第 1 章　ソフトウェア工学とは

「ソフトウェア工学」という言葉

「ソフトウェア」という言葉は、一般の「金物」を意味する「ハードウェア」という言葉から、イギリスの有名な統計学者であるジョン・ワイルダ・タケイ（John Wilder Tukey）氏が 1952 年（昭和 27 年）に作ったものである [1]。この原稿を読む人には必要がないことかもしれないが、広辞苑（第六版）では、ソフトウェアを「コンピュータのプログラムを抽象的にとらえる呼称」と定義している。私の理解では、ソフトウェアとは「プログラムに加えて、それに関連したドキュメントなどを包含したものの総称」である。

そして今我々が関心を持っている「ソフトウェア工学（ソフトウェア・エンジニアリング、Software Engineering）」という言葉は、1968 年（昭和 43 年）10 月に、当時の西ドイツのガルミッシュで開かれた NATO（北大西洋条約機構）主催の「ソフトウェア・エンジニアリングに関する国際会議」で、初めて公式に使われた。実際はその前に開かれたその準備会議での、当時のミュンヘン工科大学のフリードリッヒ・バウアー（Friedrich Bauer）教授の発言が契機となった。

バウアー教授が「ソフトウェア工学」という言葉を使った意図は、「ソフトウェアで多くのトラブルが起きている原因は、ソフトウェアが他の工学分野の製品のようにしっかりとした作られ方をしていない」と彼が考えたことにあった[BAU93]。つまり当時のソフトウェア開発が職人的な方法で行われ、組織的でも論理的でもなく、まして工学的なアプローチもとられていなかった、ということを示唆している。この事実を今、我々は重く受け止めておきたい。残念ながら今でも、一部を除いてこの状態は、大きくは変わっていない。

ソフトウェア工学の目的

この分野の言葉について世界的に最も権威があるとされているものは、IEEE[2]の定義である。その IEEE は「ソフトウェア工学」を「ソフトウェアの開発、運用、保守の系統的な方法」と定義している。

日本では何に権威を置くかについて、人それぞれの考えがあるだろう。私はこの分野では、岩波情報科学辞典を推したい [3]。それによると「ソフトウェア工学」とは、「ソフトウェアの作成と利用に関連した概念を科学的に抽出し、正しいソフトウェアを計画的に作成・利用するための理論と実践的技術」[NAG90]としている。個人的な感想だが、「科学的」、「計画的」、「理論」、「実践技術」という言葉は理解しやすい。また、「作成」だけでなく「利用」にまで範囲が広がっていることに留意したい。一方「正しいソフトウェア」という言葉はたいへんに重く、難しい。

---

[1] タケイ氏は、「ソフトウェア」に加えて「ビット（bit）」という言葉も作った。彼は新しい言葉を作る、素晴らしい才能を持っていたように思える。彼は 2000 年 7 月 28 日に、85 才で亡くなった。

[2] The Institute of Electrical and Electronics Engineers の頭文字を取ったもの。「アイ・トリプル・イー」と読む。「全米電気電子技術者協会」と訳されることがある。

[3] この辞典が出版されたのは 1990 年で、「犬の年」で世の中が進む ICT 分野では、大昔のことである。したがって、オープンシステム化やダウンサイジングなどに関わるものは一切対象になっていない。早急にこれらを含む新しい版が企画され、出版されることを、私は期待している。

「ソフトウェア工学」という言葉の生みの親であるバウアー教授は、これについて「現実のコンピュータの上で稼働する信頼性の高いソフトウェアを経済的に開発するための、完全な工学の原則の確立と活用」と述べている。

私個人の好みを述べれば、ソフトウェアプロセス改善の生みの親であるワッツ・ハンフリー（Watts S. Humphrey）氏の定義が好きである。ハンフリー氏はソフトウェア工学について、「高品質のソフトウェアを経済的に生産することを目指したエンジニアリング。科学および数学の原理、手法、及びツールの秩序だった応用」[HUM89] と記している。「高品質」、「経済的」の二つの言葉に、私自身強い共鳴を覚える。

私は、ソフトウェア工学とは「間違いばかりを犯している不完全な人間が、本来は完全であるべきソフトウェアをどのようにすれば開発し、維持することができるのかを追究するもの」であると考えている。そのためには、技術とそれを作業遂行の上で生かすプロセスから始まって、プロジェクトを含む組織の運営の問題やソフトウェア技術者個人の「人間としてのあり方」の問題まで、幅広い領域をカバーしなければこの問題を解決できない。以下で順次述べるが、まさに現代のソフトウェア工学はこの幅広い領域全体を漏れなくカバーしていることを、最初に指摘しておきたい。

「ソフトウェア危機」という言葉

「ソフトウェア危機（Software Crisis）」という言葉も「ソフトウェア工学」と同様、1968 年（昭和 43 年）のガルミッシュの会議で作られた。ソフトウェア工学が目的とするところは、端的にいえば「ソフトウェア危機を解消すること」である [4]。

しかしこの会議を主催し、あるいは出席して議論に参加した人たちは、1 つ大きな間違いを犯した。今も述べたとおり、この言葉が作られて 50 年近くになる。しかし「ソフトウェア危機」はいまだに「現役」である。つまり、「危機の状態」が 50 年近く継続していることになる。

私の友人の一人に、以前に新聞記者をしていた人がいる。25 年以上前その人にこの「ソフトウェア危機が四分の一世紀ほど続いている」と話をしたところ、彼は即座に「『危機』という言葉の使い方を間違っている」と指摘した。広辞苑（第六版）によれば、「危機」とは「大変なことになるかもしれないあやうい時や場合。危険な状態」とある。広辞苑の定義では、「たいへんな状態」がどれくらい継続すると、あるいはどれくらいで収束すると「危機」といえるのかについては何も述べていない。しかし我々は、「『危機』とは電気のパルスのように、インパクトはたいへん強いが、短時間で収束するもの」と理解している。私の友人の指摘も、この理解に基づいている。

「ソフトウェア危機」という言葉を作った人たちも、当然この理解を持っていたはずである。それでもその時点でのソフトウェアについての状態を「ソフトウェア危機」と呼んだ背景には、「今はたいへんな状態にある」という認識と同時に、「この状態は長続きしない、するはずがない、あるいはさせたくない、させてはならない」という認識があったものと思われる。後でも述べるが、この言葉を作った人たちはこの分野で、世界的に最優秀の人たちと私は評価している。「ソフトウェア危機」という言葉がいまだに現役であることは、「その最優秀の人たちでも、誤りを犯すことがある」ことの 1 つの証明であるといえる。

ソフトウェア工学は無力だったのか

[4] ソフトウェア危機については、第 2 章で改めて述べる。

ソフトウェア工学の目的がソフトウェア危機を解消することにあり、そのソフトウェア危機の状態がすでに 50 年近く継続していると、私は述べた。それでは、ソフトウェア工学は無力だったのだろうか。私は「無力だったわけではない」と考えている。

ソフトウェア危機が 50 年近く続いているといっても、同じ現象が 50 年もの間一貫して継続している訳ではない。「ソフトウェア危機」という言葉が作られた当時の「危機」の内容は、「他人が作ったプログラムを理解することが難しい」というようなものだった。それに対して最近では、

- 開発した製品の品質が悪い。
- 開発が、当初に立てたスケジュールから遅れる。
- 開発費用が、当初の予算を超過する。

さらには

- 開発プロジェクトが、最終の製品を作り出す前に解散させられる。

というようなものが、一般的な「ソフトウェア危機」の症状である。

時代とともに、多くの既存の問題には解決の方法が見いだされた。しかしその一方で常に新たな問題も発生して、「危機」の内容が変わり、ソフトウェア工学が対象とするメインテーマも変わった。

アメリカのメリーランド大学のヴィクター・バシリ（Victor R. Basili）教授は、ソフトウェア工学の発展経緯を

- 1960 年代は機能の時代
- 1970 年代はスケジュールの時代
- 1980 年代はコストの時代
- そして 1990 年代は品質の時代

と表現している[5]。ここには、メインのテーマの変遷がきわめて明確に述べられている。つまりソフトウェア工学はそれぞれの時点での大きな問題をメインのテーマに取り上げて、これまで着実に問題の解決を図ってきた。

それでは何故「ソフトウェア危機」がいまだに現役であるのか。私は、2 つの理由があると考える。1 つ目は「ソフトウェアの規模の拡大が、ソフトウェア工学の進歩を上回ったから」というもの、そして 2 つ目は「ソフトウェアの開発組織が、ソフトウェア工学の成果をこれまで充分に取り入れてこなかったから」というものである。

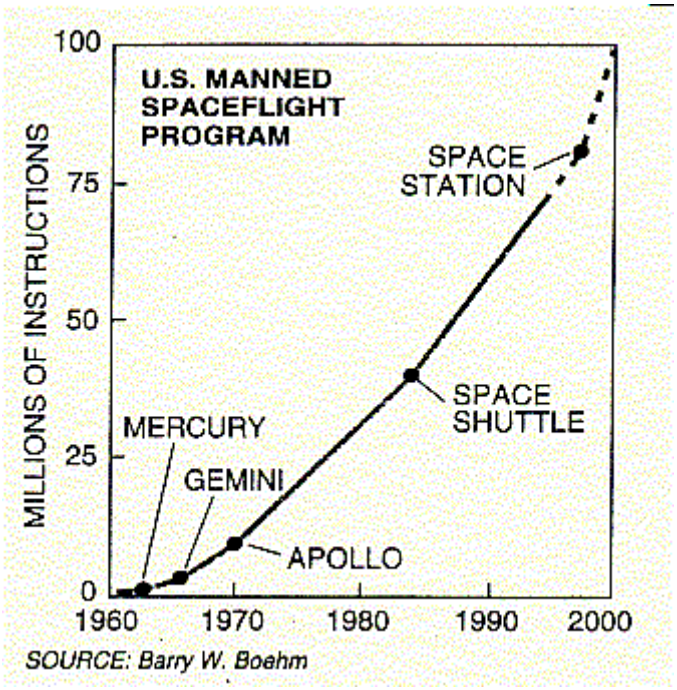
ネットワークを含むコンピュータのハードウェアの進歩があまりに急速で、人々の間でコンピュータへの期待が急速に高まった。しかし所詮コンピュータは、「ソフトがなければただの箱」でしかない。そのため急速にソフトウェアの大規模化・複雑化が進み、ソフトウェア工学の進歩がそれに充分に追いつけなかった、というのが 1 つ目の理由である。

英文のままで、しかも古い資料でたいへん恐縮だが、図表 1-1 に NASA の人間が搭乗する人

---

[5] バシリ教授はこの言葉を 1990 年代のごく最初に述べている[BAS91]。この時彼は、2000 年代が何の時代であるのかについて何も述べていない。彼が 2003 年 11 月に来日した時、私はこの質問を直接彼にする機会を持つことができた。彼は「再び“スケジュール”の時代。ただし意味するところは 1970 年代と大きく異なる」と答えた。1970 年代は、「スケジュール遅れをいかに回避するかがソフトウェア工学のテーマ」だった。しかし 2000 年代は、「ソフトウェア開発のスケジュールをいかに短縮するかがソフトウェア工学のテーマ」と彼はいいたいのだと、私は推察する。2010 年代が何の時代であるかという質問を彼にする機会を、私はまだ持っていない。是非その機会を、持ちたいものである。

工衛星関係でのソフトウェアの規模を表したグラフを示す。アポロ計画は宇宙飛行士を月に送り込んだことで名高いが、そのソフトウェアの規模はスペース・シャトル計画の 5 分の 1 程度でしかなかった。ここにこの分野での、ソフトウェアの規模の急速な拡大を見ることができる。



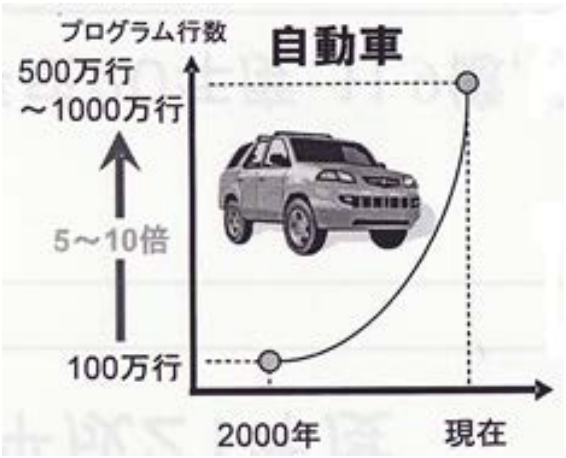
図表 1-1　人間が搭乗する人工衛星関係のソフトウェアの規模（[DOR97]より）

経済産業省のホームページで、この図表 1-1 と同じ内容の、もっと直近のものを見つけた。それを、図表 1-2 で示す。今回は自動車に組み込まれたソフトウェアの規模である。この図は 2011 年頃に 2009 年当時の情報を基に作成したと推測されるが、図表 1-1 に示したことと同じ内容を示している。ここでは人工衛星と自動車のものを示したが、他の分野でも、基本的に同じことがいえる。

アメリカのレーガン大統領時代の「スターウォーズ」構想は、「開発するべきソフトウェアの規模が大きすぎて、その時点でのソフトウェア工学の技術では、開発は不可能」といわれていた。予算上の問題やソ連の崩壊などで構想そのものが取りやめになったため、この問題は顕在化せずに終わった。

2 つ目の「ソフトウェアの開発組織が、ソフトウェア工学の成果をこれまで充分に取り入れてこなかったから」というものは、何も日本に限った話ではない。アメリカでも多くの事例が報告されている[YOU92]。逆にソフトウェア工学の成果を素直に取り入れることで、インドなどは国全体でソフトウェア産業が大きく成長するという状況を作り出している[6]。

[6] 例えばインドでは、ソフトウェアの CMMI（能力成熟度モデル統合）ですでにレベル 5 の段階に至っている開発組織の数がたいへんに多い。これは、インドでは一般にソフトウェア工学のレベルが、非常に高いことを示している。CMMI については、第 40 章で改めて述べる。



図表 1-2　自動車の組み込みソフトの推移（[METI11]より）

ソフトウェア工学と一般の工学の違い

2008 年 1 月に発行された広辞苑（第六版）では、「ソフトウェア工学」は「コンピュータのソフトウェアの設計・開発・利用・保守・品質評価などを工学的見地から研究する学問領域」と記載されている。この 1 つ前の版である第五版では、ソフトウェア工学の記載は削除されてしまっていたから、ここで見事に復活を果たしたことになる。

さらにその前の第四版では、「ソフトウェア開発の経験から生まれた理論、技法、考え方を体系化したもの。経験科学の分野の一つ」と記載されている。「経験科学」をさらに広辞苑で引けば、「対象をありのままに観察・記述・分析し、対象の法則性・説明原理を導出しようとする学問。実証的諸科学を指す」と記載されている。それに対して一般の「工学」は、「基礎科学を工業生産に適用して生産力を向上させるための応用的科学技術の総称」とある。「基礎科学」と「経験科学」の二つの言葉に、広辞苑第四版の編者達は一般の工学とソフトウェア工学の違いを凝縮したように、私には思える。

一般の工学がベースとする基礎科学とは、数学や物理学、化学、生物学などの理系の学問体系を指す。この基礎科学をベースに持つ限り、一般の工学分野では「正しい／正しくない」、「良い／悪い」が明白である。例えば、すべての物体は万有引力の法則に従わなければならない。飛行機やロケットのように一見この法則に従わないように見える物体には、それなりの仕組みが内在されている。そのような仕組みを用意せずに、単純にこの物理法則に従わない物体を作ろうとすることはナンセンスであり、正しくない。あるいは、鉛にいくら処理や加工をしても金にはならないことは、今は元素の周期律表から明らかである。したがって、鉛から金を作ろうとした昔の王様のアプローチは、正しいとはいえない。

ソフトウェア工学にも基礎科学がある。それは「コンピュータ・サイエンス（計算機科学）」である。その観点からは、広辞苑（第四版）の記述は間違っていた。コンピュータ・サイエンスは応用数学の一分野であり、「科学」として何ら不足な点はない。しかしソフトウェア工学の基礎科学という観点では、コンピュータ・サイエンスは必ずしも充分ではない。広辞苑（第四版）がソフトウェア工学を「経験科学」といった気持ちを、私は理解できる。

別のいい方をすれば、ソフトウェア工学の法則に従わなくても、情報システムを開発することができる。例えば、ソフトウェア工学の法則のうちに「ブルックスの法則」というものがある。「スケジュールから遅れ始めたプロジェクトに新たに人を追加投入すると、スケジュール遅

れが一層ひどくなる」というものである[BRO75]。この法則は、万有引力の法則や元素の周期律表のような絶対的なものではない。優秀なプロジェクト管理者がいて、プロジェクトのごく初期の段階に、経験豊富でシステム化対象の業務に精通した人材をうまく追加投入できれば、その場合にはブルックスの法則は働かない、ということもあり得る。ブルックスの法則は、残念ながら万有引力の法則とは違う。

ソフトウェア工学の成果

ブルックスの法則に限らずソフトウェア工学の成果は全て、「問題を認識し、それをどう解決するかを考え、実践し、状況をよく観察・計測し、その成果を積み上げ、体系化したもの」である。またこの成果を積み上げて体系化してきた人たちは、ガルミッシュの会議に出席した人たちも含めて、この分野で世界中の最優秀の人たちと私は評価している。したがってこの積み上げと体系化を評価するなら、ソフトウェア工学の成果を積極的に開発組織に導入するべきである。それにも関わらず、ソフトウェア工学の成果を組織内で生かして使っている開発部門は、多いとはいえない。

ソフトウェアの世界以外の部分で、我々は先人の実績の積み上げの上に生活している。先人の功績を継承できなければ、我々は万有引力の法則すら気づかずに一生を送るだろう。林檎の実が枝から落ちるところを私はまだ見たことがないが、仮にそれを見てもニュートンのように万有引力の法則を発見することは、私には多分できない。この法則は現代物理学の基本中の基本であるため、この法則に気がつかない世界では、物理法則を活用して作られた文明の利器が一切存在しない社会ということになるだろう。我々は今文明の利器を活用して、生活を大いに楽しんでいる。先人の実績を、大いに活用しているわけである。日常生活の部分では先人の実績の積み上げを活用し、ソフトウェア開発の部分でそれを無視することは、大きな矛盾である。

すでにコンピュータは、ユビキタス・コンピュータの時代を迎えている[SAK02]。ユビキタス・コンピュータとは、コンピュータが我々の身の回りのどこにでも存在する状態をいう。前にも書いたように、「コンピュータはソフトがなければただの箱」であるから、「ユビキタス・コンピュータの時代」とは、「ユビキタス・ソフトウェアの時代」でもある。そのような状態になったとき、例えばソフトウェアの品質がこれまでのような状態では、たいへんに困ったことになる。

ミシガン大学のダニエル・タイクロー（Daniel Tiechrow）教授は、「これまでソフトウェア危機と呼ばれていたものは、実は「慢性的な苦痛」でしかなかった。これから本格的な「ソフトウェア危機」が始まる」と述べている[7]。ユビキタス・ソフトウェアを考えれば、この発言は極めて妥当であると、私は評価する。

保守を含む幅広い範囲のソフトウェアの開発部門を統括し、あるいはソフトウェア開発部門の活動に何らかの責任を持っている人たちは、ソフトウェア工学の成果を充分に理解し、評価し、開発現場にその成果を積極的に取り入れて、定着させることに努める必要がある。

これが、現時点での私の意見である。この一連の原稿がその一助になることを、期待している。

---

[7] 私はこの言葉を、ロジャー・プレスマン氏の本[PRE97]で読んだ。プレスマンの本には、この言葉の出典についての記載はなかった。それ以来私はこの大元の資料を探しているのだが、まだ発見することができない。

キィワード

ソフトウェア、ハードウェア、ソフトウェア工学、ソフトウェア・エンジニアリング、Software Engineering、IEEE、ソフトウェア危機、Software Crisis、コンピュータ・サイエンス、計算機科学、ユビキタス・コンピュータ、ブルックスの法則

略語

IEEE：The Institute of Electrical and Electronics Engineers

人名

ジョン・ワイルダ・タケイ（John Wilder Tukey）、フリードリッヒ・バウアー（Friedrich Bauer）、ワッツ・ハンフリー（Watts S. Humphrey）、ヴィクター・バシリ（Victor R. Basili）、ダニエル・タイクロー（Daniel Tiechrow）

参考文献とリンク先

[BAS91] Victor R. Basili and John Musa, “The Future Engineering of Software: A Management Perspective,” IEEE Computer Magazine, Vol. 24, No. 9, pp 90-96, September, 1991.

なおこの論文は、今次の URL からダウンロードすることができる（確認日：2017 年（平成 29 年 1 月 3 日）。

http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/82.43.pdf

[BAU93] Friedrich L. Bauer, “Foreword,” from “Software Engineering – A European Perspective,” R. H. Thayler and A. D. McGettrick, eds, IEEE, 1993.

[BRO75] F.P.ブルックス Jr. 著、山内正彌訳、「ソフトウェア開発の神話」、企画センター、昭和 52 年.

この本の内容は、以下の本にそっくり含まれている。

フレデリック・P・ブルックス，Jr. 著、滝沢徹、牧野祐子、富澤昇訳、「人月の神話：狼人間を撃つ銀の弾丸はない －原著発行 20 周年記念増訂版－」、アジソン・ウェスレイ・パブリッシャーズ・ジャパン、1996 年.

この本の原書は、以下のものである。

Frederic Phillips Brooks Jr., “The Mythical man-month：essays on software engineering,” Anniversary edition, Addison Wesley, 1995.

[DOR97] Edited by Merlin Dorfman, Richard H. Thaiyer, ”Software Engineering,” IEEE、1997.

[HUM89] Watts S. Humphrey 著、藤野喜一監訳、日本電気ソフトウェアプロセス研究会訳、「ソフトウェアプロセス成熟度の改善」、日科技連、1991 年.

この本の原書は、以下のものである。

Watts S. Humphrey, “Managing the Software Process,” Addison-Wesley, 1989.

[METI11] 「産学連携ソフトウェア工学実践事業（高信頼組込みソフトウェア開発）概要」、経済産業省 、平成 23 年 1 月 7 日.

この資料は、以下の URL からダウンロードできる（確認日：2017 年（平成 29 年 1 月 3 日）。

http://www.meti.go.jp/committee/summary/0001640/035_05_04.pdf

[NAG90] 長尾真他編集、「岩波情報科学辞典」、岩波書店、1990 年.

[PRE97]　Roger S. Pressman 著、飯塚悦功他監訳、「実践ソフトウェア工学 第 1 分冊」、日科技連、2000 年.
この本は、[PRE01]、および[PRE05]であげた本の第 4 版の日本語訳である。
[PRE01] Roger S. Pressman, “Software Engineering A Practitioner’s Approach 5th Edition,” The McGraw Hill Co., 2001.
[PRE05] ロジャー S. プレスマン著、西康晴他監訳、古沢聡子他訳、「実践ソフトウェアエンジニアリング」、日科技連、2005 年.
この本の原書は、以下のものである。
Roger S. Pressman, “Software Engineering A Practitioner’s Approach 6th Edition,” The McGraw Hill Co., 2005.
[SAK02] 坂村健著、「ユビキタス・コンピュータ革命」、角川書店、2002 年.
[YOU92] E.ヨードン著、松原友夫訳、「ソフトウェア管理の落とし穴:アメリカの事例に学ぶ」、トッパン、1993 年.
この本の原書は、以下のものである。
Edward Yourdon, “Decline and Fall of the American Programmer,” Yourdon Press, 1992.

（2003 年（平成 15 年）9 月 16 日　初稿作成）
（2006 年（平成 18 年）4 月 13 日　一部修正）
（2007 年（平成 19 年）6 月 7 日　一部修正）
（2008 年（平成 20 年）7 月 23 日　一部修正）
（2013 年（平成 25 年）11 月 5 日　一部修正）
（2014 年（平成 26 年）2 月 25 日　一部修正）
（2016 年（平成 28 年）1 月 12 日　一部修正）
（2017 年（平成 29 年）1 月 3 日　一部修正）