Software Engineering C

Week 8 Group Report

Project Leader:

Adam Bramley

Team Members:

Stephen Allan

Leith Bade

Peter Williamson

Table of Contents

1. Introduction

Please refer to Appendix E for a snapshot of the project plan.

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.

1.1 Team Contribution

At the beginning of the project our team discussed the various skill sets that each person had, from this we decided on the different threads that each member would mainly work on. As the team leader, Adam is in charge of delegating work, arranging meetings with the project manager, client, and team. Along with this, Adam has been focussing for the most part on designing the interface for the system and coding the jsp files up. This included writing any Javascript and CSS as well as the obvious HTML. Peter has been in charge of the database and all things related to the database. He has designed the DB from the start including the structure and all scripts required for creating, updating, and retrieving information in the DB. Steve has been working with all the APIs for the various meta-data gathering websites (CrossRef and Scopus so far) and extracting the information out of those, along with all searching functionality of the program. Finally, Leith has been in charge of structuring the project in regards to packages, classes, builds, interfaces, and plugins.

1.1.1 Report Contribution

This report was written by the team as a whole, with each part delegated to a different team member. Along with this, Adam was responsible for editing this information given by each team member and formatting the entire report. The following list shows the section numbers each team member was responsible for:

Adam: 1, 2, 5, 8
Peter: 7
Leith: 3, 9
Steve: 6, 4

Please note, section 3 was provided in a table of checklists which Adam converted to paragraph form. Section 10 was contributed to by all team members equally.

1.2 End User Interactions

During week 4-8 of our project, we have had several meetings with Karen Sutton, including one with Craig Manning. During these meetings our team has mainly discussed the project requirements with them. We do this by asking questions around what has already been implemented in the project and matching it to requirements gathered in week 2 of the project. Almost every time we have met with them there has been a change to the requirements, this is due to the massive difference in the expectations of the product between Craig and Karen. Craig seems to want the product to benefit him in some way, for example wanting us to export our database files to XML and CSV when there is really no reason for it. He also lead us quite astray with what was actually needed by the system, we made our database far too complex and included the unnecessary emailing due to his suggestions. The most beneficial interactions we have had is our meeting with Karen herself where we could specifically ask her questions about things like the process she goes through with RIMS at the moment, and how she would like the application to run through.

For usability testing we intend to set the application in front of another person that will be using the it in the future, and ask them to try to go through a series of steps to get a desired output. Preferably, this person will not be Karen as she has seen the program in use a few times already and will have the general idea of how it works. If we test it on some people that have never seen it before, we will get an idea of how usable the product is.

1.3 Project Obstacles

There are a few obstacles the project faces before completion in week 12. First off, our code has very weak test coverage (only 5% of all classes) this poses the huge threat of bugs to our system, although there have been no severe “bugs” found as of yet, the threat is still there. Furthermore, some of our packages are very instable meaning they aren’t at all resilient to change. There are also around 150 problems in the PMD report, but most of these are very minor and won’t take much to fix at all.

The other obstacles of the project relate to the goals we have made for week 12. We still have a few things to implement in our system, and getting them all done in 3 weeks might be a stretch. We still need to fully integrate the database into the system; this was put off as we drastically changed the structure of the database recently. The code for gathering data from Scopus has been written, but it is yet to be integrated into our plugin system. The final part of our system that needs to be coded is the printing of the document after submission.

1.4 Project Plan

Please refer to Appendix E for a snapshot of the project plan and progress.

As you can see from the snapshot, we are on schedule for most things. The first thing that is behind schedule is the remodelling of the architecture. This involves changing the singleton class of the Service having one per session to mitigate any concurrency issues we would get if the Service was singleton. The reason we are behind on this is that during the study break, there was almost no contact made by Leith including the week before the study break. This meant he did almost no work for 3 weeks and put that part of the project well behind. However, we are working hard to catch up this week and next and should still be on track to achieve our goals in week 12. Secondly, the database committing and retrieving is a bit behind. This is due to having to restructure our database after realising some information stored in there would be rather redundant. The scripts for inserting and retrieving from the database have been written, they just have not been tied into the system. Finally, the support for multiple authors is a bit behind. While multiple authors are gathered and displayed we intend to separate them in to different text boxes with their own affiliation box each. This is behind due to the increased workload this week on Adam with the report writing. But since he has finished the design of the interface, this will not hinder the completion of the project.

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. These requirements were further refined after receiving feedback from our project manager.

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. The meta-data output includes: authors, article title, publication year, publisher, abstract, keywords, URL, and possibly page and volume numbers. If we are able to, we will also include author affiliations.
2. If the system cannot extract any meta-data, the raw article is to be produced.
3. The system has to display the meta-data on screen and allow the user to edit the data.
4. Access to a local database is needed. The author details and DOIs are stored in the database so author details can be automatically generated for known authors, and the system can check for duplicate DOIs.
5. The system needs to be able to print the cover sheet containing all the metadata.
6. 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 when given the author name, article title, and publication date.
7. 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 is to return local database results in under 3 seconds and web results in under 15 seconds per plugin.
3. The system needs to handle plug-in architecture
4. The system needs to have an automated deployment and maintenance procedure.
5. The system must have support for access to the internet and local database through a proxy.

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

2.3 Summary of changes

These requirements were revised again with Karen on 8th September, and thought was given to the feedback received from the project managers. We have decided to, for the mean time, not worry about scrapping the page for the author's affiliations. This is because it seems like a very difficult task with the thousands of different styled pages for the various publications. There are various ways the affiliation is displayed: Sometimes it is shown after each author in simple text, other times it is referenced by numerous characters in a superscript next to the author's name. Instead, we will store the affiliation of the authors in the database, so that once it is entered; it can automatically be retrieved the next time the author is referenced. The user will also be able to edit the affiliation in case authors change university for example.

Another feature we have removed is the verification email. After reassessing the process that Karen goes through to send the emails, it seems to be easier for her to do it herself since she doesn't email the author after every submission; it seems to be on a basis of a certain number of submissions of that particular author.

Furthermore, we have simplified our database to no longer include publication details (apart from the DOI). We first thought we needed this detail so that the user could modify the entries through our system when an author finds something wrong with a submission. We now see that this is completely redundant because the only place it is modified is directly through the RIMS system.

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.

	Taught at Massey	**Group experience**	**Easy to use**	**Technical expertise available**	**Resources at library**
Java	Y	Y (4)	Y	Y	Y
C#	Y	Y (2)	Y	N	Y
Python	N	Y (1)	N	N	Y
Ruby	N	N		N	Y
PHP	N	Y (1)	N	N	Y

	Good support	**Reliable**	**Widely used**	**Portable**	**Freely available**
Java	Y	Y	Y	Y	Y
C#	Y	Y	Y	N	Y
Python	Y	Y	Y	Y	Y
Ruby	Y	Y	Y	Y	Y
PHP	Y	Y	Y	Y	Y

	OOP	**Testable**	**Maintainable**	**Good IDE**	**Plugins**
Java	Y	Y	Y	Y	Y
C#	Y	Y	Y	Y	N
Python	N	Y	Y	N	N
Ruby	Y	Y	N	N	N
PHP	N	N	N	N	N

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.

	Taught at Massey	Group experience	Easy to use	Resources at library	Good support
Eclipse	Y	Y (4)	Y	Y	Y
NetBeans	Y	Y (3)	N	Y	Y
Visual Studio	Y	Y (2)	Y	Y	Y
SharpDevelop	N	N	N	N	N
MonoDevelop	N	N	N	N	N

	Reliable	Widely used	Portable	Freely available	Source control
Eclipse	Y	Y	Y	Y	Y
NetBeans	Y	Y	Y	Y	Y
Visual Studio	Y	Y	N	Y	Y
SharpDevelop	Y	N	Y	Y	Y
MonoDevelop	N	N	Y	Y	Y

	Refactoring	Testing	Language
Eclipse	Y	Y	Java
NetBeans	Y	Y	Java
Visual Studio	Y	Y	C#
SharpDevelop	Y	Y	C#
MonoDevelop	Y	Y	C#

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.

	Group experience	**Easy to maintain**	**Easy to update**	**Easy to install**
Desktop App	Y (4)	Y	N	Y
Client/Server	Y (1)	N	N	N
Web App	Y (3)	Y	Y	N

	Scalable	**Only need 1 computer**	**Need network connection**
Desktop App	N	Y	N
Client/Server	Y	N	Y
Web App	Y	Y	Y

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.

	Group experience	**Easy to use**	**Technical expertise available**	**Resources at library**
Guice	N	Y	N	N
OSGI	N	N	Y	Y
Eclipse Equinox	N	N	N	Y
Java Plugin Framework	N	N	N	N
ServiceLoader	N	Y	N	N
Spring Framework	N	N	N	N

	Good support	**Reliable**	**Widely used**	**Portable**
Guice	N	Y	N	Y
OSGI	Y	Y	Y	Y
Eclipse Equinox	Y	Y	Y	Y
Java Plugin Framework	Y	Y	N	Y
ServiceLoader	Y	Y	Y	Y
Spring Framework	Y	Y	Y	Y

	Freely available	**Runtime**	**Type safe**	**Interface**
Guice	Y	Y	Y	Y
OSGI	Y	Y	Y	Y
Eclipse Equinox	Y	Y	Y	Y
Java Plugin Framework	Y	Y	Y	Y
ServiceLoader	Y	Y	Y	Y
Spring Framework	Y	Y	Y	Y

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.

	Group experience	Easy to use	Technical expertise available	Resources at library	Good support	Reliable	Widely used	Portable	Freely available	Automated	Fast
Junit 3	Y (4)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Nunit	N	Y	N	N	Y	Y	Y	N	Y	Y	Y

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.

	Taught at Massey	**Group experience**	**Technical expertise available**	**Resources at library**
ASP	N	Y (1)	N	Y
Zend Framework	N	Y (1)	N	Y
Djano	N	N	N	N
JSP/Servlets	Y	Y (3)	Y	Y
EPB	Y	Y (3)	Y	Y
Ruby on Rails	N	N	N	Y

	Good support	**Reliable**	**Widely used**	**Portable**
ASP	Y	Y	Y	N
Zend Framework	Y	Y	Y	Y
Djano	Y	Y	Y	Y
JSP/Servlets	Y	Y	Y	Y
EPB	Y	Y	Y	Y
Ruby on Rails	Y	Y	Y	Y

	Freely available	**Fast**	**MVC**	**HTML**
ASP	Y	Y	Y	Y
Zend Framework	Y	Y	N	Y
Djano	Y	Y	Y	Y
JSP/Servlets	Y	Y	Y	Y
EPB	Y	Y	Y	Y
Ruby on Rails	Y	Y	Y	Y

	Other file types
ASP	Y
Zend Framework	Y
Djano	Y
JSP/Servlets	Y
EPB	Y
Ruby on Rails	Y

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.

	Group experience	**Easy to maintain**	**Easy to update**	**Easy to install**
Apache	Y (2)	N	N	N
IIS	Y (2)	N	N	N
Tomcat	Y (4)	Y	Y	Y
Jetty	N	Y	Y	Y

	Easy to use	**Technical expertise available**	**Resources at library**	**Good support**
Apache	Y	N	Y	Y
IIS	N	N	Y	Y
Tomcat	N	Y	Y	Y
Jetty	Y	N	N	Y

	Reliable	**Widely used**	**Portable**	**Freely available**
Apache	Y	Y	Y	Y
IIS	Y	Y	N	Y
Tomcat	Y	Y	Y	Y
Jetty	Y	Y	Y	Y

	Fast	**Lightweight**	**Doesn't need a server**	**Dynamic**
Apache	Y	N	Y	Y
IIS	Y	N	Y	Y
Tomcat	Y	Y	Y	Y
Jetty	Y	Y	Y	Y

	HTML	**Other file types**
Apache	Y	Y
IIS	Y	Y
Tomcat	Y	Y
Jetty	Y	Y

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.

	Group experience	**Easy to maintain**	**Easy to update**	**Easy to install**
Microsoft SQL Server	Y (2)	Y	Y	Y
Oracle	Y (3)	N	N	N
MySQL	Y (2)	Y	Y	Y
Access	Y (1)	Y	N	Y
PostgreSQL	N	Y	Y	Y
Sybase	N	N	N	N

	Easy to use	**Technical expertise available**	**Resources at library**	**Good support**
Microsoft SQL Server	Y	N	Y	Y
Oracle	N	Y	Y	Y
MySQL	Y	Y	Y	Y
Access	Y	N	Y	N
PostgreSQL	Y	N	N	Y
Sybase	N	N	N	Y

	Reliable	**Widely used**	**Portable**	**Freely available**
Microsoft SQL Server	Y	Y	N	Y
Oracle	Y	Y	Y	N
MySQL	Y	Y	Y	Y
Access	N	Y	N	N
PostgreSQL	Y	Y	Y	Y
Sybase	Y	Y	Y	N

	Fast	**Lightweight**	**Doesn't need a server**	**SQL**
Microsoft SQL Server	Y	N	Y	Y
Oracle	Y	N	N	Y
MySQL	Y	Y	Y	Y
Access	Y	N	Y	Y
PostgreSQL	Y	Y	Y	Y
Sybase	Y	N	N	Y

	Standard interface
Microsoft SQL Server	Y
Oracle	Y
MySQL	Y
Access	Y
PostgreSQL	Y
Sybase	Y

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.

	Group experience	Easy to use	Technical expertise available	Resources at library	Good support	Reliable	Widely used	Portable	Freely available	Automated	Fast
MSBuild	N	Y	N	N	Y	Y	Y	N	Y	Y	Y
ANT	Y (4)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

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.

Our system is quite different from the previous report so most of the following are changes.

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, 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 first JSP the user interacts with in the application is the index page. This is where the user enters the DOI in. When the user submits the DOI it is sent to an object called the DOIRequest (or AdvancedRequest if the user is using the advanced search). The DOI request then creates an instance of the Service class. This Service class acts as a "service" to the servlets by interacting with the plugins and database. The Service then sends the DOI to the MetaDataRetreiverFactory. The class acts as a Factory for the different plugins, these plugins are the different websites we are using to gather the meta-data. Each plugin is tested to return a publication and when one is found the Service will receive it from the Factory. The plugins work by using the sites' APIs and filling the Publication object with information gathered through XML.

Once the Service receives a Publication object, it adds it to the session and is then displayed on the results page. There are 3 separate results pages linked to each of the 3 publication types (book, conference, and journal). Once the user has finished editing the results and confirms the publication is then updated in the session and sent to the service. From there some details will be sent to the database, and a form will be printed if the user has chosen to.

The database is used to store dois as an extra safeguard against duplication. Along with this, author name, id, and affiliation will be stored in the database so that once the user has entered an affiliation for an author once, it will be automatically generated for them next time. There is a small possibility of the database returning the wrong affiliation, but our client has specified that as long as it is editable it will still be very helpful.

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 the author details and DOIs of articles that have already been entered. This is a change from our previous database as we are no longer storing all the publication data. We saw this as completely redundant; there is no need for the publication data to be stored apart from the DOI.

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

Please refer to Appendix B.

At the beginning of the project, several risks were identified with varying degrees of likelihood and impact on the project.

If the Team Leader puts too much workload on team members for a week this can lead to increased stress and unreasonable expectations, similarly if too little is assigned, then it can put the project behind schedule. Furthermore, if the Team leader leads the group away from the scope of the project then it can have dire consequences that the project will not be valid.

If the Team manager does not give the guidance that is required by the team, it could be potentially harmful towards the project. Misunderstanding Scope is another potential risk and is the most common cause of failure for software projects.

If a group member fails to produce the workload that is assigned to them for the week, it can have an effect on the flow of the project, putting the groups schedule behind and puts increased pressure on next week.

Sickness is another risk on the team. If a member falls ill work may not be completed to the deadline, or a substandard effort may be made. There is also a small risk a member of the team drops out of the paper, this will put all workload done by the group member onto the other members of the group and potentially hinder the quality of work.

There is a risk of feature creep to the project. If features keep being added to the project then it has the impact that we do not get the most important feature working. This ties into another risk of changing requirements, this could essentially mean all the previous work done to date, would be invalid

There is also the risk of team members not having enough experience and skill in specific parts of the project. If we are unable to accomplish something it may delay the schedule, and extra resources may have to be placed in solving the issues.

7.1 Risks that have been added since previous report:

The modifications to the previous risk management are highlighted in **bold**. Lack of communication and loss of communication between client/server have been added as risks to the project. Running out of time is the biggest risk that the project faces now, with only four weeks remaining. Implementing new features has an even greater impact now due to the lack of time remaining.

The solution of a client/server application poses extra risks that must be addressed. These include server connectivity, accessibility, and permissions. The risk of the client having insufficient permissions for access to the server is almost non-existent as there is no security on the application. On the other hand, accessibility and connectivity to the server is crucial for the application to run at all, and will have massive impacts if either is interrupted.

8. Issue Tracking Policy

Since the previous report, our team has decided to take a more relaxed approach to issue tracking. We have chosen to do this because we noticed all "issues" that were arising were quite simple issues that could simply be fixed within hours. Our team sees it as a burden to have to go through the process of posting it on the project site and simply unnecessary. We have decided to only post issues that we think either we cannot fix ourselves, or will take longer than 3 hours to complete. The following issue tracking workflow is for issues that need to be posted on the project website.

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.

If an issue is open for more than 7 days, the team will have an extra meeting discussing the issue until it is resolved. In most cases, we will be able to resolve the issue with all team members discussing it. If the issue is still unable to be solved then further investigation will take place by a specific team member. If the particular issue is due to an extra feature, then that feature may be removed.

9. Testing and Metrics

Please refer to Appendices F, G, and H for the reports.

For testing our system and generating metrics we will utilise as many automated systems as possible. For testing we have decided to use Junit 3. This allows us to frequently run tests on our code to test for any new bugs that may have been introduced from freshly written code or refactoring. As part of our testing policy, each time a new issue is submitted a new test case will be added to account for the newly introduced bug. This will ensure the system is safe from the bug and allow us to change our code to suit it.

To automate metrics, we will be using EMMA, PMD, and JDepend to gather information on test coverage, code complexity, and dependencies. This will give us a good range of metrics to ensure our code is up to standard. To generate the reports we are using an ANT build script. This puts all the results in html files inside the resulting build file.

There is a major problem with the EMMA reports as they are giving very low results; it seems to be something to do with running the test cases through the build script on a server. The results are included but are not accurate.

We have also included a metric for the team's participation. This is based on the percentage of commits to the repository each member has done. Keep in mind even though a percentage may be higher, different tasks require more or less work and in turn require more or less commits.

Total commits from 4/08/10 to 10/09/10:

Adam: 40%	Leith: 26%	Peter: 24%	Steve: 10%

The data in this time period is in the time since the previous report was submitted.

9.1 Compatibility testing

All testing was done using Windows XP, this is because the only platform we expect the admin at Massey to be using is Windows and since it's a web application the compatibility relies more on the browser.

Browser	**Comments**
Mozilla 3.6.8	Completely functional, all styles are showing along with AJAX animation. Meta-data gathering working as expected.
Internet Explorer 6	Something is very wrong here. Even the button to send the DOI away isn't showing. Curved borders also aren't showing but that was expected. We are not worried about this as the users are very unlikely to be using IE6.
Google Chrome 6.0.4	Completely functional, all styles are showing along with AJAX animation. Meta-data gathering working as expected.
Opera 10.62	Completely functional, all styles are showing along with AJAX animation. Meta-data gathering working as expected.

10. Appendices

10.1 Appendix A – System Architecture

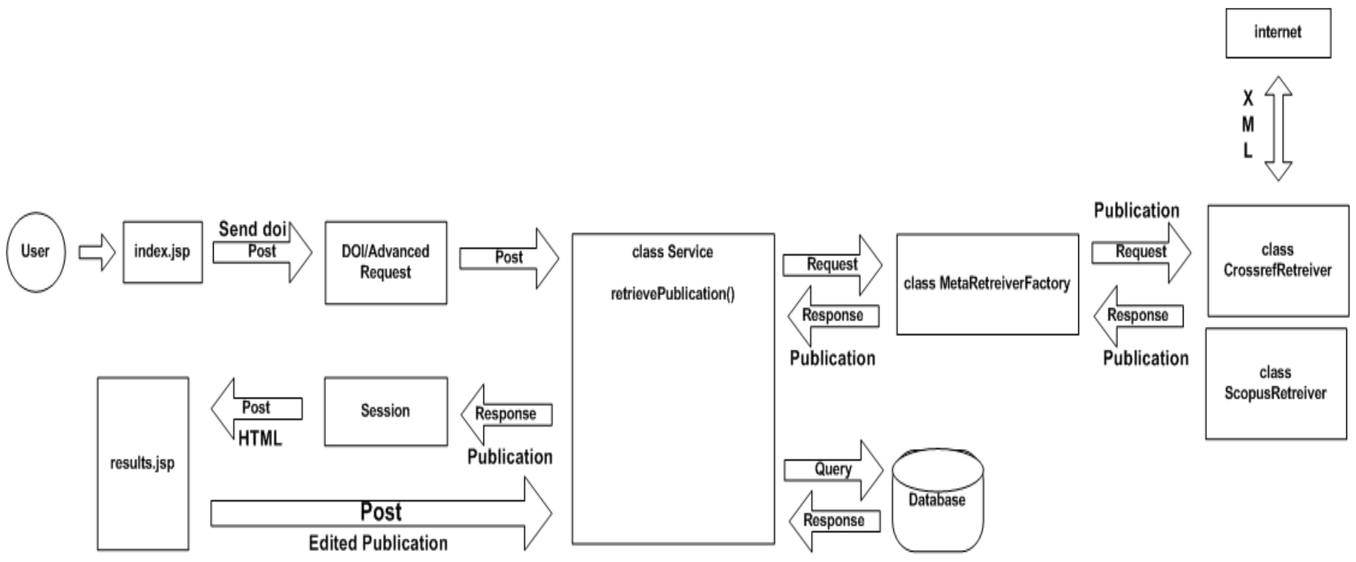


Figure 1. Improved System Architecture

10.2 Appendix B – Risk Analysis

Risk:	Team leader assigning workload improperly
Likelihood of Occurrence:	High
Potential Impact:	High. If the Team Leader puts to much workload on for a week this can lead to increased stress and unreasonable expectations, similarly if too little is assigned, then it can put the project behind schedule.
Mitigation plan:	Communication to the leader in the group meetings of what members are likely to accomplish and that they feel is the correct amount of work.

Risk:	Team leader leading group astray
Likelihood of Occurrence:	Medium
Potential Impact:	High. If the Team leader leads the group away from the scope of the project then it can have dire consequences that the project will not be valid.
Mitigation plan:	Communication to the leader and between members in the group to ensure that the team stays on the target and does not get sidetracked.

Risk:	Team manager
Likelihood of Occurrence:	Medium
Potential Impact:	Medium. If the Team manager does not give the guidance that is required by the team, it could be potentially harmful towards the project
Mitigation plan:	There is no way to prevent this from happening, although the weekly meetings should keep this risk under control

Risk:	Misunderstanding Scope
Likelihood of Occurrence:	Low
Potential Impact:	High. Misunderstanding Scope is the most common cause of failure for software projects.
Mitigation plan:	Constantly talking to the client and informing them of progress, checking what has been completed meets their requirements.

Risk:	Group member doesn't complete a weekly workload
Likelihood of Occurrence:	High
Potential Impact:	High. If a group member fails to produce the workload that is assigned to them it can have an effect on the flow of the weeks work by putting the groups schedule behind and puts increased pressure on next week.
Mitigation plan:	Regular meetings to ensure that the workload is on schedule and that if a member is unlikely to complete the task, steps can be taken to help the burden or alter the plan. Communication from the member in question is key.

Risk:	Sickness
Likelihood of Occurrence:	High
Potential Impact:	Medium. Work may not be completed to the deadline, or a substandard effort may be made
Mitigation plan:	Staying at home and not transferring the sickness to other group members is preferable. Communication via electronic means rather than face to face in order to stay in touch

Risk:	Group member resigns from Software C
Likelihood of Occurrence:	**Extremely Low**
Potential Impact:	High. This will put all workload done by the group member onto the other members of the group
Mitigation plan:	Communication between members is essential, and encouragement to stick with the project if a member considers quitting. However it is better they quit, than stay on and not do the workload

Risk:	Loss of Code/Documentation
Likelihood of Occurrence:	Extremely Low
Potential Impact:	High. All the work that is done will be lost
Mitigation plan:	The Repository is set up to prevent this from happening, and will ensure that nothing is lost

Risk:	Feature Creep
Likelihood of Occurrence:	Medium
Potential Impact:	**High.** If features keep being added to the project then it has the impact that we do not get the most important feature working
Mitigation plan:	By prioritising and evaluating what features are important and those that would be nice but are not necessary

Risk:	Changing Requirements
Likelihood of Occurrence:	**Low**
Potential Impact:	High. The Changing of requirements could essentially mean all the previous work done to date, would be invalid
Mitigation plan:	Constant communication with the client in order to catch any requirements that change early, so the project schedule suffers as little as possible

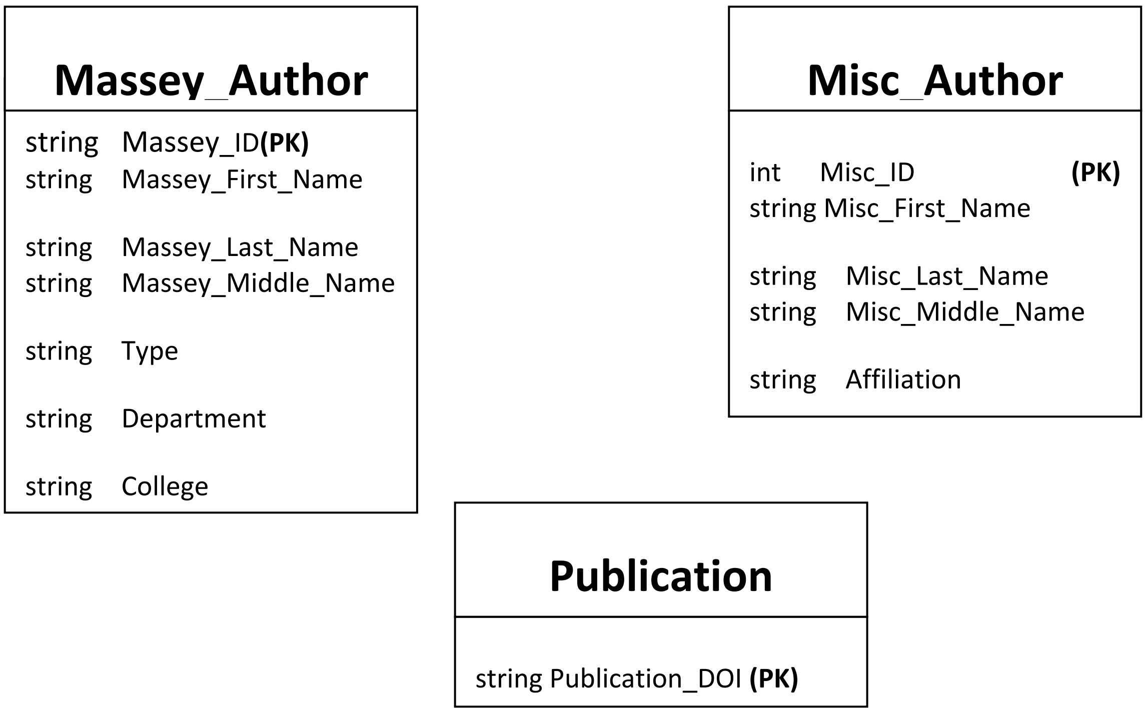
Risk:	Running out of time
Likelihood of Occurrence:	**Extremely High**
Potential Impact:	High. 14 weeks is all the time we get, everything needs to be up and running up then
Mitigation plan:	Proper planning and communication to prevent any bottle necks from occurring. Prioritising requirements which are essential will produce a working project

Risk:	Technological Issues/Lack of skills
Likelihood of Occurrence:	Medium
Potential Impact:	High. If we are unable to accomplish something it may delay the schedule, and extra resources may have to be placed in solving the issues
Mitigation plan:	A technological review and the team leader knowing the skills of individual group members will try prevent this from happening, though it may be unavoidable

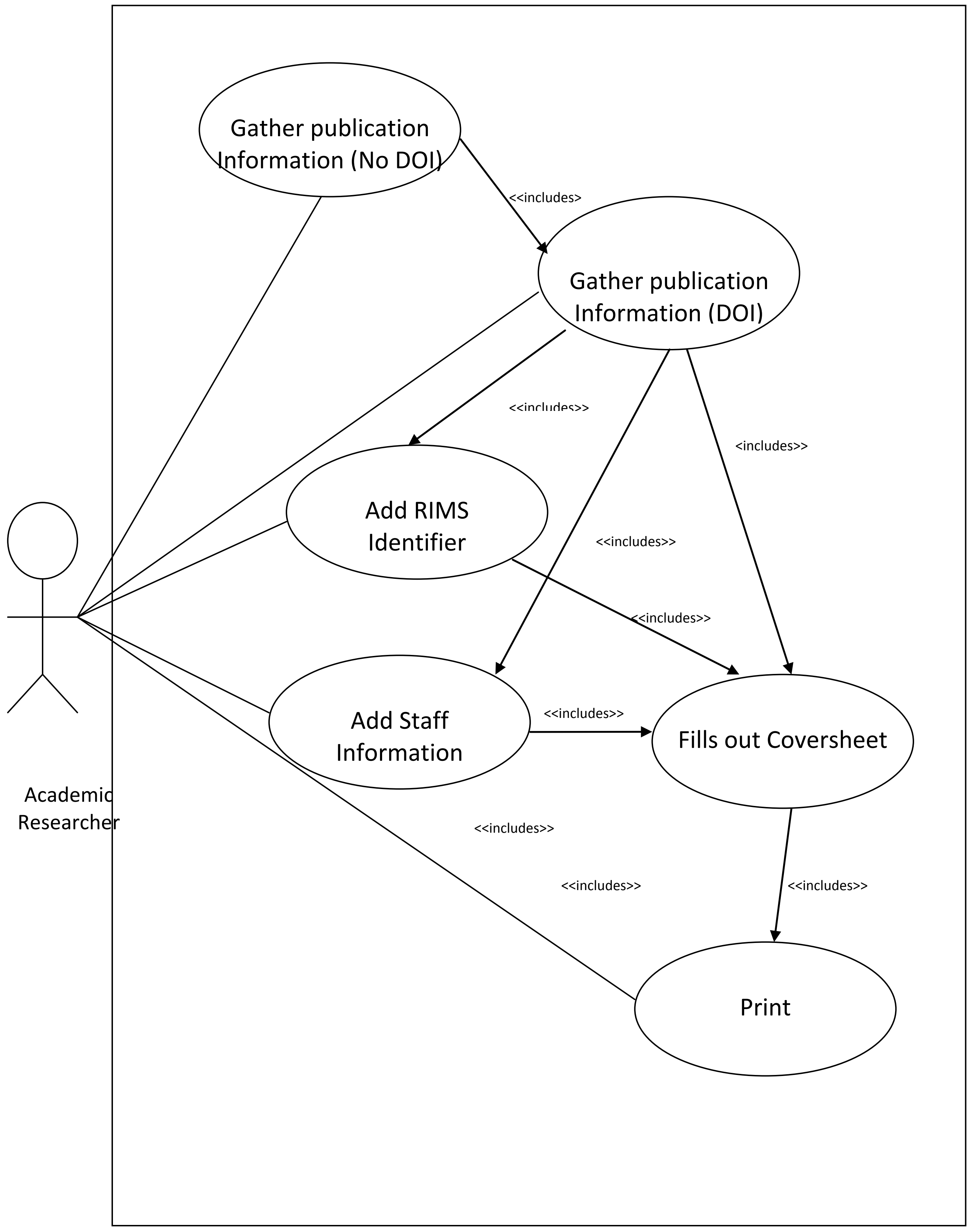
Risk:	**Lack of Communication**
Likelihood of Occurrence:	**Medium**
Potential Impact:	**High. Lack of communication the group may be behind schedule, and may cause duplication of workload.**
Mitigation plan:	**Schedule regular meetings that everybody has to show up to. Talking outside the meetings will also help alleviate the problem.**

Risk:	**Loss of client/server connection or access**
Likelihood of Occurrence:	**Low**
Potential Impact:	**High. If the client loses access or connection to the server the application will not be able to function at all.**
Mitigation plan:	**Make sure server is secure**

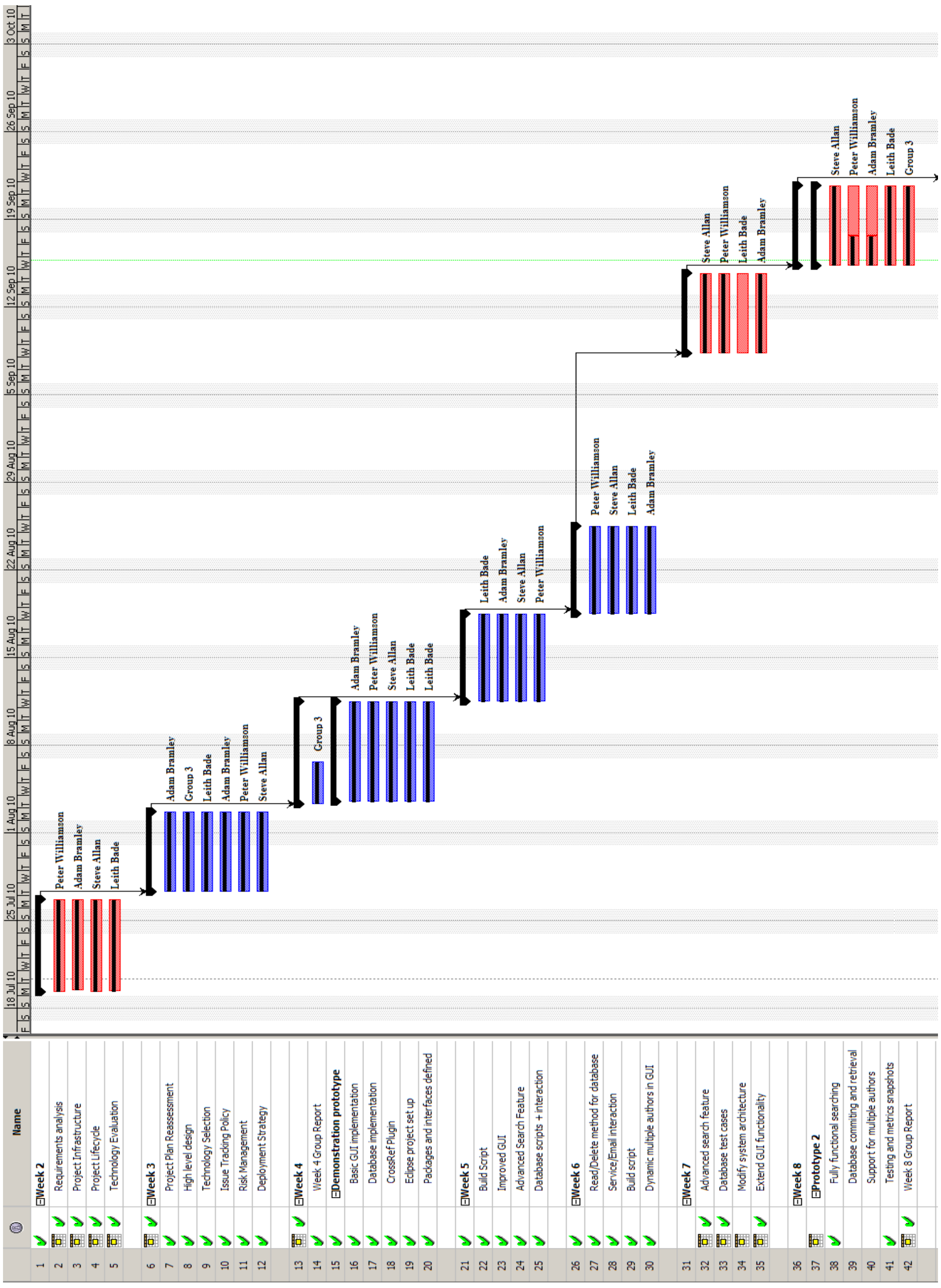
10.3 Appendix C – ER Diagram



10.4 Appendix D – Use Case Diagram



10.5 Appendix E – Gantt chart



	Name
1	Week 2
2	Requirements analysis
3	Project Infrastructure
4	Project Lifecycle
5	Technology Evaluation
6	Week 3
7	Project Plan Reassessment
8	High level design
9	Technology Selection
10	Issue Tracking Policy
11	Risk Management
12	Deployment Strategy
13	Week 4
14	Week 4 Group Report
15	Demonstration prototype
16	Basic GUI implementation
17	Database implementation
18	CrossRef Plugin
19	Eclipse project set up
20	Packages and interfaces defined
21	Week 5
22	Build Script
23	Improved GUI
24	Advanced Search Feature
25	Database scripts + interaction
26	Week 6
27	Read/Delete method for database
28	Service/Email interaction
29	Build script
30	Dynamic multiple authors in GUI
31	Week 7
32	Advanced search feature
33	Database test cases
34	Modify system architecture
35	Extend GUI functionality
36	Week 8
37	Prototype 2
38	Fully functional searching
39	Database commiting and retrieval
40	Support for multiple authors
41	Testing and metrics snapshots
42	Week 8 Group Report

10.6 Appendix F – JDepend Report

This is a report generated using JDepend:

]

10.6.1 Summary

Package	Total Classes	Abstract Classes	Concrete Classes	Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
nz.ac.massey.rimsgroup3	1	0	1	1	2	0	0.67	0.33
nz.ac.massey.rimsgroup3.DB.test	1	0	1	0	2	0	1	0
nz.ac.massey.rimsgroup3.database	7	0	7	1	1	0	0.5	0.5
nz.ac.massey.rimsgroup3.metadata	3	1	2	3	1	0.33	0.25	0.42
nz.ac.massey.rimsgroup3.metadata.bean	8	0	8	8	0	0	0	1
nz.ac.massey.rimsgroup3.metadata.plugin	2	0	2	1	2	0	0.67	0.33
nz.ac.massey.rimsgroup3.metadata.plugin.test	1	0	1	0	2	0	1	0
nz.ac.massey.rimsgroup3.metadata.test	1	0	1	0	2	0	1	0
nz.ac.massey.rimsgroup3.servlet	2	0	2	0	2	0	1	0

10.6.2 Packages:

nz.ac.massey.rimsgroup3

Afferent Couplings: 1   Efferent Couplings: 2   Abstractness: 0   Instability: 0.67   Distance: 0.33

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.Service	nz.ac.massey.rimsgroup3.servlet	nz.ac.massey.rimsgroup3.metadata nz.ac.massey.rimsgroup3.metadata.bean

nz.ac.massey.rimsgroup3.DB.test

Afferent Couplings: 0   Efferent Couplings: 2   Abstractness: 0   Instability: 1   Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.DB.test.TestingDatabase	*None*	nz.ac.massey.rimsgroup3.database nz.ac.massey.rimsgroup3.metadata.bean

nz.ac.massey.rimsgroup3.database

Afferent Couplings: 1 Efferent Couplings: 1 Abstractness: 0 Instability: 0.5 Distance: 0.5

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.database.DatabaseConnection nz.ac.massey.rimsgroup3.database.DatabaseInsert nz.ac.massey.rimsgroup3.database.DatabaseModify nz.ac.massey.rimsgroup3.database.DatabaseRead nz.ac.massey.rimsgroup3.database.InsertStatements nz.ac.massey.rimsgroup3.database.ModifyStatements nz.ac.massey.rimsgroup3.database.ReadStatements	nz.ac.massey.rimsgroup3.DB.test	nz.ac.massey.rimsgroup3.metadata.bean

nz.ac.massey.rimsgroup3.metadata

Afferent Couplings: 3 Efferent Couplings: 1 Abstractness: 0.33 Instability: 0.25 Distance: 0.42

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
nz.ac.massey.rimsgroup3.metadata.MetadataRetriever	nz.ac.massey.rimsgroup3.metadata.DOINotFoundException nz.ac.massey.rimsgroup3.metadata.MetadataRetrieverFactory	nz.ac.massey.rimsgroup3 nz.ac.massey.rimsgroup3.metadata.plugin nz.ac.massey.rimsgroup3.metadata.test	nz.ac.massey.rimsgroup3.metadata.bean

nz.ac.massey.rimsgroup3.metadata.bean

Afferent Couplings: 8 Efferent Couplings: 0 Abstractness: 0 Instability: 0 Distance: 1

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.metadata.bean.Author nz.ac.massey.rimsgroup3.metadata.bean.Book nz.ac.massey.rimsgroup3.metadata.bean.Conference nz.ac.massey.rimsgroup3.metadata.bean.Editor nz.ac.massey.rimsgroup3.metadata.bean.Information nz.ac.massey.rimsgroup3.metadata.bean.Journal nz.ac.massey.rimsgroup3.metadata.bean.Person nz.ac.massey.rimsgroup3.metadata.bean.Publication	nz.ac.massey.rimsgroup3 nz.ac.massey.rimsgroup3.DB.test nz.ac.massey.rimsgroup3.database nz.ac.massey.rimsgroup3.metadata nz.ac.massey.rimsgroup3.metadata.plugin nz.ac.massey.rimsgroup3.metadata.plugin.test nz.ac.massey.rimsgroup3.metadata.test nz.ac.massey.rimsgroup3.servlet	*None*

nz.ac.massey.rimsgroup3.metadata.plugin

Afferent Couplings: 1 Efferent Couplings: 2 Abstractness: 0 Instability: 0.67 Distance: 0.33

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.metadata.plugin.CrossrefRetriever nz.ac.massey.rimsgroup3.metadata.plugin.ScopusRetriever	nz.ac.massey.rimsgroup3.metadata.plugin.test	nz.ac.massey.rimsgroup3.metadata nz.ac.massey.rimsgroup3.metadata.bean

nz.ac.massey.rimsgroup3.metadata.plugin.test

Afferent Couplings: 0 Efferent Couplings: 2 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.metadata.plugin.test.CrossrefRetrieverTest	*None*	nz.ac.massey.rimsgroup3.metadata.bean nz.ac.massey.rimsgroup3.metadata.plugin

nz.ac.massey.rimsgroup3.metadata.test

Afferent Couplings: 0 Efferent Couplings: 2 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.metadata.test.MetadataRetrieverFactoryTest	*None*	nz.ac.massey.rimsgroup3.metadata nz.ac.massey.rimsgroup3.metadata.bean

nz.ac.massey.rimsgroup3.servlet

Afferent Couplings: 0 Efferent Couplings: 2 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	nz.ac.massey.rimsgroup3.servlet.AdvancedRequest nz.ac.massey.rimsgroup3.servlet.DoiRequest	*None*	nz.ac.massey.rimsgroup3 nz.ac.massey.rimsgroup3.metadata.bean

10.6.3 Cycles

]

There are no cyclic dependancies.

10.7 Appendix G – Emma Report

OVERALL COVERAGE SUMMARY

name	class, %	method, %	block, %	line, %
all classes	5% (1/22)	2% (3/162)	1% (24/2620)	1% (7.4/749)

OVERALL STATS SUMMARY

total packages:	6
total executable files:	22
total classes:	22
total methods:	162
total executable lines:	749

COVERAGE BREAKDOWN BY PACKAGE

name	class, %	method, %	block, %	line, %
nz.ac.massey.rimsgroup3	0% (0/1)	0% (0/3)	0% (0/19)	0% (0/7)
nz.ac.massey.rimsgroup3.database	0% (0/7)	0% (0/37)	0% (0/1557)	0% (0/419)
nz.ac.massey.rimsgroup3.metadata.bean	0% (0/8)	0% (0/89)	0% (0/351)	0% (0/132)
nz.ac.massey.rimsgroup3.metadata.plugin	0% (0/2)	0% (0/16)	0% (0/331)	0% (0/98)
nz.ac.massey.rimsgroup3.servlet	0% (0/2)	0% (0/12)	0% (0/307)	0% (0/77)
nz.ac.massey.rimsgroup3.metadata	50% (1/2)	60% (3/5)	44% (24/55)	46% (7.4/16)

10.8 Appendix H – PMD Report

The following is a report generated by PMD

#	File	Line	Problem
1	nz\ac\massey\rimsgroup3\Service.java	13	A class which only has private constructors should be final
2	nz\ac\massey\rimsgroup3\Service.java	16	Found non-transient, non-static member. Please mark as transient or provide accessors.
3	nz\ac\massey\rimsgroup3\Service.java	16	Private field 'retriever' could be made final; it is only initialized in the declaration or constructor.
4	nz\ac\massey\rimsgroup3\Service.java	22	Use block level rather than method level synchronization
5	nz\ac\massey\rimsgroup3\Service.java	23	Avoid using if statements without curly braces
6	nz\ac\massey\rimsgroup3\database\DatabaseConnection.java	11	Found non-transient, non-static member. Please mark as transient or provide accessors.
7	nz\ac\massey\rimsgroup3\database\DatabaseConnection.java	13	Avoid variables with short names like db
8	nz\ac\massey\rimsgroup3\database\DatabaseConnection.java	19	Avoid using if...else statements without curly braces
9	nz\ac\massey\rimsgroup3\database\DatabaseConnection.java	21	Avoid using if...else statements without curly braces
10	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	17	Avoid unused imports such as 'org.apache.catalina.Session'
11	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	17	Avoid unused imports such as 'org.apache.catalina.Session'
12	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	18	Avoid unused imports such as 'org.apache.tomcat.dbcp.dbcp.DataSourceConnectionFactory'
13	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	18	Avoid unused imports such as 'org.apache.tomcat.dbcp.dbcp.DataSourceConnectionFactory'
14	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	22	The class 'DatabaseInsert' has a Cyclomatic Complexity of 6 (Highest = 19).

15	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	24	Found non-transient, non-static member. Please mark as transient or provide accessors.
16	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	46	Avoid variables with short names like db
17	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	56	Avoid really long methods.
18	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	56	The method 'doGet' has a Cyclomatic Complexity of 19.
19	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	56	The method doGet() has an NPath complexity of 5072
20	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	71	Avoid using if statements without curly braces
21	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	72	Avoid using if statements without curly braces
22	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	73	Avoid using if statements without curly braces
23	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	83	Avoid excessively long variable names like statementPublication
24	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	85	Avoid using if statements without curly braces
25	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	87	Avoid variables with short names like i
26	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	91	Avoid excessively long variable names like statementPublished
27	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	95	Avoid using if statements without curly braces
28	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	96	Avoid using if statements without curly braces
29	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	103	Avoid using if statements without curly braces
30	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	108	Avoid excessively long variable names like statementConference
31	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	110	Avoid using if statements without curly braces
32	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	117	Avoid using if statements without curly braces
33	nz\ac\massey\rimsgroup3\database\DatabaseInsert.java	124	Avoid using if statements without curly braces

34	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	4	Avoid unused imports such as 'java.sql.Connection'
35	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	4	Avoid unused imports such as 'java.sql.Connection'
36	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	5	Avoid unused imports such as 'java.sql.SQLException'
37	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	5	Avoid unused imports such as 'java.sql.SQLException'
38	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	6	Avoid unused imports such as 'java.util.ArrayList'
39	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	6	Avoid unused imports such as 'java.util.ArrayList'
40	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	22	Avoid unused private fields such as 'datasource'.
41	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	50	Avoid excessively long variable names like modifyToThisInformation
42	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	51	Avoid excessively long variable names like originalInformation
43	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	54	Avoid unused local variables such as 'origAuthor'.
44	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	57	Avoid unused local variables such as 'newAuthor'.
45	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	61	Avoid if (x != y) ..; else ..;
46	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	62	Avoid empty if statements
47	nz\ac\massey\rimsgroup3\database\DatabaseModify.java	66	Avoid empty if statements
48	nz\ac\massey\rimsgroup3\database\DatabaseRead.java	25	Found non-transient, non-static member. Please mark as transient or provide accessors.
49	nz\ac\massey\rimsgroup3\database\DatabaseRead.java	46	Avoid variables with short names like db
50	nz\ac\massey\rimsgroup3\database\DatabaseRead.java	78	Position literals first in String comparisons
51	nz\ac\massey\rimsgroup3\database\DatabaseRead.java	84	Position literals first in String comparisons
52	nz\ac\massey\rimsgroup3\database\DatabaseRead.java	90	Position literals first in String comparisons

53	nz\ac\massey\rimsgroup3\database\InsertStatements.java	13	All methods are static. Consider using Singleton instead. Alternatively, you could add a private constructor or make the class abstract to silence this warning.
54	nz\ac\massey\rimsgroup3\database\InsertStatements.java	60	Avoid excessively long variable names like statementConference
55	nz\ac\massey\rimsgroup3\database\InsertStatements.java	122	Avoid excessively long variable names like statementPublication
56	nz\ac\massey\rimsgroup3\database\InsertStatements.java	136	Avoid variables with short names like i
57	nz\ac\massey\rimsgroup3\database\InsertStatements.java	158	Avoid excessively long variable names like statementPublished
58	nz\ac\massey\rimsgroup3\database\ModifyStatements.java	10	All methods are static. Consider using Singleton instead. Alternatively, you could add a private constructor or make the class abstract to silence this warning.
59	nz\ac\massey\rimsgroup3\database\ModifyStatements.java	34	Avoid variables with short names like i
60	nz\ac\massey\rimsgroup3\database\ReadStatements.java	11	All methods are static. Consider using Singleton instead. Alternatively, you could add a private constructor or make the class abstract to silence this warning.
61	nz\ac\massey\rimsgroup3\database\ReadStatements.java	22	Ensure that resources like this ResultSet object are closed after use
62	nz\ac\massey\rimsgroup3\database\ReadStatements.java	37	Avoid using if statements without curly braces
63	nz\ac\massey\rimsgroup3\database\ReadStatements.java	53	Ensure that resources like this ResultSet object are closed after use
64	nz\ac\massey\rimsgroup3\database\ReadStatements.java	62	Avoid using if statements without curly braces
65	nz\ac\massey\rimsgroup3\database\ReadStatements.java	75	Avoid excessively long variable names like statementConference
66	nz\ac\massey\rimsgroup3\database\ReadStatements.java	79	Ensure that resources like this ResultSet object are closed after use
67	nz\ac\massey\rimsgroup3\database\ReadStatements.java	90	Avoid using if statements without curly braces
68	nz\ac\massey\rimsgroup3\database\Rea	107	Ensure that resources like this ResultSet

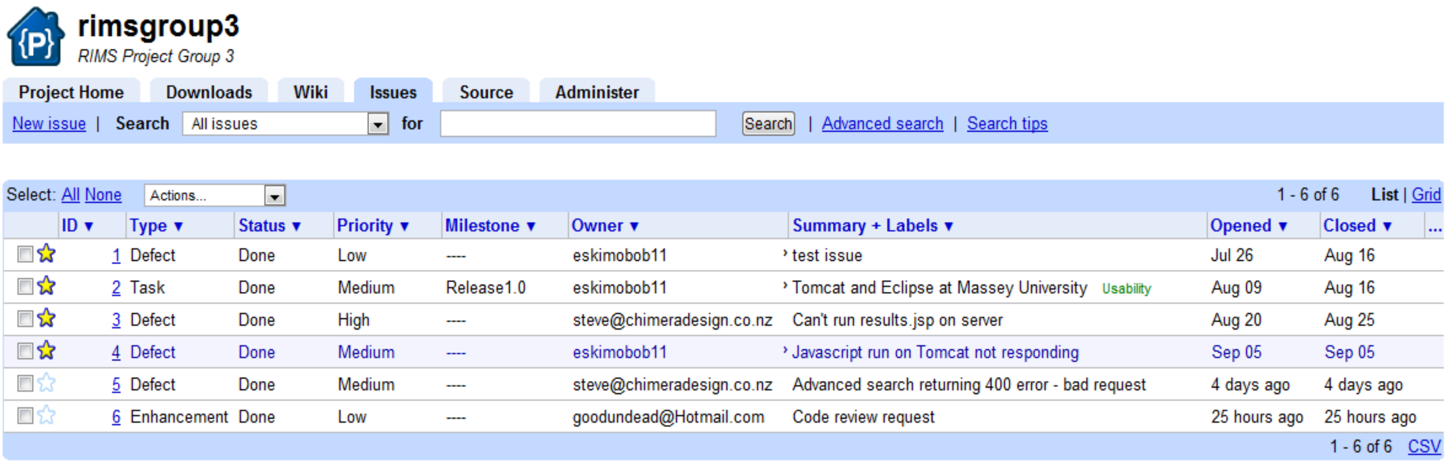
	dStatements.java		object are closed after use
69	nz\ac\massey\rimsgroup3\database\ReadStatements.java	118	Avoid using if statements without curly braces
70	nz\ac\massey\rimsgroup3\database\ReadStatements.java	136	Ensure that resources like this ResultSet object are closed after use
71	nz\ac\massey\rimsgroup3\database\ReadStatements.java	146	Avoid using if statements without curly braces
72	nz\ac\massey\rimsgroup3\database\ReadStatements.java	159	Avoid excessively long variable names like statementPublication
73	nz\ac\massey\rimsgroup3\database\ReadStatements.java	161	Ensure that resources like this ResultSet object are closed after use
74	nz\ac\massey\rimsgroup3\database\ReadStatements.java	186	Avoid using if statements without curly braces
75	nz\ac\massey\rimsgroup3\metadata\DOINotFoundException.java	14	Found non-transient, non-static member. Please mark as transient or provide accessors.
76	nz\ac\massey\rimsgroup3\metadata\DOINotFoundException.java	14	Private field 'doi' could be made final; it is only initialized in the declaration or constructor.
77	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	17	A class which only has private constructors should be final
78	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	20	Avoid excessively long variable names like metadataRetrievers
79	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	20	Found non-transient, non-static member. Please mark as transient or provide accessors.
80	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	20	Private field 'metadataRetrievers' could be made final; it is only initialized in the declaration or constructor.
81	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	29	Use block level rather than method level synchronization
82	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	30	Avoid using if statements without curly braces
83	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	36	Use block level rather than method level synchronization
84	nz\ac\massey\rimsgroup3\metadata\MetadataRetrieverFactory.java	45	Avoid using if statements without curly braces
85	nz\ac\massey\rimsgroup3\metadata\bean\Conference.java	6	Avoid unused imports such as 'java.util.Date'

86	nz\ac\massey\rimsgroup3\metadata\bean\Conference.java	6	Avoid unused imports such as 'java.util.Date'
87	nz\ac\massey\rimsgroup3\metadata\bean\Person.java	12	Avoid variables with short names like id
88	nz\ac\massey\rimsgroup3\metadata\bean\Person.java	12	Found non-transient, non-static member. Please mark as transient or provide accessors.
89	nz\ac\massey\rimsgroup3\metadata\bean\Person.java	21	Avoid variables with short names like id
90	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	14	Avoid variables with short names like id
91	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	14	Found non-transient, non-static member. Please mark as transient or provide accessors.
92	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	22	Found non-transient, non-static member. Please mark as transient or provide accessors.
93	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	22	It is somewhat confusing to have a field name with the same name as a method
94	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	25	Avoid excessively long variable names like publicationCategory
95	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	37	Avoid variables with short names like id
96	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	51	Avoid excessively long variable names like publicationCategory
97	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	79	Avoid variables with short names like id
98	nz\ac\massey\rimsgroup3\metadata\bean\Publication.java	214	Avoid if (x != y) ..; else ..;
99	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	35	The field name indicates a constant but its modifiers do not
100	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	35	Variables should start with a lowercase character
101	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	35	Variables that are not final should not contain underscores (except for underscores in standard prefix/suffix).
102	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	63	Avoid using if statements without curly braces
103	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	102	Avoid using implementation types like 'ArrayList'; use the interface instead

104	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	106	Avoid using if statements without curly braces
105	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	112	Avoid using if statements without curly braces
106	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	121	Avoid if (x != y) ..; else ..;
107	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	153	Avoid unused method parameters such as 'xmlDoc'.
108	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	153	Avoid unused private methods such as 'parseBook(Document)'.
109	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	159	Avoid unused private methods such as 'parseConference()'.
110	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	172	Avoid unused private methods such as 'safeParseInt(String)'.
111	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	191	Avoid using if statements without curly braces
112	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	196	Avoid using if statements without curly braces
113	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	213	Avoid using if statements without curly braces
114	nz\ac\massey\rimsgroup3\metadata\plugin\CrossrefRetriever.java	218	Avoid using if statements without curly braces
115	nz\ac\massey\rimsgroup3\metadata\plugin\ScopusRetriever.java	30	Avoid using if statements without curly braces
116	nz\ac\massey\rimsgroup3\metadata\plugin\ScopusRetriever.java	30	Position literals first in String comparisons
117	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	3	Avoid duplicate imports such as 'java.io.IOException'
118	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	3	Avoid duplicate imports such as 'java.io.IOException'
119	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	4	Avoid duplicate imports such as 'java.io.InputStream'
120	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	4	Avoid duplicate imports such as 'java.io.InputStream'
121	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	28	Document empty constructor
122	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	66	Avoid unused local variables such as 'test1'.
123	nz\ac\massey\rimsgroup3\servlet\Advan	67	Avoid unused local variables such as

	cedRequest.java		'test2'.
124	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	84	Method names should not start with capital letters
125	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	91	Avoid if (x != y) ..; else ..;
126	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	96	Avoid empty catch blocks
127	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	125	Avoid variables with short names like is
128	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	131	Avoid if (x != y) ..; else ..;
129	nz\ac\massey\rimsgroup3\servlet\AdvancedRequest.java	132	Avoid variables with short names like sb
130	nz\ac\massey\rimsgroup3\servlet\DoiRequest.java	38	Avoid if (x != y) ..; else ..;

10.9 Appendix I – Issue tracking



rimsgroup3
RIMS Project Group 3

Project Home
Downloads
Wiki
Issues
Source
Administer
New issue | Search All issues for Search | Advanced search | Search tips
Select: All None Actions...
1 - 6 of 6 List | Grid

ID	Type	Status	Priority	Milestone	Owner	Summary + Labels	Opened	Closed
1	Defect	Done	Low	----	eskimobob11	test issue	Jul 26	Aug 16
2	Task	Done	Medium	Release1.0	eskimobob11	Tomcat and Eclipse at Massey University Usability	Aug 09	Aug 16
3	Defect	Done	High	----	steve@chimeradesign.co.nz	Can't run results.jsp on server	Aug 20	Aug 25
4	Defect	Done	Medium	----	eskimobob11	Javascript run on Tomcat not responding	Sep 05	Sep 05
5	Defect	Done	Medium	----	steve@chimeradesign.co.nz	Advanced search returning 400 error - bad request	4 days ago	4 days ago
6	Enhancement	Done	Low	----	goodundead@Hotmail.com	Code review request	25 hours ago	25 hours ago

Number of open issues: 0
Number of closed issues: 5 (not including test issue)
Average time to close issues: 3-4 days