ソフトウェア工学 61－4
（１９８８．７．２７）

開発支援環境を新人へ教育するための考察

プログラミングの評価

石川　佳子　　太田　雅子

三菱電機東部コンピュータシステム（株）

同一のソフトウェア生産環境にて，各班（１班３～４人で構成）が同一の実習テーマで作成したソースプログラムを，静的解析ツール，カバレッジ測定ツール等で評価する教育方法を行っている。これらのツールの教育への効果的使い方について述べる。

さらに，新人研修の主眼である

（１）新人がスムーズにシステム生産ライフサイクルの流れにのれる為にドキュメントやプログラムを作成する。

（２）プロジェクト遂行上重要な種々の管理（原価・工程・品質）を実際のプロジェクトに近い形で体験する。

についても述べる。

A Consideration Of Training The Development Support Environment In The Raw Recruit Training

A technical evaluation of programming

Yoshiko ISHIKAWA　　Masako OHTA

MITSUBISHI ELECTRIC COMPUTER SYSTEMS(TOKYO) CORPORATION

973-2 Yamazaki, Kamakura city, Kanagawa, 247 Japan

Under the same S/W production environment, we are using the training method of each group (a group consists 3～ 4 persons) producing the source programs with the same theme which evaluated with Static Analysis Tools, Coverage Measurement Tool, etc. We state the effective usage of these tools for training.

Furthermore, we state the prime objectives of the raw recruits as follows:

(1) To help recruits to make a smooth transition to the system production life cycle by producing the programs, documents, etc.

(2) To experience the various management system (the cost, progress schedule, quality) for performing a project which is close to the real project form.

1．はじめに

当社の新人研修は，毎年４月から７月の約４ケ月かけて，ソフトウェア生産技術を中心に教育している。この研修の狙いは大要，次のとおりである。

（1） 技術者としての戦力化を図る

（2） 社会人としてのマナーを身に付ける

数年前までの研修の達成目標はプログラミング能力の習得が主であり，それ以外はＯＪＴ（職場内訓練）で習得していくという形であった。

近年，ソフトウェアの生産量も飛躍的に多くなり，ソフトウェアの生産性の向上が要求されるに従い，新人研修についても，トータルな仕事の流れ（プログラム設計，製作，試験）を理解し，実践もできるような教育内容が現場を中心に強く求められている。また，最近ソフトウェア技術者の教育の重要性が指摘されてきている。

そのために，研修の達成目標も以下のようにステップアップしてきた。

（1） トータルな仕事の流れをとらえ，研修終了後システム生産ライフサイクルの流れにのってスムーズに作業ができる業務遂行方法や技術，知識の習得

（2） 今後の上級教育を受ける準備

（3） 情報処理技術者試験第２種合格

これらの技術や知識の多くは，経験により培われるものであるので，講義だけで身に付けさせることは難かしく，実践を伴うことが必要となる。

そこで，目標を達成する為に研修の最後約１ケ月かけてプロジェクト体験実習を実施している。

現在の新人研修カリキュラムを整理したものを図１に示す。

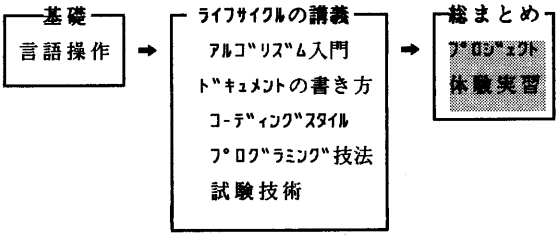

図１　新人研修カリキュラム

各カリキュラムの期間比率を表わしたのが，図２である。トータルな仕事の流れが身に付けられるような内容とするため毎年スケジュール，カリキュラムの見直しを行っている。その結果，システム生産ライフサイクルの講義（専門的な技術習得）＋プロジェクト体験実習が増加している。

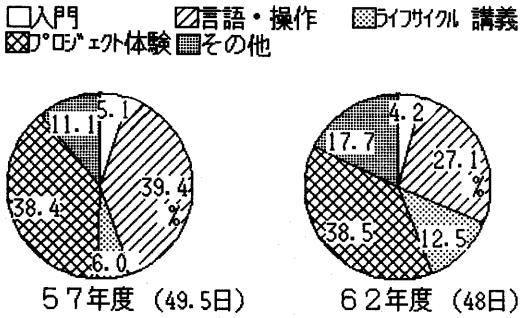

図２　新人研修研修内容の移り変わり

本稿では，プロジェクト体験実習の内容，そこで作成したソースプログラムについてツールを使用して評価した事例，そしてプロジェクト体験実習（新人研修）での効果的なツールの使用法について述べる。[9)]

2．プロジェクト体験実習とは

作業基準については，ライフサイクルの講義中の「ドキュメントの書き方」及び「試験技術」で知識の付与を行っている。しかし，講義だけでは充分には理解できないので，それを補うためにプロジェクト体験実習を行っている。

プロジェクト体験実習は，３～４人を１班としソフトウェア生産環境，課題はすべて同じで，実際のプロジェクトに近い形で進める。実際のプロジェクトとの相違は，プロジェクトリーダを決めず，工程会議（2.3 参照）出席は輪番制とし，全員が工程会議を体験することである。

目標としては，以下のことを設定している。

（1） 一つの課題を共同で作業し，課題達成の重要性を理解する。

（2） プログラム設計，製作，試験という一連のソフトウェア生産作業を理解し，実践する。

（3） プロジェクト遂行上重要な種々の管理（品質・工程・原価）について理解し，管理の基本を実践する。

２．１　共同作業の体験

一つの課題を共同で作業することで，コミュニケーションの必要性，信頼感・チームワークの重要性を身に付ける。そして，決められた期間内で課題を完了するという達成感を味わう。

２．２　作業基準の指導

プロジェクト体験実習では，ソフトウェア生産ライフサイクルの各フェーズの作業をそれぞれの作業基準に従って進めるよう指導している。特に以下のことを重要ポイントとしている。

設計	ｲﾝﾀﾌｪｰｽ	■ﾓｼﾞｭｰﾙ間でやりとりするﾃﾞｰﾀの論理的かたまりは少なくする
	構　造	■機能的な意味がなくなったり，制御が複雑になりすぎないよう考慮し，なるべく小さくする ■ﾓｼﾞｭｰﾙ間において，必要最小限の情報のやりとりしかないような接続関係とする
	機　能	■関連性の少ない処理を一つのﾓｼﾞｭｰﾙ内に同居させない
製作	形　式	■自己流にならないよう設定されているｺｰﾃﾞｨﾝｸﾞ基準に則する
試験	試験方針	■ﾌﾟﾛｸﾞﾗﾑ 単位の機能確認試験を行い，ﾄﾞｷｭﾒﾝﾄ に記述されている機能仕様どおりにﾌﾟﾛｸﾞﾗﾑ が作成されていることを確認する

２．３　各管理方法の指導

管理については，品質管理・工程管理・原価管理を実施している。[4][5]それは実際のプロジェクトと同様の形で進めることにより，チーム員全員が自然に体得できるように配慮してある。

(1) 品質管理――各種デザインレビューと障害管理の実施

(2) 工程管理――バーチャートによる１回／週の進捗チェックミーティング実施

(3) 原価管理――３回／月の作業区分（作業内容毎に分類したもの。例えば，単体試験要領書作成，デザインレビュー　単体試験　等）毎の時間計上による費用算出実施

３．開発支援環境

２で述べたプロジェクト体験実習の目標が達成できたかどうかを判断する基準は，

(1) 期間中に完成した／しない

(2) 人手で調査したデータ（プログラムサイズ，ステップ数，分岐数　等）で判断

(3) 教育担当者の経験による知識で判断

としていた。しかしながら，

(1) 完成度が完全には把握できない

(2) 研修効果の年度比較をするためのデータが，主観や経験による知識から出てくるため，客観性がない

(3) 理解度を把握することが難しいため，各講師は講義内容へフィードバックをかけにくい

という問題があり，「教育効果を定量的に把握したい」「経験の多少に関係なく客観的な判断や評価をしたい」という意見が出された。

実作業では生産効率を向上するために種々のツールを使用している。これらを新人研修にも適用することで上述の問題を解決しようとした。これらのツールの多くはｕｎｉｘ上で動作するので，開発支援環境としては，ｕｎｉｘとした。講師は研修の評価をするために，新人は単体試験を実施するために，表１のツールを使用する。[1][2][3][6][7][8]また，技法はトップダウン設計法とＨＣＰチャートを使用する。

表１　使用ツール

ツール名		概　　要
講師使用	静的解析 ﾂｰﾙ （複雑度測定） (ｿｰｽｺｰﾄﾞﾁｪｯｸ)	■ﾌﾟﾛｸﾞﾗﾑ 作成時に要した努力量や，分岐数とｽﾃｯﾌﾟ 数の指標による有効な工夫量の測定 ■ｺｰﾃﾞｨﾝｸﾞ規約違反ﾁｪｯｸ
	ﾌﾟﾛｸﾞﾗﾑ 製作管理支援 ﾂｰﾙ	■ﾓｼﾞｭｰﾙ単位のｺﾝﾊﾟｲﾙ回数・修正回数の測定
新人使用	ﾓｼﾞｭｰﾙﾃｽﾄｼｽﾃﾑ	■会話形式でﾄﾞﾗｲﾊﾞ・ｽﾀﾌﾞ作成 ■分岐網羅度，実行回数の測定
	ﾌﾟﾛｸﾞﾗﾑ 図生成 ﾂｰﾙ	■作成したｿｰｽﾌﾟﾛｸﾞﾗﾑと、ﾌﾟﾛｸﾞﾗﾑ図が一致しているか確認
	品質管理 ﾂｰﾙ	■S/W の障害ﾃﾞｰﾀ一覧表・障害件数の時系列ｸﾞﾗﾌ・障害件数の予測ｸﾞﾗﾌ等の作成

３．１　ツールの効果

（1）ソースプログラムを，複雑度測定ツールにかけることで，今まで人手で調査してきたプログラムの複雑度を，簡単に測定できる。

（2）試験終了の判定には，C_1 カバレッジが，８５％以上という設定ができる。

これらの定量的なデータをもとに、研修の評価・分析をすることが可能となった。

４．ツールを使用しての評価事例

４．１　負荷測定

コンパイルエラーがなくなった時点で，複雑度を測定した結果の一部をまとめたのが表２である。各モジュールのサイズや分岐数，各モジュール作成時の努力量や複雑度がわかり，この結果から，

（1）モジュール分割は適当だったか

（2）共同で作業する上で，班内で決めている分担に片寄りがないか

が判断できる。

表２　複雑度測定結果（一部）

項目		A班		
		ﾓｼﾞｭｰﾙa	ﾓｼﾞｭｰﾙb	ﾓｼﾞｭｰﾙc
参照データ		種類/ 回数		
	総数	16 / 70	3 / 7	10 / 22
	ｺﾒﾝﾄ無	16 / 16	3 / 7	10 / 22
	ﾎﾟｲﾝﾀ	0 / 0	0 / 0	0 / 0
	ｲﾝｸﾙｰﾄﾞ	0 / 0	0 / 0	0 / 0
	共通ﾃﾞｰﾀ	7 / 7	3 / 7	7 / 9
サイズ		文/ 行		
	手続	50 / 81	15 / 23	23 / 53
	宣言	3 / 3	3 / 3	5 / 5
	実行	46 / 50	12 / 12	17 / 18
	ｺﾒﾝﾄ	3 / 12	1 / 6	8 / 15
命令		種類/ 出現数		
	ｵﾍﾟﾚｰﾀ	18 /134	7 / 28	23 / 98
	ｵﾍﾟﾗﾝﾄﾞ	80 /132	15 / 26	33 / 43
	Halstead	24529.39	1460.90	7955.14
分岐	出現数	4	1	6
	総数	4	1	6
	最大値	1	1	1
	McCabe	6	3	8
	GOTO文	0	0	0
呼出し	種類	1	1	7
	出現数	2	9	8
	ﾊﾟﾗﾒｰﾀ総数	14	12	2

しかし，プロジェクト体験実習の期間が約１ケ月と短いために，モジュール分割が不適当と判断した場合でも，モジュール分割を再度行わせてはいない。ただしこの評価結果から，各班毎に新人が作成した工程計画の見通し，見直しについて，適切な指導を行っている。

４．２　品質

（1）２年間のデータから，複雑度とケース数に相関関係があることがわかった。試験要領書レビュー時に，各モジュールの複雑度とケース数の相関関係を調べることで，ケースの洗い出しが妥当かどうか指導を行っている。

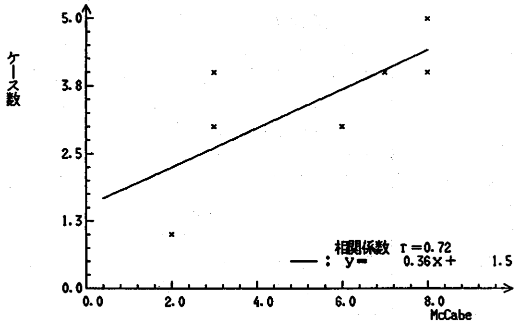

図２　相関関係グラフ（６２年度　Ａ班）

（2）各モジュール毎，試験ケースに対する C_1 カバレッジ（分岐網羅度）を測定し，その結果が８５％以上かどうかで，試験ケースの洗い出しの評価を行っている。

```
(C) COPYRIGHT 1985 [,1986 ] MITSUBISHI ELECTRIC CORPORATION ALL RIGHTS RESERVED
C TESTBED COVERAGE ANALYZER                          Histogram Report Page  1

 histogram of execution times
 ============================

 module name ; gcard
 ------------------------------------------------------------------------------
 branch  number  I   number of hits -- normalized to maximum
 number  of hits I.---10.---20.---30.---40.---50.---60.---70.---80.---90.--100.
 ------------------------------------------------------------------------------
    1          1 I**
    2 *        0 I
    3          1 I**
    4          1 I**
    5         43 I**************************************************
    6          1 I**
    7         42 I*************************************************
    8         42 I*************************************************
    9         21 I*************************
   10 *        0 I
   11         21 I*************************
   12         21 I*************************
   13         21 I*************************
   14         42 I*************************************************
   15          1 I**
 ------------------------------------------------------------------------------
                 258.  = total of execution
  total of     2 not hit     hit   13/  15    percent hit =     86.67
```

図３　カバレッジ測定結果

（3） 試験実施中に，障害件数の予測グラフを出力することで，障害予想件数と試験収束日の見通しの指導を行っている。

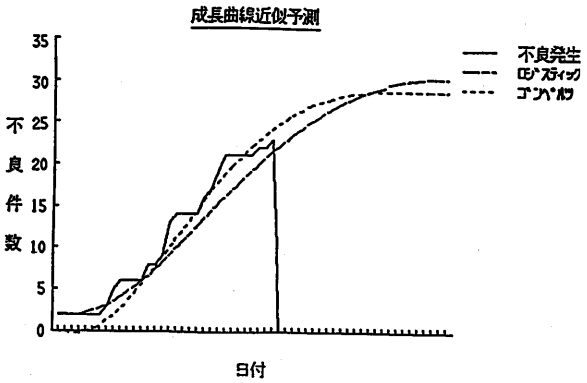

図４　障害件数予測グラフ

５．教育への効果的ツール使用法

５．１　班別比較

すべて同じ環境での作業なので，班毎のデータ（プログラムステップ数，複雑度，ケース数等）を比較して，大きな差がないか分析できる。

この分析結果をもとに，班毎に大きな差が出ないような教育ができたか検討することで，研修スケジュール，講義内容，実習テーマ，教育ポイント　等にフィードバックがかけられる。

プロジェクト体験実習進行中には，以下のような分析もできる。

(1) プログラムステップ数

講師側で予想したステップ数と比較することで

(a) 予想ステップ数より多い場合

・無駄な処理があるのではないか

・モジュール間インタフェースが多すぎないか

(b) 予想ステップ数より少ない場合

・処理に抜けはないか

そして，予想ステップ数の妥当性も分析できる。その結果は，次作業（この場合は，単体試験要領書作成）へ反映させる。

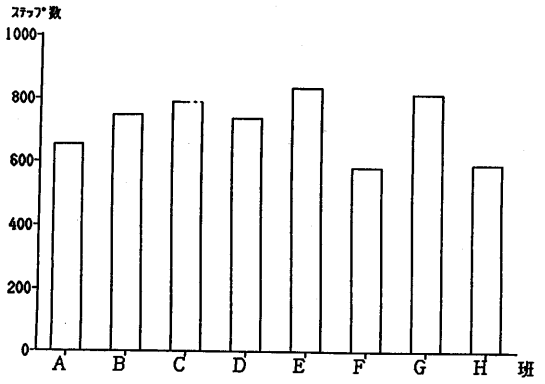

図５　ステップ数の比較

(2) 複雑度とケース数

各班の複雑度とケース数の相関関係を比較することで，

(a) 複雑度に対してケース数が多い場合

・重複したケースを設定しているのではないか

・無駄な処理が多いのではないか

(b) 複雑度に対してケース数が少ない場合

・確認しているケースが少ないのではないか

が分析できる。

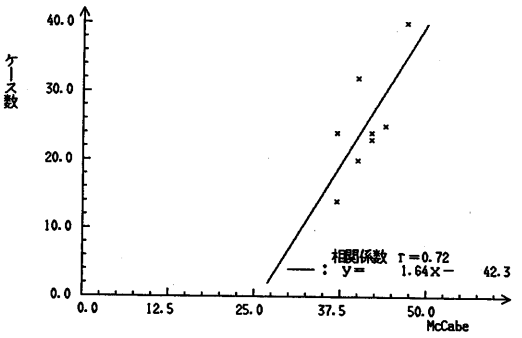

図６　相関関係グラフ

５．２　年度比較

６１年度と６２年度の相違点は，使用言語（C，FORTRAN77 ）だけであり，課題は同じであったので課題に対する言語比較ができた。

６３年度は，C言語で実施中であるので，研修終了後には，研修効果の年度比較　等もできる。

表３　年度（言語）比較

比較項目		FORTRAN77	C
年　度（班人数）		61年度 (3人×11班)	62年度 (3人× 8班)
期間/ 工数(日/ hr)		20.0/369.6	18.5/371.9
総ｽﾃｯﾌﾟ数 ( ) 内はｺﾒﾝﾄなし		907.5 (682.1)	712.5 (643.1)
ﾓｼﾞｭｰﾙ数　（平均）		4.5	7.1
複雑度	Halstead（平均）	48031.82	134484.02
	McCabe　（平均）	15	41
生産性　（ｽﾃｯﾌﾟ/hr）		1.8	1.7
障害件数　（平均）		15	8

（課題が FORTRAN77向きであったということを考慮して表３をみて欲しい）

６．おわりに

今後もプロジェクト体験実習を実施する中で，次のような目的で，ツールを使用した教育を考える。

（1） ＯＪＴ育成計画表作成

班毎だけでなく，個人毎のデータを収集し，そのデータ履歴から，プログラム作成能力　等の成長度合を評価し，研修終了後，ＯＪＴでの要教育ポイントを明確にした育成計画表を作成する。

（2） ツールの普及

新人自身が，ツールの操作法や有効な使用法を理解し，自由に使いこなせるようになる迄の時間を短縮する。そして、ＯＪＴに入っても作業効率を向上するためにツールを実作業に適合させて，使用できるようにする。

（3） 講義内容へのフィードバック

５章で述べたように，ツールを用いて定量的，客観的に評価をすることは，教育そのものの評価にもつながる。その結果から，どこの理解度が低いか，どこでつまずいているかを把握する。そして，講義内容のポイントを設定し，担当講師が代わっても，同じポイントで講義ができるようにする。

参考文献

（1） 田野口他　「ソフトウェア開発における製作管理支援の機械化－実現方式－」　情報処理学会第３１回全国大会

（2） 古川他　「静的解析システムＭＥＬＣＡＴ（４）－ＳＣＯＰ－」情報処理学会第３２回全国大会

（3） 渡辺他　「ＵＮＩＸでのテスト支援環境の構築　－ツール概要－」　情報処理学会第３３回全国大会

（4） 武田他　「ソフトウェア開発における工程管理支援の機械化－概要と特徴－」　情報処理学会第３３回全国大会

（5） 関根他　「ソフトウェア開発における工程管理支援の機械化－実現方式－」　情報処理学会第３３回全国大会

（6） 久保他　「プログラム図生成ツール」ソフトウェア・ツール・シンポジウム’87

（7） 山本他　「品質管理ツールの評価－普及のための設計方針－」　情報処理学会第３４回全国大会

（8） 小越他　「品質管理ツールの設計についての一考察－管理支援ツールの設計指標－」情報処理学会第３５回全国大会

（9） 石川他　「開発支援環境を教育するための考察－プログラミングの評価－」情報処理学会第３６回全国大会