ソフトウェア工学 114－5
（1997. 5. 23）

人の設計知識の構造とソフトウエア工学

河野 善彌　陳 慧　ベルーズ. H. ファー

埼玉大学工学部情報システム工学科
〒338 埼玉県浦和市下大久保 255
Tel : 048-858-3487, FAX: 048-858-3716
e-mail: koono@cit.ics.saitama-u.ac.jp

あらまし　この報告はソフトウエア工学の基礎を設計の知識の観点から与える試みである．設計の基本構造として知的処理の階層展開網モデルを提案し，実績でこれを裏付けた．これはにより開発の定量評価できる．高い成熟度の開発組織を説明し，これを階層知識体系でモデル化できる事，またそれから系統的に知識を獲得し再現できる事を説明し，その知識の概要を述べている．これらはソフトウエア工学での定量化，設計の構造の理解や研究，自動設計等に役立つ．

Human Design Knowledge and Software Engineering

Zenya Koono　Hui Chen　Behrouz H. FAR

Department of Information and Computer Science,
Faculty of Engineering, Saitama University
255 Shimo-okubo, Urawa 338, Saitama, Japan
Phone: +81-48-858-3487, FAX: +81-48-858-3716
e-mail: koono@cit.ics.saitama-u.ac.jp

Abstract　This paper reports on a trial for establishing a basis of Software Engineering from a view point of design knowledge.　A hierarchical expanding network is prop osed for a mode of design, and it is verified by actual data. It enables quantitative evaluation of developments. A highest maturity software organization is explained, and it is modeled by a hierarchical design knowledge system. The design knowledge is explained, and it may be acquired systematically and also reproduced systematically. These technique enable to make a quantitative evaluation, and structural research of software developments.

1．初めに

この論文は，ソフトウエア工学の対象課題である設計を最も基礎的な人の思考作業のレベルのモデルで捕らえ，ソフトウエア工学を基礎から統一的に構成する試みの初めである．

製品の「機能」「価格」「納期」や「品質」は生産上の重要な技術ポイントで，産業を先導するソフトウエア工学は，生産過程(Process)を制御してこれらの目標を達成する科学的な基礎を求められている．現在のソフトウエア工学はプログラム等(Product)を捕え純論理的に処理する事が中心である．そこで，企業では価格や納期の管理を経験的方法で行なっている．それはハードウエア産業の科学的基礎を与えたIndustrial Engineeringの流れを汲む科学的方法ではあるが，ソフトウエアでの適用根拠は明確でない．他方Total Quality Control（ＴＱＣ）は科学的で合理的な進歩方法を提供し，ソフトウエアでも大きな実績効果を挙げている．しかし，ＴＱＣは改善方法が中心で，ソフトウエアのProcessの観点からの充分な説明はなされていない．

本論文は，人の単位的な知的処理から初め，主要な基礎を構築する事を試みる．誤り[河野９３]や工数[河野９２]，これらをProcessで統合[河野９４ｂ]するボトムアップな研究の終着点になる．

2．設計の基礎

2．１設計作業の性質[河野９３]

設計の軌跡である設計図面を調べ，設計作業の性質を調べる．図１は時計のプログラムの設計を示し，データフロー図，フローチャート更にソースコードへと設計が進む．データフロー図を見ると，外部的な機能「時計」は，「時刻を得る」「時刻表示を求める」「表示する」と階層展開され，夫々は更に階層展開を繰返す．並行してデータも繰返し階層展開される．繰返す内に「何をするのか」が次第に明らかになり，個々の機能があるデータの演算に対応する事が透けて見え，ソースコードに変換する．

これより上位のシステム設計等では，経営上の目標や使用者の意図等から出発し，階層展開を繰り返してプログラムのレベルに至る．これより，

＊設計は，ある概念を階層展開する事を繰返しつつ次第に具体化し，明確化し，詳細化し，最後に実現手段に変換する事である．（これは総ての設計，人の意図的行動，に共通．）

自然言語によらない（例えば数学的な問題の）設計過程でも，設計の入力概念と出力概念の関係は階層展開である．

階層展開になるのは人の特性による．

＊人の概念は階層性を持ち，
＊展開の都度，より具体・詳細になる，
＊長時間記憶の大部分は階層的な意味記憶，
＊長時間記憶は概念の塊（チャンク）を単位とし，最大展開数は７以下にするがよい．

この階層性を破る設計もありえるし，また破っても直ちに破綻する訳ではない．階層展開によれば人の概念構成や記憶特性によく合致し，理解し易く，誤りが減り，良いProcessになる．人の特性

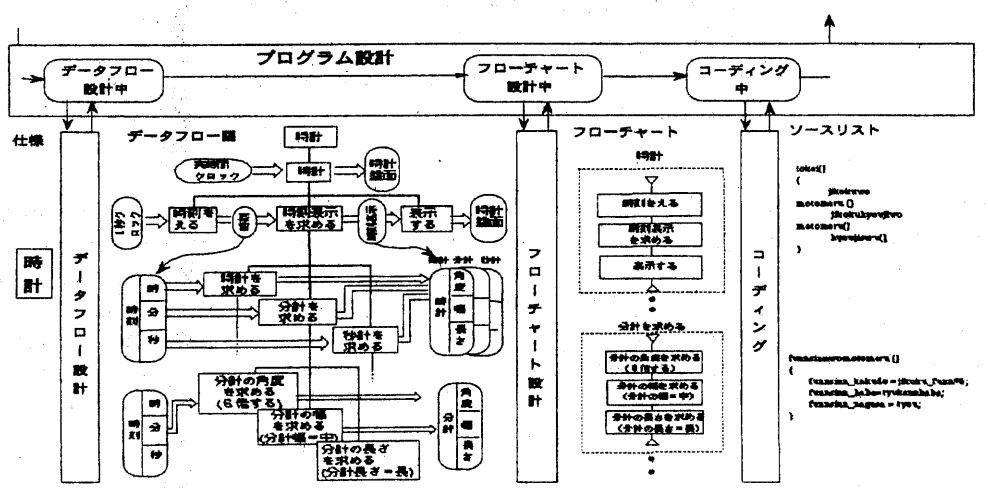


図１　時計プログラムの設計過程

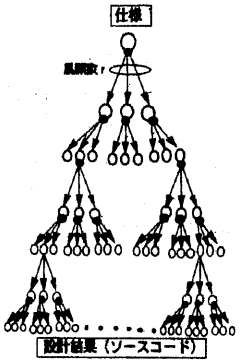


図2　設計の階層展開網

にあう構造を明確にする事がソフトウエア工学のかなめであろう．

2．2設計の特性

設計の知的処理を図2[河野９５，Koono96]の階層展開網で代表させる．図の白丸は設計の入力や結果の設計情報である概念，黒丸は単位的な知的処理を表し，各処理の展開数ｒ＝一定（図１の例では３．ｅ＝2.718らしく研究中．），ｎ段の展開と仮定する．

このモデルの特性を調べる．展開網は等比級数的であり，以下の計算ができる．

網中の知的処理数＝ $(r^n-1)/(r-1)$　(1)

最終結果の数＝ r^n　(2)

知的処理総数／最終結果数比

＝ $(1-r^{-n})/(r-1) \fallingdotseq 1/(r-1)$ ＝一定　(3)

単位知的処理の特性を与えると，３式は設計全体の特性値を与える．これを理論値として，実績と照合する．作業の誤りや作業効率は保有知識や作業方法により大きく変わる．そこで，作業の所要時間と誤りについて，

・一応の習熟者が作業する標準的な場合の実績と照合する．

大数の法則より

・「知的処理あたりの所要時間を一定」

と考え，３式より下の結果を得る．

「全作業時間(工数)は最終結果数に比例する」

「最終結果数当たりの設計工数

＝設計生産性＝一定」．

後にこれを実績と照合する．

誤りについても同様に，大数の法則より

・「知的処理あたりの誤り率を一定」

と考え，３式より，

「誤り総数は最終結果数に比例する」

「最終結果数当たりの誤り数

＝誤り作込率＝一定」．

誤り作込み率が最終結果数（ソフトウエア規模）によらず一定な事は，産業界では経験的に認めている．しかし，これも実績と照合する．

図３は最終結果数（設計の複雑度）と総工数の関係の実績を，また図４は同じく設計の複雑度と総誤り数の関係の実績を示す．広い範囲を示す為に両対数で表示した．先に示した諸式は設計一般に成り立つので，図ａはソフトウエア開発について，図ｂはハードウエア論理設計につき示している．人の作業の特性値はnから１／n倍の広範な範囲にバラつく．ここでは検証の目的にてらし，

・新設計中心で流用が少ない，

・機械力が余り用いられず人の作業が中心，

・ある程度の標準的な能力レベル，

・世界各国，各種機関の発表，

を採取して使用する．（詳細は図に添えた．）

一般傾向を得る為の各種の実例を集めるからバラツキが大きい事は避けられない．問題のポイントは「$Y=X^{+1}$の傾向であるか」なので，プロット結果の中央にX^{+1}の太実線を，またこれに並行する定数倍の補助線を点線で示した．

これら４枚の図によると，

総工数や誤り総数の実績は，勾配X^{+1}の

中心傾向線のNから1/N倍の傾向線の範囲内にある．これから

・設計は階層展開する知的処理，

・生産性や誤り率は規模にかかわらず一定，

は一応の適合性があると考えて良いであろう．

2．3工程による管理[河野９４a]

全ての管理は，対象を細分して特定し，定量的に行なう事が原則とされる．この為に設計作業を区分したい．しかし，設計過程は，外見的には色彩が緩やかに移り変わるものの，特性的には殆ど連続的な微小変換の連鎖であり，実現手段への変換点以外には特性的な断面は無い．そこで，各個に断面を定義し断面間の作業を「工程」と名付け管理に用いる．研究，開発，設計等がそれで，設計の中では，

概要設計，詳細設計，コード化

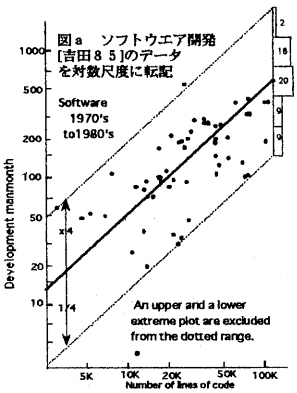
等の意味的な工程名称を与える．この断面は階層の下位まで守る事とし，詳細設計を更に

データフロー設計とフローチャート設計

等に細区分する．

工程も人の概念であり階層展開する．そこで，次の性質が現れる．

＊工程の階層展開を繰返すにつれ，作業は具体化され，詳細化され，最後には人の単位的知的作業になる．（工程は設計作業の知識）



注1　図の傾向線は
Y＝aX ＋b.
太線は分布の中央を，細線は
上限と下限に選び，極端な値
は範囲外とした.

2　ソフトウエアは開発全体
工数．ハードウエアは設計工
数であり，この時期の設計工
数は開発工数の約1/2であっ
た.

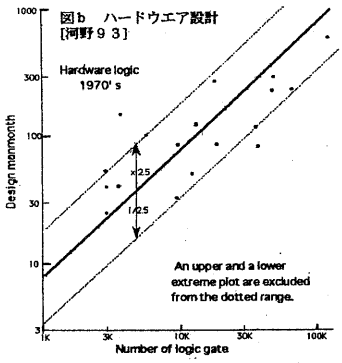
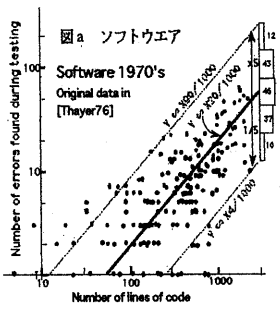


図３，４に共通な注記
'70年代中期～'80
年代初期は大規模開発
が行われはじめた技術
開発の初期で，
・初期段階を終え各機
関とも一応の経験をし
・フローチャートやゲ
ート論理回路図が中心
のツールで，他は初等
的な機械支援しかない
，世界中がほぼ同一方
法で作業していた.

図３　設計の複雑度と工数の関係



注1　ソフトウエアのデータ
はテストでの摘出数である.
テスト摘出数
＝作込み数－チェック除去数
各点はプログラム毎の値.
太傾向線は大体の中央値.

2　ハードウエアのデータは
プロジェクト毎の総計値.
作込み値は，設計図の一応の
出来上がり以後，机上チェッ
ク，テスト，使用開始後等の
総計値であり，特性的な値で
ある.

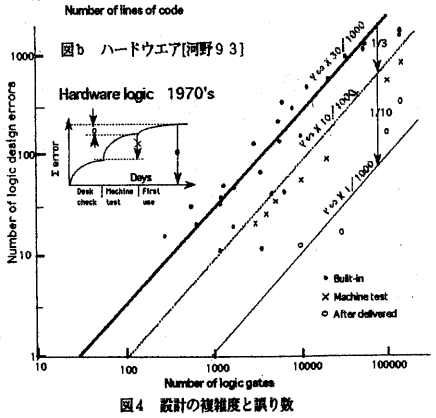


図4　設計の複雑度と誤り数

これでは展開の最終末端まで行かないと正確な定義ができない．工程を厳密に定義するには下記による.

＊工程を正確に定義するには，上下の全階層で断面を同一にして，工程の断面を指定する．これは如何様にでも厳密かつ詳細にできる.

＊設計の流れの断面での設計文書を当該設計工程の（設計全体では中間の）産物とする．具体的には設計文書の記載項目，記述のレベル，記法等を規定する.

図５はこの様相を示し，階層的工程を採ると設計文書体系もまた階層的になり，

＊工程体系と文書体系は互いに相補的である.

工程全体の規定を厳しくするには，

＊スナップショット的に多くの中間点をとる，工程階層を深くし，最下層を小単位にする.

＊各点の端面を詳細で厳密にの規定.

＊全員に規定を厳守させる.

図の右はバラツキの減少を示す模式図である.

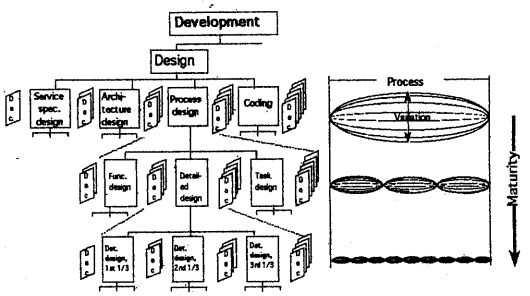


図5　階層的工程とそれを分断する階層的な設計文書体系

２．４習熟効果

ソフトウエア開発集組織には成熟効果がある事が昔から知られている．表１は成熟過程の例を示す．成熟は人の集団一般に現れる．表のBurnsのモデルは組織論の例である.

表１　各種の成熟過程モデル

INPUT社の成熟モデル

混沌→管理された→効率追及→品質追及→創造指向

CMU. SEIの成熟モデル

初期 → 再現可能 → 定義済み → 管理 → 最適化

Burnsの組織形態進化論

離散状 → 階層組織 → マトリックス状 → 網状

成熟する様相を各モデルの見方で説明する．組織の出発時点では，技術レベルは低く，各人の考えは統一されずバラバラで，作業方法が異なる等「渾沌」状態である．作業方法や各種インタフェイスが揃ってくる，すなわち工程が「管理される」につれ，作業とその指標が「再現」性を持ちはじめる．工程が「定義」されると，バラバラな個人プレーでなく，「階層組織」が確立され，階層的な命令報告系統や規律が確立していく．かように進歩すると作るのに精一杯，赤字退治の段階を脱して「効率」や「品質」が追及され，これが作業改善を加速し作業が均質化して行く．かくするうちに構成員の技術が高度化し，相互のコミュニケーションが盛んになり，モラールも向上する（人格的に向上する）．結果としてバグや改善のノルマに追われるレベルを脱却し，良い製品を出すから事業も隆盛に向かい，自由な「創造」指向などの「最適化」が顕著になる．

高成熟度では下記が顕著である．

＊高度な技術水準，

＊多面的で高度な標準化，（バラツキが小）

＊総てに均質な頭脳・知識．

ソフトウエア面では品質が極めて高い．それは入念な文書化と技術的に正しい記述と正確で明解な日本語，徹底したレビューと定量的な評価，またこれらが技術の理解や見解の統一に役立つ等の相互作用をもたらしている事が理解できる．

最低から最高まで成熟するには絶えざる改善を積重ね（技術者がエキスパートになるのと同じく）約１０年を要するという．改善は，管理指標値（生産性など）をより高める方法を求め，全員の作業（頭脳）をこれに統一する．企業は際限なく利益を増大させ，その状態の永続を求める．その結果，

＊高成熟度組織は多くの指標で優れた集団，

になりExcellent Organizationと言える．

管理指標値で見ると成熟過程には習熟効果が現れる．改善し続けると，（スポーツやゲームのスコアの向上と同じく）指標値は初めは急激に向上するが向上の程度は次第に低下し，しかし飽和はせずに何時までもなだらかに向上が続く．横軸に累計経験量を，また縦軸に指標値をとり，両対数尺度上にプロットすると，その傾向線は直線になる．これを対数習熟効果と名付け，人の知的作業効率，ハードウエア製造（直接）作業時間，更に飛行機や半導体の製造コスト等の企業レベルで成り立つ事が知られ，計画，予測や評価に用いられている．図６[Koono90]は人中心のソフトウエア開発の生産性の習熟効果を示し，各種機関，各種の実績を正規化した結果，何れも同傾向となった．（図は新設計部のみ，言語や作業が揃っていて設計支援系の発達初期である'80年代の特性例．）企業的評価として出荷量等を取ると，繰返し利用や各種の流用，作業の自動化等が影響し，より大きな勾配になる．

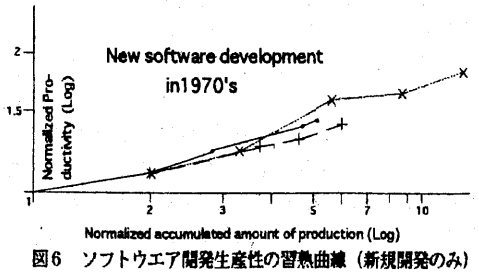


図６　ソフトウエア開発生産性の習熟曲線（新規開発のみ）

２．５誤りの合理的・定量的な把握

ソフトウエアの誤りの問題は謎であると人を嘆かせる．例えば図４のａは大きなバラツキを示し，何が何だか判らない．しかし，図ｂのバラツキは格段に小さい．（データの詳細は図に併記．）これらの研究結果[河野９３]を下に纏める．

図４ｂの黒丸印は「誤りの作込み数」を示し，図aのプロットに比べバラツキは格段に小さい．

＊作込み誤りのみならバラツキは小さい．

図bのX印は「実機テストでの検出数」であり，これは傾向線を見ると作込み数の１／３である．

・実機テストに先立つ机上チェックは誤りを１／３に抑圧している，

このテストの後使用者に引渡しの後に検出された誤り数（実機テストの漏れ）は図に○印で表わされてX印の傾向線（実機テストへの入力数）の１／１０に当たり，

・実機テストは誤りを１／１０に抑圧する

事を示す．この時代に一連のテストを行った結果は１／１０の抑圧を示した（[河野９３]，[渡辺８２]．これは以下のように理解できる．

＊机上チェックも実機テストも設計と同じく人の誤りを避けられない．設計とチェックの両者が誤った時，誤りは残留する．

・抑圧する割合はチェックの誤り率で，

・除去率は（１－誤り率）になる．

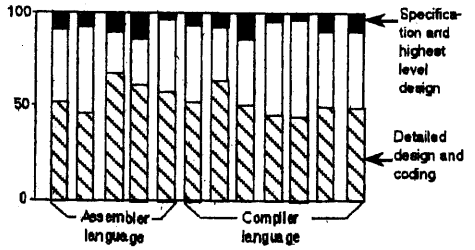


図７　作込み誤りの構成比

図７[河野９３]は作込み誤りの大工程区分毎の構成比率を示す．図のように皆同傾向を示す．過半はコード化や詳細設計で，約１／１０が仕様や上位の基本的な設計の誤りである．実現手段であるプログラム近傍の誤りは１／２程度なので，フローチャートやゲート回路図で誤りを除去していた図４bや図７の時期にはテスト前の除去は１／２程度に留まった．チェックは実現手段に過ぎないプログラムのレベルのみでなく，より上位からデザインレビュー等を行わねば効果が上がらない．

＊作込みと除去に分け管理して計測すると，誤りは安定な様相で把握できる[Koono88]．

誤作り込み率は，初心者が簡単なプログラムを作る時100件／K行程度，熟練者は専門領域内で20~40件／K行程度である．（誤り作込み率や除去率には習熟効果が現れる[Koono88b]．）

定量的な評価を行う．設計およびチェックでランダムに誤り，その確率をそれぞれＥｄとＥｃとする．作業結果の

残留誤り率 ＝（作込み率）×Ｅｃ
＝ Ed × Ec　　(4)

となる．チェックを多数回繰返しても，仕様の正しさや照合等の誤りが影響し，効果は減殺される．抑圧を大きくする効果的な方法は，設計工程を小さな区間に細分して区間毎にチェックする事である．今設計区間を等しい誤り率のM区間に分割し各区間毎に記録を残し，区間毎に設計後にチェックすれば，

全体の残留誤り率 ＝ (Ed／M)・(Ec／M)・M
＝ (Ed・Ec) ／M　　(5)

と１／Mに減少する．実際に品質向上過程を見ると，誤り再発防止策の為に設計文書が次第に追加される傾向が見え，また高品質組織ではMが大きい（多数の図面文書を使う）[倉原９０]．

３．高成熟度組織の設計作業とその知識

高成熟度組織は，長期間にわたり改善を積重ね各種を統一し標準化した．一口で言えば，

全員が均質同一で同目的に向け分担し作業するので，単純な理想形で表わされる．この統一化した体系のモデルを，

・劣等化　・ランダム化　・誤りの増大

させていくと，任意成熟度のモデルを作れる．

３．１高成熟度集団の作業[陳97]

高成熟度組織の開発作業状況を概観する．彼等は新技術の吸収に意欲的であるが，新規技術は評価し作業方法を確立した後に実施する．Systems Engineerはニーズや要求に応え，極力標準機能でシステムを構成する．これらは，念入りな作業計画と共に纏められ，全員これに準拠して作業を遂行する．

製品仕様は標準構造に従い展開され，機能／技術別の階層的組織が分担して計画に従い作業する．作業は階層的な工程に従い，各工程では標準方法で作業が行われ，結果は階層的な文書体系に記録する．これら文書は綿密に厳しくレビューしチェックされる．標準で対処できない事は上位に問題提起され，より高度で統括的能力のある上級者は上位にふさわしい判断知識で処理するか，更に上位へ問題提起する．組織の最終目的やスケジュールは順次展開され，各人は夫々の位置づけを理解し，責任を強く感じて協力する仕組みである．これを高成熟集団のモデルとする．

図９[Koono94]は，特徴的な階層的工程とこれに相補で階層的な設計文書体系を示す．（流用や機械化等は隠蔽した．）設計文書自体は特定の製品を表す知識，階層的な工程は繰返し使われる設計の知識と言える．この階層的な工程が，この組織の設計知識体系のモデルになる．

３．２個別の設計知識とその獲得[陳97]

モデル化した設計知識体系・階層的工程の最下層の作図作業から設計知識を系統的に獲得する方法を述べる．作図工程で例をフローチャート(状態遷移図でも同じ)にとる．設計作業を小さな進行段階毎に区切り，作業の軌跡である図面を残す．相隣りあう図面間で設計情報は階層的に展開する．この展開関係を単位的な設計知識と考え，「設計ル

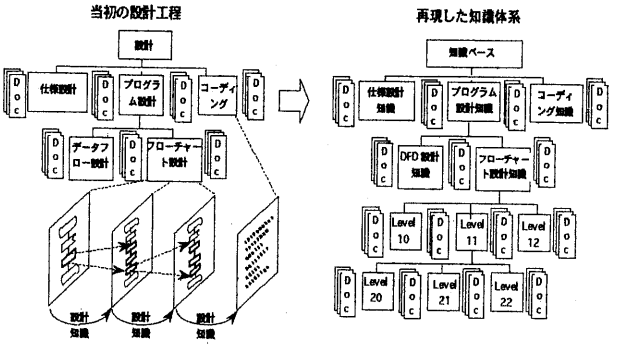


図8　当初の設計工程と再現した知識体系

ール」と名付ける．図のシンボルに併記する自然言語の宣言には，統一し標準化した用語を用いる．相隣る図面間の差からは設計ルールを，

確実に，　高い再現性と高い信頼度で，

系統的に，　誰でも，　容易に，

獲得する事ができる[Koono 94, 陳９７]．これは習熟者が持つ技能化した知識と言える．（初心者は，知識があっても技能化が低く，毎回に基礎知識から展開を試行し最終結果に到着する．）

上位階層には管理や制御の設計知識がある．作業の進行管理の知識は，下位工程を順次進行させる状態遷移で表される．工程の終了判定は，各下位工程の報告から「作業の進捗/終了」や「設計結果の評価」が行われる．次の設計を開始するにあたり，前途を予見し後続設計の方向付けをする．この評価や予測はアナログやパターン認識的な知識が中心になる．これは更に高度化できる．これらの上位の知識の獲得するには，環境と工程を特定して熟練者からその知識を獲得できる．そこで，図の右に示すように，階層的な知識体系が再現でき，これはソフトウエア工学的な各種の研究に使える．

４．新しいソフトウエア工学へ向けて

４．１科学的合理的なソフトウエア工学

本論文では，人の知的処理を基礎と考え，巨視的に階層展開網で近似し，量的な関係を示した．ハードウエア直接作業に比べ，繰返し回数が少なく習熟が低い事をソフトウエア作業の特徴として考慮すれば，ハードウエア作業で実績がある Industrial Engineeringで築かれた諸原理がソフトウエアに応用可能である事を示し，現在企業で経験的に行なう各種の定量的な管理や計画の基礎を与え，また発展させる事ができる．（定量化）

作業モデルとして成熟度が最も高い状態では単純な（理想状態の知識・工程の）モデルに帰する．逆にこれを劣化させ低成熟度集団のモデルも得られる．更に内部の知識構造も判るから，これらを仔細に研究して，

＊ソフトウエア作業は如何にするのか，

＊各種設計方式は何が違い，如何に影響するか，

＊設計文書図面は如何にあるべきか，

等の諸問題の明解な答を得る事が出来る．

工程の概念を用いると，管理面では定量的で計画的な作業の計画，監視と統制等をハードウエア並みに明確にできる．この展開として作業改善の方法も同様に確立できる．

誤りの視点からは，開発は作込みと抑圧の定量的な素性の知れた工程の連鎖になる．大工程区分毎の構成比で示したように工程毎に誤りの種別や影響が違う．そこで，これらを考えた定量的な品質管理，科学的な因果関係からそれぞれの工程毎の品質向上策等も論じる事ができる．ＴＱＣやCreanRoom Method等の品質向上策の評価と改善も行なえる．

理想モデルの標準化した知識体系は１個人の知識とも集団組織の知識の集合とも言える．これはMinskyが指摘した階層的なエージェントシステム[Minsky85]になる．成熟度を低下させ（知識の程度や評価基準等の共通基盤を狭め）任意成熟度が作り出せるから，エージェントの階層／非階層論，分散協調問題等を研究できる．本論文を基礎とし新しいソフトウエア工学を展開できよう．

４．２自動設計への適用

ソフトウエア自動設計はソフトウエア関係者の夢である．この課題は，第１に設計とは何か？であるが前記のように次第に明確化している．第２は階層的に広がる知識の取込み限度と考える．

筆者らは，人に倣った自動設計の研究を進めてきた．これは現在の作業工程を守り，設計の下流階層の下から順次段階的に自動化する．現在では，系統的に設計知識を獲得し，再現するエキスパートも系統的に構成できる事は既に説明した．

プログラム詳細設計の最下層の作図レベルでは，汎用的な知的CASEツール[陳97b,c]を研究している．これは設計知識を自動獲得して再利用により自動設計を行うものである．事前の設計知識の準備が不要でかつ汎用性があるという利点は，人の判断を要する事および使うほど自動化率が向上する人に似た特性等の知識量減少策により得られている．

上記とは別にシーケンス性のある設計の上流段階の研究[化９７]も進めている．人の設計知識「入力から出力への変換」を取出し，エキスパート（自動設計）システム化したものである．これは設計知識の量は減るがエキスパートシステム開発の手間が掛かる．今後は，開発を利用者に簡便に行わせる方策を研究する予定である．

詳細は，関連発表をご参照頂きたい．

５．纏め

この論文では，設計の知的作業に着目して微細構造のモデルや設計組織の階層モデルにより，各種の定量的なデータから，ソフトウエア工学に定量性と科学的な因果関係を捕え易くする手がかりを説明した．これは最も中核的な知の構造を捕えるからソフトウエア工学に新しい基礎になろう．

謝辞

本研究は，産業界の経験を大学で体系化したもので，多くの示唆を下さったかたがた，また研究に貢献した学生院生諸君に感謝します．日立製作所，CSK，電気通信普及財団，NTT，日立中部ソフトウエア，科学研究費基盤研究B１.07558043のご支援を頂いた．厚くお礼申し上げます．

文献

[Chen96] Chen, H., Machida, K., Far, B.H. and Koono, Z.: Software Creation: A systematic construction method of expert systems used for design, The third World Congress on Expert Systems, pp. 577 - 584, 1996.

[陳９７]陳，ファー，河野：ソフトウエ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，人工知能学会誌，１９９７．７（予定）

[陳97b] 陳，ファー，堤，河野：ソフトウエアクリエーション：Intelligent CASEツールの方式について，信学技報KBSE96-26~32, 1997.

[陳97c] 陳，勇，堤，河野：設計知識の自動獲得と自動設計を行う知的CASEツール，情処学会ソフトウエア工学研究会資料，１９９７年５月，１９９７．

[Ford93] Ford, K. M. and Bradshaw, J. M. eds.: Knowledge Acquisition as Modeling, Intl. Jour. of Intelligent Systems, vol. 8, no. 1, pp.1 - 7, Jan. 1993.

[化９７]化，河野：交換接続制御プログラムの自動設計用の専用エキスパートシステム，情処学会ソフトウエア工学研究会資料，１９９７年５月，１９９７，

[Koono88] Koono, Z. Yamato, Y. and Soga, M.: Structural way of thinking as applied to high quality design, IEEE ICC '88, 1988.

[Koono88b] Koono, Z. Yamato, Y. and Soga, M.: Structural way of thinking as applied to improvement process, IEEE GLOBECOM '88, 1988.

[Koono90] Koono, Z. Tsuiji, H. and Soga, M.: Structural way of thinking as applied to productivity, IEEE ICC '90, 1990.

[Koono92a] Z. Koono, T. Baba and T. Yabuuchi. Software Creation: A Trial for Switching software. Proc. 5th JCCNSS '92, pp. 261-265, 1992.

[河野９２]河野，ファー：ソフトウエア開発工程の定量的取り扱い，信学技報ＫＢＳＥ９２－３３，１９９３．Yamasaki, Y., Ohmori, M. and Hatae, K. : Software creation: Towards automatic software design by simulating human designers, The Fifth International Conference on Software Engineering and Knowledge Engineering 1993.

[河野93] 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.

[河野９４a] 河野，ファー，：ソフトウエア開発プロセスの構造，情報処理学会ソフトウエア工学研究会資料９８ー５，１９９４．

[Koono94b] Koono, Z. Far, B. H., Sugimoto, T., Ohmori, M. and Chen, H.: A systemtaic approach for design knowlesdge acquisition from documents, The Third Japaanese Knowledge Acquisitloin for Knowledge-Based Systems Workshop, JKAW 94, 1994.

[河野９５] 河野，ファー：系統的な設計の作業方式とその知識，第１３回設計シンポジウム，pp. 191-198, 1995.

[Koono95b] Koono, Z. and Far, B. H.: Quantitative design of a development process, Software Quality Concern For People: Proc. of Fourth European Conference on Software Quality 1995, pp. 173-181, 1995.

[Koono96] Zenya Koono, Hui Chen and Behrouz H. Far: Experts' knowledge strauctureexpains software engineering, Proc. of JCKBSE '96, pp. 193-197, 1996.

[倉原９０] 倉原他：技術集団のTQC，日科技連，1990.

[Minsky85] Minsky, M.: The society of mind.

[Thayer76] T. A. Thayer et al.: Software Reliability Study, RADC-TR-76-238, Aug. 1976.

[吉田８５] 吉田他：ソフトウエアメトリックスの現状と課題，情報処理，Vol.26, No.1, pp.48-51, 1985

[渡辺８２] 渡辺，緒方：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，１９８２．