社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
SS96-60 (1997-03)

ソフトウェア工学における最近の研究動向

松本 健一

奈良先端科学技術大学院大学 情報科学研究科

〒630-01 奈良県生駒市高山町 8916-5
matumoto@is.aist-nara.ac.jp

あらまし 本稿では，ソフトウェア工学における最近の研究動向の一端を示すため，まず，ソフトウェア工学の将来予測を試みた，90年代前半の3つの論文を紹介する．次に，最近注目されている研究形態の一つであるEmpirical Software Engineeringの概念や具体例について述べる．Empirical Software Engineeringとは，ソフトウェアの開発，利用，保守に関わる理論や技術の妥当性や有効性を，Case StudyやReplicated Experimentと呼ばれる“実験”を通じて検証しようとする，いわば“実験に基づくソフトウェア工学”である．更に，本稿では，Empirical Software Engineeringでは重要な要素である，実験用ツールの開発，配布の現状についても述べる．特に，OLEやJavaを用いたツールの実現例を報告する．

キーワード 実験的ソフトウェア工学，実験，観察，ソフトウェアシステム，OLE，Java

Recent Topics on Software Engineering

Ken-ichi Matsumoto

Graduate School of Information Science
Nara Institute of Science and Technology (NAIST)

8916-5 Takayama, Ikoma, Nara 630-01, JAPAN
matumoto@is.aist-nara.ac.jp

Abstract The paper surveys recent topics on software engineering based on three articles published in the 1990s concerning future directions in software engineering, and outlines “empirical software engineering” as one of the hottest research frameworks in software engineering. Empirical software engineering emphasizes the importance of experiments, including “case study” and “replicated experiments,” in evaluating and validating the theory and technology of software development, operation, and maintenance. This paper also presents new technologies (such as, OLE and programming language JAVA) to develop and distribute software systems and tools that can be used in the case studies and replicated experiments.

key words Empirical Software Engineering, Experiment, Observation, Software System, OLE, Java

1. はじめに

ソフトウェア工学という用語が使われるようになって約30年が経つ．その間にソフトウェア，及び，ソフトウェア工学に寄せられた期待の変遷を，BasiliとMusaは次のように表現している[2].

・1960年代:機能の時代
情報技術が社会に浸透し始めた時代．社会的ニーズを満足させるために，情報技術をいかに利用すればよいかに関心が寄せられた．また，社会の持つ機能(社会的機能)とソフトウェアの間に密接な繋がりが出来た．

・1970年代:スケジュールの時代
計画通りの期間内にソフトウェア開発が終了することが求められ始めた時代．ライフサイクルモデル (Life-cycle model) とスケジュール追跡技術 (Schedule Tracking) の研究が盛んに行われ，開発現場に導入された．

・1980年代:コストの時代
ハードウェアコストの減少とパーソナルコンピュータの普及によって，ソフトウェアの価格が下がり始めた時代．比較的低予算でもソフトウェア開発が可能となり，ソフトウェア開発における生産性が重要視されるようになった．コストモデル (Cost model) と資源追跡技術 (Resource Tracking) の研究が盛んに行われ，開発現場に導入された．

・1990年代:品質の時代
情報技術に対する社会の依存度が増大し，ソフトウェアの品質が重要視され始めた時代．ソフトウェア品質の定量化が試みられ，開発プロセスの中心的役割を果たすようになった．

品質の時代と呼ばれる今日はまた，ソフトウェアの開発／利用形態が多様化した時代でもある[2]．即ち，様々な人がソフトウェアを利用し，利用の目的や環境も多種多様なものとなってきている．そして，ソフトウェアの開発作業の多くでは，道具としてソフトウェアが利用されるため，利用形態の多様化はそのまま開発形態の多様化をもたらした．開発／利用形態の多様化を象徴する用語としては，オブジェクト指向開発，コンポーネントウェア，エンドユーザコンピューティング，World Wide Web (WWW)等がある．

ソフトウェアの開発／利用形態の多様化は，新たな理論や技術の開発を促進すると同時に，理論や技術の妥当性や有効性の検証の必要性をより高くする．即ち，理論や技術の多くは，その適用対象をある程度限定し，適用上の条件も持っている．従って，新たな理論や技術はもちろんのこと，既存の理論や技術であっても，ソフトウェアの開発／利用形態が変化すれば，従来通りに用いることが適当とは必ずしも言えなくなる．理論や技術の適用範囲を明確にし，ソフトウェアの開発／利用形態に応じてそれらを使い分ける必要がある．

ソフトウェアの新しい開発／利用形態に対して，(1)従来からの理論や技術をそのまま適用するのか，(2) 理論や技術に修正を加えるのか，あるいは，(3)新たな理論や技術を作り出し適用するのか，その判断が容易でない場合は多い．原因の一つは，適用対象や適用条件が明示的に示されていない理論や技術が意外に多いことにある．しかし，たとえ適用対象や適用条件が明示的に示されていたとしても，その理論や技術が提案された時点では想像も出来なかった新しい開発／利用形態に対しては，やはり判断は難しい．理論や技術の適用の是非は，抽象的，あるいは，定性的な情報に基づく議論だけでなく，具体的，かつ，定量的な情報に基づく議論によって決定する必要がある．

本稿では，まず，ソフトウェア工学の将来予測を試みた，90年代前半の3つの論文を紹介し，実験を中心にした分析的アプローチがこれからのソフトウェア工学に必要である，という共通の認識が存在することを示す．次に，Case StudyやReplicated Experimentと呼ばれる“実験”を基礎とする，いわば“実験に基づくソフトウェア工学”であるEmpirical Software Engineeringの概念や具体例について述べる．更に，Empirical Software Engineeringでは重要な要素である，実験用の開発，配布の現状についても述べる．特に，OLEやJavaを用いたツールの実現例を報告する．

2. ソフトウェア工学の将来予測

本章では，ソフトウェア工学の将来予測を試みた3つの論文を紹介する．いずれも90年代前半に発表された論文であり，その内容を現在でも予測と捉えることは必ずしも適当ではないが，ソフトウェア工学分野において注目されている理論や技術を知る上での参考になると考えられる．

(1) V. R. Basili and J. D. Musa: “The future engineering of software: A management perspective.”[2]

BasiliとMusaは，ソフトウェアの利用形態の多様化に伴って，ソフトウェア品質が重視されるようになる述べている．特に，定量的アプローチに移行することにより，ソフトウェア工学は真に工学となり，ソフトウェア開発に対するいかなる要求にも答えることが出来るようになると主張している．また，品質向上に寄与する要素技術として次のようなものを挙げている．

- 実用規模に適用可能な形式的方法論
- オブジェクト指向を中心とした設計方法論
- 抽象化能力を持つプログラミング言語
- ゴール，モデル，コンテキストを明確にした計測
- ユーザが使うであろう命令セットを小さくしたReduced Operation ソフトウェア
- 再利用
- 認知心理学
- ソフトウェア社会学

(2) W. F. Ticky, N. Harbermann, and L. Prechelt: “Summary of the Dagstuht Workshop on future directions in software engineering.”[10]

Tickyらによるこの論文は，1992年2月にDagstuhtで開催されたソフトウェア工学の将来についてのワークショップの報告である．ワークショップでは，Methodology, Industrial Practice, Modeling and Design, Formal Methods, Tools and Components, Education, 及び, Development Processについて議論され，ソフトウェア工学の発展に不可欠なトピックスは次の4つであると結論づけている．

- Software Architecture
  プログラム／システムの構成要素の構造，要素間の関係，設計と進化の規律やガイドライン．構造上の様式．
- Evolving Systems
  品質を犠牲にすることなく，将来において発生するであろう変更に要するコストを出来るだけ小さく押さえる仕組み．
- Scientific Basis
  ソフトウェアの部品や組立工具についての科学的研究．部品や組立工具の分析や評価を含む．
- Education
  設計技法，部品，既存のソフトウェアを利用した開発のノウハウなどを重視した若手技術者の教育．

(3) S. L. Pfleeger: "Software engineering needs to mature."[8]

Pfleegerは，ソフトウェア工学の成熟度を他の工学と同様のレベルまで上げるべきであると述べている．レベルアップには，次の3つのステップを経る必要があると主張している．

- Step 1: 問題理解のためのモデルの構築
  ソフトウェアがどのような業務において，どのようなコンテキスト(状況，条件，環境)で利用されるかを理解する必要がある．ソフトウェア工学は，実世界の問題を解決するためにあるのだから，もっと，実験に基づく議論が必要である．
- Step 2: 確固たる標準の確立
  ソフトウェア工学の標準は，現在，250を越えるが，その多くは証明された事実ではなく，習慣に基づいて定められており，必ずしも，客観的な標準とはなっていない．また，最終成果物を評価する標準は少なく，ソフトウェア工学の標準はプロセス偏重の傾向がある．
- Step 3: ソフトウェアエンジニア認証システムの確立
  簡単なプログラムしか書いたことの無い多くの人が，自分を有能な(資格を有する)ソフトウェアエンジニアと考えている．ソフトウェアエンジニアが持つべき知識やその技量を披露する方法を確立する必要がある．

以上3つの論文に共通する主張は，計測，評価，そして，実験という分析的アプローチが，これからのソフトウェア工学において必要，あるいは，重要であるということである[12]．特に，Tickyらは，分析や評価は科学的研究の基礎であるとしている[10]．また，Pfleegerは，ソフトウェア工学が実世界の問題の解決を目指している以上，実験に基づく議論がなければ工学として成熟しないと主張している[8]．

3. Empirical Software Engineering

ソフトウェア工学分野において実験を重視した研究形態をEmpirical Software Engineeringと呼ぶ[17]．Empirical Software Engineeringとは，実際のソフトウェア開発現場や研究機関において様々なコンテキストを設定し，ソフトウェアの開発，利用，保守に関わる理論や技術の適用実験(ソフトウェア工学実験)を行い，その妥当性や有効性などを検証しようとする，いわば"実験に基づくソフトウェア工学"である．

Empirical Software Engineeringにおける実験は，"Case Study"と"Replicated Experiment"の2つに分類される[9]．Case Studyとは，比較的小規模でコントロールが容易な環境下で行われる実験や観察のことである．その目的は，主に，仮説を立てることにある．仮説とは，例えば，「プログラム中の分岐命令数の数と保守コストの間には正の相関がある」といったものである．Case Studyは，ソフトウェアの開発／利用形態の実態，及び，その上で用いるべき理論や技術が何であるかを正しく認識する有効な方法の一つである．Case Studyを重視する姿勢は，大きく変化したソフトウェアの開発／利用形態に関する知識(ドメイン知識)を，ソフトウェア工学の研究者が必ずしも豊富に持っていないという認識の現れでもある．

一方，Replicated Experimentとは，仮説を検証するための実験を繰り返し行うことである．同一被験者を対象に繰り返し実験を行うというだけでなく，異なるコンテキストで実験を実施するという意味合いを持つ．実験を繰り返し行うことにより，あるいは，様々なコンテキストで行うことにより，実験結果の精度や信頼性は向上する．

Case StudyとReplicated Experimentは補完的な役割を持つ．即ち，Case Studyは，漠然とした問題意識や仮説を鮮明なものとするために，日常現象の中で問題や仮説にかかわりがありそうな場面について行われる．Case Studyで設定された仮説は，よりコントロールされた実験環境におけるReplicated Experimentによって検証される．そして，Replicated Experimentを実施する中で，研究者は漠然とした問題意識や仮説を持つことになる．

Case StudyやReplicated Experimentにおいて強調されるのは，様々なコンテキストにおいて実験を行うことの重要性である．コンテキストには次のようなものがある[6]．

- 物理的，社会的，文化的文脈
  開発部署の管理目標や方法は，会社の営業目標や方法と無関係ではありえない．
- 長期的文脈

年度の初めと終わりとでは，納期に対する考え方は異なるかもしれない．

- 短期的文脈
  「今日の目標」が何であるかによって，開発者の行動の優先順位は変わるかもしれない．

また，コンテキストを重視する主な理由は次のとおりである[3]．

- 形式的，あるいは，非形式的場面の中で，日々の活動や出来事から人がどのように知識を獲得しているのかを知ることが出来る．
- 作業の遂行を強制したり，支援したりする要因を決定することが出来る．
- 何が2つのコンテキストを機能的に等しくしているのかを理解することが出来る．

様々なコンテキストで実験を繰り返すことにより，実験結果の精度や信頼性が向上するだけでなく，個々の理論や技術の適用可能範囲が明確になると言える．

4. 実験用ツール開発，配布の現状

ソフトウェアやその開発／利用過程を対象とした実験の実施コストは決して小さくない．被験者には，ソフトウェアに関する知識やソフトウェアの開発／利用経験が要求される．また，実用規模のソフトウェアを対象とした場合，実験や観察の期間は，数週間から数ヶ月，場合によっては数年に及ぶことがある．

実験コストを小さくし，より多様なコンテキストで Case Study や Replicated Experiment を行う一つの方法は，実験データの収集，分析に関わる作業（の一部）をソフトウェアツール（実験用ツール）によって自動化，あるいは，支援することである，ツールの導入は，被験者や実験実施者の負荷を低減するだけでなく，人為的ミスによる実験データ（の一部）の欠落を防ぎ，実験データの記録形式や精度の統一性を高くする．

本章の以降では，実験用ツールの開発，配布に適していると思われる2つの技術 OLE と Java を具体例を中心に紹介する．

(1) OLE による開発

Case Study において，実験用ツールの開発コストと修正容易性は重要な要素である．**3.** で述べたように Case Study は，漠然とした問題意識や仮説を鮮明なものとするために行われる．実験結果の予測は難しく，ある程度の試行錯誤は避けられないためである．

Object Linking and Embedding (OLE)は，ソフトウェアの部品化技術の一つで，ソフトウェアアプリケーション間での情報のやり取りや共有を可能にする[13]．OLE を用いて開発されたアプリケーション（OLE アプリケーション）は，その機能の一部を他のアプリケーションから利用可能なサービスとして公開することが出来る．従って，既存のアプリケーションが公開するサービス（機能）をうまく利用することができれば，新規作成部分は小さくなり，実験用ツールの開発コストを小さくなる．また，開発期間も短くなるため，試行錯誤を伴う実験にも適している．

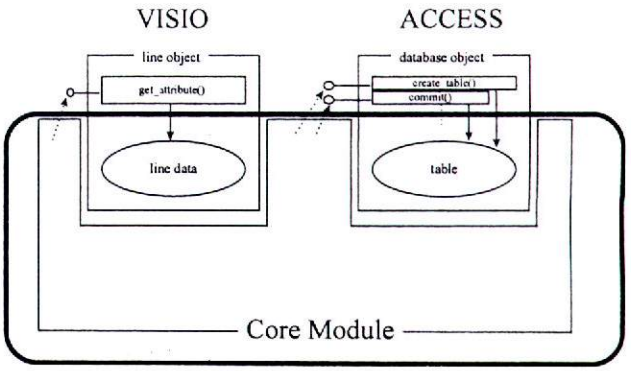

図1 プロセス生成システム

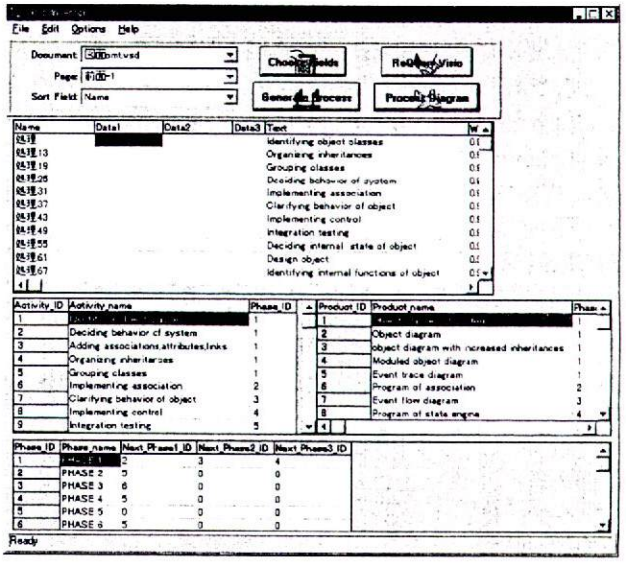

図2 プロセス生成システムのユーザ画面

花川らが開発した「オブジェクト指向開発プロセス生成システム（以下では「プロセス生成システム」と呼ぶ）」は，進捗管理作業を含むソフトウェアプロセス記述を自動生成するシステムである[5]．システムへの入力は，対象とする開発手法で定義された作業項目とプロダクトの関係である．作業項目とプロダクトの関係をデータフロー図(DFD)で表現できる開発手法であれば，オブジェクト指向開発手法に限らず，その手法に適したプロセス記述を得ることが出来る．プロセス生成システムの構成を図1に，ユーザ画面を図2に，それぞれ示す．

プロセス生成システムの主な機能は次の4つである．

- DFD 編集
- DFD とプロセス記述管理（データベース管理）
- プロセス記述生成
- プロセス記述描画

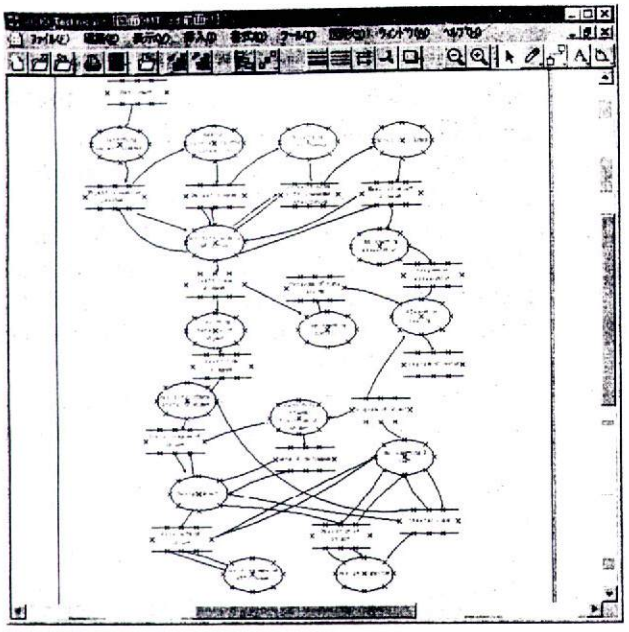

図3 DFD 編集画面

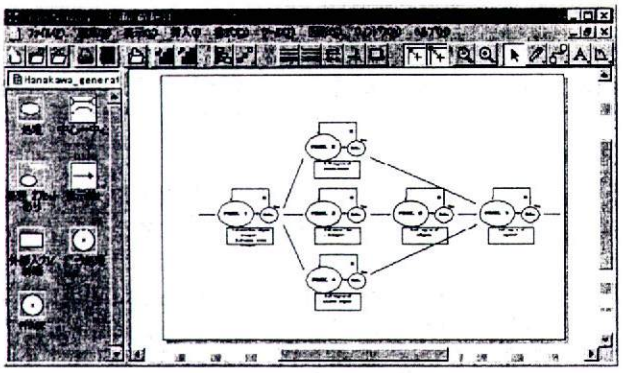

図4 プロセス記述の表示画面

これら4つの機能の実装において新規作成されたのは，花川らのオリジナルな研究成果である「プロセス記述生成」に関する部分のみである[5]．残りの「DFD 編集」と「プロセス記述描画」は，OLEアプリケーションである VISIO が提供するサービスによって実現されている[16]．また，「データベース管理」は，やはり OLE アプリケーションである Microsoft Access が提供するサービスによって実現されている[14]．DFDの編集画面を図3に，プロセス記述の表示画面を図4に，それぞれ示す．

OLE を用いて開発された実験用ツールの修正は容易である．即ち，VISIO や Microsoft Access 以外にも，ユーザインタフェース，データベース，あるいは，ネットワーク通信に関わるサービスを提供する OLE アプリケーションは既に開発されている．また，実験用ツール自体を OLE アプリケーションとして実現することも可能である．実験の目的や実施方法の変化に応じてそれらを使い分ければ，必要とする機能は比較的容易に実現される．

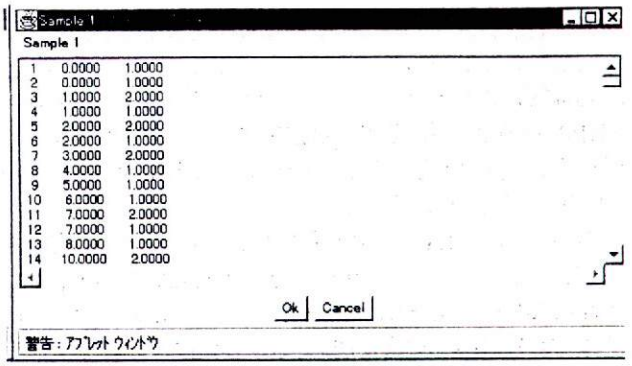

図5 データ入力画面

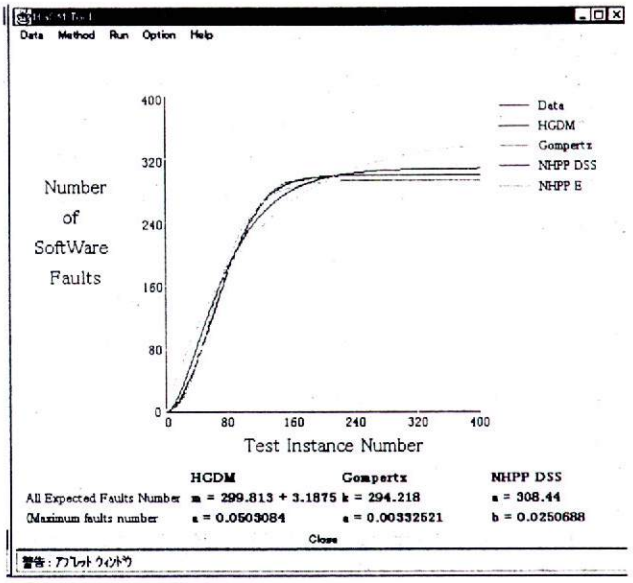

図6 推定結果の例

(2) Java による開発

Replicated Experiment において，実験用ツールの実行環境に関する制約と配布コストは重要な要素である．**3.** で述べたように Replicated Experiment は，様々なコンテキストで実施される．実験用に整備された環境だけでなく，実際にソフトウェアが開発，利用されている環境での実験も必要なためである．

Java は，C や C++に似た構文規則を持つ，オブジェクト指向プログラミング言語である[1][4]．大きな特徴の一つは，HotJava, Netscape, Internet Explorer といった主要な Web browser 上で，ハードウェアや OS を選ばず実行できる点である．また，ネットワークを介してのプログラム配布が容易で，プログラムのインストール，アンインストール，及び，更新に要する利用者の労力も小さい．従って，Java を用いて実験用ツールを開発すれば，実行環境に関する制約は非常に小さくなる．ツール配布に要するコストも，従来に比べて非常に小さくなる．

宮川らが開発した「ソフトウエア信頼性成長モデルツール（以下では「SRGM Tool」と呼ぶ）」は，ソフトウェア信頼性成長モデル(Software Reliability Growth Model)を用いて，ソフトウェア中の残存バグ数や平均故障待ち時間を推定するツールである[7][19]．ツールへ

の入力は，テストインスタンス，累積フォルト数，テスターの人数または能力の系列である．利用できるモデルは，超幾何分布モデル，ゴンペルツモデル，遅延S字型NHPP モデル，指数型 NHPP モデルの4つである．データ入力の例を図5に，モデルによる推定結果の例を図6に，それぞれ示す．

SRGM Tool は Java で記述されたアプレットとして実現されており，インターネット上で公開されている[19]．UNIX, Windows, Macintosh 上の Web Browser で実行可能であり，実行速度も速い．ソフトウェア信頼性成長モデルの解説文も合わせて公開されているため，幅広い利用と利用結果のフィードバックが期待される．

5. むすび

本稿では，ソフトウェア工学の将来予測を試みた3つの論文を紹介し，実験を中心にした分析的アプローチがこれからのソフトウェア工学に必要である，という共通の認識が存在することを示した．次に，Case Study や Replicated Experiment と呼ばれる“実験”を基礎とする Empirical Software Engineering の概念や具体例について述べた．更に，Empirical Software Engineering では重要な要素である実験用ツールを OLE や Java で実現した例を紹介した．

OLE や Java は，実験用ツールの開発，配布コストを大幅に小さくする．しかし，実験に興味を持ち参加する研究者や技術者が存在しなければ，Case Study も Replicated Experiment もその実施は難しく，有益な結果をもたらさない．産学の区別を問わず，研究者や技術者が，持てる知識や経験，実験設備を共有する枠組みが必要である[11][15][18]．

参考文献

[1] 有我 成城，衛藤 敏寿，佐藤 治，白神 一久，西村 利浩，村上 列：“Java入門”，翔泳社（1996）．

[2] V. R. Basili and J. D. Musa: “The future engineering of software: A management perspective,” IEEE Computer, 24, 9, pp.90-96 (1991).

[3] C. M. Evertson and J. L. Green: Observation as Inquiry and Method, In M. C. Wittrock (ed.): Handbook of Research on Teaching, 3rd Edition, Macmillan (1986).

[4] M. A. Hamilton: “Java and the shift to net-centric computing,” IEEE Computer, 29, 8, pp.25-39 (1996).

[5] N. Hanakawa, H. Iida, K. Matsumoto and K. Torii: “A framework of generating software process including milestones for object-oriented development method,” Proc. of 1996 Asia-Pacific Software Engineering Conference, pp.120-130 (1996).

[6] 市川 伸一（編），“心理測定法への招待 －測定からみた心理学入門－”，新心理学ライブラリ13，サイエンス社（1991）．

[7] 宮川 治，当麻 喜弘：“Javaを利用したソフトウエア信頼性成長モデルToolの遠隔操作”，1997年電子情報通信学会総合大会（発表予定）．

[8] S. L. Pfleeger: “Software engineering needs to mature,” IEEE Spectrum, 23, 1, p.64 (1995).

[9] A. A. Porter, L. G. Votta and V. R. Basili: “Comparing detection methods for software requirements inspections: A replicated experiment,” IEEE Trans. Software Engineering, 21, 6, pp.563-575 (1995).

[10] W. F. Ticky, N. Harbermann, and L. Prechelt: “Summary of the Dagstuht Workshop on future directions in software engineering,” ACM SIGSOFT Software Engineering Notes, 18, 1, pp.35-48 (1993).

[11] 鳥居 宏次：“変革する日本のソフトウェア工学”，「変革期のソフトウェア工学」シンポジウム論文集（1994）．

[12] K. Torii and K. Matsumoto : “Quantitative analytic approaches in software engineering,” Information and Software Technology, 38, pp.155-163 (1996).

[13] Microsoft Corp.（著），アスキーテクライト（訳）：“OLE2プログラマーズリファレンス”，アスキー（1994）．

[14] “Microsoft Access Database Management System アプリケーション開発ガイド”，マイクロソフト（1996）．

[15] “Software Engineering Laboratory (SEL) - Relationships, models, and management rules-,” Software Engineering Laboratory Series, SEL-91-001, NASA/GSFC (1991).

[16] “VISIOソリューション開発ガイド”，Visio Corp.（1996）．

[17] An International Journal of Empirical Software Engineering Home Page, http://www.cs.pdx.edu/emp-se/

[18] ISERN Home Page, http://wwwagse.informatik.uni-kl.de/ISERN/ isern.html

[19] SRGM Tool Home Page, http://www.dcl.c.dendai.ac.jp/hgdm/

（盛 文 社）