1. はじめに

ソフトウェアのデザインパターン本が流行っている。まだ、日本語に翻訳されているものは少ないが、米国では「デザインパターン」をキーワードとした本の出版が続いている。題名から直感的に内容を推し量ることができるのがその原因だろう。

しかし、私の目から見ると、この流行には大きな誤解がある。すなわち、「デザインパターンを採用すればすぐに生産性が上がる」という幻想である。ある前提の下ではこの命題は成り立つのだが、多くの読者はこの「前提」を忘れている。

その前提とは、「ソフトウェア工学・ソフトウェア科学の成果を土台としてデザインパターンを適用すれば、ソフトウェアの信頼性が向上する」というものである。結果として、欠陥による手直しのフィードバックが減少し、ソフトウェアの生産性が上がる。つまり、デザインパターンの本は「ソフトウェア工学・ソフトウェア科学の採用」を前提として書かれているのだ。

ところが、日本のソフトウェア開発の現場では「ソフトウェア工学・ソフトウェア科学の知識」は前提とされていない。このため、開発現場のプロジェクトでは本来なら避けられるはずの様々な問題が発生している。例えば、私がコンサルタントとして参加した「すべてのプロジェクト」でこのような問題が発生し、プロジェクトは遅れ、ソフトウェアの信頼性は低かった。

このような混乱した状態で、救いをデザインパターンやその前提となるオブジェクト指向技術に求めるのは間違いである。まず、「ソフトウェア工学・ソフトウェア科学」の成果を利用し、その上でオブジェクト指向技術を適用し、さらにそれらをベースとしたデザインパターンを採用すべきなのである。

では、ソフトウェア工学・ソフトウェア科学の成果とは何だろう？　デザインパターンに関連する成果に絞ってみれば、それは情報隠蔽やカプセル化を行う構造化技法をベースとした分析・設計方法論（すなわちオブジェクト指向分析・設計方法

論)、What あるいは「どうあるべきか」を記述する宣言的な仕様記述言語、アルゴリズムとその最先端を行くコンパイラー理論、そしてオブジェクト指向言語技術、さらに人間が主役であるというピープルウェアの概念に基づくチーム編成とプロジェクト管理、ということになる。

そこで本書は、分析・設計段階でデザインパターンの適用を目指す読者を対象として、特定のオブジェクト指向言語は想定せず、実践的に分析・設計が行えるように、分析・設計法と仕様記述法の説明を重視することにした。

また、実際に分析・設計を行う上での思考過程をなるべく明らかにしようと努めた。特に、分析工程の細かい思考過程を書いた本はあまりないので、そのあたりの記述に重点を置いた。

そこで、本書の構成を以下のようにした。

2 章ではデザインパターンを概説する。特に、デザインパターンの評価基準となる「良い設計」の要因について説明する。

3 章は、4 章以後で必要となる個々のデザインパターンの要約である。ここにはデザインパターンだけでなく、分析パターンやアーキテクチャパターンといったものも記述する。

4 章は、本書の核となる部分であり、デザインパターンを意識しながらどうやって分析・設計を進めていくかを述べる。

5 章は、制御系システムの分析・設計をどう進めていくかを述べる。

6 章は、導入の課題と今後の動向を述べる。

クラス図などの記法としては標準になる可能性が強い　UML（Unified Modeling Language)[*1] を用い、仕様記述言語も UML で定義されている OCL（Object Constraint Language)[*2] を使う。UML の記法や OCL の文法は、最初に出てきたところで概要を説明する。

7 章は、付録であり、UML と OCL の概要説明を行う。また、参考文献についても記述する。

---

1. 『UML Notation Guide version 1.1, Rational Software, 1 September 1997』及び付録 7.1 参照。
2. 『Object Constraint, Language Specification,Rational Software, 1 September 1997』及び付録 7.2 参照。