“Proficio”

Software Engineering C

Week 12 Group Report

**Project Leader:**

Adam Bramley

**Team Members:**

Stephen Allan

Leith Bade

Peter Williamson

Table of Contents

[1. Introduction 5](#_Toc272399144)

[1.1 Team Contribution 5](#_Toc272399145)

[1.1.1 Report Contribution 5](#_Toc272399146)

[1.2 End User Interactions 6](#_Toc272399147)

[1.3 Project Obstacles 6](#_Toc272399148)

[1.4 Project Plan 6](#_Toc272399149)

[2. Requirements Revision 7](#_Toc272399150)

[2.1 Functional Requirements: 7](#_Toc272399151)

[2.2 Non-Functional Requirements: 8](#_Toc272399152)

[2.3 Summary of changes 8](#_Toc272399153)

[3. Technological Assessment 8](#_Toc272399154)

[3.1 Language 8](#_Toc272399155)

[3.2 Integrated Development Environment (IDE) 9](#_Toc272399156)

[3.3 Architecture 10](#_Toc272399157)

[3.4 Plugin system 11](#_Toc272399158)

[3.5 Testing System 11](#_Toc272399159)

[3.6 Web Framework 12](#_Toc272399160)

[3.7 Web server 13](#_Toc272399161)

[3.8 Database 14](#_Toc272399162)

[3.9 Build System 16](#_Toc272399163)

[4. Project Life Cycle Methodologies 16](#_Toc272399164)

[4.1 Meetings 16](#_Toc272399165)

[4.2 Sprints 16](#_Toc272399166)

[4.3 Spike Tests 16](#_Toc272399167)

[4.4 Milestones 17](#_Toc272399168)

[5. System Architecture 17](#_Toc272399169)

[6. Deployment Strategy 18](#_Toc272399170)

[6.1 Deploying a JSP Application 18](#_Toc272399171)

[6.2 The Database 18](#_Toc272399172)

[7. Risk Management 19](#_Toc272399173)

[7.1 Risks that have been added since previous report 19](#_Toc272399174)

[8. Issue Tracking Policy 20](#_Toc272399175)

[8.1 Issue Tracking Workflow 20](#_Toc272399176)

[9. Testing and Metrics 20](#_Toc272399177)

[10. Appendices 22](#_Toc272399178)

[10.1 Appendix A – System Architecture 22](#_Toc272399179)

[10.2 Appendix B – Risk Analysis 23](#_Toc272399180)

[10.3 Appendix C – ER Diagram 27](#_Toc272399181)

[10.4 Appendix D – Use Case Diagram 28](#_Toc272399182)

[10.5 Appendix E – Gantt chart 29](#_Toc272399183)

[10.6 Appendix F – JDepend Report 30](#_Toc272399184)

[10.6.1 Summary 30](#_Toc272399185)

[10.6.2 Packages: 30](#_Toc272399186)

[10.6.3 Cycles 33](#_Toc272399187)

[10.7 Appendix G – Emma Report 34](#_Toc272399188)

[10.8 Appendix H – PMD Report 35](#_Toc272399190)

[10.9 Appendix I – Issue tracking 41](#_Toc272399191)

# 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.

## 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.

## End User Interactions

During weeks 8-12 we only needed to meet with our client one last time to clear up some interface and functionality issues we were having. During this meeting we allowed Karen to use our system for what it is intended to do, and she submitted an article to RIMS with the assistance of our product. In doing this we got a huge amount of constructive criticism about changes we could make to the interface to give a more streamlined operation. As our functionality had already been finalised with her previously there weren’t any major changes we found from this meeting.

## 1.3 Project Plan

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

Overall, we have achieved what we set out to do from the beginning of the project. Along the way there have been many changes and things we have had to leave out due to time constraints.

### 1.3.1 Features left out from final product

1. At the beginning of the project we thought we would need to include an email client to email the authors for confirmation of the submission. Before we began implementing this feature we found that it was redundant as the user must manually email the authors regardless.
2. We were planning to have more data sources than just CrossRef. Unfortunately, it took us quite a while to get Scopus working properly as we not able to obtain the proper developer’s key for the full API. Due to this we had to use the Javascript API with limited functionality. This resulted in it taking much longer to finish the Scopus plug-in, giving us no time to continue with any other possible plugins
3. Screen scrapping was going to be a last resort for us when our plug-ins had failed to return any data. Near the end of the project we realised that it was far too messy and unreliable to use this feature, and decided to stick only to the reliable sources of CrossRef and Scopus.
4. Since neither CrossRef nor Scopus return an abstract, we were working on screen scrapping for the abstract off of the publisher’s website. We ran out of time to effectively implement this feature, without having html tags and other unwanted characters left in the description.

# 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 including CrossRef and Scopus. The meta-data output includes: authors, article title, journal title, publication year, URL, page numbers, volume, and issue for journal articles. For conference articles we will also include conference location and start/end dates. For books we will also include the place published and editors of the book.
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.

## 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 have not changed much since the last report. The main changes have been that we are no longer attempting to “screen scrape” any information from publisher’s websites if our plugins fail. This is because it will be very rare that both Scopus and CrossRef return no data, and it isn’t very elegant relying on page layout and html tags to extract reliable information.

We have also decided not to change our plugin architecture to allow for article types that may need to be added in the future. We have decided this because Karen had told us at the very beginning of the project that the only things she deals with are Journal articles, Books, and Conferences. Since we structured our whole system around this there will need to be significant changes to switch to a plug-in type architecture for the types, and we simply don’t have the time to implement this.

# 3. System Architecture

*Please refer to Appendix A.*

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 DOIRequest class then sends the DOI string 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 factory will return it to the DOIRequest. The plugins work by using the sites’ APIs and filling the Publication object with information gathered through parsing the XML from the data source.

Once the DOIRequest receives the publication object it then either outputs an error message if something has gone wrong, or there is no such article; or it forwards the information to a 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 author and publication details, the publication is then sent for committing to the database and, if the checkbox is ticked, a separate window is opened for printing.

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.

# 4. 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).

## 4.1 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.

## 4.2 The Database

A locally accessed database is required for the author details and DOIs of articles that have already been entered. 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.

# 5. Issue Tracking Policy

From weeks 8-12 our team took issue tracking much more seriously, mainly because there were potential marks involved in doing so. To our surprise, using them actually helped our project quite considerably, resulting in more bugs being fixed at a faster rate than before. We are now posting issues wherever we can’t find a solution immediately, or when the issue is with another member’s code.

5.1 Issue Tracking Workflow

* Open a new issue through the Google code project page
* Fill out the summary and description of the issue.
* Cc to all team members.
* Make sure the issue type and priority are set accordingly.
* Submit the issue.   
    
  *After the issue has been submitted:*
* Adam will assign the issue to the team member that was responsible for creating the issue. For example, since Peter is the main contributor to designing the database, any issues relating to the database will belong to him. This assures that issues will be fixed as fast as possible.  
    
  *After the issue has been resolved:*
* Adam will check the issue is solved, and if it is, set the issue to “Done” on the project page.

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.

# 6. 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.

## 6.1 SVN Metrics

For the last 5 weeks (7/09/10 to 12/10/10) the following SVN statistics have been produced using TortoiseSVN’s analysis:

Total Commits analysed: 100  
Total file changes: 361

|  |  |  |  |
| --- | --- | --- | --- |
| Commit Details Per Week | | | |
|  | Average | Min | Max |
| Commits per week | 20 | 8 | 26 |
| Most active author (Adam Bramley) | 8 | 2 | 10 |
| Least Active Author  (Leith Bade) | 1 | 0 | 4 |
| File Changes per week | 72 | 22 | 93 |

The following graph shows the commits per author in a pie chart:

commit graph.wmf

Eskimobob11: Adam Bramley,

steve@chimeradesign.co.nz: Steve Allan,

goodundead@hotmail.com: Peter Williamson,

leithbade: Leith Bade

## 6.2 Compatibility testing

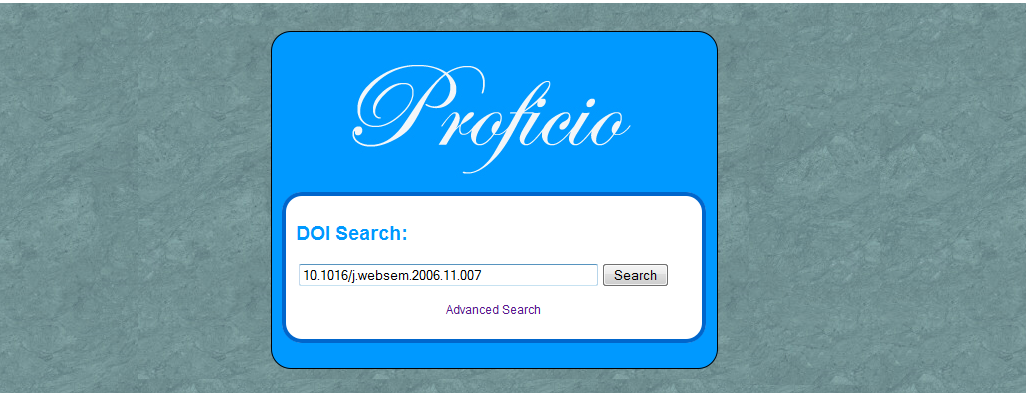
All testing was done using Windows XP, this is because the only platform we expect the admin at Massey to be using is Windows and since it’s a web application the compatibility relies more on the browser.

|  |  |
| --- | --- |
| **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. |

# 7. User Manual

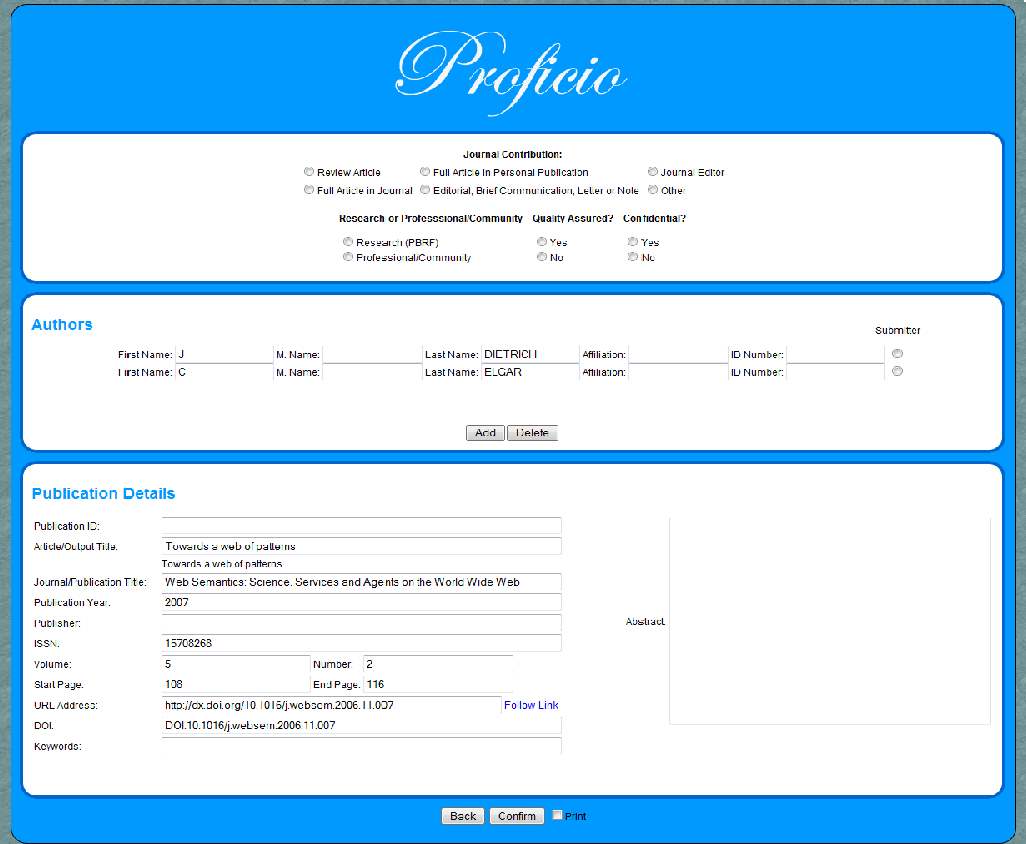
This user manual is available via the application on the index page through the help icon. Each section explains the different pages and how to use them. There is also a section at the end explaining the basic terms on the results page.

## 7.1 Basic Search



* Enter the DOI into the text box and click the Search button to search for the information regarding the DOI.
* Please wait while Proficio finds the article.

## 7.2 Results Returned



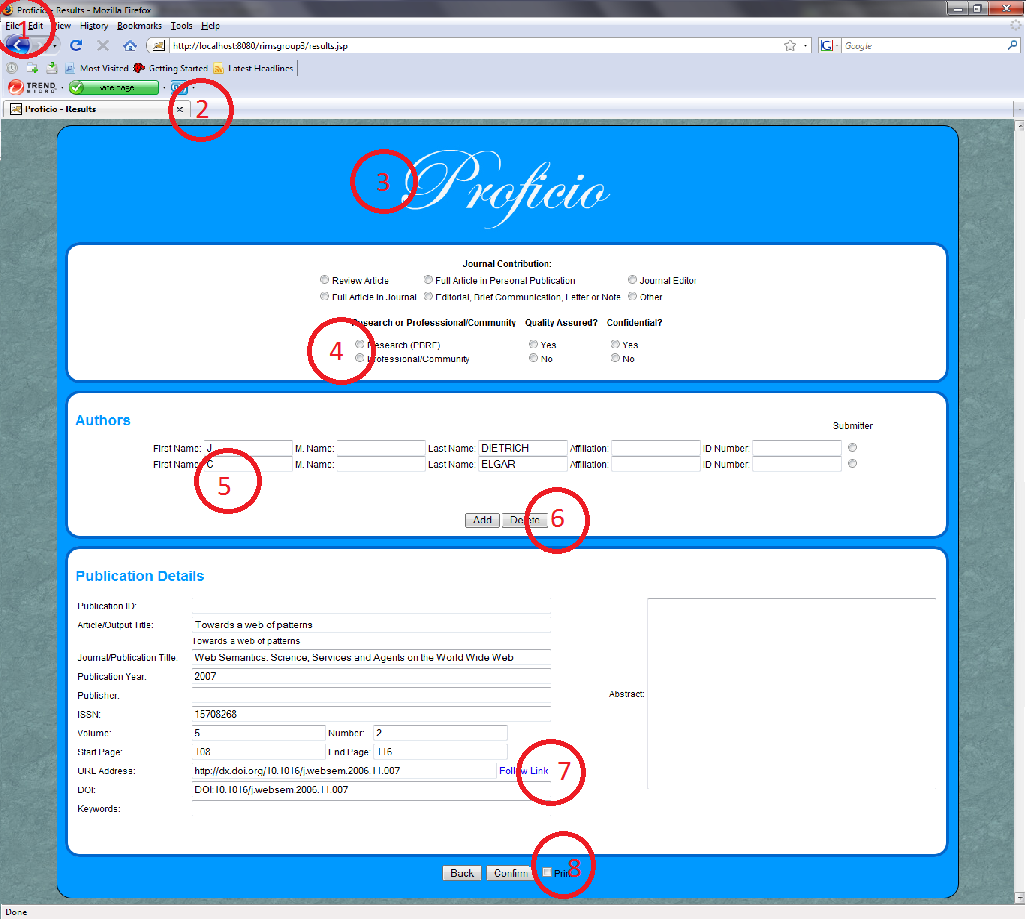
* Use the information gathered by the Proficio to fill in the details in the RIMS system.   
  This can be done by highlighting the relevant fields text, then copying (Ctrl+C) and pasting (Ctrl+V) the text into corresponding text field in the RIMS system.
* Any information not retrieved from the Proficio will have to be filled out manually.   
  Click the new tab that has appeared to access the remaining information. Alternatively click the Follow Link link
* If you wish to print set the check box to print.  
  Fill out any blank text fields and click the appropriate radio buttons in Proficio, that are relevant to information that is required for printing (any information left blank will not be printed).
* Publication ID is the RIMS system ID number after committing the article to the RIMS system.   
  This information can only be gathered from the RIMS system.
* Click the Confirm button.
* Now you are done! Any information in the DOI text box and the Authors text boxes has been saved to the database, The DOI has been saved to prevent anyone from duplicating the work, and if the same author has any more publications all the details will be returned next time.

## 7.3 Advanced Search

## indexAdvanced.jpg

* If for whatever reason you are not supplied with a DOI, click the Advanced Search link
* Fill out the all the appropriate fields, click the bottom Search button
* If the application somehow fails please use the DX DOI tool to locate the article.   
  Enter in the DOI and click the Locate Article button.   
  We apologise for the inconvenience that was caused.

## 7.3 Glossary of Terms



|  |  |  |
| --- | --- | --- |
| 1. Browser |  | 2. Tabs |
| 3. Application - Proficio |  | 4. Radio Button |
| 5. Text Box |  | 6. Button |
| 7. Link |  | 8. Check Box |

# 10. Appendices

## 10.1 Appendix A – System Architecture

Figure 1. Improved System Architecture

## 10.2 Appendix B - Database Schema

string Massey\_ID**(PK)**  
string Massey\_First\_Name

string Massey\_Last\_Name  
string Massey\_Middle\_Name

string Type

string Department

string College

**Massey\_Author**

int Misc\_ID **(PK)**  
string Misc\_First\_Name

string Misc\_Last\_Name  
string Misc\_Middle\_Name

string Affiliation

**Misc\_Author**

string Publication\_DOI **(PK)**

**Publication**

Please note this is not incomplete. Our database only contains 3 tables of which none are related to each other.

## 10.4 Appendix C – Use Case Diagram

## 

User

Gather publication Information (No DOI)

<<includes>>

Fills out Coversheet

/Adds to Database

<<includes>>

<<includes>>

<includes>>

<<includes>>

Print

Coversheet

Add Staff   
Information

<<includes>>

Add RIMS   
Identifier

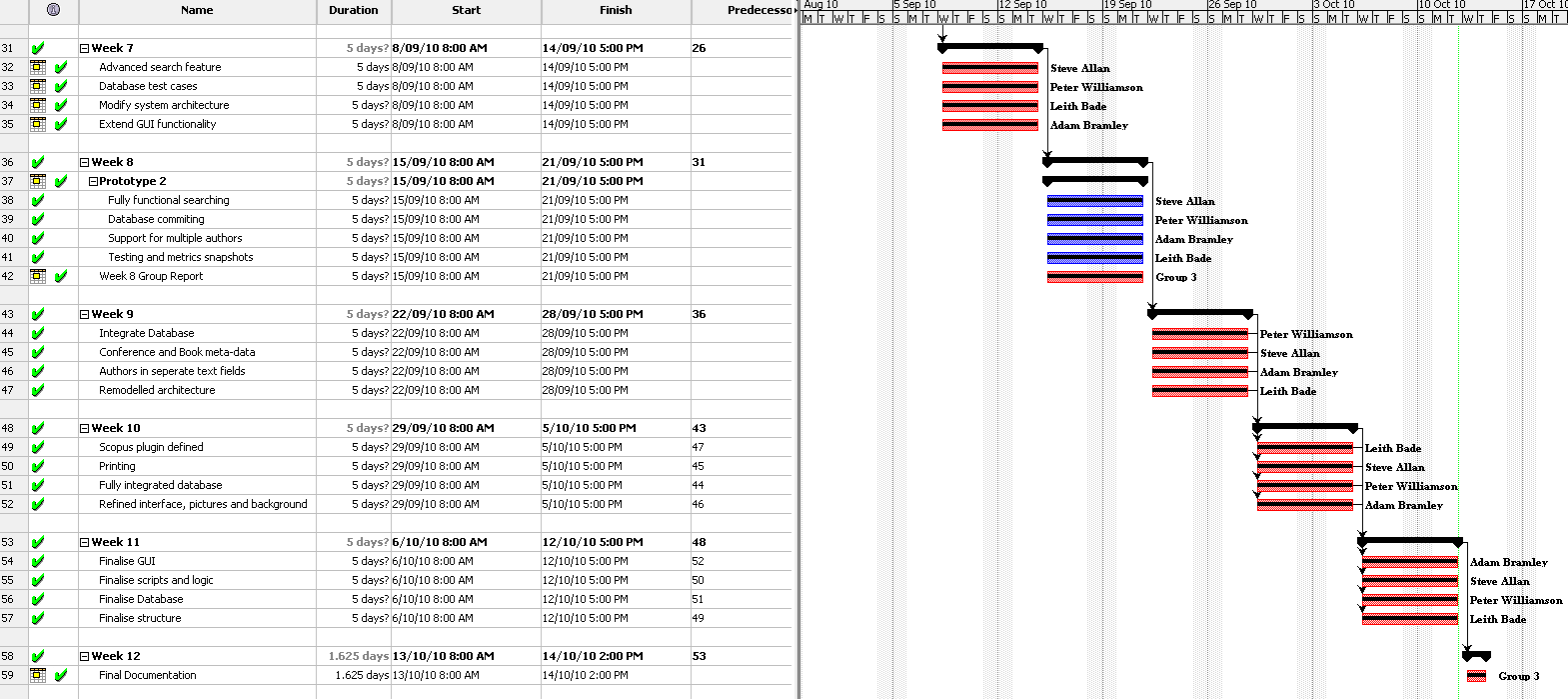
<<includes>>

<<includes>>

<<includes>>

Gather publication Information (DOI)

## 10.5 Appendix D – Gantt chart



## 10.6 Appendix E – JDepend Report

## 10.7 Appendix F – Emma Report

## 10.8 Appendix G – PMD Report

## 10.9 Appendix H – Issue tracking