ソフトウェア工学 86−12
（1992．7．21）

ソフトウェア開発実験に基づく構造化分析／設計手法の評価

岡　敦子　　山本修一郎　　磯田定宏
NTTソフトウェア研究所

抄録

本論文では、構造化分析／設計手法と従来手法を用いてソフトウェア開発実験を実施した結果、　(1) SA手法を用いて開発したソフトウェアの品質が従来手法を用いて開発したソフトウェアの品質よりも高い　(2) 機能レビュ時の仕様品質と機能テスト時のプログラム品質にはどちらの手法を用いて開発した場合にも正の相関がある　(3) SA手法では、適用回数に従って仕様品質が向上することを明らかにした。さらに、SA手法を導入した場合、1人当たり1週間程度の学習工数が必要となるが、プログラム品質が約2倍に向上することが得られた。

An Experimental Evaluation of Structured Analysis and Design Methodology

Atsuko Oka　Shuichiro Yamamoto　Sadahiro Isoda
Software Laboratories
Nippon Telegraph and Telephone Corporation

Abstract

Software development experiments are carried out to evaluate effectiveness of the Structured Analysis and Design (SA/SD) Methodology by comparing with the traditional methodology. This paper represents the following results.

(1) The software developed with SA/SD methodology has higher quality than software developed with traditional methodology.

(2) There is a positive correlation between the quality of specifications at the functional review and that of programs at the functional test, irrespective of the software development methodologies.

(3) In the software development using SA/SD methodology, the specification quality is improved in proportion to the times of appication.

This paper also makes it clear that one weak is required to learn the SA/SD methodology and that the software developed with SA/SD methodology has twice higher quality than the software developed with the traditional methodology.

1 はじめに

要求分析／設計仕様を図形を用いてわかりやすく表現する手法として構造化分析／設計(SA/SD)手法[1]が提案されている。SA/SD手法（以下、SA手法）では、データの流れに着目したデータフロー図を用いてシステムの仕様を階層的に分析し、モジュール間の呼び出し関係に着目した木構造図を用いて、プログラム構造を階層的に設計する。

これまでにSA手法をベースとするCASEツールが実用化されているが、その効果については必ずしも明確ではなかった。このため、以下のような実証的な研究が行われてきた。

(1) SA手法を商用システムに適用した結果を定量的に分析する[2][3][4][5]。

(2) SA手法と従来手法を用いてソフトウェア開発実験を行い、両者を統計的に比較分析する[6][7]。

(1) の研究では実際のソフトウェア開発に即した定量的なデータを入手できるが、ソフトウェア開発過程を制御できないため、SA手法と従来手法との厳密な比較ができないという問題がある。

また、(2) の研究ではソフトウェア開発過程を実験的に制御できるので、SA手法と従来手法とを厳密に比較できるが、①被験者がソフトウェア開発の初心者である　②開発対象規模が小さいため実際のソフトウェア開発に即したデータが得られない　という問題がある。

これらの研究がもつ問題点を解決するために、筆者らは経験のあるソフトウェア開発担当者を被験者とし、ソフトウェア開発実験を実施した。本論文では、仕様およびプログラム品質／開発工数／生産性／学習効果に関して、SA手法と従来手法による影響を統計的に比較分析する。

2. 実験方法

2.1 使用ツール

SA手法による開発では、統合化CASE SoftDA[8]の構造化分析／設計支援サブシステムSoftDA/SAを用いて、従来手法による開発ではパソコン上のワープロを用いて各手法にしたがった分析／設計ドキュメントを作成した。試験工程では、テストカバー率を測定するため、両手法ともにテストカバー率測定サブシステムSoftDA/TCAを用いた。

2.2 実験対象システム

実験対象システムは、同程度の難易度を持つ中規模（C言語,約2.5KLOC）のトランザクションシステム（技術会議管理システムと図書室管理システム）である（付録1）。各チームは、日本語で記述された要求仕様に基づいて、要求分析／設計／製造／試験の各工程を経て、SUN上でコンパイル／実行可能なプログラムを開発した。

2.3 被験者

C言語によるプログラミングならびにソフトウェアの開発経験者（1年～7年）12名を被験者とした。1チームを2名の被験者から構成し、6チーム（SA手法：4チーム、従来手法：2チーム）を構成した。C言語についての予備的なプログラミング能力テスト、レビュ能力テスト、経験年数に関して、SA手法チームと従来手法チームとの間に有意な差はなかった（有意水準5%）。

2.4 実験手順

各チームが開発したシステムと用いた手法を表1に示す。ここで、SA手法を用いた4チームをSA-n (n=1,2,3,4)、従来手法チームをJP-n(n=1,2)で示す。

表1　開発チームと開発対象システムの関係

システム名	SA手法	従来手法
技術会議管理ｼｽﾃﾑ	SA-1,SA-2	JP-1
図書室管理ｼｽﾃﾑ	SA-3,SA-4	JP-2

開発チームは、以下の手順に従って、システムの開発を行った。

[手順1] SA手法の学習

SA手法を用いる4チームに対して、SA手法とSoftDA/SAについて3日間の講習会を実施した。

[手順2] 分析工程

各チームごとに、ユーザから提示された要求仕様を分析した。分析工程では、対象システムの外部入出力の明確化、機能分析、各機能の入出力の明確化を行った。ただし、従来手法チームでは、

各仕様書を自然言語（日本語）による文章で記述した。必要があれば、テーブル設計書などの表を用いた。

分析過程で不明な点が発生した場合、質問票を作成し、その内容をユーザ（本実験の実施者）に照会した。分析が終了した後、チーム内でドキュメントを読み合わせ、分析内容をチェックした。そのチェック内容を吸収した段階で、ユーザが予め作成しておいたチェック項目（約30件）に従って機能レビュを行った（付録2）。機能レビュでは、機能の異なるプログラムが開発されると、最終的な品質／工数を比較できないため、各チームの分析結果を統一するように調整した。

[手順3] 設計工程

機能レビュでの指摘項目を修正した後、詳細設計を行った。設計工程では、モジュール構造、各モジュールの仕様、モジュール間の入出力データ、および、各データの型／サイズなどのデータ仕様を設計した。なお、SA手法チームでは、SoftDA/SAのモジュール構造図（SC）自動生成機能を用いてDFDからSCを自動生成した後、モジュールの詳細化、統合などの整形を行い、SCを完成させた[9]。従来手法チームでは、自然言語、または、HCP[10]などのチャートを用いてモジュール仕様を記述した。設計が終了した後、チーム内レビュを実施し、その結果を反映した上で、設計レビュを実施した（チェック項目数：約10件）。

[手順4] 製造工程

設計レビュでの指摘結果に基づいて必要な修正を実施した後、プログラムコードを作成した。

[手順5] 試験工程

テスト仕様を作成し、テストカバー率80%を達成させるエッジテスト、および、デバグを行った。チームごとにコーディング量に差が生じたため、テスト対象とする機能項目を約10種類に限定した。試験項目数を200件／Kstepとした。

[手順6] 機能テスト

試験が完了したプログラム、および、分析工程終了時（機能レビュ時）のドキュメントに対して、テスト対象機能に限定して、ユーザ側で機能テストを実施した（テスト項目数：約50件）。機能テスト項目例を付録2に示す。

3 実験結果の分析と考察

3.1 開発データ

開発実験で得られた工数、規模および生産性に関するデータを表2、表3に示す。なお、表2では講習会によるSA手法の修得や、マニュアルによるツール操作法の学習に要する工数を学習工数とした。

表2 開発工数

項目＼チーム		SA手法				従来手法	
		SA-1	SA-2	SA-3	SA-4	JP-1	JP-2
1）	全工数（人時）	706	653	826	648	716	585
2）	学習工数（人時）	56.4	48.4	89.4	53.5	10.3	3.0
3）	正味工数（人時）*1	650	652	784	642	706	582
4）	学習工数の割合(%)	8.0	6.9	10.2	7.7	1.44	0.51

（注）*1：全工数から学習工数を控除した正味の開発工数

表3 開発規模と生産性

項目＼チーム		SA手法				従来手法	
		SA-1	SA-2	SA-3	SA-4	JP-1	JP-2
1）	開発規模（Kstep）	6.3	2.6	2.6	4.2	2.5	2.9
2）	抽出 *1 機能数	12	24	24	31	13	12
	手法平均	22.8				12.5	
3）	検査規模（Kstep）	3.5	2.3	2.5	1.6	2.4	2.0
4）	生産性*2（Kstep/人年）	11.0	7.2	6.5	5.1	6.9	7.0
	手法平均	7.5Kstep/人年				7.0Kstep/人年	

（注）*1：SA手法では最下位のプロセス数
*2：検査規模／正味工数により算出

[観察1] 開発支援ツールの学習工数は、1人平均1週間程度である（表2-（3、4））。

従来手法ではテストカバー率測定ツールSoftDA/TCAを、SA手法では、SoftDA/TCAとSoftDA/SAを使用した。講習会によるSA手法の修得やマニュアルによるツールの操作法の学習に要

する工数は、従来手法チームでは2%以下、SA手法チームでは10%程度である。このうち、SA手法およびSoftDA/SAの学習にかかる工数は8%程度と考えられる。ただし、これらの学習工数は、開発規模／開発工数全体に依存しない講習会、および、ツールの使用方法に馴れるまでに要する固定的な工数である。したがって、SA手法の学習工数は多くても1チーム約80時間、1人平均1週間程度と考えられる。

3.2 品質分析

機能レビュ時の平均誤り発生率、および、機能テスト時の平均誤り発生率を用いて、SA手法の品質が従来手法と比較して高いかどうかを検定した（表4）。

表4 品質の検定結果

種別／テスト	機能	ﾃﾞｰﾀ	制約条件	全体	備考
機能レビュ	◎	○	×	○	図1,2
機能テスト	×	×	◎	◎	図1,3

◎：有意水準5%でSA手法の品質が高い。
○：有意水準10%でSA手法の品質が高い。
×：有意な差はなかった。

[観察2] 機能レビュ時では、SA手法の仕様品質が従来手法の仕様品質より高い（図1）。

SA手法の平均誤り発生率（51%）が従来手法の平均誤り発生率（76%）より低く、品質が高い(有意水準10%)。

[観察3] SA手法による分析では、従来手法による分析と比較して、機能に関する抜け／もれが少ない（図2）。

SA手法では、機能レビュ時における機能に関する誤り発生率が従来手法の半分以下（約30%）であり、両者には有意な差があった（有意水準5%）。この理由は次のようであると考えられる。すなわち、SA手法では機能仕様をデータフロー図を用いて記述するため、データに着目した分析を行うようになる。この結果、データの発生／加工／消滅に必要な機能を検討するようになり、必要な機能をもれなく抽出できたためと考えられる。

[観察4] 機能テストでは、SA手法のプログラム品質が従来手法のプログラム品質より高い（図1)。

SA手法の機能テストでの平均誤り発生率(19%)が従来手法の平均誤り発生率（36%）より低く、プログラム品質が高い（有意水準5%）。

[観察5] SA手法の方が従来手法と比べて、制約条件に関する機能テスト品質が高い（図3）。

テストの種別ごとの平均誤り発生率を比較すると、制約条件に関しては、SA手法の平均誤り発生率（31%）が従来手法の平均誤り発生率（48%）と比べて低い（有意水準5%）。制約条件に関するレビュ／テスト項目では、データの二重登録に関する扱いなど複数のデータや処理に関する複合的な条件をチェックする。SA手法では、このような制約条件を定式的に記述する方法がないにもかかわらず、制約条件に関する問題について従来手法より誤り発生率が低いという実験結果となった。この理由は次のようであると考えられる。すなわち、各機能に対するプロセスの入出力データをデータフロー図を用いて明確に記述することにより、データを常に意識して分析を行うことができる。この結果、データの入力により何が起こるか、機能の実行後に出力されるデータは何か等を考慮しやすいためである。

制約条件に関する誤りをさらに削減するためには、制約条件の定式的な記述方法、および、これらの誤りをチェックするためのレビュ方法を検討する必要がある。

[観察6] 機能レビュ時の誤り発生率と機能テストでの誤り発生率には正の相関がある（重相関係数：0.80）（図4）。

SA手法だけに着目するとさらに高い相関が見られる（重相関係数：0.87）。すなわち、機能レビュ時に誤り発生率が低い場合は、機能テスト時の品質が高くなることが予想される。

一方、エッジテストでのバグ数／KLOCと、機能テスト時の誤り発生率との間に相関はない（重相関係数：0.36）。これは、エッジテストでのテスト項目を設計者が作成しており、そのテスト内容が機能レベルで不十分であったことに起因する。エッジテストを効果的に行うためには、機能レベルで十分なテスト項目を作成する必要がある。

3.3 工数／生産性分析

各チームごとの工数、ならびに各工程ごとの工数の割合を図5に示す。

[観察7] 各工程ごとの工数の割合に関して、手法間に有意な差がない（有意水準5%）。

[観察8] プログラムの生産性に関して、手法間に有意な差がない。（表3-（4））

一般にSA手法では、上流工程での誤りや抜けが防げるため、試験工程で発生するバグに伴う上流工程ドキュメントへの手戻りが減少すると言われている。この結果、開発工数に対する試験工程の割合が減少し、全体の開発工数が減少するため生産性が向上すると考えられている。実験工数上の制約のため、本実験では機能テスト後のバグ吸収作業を実施していないため有意な差がなかったと考えられる。

ただし、機能テスト時の品質が従来手法よりSA手法の方が高いため、機能テスト後のバグ吸収後の工数を補正した総合的な生産性はSA手法の方が従来手法よりも高くなると思われる。

4 分析作業の学習効果

開発実験を行った後、手法の違いが分析作業経験に与える影響を評価するため、分析だけを行う実験を続けて2回実施した。分析対象システムは、同程度の難易度および規模のトランザクションシステム（医薬品在庫管理システム、部品の発注管理システム）である。各分析工程終了時には、機能レビュを行った（チェック項目：約30件）。以下では、開発実験時の分析作業を含めて合計3回の分析作業結果を分析する。

4.1 分析品質

分析経験と分析品質との間の相関を調べるため、第1回目の機能レビュ誤り発生率が第3回目の機能レビュ誤り発生率より低いかどうかを検定した（表5）。

[観察9] SA手法では、分析経験と分析品質の間には正の相関がある（図6～9）。

SA手法による分析では、第3回目の機能レビュ誤り発生率は第1回目の平均誤り率から、約45%減少し、両者の間には有意な差があった。

表5 検定結果

	SA手法	従来手法	備考
機能	◎	○	図7
データ	◎	○	図8
制約条件	×	×	図9
全体	○	×	図6

◎：有意水準5%でSA手法の品質が高い。
○：有意水準10%でSA手法の品質が高い。
×：有意な差はなかった。

（有意水準10%、図6）。すなわち、SA手法では分析経験を積むに従って、分析品質が向上する。

SA手法では、機能に関して64%、データに関して76%平均誤り発生率が低下している（図7,8）。[観察3]で述べたように、SA手法では、図形を用いることによって、機能の抜けやデータに関する誤りを検出しやすい。また、分析経験を積むに従って、図の見方に馴れてくるため、さらに品質の向上が見込まれると考えられる。

一方、SA手法での制約条件に関する平均誤り発生率は約60%減少しているが、有意な差はなかった（図9）。これは[観察5]で述べたように、制約条件について定式的に記述できないため、顕著な品質の向上はみられなかった。

4.2 分析工数

各分析対象システムごとに作成規模が異なるため、開発機能数により正規化した。各分析ごとの1機能当たりの分析工数を図10に示す。ただし、SA手法の場合には、プロセス仕様を記述する最下位プロセス数を用いて正規化している。

[観察10] 分析経験と分析工数の間には負の相関がある（図10）。

SA手法、従来手法ともに、第3回目の1機能当たりの分析工数は第1回目より少ない（有意水準10%）。これは、類似のシステムを繰り返し分析することにより、必要な機能、データを抽出しやすくなったためと考えられる。たとえば、何らかの登録機能が必要となる場合、そのキャンセル機能が必要となることは各システムに共通である。このように、どちらの手法を用いても、システム間で類似

する機能の分析工数が2回目以降は不要になるため、分析工数が減少する。

5 SA手法の効果

本実験では、1人当たり1週間程度の学習工数が必要となるが、SA手法を導入することにより、機能テスト時のプログラム品質が従来手法に比べて約47%向上するという結果が得られた（[観察4]）。これまでの評価では、平均誤り発生率が約25%減少し、品質が向上すると報告されていたが[6]、それぞれの手法で開発したプログラム品質を比較評価した結果ではない。これに対して、本実験では同一仕様のプログラムに対して両手法を適用したため、より精密にSA手法と従来手法との差が立証できたと考えれる。

SA手法では分析経験を積むに従って、機能レビュ時の仕様品質が向上する（[観察9]）。一方、機能レビュ時の仕様品質と機能テスト時のプログラム品質には正の相関がある（[観察6]）ため、プログラム品質も向上すると考えられる。

工数については、分析経験にしたがって分析工数が減少することが明らかになった（[観察10]）。さらに、2回目以降は学習工数も減少するため、SA手法では、その経験を積むことにより、プログラム品質、生産性が向上すると思われる。

[観察7,8]で述べたように、開発工数、生産性については、手法間に有意な差が認められなかった。今後は機能テストで発見されたバグの吸収工数も考慮した、より精密な実験を行う必要がある。

6 おわりに

本論文では、構造化分析／設計手法と従来手法を用いてソフトウェア開発実験を実施した結果、(1) SA手法を用いて開発したソフトウェアの品質が従来手法を用いて開発したソフトウェアの品質よりも高い (2) 機能レビュ時の仕様品質と機能テスト時のプログラム品質にはどちらの手法を用いて開発した場合にも正の相関がある (3) SA手法では、適用回数に従って仕様品質が向上することを明らかにした。さらに、SA手法を導入した場合、1人当たり1週間程度の学習工数が必要となるが、プログラム品質が約47%向上することが得られた。

今回の実験では、中規模プログラムを小人数で開発する場合について各手法を比較分析した。したがって、今回の評価結果がそのまま、大規模ソフトウェア開発へ適用できるかどうかについては、今後検討していく必要がある。

[参考文献]

[1]E.Yourdon, "Modern Structured Analysis", Prentice-Hall, 1989.

[2]P.Lempp, et al, "What productivity increases to expect from a CASE environment: Results of a user survey", ACM 27th Annual Technical Symposium, Jun, 1988, pp.1-7.

[3]T.Nakajo, "A case history analysis of software error case-effect relationship", IEEE trans. on Software Eng., Vol.17, No.8, 1991, pp.830-837.

[4]山本,磯田, "ｿﾌﾄｳｪｱの構造化分析／設計支援ｼｽﾃﾑを開発", NTT技術ｼﾞｬｰﾅﾙ, 8月号, 1988, pp.62-65.

[5]山本,国立 他,"構造化分析・設計手法に基づくｿﾌﾄｳｪｱ開発支援ｼｽﾃﾑSoftIDAの適用経験", NTT R&D, Vol.40, No.11, 1991, pp.1413-1422.

[6]J.Nosek et al, "User validation of information system requirements: some empirical results", IEEE trans. on Software Eng., Vol.14, No.9, 1988, pp.1372-1375.

[7]高橋,岡 他, "構造化分析／設計手法の一評価", ｿﾌﾄｳｪｱ工学研究会, Vol.92, No.10, 1992, pp.57-pp.64

[8]黒木,磯田, "統合化CASEシステムSoftIDAの機能とその評価", ｿﾌﾄｳｪｱ工学研究会, Vol.92, No.10, 1992, pp.17-24.

[9]西永,山本,磯田, "ﾓｼﾞｭｰﾙ構造設計支援機能の提案", ｿﾌﾄｳｪｱ工学研究会, Vol.91, No.85, 1992, pp.9-16.

[10]花田, "ﾌﾟﾛｸﾞﾗﾑ設計図法", 企画ｾﾝﾀｰ, 1983.

[11]V.R.Basili et al, "Experimentation in software engineering", IEEE trans. on Software Eng., Vol.12, No.7, 1986, pp.733-743.

[12]B.Bernstein, "Software Psychology - Human factors in computer and information systems", Winthrop Publishers, Inc., 1980.

付録1

1.技術会議管理システム要求仕様

技術会議は、複数の技術ｾｯｼｮﾝとﾁｭｰﾄﾘｱﾙから構成される。技術ｾｯｼｮﾝには、論文ｾｯｼｮﾝとパネルｾｯｼｮﾝがある。ﾌﾟﾛｸﾞﾗﾑ委員会は提出された論文に対する査読を行い、会議で発表する論文を選出し、それらを各論文ｾｯｼｮﾝに割り当てる。

ﾌﾟﾛｸﾞﾗﾑ委員会は、ﾁｭｰﾄﾘｱﾙを企画して２日間に分けて開催する。ﾁｭｰﾄﾘｱﾙには、標題(ﾀｲﾄﾙ)、講師、会議室、開催日時、定員が割り当てられる。各ｾｯｼｮﾝには、標題(ﾀｲﾄﾙ)、座長、会議室、開催日時、ｾｯｼｮﾝで発表される論文が割り当てられる。技術会議の公示を、会議の２カ月前に行ない、技術ｾｯｼｮﾝとﾁｭｰﾄﾘｱﾙの参加者を募集する。それぞれの参加費用は、ﾌﾟﾛｸﾞﾗﾑ委員会が決定する。参加費用は支払期日指定の銀行振込である。学生、各種学会会員、１カ月前までに振り込んだ人には、参加費用の割引き特典がある。参加ｷｬﾝｾﾙの場合には、ｷｬﾝｾﾙ手数料を差し引いた額を申込者に返却する。

会議の主催者側の業務を支援するｼｽﾃﾑを作成する。

2.図書室管理システム要求仕様

支店の図書室の利用しやすくするための図書室管理ｼｽﾃﾑを作成する。

この理由を以下に示す。ｾﾝﾀと支店に、それぞれ図書室がある。支店では図書室のｽﾍﾟｰｽが限られているため、古い雑誌を定期的に処分する。また、あまり利用されない図書や古い図書については廃棄処分にするか、ｾﾝﾀ倉庫に移送する。

支店では利用者が貸出返却ｶｰﾄﾞに記入せずに図書を持ち出したり、返却時に貸出ｶｰﾄﾞをそのままにしているために貸出状況の把握ができず整備に支障をきたしている。また、蔵書一覧もないので、所望の図書があるかどうかを調べる場合、書架をだけでなく、貸出ｶｰﾄﾞを調べる必要がある。しかも、全員が貸出ｶｰﾄﾞを提出しているとは限らないので、図書の検索が困難である。　利用者による購入対象は単行本と学会予稿集である。雑誌は毎年、社員にｱﾝｹｰﾄを実施し、利用頻度の低い雑誌の購読を中止する。要望の高い雑誌は新たに購読する。主要な学会予稿集、辞書や全集などは図書室が独自に購入する場合もある。

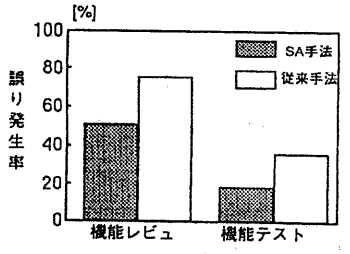

図1　平均誤り発生率

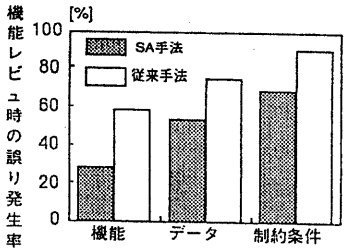

図2　機能レビュ時の側面別平均誤り発生率

付録2

付表1　機能レビュ／機能テスト項目

分類	概要	図書室管理システムテスト項目
機能	必要な機能があるか	著者名をｷｰに図書を検索できる
ﾃﾞｰﾀ	必要なﾃﾞｰﾀを管理しているか、ﾃﾞｰﾀ構造に必要なﾃﾞｰﾀ項目があるか	登録番号/著者分類番号を管理している
制約条件	ﾃﾞｰﾀの前提条件、実行後の制約条件が満たされているか	貸出時の貸出禁止図書かどうかチェック

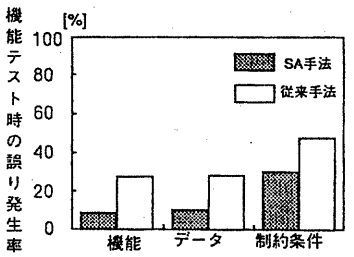

図3　機能テスト時の側面別平均誤り発生率

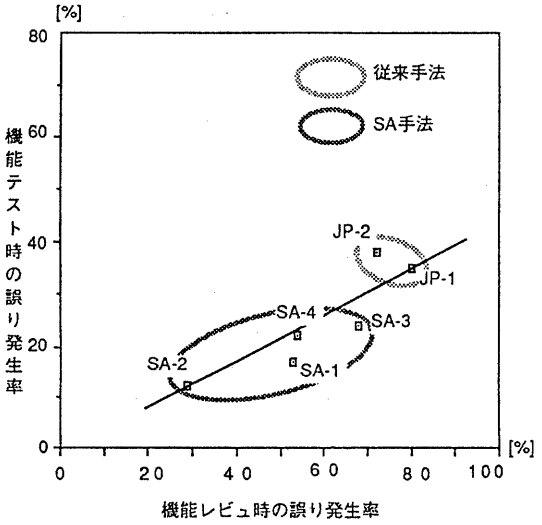

図4　機能レビュ時と機能テスト時の誤り発生率の関係

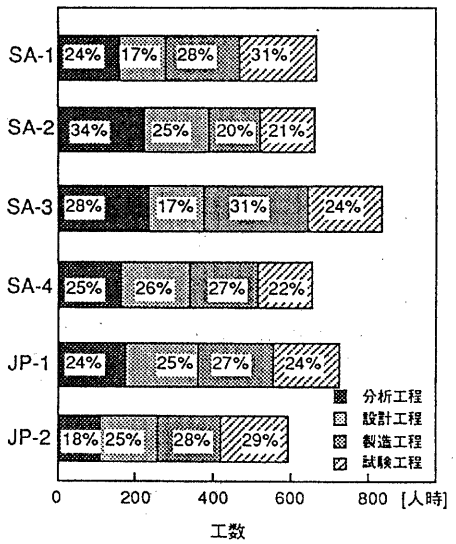

図5　開発工数分布

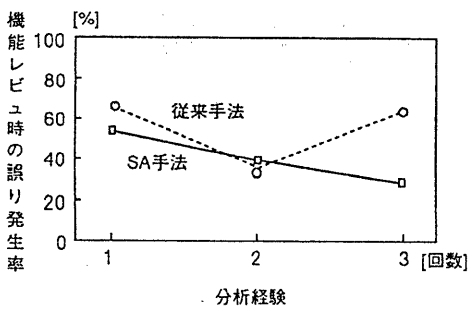

図6　機能レビュ時の平均誤り発生率

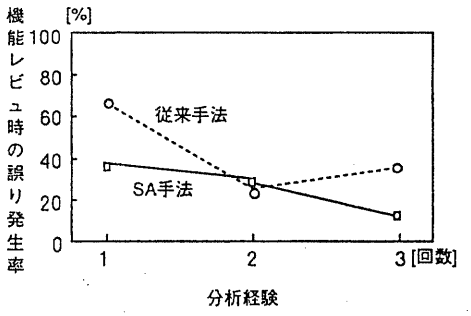

図7　機能に関する平均誤り発生率

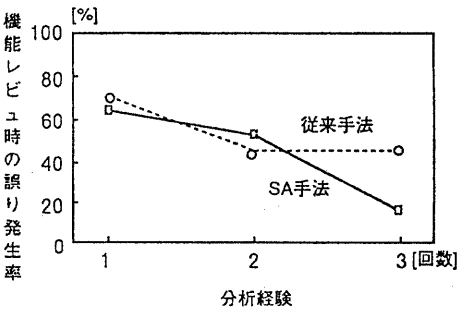

図8　データに関する平均誤り発生率

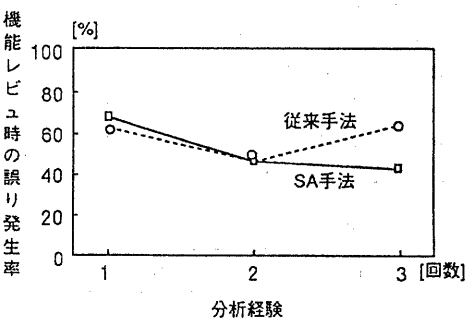

図9　制約条件に関する平均誤り発生率

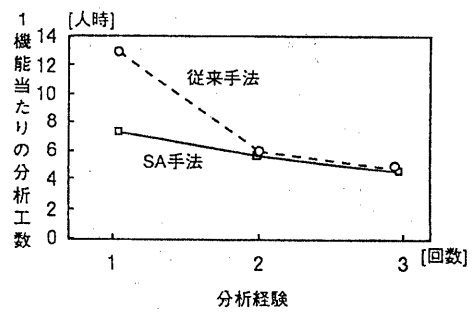

図10　1機能当たりの平均分析工数