社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE
KBSE2003-57(2004-3)

人の設計知識構造と定量評価(1/2)

河野　善彌†，　陳　慧‡

†元　埼玉大学　工学部　〒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，ソフトウエア開発の作業の定量化の基礎を説明する．基礎として人の設計過程から始め，定量的な理論推計と実績で検証する．それは線形性を持つ．開発過程を分割し，定量的に計測する為の基礎を説明する..

キーワード　設計，机上チェック，テスト，プロセス，定量的計測，人間知能

Structure of Human Design Knowledge And The Quantitative Evaluation (Part 1/2)

Zenya Koono † and Hui Chen ‡

† Formerly Saitama University, 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 paper reports on foundations for a quantitaively controlled software development. As the basis, this starts fro a human design, theoretical quantitative evaluations ans the verficiations with field data. It is a linear system. Foundations for measurements by dividing the process are described.

Keywords Design, Desk check, Test, Process, Quantitaitve measurements, Human intelligence

１．はじめに

この報告は,人の設計過程の研究をソフトウエア作業の定量管理向けに纏めたものである.本報告は開発過程とその定量化の基礎，後続報告[32]は開発過程のプロセス論を報告する．両者共に，「人間知能」のモデルを用い，定量化に重点を置く．

この報告の２章では，人の知的作業が階層展開の連鎖であり,階層展開網モデルで定量評価できる線形系であることを示し，理論推計を実績値で検証する．３章は実際の特性からテストの特性と誤り減衰モデルを説明する．４章は，設計の生産性と誤り率の両者に現れる習熟特性を説明する．５章は，線形性を用いて分割した場合の特性を説明する．

人の設計の思想は，著者等が「人に倣った自動設計」として研究してきたもので,その他は筆者等がソフトウエア開発現場で採取してきた資料や経験則等の発表論文を用いる．

２．設計

２．１　人の設計

定量評価の基礎としてヒトの設計を調べる.各種の設計方式を，その詳細手続のレベルで見ると，設計の方法は実に千差万別に見える．しかし，中核的な思考は，全てに「概要から詳細へ」の流れが共通する．上流の例として軍事科学の「目的の階層性を取上げ，最下流の例をプログラム設計で説明する．

軍事科学で計画についての最高の原則は「目的の階層性」[1]という．一般向けの表現にすると，次のようになる．

> 最高指揮者には，最終的な「目的」が課される．この「目的」を達成する為，これを実現する「手段」群に階層展開して，各「手段」を部下毎に与える．各部下にとって，これは自分の「目的」になる．夫々の部下は，これを実現する「手段」群に階層的に展開して自分の各部下に与える.以下，これを繰返す．目的は次々と階層的に展開される．

ソフトウエア工学では，この階層展開を「目的樹」と云う．少し展開を続けると,対象の概念は目的に相応しい範囲の外に出てしまう．そんなことから「目的樹」は適用範囲が狭いものと見做されている．軍事科学では，展開されたもの全ては「目的遂行」の為なので,最後迄の全てを「目的の階層性」に含める．このように拡張してシステムの設計に適用すると,システムは

効用や効果 → サービス → 外部機能 → 内部機能

のように,「内部機能」即ち「プログラム仕様」に展開できる．

これはシステムの設計の原理にほかならない[2].

Wirthの段階的詳細化[3]は，プログラムの設計法の最も早い提案と云われる．これに続き各種の構造的な設計が提唱されたが，これらは段階的詳細化を更に抽象化した「機能の階層展開」を主張している．最もプログラムに近い例として，図１[4]に示す簡単な時計の設計例を調べる．これは，最も自然な設計の見本として準備してあった教育用サンプルである．

左端は仕様である「時計」，データフロー設計，フローチャート設計，コーディングを経て，右端のソースコードになる．

左のデータフロー図の最上段は仕様「時計」，第２段は，これに入出力データを付加して「単位データフロー」にしている．これを引伸し，入出力側の最大抽象点「時刻」「時刻表示」により，直列的に３分断する．展開結果は「時刻をえる」「時刻表示を求める」「表示する」の３単位データフローを含む．(Myerは，この分割をSTS分割法と名付けた．)

この段の左上，ひしゃげたビヤ樽状の開始記号から出発した制御の流れは，機能「時刻をえる」「時刻表示を求める」「表示する」を経て，右上の終了記号に至る．即ち，単位データフローの階層展開と共に，フローチャートが生じる．この小フローチャートを，図１の中央の最上部に示す．

第３段の１例の単位データフロー「時刻表示を求める」は，(入出力データが階層的に展開されることと相俟って) 並列的に階層展開されて，下位の３本の単位データフローになる．これはJackson 法に使う特徴的な展開である．

その中の１例，単位データフロー「分針を得る」は，更に階層展開される．ここでは，「分針の角度を求める」ことは「６０進制の分の値を６０倍すれば分針の角度になる」ことが，透けて見える．次の段階では，実現手段であるソースコードでの機能表記に置換する[5]．

この設計は，親概念の「単位データフロー」を，「詳細化したデータフロー」と「フローチャート」に展開する．前者は，親概念を階層展開した子概念である「単位データフロー」群で構成したデータフロー」であり，後者はそれらの「子概念の実行順序」を定めている．次には，詳細なデータフロー中の「子概念である単位的データフロー」を親概念として，階層展開を行う．付随してデータの階層展開を伴うが，この単純なルールの繰返しで全ての設計が行える．そこで，

ヒトの設計は親概念から子概念への階層展開過程

であり，実現手段に応じて，フローチャートの例のような付随条件が付く．

筆者等は，上記の親概念と階層展開した子概念の対を単位的な設計知識として使用する方式で，人の設計の再現に成功した．この方式はZipfが指摘した「労力最小化の原則：人は簡単な知識から順次高度化して各種の知識を使う」[6] ことに倣った方式構成である．Rusmussenの指摘した３典型[7]に準拠する．新しい概念である「単位データフロー」が与えられると，エンジンはこれを詳細化したデータフローとフローチャート(構造化チャート)を作りだす．動作は以下の３モードがある．

（１）上記の設計知識をそのまま記憶し，反射的に要求に応じて出力する，最高頻度の「技能のレベル」[8,9],

（２）前記で解決できない時は，子概念のデータフローを作るスケルトンと，それを修飾するメソッドからなるフレーム形知識を使う「ルール」レベル[10,11],

（３）前記もまた解決できない稀な場合，前記を基にして，単位概念の辞書を併用して解決する「知識」レベルがある[12].

２．２人の設計の特性

色々試行する内に，この方法は，判りやすく机上チェックし易い等良い設計結果をもたらすことが判った．そこで更に，蛇の多関節的にも似て柔軟なエキスパートに倣い，

展開は単純で小さな段階毎にする

こととしてレベルを揃えて，研究資料や実際の設計に用いてきた．これ以降，階層展開の展開率が約３弱に揃う傾向が出た．

例えば図１では，全ての展開は３展開である．図２は過去の研究材料の実績で，平均値は2.64[13] と 2.93[13] である．これは，人の長期記憶についてMagic Seven説の５±２[14] より小さく，残像等の関わる短期記憶の特性に近い．設計作業中の人の知的処理の速度が早い為，長期記憶での情報の静的な展開数は大きいが，動的（高速での）特性では，小さくなる為と思われる．

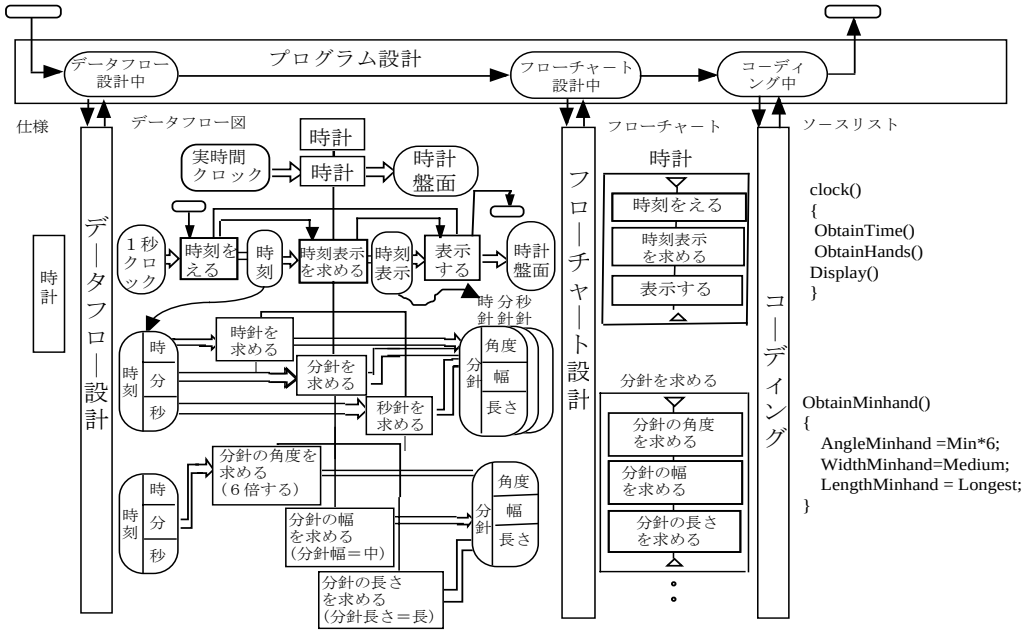


図 1 時計プログラムの設計軌跡

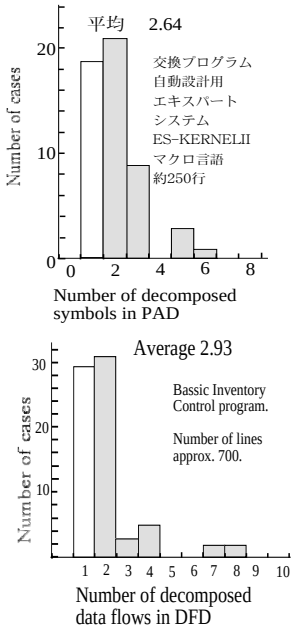


図２ 展開率の分布

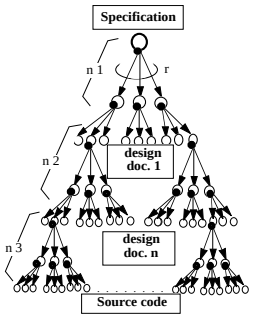
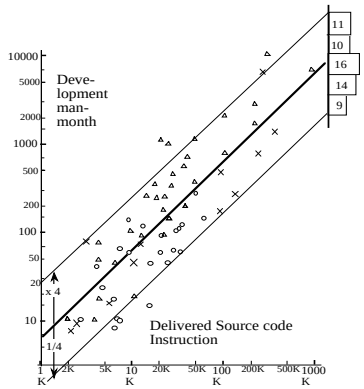
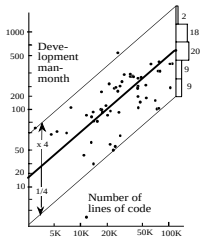


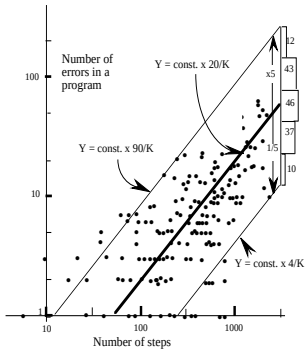
図３ 階層展開網モデル



a COCOMO data



b Yoshida data



c Thayer data

図４　理論推計と実績

展開数が一定なら，全体は等比級数で表せる．図３ [15] は，設計を表す階層展開網モデルである．白丸は情報，黒丸は階層展開の為の知的処理，矢印は情報の方向を示し，r は階層展開での一定な展開率とする．等比級数の数式を用いれば以下の計算ができる．

先頭から n 段の処理網を考えると，N_p を網中の処理総数，N_o を出力情報数は次式のようになる．

$N_p = 1+r^1+r^2+r^3+\cdots\cdots+r^{n-1} = (r^n-1)/(r-1)$

$N_o = r^n$

$N_p/N_o = (r^n-1)/(r-1)r^{n-1} = (1-1/r)/(r-1) = (r-1)/(r-1)r = 1/(r-1)$ ・一定

但し，n>1 とする．

ここで，１処理当りに微少時間 τ が消費されると考える．生産性＝（総工数／出力数）P は次のようになる[1]．

$P = N_p \times \tau / N_o \propto (N_p / N_o)^1$= 一定

今各処理当りに小さな確率 ε で誤ると考える．誤りは網中を伝搬して出力に現れる．作り込み誤り率 E は，下で与えられる．

$E = N_p \times \varepsilon / N_o \propto (N_p / N_o)^1$ ＝ 一定

設計の本質は，線形系で展開や統合ができて，生産性や誤り率は一定性を持つ．

２．３実績による検証

実績資料により前記を検証する．これには，人の特性を反映するように，ナイーブで，出来れば数桁の範囲の，偏りない，広汎な各種の，設計の資料が必要である．工数関係では，図４．a は Boehm の COCOMO 資料 [16], 図４．b は [17] の富士通の資料，また誤り関係では図４．c は Thayers の第３プロジェクト [18] を用い，原文図から読取り再プロットした．

この時期は，最初の混乱を乗切って各種システムが初開発された時期で，世界中で皆ほぼ同様な設計をしていた．すなわち，設計方式は良くても構造化設計程度，文書は少数でフローチャート位，支援系もコンパイラ，アセンブラー，リンカー程度であった．

しかし，何れも完全な資料ではない．工数資料は設計にテストを加えた開発工数である．テストの機械化が進んでいなかった当時では，設計とテストの工数はほぼ同じであった経験則，本研究では指数＝１の検証を目的として生産性の絶対値は問わない等から，「テスト工数を設計工数に繰込む」と考える．まず，生産性関係を調べ，誤り資料の問題は後に検討する．

図４の３種の資料は，人間信頼性工学（Human Reliability Engineering, HRE）[19,20] の慣例に倣い，両対数用紙にプロットした．図の太い傾向線は指数=1に対応し，プロットの密集領域を貫くように引く．これに並行して両側に等垂直距離の細鎖線の傾向線を２本引く．異常値の除外は通例の方法で帯状領域の上下に少数でほぼ同数のプロットを除外する．以上を充たすように，太傾向線のＹ切片と鎖線傾向線の垂直距離を調整した．

プロットは中央値の1/N倍からN倍（Nは，図a，bでは4, 4, また図ｃでは5）と広い範囲にバラつく．帯状領域を５区分して区分毎のプロット数を図の右端に棒グラフで示すとほぼ釣鐘状であり，プロットは対数正規分布状である．

塩見は VDT 作業につき詳しく調べ，工数と誤りの両者が対数正規分布であることを示した [21]．HRE で基準とされる人の単位作業の誤り率（Human Error Probability, HEP）の表 [26] でも，1/N 倍から N 倍（N = 3～5 が多い）のバラつきであり，対数正規分布状と理解できる．ソフトウエア工学では，ソフトウエア規模や生産性が最大／最小比が 10 倍にも達するとか，Layreigh 分布状と云う．これは何れも同根であり，無統制な人の作業の特性値は対数正規分布状で同程度のバラつきを示す．

以上から，本研究では以下を結論とする．

＊理論推計は実績資料で裏付けられた．

＊生産性と*誤り作り込み率*は，設計の特性として一定．

＊設計は線形的な系であるから，展開／統合が可能．

次に，誤りの検討を行う．

図５下部は，誤り摘出の累計曲線であり，上部の階層的に展開した工程と対応させてある．この曲線を上下反転させれば，これは左端での「作り込まれた誤り数」が次第に減衰する様相を示すもので，負の指数減衰状である．誤り作り込み総数Eは，以後の工程名の前部を添字にした各区間毎の摘出数の総合計で，次式で表せる．

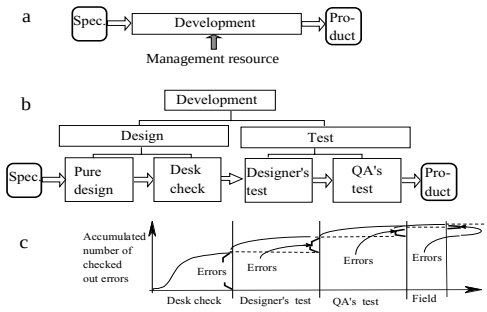


図５　開発過程と誤り累計曲線

1　Brooks は大規模化で生産性が低下すると云う [28]．大きくなると文書化や打合等が増えるなどの，作り方が変わるから，と云う．それは正しい．本報告とは議論の立場が違う．ここでは設計の本質的な特性として指数＝1を主張している．また階層展開網モデルで各種の議論をする時は，同方式で作業するから規模に関わらず一定と云っている．

$E = E_{desk} + E_{designer's} + E_{QA's} + E_{field}$

この理論式は作り込み総数である．Thayers 資料の定義が明確でなく，テスト摘出数と推定されることが第１の問題である．しかし，以下の考察を加えてこれを利用する．

右辺のE_{field}は商業的システムなら0.1E以下と思われ，無視可能．机上チェックで，誤りの事前摘出に努めれば最高 80% を事前摘出でき，努力しないなら摘出はほぼ0%．残留は20~100%である

そこで，一定の偏差と最大最小比５倍のバラつきが加わる．

図a，bの１打点はプロジェクト総計値だが，この各打点は各プログラムに対応するから，バラつきはより大きくなる．これが第２の問題である．

第３の問題点は，1プロジェクトの各プログラムなので，相互に独立ではない．成立ちから考察すると，（X = 総規模，Y = 総誤り数）を起点として Y = X の直線を対称軸とした半開き雨傘の中にプロットが分布する筈である．プロットはその形状である．従って，対称軸方向から見れば，これはバラつきの増大になる．

図a, b の N = 4 に比べ，図 c は N = 5 とバラつきはより大きい．右端の棒グラフの形状はほぼ釣鐘状で，偏りは無い．絶対値は不問なので，先の結論に影響しない．

誤り数計測の殆どの論文や資料の誤り数は，作り込み数でなくテスト摘出数である．（テスト摘出数は，当該プロジェクトの管理上は重要ではあるが．）また，多くの検討では，対数正規分布の大きなバラつきへの配慮がない．これらの資料では，何の有意な結論は引出せない．これらが「ソフトウエアの誤りの謎」であろう．

この罠に陥らない為には，以下の配慮をする．

・必ず作り込み誤りを計測する．最小限度，机上チェック摘出数とテスト摘出数を加える．それには，設計して一渡り簡単なチェックをした時に「完了」を宣言し，以後の机上チェック摘出は１件毎に記録を残す．

・少なくとも 1K 以上の規模での合計値～平均値を使う．

・1/N 倍 ~N 倍の範囲に及ぶ対数正規分布を考慮すること．

これらの処置をすれば，誤り率は安定な特性値になる．

３．テスト

テストは，通常の設計とテスト設計（何れもコーディングを含む）および両結果の照合である．これは線形系だから統合したテストも線形である．そこで，２．３は開発の検証とする．

テストはサンプリングが入り，理論解析は難しい．そこで，外部特性から特性値を求める．図５の下部の累計曲線を上下逆転すれば，残留誤りは負の指数状，等比級数的に減衰する．この減衰を計測した結果を図６ [4] に示す．

横軸はテスト開始以後の累計テスト（数，項目数，確認ポイント数）を規模で除した累計テスト密度，また縦軸は残留誤り率である．各カーブは，Y軸上の「設計で誤りを作込んだ（後に，ある机上チェックを経て誤りを摘出した後の）残留誤り率，即ちテスト開始時の残留誤り率」から始まり，右下の出荷時状態（X ＝総計テスト数／ソフト規模，Y ＝ 出荷時の残留誤り率）で終わる．出荷時残留誤り率は，使用開始後から誤りの発生がほぼ止まる迄の誤り発生数を規模で除した．

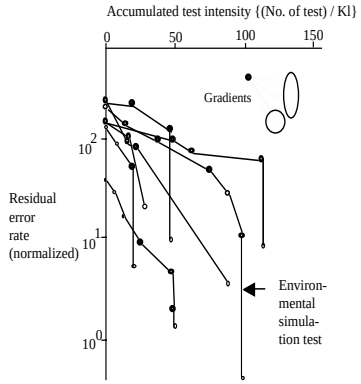
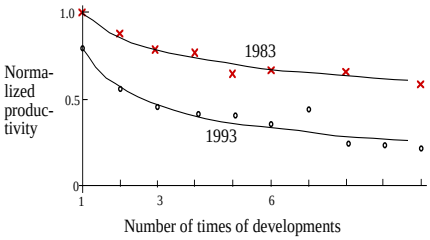
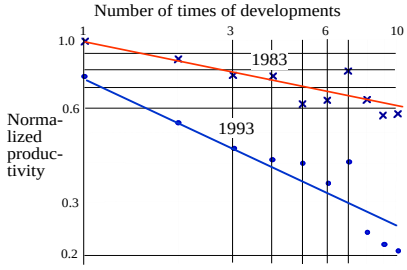


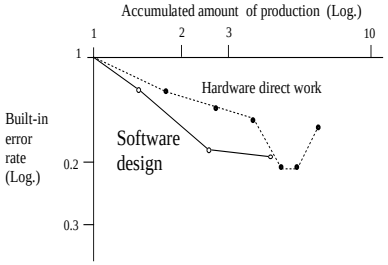
図６　テストによる残留誤り率の減衰



a 新規投入された新人チームの生産性推移（直線尺度）



b 新規投入された新人チームの生産性推移（両対数尺度）



c 誤り率の推移（両対数尺度）

図７　各種の習熟曲線

垂直に上に伸びる破線は，実使用環境に合せて負荷を掛けた実環境疑似テストの摘出した誤り総数を，テスト数１で評価している．以後，左に進む度に，当該区間でのテスト数だけ減り，当区間の摘出誤り数が加わる．

斜め右下へ降るカーブの傾向線の勾配は，１テスト/規模による誤りの減衰量であり，「テストの有効度」と名付ける．この実績の有効度は引き続く作業で保存されよう．出荷時の許容残留誤り率とテスト開始時点の残留誤り率から必要となるテストでの総減衰量を求め，テストの有効度で割れば必要な総テスト数が求まる．これは品質やテストの定量的設計を意味する．

・テストの有効度を支配する主要因は，テストの第 II 種の誤り[2]であり，設計やテストの仕様文書，テスト設計の各文書，テスト作業の正確性が関係する．有効度のバラつきを支配する要因は，テスト工程の均質度である．

４．習熟効果

人が新たなスポーツ（例：ゴルフ）を始めると，初めは急激に上達する．これは学習効果あるいは習熟効果と云う．その後，向上の度合いは次第に減少して行く．しかし，飽和はせず

2　第II種の誤りとは，テストの誤りにより，不正な対象を正しいと誤認する誤りを云う．第I種の誤りは，テストの誤りで，正しい対象を不正と誤認する誤りで，テスト結果検討で回復して問題にならない．

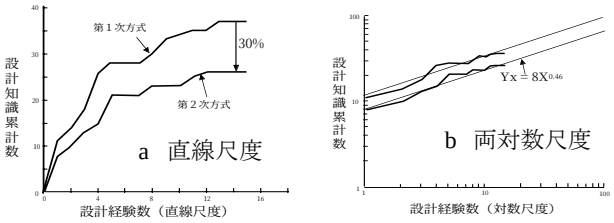


図８　技能レベル動作の場合の知識ベース蓄積量

にやればやるだけ進歩する．

筆者等はソフトウエアの生産性と誤り作り込み率に対数習熟効果があることを実証した[22,23]. 図７は各種の習熟曲線を示し，図a**[24]** は，「日本ソフトウエア工場」の一つである日立の実績例である．1983, 1993 両年のカーブは前記の傾向とおりである．図bは，毎回の規模の平均は同一として，両対数用紙に再プロットした．直線傾向線なので，習熟効果は対数と判定できる．図c[22] は，ソフトウエアの誤り作り込み率とハードウエア組立作業での誤り率を示す．

習熟効果は，IE, 経営工学で研究され，習熟性工学として纏められている [25]．対数習熟効果の時，X回目の作業時間Yは次式で表せる．

$Y = KX^{-A}$

但し，Xは繰返し回数，Kは初回の作業時間，

Aは習熟効果の指数.

人の作業の特性値（生産性，誤り率等）は対数習熟であり，習熟曲線により初期の少数実績から将来が予測できる. 生産性や誤り率を定量的に利用する場合には，（ハードウエア製造作業の場合同様に）必ず考慮する必要がある．

○ハードウエア製造は繰返し作業が特徴であり, 非常に早くから習熟効果が見出され研究されてきた. その利用は, 製造作業の工業化である分業／専業制をもたらした. 皆が一人毎に同じ一連の作業をする時，この一連の作業を作業時間が等しい1/M毎に分け，順番に１～Mの作業をする．一人当りに見ると，繰返し回数はMに増え，習熟効果が高まる．ベルトコンベヤでの流れ作業はこれを利用している．

○習熟効果は記憶によることが経験的に推定されていた. 筆者らは，自動設計の研究の中で定量的にこれを裏付けた．図８ [9] は, 技能レベル動作の場合に「経験から知識ベースへ蓄積する設計知識数」を示す図で，図のように対数習熟効果を示し，また，自動化による生産性向上も対数習熟効果を示した. 図では新経験を100% 蓄積している．もしこの率が低ければ向上もそれだけ少ない. 新パターンを認識し記憶する力が強い程, 習熟効果は高くなる. ルールレベルや知識レベルを追加すると, 経験を抽象化してルールや基礎知識を作ることが可能になり, 更に能力が高まる.

習熟曲線は，作業の工業化について示唆する所が多い．

・改善努力をしないなら，殆ど進歩しない．
・改善努力～効果が大きいほど．習熟効果（勾配）大．
・知的能力が高い程，習熟効果（勾配）大きい．
・同一改善努力×同一効果×環境（一定）なら
　傾向線は直線状，もし変動があれば曲線が乱れる．
・分業して専業化する方が，通常は効率は高まる．
　　その度毎の受注で仕事すれば，習熟効果は低い．
・繰返しを増すには，工程を標準化すればよい，
　　プログラム／アーキテクチャを標準化する，
　　製品～feature/機能は標準化する．

これらが進めば自動化が進む［**29,30**］.

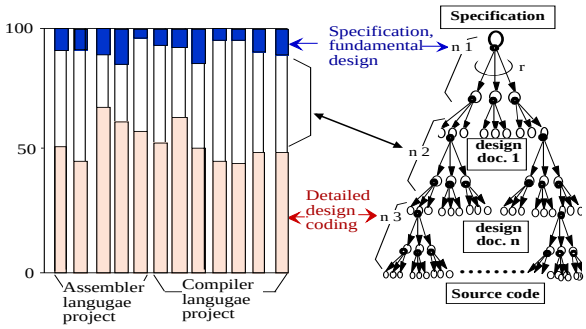


図９　設計の部分毎の誤り数の比率

５．分割と統合

テストは, テスト設計と通常の設計との動作の比較なので, テストは線形系になる. 設計とテストを統合した開発もまた線形系になる. 開発, 設計, テストは何れも分割や統合が出来る．

図9[27]に示す設計の分割実績例に基づき, 設計の分割を調べる. 各種プロジェクトの工程区分を合わせる為に，設計の全工程を以下の粗い３区間に再仕分けした．

１．仕様から基本設計迄
３．詳細設計からコーディング
２．上記の両者の中間

左の棒グラフは作り込み誤り件数の３区間毎の比率を示す. 図のように，各プロジェクト毎の比率はほぼ揃う．

・最上流：約1/10, 最下流：1/2，および中間：残余

この再区分は，右の階層展開網をn1段， n2段， n3段に水平に３区分したと等価である．作り込み誤り数と工数の両者は，区間毎の知的処理数の比に従うから，各棒グラフがほぼ同一比率を示す．これらから，次のように云える．

イ．工数や作り込み誤りは区分毎に加わる．
ロ．工数や作り込み誤り数の構成比は保たれる．

イ項は，ソフトウエア作業でも統計的な品質管理が可能であることを示し，また従来同様に作業すれば，品質や生産性は同様になる経験を裏付ける．ロ項も，全体工数を一定率で各工程に配分する等，経験的に知られている．これらは階層展開網モデルでメカニズムが説明できる．

６．纏め

この報告では，以下を報告した．

・ソフトウエア設計は人の意図的行動の１種で，概念の階層展開の連鎖であり，階層展開網モデルで定量評価が出来る．
・このメカニズムで理論推計して,「生産性や誤り率の一定性」を示す結果が得られ，実績資料でも裏付けられた．
・「生産性や誤り率には習熟効果」があることを示した．
・ソフトウエア開発工程は線形系で,「分割や統合が可能」.

これは，経営工学の生産管理の技術が使えることを示す．
また「生産管理」の基礎である「一定性や習熟効果」がメカニズムと理論モデルで説明できることは, 今迄経験ベースであった生産管理の基礎の充実でもある．

この報告では, 生産性や誤り率のバラつきが大きいこと, 知らずに犯す誤りを指摘し，正しい取扱を明らかにした．

以上の両面で, 本報告はソフトウエア工程の定量化の基礎を明らかにした．

７．謝辞

この研究は, 埼玉大学での「人に倣ったソフトウエア自動設計」の研究の成果が仕上げに使われている．Far 先生，アバールハッサニさん始め, プロジェクトに携わった多くの学生各位に厚くお礼もし上げます．また，自由な環境を与えて頂いた埼玉大学情報システム工学科の皆様, 色々な御意見を頂いた子信学会知能ソフトウエア工学研究会の皆様に感謝します.

参考文献

[1] Cluasewitz, Karl von, Vom Kriege, 1832. 淡徳三郎(訳)，戦争論，徳間書房，1965.

[2] Koono, Z. and Soga M., Structural way of thinking as applied to systems design, IEEE COMSOC Global Telecommunication Conference 1990, pp. 306.1-7, 1990.

[3] Wirth, N., Program development by stepwise refinement, CACM 14-4, 221-227, 1971.

[4] 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.

[5] Chen, H., Far, B. H. and Koono, Z., A systematic construction method of an expert system used for automatic software design - Acquisition and reproduction of design knowledge from design process-, Journal of Japan Society for Artificial Intelligence, vol. 12, no. 4, pp. 616-626, 1997.

[6] G.K. Zipf, Human Behavior and the Principle of Least Effort," Hafner Publishing, 1972.

[7] Rasmussen, J., The role of hierarchical knowledge representation in decision making and system management, IEEE Trans. on Software Engineering, 18, 6, pp. 523-533, 1985

[8] Chen, H., Tsutsumi, N. and Koono, Z., Software Creation: An intelligent CASE tool reusing elementary design knowledge, Joint conference on Knowledge Based Software Engineering 1998, p. 73-80, 1998.

[9] Chen H., Tsutsumi N., Takano H., Koono Z, Software Creation: An Intelligent CASE Tool Featuring Automatic Design for Structured Programming, The Journal of Institute of Electronics, Information and Communication Engineers -Special Issue on Knowledge-Based Software Engineering-, Vol. E81-D, No.12, pp. 1439-1449, Dec. 1998.

[10] Koono, Z., Chen, H., Abolhassani, H. and Far, B. H., Design knowledge and software engineering, Wuhan University Journal of Natural Science vol. 6, No. 1-2, pp. 46-58,2001.

[11] Abolhassani H, Chen H., Far B. H. and Koono Z., Software Creation: A Study on the Inside of Human Design Knowledge, The Journal of Institute of Electronics, Information and Communication Engineers, Vol. E83-D, No. 4, pp. 648-658, April, 2000.

[12] Abolhassani, H., Chen, H. and Koono, Z., Software Creation: Cliche as intermediate knowledge in software design, The Journal of Institute of Electronics, Information and Communication Engineers, Vol. E85-D No.1, pp. 221-232, Jan., 2002.

[13] Zhang, S., Chen, H., Far, Behrouz H. and Koono, Z., A development of a design system and some statistics, Technical report of IEICE, KBSE98-56, pp. 9-16, 1999.

[14] Miller, G. A., The magical number seven, plus minus two. Some limits on our capacity for processing information, Psychological Review, 63, 81-97, 1956.

[15] 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.

[16] Boehm, B. W., Software engineering economics, Prentice Hall, 1981.

[17] Yoshida, State of the arts and problems of software metrics, Journal of IPSJ, vol. 26, No. 1, pp. 48-51, 1985.

[18] Thayers, T. A., et al., Software reliability study, Final Techinical Report, RADC-TR-76-238, Rome Air Development Center, 1976.

[19] 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.

[20] 塩見弘，人間信頼性工学入門，日科技連，1996.

[21] Shiomi, H., On analysis and summarization of human reliability data for simple VDT operation., BICRMS 92, pp. 372-377, 1992.

[22] Koono, Z., Igawa, K. and Soga, M., Structural way of thinking as applied to improvement process, IEEE COMSOC Global Telecommunications Conference 1988, pp. 40.1.1-6, 1988.

[23] Koono, Z., Tsuji, H. and Soga, M., Structural way of thinking as applied to productivity, IEEE COMSOC International Conference on Communications 1990, 204.2.1-7, 1990.

[24] Tsuda, J. et. al., Productivity analysis of software development with an integrated CASE tool, ICSE 1993.

[25] 諸岡孝次，習熟性工学（改定版），建帛社，1982.

[26] Swain, A. D. and Guttman, H. E., Handbook of human reliability analysis with emphasis on nuclear power plant applications, NUREG/CR-1278, SAND 80-0200 (1983)

[27] 河野善彌，開発（設計）作業と誤り発生の構造，第９回ソフトウエア信頼性シンポジウム，6-3，１９８９．

[28] Brooks, F. P. Jr., The mythical man-month, Addison-Wessley, 1975 and its revised edition.

[29] 大野治，小室彦三，降旗由香里，今城哲二，古宮誠一.，多次元部品化方式によるソフトウエア開発の自動化ーバッチプログラム用スケルトンの作成とその十分性ー, 子信学会, 論文誌 J83-D-I No. 10, pp. 1055-11069, 2000.

[30] 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.

[31] Cusumano, M. A., Japan's software factory: A challenge to U. S. management, Oxford University Press, 1991.(訳) 富沢宏之，藤井留美，日本のソフトウエア戦略ーアメリカ志気経営への挑戦，三田出版会，1993.

[32] 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)