

Patterns as Signs

Gina Häussge

January 17, 2006

Abstract

In this essay a short summary of the paper “Patterns as Signs” by James Noble and Rober Biddle[1] will be given. We will see what *Design Patterns* and *Signs are*, how they can be combined, and what we can learn and achieve by looking at patterns as signs, namely answering some common questions about patterns in general.

1 Introduction

Before taking a look at what the interpretation of design patterns as signs can do for us, we first have to take a look at what they are.

1.1 What is a *Design Pattern*?

A design pattern is a solution to a common recurring problem in programming. It consists of a name, a description of the problem the pattern addresses and the context in which it can be used, a description of the solution the pattern implements and a list of consequences and trade-offs associated with the usage of the pattern. Design patterns do not give a finished and directly implementable solution to a specific situation, they rather are abstract receipts for dealing with certain kinds of problems often encountered in software design which need to be adapted each time to the specific situation encountered.

1.2 What are *Signs*?

In the field of linguistics, a sign is defined as a combination of a concept, the *signified*, and its sensorical representation, the *signifier* (see Fig. 1). The signified can be something like the concept of a tree or of a color, while the signifier is a spoken or written word representing it, like “tree” or “yellow”. Neither the signifier nor the signified are unique in their relation: A signifier can represent multiple signifieds (e.g. the word “red” can be a color or the past tense of “to read”) and a signified can have multiple signifiers (e.g. the concept of the color yellow has different words in different languages: “yellow”, “gelb”, “giallo”, “kiiro”). Only the sign, the combination of the signified with the signifier, is unique.

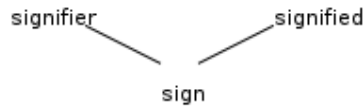


Figure 1: The components of a sign

The study of signs in society is called Semiotics.

2 Patterns as Signs

2.1 Implementation

As we learned in 1.1 **What is a *Design Pattern*?**, a pattern consists of several parts, among them a name, a description of the problem it can be applied to and a description of the solution it implements. This descriptive part of the pattern is a sign, with the signifier being the solution, and the signified being the problem and context of the pattern or - as Noble and Biddle call it - it's intent (see Fig. 2).

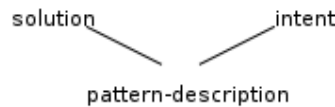


Figure 2: Pattern description as sign

The pattern itself is also a sign: The signifier is the name assigned to the pattern, the signified is the pattern-description, which we just identified as a sign as well. So we get a so called second-order semiotic system, meaning that we have a system which is composed of two signs, with one sign being a component of the other (see Fig. 3).

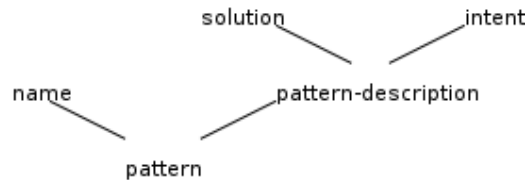


Figure 3: Pattern as sign

2.2 Answering common questions

Now that we have defined patterns as signs, we can give answers to a couple of common questions associated with patterns.

2.2.1 How can two patterns have the same implementation?

Some patterns, like for example the Strategy and State patterns, have identical structure diagrams and therefore identical solutions. However, a pattern does not only consist of it's implementation like we learned above, it's implementation is rather just the signifier of it's description. As we know, signifiers can be parts of more than one sign, and the unique sign is a composition of a signifier and a signified: A pattern is a solution to a specific problem, not just a stand-alone solution. So we still have different patterns, although we have identical solutions, as the problems, the intents, differ.

2.2.2 How can one pattern have more than one implementation?

The Adaptor pattern consists of four different sub patterns. They all share the same name - "Adaptor" - but are implemented differently, their solutions and intents are not the same (although similar). Looking at it from a semiotics angle, we see that we have different patterns, sharing the same signifier (the name "Adaptor"), but having different signifieds, the pattern-descriptions. Therefore each Adaptor variant is somehow a different unique sign, they just share a name.

2.2.3 How can one pattern solve two or more problems?

Taking a look at the Proxy pattern, we see that it claims to solve several different problems (four according to [2], seven according to [3]). In terms of the presented semiotic model, we have several different patterns, each having the same signifier ("Proxy"), and each having a different signified which consists of an identical solution (the proxy design) and different intents (the different problems the Proxy variants solve).

2.2.4 How can one pattern have more than one name?

The Decorator and the Wrapper pattern are synonyms for each other - both describe the same solution for the same problem, but are referred to by different names. From the semiotics point of view, we have two different patterns with different signifiers ("Wrapper", "Decorator"), but identical pattern-description second-order signs.

2.2.5 How can different pattern-descriptions share the same name?

The opposite to the previous point are patterns like the different versions of the "Prototype" pattern. Here we have a name shared among many completely different patterns, signs with the same signifier but different signifieds, namely the pattern-descriptions. In such a case, the intended pattern is found by context

when communicating, analogue to human language when the same word is used for different mental concepts (e.g. the german word “Bank” for a bench or a bank).

2.2.6 How can the relationships between patterns be explained?

Semiotics can also help to define the relationships which occur between patterns: Usage, alternatives and specialisation.

A pattern *uses* another pattern when we have two patterns with different intents where one needs to apply the other as part of being applied itself. In the semiotics approach, the first pattern's solution is related to the second pattern.

Two patterns are *alternatives*, when they provide different solutions to the same problem. So we have two different patterns with two different names and two different pattern-descriptions, which consist of different solutions but whose intents can be described as being similar.

There are patterns which are *specialisations* of others. Speaking in semiotics, they have similar intents and similar solutions, but the specialised pattern is more complex than the general pattern. An example of such a relation is the Factory Method pattern, which is a specialisation of the Hook Method pattern and changes the type of the object that is created.

2.3 Further Aspects

2.3.1 Misinterpretation

Semiotics explain how misinterpretations can occur when confronted with patterns. When reading the source code of a program, it is quite easy to misunderstand the patterns we see: What we think to be an implementation of the Observer pattern in truth is a Mediator pattern or something completely different that just looks and feels like an Observer pattern. Such misunderstandings can happen whenever using signs, so of course they also can happen with patterns, as patterns are nothing else than two-order signs.

2.3.2 Alternatives to pattern languages

The semiotic approach in understanding patterns offers an alternative for structuring collections of patterns. Instead of organising them in a hierarchical way, which makes it necessary to ensure that each pattern describes a single solution to a unique problem and therefore to invest quite a load of work into categorising and splitting known patterns, they are organised in an encyclopaedia like way. The advantage of this kind of organisation is the less strict structure in which the patterns are ordered, making evolutionary progress of the known vocabulary easier.

2.3.3 Future research

We so far have learned that patterns can be described as two-order signs. But the authors of the paper also plan to take a closer look at the semiotic structure of object-oriented designs and design patterns. Not only the patterns themselves, also the ways of presenting them partly, namely class diagrams, are signs, as are presentations of pattern collections. And not only patterns can be formulated as signs, data structures and algorithms can be described in a semiotic way as well.

3 Conclusion

Treating patterns as signs allows us to analyse common issues with the understanding and interpretation of patterns in a linguistic way instead of a logical or mathematical one, making it possible to give solutions to otherwise unsolvable questions and issues.

References

- [1] J. Noble, R. Biddle. *Patterns as Signs*. ECOOP 2002, LNCS 2374
- [2] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley 1994
- [3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *Pattern-Oriented Software Architecture*. John Wiley & Sons 1996
- [4] Wikipedia. *Design pattern (computer science)*.
[http://en.wikipedia.org/wiki/Design_pattern_\(computer_science\)](http://en.wikipedia.org/wiki/Design_pattern_(computer_science))
Accessed: 17.01.2006
- [5] Wikipedia. *Sign (linguistics)*.
[http://en.wikipedia.org/wiki/Sign_\(linguistics\)](http://en.wikipedia.org/wiki/Sign_(linguistics))
Accessed: 17.01.2006