ソフトウェア工学　133－8
（２００１．９．１９）

Industrial Software Engineering（産業界の期待）

河野　善彌§

最高で最善のソフトウエア設計者の設計知識構造をモデル化した．その内部構造と外部特性は，実際の人の設計の場合と合致する．それ故 Industrial Engineering がソフトウエアの場合にも適用可能になる．これらから産業界のニーズに応えて，現状より拡張した Industrial Software Engineering を確立することを呼掛けている．

Industrial Software Engineering (responding to needs in industry)

Zenya Koono §

Based on the best and expertise software designer, the design knowledge has been modeled. Its internal structure as well as its external characteristics agree with actual human design. As a result, Industrial Enigneering may be applicable also to software. This paper calls for establishing such extended Industrial Software Engineering which responds to needs in industry.

１．はじめに

この報告は, 開発Processの内部構造および外部特性を明らかにして，ハードウエアの合理的・定量的・科学的な生産管理の体系であるIndustrial Engineering（IE)がソフトウエアにも適用できることを示し, ソフトウエア工学が産業界の期待に応える道を提起する.

ソフトウエア工学は1968年のNATOの国際会議からとされる. 当時IBM System360等の第３世代コンピュータが急普及し, 大形ソフトウエア開発が急増した. その多くは「納期遅延」「品質不良」「予算超過」にあえいだ. 「工学」と銘記したのは,「開発は系統的 / 合理的 / 科学的であるべき」との社会の要求であった.

これに応え1971年以後の数年の間に各種の構造的な手法が提案された. これらは人間の知の階層性に立脚するもので，人の思考（Process）の入口を開いた . しかしこの後は攻めあぐみ, 流れは一転して説明容易な理論重点 /Product 中心になり現在に至っている．Product は重要な１面ではあるが , 当初の「納期遅延・品質不良・予算超過」の元であるProcessは未解明のまま残っている.

この報告は，Product と同じく Process も含め統合する為の研究を報告する．２章は，人を含む Process への取組みの１提案を示し, ３章ではその内部構造に基づく「人の設計の再現」を説明する．４章は内部構造の単純なモデルで，Process の主要な外部特性が説明できることを示す. これらに基づき，５章は定量的合理的科学的な管理を目指した学問 Industrial Engineering (IE) の体系がソフトウエア開発にも適用できることを示し，６章は，産業界のニーズに応えるべく呼掛を行う.

§ 東亜大学　通信制大学院
Graduate School, University of East Asia
koono@vesta.ocn.ne.jp

２．人の設計

２．１エキスパートと知識のモデル

ソフトウエア工学でのprocess研究が壁にぶつかったのは人の知を扱う困難性にあった．自動設計などでprocessの研究では「設計情報の変換関係」に着目するものが多い．一方，知に迫る人工知能では「変換関係を行わせる上位の知識」着目する．残念ながら，何れも現実の人間の優れた知の側面に留まっていると思う．

人を含めたprocessの改善については，Total Quality Management (TQM, 旧称 TQC) 実績を上げている．半導体や自動車の生産などのハードウエアだけでなく，ソフトウエアにも適用されている．現在では両者を統合したISO9000系列の国際規格を産んだ．但し，知の働きはblack boxとして外側から攻めており，人の知の解明には至らない．

これらから，筆者等は人を含むprocessを知るには新しい道を拓くしか無いと考えた．これは容易なことではない．そこで漸進的な以下の戦略を考えた．

> *設計者のシミュレータを作る．これを研究し評価してより良きものへとボトムアップに積上げ，あるいはトップダウンに掘下げていく逐次近似で人に迫ろう．*

この研究には次の目標を掲げた．

> *如何なるソフトウエアでの自動設計を可能にできる合理的で科学的な基礎を確立しよう．*

１困難な目標ゆえ，多くの試行錯誤を重ねた．現時点での研究の為の前提条件を，以下に纏める．

第１の問題は，人毎に設計方法がバラつくことである．解として「最善(バラツキ無し，誤り無し）の設計を行う最高のエキスパート[脚注1]」を想定する．

第２は，エキスパートほど多様な方法（即ち設計知識）を駆使することである．これは，蛇のしなやかさに見倣えば「小さな設計進行段階毎に設計判断を行ない，最適な方路を取ること」と考えた．そこで，この「小さな設計段階」を作業進行の単位に取る．更に，問題解決機構では各種のエンジンを順次調べて多様性に挑む．

第３は，普遍性ある結果を得る方法である．前提条件として個別の設計が適用できる*「枠組みを中心」*とし設計に共通な人の意図的行動を中心に攻める．

第４に，題材以外にはｿﾌﾄｳｴｱ独自のことは一切持込まない．（筆者自身の部品 /ﾊｰﾄﾞ/ｿﾌﾄ/ 管理の経験から）これらに共通な枠組みこそ設計の本質と考えた．唯一の例外は構造化ﾁｬｰﾄで，その優れた表記能力を利用する．

第５に，*「作業の表記手段は自然言語」*を用いる．人の高度な能力は自然言語に依存するし，普遍性と拡張性を確保でき，作業の実態にも合う

第６は，確実で再現性がある精密な研究手段である．最高のエキスパートが最善の設計を行った「設計図面～文書」を使う．これは文書類は，設計情報の変換の流れの中のノードであり，（設計の補助手段ではなく）processの中間成果物，同時に設計の履歴でもある．これは再現性に富み確実な処理ができる研究材料になる．

第７には「common sense問題」を回避したい．これは，逐次近似で研究する時，必要以外は触らないで済むことが判った．研究で確認したベースに基づき，順次積上げあるいは掘下げる時に広げていけば良い．

第８の知識の定義は，着目する知的ユニットの設計情報の入出力の変換関係とする．誤りが無いから精練は不要で，順次階層的に研究を進化させるから，抽象化は不要になる．

第９は，設計知識を系統的に獲得し，系統的にｼｽﾃﾑに再現する方法である．研究着手の時期では，これは出来ないとされていた．これは「知識構造に従えば，process中の着目部分の入出力関係である知識は系統的に求められる．獲得した知識を下敷である構造に従ってはめ込んで構築すれば，系統的に全体を再現」できる[陳97」ことが判った[脚注2]．

ｴｷｽﾊﾟｰﾄは，実存する高成熟度組織の人達を観察して理想特性を抽出した．*エキスパートのモデル*は以下のとおりである．同一領域での開発に永年の間従事し，改善を積上げた結果，人々は高い人格と優れた技術を持ち，積極的な進取の気性で高いモラールを示し，組織の目的に向け全員が一致協力する等チームワークが良く，相互のｺﾐｭﾆｹｰｼｮﾝが早く広く正確である特性を持つ．業務面では，サービス，製品，構造，作業方法（工程，Work process）が高度に標準化され自動化等が進んでおり，そのソフトウエア品質は非常に高く(e.g.誤り率は低く)，事業の成功を重ね続けている．

かような組織は階層的な工程体系を持ち，同じく階層的な設計文書体系で直交的に区切る．技術的な知見は著しく均質で，会話は極めて論理的に進み正確であり，明解な自然言語（日本語）を駆使する．用語等の標準化が徹底しており，設計文書は明解で正確な用語で書かれている．小さな進行段階毎に文書化し，これをチェックするので文書量が多い．かような人も含めた階層的工程を*知識のモデル*とする．

このｴｷｽﾊﾟｰﾄの設計結果であるような研究用サンプルを作る．自然言語での意味的な展開を単位として設計して均質な「小さな設計段階」を実現する．各種の図面文書を駆使し，各種のレビューやチェックを繰返し，簡単でも誤りの無い研究素材を作る．これを対象として，各種の特性の調査・研究および設計の再現を行わせる．

脚注１．このエキスパートモデルを劣化させて行けば如何なる成熟度でも再現でき，その様相が判る．

脚注２．同時期に提案されたKnowledge acquisition by modekingと一致する所がある．

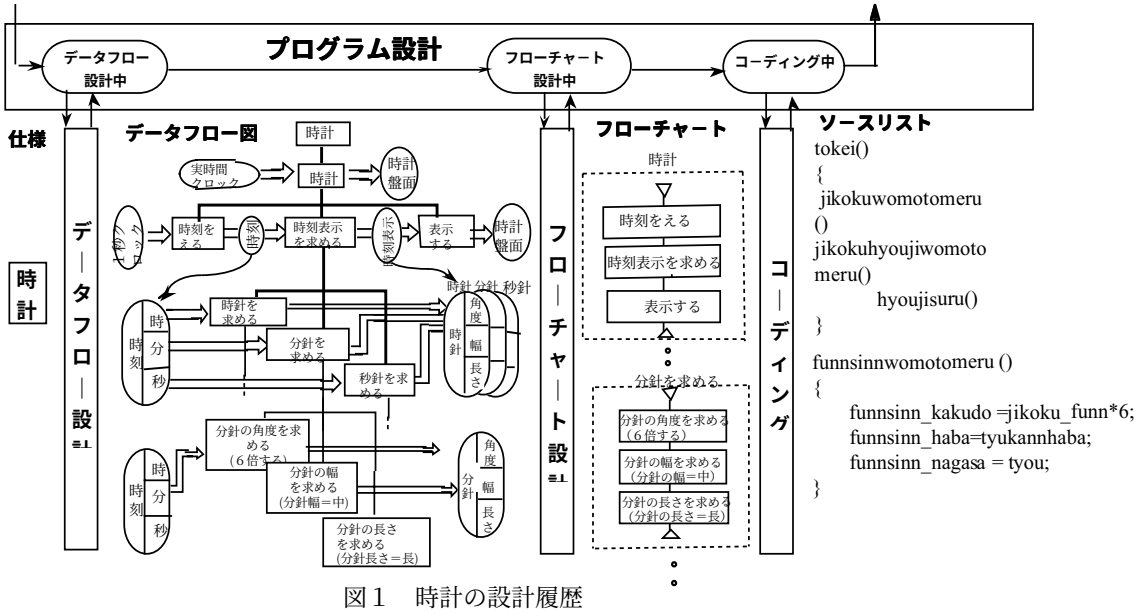

図１　時計の設計履歴

２．２設計の基本特性

簡単であるほど基本特性は認知し易い．図１[Koono96]脚注３は，「時計」の簡単なプログラムの設計履歴を示す．設計進行段階は意味的な展開単位に揃えた．この時，ﾌﾛｰﾁｬｰﾄはﾃﾞｰﾀﾌﾛｰ図の機能箱群を取出して制御の流れを加えたもの，ｿｰｽｺｰﾄﾞはﾌﾛｰﾁｬｰﾄの機能記述をﾃﾞｰﾀ演算処理に変えプログラム言語化したものになる．

設計の中心的な知的処理はﾃﾞｰﾀと機能（処理）を決定するﾃﾞｰﾀﾌﾛｰ設計にある．上から下へ各ﾃﾞｰﾀﾌﾛｰ図を見ると，階層展開を繰返すことが中心である．人の設計は次の性質を持つ[陳 97]．

人の設計は概念の階層的な展開を繰返す．展開する毎に，より明確に，より具体的に，より詳細になって行く．展開した概念が対応する実現手段（e.g. プログラム言語ステートメント）が透けて見えた時，展開はせずに実現手段による表記に変換する．

この性質はソフトやハードの設計，ビジネスプロセス，更には人の*意図的な行動全般に共通*している．

ﾊｰﾄﾞｳｪｱ直接作業と同様に，人を含めた作業の流れの１区間を「工程」と名付ける．図の上部の「プログラム設計」は，下位の「ﾃﾞｰﾀﾌﾛｰ設計」「ﾌﾛｰﾁｬｰﾄ設計」「コード化」と階層的に展開され，その「ﾃﾞｰﾀﾌﾛｰ設計」は更に何回も階層展開を繰返す．工程は次の性質を持つ．

工程は階層的に展開でき，展開するにつれ，より明確に，より具体的，より詳細になる．最後は人の単位的行動に至り，更に続けると人の単位的な思考に至る．工程は設計知識そのものである．

脚注３．図１には，MyersのSTS分割やJSP(Jackson Structured Programing)の中心形式が見える．良い設計をするとこれらのパターンが出現する．構造化設計はこれらのパターンに着目して体系化したものである．図の設計は構造化設計に適合するが，それに準じた訳ではない．

設計文書入力と出力の情報の対（入出力変換関係）を取出せば，当該工程の設計知識が得られる．これを*設計ルール*と名付ける．図２は時計の概念展開の連鎖を示す．矢印の経路で囲まれた１段階の階層展開は設計ルールであり，これは親概念と子概念の対でもある．図のように設計ルールを順次に接続すれば，全設計になる．

３．人の設計行動の再現

３章は Work process，知識モデルの内部構造の証として，ｿﾌﾄｳｪｱの自動設計の研究結果を説明し，次の４章でその外部特性を説明する．

３．１実験の計画

既に説明した設計知識の多様性につき，人間工学での [Zipf72] の「労力最小化の原則」を説明する．

人は問題に直面すると，初めに最も簡単な解法を試みる．これが不成功なら，より高度は解法を使う．問題が解けるまで，高度化を繰返す．

これは複数種の設計知識の存在を意味する．また，人工知能での [Rasmussenn85] の「技能レベル」「ルールによ

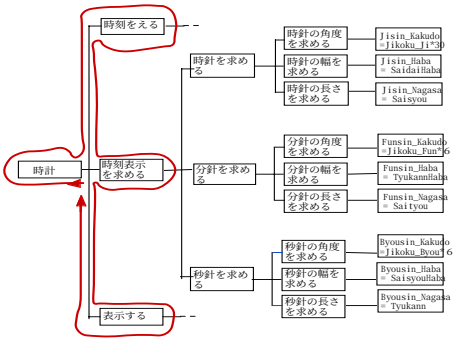

図２　時計の階層展開網

るレベル」「知識によるレベル」説とも符合する．実際に，大部分の設計は単純な作業で，中程度の難しさがこれに次ぎ，高度なものはごく少数であることは，我々が日常に実感する所である．本報告では，

１）設計ルールの再利用による自動設計

２）設計ルールを自動生成する自動設計

により人の設計の再現確認した結果の概要を示す．

３．２設計ルールの再利用による自動設計

単純な再利用による自動設計を行う知的ＣＡＳＥツールを作成した[Chen98]．これは人がＣＡＳＥツールを用いてﾌﾟﾛｸﾞﾗﾑを設計し，構造化ﾁｬｰﾄＰＡＤに表記する．PAD図面から設計ルールを自動獲得して，知識ベースに格納する（設計知識の抽出）．

図２は時計の中心部のＰＡＤの図でもある．（ＰＡＤは判断と繰返しのみ階層図と異なる．）図形を辿る動作の例を図の矢線で示す．親概念「時計」のシンボルから出発して右にツリーウオークすることにより，展開結果の子概念（群）が取出せる．

自動設計では，まず，最初の仕様となる概念をＰＡＤ上に描く．親概念のみを（前記の矢印と同様に）図式的に採取し，これを親概念として知識ベースに照会する．親概念を階層展開した「設計ルール」を得て，先のＰＡＤの対応個所以降に貼付ける．自然言語レベルでこれを繰返し，最後にソースコードへの変換を行う．

これは物真似的に自動展開する．もし，１親概念に対し複数の設計ルールがある時は，人が選択して指定する．物真似でも，既存設計の修正を繰返す所謂保守では効果が見込める．自動化手段の研究には省力効果の定量評価があるべきと考え，工夫を重ねて評価した．

図３ [Chen98]にその結果を示す．図３．ａは横軸に設計回数を，縦軸に累計設計ルール（種類）数を示す．１人の設計者が原設計し，更に別人が改良設計する等，確度向上に努めた[脚注4]．カーブは初め急激に立上り次第に緩やかになる習熟曲線状である．図３．ｂは両対数用紙にプロットした結果で，２本の同勾配の直線傾向線が現れ，対数習熟効果が確認できた．

習熟性工学[師岡94]により評価した結果，改善後の作業工数は３回目では総工数は１/３に低下する工数低減効果が判った．図３．ｃは縦軸に自動展開される率を示し，初めに急激に立上り 80% を越すが中々 100% に近づかず，浅知恵の限界が見える．

再利用される設計知識を，頻度別にソートすると指数減衰的になる．この中で上位の少数部分が高頻度で使われることで効果が出ている．同時に，設計者の記憶能力が設計の効率に関係することも判る．基本的な設計が一巡した後の（所謂保守）設計では，入門者でも役立つがそれは既出の設計知識が高頻度で使われることと理解することができる．このように，内部構造を知り，定量的な研究をすると，人の作業が良く理解できる．

３．３設計ルールを自動生成する自動設計

入出力データも含む親概念を展開すると，子概念はﾃﾞｰﾀﾌﾛｰ図とＰＡＤの対になる．これを使えば設計の上流から下流までを１方式で標準化できる[陳97][脚注5]．ﾃﾞｰﾀﾌﾛｰ図とＰＡＤの両図上で同期的に設計させる統合知的ＣＡＳＥツール[山田01他]をプラットホームとして，設計ルールを自動生成する自動設計システムの実験を行った[Abolhassani01b]．これはルールに基づく自動設

脚注４．交換接続プログラムでサービスを次々増加させる変更を行い，これに対応する運転管理プログラムも次々と変更した．この中のﾃﾞｰﾀｱｸｾｽﾌﾟﾛｸﾞﾗﾑの変更を用いた．

脚注５．構造化分析はその上位でデータﾌﾛｰのみ，下流の構造化設計は構造化ﾁｬｰﾄを用いる．その結果，中間の接続に無理がでる．

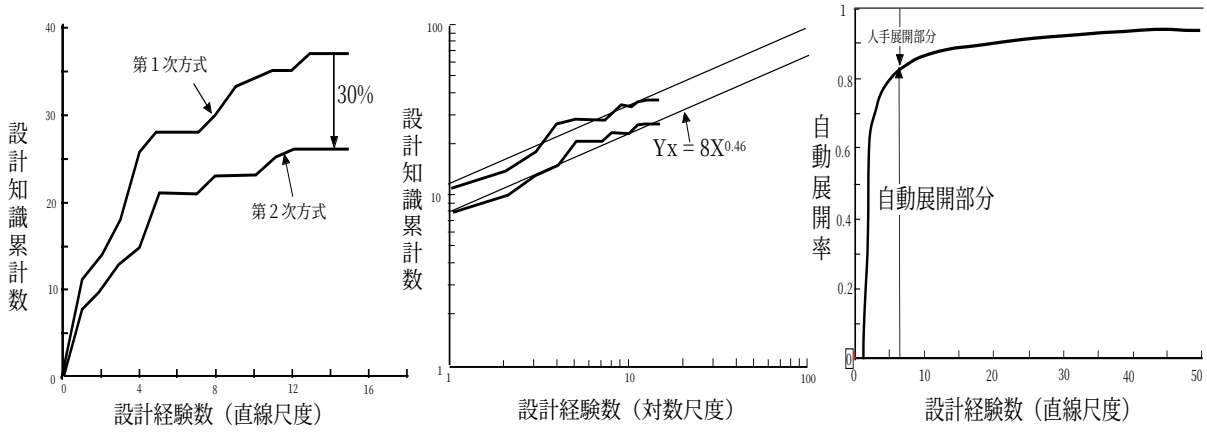

図3.a 習熟特性（直線）　図3.b 習熟特性（両対数）　図３．ｃ自動展開特性

図３　単純に再利用する自動設計の特性

計に相当する第２の様式である．

中心は図４に示す動詞辞書である．ﾃﾞｰﾀﾌﾛｰ図とＰＡＤのスケルトンをデータ，目的語の挿入や「てにをは」を付加する処理をメソッド，とするフレームを動詞の意味毎に持ち，親概念の動詞「○○する」で索引される．通常の設計ではこれのみで子概念を作れる．

入出力帳票は階層的なﾃﾞｰﾀ構造図に変換して与え，設計での機能の階層展開と同期して入出力の両側ﾃﾞｰﾀも階層展開させる．ＪＳＰ形設計の場合には，この機能を用いて複数の子概念を作る．

知的CASEツールでは人が設計ルールを選択した．ここでは「該当する意味の頁の選択」問題になる．実例調査から人の概念展開の特性を掴み，自動選択を行わせ，設計ルールの自動生成を可能にした．

簡単な在庫管理プログラム７本(約700行相当)，更に機能追加～保守設計として７本を追加作成し(合計展開数約100)，これらの設計の再現を確認した．

図５[Abolhassani01b]に評価結果を示す．図の横軸は１４プログラムの設計の進行を展開数で，縦軸には動詞辞書の累計頁数をとり，両対数用紙上に打点した．習熟効果が現れている．２折れ線の傾向線の初めの部分は動詞の増加が中心，後の部分は意味の増加が中心である．

傾向線を延長して習熟性工学の技術で評価すると，７００行程度の設計に要する動詞辞書２０頁があれば，約１０倍/約７Ｋ行の自動設計は約２０頁の追加で可能と見込める．この自動化率は９８％になる．

この方式は，事前準備を要する動詞辞書の必要量が少なく，更に各頁の内容は簡単である．前者は，少数の動詞の各意味毎に準備すれば良くて，習熟特性がある．後者は設計を図形処理に置換え部品化し簡単化したことによる．（ＣＡＳＥツール自体も徹底的な標準化と階層化を図ったので，ｿｰｽ規模は小さい．[山田01他]）

文は人を表すと云い，設計結果から設計者を識ることができる．設計結果は設計者の頭脳構造を反映するとも理解できる．この例のように，知識を細分化し標準化し多数で薄い単機能の階層構造は，実は人の頭脳の構造でもあろうか．ｴｷｽﾊﾟｰﾄは軽い負担で多くのことができる構造なのではなかろうか？

人の設計では設計しつつ記憶していくから，自動生成エンジンの生成結果は再利用エンジンで使われる．これは設計の習熟効果を説明するモデルになる．

ここでは省略するが，スケルトンは設計ルールの繰返し経験から抽象化でき，人の言語能力の獲得と同型で[Abolhassani01a]，更には人の言語学習のみでなく，各種の知識構築の基礎とも考えられる．

第３の「知識によるレベル」も大体が理解できている．内部構造の詳細な研究結果[Abolhassani01a]では，最下位の設計知識として概念辞書[脚注6]があり，上には概念

脚注６．この概念辞書の用語数もまた習熟傾向を示す．

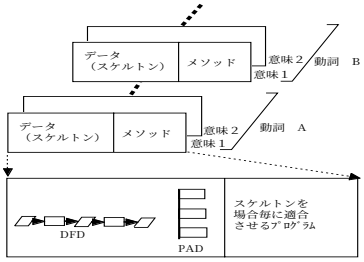

図４　動詞辞書

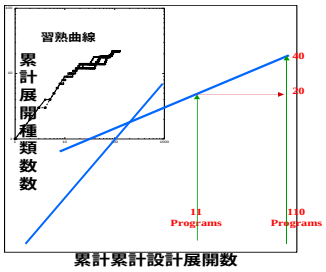

図５　自動設計の特性

辞書を用いて特定のパターンを形成するマイクロデザインルールがある，と考えている．これらを組合せると自動化可能な範囲，組合せパターン数は格段に増える．このように，人の知の本質構造は，各所での深く薄い階層構造にあるように思える．

「設計者の設計知識が判れば，ソフトウエア工学の諸問題が明解になる」と云われる．これまで数例につき，設計知識構造と静的～動的な統計資料を使うと，実感を持って理解できる．今その入口に入った所に過ぎないが，かような逐次近似的な研究を追い続けると，人の教育/訓練/育成から始まる広範囲な疑問への回答が出せそうに思える．単に，設計や製造直接作業のみでなく人のあらゆる意図的な行動が説明でき，更には人の創造過程に迫れるのではなかろうか．

以上から，本章での設計の再現は，階層展開がProcessの内部の知の中心であることの証と考える．

４．ソフトウエア開発工程の外部特性

本章は，人の設計の階層展開モデルによって工程の外部特性の定量評価[Koono96]を説明する．

４．１階層展開網モデル

図１のﾃﾞｰﾀﾌﾛｰ図は，展開の度に３倍[脚注7]に増え，階層的な展開の網状である．自然言語の意味毎に展開すると展開数の平均値は常に約３倍になる．

脚注７．人の自然言語での概念展開の特性，更には人の記憶の特性，基本はニューロンでの機能結合の特性ではなかろうか？ある場合の理論計算ではe=2.7182..が最適になる．

８．Brooksの Mythical man-month は「大規模が大きいｿﾌﾄはそれなりに複雑な作り方をするから低下する」と書いている．彼が指摘していることは，人の設計の特性の問題ではない．

９．ここでの議論は設計の本質特性である．網の中間では，工数/A4換算枚数などの尺度が使える．Function point 法は，設計の標準方式を決めて機能当りの評価を揃える目的で，設計の本質特性ではない．

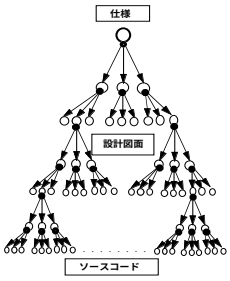

図６ 階層展開網モデル

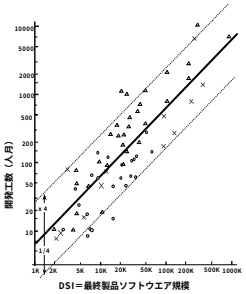

図７ 工数効率の一定性

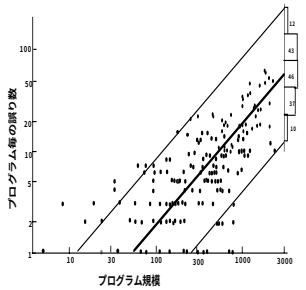

図８ 誤り率の一定性

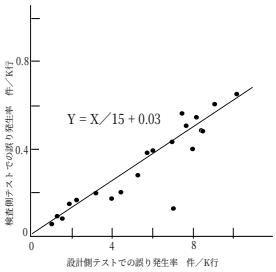

図９ 誤り減衰率の一定性

図６の階層展開網で設計をモデル化する．網は均質な構成とし，展開率ｒは一定，白丸は情報，また黒丸は単位的な知的処理を表す．ｍからｎ段の等比級数として処理の総数を求めてｎ→∞とすると，次式になる．

単位的知的処理総数∝最終出力情報数

ここで，単位知的処理当りの時間を乗ずれば全体工数になるから，上式より

総工数∝（ソースコード行数）[1]

生産性＝総工数／ｿｰｽｺｰﾄﾞ行数＝一定

が導かれる．これは産業界での常識[脚注８，９]「あるチームの生産性はかなり安定している」ことと一致する．

単位知的処理あたりにある小さな率で確率的に誤るとすれば，誤りは網中を伝搬して出力に至る．前式より

総誤り数∝（ｿｰｽｺｰﾄﾞ行数）[1]

誤り率＝総誤り数／ｿｰｽｺｰﾄﾞ行数＝一定

が導かれる．これもまた，産業界での常識と一致する．両者ともＹ∝Ｘ[1]の形で，加法性が成立つ．

４．２実績との照合

前節の理論推計と実績との照合を行う．図７，８[Koono96]に両者を示す．前者は工数に対する図で[Boehm81]のﾃﾞｰﾀ[脚注１０]，後者は誤り数に対する図で[Thayer76]のﾃﾞｰﾀ[脚注１０]，を両対数で再プロットした．

ｼｽﾃﾑ分野や人の特性測定では，環境や条件が複雑な為，条件の整った資料はまず絶対に得られず，たとえ揃えても実際はバラつく．そこで，データの処理に充分な注意を払い[脚注１１]，「バラツキ分だけ結論の統計的な確度が低い」と考える．

バラツキ×３～４倍から１／３～４倍と大きいから信頼度は低いが，理論と実績は合致する．すなわち，階層展開網モデルはprocessの代表的な外部特性，工数と誤り数は作業結果の規模に比例する．

これらは，人の作業での生産性の一定性と誤り率の一定性のメカニズムの説明でもある．

４．３誤りの減少

テストや各種の机上チェックは，誤りを摘出するから残存誤り数は減る．設計者側テストで摘出した誤り率と後続検査で摘出した誤り率の相関を図９[渡辺８２より再プロット]に示す．強い相関が明瞭である．

別の見方では「設計者側はテストで全ての誤りを摘出しても，約１／１５の漏れがある」．テストも机上チェックも誤りを減衰させる減衰器であり，その減衰率がその工程の特性値になる[脚注１２]．

以上で工程の最も基本的な下記３要素の一定性

工数効率（工数／規模），

誤り率（誤り数／規模）および誤りの減衰率

を示した．（この他筆者等は*工数効率および誤り率の習熟特性*を確認した[Koono96].）これらはﾊｰﾄﾞｳｴｱの場合と同一である．IEではこれらを基にして定量的合理的科学的な生産管理を展開した．

５．合理的定量的科学的な管理

IEの手法をｿﾌﾄｳｴｱ開発の管理に用いる例を説明する．管理とは，

合理的に目標を設定し最大効率で達成することである．基本原理として

・工程の外部特性を扱い，内部は対象技術に委ねる

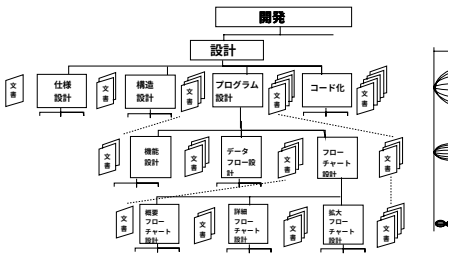

図１０ 工程による統制の強化

脚注１０．大規模ｿﾌﾄｳｴｱ開発が一応の軌道に乗った1970年代の資料である．この時期は，ﾌﾛｰﾁｬｰﾄ位しか支援ツールは無く，殆どが新規設計で，言語も手続きコンパイラー程度と工程の条件が揃い，評価に適したデータである．[Boehm81]は３種の資料群がある．[Thayer76]は，1ﾌﾟﾛｼﾞｪｸﾄで，単位はﾌﾟﾛｸﾞﾗﾑ単位であり，上記より条件が悪く，更に厳密な作り込み数ではない．これらの結果，同程度のバラツキである．

１１．人間信頼性工学などで取られている手法に倣う．経験的に人に関わる指標値は最大で平均値の×３～５倍から×１／３～５倍の広範囲にRayleigh分布状にバラつく．指標値を対数尺度で示せば，平均値を中心とする正規分布状になる．帯状の範囲を超える資料は，異常値として棄却する．打点した図上で，太線が打点群の中央になり，両側の限度線と太線の間に含まれる打点数がほぼ等しく，かつ両側で除外される異常点がほぼ同数になるように調節した．

１２．テストも設計と同様に誤りを免れない為である．

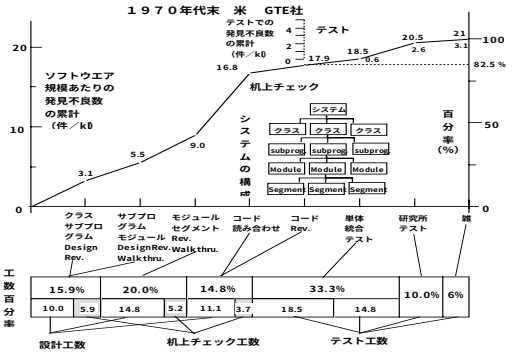

図１１　GTE社開発例

・全て定量化し，実績に準拠する

・*Divide and Conquer*

等の他あらゆる管理手段を動員する．

*Divide and Conquer*の例として，図１０[Koono96]で工程の統制を説明する．左は工程区分が階層的に降る様子を示し，統制が降るにつれバラツキは右のように順次に減る．新しく工程を細分するには，夫々の工程端面の文書/図面（あるいはその一部）を設け，作成法/基準を決め，作業はそれらに従うことを義務付ける．下の工程まで（細部まで）作業方法を規定し，厳しく守られるにほど，バラツキは減る．

定量的計測の好例を図１１[Daly76他より図面化]に示す．中の階層図はｼｽﾃﾑ構成（工程区分），下の棒グラフは工程毎工数の比率を示す．日本の多くのｿﾌﾄｳｪｱ開発組織は，この図のような大区分毎に，工程毎の作業時間，成果物（文書）数，誤り数等を捕捉する．

開発終了後には，大区分単位に工程毎工数効率（工数/A4枚など）や成果物の指標値（ﾁｬｰﾄA4枚数/K行など）などが得られる．IEでは

・工程毎の各種指標　・先行実績　・各種の補正

等により，きめ細かに定量的に管理をする．(COCOMO[Boehm81]は総数で予測等をするから，全体を教える教育には良いが，実用には不十分である．)

数例の利用方法を挙げる．ｺｰﾃﾞｨﾝｸﾞ工程の工数効率は，本来どのチームでも同程度であるべきだ．ﾌﾟﾛｼﾞｪｸﾄ毎の工数効率を調べれば，良いものと悪いものが判る．ｺｰﾃﾞｨﾝｸﾞを展開した下位小工程で，差異を分析すれば，最善の小工程（作業方法）が判る（定量的で合理的な工程の改善）．コーでイング用仕様書を定め，作業方法を統一すれば（問題の再発防止策），工数効率は改善され，そのバラツキも減る．かくて安定になった特性値を基準にする．基準値と個人実績からｺｰﾀﾞｰ毎の格付けができ，ﾁｰﾑやｸﾞﾙｰﾌﾟなど集団毎の格付けもできる．

新ﾌﾟﾛｼﾞｪｸﾄの工程毎の工数計画も過去の実績から（要すれば補正して）決める．以後は予算制度で管理する．計画累計値と累計消費工数の両カーブを対比すれば，計画からの乖離の発生が一目で判る（図式管理）．構成部分毎の対比図から，原因となる異常個所が判り，直ちに対策処置できる．中間段階でも定量管理するから，最終的な適合度は大幅に高まる．工数の他に，期限，品質など多くの管理指標があるが全て同原理である．

向上には次ぎの原則がある．

組織の業務では，管理しないと何の向上も無い．

管理すれば，問題が明確になり，向上策が現れ，

人を含む工程work processを改変すれば向上する．

ｿﾌﾄｳｪｱは極めて実験が容易にできる恵まれた環境であり，これを活かす．担当者は不良分析で自己の欠陥を知って次回作業で改善策を取れば誤り率は低下させ得る．ﾘｰﾀﾞは，自分の運営の何処が悪いかを知り，次の作業をする時に，改善策で運用すれば，適否が判る．（優れた実績を挙げた人は，自分でかような向上のサイクルを回している．これはそのｼｽﾃﾑ化である．）

全員が，たゆみなく，何処までも改善を続ける．この努力を重ねるうちに，事業も成功の機会を掴めて，２章で記したモラール（士気）の高い高成熟組織ができあがる．約１０年ほどの年月が必要．

ここに説明したことの多くは，日本のｿﾌﾄｳｪｱ組織では，かなりの程度，実際に行われている．ただ，明確さに欠け，経験的であり，説明できないことが多い．それゆえ，関係ない人には中々知られることがない．日本の平均的なｿﾌﾄｳｪｱの誤り率は，世界の平均的なそれより，平均的に良い．その良さを支えているのが，このいような経験的で非定型的なprocessである．その背景は，知らぬ内にIEの影響下にある為と考えられる．

日本のｿﾌﾄｳｪｱ組織の内，ﾒｲﾝﾌﾚｰﾑ系では本社の影響を受ける．ﾒｲﾝﾌﾚｰﾑ系のﾊｰﾄﾞの管理はIEが基本，全社的な品質保証はTQMの影響下にある．ﾒｲﾝﾌﾚｰﾑ系ｿﾌﾄｳｪｱ組織は，これらの影響を受け，それは更にユーザ系ｿﾌﾄｳｪｱ組織に及び，日本全体に波及したと考えられる．

その具体的な現れは，日本のﾒｲﾝﾌﾚｰﾑ系組織に，世界で唯一の「ｿﾌﾄｳｪｱ工場[Cusmano91]」群が形成され，高度な自動設計［大野００］が実現している．日本の殆どのｿﾌﾄｳｪｱ組織では文書で区切られた工程から出発する．そこで，CMMのレベル３（工程が定義されたいる）から出発する．これに対し米国では大半がCMMのレベル３以下と云われ，実際に文書化は非常に乏しい．

このように，日本のｿﾌﾄｳｪｱ組織の中には優れたものが存在するが，明確さを欠き，経験的で，説明できない．

本報告で説明した，ｿﾌﾄｳｪｱの設計知識構造と定量性に立脚したIEの流れの管理の体系は，その欠落したバックボーンとなり，更に，その在り方を向上させるものとなるだろう．

６．Industrial Software Engineering

productとprocessがソフトウエアを構成している．いわば，車の両輪で，均衡が取れていることが望まれる．しかし，product の典型が理論であり，process の中核が人と考えると，圧倒的に process が遅れている．

本報告は，最も簡単な設計の基本的な特性（階層的な概念展開）から始め，古典的な逐次近似のアプローチおよび，エキスパートの諸特質から，

・エキスパートの知識の構造
・この内部構造の証としての人に倣った自動設計
・その外部特性が示す基本特性

を示した．

知識構造は，まだ入口の段階で，研究していけば，

・人のソフトウエア設計について多くが解明

されるであろう．

外部特性からは

・ﾊｰﾄﾞｳｴｱと共に生長した管理の体系であるIndustrial Engineering がｿﾌﾄｳｴｱにも適用可能

であることが判った．今後，これは

・ソフトウエアの管理体系明

に発展可能である．

工学は産業～社会に貢献する責務を持つことを思う時，ソフトウエア工学は産業のニーズに触れ，そのニーズに応えうるように，

・productとprocessの両輪が揃ったIndustrial Software Engineering が必要ではなかろうか？

本報告の冒頭の基礎の部分には，現行とは異なる考えが並ぶ結果となった（奇を衒った訳ではないが）．研究対象は，人の意図的な行動を支配する知識の構造，およびその知識の成育過程であると捕らえると，この研究には幾つもの分野に関係するであろう．

ご関心があるなら，これまでの研究材料 / 資料類は必要に応じて開示できる．異見こそ研究を太らせる．共同で討論 / 検討 / 研究する機会が得られれば，大変に幸いである．

謝辞

貴重で高価な教育を与えてご指導頂いた日立製作所の方々，この研究を支えて頂いたB.H. Far教授，陳助教授，Abolhassani博士他の方々に感謝します．

文献

[Abolhassani00]Abolhassani H, Chen H., Far B. H., Koono Z., Software Creation: A Study on the Inside of Human Design Knowledge, Trans IEICE Vol. E83-D, No. 4, pp. 648Å|658, April 2000

[Abolhassani01a] Abolhassani H, Chen H., Far B. H., Koono Z.,Software Creation: Cliche an intermediate knowledge during design, Trans IEICE (Accepted for publication)

[Abolhassanio1b] Abolhassani, H., A study on automatic design based on human design knowlege, Ph.D thesis, Saitama University, 2001

[Boehm81] Boehm, B. W., Software Engineering Economics, Prentice-Hall, 1981

[陳９８] 陳 慧，Far B. H.，河野 善彌，ソフトウェア自動設計における系統的なエキスパートシステムの構築，ー設計工程からの設計知識の獲得と再現ー，人工知能学会誌Vol. 12, No. 4, pp. 616-626,1997 年 7 月

[Chen98] Chen H., Tsutsumi N., Takano H., Koono Z., Software Creation: An Intelligent CASE Tool Featuring Automatic Design for Structured Programming, Trans IEICE Vol. E81-D, No.12, pp. 1439Å|1449, 1998

[Cusmano91] Cusumano M. A,. Japan's software factories: A challenge to U.S. management, Oxford Press, 1991. 邦訳あり．

[Daly74] Daley, E. B., Mnichowitz, D. A., The management of large software development for stored program switching sysytems, Intl. Switching Symposium, 1976.

[Koono96] Koono Z., Chen H. and Far B. H, (64) Expert's Knowledge Structure Explains Software Engineering, Proc. of Joint Conference on Knowledge Based Software Engineering 1996, pp. 193-197, 1996

[師岡 94] 師岡孝次，習熟性工学，建帛社，１９９４

[大野 00] 大野治，降旗由香里，小室彦三，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ﾌﾟﾛｸﾞﾗﾑ用スケルトンの作成とその十分性ー，信学論 Vol. J83-D-I, No. 10, pp. 1055-1069, 2000

[Rasmussen85] 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

[Thayer76] Thayer, T. A. et al., Software reliability study, RADC-TR-238, 1976

[山田 01 他]山田他，ソフトウエアクリエーション：統合知的 CASE ツールに関する３編，信学技報 AI2000-71, 72, 73, pp. 21-32, 2001

[渡辺82]渡辺順平，緒方秀夫，小林英二，ソフトウエアの品質および生産性予測法の１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，pp. 7-14，１９８２

[Zipf72] G.K. Zipf, Human Behavior and the Principle of Least Effort," Hafner Publishing, 1972.