Software Engineering C

Week 4 Group Report

**Project Leader:**

Adam Bramley

**Team Members:**

Stephen Allan

Leith Bade

Peter Williamson

Table of Contents

[1. Introduction 4](#_Toc268768733)

[2. Requirements Revision 5](#_Toc268768734)

[2.1 Functional Requirements: 5](#_Toc268768735)

[2.2 Non-Functional Requirements: 5](#_Toc268768736)

[3. Technological Assessment 5](#_Toc268768737)

[3.1 Language 6](#_Toc268768738)

[3.2 Integrated Development Environment (IDE) 6](#_Toc268768739)

[3.3 Architecture 6](#_Toc268768740)

[3.4 Plugin system 6](#_Toc268768741)

[3.5 Testing System 7](#_Toc268768742)

[3.6 Web Framework 7](#_Toc268768743)

[3.7 Web server 7](#_Toc268768744)

[3.8 Database 7](#_Toc268768745)

[3.9 Build System 8](#_Toc268768746)

[4. Project Life Cycle Methodologies 8](#_Toc268768747)

[4.1 Meetings 8](#_Toc268768748)

[4.2 Sprints 8](#_Toc268768749)

[4.3 Spike Tests 8](#_Toc268768750)

[4.4 Milestones 8](#_Toc268768751)

[5. System Architecture 9](#_Toc268768752)

[6. Deployment Strategy 9](#_Toc268768753)

[6.1 Deploying a JSP Application 10](#_Toc268768754)

[6.2 The Database 10](#_Toc268768755)

[7. Risk Management 10](#_Toc268768756)

[8. Issue Tracking Policy 10](#_Toc268768757)

[8.1 Issue Tracking Workflow 10](#_Toc268768758)

[9. Testing and Metrics 11](#_Toc268768759)

[10. Appendices 12](#_Toc268768760)

[10.1 Appendix A 12](#_Toc268768761)

# 1. Introduction

The goal of this project is to provide the administration staff at Massey University with a program to assist with the submission of academic publications to the Research Administration Management System (RIMS). The program is to capture relevant information on the academic paper from a generic input (document object identifier) and give the user an automatically generated form of information to submit to RIMS.

Our project team consists of 4 members each with different skill sets and experiences to contribute to an overall solid workforce. The following paragraphs outline brief auto-biographies from the team members with relevant information about their specific skills and talents.

I, Adam Bramley, have had a lot of experience using Java and using various techniques to develop appealing interfaces, which are clear and easy to use. My experience comes from studies at Massey University where I have done numerous programming papers along side Software Engineering, Human-Computer Interaction, and Web Development. This knowledge gives me the ability to adapt to new technologies and work well in a team.

Peter is competent using Java and his experience comes solely from doing Massey University papers. ­­The papers have given him the some skills in database development, human computer interaction, web development and designing of systems. He is able to work well by himself or in a team.

…

…

# 2. Requirements Revision

After speaking with Karen and Craig on the 21st July and multiple group meetings we came out with a refined list of functional and non-functional requirements.

## 2.1 Functional Requirements:

1. The system needs to be able to take a DOI as an input, and output metadata for the article. This is to be done through a series of steps, going through the major databases like CrossRef and Scopus, then proceeding to smaller publisher’s websites.
2. If the system cannot extract any metadata, the raw article is to be produced.
3. The system has to display the metadata on screen and allow the user to edit the data.
4. The system has to be able to automatically send verification emails to authors of the academic papers when the details have been finalised.
5. Access to a local database is needed. The academic paper details are saved to a local database along with author details; this is so the user can easily retrieve past submissions and author details.
6. The system needs to be able to print the cover sheet containing all the metadata.
7. If the user doesn’t have a DOI, the system needs an advanced search function with the ability to search the local database and online sources for matches in multiple fields (i.e. by author, paper title, or keywords)
8. A further extension of the system which is slightly outside the scope of Karen’s specific requirements is for the system to be able to export the local database, or the metadata of specific papers, to a CSV or XML file which would then be sent straight to RIMS for importing into their database. *This is a requirement that will only be implemented if there is time near the end of the project.*

## 2.2 Non-Functional Requirements:

1. The system is to be run on windows
2. The system requires username and password based security.
3. The system is to return local database results in under 3 seconds and web results in under 15 seconds per plugin.
4. The system needs to handle plug-in architecture
5. The system needs to have an automated deployment and maintenance procedure.

These requirements have also been signed off by Karen on 5th August.

# 3. Technological Assessment

Throughout the project, we have done extensive technological assessment. This report will cover the technologies we have chosen to go with and reasons behind why we chose them.

## 3.1 Language

We have chosen to use Java as our main programming language. All of our team members have experience using Java through study at Massey, and some have extra experience through work and other extracurricular activities. Java is easy to use, and there is plenty of technical expertise available via lecturers at Massey. Java has a great online community which provides extensive support for users and there are plenty of resources available at the library if the online help doesn’t suffice. Furthermore, Java is widely used and freely available and fits in with our object-oriented approach to this project. Java is also easily testable through JUnit and easily maintainable; these tie in with the great integrated development environment (Eclipse). There is also a good plugin system available for Java.

We also considered C#, Python, Ruby, and PHP.

## 3.2 Integrated Development Environment (IDE)

We have chosen to use Eclipse JEE Helios as our primary IDE for the project. Again, all of our team members have a lot of experience using Eclipse through teachings at Massey and use at home. It is also easy to use and there is plenty of support available for it through Massey, the internet, and written resources. Eclipse is also very reliable and widely used in the community. It has a very wide range of plugins that can be easily installed and extend to almost any technology. There are also good features in Eclipse for testing, refactoring, and building our system. We will be using plugins such as Subclipse, which allows us to access our repository through Eclipse, and utilising Eclipse’s features that support Tomcat, SQL, and JSP. The version of Eclipse we are using supports Java 6, which is needed to use the plugin architecture we have chosen (ServiceLoader).

We also considered NetBeans, Visual Studio, SharpDevelop, and MonoDevelop.

## 3.3 Architecture

We have chosen to structure our software in a Web app fashion. All of our team members have good experience with technologies surrounding web apps through Massey and other work. Using Web app architecture allows us to have a very straight forward deployment strategy which is easy to update, install, and maintain. If the project were to grow in popularity, our architecture will allow it with its scalability. It is also a widely used architecture with a huge amount of support available through various sources. It is a lightweight approach and freely available, and also supports our SQL database.

We also considered a Desktop application, and a Client/Server application.

## 3.4 Plugin system

We have chosen to use a plugin system called ServiceLoader (SL). While our group doesn’t have much experience with SL, there isn’t a plugin system that we are all familiar with so it was a good option to go with due to its ease of use. There is ample support available online for it and it is widely used. It’s reliable due to its simplicity and the fact that it’s type safe. SL is also freely available and has a well defined interface.

We also considered Guice, OSGi, Eclipse Equinox, Java Plugin Framework, and Spring Framework

## 3.5 Testing System

For testing our system we will be mainly using JUnit 3, but could also extend testing to include HTTPUnit, DBUnit, and Apache Cactus for testing JSP pages, the database, and the server respectively. All of our team members have experience through Massey and other work using JUnit. It’s also very easy to use and there is plenty of support easily available through Massey, online, and other resources. JUnit is widely used in the community and easily portable. JUnit also follows our open-source approach. JUnit’s automated system allows for fast testing that can be done often, this fits into our agile methodology.

## 3.6 Web Framework

We have chosen to use Java Service Pages and servlets as our web framework. Most of our team members have experience through Massey and other work with this framework. Even though not everyone is totally familiar with this framework, there is still an underlying knowledge of how it works, and there is a lot of support through Massey and the internet on it. JSP is fast, portable, and reliable. It also supports our other technologies and fits in to our Model View Controller system.

We also considered ASP, Zend Framework, Djano, EPB, and Ruby on Rails.

## 3.7 Web server

We have chosen to use Tomcat as our web server. All of our team members have experience with Tomcat through papers taught at Massey. The fact that Tomcat is so easy to deploy is one of the main reasons we chose to use it. It is easy to install, update, and maintain. There is a wide range of support available online and through Massey for Tomcat. Tomcat is lightweight, portable, and fast; Tomcat also supports all of our technologies and is dynamic.

We also considered Apache, IIS, and Jetty.

## 3.8 Database

We chose to go with MySQL for our Database. Most of the team members have experience with MySQL through Massey. MySQL is easy to use, install, update, and maintain. There is a lot of support available for MySQL through Massey and other resources online. MySQL is fast, reliable, portable and freely available. It also supports SQL and has a standard interface.

We also considered Microsoft SQL Server, Oracle, Microsoft Access, PostgreSQL, and Sybase.

## 3.9 Build System

We chose to use ANT build scripts as our build system. All of our team members have experience using ANT through Massey. There is a lot of support available for ANT through Massey and other resources online. ANT is also fast and automated with follows our agile methodology.

We also considered MSBuild.

4. Project Life Cycle Methodologies

For this project the team have decided to use a design methodology mainly based on Scrum.

## 4.1 Meetings

Weekly team meetings will be held each Monday at 11am. During each meeting the members will summarise what they achieved in the previous week, what issues they had and explain what they will be working on next. Following this, each Tuesday at 3pm the team will meet with the project supervisor to update him on the team’s progress.

For the following week’s work the team leader (Adam) will assign the next round of tasks to the team members based either on relevant skills or randomly assignment if no one person is a better choice than another.

## 4.2 Sprints

As the team meets weekly to allocate tasks and detail progress, a weeklong sprint makes sense. Tasks that could take longer than a week to achieve should, where possible, be broken down into smaller sub tasks that can be finished within a sprint.

## 4.3 Spike Tests

Similar to SCRUM, we can spike test any new ideas or technologies for a day or two, no longer, to decide on the merits and either incorporate the technology into the project or discard the test. Given the week long sprints spike tests should ideally be held before the sprint that would use that technology.

## 4.4 Milestones

The project will follow the standard IID process. A working prototype should be the result of completing each milestone – full functionality isn't expected until the final completion date but the individual functions of the application will be built, tested and signed off in an incremental fashion.

Each milestone will focus on specific functions of the project, and can mostly be broken down into incremental plug-in development. That is, once the initial demonstration is finished, the project will follow a feature based development where each week the team will focus on extending the system to cope with more advanced plug-ins. This will focus on the most important, but also easiest to implement, first; then move on to the more advanced forms of gathering meta-data from the more obscure websites.

# 5. System Architecture

*Please refer to Appendix A, Figure 1.*

From the figure, you can see that there are several main features that encompass our system architecture. Beginning at the left, the user (or the internet they are using) is directly interacting with the various servlets. There is one servlets per Java Service Page, which is for each different page that the user sees, there is one servlet that acts as the interface between the user and the service. The servlets’ input is http; this is the input that the system will receive from the forms that the user is filling out. The servlets’ output is html to display results of the system working on the input. The JSPs format the output of the servlets into html.

The servlets interact with a singleton class called the Service. The fact that it is singleton means there exists only one instance of the class for the lifetime of the server. That allows the class to know the entire state of the application. All the servlets pass incoming form data to the Service, and display the results returned back from the Service, so all the main logic is done in this class. It calls on the other classes like the plug-in system, or the database interface to perform delegated tasks.

The various modules on the right of the diagram handle the different tasks delegated to them by the Service. The first of these modules is the Plug-in Factory. This acts as an interface between the Service and the multiple plugins to the system. The plugins are what process the input from the user and gather the meta-data. There will be a plugin for each of the resources we use to gather the meta-data, for example CrossRef, Scopus, BibTeX, etc. The Plug-in Factory will prioritise these plugins and step through them until sufficient meta-data is gathered.

The Database Interface is the next module interacting with the Service. This module is responsible for handling the queries to the local database. These queries will include saving to the database, searching the database, and exporting information from the database to CSV or XML.

The final module is the Email client. This module will handle requests to email the authors of the papers with the confirmation email. This will be an automated process.

# 6. Deployment Strategy

The deployment strategy focuses on how to deploy a JSP application but some thought is also given to other deployment strategies should they become relevant (e.g. The same web application could possibly be built to incorporate a web service and client that connects to it which would have a different deployment set up).

## 6.1 Deploying a JSP Application

Any JSP application is going to need a web server to run on. In the Massey environment that could possibly even include a server set up on the client's individual machine, but will most likely require being set up on an existing Massey server. Internal Massey policy will most likely have some impact on the final location and the final installation procedure.

A permanent location on a Massey server allows for easier deployment for other users if the application becomes more popular. An individual machine set up would need to be replicated on each user's computer, taking considerable time and making maintenance very difficult.

Deploying the actual files and directories that make up the application would be a simple matter of deploying as a WAR file (Web Archive) to the server or copying the file set up directly via FTP or similar.

One consideration is the client's desire to have an email function included. This would require an email server to be available to the application and, along with the difficulty of maintenance, is one of the primary reasons an individual set up is best avoided.

## 6.2 The Database

A locally accessed database is required for storing the entries to the RIMS system and supplementary tables such as Massey staff and emails.

Storing the database on a server along with the application has several advantages; namely, that the database can be used by multiple users very easily and is easier to update, backup and maintain. A database on the user's computer lacks these advantages and also has the additional issue of replication of data if other users have their own databases set up. The local database needs to be able to avoid repetition of work which it would be unable to do if several copies of the same database existed.

Relocation of the database after installation can be managed by altering the web.xml file which contains the connection information.

# 7. Risk Management

# 8. Issue Tracking Policy

Throughout the sprints when issues arise, they are to be dealt with immediately. These issues can include impediments discovered during the weekly meetings, bugs, or change requests. When dealing with bugs and other high priority issues, they are to be resolved before the feature they are associated with is signed off.

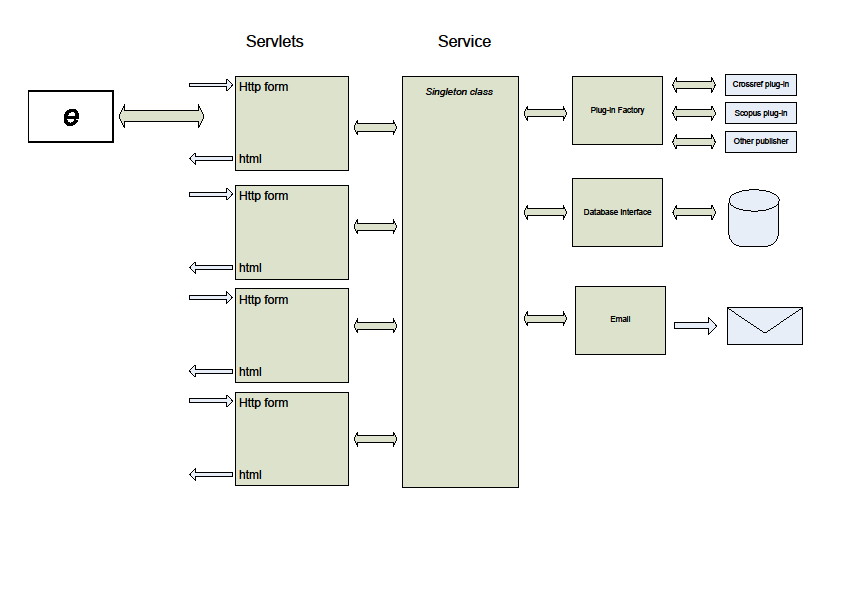
8.1 Issue Tracking Workflow

* Open a new issue through the Google code project page
* Fill out the summary and description of the issue.
* Cc to relevant team members.
* Make sure the issue type and priority are set accordingly.
* Submit the issue.   
    
  *After the issue has been submitted:*
* I (Adam) will assign the issue to a team member based on the workload of the team members at the time.  
    
  *After the issue has been resolved:*
* The two team members that have not assigned or submitted the issue will then sign the issue off.
* Issue page is then updated.

# 9. Testing and Metrics

# 10. Appendices

## 10.1 Appendix A



*Figure 1: High level system architecture.*

## 10.2 Appendix B