ソフトウェア生産過程の評価実験に関する考察†

有澤 誠††

この論文では，ソフトウェア生産過程およびソフトウェア製品に対して，種々の方法論やツールがどのような効果を及ぼすかを調べるための実験について考察している．評価実験は，方法論やツールを，どのような応用領域に，どのような実験環境で実験を行ったとき得られた結果を，どのような評価基準で調べるかというわくぐみでとらえる．応用領域，実験環境および評価基準について詳しく考察する．特に評価基準については，客観的評価基準と主観的評価基準に分け，後者については，部分的に二段階方式の実験を行うことを提唱している．

1. はじめに

ソフトウェア生産過程の向上をはかるために，種々の方法論やツールが提案されている．しかし，ソフトウェア生産では，実際の生産現場と，研究活動や要員の教育を行う組織との間でのギャップが大きく，この両者の間での協力が必ずしも効果的に行われてはいない．そのため，方法論やツールについても，現実にどのくらいの効果をみこめるものであるかという評価をくだすことが困難であり，生産現場によってこれらの方法論やツールのとりいれかたもまちまちである．さらにまた，ソフトウェア生産過程を特徴づけるデータについても同様に不統一であり，しかも公開されているデータはきわめて少ない．

ソフトウェア生産過程の向上をめざし，提案されている方法論やツールを有効に使いこなすためには，より統一的な評価の方法を確立する必要がある．現状では，まず実験的手段によって，方法論やツールの評価方法を確立することが先決であると考えられる．

1981年6月のACM-SIGSOFT主催の第1回ソフトウェア・エンジニアリング・シンポジウム* では，方法論やツールの評価をするためにどのような実験を行うべきかを集中的に討議している[1]．本稿では，筆者の見地から問題点を整理し，評価実験について考察する．

2. 評価実験のわくぐみ

ソフトウェア生産過程に，方法論やツールが及ぼす効用を評価する実験を計画する際の，実験のわくぐみを考える．ある方法論あるいはツールを，ある応用領域に適用したことの効用を，ある環境で実験し，ある基準で評価することになる．したがって，方法論やツールに対して，(i)応用領域，(ii)環境，(iii)評価基準の3点を定めなければならない．この(i)～(iii)を特徴づけるパラメータは何かを調べ，それを実験でどのように制御するかを考えることが手順になる．

方法論やツールの評価には，それを用いた場合と用いない場合を比較する絶対評価と，複数の方法論やツールの効用を比較する相対評価とがある．絶対評価では，ある方法論やツールを用いないという前提の中に，別の方法論やツールの使用がインプリシットに入りこまないように注意する必要がある．また，相対評価では，応用領域をかなり細かく限定した上での比較でないと公平を欠く恐れがある．さらにいずれの場合でも，方法論やツールを適用した実験環境による影響が十分小さいように配慮しなければならない．

やや抽象的な議論になったので，ここで例をあげておこう．ソフトウェア設計の方法論として，Jackson法，Yordon法の評価をするための実験をしたい場合を考える．どちらも，応用領域は，システム・アナリシスが終了した段階から，コーディングを開始する直前までの期間である．対象として，在庫管理を中心とした事務処理で，小型のホスト計算機による，オンラインのOSの下で，Cobolを用いた場合，といったように，具体的に応用領域を特徴づけることができる．ここではじめて，Jackson法は有効であるとか，Yordon法のほうがJackson法より効用が大きいといった判断が意味をもつ．いっぽう，実験の材料としてとりあげる例題には，引合止めや後日配達注文などを含

† On the Experimental Evaluation for the Software Production Process by MAKOTO ARISAWA (Dept. of Computer Science, Yamanashi University).

†† 山梨大学工学部計算機科学科

* IEEE主催のものは，1981年にすでに第5回を数え，1982年に東京で第6回が開催予定である．

まない基本的なものにするとか，被験者は Fortran で教育を受けたあとで Cobol を学んだ人と，はじめから Cobol で教育を受けて他の言語は知らない人の2人に依頼するとかの，具体的な実験の組み立ては，実験環境の設定である．実験そのものに対するコストの許容範囲内で，Jackson 法と Yordon 法の評価という目的に関連が大きい要素について事前に制御しなければならない．この関連度については，正確には事後でないと判定できないが，同様の実験の蓄積が進むにつれて，かなり正確な推定が行えるようになることを期待するしかない．

最も重要な点は，Jackson 法なり Yordon 法なりの効用の有無を，何の尺度で判定するかである．設計からコーディング，テスト，デバッグを含めた，インプリメント期間が短いものをよしとする，という方法がひとつ考えられる．時間という尺度の中に，コーディングやテストのしやすさまでインプリシットに含めたものとみなすことができる．あるいは被験者にアンケート調査を行って，どちらが設計手法として使いやすいかを問うことも可能である．ただし後者の場合は，被験者に2つの方法で別々に設計を行うことを求める必要がある．前者では必ずしもそうしなくてよい．

以下の章では，応用領域，実験環境，および評価尺度について，詳しく議論する．

3. 応用領域の特徴づけ

方法論やツールの応用領域は，それをソフトウェア生産過程のどの段階に適用するか，事務用あるいは科学技術計算用などどの分野のソフトウェアか，どんなプログラミング言語を用いるか，OS やホスト計算機がどのような性格をもつか，などによって細分する．

段階としては，要求定義からシステム・アナリシスまで，設計，コーディング，テストおよびデバッグ，集積統合から受入れテスト，文書化，運用保守といった，ソフトウェアのライフサイクルのどの段階かを明示する．必ずしもひとつの段階に限定されるとは限らない．たとえば，HIPO は要求定義のツールであると述べた本もあり，HIPO は構造化設計の補助手段であると書かれた本もある．HIPO の効用は，2つの段階について別々に実験を行って評価すべきであろう．方法論やツールの提案者が考えているのとは異なる領域で有効なことが発見されることもあり得る．

事務用科学技術計算用といった分野と，プログラミング言語とは，まとめて考えてもよい．大きく分けて，システムプログラミング，数値計算，事務処理，記号処理などの人工知能，センサを含むシステムの制御，システムシミュレーション，などに分類でき，さらに細分していく．また別の見かたとして，データ駆動か手続き駆動かと大別することから始める方法もある．あるいは，計算主体か制御主体かという分けかたもある．これらの分類方法は必ずしも直交性をもっていないので，分類方法をいくつかに固定してしまうよりも，むしろ対象領域を表わすキーワードを羅列するという，文献などの分類方法と同じ方法をとるほうが便利であろう．

これは OS やホスト計算機に関する分類についても同様である．小型機，大型機，小規模ネットワーク，大規模ネットワーク，専用機，汎用機といった分類から，リアルタイム処理，オンライン処理，バッチ処理，完全自動システム，半自動半マニュアル・システムといった分類についても，むりに直交座標系を固定するよりも，個々にキーワードを並べるほうが，分類を行う立場からも，分類を参照する立場からも，容易であると考えられる．

4. 実験環境の特徴づけ

実験環境は，評価すべき方法論あるいはツールが与えられ，その応用領域を固定したとき，具体的にどのような実験計画を組み立てるかを定める．被験者の選択，題材の選択，および実験の実施方法の3つに分けて考えることができる．

被験者については，それまでにどのような教育訓練を受け，どのような知識や経験をもっているかというバックグラウンドと，実験の時点での力量，さらに性格面の特徴までがパラメータとなり，方法論やツールを客観的に評価するための実験では，最もむずかしい点である．教育訓練および知識経験については，評価対象の方法論やツールに関してと，実験を行う際に固定した応用領域に関しての両面からみる．力量に関しては，実験の前後に補助的なテストを実施する方法もあり，あるいは被験者の上司同僚部下または指導教官による評価を利用することも考えられる．被験者の性格が実験に及ぼす影響を最小限におさえることが問題点である．慎重か，短気か，トップダウン的思考か，ボトムアップ的思考か，集団作業を好むか，個人作業を好むか，強気か，弱気か，パズル的凝り性か，サラリーマン的か，集中力があるか，根気があるか，などここでもキーワードを並べる方法で特徴づけをしてお

くしかない．キーワードは，自己申告とまわりから複数の目で見たものとを並記するのが妥当であろう．実験にあたっては，類似したキーワードの組み合せになる被験者を避け，できるだけキーワードがばらつくように被験者を選ぶことが第一の目安である．もし，定量化が可能であれば，これら被験者に関するパラメータとの相関をとることを実験に含めることも考えられるが，人間因子の定量化は十分慎重に行うべきであろう．

個々の被験者を単独に扱う場合のほかに，複数の被験者をグループにして実験を行う場合もある．この場合には，グループとしての特性がばらつくことだけでなく，ひとつのグループの構成のしかた，すなわち構成員の組み合せのしかたもばらつくような配慮が必要である．また，同一の被験者に複数回の実験を依頼する場合は，実験に対する慣れの要素も考えておかねばならない．もっと積極的に，何回か訓練用の仮実験を経験してもらってから，本番の実験に臨む方法もある．

実験題材は，応用領域の範囲という制限があっても，まだかなり選択の自由度がある．容量の面で，現実のソフトウェア生産に比べて，どうしても小型化したものになることはやむを得ない．むしろ，次章で述べる評価方法との関連性を大切にすべきであろう．評価すべき方法論なりツールなりが，最も有効であると提案者が述べている型の題材と，それとは異なる型の題材いくつかとについて実験を行い，比較することが望ましい．実験結果をみた上で，題材の選択にフィードバックをかけることも有益である．

5. 評価尺度の考察

実験結果をどう評価するかには，2つの見かたがある．ひとつは作られたソフトウェアの品質だけをみて評価する方法，もうひとつは作成過程に関する尺度も併用する方法である．

いずれの場合にも，尺度そのものには，客観的尺度と主観的尺度がある．以下それぞれについて個別にみていく．

5.1 ソフトウェア製品の客観的評価尺度

ソフトウェアが与えられたとき，測定者によらず同一の値が出る評価尺度であり，ソースプログラムの行数あるいはステップ数，McCabe による複雑度[2]，Halstead による尺度[3]の3つが，静的尺度としては代表的である．この内で，プログラムの大きさを示す尺度は，オブジェクトプログラムの容量を用いる方法も含めて，現在広く用いられている．学校の教師が生徒のレポートを枚数で採点するように，理想的な評価尺度でないことは明らかであるが，事前に一定の水準以上のものを別の基準で選別して，その水準以上のものについて適用すると，現実には（学校の教師の場合と同様に）かなりよい評価尺度になっている．実際，プログラムの大きさを示す尺度のほうが，McCabe や Halstead の尺度より優れているという趣旨の報告さえある[4]．

ソフトウェア製品の動的な尺度としては，実行時間を含めた性能，エラーの出現率などが客観的尺度である．実験に用いた題材の仕様の定めかたによって，性能に重きをおくか，それとも信頼性保守性をより重視するかが異なるので，性能を評価尺度とする場合には，その点を強調すべきであろう．当然のことながら，仕様の定めかたは，他の評価尺度にも影響を与えるが，性能に対する影響は他に比べて著しい．

なおこのほかに，モジュール数，モジュールごとの平均容量，モジュールごとの強度やモジュール間の結合度，あるいはモジュール間のよびだし状況などの，モジュールに関する尺度も，製品の客観的な，静的あるいは動的な尺度として使用できる．

5.2 ソフトウェア生産過程の客観的尺度

ここで代表的な尺度は，悪名高いマンマンス（人月）であり，現在広く用いられている．実験の面では，ソフトウェアが完成するまでに要した，人間の経過時間，使用した計算機時間，改訂をした回数，改訂に伴うステップ数やモジュール数の変化のようすなどが，具体的な尺度になる．ソフトウェア製品の場合と異なり，これらの尺度は間接的なもので，必ずしも生産過程の質をそのまま反映しているとはいえない．特に，人間因子に関する評価が表面に現われない．したがって，生産過程の評価では，主観的尺度に期待をせざるを得ない．

5.3 ソフトウェア製品の主観的尺度

ソフトウェア製品の品質とは，ユーザの満足度であると述べた本がある[5]．この場合の尺度は，ユーザごとにばらつきのある主観的なものである．実際，同じソフトウェアでも，使う立場によって，あるいはユーザの個人的経験や趣味のちがいによって，時にはかなり評価に差異を生じる．ソフトウェア製品では，ユーザへのサービス度のほかに，理解しやすさ，変更しやすさ，移植しやすさなど，広い意味での保守しやすさも，重要な主観的尺度である．

主観的尺度で評価するためには，ユーザや保守要員に対するアンケートや面接調査などが一般的である．より系統的に評価を行うには，実験を2段階にする必要がある．すなわち，評価のために2段目の実験を行う．ソフトウェア作成者とは別の被験者に，完成したソフトウェアを使用してもらって，あるいは変更を加える作業をしてもらって，サービス度や保守しやすさを測定する．サービス度の場合，被験者がそのソフトウェアを用いて仕事をする時の時間や，犯すミスの回数および種類などの客観的データをとり，さらにアンケートや面接で主観的評価も行う．保守性の場合も同様で，与えられた変更仕様に合う変更について，生産の場合と同様に実験を行う．2段階式実験によって，主観的尺度に部分的に客観性をとりこむことができる．

5.4 ソフトウェア生産過程の主観的尺度

方法論やツールの実験による評価の中で，この部分が最も興味深い．生産に携わる人が，方法論やツールをどう受けとめるかということが直接反映するのはここだけといってもよい．この評価には，アンケートや面接による方法しかなく，2段階式実験は使えない．被験者の体験をいかにひきだすかという問題であり，ここには統計学より心理学の知識が必要になるかもしれない．

6. 実験の管理運営に関する考察

ソフトウェア生産の向上をめざして，方法論あるいはツールを評価する実験を行うためには，まず現在までの現場での経験やデータを整理することが有益であろう．管理された実験ではないけれども，現実の場で実際に行われた数多くの生産活動は，隠れた実験とみなすこともできる．データを十分にとってないことはあっても，予備実験に代るものとして，定性的な判断の材料を提供することは可能であろう．

実験を実際に実施する場合，そのことに要する費用を見積ると，かなりの部分が人件費になるため，相当高額になることが予想できる．特に，被験者として職業的プログラマに依頼することを考えると，比較的小規模な実験であっても，容易に実施できるとは思えない．実際文献に出ている実験の諸例は，大学あるいは大学院の学生を被験者にしたものが多く，それだけいっぽうではこうした実験に対する信頼度も薄い．また，実験の方法そのものが，あらかじめ定めた結論を立証するようなバイアスをもつという指摘を受けることもしばしばあって，実験はコストがかかる割合に比べて得るところが少ないという考え方が，思いのほか広く浸透しているように見受ける．しかしながら，事前に十分調査を行い検討を加えておいて実施すれば，定量化のむずかしいソフトウェア生産の分野で，実験が果たし得る役割は決して小さくないと考えられる．

7. おわりに

冒頭に触れたソフトウェア・エンジニアリング・シンポジウムでは，8通りの実験提案書を作成するという形に討論の成果をまとめ，6)に報告している．筆者はまた，別の形でソフトウェアに関する実験を行ったことがあり，実験を通して，ソフトウェア生産をまずアートからサイエンスに変え，さらにエンジニアリングに変えていく路があることを信じている．

謝辞 本稿をまとめるにあたって，コネティカット大学の Taylor L. Booth 教授と，日立システム開発研究所小林正和氏とに，ご討論いただいたことが役にたった．ここに感謝する．

なお，本稿の一部は，文部省科学研究費補助金一般研究C（昭和55年度）課題番号558013の研究成果を引用している．

参考文献

1) ACM SIGSOFT: ACM SIGSOFT Sponsored Software Engineering Symposium 予稿集, ACM (1981).
2) McCabe, T. J.: A Complexity Measure, IEEE Trans. Soft. Eng. 2-4, 308-320 (1976).
3) Halstead, M. H.: Elements of Software Science, Elsevier North Holland (1977).
4) Woodfield, S. N., Shen, V. Y. and Dunsmore, H. E.: A Study of Several Metrics for Programming Effort, Purdue Univ. Technical Report (文献1)に含まれている.)
5) 菅野文友：ソフトウェアエンジニアリング，日科技連 (1979).
6) ACM SIGSOFT: Software Engineering Notes 7-1, 6-74 (1982) に掲載の実験プロポーザー
以下にあげる文献は，本文中では引用していないが，本稿に関連が深い．
7) 日科技連：第1回ソフトウェア生産における品質管理シンポジウム発表報文集，日科技連 (1981).
8) Basili, V. R. and Reiter, R. W. Jr.: An Investigation of Human Factors in Software Development, Computer 12-12, 21-38 (1979).
9) Booth, T. L., Ammar, R. and Lenk, R.: An Instrumentation System to Measure User Performance in Interactive Systems, Connecticut Univ. Technical Report (文献1)に含まれてい

る.)
10) Myers, G. J.: A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections, Comm. ACM 21-9, 760-768 (1978) および Comm. ACM 22-12, 687-689 (1979): ACM Forum での紙上討論.
11) 有澤 誠: プログラミング環境とプログラマの心理について, 情報処理学会ソフトウェア工学研究会資料 17-13 (1981).

(昭和56年 9 月 3 日受付)
(昭和56年11月18日採録)