ソフトウェア工学 110－13－2
（1996．7．26）

ソフトウェアシステムの新工法に向けて
－ソフトウェアプロセスはどう変わるか－

New Approaches for Constructing Software Systems
- From Software Process View Points -

井上 克郎[1]
Katsuro Inoue

大阪大学大学院基礎工学研究科情報数理系

1 はじめに

ソフトウェアシステムの開発手法が大きく変わりつつある。今までの伝統的な方法、すなわち、プログラムの命令を一つずつ積み上げてシステムのを作成するのに対して、現在は以下のような例の開発が多いと聞く。

- 過去の類似のシステムのプログラムの一部を再利用する。使える部分を探し、一部手を入れる。
- 使えそうなプログラムを買ってきて、それに手を入れたり、マクロ定義やインターフェーススクリプトを書いたり、設定を目的のものに合わせる作業をする。
- 無料で公開されているプログラムを探し、ネットワーク経由で手に入れる。必要なものがあるかどうかを検索したり、ネットワークニュースで問い合わせたりする。

このような新たな開発形態に対し、今までのソフトウェア工学的の手法は、そのままでは適用しにくい。ソフトウェア工学の一分野であるソフトウェアプロセス（簡単にプロセスと呼ぶ）も、そのままでは、適用が困難であろう。以下、その理由やどう変わかについて私見を述べる。

2 旧工法と新工法

命令を積み上げて作成したシステム $S1$（旧工法と呼ぼう）と複数の部品を組み合わせて作成したシステム $S2$（新工法の代表として）について考える。表1は、それらを比較したものである。

[1]〒560 大阪府豊中市待兼山町1－3
Department of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University,
Toyonaka, Osaka 560, JAPAN
06-850-6570 (Ph.) 06-850-6574 (Fax)
inoue@ics.es.osaka-u.ac.jp

$S1$ の構成要素は、プログラミング言語の文の長大な系列であり、非常に注意深く組み立てられている。個々の文の意味は比較的容易に把握でき、また、文と文との関係についても理解しやすい。従って、目的とするシステムに関して深い知識がない場合でも、順次作業をしながら知識を身に付けて構築できる場合があろう。しかし、知識の欠如が設計やコーディングの失敗につながり、うまく完成できない場合もあろう。

これらの開発を効率的に行なうための支援技術（いわゆるソフトウェア工学と呼ばれている技術）は、対象として均一な命令や文章、記号がかなりの分量で、いわゆる統計的な処理が可能になるくらい大きいことを仮定している場合が多い。仕様書や設計書、マニュアルなどのドキュメントのページ数や文字／記号の数、プログラムの行数、各種変数の数、サブルーチンの数や呼出数など、メトリクスデータとして活用されている。一方、統計的ではなく内部の意味や構文の詳細に関する情報に基づいて行なう支援技術は、その有用さが限られている。例えばプログラムの形式的な意味論に基づく設計や検証などの技術は、小規模例題から大規模実例へ拡大することが容易ではない。

このように開発作業全体は、粒度の小さい要素を扱う、粒度の小さい作業で構成されている。すなわち、ドキュメントに文字を追加した、エディタでソースコードを変更した、など細かな作業内容の積み重ねである。ソフトウェアプロセスは、開発に関する作業を人間が把握や管理できるぐらいの粒度に分解することが必須である。小さな作業を基本単位とすることは、全ドキュメントやソースプログラムの命令を一つずつ管理するのと同じことで、これでは、役にたたない。従って、同種の作業を一まとまりにし、その内部は統計的な情報だけを利用して、そういうまとまり単位に全体を把握／管理するのが一般的であろう。

$S2$ については、素材は、すでに複雑な機能を持つ部品である。どのような部品を使うか、これを決定するためにかなりの知識を必要とする。現在流通しているしているものは何か、それは、どのような機

表 1: $S1$ と $S2$ との比較

性質	$S1$ (旧工法)	$S2$ (新工法)
構成要素	プログラムの命令語 (ソースコード)、自然語文 (各種ドキュメント)、四角や線 (設計図など)	部品、インターフェースプログラム、過去のプログラムなど
構成要素の粒度	小	大
開発支援技法	Metrics など統計的手法	それぞれの部品の性格に依存
開発に必要な知識	最低限作業ができるためには、基本的な事項 (日本語、設計図、言語仕様) を知っていればよい	部品の内容やインターフェースに関する比較的深い事項が要求される
開発自由度	大	小

能を持ちインターフェースはどうなっているか、どれを選べば目的にもっとも簡単に近付くことができるか、など、高度な知識の上での決断をする必要があろう。

$S2$ に関する支援技術は、$S1$ のように統計的な手法はあまり意味をなさない。通常、部品の内部に関しては、隠蔽されていたり、または、その詳細を (積極的に) 知ることなしに利用する場合が多い。利用の手がかりとなるのは、情報が抽象化して記載されているマニュアルや同類の使用をしている例プログラムなどであろう。種々の部品に対して一般的に通じる支援技法というのは考えにくく、それより個々の場合に応じた技術やそのツールが有効であろう。

$S2$ の開発のなかで最も重要な作業は、設計である。どのような部品を選びそれを組み合わせるか、で、ほどんどシステムの性質が決定する。また、開発に必要な作業の性質もここで決定されるであろう。従って、作業は、かなりその部品の構成に依存したものになる。たとえば、特定の部品が使えるようにするためのセットアップ作業、その部品に関するドキュメント作業など、設計で指定する部品に対応したプロダクト作成作業が基本となる。

このような場合では、統計的な値、例えば何行プログラムを書いた、何時間テストを走らせた、などは、作業の進捗を表すのにあまり役にたたない。それより、あるプロダクトが完成した、ある部品と別の部品が協調して動いた、などのマイルストンが重要である。

3 プロセス研究の方向について

このような $S2$ に対して、既存のプロセスの技術は有効ではない。既存の技術は、(プロセスプログラムを始めとして) 基本作業を粒度の低い構成要素と見て、ループで繰り返す、階層化を行なう、などと、個々の性質ではなく、構成方法について議論されていたように思う。マイルストンの考え方では、既存のプロセス技術は、対象の粒度が大きく、その長所を引き出せない。

では、$S2$ に対して有効なプロセス技術はあるのだろうか。開発作業を、上述のようなエンジニアリングの作業の他、教育やマネジメントまで広げて、種々の作業を取り込んで、部品に関わる作業の粒度を相対的に下げると、既存のプロセス技術の延長として扱うことはできよう。

一方、プロダクトの管理技術、ソフトウェアアーキテクチャ、そして設計の問題などと深く関連付けた作業の表現や管理技術としてプロセスを位置付けることも出来よう。それらの計画の結果、作業が定義され、それに基づいて進捗を知ることが出来る。しかし、この場合、あくまでプロダクトが主である。

かなり悲観的なことを書いたが、やるべき研究テーマはたくさんあろう。

- オブジェクト指向技術を中心とした開発プロセスデータの蓄積。上述の $S2$ に最も近くかつ現実に利用されているのはオブジェクト指向関連技術であろう。数多くの実践データを得て、プロセス研究の方向性を確認することが必須である。
- 設計プロダクトに基づくプロセス管理の考え方の形式化と管理/支援手法の検討。開発作業の後のプロセスが、前のプロセスで作られたプロダクトに依存している。これは、一種の高階な作業と考え、それをどううまく扱うかが問題になろう。
- プロダクト主体のプロセスのモデル化、実働化の問題の検討。これには、マイルストンの考え方のモデル化も含まれよう。さらに、このようなプロセスモデルの有効性 (プロダクトモデルだけを利用するのに比べて) の検討も必須である。
- 設計作業のプロセスのモデル化。$S2$ では設計作業が重要である。$S1$ とは異なって部品の検索、知識の習得などの仕事が設計作業として加わっている。それらを含め、設計作業をある程度プロセスとして定式化できるであろう。