解　説

ソフトウェア開発における人間的要素

ソフトウェア開発環境†

小滝房枝†† 河崎善司郎†† 青山義彦††

1. はじめに

ソフトウェア危機がいわれはじめて久しい．この危機を打破するため，優秀なソフトウェア要員の選抜・教育，効率的開発チームや組織づくり，ソフトウェア開発技法・支援ツールの開発と利用促進，人間工学をとり入れた端末の研究，効率良いオフィスレイアウトなど，いわばソフト開発の人間的環境と物的環境の両面からの努力がなされている．

Gunther のいう「プログラミングの時代[1]」(**表-1**)にはプログラミングは特殊な能力を要するものと考えられ個人単位の業務の色彩が強く，技術は職人芸的に個人の中に集積される傾向にあり，ソフトウェア要員は少数のパイオニア的存在であった．「ソフトウェア開発の時代[1]」に入り，ソフト需要の増大と高級言語の普及がみられ，ソフトウェア要員に適する能力を持った人材を選別する道具として，プログラマ適性検査の研究が盛んに行われた．「ソフトウェア工学の時代[1]」に入ると，ソフトウェア要員の絶対数の不足が指摘されるようになり，優秀な人材を多数確保することが困難な状況となったこと，適性研究に一応の結着がついたこと等から，要員の教育，育成に重点がおかれるようになった．更には，ソフトウェアの大規模化，製品としてのソフトウェア開発の必要性等から，ソフト開発を知的分業としてとらえるようになり，業務の標準化，すぐれたチーム・組織づくりへと関心が推移してきている．

表-1　ソフト開発の3時代 (Gunther)[1]

時代名称	作られるもの	開発組織の特徴
プログラミングの時代	計算機プログラム	個人の努力
ソフトウェア開発の時代	ソフトウェア	プロジェクトチーム
ソフトウェア工学の時代	ソフトウェア製品	チーフプログラマチーム

† Software Development Environment by Fusae ODAKI, Zenshiro KAWASAKI and Yoshihiko AOYAMA (Systems Development Laboratory, Hitachi Ltd.).

†† (株)日立製作所システム開発研究所

一方，ソフト開発における物的環境の変化は，開発端末の利用，開発支援ツールの普及と意思疎通の潤滑化志向に端的に表われている．ソフト開発オフィスは従来の事務的オフィスから，ソフト開発の特質を反映させた独自のオフィス形態へと脱皮が図られており，ソフト生産性を鑑みた OA 化が期待される分野である．

本稿ではソフトウェア開発環境を人間的環境と物的環境としてとらえ，

（1） ソフトウェア開発要員の必要資質と選抜方法
（2） ソフトウェア開発における組織編成
（3） ソフトウェア開発における物的環境
（4） ソフトウェア開発技法及び開発支援ツールの人間的側面

に関する研究を紹介する．

なお本研究分野は広範囲にわたるとともに，まだ流動的であり，個別に検証された結果の集積はあっても研究分野として体系化されるには到っていない．そのため本稿では各分野の概要を述べ，続いて注目すべき論文を併記するという体裁をとっている．

2. ソフトウェア開発要員の必要資質と選択法

2.1 知的側面

ソフト開発要員の適性研究の多くは，いわゆる初・中級プログラマに関するものである．これは適性研究の基礎となる職務分析が，より上級の開発要員（上級プログラマ・システムアナリスト等）では困難なことによる．初・中級プログラマの適性として最重視されてきたものは知的能力，特に論理的思考力である．IBM の PAT[2]，日立プログラマ適性検査[3] (PATH) 等はいずれも論理的思考力を測定するテストである．職務分析や職場でのアンケート結果からは独創力，チェック能力[4),5)] 等があげられているが，①訓練効果やジョブパフォーマンスとの関連がつかみにくい，②測

定方法に適切なものがないことから検査化されているものは少ない.

近年妥当性の高いプログラマ選抜テストとして, Wolfe テスト (Wolfe Computer Aptitude Testing, Inc., Montreal, Canada) と Berger テスト (Psyehometrics, Inc., Santa Monica, CA.) が紹介されている[6]. Wolfe テストの一種である AABP (the Aptitude Assessment Battery : Programming) は, 実施に約3時間を要するパワーテストで, ①演繹能力 ②マニュアルを理解する能力 ③仕様書を理解する能力等を測定している.

● De Nelskey 等は AABP の妥当性検証を行い[7], テスト結果が, 訓練パフォーマンスとだけでなく, プログラマとしてのジョブパフォーマンスやシステム分析の潜在能力とも有意な相関を有することを見い出している.

● 適性検査の有用性を疑問視する向きに対し, F. Schmidt 等は, 米国政府のプログラマ採用に PAT を用いることによる生産性の増加をドルに換算して推定している[8].

2.2 性格的側面

「マネージャがプログラマの性格に関心を払うことはソフトウェア生産性の向上に有用である[9]」. と G. M. Weinberg が述べているように, ソフトウェア要員の適性と性格との関連研究も重要である. ここではいわゆる性格検査だけでなく, 入社後のジョブローテーションの資料としても利用が容易な評定尺度[10]等による研究も盛んである. プログラマとシステムアナリストでは性格面での必要資質に差異が指摘されており[11], プログラミングマネージャの必要資質にも指針が与えられる可能性を持った分野である.

● J. Couger 等は, ソフトウェア要員の動機づけパターンや特有な気質を他の職業人と比較して分析し, 積極的な自己成長への要求と社会的接触に対する消極的な姿勢が顕著であることを指摘している. また彼は, ソフトウェア要員が,「仕事の結果に対する監督者による評価情報」のより頻繁なフィードバックを欲していることを報告している[12].

● 筆者等は, 初級プログラマを対象に Y-G 性格結果* を用いてジョブパフォーマンスとの関連を検討し, 衝動性の低い者の上長評価が有意に高いことを見い出した[3].

* 矢田部・ギルフォード性格検査：質問紙による検査で 12 性格特性を 120 項目で測定している.

表-2 システムアナリストの職務行動次元[13]

技術的知識
顧客との関係保持
仕事への献身的努力
計画立案
デバッグ
システムの開発と修正
コミュニケーション保持
ドキュメント作成
後進の訓練
顧客のニーズを評価しそれに対する解決策提案
プレゼンテーションの指揮
監督とリーダシップ

● R. Arvey は生産性を保つ方策として採用後の定期的な評定の必要性を提唱し, ガットマンスケール* を用いた評定尺度を開発している. その中でプログラマとシステムアナリストの職務行動次元を明らかにし, (**表-2**) システムアナリストに要求される資質として「顧客との関係を保持する能力」を検出している[13].

3. ソフトウェア開発における組織編成

グループ編成に関する研究は一般論としては多いがソフトウェア開発を対象に実験的に行った研究は多くはない. まず注目すべき研究は, H. Mills が提案し, F. Baker が実践・研究したチーフプログラマチームに関するものであろう[14]. G. Shave 等はチーフプログラマチームの生産性等を詳細に検討している[15]. チーフプログラマチームに対立するチーム編成法として, G. Weinberg が提案したエゴレスチーム[9] があり, グループ検査技法, ウォークスルの基盤を提供している. 他に, 従来の慣習的なチーム編成法や, チーフプログラマチームの一変形ともとらえられるスペシャリストチーム編成法[16]がある.

● M. Mantei は, チームパフォーマンスの文献調査を行い, それを Baker 型と Weinberg 型のチーム構造の分析に応用した[17]. 彼はチーム構造を以下の3種に分類し, 開発するプログラムの特性により有効と考えられるチーム構造を **表-3** のようにまとめた. これによると Baker 型は期限の短いプロジェクトに効果的であり, Weinberg 型は, より創造的な問題解決が期待できるが, きつい計画のプロジェクトでは, Baker 型ほど効果的でないとしている (**図-1**).

① 民主的非集中型 (Democratic Decentralized): Weinberg 型でリーダはいない. 短期間のコーデ

* Guttman, L. によって考案された態度測定法の一種. 単純に「はい」「いいえ」のいずれかで答えればよい質問からできている. 任意の項目について一方が一方の先行条件になっていれば, それでつくられる尺度は尺度化可能であるという考えからなっている.

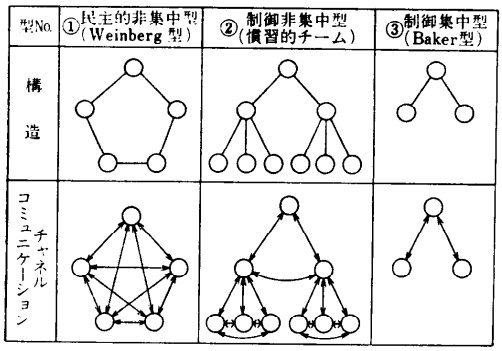

図-1　チーム構造とコミュニケーションチャネル[7]

ィネータを指定する．決定は集団の合意で行い，メンバ間のコミュニケーションは水平型である．

②　制御非集中型 (Controlled Decentralized)：

リーダ及び中間リーダがいて，コミュニケーションは，サブグループ中では非集中型，制御階層中では集中型である．従来の慣習的チーム編成法に近いものである．

③　制御集中型 (Controlled Centralized)：

Baker 型で，決定はチームリーダが行う．コミュニケーションは垂直型である．

彼は更にチーム構造の選択にあたっては，プログラム特性の優先順位決定，ウエイトづけ，特性の組み合わせ等を行う意思決定アルゴリズムを用いることをすすめている．

● R. Scott 等は，プログラミングにおけるチーム生産性の予測を，プログラマの経験や使用できる言語，ドキュメントの質，プロジェクトの経過時間等を入力とする，シミュレーションモデルを作成して行い，以下を指摘している[18]．

①　コミュニケーションストラクチャ中 (図-2) ⓑが最も生産性が低い．（1人への情報の集中が情報の遅滞と非生産的時間の増大をひきおこすため）

②　プロジェクトへの効果的人員投入には上限がある．(ⓑでは投入人員 12 名で生産性は頭うちになった．)

表-3　プログラム特性とチーム構造[17]

プログラム特性 / チーム構造	困難度		サイズ		開発期間		モジュラリティ		信頼性		期限	
	高	低	大	小	短	長	高	低	高	低	近	遠
①民主的非集中型(Weinberg 型)	○			○		○		○	○			○
②制御非集中型(慣習的チーム)		○	○		○		○		○			○
③制御集中型(Baker 型)		○	○		○		○			○	○	

○印が有効性を示す．

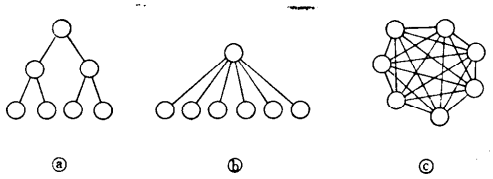

図-2　Scott の実験に用いられたチーム構造[18]

③　プログラマとして有能な人をコミュニケーション要求の多いところに位置づけるとチームの生産性は低減する．

● M. Shaw がまとめた以下のリスト[19]は，ソフトウェア開発の業務再編成及びチーム運営の指針として注目できる．

①　問題解決に対しては，個人よりグループの方が通常良い解決策を提示する．

②　試行錯誤的に進行していく仕事における決定は個人よりグループの方がすぐれている．

③　アイディアの創造中はそのアイディアに対し一切評価はしない方が良い．

④　仕事の進行は個人の方がグループより早い．

⑤　学習は個人よりグループの方が早い．

⑥　座席配置はグループ内の相互作用に影響する．

⑦　組織の発展は集中型ネットワークコミュニケーション（以後 N.C. と略）の方が早い．

⑧　メンバのモラルは非集中型 N.C. の方が高い．

⑨　困難な問題を解くには非集中型 N.C. の方が効果的である．

⑩　集中型 N.C. の方が非集中型 N.C. よりも情報チャネルの飽和により情報網の機能が失われやすい．

⑪　非集中型 N.C. の方が集団としての同質性が高く，凝集性が高い．

⑫　困難な仕事においてはメンバがグループの仕事の進捗状況に対して満足や不満を自由に表現できる方がグループのパフォーマンスは助長される．

⑬　異なる能力を持つ構成者からなるグループの方が同質能力グループよりも効果的に仕事を遂行する．

⑭　少数の部外者がグループ内にいることはメンバの動機づけを増加させる．

⑮　要求されるリーダシップの特質は仕事の性質により異なる．

4. ソフトウェア開発における物的環境

4.1 物的環境

B. Shneiderman はソフトウェア開発の物理的環境に関して検討すべき項目として ①部屋の広さ ②部屋の構造 ③明るさ ④温度・湿度 ⑤机の配列・ワークスペース ⑥端末や計算機設備の利用 ⑦騒音（量と強度）⑧他者からの干渉（電話呼び出しを含む）⑨プライバシ確保の程度をあげている[20]．⑥端末や他の計算機設備の利用には端末の応答性，端末の1プログラマあたりの台数，1プログラマにわりあてられた記憶容量等が含まれると考えられる．更には，どのようなデータや知識がコンピュータ内に蓄積されているかも1つの要因といえるだろう．

以下に(1)ソフトウェア開発オフィス関連 (2)ソフトウェア開発端末関連の研究状況を紹介する．

(1) ソフト開発オフィス

ソフトウェア開発環境の設定には，整合が困難な2つの観点即ち——①個人が集中して思考できる環境であること，②コミュニケーションが正確に円滑に行われる環境であること——のバランスをいかにとるかが重要な課題となっている．

• IBM はプログラミング新技法（階層プログラミング，ウォークスル，トップダウン開発，チーフプログラマチーム等）のソフト生産性への寄与を定量的に把握し[21]，それら新技法の効果を最大限発揮できるような建築構造設計をサンタテレサラボラトリ建設にあたり行っている[22]．建物の各階は十字型で中央に会議室，各翼に 15 のプログラマ個室と共同開発用端末室1室，小会議室1室がある．1プログラマ1個室を原則とし，各室で端末が接続できる．個室は地位に関係なく同一サイズ（3m四方）であり，そのため開発チーム再編成も容易である．この広さはプログラマが個室内を自分の作業しやすいレイアウトに仕上げるのに充分な広さである．同一プロジェクトは同一フロアに配され，プロジェクトごとに通信設備，プリンタを持つ．1端末室には 15 人収容可能である．また，ラボラトリ全体の建築構造を決定するにあたり，リストの流れを分析すると共に「居室から外景が見えること」というプログラマの要求を最大限にかなえる設計を行っているのが興味深い．

• 国内でも最近ソフト開発オフィスのレイアウトに関する検討が散見される．国内における例ではボードによるパーティションにとどめ，準個室的環境としている．会議室は座席から見える中央部に配されている[23),24)]．

表-4 ディスプレイ端末の機能チェックポイント例[26)]

〔ディスプレイ〕
ビットマップ機能，ポジティブなスクリーン上の画像，ネガティブなスクリーン上の画像，蛍光色に関するオプション，スクリーンのサイズ，傾斜角度の調整，回転できる機能，文字記号の明るさと輝度の制御，文字・記号のコントラストとその制御，ぎらつき防止及び画像の歪防止等．
〔キーボード〕
高さ，着脱可能またはCRT部に固定，彫りを施している．ファンクションキー，キートップの大きさ等．

(2) ソフトウェア開発端末

国内では端末に関する規制は特になく，端末の文字の大きさや輝度に関する研究が着手されている[25)]程度である．

• ヨーロッパでは端末に対する規制がソフトウェア要員の健康を守る目的で法律化されている西独のような国もあり，TCA (Trade Cooperative Association) は主なチェックポイントとして 表-4 のようなものをあげている[26)]．ここでは調節可能な機能を重視しているのが目立つ．また，スウェーデンの Occupational Health and Safety Board は，照明，休憩時間，眼鏡の使用に関するガイドラインを設定している[26)]．

• 米国での端末関連研究は，IBM の Human Factors Center 等で行われている[27)]．最近では，ユーザの特性にあわせた端末設計の必要性も T. Carry 等により提唱されている[28)]．

4.2 ソフトウェア開発技法及び開発支援ツールの人間的側面

(1) ソフトウェア開発技法

Dijkstra 以後，構造化プログラミングの有効性を検討する実験が多数行われると共に，構造化を主眼とする技法やトップダウン設計法・ウォークスル等が提案された．一方効率的なソフトウェア開発と保守の容易性に対し，わかりやすい仕様書が重要であるという認識と，従来利用されていたフローチャートの効果に対する議論から，仕様書の書式に関する研究が生まれてきた．更にプログラム構造化の潮流と相俟って構造化プログラミングを支援する図形化仕様書，例えば，HIPO (Hierarchy plus Input-Process-Output)[51)]，ナッシーシュナイダーチャート[52)]，PAD[29)] (Problem Analysis Diagram) 等が提案された．

• B. Shneiderman 等はディテールフローチャートのプログラミングにおける4側面（構成，理解，デバッグ，修正）での有用性の検討を実験的に行い，どの

側面においても，ディテールフローチャートが有益であるという積極的な結果は得られなかったと報告している[30]．また，彼はフローチャートの有用性は，その使用経験に依存するものであることを示唆している．彼はプログラマにとってより役立つよう，データの流れやデータ構造を強調できる仕様書書式の検討を行っている．

●R. Ramsey 等は，PDL (Program Design Language) とフローチャートを実験的に比較し，設計段階では，設計の全体的評価，設計時間等で PDL の方がフローチャートより有用であること，プログラミング段階では，設計書理解テストの成績や所要時間等で PDL とフローチャートの間に有意な差は検出されなかったことを報告している[31]．

●GE では，S. Sheppard 等が仕様書書式の特質を「記号」(自然語，制約言語：PDL，イデオグラム：フローチャート) と「空間配置」(順次型，分岐型，階層型) としてとらえて実験計画法を適用して研究し，以下の結果を得ている[32]．①記号の型は仕様書の理解に影響を与えること．(理解状況を調べる質問に対し，自然語は制約言語，イデオグラムより有意に長い回答時間を要した．) ②空間配置の型は仕様書の理解に影響を与えるが記号の型の効果ほど大きくない．(仕様書の理解に要する時間が分岐型が最も短く順次型が最も長いという傾向がうかがえるが階層型の位置づけは実施したテスト問題によって異なる．)

(2) ソフトウェア開発支援ツール

近年開発が盛んなソフトウェア開発支援ツールは，開発技法とコンピュータを結びつけ，技法利用の促進と効率化を図り，生産性向上を目指すものである．そのため端末や応答時間等ハード的面と技法的面とが混在しており，ツールの評価は，利用者レベルの多様性とも相俟って困難な問題である．定性的にはソフトウェア開発支援ツールの持つべき特性が，A. I. Wasserman[33],[34] 等によって検討されており，①会話性 ②ポータビリティ ③フレンドリネス ④統合性等が掲げられている．会話性は，ハード面とも強い関連を持つ特性で，W. J. Hansen[35]，A. I. Wasserman[36]，R. B. Miller[37] 等によって検討されており，B. Shneiderman が総括を行っている[28]．他の特性についての詳細で体系的な検討は見あたらない．ツールの評価は定性的な段階で定量的評価までは到っていない．

●筆者は実験的評価のみではソフトウェア開発技法やツールの有用性を明確化できないとの認識のもとに，

表-5 「使いやすさ」の各側面における要因とその要素

側面	要因	要素
習得しやすさ	覚えやすさ	覚えるべきものの数が少ない
		覚えるべきものに新奇なものが少ない
	理解しやすさ	内部構造が単純である
		内部構造が明確である
		例外処理が少ない (一貫性)
		論理がおいやすい
	間違えにくさ	混同しやすい事項が少ない
		主観の入る余地がない
読みやすさ (出力のわかりやすさ)	一覧性	文字の大きさ
		利用平面の大きさ
		量の少なさ
		適当な空白がある
		流れがわかる
	理解しやすさ	(習得しやすさ項参照)
	間違えにくさ	(習得しやすさ項参照)
問題解決の手助けになる	考えやすさ	思考の刺激となる
		思考過程にあっている
		思考過程を細分化する助けとなる
		確認しながら思考を進められる
書きやすさ (入力のしやすさ)	手間の少なさ	量としての手間が少ない
		重複が少ない
	入力の自由さ	制約事項の少なさ
	変更しやすさ	修正しやすい
		拡張しやすい
	間違えにくさ	(習得しやすさの項参照)
表現力の高さ		多くのことが実現できる
		機能が多い

ソフトウェア評価尺度[39],[40] や P. Reisner の DB 言語評価[41] 等を参考に，技法・ツールの具備すべき「使いやすさ」の条件の分析を行った (表-5)．「使いやすさ」の各側面は相反する点を多く含んでおり，使いやすいツールとは各側面の使いやすさを生かす機能を組み合わせたものといえる．日立がプログラム自動生成ツールとして開発した PDL/PAD[42],[43] は「PDL が書きやすく，PAD が読みやすい」という仮定のもとに2技法を統合することにより，仕様書とプログラムの乖離絶滅による保守コスト低減を目指したものである．筆者はこの仮定の正しさを実験を通じて検証すると共に構成要素表の妥当性を検討した[44]．

(3) ソフト開発における人間の思考過程モデル

ソフトウェア開発技法やツールの実験的評価研究は T. Moher[45] や B. A. Sheil[46] が批判したごとく，被験者の質，実験プログラムの規模等ネックが大きく結果の一般化に問題があり，有用性が明らかに立証されたといえる技法は多くはない．B. Shneiderman は，有効な技法の提案には人間の思考過程を検討する必要があること，技法の有用性の立証は制御実験を用いて行うべきであることを主張している[20]，ソフトウェア

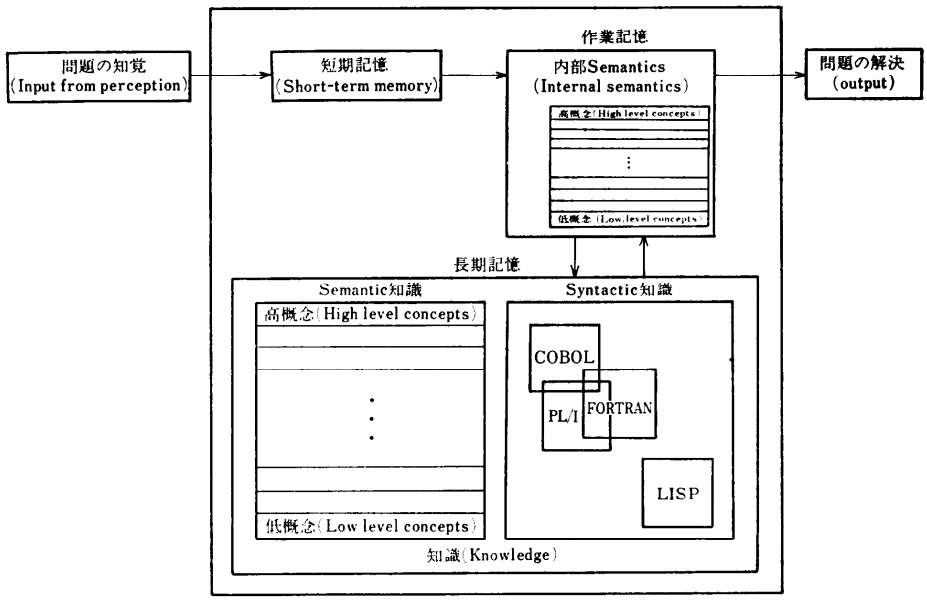

図-3 Shneiderman によるプログラミング行動の Syntactic/Semantic モデル[50]

開発における人間の思考過程モデルの研究は，認知心理学の発展に歩みをあわせるものであるが，従来技法の評価と新技法の開発に有用なものと期待がもてる．注目されるモデルには以下のようなものがある．

- R. Brooks は，Newell と Simon の問題解決モデル[47]をプログラミング業務に応用し，Newell による Production System の検討から知識がどのように長期記憶内に構造化されるかを明らかにした[48]．
- W. Tractz は，コンピュータの機構と対比させた人間の情報処理モデルを提示し[49]，良いプログラムの開発及びそのための技法の選択に際し考慮すべき人間の情報処理過程の特徴や制限事項を指摘している．
- B. Shneiderman 等は，短期記憶と長期記憶，ワーキングメモリからなるモデルを提示し，プログラミング過程の各段階（構成，理解，デバッグ等）の説明を行っている[50]．このモデルの特徴は，長期記憶内に貯えられる知識を Syntactic 知識と Semantic 知識に分類していることである（**図-3**）．

5. おわりに

ソフトウェア開発における人間的要素の占める部分は多いが，この分野においての研究はまだ局所的，部分的段階で体系化できるところまでには到っていないといえる．過去のこれらの研究成果が現実の場で評価され，そこで新たな研究が今後進んでいくことが予想される．例えばソフトウェアの生産性，信頼性，保守性向上のための１つの有効な手段として技法のツール化が進んでいくと考えているが，ツールが実用になるためには人間的側面から配慮 (user friendliness) が大きな要素として指摘されている．

ソフトウェア開発は人間の思考や概念といった抽象的世界を対象としているだけに本質的に人間的側面を無視できない．こういった側面からの研究が遅れていることもソフトウェア生産の問題を大きくしているともいえる．

いずれにせよ，今後ソフトウェア生産の問題を解決するためには，

（1） 生産技法の具体的な場での適用と定着化

（2） ソフトウェアツールの開発とそれを搭載するワークベンチの開発を中心とした生産設備の改善

（3） ソフトウェア生産マネジメント技術の開発と改善

の３つが併行して進むことが必要であると考える．

1980年代は，これらに関する過去の研究成果をふまえ，「人間的要素」の観点からそれぞれの技術が統合化され具体的な効果を上げていく実用化の時代であろう．

参 考 文 献

1) Gunther, R.C.: Management Methodology for Software Product Engineering, John Wiley

and Sons (1978).
2) McNamara, W.J. and Hughes, J.L.: A Review of Research on the Selection of Computer Programmers, Personnel Psychology (1961).
3) 小滝他：プログラマの適性に関する研究，第37回日本心理学会大会予稿集 (1973).
4) 大久保他：電子計算機要員の適性に関する一考察，産業大経営研紀要，Vol. 3 (1965).
5) 西村他：プログラマの適性検査について，鉄道労働科学，No. 17 (1965).
6) Curtis, B. ed.: Human Factors in Software Development, IEEE, Computer Society Press (1981).
7) De Nelskey, G.Y. et al.: Prediction of Computer Programmer Training and Job Performance Using the AABP Test, Personnel Psychology, Vol. 27 (1974).
8) Schmidt, F.L. et al.: Impact of Valid Selection Procedures on Work-Force Productivity, J. of Applied Psychology, Vol. 64, No. 6 (1979).
9) Weinberg, G.M.: The Psychology of Computer Programming, Van Nostrand Reinhold Company (1971).
10) 江村：コンピュータ要員の育成技法(5)「要員育成のシステム化技法」ビジネス・コミュニケーション，Vol. 11, No. 11 (1974).
11) 日本情報処理開発協会：企業内情報処理教育に関する実態調査（昭和51年3月).
12) Couger, J.D. et al.: What Motivates DP professionals?, Datamation (Sep. 1978).
13) Arvey, R.D. et al.: Rating Scales for Systems Analysts and Programmer/Analysts J. Applied Psychology, Vol. 59, No. 1 (1974).
14) Baker, F.T.: Chief Programmer Team Management of Production Programming, IBM Syst. J. 1 (1972).
15) Shave, G. et al.: Team Dynamics in Systems Development and Management, Proc. SIGCPR (1977).
16) Brooks, F.P.: The Mythical Man-Mouth; Essays on Software Engineering, Reading, Mass, Addison-Wesley (1975).
17) Mantei, M.: The Effect of Programming Team Structures on Programming Tasks, Comm. ACM, Vol. 24, No. 3 (Mar. 1981).
18) Scott, R.F.: Predicting Programming Group Productivity-A Communications Model, IEEE Trans. Softw. Eng., Vol. 1, No. 4 (Dec. 1975).
19) Shaw, M.: Group Dynamics; The Psychology of Small Group Behavior. McGraw-Hill, N.Y. (1971).
20) Shneiderman, B.: Software Psychology, Winthrop Publishers (1980).
21) Walston et al.: A Method of Programming Measurement and Estimation, IBM Syst. J. No. 1 (1977).
22) McCue, G.M.: IBM's Santa Teresa Laboratory Architectual Design for Program Development, IBM Syst. J., Vol. 17, No. 1 (1978).
23) 後藤：頭脳労働の環境づくりに工夫をこらしたソフトウェア工場，JMA ジャーナル (Nov. 1982).
24) 登家：ACOS-4 オペレーティングシステム開発のプロジェクト管理，電気学会会誌，Vol. 98, No. 1 (1978).
25) 日本経済新聞：コンピュータの文字，目への影響初の本格調査 (57.9.25)
26) T.マニュアル：欧米で広まる人間工学をとり入れた CRT 端末，日経エレクトロニクス，12-6 (1982).
27) Hirsch, R.S.: Procedures of the Human Factors Center at San Jose, IBM Syst. J. Vol. 20, No. 2 (1981).
28) Carry, T.: User Differences in Interface Design, Computer (Nov. 1982).
29) 二村他：PAD (Problem Analysis Diagram) によるプログラムの設計及び作成，情報処理学会論文誌，Vol. 21, No. 4 (1980).
30) Shneiderman, B. et al.: Experimental Investigations of the Utility of Detailed Flowcharts in Programming, Comm. ACM, Vol. 20, No. 6 (June 1977).
31) Ramsey, H.R. et al.: Flowcharts vs. Program Design Languages; An Experimental Comparison Proc. Human Factors Society (1978).
32) Sheppard, S.B. et al.: The Effects of the Symbology and Spatial Arrangement of Software Specifications, Proc. Fifth International Conference on Software Engineering (1981).
33) Shneider and Wasserman eds.: Automated Tools for Information Systems Design and Development, North Holland (1982).
34) Wasserman, A.I.: Automated Development Environment, Computer (Apr. 1981).
35) Hansen, W.J.: User Engineering Principals for Interactive Systems, AFIPS Conf. Proc. FJCC, 39 (1971).
36) Wasserman, A.I.: The Design of Idiotproof Interactive Systems, AFIPS Conf. Proc. NCC, 41 (1973).
37) Miller, R.B.: Response Time in Man-Computer Conversational Transactions, Proc, SJCC, 33 (1968).
38) Shneiderman, B.: Human Factors Experiments in Designing Interactive Systems, Computer (Dec. 1979).
39) Boehm, B.W. et al.: Quantitative Evaluation of Software Quality, Software Phenomenology Working Papers of the Software Management

Workshop (Aug. 1977).
40) Gilb, T.: Software Metrics, Winthrop Publishers (1977).
41) Reisner, P.: Human Factors Study of Database Quality Languages; A Study and Assessment, Comput. Surv. Vol. 13, No. 1 (1981).
42) 前沢他：対話型の構造化プログラム設計・製造支援システム (PDL/PAD) の開発，情報処理学会第24回全国大会論文集 (昭和57年前期)，pp. 299-300.
43) 斉藤他：図形化論理仕様出力システムの開発，情報処理学会第22回全国大会論文集 (昭和56年前期)，pp. 311-312.
44) 小滝他：プログラム論理仕様記述法 PDL と PAD の「読みやすさ」における比較評価，情報処理学会，ソフトウェア工学研究会資料 28 (1983).
45) Moher, T. et al.: Methods for Improving Controlled Experimentation in Software Engineering, Proc. fifth International Conference on Software Engineering (1981).
46) Sheil, B. A.: The Psychological Study of Programming, Computing Surveys, Vol. 13 (Mar. 1981).
47) Newell, A. and Simon, H. A.: Human Problem Solving, New York Prentice-Hall (1972).
48) Brooks, R.: Towards a Theory of the Cognitive Process in Computer Programming, International Journal of Man-Machine Studies, Vol. 9 (1977).
49) Tracz, W. J.: Computer Pragramming and the Human Thought Process, Software-Practice and Experience, Vol. 9 (1979).
50) Shneiderman, B. et al.: Syntactic/Semantic Interactions in Programmer Behavior: A Model and Experimental Results, International J. Comput. Syst. Sci. Vol. 8, No. 3 (1979).
51) Myers, G.: ソフトウェアの複合/構造化設計，近代化科学社 (1979).
52) Nassi, I. and Shneiderman, B.: Flowcharting Techniques for Structured Programming SIGPLAN Notices (ACM), Vol. 8, No. 8 (1973).

(昭和58年2月22日受付)