ソフトウェア工学 98－5
（1994．5．25）

ソフトウェア開発プロセスの構造

（Industrial Software Engineering の立場から）

河野善弥　Behrouz H. FAR
埼玉大学　工学部　情報工学科

この報告は，ソフトウエア開発作業の構造についてその特性を外部から捕えて統一的に説明する事を試み，終わりに内部すなわち知的作業で捕え，両者が同じ事に帰すると報告している．人の作業を環境等を含めて「作業工程」として外部的に捕え階層性を前提として作業を外側から見るが，階層展開により幾らでも詳しくできる．作業は繰り返しを考え，資源消費や誤り率などに反復特性とバラツキがでる．適当な原単位をとると，関係する指標値が一定性を示す．これを用い，作業工数や誤りについて統一的に合理的で定量的な計測／計画／監視／評価ができ，進化も説明できる．設計内部の知識を抽出した結果は，上の外側から見える所と一致する．かような考え方はソフトウエア，設計に限らず一般の作業に適用できると思われる．

Structure of software development process

(From a view point of Industrial Software Engineering)

Zenya KOONO and Behrouz H. FAR

Department of Information and Computer Sciences,
Faculty of Engineering, Saitama University

This paper reports on the structure of software development. Human development work is defined as work process, and the external view point of it is taken. From this view point, it is tried to explain all the characteristics rationally. The work process may be repeatable showing the same characteristics with the inherent variations. Taking a suitable basic unit, normalized indices show constancy. Using them, a unified and rational measurement, planning, surveillance and evaluation becomes possible and the maturity is explained as the progress of the work process. Finally, the internal view of design, the human design knowledge extracted based on the systematic extraction and reconstruction of knowledge, shows the same nature. This approach and results thus gained may be applied also in human works in general.

１．はじめに

ソフトウエアは，先端的なプログラム技術があるなど優れて科学的ではあるが，工学や産業として見ると最も遅れた面がある．研究開発投資比率も設備投資比率も少なく，極めて労働集約的な状態にある．このような跛行的な姿は，Engineeringとして科学的／合理的／定量的な取り組みが不足していた，と思われる．

立ち遅れの背景には，取り組むべき対象，即ち作るという事，具体的には

＊人を相手にする，なかんずく

＊人の知的面の取り扱い

の難しさ等にあると思われる．昔，ハードウエア生産でも現在のソフトウエア産業と全く同じ状況にあった．１９世紀末から発展した Industrial Engineering，ＩＥは，合理的科学的な基礎を作ったと言われている．ＩＥは人の作業に着目し繰り返し作業で実験しやすい事を活かし多くの基礎を築き，現在の行動科学，人間工学，認知科学，安全工学，経営管理などに発展した．

ソフトウエアは，初めは小さな存在であった．しかし，それは人の文化を実現し継承する力を持つゆえに，増殖して巨大になった．その過程は多くの繰り返しである．そこで，ＩＥの成果を利用できる．人の作業には習熟が現われ，特に知的作業では扱いが難しくなる．

この報告は，全ての作業に共通な人中心に外からと中から見る事，繰り返しに着目する事などこれまでと異なる進め方での研究した結果[河野92a, 河野]を纏めたものである．

２．基本的な前提

２．１作業工程

人がソフトウエアに関する作業をしている時，人や環境等を含む総体を「作業工程」と名付け，

入力を出力に変換する過程

と定義する．その実体要素は，

＊入力から出力への変換機構

＊変換機構への影響要因（環境等）

＊出力（具体的な後続への入力）

とから成り立つと考える．

２．２　基本的前定

簡単化の為，下記を前提とする．これらは人の概念展開が階層性を持つ事に起因するもので，後章にて再検討する．

Ｉ．対象入力，出力は階層的構成をなしている

作業のある断面で，抽象的な目標，外部機能次いでは機能，設計情報やプログラムは，階層的構成をとる．図２．１の左のように，設計では階層的に展開／詳細化され，テストでは階層的に統合される．（ハードウエア論理装置開発でも同形になる．）

設計で展開を続けていくと，設計対象は次第に具体的，詳細になる．最後には実現手段である論理／数値演算処理，入出力などが直ちに透けて見えるようになり，次にコードに置換する．これらの過程を通じて設計対象／結果は階層性をもっている．

ＩＩ．人の変換作業は，階層的に展開できる

（作業工程も階層的構成をなしている）

人はある作業を行なう為，幾つかの部分作業に階層的に展開して行なう．各部分作業は，それぞれ更に階層的に展開して行なう．展開を続けるに従い，作業は次第に具体性を増し，小さな処理になり，最後は人の単位的な思考や行動に帰着する．そこで図２．２のように作業工程は階層的構成を持つ．（工程の階層性）

３．作業工程の特性と評価

３．１反復再現性とバラツキ

作業工程は反復再現性を持つ．再度同じ入力を与えると，その作業工程は同様な出力を出す．これは変換機構は巨視的に不変なので，充分大きな標本をとれば結果は（大数の法則同様に）収斂する事によると考えられる．

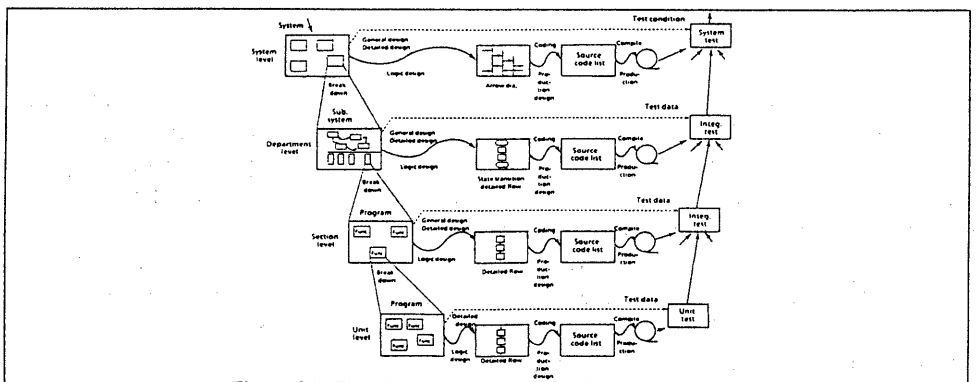

Figure 2.1 Development process net of software

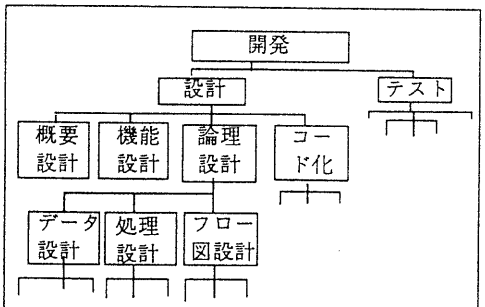

Figure 2.2 Hierarchical work process

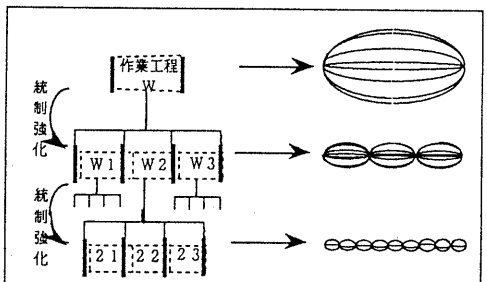

Figure 3.1 Progress of work process

結果には，次の３原因によるバラツキがある．

１．入力，出力条件のバラツキ
（例　曖昧な文書／図面記法）

２．変換機構自体のバラツキ
（例　不統制な作業方法）

３．影響要因のバラツキ
（例　不十分な時間統制）

経験を積み作業工程が熟成するとバラツキが減る．

バラツキ少なくするように変換機構そのものを統制する事は（概念では容易でも）実際には難しい．それは作業の上位概念名は言えても，変換の実体わかり易くは作業手順は一義的には定義できず，階層的に順次下位を定義しなけらばならないからである．通常は作業手順自体よりも図３．１のように

作業工程のより下位までについて
太線のインタフェイス／図面を統制する

事によるのがより有効である．このように下位まで統制するにつれバラツキは減る．図３．１の右はその概念を示す．

環境等の影響要因は，物理的環境からモラール，チームワークなどの精神面迄の広範なものがある．例えば作業の誤り確立ＨＥＰの平均値の最大と最小は約１０倍もある．人間工学ではISO/TC159で作業システムのアーゴノミックス的な一般原則として，次のものを挙げている．

１．人にあった作業場所と設備
人の寸法，姿勢，筋力，動作に適合する事

２．作業がしやすく効率的，健康的な環境
広さ，換気，温湿度，照明／色彩，音響等

３．妥当な負担で可能な工程

これらの物理的な面のみでなく，モラール（士気）が効率を高め，リーダーシップ／チームワークが作業のムラを減らし，連絡／通信が誤りを減らす事等が古くから知られている．開発は知的作業だから，直接作業以上に知的環境，精神神経面の影響が大きい．上記以外にも，

精神的な圧迫感（納期切迫等）を与えない
思考の中断のない集中
速やかな問題解決

等が知られている．ここでは，同一影響要因下で考える事としてその影響を巨視的には見ないで始める事とする．

３．２消費する資源と経過時間

作業工程は作業にあたり，下の資源を消費しまた時間が経過する．

＊「人，金，物」で表わされる経営資源
人：人時間（日／月／年）
金：設備（計算機）使用料，時間毎給料
物：計算機，用紙，文房具

これらには，経験的に下記が知られている．

＊平均すると一定性が現われる．
＊適当な原単位で正規化しかつ平均すると一定性が現われる．それは作業が繰り返しになる事による．

これら科学的合理的な計測／評価／管理／制御の基礎であり，特に後者は指標になる．一定性を確保するには，一定性を示す条件があり，指標にするには繰り返しが条件である事に留意を要する．特に，標準にするには，標準的環境，標準的能力，標準的努力が必要になる．これらが明確でない時は，同一の条件のもとで，繰り返し再現制があると考える．

＊適当な作業効率指標の例

開発初期	上位設計	具体設計
作業工数	作業工数	作業工数
検討課題	文書量	ソース行数

これらを合成した全体も一定性を示すから，設計全体では

総作業工数/(主要作業結果であるソース行数)

は一定性を示す指標になる．このようにある作業工程の指標は，下位の作業工程毎の指標とは異なる場合もあり，内容に充分注意する必要がある．

指標となるには，繰り返しが前提で，作業手順で考えた区分とは一致しない場合もある．

＊テストでの工数効率指標
高品質なら，テスト工数／テスト項目数
悪品質なら，テスト工数／不良数
中間品質なら次の２者
純テスト工数／テスト項目数
不良処置工数／不良数

実質作業に入る為の準備／段取り等の考慮が必要になる．

＊例　初期化時間＋テスト数×マシン時間／回数
初期教育期間　＋

作業個数×作業あたり期間

これらが，科学的合理的な計画／監視／評価の基礎になる．これらは，対象を現実に捕えその中から抽出するものなので，結果として出来あがったプログラム等から求めた弱い相関関係などとは異なり有用性が高いが，科学的合理的な取り扱いがいる．

＊無知　ハードウエア直接作業では，標準時間制が確立され運用されている．その簡単な様子だけを聞けば，ネジ１本当たりｘ分といった標準の時間があって作業するのだ，とわかるであろう．その話を聞くと，知的作業にそんなものがあてはまる訳がない，人を馬鹿にした事をいうな！という向きもあるかもしれない．ＩＥ普及の初期には，直接作業のベテランは全く同じ事を言って凄んだ逸話がある．

＊バラツキの配慮　実績値や標準値での計画にあたては，各種条件で起こるバラツキを考慮に入れて必ずある余裕率をとる必要がある．

計画値の例　[菅野79]

期待値

＝（悲観値＋最可能値×４＋楽観値）／６

実行にあたっては，監視管理指標用を決めて進捗を監視し，計画との齟齬を早期に発見して，対策処置をする．発達した組織ではこれらを上手に行なうので，バラツキが減る．何もしないで実績値とおりになる訳でない．

＊原因と結果の関係　これら指標やその組み合わせ指標は，多くの条件のもとで一定性があらわれる．そこで各種の指標間に色々な関係が出てくる．これらの関係は統計的な事実関係と，原因結果の関係とは注意して区別せねばならない．

管理も不十分で技術も確立していない作業組織では，学校での基礎学力とソフトウエア開発能力との相関等は計測にかからないであろう．でも，相関が見えないといって，どんな学生を採用しても良いと言う事にはならない．

作業工程が完備して標準化の進んだ組織で強い相関が現われたのに，管理不十分な組織では弱い相関しか観測できなかったとしてもそれは原理上当然である．いわゆるプログラミングレベルの改善で顕著な改善がでる発達初期の組織では，躾の大切さを理解できるか疑問であり，Customer Satisfaction を目的とした運動や経営理念が品質に及ぼす影響などは理解の外になる．

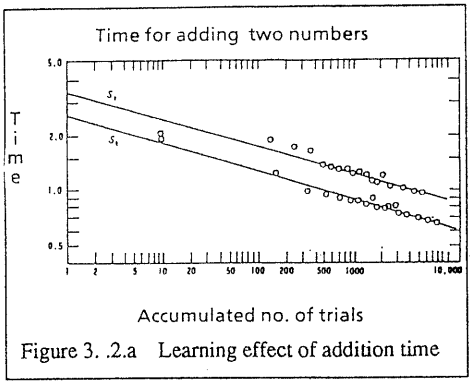

Figure 3. .2.a　Learning effect of addition time

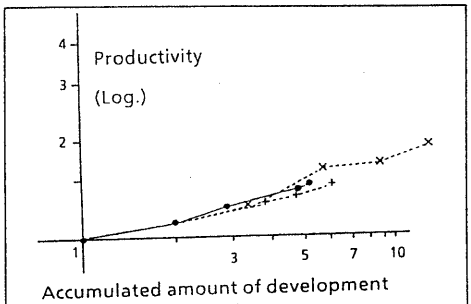

Figure 3.2.b Learning effect of initial development productivity of software

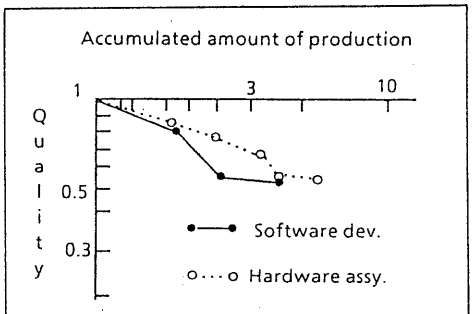

Figure 3. 2. c Learning effect ot error build-in rate

３．３習熟効果

前記の各種特性には習熟効果が現われる．一般に経験が増すにつれて，上手になるとか速度が向上する事を習熟効果という．指標値は初めは速やかに向上する．向上は次第に鈍りはするが向上はいつまでも継続する．運動技能等の向上はその例である．

厳密には習熟効果とは，次の関係がなりたつ事[Salvendy82]をいう．

評価指標＝KX

K：初期値　X：経験量　A：定数

指標値と経験量の関係は両対数用紙上で直線になる．習熟効果は，労力的な直接作業から知的作業（図３．３．ａ）[Blackburn36, Crossman59]，単純作業からLSIや複雑な装置のコスト等の広い範囲で成立つ事が知られている．ソフトウエアでは，初開発時の生産性（図３．２．ｂ[Koono90]）や誤り作り込み率（図３．２．ｃ[Koono88]）等に現われる事がわかった．

ハードウエア直接作業では，単純な繰り返し作業にして分業体制で作業する．通常は充分な習熟状態で作業する．ソフトウエアでは，過去の知識にたより作業する為，分担で複雑な作業をする事が多く，繰り返しが少ない．計測や計画にあたり習熟効果に充分な注意を要する．

3．4 作業工程の品質

品質は作業結果の出力の特性の一種と考えられる．ソフトウエアの品質は単に不良のみでなく，互換性等まで含むように拡張されている．これらは，旧来の不良率のように作業工程の結果の特性から予め条件として作り込む物まで各種ある．

前者の例として論理等の不良取り上げる．これは作成結果にある欠陥が作り込まれている．例えば出荷後の品質指標

不良数／ソフトウエア規模

には一定性があり，なかなかに向上させがたい．この種の不良は変換機構（人中心）の誤りに起因し，出力に欠陥がある状態と考えられる．この種の不良に関しては混乱した報告も多く，ソフトウエアの誤りはmysteryという向きもある．

3．4．1 設計と机上チェックの誤り

この種の不良は作業工程の欠陥により確率的に生ずる．人間信頼性工学[Swain83, 林84]によれば，ある作業の誤りは

単位操作当たりの誤り率(HEP)×単位操作数

で評価できる．これは知的作業でも成り立つ．HEPは小さい値だが作業工程に関する一種の特性で，観測値の平均値ですら最大と最小の比は１０倍に達する程バラツキが大きい．設計での知的な単位的な処理の数をｎとすると，誤り数，また誤り（作り込み）率は

誤り数＝（ｎ×HEP)

誤り（作り込み）率＝(ｎ×HEP)／Ｓ

で表わされる．（ここでソフトウエア規模はＳ行とした．）そこで，HEPの変動は直接に現われる．

設計に続く（机上）チェック（レビュー，インスペクション）も人の知的な変換処理であり，同様に誤りを免れない．チェック後の誤りは設計とチェックの両処理がともに誤る場合で，

(ｎ×HEP)×(ｎ×HEP')

になる．ここにHEP'はチェックでの誤り率である．チェックは一定率で誤りを減少させる性質を持つ．

チェックでの摘出度合いの指標

チェック摘出率＝（1-チェックでの誤り率）

具体的な数値例をあげると，設計全体について

設計での誤り（作り込み）率＝5～100件／KL

チェックでの誤り率＝0.1～0.5（無管理状態）

程度で多くの場合にチェックが立ち遅れている．

以上を組み合わせ，品質向上の為の作業工程を計画できる．大別して，誤り率自体の低下，チェックの強化と工程の分割強化になる．

＊誤り率HEPの低下
＊チェックの強化
　チェック後の誤り率
　　チェックなし　　(ｎ×HEP)
　　チェック１回　　(ｎ×HEP)　(ｎ×HEP')
　　．．．．．．．．．
　　チェックＣ回　　(ｎ×HEP)　(ｎ×HEP')
＊工程の分割強化
　チェック後の誤り率（チェック１回の時）
　　分割なし　　(ｎ×HEP)　(ｎ×HEP')
　　２分割　　(ｎ×HEP)　(ｎ×HEP')／2
　　．．．．．．．．．
　　M分割　　(ｎ×HEP)　(ｎ×HEP')／M

あとの２者は，繰り返し（多面的）チェックが有効である事，チェックは小さな区分に分けてそれぞれ毎に確実に行なうのが有効な事を示す．それは図面の規定が詳細化し強化される事，図３．１のように作業工程が階層的に細分化され，区分点の絞りが強くなっていく事と理解できる．

以上から，人の誤り関係の計測には次の配慮が必要である．

＊充分大きな，同一条件のサンプルを平均
＊誤り作り込みとチェックの分離
＊作業方法／条件の統制
＊繰り返し習熟の影響

これらの配慮を払えば，誤りの計測結果は安定になり，「ソフトウエアの誤りはmystery」ではなくなる．かように安定させてから作業工程の区分を多くしかつ証拠を残す（小さな作業工程毎の図面／文書を作る）事にすれば，何処で誤ったか？なぜ誤ったか？が追跡できる．作業現場でを合理的，定量的，科学的に自分自身で作業工程を改善向上させていくのが Software Total Quality Copntrol であり，日本のソフトウエア産業はかような裸足のSoftware Engineering People により支えられている．

3．4．2 テストでの誤り[Koono87, 93]

テスト作業は人の誤りを免れないテスト設計の結果と，通常の設計結果との比較照合で誤りを免れない．単純化すると，開発終了後の残留誤り率は

(HEP)　(HEP')　(HEP")

に比例する．ここにHEP"テストの誤り率で，

テスト全体での誤り率＝0.1～0.3

程度である．人が介在する限り，テストは誤りの減衰器に過ぎず，誤りを零にするものでない．

一つのソフトウエア開発自体が既にN Version program で指摘している信頼度構成上の多重化構成が可能で，作業工程の編成と作業方法を工夫して単一システムでも高い品質が達成できる．

テストは多数の単位テスト作業で構成される．一連のテスト全体で誤りは１／（数～１０）程度に減衰する．個々のテストは，等比級数的な小さな一定減衰量の誤り減衰器になる．定量的な実績測定からテストの有効度，テスト密度などを定量的に求める事ができる．[河野93a]（熟練者はこれを経験で変分処理している．）従って品質を工程／文書構成と併せて定量的に計画し制御できる．[koono94a,b]更に

設計方法やテスト方法も有効度に関係するので，現在検討している．

不良は作業工程の欠陥から生ずる．そこで，不良の原因と結果を求めると，あるべき作業工程がわかる．作業工程を細分化（文書／図面の強化）し，作業工程のレベルが高くなる程，微細高度な事が理解される．不良分析／再発防止策の立案は定量的な計測／評価をも併用しつつ，科学的合理的に原因と結果を追及するのでSoftware Engineeringにとって，最高の実践の場である．

4．作業の計画と改善による進歩

4．1 計画

作業工程を工数，品質などあわせて，構成を決め定量的に計画／実行／評価／改善する事ができる．図4．1は現状の品質（誤り率）を一覧するレベル図の例である．

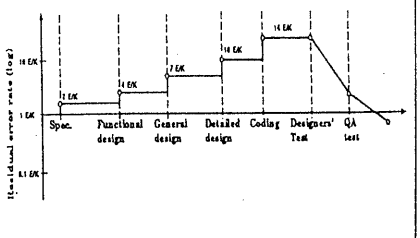

Figure 4.1 Error level diagarm

ある開発と同じ外部条件，対象，要因構成，作業区分，作業組織と陣容であるなら，多くの指標は保存される．図4．1で新しい計画の為，ある作業工程での誤りを減らしたければ，現状の作業工程を階層的にM分割してC回のチェックを行なわせる等の構成変更を考える．そこから新しい図面が定められ（あるいは図面の細部の段階毎の）チェック方式等がきまる．チェックを強化した分だけ設計作業に割当る人時間や期間は増やし，テスト期には摘出誤りが減るから不良探索修正の工数が減らせる．

この計画は資源である人を要する．作業毎の期間／作業工数等を山積表に纏めてから山崩しをして全体を平潤化し，グループ毎／個人毎の作業を調整して資源計画を纏める．新作業で期待する効率／品質が出るように，教育／動機づけも計画する．作業が計画とおりの進行であるかは各種の指標で監視し，異常なら対抗処置をとる．

これらの定量的な計画／監視／評価に必要な特性値，指標がmetricsとして重要である．従来直接作業で管理に使われた工数計画管理や進捗監視や統制は大枠を残し，知的作業の特性を加味した修正を加える．階層的な作業工程の分割と併せるとPERTやarrow diagramは合理的な管理ツールである．従来繰り返し中心で作業工程を，毎回の実績値中心に変分で，ほぼ定量的に，変更分を求める実用的方法で確度と精度の高い綿密な計画をたて実行している．それらは，一種のノウハウであるが，汎用的な尺度と方法に纏め，技術として公になる事が望まれる．それは，従来の直接作業を含みより拡張した全てのビジネスプロセスも含める新しい管理工学になろう．細分化された工程で何を作業しているか，何がそれに影響するかを知れば，銀の弾丸が無いか否かの議論は起きる余地もない．

4．2 改善と進歩 [Koono88]

Engineeringは常に進歩が求められ，系統的な改善方法が必要になる．進歩は経営資源など作業工程の経済性面と品質のような特性値の2分野がある．

作業工数等は作業工程と対応する．階層的に展開した図を作り，夫々に所要工数等着目尺度での数値を記入し，可視化する．以後の改善の計画の仕方はハードウエア生産技術者の常道，ハードウエア設計者の原価低減検討と同じで，Value Engineering (Value Analysis)の考えが使える．（無駄むら無理の追放から始まり，）ある作業についてより効率的な方法（他の作業工程）でその目的を達成する／代替する方法はないか？を探索し，評価して採否を決定する．この為には監視用に測定されている指標値では巨視的な事しかわからず，実作業時の詳細記録（実績資料管理）が重要になる．ソフトウエア生産技術が定量評価の上に立脚する事が望まれる．

品質の改善手法は，SWTQCで既に尽くされている．主たる方法として，これまでパレートの法則に従い，局所に不良が集中する事等定量評価による事，作業工程を文書／図面で区分しその現場証拠をもとに原因探索する事，クレーム／異常現象／障害メカニズム／不良箇所／作り込み工程と実証的に犯人を追及する（ナゼナゼ問答）方法，上記と同様な問題を起こす作業工程の目的を達成する別法／代替工程／悪影響排除等の技術上の基本的方法がある．人間中心の教育／動機づけ／教育には小集団活動／目標管理／発表と聴講教育など管理技術の粋が凝らされている．現場の指導で実績実証は既になされている．それを整理し，全体の関係を万人に合理的にわからせる体系化が今後の課題になる．

改善は作業工程を構成する人の改善でもある．開発集団の熟成過程は，ＩＳＯ９０００といった仕掛けだけのもの，Software Engineering Groupのみで達成できる技術だけの問題ではない．集団は一人の人に似た特性を示すが，その熟成／進歩過程は小学生から大学生への進歩と同様なものと理解する方がわかり易い．例えばCASE toolなどは進化の程度の低い組織では危険であると言う．言語概念の展開が正確にでき，ファイル等が正しく階層化れていないと構造化分析はできないし，図面を繰り返しチェックして更新を繰り返すリーダーと誤りに関する原因と

結果の知識が無ければ品質向上の効果が達成できず，繰り返し作業の中で標準機能，標準構造，標準手順を作り上げた組織でないと設計結果の再利用ができず導入の効果が上がらない．

小学生から大学生になるまでの教育と勉強の努により進歩するように，現実の作業の中で定量化／計測評価／原因と結果の関係追及／製品と作業工程の進化の努力を続け成果を挙げる事により進歩する．そこで必要な事は，短期的なプロジェクト管理ではなく，長期的教育育成論であるだろう．

5．作業工程の内部構造[河野94c, Far94]

5．1 開発対象の知識の構造

これまで主として作業工程を外部から見た．その内部につき研究の結果の概要を記す．2章の階層性の前提は実務上有益と認められている仮定である．これに基づけば次のようにして設計作業の知的処理結果を再現できる．

＊ある段階の文書／図面は設計対象の知識をもつ．
＊あい隣りあう文書／図面間の差は設計変換の結果，一種の設計過程の知識，を示す．
＊2文書／図面間の差を小さくして，単位的な設計過程の知識が階層的展開ルールとして得られる．
＊上記で図面相互間のミクロな変換が取り出せるので，図面全体を階層的に再構成する．この過程の知識は，階層構成のより上位の設計過程の知識になる．（このように再現して行くと，上位構造は人の意識する作業工程と一致する．

図5．1にその例を示す．図の下部は設計図面を階層的に展開する知識，入力のある部分を出力のシンボル群に変換する知識になる．（我々はこれを設計ルールと呼んでいる．）全体の設計はかような連鎖なので，水平方向にその連鎖を取り出せば，データや機能の大きな階層状展開をなした知識である．これは人にとって最も親和性のある構成であり，誤りを減らす働きがある．また，階層的な作業工程はこれに直交するが同様であり，階層性により人の記憶を助け作業を容易ならしめる．再現した設計知識は図5．1の左と中央のように，作業工程と同じ構成を持つ．作業工程は如何にするかが中心であるが，再現結果は実施例での動きの記述になる．

最下位は階層的展開（設計ルールによる変換）の知識になる．

色々な例を見るとその全体は

＊Liskov の指摘したデータの階層的展開
＊Jackson program development の機能の階層的展開
＊Myers のComposite design の機能とデータの交互の階層的展開

などが現われ，設計のモードの制御がある．

更に上位の特徴的な知識では，上記のようなミクロな過程から方式設計，詳細設計といった作業工程に現われる設計過程を状態遷移状に制御する知識になる．（これは変換等の結果に着目すると体系的組織的に設計知識をKnowledge Engineer なしで抽出できた事を意味する．）再現した各部は図の右のようなデータフローと制御フローで表わされる．（従ってエキスパートシステムを体系的に構成できる．）各部はExtended Finite State Machine でモデル化できる．これは，Minsky が心の社会[Minsky85]で指摘しているagent 群の構成と一致し，また上位は作業工程毎に分業する人の社会とも一致する．人の社会では，心の社会と同じ構造構成で機能していると考えられる．そのような訳で，階層的な作業工程が人に良いと意識されるのであろう．

そこで，これまで列挙した多くの関係は，人の頭脳の深い構造／構成の反映と考えられ，例をソフトウエアにとりはしたが，一般の人の合目的的な行動一般の姿を暗示するものではないであろうか？このような相似性が，人の作業の特性の共通性に現われたのではなかろうか？

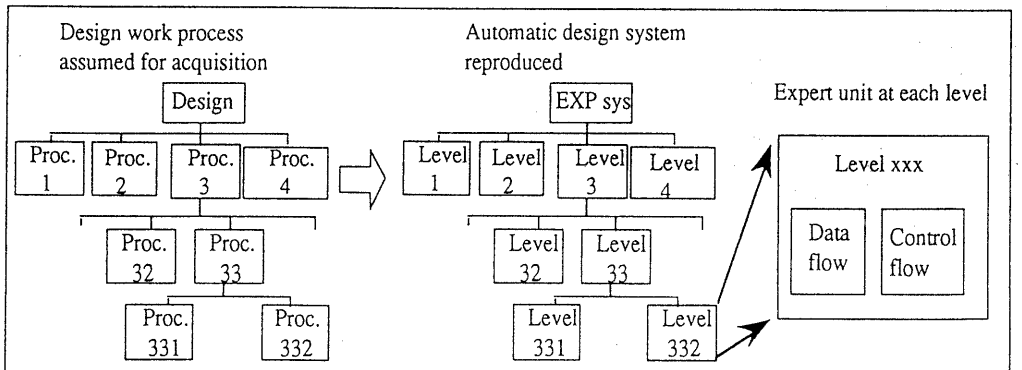

Figure 4.3 Extracted knowledge architecture

6．結び

ソフトウエア開発作業について，基本的な階層性の前提をおき，作業を外から見る立場から各種の特性を合理的に説明する事を試みた．

ハードウエア直接作業を合理的科学的なものにした Industrial Engineering 諸原理／関係を初め，多くの分野の人の知識を総合するとソフトウエア作業を統一的／定量的／科学的に計測／計画／実行／監視／評価／進歩させられる．

ソフトウエア開発作業の内部，知的領域の構成構造を説明した．説明した方法によれば，作業結果から再現するから痕跡を追う事になるが，Knowledge Engineer によらず体系的組織的に誰にでも知識を取りだし，組織的にエキスパートシステムに再構成できる．

その結果は，心の社会と相似構造で人の社会が構成され機能していると考えられ，初めにおいた前提を裏付けるものと考えられる．そこで，これらの結果一般は単にソフトウエア作業でなく，一般の人間の作業を説明するものでありはいないだろうか？

これらを通じて，次のような事が考えられる．ソフトウエアの最大の問題は人であるなら，人に取り組む事で答が得られるのではなかろうか？Engineering は現実に効果をもたらす事で存在が許される．現実の中から合理的に現実を説明するものを取りだし，有用なものに仕上げて行く事こそ科学的ではなかろうか？日本のソフトウエア産業を支える定量的合理的科学的に現実の作業を改善し実効をあげている裸足の Software Engineering People の成果を体系化する事が学問なのではなかろうか？現実に立脚した Industrial Engineering があって良いのではなかろうか？

謝辞

本報告は，筆者らがハードウエア開発や直接作業ソフトウエア開発について，多くの実践者各位から受けた教育に基づいている．特に Industrial Engineeringを初めとする関係領域については，多くの方から示唆を受けた．また，５章については本学の学生諸君の研究による所である．筆者の一人は日立製作所で永年の実践と教育の機会をえ，また上長，先輩，同僚，第一線の諸君に多くの事を教えられた．これらの各位に深く感謝します．

参考文献

[Blackburn 36] Blackburn, J. M. : Acquisition of skill : An analysis of learning curves, IHRB Report, 1936.
[Crossman 59] Crossman, E. R. F. W. : A theory of the acquisition of speed-skill, Ergonomics, 1959, 2, 153-166.
[Far94] Far, B. H., 田中，河野：ソフトウエア設計における設計知識の体系的な構築法，１９９４年人工知能学会全国大会，１９９４．（発表予定）
[林84] 林喜男，"人間信頼性工学"，海文堂，1984.
[菅野79] 菅野，ソフトウエアエンジニアリング，日科技出版社，１９７９．
[Koono87] Koono, Z., Ashihara, K., and Soga, M., "Structural Way of Thinking as Applied to Development", IEEE/IEICE Global Telecommunications Conference 1987.
[Koono88] Koono, Z., Igawa, K. and Soga, M. : Structural Way of Thinking as Applied to improvement process, IEEE Global Telecommunications Conference, 1988.
[Koono90] Koono, Z., Tsuji, H. and Soga, M., "Structural Way of Thinking as Applied to Productivity", IEEE International Conference on Communications 1990.
[河野92a] 河野，ファー：ソフトウエア開発工程の定量的取り扱い，電子情報通信学会 技術報告，KBSE 92-33 (1992-11),
[河野 93a] 河野，大坪：ソフトウエアの誤りと除去の評価，情報処理学会，ソフトウエア工学９５－５，（１９９３．１１．３０）
[Koono94a] Koono, Z. and Far, B. H. :Systematic approach for software documents and work processes, IEEE/IEICE QAMC Workshop'94, Hakone, April 1994.
[Koono94b] Koono, Z. and Far, B. H. :Structural way of thinking attaining reliable software, IEEE International Conference on Communications '94, 1994.
[河野94c] 河野，Far，杉本：設計知識の系統的な獲得，１９９４年人工知能学会全国大会，１９９４．（発表予定）
[Minsky85] Minsky, M. :The society of mind,
[Salvendy82] Salvendy, G. Ed. : Handbook of Industrial