連載　オブジェクト指向と哲学
第 16 回　パターン言語 － ソフトウェアへの浸透

河合 昭男

前回は、パターン言語のキーワードである**無名の質**について考えました。「パターン言語とは何か」の説明は後にして、外堀から埋めてゆきます。

パターン言語は本来の建築の世界よりもむしろソフトウェアの世界に取り込まれてきています。その流れは大きくふたつあります。第１はデザイン・パターンから始まった一連のソフトウェア・パターンの一大潮流です。第２は一連のオブジェクト指向開発プロセスへの影響です。統一プロセス RUP や、その対極として生まれた XP から始まったアジャイル開発ブームにパターン言語の考え方がとり込まれてきました。

●デザイン・パターン

‘94 年に出版された GoF のデザイン・パターンは、ソフトウェア開発の世界に一大センセーションを引き起こしました。

--

“Design Patterns －Elements of Reusable Object-Oriented Software”[1]

邦訳「オブジェクト指向における再利用のためのデザインパターン」[2]

--

この書籍はクリストファ・アレグザンダー（C.A.）のパターン言語に触発されてまとめられたものです。同著の始めに C.A.からの引用があります。

「それぞれのパターンは我々の身のまわりで何回も起きる問題、および、それぞれの問題に対する解法のポイントを記述している。そこで我々は、これらの解法を何万回でも使うことができる。同じ問題に対する同じ解法を何度も何度も最初から考え直さずに済むというわけだ」[2]

オブジェクト指向が「ソフトウェア部品の再利用技術」なら、パターンは「知識の再利用技術」と言える。

単純化すれば「問題と解法」のペアを C.A.はパターンと名付け、そのアイデアがデザイン・パターンになったわけです。パターンという普通に使われている名詞に C.A.は別の意味付けを行い、それが支持されたわけです。

GoF のデザイン・パターンは、過去様々な開発で何度も使われた設計のテクニックを調査し、パターンという視点で 23 のパターンとして集大成したものです。

ちなみにこの書籍の初版では、各パターンのモデルは UML ではなく OMT 法で記述されています。UML0.9 が OMG に提出されたのが'96 年なので、本書が出版された'94 年は正に UML の前夜です。オブジェクト指向開発方法論の統一が議論されていた時代です。

アーキテクチャ・パターン、アナリシス・パターンなど一連のソフトウェア・パターンが次々出版されてきました。従来の開発プロセスの問題に焦点を当てたアンチ・パターンという風変わりな書籍もやはり C.A.のスタイルを踏襲しています。

●パターン集とパターン言語

一連のソフトウェア・パターンはパターン・カタログとしてとても有益なものです。役に立つのだからそれが単なるパターン集なのかパターン言語なのかという議論は利用者にとってはささいな問題です。

UML で表すと、パターン集はパターンで構成されています。個々のパターンは、単独で存在するのでなく他のパターンと関連します。（図 1）

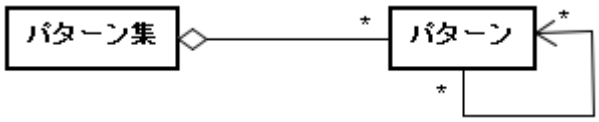

図 1　パターン集

パターン集とパターン言語の違いを UML で表現するのは困難です。パターン言語にはパターン集にはない何らかの特性と振る舞いがある筈です。（図 2）

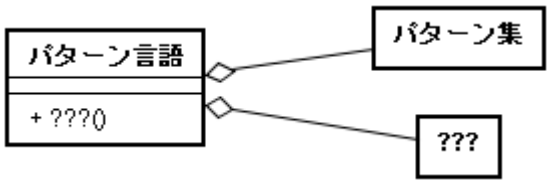

図 2　パターン言語

前回「SFC 学習パターン」をパターン言語の事例として紹介しましたが、これは単なるパターン集、ノウハウ集以上に強力なパワーが内包されています。一つひとつのパターンはそれなりに有益なものですが、相互に関連するパターンのネットワークが強力なパワーを発揮します。前回述べたようにパターン言語と SECI モデルは全く別々の概念ですが、アレグザンダー方式でパターンを整理してパターン言語化してゆくと、いつのまにか知らず SECI モデルの 4 つの象限をカバーしてしまっています。学習パターン（言語）はいつのまにか知識創造パターン（言語）に進化しているのです。

第 1 のムーブメントである一連のソフトウェア・パターンは必ずしもパターン言語化を目指したものではないようです。しかし第 2 のムーブメントであるソフトウェア開発プロセスへの流れはパターン言語を目指したものです。

●XP－eXtreme Programming

‘99 年に出版されたケント・ベックの XP[3]は、GoF のデザイン・パターンと並ぶ一大センセーションを引き起こしました。正にアレグザンダーのパターン言語なのですが、それを前面に出すと開発技術者は引けてしまうので、敢えてパターン言語という言葉を表に出さないでまとめられています。

有名な「ペア・プロ」、「オンサイト・カストマ」、「テスティング」、「週 40 時間」など 12 のプラクティスがパターンです。これらは単独でも役立ちますが、相互に関連があり、すべてを最大限に、極端に（extreme）実践するのが XP という極端な開発プロセスです。

この 12 のプラクティスは、ソフトウェア工学などの理論から導き出されたものではなく、著者達の開発プロジェクトでの経験に基づき集大成されたものなので、プログラマには説得力があります。

●場のクオリティ

C.A.の目指しているものは建物や町のハードウェアだけのクオリティではなく、そこで生活する住民も含めた場のクオリティです。場のプレイヤーである住民が生き生きと活動でき(alive)、住民と建物や町が全体として一体感があり(whole)、住民は居心地のよさを感じられる(comfortable)など人間系を重視しています。

●品質とは？

開発プロジェクトが目指すものは QCD です。顧客の要求を満足する高品質の製品を決められた予算で決められた納期までに完成させることです。

品質という言葉、当然ながら通常は製品の品質を意味します。しかし品質が必要なのは製品だけではありません。要求の品質というものもあります。要求の品質は製品の品質に大きく影響します。さらに開発プロジェクトの品質も製品の品質に大きく影響します。

XP から始まったアジャイル開発がプログラマに支持されたのは、人間系も含めた開発の場のクオリティに焦点を当てたからです。クオリティのある開発プロジェクトからクオリティのある製品ができる。

●XP が目指す「場のクオリティ」

XP が目指すものは、開発チームのメンバーをいかに活性化させるか、モチベーションを上げるか、生産性を上げるかにあります。12 のプラクティスは、開発プロジェクトという

場のクオリティを作りだす源となる実践原理となるものです。プロジェクトのクオリティが結果として製品のクオリティを創出します。

場としての開発プロジェクトの主なプレイヤーは、依頼主である顧客、完成したシステムの利用者、開発メンバーです。XP にはオンサイト・カストマというプラクティスがありますが、このカストマは依頼主と利用者の代表です。XP の 12 のプラクティスはこの開発プロジェクトのクオリティを作ります。XP がなぜプログラマに支持されているかというと、人間重視の開発の場のプレイヤーみんなが生き生きと仕事ができ(alive)、メンバー間および開発する製品の一体感のある組織であり(whole)、開発メンバーは快適で居心地のよい環境で仕事ができる(comfortable)からです。

●ふたつの知識

パターン言語とは知識の形式化の技術です。組織および個人の経験や試行錯誤から蓄積された暗黙知から新たな形式知を創出する技術です。

第 1 のムーブメントはパターンを設計知識の再利用技術として取り込んだ。第 2 のムーブメントはパターン言語を開発プロセスの知識として取り込んだのです。

【参考書籍】

[1] Erick Gamma 他, “Design Patterns－Elements of Reusable Object-Oriented Software” , ADDISON-WESLEY, 1994

[2]本位田真一、吉田和樹【監訳】、「オブジェクト指向における再利用のためのデザインパターン」、ソフトバンク、1995

[3]Kent Beck, ”eXtreme Programming eXplained”, ADDISON-WESLEY, 1999