LINGI2252 ASSIGNMENT 2 REPORT

Group 17
Xiao XU
Xavier CROCHET

Kim MENS
Sergio CASTRO

Contents

1 Introduction

In the first report, we try to do a manual analysis of *Glamour*. This first step gives us a global overview of *Glamour* and helps us to understand how *Moose* and his tools works. Because of the size of the system (about 270 classes), we use some tools in order to get some hints on where to begin. It wasn't exactly what was asked in the statement of the first report but now that we have all the tools in our hand, we can start looking more effectively for the needles in the haystack that Glamour is.

Moreover, we found some code duplication in the source code (but relatively few compared to the size of the framework) and the utilisation of the *Patern Obeserver* in the GLMPresentation class. We thus conclude that Glmamour was globally well-designed, despite the few code duplication.

The report will be divded in two parts:

1. We start by analyse the code using crieria not identifiable by metrics to see
   - How well the code is commented.
   - How good the naming convetion is.
2. Then, we develop a set of thresolded metrics we'll use later to analyse the code.

2 Code analysis

Architecture

3 Overview Pyramid

4 Complexity

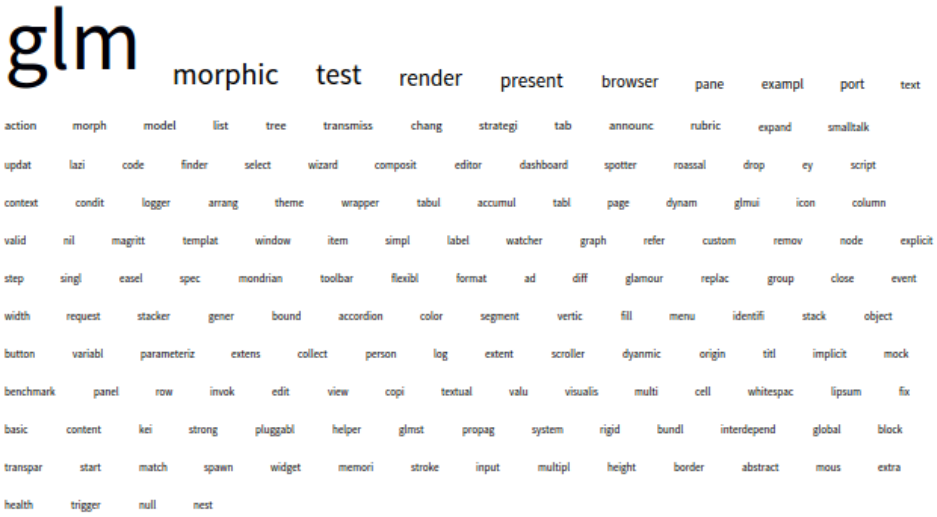
5 Comments

Commenting code is important to improve **readability** and the **reusability**. It make it easier for developpers to contiue a project or for student to analyse it.

There is so few comments in the code that we decide to not use a metric to analyse this part. In deed, there is only 768 comments! That's about 3 comments for each class.

6 Naming Convention

A good naming convention is also an important point in order to imrpove **readability** and **reusability** of the code. Using the *Name Cloud* utility from moose, we can display the following figure, showing the most used words **for naming classes** in the framework.



As we can see **GLM** is the suffix coming up the more often. Other suffix such as **test render** or **morphic** arise too. It seems that classes are named quite correcly in order to provide meta-information about their use.

We can perform the same analysis for methods or variable.Globally there is no bad smells coming up here. There is not randomly named classes variable or methods. Anyway, naming convention is quite a controversial issue, so we are not going deeper in the analysis here.

7 Metrics-based analysis

7.1 Introduction

8 Conclusion

9 Annexes