Supporting the Workflow of Archaeo-related Sciences by Providing Storage, Sharing, Analysis, and Retrieval Methods

Dissertation zur Erlangung des Doktorgrades
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

vorgelegt von

Daniel Kaltenthaler

München, den 15.02.2018

Tag der Einreichung: 15.02.2018

Erstgutachter: Prof. Dr. Peer Kröger
Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Datenbanksysteme und Data Mining

Zweitgutachter: Prof. Dr. Andreas Henrich
Otto-Friedrich-Universität Bamberg
Lehrstuhl für Medieninformatik

Drittgutachter: Prof. Dr. Dr. Joris Peters
Staatssammlung für Anthropologie und Paläoanatomie München
Abt. Paläoanatomie

Vorsitz: Prof. Dr. François Bry
Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Programmier- und Modellierungssprachen

Tag der Disputation: 27.04.2018

Eidesstattliche Versicherung

(siehe Promotionsordnung vom 12.07.2011, § 8, Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

München, den 15.02.2018

Daniel Kaltenthaler

Contents

Abstract

The recovery and analysis of material culture is the main focus of archaeo-related work. The corpus of findings like rest of buildings, artifacts, human burial remains, or faunal remains is excavated, described, categorized, and analyzed in projects all over the world. A huge amount of archaeo-related data is the basis for many analyses. The results of analyzing collected data make us learn about the past.

All disciplines of archaeo-related sciences deal with similar challenges. The workflow of the disciplines is similar, however there are still differences in the nature of the data. These circumstances result in questions how to store, share, retrieve, and analyze these heterogeneous and distributed data. The contribution of this thesis is to support archaeologists and bioarchaeologists in their work by providing methods following the archaeo-related workflow which is split in five main parts.

Therefore, the first part of this thesis describes the xBook framework that has been developed to gather and store archaeological data. It allows creating several database applications to provide necessary features for the archaeo-related context. The second part deals with methods to share information, collaborate with colleagues, and retrieve distributed data of cohesive archaeological contexts to bring together archaeo-related data. The third part addresses a dynamic framework for data analyses which features a flexible and easy to be used tool to support archaeologists and bioarchaeologists executing analyses on their data without any programming skills and without the necessity to get familiar with external technologies. The fourth part introduces an interactive tool to compare the temporal position of archaeological findings in form of a Harris Matrix with their spatial position as 2D and 3D site plan sketches by using the introduced data retrieval methods. Finally, the fifth part specifies an architecture for an information system which allows distributed and interdisciplinary data to be searched by using dynamic joins of results from heterogeneous data formats. This novel way of information retrieval enables scientists to cross-connect archaeological information with domain-extrinsic knowledge. However, the concept of this information system is not limited to the archaeo-related context. Other sciences could also benefit from this architecture.

Zusammenfassung

Die Wiederherstellung und Analyse von materieller Kultur ist der Schwerpunkt archäologischer Arbeit. Das Material von Funden wie Gebäudereste, Artefakte, menschliche Überreste aus Bestattungen oder tierische Reste wird in Projekten auf der ganzen Welt ausgegraben, beschrieben, kategorisiert und analysiert. Die große Anzahl an archäologischen Daten bildet die Grundlage für viele Analysen. Die Ergebnisse der Auswertung der gesammelten Daten gibt uns Aufschluss über die Vergangenheit.

Alle Disziplinen der archäologischen Wissenschaften setzen sich mit ähnlichen Herausforderungen auseinander. Der Arbeitsablauf ist in den einzelnen Disziplinen ähnlich, jedoch gibt es aufgrund der Art der Daten Unterschiede. Das führt zu Fragestellungen, wie heterogene und verteilte Daten erfasst, geteilt, abgerufen und analysiert werden können. Diese Dissertation beschäftigt sich mit der Unterstützung von Archäologen und Bioarchäologen bei ihrer Arbeit, indem unterstützende Methoden bereitgestellt werden, die dem archäologischen Arbeitsablauf , der in fünf Schritte unterteilt ist, folgt.

Der erste Teil dieser Arbeit beschreibt das xBook Framework, welches entwickelt wurde, um archäologische Daten zu erfassen und zu speichern. Es ermöglicht die Erstellung zahlreicher Datenbankanwendungen, um notwendige Funktionen für den archäologischen Kontext bereitzustellen. Der zweite Teil beschäftigt sich mit der Zusammentragung von archäologischen Daten und setzt sich mit Methoden zum Teilen von Informationen, Methoden zur Zusammenarbeit zwischen Kollegen und Methoden zum Abruf von verteilten, aber zusammenhängenden archäologischen Daten auseinander. Der dritte Teil stellt ein dynamisches Framework für Datenanalysen vor, welches ein flexibles und leicht zu bedienendes Tool bereitstellt, das Archäologen und Bioarchäologen in der Ausführung von Analysen ihrer Daten unterstützt, so dass weder Programmierkenntnisse noch die Einarbeitung in externe Technologien benötigt werden. Der vierte Teil führt ein interaktives Tool ein, mit dem – unter Verwendung der zuvor beschriebenen Methoden zur Datenabfrage – die zeitliche Position von archäologischen Funden in Form einer Harris Matrix mit ihrer räumlichen Position als 2D- und 3D-Lageplan verglichen werden kann. Abschließend spezifiziert der fünfte Teil eine Architektur für ein Informationssystem, das die Durchsuchung von verteilten und interdisziplinären Daten durch dynamische Joins von Suchergebnissen aus heterogenen Datenformaten ermöglicht. Diese neue Art an

Informationsabfrage erlaubt Wissenschaftlern eine Querverbindung von archäologischen Informationen mit fachfremdem Wissen. Das Konzept für dieses Informationssystem ist jedoch nicht auf den archäologischen Kontext begrenzt. Auch andere wissenschaftliche Bereiche können von dieser Architektur profitieren.

Chapter 1

Introduction

Attribution

This Chapter uses material from the following publication:

- Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, and Henriette Obermaier. A Framework for Supporting the Workflow for Archaeo-related Sciences: Managing, Synchronizing and Analyzing Data. In *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs ,,Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Workshopband*, pages 89–98, 2017. [KLKO17]

See Chapter 1.2 for a detailed overview of incorporated publications.

In the flow of archaeological and bioarchaeological excavations, loose findings such as rests of buildings, ceramics, glass, metal, human burial remains, faunal, and botanic rests are excavated and packed separately in specified units together with a find sheet. The find sheet number on the find sheet specifies the unit uniquely. Representatives of different archaeo-related scientific disciplines – like archaeologists, zooarchaeologists, anthropologists, archaeobotanists, etc. – collect the data of these findings. This archaeologically relevant data concerning the excavation site is valuable for all disciplines. The data describes the findings as precisely as possible. The manner of the description is standardized in its core. For example, zooarchaeology gathers data like animal species, skeleton elements, age at death, bone measurements, fragmentation, taphonomy, etc.

Target of all archaeological and bioarchaeological work is the best groundwork for the holistic view and interpretation of features and findings for the context of scientific analyses after the destruction of ground monuments. [Bay16b] [Bay16c] The documentation

of excavations is a description of the found situations. Once the original objects and contexts are destructed – e.g. caused by constructing projects, natural disasters, vandalism, etc. – the documentation becomes the primary source for the data. [Ver01] However, the documentation is not only pure archive material. During an excavation it serves as an error checking possibility, is used for reporting, and is a basis for strategical decisions about further excavation-related proceedings. So it is important that the documentation is present directly on-site. After the completion of an excavation and the archiving, the data of the documentation serves as a preparation for future planned investigations and for scientific analyses as part of publications, research projects, and written theses. The documentation is always an everyday working material. [Sch18]

To give the research access to the information about remains is one of the most important tasks of collections, museums, and archaeo-related institutes. This requires a full coverage of the collection. However, a full coverage does not mean that this information is also accessible. The stock records are often only available in handwritten or printed form, e.g. books or file cards which – if at all – can only be supplied on-site to the scientific community. [GHM15] Even if the information is available in digital form, the data exchange via the Internet is limited because of insufficient network infrastructures and applications. Today, one of the biggest challenges of collections is to take action to store data digitally and to provide methods to exchange archaeological and bioarchaeological data to enable complex analyses.

The gathering of the data may be carried out at the excavation site itself or at the particular institute after the excavation is finished. Based on the geographical position of the site or due to the states' political circumstances, the processing on site may be inevitable. Gathering the data online is not always possible because of the lack of an Internet connection, an instable Internet connection, or because of the limited data volume available.

Besides macroscopic investigations, specific questions concerning determination of material, absolute dating, or questions on bioarchaeological migration respectively, further investigations of chemical (qualitative analysis, stable isotopes), physical (radiocarbon dating) or molecular biological (ancient DNA) nature are carried out. This "is a popular method for solving Archaeology-related problems like determining the origin of archaeological finds, as well as analyzing the diet of individuals or entire populations, the climate of a region, and the migration patterns of people and their habitat" [MNKG15]. Specialized laboratories may be charged with these purposes. [MAK12] [AK03]

Within each discipline, the data is recorded by one person or by a group. At the end, all the data is collected. Either handwritten records are entered in tables or databases, or they had been gathered digitally from scratch. The data may be completed by those from external investigations.

The data collection forms the bases for documentations and for analyses inside the

excavation project. Most of the analyses are statistical ones. The methods follow continually developing standards. Whereas standardizing builds a great challenge and finally has to be restricted to a core, many data analyses are carried out individually without respecting any standards. One reason may be that problems and evidence of the research can be of different nature, depending on the archaeological context and the quality and quantity of the findings.

In order to be able to put the results in a historical frame or to position them in a historical process, synchronous and diachronous comparisons are carried out to enable the comparison with the results of investigations from regional or supra-regional excavation sites. In this connection, data from other investigations is included in the analyses. It may derive from published investigations or from colleagues who exchange unpublished data. This occurs inside the particular subject of research as well as in interdisciplinary collaborations of cultural sciences and natural sciences. If a big amount of data as an approach from the science of information technology is available, data mining may provide additional results.

1.1 Overview

This Chapter gives an overview of the structure of this thesis.

Concerning the archaeological and bioarchaeological work explained in the introduction, three main challenges were extracted that have to be fulfilled to support the work of archaeologists and bioarchaeologists:

1. **Data gathering:**
   Gather and store archaeo-related data digitally in a database.

2. **Data sharing:**
   Collaborate with colleagues and share data with other users.

3. **Data analyses:**
   Execute analyses on the available data.

Complemented with the excavation process itself, this sequence results in a rough workflow for the archaeological and bioarchaeological work. This is visualized in Figure 1.1. All disciplines of archaeo-related sciences deal with similar challenges that result in similar workflows. However, there are still differences in the nature of the data. These circumstances result in questions how to store, share, retrieve, and analyze these heterogeneous and distributed data. The contribution of this thesis is to support archaeologists and bioarchaeologists in their work by providing methods following this archaeo-related workflow.

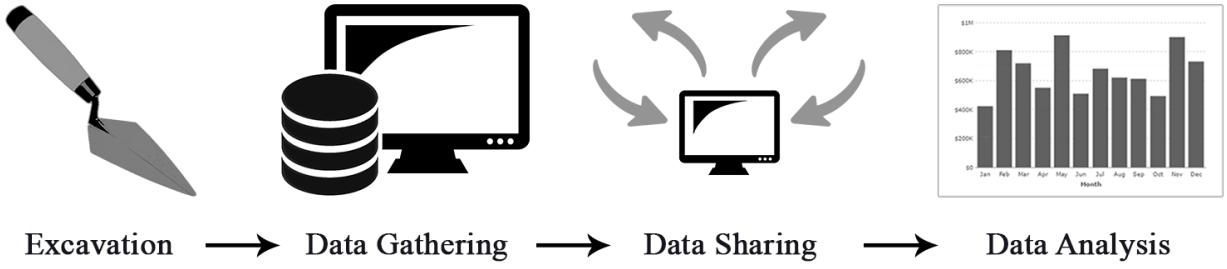


Figure 1.1: *A visualization of the workflow of archaeological and bioarchaeological work.*

Therefore, the xBOOK framework, that was developed to gather and store archaeological and bioarchaeological data, is described in Chapter 2. It allows creating several database applications to provide necessary features for the archaeo-related context. The Chapter shows the historic evolution of the zooarchaeological database OSSOBOOK which features were extracted and wrapped into the common framework xBOOK to provide the main basic functionalities for inherited databases. The framework forms the basis for the zooarchaeological databases OSSOBOOK and PALAEODEPOT, the archaeological databases ARCHAEOBOOK, EXCABOOK, and INBOOK as well as the anthropological databases ANTHRODEPOT and ANTHROBOOK.

Chapter 3 of this thesis deals with methods to share information, collaborate with colleagues, and retrieve distributed data of cohesive archaeological contexts to bring together archaeo-related data. Therefore, we explain a timestamp-based synchronization process in Chapter 3.1 that is built into the xBOOK framework and can be used for the collaboration with colleagues and the exchange of archaeological data with other scientists. In addition, we introduce REMIS (*Reverse-Mediated Information System*) in Chapter 3.2, a novel information system that allows retrieving interrelated archaeological information that is spread in several distributed, heterogeneous data sources. This is achieved by transferring the management of data sources to the client-side, allowing data owners to keep full control over the third-party data access while providing simple administration. Data owners can register their data sources directly. Queries are forwarded to be translated on the data source level. In contrast to a centralized mediator-based system, which is not suitable for dynamic retrieval and flexible adjustment of heterogeneous data, our system provides the adaptability to keep query responses accurate in case of evolving data sources.

The following Chapter 4 describes a framework for data analyses that can be embedded into a base application. Especially for less technically experienced scientists it is important to analyze the data directly inside the application to work in their familiar user environment where the data is entered. A solution is desirable to allow the scien-

tists easily to work with their data and to support and motivate the execution of further analyses of their data which would lead to new and interesting results and insights. If the analysis process is too complicated, tedious, or not apparent, this could restrict scientists, especially in non-IT-related disciplines like archaeology and bioarchaeology, from analyzing complex data circumstances. To enable this solution, we first describe the requirements for our ANALYSIS TOOL and explain the steps we took to meet those requirements. Then we describe the steps that are necessary to integrate the ANALYSIS TOOL into a base application. We explain how the ANALYSIS TOOL can be extended with new specific components that allow the users to add exactly the features they need for their analyses. We also discuss different approaches for the graphical data composition which are evaluated in a user study. The result of the evaluation contributes towards the development of a prototype of the framework. Finally, we apply the prototype to zooarchaeological data of OSSOBOOK in a case study.

In Chapter 5, we describe TARDIS, another analysis application. It is a visual analytics system for spatial and temporal data designed for the needs of archaeo-related disciplines that supports domain experts in analyzing their data. The temporal data is visualized in form of an interactive Harris Matrix that illustrates the temporal position of the layers. The 2D and 3D visualization sketches the spatial position of findings with heat map colors and a Kernel Density Estimation. The application allows the visual comparison of the temporal and spatial data. We discuss the archaeological background followed by the technical implementation of the application and the generation of the Harris Matrix. Finally, we demonstrate the usefulness of the application by analyzing real zooarchaeological data from an excavation where we use TARDIS to show the distribution of animal species in a case study.

Finally, we propose the REMIS CLOUD in Chapter 6, an information management system which allows data owners to provide and cross-connect domain-extrinsic knowledge and enhances data retrieval with a search interface that is intuitive and easy to operate. It is a further development of REMIS, but extends the architecture with a "Category" system which helps to classify the data sources. Thereby the data retrieval like in REMIS is still possible, however REMIS CLOUD additionally provides the retrieval of related, but logical separated and also of domain-extrinsic data. The coherence is determined by considering the common "Categories" of all data sources.

To conclude this thesis in Chapter 7, the presented methods and the future work are debated in Chapter 7.1 and a discussion about the development of archaeo-related technologies is conducted in Chapter 7.2.

1.2 Attribution

This Chapter gives an overview of the previously published papers that are included in this thesis (in order of appearance in this work) and clarifies the contributions of the author of this thesis.

Chapter 1 uses parts of the publication *A Framework for Supporting the Workflow for Archaeo-related Sciences: Managing, Synchronizing and Analyzing Data* [KLKO17] by Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, and Henriette Obermaier, which describes a rough overview about the archaeological workflow and contributions that consist of previous work by Daniel Kaltenthaler and Johannes-Y. Lohrer under supervision of Peer Kröger. Henriette Obermaier contributed domain-specific input.

Chapter 2 is based on the publication *The Historic Development of the Zooarchaeological Database OssoBook and the xBook framework* [KL18] by Daniel Kaltenthaler and Johannes-Y. Lohrer, which describes the development of the database OSSOBOOK and the framework XBOOK. The concept, implementation, and design of the XBOOK framework was solely realized by Daniel Kaltenthaler and Johannes-Y. Lohrer, including the technical and collaborative infrastructure, the software development, and the draft and implementation of the included features. The mentioned applications OSSOBOOK [KLK+18b] (since the new development from version 4.0), EXCABOOK [KLK+18f], ANTHRODEPOT [KLK+18c], and PALAEODEPOT [KLK+18g] are being developed exclusively by Daniel Kaltenthaler and Johannes-Y. Lohrer. ARCHAEOBOOK [KLK+18e] was originally developed by Daniel Kaltenthaler and Johannes-Y. Lohrer, and later Ciarán Harrington supported the implementation. ANTHROBOOK [KLK+18h] was developed by Tatiana Sizova and Anja Mösch on the basis of the XBOOK framework until 2016. Then Daniel Kaltenthaler and Johannes-Y. Lohrer took over the development in 2017. INBOOK [KLK+18d] is being developed by Ciarán Harrington. Primarily, the database applications consider the valuable domain expertise of Henriette Obermaier (OSSOBOOK and PALAEODEPOT), Christiaan van der Meijden (OSSOBOOK), Sonja Marzinzik, Erich Claßen, and Heiner Schwarzberg (each ARCHAEOBOOK), Michaela Harbeck and Andrea Grigat (each ANTHRODEPOT and ANTHROBOOK), Anita Toncala (ANTHROBOOK), and Silke Jantos, Agnes Rahm, Ina Sassen, Tilman Wanke, and Roland Wanninger (each EXCABOOK).

Chapter 3.1 is based on the publication *A Generic Framework for Synchronized Distributed Data Management in Archaeological Related Disciplines* [LKK+14] by Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier, which describes a timestamp-based synchronization method to enable sharing of data and collaboration with colleagues. The synchronization logic was primarily drafted and implemented by Johannes-Y. Lohrer in cooperation with Daniel Kaltenthaler. A follow-up publication of the same title [LKK+16b] by the same authors additionally explains the conflict management in more detail, which was basically drafted and im-

plemented by Daniel Kaltenthaler in cooperation with Johannes-Y. Lohrer. The publication *Synchronized Data Management and Its Integration into a Graphical User Interface for Archaeological Related Disciplines* [KLK+15] by Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier extends the described synchronization methods with the integration into the graphical user interface, which was largely designed and implemented by Daniel Kaltenthaler in cooperation with Johannes-Y. Lohrer. The work was developed under supervision of Peer Kröger and Christiaan van der Meijden. Henriette Obermaier contributed domain-specific input for the three mentioned papers.

Chapter 3.2 is based on the publications *Reverse Mediated Information System: Web-based Retrieval of Distributed, Anonymous Information* [LKK+17] by Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, and Christiaan van der Meijden, and *Retrieval of Heterogeneous Data from Dynamic and Anonymous Sources* [LKR+18] by Johannes-Y. Lohrer, Daniel Kaltenthaler, Florian Richter, Tatiana Sizova, Peer Kröger, and Christiaan van der Meijden, which describe an architecture to retrieve data from heterogeneous, distributed, and anonymous data sources. The architecture was primarily drafted by Johannes-Y. Lohrer and implemented by Daniel Kaltenthaler, Johannes-Y. Lohrer, and Tatiana Sizova, under supervision of Peer Kröger. Florian Richter attributed valuable input about the usage of the system in other domains. Christiaan van der Meijden contributed with valuable discussions. Henriette Obermaier contributed domain-specific input.

The introduction of Chapters 4 and the Chapters 4.1, 4.3, and 4.4 use parts of the publication *Leveraging Data Analysis for Domain Experts: An Embeddable Framework for Basic Data Science Tasks* [LKK16a] by Johannes-Y. Lohrer, Daniel Kaltenthaler, and Peer Kröger (used in Chapter 4), which describes an analysis tool that can be embedded into other applications to enable data analyses without the need of programming skills. The described logic of the tool was basically drafted and implemented by Johannes-Y. Lohrer under supervision of Peer Kröger. Daniel Kaltenthaler attributed valuable input towards the tool. The Chapters 4.2, 4.3, and 4.4 use parts of the publication *Supporting Domain Experts Understanding Their Data: A Visual Framework for Assembling High-Level Analysis Processes* [KLK17] by Daniel Kaltenthaler, Johannes-Y. Lohrer, and Peer Kröger, which discusses several approaches how to realize the graphical presentation of the analysis composition, the evaluation, and the realization within the ANALYSIS TOOL. The discussed approaches of the graphical representation and the implementation was primarily realized by Daniel Kaltenthaler under supervision of Peer Kröger. Johannes-Y. Lohrer attributed valuable feedback towards the tool.

Chapter 5 is based on the publication *TaRDIS, a Visual Analytics System for Spatial and Temporal Data in Archaeo-related Disciplines* [KLP+17] by Daniel Kaltenthaler, Johannes-Y. Lohrer, Ptolemaios Paxinos, Daniel Hämmerle, Henriette Obermaier, and Peer Kröger, which introduces a visual analysis tool to put spatial information in relation with tem-

poral data from archaeological sites. The concept of the tool was drafted by Daniel Kaltenthaler and Johannes-Y. Lohrer under supervision of Peer Kröger. The implementation of the presented prototype was developed by Daniel Kaltenthaler, Johannes-Y. Lohrer, and Daniel Hämmerle. Henriette Obermaier contributed domain-specific input. The executed analysis was interpreted by the domain scientist Ptolemaios Paxinos.

Chapter 6 is based on the publication *A Distributed Information Management System for Interdisciplinary Knowledge Linkage* [KLRK17] and the extended publication *Interdisciplinary Knowledge Cohesion through Distributed Information Management Systems* [KLRK18] by Daniel Kaltenthaler, Johannes-Y. Lohrer, Florian Richter, and Peer Kröger describe an information system that enables the data retrieval from interdisciplinary domains. The architecture was primarily drafted by Daniel Kaltenthaler in cooperation with Johannes-Y. Lohrer under supervision of Peer Kröger. The implementation was realized by Daniel Kaltenthaler and Johannes-Y. Lohrer. Florian Richter contributed the use case for the eLearning context and the discussion about ethics.

Chapter 2

Development of OssoBook and the xBook Framework

Attribution

This Chapter uses material from the following publication:

- Daniel Kaltenthaler and Johannes-Y. Lohrer. The Historic Development of the Zooarchaeological Database OssoBook and the xBook Framework for Scientific Databases. *ArXiv e-prints: 1801.08052*, January 2018. [KL18]

See Chapter 1.2 for a detailed overview of incorporated publications.

In general, software and its possibilities are developing to an ever more advanced level. The implementations are changing over time and new technologies have to be considered and integrated. Different ideas and concepts of developers, and different expectations of customers have to be taken into account when developing the application. Even though these approaches may differ from expected realizations, especially in the range of data gathering, the requirements are quite similar in several scientific areas.

It is often the case that different disciplines develop their own software solutions to gather and manage own data specifically for their own need. However, this causes the decisive disadvantage that new features which can be applied to several disciplines have to be implemented multiple times for each of the individual solutions. This is not only a huge temporal, but also a financial expenditure.

This situation became especially apparent during the development of OssoBook [KLK+18b], a database for zooarchaeological findings. Other disciplines in the archaeological context (like anthropology, archaeobotanic, and archaeology) also gather data with similar methods. Of course, these differ content-related, but the requirements on

the basic features strongly overlaps with the functionalities of OSSOBOOK. It is not limited to the archaeo-related context, other scientific disciplines could also benefit from similar solutions.

In other archaeo-related disciplines there is also a necessity of gathering data which consists of similar workflows like in the zooarchaeological field. In consequence, the idea of the XBOOK framework developed out of this context. In the XBOOK framework features are developed centrally and are provided to all applications that are instances of the framework. New features do not have to be implemented individually, however, custom extensions are still possible. In case of errors and bugs, it is not necessary to fix them in each application, they can be fixed centrally. Thereby, it causes the creation of similar structures in the gathering and analysis of data.

As of now, the XBOOK framework enabled the development and usage of a number of archaeo-related applications, like OSSOBOOK, ARCHAEOBOOK [KLK+18e], ANTHROBOOK [KLK+18h], EXCABOOK [KLK+18f], PALAEODEPOT [KLK+18g], ANTHRODEPOT [KLK+18c], and INBOOK [KLK+18d] (cf. Chapter 2.4). However, the XBOOK framework is not limited to the requirements of the archaeo-related context, it can be applied to many other scientific application as well.

In this Chapter we describe the development process of the OSSOBOOK application and the XBOOK framework in the recent decades. We first describe the origins of OSSOBOOK in Chapter 2.1 and explain the further development of the application and the extraction of the XBOOK framework in Chapter 2.2. Finally, we show the basic, most important features of XBOOK in Chapter 2.3 and give a short description of the available "BOOKs" (applications that instance from the XBOOK framework) in Chapter 2.4.

2.1 Origin of OssoBook

Below, we describe the original development of the OSSOBOOK application, that is being developed since 1990.

2.1.1 First OssoBook version in dBASE

The first version of OSSOBOOK was originally released in 1990 by Jörg Schibler and Dieter Kubli of the University of Basel, Switzerland. The technical basis was dBASE[1], a file based database application for computer systems running the operating system DOS. The exclusively in German language published OSSOBOOK database enabled the recording of zooarchaeological data. Five input fields for archaeological information, eleven input

[1] dBASE's underlying file format, the .DBF file, is widely used in applications needing a simple format to store structured data.

Figure 2.1: *The input mask in* OSSOBOOK *1.0. [Kal12] [Loh12]*

fields for zooarchaeological data, and for each skeleton element eight further input fields for the recording of measurements were provided (cf. Figure 2.1).

Besides the data input, the early version of the application already provided the possibility to implement simple data analyses. Three analyses were available for skeleton element representations: One for the age of long bones (cf. Figure 2.2), one set of analyses for bone parts, and one last analysis for measurements of bones. The results of these analyses could be saved to extern files which could be viewed and edited with spreadsheet like Microsoft Excel or Open Office Calc (today: LibreOffice Calc).

Besides the database itself, this version also provided an editor called OSSOINST which allowed defining custom numerical codes for different data inputs, e.g. the mapping of a species ID to a species name. This offered the advantage that each database could be used with individual data, and each user could easily extend the list of available species.

The archaeological data and the corresponding mappings were saved as ASCII files on the local hard disc drive of a computer, or on floppy disks. This made the sharing of data complicated and difficult, and therefore was rarely practiced. Nevertheless, OSSOBOOK already allowed the translation of specific texts of input values to other languages by using OSSOINST. First partial translations (especially to English and French) were already possible and done. [Sch98]

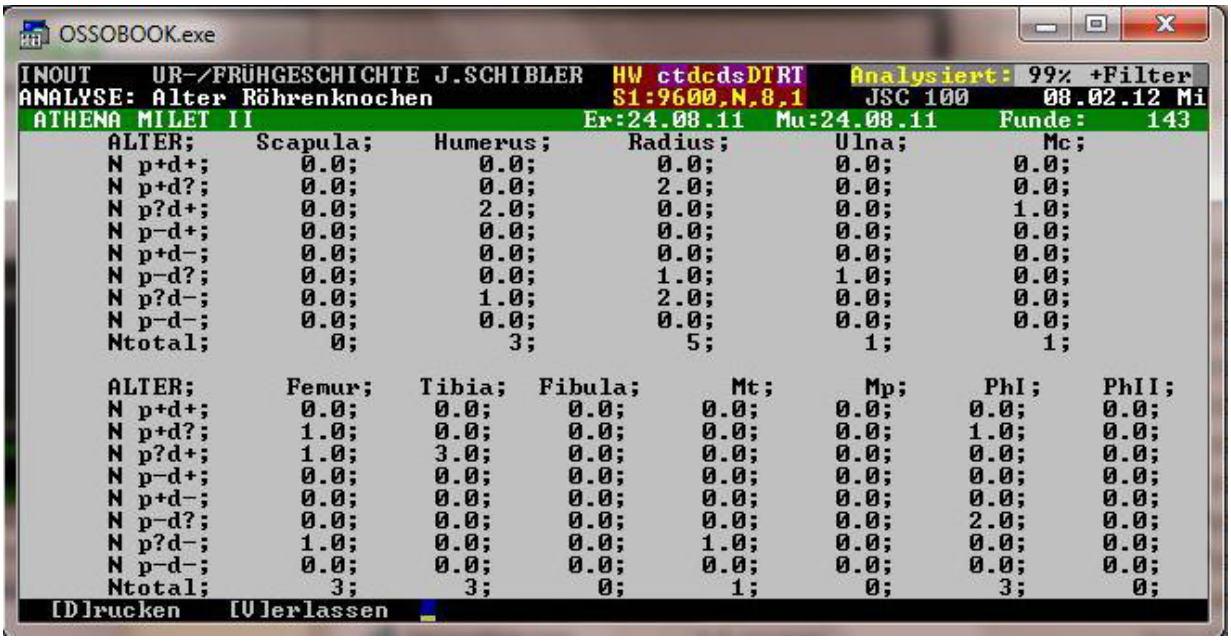


Figure 2.2: *The analysis of the age of long bones in* OssoBook *1.0. [Kal12] [Loh12]*

2.1.2 Conversion to Java

At the latest with the release of Microsoft Windows XP in 2001, the operation system DOS fades into the background. The technique of the dBASE database became increasingly obsolete. Even though the application is still running on modern operation systems (e.g. in the command-line interface on Windows computers, or the system console in other operation systems), the usability and optical presentation of OssoBook was no longer up-to-date. The disadvantage that data could only be saved on the local computers, also contributes that a new, enhanced version of OssoBook should be developed.

The new version of OssoBook was initialized by Christiaan H. van der Meijden of the *Veterinary Faculty*[2], together with the *Institut für Informatik, Lehrstuhl für Datenbanksysteme and Data Mining*[3] at the Ludwig-Maximilians-Universität München, Germany. The application was converted to the object-oriented and platform-independent programming language Java. On the servers of the university, there was installed a single, global MySQL database which should be used by the employees at the *Institute of Palaeoanatomy, Domestication Research and History of Veterinary Medicine*[4]. They used the client to connect to the server and directly work with the data of OssoBook on the global database. [Lam08]

In addition, the mapping of IDs to values for specific fields was adopted, but the

[2] http://www.vetmed.uni-muenchen.de
[3] http://www.dbs.ifi.lmu.de
[4] http://www.palaeo.vetmed.uni-muenchen.de

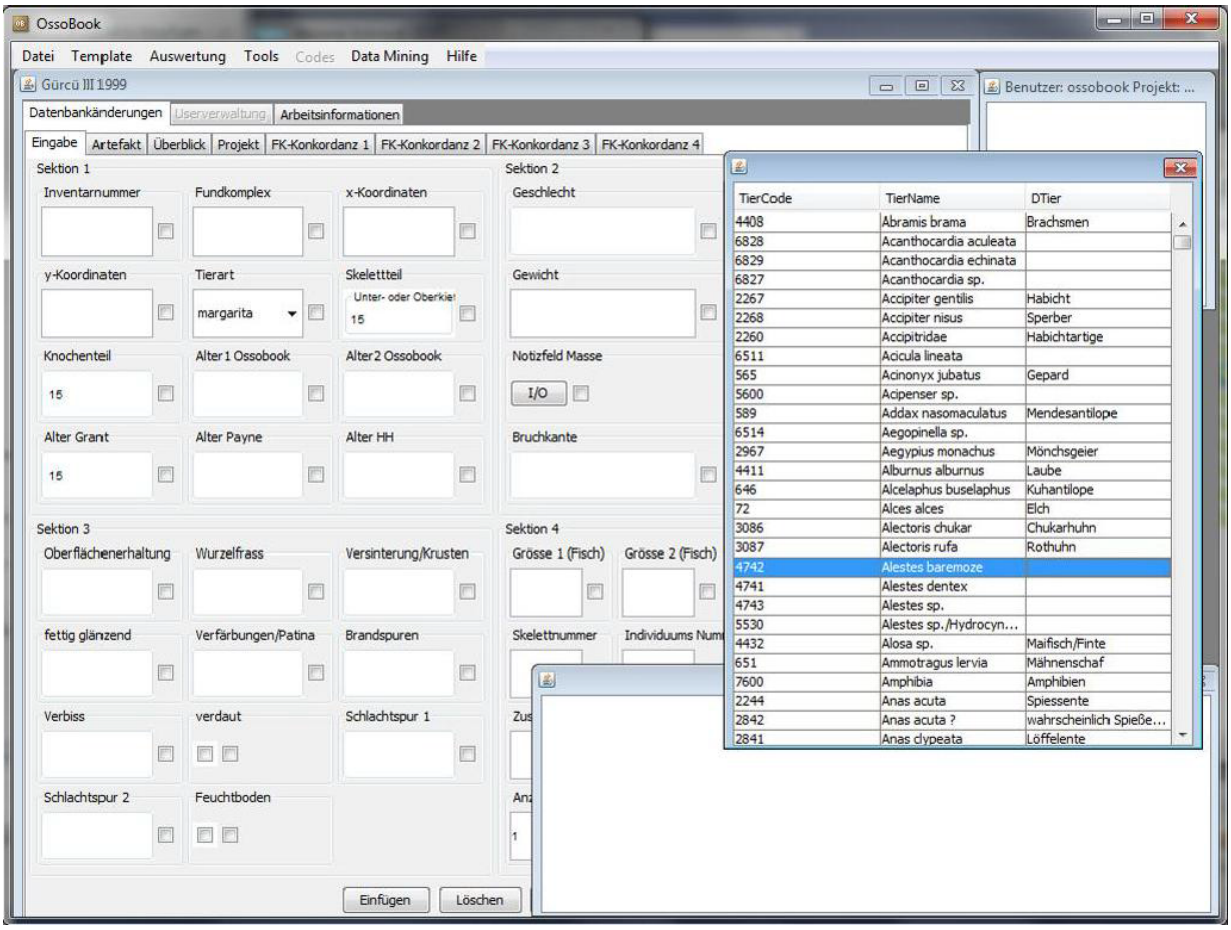


Figure 2.3: *The input mask in* OssoBook *3.4. [Kal12] [Loh12]*

functionality to add or change them manually was removed. The users worked with standardized, predefined values for the necessary input fields. This should improve the comparability of the entries that were saved in the database. Today, these mappings are called “Code Tables” in the application.

In combination with the port of the application to Java, OssoBook got a graphical user interface for the first time. As shown in Figure 2.3, the input fields were arranged in four sections and offered first input assistances, e.g. by using selection boxes for predefined values. Statistical information about the data sets and simple analyses were displayed in several tabs, which also provided more space for further input possibilities.

Version 3.4 was the first Java version of OssoBook and was released in 2007.

2.1.3 First Synchronization Implementation

Until this date, the users of OssoBook could only connect and work directly on the global database on the servers. Originally this was only possible with connections within

the network of the university, later a tunnel enabled the connection from other places.

In 2008, the first implementation of a synchronization in OSSOBOOK allowed the entry of data in a local database of the clients. The users could enter their data remotely without any connection to the network and later synchronize it to the global database.

The full development of the synchronization is described in Chapter 2.3.1. The implementation of the synchronization process is explained in detail in Chapter 3.1.

2.1.4 Application Redevelopment

At this time, we were tasked with the further development of OSSOBOOK and became the new development team for the upcoming years. However, this Java version of OSSOBOOK caused big problems for the development and usage. One could see that the application was only a port and did not use the potential of the object-orientated programming language Java. Most of the input fields were not very intuitive because of working with numerical codes in general, which meaning had to be looked-up in external spreadsheet or PDF files. In addition, the application included a lot of errors and bugs which were not displayed in the graphical user interface of the application. In some cases these problems made the data input impossible and let the application crash.

From the point of view of the development team it was nearly impossible to implement the requests of the zooarchaeologists and to add new features. The first tries to integrate new elements into the application made already clear that it is more reasonable to reimplement the application from scratch instead of trying to continue developing for the current version. The main problem of a further development of the current version was the very static program code elements that did not allow adding new input elements to the input mask. This static design also made an object-oriented design using inheritance impossible. Also the programming code was only sparsely commented and Javadoc comments were missing for the most parts. In addition, the names of the methods were not intuitive, so new developers would need a long familiarization to be able to develop the application.

It was decided that the database scheme of OSSOBOOK and the OSSOBOOK client should be newly developed considering the Model–View–Controller architecture [Kal11] [Loh11]. We put a lot of emphasis for future features being able to be fast and dynamically integrated to the application to enable a simple and resource-efficient further development. To avoid data loss and to be able to continue using the data of the previous OSSOBOOK version, we wrote a script that converts the old database scheme into the new one.

In autumn 2011, version 4.1 of OSSOBOOK was released that included the same features than the previous version of the database application at first, but was more flexible in the usage for the users and developers. Figure 2.4 shows a screenshot of this version.

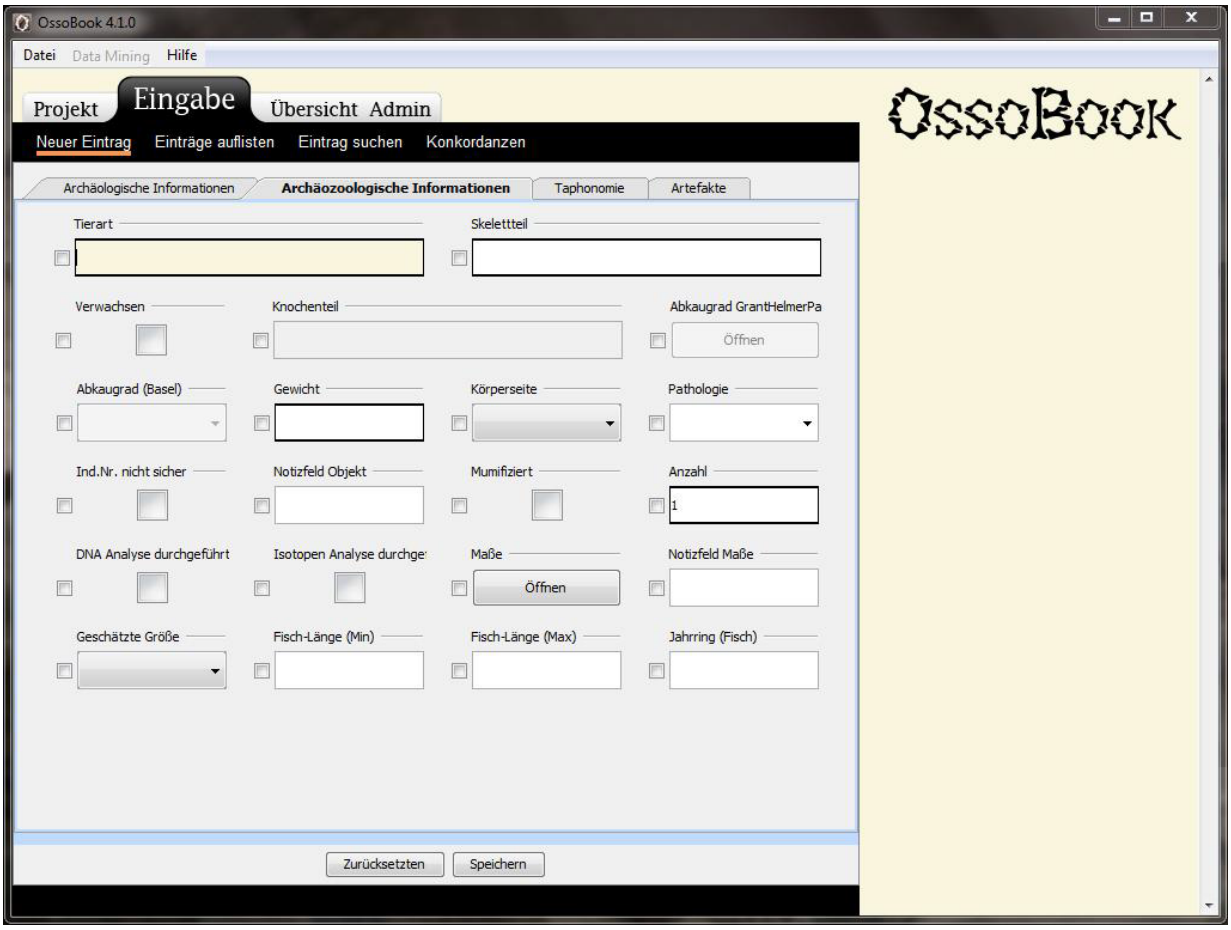


Figure 2.4: *The input mask in* OSSOBOOK *4.1. [Kal12] [Loh12]*

2.2 From the OssoBook Application to the xBook Framework

From this point of time, OSSOBOOK was ready for scholarly usage. At the same time, it was assured that further development and future adjustments to the database are possible.

Since the gathering of data is an essential task of the work in archaeo-related disciplines, other disciplines got also interested in the architecture of OSSOBOOK. While the workflow process is similar in all disciplines (archaeology, anthropology, zooarchaeology, palaeobotany, palaeontology, etc.), the collected data is different in each special field. An individual database solution based on the OSSOBOOK architecture would greatly support the scientists in their work.

So we set the challenge to provide a generic solution for supporting the scientists in all disciplines, this means that OSSOBOOK has to be as customizable as possible to

allow all required information about the specific data to be gathered. Therefore, we used the basic architecture and features of OSSOBOOK and extracted XBOOK, a generic framework including the common and basic used features for a database for archaeo-related disciplines. [KLKO18]

Below, we describe the most important functionalities that are features of the XBOOK framework.

2.2.1 Input Fields and Input Mask

The input fields were strongly enhanced and extended. Previously, there had been only four basic types of fields available: Text, numeric values, check boxes, and Code Tables. Several new types of fields were integrated which can be reused for new input fields, e.g. combo boxes for values and IDs, multi selection data, buttons to open panels for more complex data inputs, date and time choosers, etc. Furthermore, several individual input elements were added that have specifically been implemented for single data elements of OSSOBOOK. Especially the input fields for species, skeleton elements, measurements, wear stages, bone elements, and the gene bank number benefited from the individual input possibilities.

Also the visual presentation of the input mask was updated, as shown in Figure 2.5. Besides the arrangement of single elements inside an input element, they were wrapped with a visible box that was able to be colorized dependent on different states. A mandatory field that has to be filled before saving the entry is highlighted with a yellow background color. If an input is not valid in a field, this is indicated with a red background color. Further enhancements in the graphical user interface were also added, e.g. a box for temporarily displaying text like warnings and errors as a feedback for the user.

2.2.2 Update Procedure

To enable a dynamic development of the application and the version-independent use of the synchronization, it was necessary to keep the database and the program version up-to-date. Only if the local and global database match the same database scheme, the synchronization is able to exchange data correctly. So an update procedure was integrated that consisted of three steps:

1. **Program version update:**
   When the user logs in, the program version of the local client is checked if it is up-to-date. If not, the OSSOBOOK UPDATER, a small helper application, was automatically started which let the user update the program files by executing the update process.

Figure 2.5: *The input mask in OssoBook 4.1.14. [Kal12] [Loh12]*

2. **Database version update:**
   Then the update process updated the database scheme and the data itself to the current version, if necessary. In the program code it was defined which database version is required for the current version. If the current, local database version did not match with the database version in the program code, the necessary SQL queries are loaded from the server and executed. This was done recursively until the database versions matched.

3. **Code Table update:**
   The Code Tables were updated. To guarantee a consistent data structure, it was also important that the mapping of the values to the corresponding IDs is the same on each client and on the server. So the synchronization was extended with a method that updated the Code Tables in the local clients.

This procedure has not changed until today, although the process is not executed by the OSSOBOOK UPDATER anymore, but by the XBOOK LAUNCHER (cf. Chapter 2.2.7).

2.2.3 Plug-in Interface

At that time, OSSOBOOK provided an interface for plug-ins which was used for the integration of different sets of analyses that were developed by students of the Ludwig-Maximilians-Universität München, Germany:

- a plug-in for the analysis of age distribution [Kal12],
- a plug-in for the cluster analysis of measurements [Loh12],
- a plug-in for similarity search on multi-instance objects [Dan10],
- a plug-in for the execution of sample data mining methods [Tsu10], and
- a plug-in offering some analysis methods for zooarchaeological data [Neu12].

These plug-ins were able to be run directly in the application and to be used with the data from the database. Each plug-in could be imported to the application by simple copying the corresponding JAR file into the plug-in folder of OSSOBOOK.

2.2.4 Database Identification

For the synchronization and the possibility to work offline, it was essential to differentiate data sets that were entered on different computers. The problem is that one single ID for the data sets is not sufficient because the same ID could be assigned on different computers several times which would cause errors in the synchronization process. This was solved by the addition of a new column called “Database ID” for each entry.

Storing and handling of the Database ID required several iterations to prohibit errors and problems with different aspects of user interaction:

The first iteration – that was already implemented before the reimplementation of OSSOBOOK– considered a file called OBINIT which was a short SQL script generated during the registration process that was sent to the newly registered user. The file included the user name and password, and additionally assigned a unique Database ID to the local database. The main issue of this solution was that the OBINIT file was necessary to initialize the database, but if the users installed the application on two different computers and used the same OBINIT file, the identical Database ID was used for both computers. Furthermore, the users had to save the email and the OBINIT file because it was not possible to recover the data once the information is lost. This solution also bound the password to the OBINIT file, which was also a reason why the password was not editable.

The second iteration approached these main issues. The assignment of the Database ID was realized by the server while initializing the database. So every time a new database was installed and initialized on a different computer, the server was queried for a new Database ID which was used for the local database. In theory, this approach solved the problems with different computers, however some users created local back-ups of the application folder which also includes the local database. So it occurred again that the combination of identical entry IDs and Database IDs were used when the users restored the application from the backup.

The final iteration closes this gap by defining that the application could not be installed on any folder anymore and additionally saving the database ID in the registry. Now the xBook Launcher (cf. Chapter 2.2.7) installs the application data and the database in a folder which grants the logged-in user reading/writing access, e.g. in Windows we use the AppData folder. When the application is started, the Database ID inside the database is checked against the ID saved in the registry. If they do not match, or is not yet available, a new Database ID is issued which is then saved again in both, the database and the registry.

2.2.5 Registration

Originally, the users could register for an OssoBook account at a password protected homepage only. The users had to enter an email address and got an email including the user name, password, and the necessary OBinit file (cf. Chapter 2.2.4). This information was sufficient to work with the application, however, common mechanics like editing the user name or email address, or change/recover the password were not supported.

Later we changed this system to a more modern and convenient approach. The registration was moved directly into the application. At the login screen we added a button to register, where the users can enter some basic information: User name, first name, last name, email address, and a password. Now, there is no user restriction anymore, everyone can register and use the application. Once registered, the users can login to the application without the need of a OBinit file – due to the reasons described in Chapter 2.2.4. Furthermore, OssoBook was extended with profile settings where the users can manage their provided data. Additionally, the application was extended with a feature allowing the users to change and to recover their passwords.

2.2.6 Server–Client Architecture

Having a reliable Server–Client infrastructure is an important requirement to be able to synchronize and back-up data. This also helps to ensure no unauthorized changes are done, e.g. by a hacked client. In our case the Server–Client architecture has to handle

different scenarios:

The first one is the registration and login process. For this it is necessary to connect with the server from anywhere. After the user logged in, the server has to check if the client is up-to-date or first has to be updated. For this the database scheme has to be sent to the client along with values of the Code Tables.

After the version check is completed, the main task of the server is to handle the synchronization requests from the client. These use cases require the communication to handle a variety of dynamic and versatile data. Additionally – since client and server are implemented in different programming languages – built-in serialization tools like the Java Serialization [GHK+15] cannot be used. In an environment where multiple users can create, edit, and share their data, it is important to have a managed architecture that can be accessed and used from everywhere without any restrictions.

Challenges

A Server–Client architecture faces many challenges. Many of these are common in every Server–Client application, but some are very specific to the needs of the xBook framework.

- **Security:**
  Prevent unauthorized access. Users have only to be able to access the data they have the rights to. Unauthorized access has to be prevented.

- **Availability:**
  The server has to be available from everywhere. Using a Socket-based architecture [GNU] generally requires the usage of ports which have to be manually opened by an administrator in a firewall-protected secure environment. However, the opening of a port is not possible in every working environment because of strict regulations which forbids users to communicate with servers on other ports than 80 (HTTP) or 443 (HTTPS), for example in offices of state authorities or some institutes. Therefore, we had to find a solution how to make a connection from the client to the server possible in spite of restrictive firewall policies and how to use the available ports for the xBook server to accept requests.

- **Scalability:**
  The server has to work for single and also multiple users at the same time. This is true for most Server–Client architectures, still multiple users working on the same server have to be able to work simultaneously and not having to wait for one request to be completed until the next one is carried out.

- **Flexibility:**
  The server should run independent of the BOOK (an application that inherits from the XBOOK framework) in without any knowledge of data scheme inside the server. Therefore the specifics of each individual BOOK have not to be hard coded inside the server application, but dynamically loaded from a configuration file or the database.

Evolution

The first synchronization of the data in OSSOBOOK was handled by directly connecting to the database on the server from the client application. This connection required a manually entered passphrase which was given out with the registration, but was identical for all users. Additionally – apart from the client itself – no further checks for authorization were made. To address these issues, a C++ server application was created with the development of the XBOOK framework which now was the communication partner of the client application. The server is connected to the database and analyzes incoming requests if the user has the authorization. If this is the case, the server carries out the command and sends a confirmation back to the client together with data which was retrieved by this request. Both, a Thread Pool [GS02] [Goe02] and a Connection Pool [Gol14] were used to allow multiple users to work simultaneously. The communication between server and client was handled via sockets with a custom serialization of all objects that were transmitted. While this architecture provided a fast, secure, scalable, and flexible way to communicate with the server, it became clear that – due to the nature of sockets – it could not be guaranteed that the connection can be established behind proxy servers that only allow certain ports. Therefore, the communication had to be moved to a different type of protocol.

To solve the problem with proxies restricting certain ports, the communication had to be done over ports which are not restricted by most proxies. These are usually port 80 (HTTP) and port 433 (HTTPS). Of course, the possibility remains that certain IP addresses are blocked. However, it would really get into hacking to get around this. We did not explore this possibility further. The server which is running the server application is also running an Apache server. This is used to distribute the XBOOK LAUNCHER and to download the files required to start the individual BOOKs. Besides the server hosts the XBOOK Wiki[5], a MediaWiki that provides helpful information. So it was not possible to change the port to 80 or 433 the old C++ application was listening on.

A new server application was required that does not conflict with the Apache server, but can run alongside it. Many different web applications would allow this, but PHP was chosen as a scripting language, since it does not require additional server configuration.

[5] `http://xbook.vetmed.uni-muenchen.de/wiki/`

Communication

In a traditional web service, the users would enter a URL in their browsers. The web server would then analyze the request and return a website with the requested information. Since in our case the client has to communicate directly with the server, the response has not to be human readable, but interpretable by the client. Because the client was already able to communicate with the old C++ server, the serialization on the client side was already available and working. Still, it had to be modified that it would be able to communicate with a PHP server. However, PHP is not designed to work with serialized objects, but to load a script with some parameters, and then return and display results.

There are several possibilities to realize a cross-platform serialization. One is JSON [JSO17] [Bra17], a lightweight, text-based, language-independent data interchange format. Since JSON uses a human readable format, it has the disadvantage of data overhead [McA13]. There are ways to optimize the transmission size, e.g. XFJSON [XFJ] transforms the JSON format into a binary-hex form that is additionally encoded and decoded. Considering that there is no necessity to read the transmitted data and that we expect a huge amount of data sets for single projects, we focus on bandwidth-efficient solutions. So we need an alternative that is not human readable. However, a solution like FlatBuffers [Gooa] was not published at the time of the implementation of our serialization method. Protocol Buffers [Goob] do not support PHP at all. BSON [BSO] in direct comparison with Protocol Buffers can give an advantage in flexibility, but also a slight disadvantage in space efficiency due to an overhead for field names within the serialized data. However, BSON is mainly used as a data storage and network transfer format in MongoDB[6] databases. Therefore, we would have had to distribute all libraries for MongoDB which seemed unreasonable, since there is no stand-alone implementation. Because no good alternative for our serialization was available at the time of implementation, we had to implement an own solution.

Since the communication with a PHP server is asymmetrical, requests and results do not communicate the same way. Therefore, it was necessary to split the serialization in two steps:

1. The first part consists of the data transmitted to the server. For this, the request is serialized to a string which is then appended to the requested URL with the HTTP POST method. This allows the PHP server to read the data and deserialize the string back to the request which is then carried out.

2. After the server completed the request, the result is serialized again and the resulting string is displayed as the content. This is read by the Java client and is returned

[6] `http://www.mongodb.com`

deserialized.

The communication is done with a message object. The message object holds the type of the request, e.g. synchronization, login, register, and additionally a list of further data. All classes that can be added as data are instance of the interface `Serializable` which has methods to serialize and deserialize itself. For each request, the type and amount of parameters of the data that is sent is predetermined. Of course, the data itself is not known beforehand.

The serialization requires special classes that implement the `Serializable` interface even for basic data types like `String` or `Integer`. Currently 16 different classes are used. These are mostly required for the synchronization and the initialization of the database scheme.

To secure the communication HTTPS is used. The server should hold no information about the specific BOOK apart from necessary information of the corresponding database so that the independence of the database can be ensured. Information about the tables that have to be included in the synchronization, are saved in tables inside the database. This allows those tables to be dynamically adjusted without the need to update the server. If a request is carried out, those tables are checked whether and which columns can be accessed.

2.2.7 Launcher

A very important part of the XBOOK databases is the XBOOK LAUNCHER. Over the years, the application became an increasingly important part of the architecture. It was developed from a simple updater application, that was executed to check if a new OSSOBOOK update was available, to the central place where all BOOKs can be installed, updated, and started independently.

OssoBook Updater

The first idea of the OSSOBOOK UPDATER was born through the necessity to allow the user to update the program. This was required because the latest version of OSSOBOOK was ensured to be able to work in online mode and to communicate to the server, because the local database scheme has to be identical with the global scheme on the server when synchronizing data. So the update process is a frequent process that has to be executed with each minor update of the application. A screenshot of the OSSOBOOK UPDATER can be viewed in Figure 2.6.

We wanted to avoid that the users had to run and install the update process manually. They should not have to go to the website, download the latest version, and install the files in the appropriate directory. The OSSOBOOK UPDATER was automatically called

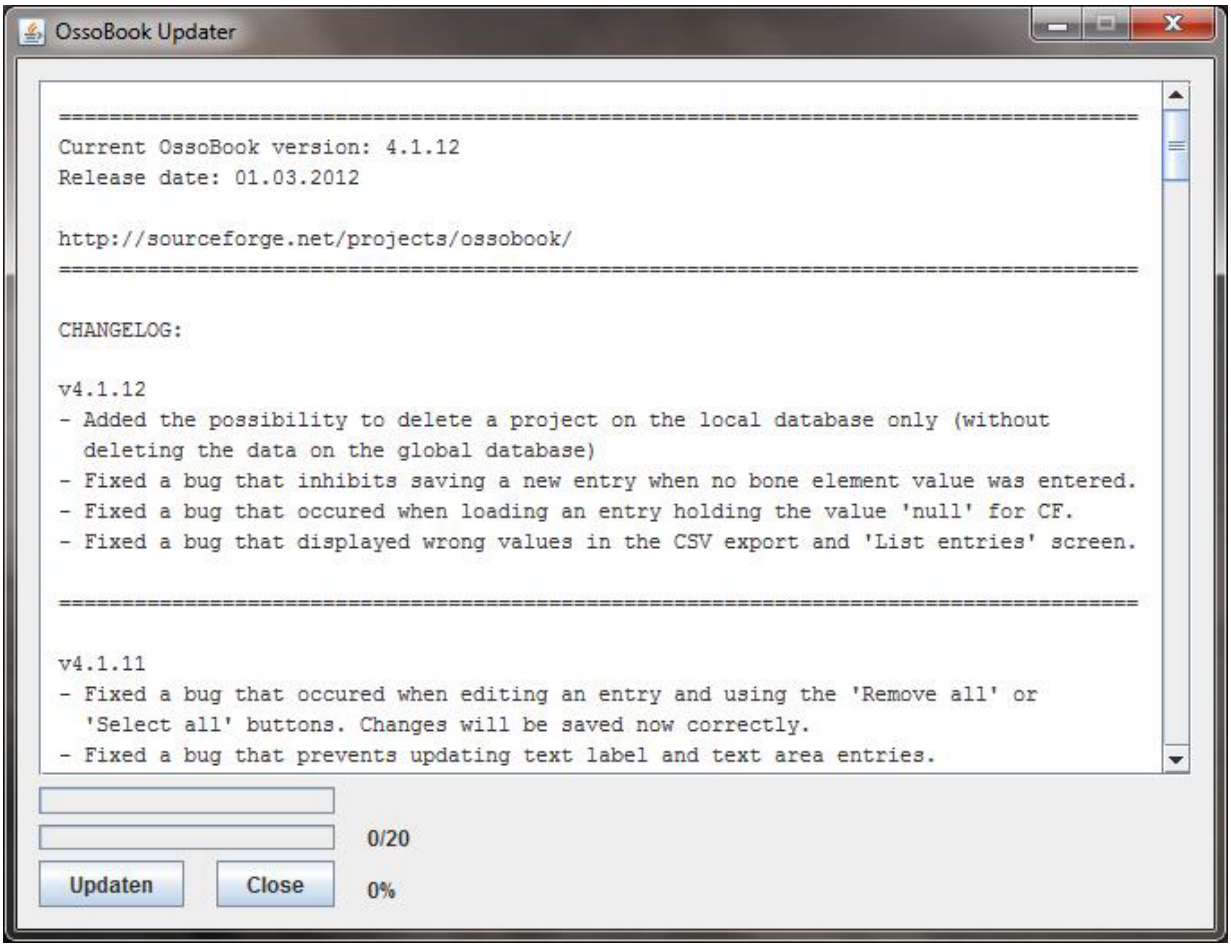


Figure 2.6: *The* OSSOBOOK UPDATER *allowed the user to update the* OSSOBOOK *application. [Kal12] [Loh12]*

when it was detected that the local program version was out-of-date when the user tried to connect to the global database. The OSSOBOOK UPDATER had a list of files that had to be checked for updates and compared them to the files available at a specific website. Since OSSOBOOK could be installed in an arbitrary directory, the OSSOBOOK UPDATER had to use this directory to update the files. For this, the updater was also stored in this directory and was updated by OSSOBOOK before the updater was executed.

To prevent different instances being run on one single computer, we had to specify the directory in which OSSOBOOK was located. This guarantees – together with the "Database ID" (see Chapter 2.2.4) – that the instance on a single computer was both, unique and could be identified uniquely. To avoid permission problems, a directory had to be used where writing permissions are guaranteed for the users. For Windows environments we chose the "AppData" directory. This allowed users to easily install OSSOBOOK on one computer, synchronize their data, and continue to work on a different computer.

Figure 2.7: *The first* XBOOK LAUNCHER *1.0 with the selection of four different* BOOKS*.*

Instead of updating the files in the directory of the updater, the updater became a independent file that from now on was called XBOOK LAUNCHER, since it also served as the entry point for the application. So instead of directly starting OSSOBOOK, now the users had to run the XBOOK LAUNCHER which then checked if all files were up-to-date and then allowed the execution of OSSOBOOK.

Development of the XBOOK LAUNCHER

With the development of more and more BOOKS for different areas of work, the requirements for the XBOOK LAUNCHER changed and – to avoid the need of several individual launcher applications for several databases – had to be adjusted to support more than one database. Therefore, the XBOOK LAUNCHER was extended with a BOOK selection. A screenshot of the version 1.0 of the XBOOK LAUNCHER is shown in Figure 2.7.

Each supported BOOK was represented as an own row in the selection, displayed by an application icon, the application name, a short description of the database, and the supported languages. The user could select the desired database and execute it. Furthermore, the XBOOK LAUNCHER was extended with general settings that affects all BOOKS (like the language selection and the selection of the automatic or manual synchro-

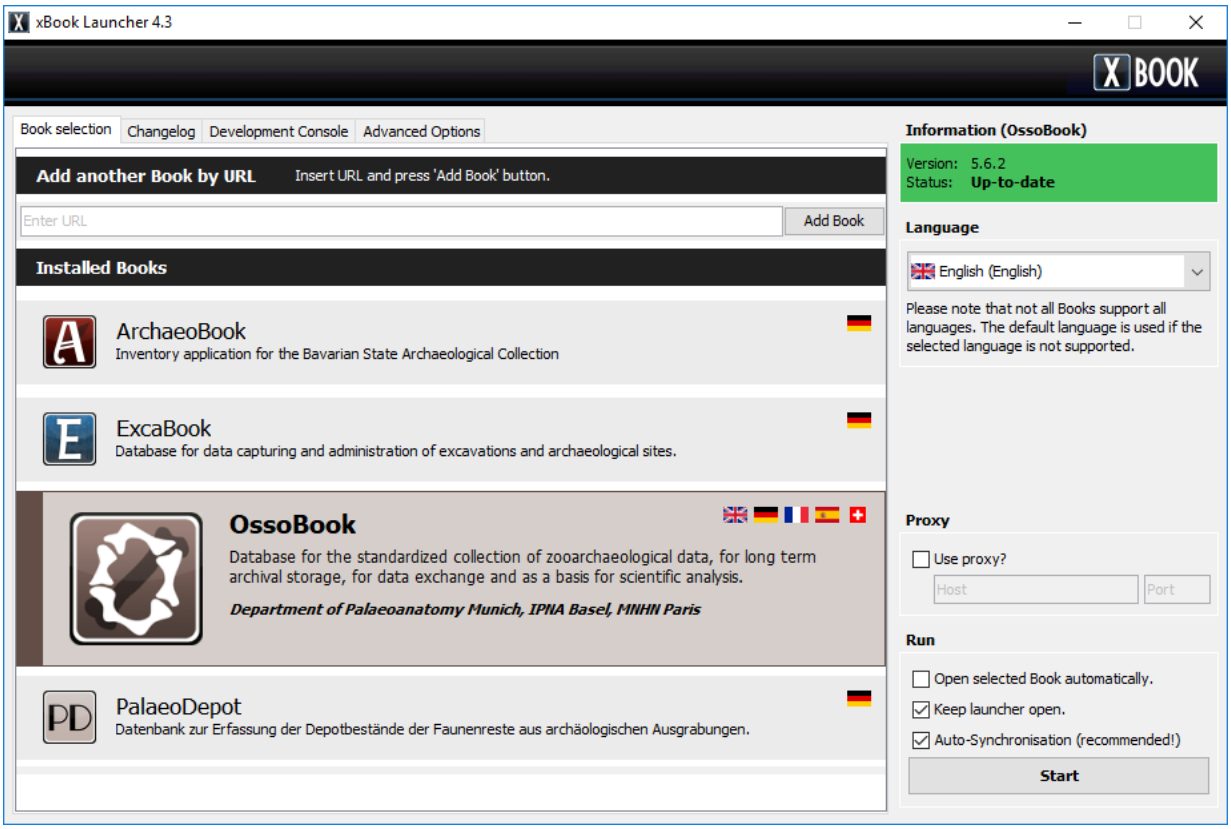


Figure 2.8: *The further developed version* XBOOK LAUNCHER *4.3.*

nization) and a frame to output the messages of the development console. The update functionality was still available which updates all BOOKs at the same time.

However, it became difficult to use one single launcher for all available BOOKs. Some of the BOOKs are scientific databases which are publicly available, others are local databases for the inventory of findings in museums or state collections. So not all BOOKs should be accessible in public. Furthermore, also the users do not need to have listed all existing BOOKs in their launcher. The previous additionally required an own instance of the XBOOK LAUNCHER for almost every BOOK, a roundabout way for the increasing number of BOOKs. Furthermore, it became necessary that the different databases were not hosted on the same server any longer. The individual BOOKs should be supported to save their data on their own servers. So the structure of the XBOOK LAUNCHER was renewed again and was developed to the current XBOOK LAUNCHER version 4.3 of which a screenshot can be seen in Figure 2.8.

Therefore, the XBOOK LAUNCHER was extended to enable adding single BOOKs dynamically to the list of BOOKs. The users can enter a valid URL where the configuration file of the corresponding "Book" is saved – for example *http://xbook.vetmed.uni-muenchen.de/books/ossobook* in the case of OSSOBOOK. The configuration file is defined

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<book>
    <displayName>OssoBook</displayName>
    <id>ossobook</id>
    <fileName>OssoBook.jar</fileName>
    <author></author>
    <bookColor>#624D47</bookColor>
    <sidebarBackgroundColor>#D5CECA</sidebarBackgroundColor>
    <defaultLanguage>en</defaultLanguage>
    <downloadLocation>
        http://xbook.vetmed.uni-muenchen.de/books/ossobook
    </downloadLocation>
    <languageFiles>
        <entry>
            <key>de</key>
            <value>
                <entry key="description" value=" [...] " />
                <entry key="descriptionShort" value=" [...] " />
                <entry key="author" value=" [...] " />
            </value>
        </entry>
        <entry>
            [...]
        </entry>
    </languageFiles>
    <languages>en</languages>
    <languages>de</languages>
    <languages>fr</languages>
    <languages>es</languages>
    <languages>de_ch</languages>
</book>
```

Algorithm 1: *The structure of the* book.xml *file that shows the* OSSOBOOK *configuration.*

to be named "book.xml" which holds all necessary information for the corresponding BOOK. This includes especially information that is used in the launcher to display the information about the BOOK (like application name, application ID, multi-language descriptions, etc.). It also defines the file that should be executed when running the BOOK and the location of the data that is required when installing or updating the application. The structure of the configuration file is illustrated with Algorithm 1.

This way, the XBOOK LAUNCHER can be used as a central platform to manage all available BOOKs, independent on their location. Especially the users benefit from this architecture. They can configure their launcher as they wish and do not have to install or integrate BOOKs they do not work with at all. Furthermore, the single BOOK applications can now be developed and hosted completely separately. In the past, all BOOKs had to use the same program version because they all depended on the same update files.

Now, every BOOK can be developed independently from other existing BOOKs and do not necessarily have to be updated to the latest version of XBOOK. This became more important with more and more different BOOKs being published by different institutions and developers.

In addition, the XBOOK LAUNCHER was extended with two technical features. With the increasing amount of users, more users required the usage of a proxy server in order to be able to connect to the server. For this, the XBOOK LAUNCHER was extended with a possibility to enter proxy information. Furthermore, it becomes apparent that a lot of users work on computers with a limited amount of computer memory which was not sufficient for using some of the features of XBOOK, like exporting the partial huge amounts of data. To solve this problem, the XBOOK LAUNCHER was extended with advanced options where the users can allocate more RAM to Java applications, and therefore for the BOOK.

2.3 Main Features of the xBook Framework

There are many different areas in the archaeological field of science which struggles with the same problems concerning the collection of data. OSSOBOOK is a database for zooarchaeological findings, but similar databases were also demanded for other areas, like anthropology and archaeology. Instead of individually implementing a completely new database for each area, we decided to extract the features from OSSOBOOK into a new framework called XBOOK.

This framework should provide all basic functionalities of the application and provide them to all instances which are called "BOOKs".

2.3.1 Synchronization

Because of the conversion of OSSOBOOK to Java, the collaboration with colleagues within the *Institute of Palaeoanatomy, Domestication Research and History of Veterinary Medicine*[7] and other institutions in gathering and maintaining zooarchaeological data is an important part of the concept. Initially, the data of the application was saved in one global database on the server of the university (cf. Chapter 2.1.2). The employees of the collaborating institutes could connect and work on this database. They could enter, view, and edit the zooarchaeological data directly on the server. This made it possible to work together on one project, enabled the exchange of data with other zooarchaeologists, and fulfills the need to sustainably store data on the cultural heritage as claimed by the UNESCO[8]. Later, a tunnel also enabled connections from other places as well.

[7] http://palaeo.vetmed.uni-muenchen.de
[8] http://www.unesco.org

But zooarchaeological data is often gathered in field work, i.e. at remote sites that do not offer a convenient environment for IT services. Therefore, it is typically not possible to enter the data into databases that have to be accessed via an Internet connection. A synchronization process was required. A Server–Client architecture was implemented to ensure working offline at remote places, but also storing data globally, where it can be shared with other users. [LKK+14]

The first concept for the synchronization was drafted in 2008.

> *"A client–server architecture ensures that each client [...] manages its own local database that is schema-equivalent to the central database at the server. This way, each client can make its updates locally, independently, and – most important – offline. At a given time, e.g. when a network connection to the server is established, the client and the server synchronize, i.e. the updates of the client are inserted into the central database at the server and the update of the server."* [KKO+09]

However, the direct connection to the database without any synchronization would still be possible within the institutes. This architecture is drafted in Figure 2.9. The concept of the synchronization was initially implemented by Jana Lamprecht in 2008 [Lam08].

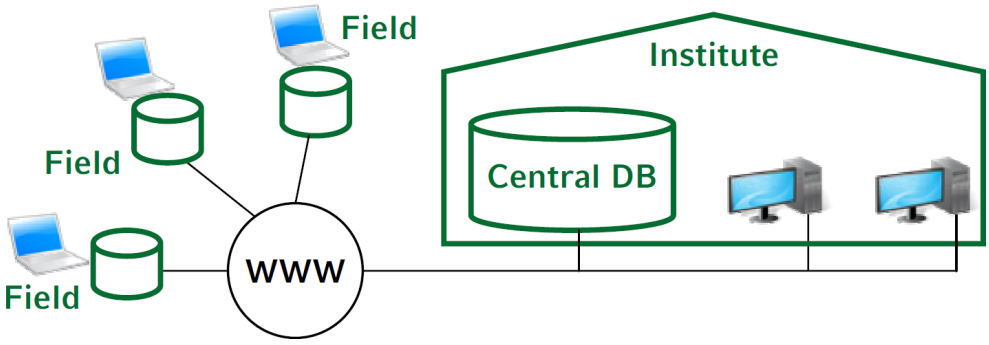


Figure 2.9: *The origin draft for the architecture for the* OssoBook *synchronization. [KKO+09]*

Since this implementation, the concept of the synchronization was updated in many development cycles. The synchronization process itself was enhanced and was extended with new features.

First of all, the clients do not directly connect to the database anymore, but to a server application that handles the requests and data manipulations. The server application also enables the feature to let the users manage their basic profile information, especially to change and recover their password if needed – a feature that was lacking before.

Additionally, the possibility to work on different computers with one account was added for a single user. In the origin implementation, this was theoretically also possible,

but the technical implementation could cause the loss of data sets during the synchronization process.

The server application was originally written in C/C++, later it was replaced by a PHP program to avoid problems with restrictive firewall settings that made a communication impossible through other ports than port 80 (HTTP) or 443 (HTTPS). Due to security reasons, all the communication has to go through the server application. Therefore, a direct connection to the central database is not supported anymore, not even inside the institutes.

At the same time the logic of the synchronization was also improved and simplified, especially the synchronization speed was optimized in several development cycles.

Also the synchronization panel in the application was extended, now it provides additional information about each project, like the number of available entries in the projects, the project owner, how many entries are not synchronized yet, and which projects were already synchronized. This serves to improve the overview about the current status of the data for the user.

The detailed implementation of the synchronization is described in Chapter 3.1, where we explain the concept and implementation of the synchronization process in XBOOK. A screenshot of the latest panel of the synchronization in OSSOBOOK can be viewed in Figure 2.10.

2.3.2 Graphical User Interface

To enable a flexible graphical user interface for the framework and its applications, it was necessary to modify the existing visual composition. All static elements had to be replaced with dynamic elements that can be individually adjusted for each BOOK. However, elements that are required for each BOOK have to be integrated to the graphical user interface by default.

In the first step, we replaced the navigation. The old navigation bar was not designed for elements being dynamically added or updated. It was composed of different images for the normal, hovered, and selected states. The displayed texts on these elements were part of the images which made translations difficult. For the XBOOK framework, the navigation elements are now arranged side by side, are clarified with an individual icon and a short text label. The elements which were displayed in the sub navigation bar before, can now be accessed by opening a pop-up menu. All required navigation elements which are necessary to be accessed in every BOOK (like ‘Projects’, ‘Data Entry’, ‘Listing’, ‘Tools’, ‘Help’, and ‘Log-out’) are displayed by default. Individual main navigation elements can individually be added by each BOOK. Adding new elements to the pop-up menu is supported.

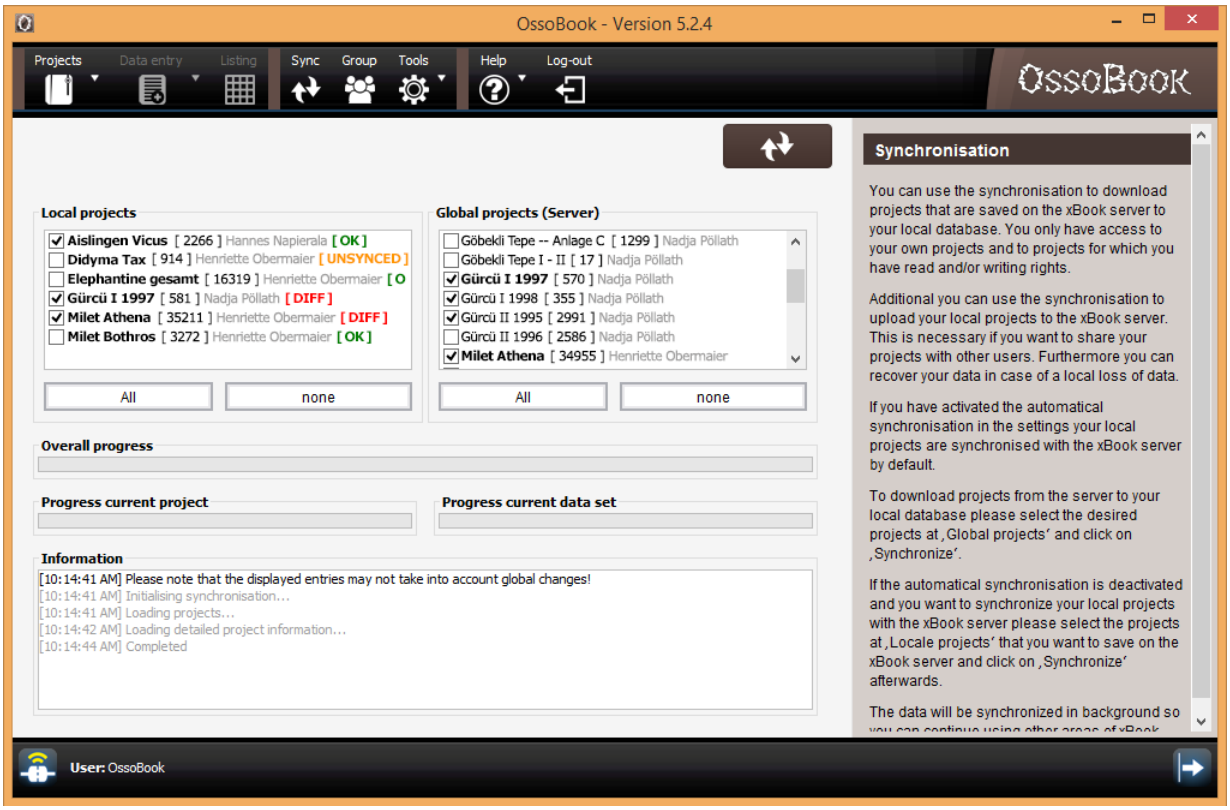


Figure 2.10: *The synchronization panel in* OssoBook *5.2.4. [LKK+16b]*

The functionality of the elements of the graphical user interface is fully implemented, but the abstract classes provides methods that have to be overridden and implemented by each Book. This way the elements can be customized by defining their contents and configurations. As an example, the input fields – either the predefined or custom, new ones – can be defined with their settings and information for the database individually for each Book. The overridden classes define the individual input fields. The logic for handling and displaying the data is defined in the implementation of the xBook framework. So each Book can have different input fields in the input mask, like indicated in Figure 2.11.

The individualization of other elements of the xBook framework is similar. The overridden, abstract methods are used to define the information displayed in the project overview screen. The location where to save settings, individual output for the export, the general style of the Book (like the logo, the name of the Book, used colors), and also the definition of the elements are displayed in the main and sub navigation.

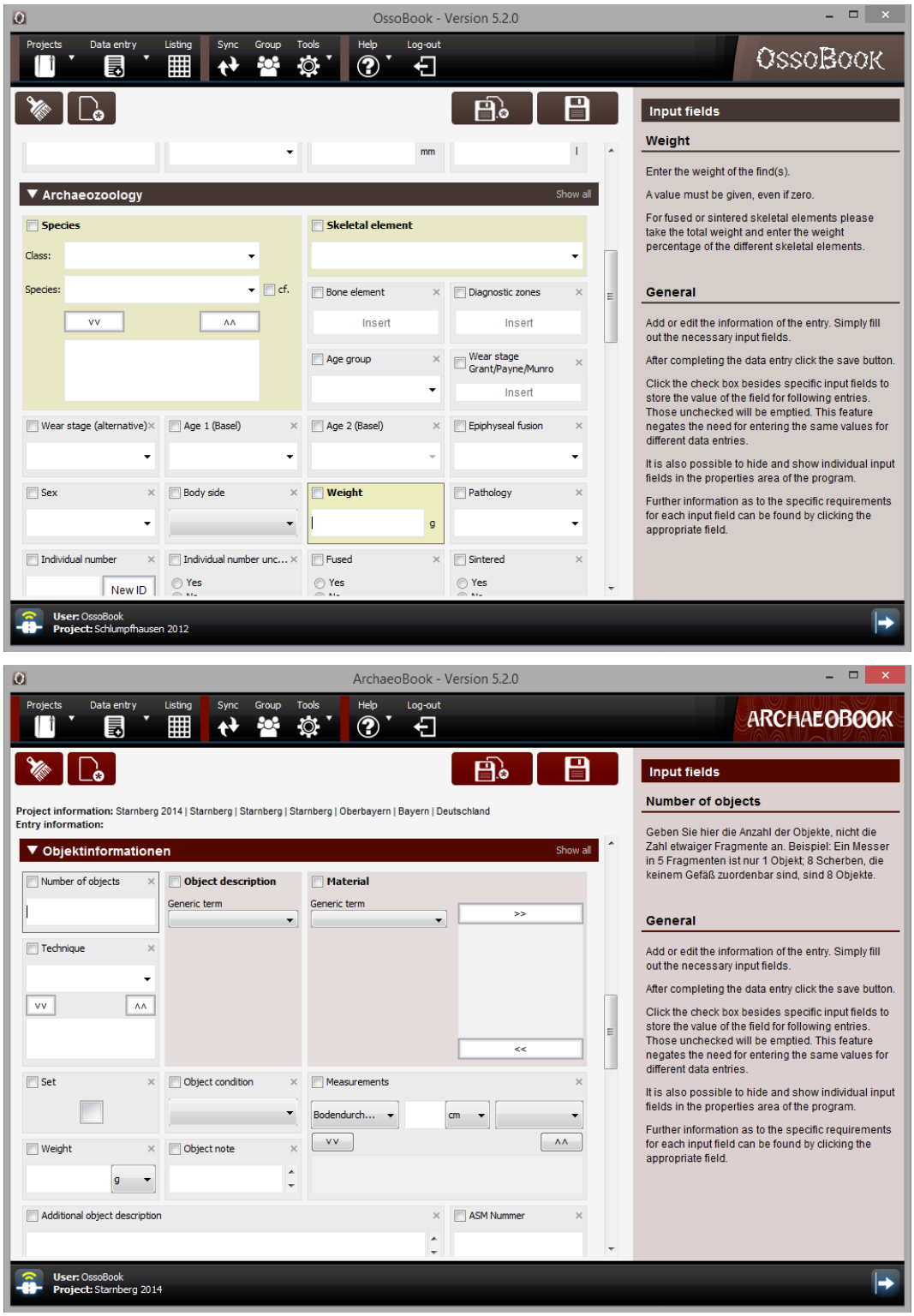


Figure 2.11: *The input mask of* OssoBook *(top) and* ArchaeoBook *(bottom). Both applications are based on the* xBook *framework that provides a basic graphical user interface and functions, but allows customization like e.g. custom input fields. [LKK+16b]*

2.3.3 Multiple and Cross-linked Input Masks

Before, each BOOK only had one single input mask with input fields where the users could enter data. During the development of the XBOOK framework and its BOOKs it became clear that different, separated input masks are required, especially to avoid redundancy and inconsistency.

So we added the possibility that each BOOK may have an arbitrary number of input masks with individual input fields. These input masks may be independent from each other, but may also be cross-linked among each other.

As an example, many archaeological findings are often grouped in boxes or bags, but the information about these boxes and bags should not be entered for each finding again. An own input mask for the boxes or bags would be an advantage to avoid extra effort when entering data. Also it is possible that the content of a box or bag changes, so it is easier to select another box or bag instead of re-entering the data for each concerned finding.

The realization of these cross-linked input masks does not need a complicated architecture of the tables in the database. But it requires some more complex SQL queries and adjustments in the graphical user interface, especially for the display and selection in the input mask, the representation in the listing and the export, and adjustments in the synchronization process.

2.3.4 Listing and Export

Besides the data recording, the listing of entered data is also important. The entered data of all input fields has to be meaningfully displayed in the data listing, i.e. the values should be human readable.

For most of the input fields the value can be directly displayed like simple text or numeric values. Other more complex input fields had to be adjusted for the readability, for example by displaying the corresponding text value instead of the corresponding ID, or by displaying a readable date or time format instead of a timestamp. Input fields with multi-inputs should also display all entered values, not only the IDs. For cross-linked values of other input masks representative text for the cross-linked entry was necessary as well.

The listing also contains a filter possibility to be able to search for entries containing a specific search term. XBOOK also supports a bulk edit mode: The entries in the listing can be selected and opened to be able to apply the same changes to multiple data sets at once, as seen in Figure 2.12.

The export of data is also an essential feature. Most of the archaeologists and bio-archaeologists still do their analyses in extern tools or applications like Excel, or want

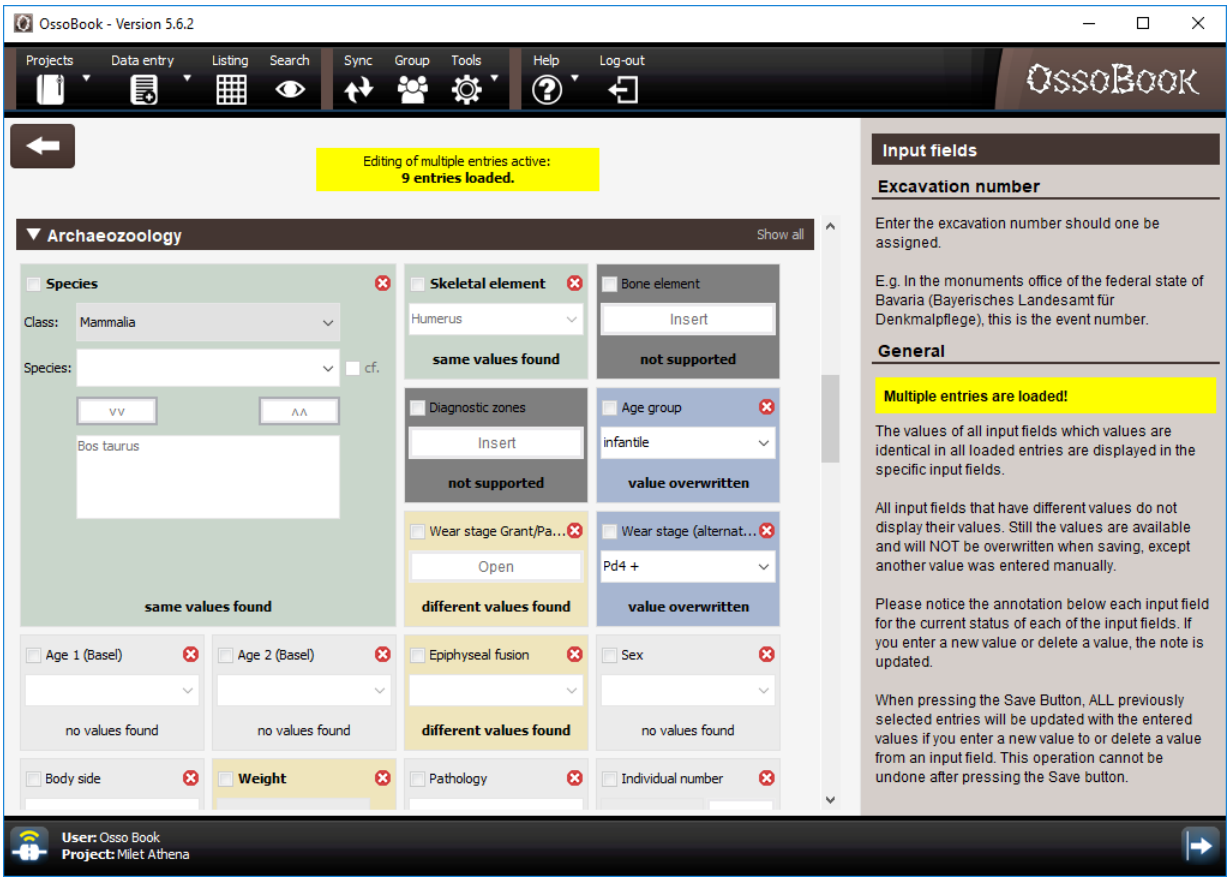


Figure 2.12: *The multi-edit mask of* OssoBook *that allows to apply the same changed to multiple data sets.*

to print them for non-digital archiving. Therefore, xBook provides an export method that saves the data in a file on the local system. There are two supported files systems to export the data: Comma-separated values (CSV) and spreadsheet (XLS/XLSX) files.

Both, the listing and the export have full support for multiple input masks. In the listing, a selection of the input mask is available where the user can select which data has to be displayed. For the export, it is possible so select single input masks. If a Book with multiple input masks is exported, the data of each mask will be saved in an own CSV file, or in an own sheet of the XLS/XLSX file.

2.4 Applications using the xBook Framework

In archaeo-related disciplines, there are many areas with the same or similar workflow. XBOOK offers a framework in which all databases benefit from the provided features. Currently, there are seven different instances of the XBOOK framework:

OssoBook [KLK+18b]

A scientific database for the standardized collection of zooarchaeological data, for long-term archival storage, for data exchange, and as a basis for scientific analyses. Recorded are archaeological, morphological, and taphonomic data from zooarchaeological analyses, and the results of molecular biological analyses and radiocarbon datings.

Figure 2.13: OSSOBOOK *icon*

OSSOBOOK is currently[9] used by approximately 350 users including scientists, Ph.D. students and students in institutes of universities, museums, and scientific collections with zooarchaeology as field of work as well as freelance zooarchaeologists. The service is available in German, English, and French. In the context of the *IANUS*[10] project of the *German Archaeological Institute*[11], Berlin, Germany, OSSOBOOK will serve as a standard for the zooarchaeology domain. As introduction of the usage of the software there are annual workshops in several European countries.

Used by

- *Bavarian State Collection for Anthropology and Palaeoanatomy Munich*, section Palaeoanatomy, `http://www.sapm.mwn.de`
- *Ludwig-Maximilians-Universität München*, *Institute of Palaeoanatomy, Domestication Research and History of Veterinary Medicine*, `http://palaeo.vetmed.uni-muenchen.de`
- *Ludwig-Maximilians-Universität München*, *ArchaeoBioCenter*, `http://www.archaeobiocenter.uni-muenchen.de`
- *University of Basel*, *Integrative Prehistory and Archaeological Science* (*IPNA*), `http://ipna.unibas.ch`
- members of *BioArch*, `http://www.archeorient.mom.fr/recherche-et-activites/participation-a-des-reseaux/GDRE-BIOARCH`
- members of *ArchaeoZoologenVerband*, `www.archaeozoologenverband.de`

[9] as of February 2018
[10] `http://www.ianus-fdz.de`
[11] `http://www.dainst.org`

ArchaeoBook [KLK+18e]

An inventory database for archaeological findings.

ARCHAEOBOOK is used as an intern inventory database of the *Bavarian State Archaeological Collection Munich*, Germany. All objects and their characteristics are saved in the database. Besides, the database is also used to track the current location of the objects within the collection and to document loans.

Figure 2.14: ARCHAEOBOOK *icon*

Used by

- *Bavarian State Archaeological Collection Munich*, `http://www.archaeologie-bayern.de`

ExcaBook [KLK+18f]

A database for homogeneous capturing and administration of archaeological excavations by qualified companies.

EXCABOOK is operated by the *Bavarian State Department of Monuments and Sites* to allow a standardized collection of archaeological data to archive a clear and transparent archaeological monument conservation [Bay16a] [Jan18]. Extern excavation companies are instructed to enter data to EXCABOOK. Once the gathering for an excavation or a part of it is completed, the data is passed to the state department using the build-in synchronization method. Hard copy reports for long-term archiving – which are prescribed by law – can directly be generated in the database application. Methods to import the available data from EXCABOOK to FIS[12], a specialized information system, is in preparation. The development of FIS and the geo web service is part of the e-government initiative of the Bavarian government[13].

Figure 2.15: EXCABOOK *icon*

Used by

- *Bavarian State Department of Monuments and Sites*, `http://www.blfd.bayern.de`
- several instructed excavation companies

[12] `http://www.blfd.bayern.de/denkmalerfassung/stabsstelle/fis/`
[13] `http://www.gdi.bayern.de`

AnthroDepot [KLK+18c] / PalaeoDepot [KLK+18g]

Figure 2.16:
ANTHRODEPOT *and* PALAEODEPOT *icons*

The sections anthropology and palaeoanatomy of the *Bavarian State Collection for Anthropology and Palaeoanatomy Munich* are spatial separated in offices and collections.

The two inventory databases ANTHRODEPOT and PALAEODEPOT are developed for the stockpile management of human skeletons in Dornach, Germany, and faunal remains in Poing, Germany, to manage the location, documentation, and loans of findings.

The collection of the section anthropology stores skeleton remains of about 68,000 individuals. The yearly increase is about 1,000 skeletons. The collection of the section palaeoanatomy contains several millions of faunal remains of archaeological excavations, every year up to 50,000 findings are added.[14]

In future, interfaces between ANTHRODEPOT and ANTHROBOOK, respectively PALAEODEPOT and OSSOBOOK are planned. These data exchange methods will enable to transfer data from the inventory databases of the collections to the scientific databases. This will save time while migrating the archaeological information. In ANTHROBOOK and OSSOBOOK, these information will be enriched with results from determinations and analyzed before being provided to the scientific infrastructures of the domains.

Used by

- *Bavarian State Collection for Anthropology and Palaeoanatomy Munich*, `http://www.sapm.mwn.de`

[14] as of end of 2017

InBook (under development) [KLK+18d]

A database for archaeological findings.

INBOOK is developed as a database of the *Bavarian State Archaeological Collection Munich*, Germany, to enable the documentation of incoming, new objects that are not yet part of the collection and therefore not captured in the database ARCHAEOBOOK.

Figure 2.17: INBOOK *icon*

Used by

- *Bavarian State Archaeological Collection Munich*, `http://www.archaeologie-bayern.de`

AnthroBook (under development) [KLK+18h]

Guidelines for the standardized collection of morphological data, to allow a consistent data collection and long-term storage were developed for the anthropology. These standards are currently being implemented in the scientific, anthropological database ANTHROBOOK.

Similar to OSSOBOOK, ANTHROBOOK is planned for long-term archival storage, for data exchange, and as a basis for scientific analyses.

Figure 2.18: ANTHROBOOK *icon*

Used by

- *Bavarian State Collection for Anthropology and Palaeoanatomy Munich*, section Anthropology, `http://www.sapm.mwn.de`

Chapter 3

Collaboration, Sharing, and Retrieval Methods for Archaeo-related Data

Attribution

This Chapter uses material from the following publications:

- Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. A Generic Framework for Synchronized Distributed Data Management in Archaeological Related Disciplines. In *10th IEEE International Conference on e-Science, eScience 2014, São Paulo, Brazil, October 20-24, 2014*, pages 5–12, 2014. [LKK+14]
- Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. Synchronized Data Management and Its Integration into a Graphical User Interface for Archaeological Related Disciplines. In *Design, User Experience, and Usability: Users and Interactions - 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part II*, pages 317–329, 2015 [KLK+15]
- Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. A generic framework for synchronized distributed data management in archaeological related disciplines. *Future Generation Computer Systems*, 56:558–570, 2016 [LKK+16b]
- Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Henriette Obermaier, and Christiaan van der Meijden. Reverse Mediated Information System: Web-based Retrieval of Distributed, Anonymous Information.

In *16th International Conference on WWW/Internet 2017, Vilamoura, Portugal, 2017*, pages 63–70, 2017 [LKK+17]

- Johannes-Y. Lohrer, Daniel Kaltenthaler, Florian Richter, Tatiana Sizova, Peer Kröger, and Christiaan van der Meijden. Retrieval of Heterogeneous Data from Dynamic and Anonymous Sources. In *8th IEEE International Conference Confluence 2018 on Cloud Computing, Data Science and Engineering, Noida, Uttar Pradesh, India*, pages 592–597, 2018 [LKR+18]

See Chapter 1.2 for a detailed overview of incorporated publications.

The technical infrastructure described in Chapter 2 builds the basis to record archaeological and bioarchaeological data. This is an important structure to enable a flexible, dynamic, and reliable way to gather data for each of the single archaeo-related disciplines. However, a functional database application that only stores data is not sufficient in the archaeo-related workflow:

1. The experience with users of OssoBook [KLK+18b] and the zooarchaeological domain has shown that the data of findings from large excavations cannot be entered into the databases by just one single scientist. A pure database running on one computer makes it hard to enter (in some cases tens or hundreds of thousands of) new data sets. Certainly, using an online database where several coworkers can collate data to the same project, would be an intuitive approach to realize this kind of collaboration. However, in the archaeo-related context this method is not applicable since data is often entered on notebooks directly on the field where an Internet connection is not available. Therefore, we need a solution to realize collaborating and collating data on the same project.

2. The exchange of data is a very important aspect in archaeo-related sciences. It is common, not only to take into account single projects and excavations for analyses, but also consider data from other excavations to figure out similarities, differences, and relationships. However, exporting data and manually sending the data to colleagues is a cumbersome process. Therefore, a method to allow the sharing of data is important and necessary to enable extensive analyses.

3. Archaeological and bioarchaeological data is not stored centrally, but distributed in several physical locations. Even the data of findings of one project is distributed, although the findings are from the same excavation: Data of animal bones is stored in zooarchaeological databases, data of human bones in anthropological databases, data of objects and artifacts in archaeological databases, etc. Furthermore, each

data source uses heterogeneous structures to store the data. Therefore, we need a method to retrieve archaeo-related information from distributed and heterogeneous data sources.

The contribution of this Chapter is to provide solutions for these issues which are split in two parts. In Chapter 3.1 we explain a generic synchronization method for XBOOK that enables collaboration and sharing of data. Then we introduce an architecture to allow retrieving archaeological and bioarchaeological data from distributed and heterogeneous data sources in Chapter 3.2.

3.1 Collaborating and Sharing Archaeo-related Data

As in many other applications, in archaeology and bioarchaeology a main part of the work comprises the collecting, sharing, and analyzing of data. Often many researchers from different institutions and even varying countries are involved in excavation projects. Therefore, entering data directly into databases is required to in order to access data from different places easily and work simultaneously on recording as well as analyzing the data. Archaeo-related data is often gathered in field work, for example at remote sites that do not offer a convenient environment for IT services. It is typically not possible to enter the data into databases that have to be accessed via an Internet connection. As a consequence, IT services are hardly used in these projects. On the contrary, data is typically recorded on paper and is (if at all) later processed electronically by using proprietary and/or file-based data management tools like Excel for doing simple descriptive statistics. This is significantly inconsistent with the need to sustainably store data on the cultural heritage as claimed by the UNESCO[15].

Obviously, researchers from the archaeological and bioarchaeological domains would significantly benefit from a solution that solves the problem of multiple users that need access to data recording and data analysis even if a permanent Internet connection cannot be established. A synchronization process is required, that implements a Client–Server architecture as visualized in Figure 3.1 to enable working offline at remote places and also storing data globally where it can be shared with other users.

Existing commercial solutions for this problem are typically integrated in a dedicated database management system and/or cloud service. However, for license or financial reasons as well as due to privacy concerns, not all institutions and departments can or want to resort to these systems. Budgets for archaeological and bioarchaeological excavation projects are typically optimized in terms of logistics and manpower. Reserving a considerable part for IT infrastructure is completely unrealistic. In addition, as long

[15] http://www.unesco.org

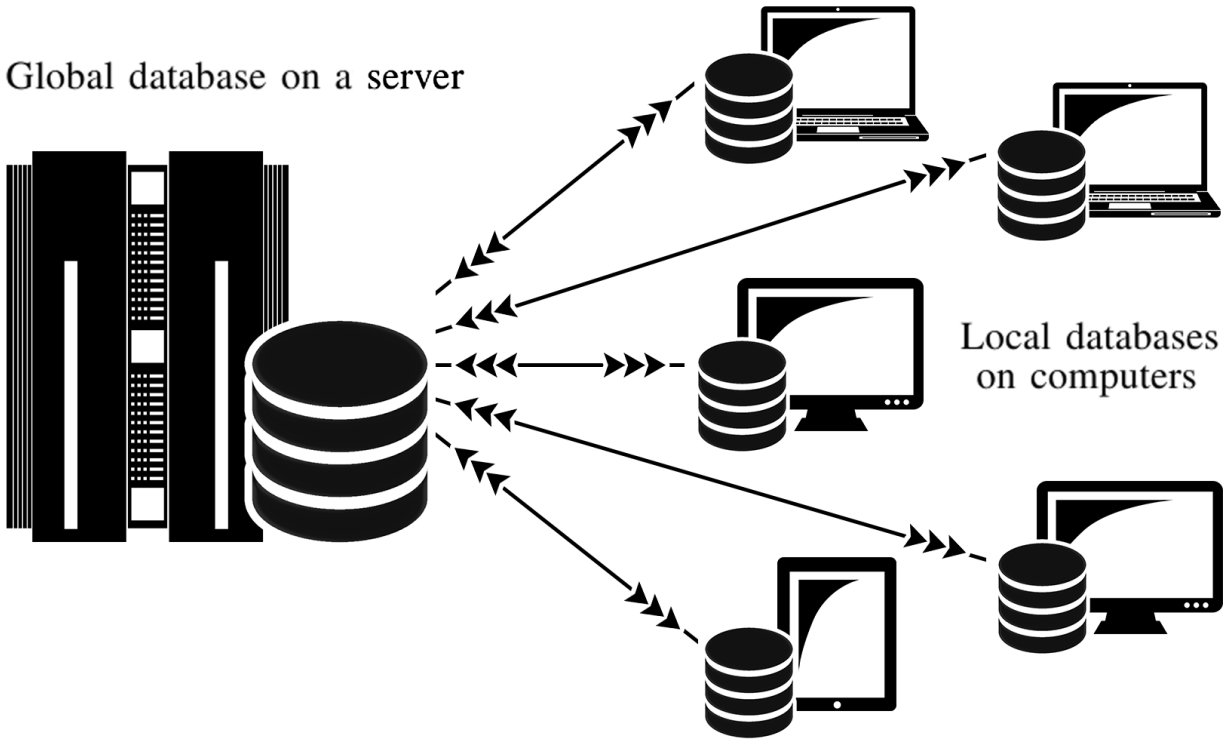


Figure 3.1: *An architecture for a synchronization process: The local clients are connected to the global server. The synchronization allows data exchange, thus data can be recorded on the local machines, but can also be backed-up and shared via the server.*

as the data is not yet analyzed and the results are not yet published, the participating researchers are very wary about transferring their data to commercial cloud services.

In summary, the main contributions of this Chapter are as follows: We list a set of requirements that should be used for a synchronized distributed data management and data analysis in the archaeo-related sciences that have been extracted from comprehensive discussions with domain experts, reflecting their typical working procedures (cf. Chapter 3.1.1). We discuss existing solutions for synchronized distributed data management and data analysis in Chapter 3.1.2. We describe the synchronization process that we implemented in XBOOK in more detail in Chapter 3.1.3. This synchronization process is independent of the data model and could potentially be used in any application domain, if needed. Finally, we show the realization of the synchronization within the application in Chapter 3.1.4.

3.1.1 Problem Formulation

In this Chapter, we discuss the requirements of a data synchronization process for the archaeo-related sciences in more detail. By developing a database for zooarchaeologists

(cf. Chapter 2), we have identified the following requirements of a synchronization that have to be fulfilled so that working with the data is possible.

Let us note that depending on the use case of the application some requirements listed below are obligatory and additional requirements might exist. However, the following list has been extracted after extensive communication with domain scientists analyzing their particular working procedures.

A synchronized distributed infrastructure for data management and data analysis in the archaeo-related sciences should address the following issues (the order of appearance is arbitrary and does not reflect priorities):

- **Distinctability of entries:**
  Two different entries have to be distinct from each other, no matter on which local database they were created.

- **Conflict Handling:**
  Different users are able to work on the same data simultaneously on different local databases. Conflicts generally occur when two users have independently made a change to the same entry. When synchronizing the global server, the server cannot figure out which of the two versions has to be used. Therefore, the synchronization has to recognize that a conflict occurred and provide options to solve it.

- **Time-delayed execution:**
  It cannot be guaranteed that a user of the database can always execute the synchronization process. This could be technical reasons like temporary Internet disconnects, but also logistic reasons e.g. there is no Internet connection available. It has to be possible to continue working offline and synchronize the data at a later time when the computer is reconnected to an Internet connection again.

- **Interruptible:**
  After a loss of connection or if the user interrupts the exchange of data, the synchronization process has to be able to continue later and no data may be lost or corrupted.

- **Modularity:**
  To avoid the transfer of unnecessary data, a selection of data is required. In general, a particular user does not need or even is not allowed to get full access to the global database on the local device. The user should be able to select only parts of data for synchronizing in order to save time and to reduce data overhead. Thereby, data should be separated in modules (“projects”). Furthermore, the local database of the users should not save any data sets for which they have no reading rights.

- **Currentness of look-up table data:**
  The look-up table data (called "Code Tables" in our architecture) that is saved on the database of the global server needs to be updated many times. This concerns regular as well as dynamic updates of this data. Administrators with the permission to edit have to be able to insert new entries and edit or delete existing ones. These changes have to be passed to the users as soon as possible to ensure that they can work with current data.

- **Rights management:**
  Not every user needs to see all the data. The synchronization have to check if the user, who has requested any data, has appropriate rights before sending or committing specific data.

- **Easy to use:**
  The synchronization should be an automatic process that can be run with few mouse clicks. The user is most likely not a skilled computer user, but has to be able to use the synchronization. Therefore, the synchronization process should be carried out without any external data devices.

As discussed above, in addition to these specific requirements, a synchronized infrastructure for the archaeological sciences should also be cost effective and ensure the privacy of unpublished data.

3.1.2 Evaluation of Existing Synchronization Methods

In general, the basic idea of synchronization is not a new one. Both, scientific and industrial databases often require functions to share data in any way, especially if several users should be able to work on the data simultaneously. Synchronization techniques exist in most commonly used database systems worldwide. In the following, we discuss the synchronization techniques of such systems, including Microsoft SQL Server, Oracle, and MySQL in the context of the requirements derived in Chapter 3.1.1.

- **Snapshot replication:**
  Snapshot Replication is an easy way to share data that is used in Microsoft SQL Server and Oracle. It "distributes data exactly as it appears at a specific moment in time and does not monitor for updates to the data" [Mic08, Types of Replication]. The entire snapshot is generated and sent to the receiver, a selection of data is not possible. But especially if there is a lot of data saved in the database, it is necessary to transmit all of this data – so the merge function allows several users to work on different sets of data, but does not reduce the amount of data that has to be sent. At

least Oracle supports partial snapshots [Ora13a, Understanding Replication], but synchronizing single data sets is not supported. Furthermore, Snapshot Replication is laid out if data changes are infrequent. This is an unrealistic scenario in our application.

- **Transaction replication:**
  Microsoft SQL Server supports a transaction replication that is not based on data sets, but on transactions. "[D]ata changes and schema modifications made at the Publisher are usually delivered to the Subscriber as they occur (in near real time). The data changes are applied to the Subscriber in the same order and within the same transaction boundaries as they occurred at the Publisher" [Mic08, Types of Replication]. Therefore, a transactional consistency is guaranteed even on high volume of insert, update and delete activities. But the transaction replication is developed for server environments. The clients must have a permanent connection with the global server. Working offline is not supported.

- **Replication with streams:**
  Using Oracle Streams it is possible to enable replication as well. "The stream can propagate information [...] from one database to another. [...] you control what information is put into a stream, how the stream flows or is routed from database to database, what happens to messages in the stream as they flow into each database, and how the stream terminates" [Ora13b, Understanding Streams Replication]. So Oracle Streams are powerful features that also support replications with the help of three background processes: The capture-process to collect information, the propagate-process to get the data out of the stream and the apply-process to handle the local changes (for more details, see [Ora13c]).

  However, a synchronization with the requirements described in Chapter 3.1.1 is not possible by using Oracle Streams. The communication cannot be managed by the global server because all processes have to be defined by the local databases. To ensure that no reading operations are executed on the global database, it is necessary that the users are not administrators of their own local databases. The other way round it means that the global server is responsible for the local databases as well – but this is not possible in practice [Lam08].

- **Merge replication:**
  Another type of replication in Microsoft SQL Server is merge replication. "[D]ata changes and schema modifications made at the Publisher and Subscribers are tracked with triggers. The Subscriber synchronizes with the Publisher when connected to the network and exchanges all rows that have changed between the Publisher and Subscriber since the last time synchronization occurred" [Mic08, Types

of Replication]. The merge replication matches with the requirements pretty well, but the complete data sets have to be in the database of the subscribers. Therefore, it is not possible only to store, or synchronize specific projects. This makes the management of permissions for the synchronization also impossible.

- **Master-slave replication system:**
  MySQL supports replication by using a master-slave system. It "enables data from one MySQL database server (the master) to be copied to one or more MySQL database servers (the slaves). Replication is asynchronous by default; slaves do not need to be connected permanently to receive updates from the master"[MyS13]. Unfortunately, it is only possible to replicate all databases, selected databases or selected tables within a database - the replication of single data sets is not supported. Furthermore, MySQL does not include any conflict management.

- **Direct working on the global database:**
  Even if it is not really a replication system, it is still a possibility that the users connect directly to the server and work on the global database. It would be an easy solution that fulfills almost all of the requirements of Chapter 3.1.1 – but of course a permanent Internet connection is required.

While all of these methods are sufficient for many areas of application, none of them cover all of the requirements stated above for the archaeo-related context. Only few of the available solutions provide the ability to work offline and none of them allow synchronization of single data sets, which are two of the most important requirements for archaeologists and bioarchaeologists. Therefore, we created a synchronization that fulfills all previously stated requirements, is independent of the database management system and hence can be used for any database system.

3.1.3 Synchronization Implementation

To achieve a synchronization that fulfills the requirements described in Chapter 3.1.1, adjustments to the database and the application are necessary. Below, we describe the concepts and methods for the synchronization.

Realization in the Database

To achieve the synchronization we are aiming for, some necessary additions in the local and global database had to be made. The database scheme of the tables has to be extended by some additionally columns: The "Database ID", the "Status", and the "Message ID". These are explained below.

Database ID

One of the most important concepts is the database ID. A column for this ID is added to each table a user can insert data. In addition to the existing primary keys, this column is marked as a primary key as well, to allow several distinct entries with the same ID from various databases. The number value of this database itself is stored in a separate table (in XBOOK we use "version") and also as a property in the configuration settings of the operation system. The database ID is generated when the user connects to the server the first time, or if the value of the ID in the database is different to the value in the properties. This is to prevent errors when a user copies his database to a different computer.

When the user enters a new entry in the database, the database ID is automatically filled in with the above defined value. If the entry is edited, the database ID will not be touched.

Because we want to enable a modular approach with projects, all tables also have to add the ID ("ProjectID") and the database ID ("ProjectDatabaseID") of the project, otherwise the entry cannot be assigned to a specific project (cf. Table 3.1).

ID	DatabaseID	ProjectID	ProjectDatabaseID	UserID	ExcavationNr	ArchaeolUnit	...
2	1073	1	1073	108	M-2011-13-1	1052	...
2	1079	1	1079	114	76/3724/230	630	...
2	1096	1	1096	132	HY.04.ISI	01.013	...
2	1147	3	1147	8	M-2013-732-1	4	...
2	1155	1	1096	132	HY.03.ISI	01.080	...
2	1174	1	1316	217	M-2007-58838	$13819C$	...
2	1218	1	1218	113	HY.81	2275	...
2	1337	7	1028	50	N.112	$154B$	...
2	1012	3	1334	6	M-2013-455-2	8	...
...	...	...	...	...	...	...	...

Table 3.1: *The table "inputunit" of the database of* OSSOBOOK *as an example for the primary keys* ID*,* DatabaseID*,* ProjectID *and* ProjectDatabaseID*. These primary keys are necessary in every data table of* XBOOK*.*

Status

To achieve conflict management as well as identification which of the entries have to be updated, locally the column "Status" (cf. Table 3.2) is added to each table. The status actually stores for each entry a timestamp of the last time the entry was modified

on the server. This value is modified only on the server via a simple trigger that is defined in Algorithm 2.

```
1 SET NEW.Status = NOW() + 0;
```

Algorithm 2: *SQL Query to update the Status for the synchronization*

This trigger updates the status as soon as the value is inserted to or updated in the database. Zero is added to the current time to convert the format from "`yyyy-MM-dd HH:mm:ss`" to "`yyyyMMddHHmmss`" to receive a sortable and numeric value.

If the entry on the client has a lower (older) status, it needs to be updated. If the entry on the server has a higher (newer) status when committing data to the server, a conflict occurred. That means that in the meantime someone else modified the entry. Then this entry can be marked locally as a conflict.

In the archaeo-related context it is not a realistic scenario with more than one scientist collecting data about the same objects (e.g. bones of an animal) simultaneously, so conflicts will not occur frequently. Anyway, conflicts may occur anytime, therefore a conflict management system is necessary. The one we implemented in XBOOK is described in Chapter 3.1.5. Other working areas may need more complex conflict management systems to avoid conflicts hampering the synchronization process.

Message ID

The next important addition is the column "MessageID" (cf. Table 3.2). It locally stores the current status of the entry. The different options for the "MessageID" are "Synchronized", "Changed", and "Conflicted", as described below:

...	Value	Status	MessageID	Deleted	...
...	24A45 H1	20150624145906	0	Y	...
...	6773	20150624150044	-1	N	...
...	Lab01 C1	20150624150058	-1	N	...
...	Lab03 B12	20150624145945	0	N	...
...	15D02 Ka3	20150624143945	1	N	...
...	Lab04 Au8	20150624143945	0	Y	...
...	...	...	...	...	...

Table 3.2: *The three important system columns in the input tables: "Status", "MessageID" and "Deleted". The message ID "1" means that the entry was edited, but not synchronized yet. Entries with Message ID "-1" are conflicted.*

- **Synchronized (0):**
  Indicates that the entry is synchronized and no uncommitted changes were made. This does not mean that the entry is up-to-date, it just means the entry has no local changes and can be synchronized.

- **Changed (1):**
  Indicates that the entry was changed locally. This prevents the entry being updated with data from the database to prevent overriding changes. In most cases, this would also mean that a conflict occurs, but conflict checking is already done when the entry is committed, so no need to take action here. After the entry is successfully committed to the server, the status is set to "synchronized" again.

- **Conflicted (–1):**
  Indicates that this entry is conflicted and therefore cannot be committed or updated. A conflict occurs when an entry that is changed locally has an older (lower) status than the entry on the server. This means that the changes on the local entry were made on an older version of the entry, and if the entry would be committed, there would be possible loss of data.

Deleted

Due to the fact that the synchronization can only identify changes with the check of the "Status" column, it is not easily possible to delete entries. The local database has to be able to inform the global database about deleted entries. If the entry is simply deleted, no information is available at all. At the same time, the server has to hold the information which entry is deleted to be able to synchronize the information to other clients.

Still, there needs to be the option to delete an entry. To solve this problem, a column "Deleted" was added (cf. Table 3.2). It is an enumeration that has only two options: "N" and "Y" – with "N" as default value. Instead of deleting an entry the value of the "Deleted" column is set to "Y". Then this change can be synchronized to the global server. From there it can also be synchronized to other clients.

When the client gets the information that an entry is deleted, it can safely delete the entry locally. On the server however an entry is never deleted, because the information about the change has always to remain available for the clients.

The same logic is applied to Code Table entries with the exception that entry tables are only synced to the clients.

Realization in the Application

Some of the basic requirements are already fulfilled in the database, but as the data has also to be sent and some features are still missing, several concepts had to be developed in the application as well: A method for updating the database, the usage of "Code Tables", the integration of "Managers" in the application, the implementation of a flexible data structure to handle the data, the rough concept of the synchronization process, and a solution to be able to delete items considering the functionality of the synchronization. These concepts are described below.

Database Update

To allow constant updates to the application including changes in the database scheme, we added a check for the version of the database. This number is stored in the "version" table and is compared to the built-in number in the program. If the numbers do not match, the server is requested for an update. This check is done after the connection to the database is established, but before the user starts working with the data. If the user has no Internet connection, of course the update can not be made, but an update could not be made without an Internet connection should not be a problem.

Code Tables

The look-up tables which contain mappings for values in different languages are named "Code Tables" in our synchronization. For example, in OSSOBOOK the name of the animals that are displayed in the graphical user interface are defined as Code Tables (cf. Table 3.3).

To be able to change or add values to the Code Tables without having to distribute a new program version, all values are stored in the database. To receive the newest version of the data only, the database has to be updated. This can also be done during the usage of the application. To find out which values are changed, all tables have to have the column "Status". Just like for entries, the value of the status is the timestamp of the last global change and is updated with triggers on insert and update (cf. Algorithm 3 and Chapter 3.1.3):

```
1 SET NEW.Status = NOW() + 0;
```

Algorithm 3: *SQL Query to update the Status of the Code Tables*

Then only the values that have a higher status than the highest value in the local table have to be updated or inserted. To avoid unnecessary update checks in each table if no

ID	language	value	LB	Status	Deleted
7	1	Mammalia > Rind/Hirsch	99	20170912163334	N
7	2	Mammalia > cattle/deer	99	20170912163334	N
7	3	Mammalia > boeuf/cerf	99	20170912163334	N
8	0	Vertebrata	0	20110118140546	N
14	0	Felix catus	16	20110118140546	N
15	0	Canis familiaris	4	20110118140546	N
16	0	Equidae	2	20111004103152	N
17	0	Equidae caballus	2	20110118140546	N
18	0	Equidae asinus	2	20110118140546	N
20	0	Sus domesticus	3	20110118140546	N
21	0	Ovis aries	1	20110118140546	N
22	1	Ovis aries (wahrscheinlich)	1	20170912163334	N
22	2	Ovis aries (probably)	1	20170912163334	N
22	3	Ovis aries (probable)	1	20170912163334	N
23	0	Capra hircus	1	20120719102831	N
26	0	Bos taurus	13	20110118140546	N
...	...	...	...	...	...

Table 3.3: *The Code Table "animal" in* OssoBook *that defines the available values for species. Adding the column "language" allows the use of terms in different languages:* 0 *for general terms,* 1 *for German,* 2 *for English,* 3 *for French, etc.*

change was made, the last update can not only be found out by run through all tables, but has to be saved directly in the "version" table. To ensure that this has always the newest timestamp on the server, the trigger is extended by another command to set the last update, so that the complete trigger now looks like as shown in Algorithm 4.

```
1 SET NEW.Status = NOW() + 0;
2 UPDATE version
3    SET version.LastUpdate = NOW() + 0;
```

Algorithm 4: *SQL Query to update the Status of the Code Tables*

The last update is set locally after all changes have been (successfully) made, and is retrieved from the server before the update progress is started. Then only a check has to be done, if the local value is lower than the global value, and only when, the Code Tables are out of date and therefore have to be updated.

Manager

To control the communication with the database from the client, we created manager classes that have all the knowledge about the columns for the table they are "responsible" for. The structure of the manager is as follows:

- **Table Manager:**
  The base class for all managers. It holds the connection object, contains the basic methods like insert and update and sends SQL queries to the database.

- **Abstract Synchronization Manager:**
  This manager holds all the important information for the synchronization. It handles the insert and update of entries from the server, retrieves uncommitted entries and sets data to synchronized or conflicted. All managers that need to be synchronized have to extend the *Abstract Synchronization Manager*.

- **Base Entry Manager:**
  The base entry manager is responsible for getting the main entry from the input unit (the main table for entries). It also calls the underlying *Extended Entry Manager*s to retrieve the data for the complete entry. This class also manages the saving, loading, and updating data for itself and forwards the call to the underlying methods.

- **Extended Entry Manager:**
  A manager for entries that extend a base entry that is needed for example if an entry can have more than one value of a specific type. So the list of values would be stored in a different table.

- **Base Project Manager:**
  The base project manager is responsible for getting the main entry for entries that are valid for the whole project (e.g. project information itself). It also calls the underlying *Extended Project Manager*s to retrieve the data for the complete entry. This class also manages the saving, loading, and updating data for itself and forwards the call to the underlying methods.

- **Extended Project Manager:**
  This manager is for entries that extend a base project entry. This manager is needed for example if an entry can have more than one value of a specific type. So the list of values would be stored in a different table.

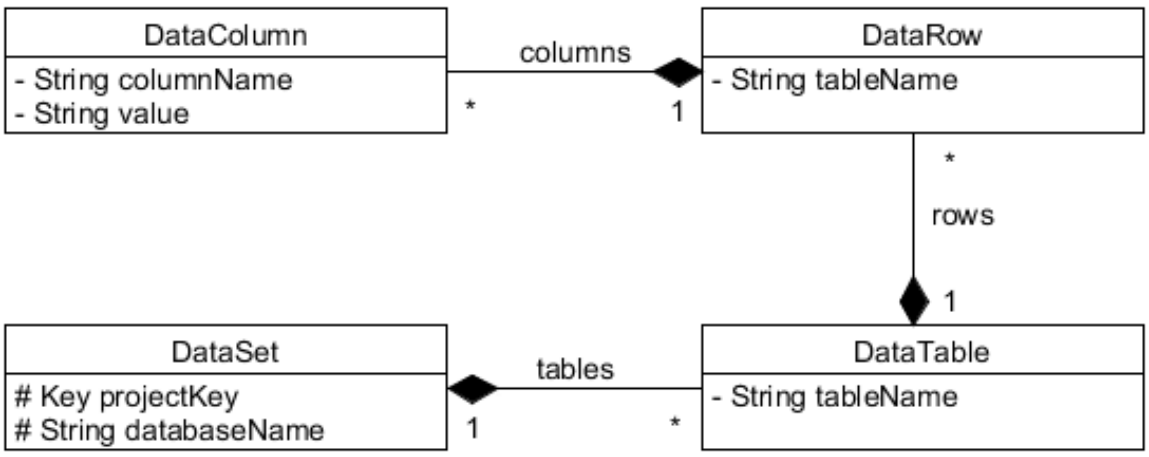


Figure 3.2: *UML visualization of the data types handling the data in* XBOOK.

Data Structure

To store the data and retrieve a complete entry we created some classes to easily load, save, and update the data. These data types are described below. The correlation between them is sketched in the UML diagram in Figure 3.2.

- **DataColumn:**
  The most basic data type is the DataColumn. In it only one value is stored together with its column name.

- **DataRow:**
  Represents one row in the database. It is an ArrayList of *DataColumn* containing all the data for this row.

- **DataTable:**
  Contains all *DataRow*s for the current entry in the specific table. In addition to the *DataRow* it only knows to which table it belongs.

- **DataSet:**
  Represents one entry. Therefore it has all *DataTable*s that define the entry, and additionally hold the key of the entry and the key of the project the entry belongs to.

Synchronization Process

The actual process of the synchronization consists of three different steps.

1. Check if the user has the required rights to access the project and therefore is allowed to synchronize it.

2. All uncommitted, not conflicted entries in project and entry tables are retrieved one by one from the database and sent to the server. This is done by iterating over all *Base Project* and *Base Entry Manager*s. These call their belonging sub managers, load all data belonging to the current entry and send their data to the server. If the entry already exists in the global database, then the server checks the sent timestamp of the entry against the timestamp of the entry which is already saved in the database. If the global timestamp is newer than the local one there is a conflict and the client is notified of it (cf. Chapter 3.1.5).

3. The project and entry data is transmitted from the server to the client. For identifying which entry has to be transmitted, the timestamp of the newest entry of the current table, i.e. the last entry that was synchronized, is transmitted. To prevent a loss of data after an incomplete synchronization, the first query requests entries that have exactly the same timestamp as the highest timestamp locally. For all later queries always the next data with a higher timestamp is retrieved. The entry is then only updated if the corresponding value in the local database either does not already exist or is not conflicted or changed (see Figure 3.3).

Local Removal and Backup possibility

Since the data is stored on the global database, the data can always be synchronized from the server to the client. Therefore it is possible to remove projects from the local databases. These are only deleted on the local machines, but not on the server. By removing local projects, the users can improve the overview over their projects, but also releasing unnecessary used memory space. The projects can be reloaded from the global database by using the synchronization at any time.

This architecture can also be used to back-up data. In case of a hard-disc crash, the data that was synchronized earlier to the global database can be restored at any time from another computer.

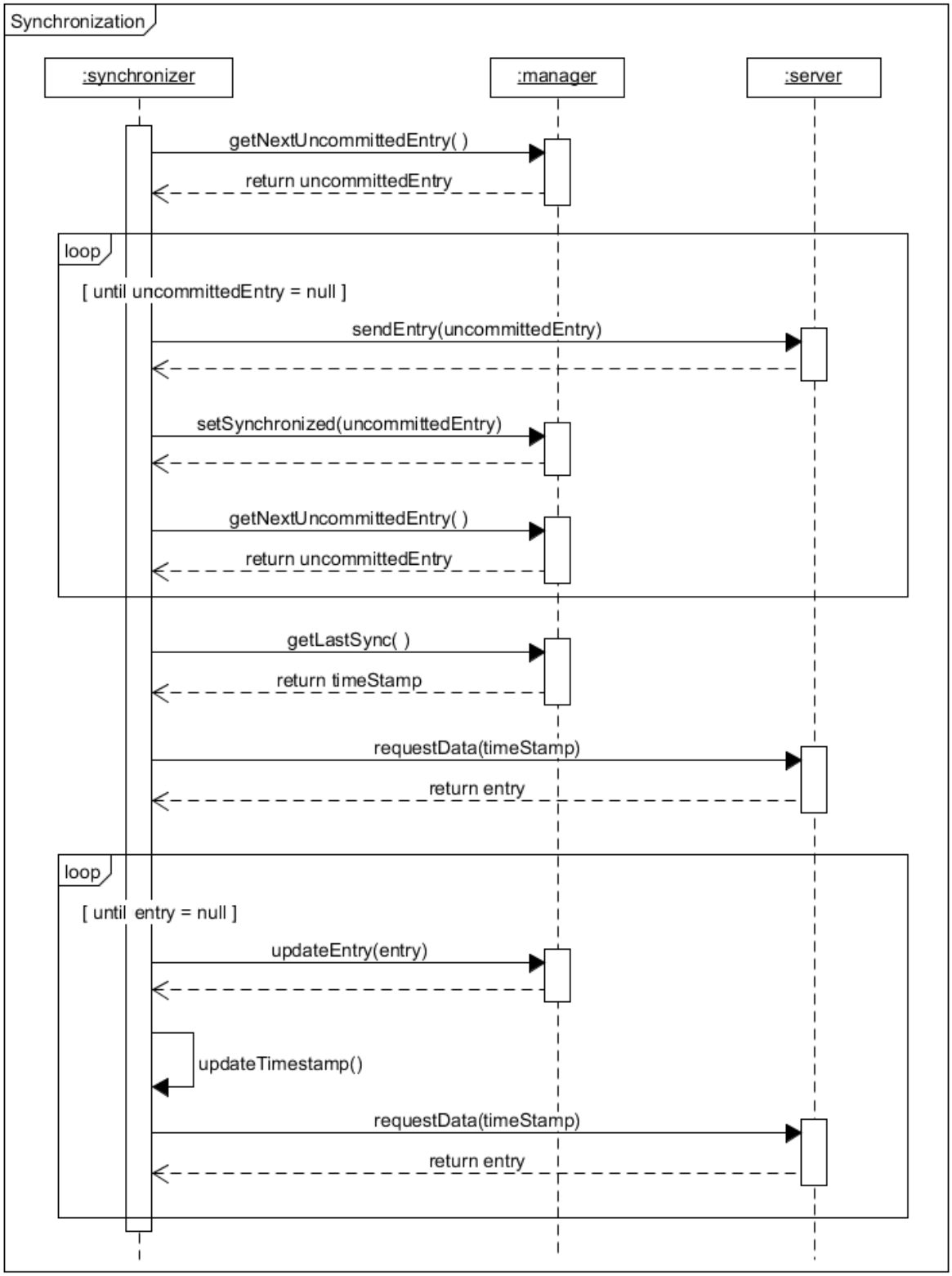


Figure 3.3: *Simplified visualization of the synchronization, displaying the commit of changed entries to the server and new data from the server.*

Figure 3.4: *The synchronization panel in* OssoBook *5.2.4. [LKK+16b]*

3.1.4 Graphical User Interface Integration

As presented in Chapter 3.1.3 the synchronization consists of a powerful, but complex architecture. The realization in the application has to consider that most of the archaeologists and bioarchaeologists are not used to work almost exclusively with a computer.

Therefore it is absolutely necessary to hide the complexity of the synchronization behind an intuitive user interface. We have to ensure that the synchronization can easily be used and that every user even if not technically skilled could execute it.

Here we describe how the synchronization is integrated into the graphical user interface of xBook:

Manual Data Synchronization

To exchange project data there are three basic procedures that have to be possible in the synchronization panel (cf. Figure 3.4).

- **Global projects** for which a user has read and/or write permission have to be downloadable from the server. Therefore the corresponding projects can be selected in the right project selection in the synchronization panel.

- **Local projects** that have not been synchronized with the server before have to be able to be uploaded to the server. These projects can be selected in the left project selection.

- **Existing projects** (as well local and global ones) have to be updateable. For this purpose the corresponding projects have to be selected, like explained above. However, the application recognizes if it was selected a project on the server project list that is also available on the local project list, and vice versa.

By pressing the "Synchronize" button the procedures are executed. Depending on the Internet speed and the number of projects and data sets (e.g. in OSSOBOOK exist projects with a six-digit number amount of entries), the synchronization may take several hours. Thus, user feedback is displayed in message boxes and progress bars (each one for general, project and data set layer).

When the procedures are running, the user can continue to work with the application. The synchronization is running in the background. It can also be interrupted by closing the application and continued at a later time.

Automatic Data Synchronization

The automatic data synchronization can be activated in the application settings. Thereby the project information and data sets are synchronized with the server automatically in the background. However, it is necessary to manually define once which projects shall be downloaded from the server. This is important to avoid that all projects are downloaded even if the user does not want to save them on the local database.

The automatic synchronization can be a big advantage when collaborating on one project in a group. Due to the fact that the automatic data synchronization is updating the data in the background, all members of the group have less unsynchronized data sets scattered in several local databases. Every collaborator immediately gets the current data and does not have to wait until the data from other collaborators is manually synchronized.

Another advantage of the automatic data synchronization is the possibility to use the synchronization as a backup method. In case of a hardware failure (in particular a hard disc crash) every synchronized data set can be restored by reloading the data from the server. When synchronizing the projects manually, there might be projects and/or data sets left that were not synchronized before, and therefore may get lost because of a hardware failure. Because the automatic data synchronization always commits data in the background the chances for a data loss are minimized.

Data sets that are conflicted will not be synchronized in general, thus the automatic data synchronization will not commit them as well. Conflicts have to be solved manually before the data is able to be synchronized to the global server.

Code Table Update

The code table update that was mentioned in Chapter 3.1.3 runs automatically by the application after the login if it is necessary. However the user can also reload all Code Tables manually by using the option in the "Tools" navigation element.

3.1.5 Conflict Management

During the synchronization process data conflicts may occur. One simple scenario for this case as an example: Two different users download data from the server by using the synchronization and update an identical data set. If user A uses the synchronization first to upload the changes to the server there will be no problem. If user B submit his changed data afterwards, it is not clear anymore whether the data of user A is the correct one or the data of user B. A conflict occurs that has to be solved manually. This example is also sketched in Figure 3.5.

If an entry was marked as conflicted during the synchronization process (that means, the "MessageID" of the entry was set to -1), the conflict has to be solved before the data can be merged with the data in the global database. Therefore, the application indicates the occurrence of conflicts at two places:

- The conflicted projects are highlighted with an yellow icon in the project overview screen. These are also displayed if the conflicted project is not loaded.
- Additionally, a yellow icon is displayed in the footer bar of the application if a conflicted project is loaded.

These icons highlight conflicted projects – i.e. the project contains at least one data set that is conflicted – to inform the user about current conflicts. Once the project is loaded, the user can click the conflict button in the footer bar to open the Conflict Management Screen (cf. Figure 3.6). The user gets displayed all current conflicts sorted by the table of the conflicted entry; e.g. on project information, entries, etc. Selecting a single conflict allows the user to solve the conflict in the corresponding Conflict Management Screen.

The Solve Conflict Screen (cf. Figure 3.7) lists all important information about the conflicted entry in a diff table – on the left, there are displayed the entry values that are currently saved in the local database. On the right, there are displayed the ones on the server. This listing includes all technical basic information to allow to definitely

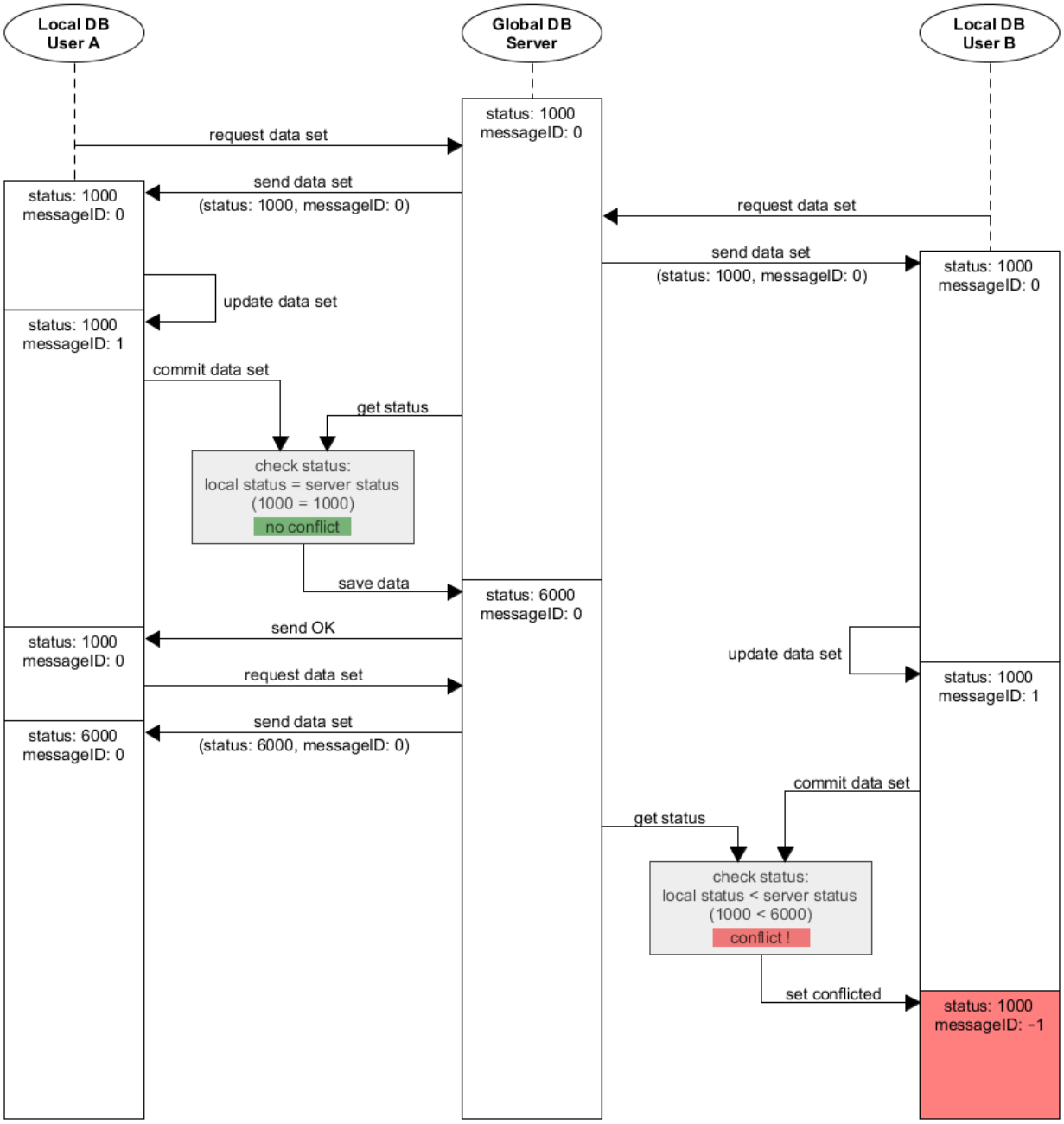


Figure 3.5: *A sketch how a conflict can occur on one data set. User A synchronizes an updated data set successfully. User B edits and synchronizes the same data set later which results in a conflict that has to be solved manually.*

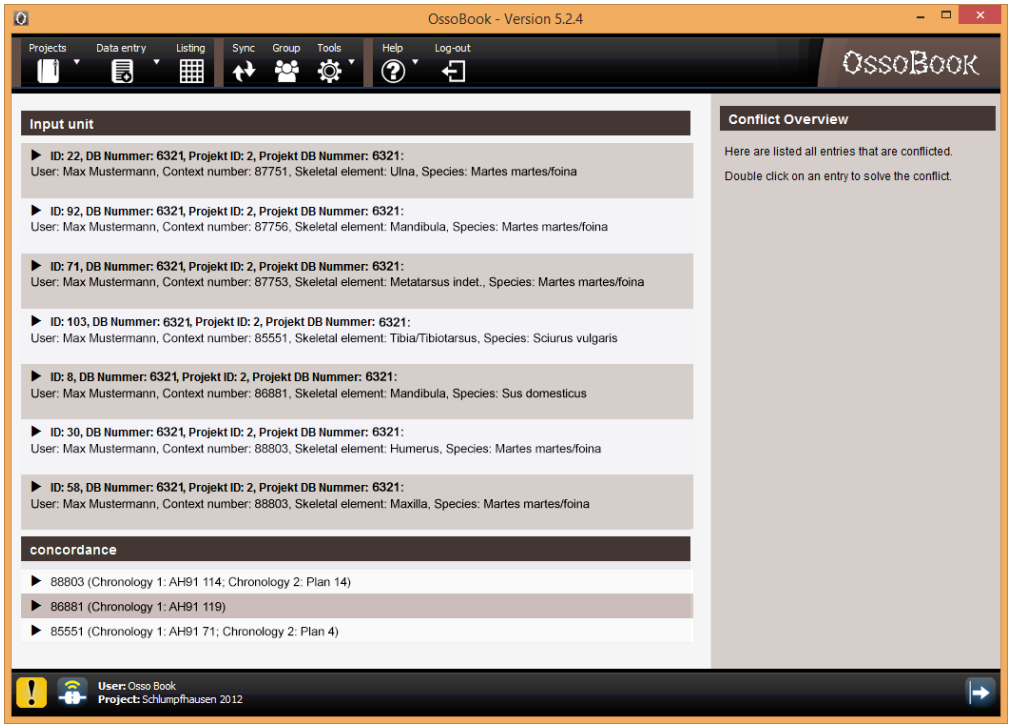


Figure 3.6: *The Conflict Management Screen that displays all conflicted entries of the loaded project.*

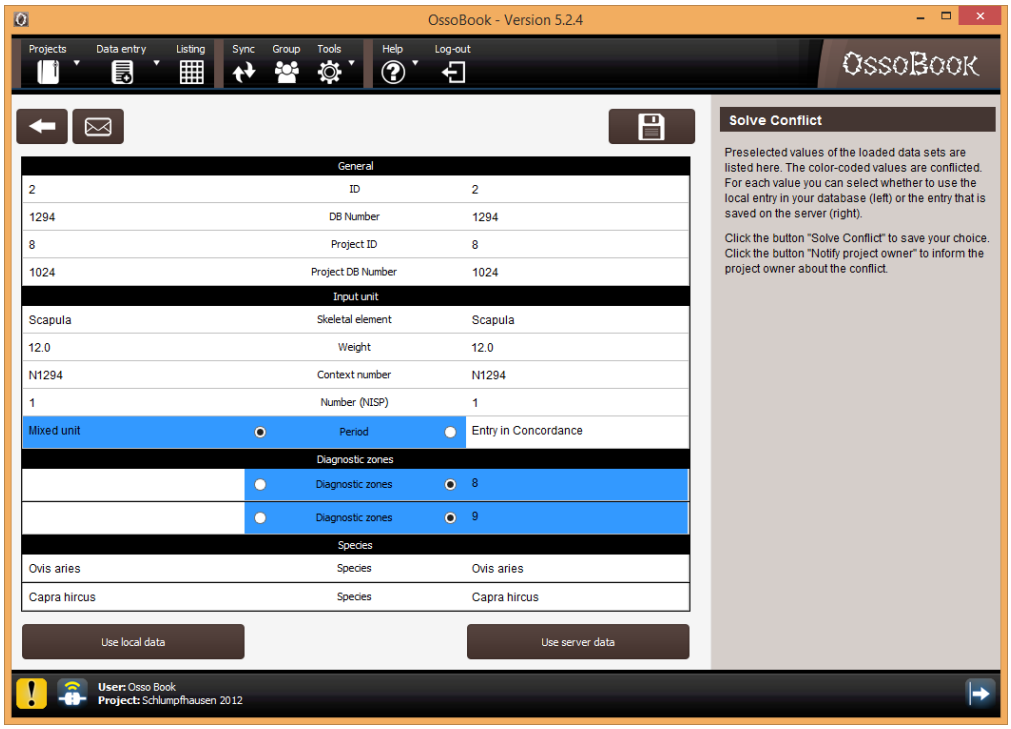


Figure 3.7: *The Solve Conflict Screen that allows selecting the local or server values for a specific conflicted entry.*

identify the data sets (like IDs and database Ids), some content-related information that are necessary to distinguish the unique entries for the scientists (most often mandatory fields), and all conflicted values. Other non-conflicted values, that are identical on the local and global database, are hidden.

To improve the usability, the displayed values are formatted to be human readable. IDs that reference to a specific Code Table value are replaced with the actual text value. Complex correlations can be individually handled, for example if several columns in the database are associated to one single input fields in the database. For example, in ARCHAEOBOOK [KLK+18e] the measurements of objects are described in a custom input field that records information about the measurement type, the measurement value, and the measurement type – although this information is associated, it is saved in separated columns in the database. A similar handling is required for the area/cut/planum/profile information in EXCABOOK [KLK+18f] where the information is saved in four different input fields (and therefore in four different colums in the database), but the combination of area/cut/planum/profile is handles as one value.

The users can select for each conflicted value whether to use the one of server or the one of the local database. An option to use all values of either the global or the local version is also possible. In case that the users do not know how to solve the conflict they can also click the "Inform Project Owner" button to let the project owner know about the conflict. The project owner will then receive an email about the conflicted entry to be able to support the scientist to solve the conflict.

To solve the conflict the merged entry is saved to the global database with the timestamp of the global entry. If the entry was updated between the solving of the entry and committing the entry to the global database – this ensures that this change will not be overridden, but a new conflict is generated. If no new conflict was generated, the local entry is updated with the merged values, but the local timestamp is not updated, and the entry is marked as synchronized (that means the "MessageID" of the entry is set to 0. Thereby, the conflict is solved in the local database and the application will not display that the entry is conflicted anymore. Now, the entry on the server has a newer timestamp than the same entry on the local database. That is the reason why the data set is updated in the local database during the next execution of the synchronization. The full conflict handling process is sketched in Figure 3.8.

Conflicts do not have to be solved immediately during or after the synchronization. It is possible to continue working on the projects even if there are still existing any conflicts in the project. This makes it also possible to inform the project owner about existing conflicts if the users are not able to solve them on their own. However, conflicted entries keep the "MessageID" -1 as long as the conflict is not solved and therefore will not be synchronized to the global server.

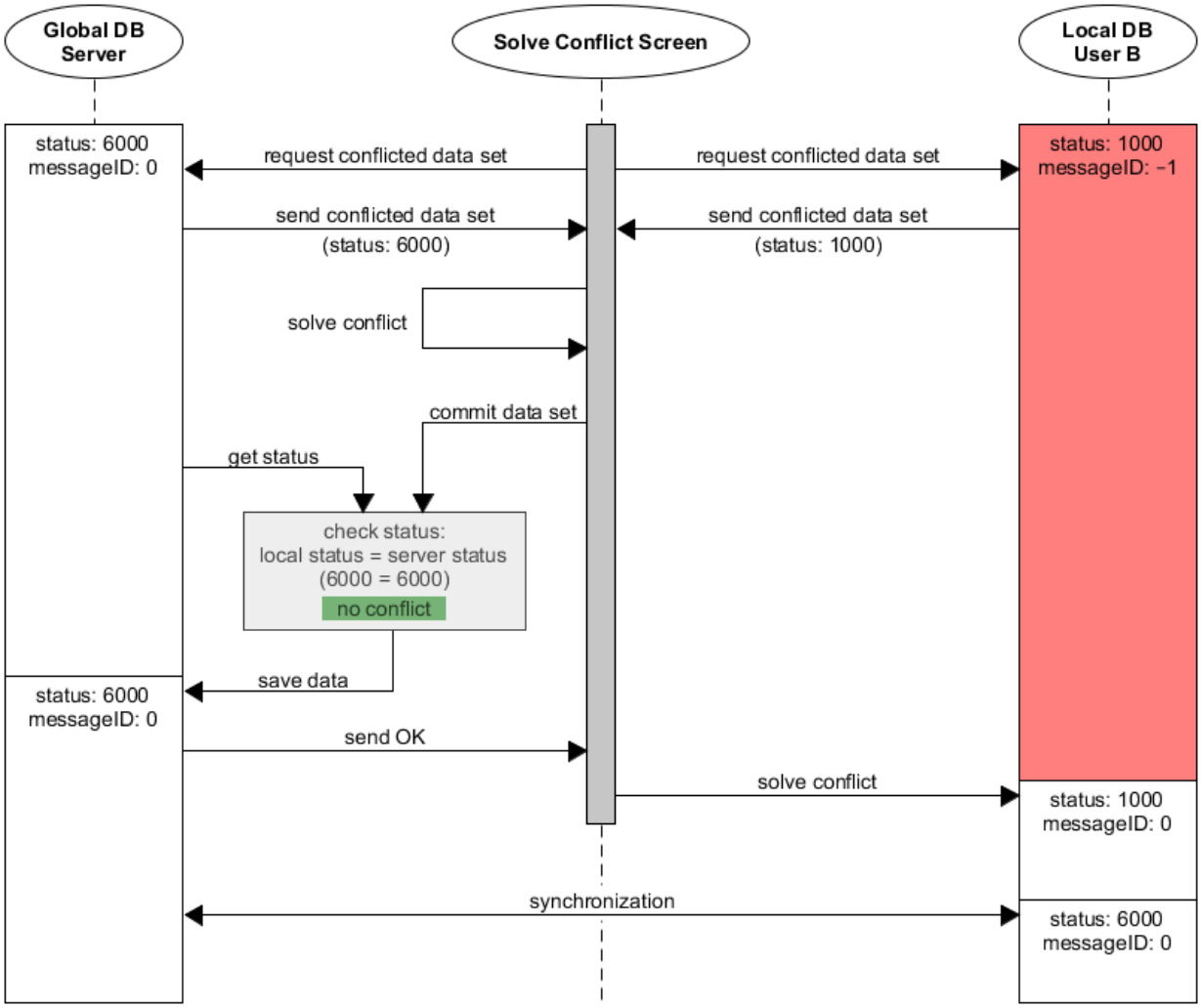


Figure 3.8: *A sketch how a conflict of one data set is solved, based on the situation of Figure 3.6.*

3.2 Retrieval of Distributed and Anonymous Archaeological Information

Conventional scientific research on prehistoric, historic, and sub-recent human legacies is done in the context of archaeological excavations. The unearthing is documented by archaeologists and bioarchaeologists in detail. Specialists from different areas of expertise like archaeologists, zooarchaeologists, anthropologists, archaeobotanists, geographers, etc. describe, categorize, and analyze the findings and produce a huge amount of data with spatial and temporal information. Analyzing all this data shall reconstruct the cultural heritage, for example the environment or the human way of life in ancient years.

Mostly, archaeological excavations in our dense populated region are so-called rescue excavations initiated by building activities. Typically, authorities for the protection of ancient monuments control these activities. In case of signs of human made changes of the surface of the bulldozed earth they initialize archaeological excavations. These authorities may be institutions of the federal state, the town, or homeland societies (hereinafter referred to as "offices"). Another possibility that leads to an actual archaeological excavation may be the continuance of an excavation started in former times or the knowledge of a (pre)historic landmark wants to be investigated in that way.

Depending on staff, financial possibilities, and level of digital development, data from these excavations is collected and stored in databases by these institutions. The data usually consists of all kinds of general archaeological information like geo-spatial information, often temporal dating on the basis of specific findings, and a rough description of the found objects (in terms of text, photos, and even 3D scans).

Afterwards, the movable findings are transferred to specialized collections or specialists like anthropologists for human remains, zooarchaeologists for animal bones, archaeobotanists for rests of plants, and archaeologists for objects and artifacts. The findings are packed separately in units according to the archaeologically defined context which can be the place inside the grid of the archaeological area.

The specialists execute corresponding investigations specific to each research field. Typically, this information is collected decentrally in subject-specific databases. The archaeological data and the data collected by the specialists build the basis for detailed analyses. The flow of archaeological and bioarchaeological data is sketched in Figure 3.9.

The problem we are dealing with is that the specialized collections and specialists do not have all data that is available about a specific finding. This is a severe limitation for their research since a comprehensive analysis can only be done when considering the entire context of a finding. Sometimes, even temporal and spatial data is not available because this information is only saved completely in the databases of the offices and archives by default.

Though this data can be retrieved by the archaeologists and bioarchaeologists by manual requests, the retrieved data is not saved in the data scheme of the corresponding database system. Usually this data is sent in Excel sheets or CSV files, as well as scanned pages and images. An automatic query system for the data exchange does not exist, so each request is aggravating, time-consuming, and needs the work of a reviser that manages the requests.

Archaeo-related data is diverse and complex, most often spatial information is enriched with temporal information and additional data on the findings, like isotopic fingerprints, morphological measures, etc. Analyzing this data provides challenges and interesting topics for data scientists in different fields. Archaeologists and bioarchaeologists would greatly benefit from query processing interfaces and analysis methods working

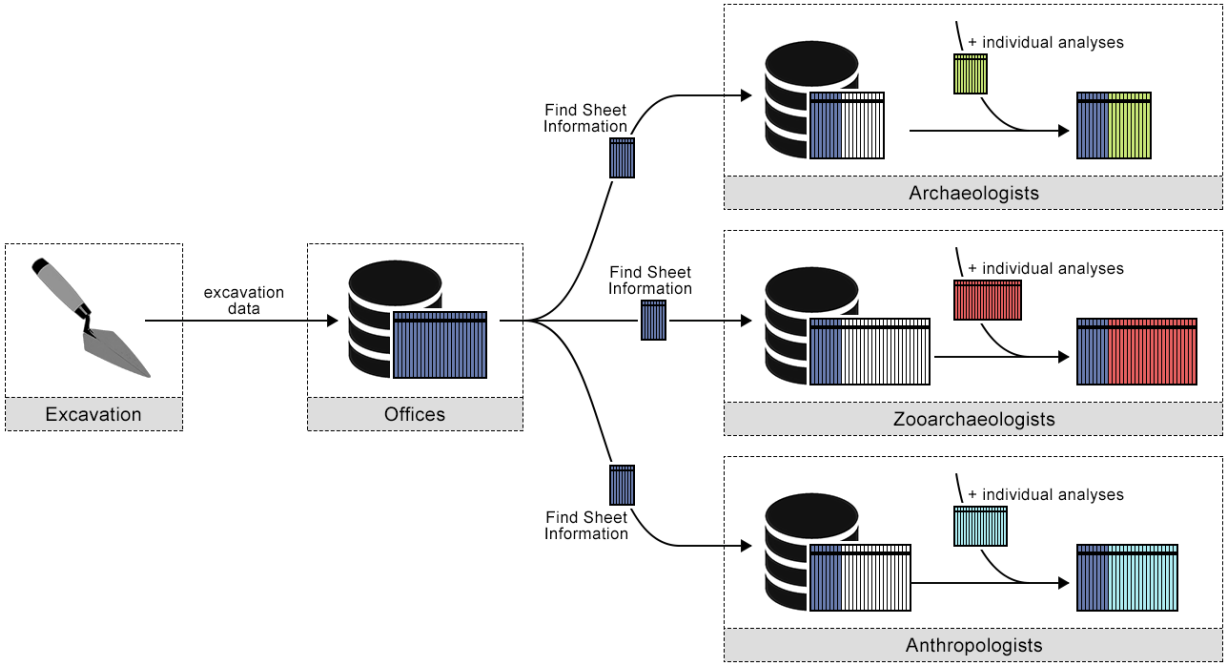


Figure 3.9: *The basic flow of archaeological data: The data from the excavation is partly passed from the offices to the specialized collections and specialists, who perform individual analyses on the findings and save the results in their databases. The results of these analyses are not accessible from the offices. In sum, neither the offices nor the specialized collections have all information about their findings.*

with such kind of complex data. Probably the highest hurdle towards this digital revolution of archaeo-related research is simply the technical gap described above.

We close this gap by proposing a *Reverse-Mediated Information System* (abbreviated with REMIS) that is based on the concept of common Mediator-based systems. It enables the provision of new data components without the need of a central administrator to manage the connections manually. It enables the users to search for more information about specific excavation contexts or findings that is spread through databases of offices, collections, and institutes.

In summary, the main contributions of this Chapter are as follows: First we discuss the challenges of the current situation of archaeo-related sciences. We list a set of requirements that should be addressed by an architecture for distributed data management of anonymous data sources (cf. Chapter 3.2.2) and discuss existing solutions and approaches for these systems (cf. Chapter 3.2.5). Then we describe the concept of our architecture, including the process of the initialization, registration, and the user search (cf. Chapter 3.2.3), and describe the concept of a compact right management for the data (cf. Chapter 3.2.3). Afterwards, we explain the necessary configuration for administrators of data sources (cf. Chapter 3.2.4).

3.2.1 Problem Formulation

For archaeological and bioarchaeological analyses it is not only interesting to analyze data from a single archaeo-related sub-domain. Considering the data from other sub-domains and the basic context information data from the offices would enable more complex and more comprehensive analysis than currently possible.

In principle, enriched geodata is available, but in practice, it is not easily accessible for every scientist because of the following reasons.

- **No direct access:**
  There is no direct external access available to data of most of the specialized databases, as sketched in Figure 3.10. Data has to be gathered from the offices manually. The composition of this data occurs requesting the so-called 'find sheet number' or – if not existing – other unique terms given by the excavator. Obviously, a more detailed data selection with complex search parameters is difficult to apply without any direct access.

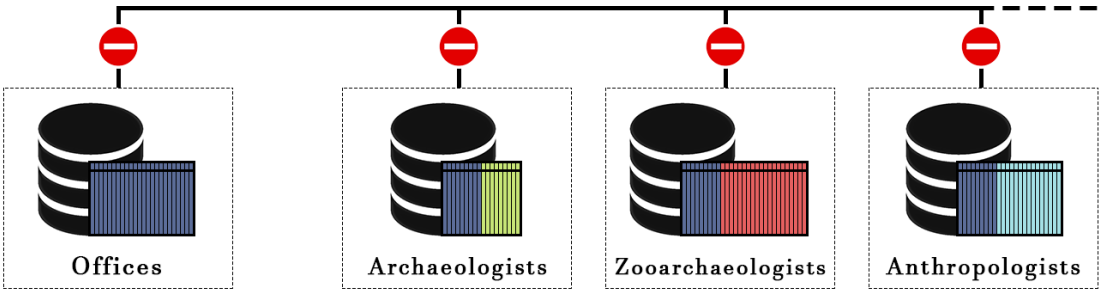


Figure 3.10: *The digital data exchange is hindered. Offices and specialized collections and specialists do each have no digital access to the detailed or individual data of other databases.*

- **Anonymous databases:**
  Data is spread over databases that – in some circumstances – are not known at all. Archaeo-related information data is not always gathered by state offices. There are also numerous city offices, local institutes, and freelancers that are gathering data as well, but do not exchange their data with the state offices. Still it would be interesting to be able to include this data in analyses, even if the archaeologist or bioarchaeologist does not know that there exists any data in these data sources.

- **Individual database schemes:**
  The databases of the offices, the specialized collections, and self-employed workers have their own architectures and individual database schemes. Although they usually have identical basic information, these are saved in different ways, and cannot be standardized to a commonly used database scheme.

- **No consistent way to exchange data:**
  As it is common practice, data is exchanged by using Excel or CSV files that have to be formatted before being able to use them for analyses. This method can be aggravating, time-consuming, and error-prone, but it is still common practice.

- **Regional and individual approaches:**
  There are regional and individual approaches for the filing of findings. They do not only vary in the way they are excavated, but also how and which data is recorded in the databases. This differs from country to country and even from state to state which is extremely frustrating for researchers that deal with cross-border excavations. Furthermore, there are numerous individual solutions even on a local level.

Especially considering the geodata would enable users to ask much more complex questions than it is currently possible. Therefore, we want to describe a solution to bring spread data together again and to achieve this without the need of a single central database, but by keeping the necessary system of distributed databases.

3.2.2 Requirements for Retrieving Distributed Data

In this Chapter, we define a set of primary requirements that should be considered for our solution. This is described below.

Our architecture has to dynamically connect new data sources that are heterogeneously distributed. Each of them can have their own data scheme and contain different types of data. Additionally, the physical location of the data is not known in advance, neither by the users nor the responsible administrators of the architecture. Therefore, it has to be possible to connect to data without having to update the information of the central server. We state these requirements as R1 and R2:

R1 *Heterogeneous Data Sources.*

R2 *Anonymous Data Sources.*

External data sources have to dynamically be connected and added to the architecture. This process shall be performed individually by the owners of the data source. Therefore, no central administrator should be required to connect a data source to the existing information system. The access to the data by third-party users should be controlled directly by the data owners – i.e. they have also to be able to remove their data source from the distributed database system at any time. We state this requirement as R3:

R3 *No central administrator.*
Administration directly by data owners.

The users have to be able to search through all connected data sources independently of their underlying database scheme. The database scheme has to not influence what the users can search, but instead map the search parameters to the local database scheme. We state this requirement as R4:

R4 *Database Scheme Independent Search.*

Finally, data sources can contain sensitive information that is not intended for everyone. Still, data sources may contain searchable data. Therefore, the architecture has to provide the possibility to restrict the access to sensible information or data that shall not be made public (yet). We state this requirement as R5:

R5 *Rights Management.*

Of course, users have to be able to retrieve information from the distributed data sources from any local machine. This may be a web interface or an embedded solution within an existing application.

Let us add that the data of the distributed databases is not intended to be edited by the users. This infrastructure is solely aimed for retrieving data.

In the following Chapter we describe our novel information retrieval system which respects the defined requirements stated above.

3.2.3 Reverse-Mediated Information System Method

In this Chapter we describe the basic concept of the *Reverse-Mediated Information System* to achieve a search through different, anonymous databases.

We consider the concept of the 'Internet of Databases' that keeps the individual scientific data sources which "are for the scientists more interesting. [...] They have their database developed in their own way. They are modular, so that every user can connect and use tools in the way they want" [vdM12] and they are used to. These databases can be connected with a simple application where meta information is managed. Considering the concept of the Internet of Databases we meet our requirements described in the previous Chapter 3.2.2.

Therefore, we distinguish the three different layers of the architecture as sketched in Figure 3.11:

- **User Layer:**
  The users who search for specific data and want to retrieve the information corresponding to the entered parameters. This can be a web service, but also a search mask embedded to an existing application.

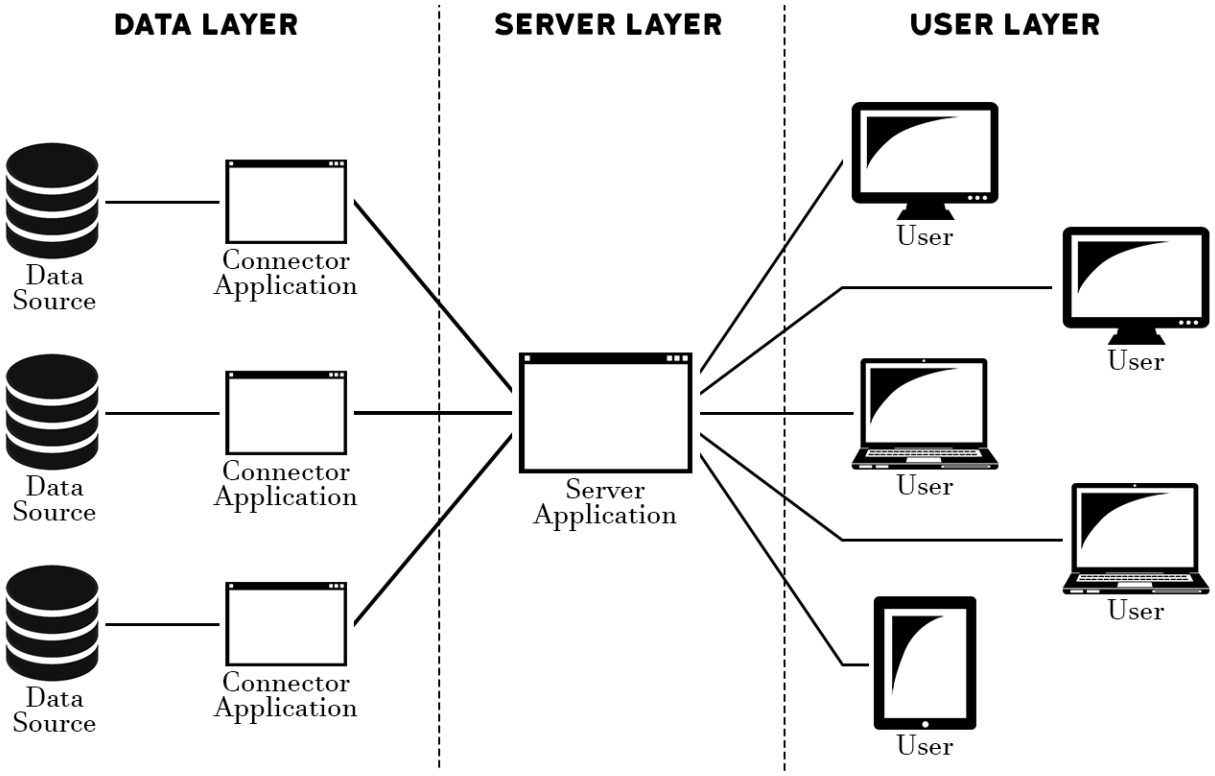


Figure 3.11: *The three layers of the ReMIS architecture: Data Layer, Server Layer, and User Layer.*

- **Server Layer:**
  Runs a stand-alone Mediator-like Server Application which accepts requests from the users and relays these to the connected databases. In addition, it receives the results from the connected databases and sends them back to the users. New databases can register themselves to the Server Application.

- **Data Layer:**
  The servers where the data sources, mostly databases, are saved. Each data source runs an independent Connector Application that is configured for the database it is connected to. This Connector Application accepts forwarded requests from the Server Layer and translates the request to a query to fetch the data from the data source. Once the data is fetched, it transforms the data to a uniformed data structure which will be sent back to the Server Application of the Server Layer.

In general, a Mediator provides data from a number of heterogeneous data sources with other users. “The autonomy of the participants enables the overall system to grow, since new sources [...] can be inserted.” [Wie94] In this context, the actual data structure of the connected sources is irrelevant:

> *"Mediation recognizes the autonomy and diversity of the data systems and information services that support the hubs, and the user applications that utilize them. The autonomy of the participants enables the overall system to grow, since new sources, new means of transport, and novel information processes can be inserted."* [Wie94]

However, a central administrator is still necessary to manage the participating data sources. Therefore, an extern connection of a data source is not possible without the integration work of the administrator, as illustrated in Figure 3.12.

The *Reverse-Mediated Information System* swaps this approach to achieve an open system for shareable data. The architecture of this system is illustrated in Figure 3.13. Data sources can be connected to the system to share data with other users, independent of the actual data type.

The usage of the *Reverse-Mediated Information System* can be categorized in three different steps to describe the workflow: The initialization, the registration, and the search. These are described in detail below.

Initialization

To be able to connect the data of a specific data source, an initialization process has to be executed for the data source in the Data Layer. The administrator of the server, where the data source is located, is guided by a setup assistant. In general, the required steps in the initialization process can depend on the type of the connected data source. Hereinafter, we use a relational database as an example for a data source. A sequence diagram of the initialization process is shown in Figure 3.14.

First, it is necessary to connect the Connector Application to the database in order to be able to access the data. Once this is done, the administrator can select the tables and columns to define which data will be sharable in general. This step determines which parts of the data are private and only visible within the database itself and which data might be transmitted to the users later.

Optionally, individual rights can be defined that specify the visibility of the data for the users. These are discussed in Chapter 3.2.3.

Finally, the administrator has to specify a set of contact information. That allows either contacting the data owners for questions or to request additional information of entries that cannot be accessed due to rights settings.

Registration

To become a part of the *Reverse-Mediated Information System*, the initialized Connector Application has to be registered to the Server Application in the Server Layer. The sequence diagram of the registration process is shown in Figure 3.15.

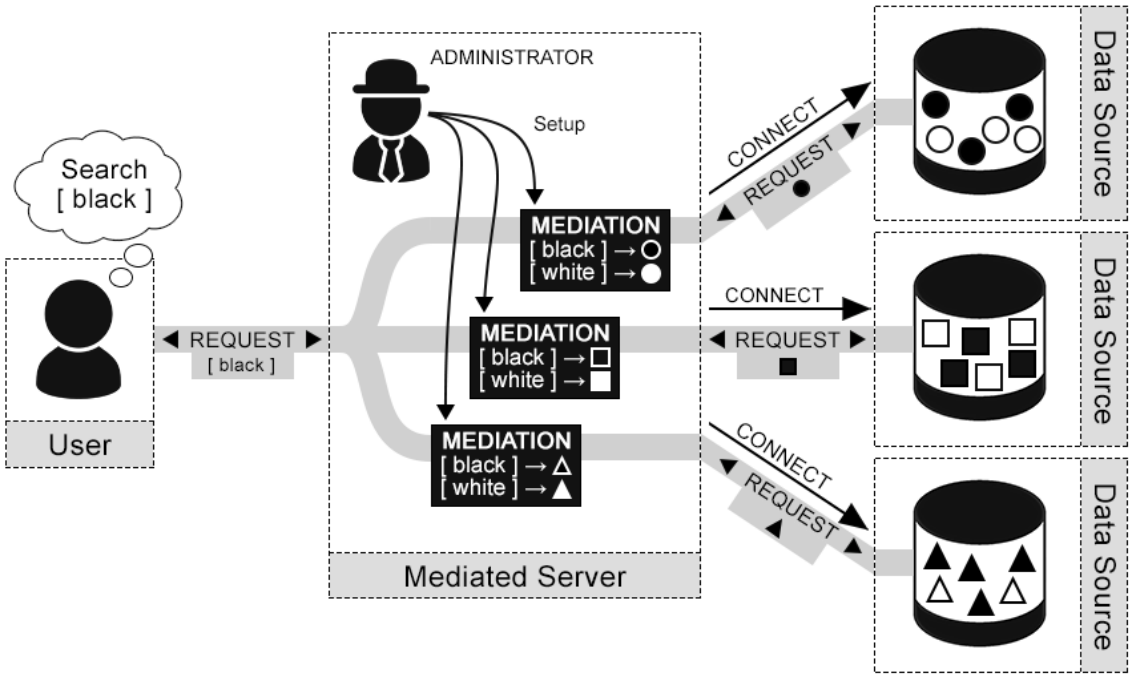


Figure 3.12: *Sketch of the well-known Mediator-based architecture. A central administrator is required to connect the data sources and to mediate the requests from the user. The administrator has to know each data source to be able to connect them.*

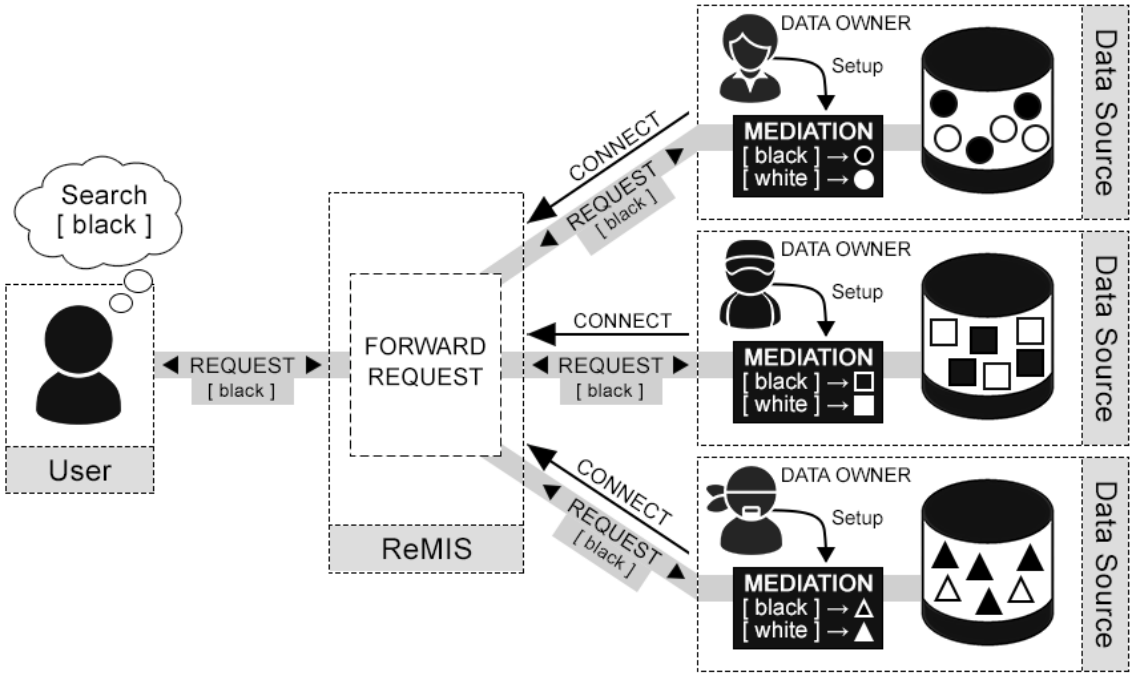


Figure 3.13: *Sketch of the Reverse-Mediated Information System. The data owners can register their databases to the system on their own. The necessary mediation setup is executed by a wizard dialog. The architecture forwards the user request to the data sources where the request is mediated.*

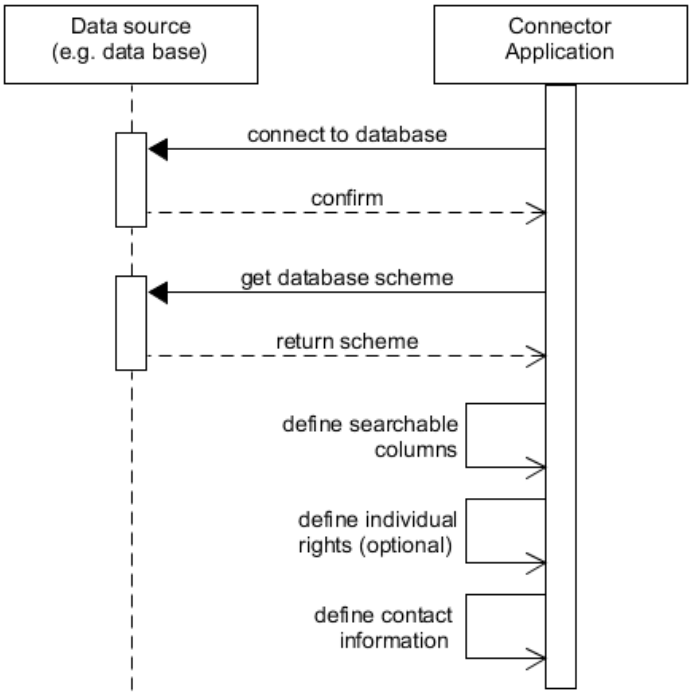


Figure 3.14: *Sequence diagram of the initialization process of* REMIS.

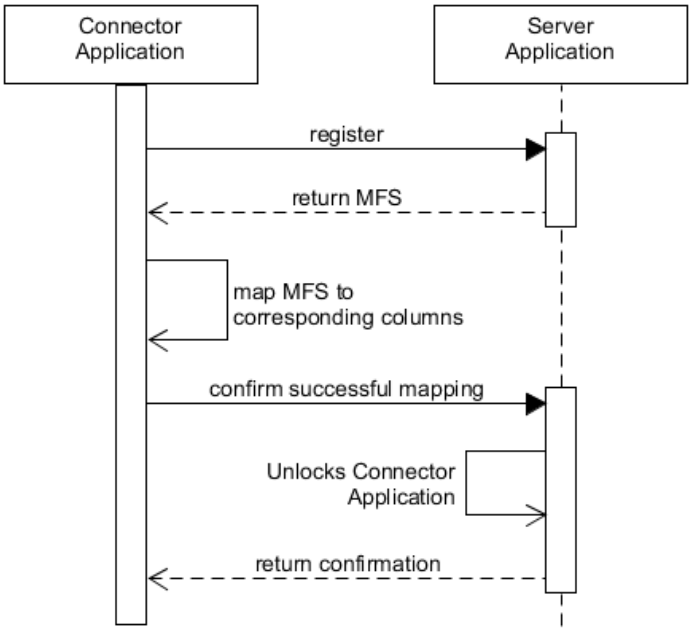


Figure 3.15: *Sequence diagram of the registration process of* REMIS.

First, a "register" command is sent to the Server Application. The Server Application then sends the "Minimal Find Sheet", abbreviated with *MFS*, to the Connector Application, a set of minimal information that are required. These are as minimal as possible and guarantee that every connected archaeological data source has saved any information about each of the parameters. These parameters p are the information that are given from the offices to the specialized collections and specialists for each finding, e.g. the excavation number, and the find sheet number. The Minimal Find Sheet also forms the options the user can search for.

$$MFS = \{p_1, \ldots, p_n\},\ n = |MFS| \tag{3.1}$$

In general, the parameters p of the Minimal Find Sheet are defined in the Server Application which has access to all connected Connector Applications and their databases. Each database D can have an individual database scheme with its own columns, but every parameter of the Minimal Find Sheet has to be mapped to a specific column in the database. Each database consists of a set of columns c.

$$D = \bigcup_{i=1}^{m_D} \{c_i\},\ m_D \geq n \tag{3.2}$$

For the registration process, the parameters are sent to the corresponding Connector Application. Then, the administrator has to map each of the parameters p of the Minimal Find Sheet to the corresponding columns in the database. To describe this mapping, we define the injective function κ_D that stands for a projection of the parameters of MFS to the corresponding columns in the database D.

$$\kappa_D : MFS \rightarrow D \tag{3.3}$$

We define r as the user search request. These may have different appearances, for example tuples containing the parameters p and the corresponding values v of the user inputs.

$$r = (r_1, \ldots, r_k),\ r_j = (p_j, v_j),\ j \leq n \tag{3.4}$$

Furthermore, we define $\tilde{\kappa}_D$, a replacement function as canonical extension over the query language, while all search parameters $r \in MFS$ are mapped according to κ_D and the remaining parts stay unchanged. Let $\epsilon = ()$ the empty search request.

$$\begin{aligned} \tilde{\kappa}_D(\epsilon) &:= \epsilon \\ \tilde{\kappa}_D(r \neq \epsilon) &:= \begin{cases} \kappa_D(r_1) \circ \tilde{\kappa}_D((r_2, \ldots, r_k)), & r_1 \in MFS \\ r_1 \circ \tilde{\kappa}_D((r_2, \ldots, r_k)), & r_1 \notin MFS \end{cases} \end{aligned} \tag{3.5}$$

Finally, r_D is yield by applying κ_D as a replacement function to r:

$$r_D = \tilde{\kappa}(r) \tag{3.6}$$

As soon as the mapping of all Minimal Find Sheet values is complete, the Connector Application informs the Server Application that it is ready to accept requests by the user. To complete the registration process, the Server Application set the Connector Application to be ready to be searched.

Search

The existing federated systems are sufficient if the users need to query data from a fixed set of (internal) databases, like sale statistics or customer information. But now, we want to provide the users the option to search for information they might not even know that it exists.

For this, we define the Minimal Find Sheet being the set of information the user can search for by default. All connected Connector Applications know how to handle these information.

Certainly, the right settings set during the initialization process has to also be considered. For reasons of simplicity, we do not discuss the implementation of the rights management in detail in this Chapter. The process of fetching data from the databases is figured in the sequence diagram in Figure 3.16.

The search enables the retrieval of entries that include values which are defined as the Minimal Find Sheet. Since this Minimal Find Sheet information is saved on the Server Application, the Server Application is first queried to get the Minimal Find Sheet MFS to be able to display its information in a search mask for the users. This is necessary to avoid input masks that are embedded to any applications getting incompatible when the definition of the Minimal Find Sheet changes.

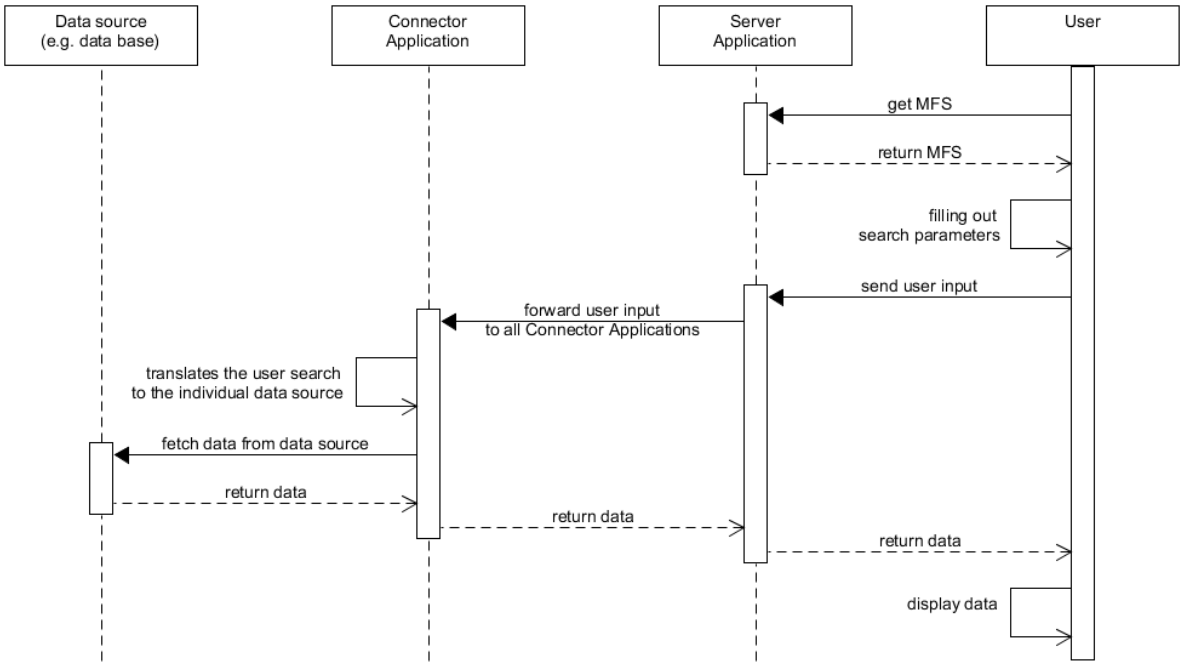


Figure 3.16: *Sequence diagram of the process to fetch data from the single data sources in* REMIS.

The users enter the data and define which parameters $MFS*$ they want to use for their search. For these applies:

$$MFS* \subseteq MFS \tag{3.7}$$

The parameters $MFS*$ and their inputs are sent to the Server Application that forwards the request to all N connected Connector Applications. Each Connector Application translates the incoming parameters to the local scheme of their corresponding databases, considering the given mapping κ_D. Each Connector Application queries the database with the modified search request r_D and checks if the search request of a user (this means, the inputs of the parameters $MFS*$) matches with the corresponding columns. Let $R(D, r)$ the result of this request.

$$R(D, r) = \sigma_{\tilde{\kappa}_D(r)}(D) \tag{3.8}$$

The entries that have been found in the database are then returned back to the Server Application. Finally, the user gets the merged data σ_r^{Σ} from the Server Application considering the search request r:

$$\sigma_r^{\Sigma} = \bigcup_{i=1}^{N} R(D_i, r) \tag{3.9}$$

The responds from the single Connector Applications are not bundled within the Server Application, but are sent directly to the users. This improved the necessary waiting time for the users, especially if a database has a slow connectivity or currently no connectivity at all.

Rights

Since we want to encourage database owners to provide their data, we have to provide a right management for the architecture. Scientists want to be able to manage the visibility of their data, not only because data of unpublished research often should not be shared until the time of the publication, but also because parts of the data has not to be shared at all, like sensitive information.

Therefore, we provide the option to limit the visibility of the data (or parts of it). The first step for the right management was already defined during the initialization process, as described in Chapter 3.2.3. In this step the data owner defines which data (tables and columns) is set as private and is not accessible for the corresponding Connector Application at all.

As well, the data owner can set individual rights that specifies the visibility and searchability of the data for the users. Therefore, we distinguish five different types of rights that can be applied to the entries. These specify the level of detail of the returned values that include the range from non-restricted to most restrictive.

- **Full-Public:**
  The default right which is used if no restrictions were applied to the entry during the initialization process. All columns of the entries are returned that are defined in the initialization process.

- **Partially-Public:**
  Limits the returned columns according to criteria specific to the data set. Still, all columns can be searched by the users, but not all of them are returned as a result. This can be achieved by excluding single columns from the visibility, but can also

be a more complex query, like checking whether a specific value is set to a column or not.

- **Numeric-Public:**
  Choosing this type makes the entries searchable, but only the number of matching enries is returned. The details of the entries are not shown to the users.

- **Hidden:**
  To reduce the visibility even more the entries can be hidden. A search request would only return a boolean value regardless if any matching data was found. The details, how many data sets are found, and any further information are hidden from the users.

- **Private:**
  A search returns no results. All data from the database is completely private.

In general, these rights can be set on single data sources like tables in databases. But if the data source has saved rights information within the data sets, these can also be used to define entry-specific rights. Therefore, the data of the result set can be put together by considering different rights.

In any case, the data owner can be contacted by the users, e.g. for further questions or a request for a higher access level on the data set. This is especially interesting for the Numeric-Public and Hidden rights. Therefore, the contact information, defined during the initialization process, is used.

3.2.4 Connector Application Configuration

Since the configuration of the Connector Application is the most important and responsible task of the *Reverse-Mediated Information System*, we describe the practical configuration process in more detail. Here, the owner of the data source defines the settings for the data and the server, and for the basic privacy settings for the data. Therefore, administrators are led by a wizard dialog to setup the application.

First, the data owner has to select the type of the data source. Currently MySQL databases and Excel files are supported. The Connector Application can be extended by other types to be supported in future. In the next step, the data owner has to establish a connection to the data. This depends on the selected type of data source. In case of a MySQL database, the data owner has to enter connection details to the database, like database name, port, user, and password. Then, the Connector Application connects to the desired database and the available tables are shown in the setup dialog. In case of Excel tables, the data owner has to select the desired files, where the data is stored, in a file dialog.

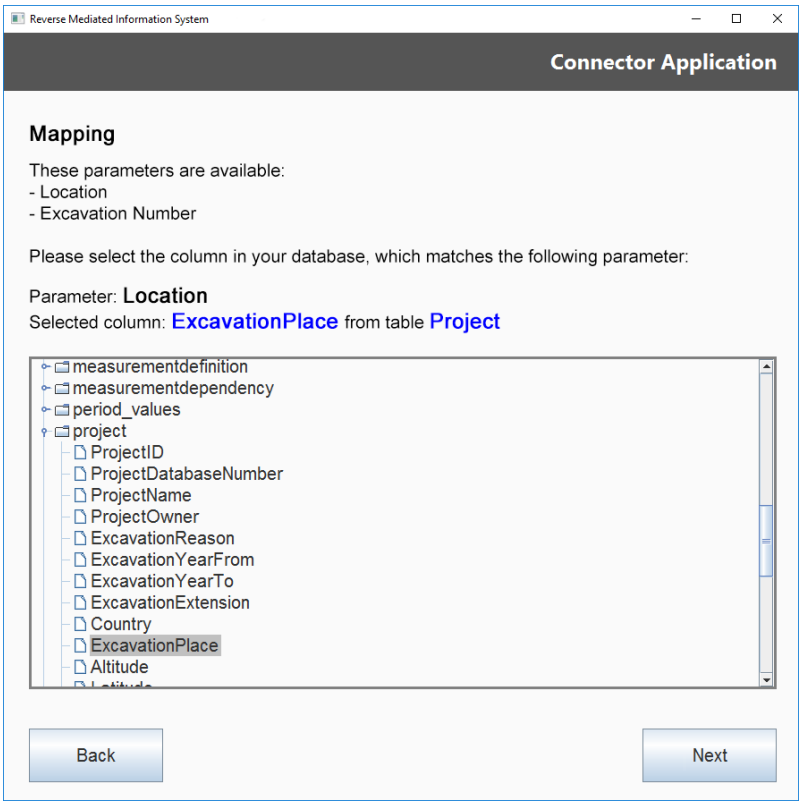


Figure 3.17: *Screenshot of the wizard dialog where administrators can map the specified parameters of the Minimal Find Sheet to the actual data of their database.*

Then, the Connector Application retrieves the Minimal Find Sheet from the Server Application. Usually, the column names in the data sources are not identical to the corresponding parameters of the Minimal Find Sheet – reasons for that might be a varying naming of the columns or different languages. Therefore, the data owner has to map each parameter of the Minimal Find Sheet to a specific column in the database, respectively the Excel file. A screenshot of this dialog is shown in Figure 3.17. The mappings ("Minimal Find Sheet parameter" $\rightarrow$ "table name" / "column name") are saved in a XML configuration file on the server. On an incoming search request, these mappings are used to translate the Minimal Find Sheet parameters to the corresponding column information.

The next step in the dialog is a listing of all available columns, where the data owner can select all those columns which data should be displayed to the users. This screen is shown in Figure 3.18. Only selected columns are considered when composing the result set. Though, the data owner can use this screen to set specific parts of data to private which will be communicated neither to any user nor to the Server Application.

If data is not saved in one single table, e.g. by using IDs and value tables, it is necessary to define dependencies between tables within one database. The data owner

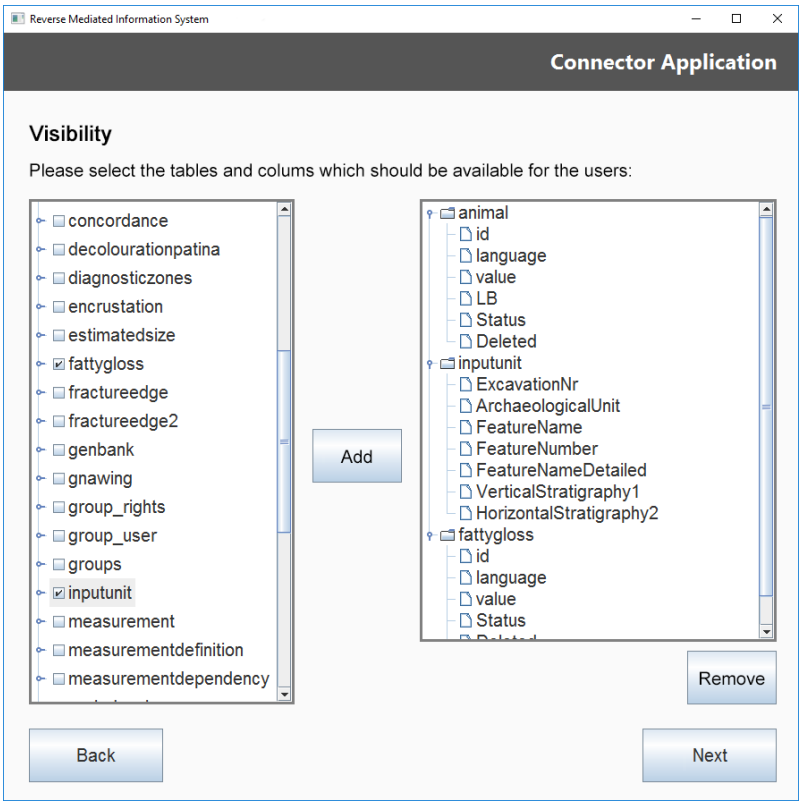


Figure 3.18: *Screenshot of the wizard dialog where administrators can determine the columns from the data source which should be communicated and transferred to the user on a request.*

should define foreign keys for each table. Multiple foreign keys are also allowed. The screen of these dependencies is shown in Figure 3.19. The settings for the result sets and the dependencies are both saved in the XML configuration file together with the mapping information of the Minimal Find Sheet parameters.

3.2.5 Related Work and Comparison

In the traditional database search, information is stored in single databases that are managed by Database Management Systems (DBMS) – but these do not meet our requirements R1, R2, R3, and R4. In general, the idea of merging data from numerous (physical or virtual) databases, is not new [OV11]. In the following, we recapitulate existing approaches for such infrastructures in the context and compare them with our requirements derived in Chapter 3.2.2.

- **Data Warehouses:**
  Data is stored in different databases to be able to store more and bigger amounts of data, or to separate logical information. Each database may hold different sets of

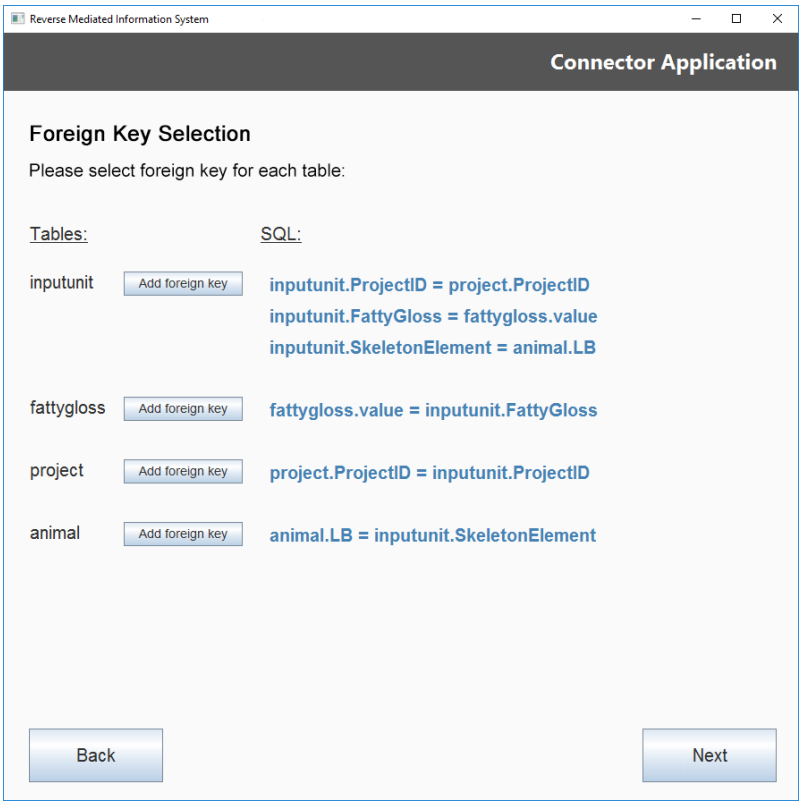


Figure 3.19: *Screenshot of the wizard dialog where administrators can define foreign keys to define related columns in separated tables, e.g. for the use of IDs and value tables.*

information. To make these heterogeneous databases searchable, the information is merged in a Data Warehouse, a central repository that retrieves and updates the integrated data from the diverse sources. The users only query the central repository. Data Warehouses do not require having to update existing databases to store new types of information, but they produce a large stream of data to query and update the central database if data is updated frequently in one of the heterogeneous databases. [Inm05] [JLVV03] In regards to our specified requirements, Data Warehouses violate the requirements R2 and R3.

- **Federated Information Systems (FIS) /**
  Federated Database Management Systems (FDBMS):
  These systems map multiple database systems into a single federated database. There is no actual data integration in the consisting disparate databases as a result of data federation. The virtual data of the FIS/FDBMS is queried directly by the user. It decomposes the query into subqueries for each of the federated databases. Finally, it composites the result sets of the subqueries. [SL90] [HM85] [BKLW99] Here again, the requirements R2 and R3 are violated.

- **Mediator:**
  An extension to the Federated Information Systems is the Mediator concept, "a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications" [Wie92]. Mediators allow the addition of new databases unknown to the user by adding a mediator layer which is responsible to translate the queries to the designated database. While the user can still query data without knowing the amount and structure of the queried databases, the Mediator needs to know all connected databases. If a new database is added, the Mediator is extended. [Wie94] [Wie13] [Bus02] However, the Mediator concept needs a central administrator who connects existing databases to the system which does not satisfy the requirement R3.

- **Other work:**
  There are further architectures and implementations for tools for accessing multiple information sources that are based on or adepts the Mediator concept. The *TSIMMIS Project* [CGMH+94] uses translators that converts the underlying data to a common information model, and incoming queries into object requests. The *Distributed Information Search Component* (DISCO) [TRV98] defines a Mediator schema for each Mediator as well as the collection of all participant data sources and their export schemes. The agent-based *Distributed Analytical Search* tool (DAS) [DAM15] deals with natural human language as a user input that is translated in a SQL query and provides a list of proposed queries that correspond to the request in a human language. However, these approaches also need an administrator that violates our requirement R3 again.

- **ADeX Standard:**
  Finally, in the archaeo-related disciplines there are also considerations to merge several distributed databases. The *Commission Archaeology and Information Systems* of the *Verband der Landesarchäologen*[16] defined the "*Archaeological Data eXport Standard*" (ADeX). Target of the project is a definition of a nationwide standard for the data exchange between the archaeological state departments and other specialized institutions and to realize a harmonized view on archaeological data throughout Germany. [GHH+17] [GHW11] However, this approach requires that the connected data sources apply the standard to their data scheme – the data retrieval is only possible if as many archaeological institutions as possible support ADeX. This will result in the databases being able to be connected in future and data being retrieved dependent on the standardized parameters (similar to the Minimal Find Sheet), but the standardized parameters are then not heterogeneous anymore and

[16] `http://www.landesarchaeologen.de`

the connection will need a central administrator again who manages the connections. Consequently, the requirements R1 and R3 are violated.

In contrast to these existing approaches, the *Reverse-Mediated Information System* fulfills all defined requirements from Chapter 3.2.2. REMIS connects heterogeneous data sources (R1) whose physical location is not known beforehand (R2). There is no need for a central administrator (R3) and the data sources can be searched independently on the database scheme (R4). The provided user rights management allows the data source owners to protect (maybe sensitive) information from unauthorized access (R5).

3.2.6 Case Study: Retrieving Archaeo-related Information

For a case study, we use real data of three database applications in archaeo-related sciences: OSSOBOOK, EXCABOOK, and ARCHAEOBOOK.

For each of these applications a Connector Application was configured to connect the data sources to the REMIS architecture. For our case, we required the research field "Archaeology" to be supported. The Server Application was initialized to require the Minimal Find Sheet information as the values for the Minimal Find Sheet. These are values that are always known in an excavation. They include the Feature Number of the excavation (which is centrally assigned by the *Bavarian State Office for Monuments and Sites*[17]), the Find Sheet Number, and location information (excavation place and x-/y-coordinates of the excavation).

The interested user can use the REMIS web interface to request data from the connected data sources, but they do not need to know that they exist. The web interface is shown in Figure 3.20 (the search mask) and in Figure 3.21 (the result view).

The search mask offers the opportunity to select the research field "Archaeology". Then the corresponding Minimal Find Sheet is loaded and can be entered which actually consist of the values of the Minimal Find Sheet.

We want to retrieve information about the excavation "Marienplatz-Haltepunkt" in Munich, Germany, of which we know the excavation number "M-2011-13-1". We enter this number into the corresponding input field of the web interface and click the "Retrieve Data" button. In the background, the ReMIS architecture will now query information from all connected databases – this means the Server Application will send the user request to the Connector Application of each of the registered databases. In our case, the Connector Application of OSSOBOOK, EXCABOOK, and ARCHAEOBOOK will each receive the user request and will then translate it to the local database scheme considering the mapping of the configuration. Then the query with the translated mapping is executed, and the retrieved information is sent back to the Server Application.

[17] `http://www.blfd.bayern.de`

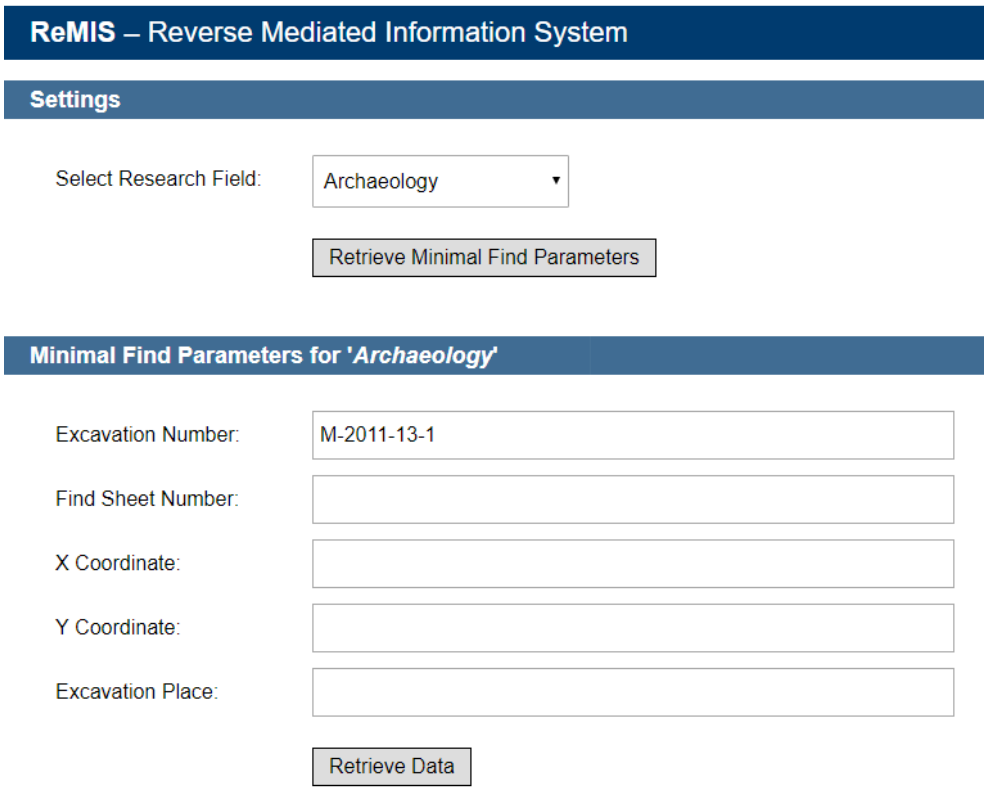


Figure 3.20: *Screenshot of the search mask of* REMIS.

ReMIS – Reverse Mediated Information System

Search Parameters

Back **Research Field:** Archaeology **Excavation Number:** M-2011-13-1 Print Export XLS Export XLSX Export CSV

OssoBook

ExcavationNumber	ArchaeologicalUnit	FeatureName	FeatureNumber	FeatureNumberDetailed	VerticalStratigraphy1	VerticalStratigraphy2	HorizontalStratigraphy1	HorizontalStratigraphy2	AddInfoArch
M-2011-13-1	4014	Stadtgraben	1229	Verfüllung	Schnitt 5		Fläche 4		Abschnitt 1 Süd, siehe auch Fz 3898
M-2011-13-1	2630	Schacht 11	1470	Verfüllung	Schnitt 11		Fläche 5	Planum 2-3	Abbau Planum 2 zu 3; Schlämmen
M-2011-13-1	2969	Stadtgraben	1508	Verfüllung	Schnitt 5	Profil 120	Fläche 4		Abbau Profil
M-2011-13-1	1760	Schacht 5	360	Verfüllung	Schnitt 3 W		Fläche 1		
M-2011-13-1	2131	Stadtgraben	1229	Verfüllung	Schnitt 5		Fläche 4	Anlage Planum 2	
M-2011-13-1	2630	Schacht 11	1470	Verfüllung	Schnitt 11		Fläche 5	Planum 2-3	Abbau Planum 2 zu 3; Schlämmen
M-2011-13-1	2969	Stadtgraben	1508	Verfüllung	Schnitt 5	Profil 120	Fläche 4		Abbau Profil
[…]									
M-2011-13-1	2106	Stadtgraben	1016	Verfüllung	Schnitt 5		Fläche 4	Planum unter 1	Abbau Befund unter Planum 1
M-2011-13-1	1344	Schacht 5	746	Verfüllung	Schnitt 3 W		Fläche 1		
M-2011-13-1	1066	Schacht 5	746	Verfüllung	Schnitt 3 W		Fläche 1		
M-2011-13-1	2726	Schacht 11	1470	Verfüllung	Schnitt 11		Fläche 5	Planum 3-4	Abbau Pl. 3-4

ExcaBook

FeatureNumber	Excavation Number	BriefDescription	Description	Comments	Date	NN_Up	NN_Down	Length	Width	Depth	Diameter	Interpretation	PreliminaryDa
4	M-2011-13-1	Befund	Befund		2011-06-07							Befund	
1	M-2011-13-1	Erschließungsstraße	Fläche 1		2011-08-03							Erschließungsstraße	
2	M-2011-13-1	Humushalde	Humushalde		2011-08-03							Humushalde	
1	M-2011-13-1	Grube	Pl. 1: rundlich, deutlich erkennbar mit verwaschenen Kon...		2011-10-27						40	Grube	
1	M-2011-13-1	Mauer	07.06.2016 Das aufgehende Mauerwerk besteht aus mi...		2011-05-22			511	45	35		Mauerwerk	NZ
2	M-2011-13-1	Wasserleitung	07.06.2016 evtl. Wasserleitung. Das aufgehende Mauerwe...		2011-05-24			134	53	33		Wasserleitung	NZ
1	M-2011-13-1	Grube	Pr. A-B (W-O): flach wannförmig, verschene Kontu...		2011-05-24				62	7		Grube	
3	M-2011-13-1	Befund	Befund		2011-06-06							Befund	

Figure 3.21: *Screenshot of the (shortened) retrieved result of* REMIS.

Then, the result is passed to the web interface which displays the information in table form to the users. Due to the different data schemes of each data source, the information from each data source is displayed in an own, collapsible block. In our search, we received information from the databases OSSOBOOK and EXCABOOK which is displayed to the users in the browser. However, there was no information stored in ARCHAEOBOOK for the selected excavation number and therefore no data was returned from its Connector Application– so it is not included in the result view.

In the result view, the users can now view and sort the results directly in the browser, but can also export the data to XLS, XLSX, or CSV files for further analyses in spreadsheet application or other tools.

Chapter 4

Supporting Individual Analyses of Archaeo-related Data

Attribution

This Chapter uses material from the following publications:

- Johannes-Y. Lohrer, Daniel Kaltenthaler, and Peer Kröger. Leveraging Data Analysis for Domain Experts: An Embeddable Framework for Basic Data Science Tasks. In *7th International Conference on Internet Technologies & Society 2016, Melbourne, VIC, Australia, 2016*, pages 51–58, 2016. [LKK16a]
- Daniel Kaltenthaler, Johannes-Y. Lohrer, and Peer Kröger. Supporting Domain Experts Understanding Their Data: A Visual Framework for Assembling High-Level Analysis Processes. In *11th International Conference on Interfaces and Human Computer Interaction 2017, Lisbon, Portugal, 2017*, pages 217–221, 2017 [KLK17]

See Chapter 1.2 for a detailed overview of incorporated publications.

In a database application for archaeo-related data, the accessibility to analyses of existing data is as important as the easiness of correct input. In Chapter 2 of this thesis a technical basis – a generic infrastructure for relational data management – was implemented to support archaeologists and bioarchaeologists in recording data of findings. In Chapter 3, we provided methods for collaboration, sharing, and retrieval of archaeo-related information. Based on these contributions, we now head to methods to support the scientists in analyzing the gathered information.

Entering and analyzing data of findings is the most important part of archaeological and bioarchaeological work. However, often the scientists require a high degree of time

and patience to learn, evaluate, and validate external tools that are necessary for analyses. This effort drains resource from their research work. Archaeologists and bioarchaeologists working in areas that are not related to IT technologies often do not have the motivation and the resources to get familiar with external applications. A built-in tool would provide the required data analysis features relevant for archaeo-related areas.

Currently, archaeological and bioarchaeological data has the form of numerous and detailed documentation about these projects. Until now, IT technology services were seldom used. The "data is archived on paper, often handwritten or typed. Even with the support of more modern technology, archaeological [and bioarchaeological] data has to be analyzed manually" [Kal12]. Archaeo-related information is stored in various databases for some of these findings. These are used in institutes, universities, museums, and scientific collections with archaeology or bioarchaeology as field of work.

Nevertheless, these database applications only support simple descriptive statistics or implement algorithms that were explicitly designed for the specific research areas of application. To meet the goals of specific archaeo-related research data is usually exported from a database in the form of spreadsheets files – e.g. Microsoft Excel, LibreOffice Calc, etc. – or comma-separated values data to be analyzed in extern applications or tools. This method can be aggravating, time-consuming, and error-prone, but it is still common practice.

But even if the analysis tools are built into the application, a crucial problem still remains: The scientists who are carrying out the analyses are often not responsible for creating the analysis tools nor are they even involved in the process of creating the tools. Not all variations of analyses can be known beforehand since there are often tasks specifically dependent on the domain-specific work. So the archaeologists and bioarchaeologists have to be able to create exactly the analyses they require. Apart from offering a rich set of options for the users to analyze the data, it is also important to offer an intuitive, user-friendly, and meaningful graphical user interface that allow them to carry out the analyses easily. To allow this, the application has to provide not only predefined analysis methods, but also offer dynamic generation of analyses by chaining together different simple configurable modules. However, some scientific tasks are so special that these modules are insufficient. Therefore, there has to be a way to add own, specific modules as well.

There are available free and commercial tools of many kinds that provide visual querying data from a database. Tools like *QueryVOWL* [HLSE15], *Query Xtractor* [Dat16], or *SQLeo Visual Query Builder* [SQL12] focus on form and graph based querying data and returning the result as a table. The majority of these tools are too complex, technical, and thus hard to be used by regular users. In addition, the possibility of further complex analyses and a graphical representation of the data is lacking. Still, there are user-friendly and clear solutions to analyze data from relational databases, like *datapine*

[dat12] or *Klipfolio* [Inc01] to visualize business data and their key performance indicators, or *Qlik Sense* [Qli05] as a more general example. While regular users can easily create analyses and the corresponding graphs, these tools are limited to their possibilities of compositions. Querying and assembling custom and complex data is rarely possible. Furthermore, all of these tools are extern applications that are not embeddable to existing applications and have to be connected to the appropriate database by the users.

Our ambition is to provide a dynamic, flexible, and powerful ANALYSIS TOOL that allows specific queries from archaeological and bioarchaeological databases without having any prior knowledge of programming. We want to support archaeologists and bioarchaeologists in their research and to provide the ability to select specific data from their database as needed with as few limitations as possible. The target is to provide a flexible tool that scientists can use for visually querying the data directly in their database applications and create almost every composition for their data analyses. Furthermore, we want to offer a possibility to generate graphical results that archaeologists and bioarchaeologists can use for their analyses and publications, without the need to export any data and use external analysis tools or spreadsheets.

Therefore, we first describe the architecture and technical realization of the ANALYSIS TOOL in Chapter 4.1. Afterwards, we discuss and evaluate possible approaches for the graphical representations of the data composition in Chapter 4.2. Then we describe how a typical analysis is built in Chapter 4.3. Finally, we integrate a prototype into OSSOBOOK and apply it to real zooarchaeological data in Chapter 4.4.

4.1 Technical Structure

In this Chapter, we describe the technical structure of the ANALYSIS TOOL, its implementation, and its integration to the XBOOK framework.

Therefore, we discuss the necessary requirements in Chapter 4.1.1 and give a quick overview about existing approaches in Chapter 4.1.2. Then, we describe the implementation of the ANALYSIS TOOL in Chapter 4.1.3 and show how to integrate the framework into existing applications in Chapter 4.1.4. Finally, we describe the most basic modules that are integrated to the ANALYSIS TOOL in Chapter 4.1.5 and explain how the integration of individually modules and analysis methods is handled in the tool in Chapter 4.1.6.

4.1.1 Necessary Requirements for Dynamic Analyses

Allowing the users to generate their own analyses out of arbitrary data brings the challenge that neither the input nor the output is known. All steps in between are also up to the users. Still, the users have to be able to work in a responsive environment which

allows them to carry out the steps they want and need. Additionally, not all operations should be allowed or are possible at a given step of the analysis pipeline.

As a consequence, we have defined several requirements that have to be met to allow a dynamic generation of analyses.

- **Dynamic data structure:**
  We need a data structure that allows data to be easily added, removed, merged, and accessed, since the data has to be generated, transformed, searched in, and of course also be displayed. This data structure has also to be usable in every part of the analyses independent of the previous steps.

- **Modular components:**
  Since the users have to define the operational steps, the order in which specific tasks are run is not known beforehand. Therefore, we need modular components that can be linked together and define a "workflow". The components have to be able to accept the data structure – no matter what components were used before, but not all inputs have to be valid for the component. Therefore it is possible that none of the inputs can be used by the component, but still the data structure itself has to be accepted. It also should provide settings like which input to use or the order to sort to make the component more dynamic. These settings should, if applicable, dynamically change depending on the input.

- **Classification:**
  All fields, for which the data can be retrieved for, have to be defined in a way that each component can decide, if or how the data for this field can be processed. For example, a date value may have to be treated differently than a pure text value. With this classification a component can also decide if it is able to work with the field or not.

- **Extendability:**
  It has to be possible for everyone to extend the application with new components that offer new, possibly very specific functions. Since there are many special analyses that cannot be put together with generic modules, it has to be possible to include these in the list of available workers.

- **Open Source:**
  The area of application might be in an open source environment, therefore we also require the analyses to be open source to be able to include it into the application. Some other licenses might be applicable as well, but still this is a requirement that is very important due to legal issues.

- **Embeddability:**
  Since the analyses shall be done with the data the users might just momentarily have entered, we require the analyses to run directly from the program without first having to extract the data into spreadsheet, CSV files, or similar. This means that the analyses have to be able to connect itself to another program and use the structure defined inside the program for its analyses.

4.1.2 Existing Analysis Methods

Now, we discuss if some of the most commonly used tools meet our requirements. Since there are many different analysis tools [Jon16], we cannot cover and evaluate all of them. Therefore, we do not claim to offer an exhaustive comparison, but still think we covered some of the most used tools that best fulfill our requirements.

- **Tableau Public:**
  As a commercial service, *Tableau Public* [Tab03] allows creating interactive data for publications in the web that can also be used for analyses of any data. The tool supports the analysis of text, numbers, dates, and coordinate values and offers extensive visualization methods.

  The free edition of the tool is provided on a limited scale. As a data source only a few file formats are supported, like Microsoft Excel, CSV files, or some files types for statistical data. More data sources, like the connection to database systems (MySQL, Oracle, Microsoft SQL, etc.), only come along with the professional edition which is subject to a fee. *Tableau Public* cannot be included directly into another application. Files are always saved on the own profile and cannot be downloaded or saved on the own computer in the free edition. Furthermore, it is not extendable – the existing analysis methods cannot be extended with own, specific ones.

- **KNIME:**
  A modular approach with graphical nodes that allows many different input methods including tables, comma-separated values files, and even images is used by *KNIME*. It enables the combination of “simple text files, databases, documents, images, networks, and [...] Hadoop based data”[KNI06] and integrates the use of modules for data blending, transformations, math and statistical functions, and predictive algorithms. The extension of new modules is supported by providing an API. However, the application is an external one that cannot be included into another application.

- **RapidMiner:**
  Just like *KNIME*, *RapidMiner* [Rap06] also uses graphical nodes for the representation of the data. It is an open source project in the basic version and can be extended with own nodes. It supports connections to databases. But, this tool has also the disadvantage that it cannot be used inside the main application and has to be run as a separate instance.

In general, all of the mentioned tools are very good in what they do. They allow a wide range of different analysis methods and also can be partially extended with new functions. Some of them also allow a connection directly to the database without having to extract the data first into an own file. Furthermore, they all offer some kind of classification of the different fields. Still, none of the tools allow an integration into an existing application.

4.1.3 Analysis Tool Realization

In Chapter 4.1.1 we discussed the requirements for dynamic analyses. Now, we take up these requirements and present the ANALYSIS TOOL, our approach to meet them. Especially for the data structure there of course can be more than one valid solution. Since we use a Java environment, we use some common names of constructs implemented in Java. These still should be available in different languages under different names.

Data structure

A data structure is required to allow a dynamic and flexible restructuring and access to the data. Therefore, the data structure for our framework consists of several elements.

- **Column Header:**
  The Column Header holds the important information about the specific field, such as the type and the display name. Since a field is basically a column in the database, we use the name of the field and the column as synonyms. The Column Header itself does not store any information about the value of the field, but serves more as information and meta data for the column. Additional information that is stored in the Column Header is described later in more detail.

- **Entry Value:**
  The smallest data type to hold the values is the Entry Value. It holds a list of strings which represents the values for a specific entry. This is necessary if a field has several options, e.g. different values for specific measurements. Therefore, every distinct value is one string and all of them are saved inside the Entry Value.

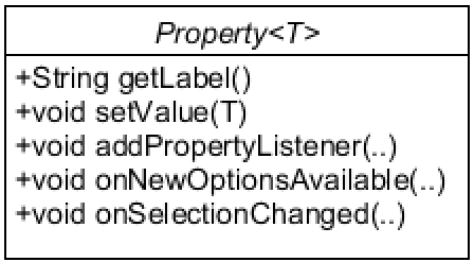


Figure 4.1: *Schematic representation of a Property.*

- **Entry Map:**
  The Entry Map maps the Column Header to the Entry Value inside a map. This map now represents a complete entry. It can easily be accessed because of the nature of the map. Therefore, writing and reading is no problem. Editing values can also easily be done.

- **Data List:**
  The Data List is a list of all Entry Maps which represents a complete data set. It also contains the list of all Column Headers that are in any of the Entry Maps. This serves as a utility method to allow components easily to access to the list of all fields without having to iterate over all entries. The list is generated by adding all unknown Column Headers whenever a new Entry Map is added to the Data List.

Worker

Each component, or "Worker" as we call it, can have multiple inputs and outputs of data. Some Workers do not necessarily have inputs, like Workers that retrieve data from the database, since they generate data without manipulating it. At the same time there can be Workers without outputs, like a Worker that allows the users to visualize the data as a diagram.

- **Properties:**
  The Worker consists of inputs and outputs, settings, and the actual logic of the Worker. The latter either uses the input(s) or creates a new Data List, runs its logic considering the settings, and finally outputs the data or visualizes it.

  The Worker defines a list of Properties which can represent a setting. Therefore, for every setting type there has to be an own property. Typical properties are text properties where the users can enter text (for example to describe a name) or combo properties where the users can select one value from a list of values. The schematic representation of the property can be seen in Figure 4.1.

Below, we describe the methods that a Property has to implement:

- `getLabel()`:
  Returns the label to describe the property. This should make clear for the users which setting they can manipulate.
- `setValue()`:
  Called by the graphical representation with the specific value. This is used to tell the Worker that the value has changed and has to update its data structure and possibly additional properties.
- `addPropertyListener()`:
  All elements that are dependent to this property can register themselves as listeners. This way the registered elements are notified if this property has changed. It may be called when either new options are available that can be selected or the value of the property itself was set.
- `onNewOptionAvailable()`:
  This is called if new options for this property are available. This causes all property listeners to be notified to update the displayed values in the graphical user interface.
- `onSelectionChanged()`:
  Almost identical to the method `onNewOptionsAvailable()`, but it is called when the value of the property is changed, for example after the `setValue()` method was called.

The Worker also defines the number of inputs and outputs. The users then connects different Workers. These are basically a directed graph or multigraph with the nodes representing the single Workers and the edges representing the connections between the Workers. The number of inputs can vary as some Workers require no input, some require an exact amount, and some can work with an arbitrary number of inputs. If the Worker provides an output this can be used to pass the data on to other Workers.

- **Data handling:**
  Each Worker fulfills a predefined task like retrieving or merging data. But still, the output of the Worker is only defined after the input(s) and settings for the Worker are set. Therefore, it is not necessary to instantly generate the output since the input(s) may change if a setting in a previous Worker was changed. It is important to know at least the Column Headers of the data to enable updating the settings of the successive connected Workers. This is the reason why the output is separated in two parts:

  - **Output Scheme:**
    The list of all Column Headers that is returned by this Worker. This list is instantaneously generated as soon as an input or setting is changed. The successive connected Workers are immediately notified with an event that the Output Scheme of this Worker was changed. This enables the Workers checking themselves if their Output Scheme changed as well. This list has the same value that the list of Column Headers inside the Data List should have in the real output. Therefore, Workers only need this list to define what settings they provide and what is their Output Scheme.
  - **Output Data:**
    The Data List with the "real" output containing the data. This is only generated if necessary since the composition of the data could take some time. A complete recalculation is not required if only the Output Scheme is important. Of course, the real data has to be generated if a diagram representing the data has to be displayed or the values should be listed. This is done recursively: Each worker – beginning from the end – requests the Output Data of the previous Workers that it is connected to. Of course, this means the starting Worker has to be able to generate data without any input.

  This separation into Output Scheme and Output Data allows a fast applying of updated settings and rearranged inputs or outputs while still ensuring that the correct type is used.

- **Interaction with the graphical user interface:**
  The Worker itself is only responsible for the logic. But since the users need to be able to easily arrange, connect, and configure the Workers there has to be a graphical representation.

  The graphical user interface is notified with events of changes in the properties. At the same time it uses the properties to notify back to the Worker if any value of the settings was changed, such as entering a text or selecting a new value. This allows the graphical user interface to be created independently of the logic and therefore is not limited to a specific format. The exact design of the graphical user interface is described in Chapter 4.2.

Level of Measurement

Since typically the database containing the data consists of multiple tables with a variety of different types of entries, not all columns can be used to carry out every operation. For example, a text cannot be used for a numeric ordering of entries or a numeric value

should not be alphabetically ordered. Therefore, for every column has to be defined what types of operation are compatible with the column.

This is achieved by using the *Level of Measurement*, a classification that describes the nature of information within the values [Ste46]. It defines four different basic scales of measurement:

- **Nominal Scale:**
  The different values can just be differentiated by their names. Therefore it can only be used to get the quantity and to check if a value equals another or not.
  Possible modules: Filter, counter.

- **Ordinal Scale:**
  The values can be ordered, but do not allow to measure the degree of difference between different values.
  Possible modules: Sorter, median, mode, and all modules of Nominal Scale.

- **Interval Scale:**
  The values can be measured by degree of difference, but not the ratio between them.
  Possible modules: Arithmetic mean, standard deviation, and all modules of Ordinal Scale.

- **Ratio Scale:**
  In addition to a degree of difference, values have a meaningful zero value.
  Possible modules: Elementary Arithmetic and all modules of Interval Scale.

Since all columns now have a definition of which scale they have, modules can define which scale they can work with. Therefore, they only work with columns that have the required scale. For our intention every column has to be assigned to one of the scales. The corresponding Level of Measurement of the columns is saved inside the Column Header. Additionally, we require information how to order the entries for every column. The default ordering is alphabetically. Still, numeric values should be ordered accordingly and also text values that have a ordinal scale should be ordered correctly. This requires a Comparator to be set inside the Column Header for the corresponding column.

4.1.4 Analysis Tool Embeddability

To be able to reuse the ANALYSIS TOOL in different base applications, it is important that the integration requires as few changes to the base application as possible. But since the

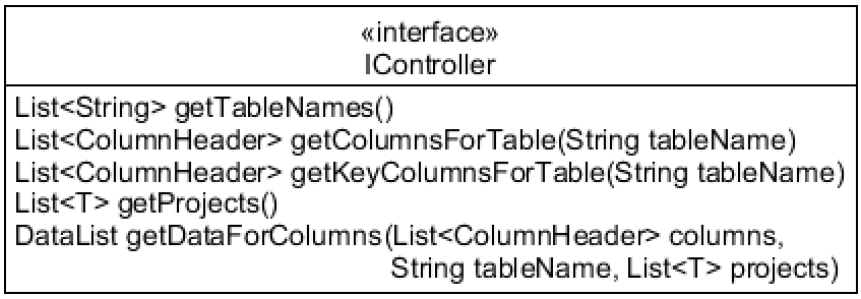


Figure 4.2: *Schematic representation of the IController interface.*

ANALYSIS TOOL cannot know how the data is stored or how the connection to the data is realized, the base application has to implement some wrapper methods.

The ANALYSIS TOOL itself is composed in a `JPanel` or `JFXScene`. The base application can either create a new `AnalysisSwing` or `AnalysisFX` instance which both require an `IController` (cf. Figure 4.2). The `IController` is the interface that serves as the combination of the base application to the ANALYSIS TOOL. It provides the following methods that the base application has to implement:

- `getTableNames()`:
  Returns the list of different names of tables that can be selected for the analysis. This should only return tables in which the users have entered data.

- `getColumnsForTable()`:
  Returns the columns for the given table that can be included in the analysis. This should return only columns that are important for the users, but no columns that contain additional information that provide no information for the users. The expected return value is a list of Column Headers. Therefore, all columns have to be transformed into this format. This is important because all future analysis options are based upon the information stored inside the Column Headers.

- `getKeysForTable()`:
  Returns the key columns for the given table. This can be used for example in the Combiner to provide default mappings. Also this method requires the returned values to be a list of Column Headers.

- `getProjects()`:
  The database can possibly be structured in different "projects" which is a logical separation of different data sets. A list of a unique identifiers can be returned to also allow this separation inside the analysis framework. This could be just a name or an integer, but can also be a more complex data structure, in which the `toString()` method returns the name of the project to be able to display it to the

users. This method can return `null` if the separation into different projects is not desired or supported.

- `getDataForColumns()`:
  Returns the Data List for the given columns in the given table for the optional project. Since the ANALYSIS TOOL has no knowledge about the structure of the database, the base application has to generate the Data List. The list of projects is only required if `getProjects()` returns a value and therefore can be ignored if it is not applicable. The values that are returned should most likely not be directly the values as they are stored in the database, but already translated into human readable form, e.g. by translating IDs into the appropriate values.

If necessary, all Workers can then use the `IController` to retrieve the data they require. This is the only connection data-wise needed for the analysis as all further analyses are built onto the retrieved data. Therefore, the base application has not to know any internals of the analysis or other way around. Since the analysis is done inside one panel, it can be easily be included into the base application which can decide where and when the analysis shall be displayed.

4.1.5 Provided Basic Workers

We have already mentioned that there are types of Workers that fulfill different tasks. Here we want to give an example of the most common Workers that are sufficient for a basic analysis that are already integrated to the ANALYSIS TOOL by default. A screenshot of these Workers from the ANALYSIS TOOL can be viewed in Figure 4.3. Of course, there are far more different Worker types possible, but we only want to give an overview over the possibilities the Workers provide.

Retriever

The Retriever is the most basic worker that is used in every analysis. The Retriever, as the name suggests, retrieves data from the database. The users can define the fields and projects they want to include in their analyses with the fields and projects being a structure for the data in the application the ANALYSIS TOOL is developed for. This would normally be the fields the users either want to filter, display, or further analyze.

The Retriever is the only Worker that needs a connection to the database to get any data. It uses the `IController` to get the required data. The retrieval from the database is most likely a simple SQL query for all fields selected by the users.

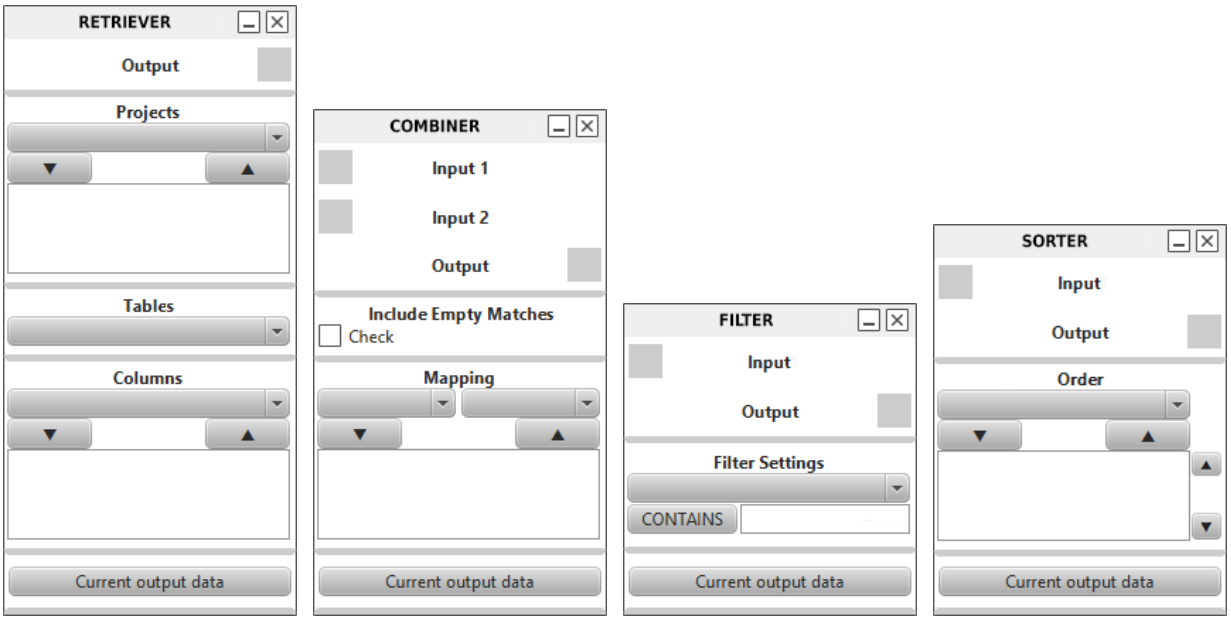


Figure 4.3: *From left to right, the screenshots of the Retriever, Combiner, Filter, and Sorter, as they are represented in the graphical user interface of the* ANALYSIS TOOL.

Combiner

The Combiner allows different data sets to be combined. The users can define the fields which should be considered when combining the data sets. The list of available fields is defined by the Output Scheme of the previous Workers as described in Chapter 4.1.3.

For example, the Combiner can be used if a user has already created two different analyses with different data sets and now wants to combine the data for a third analysis. The Combiner searches for entries in the given list of data sets that are equal on the fields the user entered. The result of the combination can be compared to the SQL join. There are two possibilities that either only entries having a match or also entries that have no match are combined. In this case the other columns are filled with empty values. Then these entries are combined into a new entry.

Filter

The Filter can be used to filter out entries that are not required in the analyses. For example, an analysis about specific species may only require the entries containing data of these species. So the users can filter out all other species using the Filter.

The users have several options: They can specify the fields for which the data shall be filtered. The list of available fields is defined by the Output Scheme of the previous Worker, as described in Chapter 4.1.3. Depending on the column header, they can specify whether the value of the field contains or equals a specific text. For dates or numbers

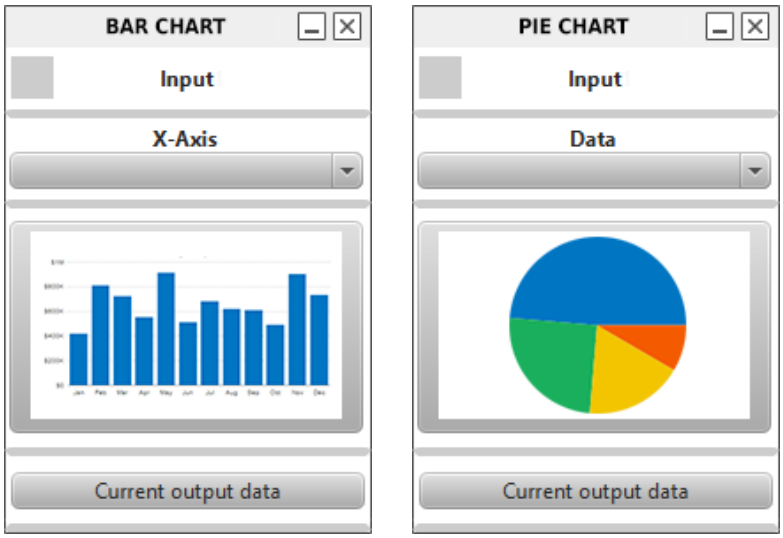


Figure 4.4: *Examples for different Diagrams Workers*

it is possible to check if the value is smaller, greater, or equal than a specific user input. It is also supported to define a combination of different fields that can be filtered at the same time.

Sorter

The Sorter sorts entries according to the comparator set in the Column Header (cf. Chapter 4.1.3). It only has one input, but allows more than one column to be set as sorting column. This allows applying different priorities if the column values with a higher priority are equal. For example, this can be useful if in a diagram a specific order of entries is required.

Diagram

The Diagram is the umbrella term for Workers that display the result in a graphical representation. They can be used in different and complex ways.

As an example, we describe a two-dimensional, axis-oriented bar chart. The users can select the field which values shall be used for the x-axis. The list of available fields is defined by the Output Scheme of the previous Worker as described in Chapter 4.1.3. The different values of this field are then listed as values of the x-axis. The number of entries for the given value are displayed on the y-axis. An example of a Diagram can be viewed in Figure 4.4.

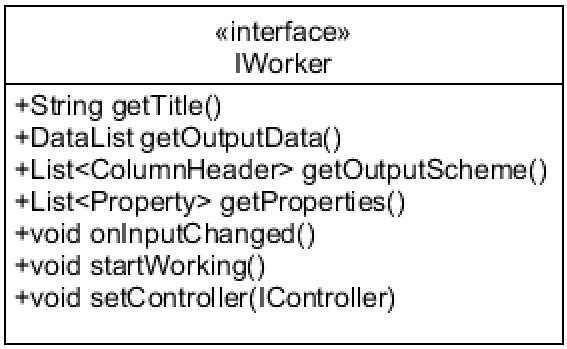


Figure 4.5: *Schematic representation of the IWorker interface.*

4.1.6 Definition of Custom Workers

Not all use cases can be known while creating the ANALYSIS TOOL since the area in which the analyses are used is not always known in advance. Therefore, it is very important to be able to add new Workers which exactly fulfill the requirements of the individual analysis that shall be carried out.

For this, we defined a very simple API to allow new Workers to be easily implemented. To add a new Worker, it has to implement the interface `IWorker` (cf. Figure 4.5). It defines the basic functions that are required for the integration in the workflow.

- `getTitle()`:
  This method expects the name of the Worker to be returned as a string value. This name is displayed for the users inside the graphical user interface. It should be a short but meaningful name which allows the users to instantly comprehend the function of the specific Worker.

- `getProperties()`:
  Returns a list of Properties which define the settings of this Worker. Specific aspects for the Worker can be set with these settings (cf. Chapter 4.1.3). This list of Properties also includes Properties for the input and output which define the connection to other Workers. The Worker is notified over the Properties if the input or a setting has been changed. Since the Properties are abstract classes, all methods of these classes have to be implemented when creating a new Worker (cf. Figure 4.1). This includes the displayed name, the logic for setting and retrieving data, and additionally what values the representation in the graphical user interface of the Property shall display – depending on the Property.

- `getOutputData()`:
  Returns the Output Data of the Worker after it has completed its job. If neither the input nor the settings of the Worker have changed this can return the generated

values from the previous call. Otherwise the process has to be run again. Therefore, this method has to call in this case the `startWorking()` method and return the generated values after it has completed.

- `getOutputScheme()`:
  Returns the Output Scheme of the Worker that would be returned in the Output Data if the Worker would return its generated data set. The Output Scheme should be calculated with regards to the Output Scheme of the connected previous Workers (if there are any) and the settings of this Worker. If the input and settings of the Worker did not change, this method does not have to recreate the Output Scheme again, but can just return the previously generated data. This method should also not run the complete calculation method but only calculate the Output Scheme.

- `onInputChanged()`:
  This method is called by the Properties to notify the Worker that something has changed and the Output Scheme and Output Data have to be recalculated, if requested. However, this method should not start the update process itself.

- `startWorking()`:
  In this method the actual logic is carried out. The input is collected by iterating over all input Properties and getting the Output Data of the connected Workers. This triggers them to generate their results themselves if needed by carrying out the same logic recursively. Then the result of the Worker is calculated with regards to the settings defined in the Properties. This method is usually only called inside the `getOutputData()` method.

- `setController()`:
  This method is used to set the `IController` which is used for interaction with the base application. This is required to be an own method and not part of the constructor since all Workers are created with reflection. Therefore, the constructor has to be the empty constructor. With the `IController` the Worker is able to query the base application for information that is possibly required for the Worker to carry out specific operations.

After the Worker has been created, there are two different options to register it to the application: A direct and an indirect approach:

- **Direct registration:**
  The API provides a registry class which allows Workers to be registered for inclusion in the ANALYSIS TOOL.

In addition to the Workers available by default, the registered Workers can then be selected by the users. This method requires of course access at runtime and therefore can usually only be done from the base application.

- **Indirect registration:**
  The indirect registration allows externally created Workers to be included in the ANALYSIS TOOL. For this all .JAR files inside a specified folder (default "./analyses/workers") are analyzed if they contain classes that implement `IWorker`. If so, they are added to the list of available Workers for analyses.

Since the Workers use Properties to tell the graphical user interface what to display, it is important to also add new Properties if the available Properties are not sufficient. This consists of two parts: The definition of the property itself and also of the definition of the graphical representation of the Property.

All new properties has to extend the `Property` class (cf. Figure 4.1). Thus it can define additional methods that are used by the representation of the graphical user interface. Then it can be registered together with the graphical representation in the registry.

4.2 Graphical Representation

In this Chapter, we discuss and evaluate the graphical representation of the data composition by offering the possibility to embed a visual querying framework to provide this functionality.

Accordingly, we list a set of requirements that we address in Chapter 4.2.1. This has been extracted from comprehensive discussions with domain experts, reflecting their typical analysis procedures. We discuss four different approaches for a potential visual querying in the ANALYSIS TOOL in Chapter 4.2.2. Then we evaluate them in a user study in Chapter 4.2.3. Finally – considering the evaluations – we revise the approaches and implement a prototype of the ANALYSIS TOOL in Chapter 4.2.4.

4.2.1 Requirements for Graphical Analyses

For the graphical solution for the ANALYSIS TOOL, we defined the requirements below as necessary for the development of the graphical composition of the data:

- **User-friendliness:**
  A graphical realization within the application for users without any programming skills. We need user-friendly database queries that have to be composed without the knowledge of programming code and writing SQL queries.

- **Any composition of data:**
  Extensive queries have to be possible due to the database structure. Queries can also be complex and versatile. The users should be able to query a database for every composition of data they need for their analyses.

- **As few restrictions as possible:**
  The users should have as few restrictions as possible while compositing data. We want to achieve a flexibility in querying data similar to SQL queries without overwhelming the user with the complexity of programming interfaces.

- **Reusable modules:**
  The different types of queries and data handling have to be packed into single modules to realize the different functionalities that can be reused in several areas.

- **Dynamic behavior:**
  Since every database may have a different structure, it is impossible to define all variations manually. Furthermore, it may happen with every application update that input fields are added, removed, or changed. To keep the maintenance effort as low as possible for the ANALYSIS TOOL, the single modules have to have a dynamic behavior to read out the available database structure.

4.2.2 Discussion of Composition Approaches

Below we introduce several options we considered for our implementation and discuss the advantages and disadvantages of the respective approaches. For simplification we only use the basic modules described in Chapter at:examples which we use in the explanations, images, and compositions below.

Puzzle Piece Approach

As sketched in Figure 4.6 the modules are represented by single puzzle pieces that are arranged from top to bottom to realize the composition. Every puzzle piece applies the functionality of the module on the input data that is handed over through the top connection(s). The manipulated data is returned to the output which is represented as the bottom connection. By putting two puzzle pieces together the output data of the upper puzzle piece becomes the input data of the lower puzzle piece.

Specific settings (optional and mandatory ones) for the modules can be set by attaching other puzzle pieces (hereinafter named as 'setting pieces') to the left or right side of the puzzle piece which can be configured individually. These are indicated with free connections where a setting piece can be attached. The shape of the connection indicates which type of setting piece can be attached. For example, the connection is

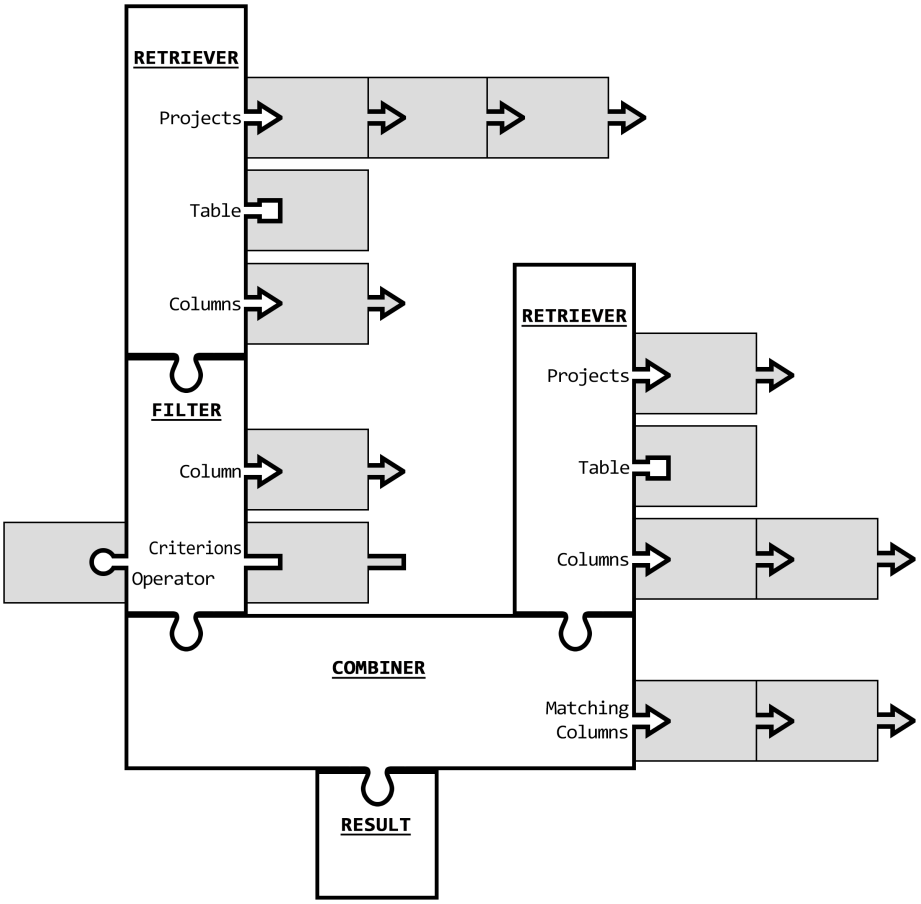


Figure 4.6: *Sketch of the Puzzle Piece Approach.*

indicated with a circular shape if a setting requires a boolean value as an input. A rectangle shaped setting piece, which represents a selection of a combo box, would not fit and cannot be attached. This way the users can recognize which setting pieces can be attached to which puzzle piece.

A composition is defined as valid if all input and output connections of the modules are assigned. The composition has to be framed by Retrievers (which have no inputs) on the top and Result Modules (which have no output) at the bottom. Furthermore, all mandatory settings have to be connected with at least one setting piece.

Rating:

- **Advantages:**
  The puzzle pieces are self-explanatory, the users can recognize where they can attach new pieces because of the shapes of the connections. The usage of shapes is also a big advantage for color-blind people. The top to bottom arrangement of

the modules is a clear and intuitive structure. The left to right arrangement of the setting pieces gives a clear impression about the adjusted settings corresponding to the specific module.

- **Disadvantages:**
  The use of complex and numerous settings makes the representation wide and confusing if there is more than one puzzle piece parallel on the same level. Furthermore, there is only less space to enter the setting criteria on the single setting pieces without making them too wide.

- **Summary:**
  The Puzzle Piece Approach is a clearly arranged way to represent a data composition in a graphical representation for simple databases with only one data table. As soon as it is necessary to use puzzle pieces with more than one input (e.g. to join several data sets) the graphical representation gets too wide and complex. So it is not applicable to our application. However, the Puzzle Piece Approach could be an operational solution if parallel pieces on the same level are not necessary.

Box Piece Approach

The Box Piece Approach, as illustrated in Figure 4.7, is a further development and variation of the Puzzle Piece Approach (see above), trying to reduce the complexity of the graphical representation.

The general top to bottom arrangement remains unchanged, only the connections are replaced with docking points to clarify the input and output locations. The setting pieces are removed completely and are not visible in the composition overview anymore. Instead the users can now double-click the single boxes to open a pop-up that provides all necessary settings for the corresponding box. This pop-up may display all available properties to set the box as well as further information to the users.

Rating:

- **Advantages:**
  Opening the settings in an own pop-up provides more space for displaying the setting options. The general overview of the graphical representation is improved a lot due to the removal of the setting elements. The layout of the rectangular elements yields a more clearly arranged representation of the composition.

- **Disadvantages:**
  There is insufficient space for the display of any information if the settings are integrated into the box itself. To get detailed information it is necessary to either

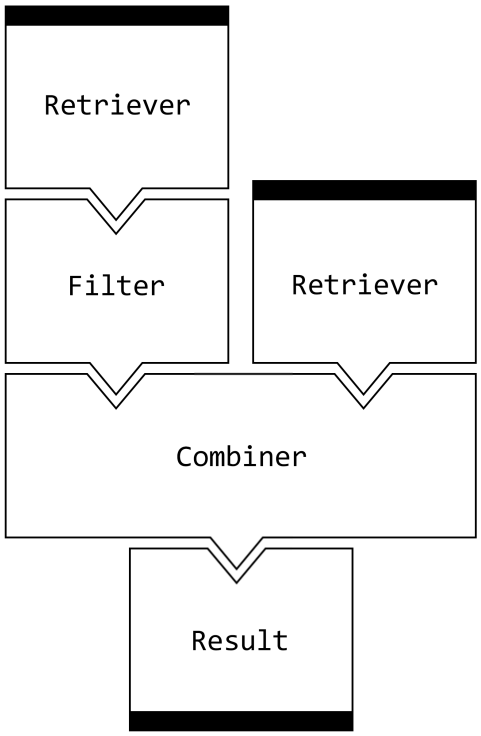


Figure 4.7: *Sketch of the Box Piece Approach.*

open the settings panel or add a kind of tooltip to the box. This might get confusing in case of complex data compositions.

- **Summary:**
  The compact display of the Box Piece Approach is suitable for complex data compositions. The ability to open the settings of each box enables the display of a high number of properties. In comparison to the Puzzle Piece Approach it might grant more overview, but the lack of space for displaying information causes a trade-off. The latter will probably be the strongest point of criticism from the users' point of view.

Directed Graph Approach

Another presentation of the data composition can be achieved by using a directed graph that is shown in Figure 4.8. Each vertex represents one module which is connected by edges to other modules where the edges have a direction associated with them. This way the output data of each module can be connected as an input of other modules.

Every module is displayed as a box holding all necessary information and settings for the module. Each property is displayed as a "row" within the box where the settings can be set directly. The input(s) and output of the data are also added each as an own row.

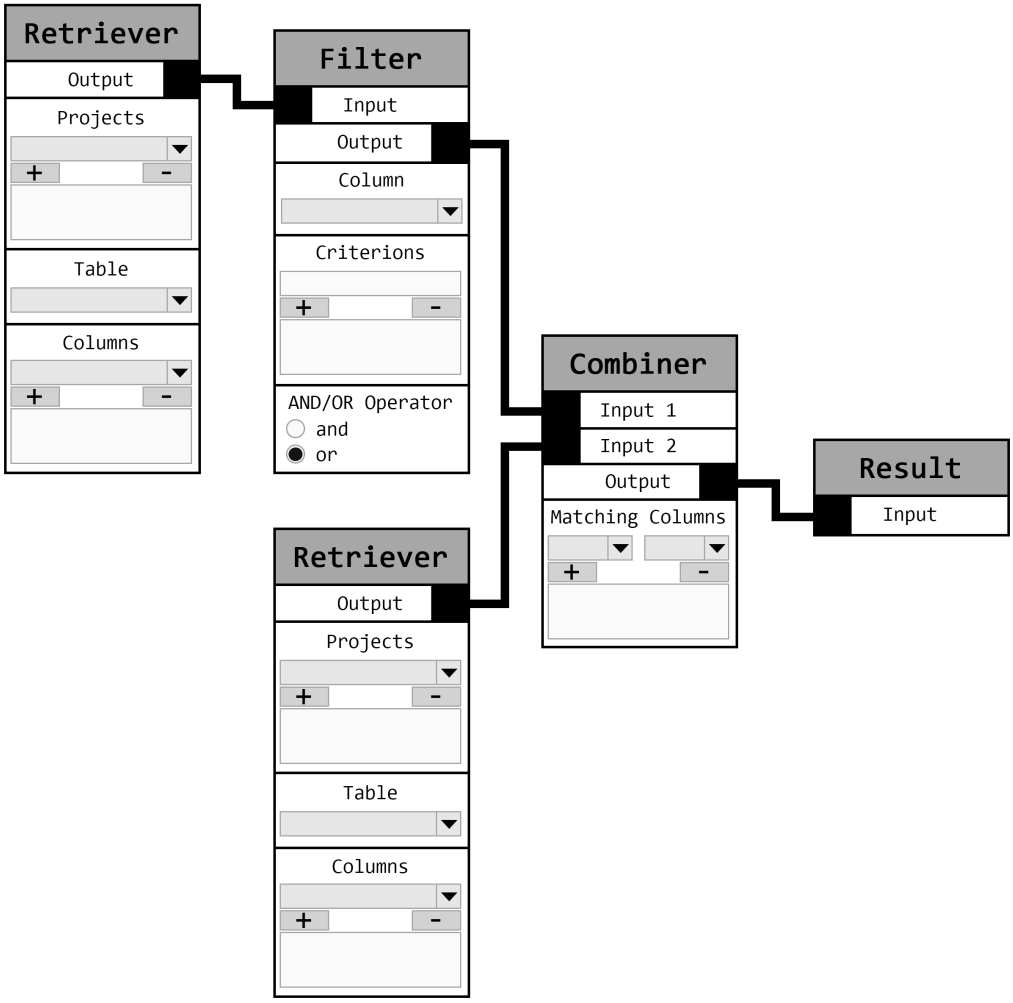


Figure 4.8: *Sketch of the Directed Graph Approach.*

To gain a valid data composition all input and output connections of the modules have to be assigned to another connection, too. Furthermore, the mandatory settings of each module have to be defined in each single row.

Rating:

- **Advantages:**
  The representation as a directed graph has the advantage that it can also be used as a directed multigraph. The data output of a module can be connected to multiple data inputs of other modules as well. This way some data can be reused in different parts of the data composition without having to redefine them again. The removal of single connections might be easier than in other approaches because the visual and logical connection can be deleted without handling the boxes.

- **Disadvantages:**
  The use of directed multigraphs may cause loops that would cause infinite loops while compositing the data. These loops have to be prevented in the program logic. Furthermore, it has to be considered how to allow the users rearrangements of the composition, whether the users can move the modules to any position or the application force specific locations. Furthermore, the size of the module boxes might get too high if there are too many or too complex setting rows, but possibly a function to hide and unhide single setting rows would avoid this problem.

- **Summary:**
  In our application the Directed Graph Approach fulfills all necessary requirements. However, the user-friendliness is lowered with more complex data compositions that are using modules with numerous setting rows.

Static Content Approach

Finally, we take a look at the Static Content Approach that is illustrated as an example in Figure 4.9. It is probably the approach for the most limited complexity for data composition. It is based on the idea to predefine specific queries in SQL, but let the users select the parameters in the graphical user interface, e.g. values, specific columns, etc. Each data composition is predefined and only the corresponding parameters have to be set by the users. This approach is common in data visualization software for smaller businesses.

It is an easy way to gain a user-friendly solution to visualize data because the users do not have to composite data by themselves. There are versatile possibilities to design the graphical user interface of the application. Some input fields – for text, combo box, check boxes, etc. – suffice for the input, but it can also be realized by more modern methods, like drag and drop elements. So a clearly structured interface is easy to be realized.

To gain a valid data composition the filling of all parameters is sufficient. The interface can display detailed feedback for missing and optional inputs that is possible because the compositions are limited in their complexity.

Rating:

- **Advantages:**
  The Static Content Approach is probably the user-friendliest approach. The users can use predefined data compositions simply by entering some data. They do not have to worry about logic and do not need knowledge about any parts of the database structure. Furthermore, the predefinition of the composition in SQL allows the optimization of the queries which can be executed more efficient and faster. Due to the lack of complexity for the data composition, it is also possible to

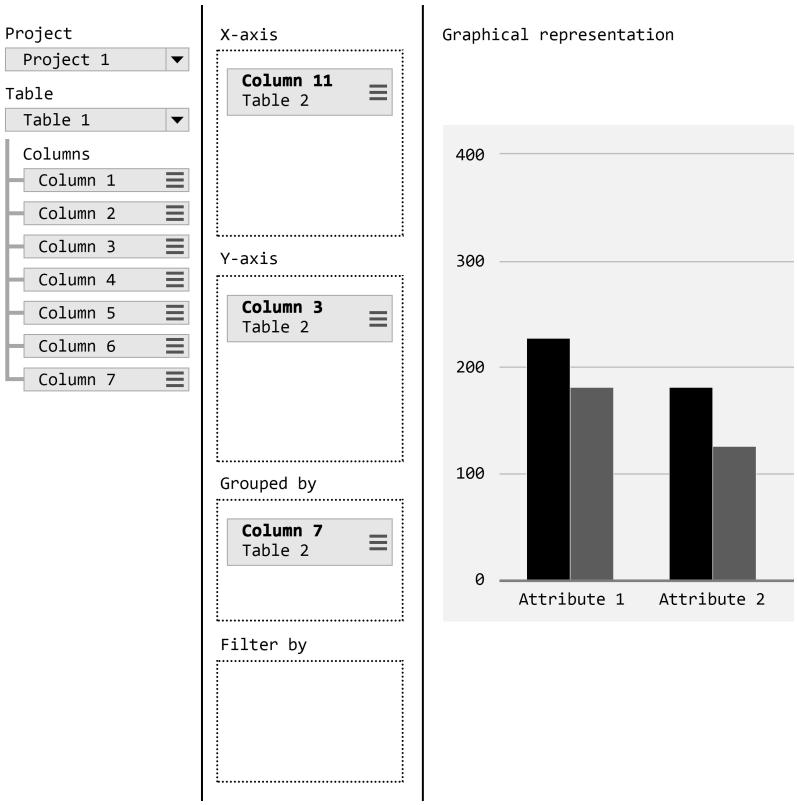


Figure 4.9: *Sketch of the Static Content Approach.*

display the user input and the data output on the same screen that results in more overview for the users.

- **Disadvantages:**
  This approach is absolutely not dynamic, only predefined SQL queries are supported, no extensions can be made by the users. To provide new or more complex data compositions it is required to implement them from scratch and directly integrate them into to the application. Without this implementation most of the specific analyses that are done by scientists are not possible.

- **Summary:**
  The Static Content Approach is not applicable to our ANALYSIS TOOL due to the lack of enabling any composition of data. Indeed, complex data compositions are possible, but every query has to be defined in the application. Even though our application is an open source project and theoretical everyone can implement the function in the code, it violates our requirement to be able of realizing data compositions without the knowledge of SQL or programming.

4.2.3 User Study: Paper Prototype

After the rating of the single possibilities we decided not to consider the Puzzle Pieces Approach (cf. Chapter 4.2.2) and the Static Content Approach (cf. Chapter 4.2.2) anymore. These two approaches are not applicable to our application due to the mentioned disadvantages.

Both, the Box Pieces Approach (cf. Chapter 4.2.2) and the Directed Graph Approach (cf. Chapter 4.2.2) are of interest for our work and should be further evaluated. We involved the actual users of the XBOOK databases to get the feedback of zooarchaeologists about the graphical composition of data.

We turned to the *ArchaeoBioCenter*[18] whose members are familiar with the XBOOK framework. We especially contacted employees of the *Institute of Palaeoanatomy, Domestication Research and History of Veterinary Medicine*[19] and the *Bavarian State Collection for Anthropology and Palaeoanatomy Munich*[20]. They are accustomed to the work with OSSOBOOK [KLK+18b]. We expected some new perceptions from the zooarchaeological point of view.

We created paper prototypes for each of the two approaches which the participants should use to compose the single elements. They were able to place, label, move, and remove them. The connections of the Directed Graph Approach were realized with a string of wool. Together with some domain experts we defined a set of realistic queries for archaeological compositions that the candidates should solve.

As a result of the observation of the participants and their feedback we registered useful facts. Below we listed the most significant ones:

- Most of the participants had difficulties in finding out how to enter any settings in the Box Pieces Approach. They were confused that they had to open them by double-clicking the box. This mechanic was not intuitive to them because there is no indication for this functionality.

- The participants preferred the compactness of the Box Pieces Approach. The single boxes of the Directed Graph Approach requires a lot of space and makes a compact arrangement difficult.

- At the same time the participants liked the visible information of the Directed Graph Approach because all setting rows and their inputs are displayed in the box. The Box Pieces Approach always requires another action to open and view the setting information that interrupt the workflow.

[18] http://www.archaeobiocenter.uni-muenchen.de
[19] http://www.palaeo.vetmed.uni-muenchen.de
[20] http://www.sapm.mwn.de

- The majority of the participants preferred the element arrangement of the Directed Graph Approach. They can reposition the single boxes and connect them afterwards with the connection lines without the box changing its position. The boxes in the Box Pieces Approach have always to be docked to another box which makes the definition of subqueries difficult.

- The participants requested a possibility to display an intermediate result of the current data.

Altogether, the larger majority favor the Directed Graph Approach, although they would prefer a more space saving display.

4.2.4 Revision of Approaches and Prototype

After the evaluation of the user study (cf. Chapter 4.2.3), we revised the named approaches and introduced the constructive feedback from the participants. We implemented a prototype that includes some adjustments of the approaches.

The Visualization Adjustment

Below we list the most important adjustments that were the results of the evaluation of Chapter 4.2.3.

- **Directed Graph Approach:**
  We decided to use the Directed Graph Approach as a basic visualization concept because the majority of the participants preferred working with the graphs instead of the Box Pieces Approach. Furthermore, it is also a great advantage to be able to connect one data output with more than only one other data input to avoid unnecessary redefinitions.
- **Outsourcing of Settings:**
  Due to the increasing complexity of extensive data compositions the single setting rows were removed from the boxes and were outsourced to an own pop-up panel. The input and output rows remain at the same position.
- **Settings Panel:**
  In the Box Piece Approach it was not intuitive enough for the users to open the settings by double-clicking on the box itself. It would be the same problem in the Directed Graph Approach as well by outsourcing the settings to another panel. That is the reason why the possibility of opening the settings is now indicated with a button in the box (cf. Chapter 4.2.4).

- **Improved Display of Information:**
  It became necessary to improve the display of information about the boxes and the graph in general since the settings are not visible in the box anymore – they are hidden inside the panel that is opened by the settings button. We added a text label that can be labeled with a custom text in the settings. If the users want to, they can enter a short information text there that can help them navigating through the graph.
- **View current data:**
  We have added a second button to the boxes that generates the current box data. This helps the users keep track of the current data. This data is displayed as a table and is always up-to-date when it is loaded by the users. This element might also be a good debug method for possible errors.

The sketch of the final concept of the Modified Directed Graph can be viewed in Figure 4.10.

Introduction of a Settings Panel

Additionally to the graph, it was now necessary to create a panel for the settings. We do not have any space limitations anymore because the content does not have to fit into the small boxes. The inputs for the settings can now be entered in an own panel.

So we designed a column layout for this purpose that is illustrated in Figure 4.11. Every setting that was displayed as a single row within the box before is now represented as an own column which each has scrollable contents. This has several advantages:

- The layout of the setting panel is still well structured as it was before with the single rows. The major difference is the layout orientation: The former rows were orientated from top to bottom, the new columns are orientated from left to right.
- The setting property elements need not to be embedded into another interface element. It is now an own panel that provides large amount of free space for graphical user interface components because scrollbars are now reasonably usable. This enables more complex and more customizable settings.
- The data input has not to be as space-saving as possible. Now, the originally planned input fields can be extended and are clearer. For example, we can now select multiple columns without having to select all desired columns in a combo box and have to add them to a list. We are now able to display all column options directly in the panel as single check box values the users can select.
- There is enough space to add additional information to the settings. This can be realized by adding text labels where needed without having to use non-visible elements like pop-ups or tooltips.

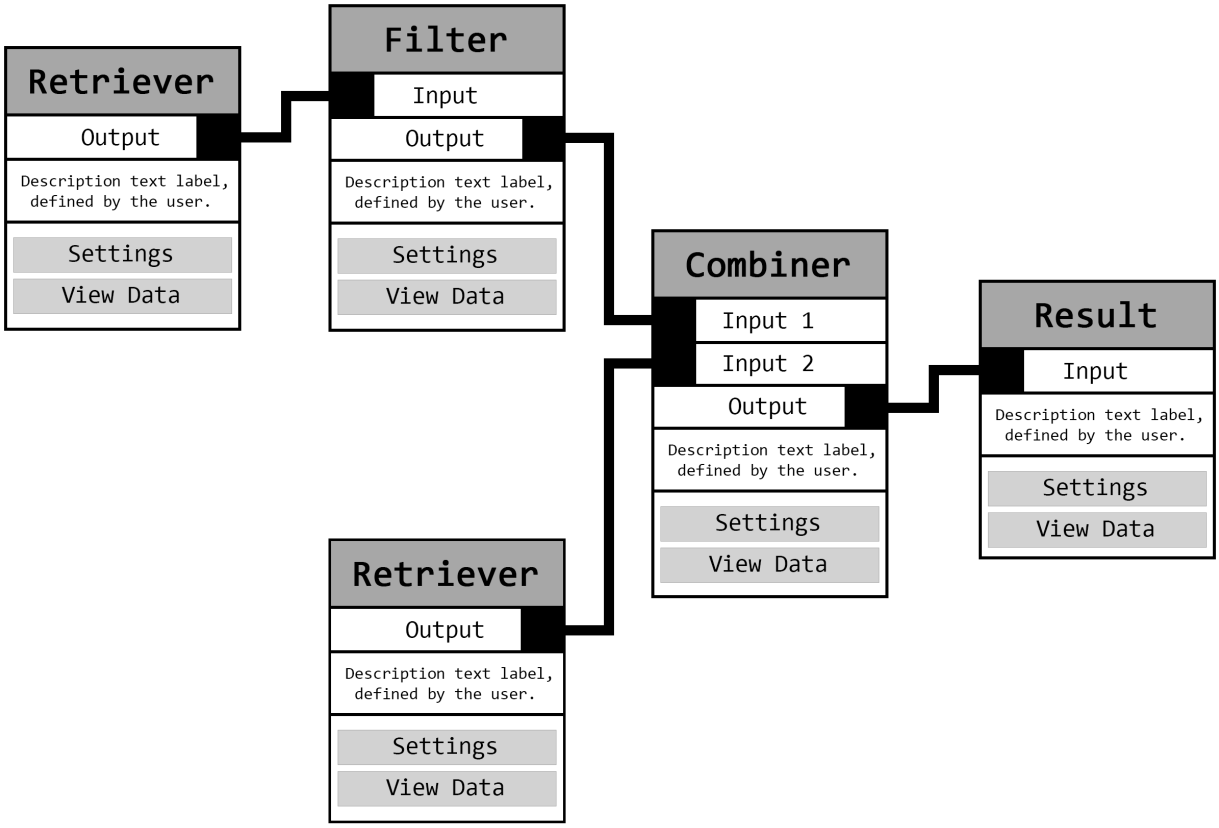


Figure 4.10: *Sketch of the modified Directed Graph Approach after the evaluation of the user study.*

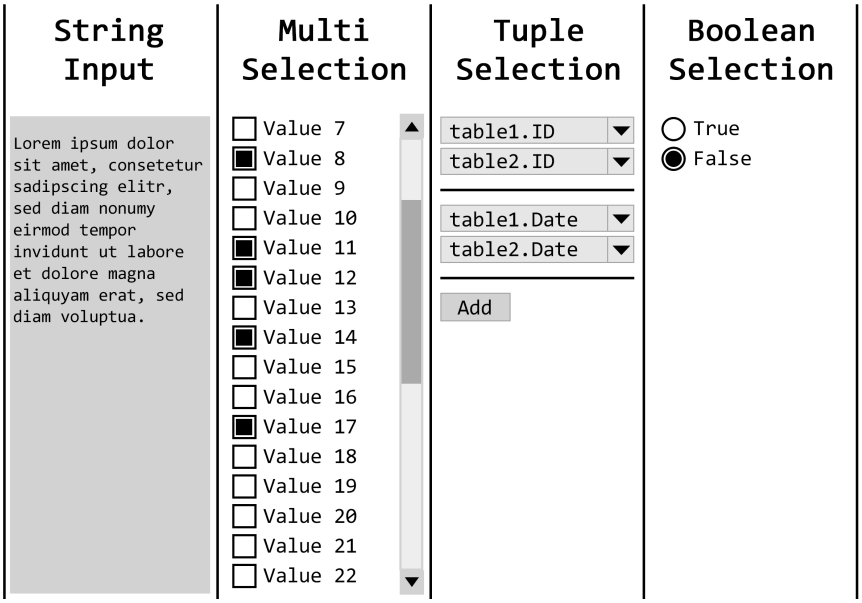


Figure 4.11: *Sketch of the Settings Panel. The visualized columns are four examples of possible setting properties.*

4.3 Workflow of Composing Analyses

We discussed the structure required for the ANALYSIS TOOL and also gave examples of the most basic Workers. In the next step, we want to put the pieces together and describe how a typical analysis is built. We will show two examples: The first one is a very simple, but still frequently used analysis to get the distribution of animals. The second one is a bit more complex, deliberately abstract, and requires a custom Worker to be implemented.

For the examples, we take a sample table with the columns "*ID*", "*Animal*", "*Location*", "*Country*", and "*Size*" as shown in Table 4.1.

ID	Animal	Location	Country	Size	...
1	Kangaroo	Melbourne	Australia	105	...
2	Kiwi	Auckland	New Zealand	37	...
3	Flying fox	Sydney	Australia	71	...
4	Dolphin	Lisbon	Portugal	227	...
5	Wombat	Melbourne	Australia	79	...
6	White Ibis	Sydney	Australia	71	...
7	Mouse	Los Angeles	United States	11	...
8	Dolphin	Auckland	New Zealand	210	...
9	Kangaroo	Sydney	Australia	127	...
10	Penguin	Melbourne	Australia	28	...
11	Turtle	Guarujá	Brazil	43	...
12	Emu	Melbourne	Australia	169	...
...	...	...	...	...	...

Table 4.1: *Sample Data: Appearance of Animals*

Of course, in real scientific databases a table like that would contain several more columns and data sets. Because of reasons of clearness we reduce the sample data to a minimum.

4.3.1 Example: A basic analysis

In the first example, we want to calculate a distribution of animals in Australia and create a graphical representation of the data as a bar chart. The composition of the data in the ANALYSIS TOOL is shown in Figure 4.12.

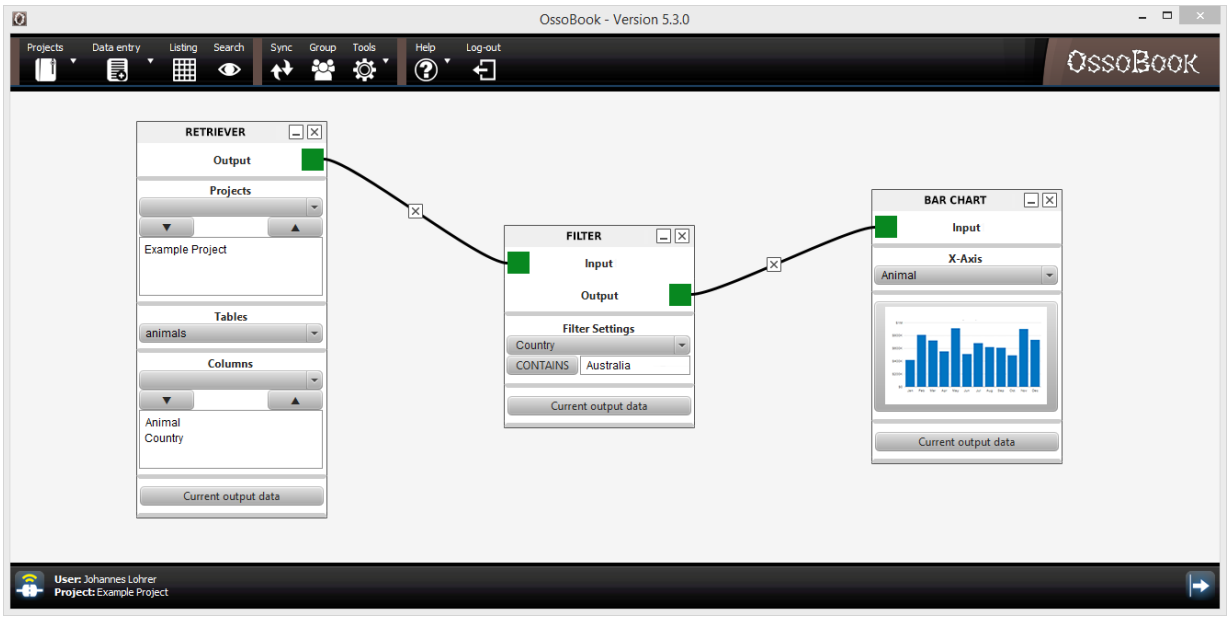


Figure 4.12: *Composition of the analysis for animal distribution in Australia, as described in Chapter 4.3.1.*

The analyzing process can be divided into three different steps that are described below:

1. **Planning and collection of data:**
   Gathering all required data from different sources and combining it to be able to use them for further processing.

   First, we identify the fields we need in our analysis. These are: "*Animal*" and "*Country*". The fields "*ID*", "*Location*", and "*Size*" are not important for our goal and can be disregarded.

   A Retriever to get the necessary data is the first Worker that we use. We configure it to get the fields "*Animal*" and "*Country*" only.

2. **Processing data:**
   The collected data is processed to remove unimportant data or structure the data, e.g. by filtering or sorting it.

   Next, we only want to filter all animals from Australia. Therefore, we add a Filter. The input of the Filter is connected to the output of the Retriever which makes the fields "*Animal*" and "*Country*" available as options for the Filter. There we select "*Country*" and define only to allow data sets in which the "*Country*" value equals the term "Australia".

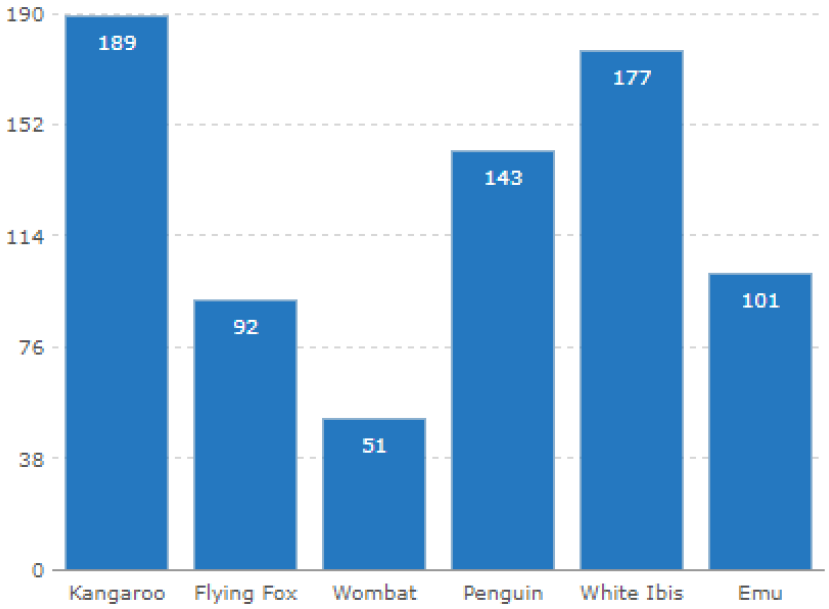


Figure 4.13: *Sample result for the animal distribution in Australia, as described in Chapter 4.3.1 and composited in Figure 4.12.*

3. **Generating result:**
   The processed data is used to display a result like a diagram or any other visualization.

   The last Worker is a Diagram. It takes the filtered data from the Filter and displays a graphical representation of it. In our case, we choose the bar chart. We can now select the column which values we want to use as the x-axis. We select the "*Animal*" column and the Worker generates the chart as seen in Figure 4.13.

4.3.2 Example: A more complex analysis

Now, we want to calculate the most common animal related to the average temperature in our sample data. For this we take data from Table 4.1 and introduce a new table with the average temperatures of the locations, as shown in Table 4.2.

ID	Location	Country	Avg. Temperature
1	Guarujá	Brazil	21
2	Los Angeles	United States	19
3	Melbourne	Australia	20
4	Lisbon	Portugal	17
5	Auckland	New Zealand	15
6	Sydney	Australia	18

Table 4.2: *Sample Data: Average Temperatures*

Again, we go through the three analyzing process steps to calculate the most common animal to the average temperature:

1. **Planning and collection of data:**
   Similar to the example before, we start with a Retriever that returns the fields "*Animal*" and "*Location*" of the table "*Animals*". Additionally, we need a Retriever that returns the fields "*Location*" and "*Avg. Temperature*" from the table "Average Temperatures".

2. **Processing data:**
   Now we only want to get the most common animal per location. Therefore, we have to create a custom Worker. This Worker accepts only one input and provides one output. The Worker consists of two `ComboProperty` elements. The first one defines the column for which the values are aggregated. The second one defines the column for which the most frequently occurring element is searched for, depending on the value selected in the first Property.

   In our example, we select "*Location*" as the column to be aggregated and "*Animal*" as the column for which we want to calculate the most common element. Then it returns a `DataList` containing only the two selected columns with the aggregated values to the output.

   We use the Combiner to merge the two `DataLists` with the grouped values and the temperatures. In it, we define the two "*Location*" columns as the key. The Combiner joins the two lists into one `DataList` with the columns "*Animal*", "*Location*", and "*Avg. Temperature*".

   As a demonstration, we want the result to be ordered by temperature. Therefore, we need the Sorter as another Worker. The Sorter uses the comparator of the selected column which would be a numeric comparison in this case.

3. **Generating result:**
   To display the results, we again use a Diagram. This time we us a scatter chart. We select the "*Animal*" column as the x-axes, and the "*Avg. Temperature*" column as the y-axis.

   Since the y-axis can only display numeric values, only columns are valid for the y-axis that are at least in interval scale.

The composition of the analysis can be seen in the screenshot of Figure 4.14. The result, visualized as a scatter chart, is shown in Figure 4.15.

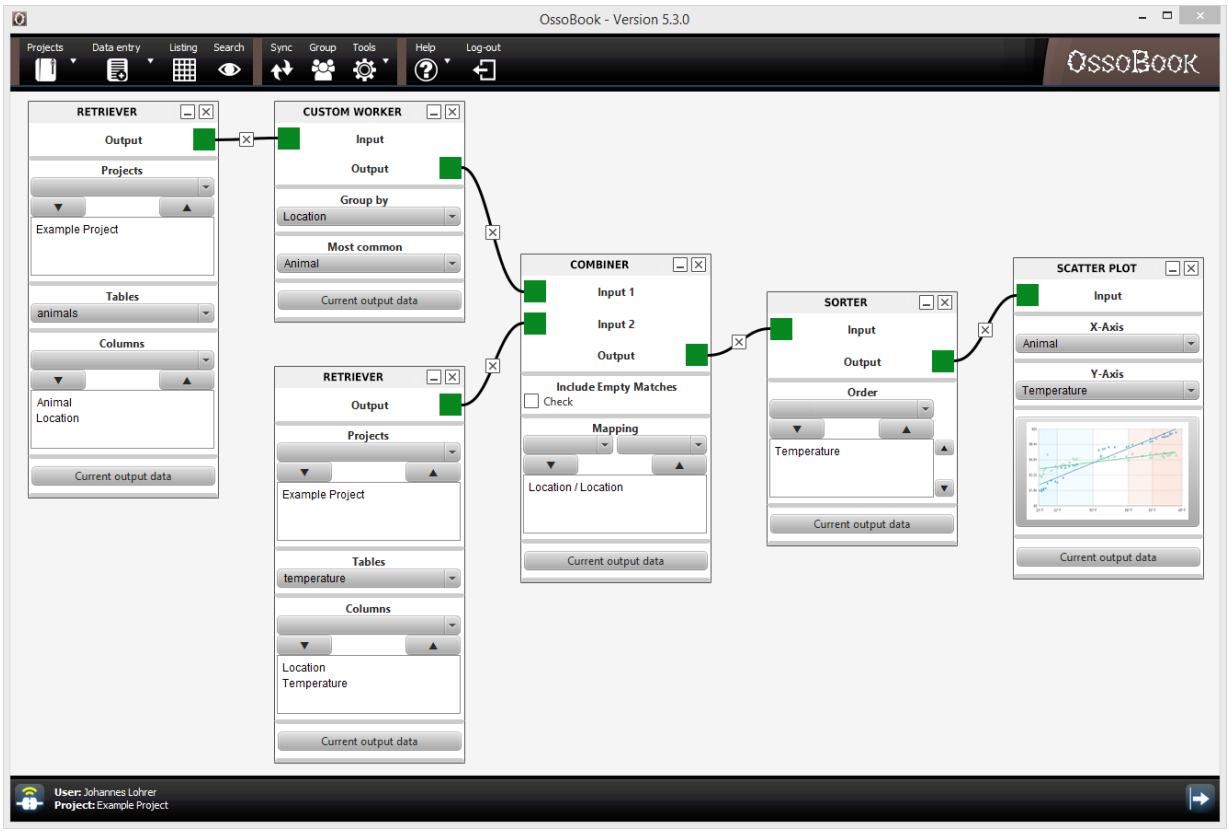


Figure 4.14: *Composition of the analysis for the most common animal, dependent on the average temperature, as described in Chapter 4.3.2.*

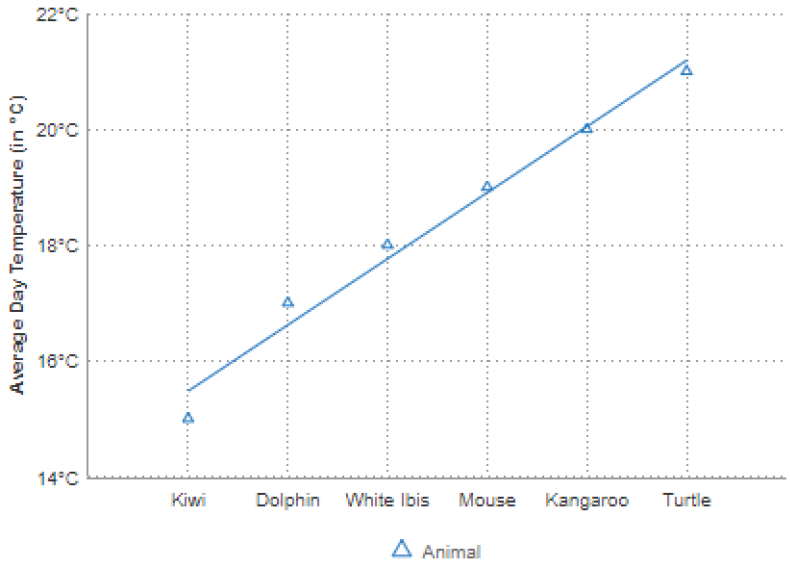


Figure 4.15: *Sample result for the most common animal, dependent on the average temperature, as described in Chapter 4.3.2 and composited in Figure 4.14.*

4.4 Analysis Tool Prototype

We now have a concept that we want to use for the realization of the ANALYSIS TOOL and apply it to real zooarchaeological data of OSSOBOOK.

4.4.1 Prototype Implementation

The ANALYSIS TOOL is implemented in Java using JavaFX as an extensible framework that can be embedded to any other base application. In this use case we integrate it into XBOOK and embed it to OSSOBOOK.

The left sidebar contains buttons representing all available modules. By clicking a button a new module is created and the corresponding box is added to the main content.

Each box is movable by using drag and drop gestures and can be removed from the working panel again by pressing the cross symbol in the top-right corner.

All boxes have the same basic layout. On the top there is the title line of the Box that describes the type of the modules, e.g. Retriever, Combiner, etc. Below the input and output rows are arranged. Two buttons are located in the last row of each box: One to open the setting panel for the box and one to open (if available) the current output data of the specific box in table form. Additionally, a description text is displayed in the box that can be used to comment or describe the logic of the composition.

The single boxes can be connected by using the input and output rows. This can be done by first pressing the connecting area of an input row and then the connecting area of an output row of another box, or the other way around. The connection of these two boxes is visualized with a line. Any connection can be removed again by clicking the cross symbol.

The setting panel is opened by clicking the “Settings” button of the corresponding box. A new panel will open that displays the available settings in a column layout.

4.4.2 Simple Example with Real Zooarchaeological Data

The prototype is applied to real zooarchaeological data in OSSOBOOK. As a simple example, we want to investigate the distribution of animal bones from an excavation site. We want to determine the distribution of horses or mules, cattle, sheep or goat, domestic pig, and chicken. Furthermore, we only consider findings of the 1st century AD.

The composition of the query can be viewed in the screenshot of Figure 4.16. In detail, we use these modules:

- **Retriever I:**
  Retrieves all data rows from the main input table (called “inputunit” in OSSOBOOK that holds most of the data of each entry).

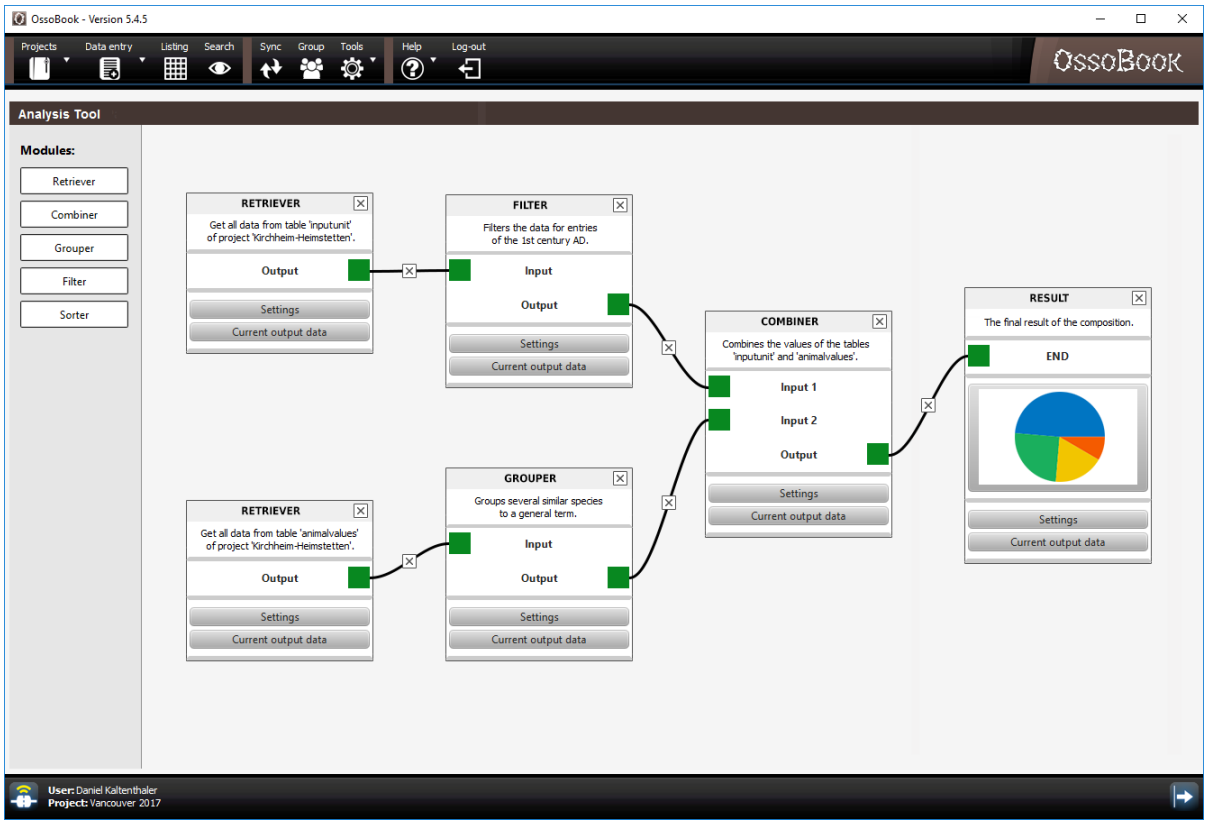


Figure 4.16: *Screenshot of the* ANALYSIS TOOL*, embedded into the* OSSOBOOK *environment. The displayed graph represents the data composition for the analysis described in Chapter 4.4.2.*

- **Filter:**
  Removes all rows of the result data from Retriever I that do not match the filter settings. These are set to the value "1^{st} century AD" in the column "Absolute dating".

- **Retriever II:**
  Gets all data rows from the table "animalvalue", an additional table that stores the saved animal values of a specific entry. This enables saving multiple values for one entry in case that the distinct determination is not possible.

- **Grouper:**
  Groups several animal values from the result data from Retriever II to a new value, e.g. the values "Ovis aries", "Capra hircus", and "Ovis aries / Capra hircus" are grouped to a new value "Sheep / Goat". Similar groupings are done for "Horse / Mule" and "other". "Cattle" and "Domestic Pig" are unique.

- **Combiner:**
  Combines the rows of the result data from the Filter and the Grouper based on the

common fields “ID” and “DatabaseNumber” which represent the primary keys in these tables. The Combiner is set to remove all matches where is no match of the columns in both tables (using an `INNER JOIN`).

- **Result:**
  Defines the output of the Combiner as the finished composition.

As a final result, we get the aimed data in table view that can be used for further processing.

4.4.3 Graphical Representation of the Result

Besides the table view the result can also be displayed as a graphical representation. The ANALYSIS TOOL enables generating different types of diagrams to display the result which the scientists can use to evaluate the data.

Each type of diagram has defined its required and optional inputs, for example which data should be set to the specific axes of the diagrams. The input data can be retrieved via an interface that provides the different types of data from the result table from the Result Module. The interface also prepares common functions, for example returning the number of rows that matches a specified criteria. The scientific users can then set the provided data information to the diagram, dependent on the purpose of the evaluation.

The logical composition of Chapter 4.4.2 was already applied to an analysis on data of an excavation from Heimstetten, Germany, but it was evaluated manually. It is part of a discussion about the development of livestock in the foothills of the Alps during the transition from the Celts to the Romans [TP15]. The manual creation of the graphical representation as a pie chart can be viewed in Figure 4.17 (left).

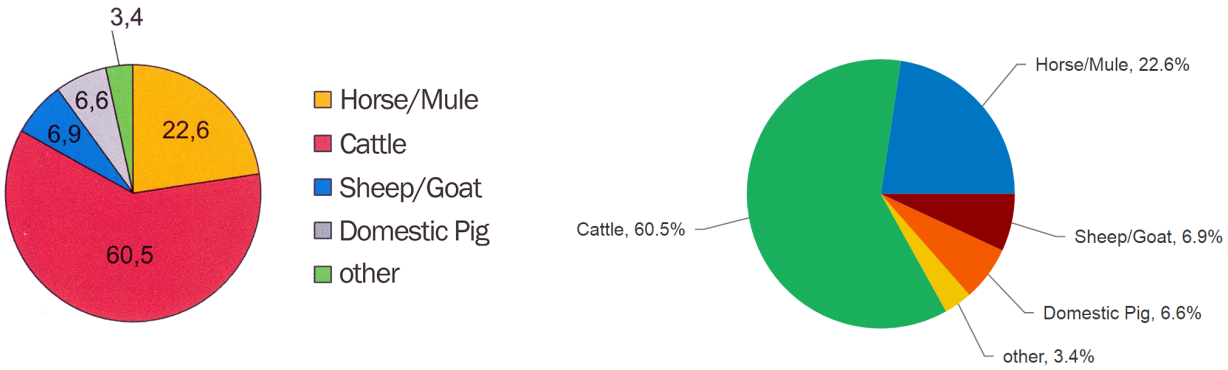


Figure 4.17: *Percentage of the most important livestock in settlements of the* Heimstettener Gruppe *(Heimstetten, Germany) in the 1st century AD, based on the number of bones [TP15]. The chart on the left is created manually, the chart on the right is generated within the* ANALYSIS TOOL *in* OSSOBOOK *showing the same result.*

The evaluated data is also available in the project "Kirchheim-Heimstetten" of OSSOBOOK. That allows us to analyze the data again, but now we use the embedded ANALYSIS TOOL. We use the method described above in Chapter 4.4.2 again. We evaluated the real data of the project and generated a pie chart that can be viewed in Figure 4.17 (right).

Comparing the chart with the original and manually executed evaluation, we have obtained the same result. Clearly, we can generate the same results in much less time without having to use extern spreadsheet applications.

Chapter 5

A Visual Analytics System for Spatial and Temporal Data

Attribution

This Chapter uses material from the following publication:

- Daniel Kaltenthaler, Johannes-Y. Lohrer, Ptolemaios Paxinos, Daniel Hämmerle, Henriette Obermaier, and Peer Kröger. TaRDIS, a Visual Analytics System for Spatial and Temporal Data in Archaeo-related Disciplines. In *13th IEEE International Conference on e-Science, eScience 2017, Auckland, New Zealand, October 24-27, 2017*, pages 345–353, 2017. [KLP+17]

See Chapter 1.2 for a detailed overview of incorporated publications.

The task of archaeologists and bioarchaeologists consists of reconstructing and describing each situation of excavations as exact as possible. Therefore, not only the data of the findings is important for archaeo-related sciences, but also the spatial position and temporal information. It can easily happen that during an excavation findings from completely different epochs are revealed. The position of findings in chambers, graves, etc. gives an insight into the temporal circumstances of the excavation site – typically, findings from older times are found below findings from more recent times. The relationship of the spatial and temporal information of the archaeo-related data allows the reconstruction of history.

Unfortunately, in archaeo-related disciplines the combination of related spatial positions and temporal information of the findings is extremely labor-intensive and therefore, if at all, has to be regarded as an exceptional occurrence. This is due to two basic barriers that we describe as B1 and B2:

B1 Non-digital data:
The spatial information of excavations is usually not explicitly available in digitalized form – even though, most excavation databases contain the necessary data to reconstruct this information. The spatial context is usually prepared by hand in a time-consuming process resulting in drawings that represent the stratigraphy of the excavation site, as seen in Figure 5.1. These hand-painted drawing sheets of the stratigraphy do not include any enriched information about the available findings in the layers. For each question the position of each layer has to be set in relation manually. This is time-consuming and error-prone.

B2 Distributed data:
The necessary spatial data is – if it is available digitally at all – rarely saved in the same data source. For example for zooarchaeological analyses, spatial data is available in the OSSOBOOK [KLK+18b] database for many projects. However, the spatial information of excavations is not saved in the database since this archaeological information is not transmitted from the *Bavarian State Department of Monuments and Sites* to the zooarchaeologists. However, the spatial information is saved in the EXCABOOK [KLK+18f] database.

Especially the REMIS architecture (described in Chapter 3.2) strongly contributes to close the barrier B2: It provides the possibility to search for more information about specific excavation contexts or findings and brings together the spatial information from one database and the temporal information from another database. It is a fundamental basis that motivates to engage with analyses considering spatial and temporal information.

Archaeologists and bioarchaeologists would greatly benefit from an automatic process to generate explicit stratigraphical models that can be visualized and analyzed by

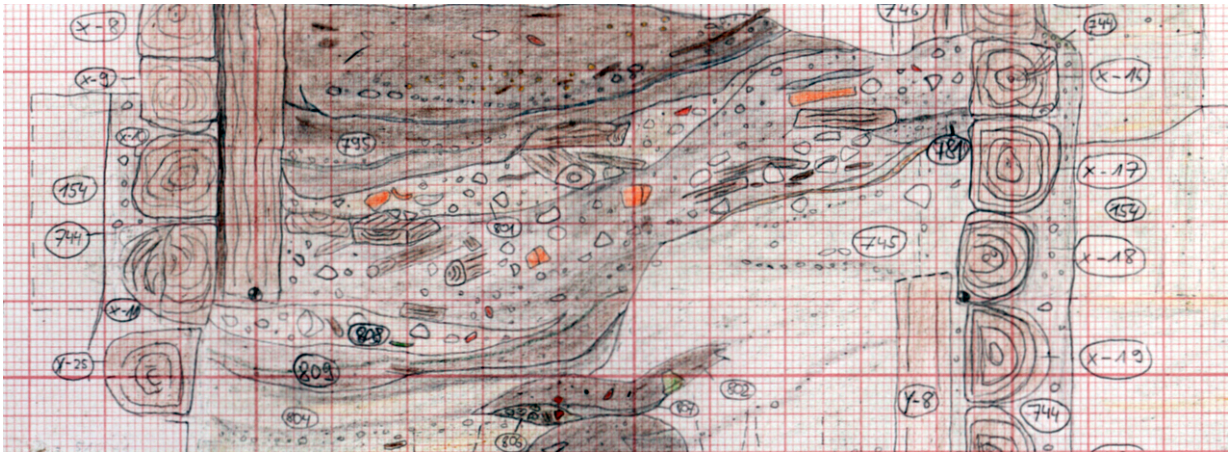


Figure 5.1: *An example for a hand-painted drawing sheet.*

dedicated tools. This would not only save processing time but would also allow further analyses to be carried out for which the manual process is too complex.

In this Chapter, we describe TARDIS (“*Temporal and Relative Diagram Interaction System*”), a tool that provides the possibility to present the extremely complex and multidimensional connections between place and dating as well as present development in time henceforth immensely simplifying the understanding of the connection of the data. It conveys an enormous amount of information in only a few descriptive and highly informative visualizations. TARDIS also features algorithms to make the implicit information hidden in common excavation databases explicit without a manual, time-consuming, and error-prone generation (e.g. drawing).

In summary, the main contributions are as follows: First of all, we explain some basic archaeological terms that are related to this work in Chapter 5.1. We discuss the background and the availability of data and the related work for this context in Chapter 5.2. Then we describe the structure of TARDIS and focus on the creation of the Harris Matrix as a key aspect in Chapter 5.3. Afterwards, we use TARDIS in a case study to analyze real data from an archaeological excavation to determine the distribution of faunal remains in Chapter 5.4.

5.1 Archaeological Terminology

For understandability and relevance reasons, we introduce definitions from the archaeological context concerning the presented work below.

Archaeological notions

Excavation and spatial documentation of archaeological findings are – amongst others – usually managed in terms of areas, sections, and layers. According to internationally agreed guidelines for the documentation of archaeological excavations, see exemplarily [Bay16b], these terms are defined as follows:

- **Layer**:
  Layers describe all structures that differ in color, consistence, and material of the directly neighbored structure, e.g. a monophased wall, a layer of earth, a discoloration of soil, but also a disruption. A layer is the smallest managed unit.

- **Area**:
  An area is a horizontal sector of the excavation that is defined by the archaeologist.

- **Section**:
  If required, single parts of an area can be split into sections.

Harris Matrix

The Harris Matrix[21] [Har75] [HIB91] is used to depict the temporal succession of archaeological contexts and thus the sequence of depositions and surfaces on an archaeological site. Two layers, i.e. their strata, can be set in temporal relationship dependent on their position.

All archaeological sites are subject to the laws of archaeological stratigraphy. Therefore, Harris [Har89] formulated a set of four basic laws for stratigraphy for the archaeological context:

- The **Law of Superposition** "assumes that the strata and layers are found in a position similar to that of their original deposition".
- The **Law of Original Horizontality** "assumes that strata, when forming, will tend towards the horizontal".
- The **Law of Original Continuity** bases "on the limited topographical extent of a deposit or an interfacial feature".
- The **Law of Stratigraphical Succession**: "A unit of archaeological stratification takes its place in the stratigraphic sequence of a site from its position between the undermost (or earliest) of the units which lie above it and the uppermost (or latest) of all the units which lie below it and with which the unit has a physical contact, all other superpositional relationships being redundant."

By applying these laws it is not necessary to sketch the relationships of each single layer to all the others. Instead it is sufficient to consider three basic relationships between them:

(A) Two layers are positioned on the same layer, but have no stratigraphic connection,

(B) two layers are positioned one above the other, and

(C) a layer is cut by another layer.

Harris also proposed a diagram that visualizes all stratigraphic relations. These visualizations are sketched in Figure 5.2. Harris called this diagram the Harris Matrix, even though this visualization is rather a diagram which is a different thing from a mathematical point of view. However, we use the term "Harris Matrix" anyway because it is the author's chosen term.

[21] http://www.harrismatrix.com

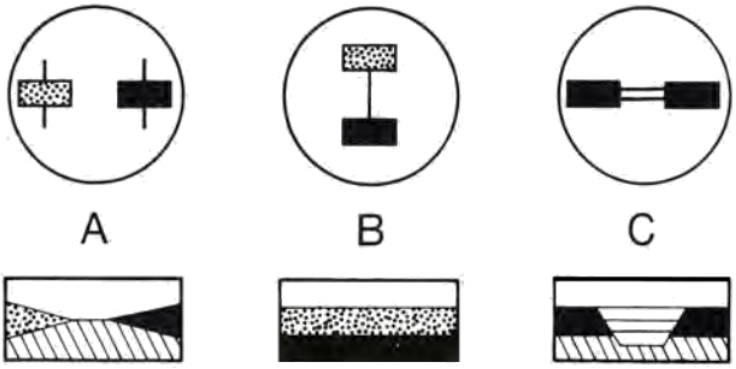


Figure 5.2: *"The Harris Matrix system recognizes only three relationships between units of archaeological stratification: (A)The units have no direct stratigraphic connection. (B)they are in superposition; and (C)the units are correlated as parts of a once-whole deposit or [layer] interface." [Har89]*

5.2 Related Work on Harris Matrices

In principle, our application TARDIS works on top of any excavation database providing implicit information on the spatial-temporal context of an excavation as described above. In this work, we will illustrate the capabilities of TARDIS on top of the database EXCABOOK running at the *Bavarian State Department of Monuments and Sites*[22]. EXCABOOK includes detailed information about the layers and the single findings from an excavation.

Each layer is saved as an own data set that is assigned to a specific position correlation information which includes – amongst others – the information for area and section. Within each layer, this information can be set in spatial relationship by defining the position of the layer in relation to other layers by defining which layer is positioned below/above another layer, which layer cuts/is cut by another layer, and which layers are disconnected, but identical. This information is sufficient to calculate the Harris Matrix [Har89] as described in Chapter 5.1.

In summary, temporal data is available in the archaeological data sets due to the definition of the stratigraphy, respectively to the temporal relation of single layers to each other. There is also available geodata for the data sets that can be taken from the spatial expansion and coordinates of areas and sections in excavations. This data is documented as precisely as possible, However, not all measurements have the same precision due to different measurement methods or level of accuracies. The data still can – at least – be used for a rough generation of a sketch of the excavation place.

The problems encountered from this context are the following:

[22] `http://www.blfd.bayern.de`

- **No considering of spatial and temporal relationship:**
  This spatial and temporal information is not set in relation in the archaeological field of work. The archaeologists do either consider the temporal or the spatial point of view. However, combining both of the aspects would result in the exploration of new scientific questions in the archaeological context.

- **No interactive Harris Matrix:**
  The visualization of the data as a Harris Matrix does not offer any dynamic interaction for the user. It is a static result that is achieved mostly by the manual composition of data.

- **Finding information is not considered:**
  The information about the single findings are not considered at all in the Harris Matrix. If a user wants only to display a filtered data set – e.g. only a specific species, bone element, sex, etc. – the input of the data has to be filtered manually before loading them to a potential application.

Since the first proposal of the Harris Matrix in 1975 [Har75], some tools were developed to support the creation of the matrix. A very first approach to generate a visualization of the data was already developed one year after the first publication of the Harris Matrix: *Strata* [BW76] assisted the creation of the Harris Matrix. However, it only generated the temporal positions of the layers, but do not consider their relationships. Interactivity and considering enriched archaeological data was not supported. A similar works is *Gnet* [Rya88] [Rya95] that do not consider all available relationships and do not support a printable output.

Furthermore, there are tools that allow the users to enter each layer and define the stratigraphy between them. Tools like the *Harris Matrix Composer* [Ima] or *ArchEd* [HMP+15] allow the setting of these relationships directly in the applications. However, extern data cannot be imported, therefore the data has to be entered to the tool for each context. Other tools, like *Stratify* [Her08] [Her06] [Her11], alternatively support the import of a CSV file or dBASE database to define these information. The entered layers are checked for errors. If no errors exist, the resulting Harris Matrix is then created. Still, these tools do not visualize the spatial position of the single layers. The actual findings inside these layers are also not considered in any way. In addition, none of these tools contain interactive elements.

Most often not all objects of an archaeological excavation site can be found. Natural circumstances (like trees) and man-made constructions (like buildings or walls), but also financial or time reasons, lead to selective or random excavations within one site. Therefore, statistical interpolation methods are used to get a representation of the distribution of objects on the excavation site. The one we focus in this work is the Kernel Density

Estimation that is commonly used in the archaeological field of work. Many tools exist for Kernel Density Estimation that allow the generation of a distribution either directly online [Wes15] or inside an application like MATLAB [Mat]. All existing tools offer various options to load data including CSV and database files. While it would be possible to filter the data by selected criteria, a 3D representation of the layers and the Harris Matrix cannot be displayed at the same time.

5.3 A Tool to Illustrate the Spatial and Temporal Distribution of Findings

In this Chapter, we introduce TARDIS, an application that supports the archaeologists in their data analyses concerning the challenges mentioned in Chapter 5.2. Its graphical user interface allows access to three major parts of the spatial-temporal context of the data from a given excavation. These includes

- a 2D representation of the site,
- a 3D representation of the layers, and
- a (graph-based) visualization of the Harris Matrix.

Furthermore, an area for the selection of the data and filter possibilities are added to the interface. Therefore, each of the three views mentioned above can be customized according to which data should be displayed. Below, we present the single areas of TARDIS in detail and describe the use of the elements which are shown in screenshot of the tool in Figure 5.3.

5.3.1 2D Representation

In TARDIS, the 2D representation sketches the rough layout of an excavation site as archaeologists are used to. Therefore, the single sections are plotted by considering the corresponding coordinates that are saved in the database. Currently EXCABOOK only allows the definition of rectangle and circular shapes to describe the dimension of sections. Thus, TARDIS currently only supports the representation of these shapes, too. In future, other shapes – like the definition and use of custom polygons – will be considered as well to leverage the flexible definition of the shape of single sections.

This rough layout of the excavation site is complemented by a visual representation of the distribution of findings which is indicated by different colors. Therefore, there are three possibilities for the representation of the available findings that are described below.

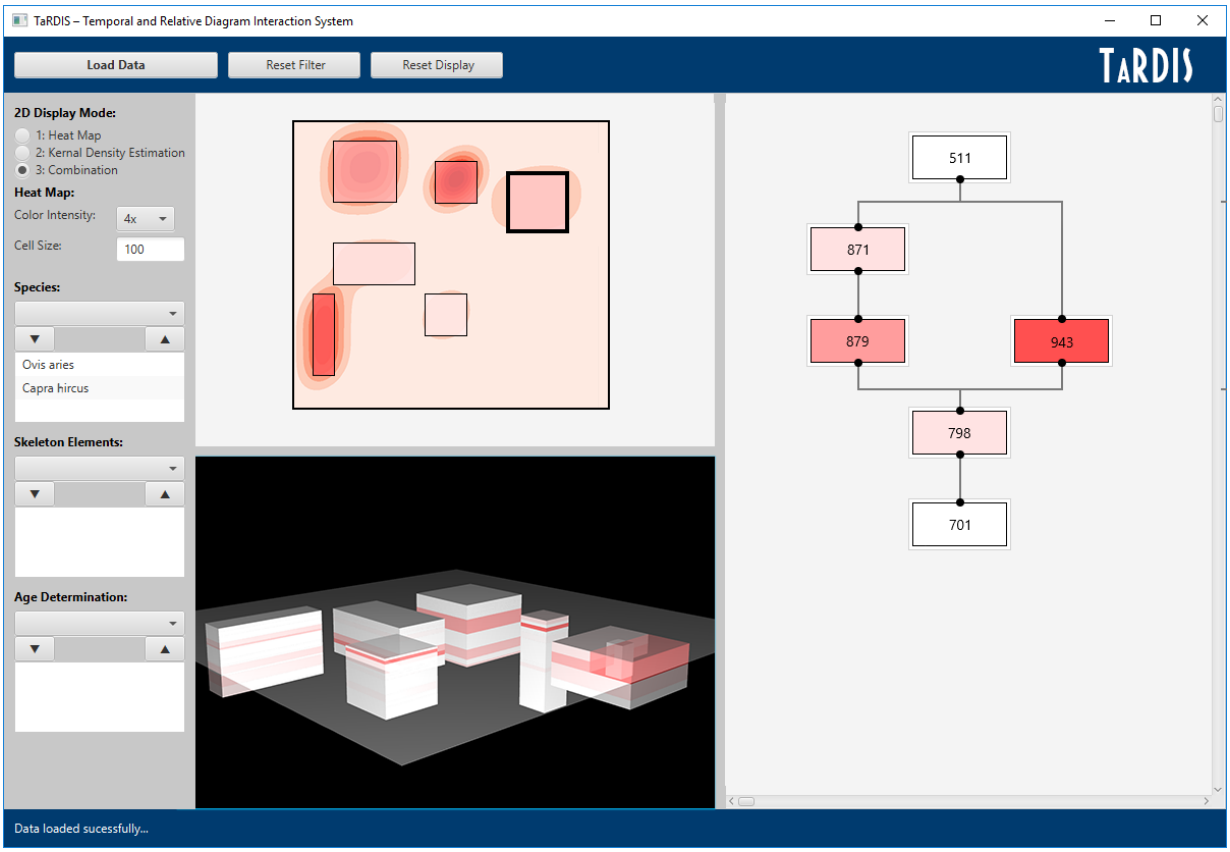


Figure 5.3: *Screenshot of the* TaRDIS *application. The 2D representation (top center) shows the areas and sections. It displays heat map colors for the absolute number of findings and the result of a Kernel Density Estimation in the background. The 3D representation (bottom center) shows the distribution of findings in the layers. The generated Harris Matrix is displayed on the right. Settings and filter options can be entered on the left.*

- **Mode 1: Heat Map Colors**
  This possibility allows the visualization of the actual findings in the database by means of a heat map. The drawn rectangles and circles in the 2D representation are colored dependent on the actual available data in the corresponding area or section. The stronger the color, the more data of findings exist in the area or section. Therefore, the colorization represents the absolute number of findings.

- **Mode 2: Kernel Density Estimation**
  The actual position of the findings is taken to calculate an estimation of the distribution of findings on the excavation place. We use a Kernel Density Estimation as a well-established statistical method. The graphical representation of the estimation is drawn to the 2D representation as an own layer behind the rectangles and boxes. The border of the sections, respectively the forms, are still displayed.

- **Mode 3: Combination**
  A combination of the previous modes allows the visualization of the absolute number of findings and the estimation. The background of the visualization shows the result of the Kernel Density Estimation, but the absolute number of findings is still indicated by a colorization of the drawn rectangles and circles. This combination can be seen in the screenshot of Figure 5.3.

The user can decide which data representation is displayed. The 2D representation of the excavation site also serves as a selection of the areas and sections which information shall be displayed as a Harris Matrix.

The 2D representation simulates the traditional way how archaeologists represent the spatial context of an excavation. However, the archaeologists usually did this by manually drawing (simply because no tool was available to support a fast digitalized data generation) so far. With TARDIS working on top of an excavation database like EXCABOOK, the generation of that representation is fully automatic (data is gathered from the underlying database). In addition, TARDIS now also allows the visual analysis of the spatial distribution of findings implementing several filters etc.

Thus, using TARDIS, archaeologists are now able to visually analyze the spatial relationships of findings in a large scale, e.g. considering different categories of findings, etc.

5.3.2 3D Representation

Similar to the 2D representation, the 3D representation does also sketch the layout of the excavation site, but it adds the depth information to the display. This a completely new feature to many archaeologists since the sketch of the 3D spatial context of an excavation is recorded not too often because it has to be done again manually so far which is very complex and time-consuming.

The x- and y-axis of the 3D representation is identical with the information of the 2D representation, but the z-axis was added to consider the height/depth information as well. These are taken from the data of layers automatically. However, for a realistic display of the stratigraphy the stored data of layers is not sufficient. Actually, only the rough dimension of a layer can be used. This can only be an approximation for the vertical value because a layer can have different thicknesses at each spot. According to this, the 3D representation is not to be seen as a detailed drawing, but as a sketch instead. However, the sketch provides a rough spatial impression of the distribution of findings for the archaeologists which is – for many projects/excavations – much more than previously available.

The distribution of findings can also be displayed in the 3D representation, but – in contrast to the 2D representation – for each single layer. Here, the stronger colorization

of a layer also indicates a higher frequency/density of findings. Therefore, the user gets an overview of the distribution of findings through the physical position of the stratigraphy. This is supported by the possibility to arbitrarily move through the 3D scene with mouse and keyboard.

Like the 2D representation, the 3D representation is a huge step ahead for the archaeologist when analyzing the spatial distribution of findings. While so far, most archaeologists are limited to 2D manual drawings that are not really flexible in terms of changing views through filters, etc., TARDIS now leverages visual analytics of the spatial context of an excavation at full scale.

5.3.3 Harris Matrix

In the context of an archaeological excavation, not only the spatial distribution of findings but also the temporal distributions and relationships of items are very important. As described in Chapter 5.1 the Harris Matrix displays the temporal relationships of different layers. Each single box of the Harris Matrix represents a layer from a specific area or section. The higher a box is displayed in the diagram, the more recent is the layer. Furthermore, an earlier layer can be recognized by a lower position in the diagram.

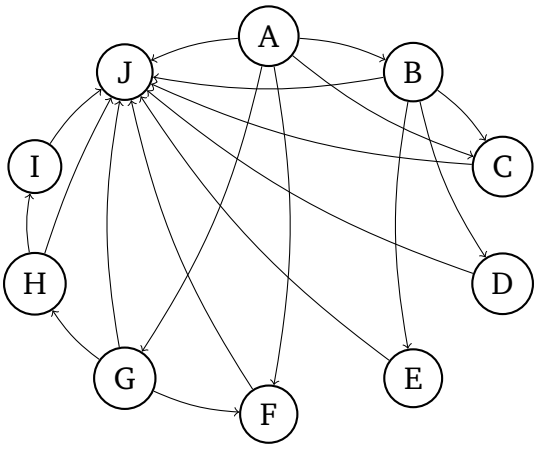
The Harris Matrix itself is an important representation of the temporal context of an excavation. However, since this representation is typically drawn manually for excavations so far, it is limited in the way it can be analyzed.

Thus, in TARDIS, the visualization of the Harris Matrix can be extended by displaying the distribution of findings over the elements of the diagram. By selecting a specific area in the 2D representation, the corresponding Harris Matrix is generated and displayed in the application. Each box in the Harris Matrix is highlighted with a color that represents the frequency/density of findings in this layer. In comparison to the 2D and 3D representation, the colorization in the Harris Matrix illustrates the temporal distribution of these findings. Again, appropriate filters allow the visualization of different categories, etc.

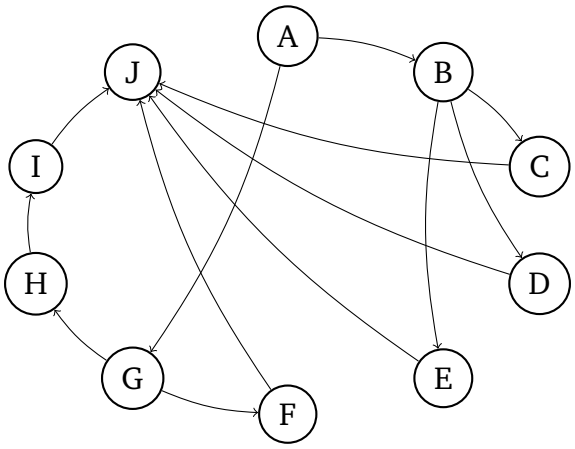
The Harris Matrix in TARDIS also features interactive elements. By selecting one box in the Harris Matrix the 3D representation will highlight the corresponding layer. Therefore, the archaeologist can identify the spatial position of the layer and compare it with the temporal position of the Harris Matrix.

Below, we describe the set of algorithms necessary to generate the Harris Matrix. The processing of the algorithms is illustrated in Figure 5.4.

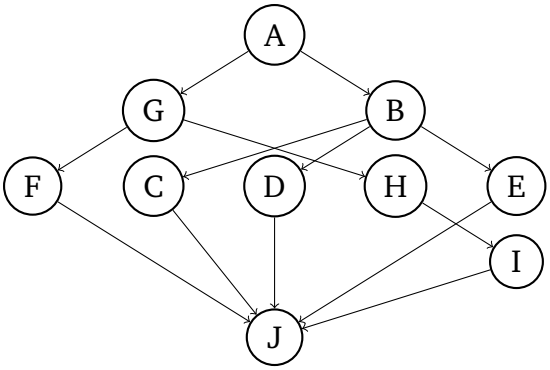
To generate a valid Harris Matrix we need suitable data. Each layer has to be connected to the network, either by being earlier, later, or in the same time as another from that data. Layers that are in the same time as other layers can be identified by having the same layers above and below. We start off by checking the available data (containing connections and names of layers) in the database by looking for loops and reporting



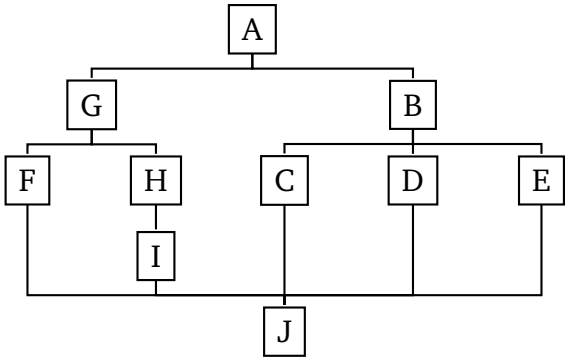
Step 1: Removal of loops from data



Step 2: Removal of redundant links



Step 3: Sorting for height of nodes



Step 4: Minimizing crossings

Figure 5.4: *Illustration of the algorithms described in Chapter 5.3.3 to create a Harris Matrix.*

them back so that they can be removed. This is done by checking if any layer is, via connections, later or earlier than itself. If so, this is a contradiction and indicates a loop that needs to be eliminated. Also in this step we can already assign depths to the nodes of the diagram. Algorithm 5 implements the identification of loops and the assignment of depths to nodes.

After we eliminated the loops, we eliminate redundant links. This is important to get a Harris Matrix that explains the excavations temporal relations with as little connections as possible. Otherwise, redundant nodes and connections produce a possibly huge overhead and an excessive visualization that is hard to comprehend. To ensure we do not have any links that are not necessary, we check each direct link if we can reach the same layer transitivity. If this is possible, we remove the direct link. Algorithm 6 implements this step.

```
1  topnode ← all nodes without incoming connection(s);

2  foreach topnode do
3  |  assign height to node;
4  |  foreach node below do
5  |  |  goDown(height++, nodeID);
6  |  end
7  |  if not all nodes have a height then
8  |  |  throw error found nodes without connection;
9  |  end
10 end

11 def goDown(height: int, node: Node) : void:
12 |  if height > nodes.length then
13 |  |  throw error loop in data;
14 |  end
15 |  if height(nodeID) < height then
16 |  |  assign height to node;
17 |  end
18 |  foreach node below do
19 |  |  goDown(height++, nodeID);
20 |  end
21 end
```

Algorithm 5: *Checking connectedness and loops*

Then we need to determine the height and the width the Harris Matrix will need. In this process we have also to take into consideration whether there will be a connection going through to a lower height but does not have a layer at that specific height. This is needed so that our Harris Matrix is clear and all connections are visible. This process is implemented in Algorithm 7.

Once we have the height and width of the Harris Matrix, we can start sorting the layers in a manner that requires as little path crossing as possible. For this purpose we sort the layers below in a way that they lie underneath the layer above and, if needed, in the right position if they are below multiple layers. In the sorting process we also need to consider empty spaces which are required if a connection stretches above multiple layers of depth without an actual layer in them. The final preparation of the data is implemented in Algorithm 8.

After we have the data well prepared we can then finish the generation of the Harris Matrix by drawing the layers and their connections.

```
1 foreach node do
2 |  if there are multiple connections downwards then
3 |  |  remove one and save ID;
4 |  |  follow the other path(s) downward;
5 |  |  if ID is not encountered then
6 |  |  |  reinsert ID;
7 |  |  end
8 |  end
9 end
```

Algorithm 6: *Check for redundant links*

```
1  maxheight ← max(height);
2  sort nodes according to height;
3  start from top do
4  |  columnwidth ← 0;
5  |  check nodes one layer above for links to nodes below that are not in this
   |   height;
6  |  foreach of those do
7  |  |  increase columnwidth by 1;
8  |  end
9  |  foreach in this height do
10 |  |  increase columnwidth by 1;
11 |  end
12 |  maxcolumn ← max(columnwidth, maxcolumn);
13 end
```

Algorithm 7: *Sort for height of nodes*

```
1  insert first layer into Harris Diagram in random order;
2  foreach layer below do
3  |  add nodes in the order that they are referenced above;
4  |  if nodes are not in this layer then
5  |  |  add a filler;
6  |  end
7  |  if nodes are referenced in multiple nodes above then
8  |  |  put these nodes in the middle;
9  |  end
10 end
```

Algorithm 8: *Sort nodes to minimize crossings*

5.3.4 Filter Options

In general, the displayed distribution of findings is not filtered. The colorization of the elements in the 2D and 3D representation and in the Harris Matrix is reflecting the distribution of all available findings, independent on their diversity.

Still, TARDIS provides some basic filter settings that can be applied to the loaded data, e.g. – in the case of zooarchaeological studies – filter for specific species, skeleton elements, or age determinations. Once a filter option is set, the colorization in the visualizations only considers the findings matching the filter settings.

This allows multiple views on the spatial-temporal distribution of findings within an excavation and leverages visual analyses at large scale that enables archaeologists to find insights on a completely new level.

5.4 Case Study: Distribution of Faunal Remains

To prove and demonstrate the usefulness of the tool as well as the quick and efficient way to show the distribution of animal species in the different features of an archaeological excavation, we run a case study on real archaeological data. Our goal is to compare the spatial and temporal distribution of animal bones excavated in several shafts, not only within the features, but between them as well.

We took data of the excavation Marienhof-Haltepunkt in Munich, Germany, excavated by the archaeological excavation company ReVe during the years 2011 and 2012. The major part of the 110 $\times$ 95 m excavated area lies within the oldest city core from the 12^{th} century whereas the northern peripheral zone is a part of city's expansion from the late 13^{th} and early 14^{th} century. Marienhof-Haltepunkt is a promising study object, due to the good condition of the animal remains as well as of the shafts themselves. The fact that flotation of material was practiced during the excavation is a great benefit for the zooarchaeologists. Thereby, smaller and more fragile bones were able to be recovered. This results not only in a higher amount of different species (e.g. rodents and small birds), but also of age groups which are usually underrepresented, especially the age groups "fetal", "neonat", and "infantile".

We chose shaft 5 of the excavation that is illustrated in the drawing sheet of Figure 5.5 and visualized the data of the shaft in the TARDIS application – the distribution of all (unfiltered) species is shown in Figure 5.6-A. Shaft 5 was constructed as an elaborate well in 1261 [Wes14], but it was abandoned short after and functioned as a latrine. Such an elaborately built well must have belonged to a wealthy owner.

The bones of frog species depict a good example of a result due to a carefully done excavation. Two distinct species of frogs, *Rana temporaria* (the common frog) and *Pelophylax lessonae/ridibunda* (the pool/marsh frog), could be identified, although the majority

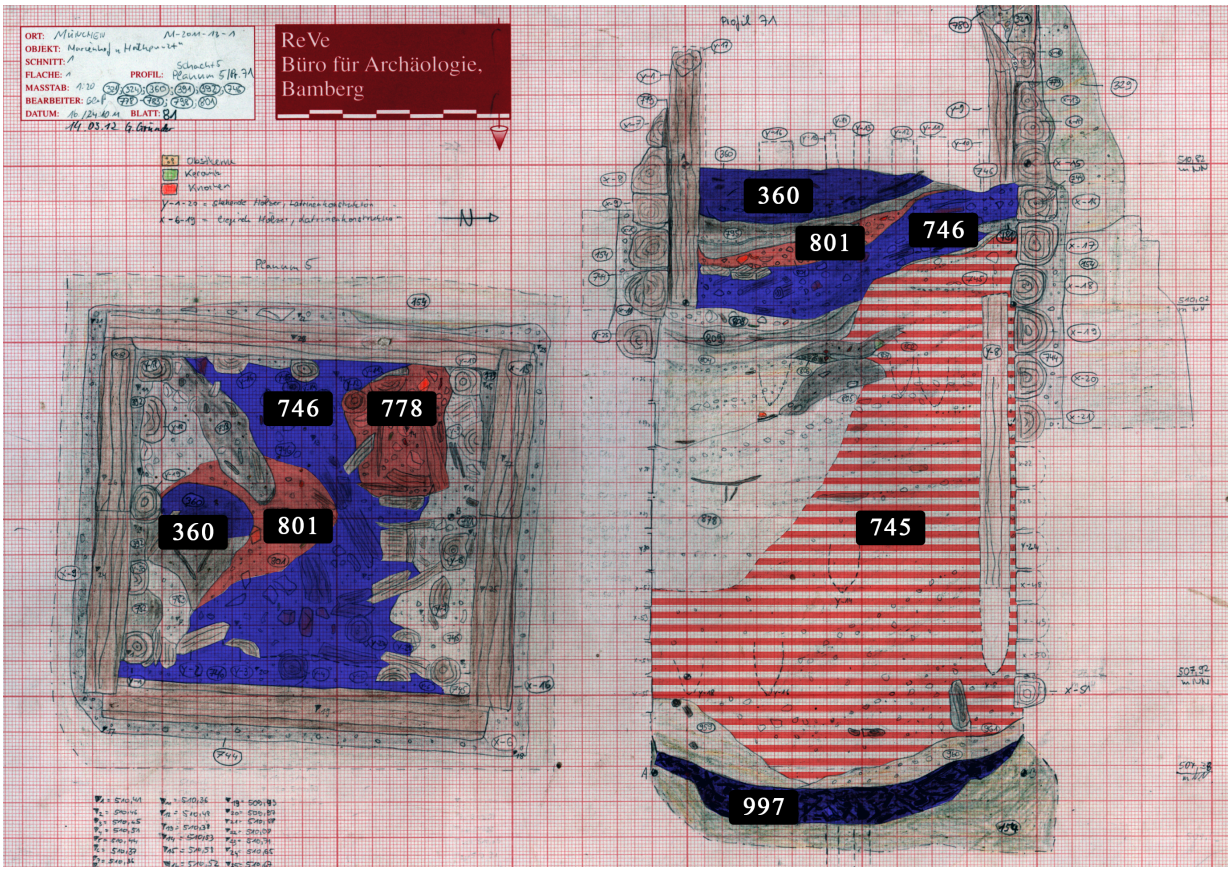


Figure 5.5: *Planum (left) and profile (right) drawing of shaft 5 of the excavation Marienhof-Haltepunkt in Munich, Germany. The layers 360, 745, 746, 778, 801, and 997, that are mentioned in Chapter 5.4, are highlighted. The mentioned layer 393 is not visible in the drawing.*

of the frog bones could be identified only to the family level (*Ranidae*). In the TARDIS visualization we filtered the available data for frogs (as seen in Figure 5.6-B) and recognize a high amount of frog bones within the shaft. The high amount of frog bones from these layers is not such an unusual phenomenon in medieval archaeology. Frogs tend to fall in shafts and are not able to get out. Another possible explanation would be the consume of frogs even if its evidence in medieval complexes is not yet rendered. In the well of the excavation "Altstadt" in Villingen, Germany, from the $13^{th}/14^{th}$ century 315 bones of *Rana temporaria* as well as three bones of *Bufo bufo* (the common toad) were found [vdDK14]. As the authors state, all of them seem to have been fallen in the well. This death assemblage which is not caused by predators or, as would be in our case, humans called *thanatocoenosis*.

Frog bones were found for the first time in layer 801 of shaft 5 although in small numbers ($n = 4$). However, the highest **N**umber of **I**dentified **Sp**ecimens (=NISP) can

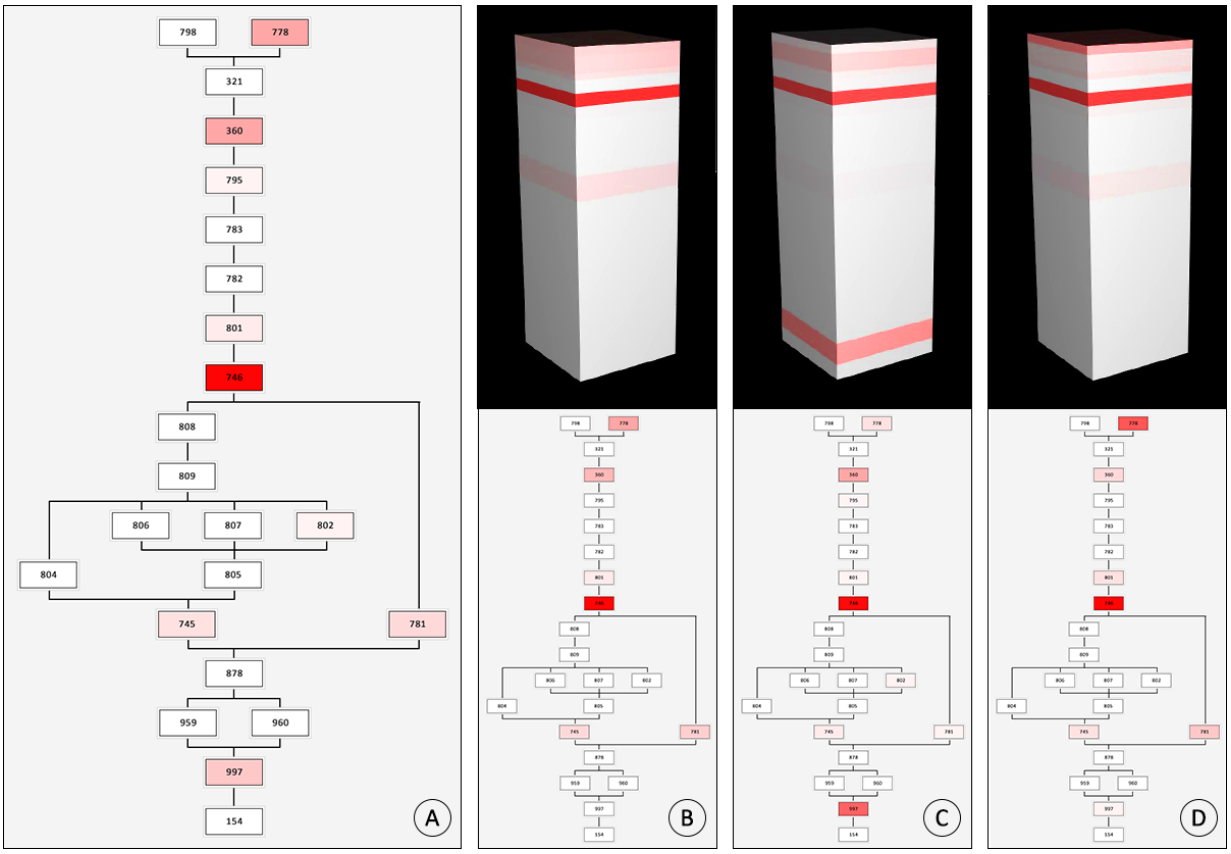


Figure 5.6: *The 3D representation and Harris Matrix of data from shaft 5 of the excavation Marienhof-Haltepunkt in Munich, Germany. The shown Harris Matrix (A) is the temporal structure of the drawing sheet in Figure 5.5 with the unfiltered distribution of findings. The other illustrations show the 3D representation of the shaft and the distribution of filtered findings of (B) frogs, (C) cattle, and (D) dogs/cats.*

be found in layer 746. Here the density of frog bones is at highest ($n = 464$). But if we take a look on the faunal material of layer 360, which superimposes layer 746, we can see that there are only 37 frog bones. In the other layers, frogs are found as well, but also in lesser numbers. The high density of frog remains in layer 746 makes clear that many frogs or rest of frogs were fallen respectively thrown in the shaft in a specific time period. Since the shaft has a dual story (at the beginning as a well and afterwards as a latrine) we could hypothesize that the distribution of the frog remains reflect the special story of this shaft.

This dual story is best reflected in the composition of the layers 997 and 745. As can be clearly seen in Figure 5.6-C, the bones of cattle are most abundant in layer 997 while the bones of other species are underrepresented. In the superimposed layer 745 there is only a small number of bones ($n = 73$) with small ruminants being most abundant.

Interestingly enough, the NISP is remarkably small even though layer 745 is the biggest of all. If we add the fact that there are virtually no other finds in the layer besides the bones one could wonder about the purpose of such a layer. The answer lies in layer 997. The cattle bones of layer 997 belong solely to one skeleton. This skeleton of a cow (as can be clearly seen on the pelvis) was fallen respectively thrown into the shaft when it was still a well. Because of water contamination, people decided to backfill the well. Hence the superimposed thick layer 745 where almost no findings were made. The visual generation through TARDIS makes it easy to spot such deviations and allows the user to quickly uncover interesting findings.

In TARDIS, we also filtered the available data for remains of cats and dogs, as visualized in Figure 5.6-D. In the subsequent layers, such as layer 778 and especially layer 746, we identified many bones of cats and dogs. Again, this is a typical phenomenon in medieval archaeology as many small animals were disposed in wells and latrines. This was for instance the case in the aforementioned well of "Altstadt" in Villingen, Germany, where six dog skeletons were uncovered [vdDK14]. In shaft 5 at least two partially preserved dog skeletons were identified. More abundant are the remains of cats. Many of them were juveniles and were disposed in the shaft, because they were "superfluous". Again, thanks to the quick visualization through TARDIS such findings can be quickly detected.

Another interesting aspect is the economic importance of livestock in the middle ages. The most important ones were cattle, pig, small ruminants (sheep or goats[23]) as well as – on a smaller scale – chicken. Trends, which can be quickly generated in TARDIS, are based on the distribution of the bones of each animal. The higher the number, the more (economically) important the animal.

In shaft 5 the raising importance of small ruminants is recognizable. While the NISP of cattle and small ruminants is balanced in layer 746, the relative share of small ruminants increases in the higher positioned layers 393 and 360. In layer 360 the domination of small ruminants becomes apparent.

As we have seen in this case study, statements on the development of the economic importance of the different livestock animals within subsequent time periods can quickly be done with TARDIS, a new application to visually analyze archaeological data.

5.5 Annotation

TARDIS is a first approach for an automated process that combines the spatial positions and temporal information of archaeo-related data. Although the data is available, the

[23]The morphological distinction of the bones of sheep and goat is not always possible in the zooarchaeological context, zooarchaeologists often use the term 'small ruminants' instead of 'sheep or goats'

combination of this data is seldom used because of the barriers B1 and B2 described in the introduction of this Chapter.

The REMIS architecture (cf. Chapter 3.2) massively contributes to close barrier B2. However, barrier B1 cannot be closed by only a technical solution. Certainly, there are possibilities to realize a detailed recording of the stratigraphy that also can be enriched with basic archaeological information. However, this kind of recording is complicated and time-consuming. Keeping in mind that methods to consider spatial and temporal information are rare, the time–benefit correlation of stratigraphy information is disproportionate.

The described use case in the previous Chapter 5.4 uses real data from an archaeological excavation. However, the spatial information was not available in digital form, although EXCABOOK supports the gathering of this information. The necessary digital data for the case study was converted to a digital form by the authors of the work. The described analysis would not have been possible without this conversion.

With TARDIS we demonstrated that the consideration of spatial positions and temporal information can lead to new research questions and to insights in the archaeo-related sciences. We hope to contribute motivating archaeologists and bioarchaeologists to provide digital stratigraphy information to close barrier B1 that would form a basis for these analyses in this vein.

Acknowledgement

We thank *Grabungsfirma ReVe, Büro für Archäologie*[24], Bamberg, Germany, in particular Barbara Wührer, and Dr. Christian Behrer of *Büro für Denkmalpflege*[25], Regensburg, Germany, for providing the stratigraphic data and archaeological information of the excavation Marienhof-Haltepunkt, Munich, Germany, used for the case study in this Chapter.

[24] http://www.reve-archaeologie.de
[25] http://www.denkmalpflege.biz

Chapter 6

Linkage of Archaeological Data with Interdisciplinary Knowledge

Attribution

This Chapter uses material from the following publications:

- Daniel Kaltenthaler, Johannes-Y. Lohrer, Florian Richter, and Peer Kröger. ReMIS Cloud: A Distributed Information Management System for Interdisciplinary Knowledge Linkage. In *8th International Conference on Internet Technologies & Society 2017, Sydney, NSW, Australia, 2017*, pages 107–114, 2017. [KLRK17]
- Daniel Kaltenthaler, Johannes-Y. Lohrer, Florian Richter, and Peer Kröger. Interdisciplinary Knowledge Cohesion through Distributed Information Management Systems. *Journal of Information, Communication and Ethics in Society*, (to appear) 2018 [KLRK18]

See Chapter 1.2 for a detailed overview of incorporated publications.

According to John Naisbitt, we "are drowning in information but starved for knowledge" [Nai82]. In this context, the gap between data and valuable information can often lead to challenges regarding the acquisition of knowledge from data. Without suitable data retrieval techniques, the analysis process loses quality and progression speed.

Within the same discipline, the corresponding learning community usually agrees on common data standards. Healthcare databases are more likely to be understandable for physicians, but they will have problems understanding archaeological databases. Sharing the data between both research fields does not yield any further information automatically.

However, it is an important point to take into account interdisciplinary data in all kind of sciences. For example, the archaeological and bioarchaeological analyses strongly benefit from considering data from archaeo-extrinsic sciences: Integrating information from e.g. meteorologic, geological, and biological disciplines into archaeo-related analyses could uncover relationships which are not immediately apparent. However, interdisciplinary data exchange is not only interesting in the scientific sector. As an example, the way to manage the data in the eLearning field could also benefit by regarding interdisciplinary data. One field of application could be students of history lessons who could be motivated by autonomous learning by being able to retrieve related data of a special historic event.

If two or more disciplines want to combine their data sources to create new knowledge, it needs an information management system in-between. This system has to manage diverse data sources which are not restricted to databases only. It has to know mappings between the attributes of different data sources, such that links of congruent information can be established. For example, the bone analysis of findings of an excavation is just an observation. However, by cross-connecting it with a medical database this might be an indicator linked to a certain food poisoning which again improves the insight of the archaeologists and bioarchaeologists.

Our novel information management system allows users from different disciplines to register their data sources in a server application. A search mask is provided so that a user can look up information without having to know each data source. The most important function and the novelty of our approach is the method of management of data sources. Each data source has to designate categories during its registration. A category specifies a set of attributes. If two data source share a common category, both sources are able to establish a link during a search. As the attribute sets of both data sources have a non-empty intersection, cross-connected knowledge can be derived. To avoid the generation of exhaustively many results, the users can also specify tags for their search. Tags are a filter mechanism to reduce the result size.

We developed the REMIS CLOUD for applications like these. We give a brief summary of our contributions:

1. An information management system capable of diverse data sources for computer-assisted collaborative content and information sharing.

2. A category-based attribute search to enable cross-connections of knowledge from different domains.

3. A filter mechanism to shrink result sets to a feasible size.

4. A dynamic and simple registration process to connect diverse sources into one information system.

Therefore, we first discuss the technical background and related work in Chapter 6.1. Then we describe the architecture of the REMIS CLOUD in Chapter 6.2. Afterwards, we explain two use cases in Chapter 6.3, where we consider two different scope of application: Archaeology as an example for an scientific use case and an example from the eLearning field. Finally, we compare the REMIS CLOUD with the original REMIS system in Chapter 6.4.

6.1 Related Work on Interdisciplinary Data Retrieval

The provision of content is known as Content Sharing in which at least one node in a community provides a data source with information. In most cases, data providers consist of more than one node. Usually, they agree on using a common format so that data is uniformly retrievable. For individual knowledge fields, there are information systems which are extensive in their closed field and can answer queries with good accuracy. To mention some examples, XBOOK as an information system framework for archaeological and bioarchaeological data (cf. Chapter 2), Weaver et al. [WBKK16] for healthcare data, and information management systems of research itself like *KRIMSON* [MK17].

However, the idea of considering interdisciplinary data is not only interesting in the scientific field. Content sharing systems also exist for the eLearning field, e.g. since social media platforms have been emerging intensely, especially young students use social media as an eLearning support [LHJ16]. In the field of eLearning, the importance of collaboration to create and share new knowledge is well-known. The paradigm of collaborative learning has already existed for a long time. Koschmann et al. [KKFB96] proposed to utilize computers to assist the collaborative learning about 20 years ago. Stahl, Koschmann, and Suthers [SKS06] outlined the history of computer-supported collaborative learning (CSCL) and stated that it is far more complex than just digitalizing sources to support teaching with few efforts. Later research by Gress et al. [GFHW10] was able to recognize the benefits of CSCL and showed its weaknesses. Recently, the evolution of mobile technology has driven the CSCL community to mobile computer-supported collaborative learning (mCSCL) [SCL16].

The *Reverse-Mediated Information System*, which we introduced and described in Chapter 3.2, is also an information system to exchange data. However, it requires each connected database to map the parameters of the Minimal Find Sheet to the corresponding information in the database. Thus, its functionality is limited to one discipline. An interdisciplinary information system requires an architecture that cannot be achieved by the initial REMIS.

Our information system is related to collaborative learning as interdisciplinary problems can only be solved by utilizing information of diverse sources and from different

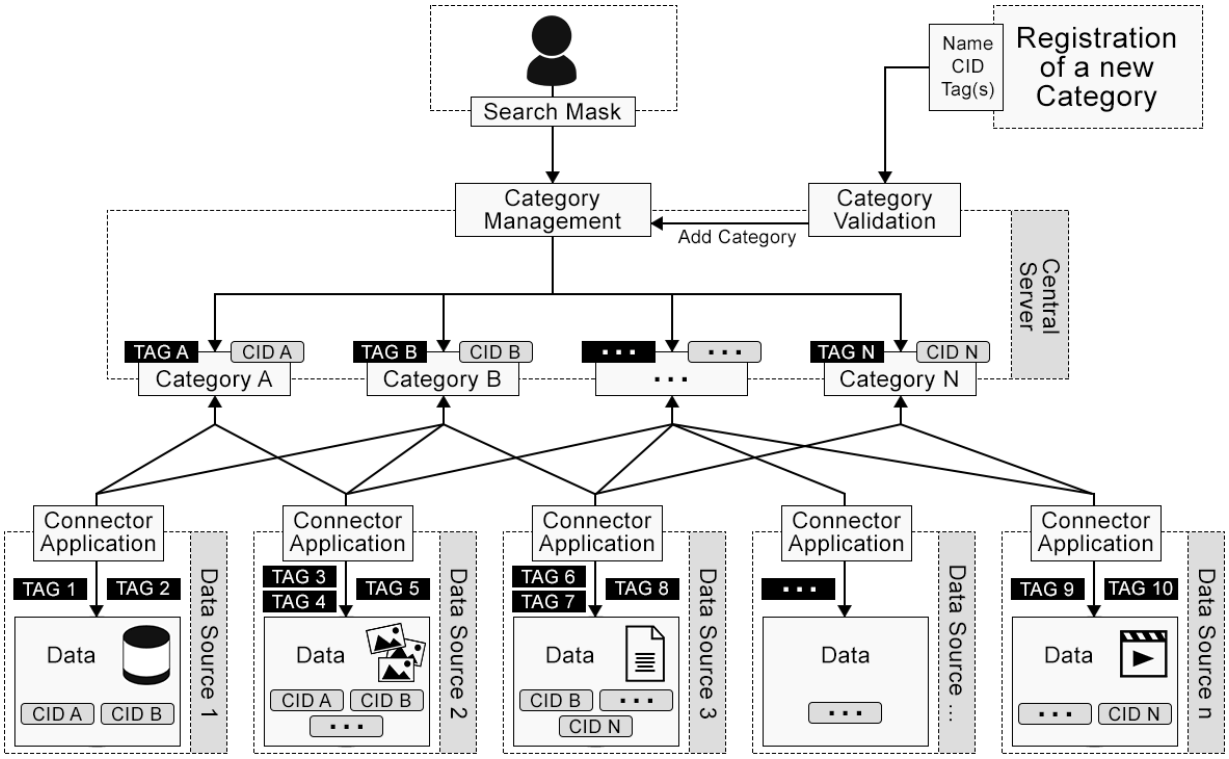


Figure 6.1: *An abstract sketch of the ReMIS Cloud architecture.*

fields. Every peer in a network of knowledge providers is able to make some information accessible which has to get linked to congruent data in other sources.

However, knowledge acquisition can be difficult in interdisciplinary sciences or learning. Formats of data are very diverse and a simple question cannot be translated easily into a complex and precise SQL query to join database tables of different database systems. To the best of our knowledge, there is no information management system which assembles diverse types of data sources (like databases, file systems, and other knowledge representations) into one information management system. Furthermore, our REMIS CLOUD system provides the ability to allow links of congruent information in order to related information can be found in diverse sources.

6.2 ReMIS Cloud Architecture

The decentral architecture of the *Reverse-Mediated Information System* (cf. Chapter 3.2) provides a solid basis. The heterogeneous data sources remain in their origin data sources. However, we provide a central architecture in which multiple REMIS are embedded. We call this architecture the REMIS CLOUD which is described below. An abstract overview of the REMIS CLOUD is sketched in Figure 6.1.

6.2.1 Architecture Structure

The basic functionality of the Server Applications in REMIS is retained, but it should now serve as a central possibility to categorize a specific field of data. Instead of distributed Server Applications for separate topics, they are now managed on a central shared server for the REMIS CLOUD technology. We want to provide a unified category system which is considered by all connected data sources. This is essential because these categories form the basis for the search requests. To point this out, we use the term "Category" as a synonym for a Server Application.

If needed, new Categories can be registered to the REMIS CLOUD and need to be activated by a responsible person. Even if this is an additional step to be managed, the REMIS CLOUD technology benefits from central managed categories to allow a better quality control, like to avoid nonsensical and duplicated Categories.

A Category basically consists of a name for the Category and the definition of the Category Information Definition (CID). The CID, which is comparable with the Minimal Search Parameters, guarantees that all connected data sources provide the defined parameters and therefore the information that is necessary for the corresponding Category. These are also the parameters the users can search for. We would like to point out that each data source is not limited to one single Category and specific columns of a data source can be mapped to several Categories. If all parameters of all CIDs are mapped, any number of Categories can be assigned.

Furthermore, the existing REMIS architecture is extended by a tagging system to provide further and more detailed filtering of data. This tagging system is integrated to the Server Application and the Connector Application. The administrator of the Server Application defines one or more appropriate tags which describe the general context of all connected data sources. In the Connector Application, the data owners can define individual and more detailed tags for their data during the initialization process. Additionally, each data source also inherits the defined tags of the dedicated Server Applications.

6.2.2 Data Retrieval Process

The REMIS CLOUD provides a central website where search requests can be defined by any users. Below, we describe the data retrieval that is split in three steps.

Step 1: Parameters of the CID

In the first step, the users can select from all available Categories that are registered to the REMIS CLOUD. Once one is selected, all parameters of the CID of the Category are loaded and displayed to an input mask. There, the users can enter their

search parameters. This is sufficient to already run the data retrieval. All databases that are connected to the corresponding Category will now be queried to gather the data, dependent on the user input.

So far, the process is comparable with the origin REMIS, but now it also allows multiple Categories to be searched. Thereby, we have to differentiate between an OR search and an AND search which the users decide. Using an OR search, data from the data sources is retrieved once at least one value of the defined search request matches, independent of the searched Category. Using an AND search, the architecture has to ensure that all search parameters match independently on the number of Categories.

All values of the filled search parameters of the CID by a user are now merged into a list. Sorted by the order of the selected Categories in the search, the Categories are now queried by the Category Management in the Central Server. Since all data sources that are connected to a specific Category are guaranteed to support the CIDs of the Category, the Category has only to check if the data source was already searched. If this is not the case, the list is passed to the Connector Application of the data source. There, all search parameters that were filled by the user (also the ones of the other selected Categories) are mapped to the local scheme of the data source which is then queried and returns the list of results.

Step 2: Search for further information

The second step is basically optional, but provides the possibility to search for further information about existing data sets, even if the information is spread through different data sources. Here, the search mask provides a selection of Categories – without the definition of any specific search parameters. This selection displays all available registered Categories from the REMIS CLOUD system. A preselection of the available Categories of the data sources is not possible in advance without an additional search to determine them. Due to the accessible data sources may be different for every new search requests, it is not intended to execute several iterations for each search run. If the users select any number of Categories, these will be considered to search for further information dependent on the CID values.

The list of result, which was returned by each data source in the first step, is checked for values of the Categories that were selected by the users for further information. These values are now used to search for further information with the values of the result as parameters. Thereby, a search request can be sent to the corresponding Category. The values of the parameters of the CIDs are not defined manually by the users in this case, but are read out from the data source. The results are sent back to the users. Finally, the results from the first step and the additional information retrieved from the second step are displayed.

Step 3: Filter the result

In the third step, the users can filter the result for specific tags or data sources. While retrieving the data, the returned search results are supplemented with the tag information and the name of the data source. This enables a filtering for this information. Therefore, the users get a selection of all available tags from the search result and names of all data sources. A selection of the tags and data source names filters the data.

6.3 Use Cases

The REMIS CLOUD can be applied to numerous sciences and applications. In this Chapter, we describe a scientific example with archaeological data and an example from the eLearning context.

6.3.1 Scientific Example: Archaeology

Archaeo-related sciences deal with distributed data which is not commonly standardized. There are countrywide and statewide, but also regional and individual approaches how to gather and store archaeological data. This makes it hard to consider all available data for large-area analysis. The distributed data sets can hardly be set in relationship due to different data schemes and standards. In contrast to this, analyses of archaeo-related disciplines would benefit strongly from considering scientific data from other disciplines as well. But up to now, the retrieval of interdisciplinary data is cumbersome, time-consuming, and error-prone.

The REMIS CLOUD supports the archaeologists and bioarchaeologists in their work by providing an architecture that allows retrieving spread and interdisciplinary data.

We basically consider three databases with archaeological data as a vivid example which is illustrated in Figure 6.2. In Bavaria, Germany, the zooarchaeological database OSSOBOOK [KLK+18b] and the archaeological database EXCABOOK [KLK+18f] follow the guidelines of the *Bavarian State Office for Monument and Sites*[26]. Therefore, these databases are connected with the Category "Archaeology BY" (with the tag "Archaeo") which requires that the parameters of its CID are mapped to the databases, that means: The find sheet number and the excavation number. The database *ADABweb*[27] is used in Baden-Württemberg and Lower Saxony, Germany, and could be connected with the Category "Archaeology BW/NI" (also with the tag "Archaeo") of which the CID requires other parameters to be mapped. Because all three databases have also saved x- and

[26] http://www.blfd.bayern.de
[27] http://www.adabweb.info

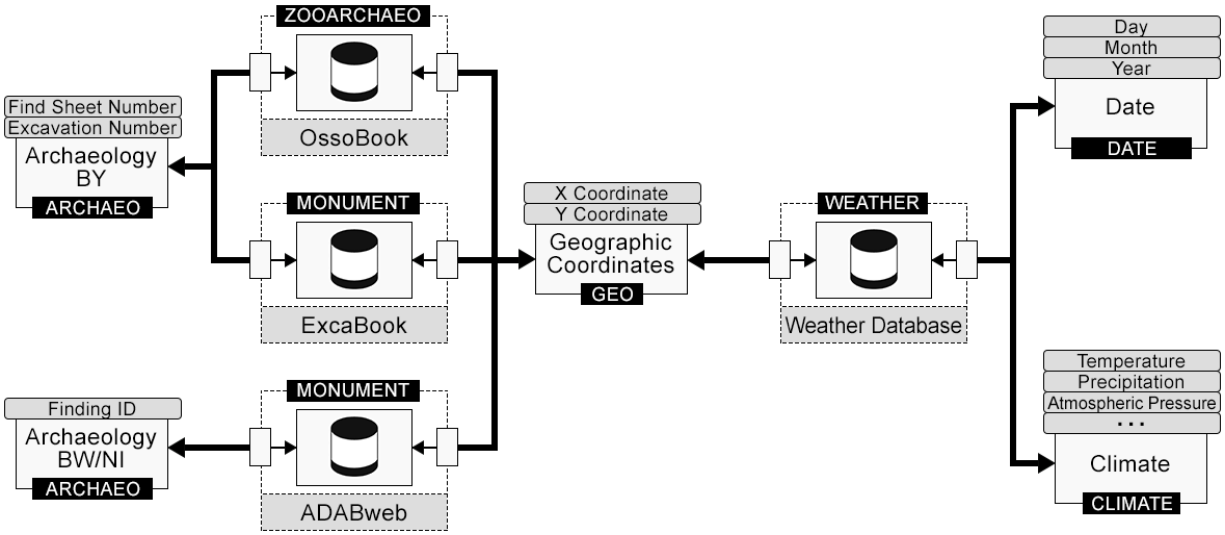


Figure 6.2: *A simplified sketch of the* REMIS CLOUD *architecture for archeo-related sciences.*

y-coordinates of the excavation places, they could also be connected to the Category "Geographic Coordinates".

Basically, archaeological and bioarchaeological scientists are interested in considering all available findings of an excavation from all databases for their analyses. Using the origin REMIS architecture, it was already possible to gather all data of a specific excavation number from the databases OSSOBOOK and EXCABOOK, but the corresponding findings of the database *ADABweb* are not considered. With the REMIS CLOUD, the users could now search for a specific excavation number in the search mask and define to search for further information about the search results. By selecting the Category "Geographic Coordinates" the system could retrieve all information with the same coordinate information from all data sources that are connected to that Category. In our example, we would retrieve matching information from the database *ADABweb* and also interdisciplinary data, like climate and time information from a weather database. The scientists could now filter the result for the tag "Archaeo" and they would only get displayed the data from the archaeological databases with this tag. As a reminder, the data sources inherits the tags of the connected Categories.

The scientists can also consider the interdisciplinary data from the result. If they filter the result for the tag "Climate" in our example, they retrieve all available climate data with the matching geographical information for the searched excavation number. This could be very useful for large-area analyses of findings where climate conditions should be considered. Of course, this is not limited to climate data. The possibilities depend on the data sources that are connected to the REMIS CLOUD.

6.3.2 eLearning Example

Imagine some students have to answer the following question:

> *"The year 2017 marked the 500th year after Martin Luther nailed his theses to the door of the Castle Church in Wittenberg. We assume that he had access to a mobile audio device. What music would he had listened to?"*

This question is not so trivial to answer by just using a query in the search engine of one's choice. The information is available in heterogeneous and distributed data sources, but it is not pre-assembled somewhere.

Using classic search strategies, one would try to use consecutive search queries to refine the results and search in different sources. With our information management system, this can be done in one step. We assume that we have the necessary data sources registered in our system. As sketched in Figure 6.3, there is a database containing historic events, corresponding persons and dates, and a file system containing music files.

First, we query "Luther" in the Category "Historic Events". The database with historic events contains the Category "Historic Events", so this data source is examined for matches with "Luther". The result set could be stated as shown in Table 6.1.

Event	Person	Year
Birth in Eisleben	Martin Luther	1483
Start studying in Erfurt	Martin Luther	1501
Vow to become a monk	Martin Luther	1507
Travel to Rome	Martin Luther	1510
Posting of 95 theses	Martin Luther	1517
Excommunication	Martin Luther	1521
Translation of Bible	Martin Luther	1534
Death in Eisleben	Martin Luther	1555
...	...	...

Table 6.1: *Extract of the result for the example query "Luther" in the Category "Historic Events"*

Our tool allows to further inspect the set of data sources for links with the current result data. So we can start the second step with one click. The database with historic

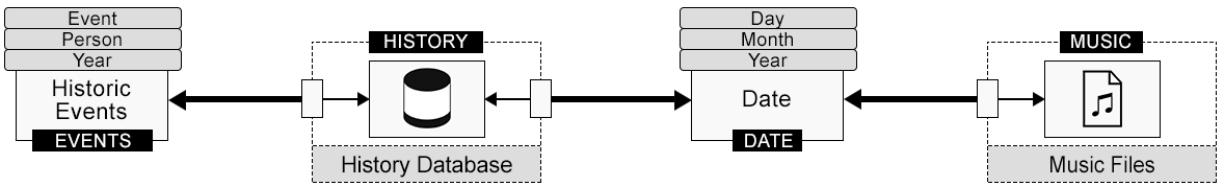


Figure 6.3: *A simplified sketch of the* REMIS CLOUD *architecture for the eLearning example.*

events also inherits the category “Dates” and for the file system containing music files it is also very usual to do this as well. Music files contain meta data like the publishing date in most cases. In a uniform database environment, we would have two database tables and can join tables using the “year” attribute. Using heterogeneous data sources makes it harder to define join operations.

Our approach utilizes the Category here. Since the data owner had to define the Categories during the registration of the source, the common semantics of “Dates” are already embedded in the system. The system matches other sources’ data with the Category “Dates” and especially with every date according to events in Martin Luther’s life. This allows us to quickly find connected information between Martin Luther and music composed in these times. The result can be seen in the screenshot of our prototype search engine in Figure 6.4.

Obviously, we need the data and a correct registration to obtain such results. On the other hand, the individual domain knowledge can be acquired as music repositories and event databases exist.

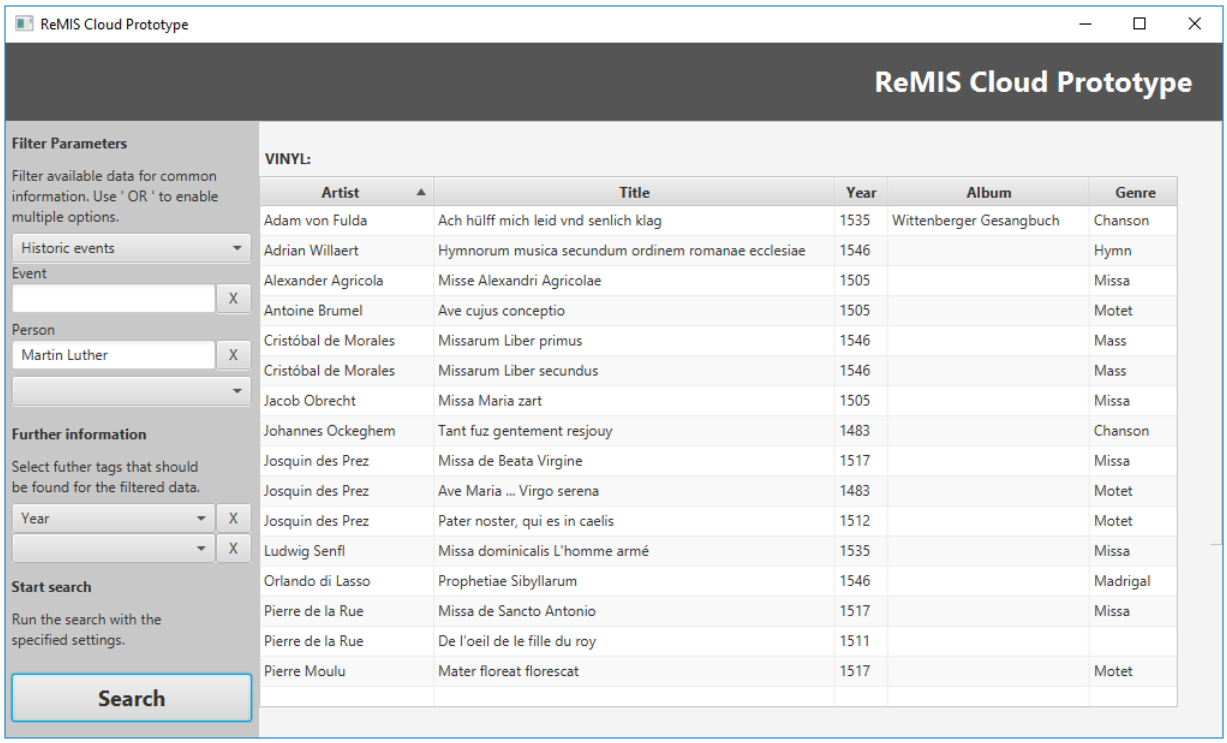


Figure 6.4: *Screenshot of the* REMIS CLOUD *Prototype that displays the result described in Chapter 6.3.2.*

6.4 Comparison of ReMIS and ReMIS Cloud

In this Chapter, we introduced the REMIS CLOUD system that is based on the *Reverse-Mediated Information System* (REMIS) technology (cf. Chapter 3.2). The two systems use similar parts of the architecture, especially for the initialization, registration, and search process (cf. Chapter 3.2.3). However, there are differences in the area of application. Below, we compare the REMIS and REMIS CLOUD systems. [KLK18a]

From the users' point of view, both information systems provide the possibility to retrieve data from distributed and heterogeneous data sources. The users need to know about neither the physical location nor the existence of the data sources from which they want to retrieve any data. They can enter their search requests which are automatically forwarded to the connected data sources. The search results are directly displayed to the user. However, only REMIS CLOUD supports the retrieval of coherent interdisciplinary data. A tagging system is necessary in REMIS CLOUD because of the limited scope of application. Thereby, filtering retrieved information using tags is only supported in REMIS CLOUD.

REMIS is an information system designed for one specialized discipline, for example for the archaeo-related context. Each instance of REMIS only supports one set of search parameters which defines the scope of application. So it is possible to set up an individual REMIS for each field, e.g. for archaeology and bioarchaeology considering the Minimal Find Sheet as parameters. REMIS CLOUD is intended to provide a central platform where all disciplines can connect, regardless of the used Category Information Definition (CID). There is no need to set up multiple instances of the information system. REMIS CLOUD also supports the assignment of multiple CIDs to one data source.

In both systems, data owners keep the full control about their data, the right management is administered locally where also the data sources are stored. Only authorized data is transmitted to the information systems. It is the data owner who decides which data can be searched and retrieved. Furthermore, there is no central administrator who connects the single data sources, it is up to the data owners to connect their data to, respectively remove their data from the network again at any time.

These differences enable the use in different field of works. REMIS could be considered as a system that is limited to one field, e.g. to specialized areas of application like archaeo-related disciplines where the necessary data is manageable. It is conceivable that each discipline sets up an own instance of REMIS that can be used within the field. On the other hand, REMIS CLOUD is a solution for generally available data that makes the integration of interdisciplinary data possible.

However, it is not excluded that data sources can be connected to both, the REMIS and the REMIS CLOUD infrastructure. Both systems are able to run in parallel on the same data sources.

Availability

A prototype of the REMIS CLOUD is available from `http://remis.dbs.ifi.lmu.de/`.

Chapter 7

Summary and Conclusion

Attribution

This Chapter uses material from all publications attributed in the previous Chapters.

See Chapter 1.2 for a detailed overview of incorporated publications.

The contribution of this thesis is to present and describe methods and concepts to provide, store, share, and retrieve archaeological and bioarchaeological data to support the workflow of archaeo-related sciences.

In Chapter 2, we described the historic development of the zooarchaeological database OssoBook and the resulting framework xBook, a generic infrastructure for distributed, relational data management that is mainly designed for the needs of scientific data. We described the concepts of the architecture and its most important features. We especially pointed out the Server–Client architecture, the synchronization process, the Launcher application, and the structure and features of the application.

In Chapter 3, we discussed collaboration, sharing, and retrieval methods for archaeological and bioarchaeological data. In Chapter 3.1, we described a list of requirements that should be fulfilled by e-Science infrastructures for a synchronized distributed data management. We discussed existing solutions for synchronized distributed data management, in relation to the previously listed requirements. Then we took an in-depth look at the synchronization process of xBook, which is independent of the data model and could be used potentially in any application domain if needed. In Chapter 3.2, we introduced the concept of a new architecture, the *Reverse-Mediated Information System* (ReMIS), motivated by the need to search distributed data of archaeological and bioarchaeological findings together with further contexts from multiple heterogeneous sources. We described the concept of the architecture where users can retrieve data without having

to know about the existence of specific databases where the data is actually saved. It should be mentioned though, that this concept is not limited to the archaeo-related context. Other sciences could also benefit from this architecture.

In Chapter 4, we described the ANALYSIS TOOL, a framework for data analyses that can be embedded into a base application. We first described the requirements for our ANALYSIS TOOL and explained the steps we took to meet these requirements and to integrate the ANALYSIS TOOL into a base application. We explained how the tool can be extended with new specific components that allow the users to add exactly the features they need for their individual analyses. Second, we set a list of requirements that should be fulfilled for a user-friendly graphical user interface for analysis tools for archaeologists and bioarchaeologists. We discussed several approaches, in relation to the previously listed requirements. Considering the results of a user study we defined a solution and implemented a prototype of the ANALYSIS TOOL. Finally, we embedded this implementation to the XBOOK framework and used it in the zooarchaeological database OSSOBOOK for a case study. We pointed out the possibilities of the graphical compositions and the fast generation of diagrams in the graphical user interface.

In Chapter 5, we described the application TARDIS, a visual analytics system for spatial and temporal data in the archaeo-related field, motivated by the challenge to set spatial and temporal data from excavations in relation. Therefore we generated 2D and 3D representations of the excavation site that are enriched with data of findings to indicate their location. Furthermore, we generated a Harris Matrix that is used to illustrate the temporal positions of the findings. We also explained the implementation of the generation of the Harris Matrix. These components are interactive, so archaeologists and bioarchaeologists can set the illustrated results in relation. Finally, we applied data from a real excavation site to TARDIS to point out the benefit of the application.

In Chapter 6, we proposed the information management system REMIS CLOUD, which enables information retrieval by linking different distributed data sources to find inter-domain knowledge with an intuitive search interface. We utilized a category-based data source registration to connect differently shaped data formats. Results are assembled using a join-operation-like retrieval mechanism. The users are able to learn new insights in interdisciplinary fields.

Finally, this Chapter concludes this thesis by first discussing the methods that were presented in the previous Chapters, address some perspectives to improve the current implementations and concepts, and offer further research challenges in Chapter 7.1. Then, Chapter 7.2 includes a discussion about the technical situation of the archaeological context where the future developments of archaeo-related technologies is debated.

7.1 Outlook

The work presented in this thesis only scratches the surface of the planned goals. This Chapter will discuss ways to enhance the applicability of the provided applications, methods, and concepts as well as give some further ideas on how to improve the way of working with them.

Synchronization

In XBOOK, the synchronization offers the possibility for several persons (work groups) to work parallely in a certain project at the same time. As mentioned above, if no Internet connection is available the instances of XBOOK can still be used locally. This is very important to enable entering data directly on field work during excavations. The synchronization checks the data sets and informs the user about conflicts that occurred in the same data set. By using the automatic synchronization each data entry is synchronized after it was entered or edited, so data is backed-up on the server and can be recovered at any time. The data of own projects can also be used by colleagues by sharing data between certain users or groups. This takes advantage of primary data that is not always published because of the large volume, to be accessed for further studies.

While the synchronization has many benefits for the user and also for someone who wants to add tables to the synchronization, there are still some limitations that have to be addressed in the future:

- **Data overhead:**
  A big data overhead is generated due to the way the data is sent and retrieved from the server. Some additional (and required) information like column names is sent that increases the amount of data that needs to be transferred. It might be possible to use a better data type for the transfer that reduces the amount of information, like – if a series of entries is sent – the column names are only sent once. This would decrease the size of data, but would require a check to prevent that entries get saved in the wrong column due to a value not being sent or an additional value being sent.

- **Unnecessary data transfer:**
  Because of the possibility of an incomplete synchronization, the entries, that have the same status as the local highest status, have to be sent again at the beginning of each synchronization process. This could mean that many entries that are already in the local database have to be transferred. The amount of data to be transferred could be reduced by only sending the key of the entries in question. Only if the cor-

responding entry does not exist locally or has a different timestamp, the complete entry would have to be resent.

- **Deletion of entries:**
  While the synchronization is working as intended, the architecture requires that single data sets cannot be finally deleted from the databases, neither the local nor the global one. If a user needs to delete an entry, we set a flag in the database instead that indicated whether the entry is deleted or not. This is necessary because otherwise we cannot recognize if an entry that was deleted in the global database and has to be removed from a local database as well, was updated in the local database when the handled entry set is not available in the global database anymore. This circumstance results in two disadvantages: First, disc space is unnecessary occupied by still having to store deleted entries in the database. This also influences the execution time of the synchronization process due to deleted entries that have to be handled as well. Second, deleted data can only be set "invisible" by considering the flag indicated that the entry is deleted. But for some kind of data, e.g. critical or personal data, it might be necessary to delete data completely from the database. However, it may be also discussed if this circumstance could also be desirable in specific fields, for example for scientific analyses where deleted entries are meaningful.

- **Detecting duplicate entries:**
  Currently it is not possible to detect directly duplicate entries due to the local databases that have to work in offline-mode, too. A helpful addition could be to generate a list of similar entries from the global server within the application. This could be used to identify possible duplicates. But since the similarity of entries is context specific, a general best approach cannot be specified.

- **Compression:**
  Since for each entry the rights are checked and one entry is sent at a time, the data is synchronized very slowly in comparison to directly viewing the data in a SQL viewer. A possible speed increase could be achieved by compressing the transmitted data. This should fasten up the transport significantly – especially if large data packages are sent.

- **Improvement of conflict detection:**
  While the synchronization is able to detect conflicts, it may detect false positives due to only comparing the timestamp and not the actual value that was changed. To improve the detection of conflicts and to provide an automatic merge, the synchronization could be enhanced to be able to keep track of changes in tuples. Thereby, each value has its own timestamp and can then be checked for conflicts.

- **Open access:**
  The synchronization architecture allows sharing data with colleagues or other users. However, it is planned that the user rights management of the synchronization is extended with an option to publish data to provide the opportunity of open access for the gathered data.

Reverse-Mediated Information System

While the architecture of *Reverse-Mediated Information System* has already many benefits for the users to retrieve data from distributed archaeological and bioarchaeological databases, there are also some improvements and extensions that can be addressed in the future.

- **Limited search possibilities:**
  In the first approach of the concept, the search possibilities for data are limited to the information of the Minimal Find Sheet. For future developments, different and more complex search options are possible. The Server Application could request all connected databases for their individual database schemes. The columns of this schemes could be displayed in the user interface as own input fields. Therefore, the user has more options and the possibility of a more detailed and individual search through the connected databases.

- **Different data formats:**
  Data can be saved in different formats, e.g. different time or date formats, geodata can be stored by defining coordinates or the name of the place. Currently we imply that data is entered in the same format, but this might not be the case in other working environments. Therefore, it is necessary to provide a possibility to translate these different formats to enable a uniform comparison.

- **Data source verification:**
  At present, the system is designed to be used for the clear excerpt of structure for archaeo-related data where most of the data sources are trustworthy. In future, the system should also be adaptable by external archaeologists, bioarchaeologists, and institutes, maybe also a full-public approach. Therefore, a verification should be considered to avoid wrong configured Connector Applications and spam.

- **Data import methods:**
  The data of the Minimal Find Sheet is handed on from the offices to the specialized collections or specialists after the excavation. These are basic information and necessary to be saved in their databases as well since they are important for further analyses. Currently this data has to be imported or entered manually. In principle,

it is conceivable to allow archaeologists and bioarchaeologists to retrieve this data via the Server Application to be imported automatically into their databases. The corresponding data can be retrieved by querying information about the Find Label Number which is unique. This would decrease the effort for the management of Minimal Find Sheet data.

- **Peer-to-Peer environment:**
  Currently the architecture requires a Server Application which is necessary for the communication between users and data sources. It could be considered to change the system to a Peer-to-Peer environment in which the Connector Applications are communicating directly to each other.

- **k-anonymity:**
  The data sources might also include sensible data, like in diary information or in meta information of pictures. However, this data could also contain interesting information for archaeologists, but especially personalized data should not be shared in plain text. A way to provide these data is using the concept of k-anonymity [Swe02] that could be implemented for this data in the Connector Application.

- **Extended rights management:**
  Finally, a more complex rights management system could help sharing data selectively to specific users who could create a user account for the Server Application. Administrators of the data sources could them authorize access to more detailed data that is set to more restricted privacy settings. This would also be a possibility for administrators to pass data to a user that was contacting him via the contact information (cf. Chapter 3.2.3).

Analysis Tool

The ANALYSIS TOOL was described as an early prototype. However, it is already ready for use and provides many benefits for the target group, especially caused by reasons of reliability and time-saving. Naturally, there are currently some limitations that still need to be addressed in future to achieve its full potential. There is still room for further improvements for the usability of the ANALYSIS TOOL as described below:

Technical aspects

- **Limited to Java environments:**
  Due to the nature of the implementation, it can only be used in a Java environment which is the main limitation of this analysis framework. While the concepts themselves are able to work in any environment, the implementation would have

to be specific for the language of the base application. An application that is not written or compatible with Java cannot currently use the ANALYSIS TOOL. So either there would have to be an instance of this tool for every programming language or a wrapper has to be developed to be able to use the framework in other programming languages as well.

- **Requirements for integration:**
  While the integration into the base application is straight forward and can easily be done, it still requires both access to the source code and knowledge how to program. Therefore, the typical users cannot do the integration themselves. This means that the developer of the base application has to integrate the ANALYSIS TOOL to provide the functionality to the users. For these cases, the framework could be extended to run as a standalone application which can be connected to the database directly. This would require additional settings which handle the connection to the database itself. At the same time, a stand-alone tool could benefit from the same level of flexibility and extendibility that the framework offers while being integrated into a base application.

- **Requirements for custom Workers:**
  Additionally, the issue that some programming skills are required for creating a new Worker still remains: The implementation of the custom Worker has to be done by a software developer.

- **Extended Worker functionality:**
  There are already possible a wide range of possible analyses, but still many additional Workers have to be created. These Workers should be part of the framework itself since they can be used for analyses in different areas. Possible Workers include clustering algorithms, different diagram types, statistical methods, etc.

- **Improved memory management:**
  The logical handling of the data sets is currently focused on processing speed. For this the `DataLists` are often just copied and remain in the primary memory. For small data sets this is a fast and efficient way, but for complex databases with thousands or millions of entries this method can quickly reach the computer's capacity limit. To avoid this problem several solutions would be possible. The generated result could only be kept in memory during one operation until succeeding Workers have used the Output Data. This would slow down the analysis process since, even for a small adjustment, all Output Data would have to be recalculated. Additionally, the generated Output Data could be saved in a temporary file. Thereby, main memory is released because the Output Data is not used for calculation. Again, this would slow down the calculation process since disk operations are relatively slow.

Interface aspects

- **Increase of the user-friendliness:**
  The integration of other elements to the graphical user interface could increase the user-friendliness by making the composition easier. For example, the use of drag and drop gestures to add new boxes and elements to a specific position would be more intuitive for the users to arrange the single elements. More feedback from the application should make it clearer for the users to set instructions and to avoid invalid operations.

- **Auto-arrangement:**
  Currently, the users have to arrange the single boxes by themselves. A function to auto-arrange, that moves the boxes to a useful position considering the available connections, would help to improve the clear overview of the composition. This function could be optional.

- **Auto-generated labels:**
  The label where the users can define a text to be displayed in the box of the single modules is a helpful, but not very comfortable solution. Still it is necessary to help the users to identify the task and/or the current data of the modules. This is a cumbersome way because the label has to be set manually with a text. A functionality to auto-generate a representing label would be helpful and time-saving.

- **Common result module:**
  By now, the Result Module is necessary to be defined at the end of the composition being an easy way to open the data in a table view. It also indicates the finished composition to be displayed in a diagram. Later, this module could be extended with the functionality to setup a specific diagram and open it directly from the composition. So the users could generate the graphical representation of the data directly from the data composition without having to select any data for the diagram in another panel.

- **Import and export of compositions:**
  In future, the ANALYSIS TOOL should provide a saving and loading functionality for the compositions. A composition can then be loaded and does not have to be recomposed from scratch. By saving the composition in a file, e.g. XML or JSON, the file can also be shared with colleagues to distribute the analysis. For these exports, it is quite conceivable to install an online distribution platform where scientists can provide and download analysis compositions.

TaRDIS

While this application has already many benefits for the domain experts, there are also some improvements and extensions that can be addressed in the future.

- **Polygon-shaped structures:**
  The 2D representation of the excavation site shall consider the definition of other shapes than rectangular and circular ones. Therefore, the plotting of new shapes, like polygon shapes, has to be supported. That is important to describe more complex structures of linear excavation sites, e.g. for motorway routes or pipelines, but also misshaped areas or sections on a smaller excavation site.

- **Addition of other statistical methods:**
  The Kernel Density Estimation is integrated to the application as one example for a statistical analysis. In future, other statistical methods shall be added to the 2D representation as well as to offer a wide selection of distribution information. The 3D representation could also be extended by statistical analysis methods that also consider the third dimension.

- **Extension of filter options:**
  The filter options that are currently realized are the most important options in the archaeo-related work. However, the filter options can be extended to also consider other information from the excavations, possibly supporting a custom filtering with user-defined attributes.

- **Comparison of data compositions:**
  The application should be extended by a feature to display two or more illustrations that show data of different filter settings. This would help the archaeologists and bioarchaeologists to visually compare two data compositions side by side.

- **More detailed 3D representation:**
  The real position of the layers in the areas shall be plotted more detailed in the 3D representation. Currently the layers are displayed as a rough estimation. Especially if layers are cut or are arranged side by side the level of detail could be increased.

- **Methods for a simpler gathering of stratigraphy data:**
  Even though the digital gathering of stratigraphy data is not part of the features of TaRDIS, it is still an important basic for the spatial analysis provided by the tool. There are possibilities to realize a detailed recording of the stratigraphy. However, this kind of recording is complicated and time-consuming, and therefore is rarely done. To provide the necessary stratigraphy data new methods have to be developed to support the difficult task of gathering this data digitally.

ReMIS Cloud

Interdisciplinary data retrieval is an emerging topic to create knowledge by collaboration of experts in diverse domains. New insights can be found by using the combined techniques and information when people have the opportunity to discuss and communicate on a common basis. We proposed the ReMIS Cloud, an information management system which allows distributed data sources to be searched using dynamic joins of result from heterogeneous data formats. This novel way of information retrieval enables data owners to cross connect domain-extrinsic knowledge and enhances collaborative learning with a search interface that is intuitive and easy to operate. We finally want to address some perspectives to improve the current system and offer further research challenges.

- **Handling of untrustworthy data sources:**
  Allowing data owners to add their data to the cloud autonomously offers a rich pool of information. However, different data owners have access to data of different quality levels. Even wrong data can be contributed to the system. An extension to report untrustworthy data sources or mark data sources as spam might be useful, so that they are not or less considered during the data retrieval.

- **Data caching:**
  If a data source, that is frequently queried and used as an intermediate link between different Categories, is temporary offline, it might disrupt the information retrieval. The information management system should be improved to cache often queried parameters and their results. This could probably also improve the retrieval time.

- **Extended search options:**
  Currently, it is only possible to search for exact matches of parameters. For certain types like dates, numbers, or coordinates more flexibility would be gainful. For example, range queries would improve the usability for the ReMIS Cloud by considering matches with locations in a spatial proximity or timestamps in a certain time interval.

- **Trade-off between quality and quantity:**
  Another important task to evaluate is the trade-off between quality and quantity of acquired data. ReMIS Cloud needs mechanisms to increase the quality of information (besides establishing interconnections).

7.2 Discussion

At the beginning of our studies, we started with reimplementing OSSOBOOK, an existing database for zooarchaeological content and extended the application with further features that were necessary in the immediate work environment of our palaeontological department. Leaving the informatic aspects out of consideration, we gained more and more insights into the archaeological and bioarchaeological workflow, discussed with domain experts from the palaeontological and other archaeo-related sub-disciplines, figured out the necessities within the fields, and got impressions from technical solutions of departments and institutions of other archaeological areas or federal states of Germany. We got revealing insights into the methodical and technical work of the archaeo-related sciences.

The deeper we immersed in this subject matter, the clearer it became that there is to solve a significant structural problem within the archaeo-related disciplines:

1. Archaeo-related data – independent of the type of data – is stored in heterogeneous, individual software solutions. These are either developed from scratch by the departments and institutions that use and operate them or are based on commercial or open source products that most often have to be extended with necessary modules or extensions. The development of these applications, tools, or modules causes a high implementation effort.

2. This distributed development causes numerous individual approaches for the data storage of similar kinds of data. Reasons like different (sub-)disciplines, non-uniformly standards, and different research questions also contribute to this situation. Consequently archaeology and bioarchaeology deal with different, heterogeneous data structures and are only comparable with difficulties, or not at all.

3. The hampered comparability causes essential problems for the long-term archiving of digital, archaeo-related data. Not all types of data are archivable, it has to be guaranteed that the archived data is also accessible and readable for the long-term perspective [Ver17]. Furthermore, the archiving context struggles with the ongoing introduction of hardware and software, discontinued products, ever-changing file types, or outdated systems.

Even though a standardized collection of data is desired in the archaeo-related domains, the definition of common standards is generally neither possible nor practicable. There are too many approaches and circumstances that have to be considered. The differences are already significant whether the intention of the application is for inventory, collection, research, or special reasons. Even for alleged simple standardizations like the

definition of common-used thesauri there is a discrepancy among the experts. Especially excavation site technicians cannot describe all detailed circumstances with predefined thesauri. They demand free text fields where they are able to describe the context in unrestricted detail, even though it is generally clear that standardized systems are important for comparability and automatic analyses. [Gö18a]

From our perspective standardizations make quite sense as long as there is a highest instance that is able to specify and enforce them and the context allows it. This is especially reasonable for inventory databases or data acquisition software like e.g. EXCABOOK. But standardized systems are even not usual within one state of Germany: While in Bavaria the excavation documentation is specified by the *Bavarian State Department of Monuments and Sites*[28] [Bay16c], there are often different approaches at regional level [Lan18] [Ver06] [LRB+17] – like in Rhineland-Palatinate where even four different standards exist in four cities of the state[29] (Koblenz[30], Mainz[31], Speyer[32] [Gen07], and Trier[33]). Already the thought of an homogeneous, standardized, country-wide system is hard to imagine – a continent- or even worldwide standardized system is rather utopia than a realistic thought.

In scientific databases the research questions, the excavation methods, and the analysis possibilities define the kind of gathering. If you would like to develop a system that reach and attract as many scientists as possible, it is necessary to afford them the freedom to record data as they need it. Standardizations are too restrictive or even not possible. As an example, in OSSOBOOK there are two parallel ways to determine the bone portion: one is based on an updated model of the *IPNA Basel*[34] ([Uer78], revised [Sch98]), the other one on the Diagnostic Zones model [HBC+03]. Further on, especially for excavation documentations there are in use several application to store data measured by a tachymeter, by using Computer-Aided Design (CAD) software, like *AutoCAD*[35] or *ArchaeoCAD*[36], or one of numerous Geographic Information Systems (GIS), for example *ArcGIS*[37] or the open-source GIS projects *QGIS*[38], *uDig*[39], *iDAI.field 2.0* [CGdO+17], or the conceptualized *TachyGIS* [Gö18b].

The solution has to be not to build a homogeneous system that handles all data

[28] http://www.blfd.bayern.de
[29] http://gdke-rlp.de/index.php?id=landesarchaeologie
[30] http://www.archaeologie-koblenz.de
[31] http://www.archaeologie-mainz.de
[32] http://www.archaeologie-speyer.de
[33] http://www.archaeologie-trier.de
[34] http://ipna.unibas.ch
[35] http://www.autodesk.eu/products/autocad/
[36] http://www.arctron.de/de/produkte/software/archaeocad/
[37] http://www.arcgis.com
[38] http://www.qgis.org/en/
[39] http://udig.refractions.net

the same way, but methods to make different formats and systems comparable. Such methods are necessary anyway. Systems, hardware, software, measurement data, and file formats change over time or are replaced by other tools or systems. It is crucial to handle these changes to enable comparing old data with current one – in future as well.

To give two examples: First, already today raw data of 20–30 years old 3D scanners cannot be read out anymore, because the related software was not being updated and is not compatible on modern computers anymore. [Sch18] However it became unusable in the meantime, the raw data is actually archived for the future. Second, in recent years the software to record site plans of excavations was mainly switched from CAD (Computer-Aided Design) to GIS (Geographic Information System) systems in many cases [Bib18]. Caused by the big amount of different, individual solutions which all have to be migrated to the new software standard, several different independent methods are developing.

However, data has to be made long-term readable for the archiving. This could be reached by updating the software to further on keeping the data compatible. Especially in the case of commercial products there is a significant risk that the maintenance and development of the applications is stopped. Alternatively, the data format has to be converted to a new, different, more modern format with as less data loss as possible. However, the more different software and data solutions are developed, the more difficult and more complicated are the compatibility issues of the data – it is rather probable that single formats cannot be longer considered. [Gö18a] [GB18]

The question rises if it is really useful to invest valuable resources like time and money in different computer programs and database solutions for different areas of application with different standards at different locations. More reasonable would be investing in a common, modular framework – considering that all archaeological and bioarchaeological disciplines, despite the differences, deal with the same problems.

The XBOOK framework is an example how such a framework could be realizable. All scientists would benefit of the common development of the framework and its features. For the data input, the framework can provide pre-defined, reusable input fields that could even use the same thesauri (cf. Chapter 2). Discussions with domain experts on conferences and workshops showed that a synchronization method, as it is implemented in XBOOK (cf. Chapter 3.1), is required in many database solutions – the reasons are exactly the same than they are in OSSOBOOK: As a possibility for a data exchange, the provision of collaborating methods with colleagues, and the necessity of an offline mode to be able to gather data directly on the excavation site. However, any kind of synchronization method is only rarely implemented in the existing software solutions. It is reasonable not having to develop a synchronization method again and again, but once for a framework that – if such is necessary – can be used for any application based on this framework. Provided analysis methods, like the ones described in Chapter 4, could also be shared with other scientists. In the end, all features that were implemented so

far and that will be implemented in future can contribute towards a powerful tool.

Even the long-term archiving benefits from a common framework: In one place it can be ensured that the stored data is readable in the long-term. Either the readable methods could be implemented directly in the provided framework (that ensures a totally independent solution compared to commercial solutions) or data could be uniformly converted by methods defined and implemented in the framework – possible is the conversion of the data into a more modern format or new formal standards and maybe additionally also the conversion vice versa.

Such a framework, mind you, does not mean that it would solve the problem with heterogeneous data. It is in the nature of things and will further exist anyway. Even if the different excavation methods would be standardized and the differences in gathering data and the used data formats would decrease by such a framework, it is realistic to state that the differences will never disappear. Data will continue to be stored heterogeneous and distributed – caused by regional needs (countries, states, cities, etc.) or technical reasons.

That is the reason why data acquisition methods remain very important. Architectures like the *Reverse-Mediated Information System* (cf. Chapter 3.2) and ReMIS Cloud (cf. Chapter 6) have proved that related data sets can be brought together, independent on a certain file format or a specific location of the data source.

Certainly, such a modular framework to this extent is a complex project. However, if the available time and money do not go to hundreds of individual software solutions, but are invested together in such a solution of this vein, it is finally the archaeo-related domain that benefits of this approach: Independence of commercial products and long-time archiving, comparability, and analyzability of the data.

References

[AK03] Stanley H. Ambrosea and John Krigbaum. Bone chemistry and bioarchaeology. *Journal of Anthropological Archaeology*, 22:193–199, 2003.

[Bay16a] Bayerisches Landesamt für Denkmalpflege. *Denkmalschutz und Denkmalpflege in Bayern 2020. Bewahren durch Erklären und Unterstützen*. Munich, Germany, 2016. Denkmalpflege Themen, 6, 2016.

[Bay16b] Bayerisches Landesamt für Denkmalpflege. *Vorgaben zur Dokumentation archäologischer Ausgrabungen in Bayern. August 2016*. Munich, Germany, 2016. Available from: `http://www.blfd.bayern.de/medien/dokuvorgaben_august_2016.pdf`.

[Bay16c] Bayerisches Landesamt für Denkmalpflege. *Vorgaben zur Fundbehandlung auf archäologischen Ausgrabungen in Bayern. August 2016*. Munich, Germany, 2016. Available from: `http://www.blfd.bayern.de/medien/fundvorgaben_2016.pdf`.

[Bib18] David Bibby. Paradigmenwechsel in der Digitalen Grabungsdokumentation im Landesamt für Denkmalpflege Baden-Württemberg: Von CAD zum Open Source GIS., 2018. Presentation at "Workshop Digitale Grabungsdokumentation – objektiv und nachhaltig", February 1–2, 2018, Dresden, Germany, slides available from: `http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_DGD-WS/V_Bibby_PARA.pdf`.

[BKLW99] Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser, and Herbert Weber. Federated Information Systems: Concepts, Terminology and Architecture. *Forschungsberichte des Fachbereichs Informatik, TU Berlin*, 99(9), 1999.

[Bra17] Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, RFC Editor, December 2017. [Online; accessed 23-January-2018].

[BSO] BSON. http://bsonspec.org/. [Online; accessed 23-January-2018].

[Bus02] Susanne Busse. Modellkorrespondenzen für die kontinuierliche Entwicklung mediatorbasierter Informationssysteme. Dissertation, TU Berlin, Berlin, Germany, 2002.

[BW76] S. Bishop and J. D. Wilcock. Archaeological Context Sorting by Computer: The Strata Program. *Science and Archaeology*, 17:3–12, 1976.

[CGdO+17] Sebastian Cuy, Philipp Gerth, Daniel de Olivera, Thomas Kleinke, and Jan Wieners. iDAIfield 2.0: Modern Apprach To Distributed Fieldwork Documentation. In *22nd International Conference on Cultural Heritage and New Technologies, Vienna, Austria, 2017*, 2017.

[CGMH+94] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Information Processing Society of Japan (IPSJ 1994), October 1994*, 1994.

[DAM15] Subrata Das, Ria Ascano, and Matthew Macarty. Distributed Big Data Search for Analyst Queries and Data Fusion. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 666–673, 2015.

[Dan10] Svetlana Danti. Cluster Analysis of Features of Animal Bones and Similarity Search on Multi Instance Objects of the Archaeozoological Data Pool. Diploma thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2010.

[dat12] datapine. datapine: Data Visualization & Business Intelligence Tool. https://www.datapine.com/, 2012. Online, retrieved 19 January 2017.

[Dat16] Data Xtractor. Query Xtractor v1.0. https://data-xtractor.com/query-xtractor/, 2016. Online, retrieved 19 January 2017.

[Gö18a] Reiner Göldner. Resümee zum Workshop 'Digitale Grabungsdokumentation – objektiv und nachhaltig', 2018. Available from: http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_DGD-WS/Res%C3%BCmee.pdf.

[Gö18b] Reiner Göldner. "TachyGIS" – Ideen zur archäologischen Grabungsvermessung mit Tachymeter und GIS, 2018. Presentation at "Workshop Digitale Grabungsdokumentation – objektiv und nachhaltig",

February 1–2, 2018, Dresden, Germany, slides available from: http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_DGD-WS/V_Goeldner1_TachyGIS_Praesentation.pdf.

[GB18] Reiner Göldner and David Bibby. Workshop Summary "Digital Excavation Documentation – Objective and Sustainable. In *2018 Computer Applications and Quantitative Methods in Archaeology (CAA) International Conference, Tübigen, Germany, March 19–23, 2018*, 2018. Summary available from: http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_DGD-WS/DGD-Workshop_Summary.pdf.

[Gen07] Generaldirektion Kulturelles Erbe (GDKA) Rheinland-Pfalz. *Ausgrabungs- und Dokumentationsrichtlinien*. Landesarchäologie Außenstelle Speyer, Germany, 2007. Available from: http://download.gdke-rlp.de/archaeologie/richtlinien_ausgrabung.pdf.

[GFHW10] Carmen L.Z. Gress, Meghann Fior, Allyson F. Hadwin, and Philip H. Winne. Measurement and assessment in computer-supported collaborative learning. *Computers in Human Behavior*, 26(5):806–814, 2010.

[GHH+17] Reiner Göldner, Irmela Herzog, Ulrich Himmelmann, Axel G. Posluschny, Thomas Richter, and Mathias Wilbertz. *ADeX – Archaeology and Information Systems*. Verband der Landesarchäologen Deutschlands, Münster, Germany, 2017. Available from: http://www.landesarchaeologen.de/verband/kommissionen/archaeologie-und-informationssysteme/projektearbeitsgruppen/adex/.

[GHK+15] Raymond Gallardo, Scott Hommel, Sowmya Kannan, Joni Gordon, and Sharon Biocca Zakhour. *The Java® Tutorial. A Short Course on the Basis. 6th Edition*, pages 124–125. Addison-Wesley, 2015.

[GHM15] Gisela Grupe, Michaela Harbeck, and George C. McGlynn. *Prähistorische Anthropologie*. Springer Spektrum, Berlin/Heidelberg, 2015.

[GHW11] Reiner Göldner, Irmela Herzog, and Roland Wanninger. ADeX Version 2.0 – Archäologischer Daten-Export – Standard für den Austausch archäologischer Fachdaten, 2011. Available from: http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_AIS_ADeX/ADeX_2-0_Doku.pdf.

[GNU] The GNU C Library. Sockets. http://www.gnu.org/savannah-checkouts/gnu/libc/manual/html_node/Sockets.html. [Online; accessed 23-January-2018].

[Goe02] Brian Goetz. Java theory and practice: Thread pools and work queues. https://www.ibm.com/developerworks/java/library/j-jtp0730/index.html, 2002. [Online; accessed 06-December-2017].

[Gol14] Joachim Goll. *Architektur- und Entwurfsmuster der Softwaretechnik*. Springer, 2014.

[Gooa] Google Developers. FlatBuffers. https://google.github.io/flatbuffers/. [Online; accessed 23-January-2018].

[Goob] Google Developers. Protocol Buffers. https://developers.google.com/protocol-buffers/. [Online; accessed 23-January-2018].

[GS02] Rajat P. Garg and Ilya Sharapov. *Techniques for Optimizing Applications: High Performance Computing*. Prentice Hall Professional Technical Reference, 2002.

[Har75] Edward C. Harris. Stratigraphic Analyses and the Computer. *Computer Applications in Archaeology*, 3:33–40, 1975.

[Har89] Edward C. Harris. *Principles of Archaeological Stratigraphy*. Academic Press, London and San Diego, 1989.

[HBC+03] Jennifer F. Harland, James H. Barrett, John Carrott, Keith Dobney, and Deborah Jaques. The York system: An integrated zooarchaeological database for research and teaching. *Internet Archaeology*, 13:149–167, 2003.

[Her06] Irmela Herzog. No News from Stratigraphic Computing? In *Workshop 10 "Archäologie und Computer. Kulturelles Erbe und neue Technologien" held in Vienna, Austria, 2006*, 2006.

[Her08] Irmela Herzog. Stratify 1.5. http://www.stratify.org/, 2008. Computer Software.

[Her11] Irmela Herzog. Possibilities for Analysing Stratigraphic Data. In *Workshop "Archäologie und Computer" held in Vienna, Austria, 2011*, 2011.

[HIB91] Edward C. Harris, Marley R. Brown III, and Gregory J. Brown. *Practices of Archaeological Stratigraphy*. Academic Press, London and San Diego, 1991.

[HLSE15] Florian Haag, Steffen Lohmann, Stephan Siek, and Thomas Ertl. Visual Querying of Linked Data with QueryVOWL. In *Joint Proceedings of the 1st International Workshop on Summarizing and Presenting Entities and Ontologies and the 3rd International Workshop on Human Semantic Web Interfaces (SumPre 2015, HSWI 2015) co-located with the 12th Extended Semantic Web Conference (ESWC 2015), Portoroz, Slovenia, June 1, 2015.*, 2015.

[HM85] Dennis Heimbigner and Dennis McLeod. A federated architecture for information management. *ACM Transactions on Information Systems*, 3(3):253–278, 1985.

[HMP+15] Christoph Hundack, Petra Mutzel, Igor Pouchkarev, Barbara Reitgruber, Barbara Schuhmacher, and Stefan Thome. ArchEd. A program for drawing Harris Matrices. `https://www.ac.tuwien.ac.at/files/archive/ArchEd/`, 2015.

[Ima] Imagination Computer Services. Harris Matrix Composer. `http://www.harrismatrixcomposer.com`. Computer Software.

[Inc01] Klipfolio Inc. Klipfolio: How it works. `https://www.klipfolio.com/features`, 2001. Online, retrieved 19 January 2017.

[Inm05] William H. Inmon. *Building the Data Warehouse, 4th*. John Wiley & Sons, 2005.

[Jan18] Silke Jantos. ExcaBook – Grabungsdatenbank des BLfD, 2018. Presentation at "Workshop Digitale Grabungsdokumentation – objektiv und nachhaltig", February 1–2, 2018, Dresden, Germany, slides available from: `http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_DGD-WS/V_Jantos_ExcaBook.pdf`.

[JLVV03] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Fundamentals of Data Warehouses, 2nd*. Springer, 2003.

[Jon16] Alex Jones. Top 10 Data Analysis Tools for Business. `http://www.kdnuggets.com/2014/06/top-10-data-analysis-tools-business.html`, 2016. Online, retrieved 3 May 2016.

[JSO17] Standard ECMA-404: The JSON Data Interchange Syntax. 2nd Edition (December 2017). `http://www.ecma-international.org/publications/standards/Ecma-404.htm`, 2017. [Online; accessed 23-January-2018].

[Kal11] Daniel Kaltenthaler. Design and Implementation of a Graphical User Interface for the Archaeozoological Database OssoBook. Project thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2011.

[Kal12] Daniel Kaltenthaler. Visual Cluster Analysis of the Archaeological Database OssoBook with special focus on Data Integrity and Consistency. Diploma thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2012.

[KKFB96] Timothy Koschmann, Ann C. Kelson, Paul J. Feltovich, and Howard S. Barrows. Computer-supported problem-based learning: A principled approach to the use of computers in collaborative learning. In Timothy Koschmann, editor, *CSCL: Theory and practice of an emerging paradigm*, pages 83–124. Routledge, New York, 1996.

[KKO+09] Hans-Peter Kriegel, Peer Kröger, Henriette Obermaier, Joris Peters, Matthias Renz, and Christiaan van der Meijden. OSSOBOOK: database and knowledgemanagement techniques for archaeozoology. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 2091–2092, 2009.

[KL18] Daniel Kaltenthaler and Johannes-Y. Lohrer. The Historic Development of the Zooarchaeological Database OssoBook and the xBook Framework for Scientific Databases. *ArXiv e-prints: 1801.08052*, January 2018.

[KLK+15] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. Synchronized Data Management and Its Integration into a Graphical User Interface for Archaeological Related Disciplines. In *Design, User Experience, and Usability: Users and Interactions - 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part II*, pages 317–329, 2015.

[KLK17] Daniel Kaltenthaler, Johannes-Y. Lohrer, and Peer Kröger. Supporting Domain Experts Understanding Their Data: A Visual Framework for Assembling High-Level Analysis Processes. In *11th International Conference on Interfaces and Human Computer Interaction 2017, Lisbon, Portugal, 2017*, pages 217–221, 2017.

[KLK18a] Daniel Kaltenthaler, Johannes-Y. Lohrer, and Peer Kröger. ReMIS and ReMIS Cloud: Information Systems for Retrieving Disciplinary and Interdisciplinary Data. In *Proceedings of 20th International Conference on Human-*

Computer Interaction, Las Vegas, NV, USA, July 15-20, 2018, (to appear) 2018.

[KLK+18b] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, Eduardo Granado, Jana Lamprecht, Florian Nücke, Henriette Obermaier, Barbara Stopp, Isabelle Baly, Cécile Callou, Lionel Gourichon, Joris Peters, Nadja Pöllath, and Jörg Schiebler. OssoBook v5.6.2. Software, Munich, Germany; Basel, Switzerland, 2018.

[KLK+18c] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, Michaela Harbeck, and Andrea Grigat. AnthroDepot (Dev version). Software, Munich, Germany, 2018.

[KLK+18d] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, and Ciarán Harrington. InBook (Dev version). Software, Munich, Germany, 2018.

[KLK+18e] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, Ciarán Harrington, Erich Claßen, Rupert Gebhard, Sonja Marzinzik, and Heiner Schwarzberg. ArchaeoBook v5.6.2. Software, Munich, Germany, 2018.

[KLK+18f] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, Silke Jantos, Agnes Rahm, Ina Sassen, Tilman Wanke, Roland Wanninger, Jochen Haberstroh, and Sebastian Sommer. ExcaBook v5.6.2. Software, Munich, Germany, 2018.

[KLK+18g] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. AnthroDepot v5.6.2. Software, Munich, Germany, 2018.

[KLK+18h] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, Tatiana Sizova, Anja Mösch, Michaela Harbeck, Andrea Grigat, and Anita Toncala. AnthroBook. Software, Munich, Germany, 2018.

[KLKO17] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, and Henriette Obermaier. A Framework for Supporting the Workflow for Archaeo-related Sciences: Managing, Synchronizing and Analyzing Data. In *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs ,,Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Workshopband*, pages 89–98, 2017.

[KLKO18] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, and Henriette Obermaier. xBook, a Framework for Common Scientific Databases. In *Proceedings of 20th International Conference on Human-Computer Interaction, Las Vegas, NV, USA, July 15-20, 2018*, (to appear) 2018.

[KLP+17] Daniel Kaltenthaler, Johannes-Y. Lohrer, Ptolemaios Paxinos, Daniel Hämmerle, Henriette Obermaier, and Peer Kröger. TaRDIS, a Visual Analytics System for Spatial and Temporal Data in Archaeo-related Disciplines. In *13th IEEE International Conference on e-Science, eScience 2017, Auckland, New Zealand, October 24-27, 2017*, pages 345–353, 2017.

[KLRK17] Daniel Kaltenthaler, Johannes-Y. Lohrer, Florian Richter, and Peer Kröger. ReMIS Cloud: A Distributed Information Management System for Interdisciplinary Knowledge Linkage. In *8th International Conference on Internet Technologies & Society 2017, Sydney, NSW, Australia, 2017*, pages 107–114, 2017.

[KLRK18] Daniel Kaltenthaler, Johannes-Y. Lohrer, Florian Richter, and Peer Kröger. Interdisciplinary Knowledge Cohesion through Distributed Information Management Systems. *Journal of Information, Communication and Ethics in Society*, (to appear) 2018.

[KNI06] KNIME.com. KNIME Analytics Platform. `http://www.knime.org/knime`, 2006. Online, retrieved 6 May 2016.

[Lam08] Jana Lamprecht. Conception and Implementation of a Intermittently Synchronized Database System for Palaeoanatomic applications. Diploma thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2008.

[Lan18] Landesamt für Denkmalpflege im Regierungspräsidium Stuttgart. *Richtlinien für Grabungsfirmen und Investoren zur Durchführung archäologischer Ausgrabungen und Prospektionen in Baden-Württemberg. 1. Fassung*. Stuttgart, Germany, 2018. Available from: `https://www.denkmalpflege-bw.de/fileadmin/media/denkmalpflege-bw/geschichte-auftrag-struktur/firmenarchaeologie/allgemein/richtlinie_bw_201801.pdf`.

[LHJ16] Jingyan Lu, Qiang Hao, and Mengguo Jing. Consuming, sharing, and creating content: How young students use new social media in and outside school. *Computers in Human Behavior*, 64:55–64, 2016.

[LKK+14] Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. A Generic Framework for Synchronized Distributed Data Management in Archaeological Related Disciplines. In *10th IEEE International Conference on e-Science, eScience 2014, São Paulo, Brazil, October 20-24, 2014*, pages 5–12, 2014.

[LKK16a] Johannes-Y. Lohrer, Daniel Kaltenthaler, and Peer Kröger. Leveraging Data Analysis for Domain Experts: An Embeddable Framework for Basic Data Science Tasks. In *7th International Conference on Internet Technologies & Society 2016, Melbourne, VIC, Australia, 2016*, pages 51–58, 2016.

[LKK+16b] Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. A generic framework for synchronized distributed data management in archaeological related disciplines. *Future Generation Computer Systems*, 56:558–570, 2016.

[LKK+17] Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Henriette Obermaier, and Christiaan van der Meijden. Reverse Mediated Information System: Web-based Retrieval of Distributed, Anonymous Information. In *16th International Conference on WWW/Internet 2017, Vilamoura, Portugal, 2017*, pages 63–70, 2017.

[LKR+18] Johannes-Y. Lohrer, Daniel Kaltenthaler, Florian Richter, Tatiana Sizova, Peer Kröger, and Christiaan van der Meijden. Retrieval of Heterogeneous Data from Dynamic and Anonymous Sources. In *8th IEEE International Conference Confluence 2018 on Cloud Computing, Data Science and Engineering, Noida, Uttar Pradesh, India*, pages 592–597, 2018.

[Loh11] Johannes-Y. Lohrer. Design and Implementation of a Dynamic Database for Archaeozoological Applications. Project thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2011.

[Loh12] Johannes-Y. Lohrer. Density Based Cluster Analysis of the Archaeological Database OssoBook in Condideration of Aspects of Data Quality. Diploma thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2012.

[LRB+17] Jens Lehmann, Bernd Rasink, Utz Böhner, Henning Haßmann, and Andreas Niemuth. *Richtlinien zur Dokumentation archäologischer Maßnahmen / Ausgrabungen. Stand August 2017*. Niedersächsisches Landesamt für Denkmalpflege, Hannover, Germany, 2017. Available from: `https://www.denkmalpflege.niedersachsen.de/download/110131/`

Richtlinien_zur_Dokumentation_Archaeologischer_Massnahmen_Ausgrabungen_in_Niedersachsen_August_2017.pdf.

[MAK12] Wolfram Meier-Augenstein and Helen F. Kemp. Stable Isotope Analysis: General Principles and Limitations. *Wiley Encyclopedia of Forensic Science*, 2012.

[Mat] MathWorks. MATLAB. https://www.mathworks.com. Computer Software.

[McA13] Colt McAnlis. JSON Compression: Transpose & Binary. http://mainroach.blogspot.de/2013/08/json-compression-transpose-binary.html, 2013. [Online; accessed 23-January-2018].

[Mic08] Microsoft. Microsoft SQL Server: Replication Features and Tasks. http://technet.microsoft.com/en-us/library/bb677158.aspx, 2008. Online, retrieved 21 January 2015.

[MK17] Renáta McDonnell and Simon Kerridge. Research Information Management System (KRIMSON) at Kent. *Procedia Computer Science*, 106:160–167, 2017.

[MNKG15] Markus Mauder, Eirini Ntoutsi, Peer Kröger, and Gisela Grupe. Data Mining for Isotopic Mapping of Bioarchaeological Finds in a Central European Alpine Passage. In *Proceedings of the 27th International Conference on Scientific and Statistical Database Management, SSDBM '15, La Jolla, CA, USA, June 29 - July 1, 2015*, pages 34–39, 2015.

[MyS13] MySQL. MySQL 5.7 Reference Manual: Replication. http://dev.mysql.com/doc/refman/5.7/en/replication.html, 2013. Online, retrieved 21 June 2015.

[Nai82] John Naisbitt. *Megatrends: Ten New Directions Transforming Our Lives*. Grand Central Pub, 6th edition, 1982.

[Neu12] Tanja Neumayer. Design and Implementation of Analysis Methods for Archaeozoological Data. Bachelor thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2012.

[Ora13a] Oracle. Oracle Documentation Library: Replication Manual. http://docs.oracle.com/cd/F49540_01/DOC/server.815/a67791/pref.htm, 2013. Online, retrieved 21 January 2015.

[Ora13b] Oracle. Oracle Streams: Part 1 Oracle Streams Concepts. http://docs.oracle.com/cd/B28359_01/server.111/b28321/pt_concepts.htm#i996787, 2013. Online, retrieved 21 January 2015.

[Ora13c] Oracle. Oracle Streams Recplication Administator's Guide: Understanding oracle Streams Replication. http://docs.oracle.com/cd/B28359_01/server.111/b28322/gen_rep.htm#STREP011, 2013. Online, retrieved 21 January 2015.

[OV11] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems, 3rd*. Springer, 2011.

[Qli05] Qlik. Qlik Sense. https://www.qlik.com/us/products/qlik-sense, 2005. Online, retrieved 19 January 2017.

[Rap06] RapidMiner. RapidMiner. http://www.rapidminer.com, 2006.

[Rya88] Nick S. Ryan. Browsing through the stratigraphic record. In Sebastian P. Q. Rahtz, editor, *Computer and Quantitative Methods in Archaeology. BAR International Series 446(2)*, pages 327–334. 1988.

[Rya95] Nick S. Ryan. The excavation archive as hyperdocument? In Jeremy Huggett and Nick S. Ryan, editors, *Computer Applications and Quantitative Methods in Archaeology 1994. BAR International Series 600*, pages 211–220. 1995.

[Sch98] Jörg Schibler. OSSOBOOK, a database system for archaeozoology. In *Man and the Animal World: Studies in Archaeozoology, Archaeology, Anthropology and Palaeolinguistics in Memoriam Sándor Bökönyi*, pages 491–510, 1998.

[Sch18] Christof Schubert. Digitale Grabungsdokumentation in Sachsen aus grabungstechnischer Sicht, 2018. Presentation at "Workshop Digitale Grabungsdokumentation – objektiv und nachhaltig", February 1–2, 2018, Dresden, Germany, slides available from: http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_DGD-WS/V_Schubert_Grabungsdokumentation.pdf.

[SCL16] Yao-Ting Sung, Kuo-En Chang, and Tzu-Chien Liu. The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. *Computers & Education*, 94:252–275, 2016.

[SKS06] Gerry Stahl, Timothy Koschmann, and Dan Suthers. Computer-supported collaborative learning: An historical perspective. In Keith K. Sawyer, editor, *Cambridge handbook of the learning sciences*, pages 409–426. Cambridge University Press, Cambridge, 2006.

[SL90] Amit P. Sheth and James A. Larsen. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surverys*, 22(3):183–236, 1990.

[SQL12] SQLeo. SQLeo Visual Query Builder: Documentation. `http://sqleo.sourceforge.net/guide.htm`, 2012. Online, retrieved 19 January 2017.

[Ste46] Stanley Smith Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):677–680, 1946.

[Swe02] Latanya Sweeney. Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[Tab03] Tableau Software. Tableau Public. `http://public.tableau.com`, 2003.

[TP15] Simon Trixl and Joris Peters. Unscheinbar und doch unverzichtbar. Tierknochen in der Diskussion um den Übergang von der Spätlatène in die römische Kaiserzeit. *Bayerische Archäologie*, 3:34–37, 2015.

[TRV98] Anthony Tomasic, Louiqa Raschid, and Patrick Valduriez. Scaling Access to Heterogeneous Data Sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):808–823, 1998.

[Tsu10] Yuliya Tsukanava. Development and Appliance of Data Mining Methods on the Palaeoanatomic Data Collection. Diploma thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2010.

[Uer78] Hans-Peter Uerpmann. The 'Knocod' System for Processing Data on Animal Bones from Archaeological Sites. *Peabody Museum Bulletin*, 2:149–167, 1978.

[vdDK14] Angela von den Driesch and Mostefa Kokabi. *Densdrochronological report*. Bayerisches Landesamt für Denkmalpflege, Munich, Germany, 2014.

[vdM12] Christiaan van der Meijden. The Internet of Databases – Generalizing the Archaeo Informatics Approach, 2012. Online Video, available from: `https://www.microsoft.com/en-us/research/video/the-internet-of-databases-generalizing-the-archaeo-informatics-approach/`.

[Ver01] Verband der Landesarchäologen Deutschlands. *Leitlinien zur Archäologischen Denkmalpflege in Deutschland*. Bonn, Germany, 2001. Available from: `http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Verband/Leitlinien_archaeol_Dkmpf.pdf`.

[Ver06] Verband der Landesarchäologen Deutschlands. *Ausgrabungen und Prospektion. Durchführung und Dokumentation*. Münster, Germany, 2006. Available from: `http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Grabungstechniker/grabungsstandards_april_06.pdf`.

[Ver17] Verband der Landesarchäologen Deutschlands. *Themenblätter zur Archivierung digitaler Daten – Archivwürdigkeit*. Münster, Germany, 2017. Available from: `http://www.landesarchaeologen.de/verband/kommissionen/archaeologie-und-informationssysteme/projektearbeitsgruppen/archivierung/`.

[WBKK16] Charlotte A. Weaver, Marion J. Ball, George R. Kim, and Joan M. Kiel. *Healthcare information management systems*. Cham: Springer International Publishing, 2016.

[Wes14] Timm Weski. *Densdrochronological report*. Bayerisches Landesamt für Denkmalpflege, Munich, Germany, 2014.

[Wes15] Patrick Wessa. Kernel Density Estimation (v1.0.12). In Free Statistics Software (v1.1.23-r7), `http://www.wessa.net/rwasp_density.wasp`, 2015. Office for Research Development and Education.

[Wie92] Gio Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

[Wie94] Gio Wiederhold. Interoperation, Mediation, and Ontologies. In *Proc. Int. Symposium on 5th Generation Computer Systems (FGCS'94), Workshop on Heterogeneous Cooperative Knowledge-Bases. Tokyo, Japan, 1994*, pages 33–48, 1994.

[Wie13] Gio Wiederhold. Mediators, Concepts and Practice. In Tansel Özyer, Keivan Kianmehr, Mehmet Tan, and Jia Zeng, editors, *Information Reuse and Integration In Academia And Industry*, pages 1–27. Springer, Vienna, 2013.

[XFJ] JFJSON. `https://github.com/mainroach/compression/tree/master/xfjson`. [Online; accessed 23-January-2018].

Own Publications

Main Publications

[KLK+15] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. Synchronized Data Management and Its Integration into a Graphical User Interface for Archaeological Related Disciplines. In *Design, User Experience, and Usability: Users and Interactions - 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part II*, pages 317–329, 2015.

[KLKO17] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, and Henriette Obermaier. A Framework for Supporting the Workflow for Archaeo-related Sciences: Managing, Synchronizing and Analyzing Data. In *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs ,,Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Workshopband*, pages 89–98, 2017.

[KLK17] Daniel Kaltenthaler, Johannes-Y. Lohrer, and Peer Kröger. Supporting Domain Experts Understanding Their Data: A Visual Framework for Assembling High-Level Analysis Processes. In *11th International Conference on Interfaces and Human Computer Interaction 2017, Lisbon, Portugal, 2017*, pages 217–221, 2017.

[KLP+17] Daniel Kaltenthaler, Johannes-Y. Lohrer, Ptolemaios Paxinos, Daniel Hämmerle, Henriette Obermaier, and Peer Kröger. TaRDIS, a Visual Analytics System for Spatial and Temporal Data in Archaeo-related Disciplines. In *13th IEEE International Conference on e-Science, eScience 2017, Auckland, New Zealand, October 24-27, 2017*, pages 345–353, 2017.

[KLRK17] Daniel Kaltenthaler, Johannes-Y. Lohrer, Florian Richter, and Peer Kröger. ReMIS Cloud: A Distributed Information Management System for Interdisciplinary Knowledge Linkage. In *8th International Conference on Internet Technologies & Society 2017, Sydney, NSW, Australia, 2017*, pages 107–114, 2017.

[KL18] Daniel Kaltenthaler and Johannes-Y. Lohrer. The Historic Development of the Zooarchaeological Database OssoBook and the xBook Framework for Scientific Databases. *ArXiv e-prints: 1801.08052*, January 2018.

[KLRK18] Daniel Kaltenthaler, Johannes-Y. Lohrer, Florian Richter, and Peer Kröger. Interdisciplinary Knowledge Cohesion through Distributed Information Management Systems. *Journal of Information, Communication and Ethics in Society*, (to appear) 2018.

Further Publications

[Kal11] Daniel Kaltenthaler. Design and Implementation of a Graphical User Interface for the Archaeozoological Database OssoBook. Project thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2011.

[Kal12] Daniel Kaltenthaler. Visual Cluster Analysis of the Archaeological Database OssoBook with special focus on Data Integrity and Consistency. Diploma thesis, Ludwig-Maximilians-Universität München, Munich, Germany, 2012.

[LKK+14] Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. A Generic Framework for Synchronized Distributed Data Management in Archaeological Related Disciplines. In *10th IEEE International Conference on e-Science, eScience 2014, São Paulo, Brazil, October 20-24, 2014*, pages 5–12, 2014.

[LKK+16b] Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Christiaan van der Meijden, and Henriette Obermaier. A generic framework for synchronized distributed data management in archaeological related disciplines. *Future Generation Computer Systems*, 56:558–570, 2016.

[LKK16a] Johannes-Y. Lohrer, Daniel Kaltenthaler, and Peer Kröger. Leveraging Data Analysis for Domain Experts: An Embeddable Framework for Basic Data Science Tasks. In *7th International Conference on Internet Technologies & Society 2016, Melbourne, VIC, Australia, 2016*, pages 51–58, 2016.

[LKK+17] Johannes-Y. Lohrer, Daniel Kaltenthaler, Peer Kröger, Henriette Obermaier, and Christiaan van der Meijden. Reverse Mediated Information System: Web-based Retrieval of Distributed, Anonymous Information. In *16th International Conference on WWW/Internet 2017, Vilamoura, Portugal, 2017*, pages 63–70, 2017.

[LKR+18] Johannes-Y. Lohrer, Daniel Kaltenthaler, Florian Richter, Tatiana Sizova, Peer Kröger, and Christiaan van der Meijden. Retrieval of Heterogeneous Data from Dynamic and Anonymous Sources. In *8th IEEE International Conference Confluence 2018 on Cloud Computing, Data Science and Engineering, Noida, Uttar Pradesh, India*, pages 592–597, 2018.

[KLKO18] Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, and Henriette Obermaier. xBook, a Framework for Common Scientific Databases. In *Proceedings of 20th International Conference on Human-Computer Interaction, Las Vegas, NV, USA, July 15-20, 2018*, (to appear) 2018.

[KLK18a] Daniel Kaltenthaler, Johannes-Y. Lohrer, and Peer Kröger. ReMIS and ReMIS Cloud: Information Systems for Retrieving Disciplinary and Interdisciplinary Data. In *Proceedings of 20th International Conference on Human-Computer Interaction, Las Vegas, NV, USA, July 15-20, 2018*, (to appear) 2018.

List of Figures

List of Tables

List of Algorithms