UNIVERSITY OF TWENTE.

KU LEUVEN

A framework for creating semantic wikis for biomedical research laboratories

Human Media Interaction - Master thesis

Author
Daniel Davison, BSc
d.p.davison@student.utwente.nl
s0115940

University of Twente supervisors
Dr. Paul van der Vet
Dr. Maurice van Keulen
Prof. Dr. Dirk Heylen
Brend Wanders, MSc

KU Leuven supervisor
Prof. Dr. Liesbet Geris

December 7, 2013

Contents

Preface

This research is based primarily on observations made during a two-month period at the Prometheus research department at the KU Leuven. During this period I had the opportunity to closely follow researchers in their day-to-day activities in order to document their habits and identify their workflows. I would like to thank the researchers and lab technicians for their excellent patience and guidance during my time spent in the laboratory, and for granting me the freedom to observe their experiments.

Additional thanks go to my supervisors Paul, Brend and Maurice for their excellent support and guidance during the planning and execution of my research, and the writing of my thesis. Although I hit several (creative) roadblocks on the way, our weekly meetings generally kept me on track and—more than once—helped me find the necessary inspiration required to continue.

Prologue

As I wander through the hallway searching for the main laboratory, I am approached by a woman wearing a long white lab coat, carrying a stack of papers and a large notebook. "Hi, you must be Daniel! My name is Mary. Follow me, I'll give you a tour of our laboratory," she says. Today is my first day as a laboratory assistant in this biomedical research laboratory and Mary is one of the many new colleagues I will be working with the coming years. After we have exchanged pleasantries, I follow her through a long hallway, passing dozens of doors marked with cryptic signs and warnings of which I can only guess the meaning. At some places the corridor narrows significantly due to the many freezers piled along the walls. "These cooling units are where we keep all our biological samples. Some are stored for years, even after the experiments have been completed," Mary tells me, "We must make sure to keep track of where all the samples are stored, because if we misplace an important sample we might lose months of work and need to re-do costly experiments."

When we enter the main laboratory, I am greeted with an overwhelming background noise of whirring and buzzing equipment, cryogenic freezer compressors and air extractors. The lab is alive with activity. The various workbenches are crowded with glass jars, pipettes, instruments, laptops, notebooks and half-completed experiments, and everywhere I look I see lab technicians vigorously working on their research. The shelves above the workbenches are filled with old notebooks and boxes containing even more supplies. Mary laughs when I ask her how they can find anything in this apparent chaos. "You will learn soon enough how we keep ourselves organized, don't worry," she reassures me.

Mary gives me a guided tour of the most frequently used equipment in the lab. "These are our centrifuges, they spin at several thousand RPM. We use them mainly during experiments to separate concentrated cell mass from solutions." "That explains the high-pitched whirring sound," I think to myself. "Over here is one of our flow chambers, which we use when working with cell cultures. The chamber blows sterile air over the bench, to ensure that no contamination occurs when cells are out in the open. We have several other chambers in quarantined areas for when you need to work with potentially dangerous cell strains". I notice that the flow chambers are stocked to the brim with hundreds of tiny pipette tips and well plates. Mary tells me that I will often have to perform experiments involving cell culturing, which implicates manually refreshing the growth medium of hundreds of tiny samples. "Often a time-consuming task," she says "but it has to be done if we want to keep our cells alive."

After having seen the main laboratory I follow her into a series of smaller rooms which *house the specialized equipment, such as cell incubators, microscopes, CT scanners and histology workstations. At each bench, a researcher is either busy configuring the equipment, or frantically writing information into his notebook. Sometimes the information is displayed on a computer screen, other times it is copied from a paper print-out. "So what's the deal with these notebooks I see everyone hauling around?" I ask Mary. "These notebooks are our way of defining experiments and keeping track of procedures, progress and experimental data. Anything of relevance to our research is recorded in such a notebook, so that we can retrace our steps when reviewing research results," she replies.*

As we walk back to the main laboratory, I ponder over the question why the laboratory still uses paper notebooks for recording their experiments in favour of a more future-proof solution. Surely in this digital age there must exist a better alternative? I ask Mary if they use any computer databases for storing research data. "Yes, of course! We use a database for our patient files and cell line information," she responds cheerfully. "Here, I'll show you". She boots up a workstation computer, navigates to a shared network drive and attempts to open an Excel document. She barely flinches when the program confronts her with an error message: "The requested file is locked by a different user". Mary raises her voice above the background noise and shouts across the room: "Is anyone using the patient records file at the moment? Or did someone forget to close it again when they were done?"

Chapter 1

Introduction

Modern biomedical research facilities typically employ dozens of researchers and technical staff, working on numerous concurrent projects. Each researcher is responsible for documenting their work using a laboratory notebook. Such a notebook contains all information related to specific experiments, such as descriptions, protocols, measurements, raw data, annotations and important observations. When project results are being reviewed before publication, it is important that a researcher can prove that his obtained results are accurate, by referring to the particular pages in their lab notebook. After a research project is completed, the corresponding notebooks are moved to archive.

Difficulties arise when multiple researchers collaborate on a certain project or when it is necessary to query historical or aggregated results. Since the data is handwritten on paper and can be scattered over multiple notebooks, the seemingly simple task of retrieving the data can become a time consuming and error-prone endeavour. If such data would be available in a digital format however, the task of searching for specific results or creating aggregated overviews could be simplified significantly.

Biomedical research is aimed at gathering and expanding knowledge about biological processes. This knowledge is used to develop and improve clinical trials which are eventually used to treat patients. Usually an iterative approach is used; experiments are repeated numerous times or in large quantities while certain parameters are tweaked. Typically, an experiment might be conducted up to a dozen times with varying parameters, while a single experiment might contain up to several hundred samples. Such experiments are usually part of pre-clinical trials, and are aimed at exploring and optimizing various parameters that are of influence to a biological process of interest. For instance, by tweaking parameters A, B and C in various combinations in a large series of biological samples, researchers can infer what their influence is on processes X and/or Y. This knowledge can then be applied when developing a clinical trial (i.e. a treatment) for a disease where these processes are malfunctioning, for instance. Pre-clinical trials are generally conducted on laboratory animals (such as mice, rats and sheep) or artificially cultured cell lines obtained from human donors.

1.1 Prometheus

Prometheus is the skeletal tissue engineering department at the KU Leuven. Their ultimate goal is to apply their developed tissue engineering methods in a clinical trial setup. Working together with the Bone4Kids foundation, the researchers hope to develop an alternative treatment method for children suffering from pseudarthrosis [1]. Pseudarthrosis is a condition where a sustained bone fracture is unable to heal, resulting in a permanently broken bone. The current treatments often involve numerous complicated operations, prolonged revalidation or in the worst case can result in an amputation of the affected limb. By using skeletal tissue engineering techniques, researchers are developing a treatment which can replace the defective (fractured) bone segment with a healthy transplant, which is grown from the patient's own donor cells. This method should significantly reduce the amount of necessary surgery and revalidation, increasing the quality of the patient's life.

The researchers in the laboratory are associated with various tracks. Each research track focuses on a specific stage in the process of bone formation, ranging from fundamental research to computational modelling. Since the ultimate goal of the group is to perform clinical trials, it is important that all experiments are meticulously recorded in the lab notebooks. All experiment designs, data, annotations and corrections must be catalogued for future reference, for instance when a medical committee is reviewing a request for a clinical procedure. Such a review generally focuses on the documentation and quality of the procedures, the extent to which researchers follow procedures and the availability of provenance information and chains of custody. In order to accurately assess a procedure, the committee typically requires access to lab notebooks and aggregated research data.

During a preliminary ethnographic research I conducted at the Prometheus department, I closely examined and documented their workflow [14]. It quickly became apparent that a large part of the data workflow in the laboratory relies heavily on standardized processes, referred to as Standard Operating Procedures (SOPs). SOPs are defined by the International Conference on Harmonisation as "detailed, written instructions to achieve uniformity of the performance of a specific function" [25]. In practice, this is typically a detailed a step-by-step instruction of the process, including details on the used materials, equipment and methods. A collection of such SOPs dictate the workflow of a research project. Since laboratory technicians are expected to operate according to the applicable SOPs, work performed by different researchers can be more easily compared and shared. Based on the conclusions from this prior research, I identified areas in the workflow where improvement could be made by introducing a digital lab notebook, as is shown in subsequent chapters. Since many scientific laboratories follow a comparable protocol-based workflow, observations made at Prometheus will most likely be applicable to a broader domain.

1.2 Research goal

The goal of this research is stated as follows:

> To create a generic platform with which laboratory researchers can construct an electronic lab notebook that supports and supplements their workflow.

Due to time constraints, it was not possible to make a complete survey of all available electronic lab notebook technologies. The choice was made to use a *semantic wiki* as a basis for the electronic lab notebook. Section 2.3 introduces and analyses this concept in more detail. The following design questions are addressed while analysing the domain, and designing and implementing the prototype application:

- Which functions should such a platform fulfil?
- Which design principles form the foundation for such a platform?
- What are limitations of current hard copy lab notebooks?
- How can can a semantic wiki address the limitations of hard copy lab notebooks?
- How should the data be organized within such a semantic wiki?
- How can the workflow of the researchers be supported by such a system?

These research questions are addressed in the following chapters of this report. In chapter 2 an analysis of the domain is made, based on an ethnographic research conducted at the Prometheus department. This chapter includes an analysis of the shortcomings of the traditional hard copy lab notebook and closes with some thoughts about possible digital alternatives which could solve some of these issues. Chapter 3 introduces the basic design concepts which address the shortcomings described in the analysis and are used as a foundation for the developed prototype discussed in chapter 4. The final chapter 5 highlights and discusses the most important findings of this research while giving some important directions for future research.

Chapter 2

Analysis

This chapter describes an analysis of the domain in which biomedical researchers operate. First of all, an analysis of the laboratory workflow is made. A clear picture of this workflow makes it possible to identify possible bottlenecks, as well as possible solutions which can be addressed with a digital lab notebook. These identified bottlenecks and solutions are seen as an important pillar on which the design and implementation of the developed prototype are based, as discussed in subsequent chapters.

2.1 Laboratory workflow

This analysis relies primarily on observations made during a case study performed at the Prometheus laboratory in a preliminary ethnographic research [14]. During the study close attention was paid to the way researchers interact with data being generated during their experiments.

While interviewing some of the researchers at the laboratory, a pattern emerged which indicated that most of the generated data could be categorized in two ways:

- Experimental data
- Organizational data

The experimental data includes all the information that can be associated with a single experiment or measurement. Such data includes annotations, tables, figures and recorded values. Raw data is usually subjected to statistical analysis in a later stage, which can still be considered as experimental data. A full research project will often consist of multiple small experiments and procedures.

Although experimental data is important for drawing conclusions, it holds little value without additional accompanying organizational data. Such data typically describes the bigger picture of individual experimental results and can be used to tie them together. In other words, organizational data provides a context in which the experimental data should be viewed. For example, a single microscopy image (experimental data) holds little value on its own, but when viewed in the context of a specific staining process performed on a certain cell line (organizational data) the image holds much more useful information [14, p. 6-7].

Information recorded in a paper lab notebook typically consists of both *structured* and *unstructured* data. For instance, tables, graphs, experiment measurements and statistical analysis can be seen as structured data holding some semantic meaning, while annotations and observations can be regarded as unstructured data consisting mostly of natural language. Since for humans the two types of data are easily distinguishable, such structured and unstructured data can be mixed freely in a paper lab notebook. A digital equivalent to such a combination can be found in the semantic wiki principle, which is further elaborated in section 2.3.

The following sections describe some of the key elements in the data workflow observed at the Prometheus laboratory.

2.1.1 Standard operating procedures

A Standard Operating Procedure (SOP) is a standardized step-by-step description of a procedure or experiment, and can be seen as a much more formalized version of a protocol [14, p. 8]. The workflow of a laboratory can be largely described as the consecutive execution of multiple SOPs. Once an SOP is in use, it is periodically reviewed and updated to reflect new insights or advances in technology. Since past research was conducted according to old SOPs, previous versions are preserved and any changes are recorded for future reference. Prometheus typically aims for a yearly review cycle, but other laboratories might follow a different schedule.

By making sure all researchers follow the same SOPs, the obtained experiment results can be more easily compared. Many of the activities in the lab are described in such a document [14, p. 23]. A medical committee will often debate whether or not to allow clinical trials of a procedure, basing part of their decision on the quality of the SOPs and the adherence to these SOPs by the researchers.

2.1.2 Research template

Perhaps the most important element of organizational data is the research template, since it initially contains references to all underlying processes [14, p. 23]. As the research progresses, this template is extended with additional data instances accompanying the experiments and measurements. When creating the template, a researcher will typically receive feedback and corrections from several colleagues. This reviewing process can generate useful information about which choices were made concerning the content of the template.

Since a template is created and approved at a specific point in time, it is important that any referenced protocol versions are preserved. For example, at time of creating and approving the template, a referenced protocol might be at version x. After conducting the experiment however, an altered version y of the protocol is released. For future reference it is important to maintain a correct coupling with the old version.

When executing the various steps of an SOP, this generally results in several consecutive experiments and measurements which produce research data. Such raw data is stored in the lab notebook for future processing and analysis [14, p. 13-22]. Once research data has been obtained it is important that a researcher can prove it has not been altered after it was entered in a lab notebook. A traditional paper notebook includes a time stamp and signature on each page, and an additional explanation when data has been altered or erased.

2.1.3 Materials

During the various experiments and procedures many different materials are used, all of which must be listed and approved in the research template. These include various types of growth flasks, scaffolds and serums, but also any used cell types and lab animals [14, p. 10-12]. By explicitly listing the materials used, the researcher shows that the protocols are followed and that his research complies with the guidelines of the laboratory. A comprehensive list of materials has been constructed by the laboratory managers, containing all approved items that can be used for any research being conducted. A researcher must adhere to this list if he is conducting routine experiments. If the goal of the research is to examine new materials, however, he is allowed to use items not present on this list.

2.1.4 Animal dossier

For many experiments a researcher uses lab animals such as mice, rabbits or sheep. Usually such research lasts for several months, during which an animal undergoes multiple surgeries while being administered anaesthetics. All such events are recorded in a special dossier of the animal, alongside its unique identification number [14, p. 13]. In addition, the animals are monitored periodically in order to keep track of their weight and overall health and recovery. The animal dossier has currently not been standardized at the laboratory and individual researchers have different methods for maintaining the animal information.

2.1.5 Donor patient dossier

Most of the research conducted by the laboratory makes use of human periosteal derived cells (HPDC), which are extracted from a human donor patient during a regular surgical procedure. Since the laboratory currently has over 200 of such donor samples in storage, it is important to keep track of them as they are used in the various experiments.

The donor patient dossier contains information about the donor of the cells, as well as the method of extracting the sample. Since some donors suffer from a genetic disease which can affect cell growth, it is important that such additional information is stored in their dossier. The information contained in the dossier is linked to the derived cell lines, so that it is always possible to retrieve this information when selecting suitable cell lines for an experiment [14, p. 7, 9, 25]. Patient dossiers are currently maintained in a shared Excel document, which is less than ideal in a multi-user environment.

2.2 Current bottlenecks

Although paper lab notebooks have been used for decades, they have some significant shortcomings when it comes to retrieving research data from archives. Usually a single researcher will be able to remember in which notebook he recorded his recent experimental results, so while conducting his own research this does not raise many difficulties. In the long run however, this can become another story as the various notebooks fill up and are moved to archive. Now imagine dozens of researchers collaborating on multiple projects, having to use each other's notes. The simple act of retrieving a measurement from a specific experiment can become a daunting task.

The above scenario only describes the routine business in the lab. If, for instance, a researcher would want to investigate the bigger picture of a certain procedure, he might need to analyse an accumulation of all data related to that procedure. This will typically involve manually browsing through countless notebooks, copying the required information by hand, with the additional risk of introducing errors in the data. After viewing various entries in the notebooks, the complexity of such a seemingly simple task becomes apparent [14, p. 6]. Section 5.2 discusses several types of questions which a researcher will typically want to answer using his lab notebook. It is currently difficult and time-consuming to answer *any* type of complex question which involves data from multiple experiments, lab notebooks or researchers.

Similarly, it can be difficult to keep track of the various SOPs which are being used in current and historic research. Since research conducted several years ago might have used an older version of a particular SOP, it is crucial that this versioning information is recorded alongside the regular data in the lab notebook. This can be important when a comparison is made between experiment outcomes that have used a different SOP version.

It is clear that the current paper lab notebook is not very well suited for working with collaborative data. The same holds true for certain other aspects of the laboratory workflow. For instance the management of donor patient records, the donor screening process and resulting cell line information is now initially recorded in a spreadsheet file [14, p. 25]. After the first few passages the cell lines are placed in cryogenic storage and individual researchers then select and culture cell lines when they require them, but generally do not update the associated records in the spreadsheet. For example, it can be the case that a batch of cells frozen after their sixth passage have been handled by up to four different researchers, being frozen and thawed an equal amount of times. Since this can negatively affect the viability of the cells, it is beneficial for a researcher to have access to this provenance information while selecting suitable cell lines for his research. Currently, this provenance information is only scarcely available and unreliable at best.

Another area where the paper lab notebook falls short, is storing animal records. Since there is currently no standardized method for creating and storing these records, each researcher has a unique approach. This makes it difficult to compare results obtained from different researchers, and can cause problems when researchers collaborate on animal-related projects [14, p. 27].

In addition to the various collaborative shortcomings mentioned above, the paper notebook has some practical issues when it comes to working with laboratory equipment. Taking measurements often involves painstakingly copying several dozen numbers or tables by hand from a computer screen or paper print into the notebook, increasing the chance that errors are made in the process. In addition, the layout of the lab notebook can be quite confusing to novices. Since a researcher is typically working on various experiments simultaneously, but only has one book, results from different experiments are organized criss cross throughout the notebook. The researcher must keep close track of which pages belong to which experiment, and must manually introduce "jumps" from one page to another when there is not sufficient room to record all data on a single page.

In conclusion, the traditional paper lab notebook exhibits the following shortcomings:

- Archived information is relatively unstructured and difficult to retrieve
- Collaboration is hindered, especially when considering remote partnerships
- Organizational data (such as SOPs, cell lines, and patient- and animal records) is difficult to establish and regulate

- There is no possibility for automating tasks, such as retrieving measurements from a machine

2.3 Semantic wiki

Similar to a paper notebook, a regular, *syntactic* wiki page typically contains human readable text, untyped hyper links and images. Because we can understand the natural language used on a regular wiki, this information may appear structured in our eyes. From the view of a computer system, however, it is relatively unstructured and therefore difficult to understand. A *semantic* wiki page contains additional structured data about the information on the page and about relationships between pages, in such a way that it *can* be interpreted by computers [11, 43].

A semantic wiki is typically used to store both structured and unstructured data together, much like how a researcher might include tables and graphics in his notebook accompanied with annotations. By using specially constructed queries it is possible to ask complex questions about the semantic data, effectively building a knowledge base. Compared to regular paper notebooks, this searchability is a major benefit and can save a researcher countless hours of ploughing through archived notebooks. In addition, semantic wiki system often include an option to export all knowledge as an RDF graph, which makes it possible to *reason* about the semantic data. In the future, it might even become possible to automatically deduce new scientific facts based on such a semantic knowledge base. The firsts steps in this direction are already being explored by researchers such as King et al. who have constructed an automated hypothesis generator and accompanying robot scientist using a large biological knowledge base concerning yeast genomes [40, 30].

2.4 Opportunities and related work

As stated in the conclusion of the preliminary ethnographic research, the following processes are responsible for generating the bulk of the important data in the laboratory, and do so in a sufficiently generic fashion that they can be stored in a digital equivalent of a lab notebook:

- Research templates and SOPs
- Donor screening, cell culturing and cell line provenance
- Various measurements performed during or after in vitro and in vivo experiments (such as metabolic activity, DNA, gene expression, nano-CT scans and histology)

This data is primarily used by the researchers directly involved with the associated research track, although some data (e.g. cell line provenance) has potential for being used on a wider range. Researchers indicate that having data available in a digital, searchable format could greatly reduce the time necessary for retrieving archived information [14, p. 28].

If a digital lab notebook would support these processes and the data generated by them, it would be well on its way to streamlining the workflow in the laboratory. In recent years, several attempts have been made at creating systems which can take on part of this role. Initially aimed at sample tracking, equipment integration and data exchange, Laboratory Information Management Systems address part of this problem [15]. Such LIMS will typically not cover detailed workflow elements such as SOPs, however. In addition, such systems can have a steep learning curve, both for configuring and using the system, which can result in a relatively costly implementation project. In recent years LIMS have grown to enterprise levels of functionality and complexity, offering increased integration and lab management possibilities. An application will usually focus on being either highly customizable (via code extensions or plugins) or configurable (via comprehensive options and settings). The customizable approach will generally result in a more tailor-made solution, but can come at a high implementation cost if exotic features are desirable. The configurable solution requires extensive knowledge of the available settings and their influences and can be a daunting task to set up as required. Additionally, it might not be able to offer uncommon functionality out of the box. Many different LIMS solutions exits, some specialized in supporting specific domains of research, while others attempt to cater to all domains.

In contrast, electronic lab notebooks, or ELNs, are aimed at replacing the archival properties of hard copy lab notebooks. They have a strong focus on regulatory and legal aspects, such as audit trails, chains of custody and protection from unauthorized changes. ELN software must comply with various regulations before being used in medical trials and will often undergo rigorous validation before being applied in a medical application. The software is therefore quite rigid and cannot cope well with dynamic situations and shifting requirements of research laboratories [17]. Although it falls outside the scope of this research to comply with these regulations, chapter 5 offers additional insight in how some of these concerns can be addressed. The focus of an ELN lies primarily on data archiving and process tracking and not so much on tracking and describing the rest of the equipment and materials, although recent implementations are shifting more toward the area of LIMS [37]. Since data stored in an ELN must follow strict guidelines, the user is somewhat confined in how he may interact with the system. Data must always be entered exactly according to the strict regulations, meaning that the researchers have no room for personal preferences when it comes to maintaining their lab notebook. Although this does ensure that the regulations are satisfied, it can seriously demoralize the users working with the system. Some of the Prometheus researchers who had recently worked with a similar digital lab notebook indicated that it was due to these restrictions that they preferred their hard copy notebooks.

My research focuses primarily on solving these restrictions which exist in regular ELN solutions, by offering a replacement for the basic, flexible nature of a paper notebook, while enhancing data searchability and supporting collaboration.

Current LIMS and ELNs solutions have two major problems, namely: they use proprietary data formats and have a lack of options for interconnectivity with other applications [17]. The fundamental principles of my proposed semantic wiki solution are a first step in the way of solving these problems. Firstly, the semantic information stored in the wiki can be made available for auxiliary applications via a so-called endpoint, which can be queried using a SPARQL-like query language (section 4.2.3 discusses the concept of endpoints in more detail).

Secondly, since at the lowest level all structured information of the semantic wiki is stored as standardized triples, it can be made available as an RDF graph which can be imported into other applications for further processing. For instance, if the triples would be imported into an inference engine, it becomes possible to automatically derive information based on the information in the knowledge base. On a higher abstraction level, the information contained in the lab notebook can be shaped to conform to a standardized domain ontology, so that other applications which share the same ontology vocabulary can understand and work with the information collected in the lab notebook. Further work in this area is necessary before this option will become feasible, however.

Another area where especially ELNs fall short, is supporting the flexibility of the paper lab notebook to which researchers have grown accustomed. This inflexibility is mainly caused by (legal) regulations and guidelines to which an ELN must adhere. The foundation of my solution is aimed more at replacing the flexibility of the paper notebook with a digital version, while greatly enhancing the functionality and usability by supporting better collaboration and offering more advanced searching options. This does imply that it is not a full replacement for an actual ELN, and should be used in conjunction with an alternative system in order to comply with legal requirements. Future work discussed in chapter 5 shows several approaches to extend the design of the system to support such legal requirements, without negatively impacting the flexibility.

In a similar attempt to overcome some of the difficulties existing in ELNs, recent work performed by Brooks et al. has experimented with automated lab notebook annotation [12]. They developed the amanuensis tool "Ami" which automatically constructs a logbook about chemistry experiments conducted in a fume hood. By using simple sensors, it is capable of monitoring which chemicals and materials are used. Short video fragments and additional sensors are used to monitor chemical reactions, while a wide-angle lens captures the complete experiment from start to end. A detailed time line of the experiment is constructed based on the collected information. A researcher might use this information when he comes across unexpected results, which might be difficult to explain without being able to replay the recorded events. In addition, the researcher is able to interact with the system during experiments using voice commands, making it possible to enrich the automated logbook annotations with additional spoken notes.

In a recent study Piho et al. constructed a foundation for a LIMS system which could solve the problem of globally sharing research data, by using archetypes and archetype patterns in a semantic context [35, 36]. Their prototype focused primarily on mapping the relationships between parties, products and inventory of the laboratory. Additionally, they propose a novel way for defining the process archetype, although its focus lies primarily on describing the relationships between stakeholders, as opposed to processes taking place inside a laboratory. Such archetype patterns seem to be a good step in the way of formalizing and sharing research data, although further work is required in order to be applied in a dynamic laboratory environment. A semantic wiki approach such as the one suggested here, coupled with less strict data types, will most likely be more suitable for such a purpose.

When focusing on the bigger picture of a research project, workflow management starts to play a more important role. In the scientific field, and especially in laboratory setups, the workflow of a researcher is largely dictated by the use of SOPs. As discussed previously, these procedures describe highly standardized methods and actions which are executed by researchers in a uniform manner. A sequence of several of such SOPs can be seen as the workflow of a research project. In order to effectively manage such workflows, Slominski proposes a web 2.0 based solution which leverages properties from ELNs to execute, monitor and troubleshoot scientific workflows while preserving provenance information [39]. He argues that the use of web 2.0 techniques can enhance scientific collaboration by promoting emerging web standards such as ATOM and AtomPub.

Wassink proposes an alternative approach to scientific workflow management, where tasks and resources can remain detached from one another until time of execution, thus making it possible to design both control flow and data flow aspects of the workflow [45]. His e-BioFlow system supports ad-hoc editing, making it possible to iteratively design and test workflows without the need for specifying all aspects of the complete workflow in advance.

In recent years several efforts have been made at enhancing the declarative and expressive nature of workflows, effectively enhancing them with semantic information. These semantic representations can be used for offering several levels of abstraction in workflow descriptions, while increasing workflow automation capabilities [16]. Alper et al. recently proposed a novel method for creating workflow summaries, which is based on primitives such as building blocks, combined with patterns of semantic annotations in the workflow [8]. This solution can be used for providing insight in workflow execution traces, and in the future could help shed light on summaries of provenance information.

Chapter 3

Design

Designing a generic lab notebook application is no trivial task and numerous different approaches are possible. In order to best support the workflow outlined in the previous chapter, a careful consideration of the various design options is made. This chapter introduces and describes the important elements and fundamental concepts of the lab notebook system, while at the same time critically reviewing advantages and disadvantages of alternative design methods.

The digital lab notebook consists roughly of two distinct parts: the fundamental system design (including the data storage concepts and user interface), and the actual data design, which describes how the workflow and research data is modeled.

3.1 System design

The foundation of the lab notebook system provides a base on which the researchers can store their experimental data. Since the nature of this data is very diverse, the storage system and user interface are designed to cope with both structured and unstructured data.

A high level overview of such a system is shown in figure 3.1. Various auxiliary devices connect to a central server, which hosts both structured and unstructured data. A regular client computer equipped with a web browser is used for manually retrieving and storing information in the system. In order to support future extensibility, additional devices can upload information to the central server automatically. Section 3.3 discusses various of such enhancements. The central server stores both structured and unstructured data using a semantic wiki approach, as previously discussed in section 2.3.

3.1.1 Semantic wiki

The principle of a semantic wiki lends itself ideally for storing diverse experimental information, containing both structured and unstructured data. In past research Hughes et al. have constructed a digital lab notebook for chemists which stores experimental procedures using semantic web technologies [24]. They developed an ontology which scientists can use to define high

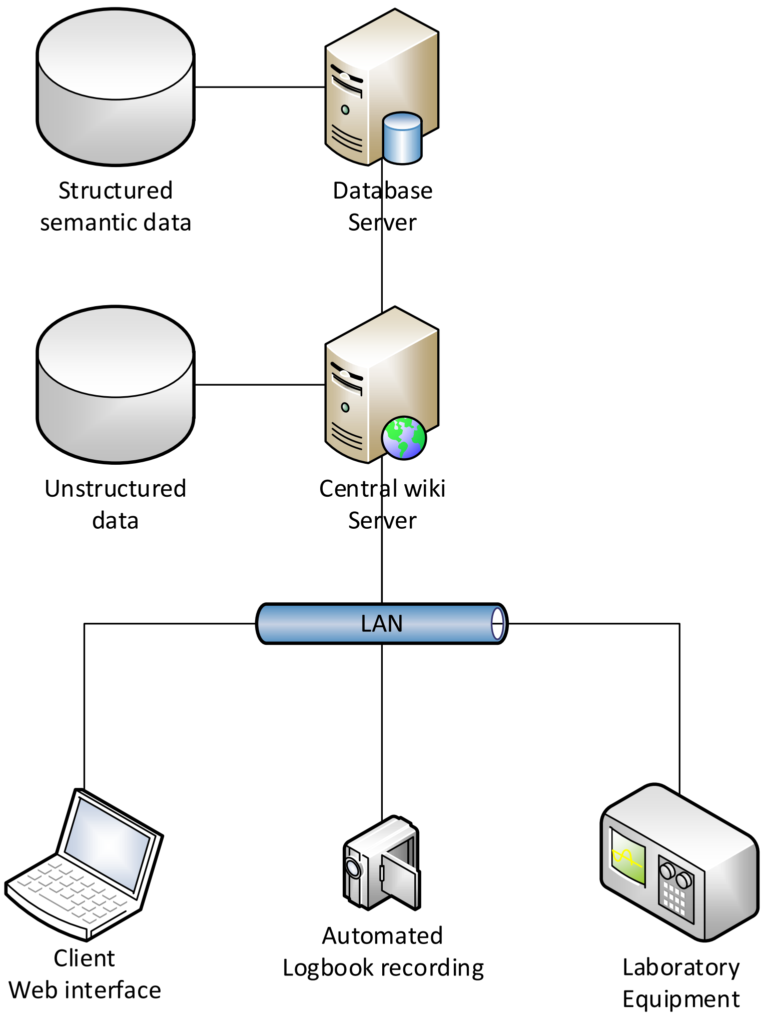


Figure 3.1: A simplified visualisation of the system architecture, showing the various connected devices and servers.

level workflows of laboratory processes, which can be consulted and annotated during experiments offering a high degree of flexibility.

Although traditional relational databases can be designed with this flexibility in mind, they are much more difficult to implement and expand by regular users. In most relational databases the data model is mirrored by the definition of, and the relationship between tables. This means that when a data model changes or is extended, the accompanying tables in the database need updating. Since this task requires knowledge about how SQL databases function, this is not something that should be left to regular users. The way data is generated and stored in the laboratory is constantly changing over time while SOPs are updated, new equipment is introduced and researchers optimize their methods. It makes sense that the data storage system should be easily updated in order to support such changes.

Since scientific collaboration has become more and more frequent over the past decade, it is critical that a digital lab notebook system supports such important partnerships accordingly [42]. Web technologies such as wikis are a natural platform for encouraging such dynamic partnerships between scientists, since the very nature of a wiki is to offer a quick and easy way for collaborative content creation [31]. In a preliminary study, Jiang et al. have found that in a collaborative context, a semantic wiki offers considerable potential for the purposes of discussion, evaluation and dissemination of information [28].

In comparison with the previously discussed research by Hughes et al. which booked success with a purely semantic storage solution, a semantic wiki approach combines the advantages of both semantic web technologies and collaborative aspects of wikis [24].

3.1.2 Free-form data

A traditional system designed for entering structured data will typically have a strict set of rules for validating any entered data. For instance, when validating an address, the system might reject an entry when it finds that the postal code doesn't match the street name and house number. The user will typically be presented with an error and will be asked to enter the information correctly before being able to continue. While this method of data validation certainly has its advantages when it comes to ensuring data consistency, it can also have its downsides. An example would be an address of a neighbourhood which is currently being built, but has not yet been included in the database. The system would prevent a user from entering this address, even though in the near future it will become perfectly valid.

Another downside of purely structured data can be that it is difficult to incorporate unstructured data (such as annotations or remarks using natural language) within the same context. An alternative is to use a so called free-form data approach, where data is initially treated as unstructured information, *unless* indicated otherwise by either the user or the system. This makes it possible to freely enter unstructured data such as text or images, and where necessary, combine it with data that does follow a structured schema. In the context of a lab notebook, this means that a researcher can describe his thoughts using regular text, while supplementing it with structured data such as experimental results, all in the same context. This approach can be seen as the digital equivalent of the traditional lab notebook, where a researcher often accompanies handwritten text with data obtained from experiments.

Unlike the traditional strict data validation of structured data systems outlined above, a free-form data system will typically not prohibit the user from entering "wrong" data or data in an unknown format. The system works on the presumption that a user knows what he is doing and should not be prohibited to enter information how he chooses. Instead, the system should be able to cope with unexpected data accordingly. In the example of an unrecognised address for instance, the system will accept the data instead of bluntly rejecting it. It will however present the user with some form of warning, in order to indicate that the data is not recognized in the current form.

This free-form approach can be ideal when dealing with the type of data commonly encountered in a laboratory. For instance, a researcher might want to partially save some research results while performing a long experiment and return at a later point to enter the remaining data. The system copes with this situation by accepting the partial data as-is, while making it possible to extend and enhance the data as an experiment progresses, without annoying the user with unnecessary obstacles. It will display warnings about any missing or inconsistent data, however, thus reminding the user which parts of the data should still be entered in the future.

3.1.3 Interface

A functional, user friendly interface is key for any application, especially one where the primary function is to interact with data. The interface for this digital lab notebook is designed with the free-form data in mind and can cope well with the dynamic and diverse nature of research data.

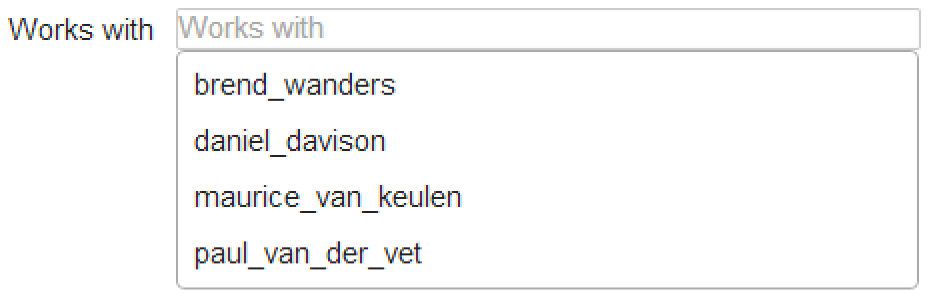
Input forms are generated dynamically based on what the data is expected look like. Section 3.2 introduces and discusses the concept of such data models in more detail. It suffices to say that by looking carefully at how the data is expected to be organized, the interface can adjust itself accordingly. An example is an autocomplete option on an input field, which lists and supplements all available options. For some fields it might contain a list of references to possible other data instances, for example.

This approach minimizes the risk that a user enters wrong information by accident, since all valid options are contained in the autocomplete dropdown. An example of such a field is shown in figure 3.2. Initially the dropdown list shows all valid references to a *person* (figure 3.2a) but as the user starts typing, the available options are filtered accordingly (figure 3.2b). If implemented correctly, this approach does not contradict with the free-form data principle. Although the interface will only suggest the valid options, it will not prohibit the user from entering "wrong" data (e.g. a reference to an instance of the wrong type, or to a non-existent instance by means of a place holder). The interface does issue a warning about any inconsistencies, but it is left to the user to decide how these issues can be solved.

Keeping in mind the free-form data principle, the interface has an intuitive way for adding additional data to an instance, without it being explicitly specified in the data model. This is accomplished by offering an additional 'open' input field, where both the name and the value of the data field can be specified by the user. In addition, the interface is able to suggest additional field names to the user (using autocomplete boxes) by filtering other data instances of the same class, searching for similar extra fields which have been entered in the past.

3.2 Data design

The base layer for the lab notebook system has been described in the previous section. Although those concepts in themselves already form a powerful tool for entering and storing semantic data, an additional layer is necessary to make the system fully functional. This section discusses the


Works with

Works with

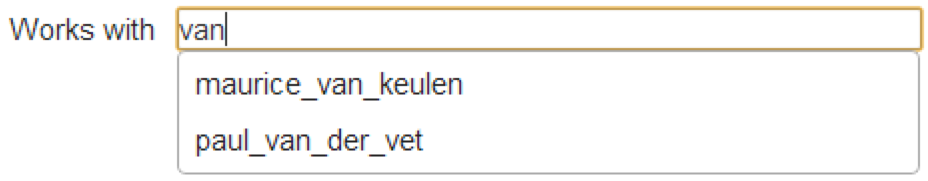
brend_wanders

daniel_davison

maurice_van_keulen

paul_van_der_vet

(a) All autocomplete options


Works with

van

maurice_van_keulen

paul_van_der_vet

(b) Filtered autocomplete options

Figure 3.2: Example of an autocomplete field

concepts related to data design, such as classes, data instances and constraints, and elaborates further on the concept of data models, which were briefly mentioned in a preceding section.

3.2.1 Data models

A semantic wiki is a very useful tool for storing and organizing information about a particular domain. In the case of a lab notebook, this domain may range from information about materials and equipment used during the experiments to actual research data being produced in the laboratory. It is clear that this wide range of data types requires a mechanism for providing structure. Without such a mechanism the quality of the semantic information stored in the wiki would soon deteriorate as multiple users attempt to enter information in different formats.

In most domains information can be categorized in various classes. A data class can be seen as a collection of all data instances that share the same type of properties. For example, all cars (e.g. your car, my neighbour's car, and so on) ultimately belong to the class *vehicle*. Generally speaking they will most likely also belong to various additional classes, such as *automobile* or *pickup truck*. Besides cars, the *vehicle* class naturally also consists of *motorcycles*, *buses* and *planes*, for instance. This concept is called the ontology of the semantic wiki and is not necessarily limited to describing physical objects such as *vehicles*, but can equally well be applied to other forms of data, such as experiment results. For instance, there can be a class for instances of *histology stainings*, of which the properties significantly differ from instances which belong to the *micro-CT scan* class.

This lab notebook system proposes the use of so-called data models for describing (a subset of) the ontology related to performing experiments. In addition, the data models are used to validate the integrity of the knowledge contained in the semantic wiki. A collection of such data models describe what the semantic data looks like, how the various pieces of data should be organized and which constraints the data should adhere to. In addition, the availability of such a description of data can be utilized to make the interface of the application more intuitive and user friendly, as described in the previous section.

Alternatively to the data model approach, it is possible to use more formalized ontologies which describe either a particular scientific domain, or aim to express all scientific research [41, 29, 35, 24]. Although such approaches are a good step towards uniformity and reuse of data, their ontologies are quite large, complex and rigid. My approach is aimed primarily at offering maximum flexibility and usability for researchers, making it possible to define and edit data models in a user friendly fashion and use them to describe the particular workflow of research conducted in the laboratory. In the long run however, it can be beneficial to investigate how the data model approach can be mapped onto regular standardized scientific ontologies in order to further increase collaboration through uniformity of data.

A data model is in essence a description of a single class of data instances. It lists the properties (referred to as fields) which are shared among all (or most) instances of the class. In the example of an *automobile*, possible properties could be *production date*, *color*, *manufacturer* or *model*, as illustrated in figure 3.3. An instance of such a class could state that the *production date* is "July, 1999", the *color* is "Silver", the *manufacturer* is "Volkswagen" and the *model* is "Golf".

Although such a straightforward list of properties describe a single data class quite well, they state nothing about any relationships that can exist *between* classes. In order to do so, additional information about the classes must be added to the data models. Since the main application of data models is data integrity checking and generation of user interface elements, the system uses a constraint-based approach to enrich the information contained in the data models. Constraints are relatively simple to construct and specify in the data model, and can be applied to both integrity checking and interface generation with relative ease.

A referential integrity constraint is a key example of such a construction. This states that the data entered in a specific property of a class must reference an instance of another specified class. For instance, the data model of a class A has a field x for which there exists a referential constraint that it must reference an instance of a class B. If property x of an instance of class A would reference an instance of a class C for instance, this is a violation of the referential integrity constraint. This will typically result in a warning in the interface.

Using a more concrete example: suppose that the data model for the *automobile* class states that the field *manufacturer* should reference an instance of a *car manufacturer* and the *model* should reference an instance of *car model*. This simplified data model is shown in figure 3.3. This implies that elsewhere in the ontology there exists a data model for the class *car manufacturer* and that there exist instances of *car manufacturers* and *car models*. Figure 3.4 illustrates an example of such an *automobile* instance, showing the relationship it has to other data instances.

The constructed data model can now be used to validate the integrity of all *automobile* instances and the interface can upgrade the *manufacturer* field to an autocomplete field, containing all instances of the class *car manufacturer*. It would even be feasible to create an additional constraint which states that the field *model* should reference an existing *car model* which matches the selected *car manufacturer*. This would catch more subtle inconsistencies such as a *manufacturer* "Audi" coupled with *model* "Polo".

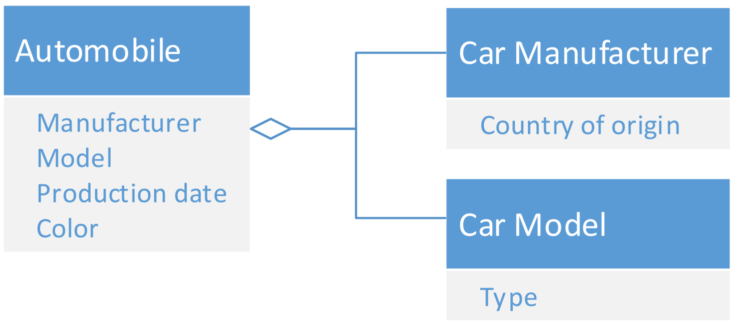


Figure 3.3: Simplified data models for classes *Automobile*, *Car Manufacturer* and *Car Model* showing the relational constraints

3.3 Supporting future extendibility

The basic system is designed while keeping several future extensions in mind. This section highlights some of the possible future developments for which the design of the lab notebook system can be extended, some of which should already be viable using technologies available today.

Integration with alternative data storage Data produced by biomedical experiments can come from many different sources, and can usually be stored in many different locations. Although a future system could be able integrate and communicate with most of the machines in the laboratory in order to retrieve data automatically, it is imaginable that not all produced research data is stored locally in the wiki. For instance, a micro-CT scan produces such a large amount of raw data that it requires a dedicated storage area outside the lab notebook system. Other data might be stored in different facilities or in cloud storage, for instance when two laboratories collaborate on a specific research project.

The lab notebook must be able to cope with this remote data in such a way that a user can add and view important metadata and can use the data to create semantic references within the wiki even when technical difficulties might prohibit direct access. Many commercial data storage solutions offer some form of API with which browsing and retrieving the data can be streamlined. Such an approach will most likely use references (such as URIs) to the data which is stored remotely, meaning that only the reference is stored locally in the lab notebook, while the actual data and metadata is stored elsewhere.

Automatic processing of data At various points during a research project, the researcher might need to process his gathered data. He will for instance generate graphs or other visualizations, or he will want to make comparisons between different instances of data. Generally, he will use programs such as excel to filter and compose the data prior to generating the necessary outputs. Since there is no standardized way to perform such operations, many different processes are maintained and it might be difficult to reproduce or compare processed results.

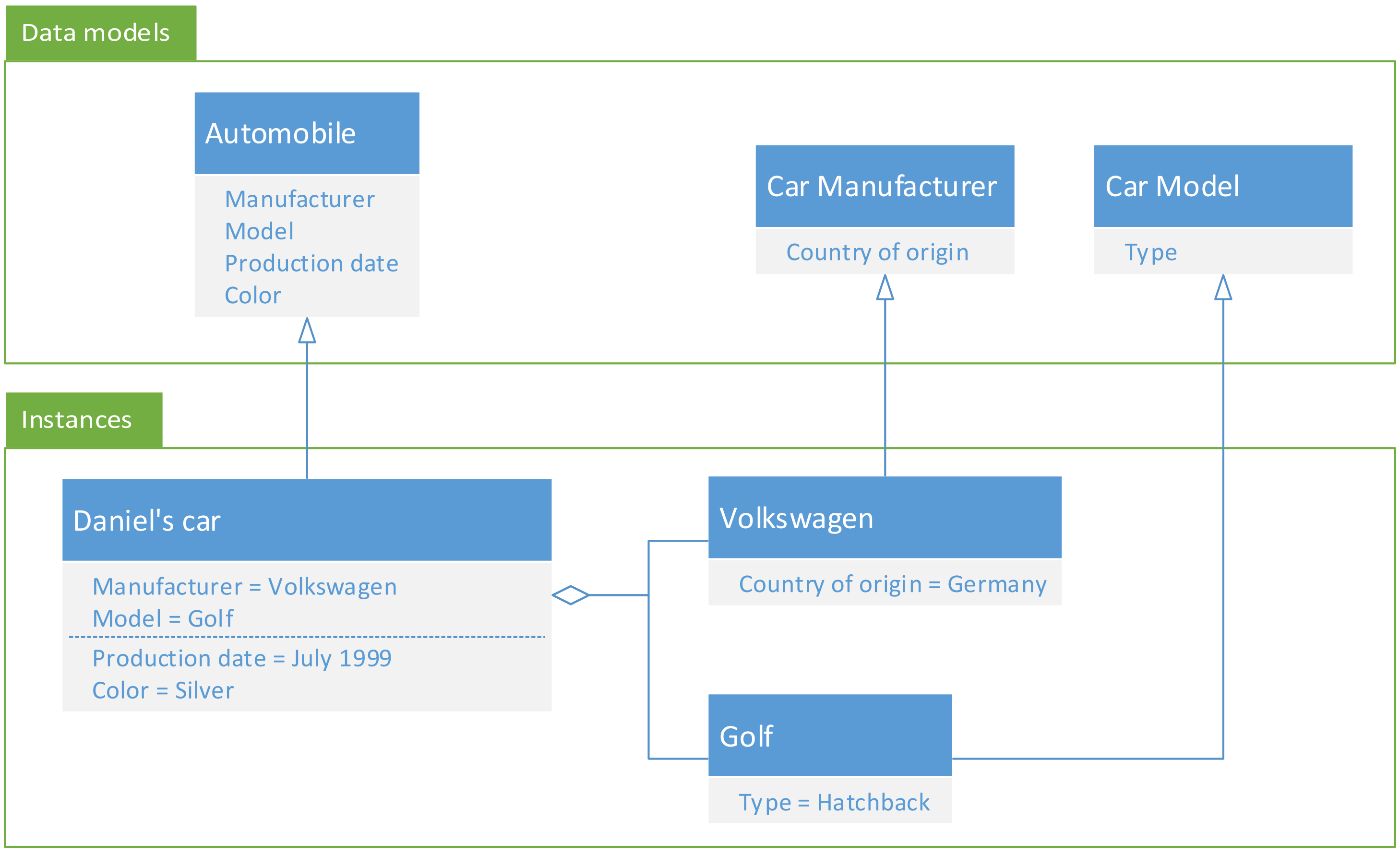


Figure 3.4: An instance of an *Automobile*, *Car Manufacturer* and *Car Model*

The lab notebook could solve part of this problem by offering standardized, automatic processing options for known data formats. The system could for instance automatically generate graphs (e.g. growth curves), perform statistical analysis or generate comparisons between data sources. It is of course important for the system to be able to recognize which type of data has been entered and how this should be processed. If this information is expressed in the workflow definition of a research project, it becomes possible to integrate the automatic processing of data into the execution of the workflow [45].

If the system is unable to automatically process data or if the researcher wishes to perform custom analysis, it should be possible to easily export (and subsequently import) data from the system in a wide range of available formats. Such formats could include CVS, Excel, XML, SPSS, etc. This makes it possible for a user to continue working with his favorite software suite, while still being able to store raw and processed data in the digital lab notebook without much issue.

Integrate with laboratory equipment Some of the equipment used in the laboratory is equipped with a form of I/O interface (such as serial or USB) which can be used for importing or exporting configuration data or measurements. It is usually essential to record such information in a lab notebook, since it is an important part of an experiment. If lab equipment can be integrated into the system, it becomes possible to automatically load their data into the semantic wiki. The researcher would no longer be tasked with manually copying over data, which can introduce unwanted errors. If the equipment is fully integrated with the digital lab notebook, the complexity of the associated workflows can be reduced significantly. Ultimately, it will even become possible to outsource the mundane part of the manual labour to robotic scientists, which can perhaps be integrated with the digital lab notebook [30].

An observed example of such a complex and awkward workflow is one where a researcher uses an ancient DOS-based machine to measure fluorescent values of several different samples. The values are displayed on a computer monitor, and are manually copied onto a piece of paper. When the researcher returns to his desk, the values are copied once again into an Excel spreadsheet, which is then printed and pasted into his lab notebook. This workflow could be significantly simplified if the fluorescence meter was directly attached to the wiki, allowing the measurements to be directly entered into the respective experiment.

The data model concept could be extended to incorporate a definition for auxiliary devices, making it possible to automatically generate semantic data instances in the wiki when certain triggers are satisfied. In addition, the system could incorporate sensors such as those used in the Ami project, in order to automatically construct a detailed logbook of the activities performed during an experiment [12]. The next section discusses this in more detail.

Speech recognition A researcher will often have his hands full while performing experiments, making it difficult or dangerous to record in data or annotations by hand. During such instances a speech-based input system is much more desirable.

There are several ways in which such a system can be used. Firstly, it can control the behaviour of the lab notebook: simple speech based commands such as “open protocol x” or “search for results from experiment y” can be used to navigate through the wiki while keeping the hands free. On the other hand, a speech recognition system can also be used to directly enter research data while an experiment is being performed. If such data belongs to a limited domain, such as numeric data, it can be recognized and entered with near-perfect accuracy. The system will need to know details about the origin of the data, however, in order to correctly enter it into the semantic wiki.

Similar to research data, a researcher might want to enter additional metadata or annotations alongside key steps in the protocol. This is another possible application for speech recognition, and although the domain of natural speech is broader than that of research data, the accuracy is of less importance, since the content of annotations is not used directly in a semantic context. The system will nevertheless need to know which stage of an experiment is currently being annotated, in order to link the annotation correctly.

The Ami project has recently demonstrated the practicality of such speech based interactions in a scientific context by training a regular speech recognition engine to perform specified actions in a digital logbook application [12].

Network of sensors For the various applications outlined above, it is necessary for the system to know what a user is currently working on in the lab. By using a network of relatively cheap sensors, it should be possible to deduce where a user is and what he is doing at a certain point in time. Motion sensors could be used to detect in which part of the lab the user is working, and pressure sensors or RFID tags could be used to detect which equipment has been picked up [12]. Multi-user concerns need addressing, however, since many researchers share the same facilities.

More advanced solutions, such as video recognition (e.g. Microsoft Kinect), can be used to further fine-tune the detection possibilities. Such a system could potentially recognize complex gestures associated with individual actions defined in protocols. The system could then, for instance, pro-actively work with the user by starting up and configuring equipment prior to the user requiring it, or automatically annotating measurements with relevant metadata.

If the information gathered by the sensors is stored in a log (annotated by timestamps and perhaps additional video data), it provides the researcher with a detailed chronology of how the experiment was executed. This information could later turn out to be valuable when explaining (unexpected) experiment results [12].

Image recognition Several protocols require the researcher to visually inspect (microscopic) images, in order to count cells for example. This is a task that is not only sensitive to errors (images will typically have hundreds of cells), but is also open for interpretation (e.g. is a cell on the border of the image counted or not?) and can be quite time-consuming for a researcher. An automated computer vision or image recognition system could be used to count such cells in a faster, standardized fashion, where the error can be statistically estimated.

A different opportunity for image recognition is selecting regions of interest for micro-CT images. A researcher must manually outline potions of a CT scan which contain bone and scaffold material, so that an algorithm can calculate the percentage of bone formation, etc. This outlining is a painstaking process, and can be automated to a certain extent. In theory, an image processing algorithm could create a rough outline of the region of interest, after which a researcher can quickly fine-tune the outline. In practice, however, this can be extremely difficult to accomplish using an automated algorithm. Further research in this field is required before being a feasible option in such a lab notebook system.

Use existing ontologies The medical and scientific communities have many existing ontologies available, which can be used to integrate with other data sources or further enhance the semantic wiki in the digital lab notebook. Especially when it comes to automated reasoning about gathered data it can be essential to use knowledge contained in existing ontologies. The task of merging existing ontologies is not straightforward however, and can require significant additional work.

In order to make the scientific workflow and the research data stored in the semantic wiki more reusable and interchangeable, it might be interesting to take a look at attempts that have been made at formalizing scientific research and results [41, 40, 29]. As discussed previously, the concepts of the data model approach would need to be mapped onto such an ontology in order to benefit from the advantages it offers.

Chapter 4

Prototype

In order to demonstrate the viability of the principles outlined in preceding chapters, a prototype was created based on the workflow of the Prometheus laboratory. This chapter covers the technical and functional capabilities of the prototype while discussing the various important design decisions which were made during the implementation phase. In addition, the implementation of the necessary data models is covered in more detail. Finally, some recommendations are made for a future version of the prototype.

As discussed in the previous chapter, a semantic wiki is a viable option for storing the type of data produced by a research facility, since it supports the free-form data concept. An ideal solution is to use an existing wiki-based platform, which can be enriched with semi-structured data. For this prototype the DokuWiki platform [18] was chosen, combined with the Strata plugin [44] which makes it possible to enter structured data blocks within the context of a wiki page. These data blocks can then be queried as if they were a knowledge base. Strata does not impose any restrictions on what the data looks like, making it ideal for the free-form laboratory notebook.

A notable alternative to the DokuWiki and Strata combination is the MediaWiki platform [3], which is most well-known for hosting the Wikipedia encyclopedia. It has a large collection of available plugins and extensions, among which the Semantic MediaWiki extension [4], responsible for adding semantic data to wiki pages. This effectively turns the wiki into a knowledge base which can be used for storing both structured and unstructured data in a free-form fashion. An additional standalone SMW+ platform [2] was based on this combination of MediaWiki and the Semantic extension, but was further enhanced specifically for creating semantic wikis for the domains biology, chemistry and physics. Unfortunately this extension is currently no longer under development due to a bankruptcy of the parent company [5].

Although this combination of a wiki and a semantic data plugin is a nice solution for integrating structured and unstructured data, it holds some disadvantages related to redundancy, scalability and usability, due to the additional layer of syntax introduced by the semantic extension. In order to overcome these problems, researchers have recently created a standalone platform OntoWiki, which can be used for distributed knowledge engineering [9]. Although their solution is inspired by the wiki free-form data paradigm, their approach is completely different than that of a traditional wiki. The information contained in the knowledge base is presented as a visual information map, which can be edited end viewed intuitively. A strong focus lies on the collaborative aspect of the system, meaning that several different users can work together to expand the information in the knowledge base. The OntoWiki Application Framework is a more recent successor to the OntoWiki platform and offers enhanced functionality which can be used for creating semantic applications, based around the intuitive information map of the original system [21]. Although this standalone framework offers an intuitive interface for working with semantic data, an electronic lab notebook will still contain a significant amount of unstructured (textual) data for which a traditional wiki is more suited.

For this prototype the DokuWiki platform was chosen above other alternatives because of the expected lesser learning curve required for building necessary extensions. Many colleagues within the faculty have accustomed themselves with this particular wiki, making it possible to benefit from their experience when configuring and developing for this platform.

Within DokuWiki there are several structured data extensions available which can be used for adding semantics to unstructured data pages in a similar fashion to how this is accomplished in MediaWiki. For many users, the 'structured data' plugin [19] is the method of choice, although less popular alternatives exist which are more aimed at incorporating relational database tables into a wiki page. This structured data extension offers the basic fundamentals which make it possible to enhance DokuWiki pages with structured data, using so called 'dataentry' blocks. The 'semanticdata' plugin extends the regular structured data plugin by adding support for the SPARQL querying language, which is ideally suited for querying semantic data triples.

The prototype discussed in this research uses the lesser-known Strata plugin, which is very similar to the structured data plugin discussed above. It offers much of the same functionality but was built with the free-form data design in mind, meaning it attempts not to make any assumptions about the data that will be entered, leaving all design choices in the hands of the user. The Strata plugin was developed by colleagues of the database group at the University of Twente and has been used in several internal projects. This has been one of the considerations which led to choosing it over the competitors, since any arising issues could be dealt with more swiftly. However, as the less popular structured data plugin, it has yet to prove itself to be stable in a wider range of situations before being suitable for more than research purposes. Strata offers a query language which is very similar to SPARQL, but is adapted in such a way that it uses more intuitive keywords and structures.

Unlike regular knowledge bases, Strata does not include an inference engine which can be used to automatically infer new facts based on previously entered rules. Strictly speaking, data entered in Strata can therefore not be considered a true knowledge base, but more a semantic database. In theory, however, it should be possible to export the subject-predicate-object triples as a regular RDF graph, which can be imported in an auxiliary knowledge base program that supports automatic inferencing. Alternatively, the Strata system could be extended to include an inference engine of its own.

4.1 Data models

As discussed in the previous chapter, data models describe what data instances look like and how they are linked together. Although in principle the semantic wiki can function without such models, it is a huge advantage for the researchers that they know what their data instances should look like. Extensive data models can help encourage different researchers to work together more efficiently, since everyone adheres to the same conventions for entering similar data.

Listing 4.1: Example Strata definition of a Person data block, including several fields

```
<data Person>
  First name: Daniel
  Last name: Davison
  Birthday[date::d-m-Y]: 10-10-1987
  Works with[ref]*: paul_van_der_vet, brend_wanders
</data>
```

Data models are relatively simple to create and modify, either when new data classes are introduced or when existing specifications change. With minimal training, it should be possible for a regular laboratory researcher to create new data models. It should be noted however, that once an existing data model is modified, it might invalidate any existing instances which will result in various warnings throughout the wiki pages. A versioning system could be able to keep track of such data evolutions, making it possible to constantly update data models while historical data instances remain valid [46, 27, 34].

The advantage of this generic data model approach is that the base system is not limited to only being applied in the Prometheus case study. The same data model principles can be adapted for any laboratory which follows a protocol-based workflow, or even broader business applications, simply by defining customized data models for the data classes of the domain in question.

4.1.1 Strata

Data models and individual data instances are defined within the wiki, using the Strata plugin. From the view of the wiki, a data block is nothing more than a snippet of text surrounded by `<data>` tags. The plugin parses such blocks and stores the fields and values as triples in a database. Each triple is of the form subject-predicate-object, where the subject is the identifier of the current data block and each field-value pair compose the predicate and object. Generally speaking, each wiki page can have one main data block associated with it, which can be supplemented with additional data fragments.

An example of such a data block is shown in listing 4.1. Here an instance of class *Person* has been created, with the following properties: first name, last name, birthday and works with. The *Birthday* field naturally has type *date*, while the *Works with* field has type *ref*, short for reference. Strata uses any specified type information for formatting the output of the data, which can be further fine-tuned with an additional type hint property. For instance, a *date* type tells Strata that the value must be displayed to the user as a recognizable date, such as 2013-12-31, while still storing the normalized date value in the database for easy processing. By specifying the type hint property for the *date* type it is possible to specify an alternative date format, such as 31-12-2013, as has been done for the *Birthday* field in the sample listing. Available types are: *date*, *image*, *link*, *page*, *ref*, *wiki* and the default *text*.

Data models are defined and stored within the syntax of Strata, so that they too can benefit from the features of Strata such as querying and templating. Even the structure of the data model itself is defined as a data model. Therefore, it is possible to define new data models with a user friendly interface in a similar fashion to entering regular data. The interface is discussed in more detail in section 4.2. A downside of this integrated approach is that it is not possible to *force* data instances to follow a certain model. At best, the system will be able to issue strong warnings when a user enters data which is considered inconsistent with the data model. When taking the intended free-form nature of the system into account, this behaviour is not considered to be a problem.

The process of loading a wiki content page is illustrated in figure 4.1. When a user browses to a specific page on the wiki, a HTTP request is sent to the DokuWiki server. The server loads the unstructured content from the file system, after which it loads the Strata plugin. The data triples related to the current page are then loaded from the database, after which they are parsed and formatted by a templating engine. The resulting structured data is inserted at specified locations in the unstructured content of the page, which is then returned to the web browser of the user.

Strata is capable of translating SPARQL-like queries to equivalent SQL queries on the fly, in order to retrieve information from the triple store which is hosted on a regular SQL server (such as SQLite or MySQL). This approach benefits from the speed and stability of SQL, while offering all semantic searching capabilities offered by SPARQL.

An alternative to this integrated data model approach would be to define and enforce the data model using an external application, interface and database. The integrity of the data could then be enforced more effectively, but the model itself would be much more rigid and difficult to create and maintain. Users would need to edit source code in order to update the data model, whereas in the current approach users can simply create and edit the data model using Strata syntax and generated forms. In addition, such a rigid enforcement method negates the advantages offered by the free-form data design.

4.1.2 Classes and fields

A data model describes properties about a single class of data, which in turn consists of various fields, each holding some information about the data instance. Fields can be defined in the data model as being either required or optional, and it is possible to define the type of the information contained in the field. In some cases this type is used to enhance both data entry interface, and the wiki interface where the data is displayed, as discussed in the previous section.

An example data model definition for the class *Researcher* is given in listing 4.2. The main data block is the *data model*, which states that is a model for the *Researcher* class. Additional *field* data fragments describe the fields which form the structure of the class. Each *field* fragment has a *name* property, which denotes the field name of class instances. Furthermore, each fragment references the data model with which it is associated (researcher_datamodel in the listed example) and it contains an additional property stating whether or not the field is required. Finally, it is possible to define the optional properties *type* and *type hint* in the data model for each field.

A data model can specify that certain fields should reference an instance of a specific type of class, this way it is possible to define rudimentary relationships between classes. For instance, field x of class A should reference an instance of class B. This concept of *constraints* is discussed in more detail in the next section. This approach is very generic and easily extendible, meaning that it is possible to describe many different types of data structures using these data model constructs. The concept of data classes and information fields is an intuitive way of reasoning about semi-structured data and has the advantage of being easily translated to the Strata data syntax. Instances of data classes are stored as a data block of which the subject is the class name, while the field names are stored as predicates of the data block and the information in the fields is stored as the object.

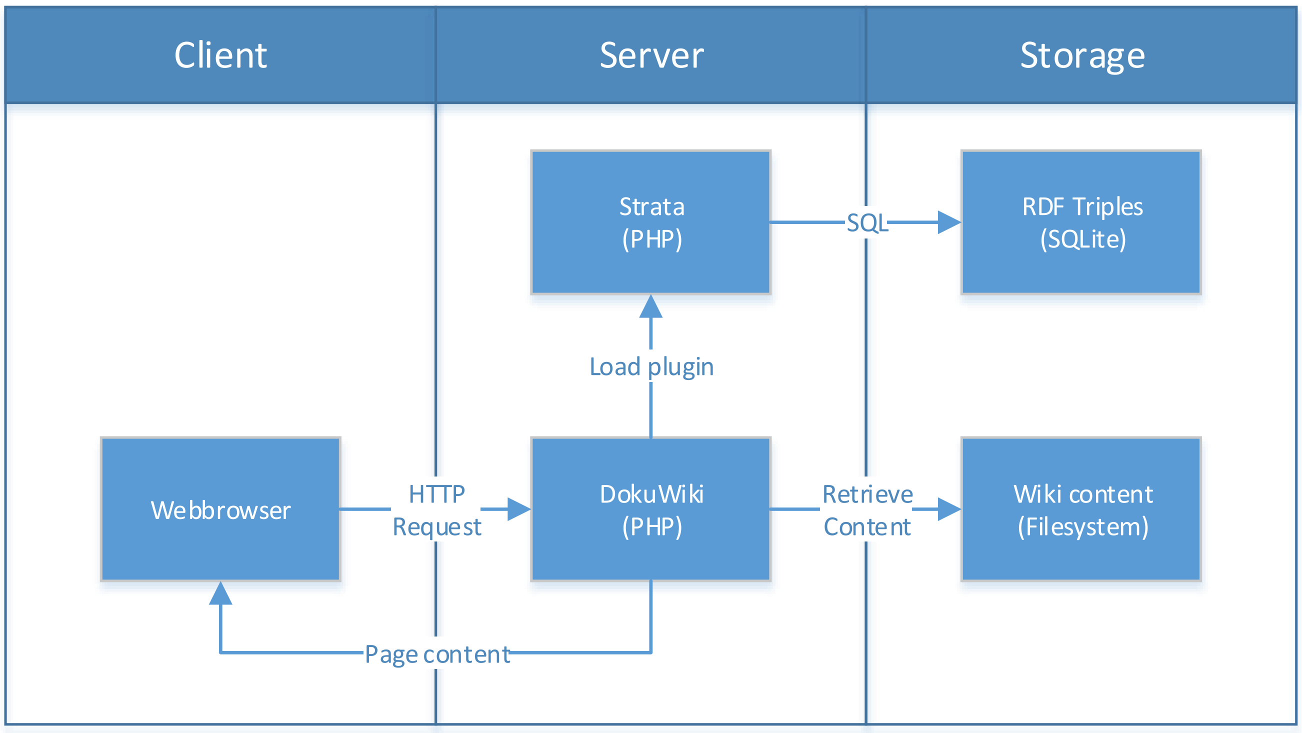


Figure 4.1: Simplified process of retrieving a wiki page, including regular unstructured data and structured semantic data

Listing 4.2: Example data model for the Researcher class

```
<data datamodel>
  Model for: Researcher
</data>

<data field#first name>
  Name: First name
  Datamodel[ref]: researcher_datamodel
  Required: true
</data>

<data field#last name>
  Name: Last name
  Datamodel[ref]: researcher_datamodel
  Required: true
</data>

<data field#birthday>
  Name: Birthday
  Datamodel[ref]: researcher_datamodel
  Required: true
  Type: date
</data>

<data field#works with>
  Name: Works with
  Datamodel[ref]: researcher_datamodel
  Type: ref
</data>
```

Listing 4.3: Example reference constraint, stating that the *Works with* property of class *Person* must reference an instance of *Person*

```
<data reference_constraint#works with ref>
  Datamodel[ref]: person_datamodel
  Field[ref]: person_datamodel#works_with
  References: person
</data>
```

Having data models defined in such a way, it is possible to generate user friendly interfaces which can be used to create new instances of data for each defined class type. The input form fields are generated and populated automatically (as much as possible) based on the structure of the data described in the accompanying model. Since data models, classes, fields and constraints each have their own underlying data model it is easy to specify all these aspects of a data model using the generated form interface which is also used for entering data. Due to this construction, if situations arise where it is necessary to define more properties for classes or fields that are not yet included in the scripting, it should be relatively simple to update both the syntax of the data model definition and the underlying Javascript plugin code.

In a future version of the system, it should be possible to link to external sources for retrieving data automatically. The class-field data model structure could be extended to incorporate this functionality. For instance, various devices and machines in the laboratory could be defined as data sources which could be coupled to specific fields of a class. When a user creates such a class, the data fields could be automatically filled by loading the associated data sources, which in turn can connect to the attached devices and retrieve the required data.

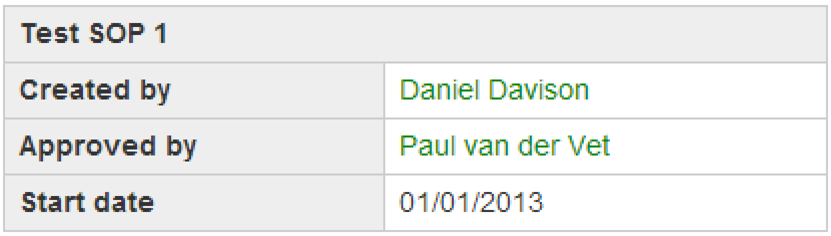
The other way around is also possible. A device has finished a task, triggering its connected data source which creates and (partially) fills out a new data block. The user could fill out the remaining fields and save the new data instance to the wiki.

4.1.3 Modular constraint structure

As discussed above, strict enforcement of the class-field data model is neither possible nor desirable in such an open environment as a wiki. It is however possible to issue warnings to users when entered data differs from the regular structure defined in the model. A simple example of this is when a required field has been omitted by a user. In order to be able to do so in more complex cases, the data model needs to be extended so it can define constraints on the data fields.

In an attempt to keep the basic class-field structure as simple as possible, a modular approach has been adopted for defining additional constraints on the data. In principle, the field definitions of a class simply describe which information belongs to that data class, but states very little about what this information should look like and how it can be related with other data. Additional constraints on these fields can describe much more about how the information looks and behaves. Once various constraints are made on fields in the data model, it is possible to use regular Strata queries and template commands to search for any classes and fields which are violating the constraints and bring them under the attention of the user.

A referential integrity constraint is an example of such a constraint that was implemented in



Test SOP 1	
Created by	Daniel Davison
Approved by	Paul van der Vet
Start date	01/01/2013

Figure 4.2: An example of how a simple template is rendered.

the prototype. It defines that a specific field of a class must always reference an instance (or multiple instances) of a specific other class. Other constraints are also feasible and could for instance define a limited-options field, which could state that a field should only contain values that match a limited set of options. This could function similar to the autocomplete option on referential fields. In principle, all constraints that are imposed on the original data fields are defined as modular data fragments. The only exception to this rule is the constraint which describes whether or not a field is required. Since this constraint is quite fundamental and must be defined for each field, it is included directly in the initial field definition, as was shown in listing 4.2.

Since it is relatively simple to define and check such constraints it is possible to extend the system to fit specific needs by adding more modular components as the need arises. Due to this generic modular design it is possible to mix and match the various available constraints to meet the specific requirements for a multitude of domains.

An alternative to this modular approach would be to store the various constraints directly into the field definitions. While this would reduce the amount of data fragments stored in the wiki significantly, it would eventually lead to large, complex lists of additional properties that would need to be deciphered and filled in when defining new data models. The current approach lends itself well for rapid prototyping situations, for instance when a quick-and-dirty version of concept data model is required. Defining classes, fields and constraints is a matter of filling out a handful of values into a few user friendly interfaces which are very similar to the ones used for creating data instances. The user interface is discussed in more detail in section 4.2.2.

4.1.4 Templates

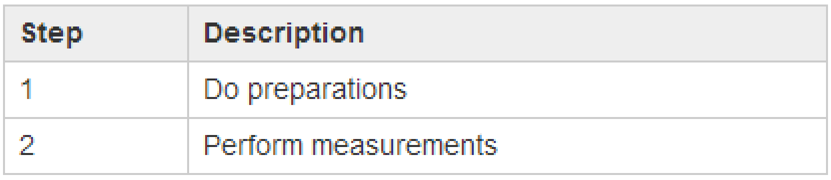
Once data blocks have been created and saved in the wiki, they must somehow be rendered on the front end of the wiki page in question. By default, Strata is able to display the various fields of data blocks as rows in a table. In addition, it is possible to run custom queries on the semantic data stored in the wiki, which can be used to generate data tables. For some applications, however, it might be necessary to display the data using a different format. The Templatery template engine is a powerful template engine which integrates nicely with both DokuWiki and Strata. Within a template it is possible to execute queries, as well as traverse through individual data blocks and their references, in order to select the information which is to be rendered to the page. An example of a very basic table template is shown in listing 4.4, and the rendered output is shown in figure 4.2. A more advanced example of a table based on a query is shown in listing 4.5 and figure 4.3. Section 4.4 discusses an additional example of such a query, which is further illustrated in figures 4.14 and 4.15 using screenshots obtained from the prototype application.

Listing 4.4: Example template of the SOP data model which defines a simple table containing some of the values defined by an SOP instance. In this context, keywords surrounded by “@@” are references to field names of class instances, which are replaced by the actual value when the template is rendered. An illustration of how such a table is rendered is shown in figure 4.2.

```
^ @@SOP name@@^^
^ Created by | @@Created by[ref]@@|
^ Approved by| @@Approved by[ref]@@|
^ Start date | @@Start date[date]@@|
```

Listing 4.5: Example of an advanced template, which uses a query to dynamically fill a table. In this example, the various steps associated with an SOP are selected and displayed in a convenient overview. An example of how such a table might be rendered is shown in figure 4.3.

```
<table ?stepnr "Step" ?description "Description">
  [[]] is a: SOP
  [[]] steps: ?s
  ?s step nr: ?stepnr
  ?s description: ?description
</table>
```



Step	Description
1	Do preparations
2	Perform measurements

Figure 4.3: An example of how an advanced template renders a table based on output from a query.

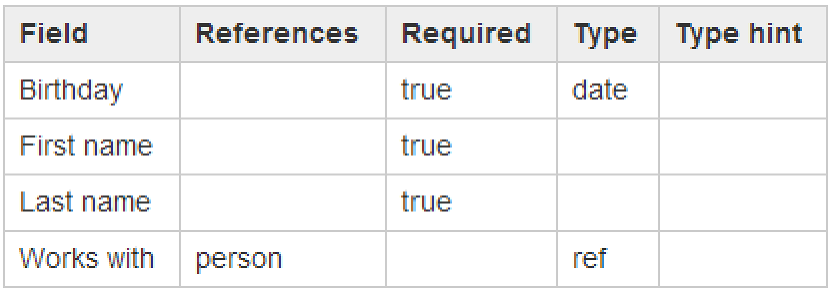
Listing 4.6: An example constraint which can be used for checking whether a required field has been specified for a certain instance. The query listed below first retrieves a list of all required fields defined in the data model for a specific class. Then it removes the fields which have actually been added to the specific instance of the class, leaving only the required fields which have not yet been entered. For each of these fields, the template then issues a warning in the form of: "<instance> misses a required field: <field name>".

```
<view ?instance ?fn>
  template {
    template:required_field_constraint#violation
  }

  ?f is a: field
  ?f datamodel[ref]: ?dm
  ?f name: ?fn
  ?f required: true

  ?dm model for: ?class

  ?instance is a: ?class
  ?instance [text] = @@.subject@@

  minus {
    ?instance ?fn: ?anything
  }
</view>

<template violation>
  :!: @@instance@@ misses a required field: ''@@fn@@''
</template>
```

In addition to being used for advanced rendering of data, templates play a crucial role in the automated constraint checking of the data models. It is these templates that are responsible for generating warning messages for users when a data inconsistency is detected.

For each type of constraint a special template is created which checks whether there are any violations in any of the data blocks defined on the page. If this is the case, a warning message is issued to the user, including details on how to resolve the conflict. Listing 4.6 illustrates an example of the constraint used for checking required fields. A user can choose to include any of the available constraint templates in the general template for each available type of data. This means that there is the freedom to exclude specific constraint checks on certain types of data, even though the other benefits of the defined constraints are still in place (such as automatically populated form fields).

The choice was made to use these templates for checking constraints, since they are well integrated in the Strata plugin and work well with the data using the powerful query language. Since the templates are defined and stored within the DokuWiki system, it is possible to modify and extend them without needing to modify any external source code. The templates can become somewhat complex, however, and regular users should probably not attempt to edit them if they are not



Field	References	Required	Type	Type hint
Birthday		true	date	
First name		true		
Last name		true		
Works with	person		ref	

Figure 4.4: An example of an overview table which is automatically generated for each data model. For each field it shows the field name, the referential class (if applicable), the type, type hint and whether or not the field is required. The example listed here is for the Person data model. Listing 4.7 shows the template code responsible for generating the overview table.

sure what they are doing.

In addition to the uses outlined above, templates are used to give a quick overview of the various fields, types and constraints of each *data model*. A user gets an overview of the features of each data model in the form of a table without the need to dive into the underlying Strata data blocks to decipher this manually, as shown in figure 4.4. The advantages of this approach become clear when classes become more complex, having many different fields and constraints.

4.2 Data input interface

Strata syntax is relatively straightforward and can most likely be taught to new users without much issue. Nevertheless, creating valid data blocks can become a daunting task when data models grow larger and more complex. Strata syntax is fully text-based and offers no real-time help when entering data. A user is presumed to *know* exactly how all data fields should be entered (i.e. he must know the exact spelling of the predicates). He must also have knowledge of the exact names of any data blocks he wishes to reference, which is not always easy when multiple users wish to reference each other's data. Naming conventions can account for some of these problems, but there are limits to how well this works when the semantic wiki becomes more complex.

Since the data model contains extensive information about field names and relations between data blocks, it is possible to construct a more user friendly interface which takes care of most of the naming- and referential issues. Figure 4.5 illustrates the process of creating new data blocks. Firstly, the web browser of the user requests information about available data models from the server, using a Strata Endpoint (see section 4.2.3). A form is then generated according to how the data models are specified (see sections 4.2.2 and 4.2.4). When a user submits this form and saves the wiki page, the information is sent to both the DokuWiki system and the Strata plugin. The structured data is extracted from the regular content and is stored as triples in an additional database.

Listing 4.7: An example template used to generate an overview table for a data model, such as the one illustrated in figure 4.4.

```
<table ?fn "Field" ?ref "References" ?req "Required" ?t "Type" ?th "Type hint">
  ?f is a: field
  ?f datamodel[ref]: [[]]
  ?f name: ?fn

  optional {
    ?rc is a: reference_constraint
    ?rc datamodel: [[]]
    ?rc field: ?f
    ?rc references: ?ref
  }
  optional {
    ?f required: ?req
  }
  optional {
    ?f type: ?t
  }
  optional {
    ?f type hint: ?th
  }
</table>
```

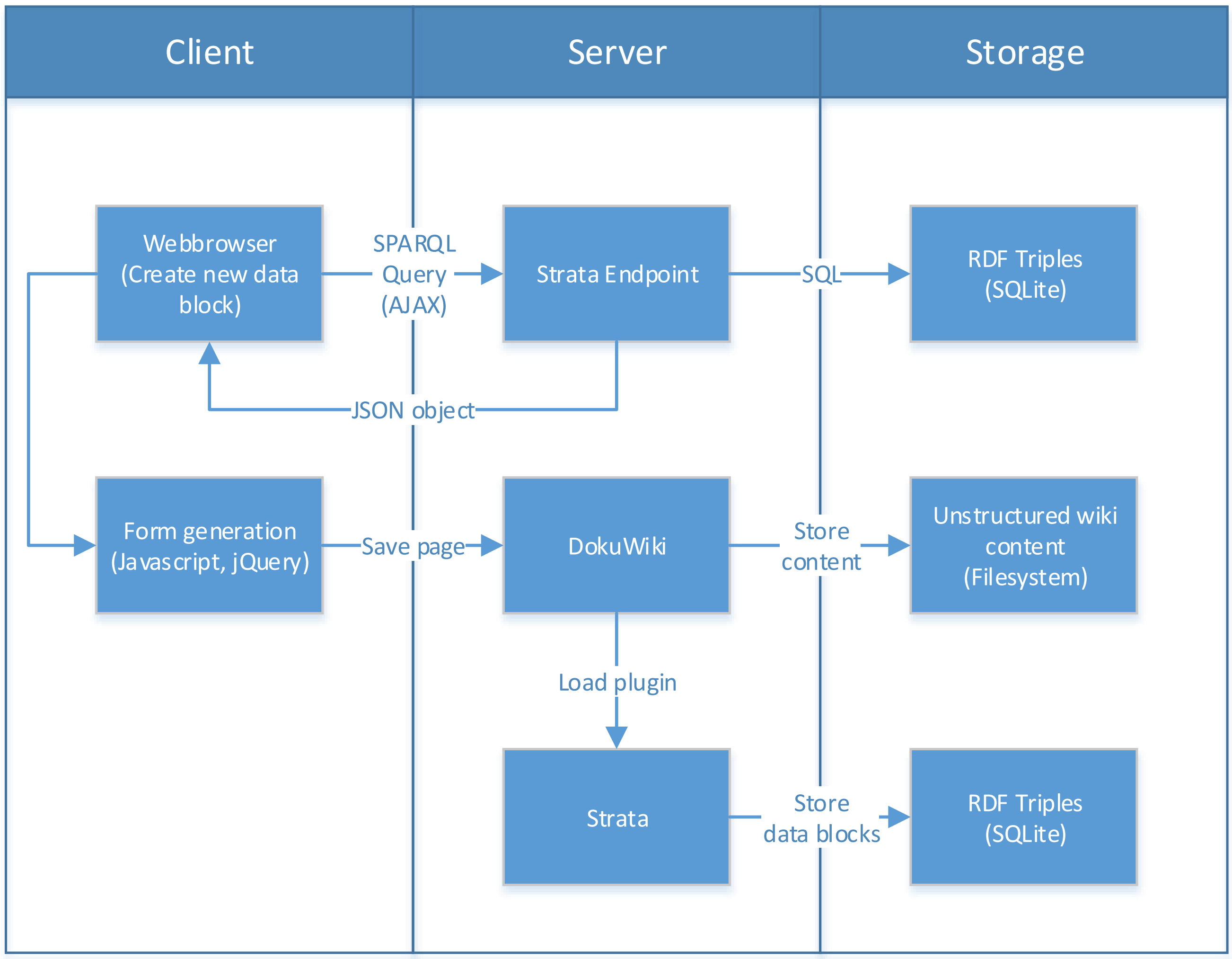


Figure 4.5: Simplified process of creating a new data block

4.2.1 Custom DokuWiki plugin

The DokuWiki platform has an extensive plugin framework, with which it is possible to create additional functionality such as the data input forms. The plugin is developed using PHP and Javascript and uses API calls to integrate nicely with the regular DokuWiki interface. This plugin approach ensures that the complete workflow of writing text, entering data and viewing pages is contained in the same platform. This provides a consistent experience for the user, since it is not necessary to constantly switch between various other systems.

The interface of the regular DokuWiki text area has been extended to include an additional custom tool bar button which, when clicked, opens a small picker box containing the names of the various available data models. Each data model represents a data class that the user can add to the page. After clicking a name, the system opens a larger popup containing the generated form fields for the selected data type. After the user has filled out the form fields, the plugin automatically translates the entered values into the Strata data block syntax.

An alternative to such a plugin is to build a separate standalone interface which stores the entered data in the Strata database using some sort of API calls. This approach would cost significantly more time to implement, however, and would add more overhead and complexity to the workflow associated with entering and maintaining data.

An exception to this would be to build a (native) interface for mobile devices such as tablets. The regular DokuWiki interface might not be suitable for such a device, so an alternative application would have to be built specifically for this purpose. This interface could offer a sub-set of the functionality of the normal plugin and could for instance be used just for entering simple data on-the-fly or for querying the semantic data.

4.2.2 JQueryUI form generator

By default, DokuWiki contains the Javascript libraries jQuery and jQueryUI. These libraries are widely used for common interface-related scripting tasks, such as generating forms and handling user input. JQuery supports its own plugin system, making it possible to add even more functionality if need may be.

The libraries combine frequently required functionality into easy to use functions, which cuts back significantly on developer time and reduces the overall complexity of scripts. Additionally, jQuery automatically takes care of the various quirks and special requirements in different browsers. Programmers can focus on creating the functionality of the script, rather than inventing obscure and complex workarounds for browser specific problems.

There are many alternatives to JQueryUI (such as ProtoypeUI or EXTJS) but they require more effort to include in DokuWiki. JQuery does the job very well and is the framework of choice for developing DokuWiki plugins. If a future version of this prototype is built using a different platform than DokuWiki, other scripting alternatives should be taken into account.

Most of the work performed by jQuery takes place by manipulating the DOM of the web page. A form object, for instance, consists of a collection of children objects such as input fields, labels and buttons which are all generated on-the-fly. The complete form object is then inserted into the DOM of the page. All forms used for data entry are contained in a modal popup which is overlaid on the regular DokuWiki background. This popup ensures that the attention of the

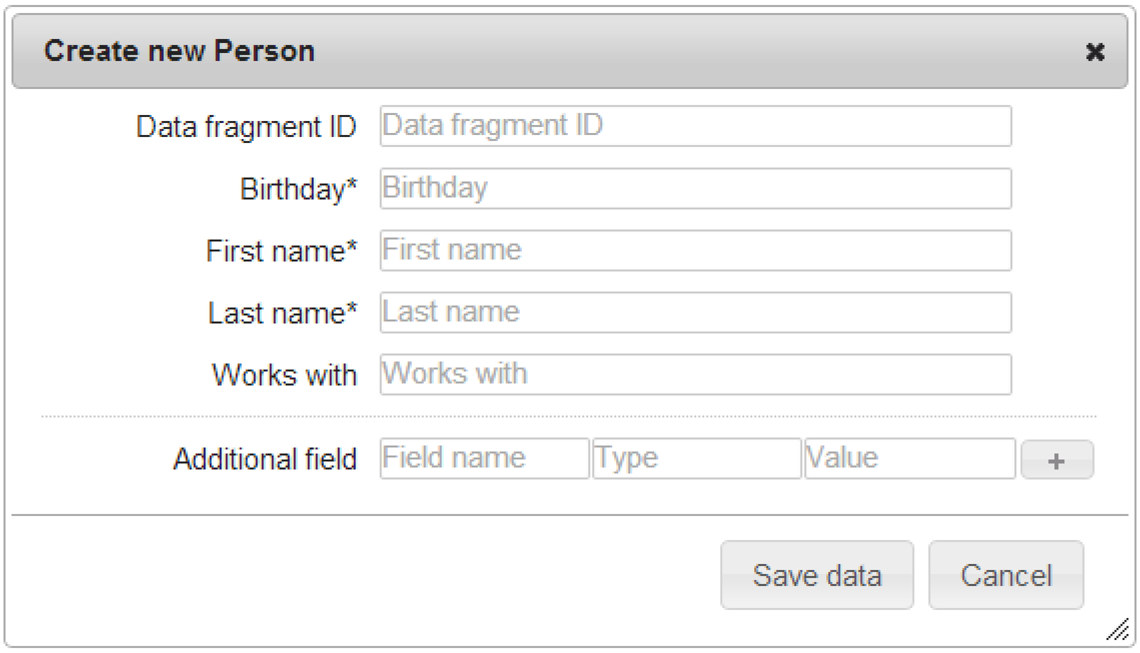


Figure 4.6: An example of a form used for entering data for the class Person. This form is generated based on the rules specified in the data model. The top field *Data fragment ID* is included in each form, and can be used to store multiple different data fragments on a single wiki page by giving each a unique name.

user is focused primarily on the form, but gives the user the reassurance that he is still working within the DokuWiki context since it is visible in the background.

Form fields themselves are generated based on the field properties defined in the data model of the chosen class. An example of such a form is included in figure 4.6 and a real world example is shown in figure 4.14. Many fields are simple text-based inputs, but some can be made more user friendly by adding additional interface elements. An example is the date input field, where a user can choose to select a date from a visual calendar, instead of entering the date manually as “yyyy/mm/dd”. Other examples could be image or file upload boxes, although the current prototype does not account for this functionality. Autocomplete is another example of such a usability enhancement and is explained in more detail in a subsequent section.

4.2.3 Strata endpoint

While generating the interface it is sometimes necessary to retrieve additional information, such as field names, types and possible references from the knowledge base. This data is loaded asynchronously as it is required, ensuring that the interface itself remains responsive while the data requests are executed in the background. By using the AJAX (Asynchronous Javascript And XML) method, the DOM can be enriched with this additional data without the need for a complete refresh of the page. This means that the user is not distracted by a constant reloading web page, and is instead presented with a streamlined interface that behaves more like conventional desktop applications.

The AJAX calls are targeted at a special Strata construct called an endpoint. This endpoint takes a SPARQL-like query as input, which it automatically translates into SQL. The result of that query is translated into a JSON object, which can then be further processed at the client side using jQuery. The endpoint executes on the server, but is defined and configured within a regular DokuWiki page. This process is illustrated in figure 4.5.

4.2.4 Autocomplete

As discussed in a previous section, a problem with making references between data blocks is that the name of the target data instance must be entered exactly as it is stored. As the wiki grows larger, this can become a challenge if data blocks do not have logical names or when multiple users are working with the system, each using their own naming convention.

A user friendly solution is to offer autocomplete functionality on applicable input fields. A user enters (part of) a data name and the system will display a list of the matching data blocks. The user then simply selects the intended item and the correct name is automatically entered into the field. When a user wishes to select multiple different references, he searches and selects them one by one and the autocomplete field takes care of concatenating the results into the correct format.

The form generator will automatically create and populate an autocomplete field when the data model for the class in question contains a reference constraint property. The possible options for the autocomplete are limited to contain only data objects of the type defined in the data model. For instance a field *is friends with* of class *Person* will have a referential integrity constraint stating that it can only reference another instance of *Person*. The autocomplete field will therefore only list data instances of type *Person*.

AJAX calls are used to pre-populate all autocomplete fields when the form is generated, as illustrated in figure 4.5. Although it is possible to retrieve the results in real-time while the user is typing, this requires an AJAX call after each letter has been typed. This is significantly slower than the pre-populated alternative and offers a laggy user experience. Since the queries are static and the underlying class instances is expected to remain relatively unchanged while filling out the form, it is not required to perform the AJAX calls required for autocomplete in real-time.

If a value entered in an autocomplete field is not recognized, it is still accepted by the form and will be stored in the wiki as it was entered. The constraint templates will issue a warning if any violations have occurred due to malformed data, and the DokuWiki system will provide a quick-link to a new empty page if an unknown reference is made. With this mechanism a user can easily (intentionally) enter a place holder reference which is unknown, if he plans to create it shortly thereafter. This fits in with the free-form nature of the system.

4.2.5 Extensible forms

The data model design can only describe the structure of the data which is predictable and constant for each instance of the same class. Some data classes might not have many predefined fields, and will instead contain much more varied information which can be different for each instance. Therefore, the user needs an easy way to enter additional open information fields into the data blocks, alongside the regular fields defined in the data model.

The regular data entry popup contains a small additional input mechanism with which it is possible to define and add more data fields on the fly. These additional fields are included in the example shown in figure 4.6. In principle, these additional form fields are a one-on-one reproduction of the Strata syntax, without much sugar coating, so they contain a field name (predicate), type, optionally a type hint and finally the value (object) which should all be filled out by the user. Once the user has entered the necessary data, the new form field is added to the regular fields in the main form, and is treated as such from there on.

In an attempt to maintain as much consistency as possible throughout the wiki, the user is assisted where possible during the creation of such a new field. For instance, the predicate input offers autocomplete options including all field names which have previously been entered for this class of data. The type input field consists of a dropdown selection box containing all available types. If the user selects the *reference* type, the value input field transforms into an autocomplete. Since the reference class type is unknown for such new fields, the autocomplete simply lists all possible instances.

4.3 Quantity type plugin

Although most of the information stored in the semantic wiki will be either text- or referential-based, a significant portion of the data will consist of quantifiable data, such as measurements that have been made during experiments. Out of the box, the Strata system can interpret and use basic numeric data in queries, but does not understand numeric data which is coupled with a unit. For instance, the system will understand "5", but not "5cm".

In order to reason about the experiment data, it is essential that one can query and compare entered values such as measurements and other stored quantities. Such measurements typically represent a physical quantity and are thus stored with the corresponding unit. This prototype contains a plugin for Strata, which can handle physical quantities and units in a similar manner to the methods used by Volkel et al. for the Semantic Wikipedia extension [43]. In fact, it can be expanded to handle any value which can be normalized one way or another, as is discussed in the following sections.

For demonstrative purposes, the current prototype supports only simple physical quantities such as *length*, *mass* and *volume*, which can be assigned to any field defined in the data model using the type system of Strata. Note that *volume* is actually not a basic quantity, since it can be reduced to $length^3$. When referring to volumes however, it is very common to refer to the unit *litre*, which behaves much like a basic quantity.

A more advanced implementation could for instance use dimensional analysis methods to decipher and understand more complex physical quantities such as *energy* (i.e. $\frac{kg \times m^2}{s^2}$) by reducing it to its basic physical quantities: $\frac{mass \times length^2)}{time^2}$). Once the algorithm understands which basic quantities *energy* consists of, it becomes possible to reason about it and convert it into *joules*, for instance.

4.3.1 SI and Imperial

The current prototype is capable of recognizing and normalizing the most common units associated with the physical quantities *length*, *mass* and *volume*. Other basic quantities such as *time* and *temperature* should be relatively simple to add. The prototype has support for both SI and

Listing 4.8: An example using the quantity type plugin to define properties of a car. Note that the values are all entered using Imperial units (foot, gallons and pounds), which will be normalized into the corresponding SI unit (meter, liter and gram) and displayed according to the preferred unit (cm, l and kg).

```
<data Car>
  Length[quantity::length->cm]: 12 foot
  Height[quantity::length->cm]: 6 foot
  Width[quantity::length->cm]: 7 foot
  Trunk volume[quantity::volume->l]: 300 gallons
  Weight[quantity::mass->kg]: 2000 pounds
</data>
```

Imperial units (such as inches or feet). All values are internally normalized to the corresponding SI unit (which are metre, gram and litre for length, mass and volume respectively). This means that it is possible to compare data which has been entered in inches with other data that has been entered in millimetres.

More complex quantities such as *surface* (i.e. $length^2$), *concentration* (i.e. $\frac{number}{volume}$) and *density* (i.e. $\frac{mass}{volume}$) are somewhat more difficult to define, and will require a future prototype to include more advanced dimensional analysis algorithms.

4.3.2 Unit recognition

Following the free-form mindset, the user should be free to enter their favourite unit of measure in the way they please. For instance, some users might prefer entering “micrometer”, while others enter “um” or “µm”. In principle, the system is able to cope with all different types of unit. Listing 4.8 contains an example of how the quantity type plugin is used.

A unit generally consists of a prefix which denotes the magnitude, followed by the suffix which denotes actual name of the unit. In the domain of biomedics, prefixes typically range from nano ($1e^{-9}$) up to kilo ($1e^3$). The unit recognition algorithm will attempt to match the input from the user with any combination of prefix and suffix it can find. In principle it is therefore possible to enter units such as microgallons or kiloinches.

If the user has neglected to enter a valid unit, the algorithm will default to the basic SI unit associated with the quantity, unless the field in question has defined a preferred display unit, in which case that unit will be used to normalize the value. The preferred display unit is discussed in more detail in the next section.

In an attempt to make the type plugin usable even in unforeseen circumstances, it will not break or throw errors when an unrecognised physical quantity has been defined. Instead it will attempt to extract and store the actual numeric value that was entered, but will not perform any normalization. This behaviour can make the quantity type useful in cases where no physical quantity is involved, but it is still useful to be able to compare numerical data. For example, an experiment might state that it requires using 5 lab mice. Instead of entering the lab mice as a normal field, the data could be entered using the quantity type, stating an *unknown* physical quantity and providing a preferred display unit “mice”. Upon entering the data, different researchers might enter various strings: “5 mice”, “5 lab mice”, “5 nude mice”, and so on. The quantity type will simply extract and store the numeric value "5" making it possible to effectively query against this specific field. The value will be displayed consistently in the interface as "5 mice" due to the preferred display unit.

4.3.3 Preferred display unit

Since only the normalized value is stored in the database, the unit that was entered by the user is unknown at time of rendering. Therefore it is possible to define an additional preferred display unit, which will be used to de-normalize the value before displaying it to the screen. For instance, data might be entered as 2 inches, it will be normalized and stored as 0.0508 metre and can be de-normalized and displayed as 0.0555 yards by entering the following field: `size[quantity::length->yards]: 2 inch`

If a field has no preferred display unit defined however, the system attempts to find the most suitable unit for displaying the value. It accomplishes this by converting the stored value into each possible unit (each combination of prefix and suffix) until it finds the "best" match. The best match is defined as the value which is closest to 1000, since this will generally be the preferred method for displaying quantities. This number was chosen arbitrarily however, and specific domains might prefer other "best" matches.

A consequence of the prefix-suffix approach is that a "best match" might sometimes turn out to be less than ideal. For instance, it could turn out that a value of 25 µm can be best displayed as 984.25 microinches, instead of the more logical 25um that was entered in the first place. Such situations can be easily solved by forcing the code to only display SI units. A future version of the prototype could deal with these situations more elegantly.

4.4 Results

With the generated form interface and the quantity type plugin in place, a case study has been implemented illustrating the viability of the semantic web lab notebook principle. The wiki platform is ideal for supporting collaboration between researchers in the laboratory, while the free-form data principle supports similar flexibility as the paper notebook to which researchers are accustomed. The key concepts of the Prometheus workflow, which were first introduced in section 2, have been implemented in the semantic wiki using the data model approach. The user friendly interface ensures that defining new data models is essentially just as easy as adding new data instances. This means that the users can adapt and extend the system to cope with the dynamic nature of the workflows used in the laboratory, without the need for a system engineer to redesign the data structures.

Basic building blocks which are easily expressed by data models are: donor patient files, researchers, materials, equipment and simple experimental procedures. Their relative static fields and constraints are simple to represent as a single data class. When multiple data blocks have more complicated interconnections, however, this simple class-based structure proves to be somewhat inefficient. Some additional constructs are discussed in section 5.1, which can be used to model the more detailed relations and dependencies that can exist in the data.

An example of such a relationship between data elements is that of the various SOPs, the research template and the experimental results. Each SOP describes a procedure in the laboratory which eventually leads to experimental data. The research template groups several of such SOPs into the context of a research project, effectively describing a workflow. When a particular SOP is added to a research template, it is often necessary to clarify additional details about that particular SOP, such as specific equipment settings. Similarly, when a research project is eventually being conducted, each individual SOP will yield unique results related to the measurements conducted in the SOP. For instance, a CT scan results in images and accompanying conclusions, while a cell count results in a growth curve.

This is a pattern which emerges frequently in a laboratory setting: a document is created which describes how a certain procedure (or collection of procedures) should be executed. This document is approved and then functions as a guideline for performing actual experiments. Each instance of such an experiment, however, will likely have slightly different variables which need to be recorded. The challenge is to define such variables in the data model of the procedure in a generic fashion, in such a way that they can be completed in a user friendly way while constructing an instance of the experiment. In other words, it should be possible to continue defining and shaping the data model in the various levels of the data hierarchy.

Since the current prototype doesn't support such a multi-level data model approach, a slight workaround is used. In addition to defining classes such as *SOP* and *research template*, additional specific data models are created to fit the most common measurements and properties of such classes. Instances of such pseudo-classes store the actual data resulting from the experiments, while still offering the users a consistent, friendly interface. The unique properties and variables of individual SOPs are addressed in a similar fashion. This approach has the obvious disadvantage that it requires many additional pseudo-data models to be defined: one for each possible measurement and property.

A simplified design of the most important data models associated with a research project is illustrated in figure 4.7. This model shows that a *research template* can be regarded as the root of the information tree, linking together the *SOPs*, *steps* and *measurements*, as well as associated *researchers* and *required materials* [14, p. 28].

Figure 4.8 illustrates an example of a research project, showing how the various instances of *SOPs*, *steps* and *measurements* are interconnected. Two pseudo-classes illustrate how unique properties for specific *SOPs* and *measurements* can be defined. The diagram omits several less important classes and fields, such as *researchers* and *materials*, in order to reduce the complexity of the illustration. Implementations of other *SOPs* and *measurements* (e.g. micro-CT scans and Histologies) follow a similar approach.

Section 5.1 discusses possible enhancements and alternative approaches to the basic data model concept, which can offer more advanced capabilities for implementing complex semantic structures.

Screenshots displayed in figures 4.9–4.15 illustrate several simple examples of how the wiki and data models look and behave from the point of view of a regular researcher. Figure 4.9 shows a regular wiki page, containing a structured data block alongside unstructured text, similar to how this would be organized in a paper lab notebook. The underlying editor interface is illustrated in figure 4.10, showing the data structure used by Strata. This relatively complicated syntax can be circumvented by using the developed data entry plugin, which is illustrated in figure 4.11, where a new *material* instance is entered. When the user saves this new data entry, the accompanying Strata data block is generated as shown in figure 4.12. When a user enters data which conflicts with the data model, the wiki issues a warning such as the one illustrated in figure 4.13.

A simple example illustrating the power of queries is shown in figure 4.14, which lists all historic and current research instances which use sheep as lab animals. This overview table is generated automatically based on the defined experiments and the required materials from the previous screenshots. The query which is responsible for generating the overview is shown in figure 4.15. Although this example table only contains two dummy experiments, a real world instance could be much larger. Performing such a query on data contained in paper lab notebooks requires a lot of labour and can introduce accidental errors when manually searching and copying the information. This process is simplified significantly by storing the data in a semantic wiki.

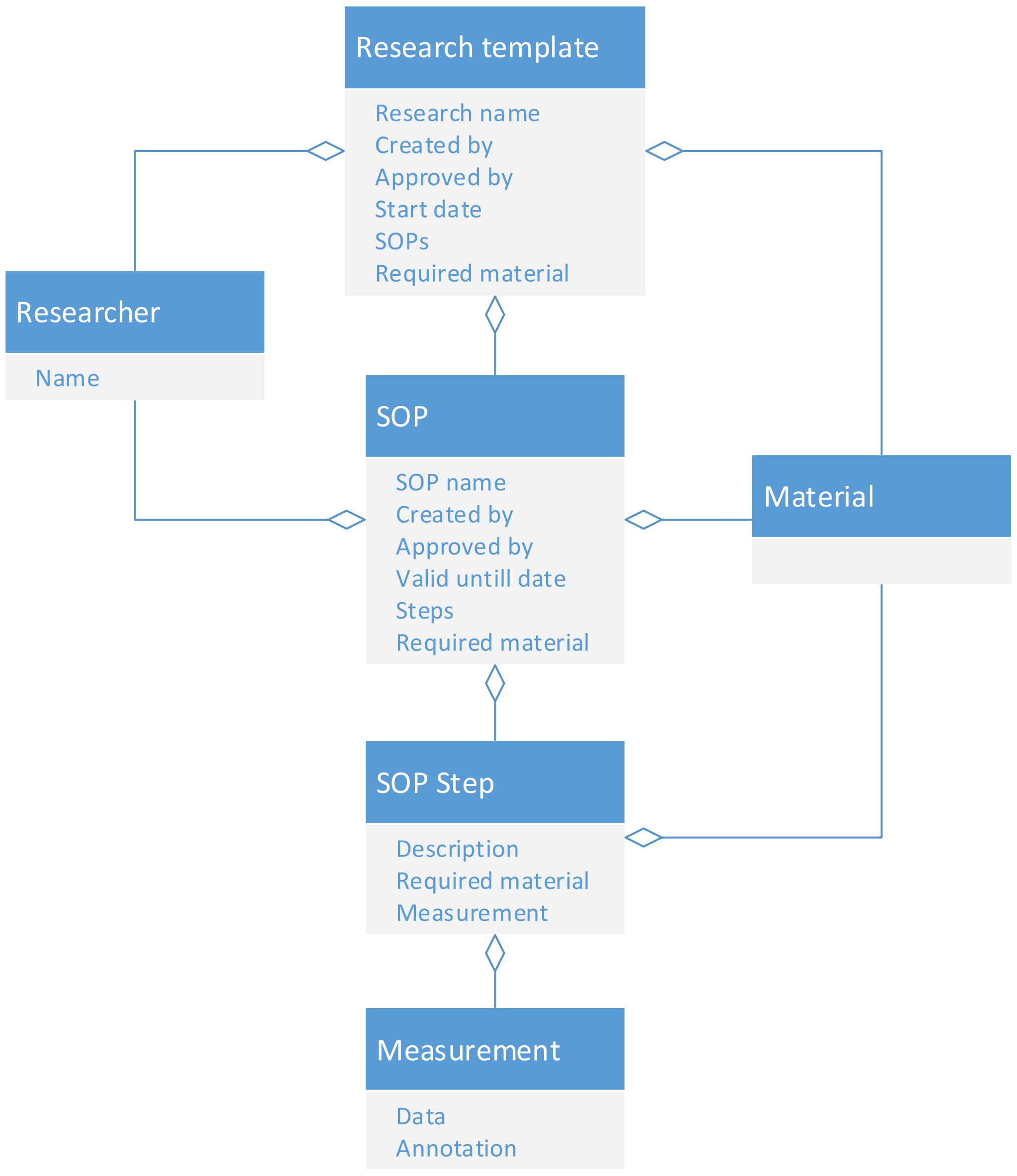


Figure 4.7: A simplified design of the various data models associated with a typical research track, such as the root research template, researchers, SOPs, steps, materials and measurements. These models include several common fields which are shared amongst most instances.

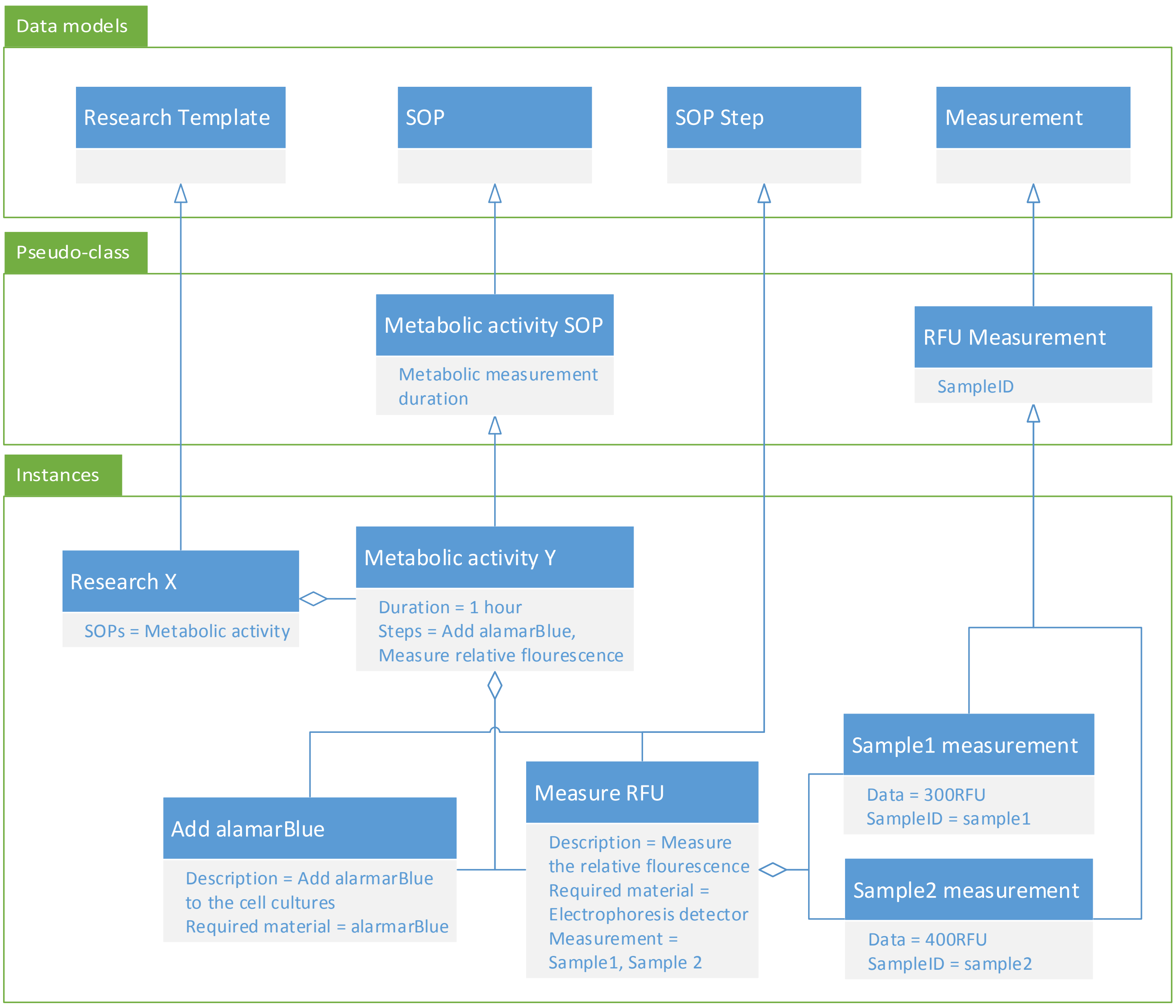
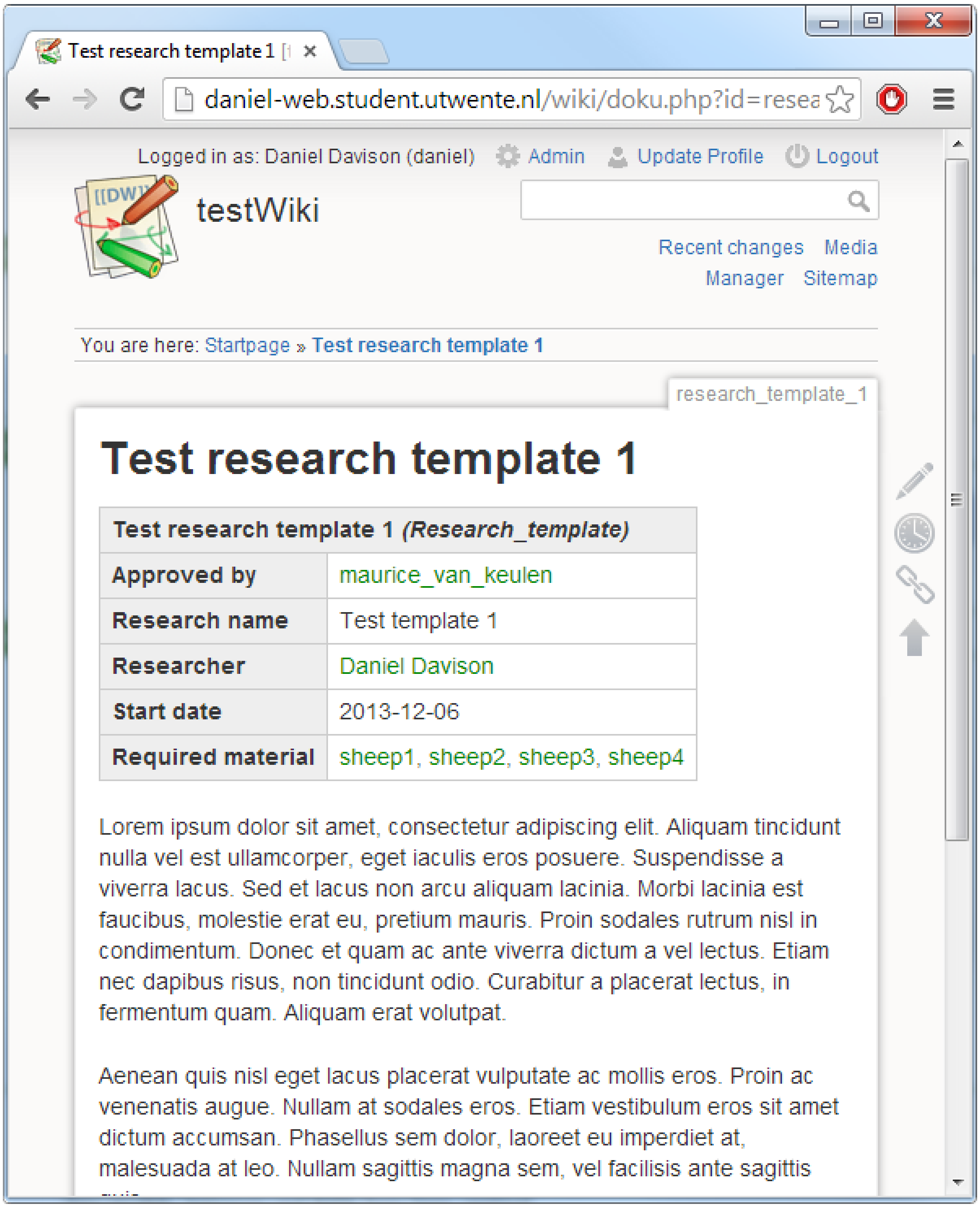

Data models
Research Template
SOP
SOP Step
Measurement
Pseudo-class
Metabolic activity SOP
Metabolic measurement duration
RFU Measurement
SampleID
Instances
Research X
SOPs = Metabolic activity
Metabolic activity Y
Duration = 1 hour
Steps = Add alamarBlue, Measure relative flourescence
Sample1 measurement
Data = 300RFU
SampleID = sample1
Measure RFU
Description = Measure the relative flourescence
Required material = Electrophoresis detector
Measurement = Sample1, Sample 2
Add alamarBlue
Description = Add alarmarBlue to the cell cultures
Required material = alarmarBlue
Sample2 measurement
Data = 400RFU
SampleID = sample2

Figure 4.8: An example instance of a simple research, illustrating the use of data models and pseudo classes. In order to keep this visualisation as compact as possible, several less important data models, instances and fields have been omitted.


Test research template 1 [

daniel-web.student.utwente.nl/wiki/doku.php?id=resea

Logged in as: Daniel Davison (daniel) Admin Update Profile Logout

testWiki

Recent changes Media Manager Sitemap

You are here: Startpage » Test research template 1

research_template_1

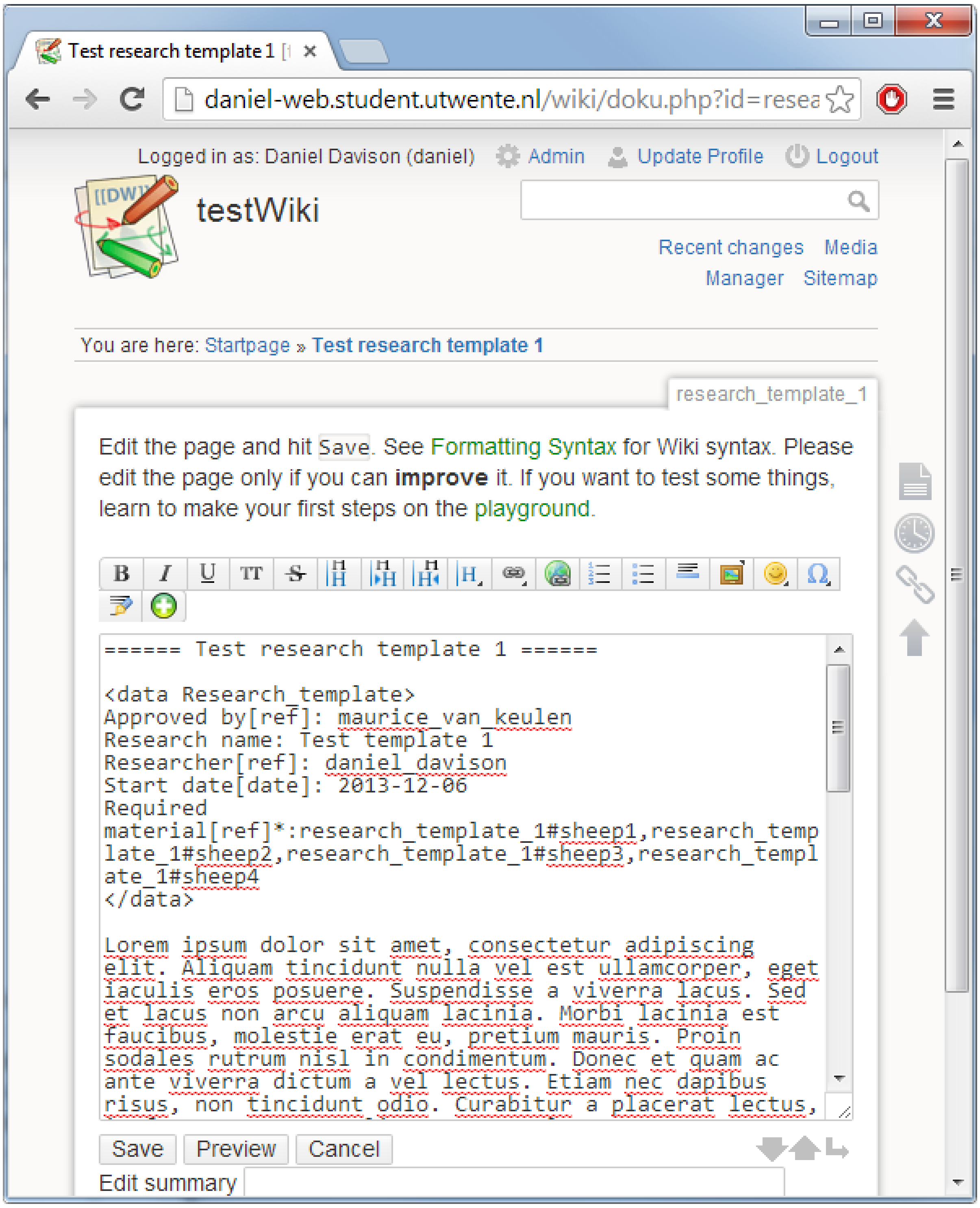
Test research template 1

Test research template 1 *(Research_template)*	
Approved by	maurice_van_keulen
Research name	Test template 1
Researcher	Daniel Davison
Start date	2013-12-06
Required material	sheep1, sheep2, sheep3, sheep4

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tincidunt nulla vel est ullamcorper, eget iaculis eros posuere. Suspendisse a viverra lacus. Sed et lacus non arcu aliquam lacinia. Morbi lacinia est faucibus, molestie erat eu, pretium mauris. Proin sodales rutrum nisl in condimentum. Donec et quam ac ante viverra dictum a vel lectus. Etiam nec dapibus risus, non tincidunt odio. Curabitur a placerat lectus, in fermentum quam. Aliquam erat volutpat.

Aenean quis nisl eget lacus placerat vulputate ac mollis eros. Proin ac venenatis augue. Nullam at sodales eros. Etiam vestibulum eros sit amet dictum accumsan. Phasellus sem dolor, laoreet eu imperdiet at, malesuada at leo. Nullam sagittis magna sem, vel facilisis ante sagittis

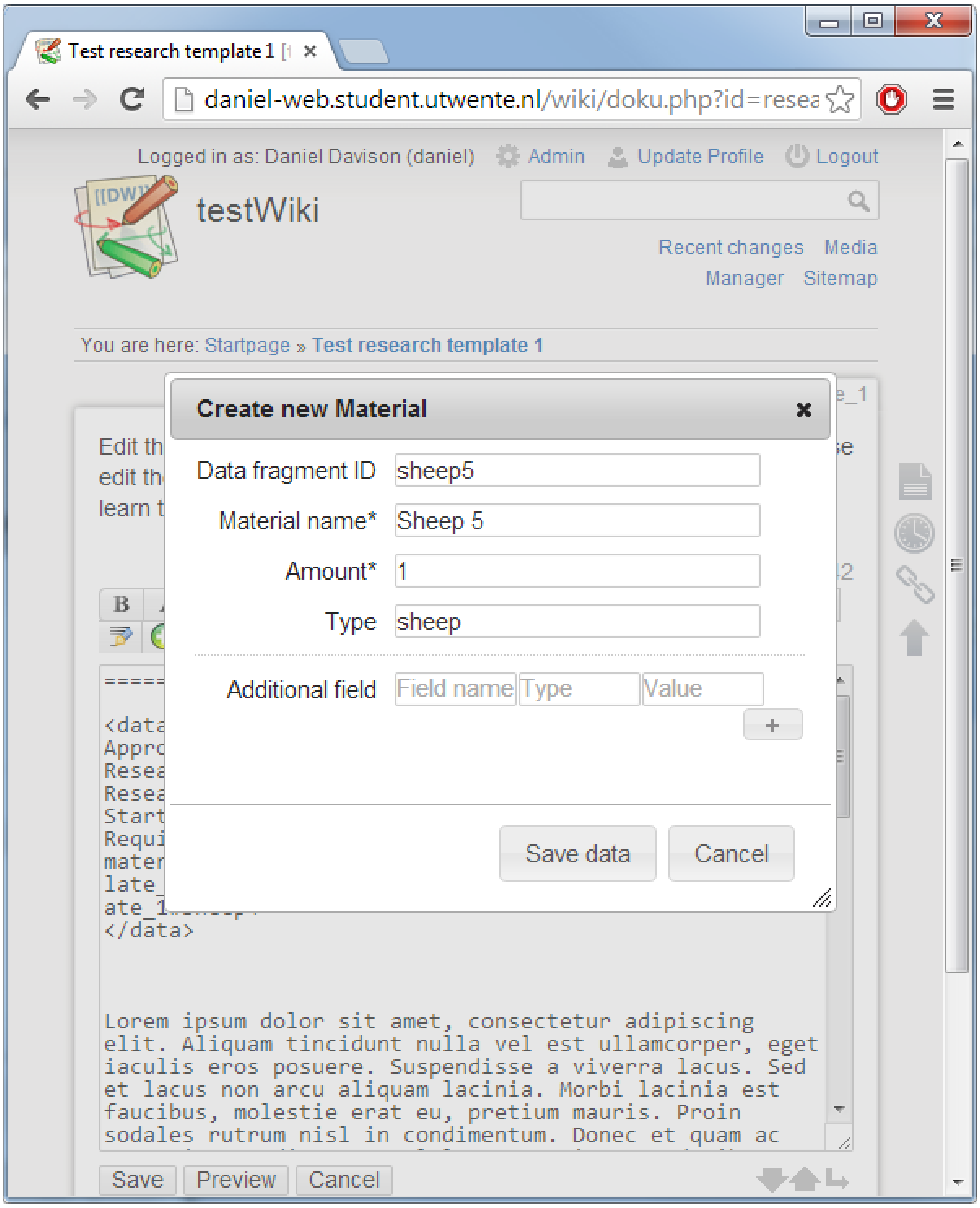
Figure 4.9: A screenshot of how an example research template might look in the wiki. The structured semantic information is contained in the table, alongside the unstructured textual information. The editor view of this same page is shown in figure 4.10.


Test research template 1
daniel-web.student.utwente.nl/wiki/doku.php?id=resea
Logged in as: Daniel Davison (daniel)
Admin
Update Profile
Logout
testWiki
Recent changes
Media Manager
Sitemap
You are here: Startpage » Test research template 1
research_template_1

Edit the page and hit Save. See Formatting Syntax for Wiki syntax. Please edit the page only if you can **improve** it. If you want to test some things, learn to make your first steps on the playground.

```
====== Test research template 1 ======

<data Research_template>
Approved by[ref]: maurice_van_keulen
Research name: Test template 1
Researcher[ref]: daniel_davison
Start date[date]: 2013-12-06
Required
material[ref]*:research_template_1#sheep1,research_temp
late_1#sheep2,research_template_1#sheep3,research_templ
ate_1#sheep4
</data>

Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Aliquam tincidunt nulla vel est ullamcorper, eget
iaculis eros posuere. Suspendisse a viverra lacus. Sed
et lacus non arcu aliquam lacinia. Morbi lacinia est
faucibus, molestie erat eu, pretium mauris. Proin
sodales rutrum nisl in condimentum. Donec et quam ac
ante viverra dictum a vel lectus. Etiam nec dapibus
risus, non tincidunt odio. Curabitur a placerat lectus,
```

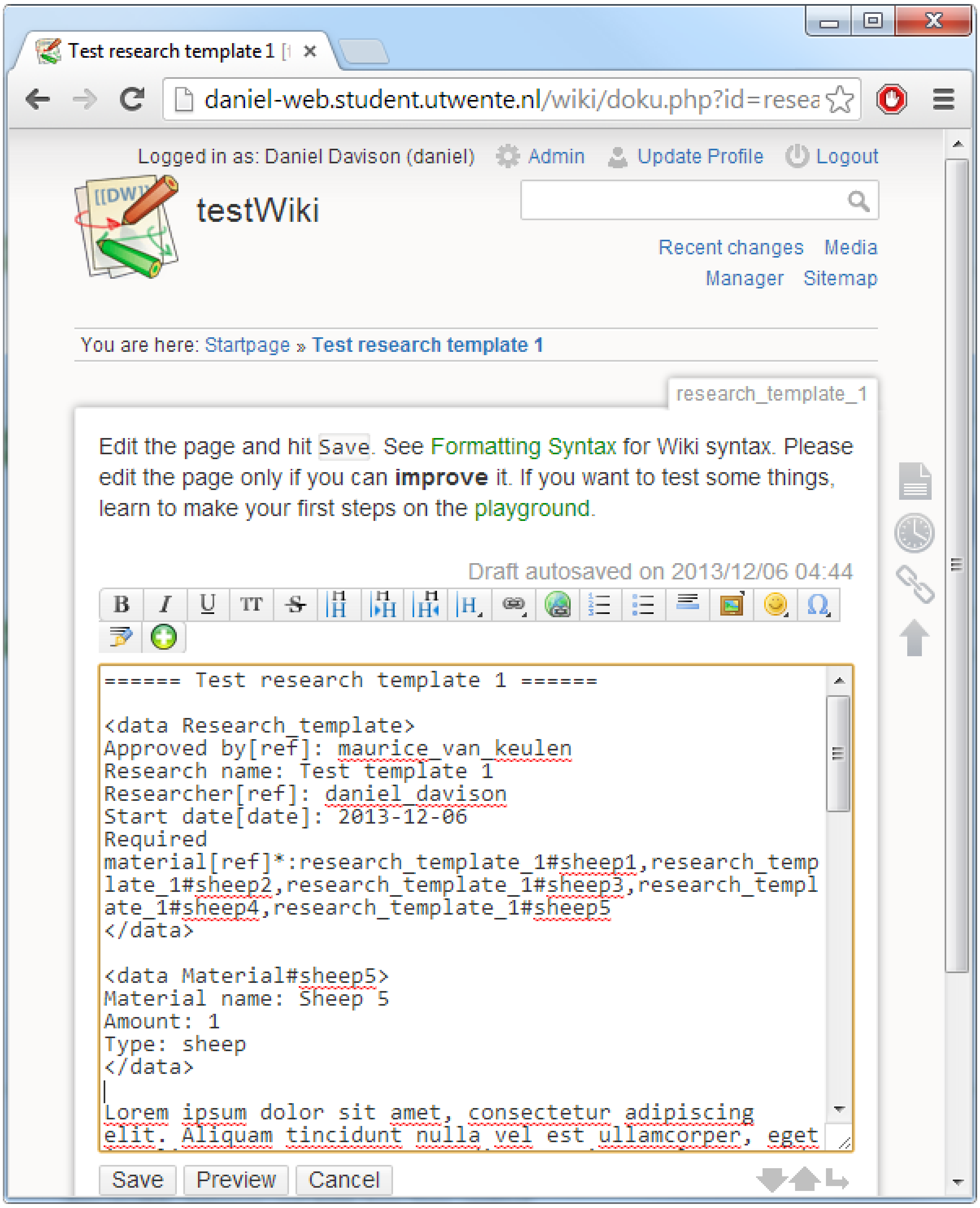
Figure 4.10: A screenshot of the editor interface of the wiki. The structured data is contained in a `<data>` block, while unstructured data is entered as regular text. An example of the data entry popup is shown in figure 4.11, while figure 4.12 illustrates the generated new data block after the popup has been saved.


Test research template 1 [
daniel-web.student.utwente.nl/wiki/doku.php?id=resea
Logged in as: Daniel Davison (daniel)
Admin
Update Profile
Logout
testWiki
Recent changes
Media Manager
Sitemap
You are here: Startpage » Test research template 1
Create new Material
Save data
Cancel
Save
Preview
Cancel

Data fragment ID	sheep5
Material name*	Sheep 5
Amount*	1
Type	sheep
Additional field	Field name / Type / Value

```
=====

<data
Appro
Resea
Resea
Start
Requi
mater
late_
ate_1
</data>


Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Aliquam tincidunt nulla vel est ullamcorper, eget
iaculis eros posuere. Suspendisse a viverra lacus. Sed
et lacus non arcu aliquam lacinia. Morbi lacinia est
faucibus, molestie erat eu, pretium mauris. Proin
sodales rutrum nisl in condimentum. Donec et quam ac
```

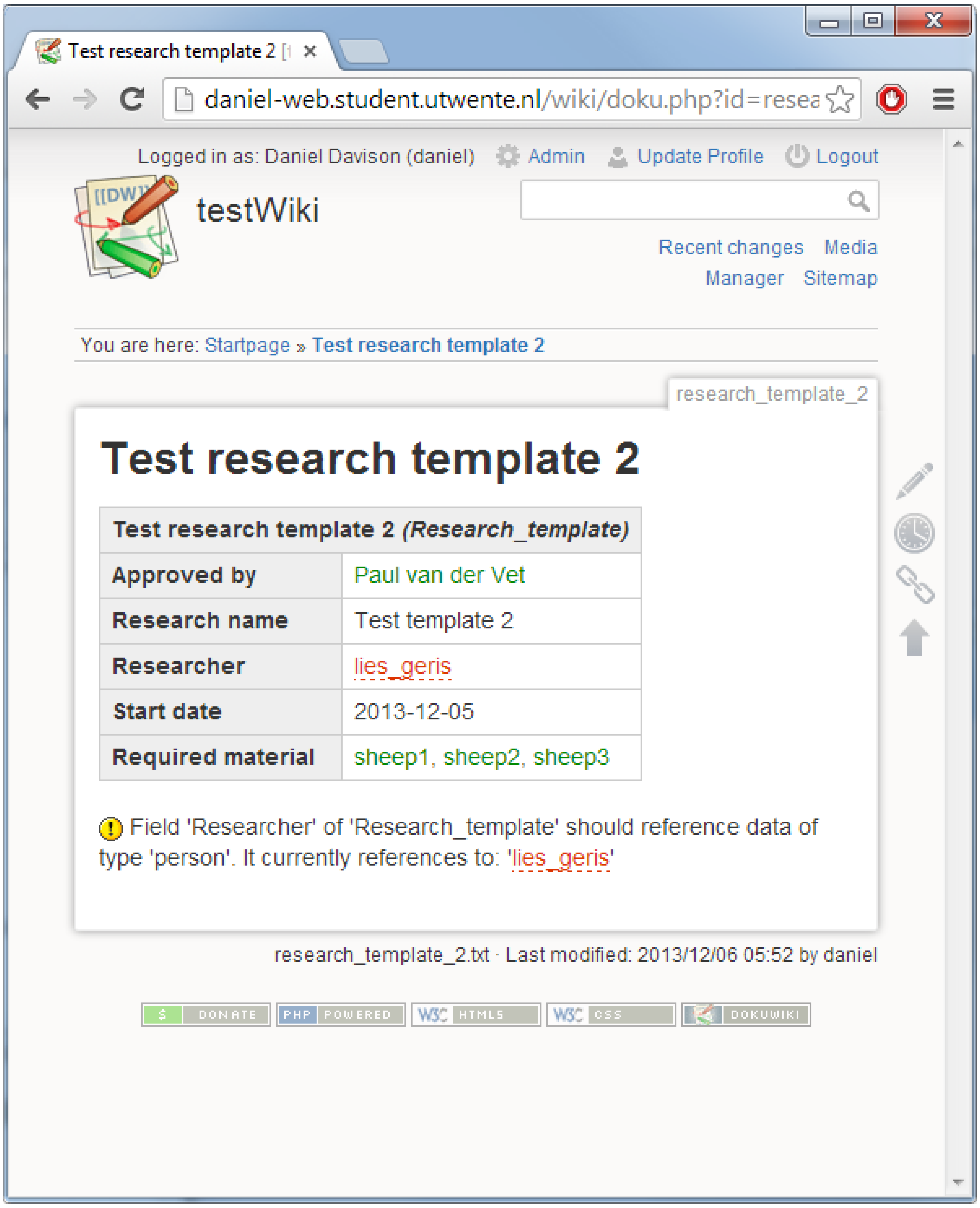
Figure 4.11: A screenshot of an example data entry popup, which is used to enter a new *material* to the existing template. The material being added is of type *sheep*. Figure 4.12 illustrates how the entered values are converted into a new `<data>` block when the popup is saved.


Test research template 1
daniel-web.student.utwente.nl/wiki/doku.php?id=resea
Logged in as: Daniel Davison (daniel)
Admin
Update Profile
Logout
testWiki
Recent changes
Media Manager
Sitemap
You are here: Startpage » Test research template 1
research_template_1
Draft autosaved on 2013/12/06 04:44
Save
Preview
Cancel

Edit the page and hit Save. See Formatting Syntax for Wiki syntax. Please edit the page only if you can **improve** it. If you want to test some things, learn to make your first steps on the playground.

```
====== Test research template 1 ======

<data Research_template>
Approved by[ref]: maurice_van_keulen
Research name: Test template 1
Researcher[ref]: daniel_davison
Start date[date]: 2013-12-06
Required
material[ref]*:research_template_1#sheep1,research_temp
late_1#sheep2,research_template_1#sheep3,research_templ
ate_1#sheep4,research_template_1#sheep5
</data>

<data Material#sheep5>
Material name: Sheep 5
Amount: 1
Type: sheep
</data>

Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Aliquam tincidunt nulla vel est ullamcorper, eget
```

Figure 4.12: A screenshot illustrating how the data entered into the popup shown in figure 4.11 is converted to the correct `<data>` block.


Test research template 2
daniel-web.student.utwente.nl/wiki/doku.php?id=resea
Logged in as: Daniel Davison (daniel)
Admin
Update Profile
Logout
testWiki
Recent changes
Media Manager
Sitemap
You are here: Startpage » Test research template 2
research_template_2

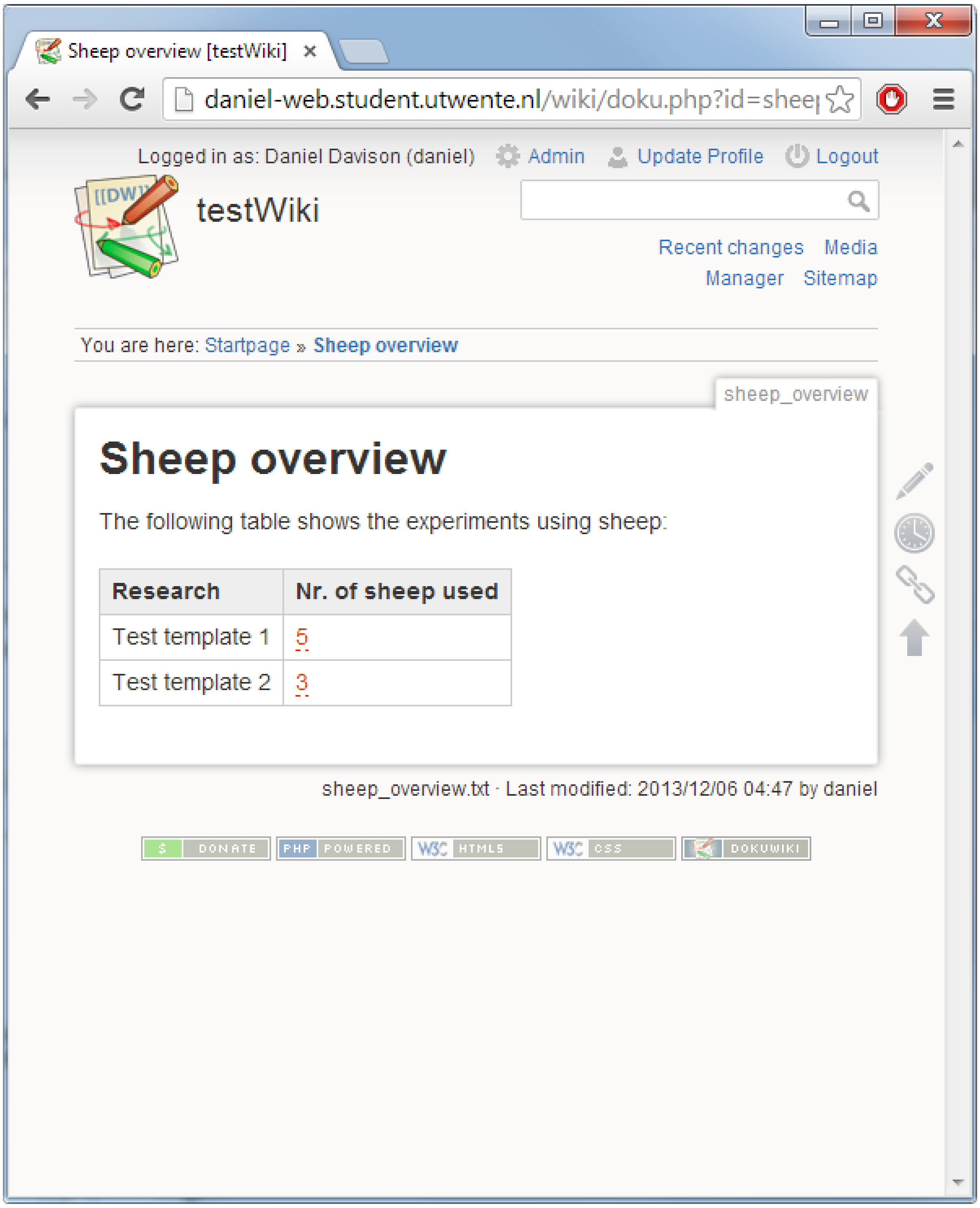
Test research template 2

Test research template 2 *(Research_template)*	
Approved by	Paul van der Vet
Research name	Test template 2
Researcher	lies_geris
Start date	2013-12-05
Required material	sheep1, sheep2, sheep3

Field 'Researcher' of 'Research_template' should reference data of type 'person'. It currently references to: 'lies_geris'

research_template_2.txt · Last modified: 2013/12/06 05:52 by daniel

Figure 4.13: A screenshot illustrating a warning message which is shown when a user has entered data which conflicts with the data model. In this case, the field *researcher* must reference an instance of type *person*, but the supplied person does not yet exist in the wiki. The solution is to create a new instance of *person* for the researcher in question.


Sheep overview [testWiki]
daniel-web.student.utwente.nl/wiki/doku.php?id=sheep
Logged in as: Daniel Davison (daniel) Admin Update Profile Logout
testWiki
Recent changes Media Manager Sitemap
You are here: Startpage » Sheep overview
sheep_overview

Sheep overview

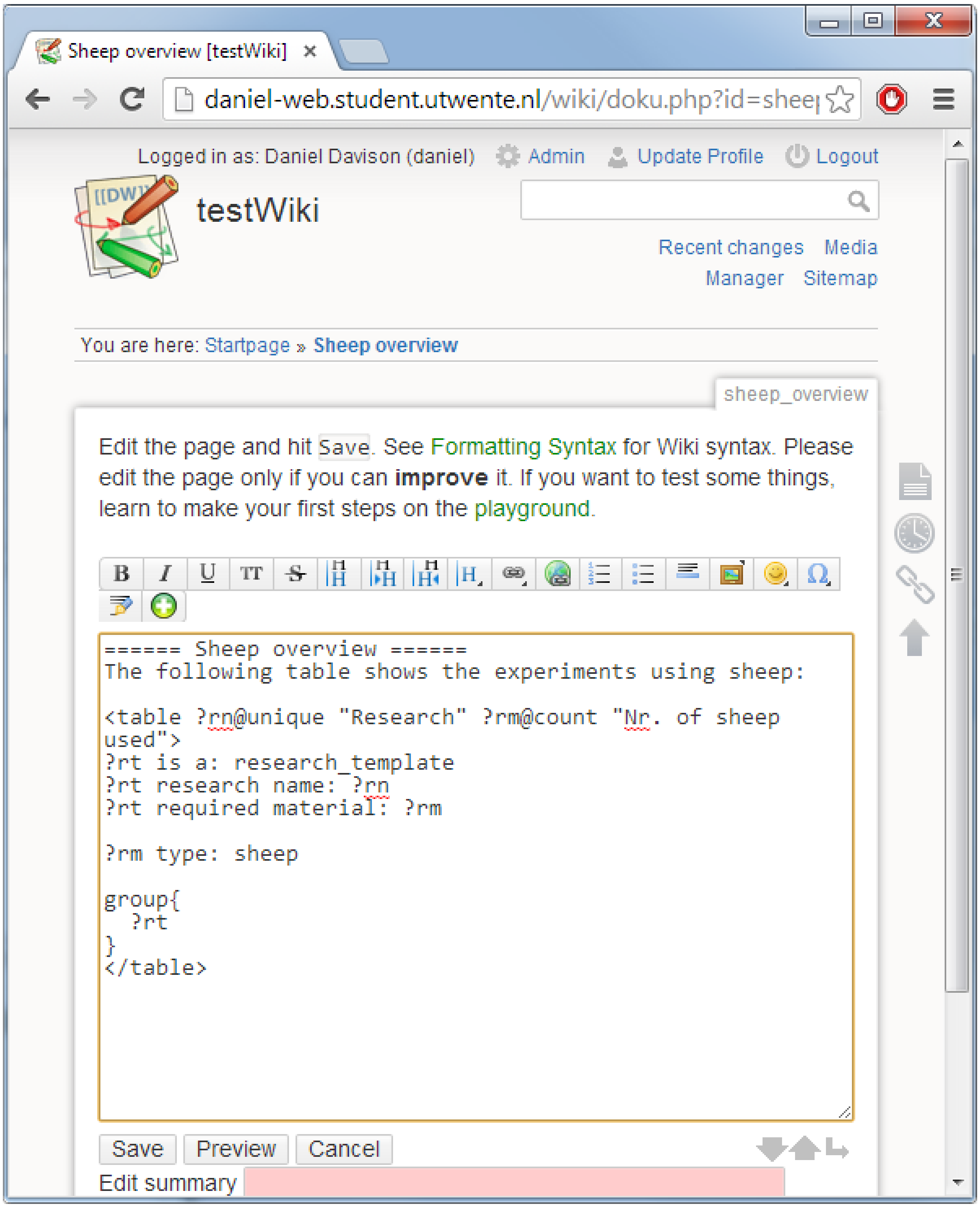
The following table shows the experiments using sheep:

Research	Nr. of sheep used
Test template 1	5
Test template 2	3

sheep_overview.txt · Last modified: 2013/12/06 04:47 by daniel

$ DONATE
PHP POWERED
W3C HTML5
W3C CSS
DOKUWIKI

Figure 4.14: A screenshot illustrating a simple example of an overview page, which uses a query to generate a table listing all experiments which have used sheep, and the amount of sheep which were used. The accompanying query is illustrated in figure 4.15. For this example, two dummy instances of *research_template* where created, each with several references to instances of *material.*


Sheep overview [testWiki]
daniel-web.student.utwente.nl/wiki/doku.php?id=sheep
Logged in as: Daniel Davison (daniel)
Admin
Update Profile
Logout
testWiki
Recent changes
Media Manager
Sitemap
You are here: Startpage » Sheep overview
sheep_overview
Save
Preview
Cancel
Edit summary

Edit the page and hit Save. See Formatting Syntax for Wiki syntax. Please edit the page only if you can **improve** it. If you want to test some things, learn to make your first steps on the playground.

```
====== Sheep overview ======
The following table shows the experiments using sheep:

<table ?rn@unique "Research" ?rm@count "Nr. of sheep
used">
?rt is a: research_template
?rt research name: ?rn
?rt required material: ?rm

?rm type: sheep

group{
  ?rt
}
</table>
```

Figure 4.15: A screenshot illustrating a simple query, which generates the overview table illustrated in figure 4.14. The query retrieves all instances of *research_template*, which have one or more references to a *material* of type *sheep*. The query then aggregates all results in order to calculate the sum of the *sheep*.

Chapter 5

Discussion and conclusion

The goal of this research is to create a platform with which researchers can construct an electronic lab notebook for organizing their workflow. To accomplish this, I propose a semantic wiki platform using a data model structure which is used to describe the various laboratory processes using flexible data classes and constraints. These processes include research templates and SOPs, as well as the experimental data resulting from measurements. In addition, this research shows how data models are used to keep track of the various materials and equipments in a laboratory. This makes it possible to construct more detailed accountancy records, such as the information on donor patients and the resulting cell lines. The current prototype focuses primarily on matching the flexibility of the paper lab notebook by using free-form data principles, as discussed in section 3.1.2, while offering significant advantages in collaboration and data searchability.

While designing this data model approach, extra attention was paid to the aspect of future extendibility, as discussed in section 3.3. By extending the way data models are defined in a future version, it becomes possible to integrate with existing ontologies and external (cloud) storage. As laboratories become more and more automated in the future, the lab notebook could be extended to include additional features such as speech and image recognition, automatic logbook generation or extensive integration with auxiliary lab equipment. Concepts such as hierarchical class definitions, hybrid data blocks and conditional fields and constraints allow for modelling more complex aspects of a laboratory's data workflow. Such additional enhancements to the data model structure are discussed further on in section 5.1.

The semantic wiki approach suggested in this research addresses some of the major bottlenecks currently present in the paper lab notebook equivalent: (these limitations are described in more detail in section 2.2)

- Archived information is relatively unstructured and difficult to retrieve
- Collaboration is hindered, especially when considering remote partnerships
- Organizational data (such as SOPs, cell lines, and patient- and animal records) is difficult to establish and regulate
- There is no possibility for automating tasks, such as retrieving measurements from a machine

Firstly, searchability of research data is increased due to the indexation of the unstructured content in the wiki and the formalization of the structured content in the semantic database. Secondly, research has shown that collaboration between researchers is facilitated due to the nature of the wiki platform [31, 28]. Finally, the use of data models makes it possible to disclose the intended structure of semantic data, increasing the overall clarity, usability and maintainability of the system.

Due to the user friendly interface, regular researchers can extend and modify the data model collection to reflect the dynamic workflows present in the laboratory. Since many scientific research laboratories follow some form of workflow in their daily business, it can be argued that the suggested approach using research templates, SOPs and measurements is applicable to a much broader domain than only the Prometheus department case study. As a matter of fact, such a semantic wiki platform using data models can be beneficial in almost any organization dealing with data or knowledge, as long as the important concepts from their domain can adequately be converted to similar data model constructs.

Although the proof of concept developed for this research illustrates how the data model approach can be applied to the workflow of a research laboratory, it is still somewhat limited in how it can be used in a real-world scenario. This is partly due to several usability concerns present in the current prototype. For instance, the user interface responsible for entering data is only available when entering *new* data, but not when updating or expanding existing data blocks. This means that users are still confronted with the underlying Strata syntax when maintaining the wiki. A future version of the system should make sure to use a similar user friendly interface for all stages of the data life cycle.

The current interface was developed with a small scale proof of concept in mind, and several user interaction constructs that were used do not scale well when the wiki grows larger. An example of this is the relatively straightforward approach that is used for creating a new instance of a data block. Currently there exist only several dozen data models, so the dropdown list is still clearly organized and finding the appropriate data block is quick and simple. This list will grow significantly larger as the wiki matures in a real-world setup, so an alternative approach is recommended. As previously discussed in section 5.1.4, it is possible to create an intelligent suggestion of relevant data blocks, effectively showing a filtered list of the entire set of available classes. Similarly, the usability of the autocomplete fields could be enhanced even further by adding a categorization in the results. The current implementation lists the results in alphabetical order, but a more intuitive approach could rank the results on relevancy, for instance.

A second major limitation of the current prototype is related to legal regulations related to electronic lab notebooks. Especially when clinical research is involved, a lab notebook is required to adhere to strict guidelines concerning the availability of data provenance, access restrictions and data integrity checks [26, 7, 6]. In the current form this semantic wiki solution can be seen more as a flexible and user friendly replacement for paper notebooks, to be used alongside a regular ELN. Future versions could be oriented at fulfilling such legal requirements, for instance by implementing constraints similar to those present in ELNs or by coupling the data gathered in the wiki with an underlying ELN storage system. In relatively open environment such as a wiki, it is often undesirable to enforce strong access restrictions and constraints on pages and data. That being said, most available wiki solutions (including DokuWiki) do offer access control with varying granularity suitable for situations where such an approach is necessary. In addition to such access restrictions, a wiki will typically maintain a history of all previous versions of a page, indicating if, what and by who data has been modified. On top of this versioning system, a future prototype could use a digital signature in order to prove data integrity. This approach is discussed in more detail later in this chapter.

Tracking provenance information in a semantic context such as a knowledge base is no straightforward task. In recent years, researchers have explored various methods for storing provenance information in the domain of e-science [10, 38]. A very promising, yet ambitious, attempt is the Open Provenance Model [32], which specifies a generic open standard for storing and exchanging provenance data by capturing the relationships and interactions between the concepts 'processes', 'artifacts' and 'agents'. A future version of this lab notebook could be developed to adhere to this model, in order to generate appropriate provenance data. More specialized approaches focus on generating and storing provenance information in a workflow context similar to the processes seen in a research laboratory [13]. Recent research has even shown the possibilities for automated logbook generation using cheap sensors and video feeds, effectively recording provenance information about the execution of the experiment itself [12].

These proposed methods for tracking the researcher's activities using sensors and video surveillance can be useful for automating laboratory work. They might pose ethical concerns, however. When the privacy of the scientists is not protected sufficiently, the recorded data might be misused for other purposes. The gathered data holds much more information about an individual scientist than might be seen at first glance. It is possible to perform statistical analysis on the timestamps obtained from sensors, for instance, to determine an individual researcher's performance in the laboratory. An unethical manager might see bad performance as a reason for firing someone, even though the gathered data might not reflect real-world performance. On the other hand, recorded video streams of the laboratory could be used for stalking if made available publicly. Employees should not have to fear for their privacy while working in the laboratory. A solution to these ethical concerns is to only make the raw sensor and video data accessible to the researcher in question, while providing anonymised and aggregated data to management. This anonymised data can still be used for calculating approximate performance statistics, but can not be used to single out individual researchers. This is obviously not applicable to actual research data and results, which ideally should be made publicly available.

An important caveat with this research is that a validation of the resulting prototype, in the form of a user research, has not taken place, partly due to usability concerns outlined above. It is therefore difficult to conclude how the developed solution behaves in a real-world setup, and whether or not it achieves the goal of supporting the data workflow of a laboratory. Although the system and data models were primarily designed and implemented based on observations made during a preliminary ethnographic research, an additional user study is essential before being able to conclude that these observations have been adequately addressed by the prototype. In addition to a user study performed at the Prometheus laboratory, it can be interesting to interview researchers of several alternative laboratories in order to investigate whether the proposed semantic wiki concepts for organizing workflow are indeed as generic as has been assumed.

That being said, it can be argued that the developed prototype addresses other important bottlenecks observed in the laboratory which exist with the regular paper lab notebooks. Searching through old research results, for instance, is significantly less time-consuming than before due to the digital indexing of the structured and unstructured data stored in the wiki. Collaboration between scientists will also be improved with this solution, since research results can be shared, accessed and edited more easily compared to the traditional paper notebook. In addition, it is expected that a more transparent connection between laboratory guidelines, such as SOPs, and resulting measurements will contribute to a better understanding of the workflow for newer members, which will ultimately lead to better overall adherence to such guidelines.

5.1 Data model enhancements

In order to support some of the more complex data model situations outlined in the previous chapter, it is necessary to introduce some changes to the way the data models are defined and handled.

5.1.1 Hierarchical data model structure

One solution is to lose the idea of defining distinct classes of data only via separate data model instances. Instead, it should be possible to extend the data model of an instance of data in such a way that descendants of this instance adhere to the updated fields and constraints set by all its ancestors. This can be accomplished by defining additional fields and constraints describing each instance, effectively opening the possibility for creating an elaborate hierarchical structure.

For instance, one could define an SOP to have generic fields such as *date of creation*. An instance of such an SOP, e.g. a histology staining procedure, could define additional fields such as *staining method*. A research template would then reference an instance of this histology SOP, which would contain fields defined in both the base SOP and the child SOP. The constraints would function in a similar fashion.

This hierarchical method is conceptually easy to understand and should be straightforward to implement. On its own, however, it will not powerful enough to offer all functionality required for the complete lab notebook data model.

5.1.2 Hybrid data blocks

Another extension to the regular class-based data model approach would be to create hybrid data blocks. Such data blocks could be multiple classes at once, combining both the fields and constraints of all classes in question. This opens up many more possibilities for representing complex data structures, for instance by introducing additional constraints that cover only specific combinations of classes.

Imagine for example an instance of an SOP which describes a CT scan experiment. The procedure describes several steps at which results are obtained. In the SOP itself, only a description of the steps is shown. Only at the actual moment that a new measurement is entered (e.g. a hybrid data block of *measurement* and *step 5* of *CT scan*), the associated fields and constraints are shown.

This can be organized as follows: The base *CT scan SOP* simply references various instances of the *step* data class. A generic class *measurement* contains fields and constraints shared by all measurement results, such as a *timestamp*, an *experiment ID*, and so on. For each specific step that should yield measurements, an extra set of fields is added to the data model. Using additional constraints, these fields will only be coupled to a data block which is of both indicated class types. It will now contain the fields of the specific step, the generic fields from the measurement class and the additional fields specified using the constraints.

This generic method offers all the functionality required for implementing the full lab notebook data model, but the various constraints acting upon multiple different classes will likely become quite complex and confusing when used inappropriately. This method should preferably be used sparingly when defining data models and can perhaps be applied in combination with the hierarchical structure to produce more organized data models. Some fields could possibly already be filled automatically from the parent class.

5.1.3 Referential- and conditional-based fields and constraints

Some data structures might benefit from the option of showing various fields from other sources, triggered by referencing from one class to another. For instance, class *A* might have some fields defined which have a constraint which states that they should be filled out and displayed only when being referenced by a specific class *B*.

This construct provides an alternative to some of the concepts outlined in previous sections. This could for instance be used for displaying parameters for specific SOPs when they are referenced in a research template.

Additionally, it could be used for measurements associated with a particular step of a procedure. For example: if step *x* of a cell seeding SOP results in the cell seeding density, a referential-based input field could be coupled to the corresponding step *x*. This additional input field would automatically be added to the interface when a reference is made to the particular step from within an experiment. The resulting fields could either be added as an additional data fragment on the page, or directly into the main data block, depending on the situation.

This functionality is less complex and easier to maintain than the hybrid data blocks solution, but is more aimed at solving problems encountered in the lab notebook setting, and is therefore less generic in design. The solution could possibly be made more generic by adding various other conditions other than reference. It can be combined with the previously mentioned solutions, together forming quite an extensive collection of possibilities for defining advanced, complex data models.

5.1.4 Conditional (smart) data block suggestions

If all above solutions are implemented and made available to the user, it might become an overwhelmingly complex task to create and edit data. In order to increase the usability, the system should be able to suggest relevant data blocks which the user is most likely going to need at a specific moment, while hiding irrelevant blocks. This should keep the interface as clean as possible, while possibly providing the user with some intuitive guidance on how to use the system.

For instance, if the user is creating an instance of a class *A*, and the data model of class B contains some conditional fields when referenced from class *A*, it makes sense to present the user with an option of creating a reference to this class. Another situation would be when there exist additional fields or constraints for a hybrid combination of class *A* and class *B*. The interface could suggest such a hybrid option when the user creates an instance of either class.

Besides these automatic suggestions, the data model should offer similar functionality for conditionally suggesting certain data blocks. For instance, an SOP should always consist of various steps. It makes sense to store each step on the same page as a data fragment (although it is possible to create and store the steps on separate pages). The data model could contain an additional rule which states that when a user is creating an instance of an SOP, he is explicitly presented with the option of adding associated *step* data fragments, even though there might not be a very strong relationship defined in the data model. This gives the designer of the data model some extra control over which options to present to the user.

5.2 Future work

Data integrity In some cases a researcher might need to prove that the data he has entered in his lab notebook has not been altered or tampered with. For instance, this is the case when he wishes to apply his methods in a clinical setup. In a paper lab notebook it is usually easy to spot when data has been changed, but in the digital equivalent this can be somewhat more difficult, especially if an outside source has deliberately tampered with data.

An anti-tamper mechanism is required for any data stored in the notebook. Although the wiki platform maintains a revision history for each page, this information can be manipulated by malicious administrators or hackers. A more robust approach is necessary in order to *prove* that data integrity has not been violated.

A possible implementation could rely on the concept of a *cryptographic digital signature*. Simply put, a digital signature makes it possible to mathematically encrypt data in such a way that its authenticity can be proven. A valid digital signature can prove that the data in question was written by the owner of the signature, and that the data has not been altered since the signature was set [33]. Although this method can be used to detect *if* data on a page has been altered, it should be noted that it cannot detect exactly *which* data has been altered. For this purpose, the regular wiki revision history system can be consulted as long as it has not been tampered with.

Research timeline A research workflow consists of many individual tasks, each of which may take several weeks. By placing all tasks on a visual timeline it can be easy to spot potential planning errors (for instance, cells have not yet been fully cultured, when they are already required in a subsequent step). Errors can then be addressed in the planning phase of the research without much trouble.

The researcher could set start and end dates for SOPs in the research template, which the system can use for automatically generating a visual timeline. This timeline can be supplemented with information about the input and output data at various time points, effectively building a workflow tool [45]. A more advanced option would work the other way around: let the user construct the timeline using a graphical interface, while the notebook generates the underlying data fields and constructs a research template and associated workflow based on the specified visual timeline.

Query generating interface Since the digital lab notebook is intended to enhance the ease of searching through research data, it should provide some interface with which search queries can be constructed. Although it is not fully clear which type of questions the system should be able to answer in the future, it is possible to make an educated guess:

- Simple queries about some aspect of a research project. For instance: "Give me all experiments that have used the CopIOs scaffold in combination with FBS serum."
- Queries requesting specific measurements. For instance: "Give me the metabolic activity measurements from the in vitro scaffolds from experiment x in weeks 2, 4, 6 and 8 (and generate a graph)."

- Comparing data between researches. For instance: “Compare the growth curves of cultured cells originating from research x with those from research y (and show me the differences in the applied methods).”
- Aggregated data from multiple researches. For instance: “Give me an overview of the metabolic activity measurements, gene expression, PCR and bone formation percentages obtained in researches conducted between 2009 and 2012.” Or: “Give me all growth statistics of cell lines x, y, and z.”

Currently it *is* possible to answer most of these queries using the paper notebooks. However, it can be an annoying and tedious task to gather the data spread over multiple notebooks written by many different researchers. Additionally, it is plausible that errors are made when searching for and copying such information from one notebook into the other. Therefore it can be argued that *any* queries executed successfully on the digital lab notebook are a significant improvement over the current situation, simply because of the time and effort that can be saved, and the lower risk of introducing errors.

Future researchers might utilize the data stored in the wiki in novel ways which are impossible to perform in the current situation using paper notebooks. Advanced big data analysis methods, machine learning and searching for statistical patterns are examples of such applications.

Auto-evolving data models A good starting point for creating a semantic wiki is to manually define the most important data models and relationships based on the observed workflow, as was done in this study. It can (and, most likely, will) be the case that data models gradually evolve over time, as procedures in the lab are changed and new techniques are adopted. Instead of relying on researchers to manually update the data models accordingly, it would be interesting to investigate whether it is possible to automatically adapt data models when changing trends are detected in the accompanying data structures.

For instance, at a certain point in time a manually defined data model for class A has fields k, l and m. After having used this class for several months, researchers frequently begin extending their instances of class A with an additional field x. An adaptive learning mechanism could recognize this evolution and could automatically update the data model to incorporate the detected additional field x. From that point onward, future instances of this class would include the additional field by default. This mechanism could work similarly for types, references and other constraints. The system should be careful to take into account when such data model mutations have taken place, since historically created data instances will still reflect the old data model and would most likely generate warnings when validated against the evolved data model. Contrariwise, in some cases it might in fact be desirable to retroactively validate historical data against an evolving data model, for instance when the updated data model corrects errors which were present in previous versions. A powerful versioning or data provenance system should be in place in order to facilitate this behaviour [46, 27, 34].

Some previous research has been done in the area of automatic data model updating. In the domain of semantic web services, Howard et al. have proposed a framework which focuses on automatically detecting and updating available specifications of services using a heuristic approach [22, 23]. It would be interesting to investigate whether the approach used in their framework can be applied to the domain of workflow data models.

A different trend which has spiked recent interest is so-called folksonomy, in which collaborative keyword tagging (usually in a social context) leads to indexing of associated information [20]. The folksonomy principle assumes that when a majority of users independently associate a specific keyword with an object (e.g. a photograph), this keyword must hold some semantic information about the object. This principle of collaborative tagging can perhaps be seen as an analogy for collaborative data model evolution: when a majority of users associates field x with class A, this can most likely be updated in the underlying data model

Epilogue

As I enter the laboratory I briefly waive my badge in front of the scanner. I am promptly greeted by my digital lab assistant Elna. "Good morning doctor Davison, today is the thirteenth of December, 2023." Her name originates from the term 'Electronic Lab Notebook', which was the very first version of the advanced digital assistant currently used in laboratories worldwide. "A lot has changed since I first started here, today is my tenth anniversary at this laboratory," I think to myself. While I walk to my bench, Elna informs me about the current state of affairs. "You have a new message concerning your research on stem cells LTG-22-HB, I have sent it to your personal workbench."

My bench is a lot different from the bench I used to work at. Instead of being cluttered with papers, equipment and tools, it displays an up-to-date overview of all my running experiments and is used to communicate with the various laboratory machines controlled by Elna. The displayed information is automatically gathered from the international biochemical knowledge base, a storage system used by researchers worldwide to describe, store and share research and data.

I open the message, which was sent by my Japanese colleagues. "We have been able to verify the promising results you have obtained for sample LTG-22-HB, our data has been added to the research page in question," they write. "We have cross-referenced the results with similar research conducted by the Americans, and have identified an unknown gene which might be interesting for closer examination. You should take a look at the graphs we have enclosed in the research entry." I open the research page on my bench, and navigate to the graph. "Elna, could you initiate a detailed gene analysis for gene 36 of sample LTG-22-HB please. I need the results by the end of this week," I tell my digital assistant. I can faintly hear the analysis equipment warming up in the room next door.

I ask Elna what the rest of my schedule for today looks like. "You have an appointment at 11:00 to discuss the results from last week's experiments," she tells me. "The cells you requested have been in culture for about a week and will be available by about 14:00, however you still need to define the subsequent experimental procedures you wish to perform on these cells," she continues. "But first I need you to review the CT scans you asked me to perform last evening. I have performed preliminary analysis on most samples, and have concluded that the third and fifth batches yield the most healthy results. I have displayed the results on your workbench, please review and confirm them so I can publish the results." She pauses briefly, as if she is unsure about her next words. "The results from batch 12 were highly irregular." This spikes *my interest. Since batch 12 was designed by my colleague Mary, who is currently on vacation, I decide to gather additional information.*

"Elna, please locate previous experiments using the cell line from batch 12 and cross reference the results with those currently obtained." After a short delay, Elna displays an empty list on my bench. "The provenance information of this cell line indicates that Mary recently revived and cloned several old cells which have not been used in experiments since at least ten years," she is able to inform me. In Mary's research proposal I read that she chose this specific cell line because she wishes to redo an experiment which failed all those years ago. Due to new technological insights, she hopes to be able to complete the work she started back then. I ask Elna to search for the experiment descriptions from ten years back, hoping that the experiments were conducted during the transition period from paper to digital lab notebooks. "The experiment in question was not stored in digital format, but I have narrowed it down to one of five possible notebooks," she tells me. Disheartened, I grab a flash light and my lunch bag. "I'm off to the old archive, please cancel my appointment and clear my schedule for this afternoon. This may take a while," I inform her. A cold chill runs over my spine as I descend the steps to the basement.

References

[1] Bone4Kids. `http://www.bone4kids.be/`, Nov. 2013.

[2] Halo extension to Semantic MediaWiki. `http://sourceforge.net/projects/halo-extension/`, Sept. 2013.

[3] MediaWiki. `http://www.mediawiki.org/wiki/MediaWiki`, Sept. 2013.

[4] Semantic MediaWiki. `http://semantic-mediawiki.org/`, Sept. 2013.

[5] Wikipedia - Ontoprise GmbH. `http://en.wikipedia.org/wiki/Ontoprise_GmbH`, Nov. 2013.

[6] 11 Electronic records; Electronic signatures. Number 21 C.F.R. pt. 1A. U.S. Food and Drug Administration, 1997.

[7] 820 Quality system regulation. Number 21 C.F.R. pt. 1H. U.S. Food and Drug Administration, 1996.

[8] P. Alper, K. Belhajjame, C. Goble, and P. Karagoz. Small Is Beautiful: Summarizing Scientific Workflows Using Semantic Annotations. In *2013 IEEE International Congress on Big Data*, pages 318–325, 2013.

[9] S. Auer, S. Dietzold, and T. Riechert. Ontowiki - a tool for social, semantic collaboration. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749. Springer Berlin Heidelberg, 2006.

[10] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey. In *ACM Comput. Surv.*, volume 37, pages 1–28, New York, NY, USA, Mar. 2005. ACM.

[11] M. N. K. Boulos. Semantic Wikis: A comprehensible introduction with examples from the health sciences. *Journal of Emerging Technologies in Web Intelligence*, 1(1):94–96, 2009.

[12] B. Brooks, A. Thorn, M. Smith, P. Matthews, S. Chen, B. O'Steen, S. Adams, J. Townsend, and P. Murray-Rust. Ami - The chemist's amanuensis. *Journal of Cheminformatics*, 3(1):1–10, 2011.

[13] S. B. Davidson, A. Eyal, B. Ludäscher, T. M. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in Scientific Workflow Systems. In *IEEE Data Eng. Bull.*, volume 30, pages 44–50, 2007.

[14] D. Davison. Investigating the data generating processes in a biomedical research laboratory. Technical report, Department of Human Media Interaction, University of Twente, April 2013.

[15] Standard Guide for Laboratory Informatics. Number E1578 - 13. ASTM International, 2013.

[16] Y. Gil. Workflow composition: Semantic representations for flexible automation. In I. Taylor, E. Deelman, D. Gannon, and M. Shields, editors, *Workflows for e-Science*, pages 244–257. Springer London, 2007.

[17] N. Goddard, R. Macneil, and J. Ritchie. eCAT: Online electronic lab notebook for scientific research. *Automated Experimentation*, 1(1):1–7, 2009.

[18] A. Gohr. DokuWiki. `https://www.dokuwiki.org/dokuwiki`, June 2013.

[19] A. Gohr. DokuWiki - Structured Data Plugin. `https://www.dokuwiki.org/plugin:data`, Sept. 2013.

[20] T. Gruber. Ontology of folksonomy: A mash-up of apples and oranges. *International Journal on Semantic Web and Information Systems*, 3(1):1–11, 2007.

[21] N. Heino, S. Dietzold, M. Martin, and S. Auer. Developing Semantic Web Applications with the OntoWiki Framework. In T. Pellegrini, S. Auer, K. Tochtermann, and S. Schaffert, editors, *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 61–77. Springer Berlin Heidelberg, 2009.

[22] R. Howard and L. Kerschberg. A Framework for Dynamic Semantic Web Services Management. In *Int. J. Cooperative Inf. Syst.*, volume 13, pages 441–485. World Scientific, 2004.

[23] R. Howard and L. Kerschberg. A knowledge-based framework for dynamic semantic web services brokering and management. In *15th International Workshop on Database and Expert Systems Applications*, pages 174–178. IEEE, IEEE Computer Society, 2004.

[24] G. Hughes, H. Mills, D. De Roure, J. G. Frey, L. Moreau, M. C. Schraefel, G. Smith, and E. Zaluska. The semantic smart laboratory: a system for supporting the chemical escientist. *Organic & Biomolecular Chemistry*, 2(22):3284–3293, 2004.

[25] Good Clinical Practice: Consolidated Guidance. Number ICH E6. International Conference on Harmonization (ICH), Apr. 1996.

[26] General requirements for the competence of testing and calibration laboratories. Number ISO/IEC 17025:2005. ISO, Geneva, Switzerland, 2005.

[27] W. Jaziri. A Methodology for Ontology Evolution and Versioning. In *Third International Conference on Advances in Semantic Processing, 2009. SEMAPRO '09*, pages 15–21, 2009.

[28] G. Jiang, H. R. Solbrig, D. Iberson-Hurst, R. D. Kush, and C. G. Chute. A collaborative framework for representation and harmonization of clinical study data elements using semantic mediawiki. *AMIA Summits on Translational Science Proceedings*, 2010:11–15, 2010.

[29] R. D. King, M. Liakata, C. Lu, S. G. Oliver, and L. N. Soldatova. On the formalization and reuse of scientific research. *Journal of The Royal Society Interface*, 8(63):1440–1448, 2011.

[30] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009.

[31] B. Leuf and W. Cunningham. *The Wiki way: quick collaboration on the Web.* Addison-Wesley Professional, Boston.

[32] L. Moreau, J. Freire, J. Futrelle, R. McGrath, J. Myers, and P. Paulson. The Open Provenance Model: An Overview. In J. Freire, D. Koop, and L. Moreau, editors, *Provenance and Annotation of Data and Processes*, volume 5272 of *Lecture Notes in Computer Science*, pages 323–326. Springer Berlin Heidelberg, 2008.

[33] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43. ACM, 1989.

[34] N. Noy, A. Chugh, W. Liu, and M. Musen. A framework for ontology evolution in collaborative environments. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 544–558. Springer Berlin Heidelberg, 2006.

[35] G. Piho, J. Tepandi, and M. Parman. Towards LIMS (Laboratory Information Management Systems) software in global context. In *MIPRO, 2012 Proceedings of the 35th International Convention*, pages 721–726, 2012.

[36] G. Piho, J. Tepandi, M. Roost, M. Parman, and V. Puusep. From archetypes based domain model via requirements to software: Exemplified by LIMS Software Factory. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 570–575, 2011.

[37] M. Rubacha, A. K. Rattan, and S. C. Hosselet. A Review of Electronic Laboratory Notebooks Available in the Market Today. *Journal of the Association for Laboratory Automation*, 16(1):90–98, 2011.

[38] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-Science. In *SIGMOD Record*, volume 34, pages 31–36, 2005.

[39] A. Slominski. Flexible Creation and Adaptive Execution of Scientific Workflows in Cloud and Grid Environments by Using Web 2.0-Based Electronic Lab Notebook Metaphor. In *6th World Congress on Services (SERVICES-1), 2010*, pages 326–327, 2010.

[40] L. N. Soldatova, A. Clare, A. Sparkes, and R. D. King. An ontology for a Robot Scientist. *Bioinformatics*, 22(14):464–471, 2006.

[41] L. N. Soldatova and R. D. King. An ontology of scientific experiments. *Journal of the Royal Society Interface*, 3(11):795–803, 2006.

[42] D. H. Sonnenwald. Scientific collaboration. *Annual review of information science and technology*, 41(1):643–681, 2007.

[43] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic Wikipedia. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 585–594, New York, NY, USA, 2006. ACM.

[44] B. Wanders. Strata - a Semi-Structured Data plugin for Dokuwiki. `https://github.com/bwanders/dokuwiki-strata`, Aug. 2013.

[45] I. Wassink. *Work flows in life science.* PhD thesis, University of Twente, Enschede, January 2010.

[46] C. Xie, L. Jiang, and H. Cai. Instance-Driven Ontology Evolution Mechanism towards Enterprise Data Management. In *8th International Conference on e-Business Engineering (ICEBE), 2011 IEEE*, pages 24–30. IEEE Computer Society, 2011.