2005－SE－147（12）
2005／3／18

ソフトウエアプロセス定量化モデルの提案

河野　善彌†，　陳　慧‡

†奈良先端科学技術大学院大学 SOHO：〒251-0875 藤沢市本藤沢 2-13-5

‡国士舘大学　情報科学センタ 〒154-8515 東京都世田谷区世田谷 4-28-1

E-mail: koono@vesta.ocn.ne.jp, chen@kokushikan.ac.jp

あらまし　この報告はソフトウエアプロセスの定量的モデルの提案である．初めに工程の線形性，プロセスの展開 / 統合が可能なことを実績で検証する．その基本単位，作業と共に誤りを作り込む「純設計」と誤りを定率減衰させる「テスト類」につき，実績を簡単な構造モデルで近似して，その外部特性を説明する．以上を統合すれば，全体が得られる．定量的で全体から細部迄良く説明できる有用な基礎方式である．

キーワード　工程，プロセス，ソフトウエアプロセス，経営工学，Industrial Engineering

A Software Process Featuring Quantitative Model

Zenya Koono † and Hui Chen ‡

† Nara Institute of Science and Technology, SOHO: Honfujisawa 2-12-5, Fujiswa, 251-0875, Kanagawa, Japan

‡ Center for Information Science, Kokushikan University, 4-21-1 Setagaya, Setagaya, 154-8515, Japan

E-mail: koono@vesta.ocn.ne.jp and chen@kokushikan.ac.jp

Abstract　This report proposes a quantitative model of Software Process. As the basis, linear nature of process, which enables decomposition as well integration of a process, is verified by filed data. External quantitative characteristics of both design (error building-in) and test (error attenuation) are explained by simple structural model derived from field data. All is simple, versatile and useful fundamental theory for actual applications.

Keywords　Process, Software Process, and Industrial Engineering

１．はじめに

この報告は，ソフトウエア開発過程を定量的合理的科学的に取扱可能にすることを目的として，ソフトウエアプロセスの定量化モデルを提案する．ソフトウエアプロセスについて各種の研究が行われているが，未だ定説に至らない．定量的取扱はBoehm[1]やJones[2]等の研究結果があるが，現場で使うには隔靴掻痒の感を免れない．

定量化尺度は通商や課税の目的から始まり，人の社会の歴史と同じく位古い．生産の為に各種の尺度を用いることが拡大し始めたのは，18 世紀に産業革命が始まる時期であろうから，約300年の歴史と思われる．しかし，ヒトの作業の定量化や計測は，19 世紀末にF. W. Taylorがストップウオッチで作業時間を計測する原価低減法を編出してからで，既に約100 年の歴史がある．

彼は，工程の開始前状態と終了後状態を同一にすれば，作業の手続きを変えうることに着目した．ある作業を構成する下位作業の方法を変えて各工数を計測し，総工数がより少ない下位作業群を決めた．彼は科学的管理を標榜し，以後多くの人々の努力により，生産の合理化から管理の広汎な研究が進展した．これら諸技術や学問はヒトの作業過程（工程）を専門とする技術や学問で，総括してIndustrial Engineering, IE と呼ばれる．

他に遅れて1960年代以降日本では特にIEが発展し，Total Quality Management, TQM やトヨタのJust-in-time, KAIZENN等の原点になった．経営に貢献するところが大きい為であろうか，公式邦名は「経営工学」と云う．

Taylorの原価低減法は事業利益を向上させるから，適用が普及し拡大して，20 世紀初頭にはFordの流れ作業や(Taylorが学会で公表した当時は社会的反発を招いた)出来高払は社会に普及し定着した．IE から発展した

TQM 等は日本産業の興隆の基となった．定量化による原因と結果の認識や,その応用である品質向上や原価低減は，大きな効果を社会 / 産業 / 個人に及ぼす.

Taylorは工程の線形性による展開/統合を経験的な前提として出発した. 筆者らは，ヒトの設計の内部構造から工程の線形性等を証明する報告[3,4]をした．ここでは経験的ソフトウエア工学の立場から線形性を実績検証し（2 章），誤りを作込む設計（3 章）と誤りを減衰させる机上チェックとテスト（4 章)の外部特性を説明して，統合等（5 章）を説明する．本報告はこのような定量化した工程の取扱の提案である.

２． 工程の線形性

初めに，ヒトの作業の特性を調べる．IE では，ある作業の両端末を指定して，それらに挟まれたヒトの行動を含む作業を「工程」と名付ける．また,「工程」の内（例えば実現技術）には立入らず，工程の「外部特性」を計測し評価する態度であり，それが汎用性を担保する．先達に倣って，本報告でもこれに従う.

初めに工程の線形性に取組む. 以下の2種の外部特性値に着目する.

原価の主体を構成する工数,

品質を害する（あるべキでない）全てを「誤り」

関係を調べる為,以下の考えで基本的な工数と誤りの実績資料を求めた．その結果を図１に示す.

ヒトの特性を反映するよう，ナイーブで,

広汎かつ各種かつ多数で，偏りがない,

出来れば数桁の範囲の開発の実績資料.

現在では各社とも実績資料は公開厳禁である. 使えても流用の影響が大きく，当該社固有の影響が強すぎて，汎用的な利用価値が無い. そこで，初期的な開発のトラブルが収まった時期の公開資料を用いた.

工数関係では，Boehm の COCOMO のデータ [1] を図 1. a に，吉田氏(富士通)資料[5]を図 1. b に，また誤り関係では Thayers の第３プロジェクト [6] を図 1. c に，原文図から読取り再プロットした. 図の横軸はソフトウエアの規模，縦軸は図 1. a，b では(設計とテストを含む開発)工数，図 1. c では誤り数を取り，両対数用紙を使った．各点はプロジェクトに対応するが，図 1. c のみ 1 プロジェクトの各(プログラムの規模, 誤り数)の資料である. 誤り数の明確な基準が原資料に無い為，テスト摘出と推定する. これは絶対値の低下とバラつきの増大を齎すが，結論には影響しないので，後で詳細に説明する.

資料を打点した所，全体として規模∝工数の傾向が見えた．そこで，更に以下の後処理を行った.

・傾向線として規模∝工数 （誤り数 ）に対応する勾配 =1の太直線を引いて打点の集中部を貫く. 実験整理法に従い異常値を除く. 中央傾向線の上下に等垂直距離に細実線で傾向線を引き, 帯状領域の外の上下同数の打点を除外する．これらの条件を満たすように，太傾向線の Y 軸上の切片と帯状領域の幅とを調整した.

・帯状領域を等しい垂直距離の斜め帯の 5 領域に分かち, 各区分領域内の打点数を各右端の棒グラフに示した．何れもほぼ釣鐘状で，対数正規分布状である．範囲の上限は中央の N 倍から下限は 1/N 倍である．工数は N=4 で，中央値の 1/4 から 4 倍（最大最小比は 16 倍），また誤り数では N=5 で，中央値の 1/5 から 5 倍（最大最小比は 25 倍）と，極めて広い範囲にバラつく.

図 1 を見ると全体として Y ∝ (X)^1 の傾向（即ち指数＝1）に沿っている．意地悪く見れば，図 1. a の打点の全体は勾配=1より大きな勾配で分布するようにも見え，また，図 1. b では勾配 =1 より小さな勾配で分布するようにも見える．更に，図 1. c ではその気配も無いから，これらは無視する．以下を粗い近似とする[7,3].

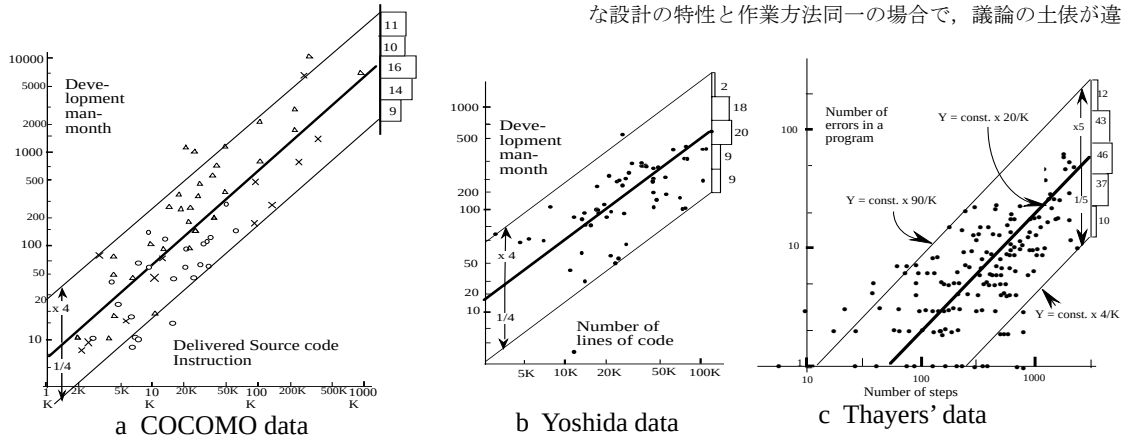
・開発過程 / 工程は線形系である.

・全体を各構成工程に展開して計画し，各工程毎の計測結果から全体的な評価が行える.

・工数は規模に比例する（生産性一定）. 注

・誤り数は規模に比例する（誤作込み率一定）.

注　Brooksはソフトウエア規模が増すと生産性が低下すると指摘した[19]. 彼は規模増大と共に作り方が低効率に移行する為と実態を説明している. 本報告でいう生産性一定とは，巨視的な設計の特性と作業方法同一の場合で，議論の土俵が違う.



a COCOMO data　　b Yoshida data　　c Thayers' data

図 1 開発工数と誤り数の規模特性

図１から，工数と誤り数，これらから誘導される工数率（生産性）と誤り作り込み率は何れも対数正規分布状である．人間信頼性工学(Human Reliability Engineering, HRE)は，原子力発電所等の巨大系の信頼度等を評価する[8, 9]．その為に，ヒトの動作の誤り率(Human Error Probability, HEP)を求め，系を評価する．そのデータブックでは，殆どのHEPは幅N=3を付けて表示してある．HREの塩見はヒトのKB操作での誤り率を精細に研究し，分布形状から対数正規分布であること，また工数にも同特性があることを報告している[10]．

1/N倍からN倍にバラつく対数正規分布を直線尺度で表示すれば，初めに立上ってピークがあり，以後はなだらかな減衰を示す．前記以外に機能同一なプログラムが同様な分布を示す．ソフトウエア工学の分野では，この種の分布をRayleigh分布と呼ぶ．数式上の違いはあるが，両者の形状はほぼ同じで呼び名の相違に過ぎない．

無統制でヒトが作業する時の外部特性値は，対数正規分布状で大きなバラつきを示す．これはハードウエア製造現場でも知られており，ソフトウエアのみバラ付きが多い訳ではない．統制を強化する（ある作業方法に揃える）につれ，分布は正規分布状に近付く[4]．

平均値は統制の技術と工程を構成するヒトの能力に依存する．(物理の定数のようなものではない．)

ハードウエア製造では繰返しが基本なので，早くからバラつきを克服する工程の特徴に気付いた．図２[4]は，バラつきを抑える技術の概念図である．図の左は階層的な工程，対応する右の弦の振動様の図はバラつきの様相を示す．最上位の設計工程では，最左端と最右端のみが統制されるに過ぎず，バラつきは最大である．以降，階層的な工程の統制が降ると，（知的変換の距離をM分断すると考えると，拡散力一定なら）バラつき幅は1/Mに減少する．階層を降り続けるとバラつきは激減する．ソフトウエアでこれを実現するには，

工程／設計情報の流れを工程の断面で定義

して統制する．図の例では，工程をデータフロー図，フローチャート，ソースコード等で区切り，その記述基準を定める．または，設計文書の記述項目を増し，記述水準を向上させる．これらにより図2のように統制を強化

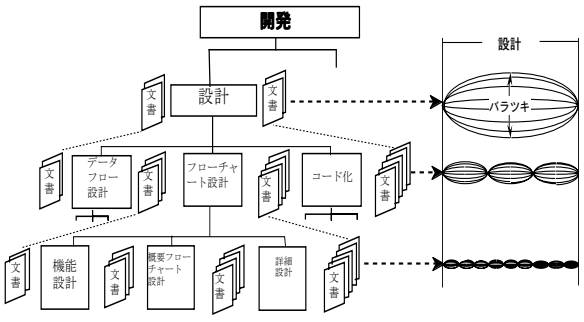


図2 階層的工程による統制の強化

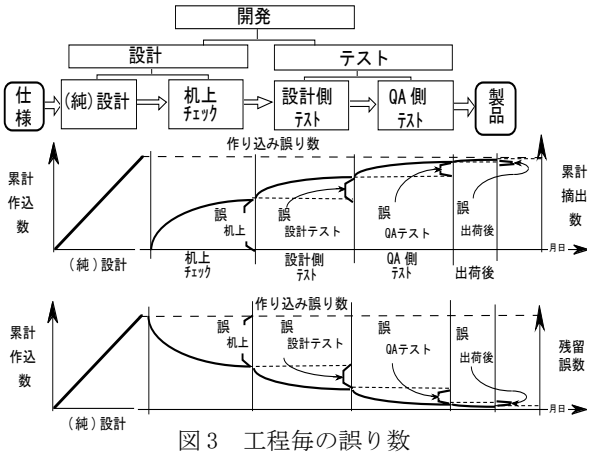


図３　工程毎の誤り数

できてバラつきが減る．

統制を強化しても人が守らない／守れないなら効果は挙がらない．ルールを守れる条件を整えればバラつきも減り，計測の不備が目立つから方法が改善される．

ヒトとルール／仕掛けと計測が歩調を合わせ進歩(バラつきが減り品質／効率も向上)する．

3．設計

ソフトウエアの作業には殆どヒトが関わり工数が消費される．この工数の消費で経営資源の消費を代表させる．原理的な工程には以下の２種類がある．

・純設計：設計入力情報を出力に変換する機能のみにする．このヒトの作業は誤りを免れない．

・机上チェックとテスト：テストとは試験条件を予め定め，主として機械を併用するもの．机上チェックとは機械的手段によらずに，頭脳作業で誤りを摘出する機能（これらのヒトの作業もまた誤りを免れない．）

純設計の重要な外部特性の第１は，１次的な出力（成果物）数と工数等の資源消費，および２次的な（出力数／工数）で定義される生産性である．

第２の重要な外部特性，誤りについて，図３により検討する．図は純設計と後続する工程群の流れ図である．工程毎に対応させた下部のグラフの横軸は作業月日の経過を示し，縦軸は誤り数である．簡単の為に，一定人数で作業する場合を想定する．

純設計の場合には日時の経過と共に出力が直線的に生じるが，その出力にはある率の誤りが避けられない．図のグラフのように誤り数も増大していく．作り込まれた誤りは，後続する机上チェック，テスト（設計側とQuality Assurance，品質保証／検査）で摘出され，残余は製品と共に出荷されるが，これは通常6ヶ月以内には殆どが摘出され，以後は稀に生じる程度になる．摘出誤りは以下の関係がある．

作り込まれた誤り数

＝机上摘出誤り数＋設計テスト摘出誤り数

＋**QA**テスト摘出誤り数＋出荷後摘出誤り数

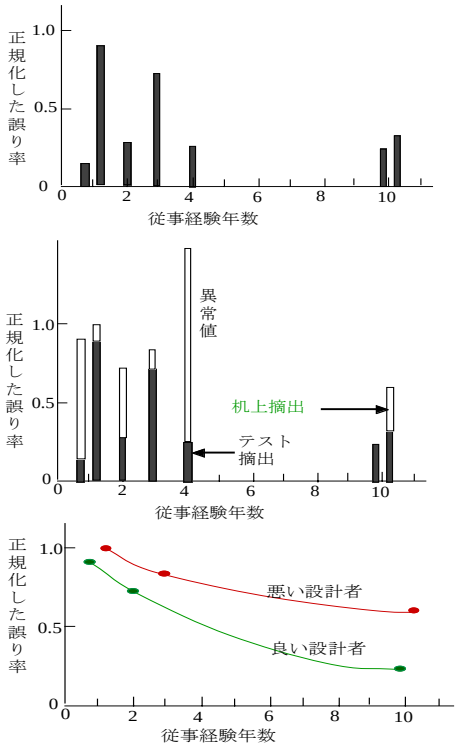


図4　誤り率と経験年数の関係

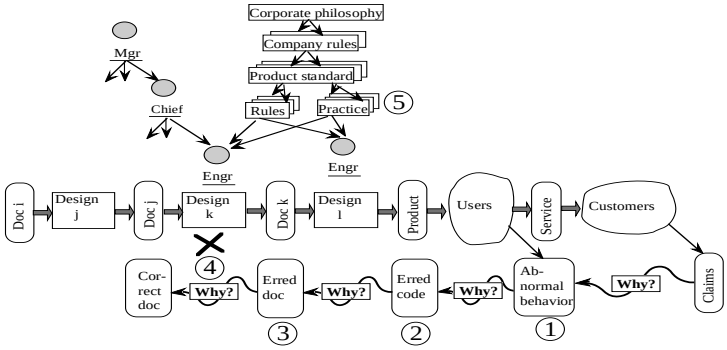


図5 誤り作り込み工程の追跡

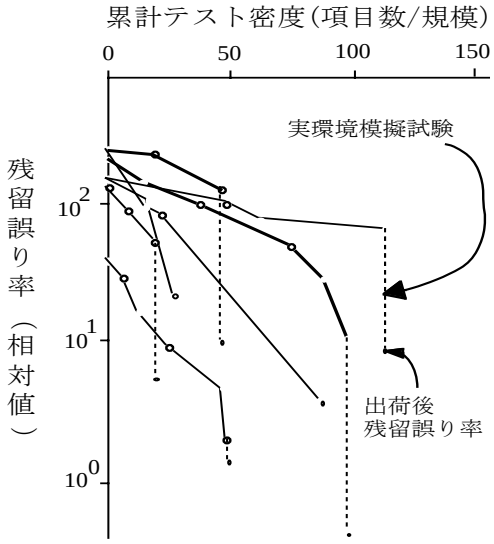


図6　累計テスト密度と残留誤り率

優秀な組織や人は机上チェックで誤りの80%を事前摘出するが，殆ど摘出しない人達も居る．後続する一連のテストでは，日本企業では誤りの90%以上が摘出される．

殆どのプロジェクトでは，最終段階あるいは一部の先行テストでの誤り摘出の累計カーブを重要な管理指標としている．誤りの検討資料にもこれを用いる例が多いが，それでは机上チェック摘出を無視している．机上摘出の率を(最高)80～(最低)0%とすると，観測値は作り込み数（0.6±0.4)のバラつきが生じて，何らの因果関係の系統性が見えなくなる．図4[11]はその1例で，横軸は経験年数，縦軸は正規化した誤り率である．上端図の棒グラフはテスト摘出の誤り率で，系統性は認め難い．下端図は作り込み誤り率の傾向線を示し，傾向が明瞭に読取れる．これは中間図に示したように，机上摘出を加えたものである．差異の様相は明白である．

作り込み誤り数を求めるには，設計作業を一渡り終えた後，一渡りの簡単なチェックを済ませる．当該部分の「設計終結」を宣言し，以後全ての摘出誤り（シンタックスエラーを除くことが多い）を記録に留める．後続工程で摘出した誤りは，その作り込み工程を識別し，作り込み工程毎に集計する．図5[4]は作り込み工程を識別する原理を示す．図の中央は開発工程である．設計工程以降で誤りが見つかると，クレーム→異常動作①→誤り箇所②と「何故？」を繰返して，誤り箇所を知って修正する．ここで留まらず更に上流に遡り「工程Xの出力文書には誤り又はその誘発要因がある③が，Xへの入力文書にはその痕跡が無い」工程を作り込み工程X④とする．

このように追跡すると，仕様以降の段階から全ての誤りが捕捉でき，プロジェクト終了後に集計できる．純設計の重要な外部特性の第2は，作り込み誤り数と成果物数で除した誤り作り込み率である．

2章で説明した大きなバラつきと階層的工程による統制，誤りの正確な計数により，「工数とは当てにならない数値」，「神秘の人月」とか，「誤りの因果関係が見えない」，「ソフトウェアの謎や神話」などの誤解は消え去る．定量化は科学的な進歩の第1歩である．

4．机上チェックとテスト

図3中央のグラフは，机上チェック以降に摘出される誤りの累計カーブであった．カーブの最右の上端は作り込み誤り数になる．このカーブを上下反転させると，全体は作り込み誤り数から出発する負の指数的な減衰状であり，また，個々のカーブもまた同様である．これは

机上チェックやテストは誤り減衰器の縦列接続

であることを意味する．

テスト摘出誤り数の累計曲線として非正常的な曲線が誤り数予測の論文類に紹介される．これは予測能力を調べる為である．テストでも机上チェックでも，理想形は負の指数的減衰状である[机上チェック報告例12]．

図3の設計側テストではホワイトボックステスト,QA側テストではブラックボックステストにより，1連のテストをした時(テスト項目数は両者ほぼ同数),夫々が誤りを約1/10に減衰することが河野等により観測され[13],これに基づき渡辺[14]が同一事業所内の一製品系列(１部署)に限った精密な調査を行ない,バラつきが少ない同程度の減衰を確認した.（これから工程のバラつきが減衰特性のバラつきを作りだすことが判明した.）

テストの特性を司るメカニズムを説明する.図6[15]は,テストの誤り減衰特性を各種プロジェクトについて調査した結果である.図の横軸はテストの正規化指標値としてテスト密度（テスト項目数/規模）を取り，縦軸は各点での残留誤り率を正規化して示す.

各カーブの最右下の点の座標は,(X: 総テスト密度,Y:最終テストが終了した出荷時点の残留誤り率). 次に左の座標は（X: 総テスト密度から最終テストを差引いた累計テスト密度，Y: 前記残留誤りに最終テストで摘出した誤り数を加えた残留誤り率），以後同操作を経て，最左端である縦軸状の切片の座標は，（累計テスト密度=0，Y: テスト開始時の残留誤り率）である.なお,右端の垂線部分は,実環境模擬テストの単一項目名で多数の誤りを摘出したテストの資料である.このテストを各項目に分割して管理した以降には，垂線は消えた.

カーブは全体として右下りで，各テストも右下りであり，定率の誤り減衰を示している.カーブの勾配（単位テスト密度当りの誤り率の減衰量）を「テストの有効度」と名付け，テストの減衰量の外部特性値とする.これを用いれば,

残留誤り率を許容水準以下にするに
必要なテスト数を設計でき,

更に残留誤り率の全体設計もできる(後続５章で説明).

テストの有効度は,ある工程の外部特性値であり,客観的な定数ではない.図に見られるように「テストの有効度」は，大きなバラつきを示す.これは前記のように各テスト工程の特性のバラつきの反映であり,工程や誤りやテスト項目の定義が揃えばバラつきは減る.設計での謂生産性のように,誤りを減衰させるテストでは「テストの有効度」は重要な指標値である.

これまで可/不可の視点で誤りの取扱を単純化した.この誤り総数は,古典的な論理誤りから可搬性等の拡張された品質の誤り迄の全てを含んでいる.ある拡張された品質基準での誤り数とかシステムダウン数等は,総数の内訳として現れる.この内訳の比率はシステム毎に違う.

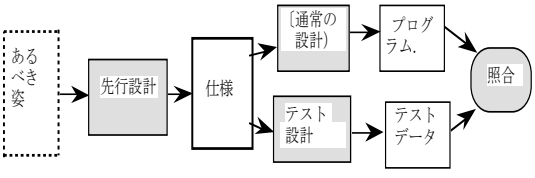


図７ テストの構造

テストは,何故定率の誤り減衰特性を示すのか？「テストの有効度」は如何にしたら大きくできるのか？図７により，これらを説明する.図は通常の設計とテストに関係する諸要因を示したデータフロー図である.右部分を見ればテストとは通常の設計とテスト用設計等との突合せに過ぎない.通常の設計がヒトの誤りを免れないと同様にテスト設計等の作業もまた誤りを免れない.

検査技術や統計学では良品を不良と誤る確率を第Ｉ種の誤り，不良品を良と誤る確率を第II種の誤りと云う.ソフトウエアでは，第Ｉ種の誤りはテスト後の見直し等で救済されることが多いので，第II種の誤りが問題である.この定義から下記が判る.

テスト結果は，元の残留誤り率×第II種の誤り率の誤りを含む.減衰は第II種の誤り率に等しい.

図７を参照すれば，知的処理であるテスト設計以外に,テストデータのコーデイング，照合確認，直接的に準拠した仕様類,更には本来的なあるべきところから出発した仕様類との食違い等の人が誤る作業が多数ある.これらが(定量的な関係は未詳ではあっても)第II種の誤りに結びつき，「テストの有効度」を支配している.

図６の資料は1970年代末から1980年代中期迄の資料である.カーブは低勾配群と高勾配群とに分れている.前者はアセンブラ時代のもの,後者はコンパイラ言語,構造化設計と対応する文書を使用するもので,スパゲッティ構造以外に上位の仕様記述の不完全が多かった.

時代と共に要求水準が向上する.1990年代には高品質組織の総テスト密度は500項目/k行を超えている.

テスト密度を増やす以外に，机上チェックを強化してテスト開始時の残留誤り率を下げることが,効率および教育の両面から有利である.システムテストで摘出される誤りは，殆どが「机上チェック」,「後続各テスト」から漏れ出したものである.この段階では,テストの設計も確認作業も複雑であり，また，摘出誤りの切分と修正にも高い技術を要するから,作業効率が悪い.単体テスト次いで統合テストで誤り摘出件数を稼ぐ方が良い.

図8[16]でテストと工数の関係を説明する.図は，あるシステムの全面新開発の資料で,各記号はサブシステムを示す.横軸はテスト摘出誤り数を各規模で除して正規化したテスト摘出誤り率,縦軸は各テスト工数を各規模で除した正規化した工数の値である.各プロットを貫く傾向線が現われ，Y軸の切片は正常なテスト１式の工数，その上部は誤りの識別/修復/再確認等の損失工数

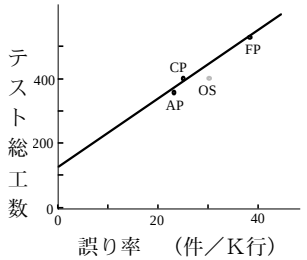


図８ 中程度品質の誤り率，工数図

で，傾向線の勾配は誤り当りの損失工数である．図のように，誤りの為に多大のコストを費やしていることが明白である．そこで，純設計に続く机上チェックで誤りを除去する方が有利である．

（初級者など）誤り率が高い場合には，テスト総工数／誤り数から誤り当り修正工数が求まる[16]．また，誤り率が極めて少ない場合は，テスト総工数/テスト項目数からテスト当り工数を求められる[16]．

以上を総合して，テストの工数は以下で表される．

テスト総工数／規模
= テスト当り正味テスト工数×テスト密度
+ 誤り当り損失工数×テスト摘出誤り率

新たにテスト密度を増加させる時や，工数配分を再編成する時には，この関係を用いれば定量的に計画できる．

次に机上チェックを検討する．中～高校教育では下記を教育している．

数学物理系の答案は，小さな段階毎に変形を明示し，段階毎に必ず念入りにチェックせよ．

この他に，経験的に小さなステップ毎の作業とチェックの確実性を承知している人も多い．この経験則は，以下の２様に解釈できる．（但し，誤りはランダムとする）

1．ある設計での誤り率をEdとする．既に記したように，これに対する机上チェックは設計工数に比例的な工数を要し，これは誤りを免れない．この第II種の誤り率をEcとする．設計終了後チェックすれば結果の誤り率はEd･Ecとなり，Ec倍に低下する．今設計を等しい知的処理数のM段階に分かつ．対応チェックもM区分されよう．小段階毎にチェックすれば，各個の誤り率は(Ed/M)･(Ec/M)＝(Ed･Ec)/M²で，M倍した全体は(Ed･Ec)/Mになり，1/Mに減る．

2．細分化するほど，入出力間の知的変換距離が小さくなり，ヒトの認識は容易になる．そこで，小区分にする程認識が容易になるから，誤り率が低下する．

机上チェックは小段階毎に行う方が数倍効率的．

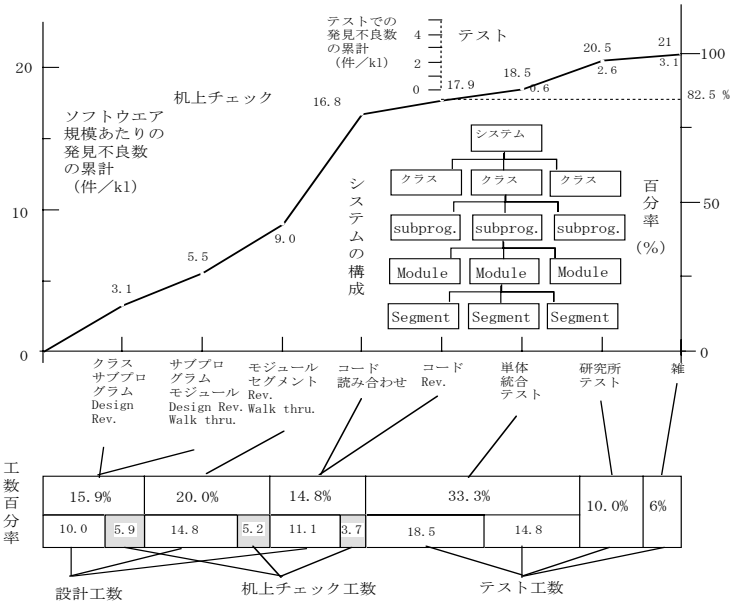


図9 GTE社の優れた開発例の記録

机上チェックの効果的な実行例を図9[17および訪問時戴いた同社資料より筆者の責任で再現]に示す．これは1970年代末の米GTE社研究所の通信系の開発例である．グラフの下部に示した階層構造を持ち，設計がトップダウンに降る階層毎に各種資料が計測された．グラフの横軸はこの工程区分，縦軸は机上チェックで摘出した誤り累計数にテストでの摘出数を加えてある．下の棒グラフは全工数を工程毎に百分率で表したもので，設計は純設計を白地で，机上チェックをハッチして示した．

この開発の優れている所は，品質の良さに現れる．摘出した誤り率は21件／k行，その82.5%を机上チェックで摘出した．机上摘出率80%は実用上の上限値である．机上チェックの工数は棒グラフに示されており，純設計の60～30%に過ぎない．累計カーブから，コードリストのチェックだけでは，誤りは少ししか摘出できないことが見てとれる．各設計でのチェック方式は横軸に添えてある．純設計直後の机上チェックで殆どの誤りを摘出するから，各机上チェックは先行純設計のしたことを中心にチェックすれば良い．即ち，設計の上流から最下流迄，小さな段階毎に具体化しながら記録し，該段階毎の机上チェックの繰返しの連鎖があり，テストになる．

これは所謂 **heavy process** である．多数工程に分割されたから，バラつきが小さく，誤りが高い率で事前摘出され，更にテストで摘出される．このレベルの高い開発は，文書作成の負担により実現可能されている．

・non-heavy な process で同成績を挙げうるか？
・heavy と non-heavy の差は科学的に検討されたか？
　夫々の成果差は定量的に研究されているか？
　ユーザである産業の基準で得失が明確にされるか？

工学は合理性科学性を要し，定量化は第１段階である．

机上チェックの摘出と工数の関係を求める．図１０はこの為のモデルで，机上チェック工数／純設計工数を正規化工数Cとする．誤りの減衰は理想的な負の指数的減衰e^{-aC}とする．工数C_0を投じ机上で摘出した件数を最終的な作り込み数で除した机上摘出率から机上減衰率D_0をえる．

机上減衰率 D_0 ＝ 1－ 机上摘出率
＝ e^{-aC_0}

であるから，希望する机上減衰率D_iを実現

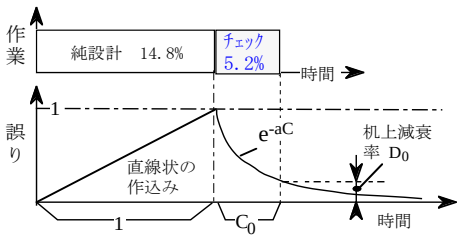


図10　机上チェックの誤り率と工数

する見込みの正規化工数C_iは次式で与えられる．

$$Ci = C_0 \cdot \mathrm{Ln}\ D_i / \mathrm{Ln}\ D_0$$

工数をC_0からC_iに増加させるとして，机上チェックの方式を求め，試行し評価して改善を積み，定着させる．

5．総合特性とその検討

線形性を用い展開した工程毎の評価を統合する．図9の誤り率関係を統合する例を図11で説明する．

図の左，設計の誤りは加算性なので，純設計毎に上向きブロック矢印で作り込みを，当該机上チェックに下向き矢印で摘出を示し，その差を各設計毎に前段の残留誤りに加えて行く（作り込みは摘出×1.0/0.825と近似，後続設計での摘出は無視）．図の右，テストでの誤り減衰量は対数尺度で加算性なので対数尺度を用いる．テスト工程入力の残留誤り率から始め，下向き矢印で各テストの減衰量を表し，下に伸ばす[18]．工程毎や全体の他に，サービス／外部機能／内部機能構造別，組織別や組織の階層別等を展望から詳細迄を表示できる．

実態を定量的に評価して可視化し，外部特性毎にその成立ちを知れば，評価が確実で精細になり，各種の改善が出来る．（例：出荷品質が不良ならテストを増す，品質向上には机上チェックを強化する方が効果的等．）ある構成部分が他より悪い／良いなら，その原因が探索できて，何が悪い／良いのか，如何に影響するか等を具体的に知ることが出来る．

> 図９のチームは，公開資料の諸提案を皆で議論する他に，開発毎に比較実験し定量評価して[17では単体テストの機械実行と人のシミュレーションを比較]皆で議論する．これで見解が統一され，皆に徹底され，工程が進歩する．

幾つか思考実験しよう．このチームが再度同種同程度規模の開発をすれば，どうなるか？先行実験している新技術が常にあるから結果は必ず向上する．（改善無くして進歩無し．集団を作業に投入する前に，問題を事前に研究し，必要なら評価実験をする事は工学の常道．）

仮に計画している改善や全員の向上努力を遮断できれば，ほぼ同様な結果になろう．この時，前回実績の各種指標と諸計画／実績に基づき，次回プロジェクトの定量的な詳細計画が作れて，実行できる．工程が進む度に，遅延日程，工数投入，設計文書など成果物量，実績生産性，机上摘出誤り数等を示せば，工程毎に計画との乖離が定量的に見える．更に工程内で計画／実績の推移カーブを見れば，より早く兆候が掴める．

Q. 現状から品質／生産性を向上するには？管理者が先頭に立って，明らかなムダムリムラを退治する．品質管理を勉強する．前記の鳥瞰から個人迄の品質関係資料を定量的に調べれば，問題の所在は判る．工程を改善するには，Total Quality Controlの再発防止を勉強し，（図５の上部に示す根本原因を断つ）再発防止策を取る．生産性は規模生産性と機能当り規模を同様に定量的に調べて悪い所良い所を絞出し，IEの基礎とValue Engineeringを勉強し，その他を含め工程を改善し続ける．

Q. 仮に，このチームに異領域の開発を無準備で行わせたらどうなるか？違いが大きくなるにつれ，バラつき始め，品質が低下し，効率も低下し始める．

Q’．更に乖離が大きい時には，如何にするであろうか？彼らは最初に，その領域の技術を勉強し，市場／使用者や運用／構成技術を調査し，机上設計し試作評価して構築技術を高め，成算を立ててから多数の人を掛けて製品を作るであろう．一旦作れば同種の受注があり，更に発展や展開があり，次第次第に事業が成長するからである．これは産業界の常道．

製品や領域毎に要となるポイントは違う．同じ製品でも攻める企業毎に戦略や戦術が違う．そのプロセスは各事業上のニーズに従い最適化されるべきもので，各種の様態があり得る．基礎段階を除き汎用的なBest practiceは存在しえない．基礎段階以降は，夫々のアクセント～価値観に基づき，夫々に最適化をしていく．品質や効率は，プロセスで決まる．プロセスを標準化するには，プログラム構造～機能構造が標準化されねばならず，それにはサービス／外部機能が標準化されねばならない．かような標準化を積重ねて上で，初めて自動化が

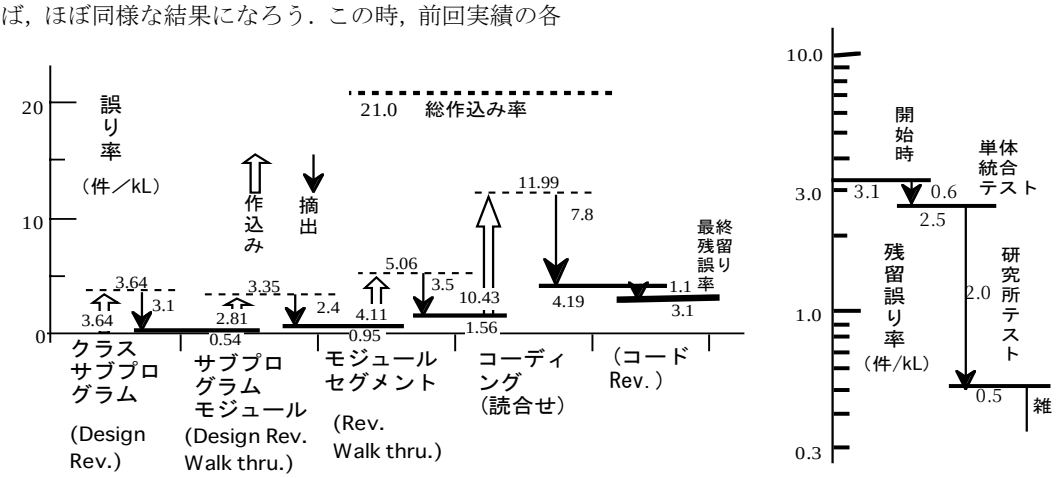


図11 残留誤り率レベル図

可能になり[20]，進展する．プロセスは，これらを可能にする，定量化を武器とした分割し統治する（Divides and conquers）管理手段である．

６．むすび

本報告は，ソフトウエアプロセスの定量的モデルを提案した．開発の現場で見られる基本的な様相を極力簡単な基本構造で近似し，定量化に結びつけた．この立脚点は以下にあるが，他とは色々と違う．

１．プロセスは線形系で展開や統合が可能である．階層構成を用いる．（COCONO との違い）
２．最終出力のみでなく設計情報/意図のモノの流れを基軸に取る．（多くのソフトウエアプロセスとの違い）
３．プロセスはモノと独立な手段．内部に立入らずありのままの特性を外部的に観る．（Osterweil との違い）
４．基本的な外部特性は，経営資源，成果物量，あるべきでない一切の誤り等少数に絞る．極力簡単な構造モデル（メカニズム）に置換える．（現場実態対応）

これらから出発して，下記を得た．

Ａ．ヒトの作業の特性と制御　対数正規分布状の分布と統制手段，誤りの正確な計数や効果的摘出法．
Ｂ．経営資源（工数等），成果物量，誤りを定量的に表現可能にし，ごく簡単なメカニズムを用いて成立ちを説明して制御可能にした．単純な定量表記は高度化しても利用可能で，簡単なメカニズムはその原因を提示して対策可能にする．以上から，事業上のニーズが何であっても，対象がビジネスプロセス，ハードウエア開発，ハードウエア製造の何れにも，レベルが低くても高くても，常に利用できる管理手段になる．
Ｃ．これらを通じて，ソフトウエアに於ても定量的合理的科学的な取組みが可能になる．

今後は以下の課題に取組みたい．

○．利用方法の明示と実地適用の中で手法の確立と効果を挙げ，提案方式を検証する．
○．非明示的な工程を使う開発過程の明確化と評価

基本技術のい一つ，習熟効果は紙面の制約上今回は省いた．次の工程の応用技術の中で，工程の進歩と共に，説明したい．現時点では[3，4]をご参照ください．

謝辞

著者河野は，日立製作所 （元）戸塚工場配属時にIEの初期教育で工程の階層展開を知って衝撃を受け，以後R&D～製造を通じて，作ることの要を学んだ．貴重な教育を賜った皆様にお礼申し上げます．これに加え，埼玉大学での10年間の研究が纏めあげる力になった．Far先生，Abolhassani さん始め，学生諸君に感謝します．

参考文献

[1] Boehm, B. W., Software engineering economics, Prentice Hall, (1981).
[2] Jones, C., Applied Software Measurement: Assurring Productivity and Quality, Mcgraw-Hill, (1996). 訳：鶴保征城，富野壽，ソフトウエア開発の定量化手法（生産性と品質の向上をめざして），共立出版，(1993)．
[3] 河野善彌， 陳慧，人の設計知識と定量評価（１／２），信学技報，Vol. 103, KBSE, (2004・3)
[4] 河野善彌， 陳慧，人の設計知識と定量評価（２／２），信学技報，Vol. 103, KBSE, (2004・3)
[5] 吉田征ほか，パネル討論会　ソフトウエアメトリックスの現状と課題：昭和58年前期第２６下位全国大会報告，情報処理，26，1，pp. 48-51，（1985）．
[6] Thayers, T. A., et al., Software reliability study, Final Techinical Report, RADC-TR-76-238, Rome Air Development Center, (1976).
[7] Koono Z., Chen H. and Far B. H, Expert's Knowledge Structure Explains Software Engineering, Joint Conference on Knowledge Based Software Engineering 1996, pp. 193-197, (1996).
[8] 林喜男，人間信頼性工学ー人間エラーの防止技術ー，海文堂，(1984)．
[9] 塩見弘，人間信頼性工学入門，日科技連，(1996)．
[10] Shiomi, H., On analysis and summarization of human reliability data for simple VDT operation., BICRMS 92, pp. 372-377, (1992).
[11] Koono, Z., Yamato, M. and Soga, M., Structural way of thinking as applied to high quality design, IEEE COMSOC International Conference on Communications 1988, pp. 8.2.1-7, (1988).
[12] 森本祥一，須藤紀久夫，陳慧，河野善彌，ソフトウエアクリエーション：統合知的CASEツールの図面処理，信学技報，KBSE80（2001-3），pp. 25-28，（2001）．
[13] 河野善弥，大坪東光，ソフトウエアの誤りと除去の評価（Industrial Software Engineering の立場から），情処研報SE95-5，pp. 31-38，（1993）．
[14] 渡辺順平，緒方秀夫，ソフトウエアの品質および生産性予測法の一事例について，第２回ソフトウエア生産における品質管理シンポジウム，(1982)．（菅野文友監修，ソフトウエア生産管理委員会，ソフトウエア品質管理事例集，」日科技連出版社，1990 に再掲）
[15] Koono, Z., Ashihara, K. and Soga, M., Structural way of thinking as applied to development, IEEE COMSOC Global Telecommunications Conference 1987, pp. 26.6.1-6,(1987).
[16] Koono, Z., Tsuji, H. and Soga, M., Structural way of thinking as applied to productivity, IEEE COMSOC International Conference on Communications 1990, pp. 204.2.1-7, (1990).
[17] Daly, E. B. and Mnichowicz, D. A., The management of large software development for stored program switching systems, International Switching Symoisium 1979,pp. 1287-1291, (1979).
[18] Koono, Z. and Far, B. H., Quantitativ design of development process, Fourth European Conference on Software Quality, pp. 173-181, (1995).
[19] Brooks, F. P., The mythical man-month: Essays on software engineering, Addison-Wiley Publishing, (1975). 改定版と訳本あり．
[20] 大野治，小室彦三，降旗由香里，渡部淳一．，多次元部品化方式によるソフトウエア開発の自動化ー自動生成系の開発とその評価ー，子信学会，論文誌J84-D-I No. 9，pp. 1372-1386，(2001)．