Richard Hohensinner, BSc

Integration of a Life Science Research Tool into a Distributed Data Management System

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering & Management

submitted to

Graz University of Technology

Supervisor
Univ.-Prof. Dipl-Ing. Dr. Stefanie Lindstaedt

Co-Supervisors
Sarah Stryeck PhD &
Dipl.-Ing. Konrad Lang

Institute for Interactive Systems and Data Science (ISDS)
Head: Univ.-Prof. Dipl-Ing. Dr. Stefanie Lindstaedt

Graz, April 2022

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

Acknowledgement

This master thesis was created in context of the *CyVerse Austria* project in collaboration of the Know Center and the University of Technology Graz.

First, I want to thank Sarah Stryeck PhD and Dipl.-Ing. Konrad Lang for taking over the role of advising this thesis as well as bringing in their expertise to the project. With this they were an excellent help throughout the course of this thesis. Additionally, I want to thank Dr. Slave Trajanoski for always being approachable and a helpful contact person from the side of the Medical University of Graz. His cooperation greatly benefited the outcome of this thesis' projects. I especially want to thank Univ.-Prof. Stefanie Lindstaedt for making this master thesis possible by supervising and examining it.

Furthermore, I want to thank my friend Inti Gabriel Mendoza Estrada for keeping me motivated as well as being a magnificent companion throughout my days as a student.

Most of all, I want to thank my parents for always believing in me and always supporting me throughout my (many) years of study.

Graz, April 2022

Richard Hohensinner

Abstract

Modern life sciences go hand in hand with computational research tools. In addition, the volume of data used in life sciences increases due to high-throughput methods as well as the large amount of available data via public data repositories. Thus, embedding these computational research tools into technological environments where the (experimental) data are kept, is crucial for efficient and robust workflows and therefore scientific progress. A plethora of aspects need to be considered to ensure reliability and usability of these workflows: (i) data security for any sensitive data (e.g., genetics), (ii) data access for controlled user access, (iii) resources for storage and computation and (iv) interoperability of tools and systems, to name a few.

The main goal of this thesis is the technical integration of an existing computational research tool from the life sciences The Galaxy Project (2021) with an existing data management system CyVerse Austria (2022) to enable an automated workflow for researchers. In order to do so, an interface between Galaxy and iRODS (2021) was developed based on design and implementation principles. This workflow was evaluated by a number of researchers to investigate usability and develop strategies for further development. Finally, the achieved goals were critically reflected and the outlook for further implementations is presented.

Summary

"The end of a melody is not its goal: but nonetheless, had the melody not reached its end it would not have reached its goal either. A parable."

Friedrich Nietzsche

The Medical University of Graz suffers from unoptimzed file transfer protocols. Often employees resort to impractical measures such as using physical storage (e.g., USB sticks) to temporarily store data and physically walking to the desired endpoint to share data. CyVerse Austria aims to provide a way to fix the issue through research data management. Unfortunately, this method does not comply with security restrictions that the Medical University of Graz demands. Thus, this thesis was born out of necessity of a research data management solution that does comply with data security restrictions.

To make an informed decision on what the solution should be, we conducted a literature survey on life sciences and their research tools, as well as distributed data management systems. Through the literature survey, we concluded that the best way to achieve the desired solution is to integrate the research tool Galaxy directly with iRODS, CyVerse Austria's data management system.

In conjunction with the Medical University of Graz, interface tools for Galaxy were developed. This partnership, alongside continuous feedback loops ensured a tool that followed the security guidelines required by the Medical University. The current version of the tool is completely functional with Galaxy, but is not bound to the research tool - it can be embedded into other research tools with minimal effort. This is a testament to the compliance of the project's requirements, which were enforced throughout the project.

Furthermore, to validate the project's results, an evaluation process was conducted. First, through interviews with the project's target user, we ensured that the functional requirements as well as the behavioral requirements were met. Additionally, by conducting an experiment, we observed quantifiable improvements over legacy methods, both in transfer speed and security assurances. Finally, we reflected on the project's life cycle. We identified development pitfalls that will be useful for any further or similar projects. We also recognize our tools' limitations as well as possible areas of improvement. In the end, we concluded that the interface serves its purpose, it is currently being used, and will continue to do so, even after the project finished.

Contents

List of Figures

1 Introduction

"The science of today is the technology of tomorrow"

Edward Teller

Life sciences (alt. Biosciences) comprise all research fields that involve a direct connection to living beings. This includes domains such as medicine, pharmacy, biochemistry, biophysics, bioinformatics, and of course the field of biology itself. Life sciences count as one of the largest groups of today's modern research areas due to their comprehension of many diverse research fields, as well as their methodology of combining these fields in an interdisciplinary way (STANDARD, 2012). It is necessary to limit the range of life sciences in this thesis because mentioning everything in detail would exceed the scope. Thus, we will primarily focus on the biological aspects in combination with medicine as well as the meaning of bioinformatics as a research method for this discipline.

Figure 1.1 illustrates the scientific research fields connected to the domain of life sciences.

Currently, life sciences are part of the most future oriented areas of research. Modern biosciences play a major role in advancing the development of new diagnostic methods, therapies, and medical technology. Because of these rapid developments and perpetually increase of extent, they are proclaimed to enable the key technologies of the current century.

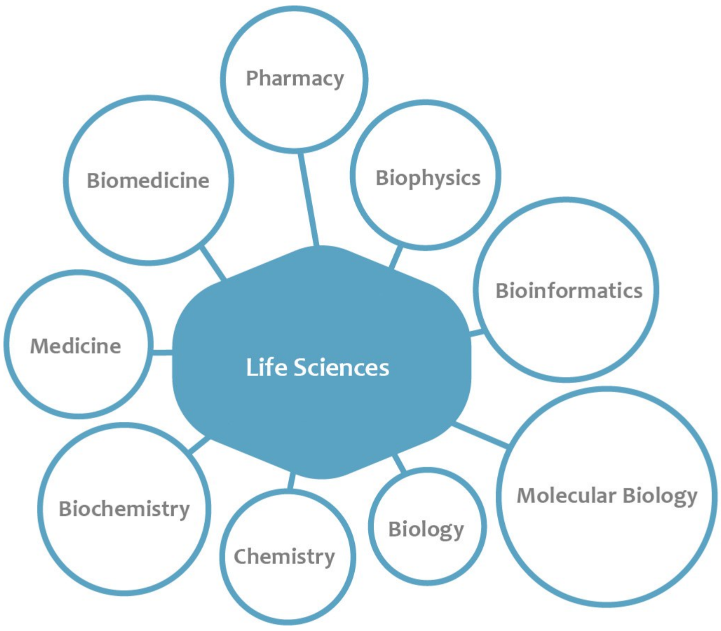


Figure 1.1: Visual representation of the fields of study connected to Life Sciences.

1.1 A brief historical background of Life Sciences

The modern term *life sciences* originates from the scientific discipline *biology*. The term biology was first introduced at the beginning of the nineteenth century and defined biology as the science of living organisms. Needless to say, the sciences related to biology are much older than the nineteenth century. In fact, some of the fields within biology, like medicine, even date back to ancient times. Alongside medicine, topics like plant science and knowledge about animals were combined to create the term that we now call biology. The most important factor in this term derivation was the realisation that all these topics had one thing in common: life. With this new insight, the creation of biology, the science of living organisms or the science of *alive* things, was conducted. Furthermore, this very concept is vital for understanding the rationale of Life Sciences in general (Magner, 2002).

In recent times, due to the vastly increasing knowledge and understanding about life itself, more and more things became connected to biology. An example would be a prosthetic arm. On first thought the arm clearly is not a living thing. Once attached, it is also technically a lifeless extension of the human body. However, with recent developments and new insights, it is possible to connect a prosthetic arm to the human body in a way that it can almost fully be controlled by its wearer, thus making it a bionic prosthesis. It may be possible, down the line, to not only control the arm with the help of the body's nervous system, but also transmit signs of touch, thus allowing the feeling of touching. This would improve the prosthesis to a point where the core functionalities of an arm are recreated. This poses the question whether the bionic prosthesis has now become truly part of a living organism and thus should be considered biotic. This topic falls under the research field of biomedical engineering & bionics, which is one many area of studies linked to life sciences and even philosophy itself.

With the knowledge about the term life sciences and the insights of which other research fields can be connected to biology, it is now possible to explain the rise of the term life sciences. Biology, the umbrella term for the study of all living beings, gets connected to prior not directly related fields of studies, like mechanical engineering, to describe the link between both, being called life sciences. This way, life sciences could be seen as biology 2.0, or an extended version of biology, by involving other disciplines into the context of biology, thus increasing its spectrum and reach.

Today's modern understanding of the term life sciences opens the scope of bio sciences to a much more manifold notion. This also brings up many new questions which emerged with currently thriving developments in the fields. As already mentioned, bionics are a big representative for life science research, along with many other medical treatments and developments. To also give an example of a potential life science topic in connection to computer science, the topic of artificial intelligence (AI) can be proposed. Ever since the idea of AI was first introduced, the question whether such a system could at some point be considered *alive* or not, emerged alongside. Assuming such a self-sustaining system, e.g. an AI robot, that would possess the power of learning, and at some point learn the ability to reproduce (or rebuild) itself, could indeed become terrifyingly close to a living being. In essence, the only difference between an AI model and a human could end up being that the former is made of copper and the latter of organic materials. This again poses the question whether a system like this could be classified as being alive, and if not, might introduce the need of an additional term for almost life-like state of being.

1.2 Life Sciences in Austria

As mentioned in the previous subchapters, life sciences comprise a variety of distinct research areas and researchers all over the world that work on projects to bring innovation and a better understanding to processes of and in living organisms. To show the importance of life sciences in Austria, some important facts and figures are presented. As of 2020, 982 companies are currently working within the life sciences research field in Austria, achieving a summed up turnover of 25.1 billion euros. In comparison to 2017, the workforce in these companies was increased by roughly 9% and the overall increase in revenues was about 12%. Due to Austria's strong performance and competence in the field, Austrian life science companies were able to receive an amount of 313.2 million euros as international venture capital investments. Additionally, the growing rate of Austrian life science research infrastructure, such as science parks or incubators, steadily increases the financial migration to Austria by attracting well-known multinational companies (FFG, 2010).

Regarding research and innovation, there are currently 17 universities, 13 universities of applied sciences, and 25 non-university institutes working in the field of life sciences. The ensemble of 55 life science institutions in Austria involves over 24,000 employees, totalled. As of 2020, there is a total number of 77,000 life science related students in Austria, this makes up about 25% of the entire count of 281,791 students in Austria (according to STATISTICS-AUSTRIA, 2021). The Medical University of Graz, Innsbruck Medical University, and the Medical University of Vienna are the most important medical universities regarding life science in Austria, due to their significant impact on research in this field (LISA, 2022). In this thesis, researchers of the Medical University of Graz will be in the focus.

1.3 Life Science Research tools - Current state of the art

As life sciences become more and more popular, so do the (research) tools linked to them. So, as with any raising field of study, with the increasing popularity of the field comes more demand for people in that sector. As well as more research projects being requested by third parties, thus increasing the market share. According to Bhisey (2021), the major reasons for life science research tools success in the past years are the growth of the pharmaceutical industry, the healthcare industry as well as the research, and development sector in relation to life sciences. Additionally to these factors, the still ongoing COVID-19 pandemic is also said to be one of the biggest driving factors when observing the rapid success in the field of life sciences. This might be caused by the threat of replication from not only the corona virus itself, but also future viruses capable of starting another pandemic. Thus, strengthening this field of research was perceived as appealing by the entire world, which may ultimately have contributed to the raising success of life science research.

Looking at the "Life Science Tools Market Size & Share Report", by GrandViewResearch (2021), supports the claim that COVID-19 was a major factor in the rise of life science research tools. The two most influencing factors were the development of COVID-19 testing tool-kits and the pressing desire to create a vaccine. Besides those factors, the life science research tool market is assumed to increase at a constant annual growth rate of around 12% and estimated to more than double its size from USD 105,5 billion (2021) to USD 227,3 billion (2028).

It is worth noting that the market of life science research tools involves a wide variety of utensils such as instruments, consumables, reagents, software and other services. Despite considering the entire extent of the current life science research tools market, the scope of this thesis lies on software tools for sophisticated data analysis. This is done to not blow the whistle of the project's scope, as it is primarily oriented at the touching point between a life science software tool and a distributed data base management system.

In the context of life sciences many different kinds of tools exist to help researchers conduct their jobs. For the scope of this thesis, three of these software solutions are especially important for its related projects. First, CyVerse Austria (2022) forms a platform for research data management, which researchers can use to store, share and analyze their data. CyVerse Austria is explained in more detail in chapter 2.2. Furthermore, iRODS (2021), the distributed data management system used within the CyVerse Austria project, enables users to securely store and access their data. Chapter 2.3 is elaborating on this system. Finally, the life science research tool that is used at the Medical University of Graz, The Galaxy Project (2021), is used for genome sequence analyses. A detailed description of Galaxy is given in 2.4.

1.4 Challenges for Life Science research

Today's life science research is undergoing rapid changes and, therefore, is becoming more and more dependent on technology. This is because modern rapid technological improvements imply that its applications are affected by it and vice versa (Ison et al., 2019). The swift changes of technology pose a lot of challenges for researchers, universities and technology providers. Therefore, one must determine the tipping point where switching to a new technology is useful taking into account the effort it takes (time, money and human resources). With this in mind, it is important to find the correct balance between these three factors. In terms of time, it is crucial to estimate the time span required to introduce new technologies. Especially, considering how long it will take researchers to adapt to new technologies. For monetary decisions, the cost-benefit ratio for switching to new tools has to be considered as well as whether it is reasonable to upgrade currently used technologies to newer versions. The coordination with human resources then closes the loop. There might be reason why researches do not wish to switch to new technologies. In contrast, researchers may prefer to keep their current tools, but wish for extensions to the current system.

A big concern in life sciences research is data handling. This is because of the large amounts of data due to high-throughput technologies and available data in online repositories. In addition, data is often sensitive - one has to protect individuals' privacy. With this in mind, many tasks related to data in this context become non trivial. For instance, any form of outsourcing computation is dangerous because the data might end up exposed to unauthorized parties. The same thing applies for sharing data with colleagues. Several universities use on-premise cloud solutions (e.g., ownCloud (2022), Nextcloud (2022)), where researchers can create open links to provide access to their data. This can lead to unauthorized access to files when the link is shared with others and becomes even more problematic when sensitive data is stored on the on-premise cloud solution. One solution to this problem would be to use physical copies of data, for example, USB drives. In a scenario like this, no third party may ever get a chance to access the data. Furthermore, this can provide additional data security by providing the ability to use encryption on the physical device. However, this solution is impractical because it requires physical interaction; which may not be a problem if two people work in the same building but becomes difficult if two people work in different countries. In conclusion, with current developments in data management, resorting to physical data storage to exchange files appears to be more of a problem, which needs a solution, rather than a solution itself.

Distributed data management systems provide a sound solution to this problem. At Medical University of Graz, University of Graz and Graz University of Technology, CyVerse Austria (2022) has been deployed. This infrastructure offers a distributed data management layer with the possibility to perform reproducible data analytics using container technologies. CyVerse Austria connects storage and computing resources from the three universities. With this system, it is possible for researchers to securely store and share sensitive data while keeping them on storage resources of their own institution. The underlying open-source technology for distributed storage is iRODS (integrated rule-oriented data system). The setup is designed to comply with security guidelines of Medical University of Graz. Researchers can give access rights for their data to collaborators. However, the data does not leave the institution, collaborators will only see what is available on the respective storage resources.

Other security measures include storage and sharing security. Storage security is achieved through fragmentation of data. Thus, unauthorized access to any single data point only yields incomplete and therefore useless results. The sharing security can be achieved by only giving owners the ability to temporarily share data with other already existing users.

For this thesis' project, the life science research tool Galaxy, which is hosted and used at Medical University of Graz, forms the point of action for the data management integration. Galaxy itself, does not provide decent enough ways to securely share data with a save storage. However, researchers are in need of a solution to share data in a safe-guarded way.

Thus, the main goal of this work is the technical integration of an existing computational research tool from the life sciences The Galaxy Project (2021) with an existing database management system iRODS (embedded within CyVerse Austria) to enable an automated workflow for researchers.

1.5 Problem Formulation

Research tools require data management. This is due to the fact that these tools work with data sets, which have to be imported into the tool in some way or another. Whereas researchers are not responsible for data management per se, they rely on the tools data management capabilities to conduct their analyses and perform data transfer tasks. This need forms the main issue this thesis and its related projects try to solve.

In the context of this thesis researchers at the Medical University of Graz face issues with a research tool named Galaxy. This tool provides methods to upload data and perform analytic tasks on these data sets. The analytics functionalities of Galaxy are focused on genetics or protein investigations with single tools, or entire analytics pipelines. However, Galaxy lacks the option to share data in a secure way, especially with collaborators outside of the tool's user base. Out of this need, the requirement to enhance data transfer tasks for sharing emerged. This thesis and its related projects' aim to tackle this precise problem setting.

1.6 Motivation

The motivation for this project is to ensure easy data integration from iRODS into Galaxy to provide an easy workflow for researchers. When working with Galaxy, researchers use provided tools to analyze their data sets. This includes uploading their data sets as well as downloading the results to visualize them with separate tools. For this, researchers often need to share data sets or result files with collaborators (e.g., to share an interesting finding or request a second opinion). However, this sometimes involves dealing with sensitive data, which cannot be shared through common ways. To avoid conflicting with security regulations, researchers often resort to suboptimal solutions. With the help of the interface module, which was developed along side this thesis, researchers are able to share their data sets and result files in a secure way. By using the interface researchers can upload their findings to CyVerse Austria to share it with collaborators. Additionally, researchers can also upload their data sets, or shared data from collaborators, directly into Galaxy.

Consequently, the main goal behind this project is to find a fitting solution to the connection of the life science research tool *TheGalaxyProject*[1] and the distributed data management system *iRODS*[2]. The need for this project was initialized by the Medical University of Graz. They requested the creation of an interface between these two systems, because they were in need of an interoperable system, which avoids or reduces the need of insecure data transfer tasks for researchers. The interface's main job is to load research data sets from a data management system directly into a life science research tool (and vice versa). This includes data and user management being handled securely in the background.

[1]An open-source software framework that aims to help researchers with no computer science expertise develop and maintain a system that allows them to perform computational web-based analyses. (The Galaxy Project, 2021)

[2]An open source data management software that implements data virtualization, enables data discovery, automates data workflows and enables secure collaboration (iRODS (2021)).

1.7 Requirements

For any project related to software engineering, requirements have to be defined first. These requirements are a mutual agreement between the client and the developer and vehemently boost the success rate of a project if done correctly. Generally requirements are split into two groups: functional requirements and non-functional (non-behavioral) requirements. (chitrasingla2001, 2020)

Functional requirements cover all required functionalities of the software project. In general, these requirements can be seen directly in the final version of the project and thus act as a list of available functionalities. An example functional requirement could look like this:

User Registration - The user is able to create an account at the registration page.

Whereas **Non-functional requirements** are hidden within the project's environment and cannot be perceived without context. Thus, non-functional requirements are often referred to as non-behavioral requirements. Non-functional requirements are used as a reflection of quality. An example for a non-functional requirement could look like this:

Scaleability - The system capable of being scaled up- or downwards.

The requirements mentioned below are abbreviated and are discussed in more detail in section 4.1.

1.7.1 Functional Requirements

R1 *User Authentication* - Users are able to authenticate their accounts when accessing the data management system.

R2 *Data Upload* - Users are able to upload (import) data from the research tool into the data management system.

R3 *Data Download* - Users can download (export) data from the data management system into the research tool.

1.7.2 Non-Functional Requirements

R4 *Security* - The used data is securely stored and linked to an authorized user.
R5 *Availability* - The interface and the connected data management system are always accessible.
R6 *Scalability* - Multiple files can be send at once and the interface can be used by N people simultaneously.
R7 *Reliability* - Data transferred via the interface is always sound and complete.
R8 *Reusability* - The interface is portable into other research tools.

1.8 Structure of Thesis

The introduction of the thesis focuses on the central aspects of this paper: life sciences and their respective research tools. It starts with the introduction of life sciences as a (scientific) discipline, and gives an overview about the history and novel challenges life science researchers have to face. The chapter already introduces three regional tools which are objects of study in this thesis (CyVerse Austria, iRODS and Galaxy). Additionally, we provide a short overview about the thesis' topic in general, including the motivation of this work, the problem formulation that led to this scientific paper, as well as giving an outline about the requirements that influenced the thesis' project and development work.

The subsequent chapter, Technological Basis, gives in-depth insight into distributed data management and further elaborates in detail on the technological aspects of CyVerse Austria, iRODS and Galaxy.

The third chapter, Interface Tools, showcases the projects results by describing the interface tools, created alongside this thesis.

The Design and Implementation chapter revolves around the design decisions that aim to fulfill the application's requirements. First, the technological background and the project's setting is explained, by stating its purpose

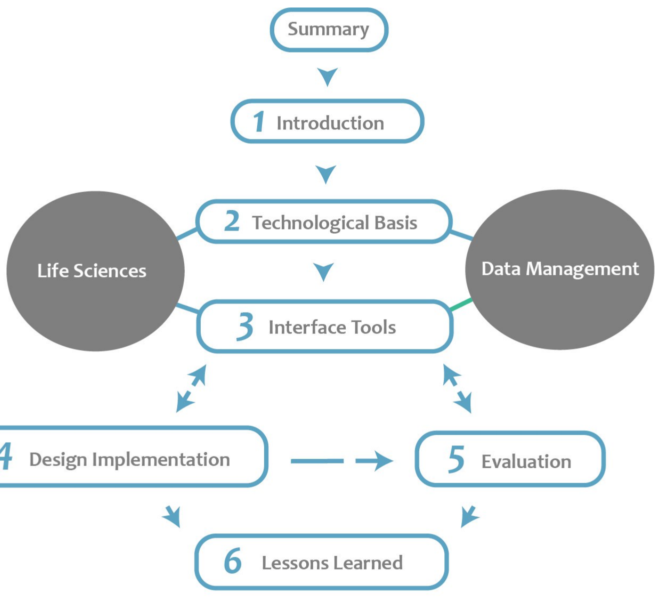


Figure 1.2: Structure of this thesis, illustrating the individual chapters and their coherences

and creation. Then, the interface implementation is elaborated via the course of development.

In the next chapter, Evaluation, the user aspect connected to this project is highlighted. First, the available methods of evaluation are elaborated. Second, user feedback and its rationale is mentioned. Third, evaluation results are presented, by including gathered user feedback.

The last chapter, Lessons Learned, serves as a way to wrap up the thesis, by pointing out both the things that went well and the encountered pitfalls, as well as anything noteworthy during our literature survey. We also provide a critical reflection of the entire project and a comparison between the previously set goals and the actually received results. Additionally, an outlook into the future and suggestions for future work is given.

Figure 1.2 showcases the structure of the thesis. Besides illustrating the flow of the chapters, it also explains how the individual chapters correlate.

2 Technological Basis

"It is not the beauty of a building
you should look at; its the
construction of the foundation
that will stand the test of time."

David Allan Coe

This chapter is divided into four parts. The first one lays the scientific foundation for distributed data management systems. This chapters gives a brief historical excursion followed by an explanation of the current state of the art as well as the recent tools being used. One particular solution of a distributed data management system is mentioned especially. Then, the next three parts emphasize important tools at Universities in Graz, CyVerse Austria, iRODS and Galaxy, because of their importance for the next chapter, the implementation of the project. Finally, a one-page summary concludes this chapter.

2.1 Distributed Data Management Systems

With the rising success of the internet, data management has become a significant factor in today's environment. While also affecting people's private lives, it is most crucial on a corporate level. Thus, in modern work life dealing with data management is inevitable, because in one way or another, data is omnipresent. Whereas data itself is broad and general, the context gives data meaning and importance.

For the scope of this project, researchers and related research data is essential. According to The University of Sheffield (2022), research data is used to support research conclusions and can exist in various forms. Simply put, research data describes files and data sets that researchers use to conduct analyses, draw conclusions and propose results. Research data management (RDM) is often used in the context of research data to describe the tasks and the environment which are necessary to enable working with research data. Thus, research data management starts at generating data, covers data sharing and ends by providing long-term archiving methods to enable viable referencing for scientific work.

The research data management for this thesis' projects is achieved by connecting Galaxy to the distributed data management system iRODS, which is embedded in a research data management platform called CyVerse Austria. The following chapters elaborate on the concept of distributed data management systems and highlight their benefits in the context of this thesis.

2.1.1 Rationale of a Distributed Data Management System

To help understanding the concept of a distributed data management system we break it down to the three elements it consists of: Distributing, Data Management, and System. This way, it is possible to look at the individual topics in detail, to then combine the knowledge and understanding to grasp the underlying concept. Therefore the three elements are explained individually.

Distributing

Distributing in this context means decentralisation. This involves the storage of data, which is usually achieved by a physical storage on a server within the network. When using a centralized approach, one single server acts as the storage device for the entire set of data. Whereas, for a decentralized approach, the data is distributed on multiple machines, also called nodes. Both approaches posses advantages, however, in terms of security and availability, the distributed approach outshines the centralized approach. While centralized data storage is generally cheaper and easier to maintain, it is also very prone to system failure. Once an error occurs at the main node, which in this case is the only node, the entire system stops working. This means that in a scenario like just described none of the data is available at all, thus, nothing is available. However, a decentralised approach allows the possibility to divide the data into smaller pieces and distribute them over multiple nodes. This way it is highly unlikely that all nodes experience failure at the exact same time, which makes this approach much more resilient to attacks. With enough storage capacity it is also possible to establish an infallible system, by keeping additional copies of one node on other node(s). This means, that even if one node would fail, other nodes could still provide the files from that unavailable node. Additionally, distribution also enables the possibility of data fragmentation, which means splitting individual files into pieces and storing them separately. Using data fragmentation increases security aspects of the system, because a malicious third party, which only has access to a single node would just find pieces of files, similar to a few pieces of multiple jigsaw puzzles. This effect can be enhanced even more by ciphering the individual files before splitting them up.

Data Management

According to Stedman (2019), **data management** is the process of ingesting, storing, organizing, and maintaining the data created and collected by an organization. Simply put, it serves three major tasks: to take data as input and to process it, to store data with a long-term plan in mind, and to provide the contained data as an output. Additionally, data management is often, but not always, combined with user management, which takes care of access rights for the data. In a simplistic version of data management, particular users are able to upload data and access it whenever needed. In this case, the author of the uploaded data is the sole person with access rights on the files. This version can be extended by allowing users to share selected files with another user, thus, making it possible for multiple users to access the same data. As a last extension, the system can be enhanced by allowing shared storage spaces, similar to a directory, to which a team of multiple users has access. This makes it possible to collaborate in real time by manipulating files synchronously. It is also worth noting, that data management knows two types of access rights: Read access and Write access. Read access allows particular users to view specified data whereas, write access grants dedicated users the ability to manipulate the files. Write access thus can be seen as an extension of reading permissions, because a user must also be able to view the data in order to manipulate it. Finally, access rights in data management are generally bound to data rather than users. Meaning, that particular files can be accessed by specified users, other than users themselves having global access rights. Besides that, data management also takes care of topics such as: Data security - by forming the boundary of what is inside and outside of the network as well as being responsible for backups, storage space - by defining physical hardware space as data storage space, and data availability - by providing access to the data via the network of the system. Data management in general, describes the purpose of handling data. However, to apply the rules for a physical solution an embedment - a system - is required.

System

A **System** is a set of individual pieces, which are put together to attain a mutual purpose. In other words, a system consists of multiple components working together to achieve a task. A system is simplest explained by looking at the human body. The human body consists of many different elements: organs, fluids, bones, muscles, and nerves. Here, the nerves play a special role, because they form a subsystem within the body itself, also known as the nervous system. Thus, when examining the individual pieces, it becomes notable that everything has its own purpose. Yet, when put together they achieve a greater good, so to say the big system of the human body. It is important to understand that, while every component has its unique task, they are unable to exist on their own. For example, the bones are reliant on the muscles to provide tension, only then providing the system with stability. The muscles depend on the bloodstream and its contents to function and provide the system with movement. The bloodstream acts as a network between the organs, similar to the nervous system, which controls their actions. Finally, the skin acts as a border for the system, determining what is inside and outside of it. Now, transposing this analogy to the world of data management, it becomes clear that it also consists of multiple components, which are working together. Thereby, they form a system, which achieves the purpose of data management. In this case, the system also limits the borderline of data management, by determining what is within the system. To start, a system like this consists of a primary node, with the purpose of running data management services. Besides holding the information on what is stored within the system, this node also orchestrates other nodes, if there are additional nodes present within the system. In addition to that, a network is required to connect the system's components to each other. At this point, the system may be extended by auxiliary services, e.g., a user management system, to supplement the existing system. Finally, the system is connected to the outside - also via the network - to users. On the one hand, the users are responsible to provide data to store within the system. On the other hand, the users also request data when accessing the system. Finally, a backup system can be placed outside the system to provide additional security.

With the in-depth knowledge about the three individual elements of a distributed data management system it is now possible to explain the connected concept of such a system. Figure 2.1 illustrates an example for a distributed data management system. Following the principles of distributing, the framework consists of multiple nodes all connected to each other. The special node amongst the others serves the purpose of being the service node which, besides providing the service for a data management system, also contains meta information about the data stored within the system. The nodes forming the data management service are enclosed by a circle, showcasing the system's boundaries. Outside the system, there are multiple users interacting with the data management framework. In addition to that, there are two extra points connected to the system; one of which being the external backup service and the other one being the external user management system. Each of which are connected to the main node via the network. When communicating with the system, the users primarily communicate with the main node of the data management system. The main node, which runs the data management service then handles the data management in the background.

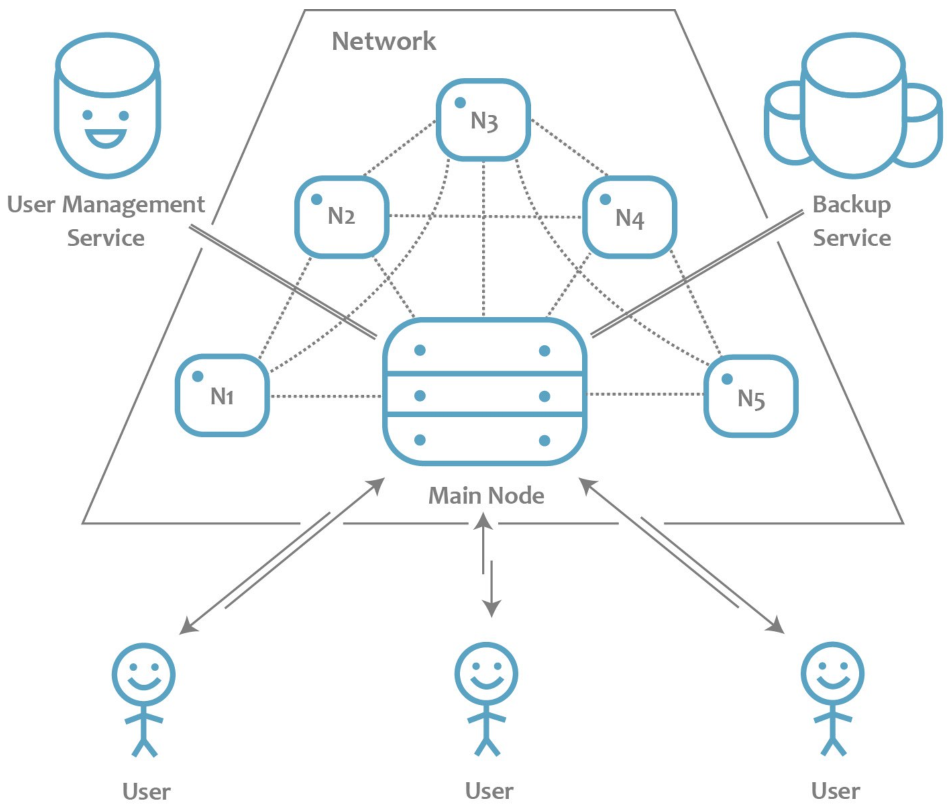


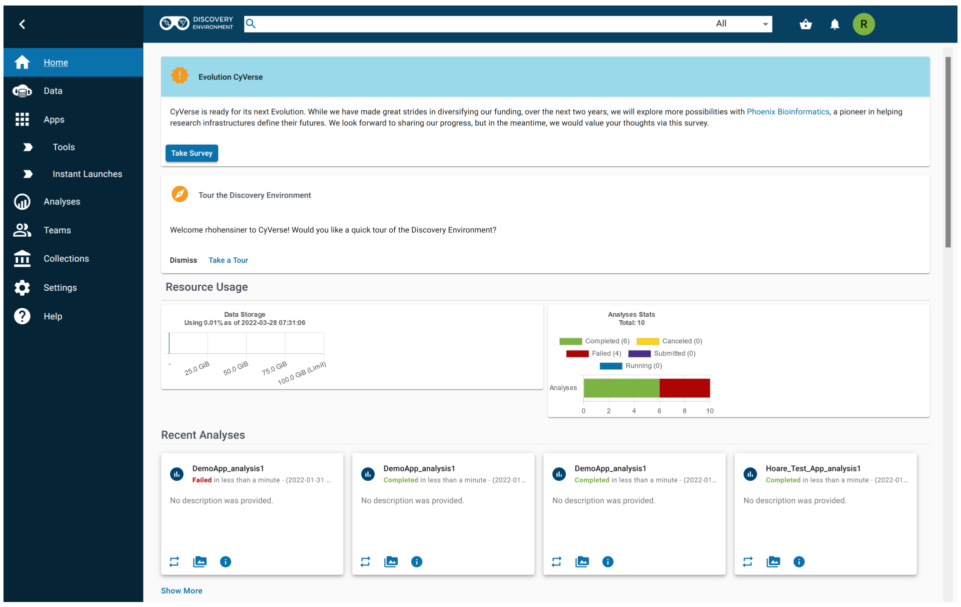
Figure 2.1: Illustration of a distributed data base management system containing nodes, users, a network and external services.

To illustrate this as an example: a user might want to upload a data file to the system. The user provides the file and additionally his user credentials to the system. When arriving at the system, the main node firstly forwards the user credentials to the external user management service, which upon successful authentication returns approval to the main node. The main node then requests one of the nodes within the system to store the data and, after successful completion, stores the data entry within the system. Finally, it acknowledges the user about successful completion.

2.2 CyVerse Austria

CyVerse is a platform designed for researchers and to help them with their data management needs. It is of importance for the context of this thesis because it involves a distributed data management system called iRODS. With this, it is elaborating the distributed data management system aspect of this technological basis.

CyVerse was originally born in collaboration with multiple universities in the United States of America (CyVerse National Science Foundation, 2022). Its goal is to establish a framework for researchers that provides a better way for them to handle their data. Additionally, it was a goal to create a system where researchers could collaborate effectively and not only share their data sets and results, but also their actual experimental procedures and methods, by the analysis tools they created. The former is achieved by the use of a distributed data management system called iRODS which manages all the data entered to the system in the background. By exploiting the distributing aspects of a DDMS, it is possible to connect various storage nodes into the system and also extend the system further. This enables multiple universities to provide a part of their hardware capabilities to CyVerse. In addition to that, CyVerse also includes a user management service, enabling verification via universities' SSO realms. To enable users to share their tools and applications, CyVerse uses the Docker (2022) framework and Docker containers. With the use of Docker containers researchers are able to wrap their research analyses into independent, often pseudo platform-independent, virtual machines, that can be reused by other researchers. This helps speeding up the



Evolution CyVerse

CyVerse is ready for its next Evolution. While we have made great strides in diversifying our funding, over the next two years, we will explore more possibilities with Phoenix Bioinformatics, a pioneer in helping research infrastructures define their futures. We look forward to sharing our progress, but in the meantime, we would value your thoughts via this survey.

Tour the Discovery Environment

Welcome rhohensiner to CyVerse! Would you like a quick tour of the Discovery Environment?

Resource Usage

Recent Analyses

DemoApp_analysis1
Failed in less than a minute - (2022-01-31 ...
No description was provided.

DemoApp_analysis1
Completed in less than a minute - (2022-01...
No description was provided.

DemoApp_analysis1
Completed in less than a minute - (2022-01...
No description was provided.

Hoare_Test_App_analysis1
Completed in less than a minute - (2022-01...
No description was provided.

Figure 2.2: Illustration of CyVerse Discovery Environment. [Screenshot taken by author from the latest build of CyVerse AT on 28.03.2022]

process of development, because by sharing their tools, researchers do not need to create their own versions of a particular analysis, but instead can use the already existing application. Additionally, it is possible to chain multiple tools together to form a sequenced application. This makes it possible to build large automated systems, which can be started with a single mouse click. One of CyVerse's biggest advantages is the joined forces for hardware capacities, which can be allocated to researchers, depending on their current workloads (CyVerse Austria, 2022).

Figure 2.2 shows CyVerse's Discovery Environment, which is used as the main interface between the user and the data management system. From this point it is possible to upload data, access data available to the user, share data with collaborators as well as import data into CyVerse from various sources. Furthermore, it is possible for researchers to create their individual tools from the discovery environment. To create tools for CyVerse, researchers need a valid Docker container, which was previously uploaded to an official *Docker Hub*[1]. Then, via CyVerse's framework, the creator of a tool is able to set performance restrictions, e.g., hardware required, which will then be taking into consideration at run time for the tool. As a next step, researchers can then build an application from their previously created tools, or already existing tools from other colleagues. An application acts as a framework surrounding one or multiple tools, sequencing them together to achieve a tool-chain. This framework contains information such as input variables, specifications for the execution environment, or output specifications. Built applications can then be started by users after issuing the execution requirements. The application will then be run on whichever hardware is currently available and fits the performance needs issued by the contained tools. After completion, results from the application can be examined either directly via CyVerse or downloaded and interpreted by another tool.

2.3 iRODS

iRODS (2021) is a widely known distributed data management solution. It defines a system that focuses on covering 4 main tasks: data virtualisation, data discovery, workflow automation, and secure collaboration. iRODS is especially important for this thesis, because it forms the second part of the connection for the the interface between a life science research tool and a distributed data management system (iRODS Consortium, 2016).

To achieve its 4 main goals, iRODS relies on a system called iRODS Zone. An iRODS Zone describes a cluster of servers, connected via network, which run the iRODS server software. Within a iRODS Zone, one node takes the role of the catalogue provider, which holds the meta information about the data stored across the cluster. In addition to that, catalogue consumer nodes exist in a varying number, which can range from one to multiple hundreds of individual server nodes. With this feature, the iRODS server software enables a great opportunity for upscaling an iRODS Zone - personalizable to fulfill specific needs. iRODS Servers in the zone accept requests from iRODS

[1]The world's largest library framework for docker containers that provides possibilities to create, manage, deliver and share docker applications. (Docker Inc., 2022)

clients, which are embodied by users who issue tasks to the system. Tasks like these could for example consist of a data upload request, or a user data pull request. iRODS is usually coupled with an external user management system. However, it possesses the capability of an internal user management registry as well. iRODS' internal user management involves the option of user administration services to create and modify user access rights.

iRODS data management stores files as data objects, which are organized in so called collections. The collections in this context work similar to directories known from common file systems. This way, the iRODS server service attains data virtualisation, which is achieved by two requirements: the collections are uncoupled from a physical storage path (data objects within a collection may be stored on multiple different physical locations) and data objects refer to one to many replicas (with replicas being exact copies of a file, stored in different locations). In essence, iRODS achieves data virtualisation, by implementing a virtual file system above the iRODS zone, which contains the physical storage devices.

Data discovery in iRODS is provided by the catalogue provider nodes within each iRODS zone. This metadata catalogue contains information about the data object, the collections, the users, and zone and storage resources. Besides traditional metadata information, iRODS additionally entails rich metadata. This extended metadata contains information such as author names, keywords, and content types. By the use of meta data, iRODS is amplifying its data discovery capability.

The third goal, workflow automation is conjunct with each iRODS servers Rule engine, which works as an event-triggered background service. The events to trigger the rule engine are called policy enforcement points, or PEPs. They define the points at which the system is making changes to access rights of files. For example, a rule could be made that before a data object is deleted, the meta information of said file is sent to an external data point, which keeps track of deleted files within the system. With newer versions of iRODS, many programming languages, e.g., Python or C++, are supported by using the Rule engine.

The last objective of iRODS is secure collaboration. This goal is of paramount importance because data management systems are usually used by a large number or users. With the rising number of users, the risk of a data breach increases accordingly. To ensure secure collaboration with iRODS, the system uses three additional features: Tickets, Permissions and Federation. Tickets allow controlled access for public sharing to data objects and collections. The iRODS Permissions resemble UNIX file system permissions, which provide additional security. Finally, the Federation feature allows iRODS systems to communicate across individual iRODS zones.

In addition to that, iRODS also comprises a powerful API, which can be used to manage iRODS servers by administrators as well as basic data manipulation performed by users.

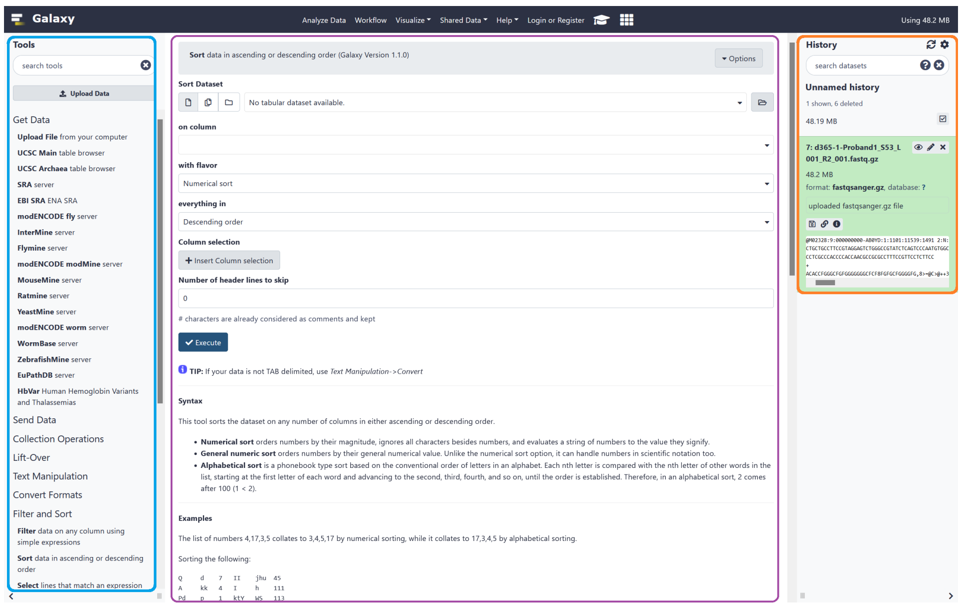
2.4 Galaxy

The Galaxy Project (2021) is an open source life science research tool solution that aims to be an all-in-one solution for researchers. It is especially important for this thesis because it represents one of the two key points involved for the interface between itself and the distributed data management system. Thus, it will be emphasized and explained in greater detail, to help the understanding in later chapters of this thesis' project.

As already mentioned, the Galaxy project embodies an all-in-one solution for FAIR data analysis. It is actively maintained and constantly expanded to increase its reach and adapt to the rapidly changing developments in the field. It is widely known in the research community, not only because it is open source, but also because of its possibility to self-administrate an entire instance. There are two different ways of installation: installing Galaxy directly onto a Linux server or installing it via Docker onto any Docker-supporting platform. With this in mind, Galaxy offers the possibility to embed itself into many different settings, thus, making it very flexible. The Galaxy project itself consists of a core which is primarily written in Python, a web client (frontend) as well as a data base. Additionally, Galaxy allows management and administration instances via its built-in user management. User credentials and permissions are stored within Galaxy's database where it can be backed up directly. Galaxy also includes lightweight user data management options called histories, which are bound to user accounts. Besides Galaxy's basic features, local data up- and downloading, its further features come from external software, imported as tools. Those tools can be installed by system administrators via so called tool sheds. Tool sheds are external tool stores, similar to app stores, which provide access to prior developed tools from various sources. This allows researches, in case a special functionality is not already present, to develop their own procedures, pack them into the framework of a tool and upload it to a tool shed to share it with other researchers. This way, Galaxy's functionality constantly increases in size and potency by researchers sharing their work. In addition to that, third party developers, e.g., software engineers, are also working on tools to create interfaces between Galaxy and other software solutions in order to make Galaxy accessible to other life science research tools. In this sense, the interface between Galaxy and the DDMS iRODS was also developed in order to enable Galaxy to connect to iRODS instances. In short, the purpose of said tool is to provide a way of secure data transfer between Galaxy's built-in histories and iRODS' secure distributed data management system. Tasks like this become increasingly important when dealing with sensitive data, e.g., protected health information, which in some cases also might be used for research purposes. Despite the data being anonymized, the risk of it ending up in the wrong hands while sharing the data with colleagues is still outside of acceptable thresholds. Thus, a solution for secure data exchange was set as an important goal.

Figure 2.3 shows a screenshot of Galaxy's user interface. The available tools are placed on the left side of the interface, while the user's history (and tasks) are placed on the right side. The space in the middle is used for the selected tool and research. A user can select a tool from the list on the left, input chosen data sets and start the tools research processing. Depending on the task, this analysis could take between minutes or hours to finish. The (task) history on the right provides information about the currently ongoing tasks, additionally notifying the user once a particular task has been finished. After processing terminates, the results are made available to the user via the history on the right. From this point out, users are able to inspect the results directly within the history task, export it to analyze it further locally or pipeline it to another tool available on the instance.

A potential workflow could look like this: a user uploads data and selects a tool to manipulate the data in a specific way, e.g., by removing noise. The manipulated data then gets "pipelined" to another tool resulting in the



Sort data in ascending or descending order (Galaxy Version 1.1.0)

Sort Dataset
No tabular dataset available.

on column

with flavor
Numerical sort

everything in
Descending order

Column selection

Number of header lines to skip
0

characters are already considered as comments and kept

TIP: If your data is not TAB delimited, use *Text Manipulation->Convert*

Syntax

This tool sorts the dataset on any number of columns in either ascending or descending order.

- **Numerical sort** orders numbers by their magnitude, ignores all characters besides numbers, and evaluates a string of numbers to the value they signify.
- **General numeric sort** orders numbers by their general numerical value. Unlike the numerical sort option, it can handle numbers in scientific notation too.
- **Alphabetical sort** is a phonebook type sort based on the conventional order of letters in an alphabet. Each nth letter is compared with the nth letter of other words in the list, starting at the first letter of each word and advancing to the second, third, fourth, and so on, until the order is established. Therefore, in an alphabetical sort, 2 comes after 100 (1 < 2).

Examples

The list of numbers 4,17,3,5 collates to 3,4,5,17 by numerical sorting, while it collates to 17,3,4,5 by alphabetical sorting.

Sorting the following:

```
Q     d    7    II    jhu   45
A     kk   4    I     h     111
Pd    p    1    ktY   WS    113
```

Unnamed history
1 shown, 6 deleted
48.19 MB

7: d365-1-Proband1_S53_L001_R2_001.fastq.gz
48.2 MB
format: **fastqsanger.gz**, database: **?**
uploaded fastqsanger.gz file

```
@M02328:9:000000000-AB0YD:1:1101:11539:1491 2:N:
CTGCTGCCTTCCGTAGGAGTCTGGGCCGTATCTCAGTCCCAATGTGGC
CCTCGCCCACCCCACCAACGCCGCGCCTTTCCGTTCCTCTTCC
+
ACACCFGGGCFGFGGGGGGGCFCF8FGFGCFGGGGFG,8>=@C>@++3
```

Figure 2.3: Illustration of the Galaxy user interface, showcasing tools, tool interfaces and user history. [Screenshot taken by author from Galaxy version 21.01 on 28.03.2022]

desired analysis of the initial data. The user then might want to export the results to visually inspect the data by a third party program.

2.5 Summary of Chapter 2

This chapter explains the technological basis required to understand the purpose of the interface on which this work is based upon. Generally, the explanation is split into four parts, reflecting the two worlds that meet through this project.

First, distributed data management, its purpose and goal, is described in detail. Then, the three technological solutions available at Medical University of Graz are introduced and described in detail.

This chapter explains what the project is trying to achieve, as well as presenting the two systems, which the interface module connects.

3 The Interface Tools - Connecting Galaxy and iRODS

"I pass with relief from the tossing sea of Cause and Theory to the firm ground of Result and Fact."

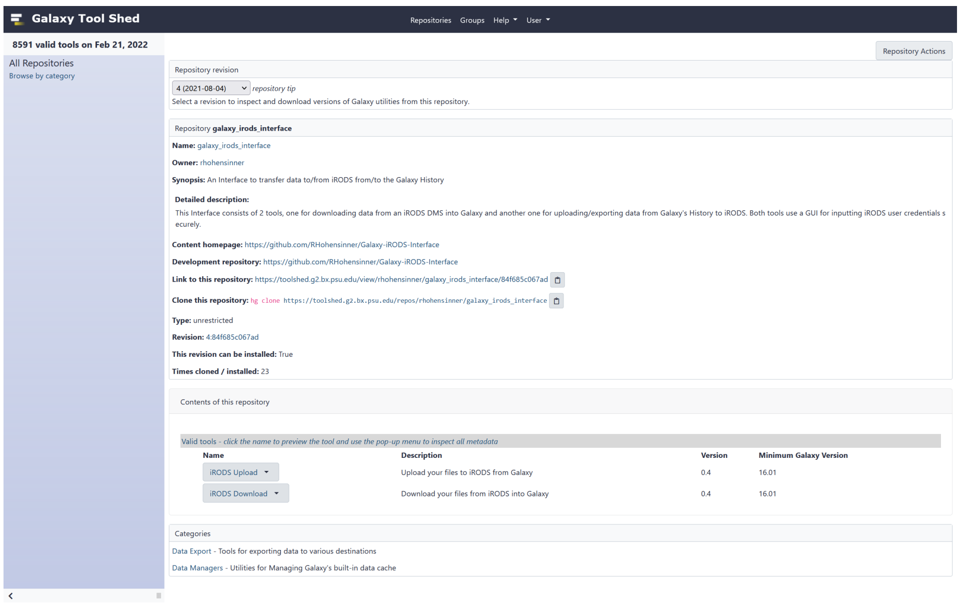
Winston Churchill

This chapter will elucidate the interface tool, which was created throughout the course of this project. This interface is a tool developed for the life science research tool Galaxy that aims to connect it to the distributed data management system iRODS. This interface was developed during the course of 2021, finally resulting in the publication of the finished tool in Galaxy's official *tool shed*[1].

The final outcome of this thesis' projects, the interface module and tools, are currently published and accessible via the dedicated tool shed entry as well as a separate public GitHub repository. The official tool shed entry can be found here: Interface Tool Shed Entry (2021). The GitHub repository is available here: Interface GitHub Repository (2021).

Figure 3.1 shows the current version of the tool shed entry to which the interface module was published.

[1]Galaxy's tool shed works similar to an app store, providing an option to install additional research tools to Galaxy instances. (Galaxy Community, 2022)



8591 valid tools on Feb 21, 2022

All Repositories
Browse by category

Repository revision

4 (2021-08-04) *repository tip*
Select a revision to inspect and download versions of Galaxy utilities from this repository.

Repository **galaxy_irods_interface**

Name: galaxy_irods_interface

Owner: rhohensinner

Synopsis: An Interface to transfer data to/from iRODS from/to the Galaxy History

Detailed description:
This Interface consists of 2 tools, one for downloading data from an iRODS DMS into Galaxy and another one for uploading/exporting data from Galaxy's History to iRODS. Both tools use a GUI for inputting iRODS user credentials s
ecurely.

Content homepage: https://github.com/RHohensinner/Galaxy-iRODS-Interface

Development repository: https://github.com/RHohensinner/Galaxy-iRODS-Interface

Link to this repository: https://toolshed.g2.bx.psu.edu/view/rhohensinner/galaxy_irods_interface/84f685c067ad

Clone this repository: `hg clone https://toolshed.g2.bx.psu.edu/repos/rhohensinner/galaxy_irods_interface`

Type: unrestricted

Revision: 4:84f685c067ad

This revision can be installed: True

Times cloned / installed: 23

Contents of this repository

Valid tools - *click the name to preview the tool and use the pop-up menu to inspect all metadata*

Name	Description	Version	Minimum Galaxy Version
iRODS Upload	Upload your files to iRODS from Galaxy	0.4	16.01
iRODS Download	Download your files from iRODS into Galaxy	0.4	16.01

Categories

Data Export - Tools for exporting data to various destinations

Data Managers - Utilities for Managing Galaxy's built-in data cache

Figure 3.1: Illustration of the Galaxy tool shed entry, showcasing the current version of the interface tools. [Screenshot taken by author from Galaxy Tool Shed on 31.03.2022]

3.1 Resulting Functionality

The interface primarily serves the purpose of allowing users to transfer data between their Galaxy user history and their iRODS Zone storage space. Thus, the iRODS zone ends up connected to CyVerse for this interface. This makes it possible to use the extra features for the data - that are inherited by CyVerse.

This interface consists of two independent tools: it is designed to export data from Galaxy to any desired iRODS zone and it is designed to import data from any iRODS zone into the users Galaxy history. Together, the two functionalities make up the data transfer feature of the interface. Thus, a file created within Galaxy can be exported to iRODS, manipulated via CyVerse, and finally imported back into the Galaxy history.

For example: Researcher "R" is working on an analysis within the Galaxy instance. R has to first upload data to their Galaxy history. After importing the data, R runs an analysis tool on the data set. After termination, R wants to share the results with a collaborator. To achieve this, R decides to use the iRODS interface tools. R exports the results from their history to CyVerse's iRODS zone. Then R logs into the CyVerse service to access the data through the discovery environment. R is able to share the data to a collaborator through CyVerse. After some time R receives a notification that the collaborator made some changes to the shared result files. R can then import the data back into his Galaxy history to run some additional analysis tools on the modified data file.

Figure 3.2 illustrates the environment surrounding the interface tools. In the middle of the image, the interface tools are presented. On the left, Galaxy is illustrated, comprising the interface tools. On the right side, iRODS and CyVerse are visible. In addition to that, researchers, data, and potential workflow of them are drawn. This illustration gives an overview about how the individual elements surrounding the interface are connected to each other.

Figure 3.3 and 3.4 illustrate screenshots of the interface tools. The interface and its tools are installed in Galaxy, and are accessed from there. Figure 3.3 shows the download tool, which allows downloading (exporting) files

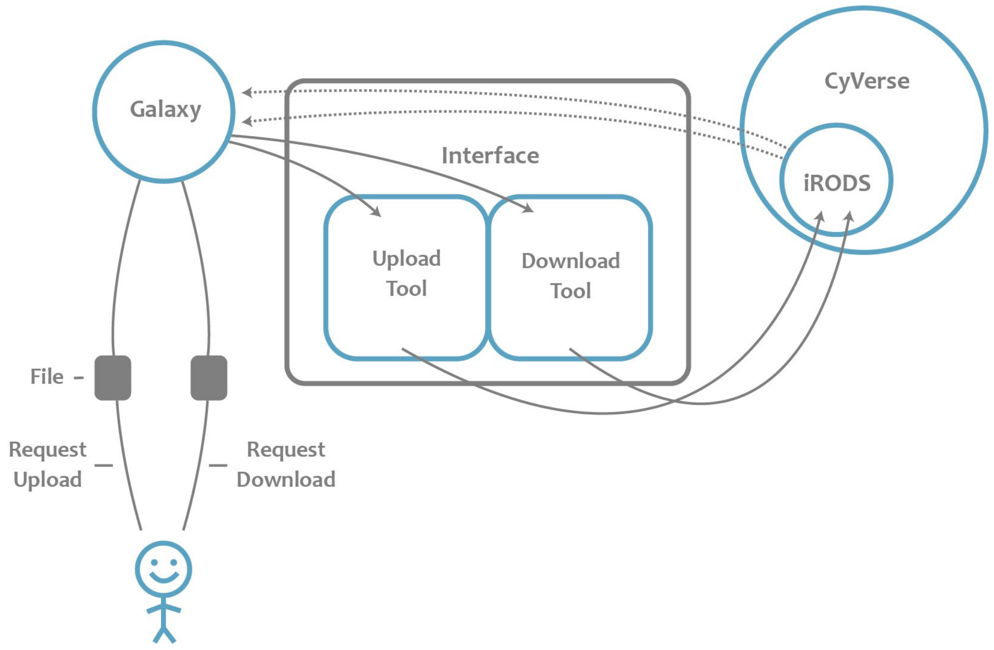
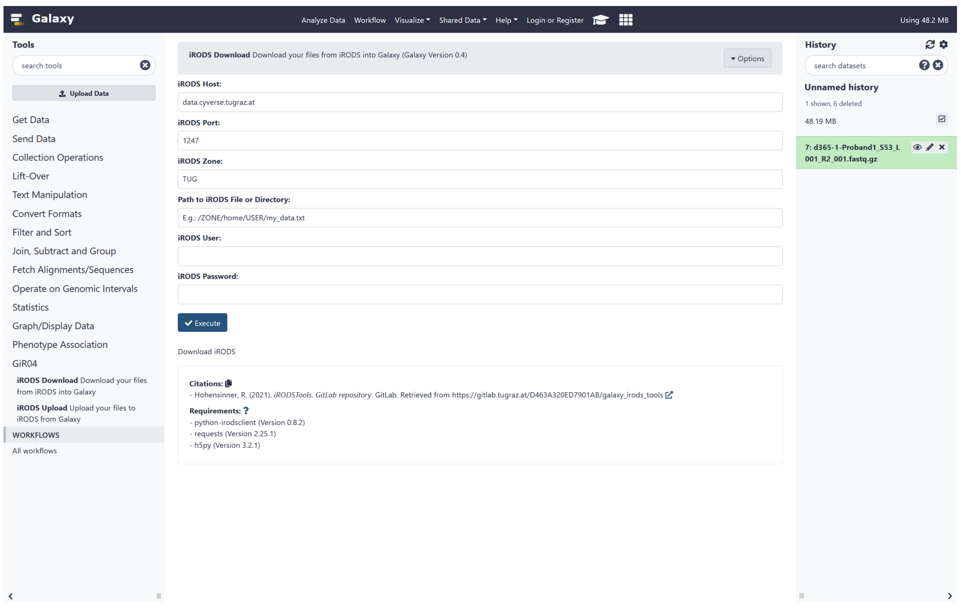


Figure 3.2: Illustration of the interface module, consisting of two separate tools, and the environment in which the interface is embedded into.



iRODS Download Download your files from iRODS into Galaxy

iRODS Upload Upload your files to iRODS from Galaxy

iRODS Download Download your files from iRODS into Galaxy (Galaxy Version 0.4)

iRODS Host:
data.cyverse.tugraz.at

iRODS Port:
1247

iRODS Zone:
TUG

Path to iRODS File or Directory:
E.g.: /ZONE/home/USER/my_data.txt

iRODS User:

iRODS Password:

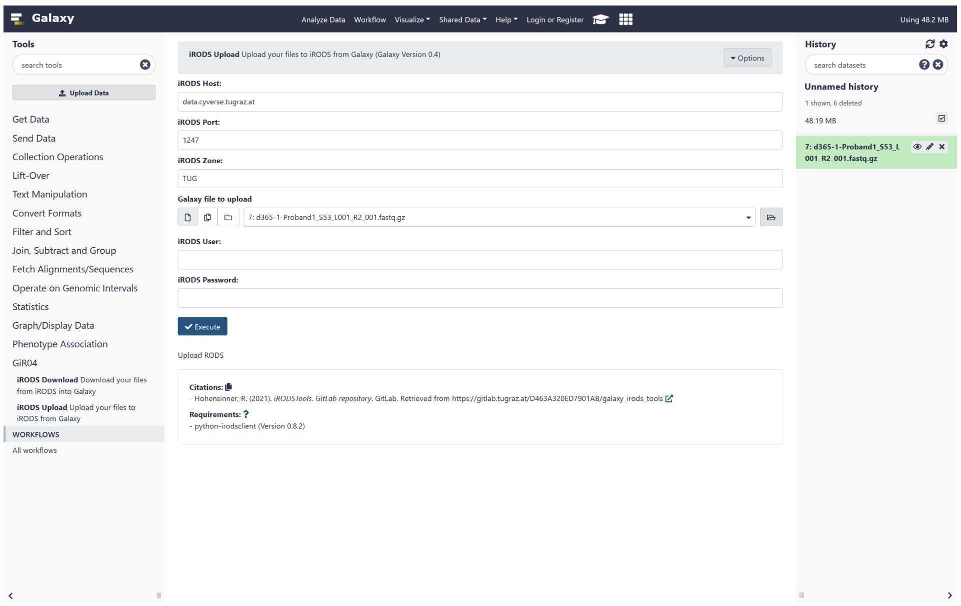
Download iRODS

Citations:
- Hohensinner, R. (2021). *iRODSTools. GitLab repository*. GitLab. Retrieved from https://gitlab.tugraz.at/D463A320ED7901AB/galaxy_irods_tools

Requirements:
- python-irodsclient (Version 0.8.2)
- requests (Version 2.25.1)
- h5py (Version 3.2.1)

7: d365-1-Proband1_S53_L001_R2_001.fastq.gz

Figure 3.3: Showcase of the download tool that was created throughout the course of this thesis' project. [Screenshot taken by author from Galaxy version 21.01 on 28.03.2022]



iRODS Download Download your files from iRODS into Galaxy

iRODS Upload Upload your files to iRODS from Galaxy

iRODS Upload Upload your files to iRODS from Galaxy (Galaxy Version 0.4)

iRODS Host: data.cyverse.tugraz.at

iRODS Port: 1247

iRODS Zone: TUG

Galaxy file to upload 7: d365-1-Proband1_S53_L001_R2_001.fastq.gz

iRODS User:

iRODS Password:

Upload RODS

Citations:
- Hohensinner, R. (2021). *iRODSTools. GitLab repository*. GitLab. Retrieved from https://gitlab.tugraz.at/D463A320ED7901AB/galaxy_irods_tools

Requirements: ?
- python-irodsclient (Version 0.8.2)

Unnamed history
1 shown, 6 deleted
48.19 MB

7: d365-1-Proband1_S53_L001_R2_001.fastq.gz

Figure 3.4: Showcase of the upload tool that was created throughout the course of this thesis' project. [Screenshot taken by author from Galaxy version 21.01 on 28.03.2022]

from an iRODS zone to the Galaxy history. For this the target iRODS zone's information, the path to the data as well as user credentials for iRODS have to be entered. Figure 3.4 shows the second interface tool, which is used to upload (import) data from the Galaxy history to the iRODS zone. First, a particular history entry, which should be uploaded, has to be selected. Then, similar to the download tool, this tool also requires the target iRODS zone information as well as the user's iRODS credentials.

3.2 Summary of Chapter 3

In this chapter the result of the project is explained. Thus, the interface, which consists of two independent tools, is presented here. In addition to giving a short explanation about the interface's context, this chapter also showcases the tools which are the direct results of this project. An illustration is used to demonstrate the interface's intentional use. Screenshots of both tools and a description then conclude this chapter.

4 Design and Implementation

"Life is soup, I am fork."

Pakalu Papito

In this chapter we will explain the design decisions taken during our development process. We will first revisit and elaborate the requirements previously mentioned in the introduction. We do this to show how our design decisions help us fulfill the requirements. Afterwards, the interface implementation and process will be explained in detail. This includes the additional decisions taken during the course of development.

4.1 Requirements

The requirements are split into two main groups - functional and non-functional requirements. During the initial planning phase of the project three functional requirements and five non-functional requirements were agreed upon by all actors. To reiterate, functional requirements describe possible actions or features of a software project, e.g., clicking a button. Whereas non-functional requirements imply passive behaviour of the system, e.g., the software is responsive by not exceeding 2 seconds without feedback. The gathered requirements form the basis for choosing a design concept for the project. In this case, the requirements directly influenced the choice of the used design concept.

4.1.1 Functional Requirements

R1 *User Authentication* - Users are able to authenticate their accounts when accessing the data management system.
R2 *Data Upload* - Users are able to upload (import) data from the research tool into the data management system.
R3 *Data Download* - Users can download (export) data from the data management system into the research tool.

R1 defines the most significant feature of the interface tool. Because data cannot exist solely with a designated user in a data management system, it is always connected to a particular author of the data. In this case, the data has to be considered for two systems. To transfer data from one system to another, user authentication has to be considered. The author of a file in one system must also be the author of the file on the other system, after transfer. For this reason, the interface tool enables users to enter user credentials in the Galaxy tools, that are used to authenticate the user for the target iRODS zone involved in the data exchange request. Furthermore, already existing user information has to be known to the target system as well as the source system. Only then user authentication can be enforced securely for the data exchange procedure.

R2 and **R3** describe the two main functionalities which enable data transfer. Data uploading allows the user to transfer data from system A (Galaxy) to system B (iRODS). Data downloading facilitate file transportation from system B (iRODS) to system A (Galaxy). By providing these functionalities, we make it possible for the user to transfer data as required. Connecting two one-way transportation routes and switching origin and destination form a data exchange cycle. Additionally, it enables users to include several destination nodes by using the Galaxy history as a juncture. E.g., User pulls data from destination X (iRODS zone 1) to the Galaxy history and pushes the files to destination Y (iRODS zone 2).

4.1.2 Non-Functional Requirements

R4 *Security* - The used data is always securely stored and linked to an authorized user.
R5 *Availability* - The interface and the connected data management system are always accessible.
R6 *Scalability* - Multiple files can be sent at once and the interface can be used by N people simultaneously.
R7 *Reliability* - Data transferred via the interface is always sound and complete.
R8 *Reusability* - The interface is portable into other research tools.

R4 - **R8** comprise important passive behaviour of the system. **R4 Security** is of utmost importance when managing data, because it induces a sense of safety to the user. To achieve this, the data always has to be in control of the authorized user solely. Also, when using the interface, only the author of the data has access to the transferred files. With this in mind, the interface tools are designed so that data is exchanged directly via the origin and destination system, with no (insecure) hops in between.

To achieve **R5 Availability**, the interface tools are designed in a stateless fashion. The tools are always ready to be used, regardless of their current state. However, to be able to transfer data from system A to system B, both systems have to be available to work. So, as long as the origin and target systems are up, running and available, the tools are utilizable.

The tools are designed to attain **R6 Scalability** by providing the option to transfer multiple files at once. In case of importing data, this can be achieved by grouping history elements together and sending them in one request. For exporting data, files can be gathered by selecting a collection instead of a single data object. Galaxy tools are being run as tasks with a unique ID, which enables the possibility of multiple tool request calls at the same time. In addition to that, iRODS zone access supports simultaneous access to the servers, which allows multiple users to transfer data with the same iRODS zone in parallel.

R7 Reliability is ensured by using iRODS data transportation framework. This prevents spurious data exchange by saving files only after the transfer process was completed successfully. For unsuccessful transfers, the users will be informed accordingly, preventing misconceptions with regards to the data transfer task. This causes the data exchange to either be sound and complete, or aborted and non interfering. The tools also check files with regards to storage space before execution. So, the data transfer will only be carried out if enough storage space is available on the target system.

R8 Reusability had the biggest influence on the implementation of the interface tools. They are designed with reusability in mind, making it possible to shift the entire software service into another host system, with the only requirement of possible input parameter adaption. Besides that, the tools can also be run entirely on their own, removing the need to be embedded into a Galaxy tool.

4.2 Design Decisions

After evaluating the requirements, design concepts for the project were investigated. This included design principles for software engineering. These were examined and chosen in coordination with the established requirements. According to Thaktur (2022), the designing phase is the stage in software development in which a blueprint for the system is created. This blueprint is created by connecting the requirements to the planned goal of a system. It defines the best fit for the requirements in consideration of the final product, while keeping software development principles in mind. The blueprint and design concepts help software engineers to stick to common principles during the course of development.

For this project four design principles were chosen:

D1 *Reusability*
D2 *Simplicity*
D3 *Usability*
D4 *Modularity*

The desired ability to be reusable formed the most important design decision. This principle influenced the structure and architecture of the interface vehemently as it tasked us to embed the interface into other (research) tools, the decision was made to detach the input parameters as much as possible from the interface. Thus, the interface was designed in a way to be usable in a stand-alone fashion and any source (Galaxy) specific behaviour was uncoupled from the main interface. This makes it possible to integrate the interface service into other projects, by simply piping the input parameters to fit the request needs.

The tool's jobs are simple: move user possessed data from point A to point B, or vice versa. Thus, the paradigm of simplistic design was chosen to keep things as self-explanatory as possible. Even though the project provides the possibility to be extended with many additional features, e.g., transfer flags, data integrity checks, etc., the interface and its code focuses solely and precisely on the task of user authenticated data transfer between two end-points.

Because the task of transferring data is considered an auxiliary task, for which users should not be forced to read pages of documentation, well designed usability is vital for this project. Thus, the UI design for the interface tools is kept as minimalist and self-explaining as possible as well as non-obstructive. This means, that the required user input is kept to its minimum, which involves just three kinds of information: target system definition, file selection, and user credentials.

Last but not least, the concept of modularity has to be mentioned. In the case of this interface service, the feature of data transfer was cut into two. One half of the tool provides the ability to export data from the source system and the other half enables the user to import data from the target system. Thus, sticking to the principles of modularity, the tools are subdivided into two pieces as well as a third shared module that is used to establish the connection with the target system, authenticate the user, and initialize the data transfer request.

4.3 Technologies used

- The Galaxy Project
- Planemo

- Galaxy Tool Shed
- CyVerse Austria
- iRODS data management system
- Python 3
  - python-irods-client (Version 0.8.2.)
  - requests (Version 2.25.1.)
  - h5py (Version 3.2.1.)
- Cheetah Template Engine
- GitHub

The interface was primarily developed in Python, with the additional usage of the Cheetah Template Engine for XML files. Cheetah allows inline Python code execution at file access. The interface was implemented as two tools for Galaxy, which are defined via XML files. The tools were created and published via the help of Planemo, a third party tool which allows tool creation, validation, and publication of Galaxy projects. Galaxy was used as the source system, whereas iRODS in context of CyVerse Austria was used as the target system for file transfer. Besides native Python code, the Python packages `python-irods-client`, `requests`, and `h5py` were used. The `python-irods-client` module provides a framework to establish a connection with an iRODS zone. The request module is used to send http requests. Finally, the `h5py` package enables file manipulation, which is used to handle files for data transfer. Throughout the course of development a GitHub repository was used to host the codebase and document the project's progress. Finally, the Galaxy Tool Shed was used to publish the interface and its tools to the public.

4.4 Interface Implementation

After the requirements were finalized, the design concepts was conceived and agreed upon, the implementation process of the interface started. First, a test version of Galaxy was set up to develop and test the tools. Additionally, research was conducted to find documentation on how tools are created for Galaxy. During that phase, Planemo was found, which prooved helpful during the development of the tools. Planemo provided a way to create templates for tools as well as provide an example tool implementation as reference. The reference tool acted as a starting point for the actual tool implementation. Thus, the first prototype of the tools were created. The first task was to ascertain the required parameters necessary for the execution of a data transfer request. With this information, the tool templates were adapted to allow users to enter the required information. After that, the module to authenticate users and perform data transfer requests was developed. The tools were then written one after the other, starting with the download version. The rest of the implementation phase was done in an iterative feedback loop involving the project sponsor.

Figure 4.1 gives an overview of the environment in which the interface was implemented. Planemo and its containing Galaxy version was primarily used to develop the tools. To test the tools whenever progress was made, they were manually transported to a previously set up Galaxy testing environment, to mimic a real system. Additionally, a GitHub repository was used to keep track of changes during the development phase. Finally, after successfully verifying the tools through testing, the tools were then published to the official Galaxy tool shed. From there, they were installed and tested on a real Galaxy environment.

4.4.1 Course of Development

The interface was implemented throughout the course of 2021, starting with the tool creation with Planemo. The two tools and the auxiliary module were then developed within the test environment of Planemo. After testing the tools on a private instance of Galaxy, the initial prototype was published to the tool shed on the 23rd of June, 2021. Subsequently, forward version

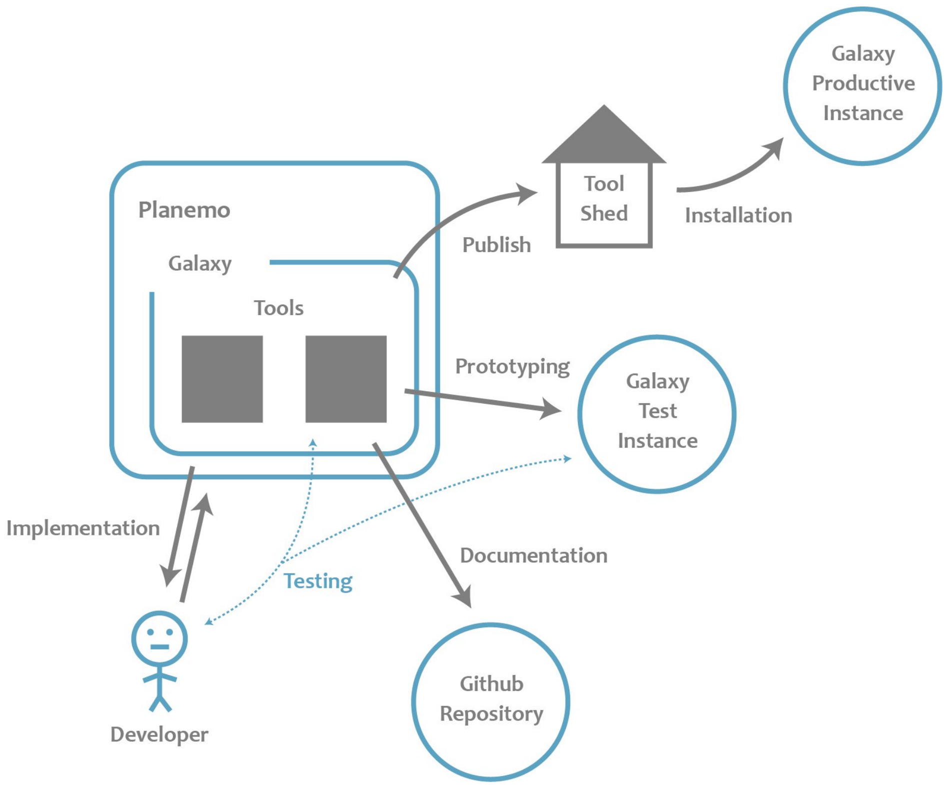


Figure 4.1: Showcase of the implementation environment which was used during the development phase of this thesis' project, providing visual clues on how the individual components interact with each other.

updates were conducted in consultation with the project sponsor. This process involved feedback loops and testing on the real Galaxy system, including on-site installation and verification. The timeline of published interface versions was as follows:

- **Version 0.0.1** published on *23rd of June 2021*
- **Version 0.0.2** published on *2nd of July 2021*
- **Version 0.0.3** published on *19th of July 2021*
- **Version 0.0.4** published on *4th of August 2021*

Version 0.0.1 was the first published prototype of the interface tools. On version 0.0.2 some bugs were resolved connected to the tool's data file paths. Version 0.0.3 resolved an issue with a required registry file as a dependency. Finally, version 0.0.4 forms the currently up-to-date and fully working implementations of the interface tools. After the final installation of the latest interface version, extensive testing was conducted. Luckily no further bugs arose. At this point the implementation phase as well as the development of the tools was completed and the evaluation phase of the project started.

4.5 Summary of Chapter 4

The design and implementation of the project is explained in detail throughout this chapter, starting off with an in-depth description of requirements which were already announced in the introduction. The individual requirements are explained and put into context, also explaining the reasoning behind them. With the functional and non-functional requirements closed, the next section deals with the design decision made for the projects. The 4 design principles, which were valued throughout the implementation phase are described and justified. Furthermore, the technologies used for the development of the interface are enlisted in the subsequent. Additionally, a description on how the individual components were used together is given.

Next, the implementation process of the tools is described. This description includes a figure to showcase the process. We clear up all actions leading to the final result. Finally, the development trajectory is particularized, explaining the progress and important milestones achieved during its course.

In conclusion, this chapter's purpose is to clarify the decisions made for the project - why they are important and how they are embodied throughout the implementation phase.

5 Evaluation

"We can't improve when we can't measure."

Bharath Mamidoju

The Implementation phase is usually followed by the Evaluation phase in software development. Its main goal is to check the results, evaluate user feedback, and, if necessary, initialize further developments as requested by clients. In other words, the evaluation phase can be broken down to addressing two simple questions: *Are (all of) the requirements met?* and *Is the project fit for its purpose?* These questions differentiate in perspective. The former question is primarily targeted at the customer or potential users, whereas the latter focuses on the project in connection to the owners. Transposing the evaluation of software to the evaluation of a physical project could be figuratively compared to the task of building a bridge. After the completion of the bridge a similar evaluation process takes place. The first question, regarding the customer, would be answered in collaboration with the client: Is the bridge built as wanted? Is it located at the right place? Does it look the way it was intended? Whereas the later question would be answered by examining the bridge itself: Is the bridge usable by whoever it was designed for? How much weight can it hold? Under which circumstances would it crumble? Switching back to the realm of software engineering, the feedback-centered question can be answered straight forwardly in a review process involving the customer. However, the project-perspective question is more complex to answer. This is caused by the versatility involved in the outcome of software engineering projects. Thus, making it seem impossible to find a unified schema to evaluate software projects. To help evaluating results, so called evaluation criteria

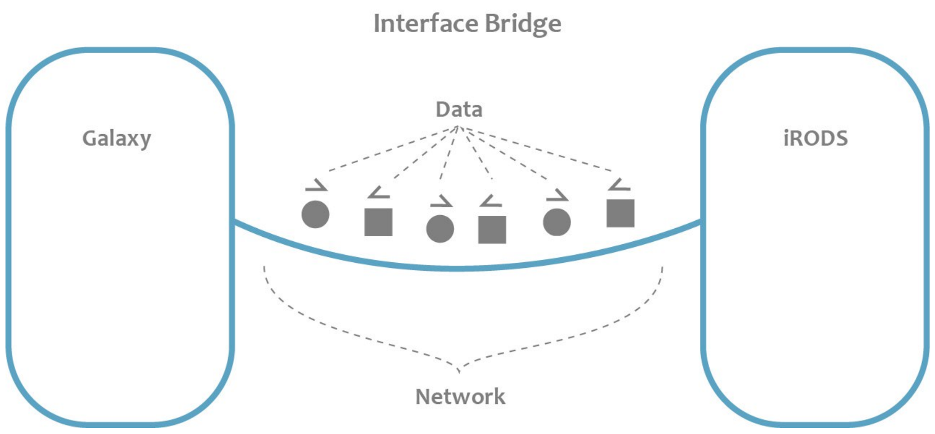


Figure 5.1: Illustration of the bridge-interface comparison, which describes how an interface could be compared to a bridge.

can be used and applied on an individual level (Hegner, 2003).

Applying the bridge example to this project, an interface module can be figuratively seen as a bridge-like object. In this case, the bridge is not traversed by humans or vehicles, but data. Similarly, the entrance and exit of the bridge can be compared to the origin and target of an interface. However, with the addition that the bridge cannot be crossed when one of the endpoints is missing (which should never be the case for a bridge) but is analog to one of the systems being unavailable for the interface. Additionally, the bearing load is comparable to the maximum load of data transmittable via the interface. However, this is not merely influenced by the interface module itself, but rather the surrounding infrastructure the module is embedded in, e.g., network bandwidth or transfer speed (channel capacity).

Figure 5.1 showcases the analogy of comparing an interface module to a bridge.

5.1 Methods of Evaluation

The methods for this project's evaluation are twofold. The prime evaluation method is linked to user feedback, whereas the second method is targeted at the project's results and performance.

First, there are many different methods of generating and evaluating **user feedback**. For software projects targeted at a larger user base, user reviews are most commonly used. This method consists of ratings, often on a scale of 1 to 5, which are used to reflect a summarized experience review of the users. However, methods like this can only be used with a larger number of users because too little data might lead to spurious evaluations. In addition to that, questionnaires can be used to attain feedback from willing users. For this method, questionnaires are sent to either the entire user base, or a part of it, with expectations of speedy user responses. Yet, evaluating projects this way requires a relatively large and diverse group of users. As a third option, direct user interviews are possible. Besides drawing a lot more time and generating more effort for evaluators, this method can be used for any size of user base. Because this project's (initial) target user base is relatively small, almost targeted at individuals, the third option was the method of choice.

Second, evaluating the **performance** and **project results** is easiest done by comparing the outcome to alternatives. In case of this project, there is no trivial way to compare the procedure of this interface to a comparable system. So, the next best option was taken, by comparing the functionality of the system to alternative ways to achieve the same task. For this, a particular user task was chosen, executed with the system and finally compared to two alternatives. In addition to that, use-case related tests were conducted to ensure full functionality of the interface.

The following chapters will elaborate on the two methods chosen to evaluate this project.

5.2 User Feedback

To avoid ending up with an incorrectly working system, user feedback was omnipresent throughout the entire phase of implementation for this project. After arranging the requirements for the interface, the project sponsor and the developer agreed upon having reoccurring meetings. During the course of development this decision proved to be beneficial for both parties involved in the project. The issuing person of the project had the ability to directly contribute his feedback, thus being sure the interface tools were designed and developed in a fitting way. Furthermore, the developer of the tools was able to propose ideas while getting immediate feedback. This way, making it impossible to drift off into unwanted functionality during development. Thus, the drawn through feedback loop had a major impact on the successful and proper completion of the project. This way of including feedback directly into the development process can best be described as agile development. Considering daily stand-up meetings between the developer and a senior developer as well as scheduled reoccurring feedback evaluations with the project sponsor and the developer. Finally, the concluding task of setting up the interface module was also done in cooperation of both parties.

After the implementation phase was completed, an evaluation interview with the project issuer was conducted. This interview verified that everything was working as intended. In addition, indirect user feedback was forwarded to the developer, including ideas for improvement and concepts for future work. During this meeting the interface module was confirmed to be fully working as intended and fulfilling all of the agreed upon requirements. A collection of ideas to improve the interface in the future, which arose from the meeting, is explained in more detail in section 6.3 - Outlook and Future Work.

5.3 Project Performance & Results

The functionality of the interface tools was verified by testing. For this, not only use-cases scenarios were applied but also system-behavior tests were conducted to ensure coverage of a broad spectrum of data transfers. The use-cases consisted of all tasks users might perform while using the particular tools. This involves data management tasks, data sharing, and redirecting the acquired data sets to other tools to ensure sound data transfer. The system-behavior tests were primarily focused on the data that was being transferred, which involved a list of commonly used file formats and a set of actual test files. Additionally, network issues were simulated to make sure that the interface still behaves as intended, even in case of environmental failure. Under any circumstances, the data transfer process should either complete successfully, or, when experiencing troubles, abort and remove any kind of artifacts. The above-mentioned properties were tested successfully on the latest version of the interface module.

To measure performance of the interface, a likely use-case was chosen and applied for three scenarios, measuring the time for users to complete each scenario. The data that was used for the use-case execution was a 50MB file, taken from the sample data set provided for testing. Furthermore, each scenario forms an alternative way of solving the particular use-case. The use-case and scenarios are as follows:

UC: *Share data stored in a Galaxy history entry with a collaborator via CyVerse's discovery environment.*

- **S1** Transfer data from Galaxy via the interface module and share the file via CyVerse's discovery environment.
- **S2** Manually export the data from Galaxy and import it into iRODS via CyVerse, then share the file via CyVerse's discovery environment.
- **S3** Manually export the data from Galaxy and transfer the data to a USB drive, then physically share the USB drive with a collaborator.

Figures 5.2, 5.3 and 5.4 show the results of the individual Scenarios' outcome. To measure the results, the Scenarios were cut to single tasks. The tasks were then performed by 4 researchers with varying experience using the interface and the time was taken for each step. Furthermore, the times of all users was summed up to calculate the average time it took to achieve the tasks. The average time is visualized in the figures. To provide a better understanding of the individual tasks which were created out of the scenarios, they are listed and explained in more detail.

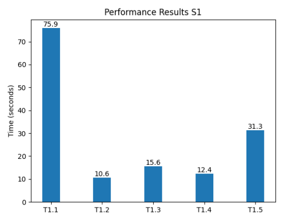


Figure 5.2: Results of S1.

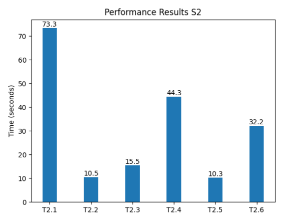


Figure 5.3: Results of S2.

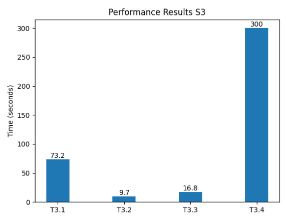


Figure 5.4: Results of S3.

Scenario 1 consisted of five tasks, which are listed below:

T1.1 Transfer data, already present in a Galaxy history entry, by using the Upload Tool from the interface module.
T1.2 Navigate to CyVerse's Discovery Environment.
T1.3 Log into the system using a test user account (credentials provided).
T1.4 Find the data in the Discovery Environment (path provided by Upload Tool).
T1.5 Use CyVerse's data sharing feature to share the data with a collaborator (collaborator username and email provided).

Scenario 2 consisted of six tasks, which are listed below:

T2.1 Download data, already present in a Galaxy history entry, by manually exporting it from the Galaxy instance.
T2.2 Navigate to CyVerse's Discovery Environment.
T2.3 Log into the system using a test user account (credentials provided).
T2.4 Upload the data, previously downloaded from Galaxy, through the Discovery Environment.
T2.5 Find the data in the Discovery Environment.
T2.6 Use CyVerse's data sharing feature to share the data with a collaborator (collaborator username and email provided).

Scenario 3 consisted of four tasks, which are listed below:

T3.1 Download data, already present in a Galaxy history entry, by manually exporting it from the Galaxy instance.
T3.2 Plug in a USB stick (Hardware provided).
T3.3 Copy the previously downloaded data to the USB stick.

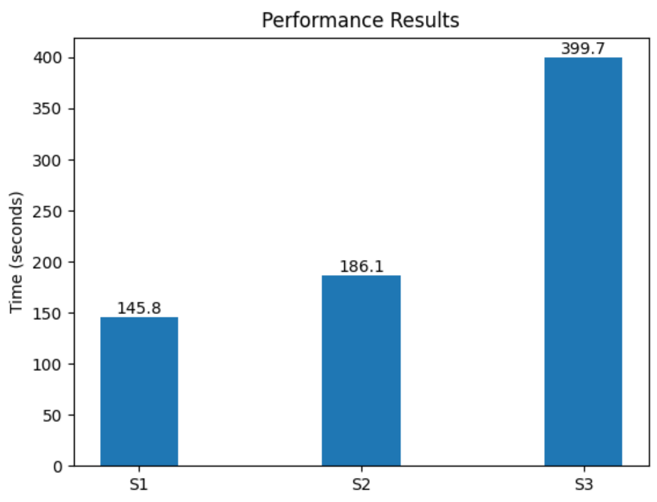


Figure 5.5: Result chart of the conducted performance testing providing a visual comparison of the three executed scenarios.

T3.4 Physically share the USB stick with a collaborator (estimation of 300 seconds).

Figure 5.5 illustrates the results of the performance testing by comparing the outcome of all three scenarios. Scenario 1, using the interface tool, proved to be the quickest with 146 seconds. The second Scenario, which involved manual recreation of the tools job, turned out to be the second fastest with 186 seconds. Finally, the third scenario emerged as the slowest option with 400 seconds required to complete the tasks.

It is important to mention, that for the third scenario an estimation of 5 minutes was taken to physically move the USB stick to another location. This was necessary, because of the high variance this task could imply. Considering the best-case scenario, the collaborator might be placed in the same office space as the user. This would reduce the time to execute the task down to few seconds. However, the worst-case scenario of this task might be that the target collaborator is located very far away (e.g., across the city, another town or even in another country). This would potentially increase the time to complete the task to hours, or maybe even days, which also increases the chances of the task not being fulfilled that way. Finally, the likely-case of 5 minutes was estimated to achieve this task. For this, it was assumed that the collaborator is located in the same company building, but placed in another office room. Thus, the task of walking from office room "A" to office room "B" was estimated with 5 minutes.

Furthermore, it is also noteworthy to highlight the performance of using the tool individually. When compared to manually handling the data, the tools execution time (T1.1) is close to the time it takes just to download the data (T2.1 & T3.1). Thus, disregarding the additional task of sharing the data, it is eligible to assume that the tool's usage outperforms manual execution clearly.

5.4 Summary of Chapter 5

This chapter comprises the evaluation results of the project. Initially, the separation between user-centered evaluation and project-centered evaluation is made. Additionally, we explain how both perspectives are important to evaluate a project successfully. To achieve that, the physical example of building a bridge is presented. With that idea, the analogy of comparing a bridge to an interface module is introduced.

After that, the chosen methods of evaluation are explained. In my thesis, we focused on user feedback with user interviews for the user-centered evaluation, and on performance and requirement fulfillment for our evaluation. The user feedback took place throughout the entire course of the project through reoccurring meetings. In addition, a completion meeting was conducted at the end of the project, which resulted in ideas for the future. The performance was measured by verifying all of the requirements that were fulfilled. As an extra measure, a use-case experiment was conducted, which compared the usage of the interface to two alternative ways of accomplishing the job of sharing data with a collaborator. The results suggest that using the interface module is more efficient than the other approaches.

With this chapter, the evaluation process of the project is embedded into the thesis, emphasising on two evaluation perspectives.

6 Lessons Learned

"Experience is a hard teacher
because she gives the test first,
the lesson afterward."

Vernon Law

When working on a long-term software project not all things work out exactly as planned. Requirements might change during the course of development, issues in management might result in set-backs for the implementation phase, or technical troubles might lead to postponement of the completion of the project. This chapter's goal is to perform a critical reflection on the realisation of this project. The course of the project will be examined in detail, elaborating on what worked out well, but also on the decisions and actions that did not turn out to be beneficial in the end. Such general observations can only be conducted in hindsight because, what might have seemed to be a save bet during the project's life cycle, might turn out to be a horrible mistake in hindsight.

Beginning with a list of encountered pitfalls along the way, this chapter will start of with *what mistakes were encountered* during execution of the project. This is followed by critical reflection on the decisions made before and throughout the implementation of the project. Finally, a target-actual comparison will conclude this work, emphasizing on the goals achieved in the end. Finally, before a short summary of mistakes and successes closes this chapter, an outlook into the future of the interface module is given.

6.1 Pitfalls

A pitfall is defined as an unexpected error or difficulty which is encountered during the course of a project. It is intended to warn and thus prevent a repeat of the same mistakes in the future. According to Zilinskas (2019), pitfalls happen especially often in context of software development projects. Thus, this section lists the most important pitfalls encountered during this project.

Test Environment != Real System

One would think that testing on what was assumed to be an identical set up of a test environment would ensure that everything behaves the same on the real system. In case of this project, that assumption proved to be wrong. Although the version numbers of the Galaxy instances matched exactly, the systems behaved differently. This was primarily due to the network embedding where the real system was running, but also due to very specific additional options set for the actual instance.
To avoid issues like this, the testing environment should be placed within the network of the actual system. Additionally, the test system should not be set up independently but rather be a derivative of the real system.

This is a 6! - No, this is obviously a 9?!

The ancient controversial subject of perspective also proved to be immanent in this project. This was mainly caused by the intersection of two worlds, becoming clearly visible during the course of this project. *"Wait, so a single galaxy history entry can consist of multiple files?"* or *"There is a difference for transferring different kind of data?"* are just two of many realizations which arose during meetings throughout the implementation of the interface module. This highlighted once more that, what might be obvious to one person, might be astonishing knowledge for someone else.
To really make another person understand what you are saying, sometimes it might be necessary to explain things as simple as possible, even if things may seem obvious.

Underestimating Framework Restrictions

When designing user interfaces, which are embedded in another piece of software, some things might not be working as planned. In case of this project, it was expected that Galaxy comprises a way of allowing to input data in a hidden way. However, during the course of development it became clear that Galaxy does not include a feature like this. Thus, a compromise had to be made to allow users to enter their passwords in plain text, which is generally considered very bad practice. But, regarding the alternative options this seemed to be the only way possible in order to still ensure simplicity of the interface tools.
Problems like these can be avoided by sketching a possible user interface before starting development. When making prototypes on the target system, issues like this become obvious immediately.

6.2 Critical Reflection

Although some minor issues arose during the entire course of the project, the results confirmed the tools to be successful in the end. The steady flow of communication, in terms of regular meetings during the development phase, proved to be very influential on the performance of the interface. Additionally, taking the time to think and discuss the requirements before the implementation phase started turned out to be very helpful. Without a doubt, the project was not always smooth sailing. During the course of the project, the development demanded a lot of effort from everyone involved (in the project). Some problems took many hours to resolve, other issues were fixed within minutes. Furthermore, many hours were spent to achieve consensus with regards to design. Yet, despite the setbacks encountered during and near the end of the interface's release, the project concluded with pleasing results, which ultimately resulted in the both-sided approval for finalisation of the project.

6.2.1 Comparison between set Goals and Results

This section, which evaluates the final result of the project to the planned goal, is divided into two parts. First, the prime evaluation factor will be the previously agreed upon requirements, whereas the second criteria will be based on the project's performance.

In regards to requirements, the development process of the interface module was able to fulfill all of the requirements. Whereby the main focus was laid upon implementing the functional requirements, non-functional requirements were also considered throughout the entire phase of implementation. In terms of functional requirements, the three requirements: **R1** - *User Authentication*, **R2** - *Data Upload* and **R3** - *Data Download* were fulfilled for the final version of the interface tool. Furthermore, the non-functional requirements **R4** - *Security*, **R5** - *Availability*, **R6** - *Scalability*, **R7** - *Reliability* and **R8** - *Reusability* were enforced throughout the entire course of implementation as well.

Finally, the performance evaluation indicated, that using the interface module resulted in a much more efficient performance by comparing the different ways of conducting the use-case *Share data stored in a Galaxy history entry with a collaborator via CyVerse's discovery environment.*. In comparison to the two alternative ways, **S2** - "Manually export the data from Galaxy and import it into CyVerse data management, then share the file via CyVerse's discovery environment." and **S3** - "Manually export the data from Galaxy and transfer the data to a USB drive, then physically share the USB drive with a collaborator.", using the interface module as **S1**, proved to be much more efficient.

6.3 Outlook and Future Work

This section covers ideas obtained from retrospect on the project with regards to possible future work as well as an outlook for the project. The aforementioned ideas are as follows:

I1 A dedicated user interface to select files when importing data objects or collections might be useful.
I2 When exporting data from iRODS into the Galaxy history, there should be an option to create multiple history entries instead of just one history entry for data collections.
I3 There should be an option to authenticate the user for the target iRODS zone by using a particular environment file, which can be stored locally.

I1 emerged from issues with file selection when using the interface tool. Currently, it is necessary to know the exact path to files stored within the iRODS zone to initialize file transfer. A dedicated user interface would allow users to browse through their available data, thus allowing them to select their files dynamically. When downloading multiple files simultaneously from iRODS into Galaxy, the set of files is treated as a collection. This means that multiple files will be combined into a single Galaxy history entry. Users can manually split this entry into individual files to run further tasks for just a particular data file. **I2** forms a solution to this issue, by allowing users to set an option to split files into separate Galaxy entries automatically. **I3** would enable users to store their user credentials in a particular environment file, which would replace the need to enter the credentials over and over for every tool request. This would provide additional comfort, especially when using the tools very frequently.

Currently, the interface tool is being used at the Medical University of Graz, it is performing all of the desired tasks, following the security requirements. The solution is expected to continue being used for the foreseeable future. This conclusion is drawn, from the feedback received throughout the project and the current user base at the target University. As of writing, the tool has been installed twenty-three times so far, suggesting that its usage is not just limited to the Medical University of Graz. Its open source nature allows for this. Furthermore, improvements to the interface tools are expected from the Galaxy community and whatever subsequent frameworks it is extended to and thus, will survive the life span of this project and thesis. The interface in its latest version can be accessed, downloaded and further developed through the Interface Tool Shed Entry (2021).

6.4 Summary of Chapter 6

This final chapter revolves around important events that happened during the project. This is done by pointing out what worked out great as well as by mentioning what went wrong and should not be done the same way in retrospect.

First, a list of pitfalls is presented, which form the most important errors that happened during the project. Besides underestimating the differences between a test and a productive environment of a system, there were issues with opposing perspectives along the way and restrictions caused by the framework which were noticed too late. Before comparing the planned goals do the achieved results in a target-actual way, a section regarding critical reflection gives a short conclusion of the project's implementation. In the end, an outlook into the future is proposed and ideas for future tasks and the interface's life cycle after the project are presented.

This chapter includes an assessment into the thesis by applying critical thinking and critical evaluation of the attained results.

Appendix

Bibliography

Bhisey, Rohit (May 2021). *Life Sciences Tools Market: Rise of the research and development sector has provided significant boost to the market*. URL: https://www.biospace.com/article/life-sciences-tools-market-rise-of-the-research-and-development-sector-has-provided-significant-boost-to-the-market/ (cit. on p. 5).

chitrasingla2001 (Apr. 2020). *Functional vs Non Functional Requirements*. URL: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/ (cit. on p. 10).

CyVerse Austria (Mar. 2022). *CyVerse Austria Homepage*. URL: https://www.tugraz.at/sites/cat/home/ (cit. on pp. v, 6, 7, 20).

CyVerse National Science Foundation (Mar. 2022). *CyVerse Homepage*. URL: https://cyverse.org/ (cit. on p. 19).

Docker (Mar. 2022). *Docker Homepage*. URL: https://www.docker.com/ (cit. on p. 19).

Docker Inc. (Mar. 2022). *Docker Hub*. URL: https://hub.docker.com (cit. on p. 21).

FFG (Jan. 2010). *Life Sciences*. URL: https://www.ffg.at/en/content/life-sciences-overview (cit. on p. 4).

Galaxy Community (Mar. 2022). *Galaxy Tool Shed*. URL: https://toolshed.g2.bx.psu.edu/ (cit. on p. 27).

GrandViewResearch (June 2021). *Life Science Tools Market Size & Share Report*. URL: https://www.grandviewresearch.com/industry-analysis/life-science-tools-market (cit. on p. 5).

Hegner, Marcus (May 2003). *Methoden zur Evaluation von Software*. URL: https://www.gesis.org/fileadmin/upload/forschung/publikationen/gesis_reihen/iz_arbeitsberichte/ab_29.pdf (cit. on p. 44).

Interface GitHub Repository (Aug. 2021). *Interface GitHub Repository*. URL: https://github.com/RHohensinner/Galaxy-iRODS-Interface (cit. on p. 27).

Interface Tool Shed Entry (Aug. 2021). *Interface Tool Shed.* URL: https://toolshed.g2.bx.psu.edu/repository?repository_id=8cf3752f7534e9b7&changeset_revision=19c1cecdfdfd (cit. on pp. 27, 57).

iRODS (Dec. 2021). *Open Source Data Management Software.* URL: https://irods.org/ (cit. on pp. v, 6, 9, 21).

iRODS Consortium (Jan. 2016). *iRODS Technical Overview.* URL: https://irods.org/uploads/2016/06/technical-overview-2016-web.pdf (cit. on p. 21).

Ison, Jon et al. (2019). "The bio. tools registry of software tools and data resources for the life sciences." In: *Genome biology* 20.1, pp. 1–4 (cit. on p. 6).

LISA (Jan. 2022). *Life Science Austria - Interesting Facts.* URL: https://www.lifescienceaustria.at/life-science-in-austria/interesting-facts (cit. on p. 4).

Magner, L.N. (2002). *A History of the Life Sciences, Revised and Expanded.* CRC Press. ISBN: 9780203911006. URL: https://books.google.at/books?id=YKJ6gVYbrGwC (cit. on p. 2).

Nextcloud (Mar. 2022). *Nextcloud Homepage.* URL: https://nextcloud.com/ (cit. on p. 7).

ownCloud (Mar. 2022). *ownCloud Homepage.* URL: https://owncloud.com/de/ (cit. on p. 7).

STANDARD (Mar. 2012). *Was alles zu Life-Sciences gehört.* URL: https://www.derstandard.at/story/1332323642715/wissen-was-alles-zu-life-sciences-gehoert (cit. on p. 1).

STATISTICS-AUSTRIA (Nov. 2021). *Students, studies.* URL: https://www.statistik.at/web_en/statistics/PeopleSociety/education/universities/students_studies/index.html (cit. on p. 4).

Stedman, Craig (Oct. 2019). *What is data mangement and why is it important?* URL: https://www.techtarget.com/searchdatamanagement/definition/data-management (cit. on p. 15).

Thaktur, Dinesh (Mar. 2022). *Principles of Software Design & Concepts in Software Engineering.* URL: https://ecomputernotes.com/software-engineering/principles-of-software-design-and-concepts (cit. on p. 36).

The Galaxy Project (Dec. 2021). *Galaxy Community Hub.* URL: https://galaxyproject.org/ (cit. on pp. v, 6, 8, 9, 23).

The University of Sheffield (Jan. 2022). *What is research data management?* URL: https://www.sheffield.ac.uk/library/rdm/whatisrdm (cit. on p. 14).

Zilinskas, Shane (Sept. 2019). *Common Pitfalls of Software Development and Tips to Avoid Them*. URL: https://devops.com/common-pitfalls-of-software-development-and-tips-to-avoid-them/ (cit. on p. 53).