

ソフトウェア・ライフサイクルをカバーする 知識体系をITシステム構築の基盤にする

ソフトウェア・エンジニアリングは、絵に描いたもちを追い求める理論研究の学問ではありません。「ソフトウェア・エンジニアリング」という言葉がこの世に登場してから40年。ソフトウェア・エンジニアリングは常に実践的な内容を追い求めて進化してきました。ソフトウェアの開発が複雑化・大規模化の道をたどっている今日、ソフトウェア・エンジニアリングのさまざまなプラクティスを再確認し、品質・コスト・納期を適切にコントロールしていくことは、ソフトウェア開発を行う一般企業にとって至上命題ともいえます。現在は、過去の経験を踏まえ、より高いレベルでのソフトウェア・エンジニアリング手法の展開が可能な土壌が整いつつあります。パワフルな環境、オープン標準の推進、モデル作成の重要性の認識、開発手法の多様性など、ソフトウェア・エンジニアリング手法の適用に関して多くの好条件がこれほどそろった時期はなかったでしょう。本稿は、ソフトウェア・エンジニアリングの定義・歴史をひも解き、ほかの記事を読解するための序説を提供するものです。

Article 3

Body of Knowledge Covering Software Life Cycle as the Basis of Building IT Systems

Software engineering is not a discipline which pursues abstract theories. Since the advent of the term "software engineering" 40 years ago, software engineering has been evolving while constantly seeking practical solutions. With software development becoming increasingly complicated and large-scale, it is of utmost importance for software developing firms to re-examine the various practices of software engineering and properly control quality, cost and delivery. Fortunately, past experiences have provided the foundation to enable the expansion of software engineering to a significantly higher level. Never has there been a time more appropriate for the application of software engineering methods than today, blessed with such favorable conditions as a powerful environment, the promotion of an open standard, the recognition of the importance of modeling and diversity of development methods. This article aims to provide an introduction to help readers understand the other articles by unraveling the definition and history of software engineering.

① ソフトウェア・エンジニアリングの定義・歴史

ソフトウェア・エンジニアリングという言葉がいつ生まれたのかは不明ですが、その定義は、1968年にスキー場で有名なドイツのガルミッシュにて開催されたNATOのソフトウェア・エンジニアリング会議において、会議のチェアマンであったフリッツ・パウアー博士が提唱したものが最初といわれています[1]。それによると、以下のような定義がなされています。

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machine.

(ソフトウェア・エンジニアリングとは)実機上で効率良く動作し、信頼できるソフトウェアを、合理的に取得するための健全なエンジニアリング原則を確立し、用いることである。(筆者訳)

もちろん、パウアーの後もさまざまな定義が試みられていますが、それらを紹介するまでもなく、ここにはソフトウェア・エンジニアリングにおける二つの重要な側面が記述されていることに気が付きます。一つは、対象が実際のソフトウェアであるという点。これは数式や理論展開のみを目的とすることなく、動く「もの」、すなわち実機上のソフトウェアを指すという点が重要です。もう一つは、原則を「用いる」ことに言及している点です。つまり、原則を確立するだけでなく、それらを実際に活用していくということに焦点が当てられているのです。

さて、この定義を念頭に入れてIEEE(Institute of Electrical and Electronics Engineers: 米国電気電子技術者学会)での定義を紹介しましょう[2]。ここではさらに汎用的な記述になっています。ひとまずこの定義が現時点でのソフトウェア・エンジニアリングの根

底を成すコンセンサスであると考えて差し支えないと思われま

Software Engineering:

(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in (1).

ソフトウェア・エンジニアリング:

(1) ソフトウェアの開発・運用・保守に関する体系的かつ規律のある定量化アプローチの適用、すなわちソフトウェアに対するエンジニアリングの適用。

(2) (1)に示したアプローチを行う方法の研究。(筆者訳)

ソフトウェア・エンジニアリングとコンピューター・サイエンスがどのように異なるか疑問に思われる方もいることと思います。ここで前述の定義などから、両者の決定的な違いを挙げるとすれば、コンピューター・サイエンスがコンピューターのアーキテクチャーや、プログラミング言語、ソフトウェアの構造など、ハードウェア/ソフトウェアを問わずコンピューターを作る技術に焦点を合わせているのに対し、ソフトウェア・エンジニアリングは(もちろんソフトウェアだけに特化しているという違いはありますが)、コンピューターを利用していく方法に焦点が当てられている点です。ソフトウェア・エンジニアリングは、あくまでも実際の「適用」が大きなテーマであるといえます。

そもそもソフトウェア・エンジニアリング(という言葉ができる前ということになるかもしれませんが)は、プログラミングに関する手法から端を発したものでした。1960年代後半には、IBMがハードウェアとソフトウェアの価格を分離する「アンバンドリング政策」を始め、これによりソフトウェアの開発生産性・品質などにコンピューター業界や研究者の問題意識が向けられ、各種の取り組みが脚光を浴びました。1970年代前半から中ごろにかけては、ソフトウェアのモジュラー設計、複合設計、構造化設計、構造化プログラミング、データ抽象化など、今日のソフトウェア開発の基礎となる手法がさまざまに開発・提唱されました。1975年には米

国ワシントンDCにて第1回のICSE(International Conference on Software Engineering:ソフトウェア・エンジニアリング国際会議)が開催されています。1980年代に入ると、コード・ジェネレーターやライブラリー管理ツールなど、ソフトウェア・エンジニアリングの手法を組み込んだ開発支援環境を提供するベンダーが多数現れるようになり、1980年代後半には、IBMもこれらを体系的に組み合わせた開発・保守フルライフサイクルの支援環境体系であるAD/Cycle®を発表しました。このムーブメントは、ソフトウェア・エンジニアリングに対するソフトウェア業界の取り組みを一気に加速させました。ところが、当時の開発支援環境(当時はCASE: Computer Aided Software Engineeringと呼ばれていました)は、稼働するプラットフォームの非力さや、その価格の高さなどによって普及しきれずに姿を消していったものも多かったことは今さら説明の必要もありません。しかしながら、その後の1990年代にはクライアント/サーバー・コンピューティングやオブジェクト指向の潮流を経て、GUI (Graphical User Interface)の普及に伴うプロトタイピングや自動テストなどの分野が発展しました。1990年代後半にはWeb、Java™といった技術が一般的になるとともに、十分な性能のハードウェアが安価に入手できるようになり、開発者1人当たりが使用するコンピューティング・パワーは10年前とは比べ物にならないほどにまで高まりました。2000年に入ってから、モデリング標準としてのUML®(Unified Modeling Language: 統一モデリング言語)も一般に広く受け入れられ始め、昨今話題のMDA(Model Driven Architecture: モデル駆動型アーキテクチャー)などもこうしたソフトウェア・エンジニアリングの系譜に連なるものといえます。

② ソフトウェア・エンジニアリングはなぜ必要か

それでは、ソフトウェア・エンジニアリングはなぜ必要なのでしょう。ソフトウェアに限らず良いシステム(ビジネス・アプリケーションのためのITシステムから、電化製品などへの組み込みシステムまでを含む)を作り上げるときに必須な要素がQ

(Quality : 品質) (Cost : コスト) (Delivery : 納期)です。プロジェクトを予定したQCDで完遂することは、プロジェクトマネジメントの基本となる重要項目ですが、これはソフトウェア・エンジニアリングの観点からも極めて重要な項目です。ソフトウェア・エンジニアリングとは、名前が示す通りソフトウェアの開発・保守をエンジニアリングの手法を用いて行うことであり、その究極の目的は、私見ながら以下の3点に集約されます。

- ・ ユーザー要求を完全にソフトウェアに反映させること。
- ・ バグがまったく混在していないソフトウェアをつくること。
- ・ ソフトウェア開発 / 保守を完全に自動化すること。

これらが非常に難しく、今もってなおこの目的を完璧に達成する手段は現れていないということは広く知られている通りです。この見方は前述QCDのうち、Qすなわち品質の部分に大きく寄与するものです。しかし、ここにコストや納期の考え方を取り込むことによってソフトウェア・エンジニアリングが実践的かつ実用的なものとなっていきます。これらの目標に近づくために、私たちは個々の分析・設計・開発技術の開発や、テスト / 検証技術の発展、見積もり精度の向上やプロセス改善の浸透などを推進しているのです。

ソフトウェアの開発を例に取ってみましょう。ソフトウェアを開発する場合には、さまざまな開発方法があります。いきなりプログラミングを始める者もいれば、何らかのモデル作成を開始する者もいることでしょう。ドキュメントを作成する場合もあれば、ドキュメントをプログラムから自動生成させるような場合もあります。しかし、いずれの場合においても重要なことは、その場における最適な方法を用いて開発していくような指針が必要になることです。「ソフトウェアの開発なんていつもプログラムさえ書いたら終わりさ」と考えるエンジニアは、エンジニアリングが何であるかを理解していません。エンジニアリングとは、一つの分野におけるさまざまな試みの中からより優れた結果を導く方法を抽出し、体系化して学習し、さらなる活動にフィードバックしてうまく活用することをいいます。ソフトウェア・エンジニアリングはまさにこれをソフトウェ

アの世界で実現するための分野なのです。少しでも完璧な目的に近づけるように、QCDの制約の中で、ベストと思われる方法は何かを先人の知恵に学び、効率よく利用しようという試みであるともいえます。

エンジニアリングは、暗黙知から形式知を導き出してこれを活用するナレッジ・マネジメントのコンセプトと同様のアプローチを取ります。ソフトウェアの開発・保守に携わるエンジニアが、先人の通った道を再び迷いながら進むのではなく、既に掲げられた道標を基に効率良く進んで行くために、蓄積された膨大な知識を使わない手はないのです。

[参考文献]

- [1] Naur, P., and B. Randall, Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee, NATO, 1969
- [2] IEEE Standards Collection: Software Engineering, IEEE Standard 610.12-1990, IEEE, 1993



日本アイ・ピー・エム株式会社 技術・ソフトウェア・エンジニアリング
エグゼクティブITアーキテクト

榊原 彰 Akira Sakakibara

[プロフィール]

1986年に日本アイ・ピー・エムに入社。以来、IT技術者として多数のプロジェクトに参画。現在は、開発方法論や開発支援ツールおよびアーキテクチャー設計技術などの開発および社内外への展開を手掛けている。情報処理学会、プロジェクトマネジメント学会、IEEE、ACMの各正会員。日科技連SPC研究委員。WWISAメンバー。ITスキル標準 ITアーキテクト委員会主査。