コンピュータと教育 38－12
数理モデル化と問題解決 4－12
（1995．11．17）

新入社員に対するレビュを中心とした プログラミングの教育方法

駒谷 昇一
ＮＴＴソフトウェア株式会社
komaya@yh.ntts.co.jp

平成７年度の新入社員研修で実施したプログラミング研修において，定量的な評価を行うために研修の前後で試験を実施した．この試験の結果から，レビュを中心としたプログラミング研修が，システム開発教育の基礎のスキルアップにおいて，有効であることが分かった．システム開発を行う人材を育成するために，当面は言語の文法を教えプログラマとして，そしてその後システム設計のできる人材へと育成する場合がある．しかし，初期の段階からシステム開発を意識した教育をプログラミング教育においても実施することができ，このシステム開発を意識したプログラミング教育のためには，レビュを中心としたプログラミング教育が有効である．

The Teaching of Programming by Review for the New Staff

Shoichi Komaya
Personnel Department, NTT Software Corporation

We performed Programming Skill Tests at, before and after the teaching of Programming by Review for The New Staff. This Test is effective against increasing Skill for the Teaching of System Development. There is a teaching style that is Teaching the System Design after Teaching the Programming Language for the present. But we can perform the Teaching of Basic System Engineering at an early stage during the teaching of Programming by Review.

１．はじめに

１．１ 新入社員研修の実施方法

新入社員研修の目的は，研修終了後の業務で必要な知識や能力を効率的に身に付けさせることである．新人研修は，修得する内容に応じて，大きく３つのプログラムで構成されている．平成７年度の新入社員研修では，以下のような内容と期間で実施した．

(1)共通研修

社会人意識，ビジネスマナー，ホウレンソウ，同期生同士の交流，同和問題などの全新入社員が共通に最低限身に付けておく必要のあることを学ぶ．入社直後から１週間の合宿研修を含め２週間実施した．

(2)専門研修

ソフトウェア開発職の新入社員を対象に，情報処理に関する基礎知識と基礎能力(タッチタイピング等)を修得する．共通研修終了後から６月上旬まで，約６週間実施した．

(3)事業部別システム設計研修

各新入社員の配属先において，システム開発の１ラウンド(上流工程から納入検査まで)を体験し，システム開発の手順やその生産物の作成

方法，事業部固有の技術について修得する．専門研修終了後から２～５カ月の間，実施した．

1．2 専門研修の目的と実施体制

専門研修の目的は，

(1)ソフトウェア開発に必要な基礎知識や基礎用語を理解する(講義中心)．
［ソフトウェア開発基礎研修］

(2)システム設計の基礎(問題把握力，レビュ力等)をプログラミング教育を通して修得する．
［プログラミング入門研修］

(3)作業標準に従いシステム開発の上流工程から下流工程を一通り体験して，システム開発の手順や生産物を学ぶ．
（工程概説演習研修）

である．

専門研修はＣ言語のプログラミング実習を中心に行うため，入社直後にＣ言語の試験を実施し，その試験結果のレベルに応じて４班(Ａ班(上級)～Ｄ班(初級))に班分けして実施している．

各班には(2)(3)を指導するため，入社３年目以上の先輩の講師を数名づつ配置し，初級者班の講師密度（講師数／生徒数）が最も高くなるように配慮している（Ａ班5/25，Ｄ班4/9）．講師は，研修の間業務を離れ講師を行うが，それ以外に新人研修開始前に約１８日間の訓練および講義準備を行う．

プログラミング入門研修では，竹田1)のカリキュラムに準拠し，社内標準に関する改良を加えた教材を用いている．

専門研修については，これまでの研修を実施して得たノウハウを独自に，研修実施計画書，各班毎の研修の到達目標，各科目の目的や実施方法が記述された教育カリキュラム，に整理しそれらに基づいて実施している．さらにその資料を，事業部別システム設計研修の参考にするため各新入社員の配属先に事前配布し，教授内容の重複を出来るだけ避けられるように配慮している．

1．3 専門研修の内容

専門研修は，図１．１に示すような科目で構成されている．

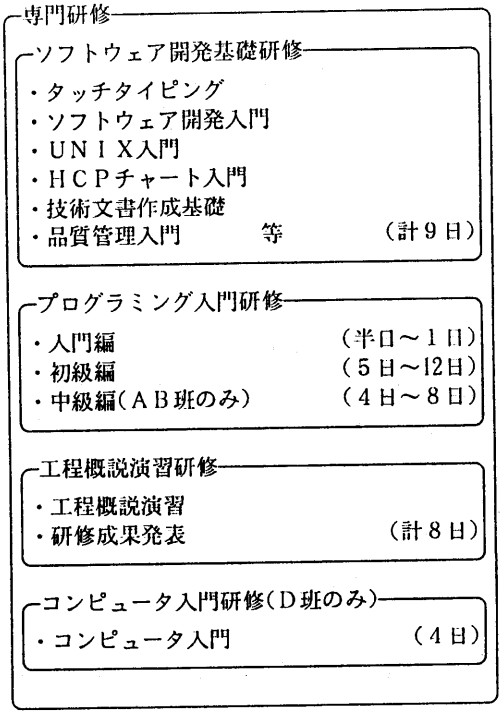

図１．１　専門研修の科目一覧

コンピュータ未経験者であるＤ班については，コンピュータ入門研修を実施する関係上，プログラミング入門研修が４日間短い．

工程概説演習は，複数メンバで開発を行うときの難しさを体験し，各工程の生産物の作成方法と品質の作り込み方法を実習で学ぶための研修である．４～５人のミニプロジェクトを作り，約１Ｋ行のミニシステムを作成し，研修最終日に成果発表とデモストレーションを行う．

プログラミング入門研修は専門研修３０日間のうち１３日間であり４０％を占めている．

2．プログラミング入門研修

2．1 研修の目的

一般的に，プログラミング教育で強調されることは，言語教育としての文法の理解であるが，Ｓ

Eには，問題発見能力やシステム構築能力やコミュニケーション能力等が不可欠である．そして，その能力を新人に対してのプログラミング教育でも高めることが可能であり，ＳＥ育成に有効である（橋本2)）．

プログラミング入門研修では，以下のことを目的として研修を実施している．

(1)問題把握力の向上

要求内容（演習課題）に対して，入力として何が必要で，何を出力するのか，その出力を得るために何をしたらよいのかを表現できる．

(2)問題解決手順の明確化

要求内容をトップダウンで詳細化および構造化することができる．また，詳細レベルの処理の流れとデータ構造を，ＨＣＰチャートを用いて記述できる．

(3)レビュ力

設計時の誤りを指摘したり，読解性向上や再利用し易さなどの点での指摘をすることができる．

(4)プログラミング力

テキストを見ながらコーディングし文法誤りの除去ができる．また，アドレス，構造体，などの概念を理解する．コーディング標準に準拠して，プログラムを作成することの目的とその重要性を理解する．プログラミングの再利用や保守を考慮して，コメントを入れることの重要性やそのコメントの入れ方，またインデント等のマナーを身に付ける．

(5)開発工程

プログラム開発手順（下流工程である詳細設計から結合試験まで）を体験することにより，各工程での生産物が何かとその作成方法を理解する．特にプログラミング工程（M工程）は，ソースプログラムだけを作るのではなく，単体試験用ＰＣＬなどを作成しなければならないことなど，何をどのように作成するのかを知る．また，品質の作り込み方法を知る．共同でプログラミングを行う場合の注意事項やその場合の手順（最初に共通モジュールの洗い出しを行うなど）を理解する．

(6)報告書作成

毎日リーダを対象に進捗会議が行われ，また全員が日報を作成する．その日報に対し講師は毎日コメントをする．このなかで，報告の仕方，問題の捉え方，その解決の仕方などを学ぶ．

(7)チームワーク

学生時代に個人でプログラムを作成した経験があっても，実務のように複数メンバで協力してプログラム作成した経験のある新人社員は殆どいない．

チームでプログラムを作成する場合，その分担をどのようにしたらよいのか，リーダは何をするのか，レビュの仕方，を学ぶ．

各項目に対して具体的な到達目標を各班毎に設定し，各個人毎にその到達目標が達成されたかどうかを評価し，配属先に対し報告している．この情報提供の目的は配属先で行う事業部別システム設計研修をスムーズに行うためである．

2．2 研修の実施方法

プログラミングの実習を個人毎に行うと正常動作させることに集中するため，言語の文法の誤りを指摘する力を向上させることはできるが，問題把握力の向上等が図れない．

これに対してレビュは問題に対しての認識を深くし，チームでの議論を通して，バグの指摘だけではなく，他人に分かり易く保守の容易な良いプログラムとはどのようなものかを効果的に学ぶことができる．またレビュでは色々な考え方を学ぶことができ，問題意識を高められる．

新入社員は自主的にレビュを行うが，必ず講師が入り，指摘漏れがないように配慮している．また講師はレビュの実施方法として，講師による知識提示型で教えるのではなく，積極的に新入社員同士が発見的に学ぶように配慮している．

レビュを行う上で，講師により指摘する内容が異なることを避けなければならない．このため，

講師については事前に研修の主旨や指導方法などの研修を実施している．さらに，研修期間中も講師間の調整を適宜班毎と，講師全員で毎日研修前と研修後に打ち合わせを行い，班を越えた情報交換や意識合わせの場を設定している．

講義については各班毎に行うが，実習については４～５名のチーム単位で行う．講師は各チームの構成を新入社員のリーダシップを参考に作成する．

テキストは，入門，初級，中級に分かれ(竹田1))ているが，各班毎に進捗が異なり，Ａ班は中級まで，Ｂ班は中級の途中まで，Ｃ班とＤ班は初級までを行う．

実習は１人１台の端末(ＷＳまたはＰＣ)を用いて，ＵＮＩＸ上でＣ言語のプログラミングを行い，自主的に学びたい者のために定時退社後も開放している．

２．３ 研修の評価方法

研修の評価は，各新入社員が，研修の目的に合わせて作成された具体的な到達目標（プログラミング力など一部班毎に異なるものもある）のどれをどれだけ到達したかで評価している．

評価の方法は，試験を行い定量的に評価するものと，各班の講師が協議して評価するものとがある．

プログラミング入門研修のプログラミング力とレビュ力の向上度合いについては試験で定量的に評価した．一通りＣ言語の文法とコーディング標準について教えた直後と専門研修を終了するときの２回，同一問題で試験を行った．なお新入社員には同一問題の試験を実施することを知らせないで行った．

その他の問題把握力の向上，問題解決手順の明確化等については直接試験で測ることができない．このため，各生徒毎に研修終了時のレベルを各班毎の講師が協議により評価した．

２．４ 効果測定試験内容

プログラミング力とレビュ力を測るための試験(以下試験と略す)は，誤りを含んだ約１３０行(コメント含む)のソースプログラムに対する指摘の個数により評価した．

問題は，ＨＣＰチャートと，そのＨＣＰチャートをもとにして作成されたとするＣ言語のソースプログラムを提示し，予めソースプログラムに混入してある文法誤り(１３個)とコーディング標準に準拠していない誤り(１２個)を指摘させるものである．

この試験を行うことにより，ポインタや配列等に関するＣ言語文法の理解度であるプログラミング力と，文法誤りではないが見やすさの点で不適切な表現である空白やインデントの入れ方やＨＣＰチャートの文章とソースプログラム上のコメントとの不一致等のコーディング標準に準拠していないものに対するレビュ力を定量的に測ることができる．

２．５ 効果測定試験の実施結果

試験は，Ａ～Ｄ班に対して実施したが，Ｄ班については研修期間が短かったため，他班との比較ができない．このため，Ａ～Ｃ班についてのみ試験結果を表２．１および図２．１に示す(数値は%)．なお１回目に指摘出来なかったが，２回目試験の研修終了時に指摘できた場合を１×２○と表す．

表２．１　各班毎の回答傾向

回答分類	１○２○			１×２○			１○２×			１×２×		
班	A	B	C	A	B	C	A	B	C	A	B	C
全　体	43	33	28	17	17	25	7	6	5	33	44	42
文法誤りの指摘	48	41	32	19	19	27	7	7	5	26	33	36
標準未準拠の指摘	37	25	24	15	14	23	7	5	4	41	56	49

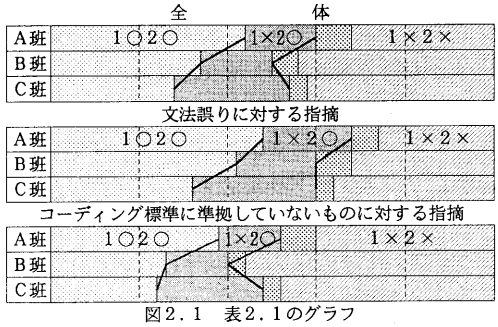

図２．１　表２．１のグラフ

１回目の試験は文法もコーディング標準も一通り講義で教えているにも係わらず２回目の試験で

は指摘率が約３５％状態から約２０％の向上があった．これは，情報提示型の講義では不十分であり，演習やレビュによる教授法が，講義を補完したことを示している．

特にコーディング標準に対する指摘が，文法誤りに対する指摘とほぼ同等の向上が図れた．これはレビュによる教授法が，文法誤りだけを学ぶ演習形態よりもバランスのとれた育成ができることを示している．

１×２○の網掛け部分は研修によりスキル向上が図れた部分である．この部分はＣ班が最も高く，ＡＢ班がほぼ同率である．このようにＣ班のレベル向上が最も大きかった理由は，

・講師の密度がＡＢ班と比較し高く，きめ細かな指導が行き届いたこと
（A班 5/25　B班 4/21　C班 3/11）
・Ｃ班は初級までを終了し，中級に進まずＡＢ班と比較して余裕のある研修ができたこと

の２点であると思われる．

誤りの傾向により，表２．２のように，２回とも指摘があった高指摘率のもの，２回目でも指摘されなかった低指摘率のもの，研修後に多く指摘された著しく向上したもの，研修では十分に高められなかった低向上のものに明らかに分類することができる．

著しく向上したものを見ても，文法誤りと標準化に反する指摘とのバランスが保たれている．

また指摘の傾向から，ソースプログラムの前後行を見なくても誤りだと分かるものやポインタに関する指摘については著しく向上した．しかし細かいスペル誤りや必要な処理が行ごと不足しているものや全体の解読ができなければ誤りを指摘できないものに対しての指摘は難しいため指摘率が伸びなかった．

ポインタに対しての指摘が多かったのは，レビュでも念入りにチェックが行われたためであると思われる．

ソースプログラムはＨＣＰを元に作成するためＨＣＰとソースプログラムとの文が一致しているという思い込みがある．このためソースプログラムとＨＣＰチャートとの文章の不一致の指摘については，レビュで軽視されたため，他と比較して指摘が少なくなったと思われる．

表２．２　指摘された誤りの傾向

区分	項番	誤り種類	○	×	×	○	１回目	
			○	×	○	×	２回目	誤り指摘の内容
高指摘率	1	文法	80	7	10	3		forの第2第3引数の順番が逆になっている
	2	文法	76	2	15	7		gets()の終わりに;がない
	3	文法	75	4	13	8		printf("～¥n);の"の終わりがない
	4	標準	83	0	10	7		インデントが１段浅い
	5	標準	82	4	10	4		１行に複数の変数定義を行っている
低指摘率	6	文法	2	91	4	3		forの前にinstr--が必要
	7	文法	3	81	10	6		defineの数値が１少ない(文字列'1'のため)
	8	標準	0	98	2	0		HCPの処理内容がソースのコメントにない
	9	標準	0	98	2	0		HCPとソースのコメントの文章が不一致
	10	標準	0	97	3	0		関数の内容のコメントが不足
	11	標準	2	89	7	2		著作権コメントのスペルミス UNPUBLISHD
著しく向上	12	文法	52	10	35	3		printf("%c～のclasでなければならない
	13	文法	26	32	32	10		outstrは*outstrでなければならない
	14	文法	19	48	31	2		"¥0"は'¥0'でなければならない
	15	文法	49	13	28	10		コメントの終わりの*/が不足
	16	文法	12	54	27	7		文字列の最後にヌルを入れる処理が不足
	17	標準	28	25	34	13		#defineの定義が小文字になっている
	18	標準	31	25	34	10		voidと関数名が別の行になっていない
	19	標準	33	31	29	7		インデントが１段浅い
	20	標準	15	48	28	9		関数名と(との間に空白がない
	21	標準	54	12	25	9		forの;の後ろに空白がない
低向上	22	文法	54	13	18	15		printfのfが欠けている
	23	文法	53	22	13	12		voidがviodになっている
	24	文法	28	45	18	9		whileの大小比較の==が=になっている
	25	標準	31	41	10	18		HCPとソースのコメントの文章が不一致

注 文法 文法誤りに対する指摘
　標準 コーディング標準に準拠していないものに対する指摘

３．まとめ

講義で言語の文法を中心に教え，個人毎にプログラムを作成し動作するためのデバグを行う演習形態では，生徒は動くプログラムを作ることに集中する．このような教育をすると，Ｃ言語のプログラムは作成できても，他人に分かり易い長い変数名を付けず，コメントを入れず，利用者の立場に立った配慮ができないプログラマが育つ．

またプログラミング演習で他人の作ったプログラムを解読し改造するという機会がないと，機能拡張のための改造や再利用を意識したプログラムを作成することの重要性が認識されない．

実務に於いては複数メンバで共同でシステムを開発する．当然ソースコードレビュも複数メンバで行う．また，改造では必ずしも自分の作ったプログラムを改造するとは限らない．このため，実務で役立つプログラマとは，他人に分かり易く作

業標準に沿ったソースプログラムが作成でき，他人の作ったプログラムを理解できレビュで誤りを指摘できる技術者である．このような技術者を育てるには，レビュを中心とした形式の教授方法が効果的である．

指摘内容の４０％以上がコーディング標準に対する指摘であった．もしレビュ中心の教授方法でなかったら，コーディング標準に対する指摘はこれほど多くはなかったと思われる．

ところで１回目の試験はコーディング標準に対して一通り講義した後に実施している．したがってレビュによる研修により，コーディング標準についての知識のレベルからコーディング標準に準拠していないことを指摘できる能力にスキルアップが図れたことを表している．

たとえプログラミング演習を行っても，個々の演習であったり，生徒が正常動作することに集中していたら標準に反する指摘能力を高めることはできないであろう．レビュを中心とした研修では，レビュの中で講師は様々な指摘を行う．当然コーディング標準についても指摘を行う訳で，標準に準拠することの重要性を認識させることができる．

また，レビュで講師は，入出力が何であるのか，ＨＣＰチャートの記述方法は適切か，再利用などを考慮したモジュール分割の仕方は適切か，などの指摘も行っており，定量的な評価を行っている訳ではないが，レビュにより問題把握力の向上や問題解決手順の明確化のスキルアップも図れていると考えられる．

４.今後の課題

レビュを中心としたプログラミング教育を効果的に行うには，事前に十分訓練された講師を大量に必要とする．このため学校教育に適用するのが難しいという問題がある．

しかし企業内教育も学校教育も，良いプログラムを作成できる技術者を育てる努力をさらに行う必要がある．ただし良いプログラムの定義は技術の進歩とともに変化する．

ハードウェアの性能や機能が指数的に向上するなか，ソフトウェア技術の進歩も非常に早く，開発言語や開発環境も非常に早いペースで変化している．ＧＵＩの向上でオブジェクト指向のシステム開発やＣ＋＋が主流になり，さらにはアイコン等の組み合わせにより所定の機能を実現するビジュアルプログラミングが今後主流になるだろう．そうなると言語に対する文法の知識が不要になることもあり得る．

今後，『特定の言語を用いて，詳細設計書のとおりにコーディングができる能力』よりも，『適材適所で色々な言語を使い分け，場合によってはビジュアルプログラミングや市販のツールの組み合わせなどで，目的とする機能を実現できる能力』が求められると思われる．

このため新入社員に対するプログラミング教育でも，言語の文法を中心に教える教育ではなく，システム設計やプロジェクト管理や品質管理に必要となる基礎的な知識や能力を高めることに主眼を置いた研修をさらに実施する必要がある．

教育を企画する者に対しては，従来の教え方に固執しないこと，何を教えるのかの目的意識を失わないこと，そして新しいコンセプトを考え，それを研修企画に反映できることがさらに求められると思う．

参考文献

1) 竹田尚彦，大岩元：プログラム開発体験に基づくソフトウェア技術者育成カリキュラム，情報処理学会論文誌，第３３巻，第７号，PP.944-954，1991

2) 橋本千恵子：プログラム開発体験に基づく上流工程ＳＥ育成カリキュラムの開発と実施，情報処理学会コンピュータと教育研究会報告，No.３５，PP.1-12，1995

3) 新しいソフトウェア基礎教育の提案，日本科学技術連盟第14回ソフトウェア生産における品質管理シンポジウム発表報文集，PP.49-56，1994