Evaluation of the Application of Design Patterns by Using Classification with a Support Vector Machine

Jonatan HERNANDEZ[†], Kubo ATSUTO[††], Hironori WASHIZAKI[†], Yoshiaki FUKAZAWA[†]

Design patterns are well known for their usefulness in software development. However how to measure its correct application quantitatively is still a problem. The objective of this proposed technique is to offer a process to make an evaluation when a Design Pattern was applied by using the change of values of selected metrics in the source code.

1. Introduction

Design patterns are well known for their usefulness in software development. The question that I try to solve in this work is: Is it possible to measure them quantitatively? Is it possible to know if a Design Pattern has been applied correctly? This is a problem that has not been treated deeply enough.

The objective is to provide a view of the evolution of the source code from a metrics point of view. This metrics will be related to one particular structure: Design Patterns. This structure is the one that we are trying to identify and to measure in the source code. We also make the assumption that the quality of the design is related to the presence of Design Patterns. Summarizing the characteristic in which we will focus is the application of the Design Patterns and its effects in the metrics.

To measure the metrics, we devised a small process in which we have different states of the source code that represent its evolution through time.

Finally the metrics chosen to measure are a set of Object Oriented metrics. This metrics are obtained by using an Eclipse plug-in.

In the following section will be a description of the process use to obtain the metrics, and to generate a model for classification using a SVM.

2. Proposed Technique

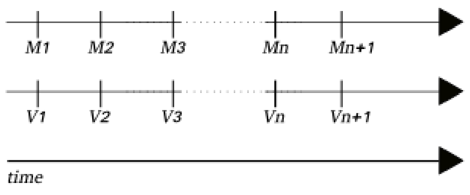
2.1. The Process

The process proposed for evaluation consists of the following general steps: 1) Obtain source code *before* the application of a Design Pattern. 2) Measure the metrics of this source code M_b3) Obtain the source code *after* the application of a Design Pattern. 4) Measure the metrics of this source code M_a. 5) Obtain the difference of the values of the metrics (M_b -M_a) 6) Use this difference to create an input file for the SVM. 7) From this data the SVM can creates a model for classification 8) This model can be used to classify source code were a Design Pattern is applied and made a prediction if it is correctly applied or not.

In the previous process there are to possible scenarios for an application of a Design Pattern: 1) Failure: The Design Pattern has been removed, or 2) Success: the Design Patterns has been added. For the purposes of this study, we will assume that if a Pattern was removed from the source code, always can be considered as a failure.

In the case of a failure scenario, only the steps 1) and 3) change: 1) Obtain the source code when the Design Pattern *is included* in the code … 3) Obtain the source code *after* the pattern has been removed from the source code. The rest of the steps are the same.

To find the metrics it is necessary to identity when to measure the metrics. In order to do this we can see the following diagram where M = set of metrics, V = version. Each set of metrics is composed of individual metrics *m1*, *m2*, *m3* … *mk*. I.E. *m1* could be Depth of Inheritance Tree (DIT), an *m2* could be Number of Methods NOM, etc.



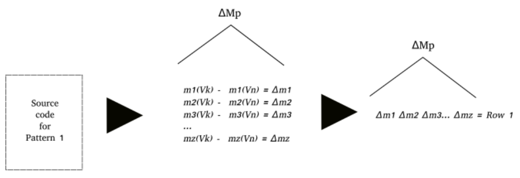
Now, supposing that version V3 has a Design Pattern, i.e. Singleton, and then in version Vn this Design Pattern (Singleton) was deleted, we obtain the metrics in the following way:

Set of metrics from V3 = M3, and set of metrics from

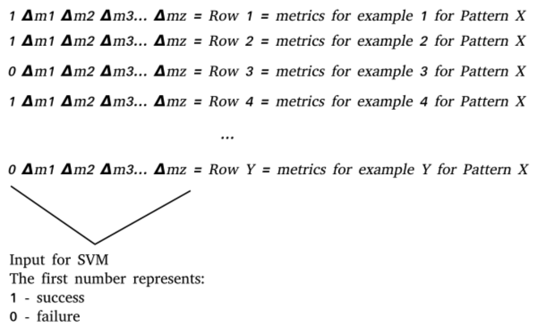
†Waseda University, ††National Institute of Informatics

Vn = Mn.

Mn – M3 = Δ*Mp* = Delta of Metrics for a Pattern *p*, which is the difference of value of each individual metric from version *V3* and version *Vn*.



The next step is input this data in a format suitable for the input of an SVM.


1 Δm1 Δm2 Δm3... Δmz = Row 1 = metrics for example 1 for Pattern X
1 Δm1 Δm2 Δm3... Δmz = Row 2 = metrics for example 2 for Pattern X
0 Δm1 Δm2 Δm3... Δmz = Row 3 = metrics for example 3 for Pattern X
1 Δm1 Δm2 Δm3... Δmz = Row 4 = metrics for example 4 for Pattern X
...
0 Δm1 Δm2 Δm3... Δmz = Row Y = metrics for example Y for Pattern X
Input for SVM
The first number represents:
1 - success
0 - failure

In this way we fill a file for a particular type of Design Pattern. Each row is an example of a success or a failure of that particular Design Pattern. Then we give that file as an input for the SVM, and we obtain a model that can be use to classify future inputs.

2.2. Expectations.

When applying a Design Pattern certain values from the difference between the metrics can be interpreted as an indication of a success or a failure. This can be done because of the model previously obtained.

2.3. Problems.

One problem that has to be faced since the beginning is the recollection of data. One part of the problem is identifying the Design Patterns inside the code. The second part of the problem is getting the source code for that pattern. A variation of the problem is to find information from a repository history (i.e. CVS, SVN) where a pattern was deleted. These two problems have proved to be very difficult to solve in an automatic way.

2.3.1. Problems with Open Source Projects.

One will be oriented to think that in the open source projects there is a lot of pattern work. Here the problem is to identify were the Design Patterns have been used. The problems with the open source projects have been to find the patterns, and then retrieve the appropriate versions of the source code in an efficient way.

2.4. Results

The analysis made so far was made with examples from books, primarily two sources: Refactoring to Patterns and Pattern Hatching.

From these examples some models have been obtained, but the samples are still few, so in this point there is a need for more samples.

2.5. Who can benefit from this

After the model has been created it can have several uses. The most important is the capacity for a developer to judge if a Design Pattern is correctly applied or not. This can be done by measuring the metrics of the code to check, and then giving those metrics as an input to the SVM and the model previously obtained.

2.6. Points for improvement.

So far there have been some problems, specifically with the collection of data. To improve this part one way is the automation of parts/all of the process.

Another problem is to find suitable examples; so far all the examples have been extracted from books. One more point is to get data from other sources, like open source projects (overcoming the current problems).

Reference Works

[1] Marek Vokac, Defect Frequency and Design Patterns: An Empirical Study of Industrial Code, IEEE Transactions on Software Engineering, 2004

[2] Kerievsky J.: Refactoring to Patterns, Addison-Wesley, 2008

[3] Vlissides, J: Pattern Hatching – Design Patterns Applied, Addison-Wesley, 1998

[4] L. Prechelt et al., A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions, IEEE Transactions on Software Engineering, 2001

[5] T. H. Ng et al., Work experience versus refactoring to design patterns: a controlled experiment, 14th ACM SIGSOFT , 2006

[6] http://metrics.sourceforge.net/, Eclipse Plugin Metrics 1.3.6