

ソフトウェア・プロジェクトの完成遅延

安部城一(日立S K)

坂村 健(慶応大学)

坂前和希(慶応大学)

相磯秀夫(慶応大学)

1. はじめに

ソフトウェア工学が発展期に入った今日、なお依然ソフトウェア開発の失敗、特に完成遅延や予定費用超過等の事例が跡を絶つことがない。開発管理の破綻についてその性質を究明し、予知・損失軽減を図ることは情報処理の分野でソフトウェアの占める重要性が増大している今日、以前に増して重要である。

従来ソフトウェアのプロジェクト管理について[1], [2], [3], [4], [5], [7]等の研究が知られている。しかし開発の破綻そのものに着目した追及は筆者等の知る限り学会、実業界を通して組織的に行なわれていない。

この課題に切実な関心を持つ実業界においては、従来個別失敗例に基づき、原因を分析評価して、同様な状況においてどうすべきかと云う知識を蓄積してきた。しかし考察の範囲は個別の状況下に限定され、全体として破綻に關する普遍的特性について知る術がなかった。この問題を打開するため慶応大学と(株)日立ソフトウェア・エンジニアリングの有志がプロジェクト・チームを組み研究に当り、或種の結果を得たので報告する。

本文ではこの開発管理の破綻を破産と呼ぶこととし、この破産に係る主要な問題点を要約すると次の通りである。

- (1) この破産現象の性質について分析抽象化の試みが成功していない。
- (2) 破産の早期検出が困難である。多くの破産はテスト期に入ってから検出され、時には納期直前に検出されることもある。
- (3) 失敗を防止し、万一避けられぬ場合に被害を極小化する有効な管理手

段の研究が進んでいない。

本研究は上記問題点に關して、(1)および(2)の点からアプローチし、破産の対策に關しては後の研究にこれをゆずった。

筆者等は進捗管理が最も困難でありまた破産が検出される可能性が高いテスト期に着目し、広く大型よりミニコンに至る迄の33プロジェクト延100万ステップについてテストする条件を定めるチェックリストの項目数、検出されたバグ数のデータを時間をパラメータとして収集した。次にこのデータをプロジェクトの規模、期間につき正規化し、共通の傾向を求めるためモデル化を行ない破産例と正常例につき考察を行なった結果

- (1) ソフトウェアプロジェクトがテスト期に示す進捗特性を数件指摘した。
- (2) テスト途上のマクロな進捗度評価と破産の予知方法を提案し、方法の評価を行なった。
- (3) 破産の原因について要因を分類し、成否に最も影響力を持つ管理者はいかなる場合に失敗することが多いかについて論じた。

2. 問題の設定—破産現象

本文では破産とは納期遅延、または予定費用超過により経済的あるいは信用上の大きな損失をもたらす場合を言う。遅延は通例予定以上の期間担当者拘束するので費用の超過を伴う。遅延と費用超過は現象を観測する立場を変えた測定量であり、実質は一個の現象である場合が多い。従って破産の検出については代表的特性である遅延の測定によって行なうものとする。

2.1 大損失を伴う現実の破産

コンピュータにより処理をする業務の範囲は応用分野の拡大につれて、重大な責任を伴うものを含む傾向がある。例えば現代製鉄所の総合管理にコンピュータを応用する場合、この神経中枢が動作しない限り製鉄所はフル稼働に入り得ない。20万トン/月クラスの製鉄所建設には約0.3兆円[18]、[9])を要する。今総合管理用ソフトウェアの開発遅延の責任でこの投資対象が1ヶ月雨ざらしになったとすると、少なくとも15億円の金利損失を考えねばならない。10万トンの鉱石運搬船が接岸する港湾設備や建設中の高炉、転炉、連続鑄造工場、および数十コースのゴルフ場に匹敵する敷地を見て、社会的責任を痛感しないソフトウェア管理者はいない。見渡す限りパイプと装置が並ぶ石油コンビナートの制御や山を抜き国土をおく高速鉄道の運行制御の場合もソフトウェアの開発は同様な立場にあり、納期・品質に関する社会的責任は益々重大である。

一般のソフトウェアの売買においてはソフトハウスの責任は契約金の範囲内に限られる。特殊な契約条項がない限り品質や納期に問題を生じた時、最悪でも契約破棄の範囲の損失にとどまるであらう。しかし大プロジェクトにあっては管理者はソフトウェアを含めて、どのサブプロジェクトに遅れが発生してもこれに対処し、いかなる犠牲を払っても遅れを挽回せねばならない。

従ってその余力を保有する者だけが、大プロジェクトに責任を持って参加することが可能である。

このような重責任を伴うソフトウェアの開発に遅れがある場合、可能なあらゆるリソースを投入してどれだけ遅れを回復できるかが問題となる。結果として単位時間当りの投入費用はFig. 1に見られるような傾向を示し

大きな損失は破産検出以降に発生する。

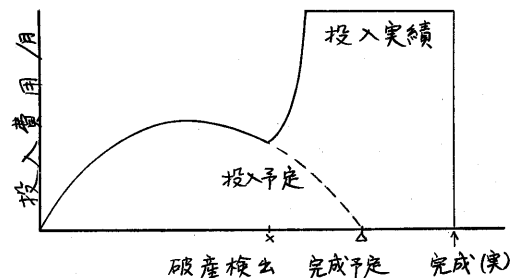


Fig. 1 破産ソフトウェアプロジェクトの月別費用投入例

2.2 破産時に起る現象

破産を検出した時、次の現象が起る。対策が可能で期限に間に合う可能性があればFig. 1の投入費用のピークは前にあわせるに必要な高さの範囲に限定することができる。

しかし間に合う可能性がないか、どれだけ遅れるのか見通しを失った場合はFig. 1のピークの高さを制限する要素はなくなり、高さは投入可能な人員および計算機時間の最大値により定まる。特に見通しを失った場合はピークの中が判らなくなり、累積する赤字の予想がつかないと云う常識的破産に近い状態に陥る。

この追加投入の目的は遅延の短縮にあり、先の例のような重責任ソフトウェア開発の場合、プロジェクト全体の投資回収遅延損失とピーク用追加投資による損失の合計を最少化することが合理的であるが、一般には遅延期間の最少化を損失ミニマイズ条件と近似する考えが有力である。

例え一日でも遅延が予想されれば、動員可能な最大限を投入することが普通である。進捗予定が立てられる程度に管理能力が回復しても費用投入のピークを低減することはできない。しかしその持続中つまり損失金額を限定する効果を期待することは可能である。

2.3 検出時期

破産の検出は早いに越したことはない。しかし例えばテスト期の中間時点では遅すぎてソフトウェア管理者の役に立たないとは言えない。所要人月の例えば9割が投入済みとなり残余1割の時期に破産を検出するのでなく、重責ソフトウェアの場合大損失は検出後に発生するため、損失の大部分を事前に検出する意味がある。更に大プロジェクトの管理者に遅延を予告し、対策を協議する上でも意味がある。

検出の時期がいつであれ、大きな犠牲を伴うので管理者は明確な根拠なしでソフトウェア・プロジェクトを破産したと判定する決心がつかない。そしてかなり明らかな根拠が得られるのは後述によればテスト期なのである。

2.4 破産による損失の軽減策

破産時の損失軽減策について、その概要を承知して置くことは、検出に關する理解を進める上で有用であろう。多くの対策の中で効果の定量的裏付けが行なわれていないが、経験的に有用と考えられるものを次に示す。

(1) 進捗見通しの精度向上

破産の判定は進捗見通しに基づいて行なう故に、進捗見通しは達成率の面で出来るだけ高精度でありたい。それだけでなく、精度が高い程早期に破産の判定を行なうことが可能となり、プロジェクトの遅延を防止する可能性が向上し、防止できると判れば投入ピークを必要限度に止めることが出来るのである。

この精度向上は主として問題点の情報収集、並に近い将来の問題予測の作業を通して実現される。

(2) 対策の事前準備

ピーク投入量を低減する事は基本的には間に合わせる方策を見出すことに帰着する。その一つとして破産の判定に先立って、投入予定技術者の

事前準備(例えばシステム仕様書や機能設計に關する記録等を調査させて置く)があり、更には重点投入目標を定めるため問題モジュール、または問題機能はどれか、等の状態を把握しておくことも準備として役立つ場合が多い。

3. 方法

3.1 マクロな進捗評価

進捗状態をマクロに把握しプロジェクトが破産に至る可能性を視覚的に表わすための採った一方法について述べる。破産には予定費用を越えるものと期限より遅れるものとがある。前者についてはどこまで人員と他のリソースを削減できるか調べねばならぬが、これは削減後のリソースによる進捗度を見積ることにより期限に丁度間に合うリソース投入量を定めることができる。その結果管理者は生じる費用の大きさを予想することができる。後者の場合および前者においても、破産を予測する上での主要な問題は進捗度を評価し、完成日が期限に間に合うか否かを判断する問題に帰着する。

本研究ではソフトウェアプロジェクトが破産すると判明するのはテスト期であると云う事実に着目した。ここでテスト期とは開発工程をプランニング期(機能設計)、プログラミング期(機能実現法決定よりモジュール単体デバッグ完了)、およびそれ以降に区別した最終部分に当り、モジュール結合から統合テスト迄を含んでいる。

テスト期における進捗状況を観察するためにプログラムチェックリスト(以下PCL)の未完了テスト項目数対作業日数、および累計検出バグ数対作業日数を示すPB曲線図を使用した。これは残作業量を未完テスト項目数で表現し、作業日数の経過と共にその減少

状態を示すものである。

3.2 PB曲線図

Fig. 2にPB曲線図の例を示す、縦軸にPCL残存項目件数、横軸に作業日数を示している。PCLとはプログラムに加えるテスト方法を規定するリストである。PCL項目には簡単な確認もあり、複雑なテストも含まれている。テストの順番はテスト項目の難さとは無関係に定まると仮定を設け、プログラムの大きさをステップ数で計る様に残作業量は残存PCL項目数で表わされると考え、残存件数が0件に達した時テストは完了したと見るのであ

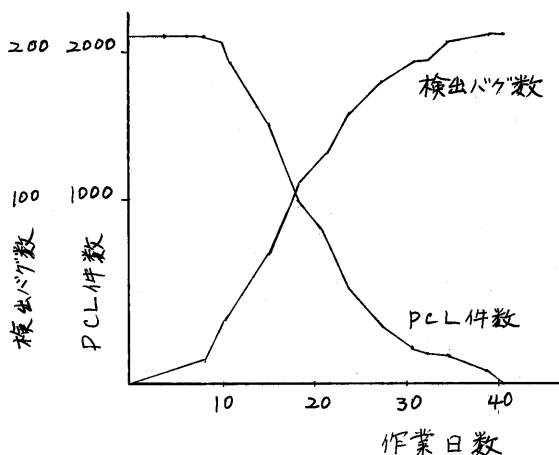


Fig. 2 PB曲線図の例

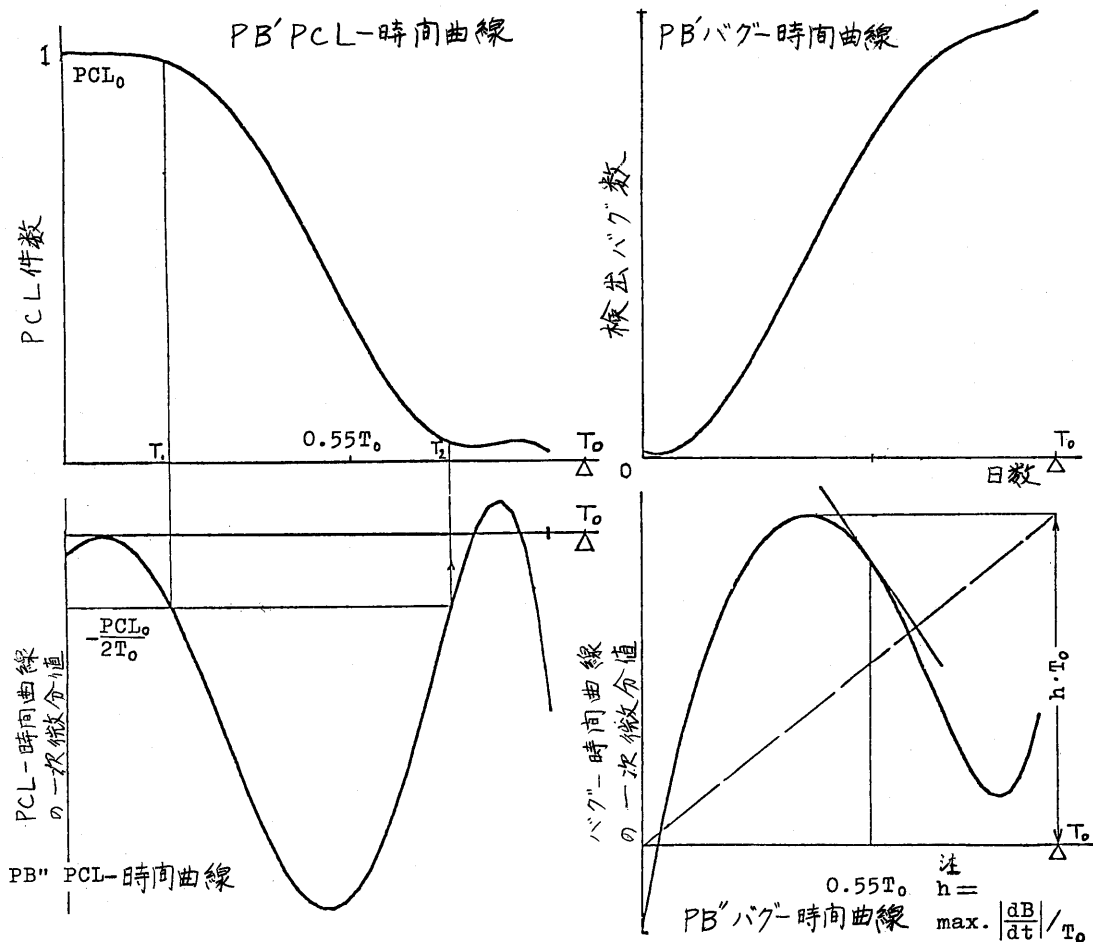


Fig. 3 PB曲線図例

る。

検出したバグはこれを除去し、修正の確認を行なわねばならぬ。この修正と確認が終わって初めて当該テスト項目がPCL残存件数から削除されるのでテストの総合的進捗度はPB曲線によりマクロに表示される。

多くのプロジェクトより得たPB曲線に共通な傾向を抽出しFig. 4に示す。この傾向はテストを三期に区分し夫々を直線で示すものである。初期 t_1 には一般に制御が完結せずPCL項目を合格・消化するペースが低い。

バグの摘出が進むにつれて制御がプログラムの末端に近到達すると急速にPCL消化が進展する。これを中期 t_2 とする。

やがて検出・再現に困難を伴う複雑なバグにより消化が困難なPCL項目が残ることと同消化ペースは再び低下し終期 t_3 に入る。PB曲線よりこの三直線モデルを求めた方法を次に示す。

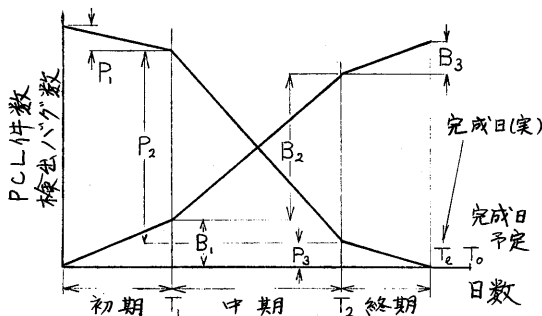


Fig. 4 三直線モデル化したPB曲線

3.3 PB曲線の多項式回帰

作業日毎のテストの成果には変動がありPB曲線は微かな凹凸を示すので大勢を判断するに必ずしも適切でない。このためにPB曲線上の座標値をコンピュータに入力し、6次式で回帰曲線を求めた。この出力曲線をPB曲線と呼ぶことにし、一例をFig. 3に示す。PB曲線はPCL件数、バグ数およびテスト日数について最大値により正規

化をしておくものとする。

バグ数の最大値はテスト中に求めることができない故、累計値を常に1として正規化する。テストが完了した時点でPB'バグ-時間曲線は検出バグ総数により正規化された値を示す。

作業日数についてはテスト期に入ってから期限迄の長さを1として正規化を行なう。以上により諸プロジェクト間のPB'曲線の比較を行なう基盤を求めた。

PB'PCL-時間曲線を時間について微分した曲線をFig. 3のPB'PCL-時間曲線に示す。初期が終る T_1 は次により定義する。

$$PB'(T_1) = -PCL_0 / 2T_0$$

但し $PCL_0 = PCL$ 項目数初期値

T_0 = テスト期の予定日数

PB'PCL-時間曲線は T_1 を過ぎてしばらくは減少を続けやがて終期の終りに0に接近する。曲線が再び $-\frac{PCL_0}{2T_0}$ の値を示す時を T_2 としこれを中期と終期の境界とする。

3.4 データの正規化

延100万ステップにおよぶプロジェクトからテスト期のデータを集めPB'およびPB'曲線を求めた。次に T_1 , T_2 におけるPCL値 $(1 - P_1)$,

$1 - (P_1 + P_2)$ およびバグ数 B_1 ,

$B_1 + B_2$ を全ての曲線から求め、Fig. 4に示す三直線モデルに変換し、直線モデルの分類・研究を進めた。なおここで初期、中期、終期におけるPCL消化数を P_1 , P_2 , P_3 と P_i ($i=1, 2, 3$)を $P_i = p_i / PCL_0$ としている。

PCLと同様バグ検出数についても初期、中期、終期夫々において b_1 , b_2 , b_3 であるとし、 B_i ($i=1, 2, 3$)を次のように定めた。 $B_i = b_i / (b_1 + b_2 + b_3)$

作業日については二つの正規化方法がある。テスト期のソフトウェアの性質を考察するためには実際の所要日数

(t_e)により正規化し、予測方法を考察するためには計画所要日数(T_0)により正規化を行なった。Table 1 に収集した諸プロジェクトの P_i , B_i , 等の値の平均値を示す。

4. PCL, バグ数, 日数の間の関係

収集したデータより上記三変数の間に次の関係を見出した。

4.1 初期に見出される性質

(1) デバグ効率

ソフトウェアを出荷した後に検出されるバグを除くと初期のテストによるデバグ効率([3])は B_1 により近似される。以後値は指定ない限り平均値とする。全プロジェクト平均値 $\overline{B_1}$ は 0.2178 であった。3.1 の定義によりテストに入る前に全モジュールはデバグされている。故にこの時期に検出されるバグは制御の主要路の周辺にあるインタ・フェースバグであると推論されそれはテスト中に見出されるバグ全体の2割に相当する。

(2) バグ検出の難易度

初期におけるバグ検出は中期より困難であると言えない。Table 1a によれば $\overline{B_1}/\overline{t_1}$ は 0.8776 であり中期の $\overline{B_2}/\overline{t_2}$ の 67% に当る。これは PCL 消化速度のこの二期間の変化に比して 1/6.8 倍でない。

(3) 初期の長さ

$\overline{t_1}$ は Table 1a によれば 0.208 である。

この時期に検出されるバグは主要路の周辺のインタ・フェース・ミスが多く、主要路に沿ってのコーディングレベルにより効果的に除去し易いと考えられる。全テスト費用の1/6倍はこの除去効果の最大期待値を示している。

4.2 中期に見出される性質

(1) PCL 消化率

$\overline{P_2}/\overline{t_2}$ は 1.72 であり、初期のそれの 12.9 倍を示した。

(2) テスト中期によるデバグ効率

上記を B_2/B_2+B_3 と定義し Table 1b に平均値を示した。中期には PCL 消化率が高い。PCL 残存項目数は残作業量、作業日数は費用を意味す

Table 1a P_i , B_i , t_i 類の平均値

項目	$\overline{B_1}$	$\overline{t_1}$	$\overline{B_1/t_1}$	$\overline{P_1/t_1}$	$\overline{P_2}$	$\overline{B_2/t_2}$	$\overline{P_2/t_2}$
値	0.2178	0.2048	0.8776	0.1343	0.8847	1.307	1.720

項目	$\overline{B_3}$	$\overline{B_3/t_3}$	$\overline{P_3/t_3}$	$\overline{B_1/P_1}$	$\overline{B_2/P_2}$	$\overline{B_3/P_3}$
値	0.1553	0.7081	0.3212	11.200	0.8009	2.7002

Table 1b 中期の期間・デバグ効率の平均値

項目	総平均	正常例平均	準正常平均	破産例平均
期間(t_2)	0.5735	0.6533	0.5679	0.2881
デバグ効率	0.8350	0.8900	0.7335	0.6368

注: 中期のデバグ効率は $\frac{B_2}{B_2+B_3}$ とする。

Table 2 PCL 消化およびバグ検出速度の平均値および変動係数

テスト 期別	平均値		変動係数	
	P_i/t_i	B_i/t_i	P_i/t_i	B_i/t_i
初期	0.1343	0.8776	1.225	0.6508
中期	1.720	1.307	0.3330	0.2672
終期	0.3212	0.7081	0.8453	0.9532

るのでPCL消化率が高いとは単位費用当りの作業量が大きく効率が良い事を意味する。よって中期のデバッグ効率が低いことが望ましい。破産および準正常例の平均は0.714であり正常例平均は0.89を示した。

(3) 中期の長さ

t_2 の平均値をTable 1 bに示す。正常例平均0.5753, 破産例0.288であり、後者が如何に短かく効率が低いかを示している。

(4) 変動係数

P_n/t_n ($n=1,2,3$) および B_n/t_n ($n=1,2,3$) の全プロジェクトによる変動係数をTable 2に示す。両者の変動係数は中期において最少であり、三直線モデル化による共通特性を抽出する試みは中期において最も成功している。

4.3 終期に見出される性質

(1) PCL消化率

Table 1 aによれば P_3/t_3 は0.321であり中期の0.186倍に相当する。

(2) 検出バグ数対PCL消化件数比によりPCL項目のバグヒット率を求めTable 1 aに示す。終期の B_3/P_3 は2.7であり中期のそれの3.4倍を示している。

4.4 破産の検出

(1) T_1 の値による判定

T_1 はPCL消化率が平均値の半分に達する時点であった。 T_1/T_0 の値が1に近づく時破産は不可避である。 T_1/T_0 の値を正常, 準正常, および破産例別にFig. 5に示す。 T_1/T_0 の値が0.55を越えた時、破産は不可避であると判定するものとし、この方法の評価を行なう。

T_1/T_0 は正常および準正常例について0.146であり、破産例については0.966を得た。 T_1/T_0 の推定標準偏

差は夫々0.140および0.659を得たので、正常または準正常例を破産に到ると誤って判定する危険度は0.5%程度と評価される。一方破産例を破産に到らぬと誤る方の危険度は約24%と評価されるので何等かの改善策が必要である。その一例を次に示す。

(2) バグ検出速度の変化による判定

PCL消化率が向上(加速)しながらPB'バグー時間曲線が上昇しなければ、テスト中のPCL項目により定まる制御の流れの周辺においてバグ密度が低下しつつあり、品質向上の見通しが明るいと考えられる。

時刻 $t = 0.55 T_0$ におけるPB'バグー時間曲線の二次微分値を求め、次にこれをバグー時間曲線の1次微分の最大値を T_0 で除した値: $\max |dB'/dt| \cdot T_0$ により正規化を行なう。Fig. 6にこの値を示す。この値が+0.3以上であれば破産に到ると判定すると、Fig. 6には+0.3以下が13例あり内1件が破産に到っているので破産見落とし確率は約8%と期待される。この方法と T_1 による判定方法を結合させることにより破産見落とし危険率は1.8%程度に迄改善が期待される。

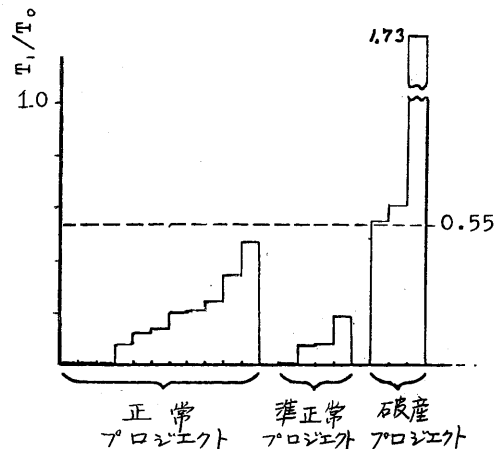


Fig. 5 T_1/T_0 の分布

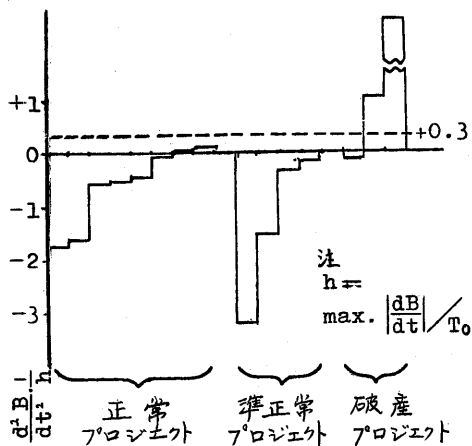


Fig. 6 $\frac{d^2 B}{dt^2} / h$ の分布

5. 破産原因の分析

夫々の破産例の原因につき分析し、情報を集めることにより、同様な状況の下ではどうするべきかを知ることが出来る。更に破産原因を集め分類し共通の性質を研究することにより、プロジェクト管理に有用な新しい知識を得ることが出来る。

Table 3 破産要素の状態分類表

要素	状態：定義	要素	状態：定義
管理者	計画不良あり : 見積り誤り : テスト計画不良 : 仕様決定遅延 判断不良あり : 類推不良 : 実態把握不良, 実態を知らず不適切な根拠で出来ると判断 : 報告過信, 悪い報告を欠いた粉飾報告を過信した。 指揮不良あり : 放任・無指導 : 管理過剰 無過失 : 管理者の申告によれば無過失	担当者	担当者の過半数が未経験 : 対象分野未経験 : 使用するリソース未経験 担当者の過半数は経験有 : 対象分野とリソースに経験有
		他プロジェクト	に依る影響あり : 本プロジェクトチームは他の困窮プロジェクトにメンバーを貸出せねばならず、工期の10%以上発足が遅れた。 に依る影響なし : 影響は10%以下であった。
		新規形態	新規 : 新規開発 改造 : 既製品を一部利用して開発

破産の原因は一般に多くの要素が結合している。例えば進捗報告から判断してプロジェクト管理者が順調と考えていた所コーディングを終った時点で規模が所期の倍に上ることを発見した。一方このプロジェクトのメンバーの一人はこの対象分野に未経験でありある種のテストケースに気付かず、問題の発見が遅れて期限の前日に4ステップ程の修正が必要となることもあろう。

このような複合した形を含む諸原因を我々は三つの要素に大別した。即ち管理者、担当者、および環境である。この内、環境は更に二つの独立な要素に分け、他プロジェクトによる影響と開発の形態を採り上げた。各要素が破産に関与する状態を分類しTable 3に示す。これらの情報は研究中のプロジェクトの管理者に面接し、類似原因をグループ化して整理したものである。

この表を用意し、各管理者に破産に最も大きく貢献した状態を各要素毎に1件選択するように求めた。Table 4に破産および破産寸前で回復するを得た準正常プロジェクト11件の要素別

分布を示す。

この表より管理上の失敗の大半は判断不良に帰し、その次に多い計画不良との間に差が認められる。失敗例の数は充分とは言えないが、このサンプルがソフトウェア産業の一般的傾向を代表していると仮定すれば管理者はTable 3の判断不良に最も注意すべきであることが判る。

表のオ3欄より右は特定の管理上の状態にかかわる、他の三要素の状態区分毎のプロジェクト件数を示す。管理上過失なしとするオ4行の2件は他プロジェクトの影響なく、担当者未経験に偏りを見せている。

最も頻度が高い判断不良について、管理者はいかなる場合に判断を誤ることが多いかを探るための他の要素の状態との関係をTable 5に示した。この表から改造物件の場合に判断を誤った件数が6件中4件を占め、かつ4件其他

プロジェクトの影響を受けていた。

全体のプロジェクト数に占める改造の割合が高くてこのように見えるであろう。そこでTable 5のオ6欄に問題の11プロジェクトと同年代の全プロジェクトの新規・改造比を示した。オ7欄に破産および準正常例の同比を、オ8欄に判断不良例の同比を示す。

全体として改造物件を扱い、他プロジェクトの影響を受けている場合、管理者はTable 3の判断不良を防ぐよう特に留意すべきである。

6. おわりに

本文ではソフトウェア開発プロジェクト一般の破産について、その原因となり得る要素を分析し、更に破産の検出方法について提案を行なった。

事例23件延100万ステップを調査し、テスト期において、PCL数、模

Table 4 破産(含準)原因分析表(管理状態別各要素状態発生件数)

管理者の状態	破産(含準) プロジェクト		左記に関する各要素の状態(内訳件数)					
	件数	%	担当者経験		他プロジェクト		開発形態	
			有	無	影響有	影響無	新規	改造
計画不良	2	18	1	1	1	1		2
判断不良	6	55	4	2	4	2	2	4
指揮不良	1	9	1			1	1	
無過失	2	18	2			2	1	1
計	11	100	7	4	5	6	4	7

Table 5 判断不良と他要素との関連

開発形態	件数	担当者経験		他プロジェクト		開発形態比率		
		有	無	影響有	影響無	全体	破産(含準)プロジェクト	判断不良ありプロジェクト
新規	2	2			2	53%	36%	33%
改造	4	2	2	4		47%	64%	67%

出バグ数、および時間の間の関係が単純なモデルに置換えられることを示しこれによって破産の判定が可能となった。更にテスト期にソフトウェアが示す一般的性質を9件指摘し、工程計画の信頼度向上に役立たしめた。次に破産または破産をようやく回避した準正常プロジェクトにつき、その原因を分析し、特定の要素の組合せ状況を追及し、どのような環境の下で管理者が失敗することが多いかを論じた。

ソフトウェアの開発に際し、テスト期以前に破産の危険性が高いと指摘した状態を発見した場合、管理者は破産損失軽減策を並用することが出来る。更にテスト期以降には破産の検出法によりこれを予知することが可能である。これによってソフトウェア開発者およびコンピュータを含んだ大プロジェクトの管理者が、ソフトウェア開発の不確実性により損害を受ける可能性を低減し得るものと考えられる。

最後にこの研究を進める上で程々御援助を頂いた慶応大学部工学部、相模研究室の方々、並にデータ収集に御協力頂いた日立ソフトウェア・エンジニアリング社幹部の方々に感謝する。

文献

- (1) L. H. Putnum : " A Macro Estimating Methodology for Software Development " COMPCON PP 138-143 (1976)
- (2) M. K. Starr : " Systems Management of Operations " PP 216-218 Prentice-Hall Inc. Englewood Cliffs. N. J. (1974)
- (3) T. C. Jones : " Measuring Programming Quality and Productivity " IBM Syst. J. Vol 17. No 1 PP 39-63 (1978)
- (4) F. Brooks : " The Mythical Man-month " PP 14 f. Addison-Wesley Publishing Co. Reading, Mass.
- (5) 坂村 健 : " A Debugging Machine an Approach to an Adaptive Computer " IFIP PP 23-28 (1977)
- (6) 芝田 寛二 : " ソフトウェア開発管理の改善 " 鉄鋼のIE Vol 15-4 PP 22-27 (1977)
- (7) E. M. Weinberg : " The Psychology of Computer Programming " PP 100-112 Van Nostrand Reinhold Co. New York, N. Y. (1971)
- (8) " 有価証券報告書総覧 " No 11-68 (新日鉄) PP 29. 大蔵省印刷局
- (9) " 有価証券報告書総覧 " No 11-4 (日本鋼管) PP 20-21 大蔵省印刷局