ソフトウエア工学 23－1
（1982. 5. 7）

ソフトウェア・テストにおける エラー検出力の個人差

武田 健二
（日本アイビーエム 製品保証）

1. はじめに

「ソフトウェア危機」という言葉に表わされるように，ソフトウエア開発に対する高信頼性・高生産性の要求はますます強くなっていく。（たとえば[1]） その解決に技術的指針を与えるために，ソフトウエア工学が発展してきた。 要求分析・定義技術，設計技術，テスト技術，開発支援ツール等である。

これらの工学的方法が実験的には進展しつつも，実用ソフトウェア開発現場ではその適用は進んでいない。 今だにソフトウエア開発作業はプログラマ個々人の技術・嗜好にまかされている面が強い。

"Programming is an individual activity." [2]

工業というよりは技芸というべき状態で，ソフトウエア工学の先端技術の採用はなかなか進まない。 開発現場のプログラマは，それらの先端技術を，まだ使いこなせる程には洗練されていない，あるいは有用性が実証されていないと考えているからであろう。

このような現状認識に立つとき，ソフトウェア開発に対する高信頼性・高生産性要求をどのようにして達成すべきか。 それには次の２つが重要である。

1) 工程毎審査活動の拡充
2) チーム編成の見直し

前者については設計審査を中心に既にいろいろなところで検討を加えられているのでここでは論及しない。

チーム編成の見直しとは，分業のあり方を考え直すことである。 ソフトウエア開発管理者は，作業工程を中心に分担する時間分業を積極的に取り入れる。[3] そのときプログラマ個々人のスキル・レベルを適切に把握し，それをもとに分担作業を割当てる。 また教育・経験を通して工程毎の専門家を育成する。 高信頼性・高生産性の達成にはこのような管理技術がソフトウェア工学に劣らず重要である。

この主張は次の仮説を前提にしている。

> 作業工程毎にスキル・レベルの大きな個人差がある。

プログラム作成能力の個人差についていくつかの実証研究がある。（たとえば[4]，[5]） 筆者はこのたびテスト能力（以下テスト力と呼ぶ）の個人差について実験を行ったので，ここに結果を発表する。

2. 実験

2.1 実験方法

本実験は次のように行われた。

まず実際に使用されているプログラムを用意して，それに故意にバグを埋め込んだ。

言語 ----- PL/I
サイズ----- 170 ステップ
機能 ----- いくつかのファイルの内容により，他のファイルを更新する事務処理系プログラム
バグ数----- 15個

このプログラムと，それをテストするためのサンプル・データ（ファイル）が被験者に渡される。 被験者にバグの数は知らされない。 渡されるサンプル・データだけでは少数のバグしか発見できない。 被験者はタイム・シェアリング環境（VM/370 を使用）の

下で適宜データを準備しながらこのプログラムのテストを行う。 このプログラム実行後のファイルの内容からエラーを見つけ出し，エラーの現象・発見時刻を記録用紙に記入する。 プログラム・リストを見てバグを除去する必要はない。 被験者は必要に応じてそれまで発見されたエラーを引起こすバグを実験者（筆者）に除去してもらい，その後またテストを続ける。

このようなテストを1人ほぼ3時間づつ行い，被験者毎にテスト開始後何分後に何個目のエラーが見つかったかをトレースした。 この時間に，プログラム修正の時間は含めない。 データ準備の時間は含める。

2.2 被験者

本実験の被験者は今回は全員理工系学卒で，何らかの形でソフトウェアに関する仕事に従事している。 現在本格的なソフトウェア開発（テストも含めて）には 従事していない。 小規模なプログラムを作成することはある。 被験者のソフトウェア経験は次のようなものである。

- 品質保証業務
- DP部門での適用プログラム開発
- メーカでのコンパイラ開発
- システムズ・エンジニア
- 保守業務
- 端末テスト・ドライバのテスト

2.3 実験結果

実験結果を表1および図1で示す。表中の数字は，被験者毎の1個目のエラー，2個目のエラー，…が見つかった時のテスト開始からの経過時間（分単位）である。

n個目のエラーを見つけるまでの時間の，最長対最短の比が表1最下段にある。 8番のデータを除外した場合の比が下から2段目にある。

7名分のそろっている最後のデータを見るとエラー8個目で2.69倍となっている。 この見方から，限られた時間にどれだけエラーを摘出できるかという生産性の差がわかる。 十分な時間が与えられた時にどこまで摘出できるかで示される信頼性の差はこの実験からは確証できない。 しかし十分予想しうる。

またこの結果からただちに本格的なソフトウェア開発におけるテスト力を推測するのは危険であるが，テスト対象が大規模化・複雑化する程，この差は開くと考えるのが妥当ではないか。

3. 考察

3.1 テスト力

テスト力が大となるためには，次の2つの点が秀れている必要がある。一方のみでは完全でない。

1) 良いテスト・ケースを作る。
   ソフトウェアの中に含まれるエラーを顕在化させるテスト・ケースを過不足なく作る。 抜けがあれば信頼性を損ない，余分なテストは生産性を阻害する。
2) エラーを見逃さない。
   テストの実行結果に現れたエラーを見逃さずに捕促する。 特にオンライン・プログラムにおいてこの力が要求される。

3.2 個人差の原因

何故テスト力に個人差が生じるか。筆者は次の3点によると考えている。

1) 心理特性・思考様式の違い
   次のような個人的特性がテスト力に影響を及ぼす。
   - システム的・論理的思考を行う。
   - 細かいところまで注意が行きとどく。
   - 批判的態度を取る。

・物事をとことんまでつきとめようとし，途中で投げ出さない。

他の作業と異なり，ソフトウェア・テストは終了規準が明確でない。テスト担当者には最後の特性が特に要求される。そしてこれらの総体をテスト適性と呼んでよいのではないか。

2)技術力の違い

テスト力は以下の手段によって強化できる。これらによって身についた力をテスト技術力と呼ぶことにする。

・教育・研究

ソフトウェアのテスト・ケース設定法，テスト・ツール等について研究し，あるいは教育を受ける。

・経験

繰返しソフトウェア・テストに参加し，どうすれば漏れなく効率よくテストできるか身をもって体験する。

・エラー例の収集・分析

ソフトウェア・エラーの実例を収集・分析し，どのようなエラーがあるか，起こしやすいエラーは何かを知る。

([6],[7])

3)テスト対象との関係の違い

テスト対象のソフトウェアとの関係によって，対象に対する熟知度が異なり，また心理的緊張度合(批判的態度の強さ)が異なる。また物の見方も異なる。それ故開発するソフトウェアの種類・規模等により，テスト工程のあり方(誰が，いつ，どのような観点から)を工夫する必要がある。

テストを担当する関係者には次のような人が考えられる。

・プログラミング担当者
・開発チーム内テスト専任者
・保守担当者
・品質保証・検査部担当者
・ユーザ

4. まとめ

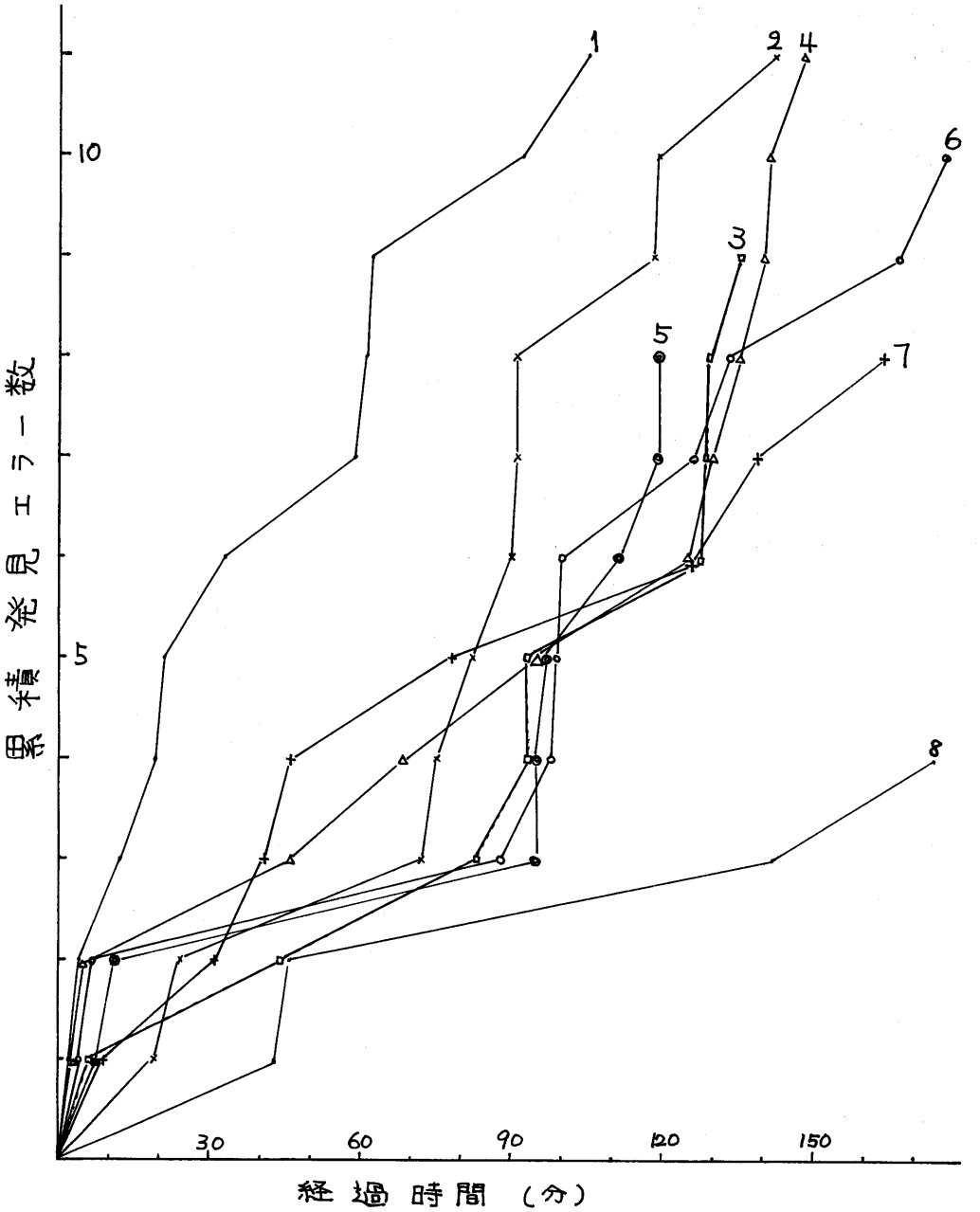
ソフトウェア工学の発展にもかかわらず，ソフトウェアの信頼性・生産性向上へのアプローチが，実用ソフトウェアの開発現場では，まだ確立されたとは言えない。ソフトウェア開発はまだプログラマ個々人の技倆に依存するところが大きい。そしてその技倆も，本実験結果に見られるように，個人差がある。プログラマの工程別技倆程度を把握し，それによって教育を行い，最適なチーム編成を行う。このような管理技術の向上がまだまだソフトウェア開発には必要である。

表1.　実験結果（表）

	EXP	-01-	-02-	-03-	-04-	-05-	-06-	-07-	-08-	-09-	-10-	-11-	-END-	-PROB-	-MIN/PROB
1.	7	2	4	12	19	21	33	59	61	62	92	105	105	11	9.55
2.	9	19	24	72	75	82	90	91	91	118	119	142	145	11	13.18
3.	3	6	44	83	93	93	127	128	129	135			135	9	15.00
4.	6	2	4	46	68	95	125	130	135	140	141	148	173	11	15.73
5.	7	7	11	95	95	97	111	119	119				138	8	17.25
6.	7	4	6	88	98	99	100	126	133	167	176		197	10	19.70
7.	1	9	31	41	46	78	125	139	164				177	8	22.13
8.	3	43	46	142	174	197							213	5	42.60
A		9.5	11.0	7.92	5.16	4.71	3.85	2.36	2.69	(2.69)	(1.91)	(1.41)			
B		21.5	11.5	11.83	9.16	9.38									

EXP：ソフトウェア経験年数
END：テストを中止するまでの時間
PROB：見つけたエラー数
MIN／PROB：エラー1個当たりの平均所要時間（＝END／PROB）
A：8番のデータを除いた場合の最長所要時間／最短所要時間
B：最長所要時間／最短所要時間

図1. 実験結果（グラフ）



参照文献

[1] Harrison T.J., 1980年代へのソフトウェア・チャレンジ, 日経コンピュータ, 1981.11.30

[2] Weinberg G.M., Thy Psychology of Computer Programming, Van Nostrand Reinhold Company, 1971

[3] 小林, ソフトウェア製品生産における知的分業, 情報処理, Vol21 No10, 1980.10

[4] 伊土他, プログラマのスキルを考慮した作成分担に関する一提案, 情報処理学会22回全国大会講演論文集, 1981.3

[5] 平井他, プログラム通性検査の信頼性について, 同24回大会講演論文集, 1982.3

[6] 篠田他, 制御欠陥と機能試験の妥当性評価, 同22回大会講演論文集, 1981.3

[7] 武田他, ソフトウェア・エラーの性質, 同24回大会講演論文集, 1982.3