Vol.2010-SE-169 No.11
2010/7/23

# ソフトウエア開発工程の定量的特性

（ソフトウエアの謎は有るか？　銀の弾丸は無いか？）

河野善彌[†1]　陳慧[†2]

産物はそれを作る工程(プロセス)の結果である．この論文は各種のプロセス特性の実績例，その内部メカニズム等の総覧である．これらの本質はヒトの意図的行動～ヒトの知にあると考えている．現在のソフトウエアエンジニアリングについて，各種のコメントを付けている．

# Quantitative Characteristics of Software Process

**(Is There any Myth, Mystery or Anomaly? No Silver Bullet?)**

Zenya Koono[†] and Hui Chen[††]

A process creates a product. This paper reviews various samples of observed data, each internal structure to bring such results. The substantial structure behind these seems to be “human intentional activity” of human knowledge. Various comments are made on so-called software engineering.

## 1. はじめに

筆者らはソフトウエア開発工程の定量的特性を研究してきた．この論文の目的はそれらの見通しよい総覧にすることであり，ビジネスとの関わりを明らかにする．

計算機の利用が始まった初期，メインフレーマーには自社計算機用制御プログラムを開発する組織が作られた．初期のモニターは計算機の効果効用を増す手段 OS に変身し，計算機ハードウエアと同様に大型化していく．かくて，OS は自社の研究者と製品計画者に牽引され伸びた．計算機メーカは計算機を売り，これを動かすソフトウエアを添付した．更に要員も「システムエンジニア」の名で無償派遣した．計算機システムという未来の大物に育つ見込みのある子供の教育を託したとも云えよう．利用の高度化につれ，ユーザ側システム計画要員と「システムエンジニア」のペヤーは「システム企画計画部署」に成長し，日本のコンピュータシステムは第１次オンライン，第２次さらに第３次と巨大化した．

世界トップのリーダは巨人 IBM であった．IBM は後続する同業者を牽引する責任感を持っていた．同時期，通信の巨人(AT&T)ベル電話研究所は電磁技術で固まっていた通信業者とその製造業界を電子化しソフトウエア化する育成の責任感を持っていた．やがて電電公社(現 NTT)が交換機 DEX と計算機 DIPS を携えて，これらリーダの中に入る．1960 年以降は産労官の合意で，「生産性運動」が日本全国に展開された．日本のメインフレーマーは主たる通信機製造業者，主たる半導体製造業者であり，前記運動の中核メンバでもあり，一斉に工学と経営の新技術の洗礼を浴びた．特に日本のメインフレーマーは整備された Industrial Engineering, IE, 経営工学の力でハードウエア製造が一新されたので，それに倣って世界に先駆け「ソフトウエア工場」が発足した．

ソフト/ハードを分離するアンバンドリングの普及と共に，メインフレーマーは「製品企画計画部署」が OS 等の開発の牽引力になり，利用者側の「システム企画計画部署」が自社システム開発の牽引力になった．しかし，内作より外注が有利と考える経営者が増えた．受託開発企業が増える．新しいユーザが急増するが，自らシステム企画計画する力が無いから，受託開発企業に頼る．かくて，「自分達のものとしてソフトウエアを企画し作れる組織」と「受託ソフトウエア開発組織」に依存する組織の２種に分化した．**両者はビジネスモデルが異なるから，それぞれの工学は違う．**

本論文では，自由で競争的市場で営業するビジネスモデルを「通常（ビジネス）モデル」と名付ける．自社 OS 開発，組込みシステム，ゲーム，ツールやミドルウエア等のパッケージソフトウエア，ハードウエアとそれのソフトウエア等自由競争市場で営業するビジネスモデルでもある．

後者の第１は，ミドルソフトウエアや部品的プログラム，各種アプリケーションの母胎あるいは専用開発システムを構築しており，これを用いて受注ソフトウエアを開発するもので，VF 形（ビジネス）モデルと名付ける．後者の第 2 は VC 形（ビジネス）モデルと名付けるもので，前記の母胎や専用会自発システムを持たないものである．

[†] クリエーション プロジェクト，神奈川県　koono@vesta.ocn.ne.jp;　URL: http://www.creationproj.org
Creation Project, koono@vesta.ocn.ne.jp.　URL: http://www.creationproj.org

[††] 国士舘大学 情報科学センタ，世田谷区，東京都．
Center for Information Science, Kokushikan University, Setagaya, Tokyo.

Vol.2010-SE-169 No.11
2010/7/23

## 2. 最も基本的な規模対工数の特性

『共通』図１[1]は米空軍 RADC (Rome Air Development Center)での多数の実績資料を統計処理した Nelson 資料[2]に準拠した図である．図は Nelson 資料中の Fig. 2 のソフトウエア規模対工数の両対数尺度グラフから，統計処理で求めた中央傾向線を残しつつ，±1σの副傾向線を削除し±3σの副傾向線を追加した．Nelson は中央傾向線の勾配の指数部分である X 項は $x^{0.975873}$ と報告している．また，アセンブラ使用等の在来手法の Fig. 12 では X 項は $x^{0.995841}$，また構造化設計等の近代的な手法の Fig. 9 では X 項は $x^{1.045325}$ と報告している．Nelson 資料は 407 のプロジェクトデータ，自ら研究し開発することが当然であった初期，幅広い各種データ，を統計処理したもので，最も信頼度できると考える．そこで，指数部分である X 項は $x^{1.0}$ を採用する．

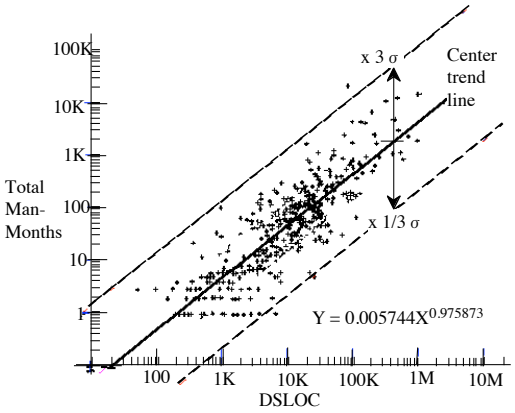


図 1. 規模対工数

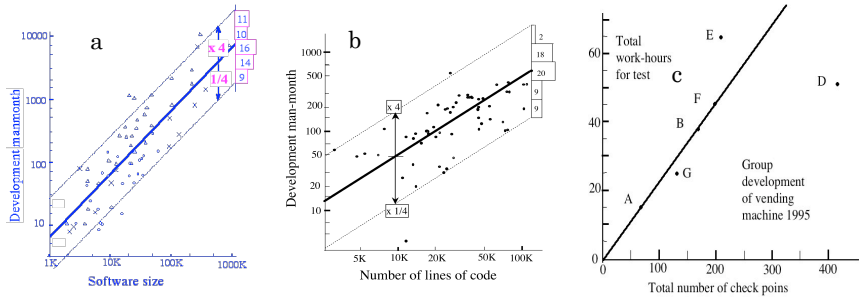


図 2. 規模対工数

筆者等は 1996 年に図２a，b を報告した．図 a[3]は COCOMO データ[4]の図の X，Y 軸を入替え，Boehm の３種の傾向線(後述)を除いたものである．図 2b[3]は富士通の吉田の報告[5]した直線尺度表示のグラフを両対数尺度図に再プロットした図である．両図のプロット群は帯状で，帯状領域の勾配は $x^1$ に非常に近い．帯状の中心に $x^1$ の中央傾向線，上下に等間隔で副傾向線を２本引き，プロットを帯状領域におさめた．

統計学は，「充分に多数の資料を得れば，統計処理結果は実体に近づく」ことを保証している．これに従えば，まずは中央傾向線で問題は処理でき，バラツキはバラツキで計測条件の問題になる．これを実地に確認する為に図 2c のバラツキの小さい例を調べて見る．図 2c の横軸はテスト総数，縦軸はテスト設計から確認終了迄の総工数であり，筆者等の学生演習「自販機の開発」[6]のある年度の実績で，規模対工数の一例である．図 2c の実績プロットで(チーム A，G，B，F) は傾向線に沿いバラツキは小さい．これは，３年生と技術レベルが揃っている．テストは簡単のもので，作成/記述/作業方法は規定してある．前記の揃ったチームはこれらに準拠した．効率が良い方にずれたチーム D は，発券側と収集側の２システムを開発したので，習熟効果が現れ全合計値ではより効率的に現れた．一方，チーム E は欠陥減少の意識が強すぎて，全作業の効率を下げてしまった．以上から特性値のバラツキはプロセスのバラツキに起因することが理解できる．

## 3. 工数の定量化

『共通』ソフトウエア開発工程は線形系であるから，ある作業方法の工程について

工数∝（規模）$^1$ である，あるいは生産性＝工数/規模＝一定

の関係を実測さえすれば，工程の工数に関する定量的特性は掌握できる．対象の規模が m 倍に増えたら，所用工数は m 倍になる．あるいは，所用工数を規模で割ると，１単位当りに必要な工数は一定である．

- [作業期間] ある作業 W の１回の所用工数を t とする．N 回の作業を一人でするなら，所用工数は N･t，所用期間も N･t である．今，M 人が共同で作業するなら，所用工数は同一だが，作業期間は N･t/M に減少する．
- 工程は階層的に展開でき，展開した下位工程を上位に再構築できる．

この工数に関する性質はハードウエア製造の場合と同一である．ハードウエア製造作業の管理技術は Industrial Engineering, (IE, 経営工学)[7]が築き上げた．その嚆矢は，19 世紀末 F. W. Taylor の提唱した下記の系統的な原価低減法（Time Study）である．これは前記の諸関係を経験則としてもちいた．

- ある作業(工程)をその下位作業(工程)群に展開し，
- 何れかの下位作業につき数種の（環境を含む）作業方法/手続を考え，
- 作業時間を計測し最高効率を産む環境/方法/手続きを選び，
- これらを組合せて元工程に復元する．

これに続き Gilbreth 夫妻は基礎的な単位作業の標準作業時間を定め，これを組合せて

（作業時間を含む）経済的な生産工程を計画する技法(Motion study)を発表した．Time study と Motion study の両方法は現在でも重要な基礎である．筆者等の技術思想を下記に示す．

『共通』Nelson 資料他から得たことは，ソフトウエア開発工程の工数の特性は線形系である．作業実態の特性は定量的に計測でき，実測結果を基に，より良い作業方法を求め，悪い作業方法を廃止することで，原価が低減できる．IE の工数にかかわる諸技術はソフトウエアに於いても踏襲できる．

『COCOMO モデル』Boehm は，工数と規模の関係を得るために，開発実績例を収集し，それから経験則を得て COnstructive Cost MOdel と名付けた．資料の集積が進むにつれ幾つかの版が生じた．彼の著 Software Engineering Economics に収めた実績資料の図は既に図 2a で示した．筆者らはこの図から工数∝（規模)[1] の中央傾向線と帯状領域に着目して２節で説明した性質を帰納した．

Boehm は，指数項の値が議論された時期を経験し，図 1 の Nelson 資料を承知しており，彼の著書に工数と規模の関係を示す図を引用している．かれは開発陣容に着目し，表１に示す３モード（自組織: Organic, 中間:Semidetached, 強統制 Embedded）の開発工程の種別で実績資料をプロットし，モード毎の傾向線式を求めた．その傾向線式の指数項の指数を表の右端に示す．

表　1

| シンボルと名称　　定義 | x 項指数表示 |
|---|---|
| ○自組織モード：自社システムを自社社員が開発する． | 1.05 |
| ×中間モード　：上下の両モードの中間的な場合． | 1.12 |
| △強統制モード：強い統制条件があり，これまで未経験な業務システムを顧客と打合せて開発する． | 1.20[1] |

Boehm の工数の定量化の目的は規模から工数（大工程毎の区分を含め）を予測する工学的な思想である．第１の思想は前記のカテゴリー化である．第２の思想は，これらモード内での，全体工数中の代表的な工程区分毎の比率で分ける．これでプロジェクトの見込み生産性と代表的な工程区分毎比率が得られる．簡単には，前記の指数値に従って過去の実績を参照したプロジェクトの全体～主要工程毎の生産性予測で中心である．これが第２の技術思想である．

『Function Point』FP 法は IBM の A. J. Albrecht の提案から標準化したものである．その狙いは，使用言語等に関わらない標準化した Function Point で，機能ベースの相互比較や事前予測である．簡単な骨子は，仕様がある程度に明らかになり概要の設計が行えるようになった段階で，主要なデータ等のソフトウエア構成条件を入れて数値計算により Function Point を求める．これに基づいて開発業者側では所用工数等を求めたり，クライエント側ではベンダー毎あるいは方式別の作業量の見込み予測に役立てるものである．これが第 3 の技術思想である．

## 4.　欠陥密度と減衰率

『共通』工数に続き，欠陥関係に欠陥に対抗するテストを含めて検討する．図３[3] はソフトウエアの規模と設計で作込んだ欠陥数の関係を両対数尺度で表したもので，Thayers 資料[8]の第３プロジェクトの直線尺度表示のグラフから再プロットした．図の各プロットは各プログラムに対応し，プロジェクト単位ではない．中央傾向線と副中央傾向線は図 2a, 2b と同様に引いた．

図の表す状況は図 1, 2a, 2b と同様だから，中央傾向線の x 項は $x^1$ とする.図 1, 2a, 2b と同様に，図 3 は「系は線形性」を示している．規模の 1 行当りの人の知的処理時間が工数になると看做せば，欠陥数は（知的処理時間）が(知的処理の欠陥率）に替るに過ぎず，グラフは同傾向と期待する．しかし仔細に見ると，図３は図 1 や図 2 とは異なる形状を示している．差異の第１は下記である．

図 3 の帯の半幅は N = 5，先行例の N = 4 より大，バラツキが大きい．

欠陥数を計数する過程を吟味しよう．図４は，最上部の開発工程から順次階層展開され，最下部のグラフの各区分に対応させた．最下部では設計が何の机上チェックも行わない純設計とその直後の机上チェックに分けてある．設計を行うと，純設計で欠陥が直線的に作込まれた後，机上チェック，第１テスト，第２テストで欠陥が摘出されて出荷されるが，現場でも欠陥が摘出される．図から作込んだ欠陥総数は，図の右端の現地摘出数から始まり，第２と第１テストでの摘出数および机上チェックでの摘出数の総和に等しい．

Thayers 資料での摘出数は資料中に明確に定義されてはいない．そこで，純設計の直後の机上チェックの義務づけや，そこでの摘出数の計上はしていない米国流と想定する．机上チェックは最も欠陥が摘出し易く，何もしないなら（作込み数を基準にして）摘出ゼロだが，厳密に摘出するなら約 80%を摘出できる．即ち，机上チェックを

1 指数項の値 x>1 は，$X^1$ より大きな勾配になり，規模が大きい程生産性が低い．プログラムは，前半は展開，中央のコード，以降は収束する V 字形網であり，充分に小さい規模の全体特性の x 項は充分に 1 に近い．より規模の大きいプログラムを考える，仕様を展開する段階があり，その後には数個の前記 V 字形網群があり，最後に統合テスト段階がある．受注開発では未経験なソフトを作るから，最初の仕様展開と最後の統合テスト段階の生産性は前記の最下位 V 型群より悪化する．この新しい V 型網に更新を続けると規模が大きい程品質と生産性が低下する強統制モデルができあがる．

Vol.2010-SE-169 No.11
2010/7/23

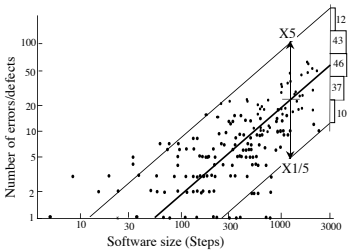


図 3. 規模対欠陥数

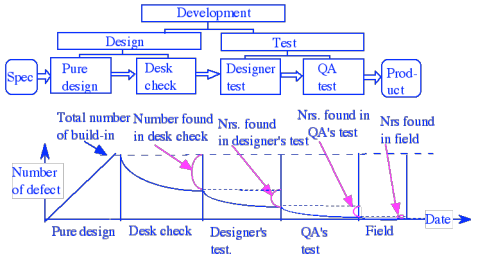


図 4．欠陥の作込みと摘出

義務づけ計上しないなら，0〜80%にバラ付く．下式のように(作込み数)/2 の定常値に(バラツキ)/2 が重畳すると看做す．

(作込み総数)/2 ± {(作込み総数)· (100 - 20)/100}/2

欠陥数自体とほぼ同大の擾乱が重畳するから,電力和で略式評価するとバラツキは 1.4 倍程度増える．これが N＝4 より増えた原因である．

欠陥の資料からバグの原因を探ることは研究の常道である．前記のように，(固定分)に近い大きさの(変動分)が重畳したのでは，*探索結果は略ランダムになり，安定な傾向が現れない．多くの研究者が敗退し，ソフトウエアの誤りは謎とされた原因はここにある.筆者等も大いに悩んだが,（机上チェック摘出数とテスト摘出数)/規模が安定な特性値になることから，計数法の欠陥に気付いた．また,「謎」や「ミステリー」は心の中の闇が作出すと銘記した.*

差異の第 2 はプロットが一様な帯状ではないことである．思考実験する．仮に第 3 プロジェクト全体を 1 プログラムと看做せば,これは（X＝全体規模，Y=欠陥総数）で中央傾向線上の一点になる．これを開始点と名付ける．今全体を N (N ＝ 1〜N)のプログラム群に分ける．多くの組合せが生じる．次に，N を M 倍に増すと組合せ数は今迄より飛躍的に増える．従って，図 3 の開始点から下の中央傾向線を回転軸として，開始点から下に指数状に広がる立体曲面錐状になろう．プログラムは，数百行以下だから，この立体曲面の上部は存在しない．残った立体錐が投影されて図 3 の末広がり状になるのであろう．これらが差異を作り出したと考える．

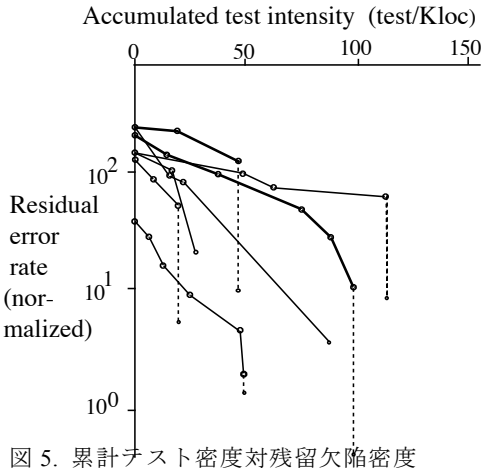


図 5. 累計テスト密度対残留欠陥密度

図 4 を更に検討する．各種の摘出曲線を粗く近似すると，負の指数減衰状である．これらを対数尺度用紙にプロットすれば,指数状減衰部分は直線状に現れるであろう．図 5[2]は各種実績の整理結果で，予測とおりである．

図の横軸はテスト数を規模で除したテスト蜜度の累計値を直線尺度で表し，縦軸は残留欠陥数を規模で除した残留欠陥率の対数尺度表示である．図の破線で表した垂線は，実環境模擬テストの名の 1 項目で摘出した数多い摘出数に対応する．

最右下の座標点（X＝ 最終的なテスト数/規模，Y＝ 現地摘出欠陥数）から始めて，左方向に移動して最終テスト，次いでその前のテストと遡及し，最後は座標( X = 0. Y ＝ 設計終了時即ちテスト開始時の残留欠陥率）に至る．

図 5 の曲線(群)で,ある点と曲線上の他の点の 2 点を考える．2 点を結ぶ線分の負の勾配は，急峻（負で大きい）程少ないテストで多くの欠陥を摘出できるから,「テストの有効度」と名付けた．あるテストの「テストの有効度」と該テストの，欠陥率の減衰量になる．これは，ハードウエア製造工程と全く同様にソフトウエア開発過程でも品質が定量的に計画/制御/管理できる[2]．

テストは統計学の用語では検査である．統計学の定義に従えば，テストは，対象の正邪を識別する過程である．その判定結果については，２種の誤り率が伴う．第 1 種の過誤率（$Ec_1$）は良品を不良品と誤る率，第 2 種の過誤率（$Ec_2$）は不良品を良品と誤る率である．通常テストで弾かれたケースは再チェックされるので，第 1 種の過誤は実務上はたいして問題にならない．しかし，第 2 種の過誤は大きな問題になる．対象の欠陥密度が Ed であったと考えると，テスト後の対象物の欠陥密度は，定義に従い，Ed•$Ec_2$ であり，欠陥密度は $Ec_2$ だけ減衰されており，前記と一致する．

『共通』工数，欠陥率とテストの減衰率を説明したので，システム全体の定量的な特性の基礎ができた．システム全体の様相を例により説明する．図 6[10]は 1970 年代末の優れたシステム開発の記録[11]他から実績を再現した．この図は，欠陥率の全容を示す図である．縦軸は欠陥密度，横軸は各工程に対応する．

左図は設計工程を示し，縦軸は直線尺度である．設計の 4 下位工程毎に，設計と共に欠陥が作り込まれ(↑)，直後のチェックで約 80%が摘出され(↓)，残留分が次工程に送られることを繰返している．最終的には残留欠陥密度 3.1 件/K 行で後続するテスト

２ 製造工程は多数回の繰返しを前提とする．一般の開発は，一回毎に異なる非繰返し性である．しかし，筆者等が提唱している究極的な開発方式のように，全ての開発が全て同様な基本パターンで行われるなら，同一の面からみた管理と監視は可能になるから，いわば個人毎の管理と同様なスタンスで品質管理できる．

Vol.2010-SE-169 No.11
2010/7/23

に引継ぐ．右図はテストを示し，縦軸は対数尺度である．図からテスト毎の残留欠陥率が直線尺度で読み取れる（テスト毎の減衰率も読取れる）．これは，将にハードウエア製造工程並みの監視であり，この図からどの工程がどれだけ貢献しているか，即座にかつ精密に判る．

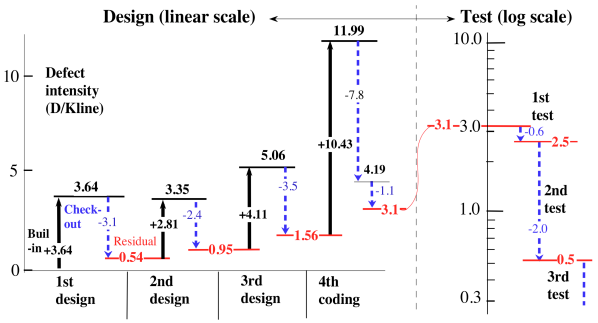


図６．残留欠陥密度

## 5. 帯状領域と改善

『共通：対数正規分布』　既に図 1 と図 2a，2b の工数の特性の帯状領域を示し，また図 3 では欠陥数の特性に帯状領域が現れる事を示した．図 2a，2b および図 3 には中央傾向線の上下の帯状領域を 5 部分帯状領域に分ち，各部分毎のプロット数を表す棒グラフがあり，ほぼ釣鐘状である．これらは開発工程に於ける工数と規模の関係，および欠陥数と規模の関係は対数正規分布であることを示す．

図 7[9]は，製品の出荷に先立ち実環境を模擬して重負荷を加える加速テストの資料である．重負荷を印加すると，常用運転よりも短時間で潜在欠陥が現れることを用いて，欠陥を摘出し品質を評価する．図の横軸は正常にランした時間の累計値を直線尺度で示す．縦軸はスタートしてから負荷を処理中に(何等かの欠陥により)アボートする迄の時間(RI)を対数尺度で示す．アボートすると，その原因を探求して修復し，再度ランさせて負荷を印加して試験を続ける．図の中央傾向線は右上に向けて直線的に伸びる．これはシステムから欠陥が順次取除かれて RI が指数的に伸びている，即ち残留欠陥密度の負の指数状の減衰を示す．図には帯状領域が現れ，残留欠陥密

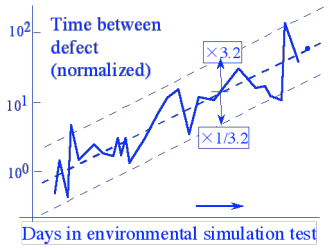


図 7．連続走行時間

度もまた対数正規分布である．

対数正規分布とは，対数尺度上で中央傾向線を中心とする正規分布であるから，σ を標準偏差として±3σ の範囲に 99.74%の資料が含まれる．±3σ の幅とは直線尺度では中央傾向線の 1/N 倍から N 倍の範囲になる．従って最大/最小の比は図 2a, 2b では N = 4 だから 16 倍に達する．

『受注開発形』以上はソフトウエア開発工程およびソフトウエア運用状態の特性であった．図 8 は開発結果のモノの特性であるプログラム規模を示す図で，DeMarco の実験結果[12]である．彼はプログラマー達に同一仕様を与え，プログラム化してデバッ

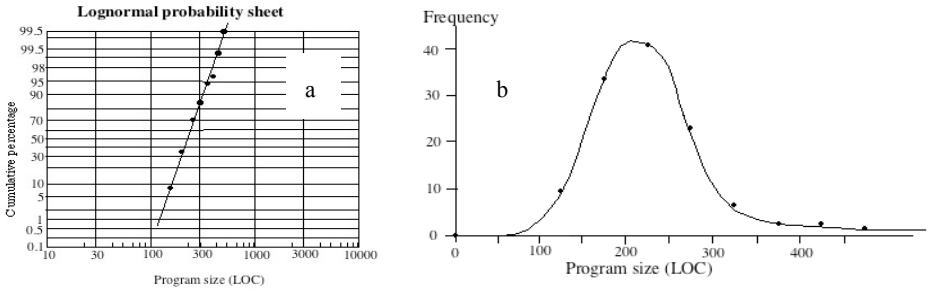


図 8. DeMarco の実験

グを行わせた後，各プログラムの規模を調べた．その最大/最小の比は 10 倍に近居大きさで，彼を驚かせた．彼はこれを“anomaly”と表現した．図 8a は対数正規確率紙に累計分布をプロットした結果を示す．この用紙は対数正規分布である資料をプロットすると直線状の傾向線が現れる．即ち，開発結果のモノも対数正規分布を示した．

図 8b は DeMarco の実験結果のデータを連ねる曲線を描き分布を再現した．図から右端で中々横軸に接近しない様相である．前記の N を使えば，曲線は 1/N から立上がって中央の平均値に達し，次にその N 倍に向けてだらだらと低下する．これもヒトの作業結果に現れる特性で anomally ではない．

統計学によると，対数正規分布は以下の場合に生じると云う．

「正で相互に独立である多数のランダム要因があり，これらの相乗積である変数は対数正規分布である．」

今迄に見た各種の特性の大きなバラツキは，全て上記に起因する．我々は意識しないが，我々の行動はこの構造に捕われており，この構造から逃れられない．

あるシステムを設計する時，その規模は，例えば N = 3 なら

「最小：平均値の 1/3 倍，最大：平均値の 3 倍，最大/最小比は約 10 倍」

のばらつきが生じる．これはシステムの開発コストあるいは生産対象あたりの開発コスト負担である．

○自由競争市場で営業している場合には，このような差異は許容できない．工夫を重ね，研究費を投じても規模を最小化し，開発費やコストを低減すべきである．
×受注開発で営業しており，例えば機能で契約したなら，ユーザはこの設計をとがめる権利はなく，規模が大きくても機能等の契約に違反しなら問題ではない．
このように，ビジネスモデルは技術の評価に大きな影響を与える．

通常モデルは自由だが競争市場で営業している．従って，常に競争会社との競争を意識しており，競争会社/競争製品に勝つ為に，常に並より以上の設計，できれば業界で誰もができない設計を心がけさせ，現在のトレンドラインを超える心方式の創造に挑戦するように教育する．従って，組込みシステムなど競争市場での営業製品の仕事は，通常モデルの人々，できれば自社のモラールの高い人々で行うべきである．

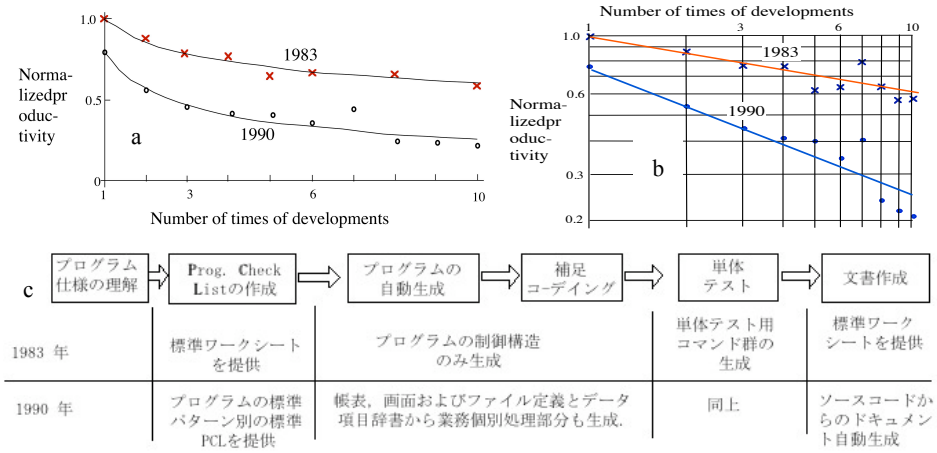


図 9. 習熟効果の実測例

『共通：習熟効果』対数正規分布と同様な自然の法則のひとつである習熟効果を説明する．図 9a は「日本のソフトウエア工場」の一つである日立の発表である．図の横軸は作業の繰返し回数，縦軸は作業の総工数を正規化して示す．図のように，作業を繰返す度毎に，総工数は低下する．これは習熟効果と云う．繰返す度に効率指標類が向上する．向上は始めは急激だが次第に緩やかになり，しかし，何時迄も向上し続ける．繰返し作業には必ず現れる．筆者等の責任に於いて，毎回のソフトウエア規模は等しいと仮定して，図 9a の横軸と縦軸を両対数尺度に取り図 9b に再プロットした．両年のプロットは直線状傾向線を示す．この種の習熟効果を対数習熟効果[13, 例えば 7 の第 4 部 2 章]と云う．
対数習熟効果は数式的に表示すると，以下になる．

$$Y = K X^{-A}$$

但し，K は初回の作業時間，Y は X 回目の作業時間，A は習熟効果の指数
このように初期の作業実績から未来の作業効率を予測できるので，各種の将来予測は全てこれを使っていると云って過言でない

図 9a の 1983 年で見ると，10 回目には初回の約 1/2 に低下している．図 9c は 1983 年の設計支援システムから 1990 年の支援システムへの強化を工程毎に示したもので，プログラムの自動生成等が加わる．この機能強化は初回で評価すると，僅かに 20%の生産性向上に過ぎない．しかし，図 9b で 1990 年の 10 回目では約 1/3 に低下しており，1983 年よりも低減の度合いが大きい．通常は改善前と後との工数を比較するが，この場合には習熟効果迄を評価しないと正当に評価ができない事が判る．

筆者等は設計の繰返しで，設計知識が対数習熟状に増加して向上が起こることを確認した．習熟効果は記憶によると思われていたが，発見後約 70 年後に証明された．
『通常モデル』生産工程の改善の投資は，経営者に下記等を約束する．

・設備投資前の具体的な作業と投資後の具体的な作業（作業の変更内容）
現状の製造原価 ― 変更後の減少した製造原価，および効果発揮時期
現状総工数 → 変更後の総工数，人員増減等を含む．
・新しく備える設備類　必要な設備，治工具類　仕様，金額，納期，効果時期

例えば図 9 の習熟効果は同一チームが作業する場合である．人，チームや仕事がガラガラと変る場合には習熟効果は現れない．従業員に安定して教育し作業を高度化させる等，経験により知識が蓄積され使われて強化され安定な状態にする．基準とする生産性等が安定し，各種特性値がバラツキ少なく正確に計られれば前記の約束ができる．

『通常モデル』対数正規分布で説明したことから，影響する各種要因群を夫々好ましい値に固定すれば，着目する特性は改善される．そこで影響する各種要因群から効果的に好ましい値が判り固定し易いものから取上げて改善すれば効率が良いことは云うまでもない．イタリアの経済学者 V. Pareto は 80%の富が 20%の人口により所有される集中性があることを見出した．品質管理の W. E. Deming 博士は終戦後の日本産業に以下を教えた．品質を悪化させる要因を突き止め，要因別に損失を調べて，損失順にソートする．過半の損失は，上位の少数項目から生じるから，この少数項目に絞って（パレート分析と云う）向上対策をする．品質が向上してきたら，再度パレート分析をす

ると，別の少数項目がクローズアップする．そこで，これを対策する．これを（効果的に）繰返す内にドンドンと品質は向上する．これは日本の品質を世界レベルに向上させた原動力とされている．

現場でこれらの品質向上策が推進されている時，設計では原価低減の為の類似活動（Value Analysis/Engineering)と云う活動を行っていた．筆者河野は，採算が極めて悪い，ある I/O 機器の VA に上記を３年余り適用した結果，これは工場でも高利益率のチャンピオンになり，この VA 担当者は本社 VA 推進センターに引き抜かれた．この時期になると，皆がこの原理に気付きだして，パレート分析は一般化した．

新規設計でも同じで，設計の時に必死に考えて各種方式を定量比較して方式を決め，製品化して評価する（多くの企業は研究報告書を書かせる）．これを何度も繰返すうちに，設計が上手になり楽になって行き，10 年経つとその道の権威になる．原理は（云うなれば対数正規分布の逆だから）全て同一である．上記のように，定量的に計測して改善することを行えば，生産性も品質も向上する．従って「銀の弾丸」は図 9 に示したように実在する．

## 6. 設計の原理

『共通』図 10[1, 3, 9]は時計プログラムの設計のデータフロー図の軌跡を示す．報告済だから要点を記す．データフロー図群は設計の進行を示す．仕様「時計」に出力データ次に入力データを加え明確化し，概念「時計」の「単位データフロー」にする．第３段のデータフローは親概念「時計」を階層展開し詳細化した子概念群で，{「時を得る」，「時刻表示を求める」および「表示する」}の各概念（「単位データフロー」）の 3 直列展開で，３概念（機能）の実行フローチャートも得られる．これら子概念を順次取出し，これを親として階層展開する．第４段は入力と出力データの両方を階層展開したデータ群を用いる 3 並列展開になる．最下段では展開の繰返しで概念が単純化しており「分針の角度を得る」の展開後は，60 進制の分の値を 360 進制の角度に変えるソースコード群に置換する．

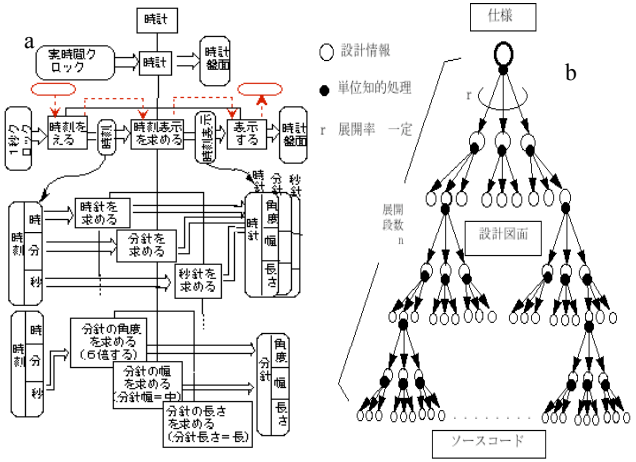


図 10. 時計データフロー展開




図 11.階層展開網モデル

設計の中心的機能は，自然言語表記の概念の階層展開連鎖の段階的具体化過程であり，ソースコードは 1 実現手段に過ぎず，この図は論理ゲート回路でも実現できる．これは拡張した構造化設計と看做せる．人はある意図を持つと，その意図／目的を達成する実現手段群を考える．それらの実現手段群の各 1 を新しい目的として，これを実現するための方法を考え，階層展開を繰返す．展開の度に下位概念になり，明確化，具体化，詳細化する．これは「ヒトの意図的行動」のパターンである．従って，経営レベル，設計レベル（ハードウエア/ソフトウエア），身体的行動レベルは同一構造に帰するから，工数や欠陥数などの process の特性，および規模等の product の特性は全て同一の性質を示す．ソフトウエアのみが異なることなど，あり得ない．

図 11 はデータフローの階層展開連鎖である．平均展開率は約３弱なので，定率展開として最後にソースコードになると考え，等比級数で計算すると出力数や中間の黒丸の展開処理数等が求まる．知的処理当りに微小工数が消費されると考えると，総工数が計算でき，総工数対出力数の関係が求まる．この指数項は $x^1$ であり，冒頭の図 1 の理論証明ができ，同様に知的処理当りに微小な誤り率を想定すると，図 3 の理論証明ができ，ソフトウエアでもハードウエアでも，経営～設計～身体的行動を通じて人の意図的行動は線形系と看做せる．ソフトウエアのみ他と異なることは無い．IE ではこれらの（理論証明無しの）経験式で世界の産業界を引っ張ってきた．工学 2 は理論証明は必須ではない．技術思想の齎す具体的な効果効用の大きさで判断する．論文の査読基準を改めるべきではないか？

これは言語を使い，親概念から子概念群に階層展開する繰返しである．親概念から子概念群の知識ベースを使えば自動設計できる．親概念でフレーム記憶に蓄えたデータである子概念群のスケルトンとこれで子概念群の記述を完成するメソッドからなるルールベースを設け，親単位データフローの動詞で前記フレーム群を読み出し，親の出力データと入力データでフレーム群中の該当フレームを選び実行させて 1 階層展開を実現し，一筆書き風に連続動作させればソフトウエアは自動設計[14, 15, 16]できる．これは米国および中国で特許を認可（言語表記した概念展開連鎖は中，米政府の公認済み）され，特許公報は一般配布されている．この例でソースコードでなく論理ゲートにすれば論理の自動展開が行える．従って，「銀の弾丸が無い」ことは誤りである．

## 7. おわりに

工学は厳しい商戦を闘う『通常ビジネスモデル』が前提である．通常ビジネスモ

デルの代表的な製品は組込みシステム類であり，厳しい世界的競争市場で勝抜く為の工程には，正確な定量評価技術を基礎にした，世界レベルの高い品質と優れた経済性が求められる．**これは受託開発に委ねるべきではない．**この領域は，アップルの iPhone や iPad，アマゾンの Kindle 等の始めから世界制覇を目指した量産品の多量販売が始まっている．本論文はハードウエアと同根で対処する方法を説明した．

『受注開発ビジネスモデル』は最大の営業形態の如くジャーナリスチックに書立てられている．しかし，その開発工程プロセスは各種指摘したように 30 数年前の水準に留まる．誤った謎や神秘をご認識頂きたい．

表２．売上高研究開発費率[17]

| Year | '94 | 1995 | '96 | '97 | '98 | '99 | 2000 | '01 | '02 | '03 | '04 | 2005 | '06 | '07 | '08 | '09 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R&D I (%) | 1.10 % | 1.04 % | 1.33 % | 1.58 % | 1.62 % | 1.72 % | 1.2 % | 1.1 % | 1.0 % | 1.0 % | 1.1 % | 1.02 % | 1.15 % | 0.84 % | 0.47 % | 0.82 % |

この種企業約 700 社で構成する日本情報サービス産業協会の売上高研究開発費率は表 2[JISA]のように，永年に約 1%程度である．かような低率の企業は，超巨大設備産業である鉄鋼業と食品関係のみである．因に Microsoft は約 15%と推定されている．

『通常ビジネスモデル』あるいは『受注開発』するが，生産設備あるいは母胎や部品パケージ等の技術を蓄積した『VF 形ビジネスモデル』の各企業の方々は，本論文を参考にしてますます技術を向上されることを期待する．

工学は，科学的態度を持つ人が今迄未知であった中から etwas Neues を掴む，あるいは「これが出来たら儲かる」と考えて，色々と苦心する内に不可能が可能になる領域に入るなどで，経験的に進歩する事が多い．経験主義的でプラグマチックな「工学」と，一応は理論で理論的な手順を踏まねばならない理論的「科学」とは，価値観もアプローチも違う．最小限度，社会が求める通常ビジネスの見地に立たれては如何か？

## 謝辞

この研究に各種のご示唆を賜った各位，特に自動設計の研究に貢献された埼玉大学河野研究室での学生諸君に感謝します．

## 参考文献

[1] Z. Koono, H. Chen and H. Abolhassani, An Introduction to Quantitative, Rational and Scientific Process in Software Development (Part 1 and 2), SoMeT 07, pp. 361-390, 2007.

[2] R. Nelson, Software Data Collection and Analysis at RADC, Rome Air Development Center, Rome, NY, 1978. (Western Michigan Univ. LibraryにPartial copyあり）

[3] Z. Koono, H. Chen and B.H. Far: Expert's Knowledge Structure Explains Software Engineering, Joint Conference on Knowledge-Based Software Engineering (1996), 193-197, 1996.

[4] B.W. Boehm, Software Engineering Economics, Prentice Hall, 1981.

[5] 吉田征，ソフトウエアの軽量化：事例と実感に違いはないか（統計的な検証)，情報処理，Vol. 26, No. 1, pp. 48-51, 1985.

[6] 河野善彌, 陳慧, 高野英樹, 森本祥一，学生チームによる組込システムの開発～10年間の教育から～,第26回ソフトウエア品質シンポジウム報文集，一般3-3, p. 171，日科技連，2007.9

[7] G. Salvendi, eds, Handbook of Industrial Engineering, John & Sons, 1982. 訳本　IEハンドブック，日本能率協会1986.

[8] Thayers et al., Software Reliability Study, Final Technical Report, RADC-TR-76-238, Rome Air Development Center, 1976. (英国，British Libraryにあり）

[9] Z. Koono, K. Ashihara and M. Soga, Structural Way of Thinking as Applied to Development, IEEE/IEICE GLOBECOM 1987, pp. 26. 6. 1-6. 6, 1987.

[10] 河野善彌，陳慧，ソフトウエアプロセス定量化モデルの提案，情処研報，2005-SE-147(12), 2005.

[11] E. B. Daly and D. A. Mnichowitz, The Management of large Software Development for Stored Program Controlled Switching Systems, International Switching Symposium 1979, pp. 187-1291, 1979.

[12] T. Demarco and T. Leister, Software Development: State of the Art and State of the Practice, 11th International Conference on Software Engineering, pp. 271-275, 1998.

[13] 師岡孝次，習熟性工学 ：動的評価と計画の技術（改訂版）,建帛社，1982.

[14] 河野善彌, ハッサンアボールハッサニ，陳慧，PCT 特許出願番号*W02002/097727,* 優先権主張　2001. 5. 28 ,
*中国特許 ZL 02 8 10859.0, July 26, 2006.
*米国特許 US 7,480,642 B2, Jan. 20, 2009.

[15] Hassan Abolhassani，河野善彌，陳慧，ソフトウエアクリエーション：ルールによる自動設計と知識による自動設計，情処研報，ソフトウエア工学，138-15, pp. 105-112, 2002.

[16] Z. Koono, H. Abolhassani and H. Chen, A new way of automatic design of software (Simulating human intentional activity), SoMeT 06, p. 407-420, 2006.

[17] 日本情報サービス産業協会,JISA 基本統計調査，　各年版概要，URL: http://www.jisa.or.jp/index.html