Requirements Analysis

B.Kendrick, A.Syed

Contents

[Introduction 1](#_Toc412623488)

[Current System 1](#_Toc412623489)

[Proposed System 1](#_Toc412623490)

[The Useful Features of BlueJ 1](#_Toc412623491)

[Our Plugin 2](#_Toc412623492)

[Stakeholder identification 2](#_Toc412623493)

[The Client 2](#_Toc412623494)

[Lecturers 2](#_Toc412623495)

[Students 2](#_Toc412623496)

[Programmers 3](#_Toc412623497)

[Stakeholder Interviews 3](#_Toc412623498)

[The Client – Gary Allen 3](#_Toc412623499)

[Questionnaire to Students 3](#_Toc412623500)

[Lecturer – Andrew Crampton 3](#_Toc412623501)

[Stakeholder issues 4](#_Toc412623502)

[Requirements 4](#_Toc412623503)

[List of requirements 4](#_Toc412623504)

[Functional 4](#_Toc412623505)

[Non-Functional 5](#_Toc412623506)

[Process Constraints 5](#_Toc412623507)

[Software 5](#_Toc412623508)

[Development Time 6](#_Toc412623509)

[Requirements Prioritisation 6](#_Toc412623510)

[Resources 7](#_Toc412623511)

[Requirements Confirmation 7](#_Toc412623512)

[Conclusion 7](#_Toc412623513)

# Introduction

The main purpose of this document is to address the requirements of the project in more detail. However additionally there will be a brief description of the current and proposed system, more in detail stakeholder identification and lastly some UML.

This project aims to provide potential students and teachers with the tools found in BlueJ that are beneficial to learning within the environment of an industry standard Integrated Development Environment (IDE). We aim to take the most useful features and concepts from BlueJ and integrate them into the Eclipse IDE. This will allow a more flexible approach to transitioning between an IDE intended for a new and in-training programmer and one preparing to take steps into professional development.

Our tool also proposes to provide lecturers a tool to assist in object oriented concepts such as instantiation, inheritance and assists the usage of classes and objects.

# Current System

The current system and process at the University of Huddersfield involves students learning programming by using BlueJ in the first year. The object oriented aspects of programming are taught using BlueJ to the new students; this is due to the helpful visual features in the IDE. BlueJ also provides functionality to allow for testing code without the need to create a full build meaning small code snippets can be tested quickly and easily via instantiation. From the first year the students are then advised to make use the Eclipse IDE. The Eclipse IDE provides a more professional and traditional interface for software development projects and provides an extensible framework to fit the needs of both teams and individuals.

Currently the only plugin that provides an object bench, with similar functionality to our proposed system, in Eclipse is E-BOB; however this is poorly maintained, and, according to our client, unstable and ‘highly buggy’.

# Proposed System

Our proposed system is an up to date Eclipse object bench plugin, this plugin will attempt to bring the ‘useful’ features that are available in BlueJ into the Eclipse environment.

The purpose of the proposed system is to help new students from the first year have an easier time transitioning to using Eclipse after utilising BlueJ for a variable amount of time.

## The Useful Features of BlueJ

The useful features of BlueJ we wish to incorporate into Eclipse are:

* Object Bench – The Object Bench controls instantiated instances of classes. From the object bench methods can be called to test functionality and determine results.
* Instance Inspector – The instance inspector allows a user to view the fields within an instance and see their values. This is useful when learning because it allows the user to see the results of declarations and the results of methods effecting individual fields.
* Class Diagram – The class diagram view within BlueJ allows the user to see the classes and how inheritance can be visualised. The class diagram is useful as it helps teach both object oriented concepts and basic UML.
* Scope Highlighting – BlueJ provides scope highlighting when writing code. Individual code blocks are surrounded in a different colour depending on the depth of code being written.
* Code Templates - When a user creates a new class in BlueJ the user is given a barebones class with a constructor and a sample method. This is helpful in re-enforcing how a class is constructed and reduce errors in new class creation.
* Test Class Generation – BlueJ can generate test classes which are used for unit testing. This is highly useful for testing written code in a formal manner, this can also be used to design code for the sole purpose of “beating” the series of unit tests defined in a test class.

These features will be looked at in more detail within the “Research” document.

## Our Plugin

Our plugin will attempt to bring the useful features of BlueJ into Eclipse whilst expanding and adding features based on the requirements outlined by our client and from data obtained in our research. Our plugin will integrate with the Eclipse environment and become an easy to use solution for assisting new programmers in using a fully featured IDE whilst still having access to tools that make things easier to both test and understand fully.

# Stakeholder identification

## The Client

The client for this project is Gary Allen, a lecturer at the University of Huddersfield. He gave us the initial problem to solve and the fundamental requirements. He has been looking to use a plugin for Eclipse which uses the most useful features of BlueJ to help teach his students concepts of object oriented programming. The client also wants this plugin to help test code snippets and assist in modelling the effects of various instances acting upon other instances through different methods.

There will be weekly meetings with Gary to ensure that the project is on track and help us make sure that the requirements are up to date and continually valid.

## Lecturers

Lecturers are another potential stakeholder which may want to use the plugin to help in their teaching. Other lecturers may well agree with the client in using the visual learning tools from BlueJ in Eclipse via the use of the plugin.

## Students

Students are the third main stakeholder that this plugin is targeted towards. Students will actually be using this plugin in their studies and possibly assignments set out by the lecturers. Students who are still unfamiliar with their object oriented programming aspects in the second year will find the plugin the most useful, especially if they found BlueJ useful.

## Programmers

Programmers in general are another stakeholder. One can argue that all the previous stakeholders can be classed as programmers; however the plugin was mainly intended for the teaching of students in programming, not industry professional level programmers.

The plugin can still be used by people who are proficient in OO programming who prefer to use visual tools in their coding.

# Stakeholder Interviews

In this section the initial interview with the client will be described, this is where requirements and possible issues with them were outlined before the start of the group project.

## The Client – Gary Allen

The Client Gary Allen pitched the idea of creating a plugin which will ease the students on moving from BlueJ in the first year to Eclipse in the second year.

The requirements were outlined along with the context and nature of the project. These are all documented with the proposal, product specification and requirements analysis. These requirements will later be confirmed again by the client before the design phase.

## Questionnaire to Students

A questionnaire to the students was sent out on the 24th November 201424 November 2014 24 November 2014 24. This questionnaire mainly asked the students on giving their feedback on their experience from learning programming from the first year to the second, especially in regards to the IDE’s BlueJ and Eclipse. The final aim of the questionnaire was to reaffirm the requirements on the need for a plugin which acts as a stepping stone from BlueJ to Eclipse and perhaps outline any missed requirements.

However the result was disappointing as only 3 completed questionnaires were returned 2 of which from by students who had previous programming experience (which isn’t the main target audience of the projects product).

We can attempt to gain more responses on the questionnaire as the project progresses. All feedback will be valuable in shaping future prototypes and fine tuning requirements and features. This will be kept in an additional document that will be referenced as and when more responses become available to us.

## Lecturer – Andrew Crampton

We had a meeting with the leader of the first year software development module to gather feedback on the proposed goals of the plugin. He provided us with a positive amount of feedback regarding the usefulness of the tool and expressed an interest in the merit to its use in teaching.

The meeting allowed us a further verification of our requirements and allowed us more justification for proceeding with the project based on the information we already have.

Andrew has given us access to the first year teaching materials as a means to benchmark the usefulness of our tool. If our tool can be used to complete the teaching exercises outlined in the materials in an efficient manner then this may serve as a proof of usefulness for our tool in regards to its pedagogical merit.

This early meeting also has given us insight into the scope of the project, we may need to aim for a strong proof-of-concept tool which showcases the potential of a tool of this manner over a polished and fully complete end product. Constraints on time will have to shape how much of the project we can complete well.

We will meet with Andrew again at later stages in the project as a means to evaluate progress and gather feedback on the use-ability and usefulness of the product.

# Stakeholder issues

In this section any issues will be outlined that stakeholders may have or present in the future according to the progress of the projects requirements.

Students may find some aspects of the project unneeded / unnecessary, this could affect the state of the requirements list, and this can be avoided to some extent by having students try out a stable prototype. However if we follow a modular development structure we can provide the means for students to only have to use the parts they need and disable or hide parts they feel they don’t need.

Lecturers as well may find issues with usability or even find that they require more from the plugin than what is provided. In this instance we will have to review requests from lecturers and attempt to determine whether the feature will be a worthwhile addition. It is important that we address issues in the best manner possible to create a tool that is highly useful for new programmers.

All these possible issues will be recorded when they happen with a possible risk solution to this, however due to variables such as deadlines and man hours this may be difficult to accommodate appropriately. Additionally, due to the nature of this project often issues will be solved organically and in the moment, this can often be difficult to record as there is no quantifiable data that enables us to create a meaningful log of the events.

# Requirements

This section initially will list all the requirements each with some description, these requirements will be categorised into functional, non-functional and process constraints sections. Finally the requirements will be prioritised with the chosen prioritisation method.

## List of requirements

### Functional

* Class Diagram GUI – The plugin will the creation of classes via the GUI and draw a class diagrams showing the relation between the various classes.
* Class Templates – The user should be able to create a class based on a template. (e.g. Class, Abstract, Interface, Enum etc.)
* Object Workbench – A workbench is required to instantiate instances of individual classes and makes calls to their methods as required.
* Inspector Functionality – There must be some form of inspector utility that enables to monitor the variables within a class. We could also enable this to edit fields to give a user better control over instances.
* Enhanced Inspector Functionality – As well as being able to view the values of variables it should also be possible to change them as the program is running to see the results.
* Package Management – Separate class diagrams for individual packages, these should be navigable.
* Scope Highlighting – The ability to highlight the background of each code block depending on its depth within the code. For example an if method would be surrounded with its own colour for easier code readability and visual scanning.
* Syntax Error Highlighting – Whilst Eclipse has basic syntax highlighting it would be useful to better explain mistakes made by potential students and perhaps have definable colour editing per error to make specific, but basic, errors easier to find.
* Unit Test Generation – The plugin may be able to generate test classes for a user defined class.

### Non-Functional

These requirements are not part of the plugins functionality but rather can be used to judge the operation of the plugin and the experience of the user.

**Usability**

* Modular – Parts of the plugin can be switched off and removed in respect to the users programming proficiency.

**Reliability**

* Robust – The plugin needs to be as reliable as possible as it will be used in teaching. This means it needs to work for the version of Eclipse that the university is using and be stable.

**Performance**

* Responsive – The plugin should be responsive and quick with minimal delay times for actions.

**Supportability**

* Documentation – If at all possible have documentation that can be used as support for other people trying to use the plugin
* Communication – We should be available to respond to queries regarding issues with the plugin and provide assistance when required.

**Interface**

* Easy to Use UI – The UI should be simple to use, intuitive and familiar to students learning java through BlueJ.

# Process Constraints

Throughout development of the project we have to be wary of constraints that may hamper progress or deny the application of specific features.

## Software

The plugin will be developed using the Eclipse SDK within the Plugin Development Environment (PDE). The version we are using is part of the Eclipse Luna distribution. This allows us to make use of newer Eclipse tools and libraries. The only major issue with using a newer version of Eclipse is that the plugin will not work with older versions of Eclipse.

## Development Time

The plugin and all documentation will need to be finished by Week 25, this means that we have a relatively short development period in which we can actively develop and complete the project. There may be times where we have to review less important features and cut content accordingly.

## Requirements Prioritisation

Out of a few methods that were available, MOSCOW was chosen to prioritise the requirements. More detail on the other methods can be found in the research document.

MOSCOW is one of a few methods of requirements prioritisation; it involves sorting the requirements into four sections. Must, Should, Could and Wont. “Must” requirements hold the highest priority and “won’t” hold the least. The table below displays the requirements being prioritised in the appropriate sections of the MOSCOW.

Table - MOSCOW Classification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirements | MUST | SHOULD | COULD | WONT |
| Class Diagram GUI |  | ✓ |  |  |
| Class Templates |  | ✓ |  |  |
| Object Workbench | ✓ |  |  |  |
| Inspector Functionality | ✓ |  |  |  |
| Enhanced Inspector Functionality |  | ✓ |  |  |
| Package Management |  | ✓ |  |  |
| Scope Highlighting |  |  | ✓ |  |
| Syntax Error Highlighting |  |  | ✓ |  |
| Unit Test Generation |  |  | ✓ |  |
| Modular |  | ✓ |  |  |
| Robust | ✓ |  |  |  |
| Responsive | ✓ |  |  |  |
| Documentation for Support |  |  | ✓ |  |
| Easy to Use UI |  | ✓ |  |  |

# Resources

For this project several resources will be needed to enable a smooth development process. We will attempt to utilise a variety of different resources that will make production easier.

A GIT repository will be set up for the group to allow both members to share and work on documents or code. This will help collate all the work in one location and allow for the group to work simultaneously on a piece of work. The repository can also track individual contributions to the project so a measure of work done can be applied.

The Eclipse PDE is another resource that will be used by the group during implementation of the projects product. This resource however is already available within Eclipse when developing plugins, therefore no additional action will be required.

Email is another resource that will be used, this is to help notify each other of any new work to be done or ask for any feedback. This resource was already readily available and therefore was used from day one of the project.

# Requirements Confirmation

The requirements in this document were confirmed to be accurate and correct by the client Gary Allen on the 8th December 2014.

# Conclusion

From the analysis of the requirements several concluding thoughts are to be made. It is important that we consider the needs of both our client and the potential users of the plugin. The requirements outlined present a challenging problem that when solved will lead to a new means of transitioning new programmers from a basic environment like BlueJ into utilising he features of a fully-fledged IDE such as Eclipse.

This project should attempt to meet all requirements outlined in this document. Failure to meet the requirements would lead to a failure of the project as a whole. Throughout the development of the project we should be mindful of the various requirements at all times, if time becomes an issue we can systematically remove features fulfilling requirements as needed.