パターン指向開発プロセス再構築
A pattern oriented process refactoring

河合昭男†
Akio Kawai
†（有）オブジェクトデザイン研究所
† Object Design Laboratory,Inc.

要旨
大規模ソフトウェア開発プロセス RUP や XP などアジャイル系と呼ばれる短期小規模開発プロセスは過去の様々な個人・組織の経験・試行錯誤を通して得られた知見からなるプラクティスをベースにして構築されたものである．これらのプラクティスはそれぞれ開発上起きる問題の解決策であり，パターンとして捉えることができる．パターンから構築された既成のプロセスをカスタマイズして開発現場で使用するのではなく，各プロジェクト利害関係者が共有する価値と基本原則に基づく最適なパターンを抽出し固有のプロセスを作るのが本来あるべき姿である．そのために，パターンを知識ベースとして登録・検索・活用できる仕組みにアレグザンダーのパターン言語の技法が適用できないだろうか．本稿はそのアイデアを述べるにとどまる．

1. はじめに

メインフレーム主流の時代から各社試行錯誤を繰り返しながら社内標準開発プロセスを構築してきた．それらの大部分はウォーターフォール（以下 WF と略称）型と総称される．WF 型の問題はリスク先送りとなり，問題が噴出した時点での設計変更が容易でないことである．また利用者からの仕様変更・拡張の要求に対処できる開発スタイルは，今や時代の要請として避けて通れない．そのためプロトタイピングやスパイラルなど反復型開発の走りとも言うべきスタイルが試行されてきたが，それらは変更・拡張に強いオブジェクト指向技術の特徴を取り込み RUP に集大成された．一方，短期小規模開発にはもっと軽量なものが必要になり，RUP とは異なる価値観から XP を始めとするアジャイル開発が提唱された．WF 型->反復型，RUP->アジャイル開発という開発プロセスの潮流の今後の展望について，今回アレグザンダーのパターン言語の視点から考えて見たい．

2. アレグザンダーの３部作

ソフトウェア開発の世界に大きな影響を与えたアレグザンダーのパターン言語 3 部作について簡単に整理する．

①「時を超えた建設の道」鹿島出版会，1993
②「オレゴン大学の実験」鹿島出版会，1977
③「パタン・ランゲージ - 環境設計の手引き 」鹿島出版会，1984

これらの３部作は，アレグザンダーの考える理想の建築を実現するための価値・基本原則（戦略）・プラクティス（戦術）の３階層の戦略ピラミッド（図 1）を構成する．

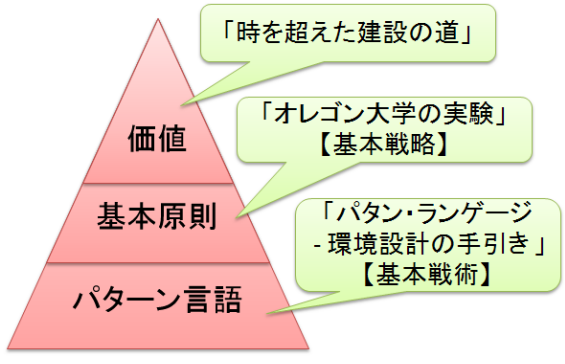

図 1　アレグザンダーの３部作

①は人に感動を与える街とはどのようなものか，そこにはどのような質が備わっているのか，それはどのようにしたらできるのだろうか，というアレグザンダーの基本思想を表した理論書の第 1 冊である．その質は名前のつけられない「無名の質 - Quality Without A Name(QWAN)」である．この「無名の質」はアレグザンダーの価値観の根底にあるもので，戦略ピラミッドの頂点に位置付けられる．

②はオレゴン大学でこの理論を実践した記録で，ここでは「無名の質」を実践するための６つの基本

原則（表 1）を挙げ，これは戦略ピラミッドの２層目の基本戦略に位置付けられる．

表 1　基本原則（基本戦略）

1.有機的秩序 organic order
2.参加 participation
3.漸進的成長 piecemeal growth
4.パターン patterns
5.診断 diagnosis
6.調整 coodination

表 2　無名の質の 7 つの特性

1.生き生きとした　alive
2.全一的　whole
3.居心地のよい comfortable
4.捕われのない free
5.正確な exact
6.無我の egoless
7.永遠の eternal

ソフトウェア開発プロセスとの類似点として興味深いのは，ソフトウェア開発を成功させるために大切なユーザー参加(2)と反復・成長型プロセス(3)が基本原則に挙げられている点である．この２つは RUP やアジャイル開発のベースにもなっている．

誰のために街づくりを行うかといえば，そこで生活する住人のためである．住人にとって町には無名の質，具体的には７つの特性（表 2）が備わっていなければならない．誰のためにシステム開発を行うのかといえば，当然顧客のためであり，ビジネスを実践する現場担当者や管理者，経営者のためである．無名の質は機能要求や非機能要求の上位あるいは裏にあって要求の対象外になっているが，その質についてもアレグザンダー流の考慮が必要ではないだろうか．その質を生み出す方法として，この６つの基本原則は一考の余地は十分にある．

③は基本原則に従って「無名の質」を作りだすためのプラクティスを，アレグザンダーの提唱するパターン言語という形式で具体的に 253 パターンとして列挙したものである．これは，基本原則を戦略とする具体的戦術を示すもので，戦略ピラミッドの３層目に位置付けられる．

3. RUP/XP をパターン言語として捉える

従来の RUP は 6 つのベスト・プラクティス（表 3）がベースになっている．それらは過去大規模開発に携わった人達の様々な経験や試行錯誤から得られた知見を集大成したものである．

表 3　RUP のベスト・プラクティス

1.反復型開発　Develop Iteratively
2.要求管理　Manage Requirements
3.コンポーネント・アーキテクチャ　Use Component Architecture
4.UML による可視化　Model Visually
5.継続的品質検証　Continuous Verify Quality
6.変更管理　Manage Change

これら 6 項目はそれぞれ開発上の問題とそれを解決する方策を示しており，パターンとして捉えることができる．これらのパターン言語化を試みた[1]（表 4）．

例えば「(1)反復型開発」はリスクを早期発見して解決する方策である．その中でも開発に付きもののユーザーからの要求の変更という問題の対応策でもある．WF 型ベースの従来型開発プロセスでいつも起きる大きな問題は仕様変更・設計変更である．なぜこのような問題がいつも起きるのか，その原因のひとつは要求の引き出しの困難さや要求の品質の問題にもある．これは「(2)要求管理」につながる．

表 4　開発プロセスのパターン言語の 1 例

(1)反復型開発

状況：

WF 型開発ではリスクを抱えたまま開発が進行し、開発後半になって実際に動き始めた時点で大きな問題が噴出する. 結合テストでモジュール間の不整合が問題となり設計見直しとなる. エンドユーザーの評価が始まって要求仕様の問題点が明確になり要求仕様の見直しとなる．これらはリリースの遅れ，予算超過などプロジェクトに重大な影響を及ぼす．これらの問題を根本的に解決する「反復型開発(1)」を成功させる鍵は「要求管理(2)」，「コンポーネント・アーキテクチャ(3)」，「UML による可視化(4)」，「継続的品質検証(5)」，「変更管理(6)」の実践である．

問題：

・重大な問題の発見が遅れる．

・変更の対応が難しい．

解決策：

・反復型開発により早期にリスクを軽減する．

・反復毎にユーザー参加で評価を行い，以降の反復で修正する．

RUP はこれら 6 つのベスト・プラクティスをベースに具体的なプロセスを構築したものである. 役割，作業，成果物について詳細に示されている．RUP は大掛かりなものであり実際のプロジェクトでそのまま 100%適用できるものではない．カスタマイズが必要になる．

従来の RUP は RUP7 で改訂され，この 6 つのベストプラクティスは見直しが行われ，6 つの基本原則とパターンの形に再構築された[2]．RUP7 では，従来のプラクティスはパターンという言葉で表現されているが，特にアレグザンダーの表記法は用いられていない．RUP はこれで基本原則・パターンの 2 階層になったが，価値は XP のようには明示されていない．

一方 XP は価値・基本原則・プラクティスの 3 階層で構成されている．このプラクティスはパターン言語として表現できる．つまり RUP も XP もパターンを基にしてプロセスが構築されている．

4. プロセス指向からパターン指向へ

開発プロセスは一つには決められない．プロジェクトのメンバーや顧客，構築するシステムの規模や特性などによって最適なものが必要である．RUP もそれぞれのプロジェクト毎にカスタマイズして利用するものである．

プロセスがどのようにしてできたのかその原点に立ち返れば，過去の経験から得られた知見がベースになっている．その知見であるプラクティスはパターンとして捉えることができ，それらパターンの知識ベース作成のためのひとつの技法がパターン言語である．プラクティスをパターンとして知識ベースに蓄積・共有し，そこからプロジェクトに使えるパターンを検索して引き出し，そこから最適なプロセスを再構築できないであろうか. つまり RUP などの既存のプロセスをカスタマイズするのではなく，パターンからプロセスを再構築するということである．例えて言えば，ソフトウェア・パッケージをカスタマイズして使用するのではなく，既存の実績のある部品を組み合わせてアプリケーションを構築するスタイルである．

では何を根拠にしてパターンを抽出し，プロセスを再構築すれば良いのだろうか．実績のあるパターンといってもプロジェクトの目的にそぐわないものもあるかも知れない．プロジェクトはまず第 1 に価値を明確にして利害関係者の間で共有しなければならない. ちなみにアジャイル開発は RUP とは明白に価値観の異なる開発プロセスを目指すものである（表 5）.

表 5　アジャイル開発宣言[3]

プロセスやツールよりも、個人と相互作用 包括的なドキュメントよりも、動作するソフトウェア 契約交渉よりも、ユーザとの協調 計画に従うよりも、変化に対応

次に基本戦略となる基本原則を決める必要がある．その次の具体的な戦術として共有知である知識ベースから適当なパターンを検索して再利用し，そこからプロジェクトに最適なプロセスをデザインし再構築する訳である．

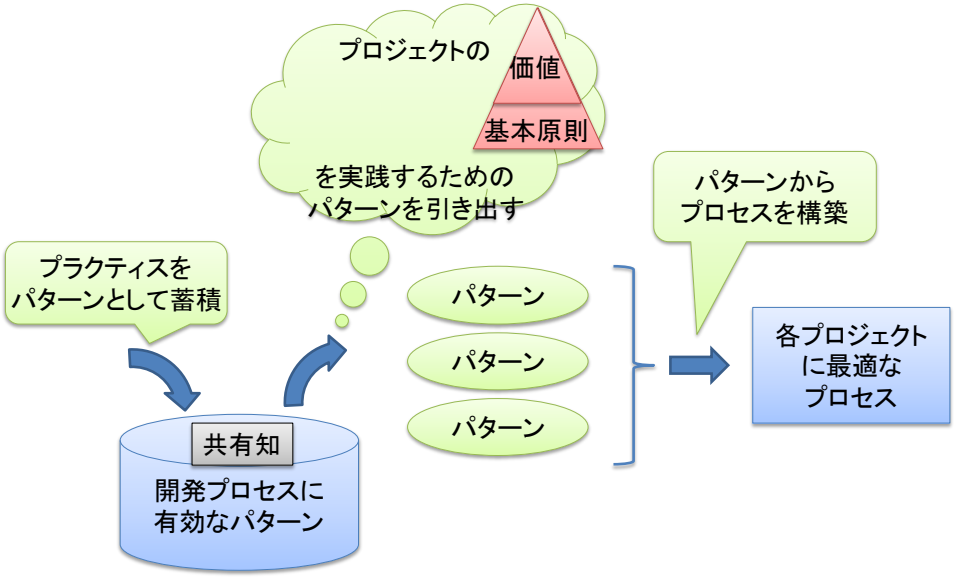

図 **2**　パターン指向の開発プロセス再構築

5. おわりに

『世界のビジネスを取り巻く環境においては「品質ムーブメント対創造ムーブメント」ともいうべきせめぎあいが本格化している』[4]というビジネス環境にあって，システム開発も従来のままではいられない．同書[4]では「無名の質」を生み出す知識デザインの技法として，パターン言語の可能性が指摘されている．

アレグザンダーの 3 部作に記された価値・基本原則・パターンの戦略ピラミッドは，住人が快適に生活できるための「無名の質」を街や建物に吹き込むための建築プロセスを方向付けるものである．従来のソフトウェア開発プロセスはいわば QCD を価値とする QCD 指向といえる．QCD 指向だけではこれからのビジネス環境に十分とはいえない．顧客に本当に必要な「無名の質」は機能要求にも非機能要求にも挙げられないし評価も難しい．この「無名の質」こそが製品を差別化しビジネスを差別化する鍵である．この質を製品に注入する技法としてパターン言語を見直したい．

参考文献

[1] 河合昭男, “RUP をパターン言語として考える”, http://www.atmarkit.co.jp/im/carc/serial/world19/world19.html, 2007.
[2] P.Kroll, W.Royce, “Key principles for business - driven development”, http://www-128.ibm.com/developerworks/rational/library/oct05/kroll/, 2005.
[3] Agile Alliance, “Manifesto for Agile Software Development”, http://agilemanifesto.org/, 2001.
[4] 紺野登, 知識デザイン企業, 日本経済新聞社, 2008.