Honours Project Report

Spacial Management of African Sites and Heritage: Workflow Management System

Author:
Michiel Johan Baird

Supervisor:
Dr. Hussein Suleman

	Category	Min	Max	Chosen
1	Requirements Analysis and Design	0	20	15
2	Theoretical Analysis	0	25	0
3	Experiment Design and Execution	0	20	10
4	System Development and Implementation	0	15	15
5	Results Findings and Conclusion	10	20	10
6	Aim Formulation and Background Work	10	15	10
7	Quality of Report Writing and Presentation	10		10
8	Adherence to Project Proposal and Quality of Deliverables	10		10
9	Overall General Project Evaluation	0	10	0
Total		**80**		**80**

University of Cape Town

Department of Computer Science

October 29, 2012

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Abstract

Workflow Management Systems have been highly successful in managing complicated tasks and procedures, both for business and scientific applications. This research project follows the development of an *Automatic Workflow System* designed to meet the needs of the tasks that are present within the *Zamani Project*. This is developed to handle very large files, which is a common within the project. The system was developed in three design iterations built from the requirements and considerations of the group. A Web interface is used to control the system. The system was built in Python using the *Django* framework along with JQuery and other front-end libraries.

Both automated and user tasks are supported in the system. For user tasks, the required files are automatically copied to the user's workstation. This is done using *rsync* to reduce transfer times; as well as protect effectively against network failure that can occur. Once the task has been completed, the transfer process is initiated in the opposite direction. To improve the quality of the work, all user tasks require being validated by senior members on the team. Simple error recovery was built into the system, allowing users to inspect the logging information to determine the cause of the failure, allowing the tasks to be restarted as soon as the problem has been alleviated.

The system was evaluated by implementing a subsection of the Zamani tasks. These tasks were successfully completed within the system. A User Experience evaluation was conducted that shows that users were satisfied with the system. It was also found that the system was easy to use and easy to learn.

Acknowledgements

First of all I would like to thank my supervisor in the project, Dr Hussein Suleman. Your advice and leadership was instrumental throughout this entire project. Thank you so much for the early mornings spent reviewing drafts and the advice given when I had no idea of the direction that I needed to go in. Your ability to keep conversation interesting helped me keep my sanity through this process. For that I thank you.

To the National Research Foundation(NRF), thank you for the financial assistance given towards this degree and in particular the project. This made the process a lot more comfortable and enjoyable.

To the members of the Zamani Project I interacted with: Professor Heinz Ruther, Stephan Wessels and Ralf Schroeder, thank you for the time and effort that you put in to help make this project possible. The use of your data is highly appreciated and valued. A special thank to Roshan Bhurtha, who was my main point of contact within the team. Thank you for the quick responses when I had any queries.

To my project partner, Tim: thanks for being one of the most exceptional people I have ever met. I really look up to you. Your intelligence is only matched by your kindness. Thank you for an amazing year and I will treasure your friendship always. To the rest of the CS Honours class of 2012, it has been a great experience getting to know you guys; sharing in both the fun and hard times. A special thank you to Simba Nyatsanga and Marco Lawrence, for the long nights spent in the laboratory, the food runs, the jokes, the emotional support. Thank you for making this project enjoyable. To Kaitlyn Crawford and Henk Joubert, thank you for being the exceptional people you are. Your friendship during this year has been great. I look forward to working with you next year.

Lastly, I would like to thank my parents: John and Wilma Baird. Without you I literally would not have been here. Thank you for all the support, both emotionally and financially. It was quite convenient being able to phone you up at 3am in the mornings. Thank you for being the best parents anyone could ask for. I am extremely proud to be called your son. I love you, and I will always strive to make you proud.

Contents

List of Figures

Chapter 1

Introduction

Workflow Management Systems are systems designed to facilitate and manage tasks within an organisation. These tasks are topologically organised and the system then executes the workflow such that each task's dependencies are first met. Such a system would allow a user to set up, monitor and execute tasks [Slot and van Zoolingen, 2005].

Workflow systems have been successfully implemented in various business and Scientific fields [Brahe and Schmidt, 2007]. This has not only increased productivity but in the field of science has allowed that the reproducibility of results be greatly increased[De Roure and Goble, 2009].

In recent years scientific discovery has move towards being more data-driven[Gray and Szalay, 2007]. This has created a process where large amounts of data is collected, processed and then analysed to produce results. Where in the past results were based in simulating and evaluating experimental models, it has become important that results are based in data. This method has been highly effective across various fields including: Pharmacology[Harpaz et al., 2012], Astrophysics[Thomas et al., 2011] and Bioinformatics[Greene and Troyanskaya, 2010]. In order to support these processes, sophisticated tools are required to process and analyse this data[Shneiderman, 2002].

Management of this data is an extremely important process that ensures the ability to produce these scientific results[Gray et al., 2005]. Workflow Management Systems, is an important component in this as it provides a means to very accurately record the scientific process followed [Davidson et al., 2007]. Effective management of the data allows for complex analysis to be performed, which ultimately leads to scientific discovery[Ludäscher et al., 2009].

1.1 Zamani Project and Cultural Heritage

The *Zamani Project*,[1] based at the Geomatics Department at the University of Cape Town, aims to accurately record the physical and architectural nature and dimensions of African Heritage sites.

Heritage sites are mapped using sophisticated technology that is used to create a variety of digital objects, such as: i) Geographic Information Systems; ii) 3D Computer Models; and iii) Panoramic Photographs. The team has recorded sites in Ghana, Mali, Kenya, Ethiopia, Tanzania and South Africa. Other sites are currently in progress or being planned.

These are some of the best, and most accurate heritage documentation in the world. A very large collection of data items has to be managed within the project. Currently, the team is experiencing difficulty managing the vast amount of data items that needs to be processed. The size of these data items provides a challenge to traditional workflow systems that are currently available.

To document these heritage sites require a great deal of effort. The process includes the capture, storage, manipulation, analysis and management of this geographic, architectural and photographic data. This data is very large and diverse. It gets used in very diverse ways. This involves processes

[1]Zamani Project:http://www.zamani-project.org/

abstracting these data items into various forms. Managing the data in such a way that it is accessible at any point presents a challenge due to the size of the data.

Currently, the documentation process of the heritage sites involves the data being manually copied to each point where it is required. This is an extremely slow and laborious process. By automating this process time could be saved.

1.2 General Outline

The project is divided into two major components. The first is a workflow management system developed by Michiel Johan Baird. The second is an indexing and streaming system for large point clouds developed by Timothy Trewartha. These two components aim to solve some of the challenges faced by the members of the Zamani Project.

1.2.1 Workflow Managment System

In order to assists with the difficulty in managing the data and the tasks involved in creating the heritage data, a Workflow Management System was developed using the requirements of the team.

This system is able to handle both automated and manual user tasks. The aim is to increase the efficiency of the task execution and improving reusability between different heritage sites sites.

1.2.2 Point Cloud Indexing

The other component tackles the difficulty involved in managing large point clouds. The approach taken is to build an index for the point cloud. This index allows for efficient region extraction. It also allows the user to specify the resolution at which they wish the extraction to be performed. The system also enables the extractions from the point cloud index to be streamed from a central server. For example, using the index a low resolution representation can be efficiently extracted and sent to the client. Alternatively, the user may request a high resolution extraction of a subregion.

These two separate components can interact in a number of ways. For example, at various stages the Workflow Management System requires point clouds as input to certain processes. This input can be efficiently extracted from the indexed point cloud using the point cloud streaming system. Alternatively, if the Workflow Managment System produces a point cloud as output (after cleaning for example) it can be sent to the indexer so it can be more efficiently accessed further down the workflow pipeline. System.

This report discusses the development and evaluation of the Workflow Management System.

1.3 Problem Statement

Workflow management systems decompose complicated procedures into small atomic tasks that are dependent on one an other[Taylor et al., 2006]. This decomposition often leads to an increase in overall efficiency of the process. Workflow Systems are particularly efficient in instances where there are multi-person teams and work is done in mostly independent tasks. These are the exact conditions that are present within the Zamani-project.

The aim of this project is to develop and evaluate a system that is is applicable to workflows similar to those found within the Zamani Project. This should allow the system to: i) interface with existing systems; ii) manage the work both at a site and a user level; iii) provide local data availability when users require it for a task; and iv) increase the overall efficiency of the process.

1.4 System Outline

This project is about a Workflow Management system that could effectively automate the management of the heritage preservation processes. This would manage the process starting from the raw scans and photographs through the creation of all the digital artifacts. Figure 1.1 shows the simplified operation of the system.

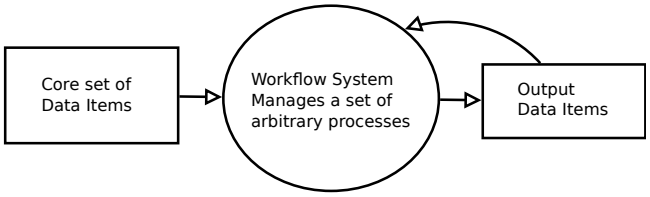


Figure 1.1: Basic System Overview

The system would starts with a core set of input files. The workflow is then executed, which initiates a set of topologically-sorted tasks. These tasks would then be executed in order, such that the processes then generate the full set of output data items. These output items can then be used as input files in remaining tasks.

These processes can be ordered into two main categories: (i) tasks that can automatically be executed by the system; and (ii) tasks that need to be completed by a user. The system aims to coordinate an automate these tasks to the maximum extent possible whilst managing the information.

Executing these processes involves using various software packages. This functionality can not be replaced by the system so it would need to be integrated in a very general way, this would be done by using wrapper software software that would use these applications to complete the required goal.

1.5 System Evaluation

In order to determine the applicability of the proposed solution, the system was evaluated in the following ways:

Partial Integration Case Study
: In order to determine whether the proposed system would be applicable to the problem at hand, a portion of the workflow of an existing site was implemented and tested within the system. This test provides a means to determine whether the problem could be solved by the proposed system.

User Experience Testing
: Even the most effective system can fail if it cannot be successfully be adopted by users. Therefore, the system's user experience was tested. This allowed problems to be detected and shortcomings of the system to be identified[Tullis and Albert, 2008].

1.6 Legal and Ethical Issues

All the software used in the development of the workflow system is either free or open source. If this project were to continue or be extended, there would not be any legal issues. The core of the system is written using the Django Framework, using Python. These are, respectively, licenced by the Django Licence and the PSF license. Further tools used during the implementation include Javascript which

is licenced under the LGPL. JQuery, JsPlumb and tablesorter, which are dual licenced the MIT license and GPL, Version 2.

Further data and information was obtained from the Zamani Project for testing the application. Permission was obtained to use this data.

User testing was also done during the course of the project. For this purpose, ethical clearance was granted from the Faculty of Science Research Ethics Committee. Access to UCT Students was also granted by the Department of Student Affairs.

1.7 Report Outline

This report outlines the design, implementation and evaluation of *Zamani Workflow*, a workflow system that sets out to solve the data management problems experienced within the Zamani Project. The report starts with a review of the literature in Chapter 2.

This is then followed by the Design and Implementation of the system in Chapter 3. The considerations for the design are defined before the full system design is reported. This design process involved three iterations.

The final system in then evaluated in Chapter 4. This was accomplished using a *User Experience* evaluation and an *Integration Case Study*.

The report is then concluded in Chapter 5 where the system is reviewed. Here possible future avenues of research is discussed that could be used to build on the project.

Chapter 2

Background

Workflow management systems define a complex process in terms well-defined tasks and coordinate process completion [Qiu et al., 2003]. Automated workflow management has been in wide use across various disciplines since the concept was formalised in 1996[Carstensen and Srensen, 1996]. Successful systems have been implemented across various, fields including banking and pharmaceuticals [Brahe and Schmidt, 2007, Johnson et al., 2009].

It has been shown to be very successful in the sciences as the same scientific process can easily be repeated on a different set of data[De Roure and Goble, 2009]. This not only aids in reproducibility but also saves time. This is done by efficiently abstracting the operations in the flow, allowing it to be automatically handled.

Geomatics is the field that concerns itself with the organisation, representation and processing of geographic data, for the purpose of querying it and making decissions off of the data [Di Martino et al., 2007]. The workflow in Geomatics is very distributed and the set of data that is operated on is large and diverse. Workflow management within Geomatics has been considered and solutions have been proposed, but not implemented or evaluated[Migliorini et al., 2011].

This chapter presents a discussion on Workflow Systems. Firstly presenting an overview of what these systems are and briefly looking into the histories of these systems. This is followed by a review of the factors that have influenced the success an failure of theses systems.

Section 2.2 does a review of the data and processing involved within the field of Geomatics.

This is then followed in Section 2.3 by a review of existing implementations of Workflow Management Systems namely: Kepler, Trident and Taverna. A variety of Case Studies ar presented in Section 2.4.

2.1 Overview

A workflow management system consists of definitions on how a set of tasks should be executed [Carstensen and Srensen, 1996, van der Aalst and Basten, 2002]. The overall procedure is defined by the following components: (i) actors, (ii) roles, (iii) responsibilities and obligations, (iv) tasks, (v) activities,(vi) conceptual structures and (vii) resources.

A real life problem or task can then be broken up into these components in such a way that the tasks represent a flow network. These tasks then connect to the actors and resources via the other components[Taylor et al., 2006, p. 4]. This allows tasks to be executed efficiently in a distributed manner.

The initial implementations of a workflow system, however, almost immediately failed. The systems were too rigid and was unable to accommodate the high levels of change that was required by the users [Suchman, 1983].

These changes come from a number of sources, including: ill-specification of initial problems, change in actors or resources, exceptions that occurred and new requirements. Adaptive work-

flow systems were proposed to solve this problem by providing a mechanism for allowing change in the system[van der Aalst and Basten, 2002]. This allows processes to be extended, replaced or re-ordered. It also adds the ability to change already running tasks by providing restart, transfer and proceed options.

Scientific workflow management has also been very successful with how experiments are defined, and, more importantly, reused. Another benefit that was quickly discovered was that it also allowed researchers to trade workflows, making the replication of results much easier than they were previously[De Roure and Goble, 2009]. Keys to this success were: that the workflow systems were made to fit the researchers; quick responses to adding required features when needed; listening to user input and making sharing of workflows as easy as possible.

Such a system has also been applied in fields that operate on large data sets, as would be the case if applied to problems in Geomatics Workflow systems were found to work well in the management of getting this data processed. Applying the concept to Observational Astrophysics, it revealed that it could be used to identify bottlenecks that could be optimised [Aragon and Runge, 2009]. Further, it was used to automatically ensure local access of large files that needed to be processed.

2.2 Geographic Data

Geomatics concerns itself with the collection, organisation and query of geographic data [Di Martino et al., 2007]. This data includes but is not limited to landscapes, coordinate data, building models, statistics, pictures, textures and routes. This is a very broad set of data, varying from very large to very small. That variation, however, means that there exists no uniform method to efficiently deal with the data.

The processing of this data can vary from human to software processing [Di Martino et al., 2007]. Various Web applications have been written to facilitate the tasks that need to be accomplished. This software is known as WebGIS and is becoming more popular with scientists; it also means that even within the field there is a strong shift toward Web-based services.

A key realisation with the usage of this data is that the same data is used across various applications, to create various amounts of abstractions[El Adnani et al., 2001]. The core data is seldom changed. Instead a new abstraction layer is added on top of it. The data can be thought of as a graph, where the nodes represent either a data or abstraction element, and the edges represent the functions/tasks required to create the particular abstraction as a set of topological relationships.

2.3 Implementations

There are various products available that can compose scientific workflows. *The Trident workbench* [Simmhan et al., 2009] is an open source workflow management system developed by Microsoft Research that also adds middleware services and a graphical composition interface. Trident builds workflows of control and data flows, off of built-in, user defined activities and nested subflows. The flows are represented using XOML, an XML Specification, while the activities are stored as a set of sub-routines[Simmhan and Barga, 2011]. Trident can be used on a local system, remote systems and even clusters. Queries on the system can be performed using LINQ.

Kepler is another scientific workflow management system that provides workflow design and execution. Actors are designed to perform independent tasks that can either be atomic or composite [Wang et al., 2009]. Composite actors(subflows) consist of multiple atomic actors bundled together. Actors can consume data and produce output, called tokens. Actors communicate tokens with each other via links. The order of execution and the links are defined by an independent entity called the director. As a consequence, the workflow can either be executed in a sequential or parallel manner.

Kepler effectively separates the workflow from its execution, allowing for easy batch execution. Actors can easily be exported and shared. Kepler is very popular due to its adaptability and easy integration.

Taverna is a scientific workbench that supports application-level workflow and does not focus on scheduling as much as others[De Roure and Goble, 2009]. Taverna has a strong focus on workflow sharing. Taverna is quite popular, since there exists a social network designed to facilitate workflow sharing among scientists(*myExperiment*). Services are linked to the model to execute the various tasks. Taverna can be used in such a way that it can utilize all the services a client has to facilitate the flow by easily adding services. The Taverna language is a simple data-flow language called the Simple Conceptual Unified Flow Language(SCUFL), that can be encoded in XML.

In order for these workbenches to be successful, there needs to exist a high level of interoperability between the workflow management and the services that are required [Shegalov et al., 2001]. However, due to the fact that there is a relatively high chance of failure when building this interoperability into the services as a core component. It is an extremely high risk and therefore is not typically done. A cheaper way of doing this is providing middleware that can wrap around the service to provide the required interfaces.

This need for interoperability has led to the popularisation of SOA(Service Orientated Architecture) [Sanders et al., 2008]. It should be noted that SOA is *not* an implementation, but rather an *Architectural Model*; SOA refers to a collection of loosely coupled services, that individually carry out a particular process. Each service should have a well defined interface with self-contained functionality. It should allow other applications or services to use this functionality without knowing the underlying technical details. These services should be hidden from the end-user and their usage should preferably be platform-independent.

Although the concept has been around since the 1970s, it has only recently gained favour due to Web services. Web services are software components that run on the Internet through XML standards-based interfaces[Tai et al., 2004]. Each service provides a functional description using the *Web Services Description Language*(WSDL). This description provides the supported operations, as well as the definition of the input and output messages.

By using these concepts, a workflow system can be built that automatically uses these Web Services to facilitate both the data and control flow using well-defined interfaces in standards such as XML/JSON [Shegalov et al., 2001]. With the advancement of WebGIS, a lot of Web Services that facilitate Geomatics processing already exist.

2.4 Case Studies

The next section will look at two instances where workflow management systems were implemented and used. These case studies will look at both a business and a scientific application.

Danske Bank

The workflow management system at *Danske bank* was incrementally implemented as their system moved from a manual system[Brahe and Schmidt, 2007].

This system was developed as an in-house solution when the manual system could not cope any longer. Several lessons were learnt that are applicable to other workflow systems. When work was divided purely from an efficiency point of view, the workers became complacent as they felt that they did not understand the overall mechanism and felt that they were not involved. They discovered that the system did not handle change very well. This change was expensive and inevitable. Their system had to be adapted to handle this change. The success of the system is mainly attributed to the interoperability and close relationship between the users and the developers

OrthoSearch

OrophoSearch is a workflow, built on *Kepler*, that is designed to work on data in the field of Bioinformatics. [da Cruz et al., 2008]

A workflow system was implemented in *Kepler* as it addressed the requirements they had, including: (i) workflow definition and design; (ii) workflow execution control; (iii) fault tolerance; (iv) intermediate data management; and (v) data provenance support.

Although the system was not without its hiccups and changes, the integration with Kepler provided the workflow with increased overall productivity.

Sunfall

Sunfall is a workflow system that was created to assist in locating supernovas from large amount of telescope data[Aragon and Runge, 2009].

Sunfall consists of four components: (i) Search, (ii) Workflow Status Monitor, (iii) Data Forklift and (iv) Supernova Warehouse.

The Search component is responsible for coordinating the tasks responsible for coordinating tasks involved in finding supernovas, within the data. The system is also tasked with dealing with an enormous amount of data, up to 100TB. The data movement is carried out using the *Data Forklift* component.

This project used a Parallel File system, to aid in data replication within the project and used middle ware to interface with legacy software.

Sunfall was deemed a great success as it not only successfully improved the efficiency and identified bottlenecks within the process

2.5 Summary

This chapter reviewed the appropriate literature for Workflow Management Systems and the data variety and processing within Geomatics. This has provided the necessary insight to determine what components would be required in order to build such a Workflow System for the Zamani Project.

The process to create the heritage artifacts from the raw-scans and photographs generates a large amount of varied data. A Workflow Management System would need to be able to specifically cater for this constraint similarly to the large scaled data involved in the implementation of both *OrthoSearch* and *Sunfall*.

Since the tasks are however a mixture between automated and manual tasks. Such a system would be able to map to a more grid-based approach and was shown to be done at *Danske Bank*. Middleware would need to be provided in order for the system to uniformly integrate with applications required throughout the process[Montella et al., 2007]. The model has been shown to be able to be effectively automated using a Workflow Management System[Withana et al., 2010].

Chapter 3

Design

3.1 Introduction

The system was implemented in three design iterations. These iterations are as follows: (i) Feasibility design; (ii) Workflow system design; and (iii) User Interface design. The methodology used was to design, implement and evaluate the system at each iteration. The evaluation at each step was then used as additional input at the following design step. This is illustrated in Figure 3.1.

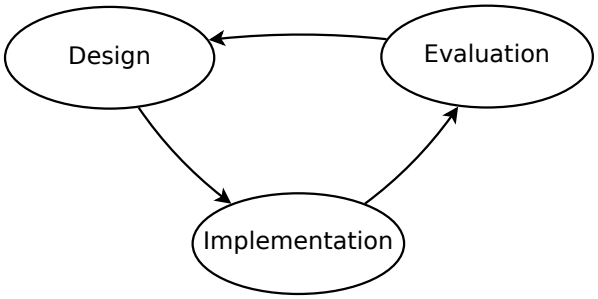


Figure 3.1: Design Process

This chapter describes the design process followed in the building of the workflow system. Section 3.2 describes the items taken into consideration in the initial design. This was then followed by the process used during the first iteration in Section 3.3. Section 3.4 follows with the design and implementation of the first functional product; this product was then evaluated using user a small group of test users. The final design iteration was done by correcting the criticisms that test experiences with the system. In Section 3.5 the final improvements and features are described.

3.2 Design Considerations

This system was developed for the Zamani Project. This requires a thorough understanding of the requirements that exist within the team. This then was then combined with the general requirements of a workflow system.

Workstation configuration

Within the Zamani team, each member has a workstation on campus. These workstations vary in specification but are all locally connected to a high speed local network. These workstations are predominantly *Microsoft Windows 7* based machines. On this network there is also a server present. This server is used as a repository for the sites. Figure 3.2 shows the setup of the

workstations. Files required to process any particular task are typically transferred over the network or via removable drives.

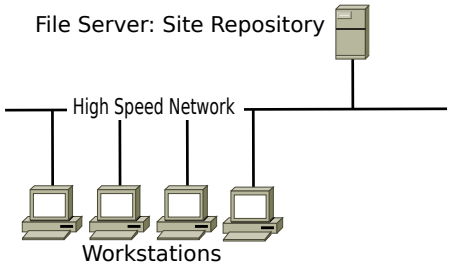


Figure 3.2: Zamani Network Configuration

Task Variety

The team's main tasks involve creating derivative data off of laser scans, photographs and other source data. A large portion of these tasks can be automated, while others require manual processing by the team. As such, the tasks can be divided into two types: user tasks and server tasks.

Dataflow model

The process involves the data constantly moving between the server and workstations with data elements being added at every step. These flows are rather complex and relate closely to the workflow of the project. Given the scale of the data being processed, transferring the data from one end to another has a large cost. As such, this cost should be factored into the design.

Staff Turnover

The project consists of a core number of individuals as well as a group of interns who cycle in and out of the project on a regular basis. This would have an impact at the task level if a user fails to complete a particular task before leaving the project. Due to the high staff turnover rate, this would be a frequent occurrence and would have to be sufficiently dealt with by the system.

Repeatability

In creating this data the same processes are essentially followed on all sites. In the design of the processes these would need to be reusable on different sites with very little effort. This will aid in the easy repeatability of experiments.

Integration with existing software

The Zamani project uses a wide variety of programs during the processing of this data. The workflow system would have to integrate with this software as well.

System Backup

Each site consists of a set of data items. This data is very valuable and is often irreplaceable. In the case of a system failure, the loss of the data would have great costs. As this system would be controlling these items, it would have to allow for the site to be backed up.

Provenance

Provenance refers to the documented history of an object[Moreau et al., 2008]. For digital objects this refers to all the objects, processes and agents involved in the life cycle of the object. Provenance is perceived as a crucial component in workflow systems. This is designed to aid in analysis of systems and help ensure reproducibility. The Open Provenance Model

was designed to represent the capabilities of a system. This the technology to be defined in an agnostic way and allows systems such as this can be built using this model.

This model consists of three primary entities: (i) Artifacts; (ii) Process; and (iii) Agents.These artifacts are joined using relations and dependencies. Figure 3.3 shows the relations between the entities of the model. The benefit of using such a model is that it has been extensively tested in other similar systems with great success.

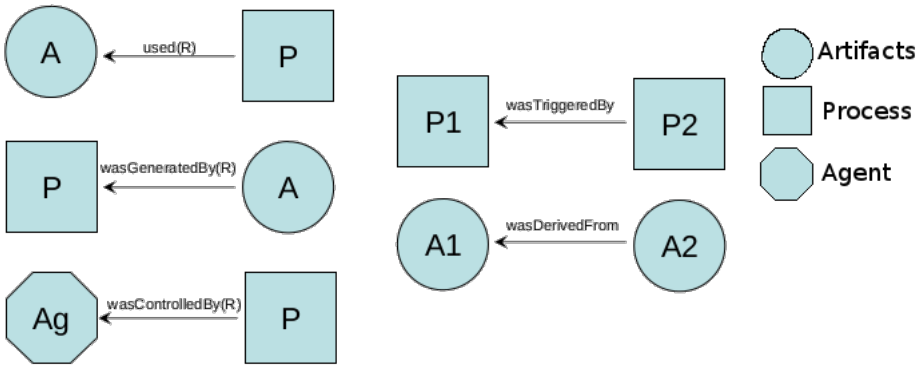


Figure 3.3: Edges in the provenance model

3.3 Iteration 1: Feasibility Demonstration

The focus of this iteration was to ensure that the core types of tasks in the system would be supported. The main tasks were: user-tasks and tasks that can be automated. During this portion, a significant amount of time and effort was spent on comparing different systems with one an other.

This phase produced a small system that supported very primitive manual and automatic tasks. The system was integrated with *Kepler* [1], and could copy over files to a remote host. The tasks could be signaled as being: "Not Done"; "In Progress"; or "Complete".

3.3.1 Design

The initial design focussed on enabling the type of tasks required by the system. These tasks can be abstracted into two main types: (i) Manual User Tasks; and (ii) Automated tasks.

The workflow-management system would run on the same server that hosts the central site repository. This would allow inherent data locality, therefore would allow for efficient server operations on the data. These operations would be performed on data of a given site. Sites can be abstracted to a large hierarchical collection of data items. To aid in the repeatability of experiments, the hierarchy of each of these sites should kept consistent both within the main repository and on the user hosts.

Server Task

The first type of task that would need to be facilitated by the system are those tasks that can be done automatically by the system and require no interaction from a user. These include removing duplicate points in input data, or changing a file to a particular format. To avoid the high cost of transferring data, these tasks are performed on the server itself.

[1]The Kepler Project: www.kepler-project.org

User Task

The second type of task that needs to be facilitated within the system is manual tasks performed by the members of the team. During such a task, particular data elements of the site are required to create new derived data elements. As all the team members cannot work on the server, the required files would need to be moved to the respective workstations. Figure 3.4 shows the data flow for any given User task.

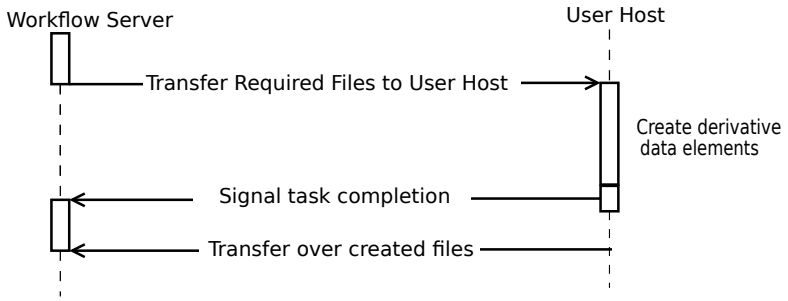


Figure 3.4: Data flow in user task

Basic Task Model

These task types can be abstracted for the workflow system into a general task type. This allows the system to handle them indiscriminately. The basic requirements of a task are to take a set of input data and produce a set of output data; these are mapped to artifacts in the provenance model. This will allow the tasks to be chained together to form a complex workflow. This task maps to the *Process* entity in the provenance model. Each task would be assigned to a user, which maps to an agent in the provenance model.

Kepler integration

Kepler is a popular workflow system, with pre-built models to facilitate complicated workflows. Ideally, the system would need to be able to integrate with Kepler so that it can harness the existing capability of the system.

Site Layout

Within the Zamani Project the data is organised at the root level in terms of the physical site with which the data corresponds. Since the goal is to have the tasks executed on a variety of different workstations, the data management would be greatly simplified if the folder structure is the same on the workstations and the server. For this reason workstations and the server will have an identical directory structure with the site being regarded as the root. This layout could be changed from site to site to allow for more flexibility.

3.3.2 Implementation

The first round of implementation set out to test the feasibility of the core components required within the system. As shown above, this is identified as setting up the framework for the User and Server Tasks. This was done using Kepler, Java and a series of command line tools. This was split into the following components: (i) Database; (ii) Kepler Task Model; (iii) and a Kepler File Copier. These could then be packaged to create complicated workflows that could represent a site. Figure 3.5

shows how these components interact with one another to form the core components of the workflow system.

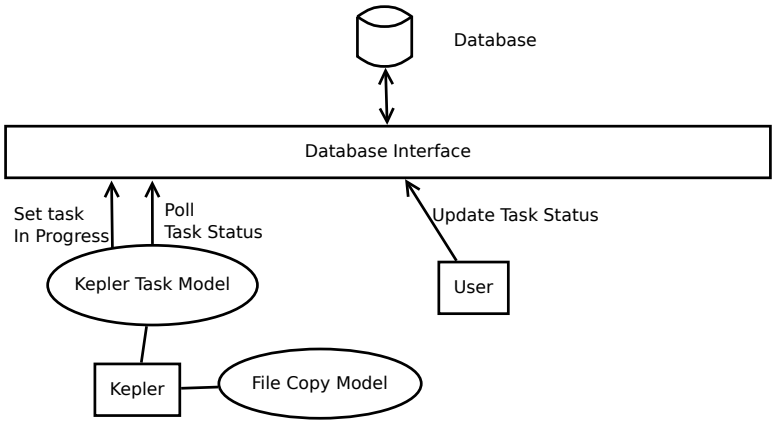


Figure 3.5: Feasibility components

Database

In order to keep track of the tasks, a MySQL database was used. Tasks could be added to the database along with a status. As a task progressed thorough the process its status would be updated, which would trigger events in the system. Initially tasks would have three valid statuses: "NOT DONE", "IN PROGRESS" and "DONE". The database would be updated by small scripts that could be called by Kepler.

Kepler Task Model

This model is used to launch user tasks. The model then polls the database for status changes of the task. As soon as a user completes the task, the database is updated; the model would pick this up and continue with the process. Figure 3.6 shows the workflow in Kepler for the task model.

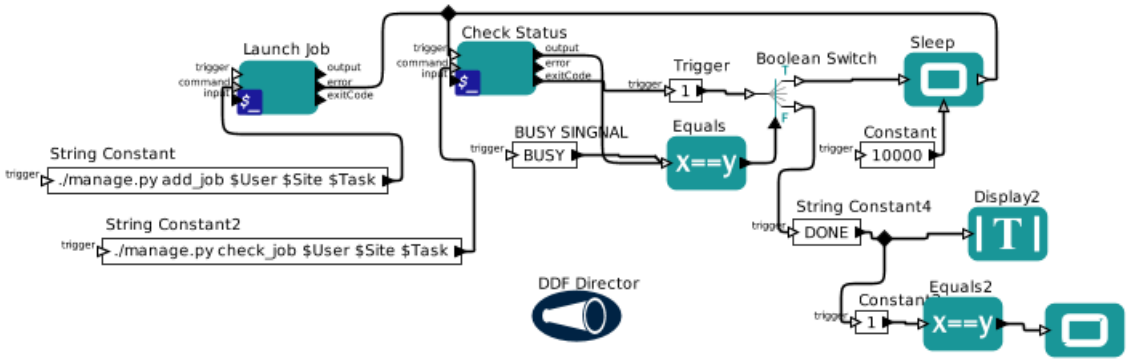


Figure 3.6: Kepler model that launches a user task

Kepler is not ideally suited for user-tasks as its primary focus is complex automation. It is also highly suboptimal as it needs to continuously poll to determine the status of a given task.

Kepler File Copier

User tasks also need files to be copied to the user workstations. For this a Kepler workflow was built that copies files. Each task has a list with the files required to complete the task. The model uses SCP to copy over files to the respective user. Figure 3.7 shows how the model is built in Kepler.

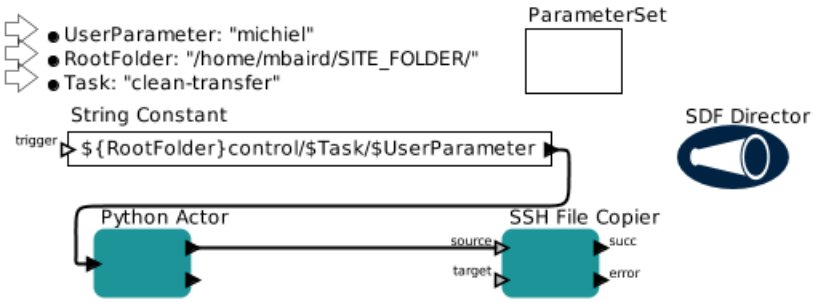


Figure 3.7: Kepler model that copies files to workstations

3.3.3 Evaluation

Due to the lack of functionality of the first prototype, the first phase of evaluation was rather short. This prototype was demonstrated to the second reader (Dr. Bagula). The prototype showed that the core requirements could be implemented and was critical to develop a more complete understanding of the problem.

Dropping Kepler

Kepler is rather bulky and the user task model is ill-suited to the system. The Zamani Project's tasks fit better to an operational model, where user activities are core to the creation of the data, whereas Kepler is suited to a more automated scientific data processing model.

Kepler also does not integrate very well from an interface point of view. As such, it was decided that Kepler would not be used in the next iteration. This requires that its functionality be replicated in the next iteration.

3.4 Iteration 2: Functional Design

This section follows the second design iteration using the first iteration as a base. The goal of this iteration is to create a workflow system that is functionally adequate to support the automation of the workflow for the Zamani Project.

3.4.1 Design

The design of the second iteration followed a more rigorous process and had the primary goal of supporting most of the required functionality. The system should support multiple users, each being able to execute a portion of the workflow assigned to them.

Web Interface

Using a Web interface is well suited to the problem. This will allow multiple users to log in, track and manage their tasks. This could also be hosted on the same server that would host the primary site repository. Preference and prior experience led to the adoption of Django[2] for this purpose. Django is a Python framework that allows for rapid Web application development. Due to the type of iteration cycles, Django is well suited to be used as the supporting framework for the system.

Replication of Kepler functionality

The feasibilty demonstration showed that Kepler has poor support for the user tasks required within the workflow system. This functionality would need to be replicated within the new framework. This requires classification of the Zamani workflow such that it could be correctly implemented. A workflow management system can be classified by four different elements [Yu and Buyya, 2005]:

Workflow Design
: The tasks required within the project are well defined. Data is transformed in a linear manner. Data items do not get iterated over once they are complete. Therefore the system would fit a user defined, directed acyclic graph. This workflow would be designed by the user to allow for arbitrary dependencies.

Workflow Scheduling
: Based on the way that the project is currently executing tasks, the scheduling would need to happen dynamically with a priority to determine the urgency of any particular task.

Fault Tolerance
: Since the system would be dealing with a large amount of data, a lot of which will be generated by the users themselves, the likelihood that tasks may fail is quite high. Therefore it needs to be mediated. Since the workflow models would be manually created by a user, workflow level errors will not be taken into account. The system will, however, have to be able to deal with task level failures. The mediation process would include: the ability to retry a task and modify the node on failure. Both of these would have to be mediated manually by the user.

Data Movement
: New data items are continuously generated and as such the movement of the data needs to be controlled. The goal is to automate the data movement, using a centralised repository that maintains the authority on data.

Django is built using Python and uses a *Model View Controller* architecture[Leff and Rayfield, 2001]. This model separates the system into three parts: the data, the control logic and the presentation of the data. The MVC framework would allow for the workflow logic to be built and executed independently of the User representation of it.

Participatory Design Session

To ensure that the product meets the needs of the client, they were made active participants in the design process. The methodology used during this is Participatory design [Muller and Kuhn, 1993]. Due to limited time, this was selectively applied. Four members of the Zamani Project were brought together for a PICTIVE design session [Muller, 1991]. The method used can be outlined as follows: (i) General Requirements gathering; (ii) defining the task flows; and (iii) a graphical user interface PICTIVE design session.

The session identified the core tasks that need to be supported. These include the following:

[2]Django Web Framework www.djangoproject.com

- Workflow Setup
- Template Functionality
- Progress Indicators
- Task time information
- Task control from a user perspective
- Logging Functionality

These tasks were then further developed to create a logical use-case. The most important functionality defined was as follows:

Task Definition

The focus was on what users would need regarding tasks. The suggestion was made that all tasks be separated at site level. Each task should have the following properties:(i) Name; (ii) Priority; (iii) Category; (iv) Input Files; (v) Status; (vi) Description; and (vii) an Output Directory. Users should see an overview of their tasks, and on demand be able to obtain further details on a particular task. Tasks would be assigned to users. User Tasks performed by unprivileged users should also be checked by privileged users before being marked as complete. Users should be able to re-initiate downloads of the required files to their hosts.

Task Setup

In order to set up the workflow, an administrative interface is required. This should allow for the creation of tasks. Each task would have a set of dependencies, linking to other tasks that need to be completed before the task can start. It would also require a graphical overview of the site's task. From this view users should be able to add and edit tasks, having access to all the parameters of the task. Each task would have a separate log, allowing users to determine what the causes of failures are. Sites should have an initial set of tasks populated using a previous site as a base. This would allow workflows to be reusable, which is preferable as each site follows roughly the same process. The workflow for the derived site can then be modified independently to allow for site-specific conditions.

The final task of the session was to create mock user interfaces for the identified tasks. This was using low-fidelity items, which included small pieces of paper that represented the components that need to be added. This activity was performed with a small group of users who are involved with the Zamani Project. Three screens were set up.

Figure 3.8 contains the controls for each site. The main component on this screen is the visualisation of the workflow. Below this is a table that contains a list of all the tasks in the site. Controls to add tasks and categories were also added. The tasks to edit and add tasks should only be available to privileged users. The second screen developed is aimed to give an overview of the tasks. Figure 3.9 is separated into two sections. The top section contains a list of the tasks that are outstanding. The bottom portion contains a list of tasks that are currently in progress by the rest of the team. An additional feature is a progress bar that indicates the progress of the completed tasks. The last screen that was developed during the session was the task control screen. Figure 3.10 gives a detailed description of a task. It should allow users to indicate their task's completion. With the correct permissions, users should be able to validate completed tasks of general users within the system.

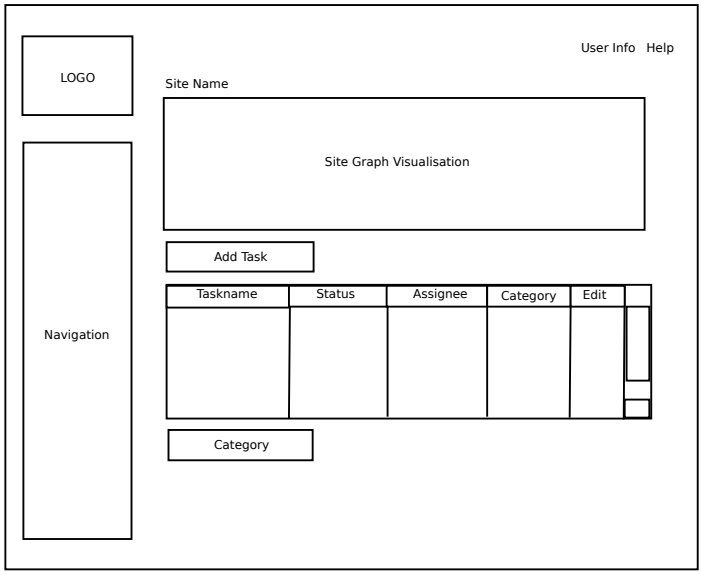


Figure 3.8: Site screen developed during the PICTIVE session

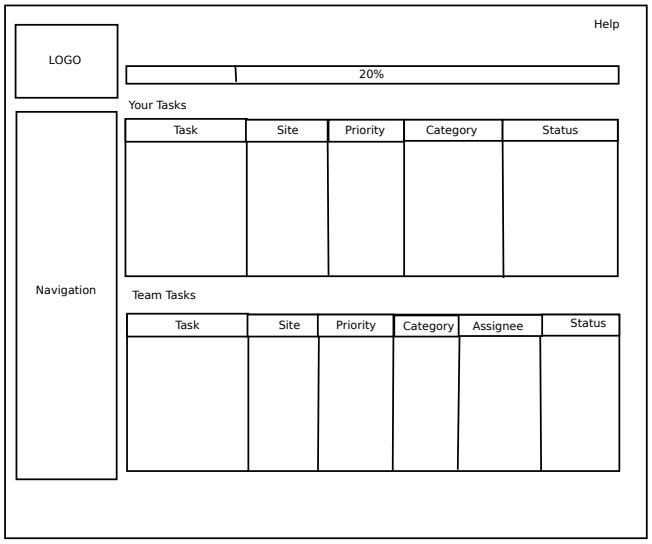


Figure 3.9: Task Overview screen developed during the PICTIVE session

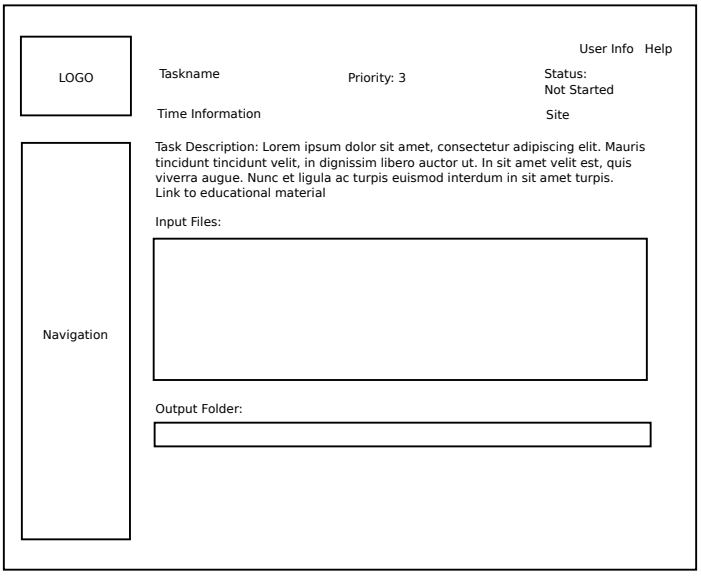


Figure 3.10: Task Control screen developed during the PICTIVE session

Database Schema

The system needs to manage a large amount of data. This requires a database that will control all the information required for the workflow system to function. The primary tables required for this are: (i) Sites; (ii) Jobs; (iii) Hosts; (iv) Files; and (v) Tasks.
Each of these tables is described below:

Site	
Name	Type
Id	VARCHAR(30) PK
Name	STRING
Folder Name	STRING
Active	BOOLEAN

Figure 3.11: Site Schema

Site: This table stores the root information for the various sites that the project is working on. The "Id" field is used for short *URLs*, while "Name" is used for display puposes. "Folder Name" provides the root folder that the site data will be stored in. This allows for consistency between the repository and the client workstations. As soon as a site is completed, it will also have the option of being marked inactive. This will prevent the clutter of old sites that no longer require workflow to be managed.

Job	
Name	Type
Id	AUTOINT PK
Name	STRING
Type	CHOICE
Script	PATH
Description	TEXTFIELD

Figure 3.12: Job Schema

Job: This table was created to improve the reusability of tasks. The purpose of a given task will be determined by the *Job* that the task is linked to. This table supports three main type of jobs, namely: (i) User Jobs; (ii) Server Job: one-to-one; and (iii) Server Job: many-to-one.This is represented by the type field. For server tasks, the *Script* field is set to a path that contains the server-side script that will execute on the input files defined within a *task*. For one-to-one tasks the script is executed once per input file, while it is executed only once for many-to-one jobs. Information about the job is stored in the *description* field. The rest of the fields are used to facilitate naming and queries.

Host	
Name	Type
User	FOREIGNKEY(User)
Host Name	STRING
Root Directory	PATH
Primary	BOOLEAN

Figure 3.13: Host Schema

Host: This table manages the control of a user's workstation. The *user* field points to the user attached to the particular host. The *hostname* contains the connection string that connects to the user's workstation. Since every user is likely to have a different folder structure on their workstations, a *root directory* field was added to indicate a path on the workstation that will be used as the root folder for the sites. It is very likely that a user could have more than one workstation, therefore a *primary* field was added to indicate which host would be the default for that user.

File	
Name	Type
Filename	PATH
Site	FOREIGNKEY(SITE)

Figure 3.14: File Schema

File: Each file will have a database entry. This will add the ability of a quick file lookup and aid in maintaining files between the repository and the workstations. Each entry will have a *filename*: this will store a relative path from the site root to the file name. By only storing the relative path we are able to access the file without making assumptions on where the root directory is. The second field links a file to the site that it belongs to.

Task: This controls how the workflow is put together in the system. Each task is defined as an independent activity that can be executed by the server or by a user. Each task has a list of dependencies that need to be satisfied in order for the task to be executed. When a task is complete, its successors are then checked, and run if all the dependencies are met. In order to ensure that this can be done efficiently, both the predecessors and the successors are stored in the table. The *Id* field is used for task identification in URLs. Each task also has time information. The execution time of

a

Task

Name	Type
Id	AUTOINT PK
Priority	INTEGER
Started	DATETIME
Ended	DATETIME
Log	TEXTFIELD
Site	FK(Site)
Job	FK(Job)
Assignee	FK(User)
Input Files	MTM(File)
OuputFolder	PATH
Predecessors	MTM(Task)
Successors	MTM(Task)
Job Status	CHOICE FIELD

Figure 3.15: Task Schema

task can be calculated by subtracting the start time from the end time. Each task is based on a *Job*: this allows jobs to be reused in various tasks, making the system easier to use. A task is explicitly assigned to a user; for automatic tasks a dummy user named "server" is used to ensure database consistency. The input-files of a task is defined in a "Many-to-Many"[3] field; this enables the system to only transfer required files to workstations as well as running the script on appropriate data. When a task is executing, the system expects the output files to be placed in the output folder field, this allows the system to manage the file dependencies of the successor tasks.

Along with the system being failure-tolerant, it also needs to be able to determine what has happened when a failure has occurred. For this reason a *Log* field was added. This field stored all the activity that a task performs. This includes the transfer logs as well as any output generated during the execution of the server scripts. Tasks are also stateful; this state is represented by the *Job Status* field. The possible states for a task are: (i) NOT DONE; (ii) TRANSFERRING; (iii) IN PROGRESS; (iv) TRANSFERRING BACK; (v) VALIDATION REQUIRED; (vi) DONE; and (vii) FAILED.

Features

The combination of the interface, the and the database provides the required components to support the features of the system. Below are a list of features that were implement during this design iteration.

File Transfer

User tasks are able to transfer files to the user workstations, from the input file list. As soon as a user task is triggered, the file list is retrieved from the task entry. The server then connects to the host and copies over all the required files. Since a user can have more than one host, this transfer can also be manually triggered by the user. When a task is then completed, the server will again initialise the file transfer but in the opposite direction. This will upload all files that are present in the *Output Directory*. In order to support switching of workstations, this process can also be triggered at any stage, before the task is completed. Extra functionality is also added to download all the files in the output directory located on the server.

Task Automation

The system's ability to facilitate workflows is entirely dependent on its ability to automate tasks. This is done by adding a control system within the server that works out which tasks have no unmet dependencies. As soon as these tasks have been determined, an asynchronous task is launched to run the task. Depending on the type of job, either files are transferred or a script is run. When a task is completed, either by verification or the script finishing, the task is finalised. This process involves adding all the generated files to the database, updating the input files of the successors. If all the dependencies of a successor are met, that task is then started. When a task fails, however, a user is able to make the required changes and restart the process.

[3]MTM refers to Many-To-Many Field

Logging

Logging is added to help with identifying and controlling task failures. Within the automation framework all events happening with the task are added to an append-only log field. This includes the transfer logs for user tasks, and both *stdout* and *stderr*.

Site Visualisation

The tasks for a site can be represented as a *Directed-Acyclic-Graph*. To make the system more usable, the tasks will be represented as a graph within the interface. This is done by calculating the graph of the sites and then applying a layout algorithm to it. For the purposes of this visualisation, a spring layout is used. The nodes are then joined by edges. A summary of each task is displayed on the node.

Site Setup

Privileged users should be able to compose workflows with relative ease. Options are added to allow users to create new *Jobs* and *Tasks*. Once a task is added, dependencies can then be added in a visual way on the visualisation. Workflows between sites are very similar; creating a new workflow for each site would be very time consuming. An additional feature is added that copies over the structure of an existing site. Once the new site has been built, a user would just be required to select the input files and assign tasks to users.

3.4.2 Implementation

The system was implemented using the *Django framework*. This was done in three parts: (i) database setup; (ii) workflow framework; and (iii) user interface.

Database

The *database* was set up using the model subsystem in *Django*, from the chosen schema. This is a system that defines the database using Python; this allows the schema to be implemented without needing to worry about the database system being used. For the purpose of this iteration, *SQLite*[4] was used to support the database. This can, however, easily be changed to a more robust system as the demand on the system grows. User data was implemented using the authentication module in Django. This module manages users and permissions on the system. As the schema was discussed in Section 3.4.1, it will not be discussed further.

Workflow Framework

The underlying workflow system is implemented using *Python*[5], within the Django framework. Tasks are executed asynchronously. Each job type is handled separately.

Server Job - One to One

This job type executed a given script on all the input files. The system generates a list of absolute paths and executes the script given two parameters: the input file, and the output directory. This is done for all the input files of the task.

Server Job - Many to One

This job type executes nearly identically to the *One to One* task; however, instead of the script being run run multiple times, it is run once. Here, however, all input files are passed in as arguments, along with the output directory. Both server jobs can use any script or applications available on the system. With applications where the parameter structure does not match the required utility, wrapper scripts can be used.

[4]SQLite: www.sqlite.org
[5]Python: www.python.org

User Job

This type of job is split into multiple parts. Since there is a distinct separation between the server and the execution type task, the goal is to ensure the effective communication between workstations and servers. The goal is to be able to transfer large amounts of files with minimal redundancy. For this reason *rsync*[6] is used. This only transfers files that have not been transfered yet. Redundancy is greatly reduced, and this enables efficient recovery in case of network failure.

Figure 3.16 shows the components used to execute a user task. Each of these components are built using an asynchronous object. Note that the transfer components can be used at any point during the task's life cycle.

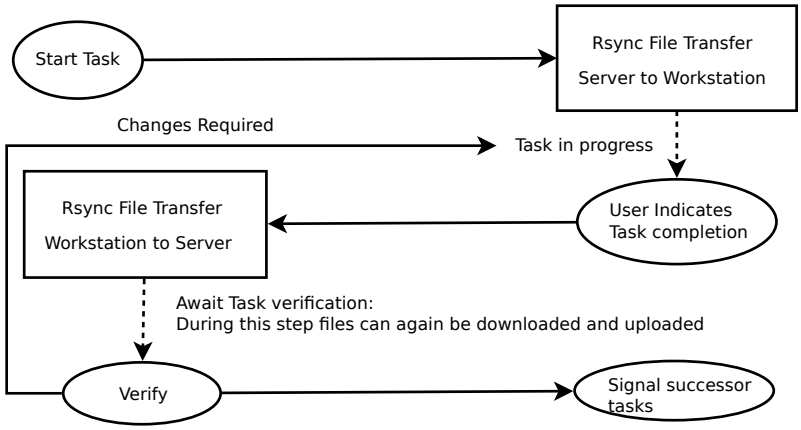


Figure 3.16: Components that support user jobs

Once a task is finished it passes on to finalisation. All tasks follow this model. It is assumed that all the appropriate files are present on the server at this point and that the task has completed successfully. At this the output directory is scanned for files recursively and new files are indexed and added to the database. The new files are then added to the input files of all successor tasks. If all the dependencies of a successor task are met the task is then launched.

User Interface

The interface was implemented using *HTML* served using the View subsystem in Django. These views aim to create a usable system. These are based on the designs that were created during the *PICTIVE* session. Slight alterations were made to the design to support necessary functionality that was not considered during the session. The interface comprises of several views.

Task Overview

This view shows a user which tasks are outstanding, as well as what is happening within the team.(See Figure 3.17) This also provides links to the task control page for each task. An additional feature that was implemented allows users to filter tasks based on a choice of sites.

Task Control

This is the view that non-privileged users will be interacting with most often.(See Figure 3.18) It provides a detailed overview of what is required to be done with the task. Privileged users also get a link to edit the task. This page also contains tools to allow users to download and upload required files. Once a user is finished with the task at hand, he/she can simply click on the *Finish Task* button. This triggers the transfer back routine and queues the task

[6]Rsync Incremental file transfer: rsync.samba.org

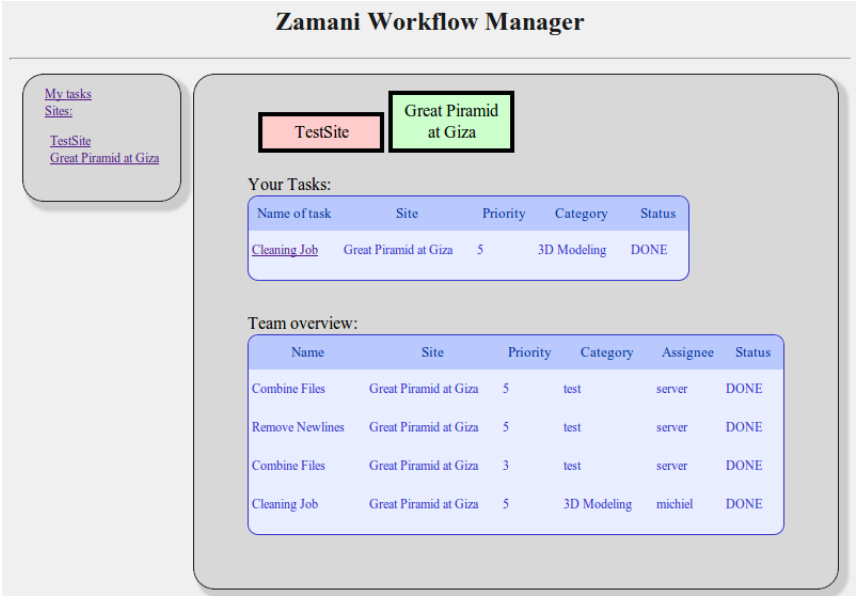


Figure 3.17: First implementation of the task overview screen

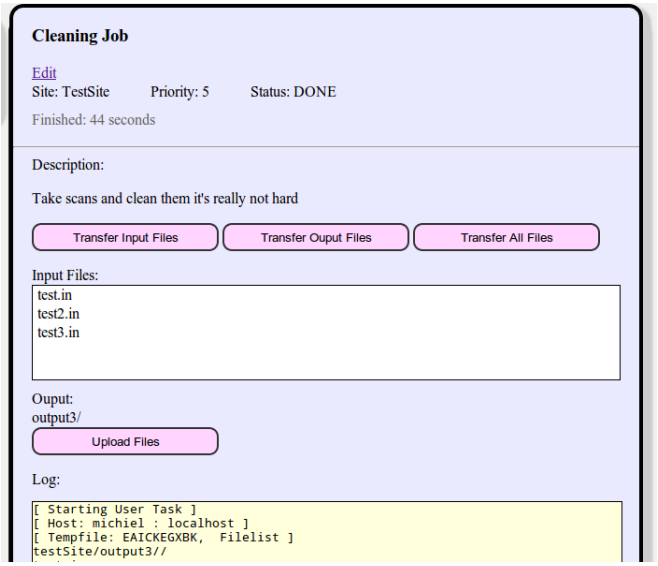


Figure 3.18: Task Control Implementation

for verification. Users who are able to verify the task will use the same view, however the *Finish Task* button is replaced with two buttons that will accept or reject the task. Logging information is also made visible to ensure that the complete task history is visible.

This view is also used for server tasks. In this case the emphasis is on failure control. With the log visible, users can trace the source of the failure and rerun the task as soon as it has been dealt with.

Task Editing

During the initial task setup, the parameters of a task need to be set. This screen enables the user to set who the task is assigned to, what the input files are and what the task depends on. The dependencies can also be set the task interface. When something goes wrong with a task, privileged users are also able to edit these setting when the workflow is already active. The workflow should be able to dynamically adapt in this case.(See Figure 3.19)

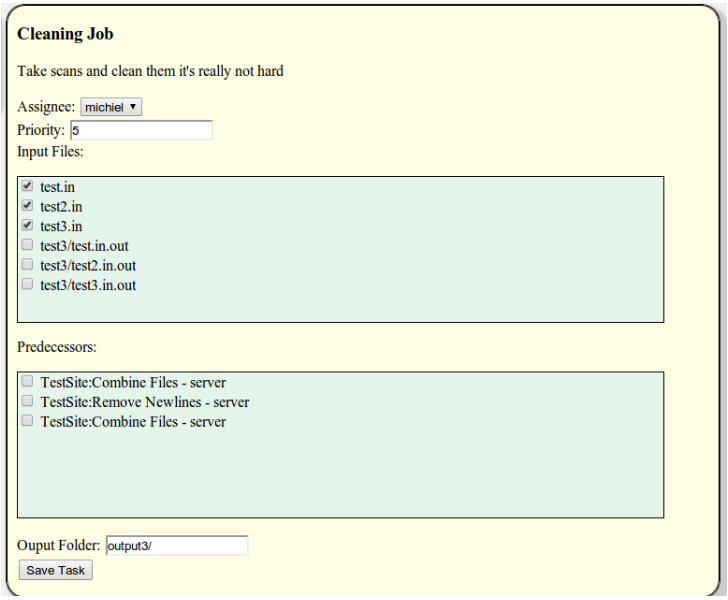


Figure 3.19: Task Edit Implementation

Site View

This view is dominated by the visualisation of the site.(See Figure 3.20) The visualisation was implemented using *jsPlumb*[7]. This is a library that builds interactive graphs. Each node provides an overview of the task. Dependencies are represented with edges while the colour of the node indicates the status of task. Red indicates that there is a problem with the task,

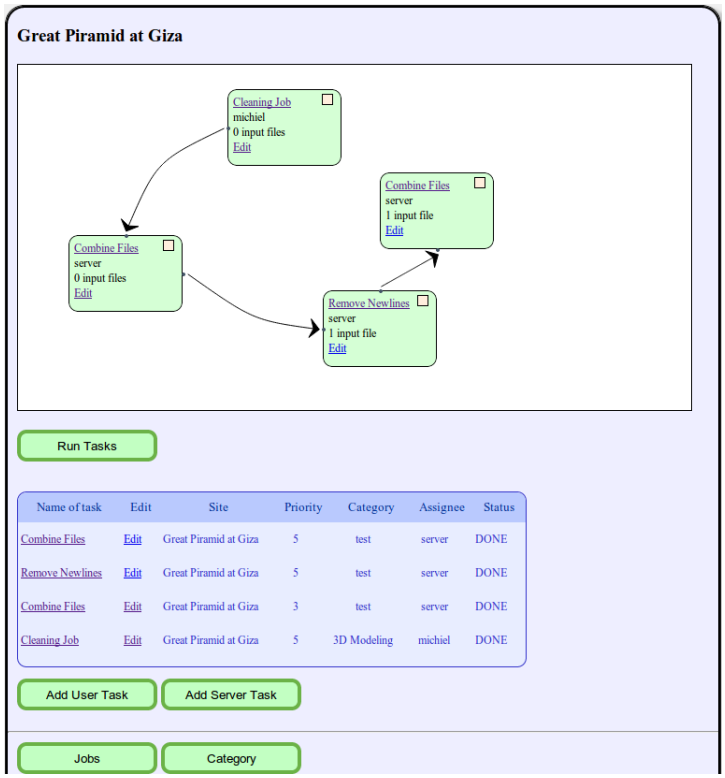


Figure 3.20: Site View Implementation

[7]jsPlumb Graph UI: jsPlumb.org

grey indicates that the task has not been run, blue indicates the task is in progress and green indicates that the task has been successfully completed. Dependencies for each task can be added by clicking and dragging the arrow in the corner of the node to a successor node.

Below the visualisation is a list of the tasks for the site. This gives an overview similar to the task overview screen. A privileged user can select tasks off this list, which will then forward to the task editing screen. As soon as the tasks in a site have been set up, the workflow can be started from this view.

The final set of components is there to add/edit jobs and categories in the system. These are not specifically linked to a site and can be used throughout the system, allowing for easy reuse.

In addition to *jsPlumb*, *jQuery* and *jQuery UI* is used extensively to build the client side interface.

3.4.3 Evaluation

To evaluate the second iteration, the system was given to five Computer Science honours students. They were asked to critically evaluate the system. The purpose was to highlight any major issues.

The issues found are as follows:

Aesthetic Appeal
The apparent usability is impaired by the lack of aesthetic appeal, which distracts from the task at hand[Tractinsky, 1997].

Lack of logical Flow
Some controls seem to be out of place, making their purpose hard to figure out; this would require some reordering.

Site Filter is confusing
The site filter appears like a button and its purpose is misleading.

Technical Namings
The use of technical language makes the system unsuited to a user without a Computer Science background. Naming should be changed to include a wider audience.

Cycles aren't being checked
The system uses a *Directed Acyclic Graph* to represent the workflow. However, when the workflow is updated cycle checking is not done. This could lead a workflow to enter an unsatisfiable state.

Allow User Specified ordering
It was noted that users cannot order tasks. This could make a specific task very difficult to find when the list becomes increasingly large.

File Picker is too simple
The file picker is severely lacking in functionality and would be aided by using a more standard tree view.

3.5 Iteration 3: Final Design

This iteration builds on Section 3.4 and addresses the issues raised during the evaluation phase. This iterations aims to improve usability issues encountered in the previous iteration. The system was then evaluated for user experience by performing a user experience study with several users.

3.5.1 Design

The system was altered to include a number of changes that improve the usability of the system. A templated stylesheet was used to improve the aesthetics of the system. Some components were reordered to improve the logical coherency and layout of the interface.

The task overview interface was reworked and the site filter was removed as it caused too much confusion. Sorting controls were added to the task-list, allowing tasks to be located easier. The user facing terminology was changed such that the terms are more general and understandable to users without background in technology.

The last focus of the design was to avoid unintentional user errors. Cycle checking was added during the setup of the tasks, this avoids unintentional user errors and ensures that the system remains in a consistent and executable state. The *File Picker* was changed to a more intuitive component. This further improves the usability of the system.

3.5.2 Implementation

During this phase several features were improved upon from the previous iteration. These changes were all made on the client side of the application and the focus was to increase the usability and make the system more understandable.

The most noticeable change of the system is that the style-sheet was changed. The theme that was chosen is called *Blue Nile Admin* and was designed by *Portnine* [8]. This is a two panel design that roughly matches the original design. This theme also includes a range of icons, which, overall makes the system more appealing(See Figure 3.21).

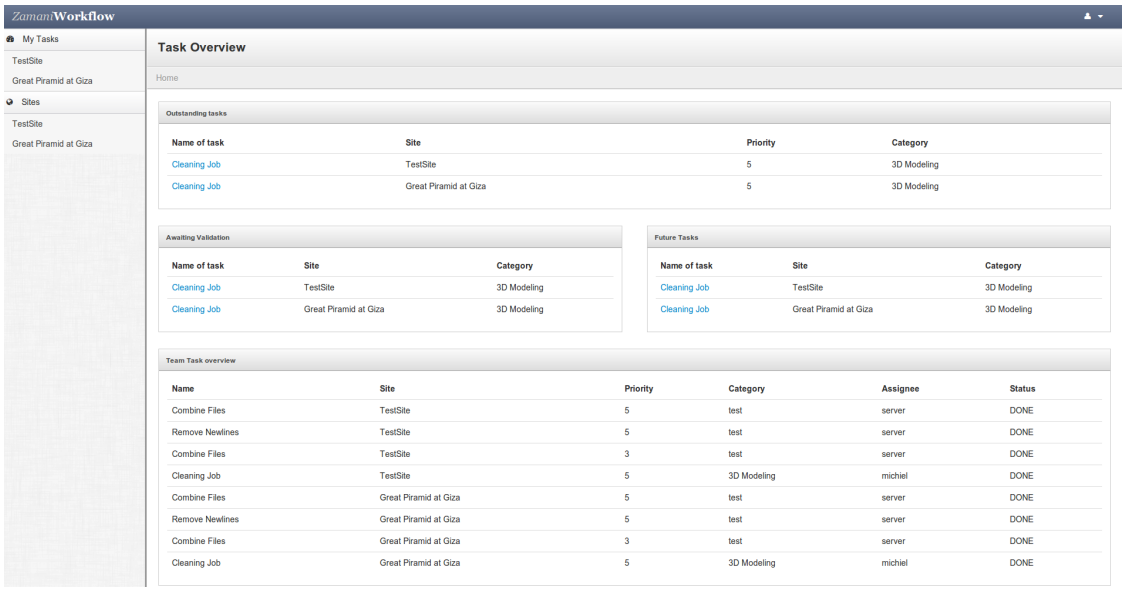


Figure 3.21: Final Implementation: Task Overview

The task overview screen was altered to make tasks more clear. Figure 3.21 shows the implementation of this. Instead of using a two table design, this esxpanded to using separate for tables tasks that have different statuses. Since *outstanding tasks* are the main points of action for users, these tasks are placed in a large table on top. Task that are *awaiting validation* are placed in a smaller table below: privileged users will also have tasks from other users appear here in order to easily see which tasks needs validation. Tables for *tasks with unmet dependencies* and *completed tasks* are

[8]Portnine: http://www.portnine.com/bootstrap-themes

also included. The team overview task has been filtered to only include tasks that are currently in progress. All tables are sortable using *JQuery: TableSorter*[9]. Tasks can also be filtered by site.

The file interface was changed to be more appealing and usable. *JQuery Widgets* [10] was used for this purpose, specifically the *JqxTree* widget. This allows users to easily select and view files in a hierarchical manner. Figure 3.22 shows how files can be viewed and how files can also be selected. The selection and view interfaces are similar to be consistent throughout the system.

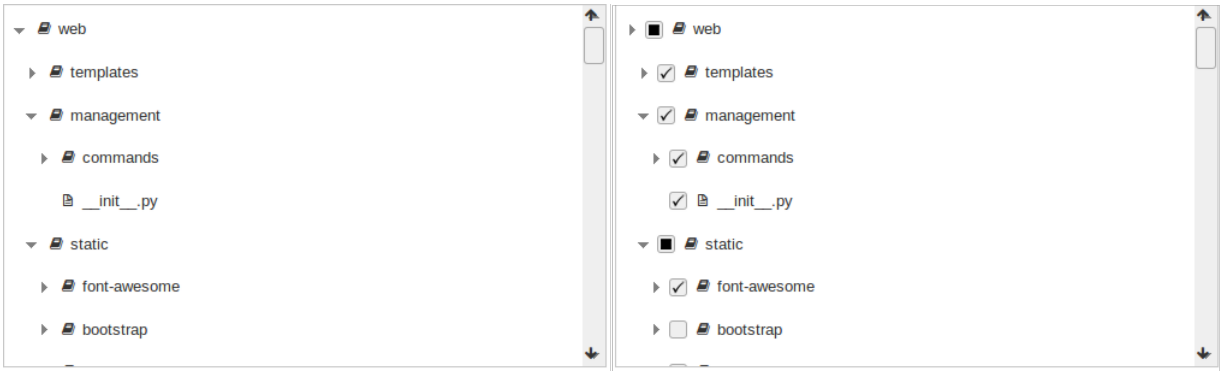


Figure 3.22: Final Implementation: File interface

The task visualisation was changed to be visible on both the site interface, as well as the edit interface. This allows for the removal of the unintuitive predecessor selector that was present in Section 3.4.2. In the task edit interface, a specific task is selected. To indicate this the selected task will be coloured in such a way that its stands out. The visualisation was also changed such that task positions are static and persistent. This allows for the tasks to be arranged by users. Figure 3.23 shows how the visualisation interface was changed.

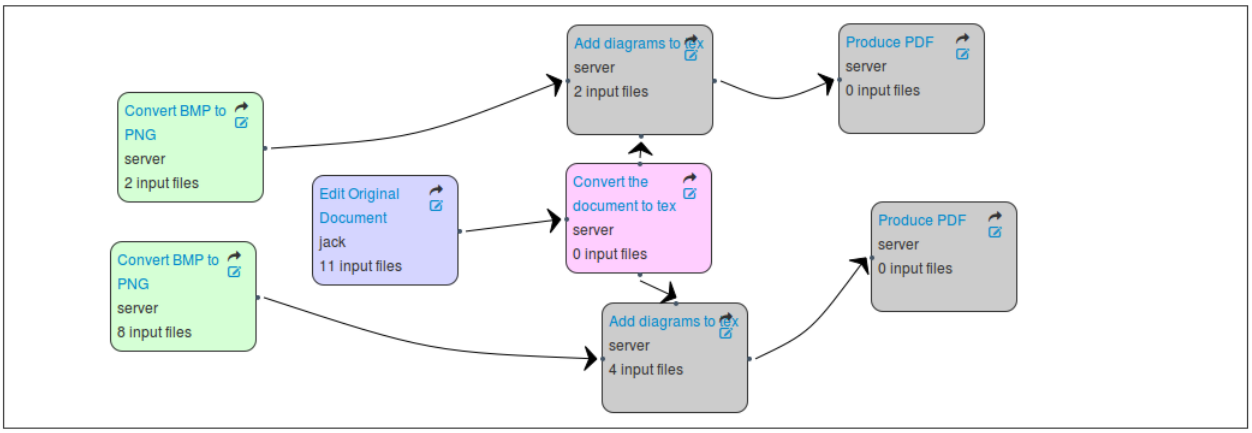


Figure 3.23: Final Implementation: Site visualisation

[9]Tablesorter: tablesorter.com
[10]JQWidgets http://www.jqwidgets.com/

3.6 Summary

This chapter reported on the over the design of Zamani Workflow. This was done in three design iterations. That covered a Feasibility Demonstration, a *Functional Design* and a *Final Design*. This was developed from the requirements of the client. The features of the final system are as follows:

- User Management
- Automated Server Tasks
- Remote User Tasks
- Web interface
- Task Validation
- Error Recovery
- Task Logging
- Visual Workflow Setup
- Workflow execution

Chapter 4

Evaluation

For the system to be successfully utilized within the project it needs to accomplish the following goals:

1. It needs to be able to integrate the daily tasks of the users without it being a burden.
2. It needs to effectively manage the data items and coordinate the tasks between the users and the system.

In order to answer these questions, the system needed to be evaluated on two fronts. *User Experience* evaluations were performed to test whether the system could be easily adopted by users if implemented.

The system was then tested further to see if it could perform a subset of the tasks it would be required to perform if implemented within the *Zamani Project*. Due to time constraints, the entire set of tasks required to support the activities of the Zamani Project could not be implemented.

4.1 User Experience Evaluation

Users and their experience of a system is becoming a critical component in designing a software system[Forlizzi and Battarbee, 2004].

User Experience evaluations aim to understand the needs and experiences of a user. In order to do this, user engagement needs to be tested on both a visual and emotional level. These evaluations are used to link the needs of the users to the functionality provided by the system. The benefit gained from having a usable system is that the productivity of the users is greatly increased[Nielsen, 2003]. User Experience experiments were performed to ensure that the system provides a good, usable interface.

4.1.1 Aim

The aim of the usability test is to assess the quality of the system. This assesses the user's ability to complete required tasks efficiently while remaining satisfied with the system[Bevan, 1995]. This is done in order to determine how suitable the solution is for the user.

The usability of the system was be rated using the following attributes [Zhang and Adipat, 2005]:

Learnability Relates to the capability of the software to enable a user to learn how to use it. This is a goal for the system as this would enable users to quickly start using the software effectively.

Efficiency Refers to the time it takes for the user to complete a certain goal or task.

Satisfaction Measures the attitude of the users towards software. This includes: (i) Difficulty;(ii) Confidence; and (iii) Like/Dislike towards the system.

Error Number of errors that a user makes, including deviations from the intended path.

Effectiveness This tests user efficiency based on a predetermined level in terms of speed, number of errors and steps.

Simplicity Amount of effort that is required for a user to complete a task. This can be traced in terms of number of selections or time taken to search for a function.

To measure the user experience, a simple performance evaluation is not enough. We need to be able to gain insight on how users feel about the system [Vermeeren et al., 2010]. This requires that the emotional state of the user be evaluated.

4.1.2 Methodology

In order to determine the attributes mentioned above, a quantitative *User Experience* experiment was set up. Users were asked to complete two task using the system. The system was designed to support a workflow. This involves two main user activities: (i) Management of individual tasks; and (ii) Setting up workflows . The tasks were designed to test these operations. Their experiences were recorded and then evaluated. The detailed process is outlined below.

Task Setup

Tasks were set up for users to complete. The aim of these tasks was to simulate the main activities of the system. This led to the formalisation of two tasks that are described as follows:

Complete a set of tasks as a user

The first part was to simulate the actions of an unprivileged user who is required to do outstanding tasks. A Site was created containing three tasks. These tasks were assigned to the test user. Since the system aims to support tasks that are executed on the user's workstation, the tasks were designed to be as simple as possible. Since the general usability of the system was to be evaluated, the tasks were not related to processing digital cultural artifacts.

Three *Python* programs were set up to represent the user tasks that were to be executed. For *Task one* and *Task two*, the user was simply required to run the desktop application. The third task, however, aims to test the user's understanding of the system by asking the user questions about the task. These questions are: (a) What is the output directory for the task?; and (b) Please select the input files that are used in this task? The task descriptions all convey what the purposes of the tasks are. As soon as the user completes a task, they need to indicate on the system that the task has been completed.

Build a simple workflow

For the second part the user had to simulate the role of a privileged user, and had to set up a sample workflow. The tasks represent a workflow that produces a PDF file starting with a text file. This was given as a diagram to the users. The diagram can be seen in Figure 4.1. The user was given very little instruction on how to complete the task. This allows them to explore the system and intuitively complete the task.

To ensure that each user experienced the system the same way, it was restored to its previous state after each test.

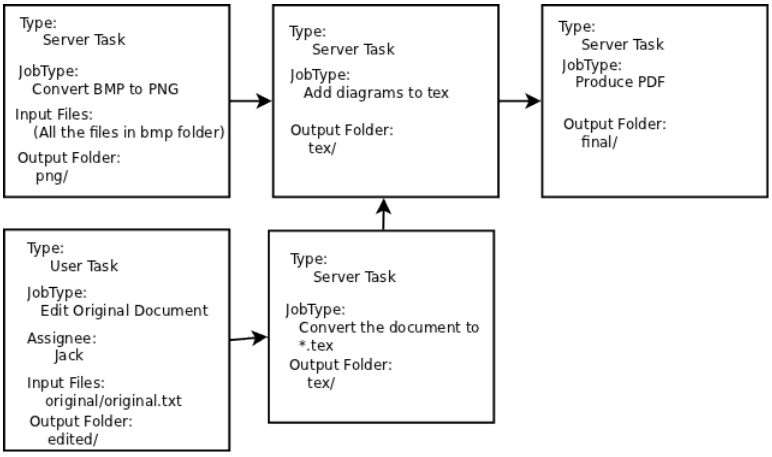


Figure 4.1: Workflow that needed to be recreated

Questionnaire

The primary method used to evaluate the *User Experience* of the system was using a questionnaire. Questionnaires attempt to obtain the subjective feelings of the user towards the system[Chin et al., 1988]. In order to measure the experience, the emotional response of the user also has to be obtained. The USE questionnaire was chosen to evaluate the User Experience [Lund, 2001]. This questionnaire is designed to determine the usability attributes in terms of: (i) Usefulness;(ii) Ease of Use; (iii) Ease of learning; and (iv) Satisfaction . It is designed in such a way that an emotional response is triggered. In order to avoid *Acquiescence Response Bias*, the questionnaire duplicates responses in such a way that the they were phrased both negatively and positively. In order to save on paper, the survey was conducted using *Lime Survey*[1], The questionnaire used can be found in Appendix A.

Monitoring and Pilot Tests

During the tests the users were monitored and their actions noted. This was to determine what the overall process was for each of the users. All questions, actions and errors were also noted during this process. These events were noted by time to produce the remaining usability attributes namely: (i) Efficiency; (ii) Error; (iii) Effectiveness; and (iv) Simplicity

After the tasks were set up, the user tests were run using two users. This was used to determine problems that would adversely affect the results before it was run using a larger test group. The results of the pilot test identified some minor issues regarding how the questions were worded, that distracted the users from the task at hand. This was fixed before the full tests were conducted.

User Selection

The participants chosen for the user experiment were not the users that would be interacting with the system on a daily basis. This is to avoid confirmation bias that is likely to occur due to the fact that they are stakeholders in the project[Kaptchuk, 2003].

The participants varied in age ranging from 18 to 24. Due to cost and time constraints all subjects tested were students. The level of technical competence varied from novice to expert. The students were highly diverse in terms of *Field of Study*, being drawn from four faculties: Economics, Engineering, Science, and Humanities. 24 participants were used to perform the evaluation.

[1]Lime Survey: http://www.limesurvey.org/

4.1.3 Results

The user evaluation produced a number of results. Users were on average able to finish the task 21 minutes, with an interquartile width of 7 minutes. This indicates that the users were mostly able to complete the task in the expected time. All but one of the users were able to complete the entire set of tasks that were provided. Results of the test were acquired using two methods: the survey; and the data from the observations.

The survey independently measured data for both tasks that the participants were required to execute. This aimed to get the users' opinions on the categories regarding: (i) Usefulness; (ii) Ease of use; (iii) Ease of Learning; and (iv) Satisfaction . Values with a high amount of variance were not considered to be significant.

Execute Simple Tasks

The participants were asked to evaluate their experiences with regard to completing the tasks. The survey, along with the observations was used to rate the experience in terms of the following categories:

Usefulness

Overall, the users found that the system was useful and allowed them to be productive. There is a general indication that the system was well equipped to support the tasks that were executed by the participants. 76% of participants found the system useful.

From observation it seemed that users were easily able to use the system to indicate the tasks. There appeared to be very little confusion as to what the purpose of the buttons on the interface was.

There was a slight delay, for about 30% of users, between the completion of the first task and when the users started the second task since the task screen displayed either that *data was being transferred back* or that the task was *awaiting validation*. They seemed to wait for the system to indicate that the status has changed, without realising that the validation process was manual. This reduced the initial usefulness of the system, prompting that more interactive task reporting is possibly required.

Ease of use

The number of steps required to complete the task was considered to be minimal by 80% of the users, and very few inconsistencies were noted. 71% of participants agreed that the system was easy and simple to use. The system was described as flexible and users were able to recover from mistakes easily.

It should be noted that participants initially found the concept of specifying *input files* and the *output folder* while completing *Task* 3 confusing. 50% of users had to be told what the task was asking. From the user responses, this can likely be attributed to phrasing of *Task* 3 within the system.

Ease of Learning

The system was described as being very easy to learn by 90% of participants. They found that the system was very easy to remember. 75% of users described themselves as somewhat skilful after using it for a short period of time.

This corresponds with the observations made during the test. Since participants were given no instructions on how to use the system, they were initially quite hesitant to perform actions. However, as the experiment moved along, their actions became much faster and more decisive.

Satisfaction

85% of participants were satisfied with the experience with the system and that it allowed them to complete the tasks with relative ease. Users emotionally responded to the system in the following way: (i) it was pleasant to use; and (ii) that the system was wonderful.

Some users showed clear signs of satisfaction when they started getting used to the interface. At no point did any of the participants appear frustrated while completing the first part of the experiment.

Efficiency

The efficiency of the survey was not measured by the survey and all findings are based on the observations of the users. Users took an average time of 7 minutes in completing the first task. The slowest time taken was 12 minutes. The interquartile range was determined to be between 5 minutes and 8 minutes, showing that most of the users were able to efficiently accomplish the tasks provided.

Some efficiency issues were noted in the hesitance of participants after completing the first task, before starting the second task. This could be attributed to the participants being unfamiliar with the system; however it is an indication that better task feedback would likely increase efficiency.

Overall, the task interface yielded positive results on the *User Experience* of the system. Layout issues were identified that caused users to navigate to unexpected places, but users were quickly able to recover from this without needing assistance. The learnability tests showed that the system was very easy to learn and it required very little effort on the part of the participants to use it successfully.

Build Simple Workflow

For this task the participants were asked to build a simple workflow, that had been provided on a diagram. (See Figure 4.1) The instructions did not provide any information on how the system should be used and participants were required to determine how to complete the task on their own.

Usefulness

85% of participants indicated that they found the system to be effective for the task that was required. 80% also indicated that the system aided them in being productive and completing the task. Overall, 94% of users found the system useful.

Users found that the system gave them good control over what they were doing and made the task relatively easy to complete, and felt the number of steps involved were appropriate.

Observations of the participants revealed that the participants quickly understood what was expected and were able to use the system to effectively build the workflow that was required. All users were able to complete the task.

Ease of Use

85% of participants indicated that they found the system simple and easy to use. They believed that the number of steps were minimal to complete the tasks that were required. Inconsistencies within the system was not noted by the participants. 75% of users felt like they were easily able to recover after making an error. They were confident that the system was being used successfully. In addition, 75% of users commented very favourably on ease of use.

While monitoring the user interaction, users navigated the entire page before deciding on how they would add a task. Typically they would select the *User Task* and be confused that the option for the *Server Task* was not available. Several users were directed to the cancel button. This indicates that the option should be combined and indication as to what the task type is should appear in the combobox.

Users did seem to intuitively understand that the tasks could be dragged within the visualisation. Furthermore, since the creation of a task placed the node in a fixed position in the visualisation, users seemed to believe that their previous task had been overwritten. Stronger visual cues on the nodes in the visualisation should be added to indicate that drag options are available. Nodes should also initially be randomly placed to avoid confusion. A similar problem was noted in dragging the *arrow icon* to indicate dependencies.

A rather significant problem was encountered during the testing. While the users were navigating the page to determine what to do, they encountered the Jobs section, believing this is where they needed to add tasks. This wasted a lot of time for some users. This is caused by the confusing terminology that was used. Better naming convention should be established to avoid this problem.

Ease of Learning

Users were very quickly able to determine how the system worked and how to use it efficiently. Participants grew much more confident and decisive as the task progressed. This is also validated by 85% of the responses in the survey, that indicated that the system was very easy learn. Participants did not hesitate between tasks and quickly added all the required nodes.

Satisfaction

90% of users were very satisfied with the system; and 57% agreed that the system was fun to use. Participants felt that the system worked the way they expected it to work. Emotional responses, that give a better indication of the user experience, indicated that a significant portion of the users believed the system to be wonderful and that they need to have it. Overall, 80% users found the system pleasant to use.

During the experiment, the users were noted to express excitement and happiness was expressed when they realised how the system works.

Efficiency

Users took an average of 7 minutes to complete, with an interquartile range of 5 to 9 minutes. Once the participants had placed the first task, they were immediately more efficient. The efficiency of the participants was quite varied. Some participants spent a significant amount of time staring at the visualisation, as they thought that the previous tasks had been removed, due to the *drag-and-drop* functionality not being apparent.

The second task was overall rated as a positive experience; while some participants expressed the need for more complete instructions, on using the system. However, based on the data acquired on the *learnability* of the system, this appears to have been the appropriate choice. The experiment strongly indicated that the *Job* terminology would need to be reworked. Further observations showed that more visual cues are required to indicate that the tasks nodes are *draggable*.

The user study was able to highlight problems associated with the system. These include: (i) Task Feedback; (ii) Navigation; (iii) Visual cues; and (iv) certain Terminology. In terms of usability criteria, *Effectiveness* and *Simplicity* could not be tested due to resource and time constraints. Findings on the usability were as follows:

Learnability Users were very quickly able to get used to the system and became competent in completing the tasks.

Efficiency Participants were able to complete the tasks in the required amount of time. This was done intuitively without giving specific information on how to use the interface.

Satisfaction Users indicated that they were satisfied with the system. It was also noted that overall the participants were not frustrated with the system. Users confidently used the system and did so with ease.

Error Participants who did make errors were able to recover swiftly using the interface. Problems however did arise with the job terminology where users were confusing jobs with tasks, causing errors. Some detours were also noted where a group of participants were not able to intuitively navigate back to the *Task Overview* screen.

4.2 System Test

The system needs to be evaluated in terms of its applicability to successfully manage and coordinate the tasks of the Zamani Project. Due to time constraints, however, the full set of the tasks within the project could not be implemented. In order to evaluate its applicability, two tests were performed to test whether the solution would work. The first test was designed to determine the general applicability of such a system, by executing a workflow that is unrelated, while the second test involves implementing small portion of the tasks of the Zamani Project to determine if if could be successfully implemented.

4.2.1 Applicability Tests

Two test were performed. The first test was to build a simple workflow that creates a document. The second test builds a workflow that represents a portion of the process that creates a 3D model from the raw scan.

Creating Document

The purpose of this task is to mix user tasks along with server tasks. This workflow is the same workflow that was used in the user test. A representation of the workflow can be seen in Figure 4.1. The process involves: a user creating a "*.txt*" file. Images are then converted to a different format. These are then converted to the LaTeX format with the pictures added. The document is then compiled to a *PDF*.

The workflow was successfully able to build the document. The completed workflow can be seen in Figure 4.2.

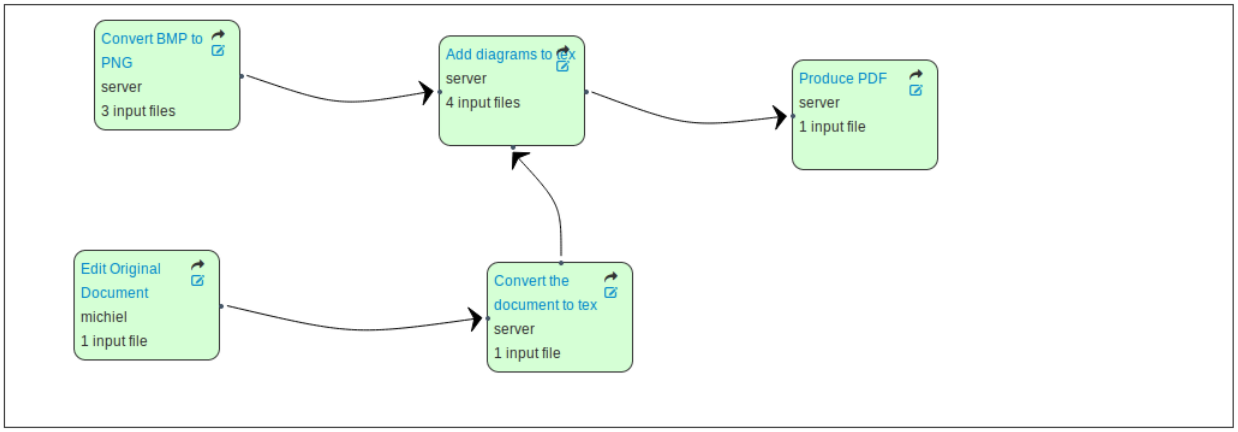


Figure 4.2: Completed workflow that creates a simple PDF document

Integration Test

A lengthy process is requires to build the 3D models from the initial laser scans require a lengthy process. This process can be illustrated in Figure 4.3 [2]. In order to test whether the system can be applied successfully, a portion of this process was implemented in the system.

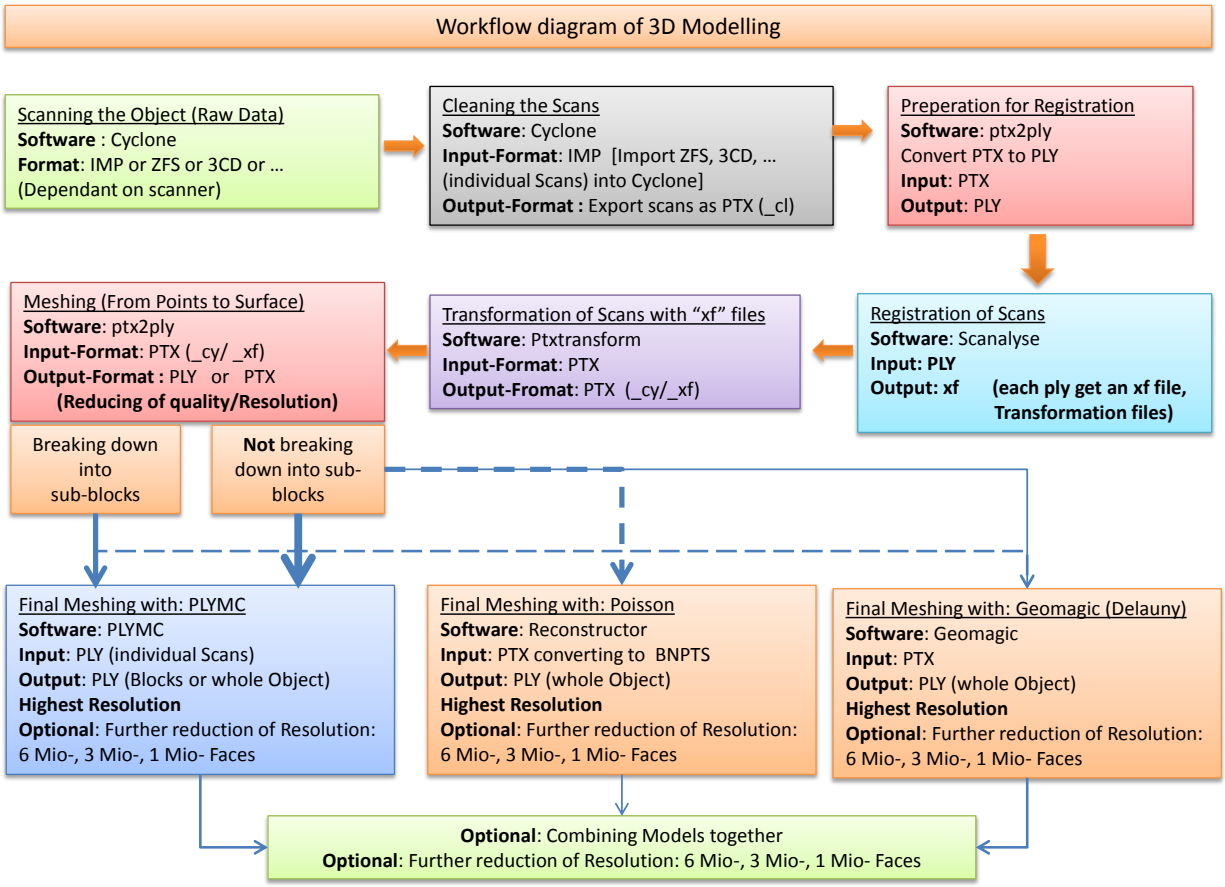


Figure 4.3: Zamani - 3D Modelling Workflow

The first five steps in the modelling process were implemented using the workflow system to determine how the system would support running on actual data. Since data from a previous site was available, the user tasks were merely simulated by copying over the required data for this task. This was implemented successfully. Figure 4.4 shows the workflow being executed.
This successful integration shows that the system is able to support some workflows required by the Zamani project.

4.2.2 Filtering

The system can successfully implement the workflow required for the Zamani Project, however filtering workflow at a site level is not ideal. Many workflows get replicated at a building level and beyond this the artifact creation can also be distinguished by category. In order for the workflows built within the system to be more reusable, it would need to be extended to store workflows at finer level. These could then be nested in a hierarchical manner. This would allow entire workflows to be abstracted as a single node. This would decrease set up time and increase reusability of the system, however, further experimentation would be required to determine the effect on User Experience.

[2]Source: Zamani Project at UCT

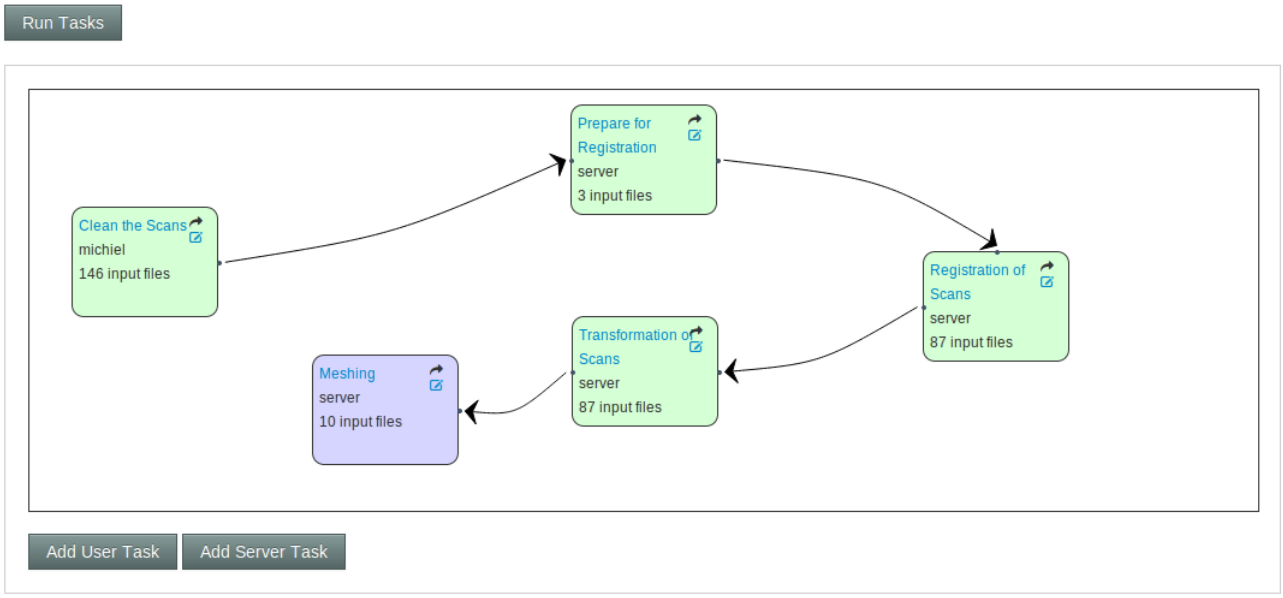


Figure 4.4: Partially implemented Zamani modelling workflow

4.2.3 Performance

The movement and processing of data is regarded as being the most expensive portion of the process. The data movement is directly offloaded to *rsync*, whereas the data processing is offloaded to the native applications. The system management of these activities outperforms these tasks. As such, the system's performance is bounded by these activities. This offloading procedure is illustrated in Figure 4.5.

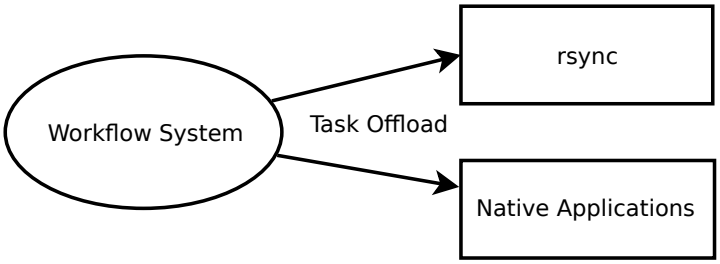


Figure 4.5: Performance: Task offload

Due to this, a performance evaluation of the system was not deemed necessary. Performance of the system could, however, be affected by several processes being executed simultaneously. In order to speed up the process in this case the system would need to be distributed to include multiple processing servers; this was decided outside the scope of the project.

4.3 Summary

The system was evaluated and it was determined that: (i) it can successfully integrate a portion of the Zamani Workflow; and (ii) user could successfully use the system with very little effort.The terminology was found to be somewhat confusing and it was found that better filer filtering options would be beneficial. Overall, however, the system was evaluated as being a success.

Chapter 5

Conclusion and Future Work

This system can successfully manage a complex set of tasks with arbitrary dependencies. Tasks could be either be fully automated by the system or could be completed by users. When user tasks are started, the required files are incrementally transferred to the desktop host of the user using *rsync* to only transfer files that have not been transfered or are out of date. To enforce quality and accuracy, a feature was added that enforces that user tasks be validated by experienced members of the team before the tasks can be labelled as complete.

Automated tasks are executed on the server due to the fact that these tasks operate on very large files. By executing them locally, data does not need to be transferred, which would be an expensive process. Tasks are automatically started when all dependencies are met.

In the event that a task fails, the system also allows the user to inspect the logging information that is generated during the execution of the task. Once the problem is identified the tasks can then be manually restarted.

5.1 Conclusion

This research project was concluded with the successful implementation of a Workflow Management System. The design and implementation was done in three iterations. This was explained in depth in Chapter 3.

The system was then successfully evaluated in Chapter 4 both for its usability and its effectiveness at solving the problem. The following positive results was obtained during the evaluation of the system:

1. The system was successfully able to implement and execute a portion of the workflow in the modelling section of the modelling tasks that are present in the Zamani-Project. This sample workflow used a mix of system and user task.

2. The system was positively evaluated using a sample group of 24 users. This evaluation revealed that users found the system useful, easy to use and users were satisfied using the system. User responses and the observations made during the test it was found that the system is effective and is very easy to learn.

This system was however not implemented within the production Zamani Project. This was mainly due to time constraints, caused by the scale and time required to implement it. Functionally the system could be implemented, however this process could be significantly simplified by the addition of some features. These are mentioned in the future work session.

5.2 Future Work

During the implementation of the workflow system, various possible extensions that could be added to the system could not be implemented. These features would improve the system both in terms of performance, usability and set up time.

Hierarchical Workflows

To allow better control and re usability over tasks, workflows should be abstracted to include a hierarchy. Such a hierarchy would allow entire workflows to be represented as singular nodes. These workflow, could then be repackaged and reused in different sites, or even the same site. This would also allow the setup for new sites to be much faster, as prepackaged workflows could easily be used as drop-in components.

Parameterized Scripts

Oftentimes particular parameters of a script can change from one site to another. This change does not necessarily affect the *Task type*; however, with the current implementation of the system the change would need to be made at this point. This can be greatly improved by allowing a *Task* to send parameters to the job. This would require the Task Subsystem to allow parameters to be sent dynamically to the *Task Type*.

Rule Based File Filters

Currently within the system all the files in the output directory of are task is treated as input to successor tasks. Tasks often only use a portion of the files created by the predecessor. In order to currently facilitate this with the system an additional filtering, task would need to be set up that filters out unused files. By including a rule based filtering system much greater control can be placed on the output files. Such rule based filters have been successfully implemented in other systems[Conery et al., 2005].

Interactive Task Feedback Options

In order to avoid one of the problems that were found in Section 4.1.3, more interactivity is required for *Tasks*. This primarily includes real-time updates on the status of tasks. Further developments include the ability to do more interactive validation such as discussion integration. The addition of these collaborative tasks could resolves issues with tasks in a uniform manner[Guimarães et al., 1998].

Transformation-based Task Support

Currently the system is built around creating derivative data items. However, it is often common for certain files within a site to change, without creating an additional copy. Although this behaviour is implicitly allowed, it should be extended to be better defined within the system.

Parallel Task Processing

One of the most crucial aspects affecting the long term feasibility of the system is its ability to scale and handle larger and more complicated workflows. In this regard the server node would become a significant bottleneck in processing *Server Tasks*. In order to alleviate this problem, the system would need to become distributed. This would present its own set of problems as data would need to be efficiently distributed along the computation nodes to ensure efficiency.

Bibliography

[Aragon and Runge, 2009] Aragon, C. R. and Runge, K. J. (2009). Workflow management for high volume supernova search. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 949–955, New York, NY, USA. ACM.

[Bevan, 1995] Bevan, N. (1995). Measuring usability as quality of use. *Software Quality Journal*, 4(2):115–130.

[Brahe and Schmidt, 2007] Brahe, S. and Schmidt, K. (2007). The story of a working workflow management system. In *Proceedings of the 2007 international ACM conference on Supporting group work*, GROUP '07, pages 249–258, New York, NY, USA. ACM.

[Carstensen and Srensen, 1996] Carstensen, P. H. and Srensen, C. (1996). From the social to the systematic. *Computer Supported Cooperative Work (CSCW)*, 5:387–413. 10.1007/BF00136712.

[Chin et al., 1988] Chin, J. P., Diehl, V. A., and Norman, K. L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 213–218, New York, NY, USA. ACM.

[Conery et al., 2005] Conery, S., Catchen, M., and Lynch, M. (2005). Rule-based workflow management for bioinformatics. *The VLDB JournalThe International Journal on Very Large Data Bases*, 14(3):318–329.

[da Cruz et al., 2008] da Cruz, S. M. S., Batista, V., Dávila, A. M. R., Silva, E., Tosta, F., Vilela, C., Campos, M. L. M., Cuadrat, R., Tschoeke, D., and Mattoso, M. (2008). Orthosearch: a scientific workflow approach to detect distant homologies on protozoans. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 1282–1286, New York, NY, USA. ACM.

[Davidson et al., 2007] Davidson, S., Boulakia, S., Eyal, A., Ludäscher, B., McPhillips, T., Bowers, S., Anand, M., and Freire, J. (2007). Provenance in scientific workflow systems. *IEEE Data Eng. Bull*, 30(4):44–50.

[De Roure and Goble, 2009] De Roure, D. and Goble, C. (2009). Software design for empowering scientists. *Software, IEEE*, 26(1):88 –95.

[Di Martino et al., 2007] Di Martino, S., Ferrucci, F., Paolino, L., Sebillo, M., Tortora, G., Vitiello, G., and Avagliano, G. (2007). Towards the automatic generation of web gis. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, GIS '07, pages 57:1–57:4, New York, NY, USA. ACM.

[El Adnani et al., 2001] El Adnani, M., Yétongnon, K., and Benslimane, D. (2001). A multiple layered functional data model to support multiple representations and interoperability of gis: application to urban management systems. In *Proceedings of the 9th ACM international symposium on Advances in geographic information systems*, GIS '01, pages 70–75, New York, NY, USA. ACM.

[Forlizzi and Battarbee, 2004] Forlizzi, J. and Battarbee, K. (2004). Understanding experience in interactive systems. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS '04, pages 261–268, New York, NY, USA. ACM.

[Gray et al., 2005] Gray, J., Liu, D., Nieto-Santisteban, M., Szalay, A., DeWitt, D., and Heber, G. (2005). Scientific data management in the coming decade. *ACM SIGMOD Record*, 34(4):34–41.

[Gray and Szalay, 2007] Gray, J. and Szalay, A. (2007). escience-a transformed scientific method. *presentation to the Computer Science and Technology Board of the National Research Council, Mountain View, CA*.

[Greene and Troyanskaya, 2010] Greene, C. and Troyanskaya, O. (2010). Integrative systems biology for data-driven knowledge discovery. In *Seminars in nephrology*, volume 30, pages 443–454. Elsevier.

[Guimarães et al., 1998] Guimarães, N., Antunes, P., and Pereira, A. (1998). The integration of workflow systems and collaboration tools. *Workflow Management Systems and Interoperability*, pages 222–245.

[Harpaz et al., 2012] Harpaz, R., DuMouchel, W., Shah, N., Madigan, D., Ryan, P., and Friedman, C. (2012). Novel data-mining methodologies for adverse drug event discovery and analysis. *Clinical Pharmacology & Therapeutics*, 91(6):1010–1021.

[Johnson et al., 2009] Johnson, D., Meacham, K., and Kornmayer, H. (2009). A middleware independent grid workflow builder for scientific applications. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 86 –91.

[Kaptchuk, 2003] Kaptchuk, T. (2003). Effect of interpretive bias on research evidence. *BMJ*, 326(7404):1453–1455.

[Leff and Rayfield, 2001] Leff, A. and Rayfield, J. (2001). Web-application development using the model/view/controller design pattern. In *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*, pages 118–127. IEEE.

[Ludäscher et al., 2009] Ludäscher, B., Altintas, I., Bowers, S., Cummings, J., Critchlow, T., Deelman, E., Roure, D., Freire, J., Goble, C., Jones, M., et al. (2009). Scientific process automation and workflow management. *Scientific Data Management: Challenges, Existing Technology, and Deployment, Computational Science Series*, pages 476–508.

[Lund, 2001] Lund, A. (2001). Measuring usability with the use questionnaire. stc usability sig newsletter.

[Migliorini et al., 2011] Migliorini, S., Gambini, M., Belussi, A., Negri, M., and Pelagatti, G. (2011). Workflow technology for geo-processing: the missing link. In *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, COM.Geo '11, pages 36:1–36:6, New York, NY, USA. ACM.

[Montella et al., 2007] Montella, R., Giunta, G., and Riccio, A. (2007). Using grid computing based components in on demand environmental data delivery. In *Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, UPGRADE '07, pages 81–86, New York, NY, USA. ACM.

[Moreau et al., 2008] Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., and Paulson, P. (2008). The open provenance model: An overview. *Provenance and Annotation of Data and Processes*, pages 323–326.

[Muller, 1991] Muller, M. (1991). Pictivean exploration in participatory design. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 225–231. ACM.

[Muller and Kuhn, 1993] Muller, M. and Kuhn, S. (1993). Participatory design. *Communications of the ACM*, 36(6):24–28.

[Nielsen, 2003] Nielsen, J. (2003). Usability 101: Introduction to usability. *Jakob Nielse's Alertbox, August*, 25.

[Qiu et al., 2003] Qiu, X., Du, Y., and Zheng, F. (2003). An automated reasoning method used in workflow management system. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 5, pages 5016 – 5021 vol.5.

[Sanders et al., 2008] Sanders, D. T., Hamilton, Jr., J. A., and MacDonald, R. A. (2008). Supporting a service-oriented architecture. In *Proceedings of the 2008 Spring simulation multiconference*, SpringSim '08, pages 325–334, San Diego, CA, USA. Society for Computer Simulation International.

[Shegalov et al., 2001] Shegalov, G., Gillmann, M., and Weikum, G. (2001). Xml-enabled workflow management for e-services across heterogeneous platforms. *The VLDB Journal*, 10(1):91–103.

[Shneiderman, 2002] Shneiderman, B. (2002). Inventing discovery tools: combining information visualization with data mining1. *Information Visualization*, 1(1):5–12.

[Simmhan and Barga, 2011] Simmhan, Y. and Barga, R. (2011). Analysis of approaches for supporting the open provenance model: A case study of the trident workflow workbench. *Future Generation Computer Systems*, 27(6):790 – 796.

[Simmhan et al., 2009] Simmhan, Y., Barga, R., Ingen, C. v., Lazowska, E., and Szalay, A. (2009). Building the trident scientific workflow workbench for data management in the cloud. In *Proceedings of the 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences*, ADVCOMP '09, pages 41–50, Washington, DC, USA. IEEE Computer Society.

[Slot and van Zoolingen, 2005] Slot, M. and van Zoolingen, P. (2005). Workflow management systems. Technical report, Technical report, Division of Mathematics and Computer Science, Vrije Universiteit, The Nerherlands.

[Suchman, 1983] Suchman, L. A. (1983). Office procedure as practical action: models of work and system design. *ACM Trans. Inf. Syst.*, 1(4):320–328.

[Tai et al., 2004] Tai, S., Khalaf, R., and Mikalsen, T. (2004). Composition of coordinated web services. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, Middleware '04, pages 294–310, New York, NY, USA. Springer-Verlag New York, Inc.

[Taylor et al., 2006] Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M. (2006). *Workflows for e-Science: Scientific Workflows for Grids*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Thomas et al., 2011] Thomas, R., Nugent, P., and Meza, J. (2011). Synapps: Data-driven analysis for supernova spectroscopy. *Publications of the Astronomical Society of the Pacific*, 123(900):237–248.

[Tractinsky, 1997] Tractinsky, N. (1997). Aesthetics and apparent usability: empirically assessing cultural and methodological issues. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, pages 115–122, New York, NY, USA. ACM.

[Tullis and Albert, 2008] Tullis, T. and Albert, B. (2008). Measuring the user experience. *Collecting, Analyzing, and Presenting Usability Metrics*.

[van der Aalst and Basten, 2002] van der Aalst, W. and Basten, T. (2002). Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science*, 270(12):125 – 203.

[Vermeeren et al., 2010] Vermeeren, A., Law, E., Roto, V., Obrist, M., Hoonhout, J., and Väänänen-Vainio-Mattila, K. (2010). User experience evaluation methods: current state and development needs. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 521–530. ACM.

[Wang et al., 2009] Wang, J., Crawl, D., and Altintas, I. (2009). Kepler + hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, WORKS '09, pages 12:1–12:8, New York, NY, USA. ACM.

[Withana et al., 2010] Withana, E. C., Plale, B., Barga, R., and Araujo, N. (2010). Versioning for workflow evolution. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 756–765, New York, NY, USA. ACM.

[Yu and Buyya, 2005] Yu, J. and Buyya, R. (2005). A taxonomy of scientific workflow systems for grid computing. *Sigmod Record*, 34(3):44.

[Zhang and Adipat, 2005] Zhang, D. and Adipat, B. (2005). Challenges, methodologies, and issues in the usability testing of mobile applications. *International Journal of Human-Computer Interaction*, 18(3):293–308.

Appendix A

Questionnaire

Workflow Management System Usability Test

User Test of the Zamani Workflow system.
STUDENT NUMBER: BRDMIC026
CONTACT: mbaird@cs.uct.ac.za OR michiel@baird.za.net
Supervisor:
NAME: Hussein Suleman
CONTACT: hussein@cs.uct.ac.za
BSc Computer Science Honours Project:
TITLE: Workflow Management
AIM: To create a system that can easily create and facilitate the workflow of Geomatics based research project

1 -- INTRODUCTORY STATEMENT
Zamani Workflow is a system built to create and support workflows within the Zamani Project http://www.zamani-project.org/ This system is used to manage the tasks associated with the process of creating the digital heritage artifacts. These tasks are a mix between manual user tasks and tasks that can be fully automated and run on a server.

Two main focusses exists

- Manage the tasks that each user has to complete.
- Building workflows

To manage user tasks, a web interface is used to indicate the status of the task on the desktop of the user. The tasks assigned to users are however, not performed on the web interface.

2 -- PURPOSE
This survey will analyse Zamani Workflow

3 -- TASKS

- Complete tasks in an existing workflow
- Set Up a simple workflow, that can be executed.

****These tasks will be detailed on the next stages of the survey***

4 -- ENVIRONMENT & DURATION
You will perform the task will be performed in a lab environment. During this period your actions will be noted and you will be timed in the completion of the tasks. Time to complete the tasks are user-dependent and may take around 20 minutes.

5 -- CONFIDENTIALITY
If you consent to participate in this evaluation, your personal information will be kept confidential. Any information you provide in the survey or the test repository is kept confidential between you and the researcher. You may also request the removal of any items you deposit.

6 -- STATEMENT OF CONSENT
"I acknowledge that I have read the above explanation of this evaluation. I understand that the collected data from this survey will be analysed and used to evaluate the mentioned system. I also understand that the researcher will not disclose my personal information. By selecting the option 'next' below I agree to participate in this evaluation."

There are 16 questions in this survey

Background Information

This section obtains some background information on you.

1 [1]How old are you *

Please write your answer here:

-

2 [2]Please rate your experience with in the following

Please choose the appropriate response for each item:

	Novice				Expert
How experienced are you at using a computer?	○	○	○	○	○
Rate your personal competence in using a Web Browser	○	○	○	○	○
Rate your understanding of workflow management	○	○	○	○	○

3 [4]Do you use a Day Planner or similar tool to manage your day to day activities.

Please choose **only one** of the following:

- ○ Yes
- ○ No

4 [5]What is your field of study?

Please write your answer here:

Complete a simple Task

One of the goals of the Zamani workflow system is to facilitate user tasks in a multi-user environment. Your goal will be to use the system to execute a series of tasks .

Step 1 - Log into the system

A tab on the browser shoulf be open waiting for you to log in.

- Username: user_test1
- Password: usertest

Step 2 - Complete the three outstanding tasks

There are 3 outstanding tasks listed. Complete the tasks then answer the questions below,

5 [1]Did you successfully complete all the outstanding tasks? *

Please choose **only one** of the following:

- ○ Yes
- ○ Two tasks successfully completed
- ○ One Task successfully completed
- ○ No

6 [2]

Rate the system in terms of usefulness in the following categories.

Please choose the appropriate response for each item:

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
It helps me be more effective.	○	○	○	○	○	○	○
It feels unnecessary to use the system	○	○	○	○	○	○	○
It does not meet my expected requirements	○	○	○	○	○	○	○
It helps me be more productive.	○	○	○	○	○	○	○
It feels like I have little control over what is happening with the activities	○	○	○	○	○	○	○
It is useful.	○	○	○	○	○	○	○
I find it difficult to get things done with the system	○	○	○	○	○	○	○
It gives me more control over the activities.	○	○	○	○	○	○	○
It makes the things I want to accomplish easy to get done.	○	○	○	○	○	○	○
It saves me time when I use it.	○	○	○	○	○	○	○
It meets my needs.	○	○	○	○	○	○	○
It does everything I would expect it to do.	○	○	○	○	○	○	○
It takes up a lot of time.	○	○	○	○	○	○	○

7 [3]Rate the system in terms of Ease of Use

Please choose the appropriate response for each item:

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
There were too many steps involved	○	○	○	○	○	○	○
It is easy to use.	○	○	○	○	○	○	○
It is simple to use.	○	○	○	○	○	○	○
It is user friendly.	○	○	○	○	○	○	○
It requires the fewest steps possible to accomplish what I want to do with it.	○	○	○	○	○	○	○
It is flexible.	○	○	○	○	○	○	○
Using it is effortless.	○	○	○	○	○	○	○
Mistakes are difficult to recover from	○	○	○	○	○	○	○
It is unnecessarily complex	○	○	○	○	○	○	○
I can use it without written instructions.	○	○	○	○	○	○	○
I don't notice any inconsistencies as I use it.	○	○	○	○	○	○	○
It is difficult to use	○	○	○	○	○	○	○

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
Both occasional and regular users would like it.	○	○	○	○	○	○	○
I can recover from mistakes quickly and easily.	○	○	○	○	○	○	○
It required a lot of effort to use	○	○	○	○	○	○	○
I can use it successfully every time.	○	○	○	○	○	○	○

8 [4]Rate the system in terms of Ease of Learning

Please choose the appropriate response for each item:

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
I learned to use it quickly.	○	○	○	○	○	○	○
I easily remember how to use it.	○	○	○	○	○	○	○
I found it difficult to get used to	○	○	○	○	○	○	○
It is easy to learn to use it.	○	○	○	○	○	○	○
I quickly became skillful with it.	○	○	○	○	○	○	○
I could not remember what I needed to do	○	○	○	○	○	○	○

9 [5]Satisfaction in using the system

Please choose the appropriate response for each item:

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
I am satisfied with it.	○	○	○	○	○	○	○
I would recommend it to a friend.	○	○	○	○	○	○	○
It is fun to use.	○	○	○	○	○	○	○
It works the way I want it to work.	○	○	○	○	○	○	○
It is wonderful.	○	○	○	○	○	○	○
I felt disappointed with the system	○	○	○	○	○	○	○
I feel I need to have it.	○	○	○	○	○	○	○
It is pleasant to use.	○	○	○	○	○	○	○
The system did not do what I expected.	○	○	○	○	○	○	○

10 [6]Any general comments on completing the outstanding tasks.

Please write your answer here:

Set up a simple workflow

A second part of the system is to build interactive workflows.

Step 1 - Log out of system

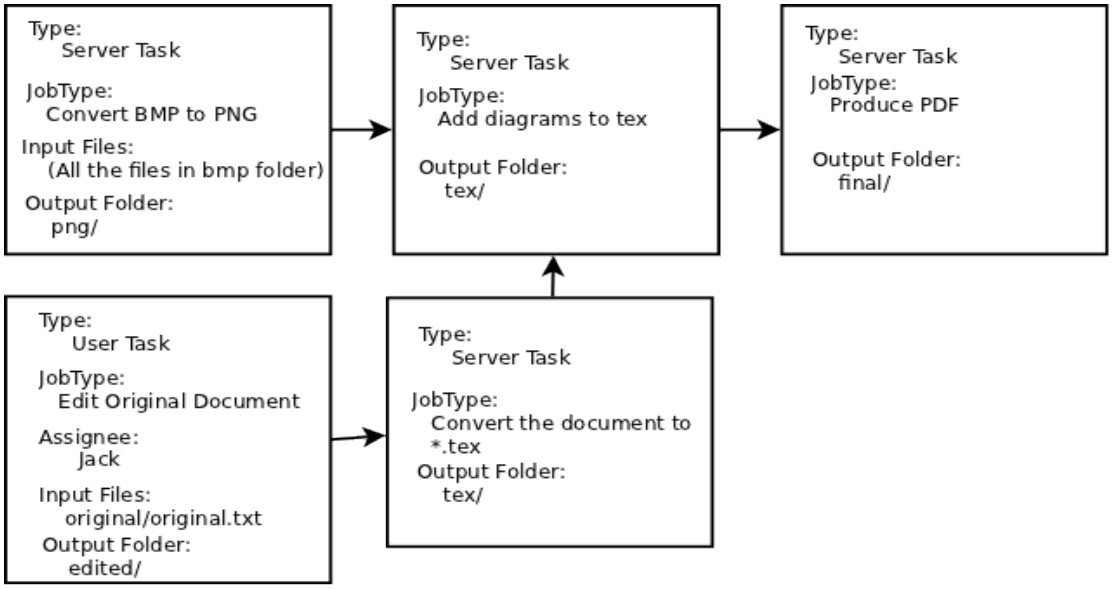
Locate the log out button and log off the system.

Step 2 - Log in with a new account

- Username: user_test2
- Password: testpassword

Step 3 - Set up the a sample work flow

Open the site called "UCT Site" and use the system to recreate the workflow below:



11 [1]Could you successfully set up the workflow as it was described. *

Please choose **only one** of the following:

- ◯ Yes
- ◯ No

12 [2]

Rate the system in terms of usefulness in the following categories.

Please choose the appropriate response for each item:

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
It helps me be more effective.	○	○	○	○	○	○	○
It feels unnecessary to use the system	○	○	○	○	○	○	○
It does not meet my expected requirements	○	○	○	○	○	○	○
It helps me be more productive.	○	○	○	○	○	○	○
It feels like I have little control over what is happening with the activities	○	○	○	○	○	○	○
It is useful.	○	○	○	○	○	○	○
I find it difficult to get things done with the system	○	○	○	○	○	○	○
It gives me more control over the activities.	○	○	○	○	○	○	○
It makes the things I want to accomplish easy to get done.	○	○	○	○	○	○	○
It saves me time when I use it.	○	○	○	○	○	○	○
It meets my needs.	○	○	○	○	○	○	○
It does everything I would expect it to do.	○	○	○	○	○	○	○
It takes up a lot of time.	○	○	○	○	○	○	○

13 [3]Rate the system in terms of Ease of Use

Please choose the appropriate response for each item:

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
There were too many steps involved	○	○	○	○	○	○	○
It is easy to use.	○	○	○	○	○	○	○
It is simple to use.	○	○	○	○	○	○	○
It is user friendly.	○	○	○	○	○	○	○
It requires the fewest steps possible to accomplish what I want to do with it.	○	○	○	○	○	○	○
It is flexible.	○	○	○	○	○	○	○
Using it is effortless.	○	○	○	○	○	○	○
Mistakes are difficult to recover from	○	○	○	○	○	○	○
It is unnecessarily complex	○	○	○	○	○	○	○
I can use it without	○	○	○	○	○	○	○

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
written instructions.							
I don't notice any inconsistencies as I use it.	○	○	○	○	○	○	○
It is difficult to use	○	○	○	○	○	○	○
Both occasional and regular users would like it.	○	○	○	○	○	○	○
I can recover from mistakes quickly and easily.	○	○	○	○	○	○	○
It required a lot of effort to use	○	○	○	○	○	○	○
I can use it successfully every time.	○	○	○	○	○	○	○

14 [4]Rate the system in terms of Ease of Learning

Please choose the appropriate response for each item:

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
I learned to use it quickly.	○	○	○	○	○	○	○
I easily remember how to use it.	○	○	○	○	○	○	○
I found it difficult to get used to	○	○	○	○	○	○	○
It is easy to learn to use it.	○	○	○	○	○	○	○
I quickly became skillful with it.	○	○	○	○	○	○	○
I could not remember what I needed to do	○	○	○	○	○	○	○

15 [5]Satisfaction in using the system

Please choose the appropriate response for each item:

16 [6]Any general comments on building the workflow

Please write your answer here:

Thank you very much for participating, if you have any comments or concerns please email me at michiel@baird.za.net .
1970/01/01 – 02:00

Submit your survey.
Thank you for completing this survey.