Chapter 2.12
Object-Oriented Software Design Patterns Applied to IT Management Theory

Eric Tachibana
National University of Singapore, Singapore

David Ross Florey
Merrill Lynch Global Business Technology, Singapore

ABSTRACT

Since the mid to late 1990's, object-oriented software design patterns have proven to be a powerful tool in support of software design and product management. However, the usefulness of the methodology need not be restricted to the technical domain alone. In fact, the design pattern methodology represents a powerful tool that can also be used in support of it management at a business level. In this paper, we discuss the design pattern methodology, provide an example of how the methodology could be implemented to solve a business problem, the multivariate vector map (mvm), and then apply the mvm pattern to the problem of choosing an it outsourcing strategy as a means to demonstrate its effectiveness to it managers and to it outsourcing vendors.

INTRODUCTION

Since they were first introduced in the mid to late 1990's, Design Patterns for object-oriented software development have become a powerful force in software engineering. However, the efficacy of Design Patterns need not be limited to the technical realm.

In this chapter, we present a broad paradigm through which business managers can apply the object-oriented Design Pattern methodology to help solve strategic management problems and to better predict the market outcomes of their choices.

We will begin the discussion by briefly recounting the history of Design Patterns in software engineering. Next, we will discuss and define Design Patterns themselves and explain how the Design Pattern paradigm is used effectively in object-oriented software development.

Having defined our base of reference, we will then explore the ways in which the Design Pattern paradigm can be applied to the process of strategic management decision-making by examining how several popular management decision-making tools can be refactored as Design Patterns.

Finally, we will propose our own Design Pattern, the Multivariate Vector Map (MVM), and demonstrate its effectiveness by applying it to the problem of strategic IT Outsourcing. In support of the case study, we have gathered data from a large multinational IT Services company based in Singapore that was able to provide the information and statistics required for the generation and implementation of the MVM Design Pattern.

WHAT ARE DESIGN PATTERNS?

Over the last decade, Design Patterns have emerged at the forefront of object-oriented software engineering and have inspired dozens of books, conferences, online communities, and well-designed software solutions.

Although the conceptual origins of Design Patterns were being postulated in the early 1990s by several sources, it was the groundbreaking book, "*Design Patterns: Elements of Reusable Object Oriented Software*" by Gamma, Helm, Johnson and Vlissides, affectionately known as the Gang of Four (GoF), which truly launched the Design Pattern paradigm into the popular consciousness. [FOWLER, 5]

The core thesis argued by the GoF explains that rather than solving every software engineering design problem from first principles, good designers reuse abstracted designs developed (and copied) throughout their careers. When software architects discover designs that work, they continue using those designs over and over again. In the words of the GoF,

> *These patterns solve design problems and make object-oriented designs more flexible, elegant, and ultimately reusable….A designer who is familiar with such patterns can apply them immediately to design problems without having to rediscover them.…Once you know the pattern, a lot of design decisions follow automatically.* [GoF, 1]

Of course, the recognition of 'good' designs within software architecture was no epiphany. Any of the almost two dozen design patterns cataloged in the GoF's revolutionary book were all designs that any senior, seasoned architect of the era would have intuitively recognized. The GoF's truly novel contribution to the field was their definition of a paradigm with which to understand the process of recognizing, documenting, communicating and implementing Design Patterns. As they explain it,

> *The purpose of [our] book is to record experience in designing object-oriented software as design patterns. Each design pattern systematically names, explains, and evaluates an important and recurring design. Our goal is to capture design experience in a form that people can use effectively.* [GoF, 1]

The benefits of using design patterns for object-oriented software engineering have proven to be numerous and significant.

For one, while the object-oriented paradigm addressed reusability at the algorithm and object level, Design Patterns make it easier to reuse successful designs at the architectural level. This result is faster development turnaround and more reliable code.

In addition, Design Patterns make more starkly visible the assumptions and consequences of design choices. This elevates the process of coding from fire-fighting to architecture, allowing developers to think more strategically about their designs. As noted by Shalloway and Trott,