# Feasibility Study into Software Design and Collaboration Tools

## Executive Summary

This document outlines some of the considerations associated with developing a source code, bug tracking, and software design collaboration tool. Although individually these tools have existed independently of each other, it has been decided that I would be best to incorporate these tools into a ‘super collaboration’ tool, giving all the power of source control, mixed with bug tracking and reporting; while allowing communications between the team to be tracked during the software development process.  
One such software comes close in providing this service, but with the ready availability of multiple tools are open-source or freeware, so why not apply a mix readily existing tools, providing the development community at large, with the tools necessary to complete complex design tasks with minimal concern.

## Product Services

The problem with a product such as this is that, a single all in one solution isn’t readily available on the market, software such as ‘Bugzilla’, is able to accurately keep track of bugs, whereas software such as ‘Gource’ is excellent visual tool for seeing member contributions to software repositories. But none of these packages do everything, ‘Unfuddle’ probably being the closes

### Gource

One aspect of the software that’s being commissioned is being able to graphically visualise the changes being performed to the repository, by tracking the logs (through trunks, commits and checkouts) (acaudwell, 2013). Gource, another open source project provides an amazing display to visualise the data as it makes changes- of course this software is inherently limited since it provides no real other features related to this specification.

### Bugzilla

According to the Bugzilla website, this open source software tracks bugs and code changes, allows you to communicate with teammates and submit/review patches (Bugzilla.org, 2012). It markets itself as being able to “reduce downtime, increase productivity, raise customer satisfaction, and improve communication”.

### Fossil

Fossil (Fossil, 2014) is probably the closest to the sort of software that we’re after. It provides a web-based interface with graphical tools, bug tracking and support for creating your own wiki. The main issues with using this software is that email control, since the requirements specify importing project emails into the source control, although the use of the Wiki could work around this.

### Unfuddle

Unfuddle appears to fill most of the requirements, allowing bug tracking and management of source code, this software can also allow collaboration through a messaging service and wiki. The major downside of this software is that unlike all the previous software, its not freeware, and requires a minimum $15 a month (Unfuddle).

### Product/Service Marketplace

The only reputable competition to this software came from Unfuddle, and with their signup fees, would be a major upset, especially for the casual developer who prefers to use free and easy tools.  
All the other tools failed to capture all the necessity, and it should be noted that even Unfuddle still doesn’t quite do what’s required, its just happens to be the closest thing.  
This sort of software, due to its cost, would be targeted at developers and businesses that can both afford and are large enough to require using a comprehensive source control strategy. Smaller developers have to rely on existing open source and freeware tools, in order to collaborate and create their products, its here that exists a niche market for the serious but small developer.

## Marketing Strategy

Since this software is more likely to be used in-house, there doesn’t really require marketing a program like this. However, if this program is to be released externally, then it should be released as free, allowing for all users to popularise its use and become a staple of the software development community. In order to generate a profit (or at lease break even), advertising space can be bought to complement this software and pay for its development. As well as this, for large organisations a premium service can be offered that gives companies extra support and training for its staff, for a nominal fee, and as such generating a profit.

## Technology Considerations

#### Deployment Environment

This service would potentially be used by a wide number of users each with many different Operating Systems and Configurations, as such, to combat this, the service needs to be platform independent, which can be achieved through use of a web service, requiring a web browser.

#### Web Services

##### Apache

Pros:

* Supports FastCGI
* Has a Module for PHP
* Widely used and well documentation, Good Support
* Runs on All Operating Systems (Works Better on Linux)
* Open source (Free)

Cons:

* Can be very heavy as it supports many features

##### Nginx

*Pros:*

* Supports FastCGI
* Runs on All Operating Systems (Works Better on Linux)
* Extremely Light Weight
* Growing and becoming more popular
* Open Source (Free)
* Has easiest to read configuration

*Cons:*

* Does not support as many features as Apache
* Relatively new to other web servers, not as mature or tested

#### Language Implementation

##### PHP

* Object Oriented
* Weakly Typed
* Just in Time Compilation
* Normally has a web server running in front of it. (Nginx, Apache, ISS)

##### Java

* Object Oriented
* Strongly Typed
* Compiled to byte code, then interpreted by Java VM
* Normally has a web server running in front of it. (Apache Tomcat, Oracle)

##### C++

* Object Oriented
* Strongly Typed
* Compiled
* Robust
* Not easy to parse files with

##### Ruby on Rails

* Object Oriented
* Can easily pass files
* Database management somewhat automatic
* Easy to learn
* High Overhead

#### Storage Solution

##### SQLITE

Creates files on disk containing Data, not good for reading from or storing large amounts of data.

Does not support all SQL statements

Does not support multiple users

##### MySQL

* Widely supported and widely used on the web, standard to have behind implements well with scripting language such as PHP or Ruby
* Open source

## Organisation and Staffing

Joshua Brown- System Data Specialist  
Josh has experience in database implementation and as such is fully capable of designing and implementing our data storage solution.

###### Peter Brown-Requirements Analyst and User Interface Engineer

Peter is in charge of implementing the front-end of the software, specifically regarding how our user interacts with the product. Peter will also ensure that the interfaces are intuitive, and well documented such that anybody with limited knowledge of these systems will be able learn how to use the product.

###### James Glennan- Software Architect

James is required to help implement software solutions and learn new languages that may be required. This job will require specific detail in ensuring that tests are designed for all of the system’s modules as well as ensuring that code is implemented as efficiently as possible.

###### Kurt Robinson- Tool Specialist and Change Control Manager

Kurt is hosting the software repository as well conducting research and learning about specific open-source components that can be implemented in to this project. He is tasked with third party implementation, as well as liaising with the relevant authorities to ensure that the product’s modules connect cohesively.

###### James Wilson- Project Manager

James is directing the group as well as coordinating back and front-end integration. James will monitor the individual module’s development direction and progress as well as ensuring that weekly reporting/testing suites are beinf provided for the relevant sections.

## Schedule

For a more comprehensive schedule, please see project plan.

1. Environment Set Up
   1. Set up development environment
   2. Set up MySQL database
   3. Set up coding environment
   4. Set up website tools
2. Iteration 1.1 - Website Construction
   1. Analysis
   2. Design
   3. Development
   4. Testing
   5. Review
   6. Milestone - Website Constructed
3. Iteration 1.2 - User Management System
   1. Analysis
   2. Design
   3. Development
   4. Testing
   5. Review
   6. Milestone - User Management System Complete
4. Iteration 2.1 - Import Data
   1. Analysis
   2. Design
   3. Code
   4. Testing
   5. Review
   6. Milestone - Import data complete
5. Iteration 2.2 - Data Translation
   1. Analysis
   2. Design
   3. Code
   4. Testing
   5. Review
6. 2.2 Milestone - Data translation complete
7. Iteration 2.3 - Data Manipulation
   1. Analysis
   2. Design
   3. Code
   4. Testing
   5. Review
   6. Milestone - Data manipulation complete
8. Iteration 2.4 - Data Visualisation
   1. Analysis
   2. Design
   3. Code
   4. Testing
   5. Review
   6. Milestone - Data visualisation complete
9. Develop Final Test Script
10. Deployment
11. User Manual
12. Testing with Client
13. Approved by Client
14. Evaluation

## Financial Projections

At this stage there isn’t a real projection of financial income due to the nature of this software. If the above suggestions are taken on-board, that is, use the software to advertise and offer training programmes, it can be assumed that money can be made. However this hinges on the fact that this software must become popular to start with, and that is why it will be released as freeware.

## Findings and Recommendations

It is our recommendation that software be immediately be commissioned to fill a need since collaboration isn’t cost effectively managed by any existing software. The conclusion being that a better and more tailored piece of software can be developed, that meets the requirements specified for this project.

This product will be designed around a three module system. Front-end that interfaces with the user, back-end that parses and stores the RAW data and the middle component that provides the bridge between both by making sense of the raw data and delivering it to the front to be displayed in a aesthetic and intuitive way. To do this, a web server utilising NGineX, will serve a web page that draws its data from a MYSQL database imported by our Ruby On Rails back-end reader. We believe this modular design will assist in ease of development, as well as allowing us to make modification as we go to ensure correctness and function of the system.

# Works Cited

Unfuddle. (n.d.). *Unfuddle Signup.* Retrieved 03 26, 2014, from https://unfuddle.com/signup

acaudwell. (2013, 04 26). *Gource-Software Version Control Visualisation.* Retrieved 03 26, 2014, from https://code.google.com/p/gource/

Bugzilla.org. (2012, 11 19). Retrieved 03 26, 2014, from http://www.bugzilla.org/about/

Fossil. (2014, 03 15). *Fossil.* Retrieved 03 26, 2014, from https://www.fossil-scm.org/index.html/doc/tip/www/index.wiki