Observer Pattern in FreeCol

Karl Fredrik Gudjonsson
Linköpings University, Sweden
kargu328@student.liu.se

***Abstract*—The abstract goes here. This paper research if the implementation of the Observer pattern increase the maintainability/extendability in FreeCol...**

***Index Terms*—Design patterns; FreeCol; JAVA; NOP; Observer Pattern**

I. Introduction

ONE of the hardest problems to solve in programming is to develop a good software design, in order to keep the project easy to understand, extensible, maintainable, and efficient. A solution to this are thought to be using design patterns when designing software. The main advantages of using design patterns are[1]:

1) Using design patterns improves the programmer productivity and the software quality.
2) Beginners can increase their programming skills by applying design patterns in their learning process, by recognizing similarities between problems they encounter.
3) Design patterns improves the understanding between both developers and maintainers working on the same project

As of yet there are no concrete evidence for this, and this subject needs further studying. However there are some claims that supports that the use of design patterns improves the understanding between developers and maintainers. This papers main focus is how the use of the observer pattern in *FreeCol* improves the maintainability/extendability.

In this paper the aim is to answer the following question:

How does the implementation of observer patters in FreeCol affect the maintainability/extendability as measured by NOP (number of polymorphic methods)?

II. Theory

A. Design Patterns

Design patterns are a way of structuring code to solve reoccurring problems when designing software. A design pattern is not a final design that can be copied straight into the source code. It is more of a description for how to solve a problem. [2]

B. The Observer Pattern

The observer pattern goes under the category of behavioral patterns, with the intet to: ”Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.” [3]

The problem we want to solve by using the observer pattern is to contain consistency between objects that all depend on the same data. The observer pattern consists of one subject class containing a state which is of interest of different observer classes, and notifies the observer when there are changes made.

C. FreeCol

FreeCol is an open source game written in JAVA based on the turn-based strategy game Colonization[4]. In this paper the focus in the code will be in and arround the class **net.sf.freecol.common.model.TrasactionListener.java**.

III. method

A. Number of polymorphic methods

The number of methods that are using polymorphism (the ability to override an existing method when compiling or running the program) to override a method of a base class. The purpose of using this is mainly to increase the extendability and flexibility. But this comes with the cost of the design being harder to understand [5].

IV. result

Results goes here.

V. analysis

Analysis goes here.

VI. Conclusion

The conclusion goes here.

References

[1] C. Zhang and D. Budgen, "What do we know about the effectiveness of software design patterns?" *Software Engineering, IEEE Transactions on*, vol. 38, no. 5, pp. 1213–1231, 2012.

[2] M. P. L. Prechelt, B. Unger-Lamprecht and W. Tichy, "Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance," *IEEE Trans. Software Eng*, vol. 28, no. 6, pp. 595–606, 2002.

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[4] Freecol. [Online]. Available: http://www.freecol.org/about.html

[5] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *Software Engineering, IEEE Transactions on*, vol. 28, no. 1, pp. 4–17, 2002.