通信制御ソフトウエア開発における プログラマ個人差の分析

田村悦夫（富士通(株) ソフトウエア事業部）

1. はじめに

ソフトウエアの開発においては，プログラマ個人の能力に依存する要素が多く，プログラマの個人差がソフトウエアの信頼性およびソフトウエア開発の生産性におよぼす影響は少くない。

一方，ソフトウエアが大規模化，複雑化するにつれ，その高信頼性，高生産性に対する要求は切実となりつつあり，プログラマの個人差を吸収できるソフトウエア開発管理技法の確立が望まれている。

ここでは，通信制御ソフトウエア開発における製造工程の中で収集したデータをもとに，プログラマの個人差の実態について調査・分析し，更にこれらの個人差を吸収し高信頼性，高生産性ソフトウエアを開発する上で有効と思われる開発管理技法について分析・評価する。

2. 個人差の実態とその分析

2.1 プログラマの能力の定義とその定量的把握

プログラマの能力は，知識，経験，性格などのさまざまな要素より成り立っているものと考えられ，一般的にその定量的把握は簡単ではないが，ここでは能力をプログラミング力に限定し，定量的把握を試みる。

すなわち，プログラムの開発過程（大きく設計，製造，保守の3工程から構成される）のうち，製造工程で検出したプログラマの個人別プログラム・バグに着目し，定量化を試みるものである。

プログラミング能力

製造工程の作業は大きく分けて，仕様書に沿ってジェネラル・フロー・チャートを作成する詳細設計作業，ジェネラル・フロー・チャートに従ってコーディングを行うコーディング作業，およびプログラムのデバグ作業に分けられる。これらの作業におけるプログラマの能力を表1のように定義して定

表1. プログラミング能力の定義と定量的把握法

		能力の定義範囲	定量的把握方法
プログラミング能力	設計力	仕様書に従ってフローチャートに記述する力	KS当りの論理ミス件数
	コーディング力	フローチャートに従ってプログラミング言語で記述する力	KS当りのコーディングミス件数
	デバグ力	検査工程以前にバグを検出し修正する力	KS当りの検査工程で発見された全種のバグ件数

量化する。

2.2 個人差の傾向

設計力，コーディング力およびデバグ力についての個人差を，ソフトウエア経験年数との関係において調査した結果，次のような傾向のあることがわかった。（図1，図2および図3参照）

(i) 設計力はソフトウエアの経験を積んでも必ずしも向上していない。本例では個人差が3倍以上ある場合がある。

(ii) コーディング力はソフトウエアの経験を積むにつれ向上する傾向がみられる。本例では個人差は大体4倍以内におさまるが，中には特にコーディング力の劣るプログラマもおり，この場合は10倍位の個人差がある。

(iii) デバグ力はソフトウエアの経験を積んでも必ずしも向上していない。本例では10倍以上の個人差のある場合がある。

3. 生産性，信頼性と個人差の関係

3.1 定義と定量的把握

プログラミング能力を総合的に評価する場合，プログラミングの生産性，プログラムの信頼性の面から評価するのが一般的であると思われる。

ここでは生産性，信頼性について，それぞれバグ検出工程とプログラミング力の関係において，次の様に定義する。

図1 設計力の個人差

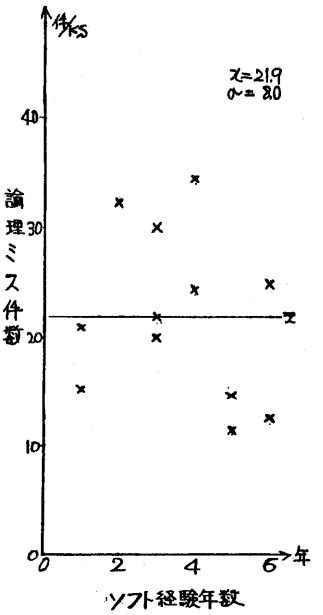


図2 コーディング力の個人差

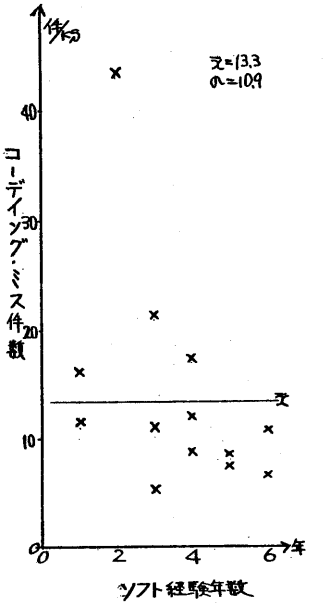
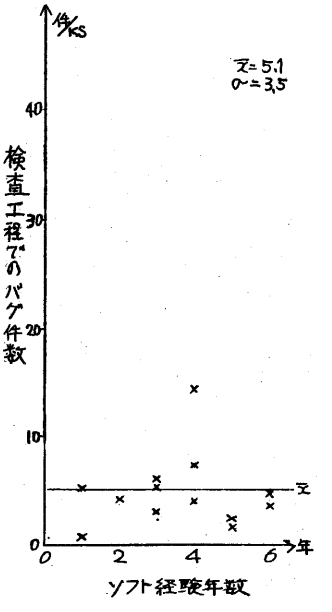


図3 デバグ力の個人差



(i) プログラミングの生産性

製造工程におけるバグの検出および解決に要する工数は，その時期が早い程少くてすむ傾向のあることが分った。(表2 参照)

ここでプログラマの生産性指標を，

$$\Sigma\left[\left(\frac{\text{バグ1件の検出・解決工数}}{*}\times \text{バグ件数}\right)_S\right]$$

*表2による値

でもって定義することにする。

すなわち，プログラミング能力が高くバグの絶対件数の少いプログラマ，およびバグを早い工程で検出できるプログラマを，生産性が高いプログラマと評価する。

表2 バグの検出・解決に要する工数 (人/件)

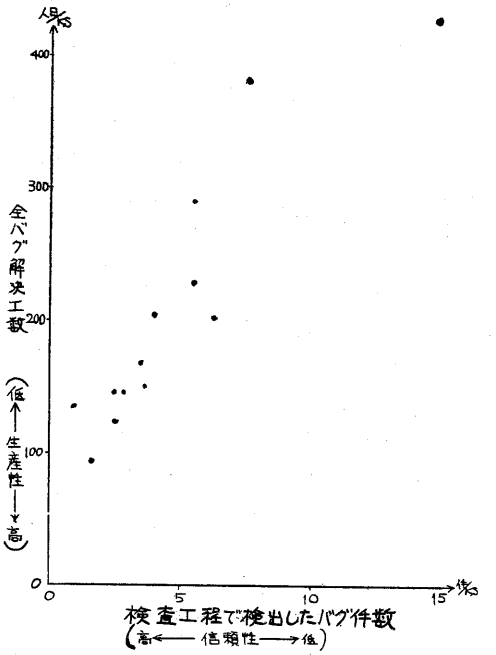
工程	机上レビュ	デバグ	検査
バグの検出,解決に要する工数	0.6	5.6	17.6

(ii) プログラムの信頼性

製造工程でのバグの多寡にかかわらず，バグを検査工程前に解決することができれば，そのプログラムの信頼性は高いと評価できる。

すなわち信頼性の指標を『検査工程で検出されるバグ件数』でもって定義する。

図4 生産性，信頼性の個人差



3.2 生産性，信頼性の個人差

以上の定義をもとにして，生産性，信頼性と個人差の傾向を分析すると，以下のような傾向がみられる。(図4 参照)

(i) プログラマの生産性の個人差は4倍以上ある場合があり，またプログラムの信頼性の差は10倍以上ある場合があり，各々かなりのバラツキがある。

(ii) 生産性の高いプログラマは，信頼性の高いプログラムを作るという傾向がみられる。

4. 個人差を吸収する開発管理体制

4.1 開発体制

以上より，プログラマのプログラミング力および生産性，信頼性の個人差のバラツキがかなり大きいことが分った。

しかし，ソフウエア開発プロジェクトでは，プログラムの機能とかモジュールの単位でグループに分割して作業を進める場合が多い。それで個人差のあるプログラマがこのようなグループ作業を進める場合，グループ構成(開発体制)

の違いにより生産性，信頼性にどのように影響を与えるかについて調査した。

通信制御ソフトウエア開発プロジエクトは2回編成された。最初のプロジエクトではハードウエア，ソフトウエア的に新しい分野であり，そのノウハウの蓄積が少くグループの全員が同じレベルで製造作業を分担した。
その結果は特定の個人，又はモジュールの信頼性が低く，それがプロジエクトのネックになるという問題が発生した。
次のプロジエクトは最初のプロジエクトとほぼ平行的に作業が進められたため，核要員として最初の経験者を何名か投入したがやはり経験の浅いプログラマの率が高くならざるを得なかった。
このため従来のプロジエクトを反省し，各グループ毎にこれらの経験者を核要員として配し，直接プログラム製造作業は分担させず，プログラマの技術指導と調整に専担する体制（スタッフ制と称す。4.2節参照）をとることにより対応した。
その結果は，プログラミング力の個人差がプロジエクト全体の生産性，信頼性に影響を与えることはなかった。

以上のことから個人のプログラミング能力と，グループの生産性，信頼性との関係を分析すると次のようになる。（表3 参照）

平均的にプログラミング力の高くないグループがスタッフ制を採用すると，グループとしての生産性，信頼性が向上し，スタッフ制を採用した効果が大きい。
また，平均的にプログラミング力の高いグループがスタッフ制を採用すると，信頼性に関してはあまり差はないが，生産性に関してはかえって低下する場合がある。

4.2 スタッフ制

次にスタッフ制の目的と，その構成および適用の範囲について述べる。

表3. 開発体制と生産性，信頼性

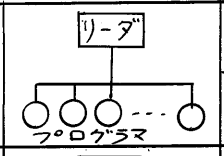
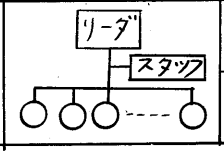
開発体制		プログラマの構成（人）					信頼性	生産性
		リーダ	スタッフ	プログラマ（プログラミング力） 高い	普通	低い	検査工程バグ件数（件/KS）	1日当り生産量（ステップ/人日）
スタッフ制を採用せず		1		2	1	1	3.3	4.6
		1		0	1	1	10.0	2.5
スタッフ制		1	1	1	2	0	3.5	4.0
		1	1	0	3	1	4.2	3.5

表4. スタッフ制の構成

担当	人数	担務	基準
リーダ	1	進捗管理/報告，管理業務，工数/マシン/工程計画 等	・プログラム全体の把握， ・指導性のある者
スタッフ	1	プログラマの技術指導， 机上レビュ，デバグ法検討	・全体把握と担当分の詳細把握 仕様決定に参加
プログラマ	2 〜 6	プログラミング（設計，コーディング，デバグ），保守 等	プログラマとしての適性

(i) スタッフ制の目的

プログラマの個人差を吸収するためには，HIPO，トップダウン等の生産技法の他に，開発体制面での考慮が必要と思われた。

一方プログラマはソフトウエアの経験を積んでも必ずしもそのプログラミング力の向上は期待できない面もあり，少数の優秀なプログラマを有効に活用するために，技術力があまり問題にならないような作業は他のプログラマに任せ，高度な技術を要する作業だけをこれらのプログラマに専任させるという体制をとったのがポイントである。

特に生産性と信頼性の向上に関しては，プログラマ個人のバグ絶対件数は減少させることができなくても，スタッフの技術指導によって，できるだけ早い工程でバグを解決させることでその目的を達成しようとするものである。

(ii) スタッフ制の構成

リーダ ---- グループが開発するプログラムを全体的に把握し，工数計画，工程計画等の調整を行い，作業の進捗を管理する。

スタッフ ---技術力の高いプログラマが担当し，仕様決定に参加してプログラム全体の把握をすると共に，担当分の詳細を把握する。

仕様の充足性，プログラム・インタフエースのチエック，試験項目のチエック等に関して指導をし，デバグ法の検討を通して次工程で行なう作業の準備をする。

机上レビューを指導すると共に，自らもプログラム全体の立場から机上レビューを行なう。

プログラマ--- プログラムの製造作業を担当するが，グループの構成はスタッフの技術指導の範囲から，5，6名位までの人数と思われる。

以上，スタッフ制の構成を表4に示す。

(iii) 評価 ： 開発体制としてスタッフ制を採用した方が効果があると思われる場合は，次の様な場合である。

① グループのプログラマの平均的なプログラミング力が低い場合。

② プログラマのプログラミング力が充分に把握できないような場合。

4.3 まとめ

通信制御ソフトウエアの開発という限られた範囲と，適用ケースが少いということはあるが，プログラマの個人差の実態について調査し，個人差はかなり大きくまたこれはソフトウエアの経験年数によっても必ずしも解決されないことが分った。

またこのような個人差を吸収し，高生産性，高信頼性ソフトウエアを開発するための開発管理体制としてスタッフ制について評価し，プログラミング力の低いプログラマのグループに対しては有効な開発管理体制の一つであることが分った。

参考文献

1. F.B.Brooks Jr "The Mythical Man-Month: Essay on Software Engineering"
2. F.T.Baker "Chief Programmer Team Management of Production Programming" IBM System Journal Vol.11 No1.1972
3. Mills H., "Chief Programmer Teams, Principles, and Procedures," IBM Federal Systems Division Report FSC 71-5108, 1971
4. B.W. Boehm, et al. "Quantative Evaluation of Software Quality" Proc. 2nd Int. Conf. on Software Eng., OCT. 1976