特集

ソフトウェア工学の新しい流れ

井上克郎 監修

ソフトウェアシステムの作成現場を取り巻く状況は急速に変わりつつある．

少し前まで技術者たちは，大型メインフレーム計算機，ワークステーション，パソコンなど，安定したハードウェアやＯＳの上で，安定したシステム要求に基づいて作業を積み重ね，じっくりといいものを練り上げようと努力してきた．このような「落ち着いた」開発は，これからも存続していき，決してなくなるものではなかろう．

しかし今日，急速に発展しつつあるネットワークビジネスや情報産業では，システムの基盤の変更やシステムの仕様の変更は日常茶飯事である．じっくりいいものを作ろうとしても，システムの基盤の方がどんどん変化していったり，システムへの要求が瞬く間に変わってしまう．この変化に間に合わなければビジネスチャンスを失い，開発の意義を失ってしまう．このような「慌ただしい」開発がこれからの技術者の仕事の大半を占めるのではなかろうか．

過去，ソフトウェア工学の分野では，数多くの技術の提案や適用が行われてきた．しかしその多くのものは，（暗黙ではあるが）落ち着いた開発を前提としている．今までの技術がそのまま慌ただしい開発にも適用できるのだろうか？　筆者だけではなく，多くの開発現場の人々や研究者は，このような疑問や不安を現在抱いているのではなかろうか．

環境の変化を背景として，ここ数年のソフトウェア工学を研究する人々の間では，新たなテーマへの取り組みが行われている．本特集では，近年ソフトウェア工学の研究集会や会議などで取り上げられて，今後重要になると思われる４つのテーマについて取り上げる．

まず，慌ただしい開発の代表例として，現在急速に拡大しつつあるコンポーネント指向開発について，その背景や最新動向に詳しい青山幹雄氏に解説していただく．次に，開発対象となるソフトウェアシステムをモデル化し，設計，評価するために必須な技術として注目を集めているソフトウェアアーキテクチャについて，岸知二氏に紹介していただく．

また，ソフトウェア工学に関する種々の技術を効率よく評価するメタ技法として，近年，関心が持たれているソフトウェア工学の実験的手法を筆者が概説する．最後に，ソフトウェアの保守，改変，新規バージョン作成など，ソフトウェアの変革を発展という視点で整理する試みが盛んに行われてきている．このソフトウェア発展について片山卓也氏に解説していただく．

特集 ソフトウェア工学の新しい流れ

③ ソフトウェア工学における実験と評価

井上 克郎

1 はじめに

品質の良いソフトウェアを，いかに効率よく作成するかに関する技術や技法をソフトウェア工学と呼んでいる．ソフトウェア工学の名前が初めて使われたのは，1968年のNATO科学委員会においてで，それ以来この分野で，いろいろな技法の提案や新たなシステム・ツールの作成が行われてきている．表1にソフトウェア工学に関連するキーワードを示す．

この表に示すように数多くのキーワードが現れている．各時代で，いろいろなキーワードがブームなり，競ってツールが作られたりシステムが開発されたりしてきた．しかし，提案された方法が本当に有効かどうか，その時点で判断することは難しい．

例えば，CASEやAI的アプローチは，一時期，数多くのツールや適用事例が論文や会議で報告され，多くの研究者がそれに取り組んだが，最近ではずっと下火になっているように見える（地道に研究を続けている人もいるが……）．

多くの場合，ソフトウェア工学での技法やツールの善し悪しを議論しようとすると，定性的で主観的な議論になりやすく，白黒の決着をつけにくい．結局，歴史的な評価を待つことになる．

〈表1〉ソフトウェア工学におけるキーワードの出現世代

1970年代	ウォータフォールモデル 段階的詳細法・トップダウン設計 構造化プログラミング・高級プログラミング言語 構造化設計・ジャクソン法 構成管理
1980年代	メトリクス技法 ラピッドプロトタイピング ADA ソフトウェア開発環境 形式的開発手法 エキスパートによる開発支援 オブジェクト指向言語，設計
1990年代	簡易言語，第4世代言語 CASE（Computer Aided Software Engineering） CSCW（Computer Supported Cooperative Work） ソフトウェアプロセス，プロセス改善 オブジェクト指向開発 パターン ドメインエンジニアリング ソフトウェアアーキテクチャ コンポーネント エボリューション
2000年代	？？？

2 ソフトウェア工学と評価

ソフトウェアは，人間が汗水流して1つひとつソースコードや設計図，仕様書などを作り上げた後に得られるものである．このような人間的な活動を含むソフトウェア工学は，「科学的な」評価にさらし得るものであろうか．

もし，何か物理的/化学的な仕掛けで，ソフトウェアが自動生成されるものならば，その仕掛けの効率や精度を議論するための，科学的な評価方法を考えることができる．しかし，人間の行う作業，特に非常に知的であると考えられているソフトウェア作成作業は，同じように評価できるであろうか．

新しく提案される方法やツールの良い性質（例えばプログラミングの効率が100倍になる，バグが無くなる，

〈表 2〉ソフトウェア工学技術の評価方法の分類

観測型 (Observational)：実行しているプロジェクトを観察する.	プロジェクトモニタ	対象を漠然と観察. 目標不明確な場合も. 簡単な方法.
	事例研究	対象をより深く解析. まだ, 変動要素の制御が不十分だが, 比較的簡便.
	アサーション	主張が成り立つことを簡単なプロジェクトで実証. 厳密な評価としては不十分.
	野外調査(Field Study)	いろいろなプロジェクトを見て回る. 条件を揃えるのが困難だが追証しやすい.
履歴型 (Historical)：過去の情報・データを探す.	文献調査	過去発表された論文を探す. 条件や視点の統一不可能. 簡単.
	事例調査	過去のプロジェクトデータを探す. 条件が不統一で得られるデータが限られている.
	経験	過去のプロジェクトの定性的なデータを調べる. 定量的な議論ができない. やりやすくて簡単に傾向が分かる.
	(静的解析)	作ったプロダクトの解析をする. 方法には適用できない.
制御型 (Controlled)：条件を整えてプロジェクトを実行する.	繰り返し	条件を揃えていくつもの実プロジェクトで繰り返す. 理想的. 高価.
	実験室	条件を揃えて実験室で繰り返す. スケーラビリティ問題. 条件を制御しやすく比較的安価.
	(動的解析)	プロダクトの効率を実行させて計測. 方法には適用できない.
	シミュレーション	仮想データで実行. 手軽に定量的なデータが得られる. 信頼性問題.

…) を示すためにどのような方法が取れるだろうか. きっちり基礎が定まった理論体系があって, その体系の中で性質が証明できればいいのだが, 納得できるような理論体系が現れそうにない.

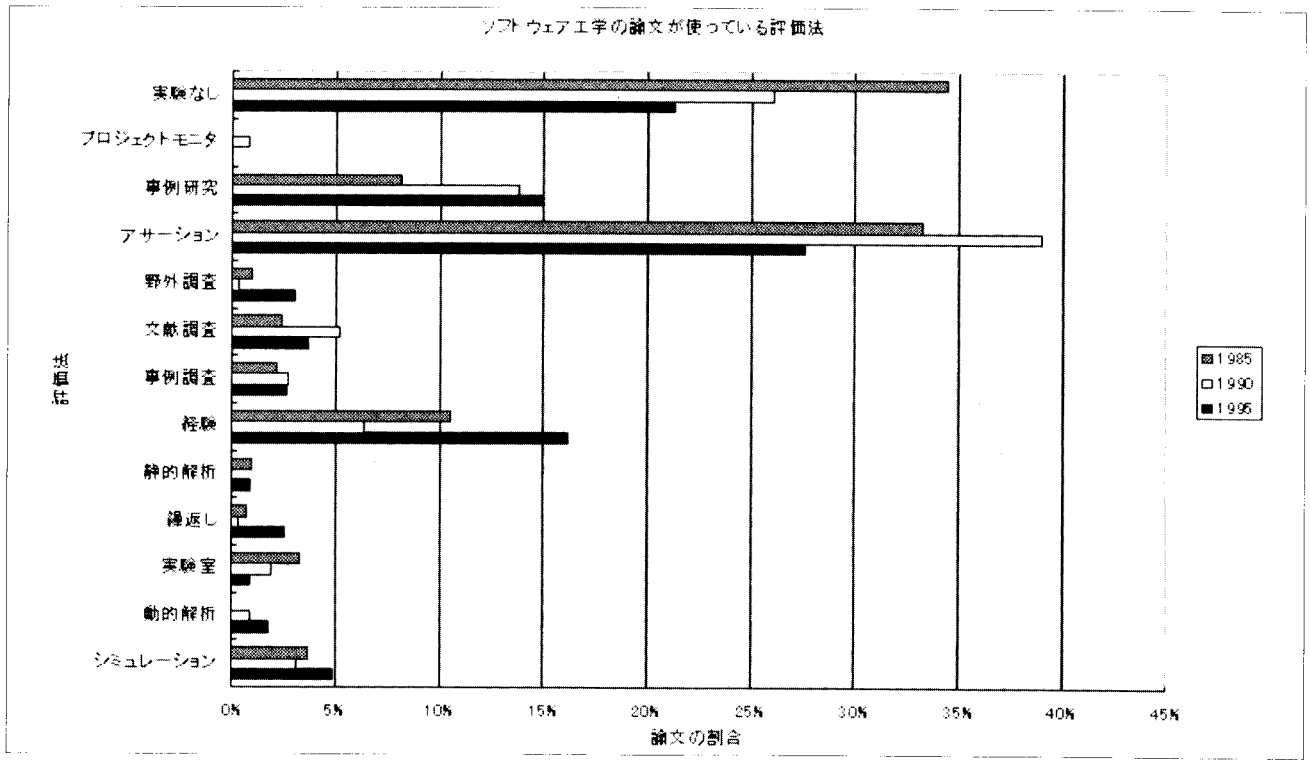
ならば, 採り得る方法としては, 実験的にその方法やツールの性質を示すことであろうか.

3 評価方法の分類

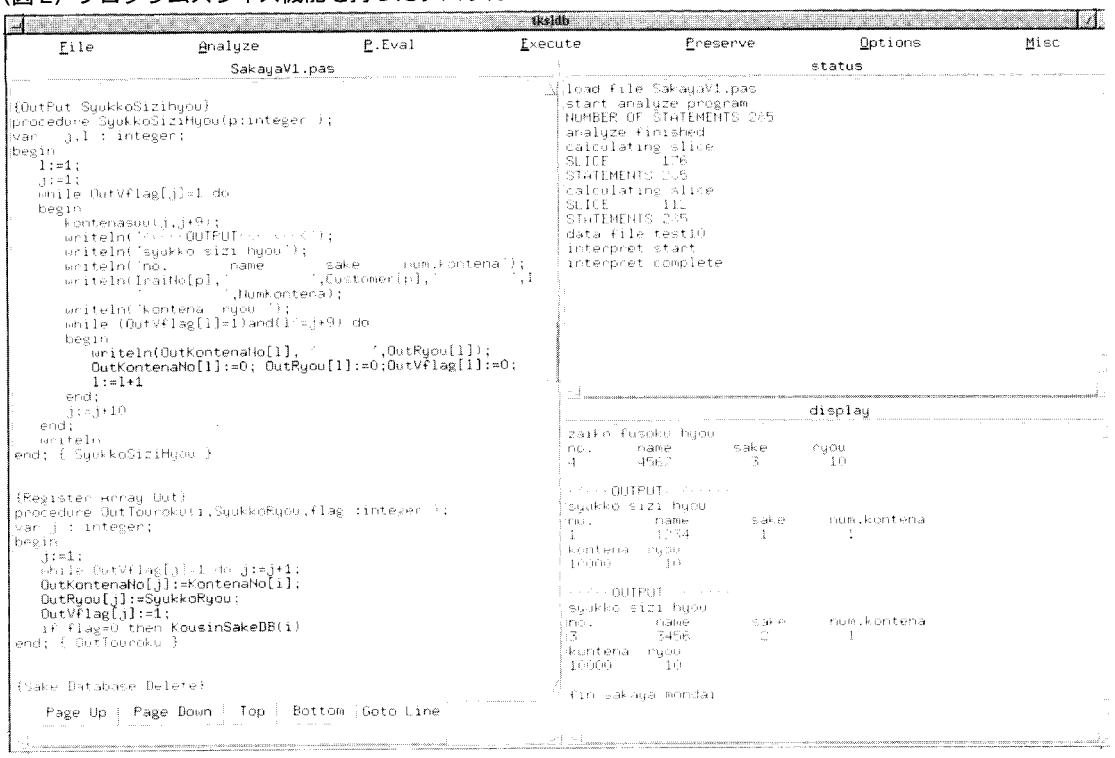
メリーランド大学の Zelkowitz と NIST の Wallace は, ソフトウェア工学の分野で扱われる種々の技術の評価方法を, 表 2 のように大きく 3 種類, 詳細に 12 種類に分類した[1].

観測型は実プロジェクトを調査する手法で, 比較的大規模な対象のデータが容易に得られるが, 正当な評価を行うための環境や条件の統一が困難である. 履歴型は過去のデータを調べるだけなので, より一層, 簡便な方法であるが, 目的に合致した情報が得られる

〈図 1〉ソフトウェア工学の論文が使っている評価法



〈図2〉プログラムスライス機能を持ったデバッガ



保証がない．制御型は目的とする評価に合った環境・条件を設定するため主張が行いやすいが，手間がかかる．括弧で表した静的解析，動的解析は，主にツールや環境の性能の解析に用いられるもので，通常，手法の評価には適用できない．

図1は，1985年，1990年，1995年の各年に，ソフトウェア工学国際会議(ICSE)論文集，IEEE Software Magazine，IEEE Transaction on Software Engineeringの3種類の雑誌に発表された全論文612編を分類したものである．

これによると，実験を行っていない論文は多数存在するが，その割合は近年，減少してきている．また，アサーションの論文が多いことが分かる．これは，環境や条件を厳密に整えることなく，提案する手法をとりあえずやってみて，「うまくいったよ」と主張している論文が多いのではないかと著者は推測している．

余談ではあるが，他分野の論文の同様な分類も試みている．それによると，例えば臨床医学分野ではほとんどの論文では何らかの実験を行っている，人類学では文献調査や事例調査が多い，など，分野による特徴がある．

4 実験的評価

実際，手法やツールをどう評価すればよいのだろうか．観測型，履歴型でも説得力あるデータが収集できればよいが，そううまくいくケースは少ないだろう．手間がかかるが制御された実験を行えば，主張したいことに沿ったデータが得られるはずである．

図2は，我々が作成したスライス機能を持ったデバッガの一部である．このツールは，通常のデバッガの機能のほか，着目する変数の値に影響を与え得る文（スライス）をハイライトで表示する機能がある（左側窓）．デバッグ作業中，値の怪しい変数のスライスを表示させると便利そうである．

実際に学生に，被験者になってもらって，バグを潜ませたプログラムの虫取りをしてもらった[7]．6人の学生を2グループに分け，スライス機能を持つデバッガを使うグループと，スライス機能を持たないデバッガを使うグループで，デバッグ時間の差を計測した．グループ構成員の能力差があることを考慮して2回実験を行った．2回目の実験では，グループの使う環境を入れ換えている．

この実験の結果，スライス機能を使うと，デバッグ時間の平均が14〜26%程度短くなることが分かった（被験者の数が少ないため，この差は有意な差ではなく，これを確認するために別の方法で追証実験を行っている）．

この実験を行うために，被験者をあらかじめ教育し，実験の指示を与え，実験に集中させ，その行動を監督者が記録し，その記録を分析する，といった作業が必要であった．特に行動の記

〈図3〉統合的実験環境

〈図4〉アイカメラを付けた奈良先端大の大和君

録，分析は，負担が大きい．

このような実験を効率的に行うためには，整った実験環境が必須である．図3は，筆者の研究室で作成した統合的実験環境である[3]．ワークステーションの画面出力，キーボード/マウス入力はすべて自動的に記録されている．被験者の視線の動きはアイカメラ(図4)で捕らえられ，画面の何を見ていたか記録される（このほか脳波や発汗も，必要ならば計測できる）．

このように，ソフトウェア工学における実験を促進し，得られたデータや実験方法を交換する場として，実験的ソフトウェア工学（Journal of Empirical Software Engineering）という論文誌が1996年より発刊されるようになった（図5）．

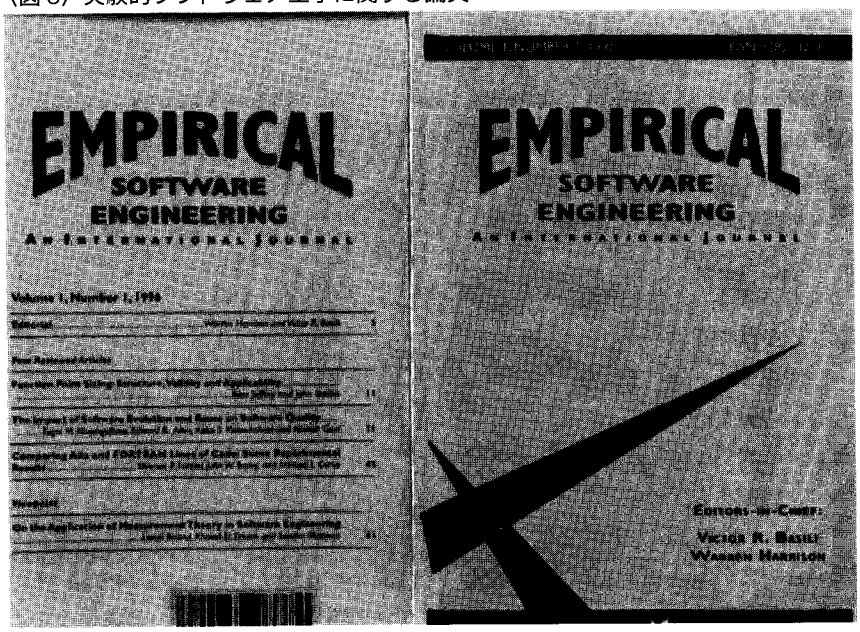
5 実践的アプローチ

ソフトウェア工学の性質上，シミュレーションや実験室レベルの評価で合格となっても，実プロジェクトでは，不合格になるかもしれない．高層ビルの建築技術を模型や平屋で評価するには難しい．スケーラビリティはソフトウェア工学の大きな課題である．

そこで，より実践に近い環境で評価実験が行える環境が欲しい．大学や研究所の中だけでは限界がある．一方，開発現場だけで繰り返し実験，評価を行うことは，予算や制度上，非常に困難である．そこで，大学や研究所の人々と開発現場の人々が，協調して大規模な実験や評価が自由に行えるような組織が望まれる．例えてみると，医学部に対する附属病院のようなもので，実践における諸問題をそこで解決するとともに新しい治療法を開発し，その評価を行う．

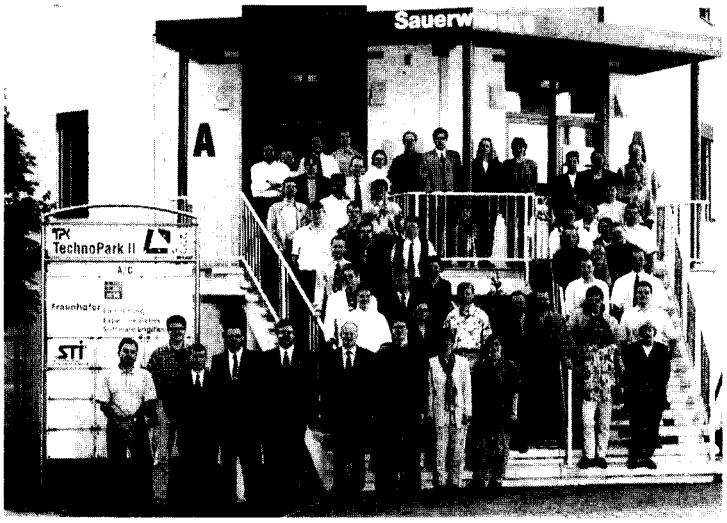
これに近いことが，ドイツで実際に行われている．Kaiserslautern 大学の Dieter Rombach 教授は，ソフトウェア工学，特に，メトリクスの専門家で非常にアクティブな研究者である（図6）．彼は，Fraunhofer 財団の支援を受けて，大学のそばに実験的ソフ

〈図5〉実験的ソフトウェア工学に関する論文



〈図6〉ロンバック教授

〈図7〉フランホッファ財団実験的ソフトウェア工学研究所とその人々



トウェア工学研究所（Fraunhofer Institute for Experimental Software Engineering：IESE）を1996年1月に作った（図7）．

この研究所では，現在，80名の常勤，60名の非常勤の研究員がおり，Kaiserslautern大学と多くの企業（ダイムラークライスラー，ボッシュなど多数）との間に位置して，研究と実践の橋渡しをしている．大学からは教授や学生などの人材，基礎研究の成果がIESEに渡り，IESEからは大学に，研究テーマや外的な評価結果がフィードバックされる．企業からは，問題や資金がIESEに提供され，問題の解決方法や研究成果，教育を受けた要員などがIESEから企業にもたらされる．

ソフトウェア工学に関する国際会議などを見てみると，IESEからの発表が非常に多く，活発な研究活動を行っていることがうかがえる．このような仕組みが今後のソフトウェア工学の研究に必須になるかもしれない．

6 おわりに

ソフトウェア工学における実験と評価についての最近の話題について述べた．ソフトウェアという対象物は，扱いがやっかいであるために，今までは，単に「新しいから」「面白そうだから」「好きだから」という個人の情緒的な判断で先に進む場合も多かったであろう．このような直観は大事にされるべきであるが，何百人月もかかるような大プロジェクトでは非常に危険である．ここで述べているような実験的な評価方法は，より安定した判断を，簡便に得るためのよりどころとなることが期待されている．

参考文献

[1] M. V. Zelkowitz, D. R. Wallace: Experimental Models for Validating Technology, *IEEE Computer*, pp. 23-31, May 1998.
[2] 西松，西江，楠本，井上：フォールト位置特定におけるプログラムスライスの実験的評価，電子情報通信学会誌，J 82-D-I，11，pp. 1336-1344，1999.
[3] K. Torii, K. Matsumoto, K. Nakakoji, Y. Takada, S. Takada, K. Shima: Ginger 2: An Environment for Computer-Aided Empirical Software Engineering, *IEEE Trans. Software Engineering*, 25, No. 4, pp. 474-492, 1999.

●井上克郎
（いのうえ・かつろう）
大阪大学大学院基礎工学研究科/
奈良先端科学技術大学院大学
情報科学研究科

15年ぶりにアマチュア無線の機器を買った．カーラジオのサイズで，HFからUHFまで全バンドに対応する最新のトランシーバーは，真空管育ちにとっては驚異である．オートチューナー内蔵のホイップアンテナも技術の進歩を感じさせるが，やはり電波の飛びは所詮ホイップである．屋根の上に八木を上げようか，寒空を見上げる今日この頃である．