

# From the OO Design Principles to the Formal Understanding of the OO Design Patterns

JAVIER GARZÁS  
ALTRAN SDB Consultant  
Projects Engineering Research Group  
ALTRAN SDB  
C/ Ramírez de Arellano, 15. 28043,  
Madrid - SPAIN  
jgarzas@altransdb.com

MARIO PIATTINI  
Alarcos Research Group  
Escuela Superior de Informática,  
University of Castilla-La Mancha  
Ronda de Calatrava, s/n. 13071,  
Ciudad Real - SPAIN  
mpiattini@inf-cr.uclm.es

## 1 Introduction

Up until a few years ago the Object Oriented (OO) knowledge was totally implicit but fortunately it is now being specified and popularized in different forms: principles, heuristics, patterns and more recently, refactoring techniques. The difference between these concepts is generally unclear and moreover not all of them have received the same amount of attention or have reached the same degree of maturity. In fact, with the exception of the contributions of Liskov, Meyer and R.C.Martin, a strong knowledge does not exist on design principles. Regarding OO design heuristics the only works to which we can refer are those of Riel and Booch. Patterns, however, are the elements that have undergone the greatest evolution and proof of this is the existence of numerous publications on the theme (Coad, Gamma, Buschmann, Fowler and Rising , etc.). Lastly, refactoring techniques are characterized by their immaturity, although it is true to say that this topic is rapidly gaining acceptance, largely thanks to Fowler's work

The problem confronting the designer is how to articulate all this explicit knowledge and to apply it in an orderly and efficient process in the OODA, in such a way that it is really of use to him. In fact, in practice, even such advanced subjects like patterns have this problem, this situation could give rise to incorrect applications of the patterns.

In this paper we are going to make an in-depth analysis of the relationships between principles and patterns, as we believe that principles can be useful when systematizing the application of patterns in OODA

## 2 Relationship Between Principles and Patterns

An OOD principle can be defined as a set of proposals or truths based on experience that form the foundation of OOD and whose purpose is to control this process. Some principles are the next (other principles apart from those described here may exist but we are limited by the length of this paper):

- **Open-Closed Principle (OCP):** A module should be open for its extension and closed for its modification.
- **Substitution Principle (SP):** The subclasses must be substitutable by their base classes.
- **Dependency Inversion Principle (DIP):** Depend upon abstractions. Do not depend upon specifications.
- **Interface Segregation Principle (ISP):** Many client specific interfaces are better than one general purpose interface.
- **Default Abstraction Principle (DAP)**
- **Interface Design Principle (IDP):** "Program" an interface, not an implementation.
- **Black Box Principle (BBP):** Favour the object composition over class inheritance.
- **Don't Concrete Superclass Principle (DCSP):** Avoid maintaining concrete superclasses.

In general, we can state that in order for an OO system to be of a certain quality this shouldn't violate any principles. On the other hand, patterns contribute to an efficient design but in general the exact relationship between principles and patterns is unknown or more specifically we do not know which principles/s is/are ensuring each pattern.

So, for example, in order to conform to the DIP, one of the strategies could be to use the abstract Factory pattern. The purpose of other patterns such as Prototype, Factory method, etc. is more to perform the Abstract Factory than to directly conform to a principle. Therefore, we can conclude that there are patterns that directly allow a principle to be complied with, whilst other patterns are more related to patterns than to principles. Consequently, patterns could be classified according to the principles they follow. The principles would even enable us to create a different catalogue of patterns to that currently existing (in most cases they are simply presented in alphabetical order). Checklists of principles could also be drawn up which assess the design and offer us solutions patterns that ensure that they are complied with. We can specify more and considering their relationship with the patterns, the principles can contemplate one or several of the following types:

<b>Type 1</b> , the pattern contributes a good solution to the resulting model of the application of the principle (“from the principle toward the pattern”).	<b>Type 2</b> , the pattern completes or contains at the principle.	<b>Type 3</b> , the principle can improve a solution to which has been applied a pattern previously (“from the pattern toward the principle”).
---	---	--

The next table shows an analysis of the principles mentioned in previous epigraphs and their relationship with each pattern of the detailed by Gamma in function of the previous types. We can observe as the relationship of patterns has been ordered alphabetically, we can obtain this way an objective order, later on, and based on the principles, we will be able to obtain analogies.

Several considerations, uses and investigation lines can be extracted, some examples are the following ones:

- It allows to break down in forces of smaller grain each one of the patterns, facilitating the study of elements common to all the patterns of oneself character: “patterns in the patterns” or “meta-patterns”.
- It allows to guide the use of patterns, since it is easier to know how to apply in a correct way a principle that a pattern and once applied the principle is easy to arrive to the pattern. This facilitates us the pattern's good use, in their fair measure, without abuse. For example, the use of NSCP implies us to use creational patterns and this assures us that our system is written in function of interfaces and not in function of implementations.
- It allows a formal study of micro architectures.
- It allows to obtain the forces (principles) that conform the pattern and how depending in its way of incidence in the pattern (type 1, 2 or 3) this can be of different character, examples:
  - We can observe as Abstract Factory, Builder, Factory Method and Prototype maintain an almost identical kernel of principles while Singleton doesn't complete any principle. Singleton is not a micro architecture (it only describes a class), Singleton treats the creation of objects but he doesn't make it with the same character and the same abstraction that the other four creational patterns, an Idiom considers it. With regard to the four remaining creation patterns, we observe that they complete the same principles except Builder, since this has the same character that the previous ones but by means of a composition strategy. As we see the study of the principles that intervene in a pattern allows us, among other many things, a finer and based classification.
  - We observe as any micro architecture with some hierarchy that we want to consider design pattern should complete (type 2), at least, the following principles: OCP, SP, DIP, IDP and DCSP.
  - We observe that in patterns structurally identical as State and Strategy the same principles are completed and with the same character.
  - All pattern that completes OCP, SP, DIP, IDP and DCSP in type 2, ISP and DAP in type 3 and BBP don't contemplate it is classified (according to Gamma's book) as of behavior.
- We will be able to look for and/or to validate new design patterns observing if they complete certain of meta-patterns.

Principio	OCP			SP			DIP			ISP			DAP			IDP			BBP			DCSP		
Patrón	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Abstract F.																								
Adapter Class																								
Adapter Object																								
Bridge																								
Builder																								
Chain R.																								
Observer																								
Prototype																								
Proxy																								
Singleton																								
State																								
Strategy																								
Template Met.																								
Visitor																								

The principles allow us to extract good practical OO, observing how the patterns are based and how they are connected with the design. Our final aim is to offer a detailed systematization of principles, heuristics, patterns and refactoring techniques (together with their respective interrelationships) which will facilitate their application for the designer.