Introduction

I have been tasked to investigate and re-engineer Project “JFreeChart”. Firstly I investigate the system to gain an understanding and get familiar with the technical and business logic. Secondly we carry out static analysis and dynamic analysis to get some visualizations and graphs. We analyse our visualizations to see which parts of the system could be problematic or good candidates for refactoring or rewrites.

Project Investigation

There were multiple re-engineering patterns used in the investigation of “JFreeChart” project. The patterns used have been listed below along with how it was applied and what information was found using the pattern. I summarize the initial investigation with my findings using all the patterns listed.

Patterns

**Read all the Code in One Hour** (First Contact)(NEED TO COME BACK)

Pattern Application – The project source code was cloned from Github. JFreeChart uses Maven where I setup the packages and prepared it for Eclipse. I inspected the code using Eclipse. I had firstly went to see if there was functional / unit tests and ran all of those to make sure the system was running as expected. The project uses Javadocs and it is extensively commented within the sourcecode where I was able to find the developers intentions on some of the project functionality.

Pattern Information Finding - The project was developed using Java where it uses Object-Oriented principles. The project seems to be using the generic Java Programing Style Guidelines even though this isn’t explicitly declared on their website or forums. The project is considered to be classed as a medium sized project. There are varying degrees of code quality found within the project but the overall quality was considered to be above average. The project uses various programming idioms / design patterns used in most modern Object-oriented programming projects although I did manage to find a few code smells whilst looking at the org.jfree.chart.renderer.\* and org.jfree.chart packages. I had a look at some abstract classes and interfaces to see the design intention. I had found some very large classes with over 1900 lines of code and uses conditional statements extensively mainly from org.jfree.chart.renderer.\*

**Skim the Documentation** (First Contact)

Pattern Application – The majority of the documentation found in the project was found on

* Website - <http://www.jfree.org/jfreechart/>
  + Project overview
  + Project Samples (Screenshots and a Demo application that needs to be installed)
  + API Documentation (Up to date JavaDocs)
  + Frequently Asked Questions
* Project Source Code - <https://github.com/jfree/jfreechart>
  + ReadMe.md (Quick overview and versioned change logs)
  + ReadMe.txt (Project setup and technical documentation)
* Documentation - <https://sourceforge.net/projects/jfreechart/files/> (Includes versioning of documentation)
* Project History - <https://sourceforge.net/projects/jfreechart/files/1.%20JFreeChart/>
* Bugs - https://sourceforge.net/p/jfreechart/bugs/
* Patches - https://sourceforge.net/p/jfreechart/patches/
* Feature Requests - https://sourceforge.net/p/jfreechart/feature-requests/

We elicited as much information from the above sources of information. By reading through the various sources I was able to find out –

* the project’s overview and project intention for developers and users
* How up to date the documentation is compared to the source code
* Key points in the project and any common objects
* The technology stack and a little bit of the project design
* Requirements specifications came from the projects version control and desired features from the featured requests on source forge.

Information Finding – Our findings found out that the documentation is up to date with the most recent core project release even though this was in the year 2014 it still addresses the main functionality. We were not able to find any architectural information, design information and models describing the system. We did not find any database information either however we did find an up to date JavaDoc / Api Documentation. The Website provides a good overview of what the system does and what charts it supports. There is also instructions on how to install and build the system. We did find any email address of the chief designer on the system which we could contact later on if we need to. There is a Forum where we are able to find out other information if we need to.

Do a Mock Installation (First Contact)

Pattern Application – A JFreeChart Demo from <http://www.jfree.org/jfreechart/samples.html> was used to install and use JFreeChart Samples.

1. Analyze the Persistent Data (Initial Understanding) //Probably not going to do this
2. Speculate about the design (Initial Understanding)
3. Study the Exceptional Entities (Initial Understanding)
4. Refactor to understand (Detailed Model Capture)
5. Step through the execution (Detailed Model Capture)
6. Look for the contracts (Detailed Model Capture)
7. Learn from the Past

Investigation, Analysis, use evidence, refactor

Investigation (1-2 pages)

1. Patterns
   1. Pattern name
   2. How it was applied to this project
   3. Which information was identified by applying the pattern
2. Findings
   1. What did you learn about the system? Use the object-oriented reengineering patterns chapters (specifically the kinds of questions they consider) to guide your discussion

From your analysis you should identify one class, or a group of classes, upon which to focus your re-engineering efforts.

You should write up your findings over 4-7 pages (including any charts or other visualisations)

Document changes 2-3 pages (include some evidence that your change has made a significant difference)