適用業務システム開発における生産性とエラー

工藤市兵衛・鈴木　達夫・近藤　高司

Productivities and Errors on the Development of Application System

Ichibei KUDO, Tatsuo SUZUKI and Takashi KONDO

The big scale and complicated software system is demanded by the system user.
Recently, this development requires a large quantity of human power which calls for agreat deal of cost. Rasing productivity on the system development is necessary. This paper discusses productivity of programming from the point of view of examining the causes of programming errors.

Ⅰ．はじめに

高度情報化社会の進展する今日，その中心的役割をするコンピュータは，ハードウェアとソフトウェアの両方が相まって機能し高度な情報処理システムが完成する。ハードウェアはエレクトロニクス技術の革新によりその性能は大幅に向上して，その価格は年々低下して来ている。それに比べソフトウェアは労働集約的な作業によって作成されソフトウェア開発費用の大部分はシステムズエンジニアやプログラマー等に支払われる人件費であり年々上昇する傾向にある。ハードウェアの性能価格費は10年間でほぼ100倍向上するのに対し，ソフトウェアの生産性は10年間で約2.5倍にしかなっていないと言われている。コンピュータシステムに不可欠の，ソフトウェアは，①開発，保守費用が高くなる。②信頼性が低い。③大規模なソフトウェアシステムの開発は困難。④ソフトウェアの開発管理が困難，等の問題が存在している。一般に企業のEDPシステムでは，ハードウェアとオペレーティングシステム等の基本ソフトウェアはメーカーから購入し，各企業の業務に合致する適用業務ソフトウェアは独自で開発を実施している。今日，企業経営の観点から，オンラインシステムの様に複雑で高度なEDPシステムを開発するためには莫大な資本投資をしなければならず，この点の合理化は増々重要となると考えられる。ソフトウェアは人間の知的労働により作成されるが，人間の人間らしさであるミスにより，エラーが内在してしまう。ソフトウェア開発の工程で，その半分の労力は人間が作り込んでしまったミスを検出・修正するデバッグ作業であり，ソフトウェアの信頼性，またはEDPシステムの全体の信頼性に大きくかかわりあう仕事である。当研究においては，企業独自で開発する適用業務システムの開発時におけるエラー発生の問題を詳細に検討せんと試みるものである。

Ⅱ．システム開発工程とエラー

ソフトウェアの開発は種々な作業工程に分割して，複数のコンピュータ要員により作業がなされる。表1には標準的な開発工程と，その各工程により生産されたドキュメント類やプログラムを示している。ソフトウェア開発は設計・製造の過程が直接に見ることができない頭脳的労働によって作業が進行する特殊性があり人的要素が大きく寄与する。調査分析工程では，現状等の分析がなされシステム設計では，調査報告書を基にしてコンピュータシステムの各使用等を考察し，システムの機能を細分化する。この時，作業者のミスにより表2に示す様な設計不良や仕様不良，条件設定エラーが内在してしまう。これらのエラー原因が入ったまま次のプログラミング工程に進みプログラム設計書，テスト仕様書等が作成されるがここでフローチャートの作成不良やコーディングのミスが発生する。プログラムは単体又はモジュールに分割されて最少機能単位となり，エラーの検出（単体テスト）が行われる。総合テストでは，モジュールのインターフェスのテスト（結合テスト）と総合システムテストに分かれるがテスト中に，不良を修正するが一部は改悪となり機能が不良になるデグレートが発生する。システムの開発工程は直列システムであり前工程のエラーが次工程に反映してしまう。これらのシステム開発工程は，情報処理の手順をコンピュータに指示する詳細な命令語

表1　ソフトウエア開発における標準工程とドキュメント，プログラム類

標準工程	作業範囲	アウトプット
調査分析	現状分析 予備設計 開発計画 技術計算の場合の理論解析	調査報告書 理論解析説明書
システム設計		
基本設計	機能設計 システム構成とサブシステム分析 ファイル設計 入出力帳票と基本ファイルの設計 システム評価方法の設定 テスト仕様の計画・内容	システム基本設計書
詳細設計	プログラム構成の設定 プログラム基本設計 プログラム機能仕様の作成共通エリア，メッセージ，詳細ファイル，内部コード等の定義 技術計算の場合の数値解析	詳細設計書 数値解析説明書
プログラミング		
プログラム設計	プログラム仕様の作成 メッセージ覧，エリアの定義含む 単体テスト仕様の作成	プログラム設計書 プログラムテスト仕様書
プログラム作成	プログラム・フローチャートの作成 コーディング 机上デバッグ，アセンブル，コンパイル 単体テストデータの作成 単体テスト（＝デバッグ）	プログラム・フローチャート プログラムリスト（媒体含む） テスト結果
総合テスト	サブシステム総合テスト システムテスト テストラン 総合テストデータ作成	総合テスト結果報告書 プログラム（完成品）
マニュアルの作成	操作説明書，利用者マニュアルの作成 納品物件整理を含む	操作説明書 利用者マニュアル

（出典　ソフトウエア産業振興協会）

の大規模な集積化作業であり，情報処理手順・解法を表現する手段の集大成されたものの構築であり，その為に，報告書をはじめとする情報（表現）を詳細に分割変換する過程であり，その分割変換作業を人間が行うので完全に変換されない時にエラーが内在してしまうと考えられる。システム開発工程の4割以上の作業はエラーを検出し修正するデバッグ作業であり，そこに多額の費用が発生する。期待される情報処理の機能が完全に作動するソフトウェアの開発，すなわち，コンピュータ上にインプリメントしてエラー発生が皆無で運転できるプログラムの作成はコンピュータ化した社会において重要な問題であろうと考える。

III．適用業務システムの開発例

今日，銀行や証券会社等の金融機関におけるコンピュータシステムは大変高度な利用形態を採用し，事務処理

表2　システム開発時のエラー発生要因

作業工程	内容
基本設計	要求把握エラー（基本設計不良）
	その他
詳細設計	機能～論理展開不良（仕様不良）
	インターフェース不良
	テーブル設計エラー
	ファイル設計エラー
	その他領域設計エラー
	境界条件設定エラー
	その他
フローチャート	ロジック不良
	初期化エラー
	比較エラー
	フラグ設定エラー
	その他
コーディング	エリア（領域名称）不良
	命令語不良
	レジスター（Register）不良
	名標／分岐先不良
	その他
システムテスト（結合：総合）	ディグレード（修正ミス）
	その他

業務の機械化に始まり，昭和50年頃からは第2次オンライン化と呼ばれる全勘定科目のコンピュータ化，全営業店のオンライン化，顧客情報管理の充実を目ざして効率的な情報処理システムの構築が進んで来ている。この様な状況のなか，新規取扱い業務の事務処理をオンライン処理するためのシステム開発が始められた。

III-1．開発システムの概要

システムの概要は図1に示す様にオンライン処理系とオフライン処理系から構成されて，前者は各営業店の端末機から取引情報が入力されると元帳ファイルの内容と照合され取引ジャーナル用の磁気テープに記憶するデータ収集機能と入力されたデータの検証機能を持つ様設計されている。オフライン処理系は，オンライン処理で収集された取引情報に基づいて元帳ファイルと追加ファイルの内容を更新する機能と業務に必要なデータテープの準備，各種の報告帳票の編集と印字出力等の機能から成り立っている。オンライン処理系は約50本のプログラムから成り開発に使用した言語はアセンブリー語12084ステップでオフライン処理系は約20本でアセンブリー語12386ステップPL/I言語で7349ステップのプログラム規模となった。ここでPL/Iは主に帳票編集や印字出力に関係するプログラム開発に使用し生産性の向上を計っ

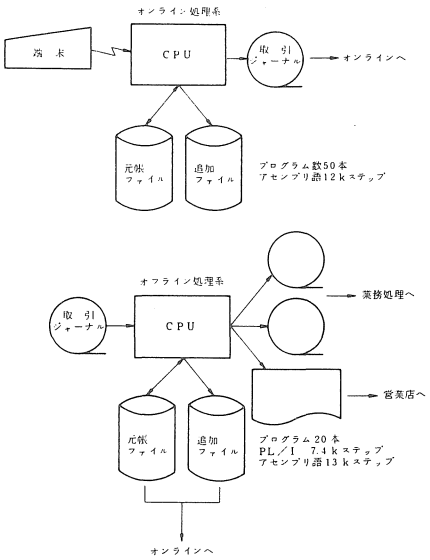

図1　適用業務処理システムの概要

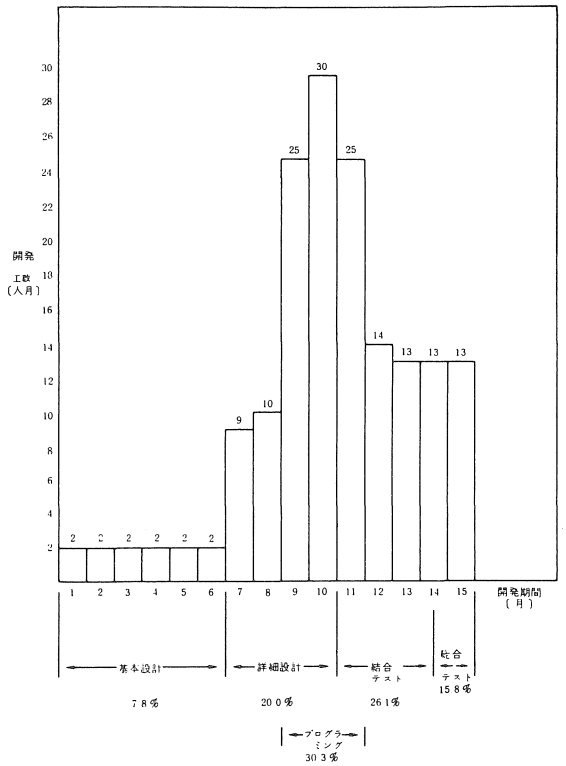

図2　適用業務システム開発の作業工数

ている。

III-2. システム開発工程

開発期間は15カ月で図2の様な要員の割振りで行なわれた。開発工程は，機能別に分割できるシステムであるため詳細設計工程以降は同時に並行して作業が進行できるため図の様に重複している。工程の中ほど(詳細設計，プログラム開発，結合テスト）に要した作業工数は全体の76.4%であり，特にプログラミング工程に膨大な要員の作業を必要としていることがわかる。システム開発に要した直接作業工数は165人・月であり，それに間接的な運用や各種の手続き，報告書作成準備等の補助作業に常時20名が係わっていた。表3に工程別作業工数割合の比較を示す。表中の1は当システム開発における工数割合であり，2～5は表の下に書かれている資料から採用したデータで比較のため書いてあるが2の場合は事務処理分野における調査データでプログラム開発に49%も割振られている。4は効果的プログラム開発技法を使用した場合に合理的なテストを行い，その分設計段階に重点を置く最近の傾向を示している。5はプロセス制御システム開発の25例の平均値を示す。以上の代表的な工数割合の資料と比較すると当システム開発においては特にテスト工程，つまり最小単位のプログラムモジュール間のインターフェスをテストする結合テストと実運営にきわめて近い環境で全システムのテストを行う総合システムテストに重点を置いてエラー発生を極力減少すべき努力がなされた。

表3　システム開発の工程別作業工数割合　（%）

工　程	1	2	3	4	5
基本設計	7.8	7	4	26	10.3
詳細設計	20.8	16	6	25	16.6
プログラム開発	30.3	49	43	30	41.2
テスト	41.9	28	39	19	24.8

1：当システム開発データ
2：ソフトウエア産業振興協会資料
3, 4：関東IBM研究会調査
5：ソフトウエア開発ハンドブック

III-3. システム開発の生産性

図2から明らかの様に開発工程の中ほどに多くの作業工数を必要とする。この部分の生産性向上の目的のため高水準言語をできるだけ多く使用する計画で開発が進められた。昭和40年代から始まった1次オンライン化には主にアセンブリー言語が使用されていたが，ハードウェアの能力向上と処理速度の向上により，しだいに高水準言語が実用上使用できるようになり，50年代から2次オンライン化ではCOBOL等の言語が多く使用され今日に至っている。60年代に向けて第3次オンライン計画が，現在，都市銀行，長信銀，信託銀行で進められているが非常に大規模なソフトウェアシステムの開発が必要であ

表4　システム開発の生産性（1日当りのステップ数）

	PL/I	アセンブリ語
コーディング量	146	342
デバッグ量（単体テスト）	48	111
プログラム開発量	31	53

り，1万人・月程度の労力を投入し500万ステップのプログラムが必要と言われている。開発したソフトウェアの保守作業にも多くの要員を要し，開発の70%以上のプログラマーが必要であろう。生産性向上のため効果のある手段として高水準言語の採用の比重が増大する傾向にありPL/I言語の使用による効率の測定を行った。表4に結果を示す。プログラムの開発量はPL/Iで31ステップ/日，アセンブリー語では53ステップ/日であった。但し，高水準言語であるPL/Iは，アセンブリー語の3～5ステップの機能を表現することができるので，4ステップと考えると，124ステップのアセンブリー語に相当することになる。従来，処理効率の点からアセンブリー言語を使用しているが，特にオンライン処理系では機能が複雑で，迅速な処理を要求するので重宝されているが，ソフトウェア保守の面から，又開発生産性向上からも高水準言語を使用すべきであろう。そして，プログラム開発工程におけるフローチャート作成工数が，PL/Iの場合に，13.7%短縮できることがわかった。しかし，システム開発の生産性は開発要員個人の能力に大きく影響し今回，PL/I言語の経験年数が1年未満のプログラマーは平均20ステップ/日程度であり，経験が1年以上の場合平均34ステップ/日のプログラム開発量をこなしている。

III-4. テスト工程とエラー検出

適用業務システムの品質を保証し，業務に必要な機能を検査する工程がテストである。業務で使用するオブジェクト・プログラムは前述の設計工程，プログラム開発工程で作成されるが，人間の思考論理により，そして，情報を分割変換する過程で思い違いや思考不足等により不良（バグ）が混入してしまう。それは，プログラムをコンピュータに入れ実行させるとエラーとなり期待する処理機能を満足にはたさない。開発工程の最後に，バグを検出し修正するテスト工程があり，デバッグ作業が行われる。それは単体テスト，結合テスト，総合テストから構成されるが，モジュール化された単体プログラムのデバッグはプログラム開発工程に含まれ，フローチャート作成時やコーディング作業時のバグを検出し修正する。結合テストではモジュール・プログラムを組合わせて相互のモジュールのインターフェスの整合をテストし，詳細設計時のバグを検出，修正する。総合テストは運用時のデータによる実動機能のテストを行い，基本設

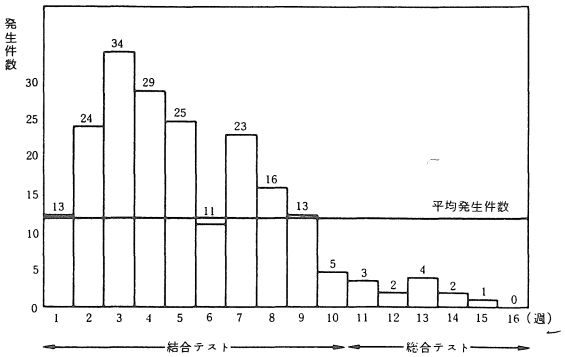

図3　テスト工程のエラー検出状況

計で組込まれた機能のテストが行われる。オフライン処理系の20本のプログラムに対し，各々につき100項目から1,200項目のチェックリストに従いデバッグ作業が行われた。単体テストには5,550項目，結合テストには2,100項目，総合テストには377項目のチェックリストに対してテストが実施された。全般的には8.5件/Kステップから13件/Kステップのバグが検出された。結合テストと総合テスト工程におけるバグの検出状況を図3に示す。結合テストは，2段階に分けられ，結合テスト1（1～6週）は正常項目のテストで結合テスト2（7～11週）は異常項目，限界項目，特異項目等の特例項目についてテストしている。各テスト作業で検出されたバグ1件当りの平均テスト件数は，結合テスト1で8.8件，結合テスト2で10.9件，総合テスト37.4件であった。テスト工程（結合と総合テスト）全体で検出されたバグは図4に示す様に，設計不良（機能不備，基本設計不良）を原因にする件数が41.9%であり，設計工程の管理が非常に重要であることがわかる。次にプログラム開発工程（フローチャート作成，コーディング）に混入するバグが36.6%検出されたが，本来は前工程の単体テスト時に取り除かれるべきバグがテスト工程にまで残っていたことになる。単体テスト作業方法の改善が必要であることになる。テスト中に検出されたバグを修正したがその結果，新しいバグを作り込んでしまうデグレードが13.7%も検出されてプログラムのデバッグ作業の困難性が現れたものと考える。開発に使用した言語間で比較すると，アセンブリー語を使用した場合フローチャート作成の不良が28.7%でPL/Iでは12.4%である開発の生産性と共に複雑で詳細に記述したフローチャート作成に伴ないバグが混入するためであろう。コーディング不良がアセンブリー語のばあい4%程度多く高水準言語使用によってプログラミング時のバグ発生が押えられることになろう。PL/Iによるプログラムのバグ発生率は，結合テストで12件/Kステッ

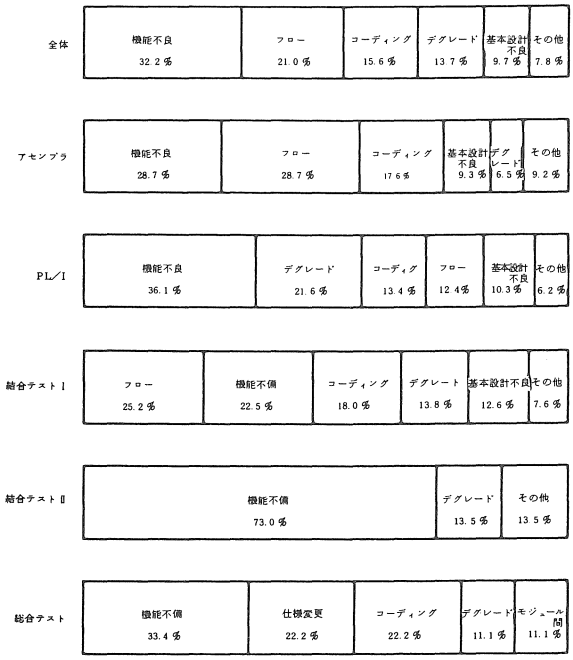

図4　エラー発生原因

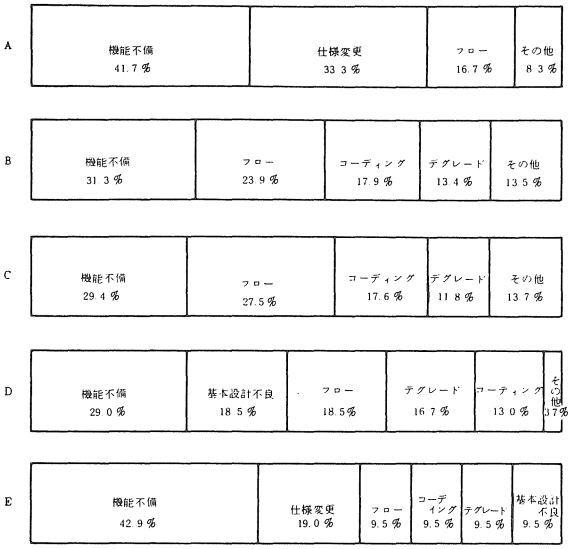

図5　システム設計者別エラー発生原因

プ，総合テストでは0.6件/Kステップ，アセンブリー語の場合，それぞれ8件/Kステップ，0.4件/Kステップであった。しかし機能から見るとアセンブリー語の場合PL/Iの4分の1のステップ数に相当するので，それぞれ32件/Kステップ，1.6ステップ/Kステップに相当すると考えればPL/I，エラーに対しても有効なプログラミング言語であることは明白であろう。次に結合テスト2で，機能不備（詳細設計不良）が，73%も出現しているが，異例データの処理に対応する設計方法の不備が原因で，異常時の予想が困難であった。システム設計者別のエラー発生原因をまとめて図5に示す。全設計者について詳細設計工程の機能不備が目立ち29%～42.9%のバグを作っている。設計者AとEの図から使用変更によるプログラム修正が33.3%と19%あるが設計後に業務部門や日銀，大蔵省から報告様式の変更が持ち込まれた結果でありバグと同様に集計している。フローチャートとコーディング不良のバグが次に多く出現しているが，設計者の仕様がプログラマーに充分理解できない結果によるものと考えられる。

IV. まとめ

企業における適用業務処理システムのソフトウェア開発の効率化は，増々重要な解決すべき課題であろう。今回，特に高い信頼性を要求する銀行システムの開発時におけるエラー発生の原因を究明するため，開発工程別又使用言語による差異等について考察を試みた。当システム開発では構造化プログラミング等の手法を十分に使用していないが，従来の開発よりも多くの工数を投入してテストがなされている。ソフトウェア危機が言われて十数年がたつがシステム開発の生産性に大きく影響するソフトウェアのエラー原因（バグ）を合理的に減ずる手法又はシステム開発の管理手法の早期確立をしなければならないと考える。

参考文献

1）Glenford J. Myers, 有沢　誠訳，ソフトウェアの信頼性，近代科学社，1977
2）Chin-Kuei Cho，後藤公雄監訳，高品質ソフトウェア，近代科学社，1982
3）国友義久，効果的プログラム開発技法，近代科学社，1983
4）国友義久，プログラム開発管理，オーム社，1982
5）コンピュータアプリケーションズ編，ソフトウェア開発ハンドブック，オーム社，1979
6）F. P. Brooks, Jr.，山内正也訳，ソフトウェア開発の神話，企画センター，1980
7．菅野文友，プログラミングの生産・管理，昭晃堂，1984
8）菅野文友，ソフトウェア・エンジニアリング，日科技連出版社，1983

（受理　昭和60年1月30日）