1989年9月5月

第7回　夏のプログラミング・ワークショップ

ポジション・ペーパ

高田義広

大阪大学基礎工学部鳥居研究室

E-mail:takada@tori2.ics.osaka-u.junet

我々の研究室では，ソフトウェア信頼性シンポジウムにおける討論に基づいて提案された枠組みに基づいて，要求品質の実現を支援するシステムの開発を行っています．枠組みでは，(1)要求を明確にする，(2)要求を実現するために適切な開発プランを立てる，(3)開発プランを十分に実行する，(4)実現された製品の品質を確認する，ことが必要と考えています．我々のシステムでは，これらの機能を実現するためにデータベースを利用します．データベースの内容は，図１の３種類の品質表にまとめられます．

現在，我々の抱えている問題は，品質表の内容が充実していない点にあります．（特に，要求品質の実現のための活動に関して情報が不足しています．）品質表の内容が充実していないのでシステムの適用事例が得られない，逆に，適用事例が得られないので品質表の内容を改善できないと言うジレンマの状態にあります．

そこで，今回のワークショップでは，現場で要求品質の実現のために行われている活動にどのようなものがあるかを教えて頂ければと考えています．そのために，次の３つのアプローチを考えています．

１）大学で考えられる限りの品質表（管理者視点の品質表T2）を提供し，その批評，修正案を頂く．

今回用意した品質表を図２に示します．品質表中のActivityは次の文献の中から選択したものです．

[1]"IEEE Standard Glossary of Software Engineering Terminology", IEEE, 342 E.47th St., New York, Rep.IEEE-Std-729-1983(1983).

[2]情報処理進行事業協会技術センター 編："品質機能展開による高品質ソフトウェアの開発手法「活用事例編」", コンピュータ・エージ社 (1989).

[3]宮本勲："ソフトウェアエンジニアリング：現状と展望", TBS出版会 (1982).

選択に当たっては，なるべく特定のソフトウェアに依存しないもの，広く行われていると思われるものを選びました．

2）現場で作成されている報告書，記録等の中間生成物に注目して，行われている活動を教えて頂く．

ソフトウェア開発中にある活動が行われた場合，それを報告したり記録するために中間生成物が作成されるはずであると考えられます．それらの中間生成物の種類に注目して，それに対応する活動を洗い出せば，行われている活動を客観的に知ることができると考えられます．この洗い出しを十分に行うためにはできるだけ細かく中間生成物の種類を教えて頂くことが重要です．

3）現場で使用されているソフトウェアツールに注目して，行われている活動を教えて頂く．

ソフトウェアツールは開発中のある活動を支援するためのものです．それらのソフトウェアツールに注目して，それに対応する活動を洗い出せば，行われている活動をやはり客観的に知ることができると考えられます．この時，1つのツールでいくつかの活動を支援している場合の配慮が重要になります．

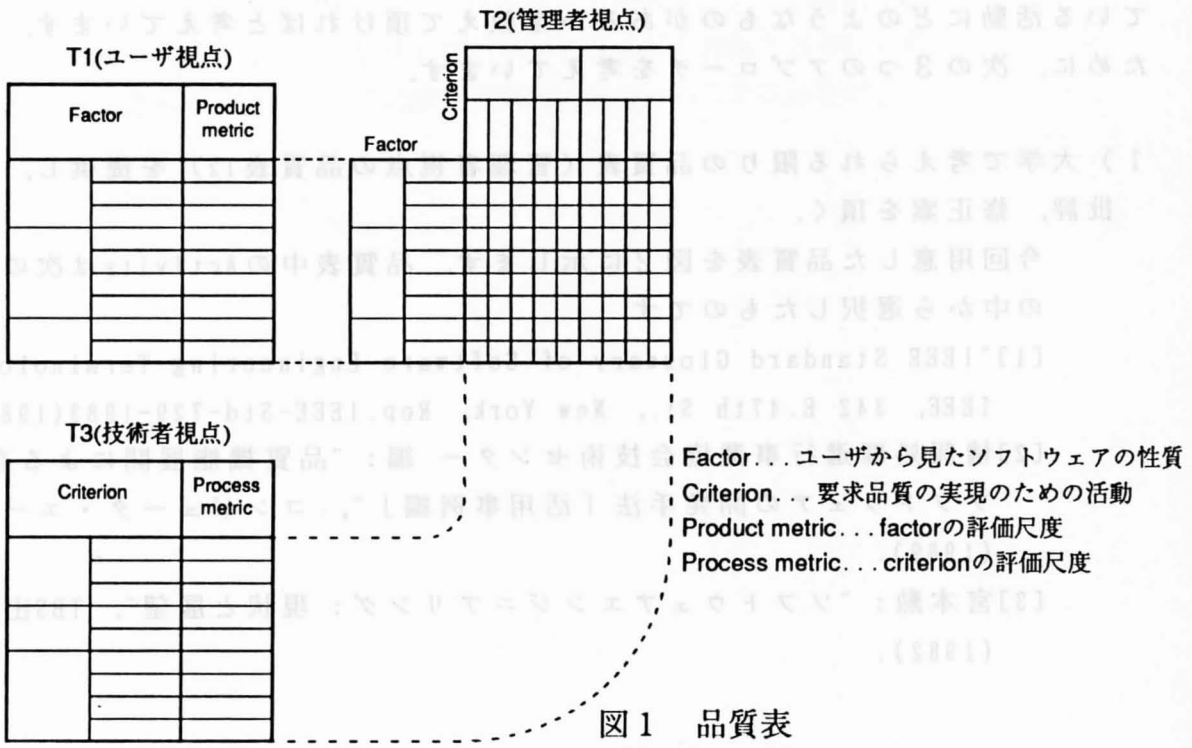


図1　品質表

		requirements analysis		design																
										modular decomposition functinal decomposition			structured programming							walk-through design inspection		
		システムの運用計画の明確化	システムの拡張計画の明確化	アルゴリズムの明確化	入出力データ項目の明確化	他のソフトウェアとのインターフェースの検討	ハードウェアとのインタフェースの検討	障害回復時の復旧方式の決定	セキュリティ要件の検討	論理フローの明確化	データフローの明確化	モジュールのインターフェースの明確化	論理フローの明確化	入出力データ形式の明確化	コマンド形式の明確化	画面出力形式の明確化	稼働時に必要な資源の見積り	入出力量の見積り	実行速度の見積り	全ての機能が実現されているかの確認	論理エラーのチェック	あいまいな表現のチェック
機能性	合目的性				3									3					2	7	3	3
	正確性			5	5									7							5	3
	セキュリティ								7													
	互換性				5	5																
	接続性					9					2											
信頼性	無欠陥性			3	3									3							7	5
	誤り許容性							9														
	可用性																				5	
使用性	理解性													3	5	5						
	習得容易性														5	5						
	操作性														5	5			2			
	対話性														5	5			2			
効率性	実行効率性			5															8			
	資源効率性										5						5	5				
保守性	修正容易性		5							3	3	5	3									5
	拡張性		9									5										5
	テスト容易性			3	5					3	3	7	5									3
移植性	ハードウェア独立性						9															
	ソフトウェア独立性					9																
	導入容易性	8																				
	再利用性					7	7				5	8										

		coding							
					walk-through code inspection				
		規約に基づくコーディング	program protection	inductive assertion method	設計に従っているかの確認	規約に従っているかの確認	論理誤りの解析	プログラム構造の解析	timing analysis
機能性	合目的性				2		2		
	正確性				2				
	セキュリティ		3						
	互換性								
	接続性								
信頼性	無欠陥性				5		7	5	
	誤り許容性								
	可用性				3		7	5	
使用性	理解性								
	習得容易性								
	操作性								
	対話性								
効率性	実行効率性							2	8
	資源効率性								
保守性	修正容易性	3			2	2			
	拡張性	3							
	テスト容易性	5							
移植性	ハードウェア独立性								
	ソフトウェア独立性								
	導入容易性								
	再利用性	7				3			

		testing											
		test design		unit testing			interface testing		system testing				
		仕様に基づくテストデータの選択	内部構造に基づくテストデータの選択	機能の実現の確認	fault seeding	symbolic execution	機能の実現の確認	program mutation	機能の実現の確認	セキュリティ要件を達成しているかの確認	障害対策の確認	operational testing	error analysis
機能性	合目的性	2							7			9	
	正確性	3	3						7			5	
	セキュリティ									6			
	互換性											5	
	接続性												
信頼性	無欠陥性	8	8	7			7	5	8			9	5
	誤り許容性										5		
	可用性		5	5					5		3	9	5
使用性	理解性											2	
	習得容易性											2	
	操作性											5	
	対話性											5	
効率性	実行効率性								3			1	
	資源効率性								3			1	
保守性	修正容易性												
	拡張性												
	テスト容易性												
移植性	ハードウェア独立性												
	ソフトウェア独立性												
	導入容易性											5	
	再利用性												

図2 管理者視点の品質表T2

component testing. Testing conducted to verify the implementation of the design for one software element (for example, unit, module) or a collection of software elements.

design review. (1) A formal meeting at which the preliminary or **detailed design** of a **system** is presented to the user, customer, or other interested parties for comment and approval.

(2) The formal review of an existing or proposed **design** for the purpose of detection and remedy of design deficiencies that could affect fitness-for-use and environmental aspects of the product, process or service, and/or for identification of potential improvements of **performance,** safety and economic aspects. (ANSI/ASQC A3-1978)

error analysis. (1) The process of investigating an observed **software fault** with the purpose of tracing the fault to its source.

(2) The process of investigating an observed **software fault** to identify such information as the cause of the fault, the phase of the development process during which the fault was introduced, methods by which the fault could have been prevented or detected earlier, and the method by which the fault was detected.

(3) The process of investigating **software errors, failures,** and **faults** to determine quantitative rates and trends.

fault seeding. The process of intentionally adding a known number of **faults** to those already in a **computer program** for the purpose of estimating the number of **indigenous faults** in the **program.** Synonymous with **bug seeding.**

functional decomposition. A method of designing a **system** by breaking it down into its **components** in such a way that the components correspond directly to system **functions** and subfunctions. See also **hierarchical decomposition.**

functional design. The **specification** of the working relationships among the parts of a **data processing system.** (ISO) See also **preliminary design.**

inductive assertion method. A **proof of correctness** technique in which **assertions** are written describing **program** inputs, outputs, and intermediate conditions, a set of theorems is developed relating satisfaction of the **input assertions** to satisfaction of the **output assertions,** and the theorems are proved to be true.

inspection. (1) A formal evaluation technique in which **software requirements, design,** or **code** are examined in detail by a person or group other than the author to detect **faults,** violations of development standards, and other problems. Contrast with **walk-through.** See also **code audit.**

(2) A phase of quality control that by means of examination, observation or measurement determines the conformance of materials, supplies, components, parts, appurtenances, **systems, processes** or structures to predetermined **quality requirements.** (ANSI N45.2.10-1973)

integration testing. An orderly progression of **testing** in which **software** elements, **hardware** elements, or both are combined and tested until the entire **system** has been integrated. See also **system testing.**

interface testing. Testing conducted to ensure that **program** or **system components** pass information or control correctly to one another.

modular decomposition. A method of designing a **system** by breaking it down into **modules.** See also **hierarchical decomposition.**

program mutation. (1) A **program** version purposely altered from the intended version to evaluate the ability of program **test cases** to detect the alteration. Synonymous with program mutant.

(2) The process of creating program mutations in order to evaluate the adequacy of **program test data.**

program protection. The application of internal or external controls to preclude any unauthorized access or **modification** to a **computer program.**

requirements analysis. (1) The process of studying user needs to arrive at a definition of **system** or **software requirements.**

(2) The **verification** of **system** or **software requirements.**

structured programming. (1) A well-defined **software** development technique that incorporates **top-down design** and **implementation** and strict use of **structured program** control constructs.

(2) Loosely, any technique for organizing and coding **programs** that reduces **complexity,** improves clarity, and facilitates **debugging** and **modification.**

symbolic execution. A verification technique in which **program execution** is simulated using symbols rather than actual values for input **data,** and program outputs are expressed as logical or mathematical expressions involving these symbols.

system testing. The process of **testing** an integrated **hardware** and **software system** to verify that the system meets its specified **requirements.** See also **acceptance testing, qualification testing.**

test design. Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests. (See ANSI / IEEE Std 829-1983 [4].)

timing analyzer. A software tool that estimates or measures the **execution time** of a **computer program** or portions of a computer program either by summing the execution times of the **instructions** in each path, or by inserting probes at specific points in the **program** and measuring the execution time between probes.

walk-through. A review process in which a designer or programmer leads one or more other members of the development team through a **segment** of **design** or **code** that he or she has written, while the other members ask questions and make comments about technique, style, possible **errors,** violation of development standards, and other problems. Contrast with **inspection.**

第７回夏のプログラミング・ワークショップの感想

高田義広
大阪大学　基礎工学部

今回のワークショップでは，企業の（若手の）方々の問題意識に接することができ，大変有意義だったと思います．特に，実際ソフトウェア開発に携わっておられる方々の一部の方々の思考には，いくつかのパターンが存在すると感じました．自分なりにまとめてみると，１つは次のようなパターンとなりました．

(1)納期が守れない．バグが取りきれない．
↓
(2)数字を利用して工程管理，品質管理，等を行いたい．
↓
(3)どんなデータを収集するべきか，収集したデータをどのように利用するべか分からない．
↓
(4)実験的にあるデータを収集してみたい．
↓
(5)データ収集に労力がかかりすぎる．
（上司の承諾が得られない場合(1)に戻る．）
↓
(6)データ収集，分析を機械化したい．
（方法がみつからない場合(1)に戻る．）

もう１つ次のようなパターンもあったと思います．

(1)納期が守れない．バグが取りきれない．
↓
(2)数字を利用して工程管理，品質管理を行いたい．
↓
(3)プロジェクトによって要求，工程，等があまりに違うので，比較できるデータが少ない．
↓
(4)工程等を標準化したい．
（どうしてよいか分からない場合(1)に戻る．）

思考のループから抜け出して，議論を前進させるためには，(1)データ収集の機械化，(2)工程の標準化，等を考える必要があると思います．

Position Statement

前澤　伸一（２８歳　Ｏ型）
日本システム（株）
第２システム技術部
〒190 立川市曙町１－１８－２　一清ビル６Ｆ立川事業所
0425-26-5911 内線(612)

経験：
電力会社　給電所システム（事故時自動復旧）
　　基幹系統故障復旧支援システム　など

ポジション・ステートメント：

ソフトウェアの品質を把握するには，主に３つの問題が上げられる．

１．ソフトウェア品質の定義

現状では，ソフトウェアの品質を客観的に定義せず，仕様を満足するだけで良しとしている．従って，でき上がったソフトウェアの品質がシステムの中でどのレベルにあるのか，他のソフトウェアに比べ充分な品質を持っていあるかなどの評価を正しく行なうことができない．

２．ソフトウェア品質の評価基準（単位）

仮に，ソフトウェアの品質を客観的に定義したとすると，次に品質を計るための物差しが必要となる．そこで，大事なのは物差しの単位で，客観的かつ不変でなければ意味がない．

３．ソフトウェアに求められる品質

定義や評価基準が決まることでソフトウェア品質の把握は可能となる．しかし，ソフトウェアの品質を管理するには，ソフトウェアの仕様を同じ基準で評価し，その結果から管理しないと出来上がったソフトウェアの品質がオーバースペックで赤字となることも考えられる．

これらの問題を改善するには，まず品質の定義を明確にし，客観的な評価基準を作る．次に，ソフトウェアはハードウェアと違い直接目で確認しながら物差しで計ることができないので，評価者が主観的に評価することで，客観的な数字を求められるような変換表を作る．以上の改善により，ソフトウェアの品質を定量的に把握・管理することが可能と思われる．

ポジション・ステートメント：

1989/09/21

日本システム（株）
第２システム技術部
前澤　伸一

ソフトウェア品質の定量的把握を目指して

現状：

現在も含め今までに，ソフトウェアの品質を定量的に評価・把握することは行っていない。ユーザーの要求（納期も含む）を満足し，システムに問題（バグ）がないと判断することで“ＯＫ”としている。しかし，実際にはシステムの出荷後にも問題点が発生し苦労することがある訳で，そんなときのためにもソフトウェアの品質がきちんと把握できていれば良いなと考えている。

問題点：

簡単に『ソフトウェア品質の定量的把握を目指して』と言っているが，先ず最初に“ソフトウェア品質”とは何かと言う疑問がわく。ユーザーにとっての品質は，機能や性能と考えることができる。しかし，我々のような開発者や管理者にとっての品質を考えた場合，ステップ数や開発時間，保守性とか標準化などのキーワードを思い浮かべることはできるが品質を定義することはなかなかできない。

仮にソフトウェア品質を定義できたとした場合，品質を定量化するための評価基準が必要となる。一定の評価基準を用いて品質を評価したとき，初めて意味のある品質評価ができる。

今後：

もしもソフトウェア品質の定量化ができたなら，次ぎのようなことに利用したいと考えている。

１．見積もりの簡素化

要求されるソフトウェアの品質を明確にすることで，ソフトウェアの規模や工数が簡単に逆算できる。

２．工程管理の質的把握

今までステップ数やバグ件数などしか把握できなかったものが，ソフトウェアがどの程度の完成度か適切に把握できる。

３．ソフトウェアの品質保証

品質が定量化できれば，家電製品のような保証書を付けてソフトウェアを出荷できるようになる。

第７回夏のプログラミング・ワークショップを終えて

1989/09/22

日本システム（株）前澤伸一

今まで品質について考えること無く，いったいどうなるのかと思いながら，不勉強のまま今回のワークショップに参加することとなりました。

基調講演ではいろいろなことを教えてもらいましたが，まだまだ品質を定量化することができずにいることを知り，自分にも討論の中に入り込む余地をみつけホッとするやら，収穫が余り無いと残念に思うやら，チョット複雑な気持ちでした。グループ討論では様々な立場で，ソフトウェアを作っている人達が集まり，それぞれの現場における苦労話がメインとなってしまい，まさに傷の舐めあいでしたが，最終的にはみんな何かしらの結論（途中結果とは思うが）を得たと思います。私の場合は，品質を定量化する方法はまだ研究段階でもあり，どのようになるかはわかりませんが，品質が定量化できると仮定したなら，私達ソフトウェア技術者にとって非常に有用な武器となると改めて認識しました。

Position Statement

後藤　浩（２４歳　Ｏ型）
（株）シスプラン
技術部
〒107 港区赤坂２－１４－１１　赤坂斎藤ビル４Ｆ
03-224-1620

これまでの主な開発経験：

私は，転職の経験１回あります．いわゆる業界の中での転職ですが，最初の会社ではマイコン関係の仕事が主でした．余り覚えてませんが，横浜の市営地下鉄の運行管理システムの１部を担当したはづです．（横浜の地下鉄には恐ろしくて乗れません．

現在は Unix ワークステーション，あるいは MS-DOS パソコン上のアプリケーションを開発しています．Unix ワークステーション上では，データベースシステムの保守や拡張の仕事，各種フィルタの開発などを行っています．MS-DOS パソコンでは，製品として，ターミナルエミュレータを創りました．

ポジション・ステートメント：

司会：　本日は，マイコン関係のソフト開発に従事されている後藤硬さんとワークステーション上のアプリを開発されている後藤軟さんにおいでいただき（上記これまでの主な開発経験参照），ソフトウェア品質の定量的な把握や管理というテーマに関してお話をお聞きします．それではまず初めに硬さん，お願いします．

硬：　定量的ってなんですか？

軟：　定量分析とか定性分析とか聞きますね．

司会：　いっ，いや，ですから，ソフトウェア品質の定量的な把握と管理....

軟：　私は，ほとんど一人で仕事してますから，私を把握している人間は私ですし，管理している人間も私です．

司会：　わっ，分かりました．では，仕事上のリスクに関してどの様な観点をお持ちですか，軟さんどうぞ．

軟：　RISC ですか，まだ RISC マシンには触れたことがありません．

硬：　軟さん，ふざけないでください．私の場合は，ハードのチームと一緒に仕事を進めます．私がメンバに加わる段階では，CPU は決定されてますが，ROM や RAM の容量に関してハードのチームとよくやりあいます．機能を実現するために充分必要な大きさを要求するわけです．ROM/RAM の容量と，システムの要求するスピードなどを充分考慮し基本設計を行います．あと，結構問題になるのは，デバクの環境です．ターゲットが大きいものだと社内ではシステムテストが出来ない場合がありますから，リスクの話とは関係ありませんが，経験の無い TTL をつかっている仕事は避けたいですね．

軟：　なるほど，リスクですね．私は，ターゲットマシンも開発マシンも WS か PC ですから，ターゲットマシンが開発マシンと異なることによるリスクはありません．しいていえば，ユーザの要求をはっきりさせる点に注意を払っています．画面のイメージや印刷イメージのことです．出来上がってユーザに見てもらって「こんなはづじゃなかった」ということだけは避けたいと思っています．それと，WS や PC 上の部品が蓄積されていますから，それら利用できるものと新しく創る必要のあるもののきりわけを考えたりしますね．部品の再利用を含め，なるべくプログラムを書かないようにしています．

司会：　プログラムを書かないというのは？

軟：　プログラムを書かなければバグはでないのです．私の経験では，100 ステップあたり 3 個位の割合

でケアレスミスによるバグが発生します．C 言語ですよ．できるだけ書かないようにするために部品の充実に時間をかけます．ソース・モジュール単位の独立性を高めるためにそのプロジェクトとは直接関係のない部分も書き直したりしています．

司会：　有難うございます．そろそろ，ソフトウェア品質の定量的な....　平たく言えば，軟さんの「100 ステップあたりにバグが 3 つある」とか，とにかくなんでもいいから数字で現してしまおうという事....（ほんとなのかしら：本人）

硬：　なるほど，私の会社ではドキュメントを標準化することにより，開発効率やメンテナンス性の向上に努めています．コーディングスタイルやコメントの書き方までプロジェクト単位で標準化します．バグ死滅曲線を作って，品質に関する判断基準にします．なかなか，掴みにくい個人の問題点などは毎朝のミーティングで各人が報告しあいます．

軟：　標準化ですか，んーっ．私の会社のドキュメントに関する唯一の標準は LBP 出力Ａ４タテ版です．コーディングスタイルなど気にしません．どうしても読みづらい場合は cb するか，捨てます．バグ死滅曲線は面白いかもしれませんね．こんど色分けして壁に貼ろう！定量化という意味では，先にもお話しましたが，蓄積された部品がかなり役に立っています．

司会：　えーっと，テーマに関して適切な意見が....　最後に，ソフトウェアの品質に関してご意見を．

硬：　ソフトウェアというよりもシステム全体の品質ということが重要です．極端な話，システムが要求通りに動作すれば品質は関係無いんじゃーないかなー．あっ，動かなきゃ駄目ですよ．

軟：　そのシステムの性格によりますね．使い易さという点では，ユーザと動きを確認しながら行なうことによりほとんどの部分をカバーできます．スピードに関しては，profile を通して確認します．そうですね，品質を第一に考えるのは我々の使命のような気がします．「納期よりも品質をとりますか」と言われると困りますが....

司会：　ありがとうございます．話はつきませんが，私ちょっと用事がありますのでこのへんでお開きにしたいと思います．

軟：　デートですか？

司会：　だといいんですけどね．どうも長い間ありがとうございました．これからのお二人のご活躍を期待します．

「ソフトウェア品質の定量的把握を目指して」
ワークショップを終えて

株式会社　シスプラン　後藤 浩

ポジションペーパでうけを狙ったのは私一人だったようです。
ここまできたら、最後まであの調子で行きます。

司会：　えーっ、今回はワークショップを終えて盛岡からお帰りになって間もない
お二人（昼：後藤　夜：？？）に成果を報告して頂きます。
始めに、全体的な流れからお話して頂けますか。

夜：　最初の夜はパーティの後に、グループのみんなでドブロクを飲みました。
ホテルに帰っても寝つけなかったのでバーで一杯やって寝ました。
３泊あるのでボトルをいれようかと思いました。
次の日は、ワンコソバを食べました。流れ解散で気が付いたら知らない人たちと
アイスクリームを食べて・・・

昼：　ちょっと待って下さい。ここに「プログラム」があります。

司会：　ふむふむ。グループ討論では何を？

夜：　小岩井農場でジンギスカンを食べました。

昼：　いい加減にしてください。
私たちのグループは、「ソフトウェア品質の定量的把握」に関して
既にソフトウェアの品質管理部門を設けて取り組んでおられるある会社の
事例をもとに討論を行いました。
その事例を極論すれば、「バグを記録すること。蓄積されたデータから
プロジェクトのバグの総数を試算し、その数だけバグを潰すことを
目標にすること。」という感じに解釈しました。
実際にその会社では、大きな成果があったそうです。
事例に関して、各自の立場で意見を交換しあいました。

司会：　で、成果は？

夜：　有形無形それぞれの大きな成果があります。
私は、無形の成果が大だったのでここで口にすることはできません。

昼：　私の場合は、バグを記録することに対して積極的になれるような気がします。
次のバグを出さない為に、今のバグを記録するのです。

夜：　かっこいぃー！

昼：　馬鹿にしないで下さい。

司会：　今日も、時間が無いですね。Ａ４　１枚分の時間しかないです。
感想を一言どうぞ。

夜：　盛岡から戻る新幹線の中から見えた景色が印象的です。
屋根が赤い家ばかりだったのを覚えています。

昼：　多くの方と様々なことに対して話が出来たことが良かったと思います。

司会：　時間が来ました。お疲れさまでした。
皆さんこれからの予定は？　一杯やってきますか。

夜：　賛成！！　赤坂の「ままや」に行きましょう。

Ｐｏｓｉｔｉｏｎ　Ｓｔａｔｅｍｅｎｔ（第２版）

安喜　裕師（２７歳　Ａ型）
（株）富士通徳島システムエンジニアリング
システム部第一システム課
〒770 徳島県徳島市寺島本町西１－７－１
0886-25-5555(309)

経験：地方公共団体（市役所）の住民記録オンラインシステムの開発
（住民票，印鑑証明，戸籍等）

ポジション・ステートメント

現在まで４年余り、同一ユーザのシステム開発を行ってきたが、その間の経験から、ソフトウエアの品質管理に関して、次の様な問題があると考える。

１．開発者側の問題

（１）各システムに開発担当者が１名しかいないため、充分なレビューが行われていない。

（２）開発担当者は翌年も別のシステムを開発するため、前年のシステムについてのメンテナンスが、充分に行えない。

２．ユーザ側の問題

（１）人事異動によるシステムの理解者の不足

（２）プロジェクト完了後の、ユーザ担当者の不在

これらの問題に対して、開発に直接従事した人が不在であっても、何とかシステムのメンテナンスが可能となるように、いろいろと考えてきた。

勿論、メンテナンスを行う上で重要なことは、ドキュメントがどこまで整備できているかということになる。しかし、いかにドキュメントが充分あっても、いきなり開発担当者以外の人間がメンテナンスできるわけもなく、結果的に前年の開発担当者が、メンテナンスを行うことになる。

こうなった原因として、次の様なことが考えられると思う。

１．メンテナンスが必要になるということは、それだけ品質か悪いということであり、開発時に何らかの問題があった。

２．開発担当者以外の人間にシステムを知っている人が全くいない。

上記のような問題を解決するために、私の担当するユーザでは、システムの維持管理に関しては、別会社にすべて任せるという方法を取った。これにより、ユーザ側のシステム理解者がいなくても、別会社の手によりシステムはメンテナンスされ続けるというわけである。

しかし、上記１の問題については、開発時の問題であり、今後品質の向上や、品質の把握を考えていかなければならない。

夏のプログラミングワークショップ感想文　　安喜　裕師

今回、夏のプログラミングワークショップに参加して、「ソフトウエア品質の定量的把握」について議論できたことは、私が今後仕事をしていく上で、非常に有益であったと考えています。現在の会社に入社してから今まで、私のソフトウエア開発に従事してきましたが、作業人員の不足、作業時間の不足から、かなりいい加減な開発作業をしてきたようにおもいます。以下に、今回のワークショップを終了してからの感想を述べたいと思います。

1．レビューについて

現在の会社に入社してから4年余りの間、私にとってのソフトウエア開発とは、プランニング工程から始まって、テスティング工程まで、すべて一人の担当者が独自に行うというものでした。というのは、人員の不足ということもありますが、一人で作業を行った方が無駄なレビューをせずにすむし、どうせ誰も手伝ってくれたり助けてくれたりすることはないのだから、レビューを行っても時間が無駄になるだけで、得るものは殆どないと思っていたからでした。

しかし、他の会社の人たちの意見を聞いたり、その後自分なりに考えてみたりした結果やはりレビューとは重要なものであるということが、わかってきました。確かに、今までレビューの重要性はわかっていたのですが、レビューにかける手間を省くために、殆ど何も行っていなかったのですが、思わぬ勘違いが発見されたり、他の人の意見が聞けたり、やはり、レビューはソフトウエアの開発には欠かすことのできないものであると、現在では思っています。

現在も、ソフトウエア開発を行っているわけですが、今後なるべく多くのレビューを行っていきたいと現在は思っています。

2．テスティングについて

テスト方法については、グループ内でもかなり意見がわかれた問題でした。

私の作業は、ユーザとの共同作業の形態を取っているため、テスト作業とは、ユーザが中心になって行うものであると思っていましたし、ユーザに対してもそのように働きかけていました。勿論、それがベストであるとは思いませんし、もっと他のテスト方法が沢山

あることは知っていますが、作業形態上この方法が一番開発者にとっては楽な方法でした

現在結論はでていませんが、テスト方法についていろいろと考えているところです。

いかに多くのテストを行ってバグを発見し、いかに開発者にとって楽なテストを行うかこの二つは全く矛盾したことですが、実際に開発作業をしていくうえで、何かしらの回答が発見できるのではないかと考えています。

3．開発作業全般について

当たり前のことですが、ソフトウエア開発とは多くの人がチームを作って行うものですが、私にとっては一人で全て行うものでした。工数面の制約で、多くの人員を投入できないこともありますが、そのような場合でも、できるだけ多くの人の意見を聞くことが重要であるということを再認識しました。

意見を聞くのは、自分の仕事とは全く関係のない人であっても、何かしら自分が考えていなかったような良い意見がでてくることも考えられます。これからは、多くの人と会話を行いながら、作業を行っていきたいと考えています。

4．ソフトウエア品質の定量的把握について

今回のワークショップの議題はこれでした。しかし、グループ討論では参加者のソフトウエアの開発方法についての議論が中心となってしまい、いかにして品質を上げていくかを議論していったと考えています。

品質の定量的把握は、ソフトウエア産業に従事するものとして、必ず考えていかなければならない問題ですが、それもまず、品質の向上を考えての上での話しであると思います

今回は、具体的な品質の定量的把握の話しをするところまで行きませんでしたが、今回議論した、レビュー方法、テスティング方法などから品質の向上を図っていくことにより品質を把握できるようになるのではないか、と考えています。

私にとっては、今後もソフトウエア開発作業は続いていきますし、今回のワークショップだけに終わらせずに、今後もソフトウエアの品質について、いろいろと考えていきたいと思っています。

ソフトウェア品質の定量的把握について
～ある試みとその問題点～

（株）ソフトウェアコントロール
山崎哲彰

ソフトウェア品質の定量的把握について、私自身が現在担当しているプロジェクトを材料に述べてみたい。

1．その背景

仮にＯプロジェクトと名付けよう。これは、銀行オンラインシステム用のＯＰ（オフィスプロセッサ）の開発を目的としたプロジェクトである。銀行サイドで現行システムを見直し、総入れ換えを行うことになった。ホスト側、営業店側、端末側、その中間に立つＯＰ、すべて開発しなおしである。その中でも、ＯＰのアプリケーション（ＡＰ）と、アプリケーションプログラムを支援する共通プログラム（共通サブルーチンと通信制御）を、筆者のチーム（というより、筆者の勤務する会社）が担当することとなった。

＜チーム担当分＞

開発規模:　約１００Ｋｓｔｅｐ
使用言語:　ＣＯＢＯＬ（ＡＰ）、及び、ｃ（共通）
開発期間:　１９８８　８／ｂ～１９８９　７／ｅ

現在の工程は組合せテストである

プロジェクトのスタートに当り、チーム内で工程管理と品質管理の方法を明確にする必要があった。開発規模も投入工数も筆者の所属する課（課ではなくグループと呼ばれているが）で経験のないｖｏｌｕｍｅであり、従来行ってきたような、経験と勘に頼る管理ではとても成功はおぼつかないと思われたからである。

それまでのやり方というのは、リーダ、またはマネージャが、製作担当者の報告から進捗を判断するという、非常に曖昧なものであった。製作された物件の品質についても担当者まかせであり、製作物の品質を保証する客観的な基準がなかったわけである。その結果、なんだか分からないうちに工程が終了し、次工程でバグが出る、という悪い（おなじみの）パタンを繰り返していたのだ。これではやっていられない、というので、製作するプログラムの品質を定量的に把握し、そのデータで管理しようという計画が立てられた。以下は、その成果と分析である。

☆☆☆☆☆☆☆☆★★★★★★★★★☆☆☆☆☆☆☆★★★★★★★★

２つの問題があった。

①ソフトウェアの品質を何で評価するのか。
②評価基準をどうやって定量データに置き換えるか。

報告に入る前に、ちょっとこの問題について考えたい。

①ソフトウェアの品質を何で評価するか。

・仕様の満足度
・実行効率、メモリ効率
・使用容易性
・保守性
・移植性

ソフトウェアの品質を評価する基準として、以上のようなものが考えられるだろう。しかし、このすべてについてデータを採り、分析を加えるには、あまりにも手不足であり、また、経験が不足していた。われわれのチームで採用したのは、”バグがないことを最低限の品質目標とする”という考え方である。少々プログラムの作りが不器用でも、問題なく動作すればとりあえずよしとしよう、なんせ、これまでは、そのレベルさえクリアできていなかったのだから．．．　これが、チームの誰にも、また、技術部の同僚にも共通した感慨であったと思う。

②評価基準をどうやって定量データに置き換えるか。

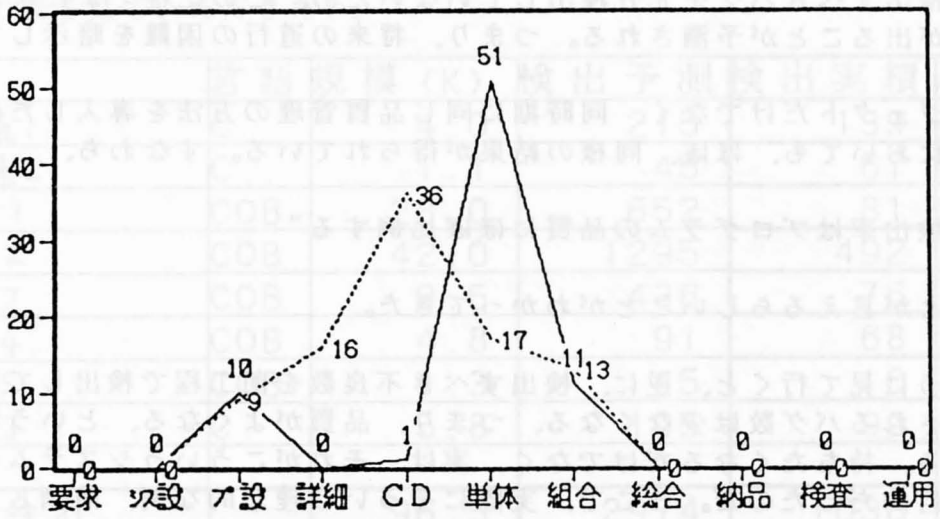
プログラムにバグがないことを論理的に証明することはできないといわれる。では、バグがないことをどうやって定量的に示せばよいのか。
バグが残っていないことを統計的に保証しよう、というのがわれわれが採った方法である。具体的には、

・開発規模をもとに、混入されるバグ数を予測する。
・開発過程で検出されるバグ数から、検出率と収束率を算出する。

というステップを踏むことにした。予測したバグ数に見合う数のバグが検出されていることで、プログラムの安定度を評価しようとしたのである。予測バグ数は開発スタート時の開発予測規模から割り出し、各工程に入る前に調整することとした。そして、検出数の把握は、製作担当者、レビューア、テスト担当者が記録するバグ票（不良記録）によって行うことを決めた。

☆☆☆☆☆☆☆☆★★★★★★★★★☆☆☆☆☆☆☆★★★★★★★★

2．その結果と分析



〈グラフ1〉

グラフ１は、あるアプリケーションの工程ごとの不良検出予測と検出実績を示したものである。

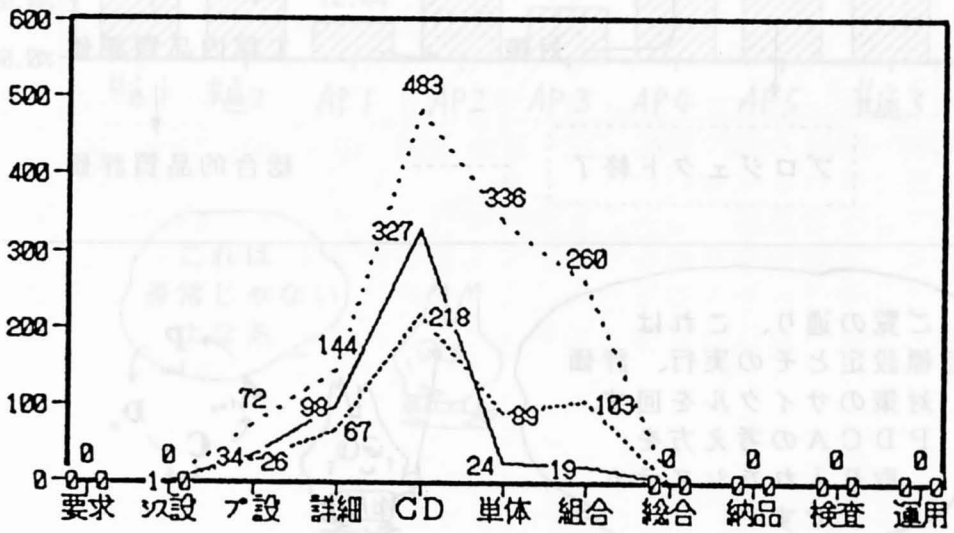
一見して分かることは、詳細設計、ＣＤ（コーディング）工程の検出率が悪い。その分、後工程、つまり単体試験で多くのバグを検出している。ここから言えることは、

①詳細設計、ＣＤ工程で検出すべき数の不良を検出していない。
②その結果、次工程に不良を持ち込んでいる。

ということである。つまり、ＣＤ工程が終了した時点でこのプログラムを評価すると、

”不良検出率が低い。すなわち、品質が悪い（バグを残している）”

という判定ができる。



〈グラフ2〉

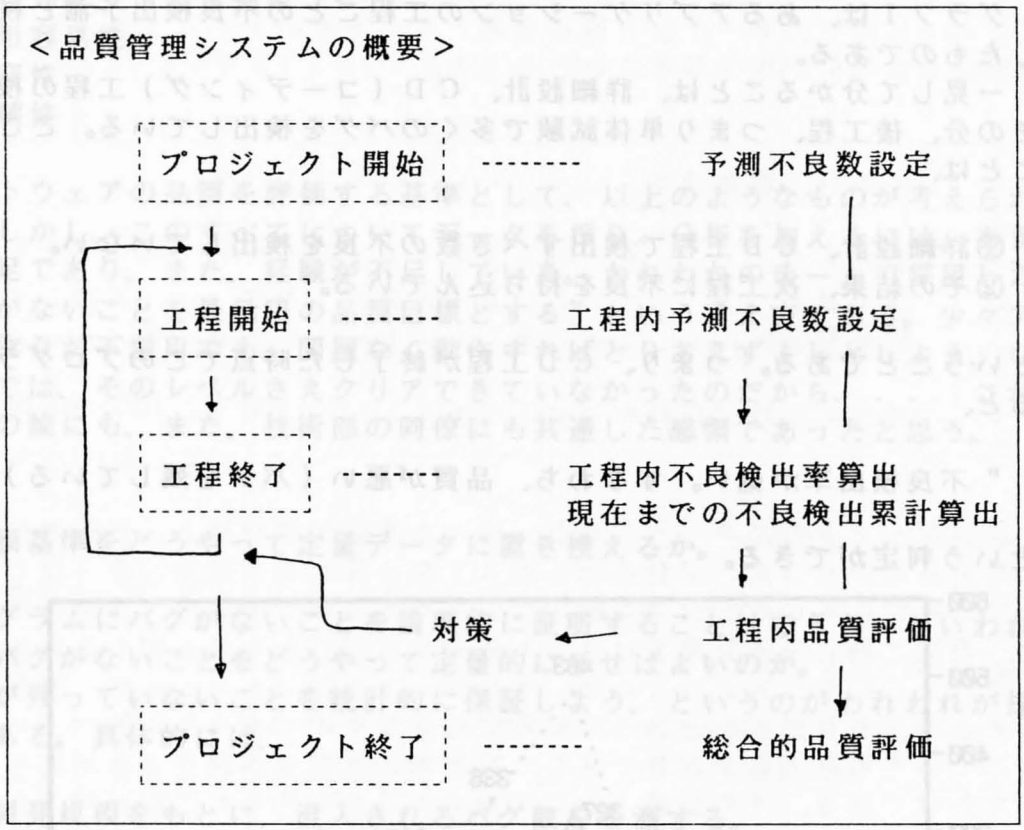
グラフ２は別のアプリケーションについて、同じデータを示したものである。単体試験工程での不良検出率が非常に低い。現在は組合せ試験工程であるが、単体試験で検出すべきバグを充分検出していないため、この工程でまだまだたくさんのバグが出ることが予測される。つまり、将来の道行の困難を暗示しているわけである。

本プロジェクトだけでなく、同時期に同じ品質管理の方法を導入した他のプロジェクトにおいても、ほぼ、同様の結果が得られている。すなわち、

”不良検出率はプログラムの品質にほぼ比例する”

ということが言えるらしいことがわかってきた。

このように見て行くと、逆に、検出すべき不良数を前工程で検出していると後工程に残されるバグ数は少なくなる、つまり、品質がよくなる、という希望を持ちたくなる。持ちたくなるだけでなく、実は、それがこういうシステムを採用した本来の目的だったのだ。ここで、実際にそういう理想的な例、前倒しでどんどんバグを検出して、プログラムが安定したという実績をお見せしたいところであるが、残念ながら手元に見つからない。世の中、うまくいかないものである。



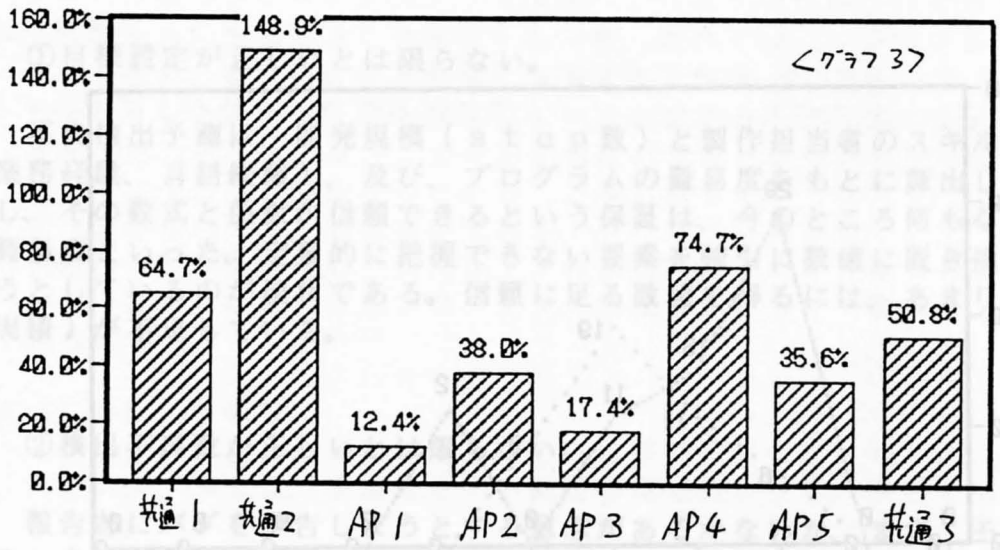
3．その問題点

では、このシステムでソフトウェア品質の定量的把握ができると言ってよいのだろうか。

下の表を見ていただきたい。

	言語	規模（K）	検出予測	検出実績	検出率
共通1	C	4.0	215	139	64.7%
共通2	C	1.1	45	67	148.9%
AP 1	COB	28.0	652	81	12.4%
AP 2	COB	42.0	1295	492	38.0%
AP 3	COB	9.5	436	76	17.4%
AP 4	COB	4.8	91	68	74.7%
AP 5	COB	1.3	45	16	35.6%
共通3	C	5.8	195	99	50.8%
合計		96.5	2974	1038	34.9%

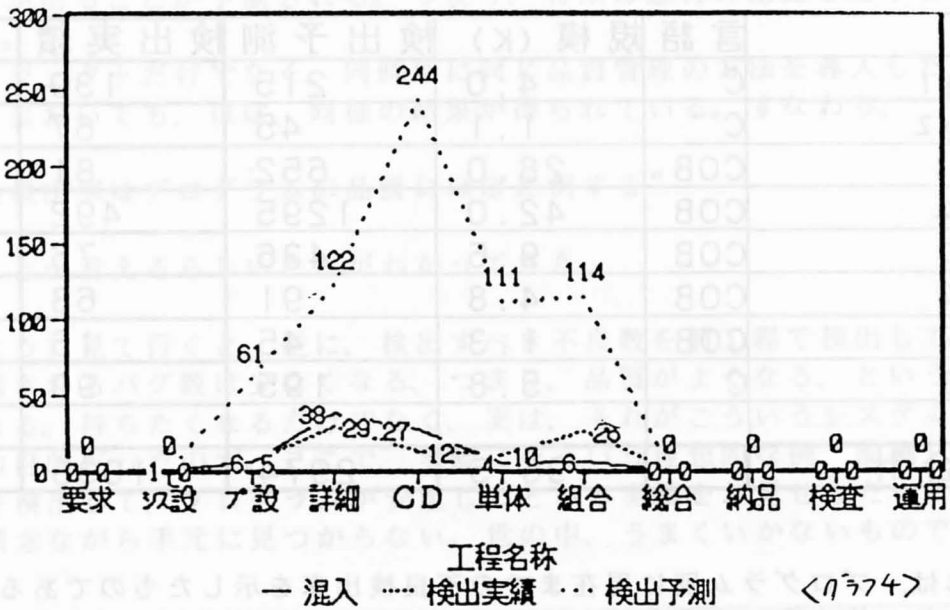
これは、プログラム別に現在までの不良検出率を示したものである。プログラムにより、検出率に大きなばらつきがある。１０％台のものから、１５０％近いものまで、そのばらつき方は、誤差範囲といって見逃すには、あまりに大きすぎる。参考に、プログラム別検出率をグラフ化したものを示す。



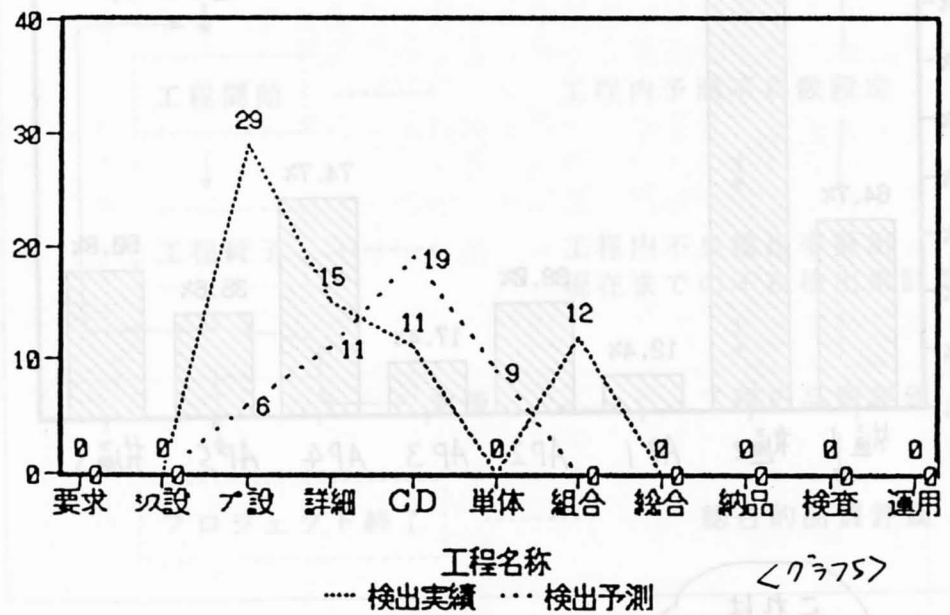
<グラフ3>

もっと具体的に、ＡＰ１と共通プログラム２の検出予実カーブを見てみよう。

グラフ４は、検出率１２％のＡＰ１の検出予実カーブ、グラフ５は、検出率１４８％の共通プログラムの検出予実カーブである。



<グラフ4>



<グラフ5>

ＡＰ１は開発開始時から一貫して検出率が悪い。しかし、不良が検出されてないわけではなく、ｃｏｎｓｔａｎｔに検出されているのである。現在は組合せ試験工程であるが、この後、検出率が急に上昇するとは考えられない。この場合、バグが検出されていないのではなく、最初の検出予測が誤っていた、もしくは、検出されたバグが報告されていない、または、バグとして計数されていない、と考えられる。

一方の共通プログラムを見てみよう。すぐ気付くのは、プログラム設計、詳細設計段階で、予測値を上回るバグを検出していることである。ＣＤ工程はやや検出率が落ちるが、前倒しでバグを検出したと考えれば納得がいく。そして、単体試験での検出数は０である。すでに予想検出数をクリアしているし、プログラムソースのバグをマシンテストまでに出し尽くした、なんと、理想のプログラム開発の姿ではないか！　と思ったいたら（ま、誰もそんなこと思ってはいないが）、組合せ試験で予想を上回るバグが出てしまった。たとえ、検出率が１００％に達しようと、それがプログラムの安定度を保証することにはならないという例である。

以上の事から、次の問題点が導かれる。

①目標設定が正しいとは限らない。
②検出不良件数が正しいとは限らない。

これらをまとめて、精度の問題ということもできるだろう。

①目標設定が正しいとは限らない。

不良検出予測は、開発規模（ｓｔｅｐ数）と製作担当者のスキル、経験年数（業務経験、言語経験）、及び、プログラムの難易度をもとに算出している。しかし、その数式と係数が信頼できるという保証は、今のところ何もない。スキルや難易度といった、定量的に把握できない要素を強引に数値に置き換えて計算しようとしているのが現状である。信頼に足る数値を得るには、あまりにもデータ（実績）が不足している。

②検出不良数が正しいとは限らない。

報告者にバグを申告しようとする意志があるかないか、まず、それが問題になる。さらに、何をもってバグとするかという基準が個人によってまちまちであれば、報告精度はばらばらになってしまう。どの段階からバグと見なすか、同様のバグが一度に検出されたとき１件と計数するのか、複数件として申告するのか。

実は、今回の方法でも不良を計数することの趣旨を説明し、かなり厳密に基準の設定を行ったのである。しかし、投入人数が多くなってくると、それを徹底するだけでも（チェックまで考えると）専任の要員が必要になる。開発チームだけでは、手が回らなくなるのだ。

4．その対策と課題

①予測精度を上げる。
②個人による計数のばらつきをなくす。

上のような対策が必要であろう。では、具体的にどうすればよいかであるが、

①については、長期間にわたってデータを収集し、予測値と実際との差を少なくしていく。
②については、フィードバックと啓蒙を繰り返して、バグの計数基準と計数を徹底する。

という具合いに気長に対応していくしかない。頼りない対策であるが、では、すぐに効果のある抜本的な対策があるかというと、人間がやっている限り、根本的な対策はありませんという以外ない。

むしろ、本当の問題は、このような、個人の申告に依存する方法でしかソフトウェアの品質を把握できないというところにあるのだろう。人間が作り出す不良を人間が検出する、つまり、評価の過程に人間が介在してしまうということそのものが情報の劣化を招いてしまう。工業がそうであったように、人間の介在する部分をできるだけ少なくしていくことが、品質の評価のためにも、その均一化のためにも有効であるはずだと思う。現在のソフトウェアでは、品質の定量的な判定は期待できないというのであれば、始めからそれをめざした言語とＯＳとマシンを開発するしかない。いずれは、今のような手作りのソフトウェア開発は時代遅れになる、というのが、筆者の希望であり予測であるのだが、それまでは、そのいいかげんさにうんざりしながら、「人間的品質管理」を積み上げていくしかあるまい。

１９８９．７．２０

その後の仁義なき戦い

ソフトウェアコントロール
山崎哲彰

1．精度を上げる

プロジェクト計画時に立てた予測値をもとに不良検出率を算出すると、'プログラムによって著しいバラつきがあることは、ポジションペーパに書いた通りである。精度を上げるためには、気長にデータを収集するしかない、と前回は述べたのだが、中間評価をもとに予測を立て直す、つまり、軌道修正をするという手があることを書かなかったので、追記したい。

計画時に立てた予測値が、なぜ精度が悪くなるか。これには以下の理由が考えられる。

①開発規模が推定値である。

予測値は、開発開始時に推定した開発ステップ数を基準にして算出している。しかし、実際に開発されたプログラムが何Kステップになるかは、プログラムを書いてみなければ判らない。また、書かれたプログラムも、テストが進むにつれ、障害対応や仕様変更等で更新されていく。

本来は、ステップ数が更新される度に検出予測も立て直すべきなのだが、工程の遅れに追われて、実行できなかった、というのがこれまでの状況である。

②開発担当者が交代する。

担当者の経験年数、スキル等を考慮して、予測値を算出するための係数を設定しているのだが、プロジェクトが進むにつれ、担当者の交代というのが起きる場合がある。また、スキルを係数化するといっても、そのスキルを判定する基準自体が曖昧なのだから、どうしたって、”えいや”で決めてしまうことが多いのだ。蓋を空けてみて、”こいつ案外よくできた”とか（あんまりない）、”期待していたら、大外れだった”とか（結構ある）いうことがあるため、この係数もプロジェクトの進行に合わせて更新していく必要があるだろう。

③いいかげんな係数。

係数自体の根拠が希薄である。これは前回書いたこととダブるので繰り返さないが、本来は、予測に対する実績の出方によって、係数を調整する必要がある。もっとも、うっかりすると現実に予測値の方を会わせていくことにもなりかねないので、注意が必要である。

以上３点を上げたが、この中で、軌道修正の対象となるのは、①と②である。③については、軌道修正しても、もともと根拠の希薄な係数を、さらに根拠希薄にしていくだけなので、長期的にデータを収集して、調整していくしかないと思う。

実際に設定し直した予測値と、それに対して算出し直した検出率を次に示す。

機能	予測不良件数	検出実績	検出率（%）	前データに基づく検出率（%）
AP1	140	89	63.6	13.7
AP2	840	588	70.0	45.4
AP3	84	84	100.0	19.3
AP4	96	79	82.3	86.8
AP5	28	16	57.1	35.6

こうして算出した検出率は、実際にテストを続けていて感じるプログラムの安定度から判断しても、なかなか納得のいく値である。

忙しいとか、めんどくさいとか言わずにデータの見直しをしていけば、結構精度を上げることができるのが分かるだろう。こうした見直しを工程の区切り毎に行っていけば、工程が進むにつれ、精度を上げていくことが可能ではないかと思う。

2．仕様変更という奴

プロジェクトの進行に従って、動的に予測値を更新していく話をしたが、ここで、さらに予測値に影響を及ぼす要素を登場させたい。それは、仕様変更という奴である。

エンドユーザは、実際に動いたシステムを見るまで、最終的な仕様を決められないと言う。これは、どんなに仕様書上で仕様確認しても、所詮は紙の上のことにすぎないという事情による。そこでプロトタイピングといった手法も生まれたのだが、わがOプロジェクトでは、プロトタイピングが採用できず（技術的理由から）、昔ながらの、労働集約型開発を進めざるをえなかった。

結果、どうなったか。

デモンストレーションを兼ね、エンドユーザのもとで総合試験を行ったところ、なんと2ヶ月で40件近い仕様変更が発生してしまったのだ。工数にして700H近い量である。

仕様変更は工程を遡っての手直しになるため、当然、品質管理の基準となるプログラム規模、難易度に影響する。こういう場合、どうやって予測値を立て直せばいいのか、現在のところ明確でない。設計時に遡るという気もするが、不良と同じ扱いにして、修正によるデグレード不良の予測を追加すべきという気もする。

この辺も、将来的には算出式を作って管理すべき点であろう。

3．現実の話

現実の話、変動要素が多すぎる。プログラムに対する不良数の予測は、ある程度まで精度を上げることができるだろうが、それをプロジェクトの進行に沿って、動的に更新し、品質管理に役立てるためには、強力な品質管理ツールといったものがないととても無理だろう。どうも話が品質の定量的把握の話題からずれて、ただのぐちになりつつあるので焦っているのだが、不良検出率による品質把握の問題点として、２点を上げて、もうこんな不毛な文章は終わりにしよう（苦しい逃げ方だな）。

①プロジェクトがある程度進まないと、予測値の精度を上げることができない。

②プロジェクト進行に従って、変動する要素が多すぎる。

１９８９．９．４

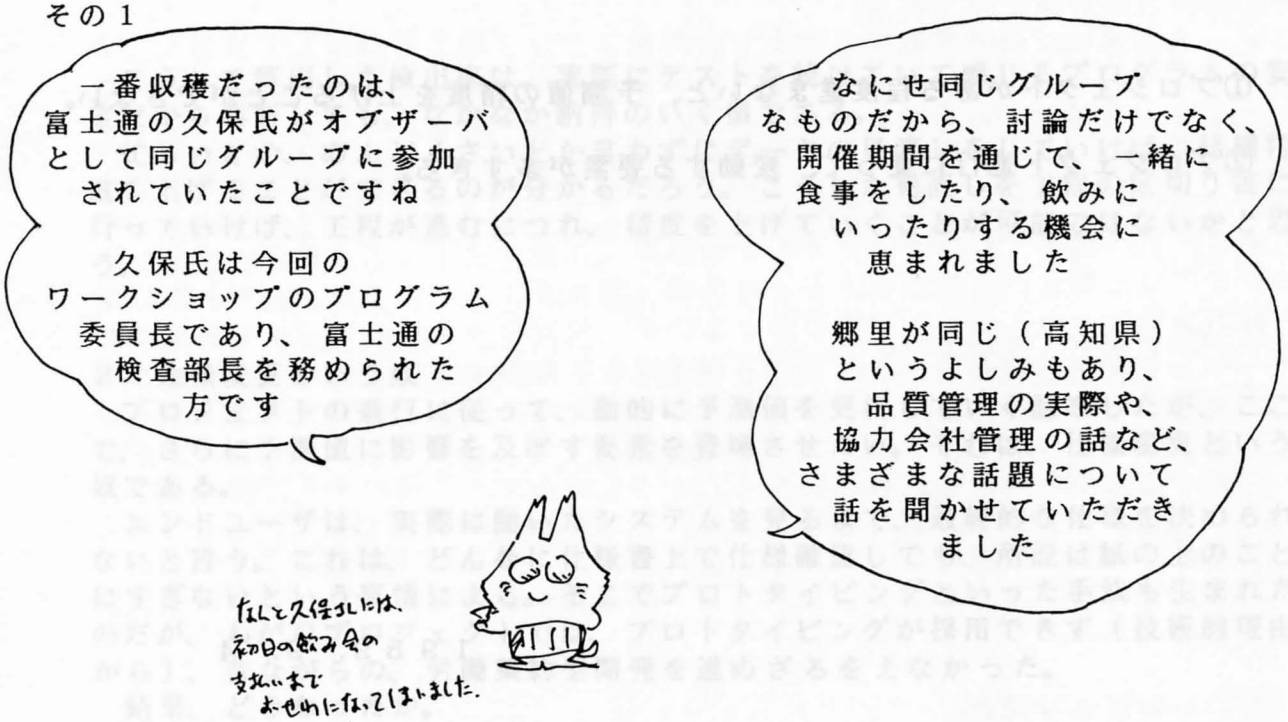
'８９　夏のプログラミングワークショップに参加して
～３つの収穫～

ソフトウェアコントロール
山崎哲彰

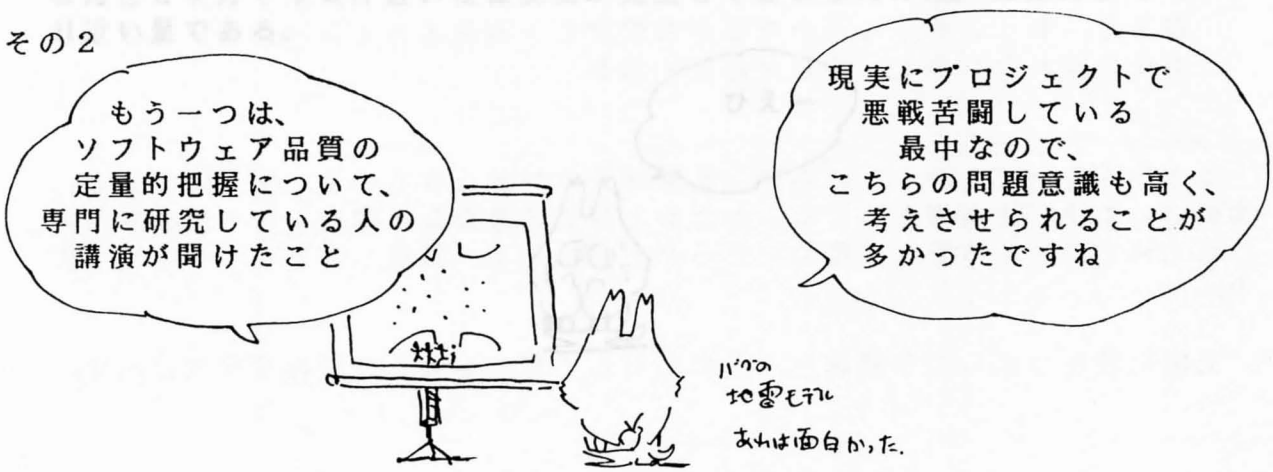
ポジションペーパは、誤字脱字を訂正して、そのまま載せます。ワークショップには、ほとんど満足していますが、実際にデータを取ってみて、その上で提出された問題についてのディスカッションがあまりできなかったのが、やや不満です。１０月のソフトウェア信頼性シンポジウムでは、その辺の突っ込んだ議論が聞けるのではないかと楽しみにしているのですが...

以下、収穫だった点を書いておきます。

その１



その２



その３

収穫の３つめとして、
やはり、他社の人と交流
できたことを上げて
おきましょう

会社、業務形態が
異なることで、
こんなことが違うのか、
とか、

こんなところは変わらない、
といった発見がありました

ぐちの自慢しあいになった部分も
ありましたけど

この世界、おしなしところで仕事してると、
すぐ、井の中の蛙になって
しまいますから。

Position Statement

山口　敬介（２４歳　A型）
（株）ケーシーエス
ソフトウェア技術部
〒650 神戸市中央区浪花町６４　三宮電々ビル６Ｆ
078-391-7733 内線(2555)

経験：
C 言語によるアプリケーション開発
Lisp 言語による設計支援ツールの開発

ポジション・ステートメント：

１．ソフトウェア品質の定量的把握について

ソフトウェアの品質といっても一概に表現することが，困難である．例えば，開発言語によっても異なるであろうし，開発するソフトウェアの種類にもよるものである．

ここでは，業務アプリを開発するうえでの品質の定量的把握について述べることにします．一般に，ソフトウェア品質を計る尺度として基準となるキーワードは，次に挙げる５つに絞られるのではないかと考えています．

- Maitainability（保守性）
- Reliability（信頼性）
- Prtability（可搬性）
- responsibility（処理速度）
- Functionability（機能要求）

(1) Maintainability（保守性）

「ソフトウェアのライフ・サイクルと言うことを考えた場合，その 60～70 %は保守であり，保守の生産性と品質を上げることが今後のソフトウェアの課題である」と書いてある書籍を目にしたことがある．私達は，ものが出来上がったらそのプロジェクトは，ほぼ完成したと考えがちになり，保守を頭に置いて開発することは少ないように思える．

(2) Reliability（信頼性）

バグがないのはもちろんのこと，操作性に優れていることである．操作性にも色々あるが，ユーザに不安感を与えることなく，ユーザを惑わしたりしない操作手順であり，オペレションに一貫性があること．バグの多い少ないは，設計方法に左右されるし，設定するテストケースのパターンが，ソースコードに比較して少なすぎた結果の出来事である．もう一つ言えることは，「Simple is Best」と言う言葉からも伺える通り，スマートな設計にバグはない（少ない）と考えています．

(3) Portability（可搬性）

移植性も含めた意味での Portability であり，稼働マシンにあった実効サイズをしていること．固有のマシンに特価されないソースコードである（移植性）．

(4) Responsibility（処理速度）

昔のようなハードウェア側の処理速度の遅さは，現在では無くなりつつありますが，エンドユーザにとって遅すぎず早すぎないことが必要です．

(5) Functionability（機能要求）

Position Statement

ユーザが望むアプリケーションにする．信頼性が高く，保守も容易，処理速度も問題がなくてもユーザ要件を満たしていないようなソフトウェアには，価値がありません．いかに，ユーザ要件を把握し，適切なアドバイスをし，一定期間内に仕様どおりに仕上げるか，当然ですが，なかなか大変な問題です．

以上の５つが私の考えるソフトウェアの品質を語るためのキーワードであり，品質を評価する際の条件と考えます．

２．ソフトウェア品質の現状と問題点

上記内容のうち最近，切実に思うことは，数計の甘さが及ぼす PG 作業への影響です．PG 作業に入ってから，設計へのフィードバックを繰り返すといった事態に陥ります．

ソフトウェアにとって，ユーザ要件を満たしていることは必須のことであるが，一定期間内に豊富な要件を盛り込み，バグのない，信頼性の高いソフトウェアを完成することは大変である．いままでの，経験から言えることは，全体のスケジュールのうち，仕様理解と数計に時間を割き過ぎて，プログラミングの期間迄食い込むといったケースになることがよくありました．

プログラム設計及びモジュール設計の甘さからソースコードが複雑になり，バグの原因となり，悪循環を繰り返してしまいます．

プログラム設計及びモジュール設計段階でのアプローチとして，フローチャトからプログラミングを行っていましたが，フローチャートだけでは，制御の仕組みは読み取れるが，データの流れを追うことができず，プログラミングレベルでの戸惑いが生じてしまう．このことが，スマートなモジュール間及び，プログラム間のインタフェースを損ねてしまいます．

３．ソフトウェア品質向上の解決法

現在，Lisp 言語による設計支援ツールの開発を手掛けていますが，ここでは，フローチャートで分からないデータの流れについては，データフローダイアグラムを書くことによって制御とデータの両面から設計を行っています．

このように，全体を通してフローチャートとデータフローダイアグラムを最下位レベルまで作成しておくと，以前見えなかった部分まで見えてくるし，設計段階でのバグをも未然に防げ，メンテナンスのしやすいソースを生成できる．

ドキュメントなどの毎回決まった作業には，時間を費やさないようにする（例えば，ドキュメント作成ツールによる作業の軽減）．

PG 作業を軽減するために，モジュール，プログラムあるいは，サブシステムの再利用できるものは，できるだけ再利用する．

そこで，データの流れを把握するために，データフローダイアグラムを作成し，制御系の流れだけではなく，データの流れも掴んだうえで，プログラミングに臨めば完成したソフトウェアの仕様と設計時の仕様の食い違いを最小限に押さえることができ，仕様変更の際にも早急に対応できます．当たり前のように思うことですが，大規模な開発になるほど実際には，なかなか難しいことです．

私自身，ソフトウェアの品質の定量的な把握について，まだ，考えの及ばぬものとして捉えています．

盛岡ワークショップを終えて

株式会社 ケーシーエス
ソフトウェア事業部
ソフトウェア技術部
山口 敬介

ソフトウェア品質向上について

ソフトウェア開発においてソフトウェアの品質を向上させるには、開発工程間あるいは、プロジェクト間におけるスムーズな開発方針を設定する必要がある。実際の作業分担の中では、開発工程間あるいは、プロジェクト間の相互に与える影響は大きいはずである。

ワークショップでは、開発工程間あるいは、プロジェクト間開発作業の中の仕様を決定してプログラム作成作業に移る段階でスムーズに開発を行なうためには何が必要かを考えてみた。

下位工程に与える情報としては、次のものが考えられる。

	プロダクト	プロセス
要求	機能、性能（品質）	納期、コスト
評価	プロダクトメトリック （機能実現率） （レスポンスタイム）	プロセスメトリック （レビュー時間） （残バグ数）
支援	サンプル、ライブラリ	ツール

現状との比較

グループミーティング中には、多くの意見が出たのですが、現実との比較としては、表中にかかれたものすべてを下位工程に渡しているわけではなく、プロジェクトごとに違い簡単なドキュメントで開発を行なっているのが現状です。

第七回夏のプログラミングワークショップ（若手の会）

---- 盛岡ワークショップについての感想 ----

（株）ケーシーエス
ソフトウェア事業部　ソフトウェア技術部
山口　敬介

初めて盛岡に行きました。というより初めて東京より東に行きました。やはり、討論の場としては非常にいいですね。３泊４日の短い間でしたが、とても有意義な４日間でした。

毎晩数人で居酒屋で語らい、昔ながらの古風な作りの居酒屋で時間も忘れて喋ってました。初めて体験したわんこ蕎麦大会とか、盛りあっが情報交換パーティーは、非常に楽しいものでした。この場を借りてプログラム委員の方々にお礼を言いたいと思います。どうもありがとうございました。
成果発表の前日には、小岩井農場や手作り村などに行き、そしてまた、温泉にも入りました。案内してくださった岩手電子計算センターの三浦さんと吉田さんどうも有難うございました。

このように、遊んだことばかり書いてしまうと、『おまえは、何しに行った』と言われそうなので、ワークショップの内容にも触れたいと思います。

私の現在の仕事として、ｕｎｉｘ上で開発支援ツールの開発に携わっているのですが、プログラムの構造とか、モジュール分けがどうだとか、といった範囲のことしか日頃考えていない状態で、『ソフトウェア品質の定量的把握を目指して』などと言う事について考えたことがありませんでした。このような姿勢でワークショップに望みましたので、他社の方々の意見から非常に得るものがありました。

わたしのセクションでは、まだ行なっていないのですが、開発工程においてメトリックスによる品質を定量的に測っておられる企業もありました。
また、ＩＢＭの大場さんには、地雷の話など具体的でわかりやすく話していただき、大阪大学の松本さんには品質表を管理するシステム（ＳＱＵＡＲＥ）の話など品質の定量的把握に関する話題が満載されていました。

品質の定量的把握と言う事をソフトウェア全体で考えようとした場合、やはり一長一短であり、統一的にまとまらないように私自身思っています。品質の定量的把握の目的は、ソフトウェア品質の向上であると受け取っていますので、現段階では、『品質を向上させる』と言う事を考えていきたいと思います。

以上

Position Statement V2.0

湊　幸雄 (24歳　A型)

(株) ケーシーエス　(旧社名:神戸コンピューターサービス)
ソフトウェア事業部　第二ソフトウェアシステム部
〒650 神戸市中央区浪花長64　三宮電々ビル3F
☎ 078-391-8355 内線(2456)

経験:
公共自治団体財務会計システム (COBOL)
ドキュメント管理システム (COBOL)
マイコン用プログラム自動解析システム (C)

ポジション･ステートメント(rewrite版):

始め『ソフトウェア品質の特性が使用環境･目的などによって多岐に渡っており、画一性に欠けているので、定量化は難しいだろう』と考えていたが、他の人の話を聞いていると、本当に大きな相違点が有り、定量化の困難さを改めて感じた。

しかし、『困難さを嘆いていてもソフトウェア品質は改善されない!!改善するために、何でもいいから出来ることを積極的に実行する』ことの重要さを感じることが出来た。

実行出来ることは幾つか有るが、基本的に品質に対する『意識改革』が一番重要であることが討論の中でクローズアップされたので、それについて幾つかを述べる。

ソフトウェアの品質が悪い、どうしよう	→	バグを無くそう
バグを無くす為にどうするか	→	バグ表を作成して管理してみよう
正確なバグ表を作成する為にどうするか	→	バグを漏れなく報告させる
バグを報告し易くする為にどうするか	→	バグに対する従来の価値観(意識)を改革する
意識改革を全社的に進める為にどうするか	→	バグ表神話をでっちあげる

ワークショップで討論していた中で実際にデータを取っている所が有り、何らかの形でデータをフィードバックすると品質の定量化(品質の向上)を実践できると言う事例が有り、非常に影響された感じで有る。

しかし、実際にデータを取ることが良いと行っても現実はプログラムの仕様目的が多様でフィードバックの意味が無くなる場合、また設計段階ではバグをバグと認識していない(システムを実際に使用してみないと気が付かない)ことも有り前途多難である。

よって開発作業標準書を早急に作成する必要を以前より感じた。
設計段階のバグに対しては開発作業標準書を作って、チェック項目を明確にすることにより、レビューアの能力に頼らなくてもある程度の品質の開発が出来る制度を作れば、その後の保守･管理も「先ず理解する」という工数が省け、作業し易くなると共に定量化に向けて前進出来ると考える。

感想

日頃、システムを開発していると開発方法などに疑問を持つことが多いが、業務の忙しさに押し流され、考えてみる時間が無かったので、今回のワークショップでゆっくり討論できたことは有意義でした。

しかし、会社に帰ると再び業務の忙しさに押し流され、成果をフィードバックする前に忘れてしまいそうで、現実の厳しさに涙を流しています。それから私は蕎麦は嫌いでは無かった筈ですが、あれから一度も蕎麦を食べていません。

雨が多くて残念でしたが、このワークショップを運営して下さったSEA事務局の方、並びに地元の世話(特に夜)をして下さった岩手電子計算センターの方々に感謝します。

Position Statement

伊佐　厚司（２３歳　ＡＢ型）
（株）ケーシーエス
第２ソフトウェアシステム部
〒650 神戸市中央区浪花町６４　三宮電々ビル６Ｆ

ポジション・ステートメント：　ソフトウェア品質の定量的把握を目指して

ソフトウェア品質の定量的把握について考えるにあたって，まず，品質を低下させている要因について考えてみました．ソフトウェアの品質を低下させてる要因は設計段階にもありますが，ここでは，開発段階に的を絞って，考えていきます．

(1) 概要書・仕様書の標準化がされていない
(2) テスト項目の洗い出しの漏れとレビュー不足
(3) ひな型，部品化の再利用が徹底されていない

それぞれの原因についての説明と対策案

(1)　単に，概要書・仕様書といっても，プロジェクト毎に揃える資料のバラツキがある．仕様書を例にとると，あるプロジェクトでは，処理の流れを表現する手段として，モジュール構造図を用いる場合もあれば，YAC-II の様に，１ステップ対応で表現する場合もある．どちらが良いかという問題は別として，この２種類の仕様書を同一の基準で品質の評価を行ったとしても，全く意味のない事である．したがって，品質の定量化を行なうにあたって，開発に必要なドキュメント類の標準化を推進し，ドキュメント品質の評価も行っていかなければならない．

(2)　抽出・更新・編集といったプログラムのテストケースには，ある一定のパターンがある．例えば，更新プログラムなら，「更新される項目に誤りはないか」とか，編集プログラムなら「編集項目に漏れはないか」といった様な内容である．こういった一定のパターンを処理毎に作成し，テスト仕様書の一部とする．テストは，この一覧をもとに行えば，基本パターンの漏れもなくなり，その一覧のパターンＮＯ毎に報告書をまとめれば，報告書の統一も可能となる．

又，こういった項目の洗い出しも，仕様担当者やプログラマで行なうのではなく，レビューを兼ねて，第三者による項目漏れのチェックをする必要がある．品質を維持していくには，こういう第三者の人を品質管理グループというような形で専属化する必要がある．

(3)　ひな型を用いてソース作成を行えば，処理の流れが一定になり，ソースに対する標準化が実現できる．しかし，よく問題となる事に，ひな型に固執してしまうと逆にロジックが複雑になってしまうので，ひな型に柔軟性を持たせるか，必要に応じて，作成していかなければならない．

部品に関していえば，現在のプロジェクトでは，ひな型同様，部品組立てによるソース作成もよく実施されている．この部品組立ての要になるのは，やはり部品一つ一つの品質のよさである．しかし，部品一つ一つを１から作成してしまうと，品質の保証ができない．

その為，１部品が１プロジェクトのみに使用されるのではなく，全プロジェクトを通じて，繰り返し利用されなければならない．それによって，高品質の部品が蓄積されていく．（これは，部品に限らず，ひな型にも同様の事がいえる．）そして，蓄積された部品をプログラムの４０％をしめれば，後の６０％の品質向上に力を注げばよいことになる．

ここで，部品作成に関して少し見方をかえると，既に，本番で動いているソースから，ある一部の処理を部品として抽出できれば，部品の品質は望めるのではないでしょうか．

以上，開発段階でのソフトウェアの品質低下の原因と対策案を述べてきました．私の定量化へのアプローチとして，まず，ドキュメントやソースコードの標準化と再利用を実現するを考えています．これを実現すれば，定量化を行なう為のきっちりとした「物差し」が出来る為，明確な評価が可能になると思います．

１９８９．９．２１

伊佐　厚司　（２４歳　ＡＢ型）
（株）ケーシーエス
第２ソフトウェアシステム部
〒６５０　神戸市中央区浪花町６４　三宮電電ビル６Ｆ

ポジション・ステートメント：ソフトウェア品質の定量的把握を目指して

私がこのワークショップに参加する前は、「ソフトウェア品質の定量化を実現するには、ドキュメントやソースコードの標準化と再利用を実践する事によって、品質を測定する物差しができる」と述べていました。しかし、討論を行なう中で、この物差しは、決して１っではないという事がわかりました。というのは、システムを作る側、使う側と言うように立場によって、品質の基準がさまざまだからです。その品質の基準として討論中に出てきた項目は、次の通りです。

１）仕様書に明記してある機能をどの程度満たしているか
２）操作は、簡素・簡単であるか
３）品質を向上させるコストは十分にあるか
４）納期は厳守できるか

作る側からみれば、１）さえ満たせば良いと考えがちですが、使う側からみれば、１）は当然のことであり、２）～４）も満足してこそ、高品質といえるのです。この事は、分かっていても作る側としては、納期に追われての開発の為、忘れがちになっていましたが、グループ・メンバーの実体験をあからさまに聞き、考えさせられました。いくら私達開発者が工数をかけて、機能を１００％満たしたシステムを作成したとしても、ユーザーにクレームをつけられれば、低品質のソフトウェアという烙印を押されてしまいます。そういった事から、品質の定量化の実現は、非常に困難なテーマである事を改めて確認した次第です。

次に、少し今回のテーマに逆らった討論内容として、「本当に定量化は必要なのか」という事をグループで討論しました。私は、品質の定量化は必要だと考えていました。その理由として、現在、プログラムの完成は最終的にユーザーからの障害通知や変更依頼が無くなった時点と暗黙の内になっています。もし、品質を確実に保証する手段があれば、ユーザーに左右される事なく、開発者側でプログラムの完成時点を決定出来るようになると思うからです。しかし、現実では、品質を測定した結果、プログラマの質を見極める為だけに使われるような意識を持ちました。これが決していけないという訳ではありません。ただ、開発者の精神面にフィードバックするのではなく、実作業の軽減につなが

＜ワークショップの感想＞

この度は、プログラム委員の方々には、いろいろと御世話になり、ありがとうございました。品質の定量的把握とは、具体的にどういった事なのだろうと考えながら、盛岡までやってきた次第です。しかし、参加させて頂いたおかげで、定量化への試みや定量化の問題点が明確になりました。

私の部所では、品質の定量的把握を試みていません。しかし、近い時期に何等彼の試みは必要となる為、今後も品質の定量化についてのセミナーやワークショップに参加して、実作業に役立てていきたいと思います。

盛岡には、こういった形でしかくる事がありませんので、来年、上司の許可が下りれば、参加させて頂きたく思いますので、その時はまた、よろしくお願いいたします。

１９８９．９．２１

（株）ケーシーエス
第２ソフトウェアシステム部
伊佐　厚司

Position Statement

根本　慶一（２６歳　Ｏ型）
（株）ＳＲＡ
特別開発第１部　チーフ・エンジニア
〒102 千代田区平河町１－１－１
03-234-2625 内線(3710)

経験：
・大規模事務アプリケーションの設計と開発
・データベース設計，及びユーティリティー作成などの環境設定

ポジション・ステートメント：　ソフトウェア品質の定量的把握に関して

１．はじめに

自分は主に大規模事務アプリケーションの開発に従事してきており，他のソフトウェア開発の現状についてなあまり把握していないと思っている．従って，ある程度大規模アプリケーション・システム開発にしぼり，この問題についてまず理解し，意見交換の中で検討していきたいと思っている．

そして，品質の定量化について，「どの視点からアプローチするのが最も適当か」を自分なりに検討し，今後の自分の仕事に活していきたいと考えている．

また，理論に基づくトップダウン型アプローチ（“ソフトウェア品質定量化の枠組”等）と，現場サイドからのボトムアップ型アプローチの接点が見つけられれば，と考えている．

２．品質の定量化を検討する上での問題点及び疑問点

（１）　品質を定量化するための背景となっている理論を，いきなりソフトウェア全般にあてはめる議論していくには，現時点では無理が多すぎるのではないかと思われる．また，その理論を実戦してデータ収拾する場合では，ソフトウェアを大きく分類しても，事務アプリケーション，ＣＡＤ／ＣＡＭ，その他制御系等々，それぞれタイプがかなり異なっている．

従って，各プロジェクトのタイプをもっと細分化してそれぞれのタイプ毎にも十分検討を行なわなければ，現実の問題点が明確にならないと思われる．少なくとも，そういう視点から検討することが必要である．

（２）　プロジェクトのタイプを考える上で，重要な要素に，「コスト（特に，金と時間）をどれだけかけられるか？」という問題があり，それによって品質への影響も相当にあるはずであるが，この“コストと品質の相関”についてはどういう考えているのか？

（３）　プロジェクトの性格を検討する上では，「プロジェクトの組織構造」(*)などによっても質的に変化してくると思われるが，そういった要因は反映できるのか？

(*)
ユーザー／管理者／技術者　すべてを含めて考える
開発規模が大きくなると，技術者間内部の組織上のあり方も重要になってくる

（４）　データ収拾／モデルの理論の実証の難しさについて　→　特に，現場の技術者に負担の少ない，効果的なデータ収拾の方法の検討が必要と思う．

（５）　“定量化によって，実際にはどういう効果が得られるのか？”
“今後のソフトウェアについての考え方はどうかわってくるのか？”
といったことを把握したい．

Position Statement

3．現在，携わっているプロジェクトでの品質評価について

規模　－　約 30 万ステップ（COBOL）

＜品質評価の方法＞

基本設計レベル　－　ドキュメントのレビュー

詳細設計レベル　－　ドキュメントのレビュー

（画面／帳票）単位

ＰＧ開発／テスト　－　ＤＹＡＮＡによるテスト・カバレージ測定

コードレビュー（任意選択による一部に限る）

結合／総合テスト　－　事前の用意したテスト計画表によるチェック

ユーザーテストの前に行い，信頼度を類類

BUG 表の管理

（但し，タイプ別の十分な整理はされていない）

＊総合テストでの“Bug　Fixibg”はどうやって判定するか？”

結合／総合テストで，ＢＵＧ発生件数が収束に向かった否か？

サブシステムレベルの担当者の過去の実績とその人個人の技術力，性格より判断

Position Statement

小池 誠（２９歳　A型）
（株）ＳＲＡ
名古屋営業所　チーフ・エンジニア
〒460 名古屋市中区栄２－３－１　名古屋広小路ビルヂング
052-204-5411

経験：
ＮＣ制御
ワープロアプリケーション開発

ポジション・ステートメント：　ソフトウェア品質の定量的把握を目指して

ソフトウェアの品質とは，いったい何なのか？　バグが少ないとか，汎用性があるとか，拡張性に優れているという事なのか？

物の場合，品質がいいと言うことは，機能的に優れていて使いやすくて長持ちするということになると思う．我々は，物を買うときに品質がよくて，安い物を買う．

しかし，我々のお客さんは，品質よりも納期を重視する．即ち，まずは動くものを納期に間に合わせればそれでよしとなる．我々エンジニアも，納期に追われて品質どころではない．本来ならレビューをすべきところで，時間がないと言ってやらない．動けばそれで終り，納期までに出来るのはこの線までであるという意識が，お客さん，我々エンジニアに深く浸透していっまっているのではないか？特に会社との契約での仕事となると，お金がいくらというのが重要となっていて，そのシステムをいかにいいシステムにするか，という点にはあまり気をつかわない．私が入社当時はまだメンバーだったわけだが，お金は問題ではなく，いいシステムを作りたいと思った．しかし，現在はリーダーという立場上，品質よりも納期という考えが先に立ってしまう．

品質をよくする方法は，考えれば何かでてくると思うが，今は品質をよくするという意識を高める事が，重要課題である．

感想文

SRA名古屋営業所　小池　誠

このワークショップに参加する前は、ソフトウェア品質よりは納期という考えが先に立ってプロジェクト管理をしていた。他の会社に比べると品質管理部門もなく、開発時のマニュアル、標準もない（コーディング規約はどっかにあったかもしれない）。ないからではないが、品質管理に対する意識が低かった。しかし、今は”ソフトウェア品質の定量的把握”をやってみようか、という気になっている。まずは、自分のプロジェクト内で標準化をしてデータをとってみる。私が、ポジション・ペーパに書いた”品質に対する意識革命”を、自分から実践してみるつもり。

3泊4日の盛岡でのワークショップは、いろんな人との出会いがあり、有益な話、おもしろい話など聞いたり、特にグループでの活動が多く、好きな場所で討論できるというのはとてもよかった。又、仕事を離れて（地理的にも離れて）同じ年代の人達と同じ様な悩み、問題を話せる場であり、今後も参加したいと思った。

Position Statement

岡田　正之（２７歳　Ｏ型）
（株）ＳＲＡ
システム開発第５部
〒170 豊島区南大塚２－２６－１５　南大塚第一生命ビル
03-942-4455 内線(5506)

ポジション・ステートメント：

ソフトウェア品質の定量的な把握や管理に関してと言われて，まず品質とは何かと考えてしまった．品質とは，何をもって「品質が良い」だとか，「悪い」だとか判断するのだろう．信頼性であろうか，操作性であろうか，はたまた，拡張性であろうか．全てが良いにこしたことがないが，結構品質について，把握するのは難しいようである．

私達が現在ソフトウェアを開発しているなかで，品質の一応の目安にしている事は，ステップ当りのテストの量であったり，プログラムの全てのパスを通りプログラムが正常に動作することを確認したりすることである．

しかし，以前データベースを使用したプログラムの開発を行った際に，テストケースが全て満たされて納品を行ったにも関わらず，多くのユーザがデータベースを同時にアクセスしたり，または，データが多くなってくると，あまりにも処理速度が遅くなってしまい，運用上に問題をきたした事があった．このプログラムは後に，処理速度をあげるためにバージョンアップが行われ現在使用されているが，運用時をあまり考慮にいれずにテスト等を行っていたためにこの様な事が起こってしまった様に思われる．

この様に，品質についての多角的評価がなかなか行われていないのが現状であり，その事はまた，品質を評価する指標が良くわからないためであると思う．今後，良いソフトウェアを開発していくには品質について多くの面から定量的に検討をしていく事が大事なことであると思うが，なかなか一人の力で行なうのは難しいので，ある程度の指標になるような枠組みがあればぜひ身に付けて実践してみたいと思う．

<< ワークショップに参加して >>

(株)ＳＲＡ
岡田 正之

ワークショップには、ソフトウェアの品質の定量的把握の方法を得る、糸口が見つかれば良いと思って参加しました。

実際にワークショップに参加して、大型汎用機のアプリケーションを作成されている人や、組み込み型のプログラムを作っている人、また、検査をしている人など、立場や環境が全然違う所で働いている方のお話をうかがう事ができて、たいへん有意義でした。

また、基調講演やグループ討論を通じて、ソフトウェアの品質を定量的に把握して、高品質のソフトウェアを作る事は大変な事だと感じました。しかし、自分が今、取っている開発データをより有効に利用していけば、もっと高品質のソフトウェアを作る事ができるのではないかと思いました。

Position Statement

三田村　英生（２８歳　Ａ型）
（株）ＳＲＡ
特別開発第２部
03-234-2641

ポジション・ステートメント：

１．はじめに

私は現在，某都市銀行の第３次オンラインのメンテナンスをしておりますが，ソフトウェアの品質管理について，同プロジェクトでの経験及び現状より，その問題点をあげ，今後の改善策を私の考えより述べていきます．

２．ソフトウェアの品質とは？

ソフトウェアの品質管理について述べる前に，私の考えているソフトウェア品質について述べておきます．

私はソフトウェアの品質は次の要因により決定されると考えます．

・定められた規格に従い作られていること（標準化）
・求められた機能を全て持っていること

また，ソフトウェアは他の生産物，製品等と違い，その品質の評価に少しの誤差も許されないと言うことができると考えています．

３．現状における問題点

現在，私のしている作業において発生するトラブル原因として次の様なことがあげられます．

・他のサブシステムの変更が知らされない
・リリースまでの期間が短い
・ドキュメントの不備
・ケアレス　ミス
・開発当時からのバグ
・確認テストにおいて，既存機能を全て消化できない．

また，ユーザーの担当者及びモジュール担当者の業務への理解が足りず，不完全なままでリリースしてしまうケースもありました．（設計担当者の不在）

現在所属しているチームでは品質管理としては，修正の履歴，トラブルの履歴位しかしておらず後は，個人の判断に任されているのが現状です．

４．今後の改善策

早急に品質管理の体制を細分化して導入することが必要と考えています．

・サブシステム担当者間の連絡の徹底
・検証体制の強化
　業務を知らない人でも容易に検証できる．
・ドキュメントの整備
・全機能のチェックシートを作成し，変更毎に確認する．
・リリース後，ある一定期間により再評価する．

Position Statement

山城　英彦（３３歳　ＡＢ型）
カシオ計算機（株）羽村技術センター
情報機器事業本部第１開発部
〒190-11 西多摩郡羽村町栄町３－２－１
0425-79-7111 内線(7565)

経験：

１．ミニコンピュータ　オンライン　システム
２．マイクロコンピュータ　制御ソフト

ポジション・ステートメント：

１．　従来から，統計的品質管理と称される手法による品質評価，管理が行われて来た．この手法自体は，検査側から得られる評価尺度として十分価値があると思われる．しかし，この評価尺度を持って，開発側（設計側と言っても良い）の品質評価が行いえるかと考えた時，十分とは言えないのではないだろうか．

　例えば，「保守性を全く無視して，ＤＥＦＢ（定義文）で記述したプログラムにバグが１件も無い」場合，このプログラムは品質が良いと言えるのかという問題が存在する．

　従って，現時点で不可能としても，品質特性要因，品質特性と代用特性の因果関係等が，量子化され定量的モデルとして扱える様になることが重要と思われる．

２．　品質の定量的把握が行えたとして，マネジメントレベルから考えた場合，属人的要素の多いソフトウェア開発に於いて，定量的品質尺度が，開発担当者の評価基準となりえるかどうか疑問が残る．

３．　今回のワークショップへの参加を通じて，「ソフトウェアの品質定量化の枠組」を当社の開発システムへどう展開，応用していくか検討したい．

山城　英彦　　（３３歳　ＡＢ型）
カシオ計算機（株）羽村技術センター
情報機器事業本部第１開発部
郵便番号：１９０－１１
住所　：西多摩郡羽村町栄町３－２－１
電話番号：０４２５－７９－７５６３

経験：ミニコンピュータ　オンライン　システム
　　マイクロコンピュータ　制御ソフト

ポジション　ステートメント＜第２版＞

１．はじめに

先ず、本ワークショップの、参加資格（年齢制限）をオーバしているにもかかわらず、参加できるようになったことを感謝しています。ありがとうございます。
さて、私は、本年６月にカシオ計算機（株）にデューダし、開発の現場からはなれることになりました。
そこで、このポジション　ステートメントには、「ソフトウエア品質定量化の枠組み」についての意見と、経験事例について述べて見ようと思います。

２．ソフトウエア品質定量化の枠組みについて

「ソフトウエア品質定量化の枠組み」は、これまで抽象的に論議されていた品質について定量的なアプローチがなされた点で賞賛に値するものと考えます。
しかし、これを現場に適用するのには、以下の疑問を感じます。

１）表Ｔ２を用いた、選択アルゴリズム（Ｐ．１４）で加重平均を行っていますが、各係数に重みづけする必要が出てくるように思われます。また、各係数の設定基準が明確になっていないと、現場の人間は、納得しないのではないでしょうか？

２）表Ｔ２で、係数値が１０のものがありますが、そのＳｕｂ　Ｆａｃｔｏｒは、対応するＳｕｂ　Ｃｒｉｔｅｒｉｏｎですべて説明しきれるものなのでしょうか？

３）ｔ１，ｔ２，ｔ３，ｄ１，ｄ２，ｄ３の量子化の方法論が無いと、実施に際して現場の人間を納得させられない思います。

４）現場の人間は、「明日の納期と、今日の品質」の間で苦労しているのが現状の様な気がします。
その人達に、品質を定量化して確認してもらうには、「率」とか「係数」という数字ではなく、ＱＣ７つ道具のような「直感的な」表現方法が必要と思われます。

自分自身としては、何の知識もないのに、批判的なことを言うのは、差し出がましいと思いますが、以上が私の感じた点です。

3．事例

実際、開発の現場で実作業に携わりながら、何らかの形でソフトウエアの品質向上の努力をするのは並大抵なことではないと（私は）思います。
しかし、何もせずに手をこまねいている訳にもいかないので、あれこれ手を打った中で、有効と思われるものを紹介します。
（多分皆さんは、すでに経験されて、周知のことと思われますが．．．）

（１）テスト作業の評価を行う上で、チェック項目消化曲線と、バグ累積曲線を書いて、分析しました。
その結果、以下の点が、参考となったので一応書きます。

ａ）チェック項目消化曲線と、バグ累積曲線を書くだけでは、テスト作業の管理を行なう上では不十分であり、テスト作業の実施状況も併せて並記することが必要である。

ｂ）チェック項目の消化が予定より進んでいることが、必ずしも、テスト作業の進捗が良いことを示すものではない。（進捗管理に用いる上では、注意が必要である）

ｃ）類似するシステム間では、テスト項目が似てくるためバグ累積曲線の収束状況を終了基準の一つとして用いることが出来る。

4．終わりに

このような経験しかないのにもかかわらず、ソフトウエアの品質に関する仕事に携わってしまいました。
身の程知らずと思いますが、宜しくお願いします。

以上

テスト技術の定量的把握

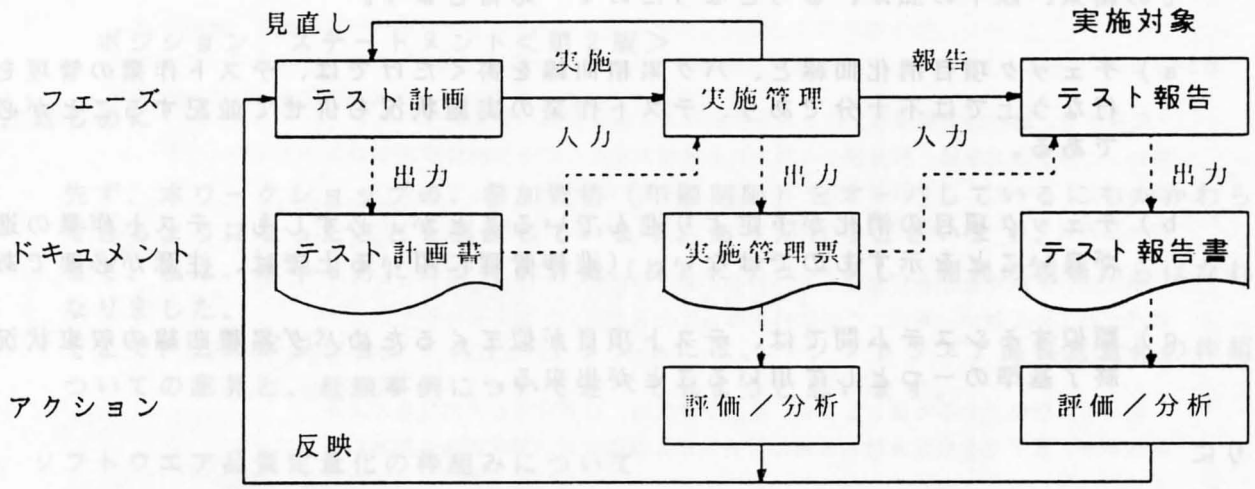
1）定量化の中の位置付け

Ｃｒｉｔｅｒｉｏｎ　＝　テスト技術

Ｓｕｂ　Ｃｒｉｔｅｒｉｏｎ　＝　機能テスト

Ｐｒｏｃｅｓｓ　Ｍｅｔｒｉｃ　＝　詳細化が必要

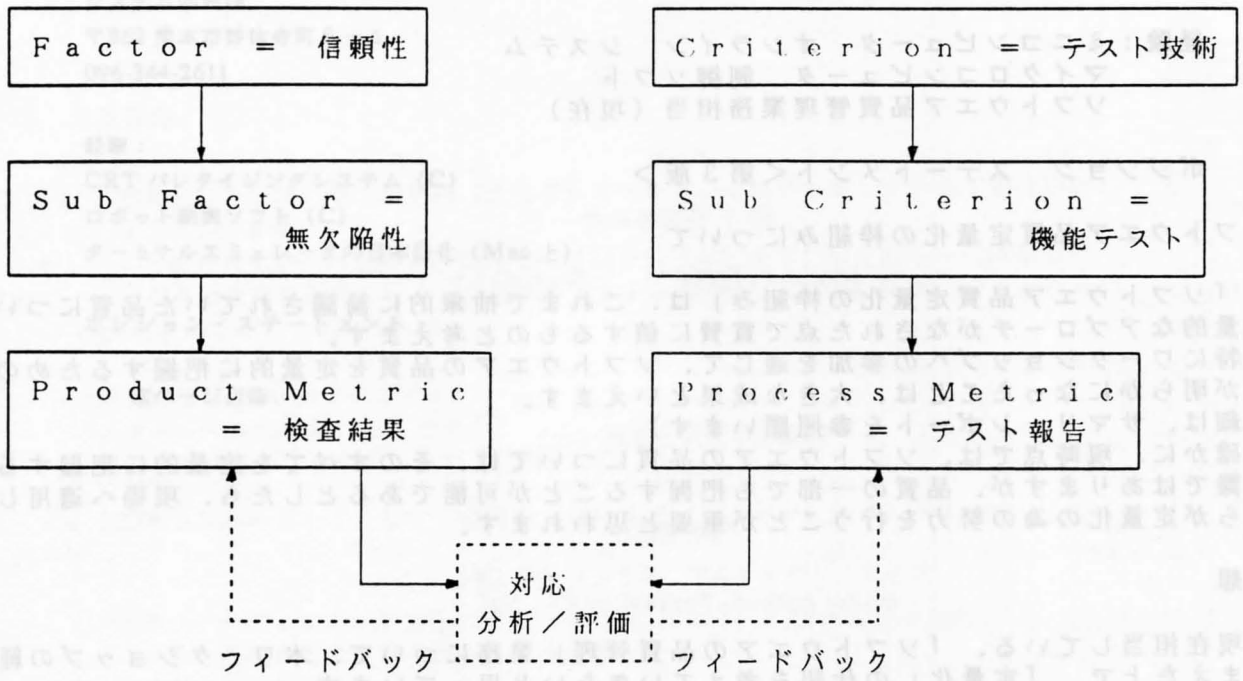
2）テストの体系化



3）テスト報告の定量化

項番	代用特性		計測項目	評価（予定）
1	障害摘出状況	バグ	発生バグ数 未解決バグ数	未解決バグ占有率
		テスト	未実施テスト項目 未解決バグ数	未実施テスト項目占有率 テスト実施率
2	プログラム網羅状況		（カバレージ） （未実施）	（カバレージ率）
3	作業状況	テスト時間	テスト期間 累積テスト時間	単位項目テスト時間
		作業量	テスト項目数	
		作業方法	テストツール テスト環境 未実施テスト項目	ツール使用率 テスト環境網羅率 バグ収束率

4）今後の課題


Ｆａｃｔｏｒ　＝　信頼性
Ｓｕｂ　Ｆａｃｔｏｒ　＝
無欠陥性
Ｐｒｏｄｕｃｔ　Ｍｅｔｒｉｃ
＝　検査結果
Ｃｒｉｔｅｒｉｏｎ　＝　テスト技術
Ｓｕｂ　Ｃｒｉｔｅｒｉｏｎ　＝
機能テスト
Ｐｒｏｃｅｓｓ　Ｍｅｔｒｉｃ
＝　テスト報告
対応
分析／評価
フィードバック
フィードバック

山城　英彦　　（３３歳　ＡＢ型）
カシオ計算機（株）羽村技術センター
情報機器事業本部第１開発部
郵便番号：１９０－１１
住所　：西多摩郡羽村町栄町３－２－１
電話番号：０４２５－７９－７５６３

経験：ミニコンピュータ　オンライン　システム
マイクロコンピュータ　制御ソフト
ソフトウエア品質管理業務担当（現在）

ポジション　ステートメント＜第３版＞

１．ソフトウエア品質定量化の枠組みについて

「ソフトウエア品質定量化の枠組み」は、これまで抽象的に論議されていた品質について定量的なアプローチがなされた点で賞賛に値するものと考えます。
特にワークショップへの参加を通じて、ソフトウエアの品質を定量的に把握するための方向性が明らかになったことは、大きな成果といえます。
（詳細は、サマリーレポートを参照願います）
確かに、現時点では、ソフトウエアの品質については、そのすべてを定量的に把握する事は困難ではありますが、品質の一部でも把握することが可能であるとしたら、現場へ適用して、自らが定量化の為の努力を行うことが重要と思われます。

２．感想

現在担当している、「ソフトウエアの品質管理」業務について、本ワークショップの経験を踏まえた上で、「定量化」の仕組を考えていきたいと思っています。
しかし、「定量化」の前提となる障害報告の収集等の課題についても、今後検討して行かなければなりませんが、ソフトウエア品質の定量化を前提として、アプローチすることが出来ればよいと考えています。
ワークショップへの参加により、今まで漠然としていた「定量化」の手法が少しは、明確になってきました。
この点だけでも、私にとっては、大きな成果といえると思っています。

以上

Position Statement

門岡　春雄（２４歳　Ａ型）
ヒラタソフトウェアテクノロジー（株）
システム開発課
〒862 熊本市妙体寺町５－４
096-344-2611

経験：
CRT パレタイジングシステム（C）
ロボット制御ソフト（C）
ターミナルエミュレータの日本語化（Mac 上）

ポジション・ステートメント：

次ページ以降.

Position Statement

第 7 回若手の会

定量化の基準？、その前に

平成元年７月２２日（土）

門岡 春雄

ヒラタ　ソフトウエア　テクノロジー（株）
Hirata Software Technology Co., Ltd.

1．　はじめに

私は、「・・的」という言葉が好きになれません。プロ野球中継で「今のは、タイミング的にあってませんね」とか、「トータル的に見ればそうであって・・・」とか、「的」をつけた言葉をよく耳にします。

それぞれ何をいいたいのかわからず、説得力がない解説になってしまいます。中には、「的」がつかないとわからない「具体的」や、わかって「的」をつけている文を見かけます。しかし、普段の会話や政見放送にまで「的」を乱発して人々の頭を乱しているのが現状でしょう。相手に意志を伝えられなければ誰も耳を傾けません。他人が聞いてわかるような解説をしてほしいものです。

2．　私がプロジェクトリーダに

私は、つい先日 3 カ月の短期プロジェクトのリーダー（以後 P.L）をやりました。定量化なんて遠い存在だったそんな私がリーダーになったプロジェクトは、 3 カ月の短期プロジェクトにも関わらず、納期は半月ほど延びてしまい各方面に御迷惑をかけてしまいました。

原因として、遅れの先見が出来なくてマネージャクラスへの報告が遅れ、結果報告になったことが考えられます。もう少し掘り下げるてみるとすべての期限の見積りミス、それ以前に定量化の基準となるものがわからないままリーダーをやったことに問題があると思っています。

定量化が"まだ"出来ない P.L がどうやったら円滑にプロジェクトを進められるか考えて見ました。

Position Statement

3. 先見報告

P.L に定量化の力がない時、必然的（あら？）にマネージャがプロジェクトの定量化を計ることになると思います。
しかし、マネージャというお仕事は、私たちには計ることが出来ないほど難しい仕事らしいので、P.L が今の状況を正確に伝える必要があります。しかし、この P.L は、マネージャとの Meeting の中でこう言いました。

「量的に言うと 3 日遅れです」
「スケジュール的に遅れそうです」
「残作業的に見ればもう少しです」（実話です）

ここで問題となるのは、どうしてこの報告でははまずいか、です。たぶんマネージャから見れば、

1. 作業（コーディング、テスト、デバッグ etc..）が何 % 残っているのか
2. このまま進むとどの程度の遅れになるのか
3. いつ頃終りそうなのか

などが知りたいことでしょう。そうなると 1 ～ 3 までを具体的に数値であらわさないと判断しようがありません。ここで、最初にいった、《相手に気持ちを上手に伝える》ことが大切になってきます。
「的」の話はちょっと極端ですが人は、窮地に追い込まれると悪いことを出来るだけベールにつつみたがります。そのベールに「的」というのはピッタシなんですね。（よく人の話を聞いているとわかります）
私は、今の現状をどれだけ確実に人に伝えられるかが定量化への第一歩だと思います。今をしっかり掴んでいないのにその先の納期がどうのこうのというのは早すぎる話です。

プロジェクトのあり方の理想は、リーダーは、「出来上がったら、呼んでね」の一言で済ませ、定量化も何もいらないプロジェクトだと考えます。
しかし、そのレベルまで"まだ"到達していなければまずは、【結果報告】ではなく、【先見報告】が出来るようになるのが先だと思います。

4. 最後に

今回の若手の会には、長期プロジェクトの達人が集まると聞いています。今後長期プロジェクトを任される日がきっと来ると思います。その時に役立つような話をたくさん聞いて肥しになれば幸いです。参加することになれば、3 カ月プロジェクトのバグの数とその傾向をまとめたものを持っていきたいと思っています。

定量化なんてラララ～ラララララ～

ヒラタ　ソフトウエア　テクノロジー（株）
Hirata Software Technology Co., Ltd.

*1989.9.*吉日　　門岡 春雄

1．　定量化への第 1 歩

私は、定量化への第 1 歩として、【先見報告】という言葉を挙げました。それには、現状把握と自分が抱えている分の仕事の見積りが含まれています。
最近自分で立てた見積り（X 人日）に "* 1.5" やって報告しています。どうせ * 1.0 の見積りなんてあてにはならないから。:-)
不思議な話ですが "* 1.5" の見積りを立てるとその仕事は "* 1.2" ぐらいで済んでしまうのです。
私のここまでのソフトウェア人生の中の定量化とは、

「自見積りより * 1.5」

という言葉に集約されました。

2．　定量化なんて

前回の P.P には、少しだけきれい事を書きました。盛岡に来たかったからなんです。:-)
今思っていることは、「この業界に定量化なんて存在するの？」ということです。
創ったソフトの品質は、色んな本でその計り方を明記してあります。しかし、ソフトウェアを創るというのは所詮人がやることです。人がやることを、「過去にこんなバグが出た」とか「彼はこのぐらいのシステムにはこのぐらいのタイプミスを犯した」で計るなんて絵に書いた餅だと思います。
環境も違うし、対人も違うのです。
それでも計ろうとするならば、しないほうが賢明だと思います。データ通りに事が進むならマネージャやリーダなんかいらないでしょ！

誰か、「お前は、考えが浅い。こうは考えられないかい」と言ってくれる人を探しています。

Forever 「若手の会」

ヒラタ　ソフトウエア　テクノロジー（株）
Hirata Software Technology Co., Ltd.

1989.9.22(Fri)　　門岡 春雄

1.　ポジションの変化

（この項が事後の P.P だと思って下さい）

どう考えても、"手近な" バグ数で定量化を計ろうなんていうのは、やれるだろうけど、意味がないものではないかと思っています。
似たような Project で同じメンバで、環境（体調）も同じであれば前回のバグ数での見積りは可能かもしれませんが、そんなことはまれだと思います。
過去たくさんの Project のバグを色々分析して今後の Project の定量化を計るというメリットより、人を計るというデメリットの方が大きくなると思います。そういことをやる上司やマネージャは"クビ"ですね。:-)
ただ、そのバグ数で人を攻撃するのではなく不特定の人の意識革命の材料に用いるのであれば大賛成です。しかし、バグ数を意識革命に使うんだ、という意識革命をしないといけません。それだけで、定量化がどうのこうのとはまったくいえませんが・・・。

で、私の立っている場所の変化ですが、あまり変わっていません。
しかし、定量化のことをまったくわからなくてただ「意味がない」といっていた時よりも、大場さんや松本さんの話を聞いたり、みんなと談笑したりした今のほうがより身近になった気がしています。もう少し踏み込んでみたいな、と思ったことがポジションの移動です。
久保さん著の「富士通におけるソフトウェア品質保証の実際」を読んでもう少し勉強したいと思います。

2.　完走文

2.1.　プロローグ

新幹線に乗るのは高校の修学旅行以来だな、と思いながら上野駅から東北新幹線に乗り込んだ。東北が初めての僕は、車窓から見えるのどかな田園風景に胸を踊らせた。
果たして、僕の思惑は見事にはずれた。窓から見える景色は私の住んでいるところとあまり変わらなかったからだ。つまり、私の家の回りも田園だった。

そんなこんなで盛岡に到着した。

2.2.　自己紹介

さて、私の盛岡の感想です。

アルコールがダメな私は酒のことはよくわかりませんが食べ物は確かに・・・うまい。わんこといい冷麺といい南部料理といい。ただ、わんこそばだけはあと半年ほどお断りしたい。

食い気はこの辺でおいといて、もう一つ思ったことがあった。
それは、みんな結構おしゃべりということだ。プログラム委員の自己紹介も 10 分で終らせた人はおられなかったし、参加者の自己紹介も 1 分半なんて誰も守っていない。（私も人のことを言えた義理ではないのですが）何か溜っているものがあるのか、本当に喋るのが好きなのかは定かではない。
しかし、正式の場での発言が偏った人に集まっていたのは残念だった。もっと泥臭い話が活発に出るのかな、と思っていた私には期待はずれだった。

もう一つ残念なのは、回りの雰囲気が「定量化 -- 正しい」の空気が漂っていたことだ。結果としてこの公式がアタリだとしてもあれだけの人数の中に私だけが否定意見を持っているのはおかしい。それとも私自身がおかしいのか？:-)
もっと千差万別な意見が出てもよかったのではないか。世の中 " 右へならい " なのかな。

2.3. 雨の小岩井農場

あいにくの雨でせっかくの旅の楽しさも半減してしまった。小岩井農場でのソフトクリームも美味しかった。降り出したら屋内に討論場を移すなどして雨に振り回される人間の弱さを感じた。:-)

グループでの討論の感想は、第 1 班以外の班の討論内容のスケールの大きさに驚いた。他班の発表を聞いていると自分の小ささに恥ずかしくなった。ただ、最前線にいるものにとっては「机上の空論」の話が多いように感じた。
それはそれでいいことだと思う。若い人達が集まったのだから理想を語り合うのはいいことだと思うし、それが【若さ】だと思うから。
以前、誰からか聞いた、

「トップダウンのものに良いものはない」

というのを思い出した。

それにしても、そびえ立つ岩手山を見たかった・・・。

2.4. エピローグ

私なりにつっ走った 3 泊 4 日だった。途中で逃げ出しも、投げ出しもせずに完走したつもりだ。しかし、ゴールなんて気のきいたものはどこにもなかった。ひょっとしたら永遠にゴールなんてないのかもしれない。それがこの世界かな、とも思った。

どこかの班から、「しゃべるエアコンがあるのに何故コンピュータはしゃべらないのだろう？」という意見が出された。私の近くにはマシンに話かける人はいる。皆さんの回りにもおられると思う。:-)

いろんな環境の人と出会えたことが 1 番の収穫になった。こういった WS に参加すると閉じた世界が開けたような気分になる。来て良かったなあ、また参加したいと思った「若手の会」だった。

最後に、関係者各位の皆さんにあやまりたいと思います。ノイズばかりで御迷惑をかけました。どうもすいませんでした。また、みなさんとお会い出来る日を楽しみにしています。懲りずにつき合ってやって下さい。

Position Statement

小森林　正樹（２９歳　Ａ型）
（株）岩手電子計算センター
流通システム課
〒020 盛岡市松尾町１７－８
0196-51-2626 内線(255)

ポジション・ステートメント：

現在は，システムの設計からプログラムの作成テストと一連の作業を自分で行っている．そのための利点として全体の処理が把握でき個々のプログラムの必要機能が把握しやすいため，単体のテスト連動テストがスムーズに行なう事ができる．ただ，プログラムのチェックリスト等を使用していないため，チェックしたと思い込んでしまい，バグが発生してしまうケースも多々あるように思える．このことからプログラムの作成者と最終チェック者（納品前のチェック）は別々の人がすつ方がよいように思える．

又，プログラムの管理はできるだけ，プログラムリスト内に修正等のあった事をコメントとして残すようにしている．これにより，修正ミスは少ないように思っている．

小森林正樹
（株）岩手電子計算センター

（所感）

今回、品質の定量化について話をしたわけだが、自分の立場ではというとそのまま適用できないようである。

1．プログラム開発というよりは修正が多い。
2．日程的に短い。
3．工数が少ない（当初開発は別）。
4．日々の運用業務を個々で持っているため、プロジェクトによる開発になることが少ない、

など、色々と違いがあるように思われる。

特に委託の場合には、仕様書、プログラムが納品とならず成果物が問われることになるため、仕様書等の管理が不十分になり、プログラム仕様書に書く前にプログラム作成したりすることが多くなってしまう。というのが現状であるように思える。

そのため、今すぐにということは無理であろう。

又、話をしているとよく専門用語が出てきて意味不明なことも多々あり、いかに自分が新しい知識が乏しいかを痛感した。

多くの人々と出合い、いろんな人の考えを聞くことは、大変今後役に立ちそうな気がする。

今回同じワークショップに参加された皆さん、これからもよろしく！！

Position Statement

三浦　博（２８歳　Ｏ型）
（株）岩手電子計算センター
システム流通課
〒020 盛岡市松尾町１７－８
0196-51-2626 内線(256)

ポジション・ステートメント：

１．開発パターン

（１）ユーザーとの打ち合わせ
（２）設計
（３）プログラミング
（４）テスト（単体，総合）

現状の開発パターンは，（１）～（４）１人の人間が行っているが，時として，（３）（４）は他の人間も加わる場合もある．

（３）（４）への移行の場合，思い違いもあるため，十分に机上デバックを行なってから，（４）テストへ移行した方が，スムーズに作業行える．

テストの方法として，多種パターンのテストデータ，（数値などの羅列しているものなどは，同じ数値を用いない．数量と分割回数が０１と入っていても，誤って，数量の値が分割回数に入っているかもしれない．）を入力とし，出力をエデットをとって確認する．

２．改善策

（３）プログラミングと（４）テストを行なう人間は，異なった人間に作業を行わせる．（３）と（４）を同じ人間が行なうと，あるパターンしか頭にない又，思い込みで簡単なミスにも気付かないこともあるからである．

三浦　　博
（株）岩手電子計算センター
企業システム部流通システム課

＜所感＞

今回のワークショップのそれぞれの所属には色々の人がいた。メーカ、ロボット、計算センターなど、普段、余り顔を合わすことがない異色のメンバーであった。しかし所属は異なっても、悩みは同じで私はホッとした面があった。

地方に住む私と、他の人たちは、ほとんどが中央からきていて、技術、専門の知識にギャップがあり、世の中も広いものだと刺激を受けました。

今回、討論し合ったテーマについても、私自身、大変興味を持ちましたが、それを応用するとなると問題点がある様です。

①　プロジェクトの問題
②　納期の問題
③　意識の問題

しかし、今回刺激を受けたことについては、大いに仕事に役立てたいと思います。

”やさしい気持でシステムを作れば、やさしいシステムが出来る”いい名言です。日頃、私もそう言った気持ちで仕事をしたいと思います。

Position Statement **No. 29**

金沢　美佐雄（２９歳　Ａ型）
（株）岩手電子計算センター
公共システム部公共第１課
〒020 盛岡市松尾町１７－８
0196-51-2626 内線 (237)

経験：
水道料金計算及び収納消込業務
水田農業確立対策

ポジション・ステートメント：　ソフトウェア品質の定量的把握を目指して

ソフトウェアの品質を高める手段として，要求仕様や基本設計などソフトウェア開発の早い段階でレビューを行なうことは重要なことである．また同時にプログラムをコーディング後，実際に実行させてみる「テスト」も重要な手段である．

しかし，「テスト」作業は，限られた期間，コストまた環境のもとで行なわければならないことがほとんどである．『まあ，こんなもんでいいか』といってテストを終えてしまったことも幾度かある．

品質を高めたと自分自身で納得するために，数多くのテストデータをつくってみたりする．しかし数多くのテストデータを作るには時間と手間がかかる．どうすれば効率よくテストできるかということが問題である．またテストを終了すればよいかという問題もある．テストを長期間行ったからといってプログラムにエラーが全くなくなったという保証はない．ユーザーが予想外のデータを入力したり，使い方をして正常に動作しなかったことは結構ある．

それでもテスト作業をすることでプログラムの中にあるエラーが減少するのは確実であるので，ある一定の期間テストをする必要はある．そこで次のようにテストを完了する基準をつくる．

（１）テスト期間の設定

テストのためあらかじめ期間を設定し，その期間が終了したらテストを完了する（実際，私の経験ではその期間の終了日はその製品の納期にあたることが多い）．この基準だと，テスト作業の中身がうすくても期限がくると終ってしまうが，実際は期限が近づくと残業してでも作業してしまうので，ある程度エラーが消え品質が向上すると思われる．

（２）テスト項目の設定

テストする前に，テストの項目数を設定する．そして，設定された項目が終了したらテストを完了する．しかし，この項目をどう設定するかが重要な作業である．やはりある程度経験がものをいうように思われる．また，プログラムの重要度をみて重要なものには項目数を多く設定する．

以上の基準によってある程度ソフトウェアの品質を把握できると思われる．

Position Statement

金澤　美佐雄
(株)岩手電子計算センター
〒０２０　岩手県盛岡市松尾町１７－８

ソフトウェア品質の定量化を検討する上で重要なことは、最終的にはユーザーの立場になって品質というものを考えることだと思う。
ユーザーがソフトウェアに対して、どのような品質をどの程度求めているかということが評価の基準を大きく左右する。

信頼性、機能性、効率性、使用性、保守性、移植性などと品質を評価する要素はあるがユーザーがどのような品質特性を重要視しているかということになる。
例えば、一回切りのシステムならば、そのシステムの拡張性を評価したところでそれは無意味である。またシステムが特に実行効率性を求めているのであれば、その特性を特に考慮して品質を評価しなければならない。

つまり、ユーザーがどのような品質を求めているかということを考え、品質特性を測る要素に重みづけをして評価しなければならない。
重みづけをすることによってバランスのとれた品質の評価というものができるのではないかと思う。

最後に、現在自分が開発したソフトウェアの品質の評価を自分で評価しているというよりも、納品後ユーザーに評価してもらっているという現状を考え、このソフトウェア品質定量化の問題をどう現場に活用していくかということを今後考えていきたいと思っている。

感想

今回初めてこのようなワークショップに参加しましたが、雰囲気が固苦しくなくグループの仲間もたのしい人達ばかりで、とても有意義な時間を過ごせたと思います。

地方の一会社で仕事をしている私にとって、全国のソフトウェア業界のいろいろな人と話ができたことは、大変貴重な経験だったと思います。

Position Statement（ワークショップ終了後改訂版）
注！！　：　ＷＳ後に変化のあった部分に”＞＞”マークを付与して追記しました。

馬島　宗平（30歳 B型）
NTTソフトウェア開発センタ　技師
〒108　港区港南 1-9-1 NTT品川TWINSデータ棟 8F
TEL 03-740-5565

経験：　１．CHILLリース解析・表示プログラム
　　２．問処管理システム
　　３．ドキュメント作成支援システム（開発期間：２年、機能追加／保守：２年）

P.S:

１．はじめに
　私は交換機の制御プログラムを開発しているセンター内のプロジェクトに対して、ドキュメント絡みの開発をより効率化するためのサポートツールを開発・提供しています。部隊は、適用Ｇと開発・維持管Ｇの２つに分かれ、私は、その開発・維持管理Ｇのリーダをしております。（総規模：約200KL　（C言語及び自作言語））

２．品質の定量化についての問題点
(1)　定量化を測り、品質の尺度を与えるには、定量化の方法が開発技法・対象分野等に依存しないものである必要がある。（その結果の値が異なるのは当然？）
　ところがバグの定義、バグ要因の分類項目やその定義、レビュー指摘事項の分類等の多くの品質管理データは実際のデータを取ろうとする場合に一意に分類できず、結局個人の主観で分類されてしまうものが多い。
　では、なぜ分類できずに個人の主観が入るかと考えると、プロジェクトで採用する開発技法やソフトウェアの対象により品質の考え方が異なるためと考えられる。
>>　データをとる目的が明確であり、それを全員がよく理解していればそんなにひど
>>　くバラつかないのではないか、という意見をもらった。確かにそうであるが全員が
>>　理解するという事はなかなか実現が難しい　：＜
定量化のためには－＞①　どの様な場合にも一意に決まるような品質管理データを定義する必要がある。
>>　ここの所は、現在はどんぶり勘定的やり方になっているところが多く、
>>　そこを少しでも合理的にしようというのが品質定量化の課題と認識した。
②　対象毎の品質の考え方と品質管理データの関係を明確化する
>>　単に対象毎と考えていたが、ＩＳＯの考え方、ユーザ、開発、管理といった見方
>>　と各々の品質の見方があり、より整理して考えることができるようになった。
(2)　計画当初の見積は経験則である
　何か開発を行おうとすると、多くの場合にＢＩ終了程度で規模と工数を見積るが、実はその時点では見積るために必要な情報が揃っておらず過去の経験から推定することとなる。
　ところが、割り当てられたメンバが新人ばかりだったり、ＢＩで重要な見落としがあるとこの見積は大きく狂い、計画の変更ができない時期になっていると品質の低下で不足工数を埋め合わせることになる。
定量化のためには－＞①　工数見積には、メンバの生産性やＢＩの品質と言った要素を埋め込む必要がある。
>>　メンバの生産性については、プログラマ性能（個人／グループ）の話が参考に
>>　なったが、感情的な要素を含む話なのでなかなか難しいと思った。

(3) 異常データの対策は、一意には決まらない。

定量化を行うことは必要なのですが、そのデータが異常値を示し始めた場合の対策がマニュアル化されていない。例えば、レビューでの指摘件数が標準値を下回っている場合などに単に標準値を下回っているからレビューをやり直すでは、メンバは何をして良いか分からず困ってしまう。このような場合に何をどうすれば良いかを適切に指示するのがＰＬの力量であるが、データをどう読みどの様に指示するべき化までマニュアル化したいものである。

(4) 設計の品質は管理しにくい

設計の善し悪しが重要であることは分かっているのだが、その品質の測定は困難をきわめる。特に、漏れ、抜け、不足の類はどの様なドキュメントを使用してなん枚ドキュメントを書いたか、等で測定されるようだが、どの様な設計にはどんなドキュメントを使用するべきかは設計者に任されているようである。

（特にＢＤ，ＦＤ工程が著しい）

>> 図表記載率、ユーザ用件展開率等、このような尺度を検討されていることが
>> 分かり、参考になった。討論のなかでも話題になったが結局は定説はなく、
>> 地道な体系化（まずできそうな事からやる）が必要と痛感した。

(5) 所詮人間は間違える

品質の定量化のためにデータの収集、ＤＢへのデータ投入、転記、計算、グラフ化等を行う訳だが、その手順が余りに複雑であると、且つツールが完備していないと、人は面倒になり間違いも引き起こせばずるをしようと考えても不思議ではない。しかし、こうなると品質管理データは正確にプロジェクトの状態を示さなくなる。

定量化のために－＞① 定量化の方法、データ収集は単純明快である必要がある。
② 複雑になる場合には、ツール化して提供する必要がある。

>> ここの話は、定量化に必要な条件として同意が得られたと思っている。

＜感想＞

今回の盛岡でのワークショップは、大変自由な雰囲気で形式ばらずに議論ができたことが最大のポイントであった。メンバが若い人中心ということもあろうが、発想、着想を得るにはよい環境であったと思う。また、事前の講義内容も以後の話を展開するのに必要十分な内容で良かったと感じた。ＰＣの方の御尽力のおかげで昼も夜も快適に作業？でき、グループとしてもいい雰囲気で取り組めた事に感謝しています。

内容的には、いま一歩突っ込みが足らなかった気がしますが専門家でなくこの分野の素人の作業結果としては一応の評価ができるレベルと考えています。残念なことは、最終発表の時に発表の不備を前夜の飲み過ぎを言い訳とするような内容があったことです。社会人としてこのような発表を行う以上しっかりと行わないと次回以後にこの自由な雰囲気での討論ができなくなる可能性があると思います。

ワークショップのコントロールは難しい事もあると思いますが、あの自由な雰囲気を失わず、来年以後もこのワークショップが開催され有意義な成果が出ることを期待したいと思います。

富士通株式会社
ソフトウェア事業部
中川　明生

ソフトウェアに対する品質管理について

1. 現行のソフトウェアに対する品質管理

現行のソフトウェアに対する品質管理は，以下のようなレビュー／テストを中心とした方法により行われている．

- 各設計工程における設計資料のレビュー
- プログラミングソースのレビュー
- モジュール単体テスト
- モジュール結合テスト
- コンポーネントテスト
- システムテスト
- 製品テスト

すなわち，ソフトウェアに対するレビュー／テストを行い，製品の品質を見極めている．また，ソフトウェアに不良が発生した場合には，不良原因の調査，類似不良の検出，不良の修正（類似不良箇所の修正も含む），不良が発生した要因の分析等を行っている．

このように，現行のソフトウェアに対する品質管理は，不良が発生した場合にそれに応じた対策をとるというような事後的なものが中心となっている．

2. 今後のソフトウェアに対する品質管理

現行の事後的な品質管理方式から，より前向きの品質確保対策へ移行していくためには，以下に示すような対策が考えられる．

- ソフトウェア製品に対する品質情報の収集
- モジュール相関関係情報の収集
- 各工程ごとの品質分析の推進（工程毎品質管理の強化）
- ソフトウェアの部品化の推進
- 理解の仕易いドキュメントの作成
- ハード／ソフトインタフェースの論理化
- 開発工程に応じた業務の専門化

2.1　ソフトウェア製品に対する品質情報の収集

開発が行われているソフトウェアの品質情報を各工程単位（機能設計／構成設計／詳細設計など）で収集する．この情報の蓄積により，品質の評価をデータに基づいて行うことができる．また，複数ソフトウェアのデータを基に相関関係の分析等も行うことができる．

2.2　モジュール相関関係情報の収集

ソフトウェアを構成する各モジュール間の関連情報を収集する．これにより，あるモジュールに障害が発生した場合の影響範囲を見極めることができる．

2.3　各工程ごとの品質分析の推進（工程毎品質管理の強化）

各開発工程での品質見極めを行い，十分な品質を確保した上で次工程の作業を開始する．これにより，前工程を原因とした品質不良を極力抑えることができ，各工程での作業を円滑に行うことができる．

2.4　ソフトウェアの部品化の推進

ソフトウェアのモジュール化は行われてきているが，そのモジュールを共通に利用できる部品としては，作られていない．しかし，モジュールを共通に利用できる部品（十分に品質が確保されているものとする）として作成すれば，利用する部品についての品質は考慮する必要がなくなり，部品を組み合わせることにより実現されるコンポーネント全体の品質についての考慮だけでよいことになる．よって，部品化により，品質にたいする考慮部分を極小化できる．

なお，基本ソフトウェアについては，各コンポーネントの性能がシステム全体の性能に影響を及ぼす．このため，各コンポーネント単位で性能優先の設計をしており，各コンポーネント間でのモジュール共通化は，難しい環境にある．

2.5　理解の仕易いドキュメントの作成

各工程での設計資料，ユーザマニュアル等のドキュメント類は，次期の開発を行う際に従来の機能を理解するための参考資料となる．このため，ドキュメントに不備があると，従来機能の理解不足／誤解により障害を作り込む可能性がある．よって，設計ドキュメントは，誰が見ても理解ができるような記述内容である必要がある．

なお，現行の開発作業は，同一担当者が一貫して行う場合が多く，その担当者の熟練レベルに応じてドキュメント類の記述内容が左右される場合が多い．すなわち，熟練度の高い担当者の場合，常識としてとらえられて記述されない部分が多い．また，熟練度の低い場合は，担当部分のみの記述となり，相互関係等の関連情報については，欠落し易くなる．よって，理解の仕易いドキュメントを作成するためには，担当者の技術水準に左右されない標準化を図る必要がある．

2.6　ハード／ソフトインタフェースの簡素化／論理化

ハードウェアを制御するためのサポートソフトウェアが多く出現してきているが，ハードウェアとソフトウェアとのインタフェースが複雑であるとその制御も必然的に複雑になってくる．よって，ハードウェアとソフトウェアとのインタフェースは，必要最小限度のものとし，ハードウェアの物理構造を極力意識する必要がないインタフェースとする．これにより，ソフトウェアは，限定されたハードウェアインタフェースのみを意識すればよく，処理の簡素化が進み，障害が入り込む余地を抑えることができる．

2.7　開発工程に応じた業務の専門化

ソフトウェアの開発も多様化しており，様々な作業が必要となってきている．このため，作業に対する熟練不足／理解不足から作り込まれる障害，または，障害の見落としを極力抑えるために，各作業における専門化を推進し，作業水準の高品質化を図ることができる．さらに，この環境で生産されるソフトウェア，ドキュメント類の品質も高いものとなる．

3.　各対策の評価

2.章で示した対策で，現状のソフトウェア開発におけるその必要性と実現性を評価してみると以下のようになる．

- ソフトウェア製品に対する品質情報の収集，モジュール相関関係情報の収集

  製品の品質を評価するための基礎データの収集であり，その必要性はある．しかし，品質そのものを確保するための効果としては，間接的であり，データ収集のための制度の確立，データの利用方法の検討など課題も多い．このため，短期的な対策ではなく，長期的な対策として取り組む必要がある．

- 各工程ごとの品質分析の推進（工程毎品質管理の強化）

  開発工程そのものに関与する対策であり，その効果は，直接，ソフトウェアの品質を左右するものと考えられる．この対策を通じて，各工程において品質を捕らえることから，一面的に品質を捕らえるのではなく，品質を多面的に捕らえることができる．また，この対策は，日々の業務への導入も容易であり，その効果は大きいものと考える．

- ソフトウェアの部品化の推進

  部品化を推進することにより，製品の開発／品質分析を行う際の考慮範囲を極小化する意味はあるが，品質を確保する意味においては間接的である．また，基本ソフトウェアについては，共通化が可能なのは，同一コンポーネント内に限られる場合が多く，それほどの効果は期待できない．

- 理解の仕易いドキュメントの作成

  設計資料の高品質化を目的としたものであるが，その効果がでるのは，機能追加，機能改善等の二次開発以降が中心である．このため，一次開発で作成するソフトウェアの品質確保への貢献度は，それほど大きいものでないと考える．

- ハード／ソフトインタフェースの論理化

  インタフェースの簡素化／論理化により，製品の開発／品質分析を行う際の考慮範囲を極小化する意味はあるが，品質を確保する意味においては間接的である．また，サポートするハードウェアによっては，現行の技術レベルでは論理化／簡素化が難しい場合があり，その効果を期待できない場合もある．よって，この対策は，長期的にハード／ソフト間で検討し，取り組む必要がある．

- 開発工程に応じた業務の専門化

  現行の開発体制への影響が大きく，大規模な組織改編を必要とする．このため，開発工程をどのように分類するか，分類された開発工程に応じてどのように組織を編成するか，などを十分検討した後に実際の対策を実行に移すことができる．このため，その効果は認めるが，対策の実現に向けては検討課題が多くあり，現状では，その実現性は，低いと思われる．

以上のことから，“各工程ごとの品質分析の推進”が最も効果が現れるので優先的に実施する必要があると考えられる．それを実際のケースをもとに検証してみる．

社内において品質不良と評価された製品（２１製品）について，各開発工程（機能設計／構成設計／詳細設計／コーディング）における品質分析作業量（レビュー量）の当社作業基準（単位ステップ当たりのレビュー量）のクリア率を分析した結果を図１に示す．

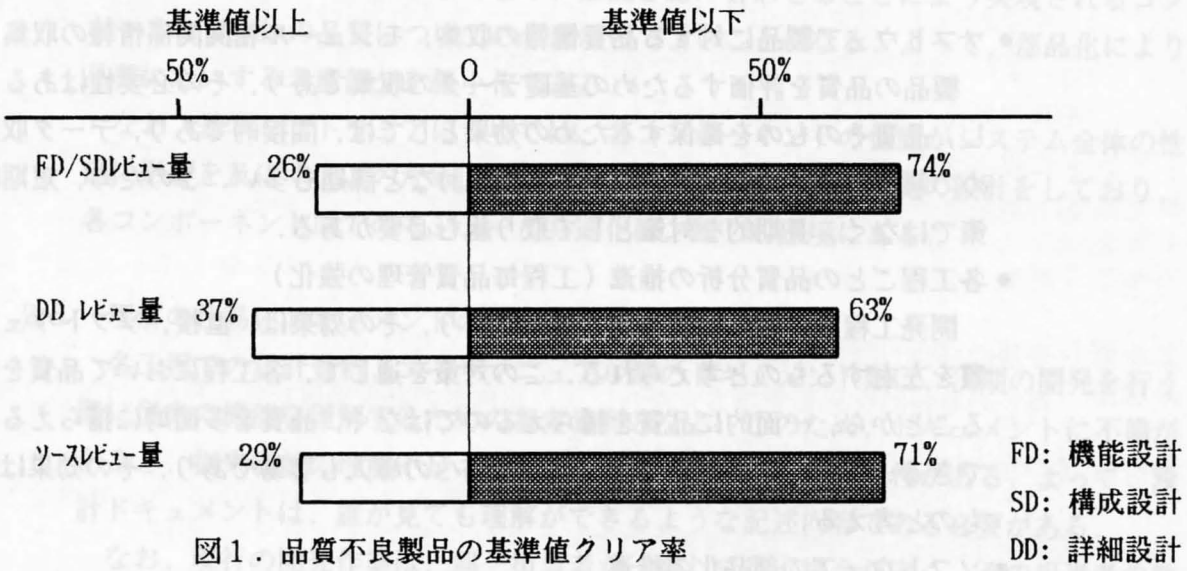


図１．品質不良製品の基準値クリア率

上記の分析結果から分かることは，品質不良と判断された製品は，各工程における品質分析作業が，当社の基準値をクリアしていない割合が高い．よって，品質不良を発生させないためには，この品質分析作業により各工程における品質の見極めを十分に行い，前工程が原因となる品質不良発生の可能性を極力減少させた上で次工程の作業を開始することが必要であると考える．

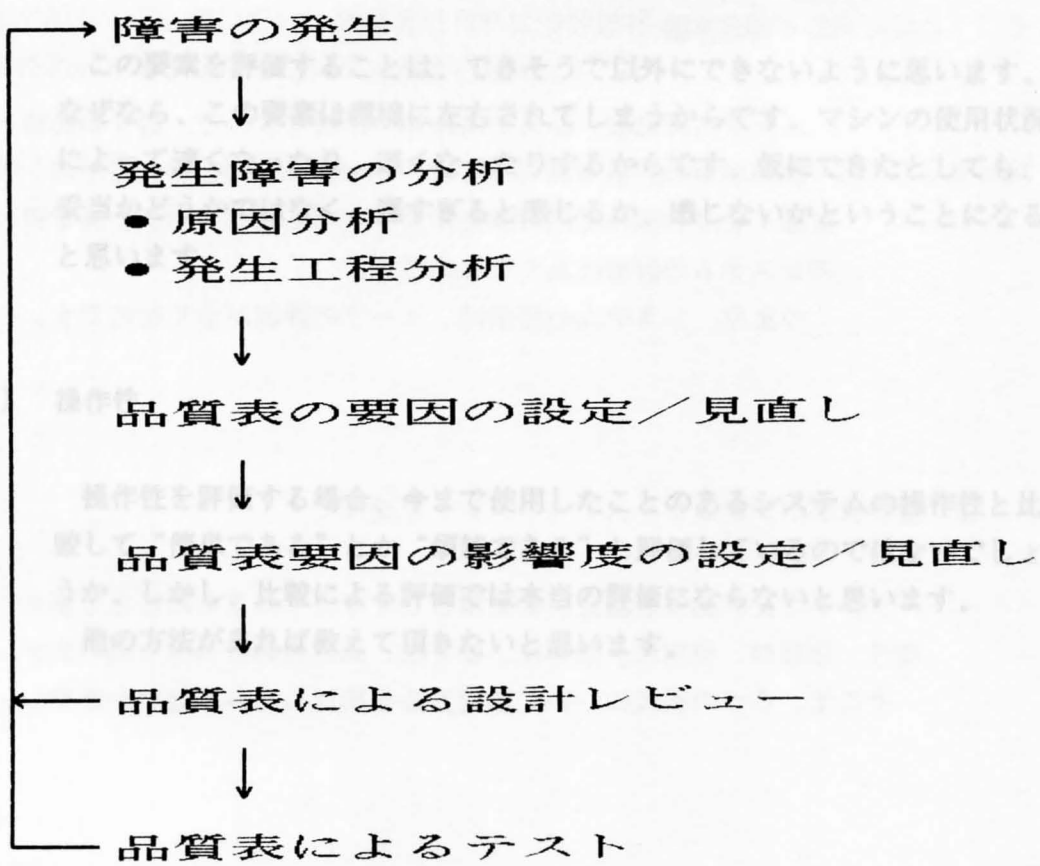
富士通株式会社
ソフトウェア事業部
中川　明生

プログラミングワークショップに参加して（感想）

このワークショップに参加して感じたことは，かなりの人が品質問題で困っているのだなということです．即ち，品質不良を検出し，それに対する対処はしているが，根本的な品質対策をどのように行ったら良いかが明確になっていないということです．皆さん，暗中模索の状態のようである．

このワークショップを通じて得られた実際の品質対策としては，まず初めに，ソフトウェア開発の現状を分析する必要がある．次に，この分析結果をもとにソフトウェア開発における品質対策を立案し，実施する．さらに，実施後には，ある一定期間ごとに，対策の見直しを行い，対策の陳腐化を防ぐことも必要である．

また，ソフトウェアの品質をよくするためには，設計段階での品質対策が有効であることが再認識させられた．その理由としては，後工程へ行けば行く程，障害の検出／修正作業が困難になり，開発工程への影響も大きいためである．よって，このことをもとに，設計工程における品質管理をどのように行うかを検討した結果，以下のようなアプローチにまとめられた．この検討結果をもとに開発工程における品質管理に取り組んでいきたいと考えている．



Position Statement

高橋　良一（２８歳　A型）
日本電子計算（株）
情報システム事業部システム１課
〒135 江東区東陽２－４－２４
03-5690-3231 内線(4040)

ポジション・ステートメント：

ソフトウェアの品質とは，一体何でしょうか．品質の意味がよく理解出来ませんのでソフトウェアの評価ということで述べたいと思います．

ソフトウェアを評価する要素（６つ）と問題点を挙げたいと思います．

（１）信頼性　　：全パターンのテストデータを作成するのは難しい
（２）処理時間及びレスポンス：環境に左右されてしまう
（３）操作性　　：比較するものがないと評価しにくい
（４）保守性及び拡張性　：評価しづらい
（５）マニュアル　　：評価できる
（６）ユーザの評価　　：使用してもらわないと正しい評価ができない．

以上のことから考えると，要素ごとに定量的評価はできるが，ソフトウェア全体についての評価となると，各要素の評価をプラスしたものと同じということにはならないと思います．そこで，ソフトウェアを評価しようとした場合，要素を洗出し，ソフトウェア全体を１００％として，各要素の％を決めて評価するのも１つの方法だと思います．

Position Statement (Part-Ⅱ)

高橋　良一
日本電子計算（株）

1．ソフトウェアを評価する要素（6つ）について

（1）　信頼性

信頼性があるかないかを判断する方法は、テストデータで処理を行い、その結果が正しいかどうかで判断しています。この方法で信頼性を確かめるためには、全パターンのテストデータを作成しなければなりません。しかし、全パターンのテストデータを作成するとなると時間がかかり過ぎてしまう。もう一つの方法としてカバレッジで判断する方法があります。この方法は、単体テストのようなもので、組み合わせのテストではないという問題が残ると思います。

しかし、現状から考えるとカバレッジで判断するしかないと思います。

（2）　処理時間及びレスポンス

この要素を評価することは、できそうで以外にできないように思います。なぜなら、この要素は環境に左右されてしまうからです。マシンの使用状況によって速くなったり、遅くなったりするからです。仮にできたとしても、妥当かどうかではなく、遅すぎると感じるか、感じないかということになると思います。

（3）　操作性

操作性を評価する場合、今まで使用したことのあるシステムの操作性と比較して“簡単である”とか“煩雑である”と評価しているのではないでしょうか。しかし、比較による評価では本当の評価にならないと思います。

他の方法があれば教えて頂きたいと思います。

（４） 保守性及び拡張性

保守性及び拡張性を評価する場合の要素として、“ドキュメントの正確さ”又は“いかに部品化されているか”などが考えられると思います。ドキュメントについては、システムと一致しているかで評価できると思います。部品化については、システム全体の中の部品化の割合で評価できるような気もしますが、部品化がされていないからといって保守性が悪いということにもならないと思いますので、部品化によって保守性及び拡張性を評価することはできないと思います。

（５） マニュアル

マニュアル（運用マニュアル、操作マニュアル）の評価は、マニュアルに書かれている内容が正しいかどうか、またシステムと一致しているかで評価できると思います。しかし、マニュアルの書き方又は使いやすさについては個人の感覚により異なるので、基準を設定するのは難しいと思います。

（６） ユーザの評価

ユーザの評価は、システム全体の評価ですので、個々の要素と異なります。

他の要素の評価が良くてもユーザが良くないといえばそのシステムの評価は悪くなり、他の要素の評価が良くなくてもユーザがこれで良いといえばそのシステムの評価は良くなるのです。

つまり、システムの評価は、ユーザの評価が全てなのです。

２．資料及びポジション・ステートメントを読んで

ソフトウェア品質を測定する要素として考えられるものとして多かったのは、機能性、信頼性、使用性、効率性、保守性、異色性の６つだと思います。

そこで、６つの要素について測定方法を検討してみてはいかがでしょうか。

1989.9.25

ワークショップの感想

日本電子計算（株）
高橋　良一

“ソフトウェア品質の定量的把握”は可能かどうか疑問をもちながら参加しました。そして参加者の意見を聞いてみると、参加者の全員が定量的把握を望み、ほとんどの人が可能であると考えていることがわかりました。そして各社が何らかのデータを収集し、ソフトウェアを定量的に把握しようと努力していることがわかりました。

定量的把握については、今回のワークショップに参加して可能であると思うようになりました。定量的には把握するためのメトリクスについて具体的な意見は言えないのですが、おそらくまずプロセスメトリクスが確立され、次にプロダクトメトリクスが確立されると思います。そして、メトリクスはおおまかすぎず、細かすぎないものになると思います。（大変抽象的な意見ですいません）

その他に、今回ワークショップで感じたことは、それぞれの立場の人の意識改革を望んでいる人が多いことでした。（例えば、システムはバグはあってあたりまえ、バグは恥ずかしいことではない、管理者は何かにつけ納得のいく数字で説明しろと言うが、納得のいく数字は出せないことが多いｅｔｃ）

最後に、今回いろいろお世話して下さった岩手電子計算センターの皆様に心から御礼申し上げたいと思います。本当にありがとうございました。

Position Statement

岩井　邦明（２２歳　Ｂ型）
日本電子計算（株）
情報システム事業部システム課
〒135 江東区東陽２－４－２４
03-5690-3231

ポジション・ステートメント：

・品質の方向性

品質に対する考え方は人それぞれ求めるものが違う．
それぞれの立場の人達が品質として何を要求するのか話し合い
明確にしなければ何もスタートできないと思います．

・ドキュメントの整理がされているか．

今まで開発してきた過程が整理されていないと保守・管理していく
上でせっかく築きあげてまものがくずれてしまう可能性がでてくると思います．

・いつまでも古い品質のものを使用していないか．

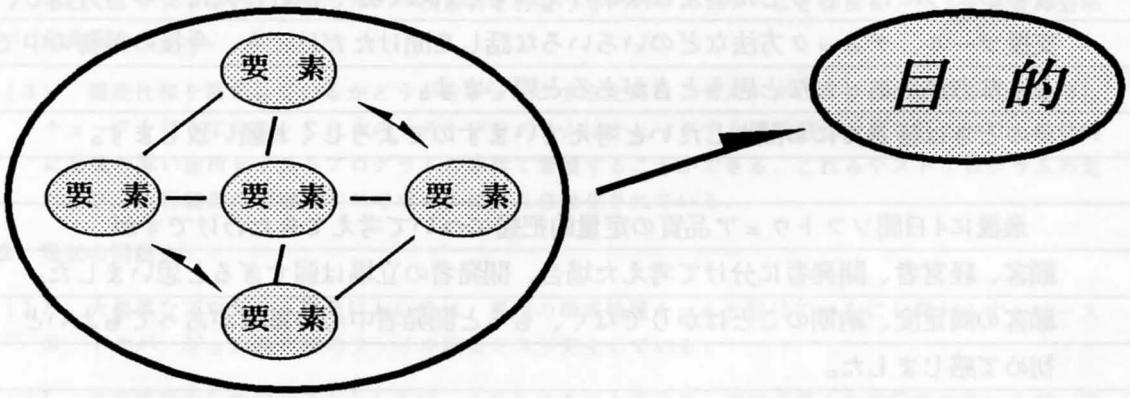
効率，信頼，柔軟，保守などを考え，つねに品質の管理をしなけらばならないと思います．

＜ソフトウェア品質に対する考え方＞

ソフトウェアの品質に対する認識は各個人の立場によってことなる。
したがって品質に対する認識を明確にしておかないと個人により達成しようとする品質の属性もその達成の程度も異なることになる。

　また、問題になるのがソフトウェアごとにその品質を構成する要素が異なることである。
単一機能を有する個々の要素がある相互関係のもとに集約された集合体となり、
ソフトウェア全体としての機能が形成される。
ソフトウェアを構成する複数の機能が組織的に結合しあって目的達成のために活動するわけである。
本来、ソフトウェアの品質とは集合体としてシステムの目的をどれだけ満たすかによって論じられるべきものであり、個々の単体についての品質とは内容も意味も違うものであると考える。

＜システム全体＞



しかし、現状ではソフトウェアの品質と単体の品質が明確に区分されていない。
そのため、ソフトウェアの品質に対する実際的な枠組やモデルがなければソフトウェアに対する品質要求の設定がおこなえない。
そのことがソフトウェアの品質の定量的測定を困難にしている。
開発者の立場からすれば、最終的ソフトウェアの属性を何によって測ればよいのかわからない。
しかし、単体の品質はあくまでも最終的ソフトウェアを実現するための補助でありそれ自体を最終的な品質であるとはいえない。

TITLE	記入年月日	担当者

平成元年9月25日
日本電子計算(株)
情報システム事業部
岩 井 邦 明

'89ワークショップ in 盛岡 に参加して

刺激されました。
皆さんの考え、意見を聞き自分自身もっともっとしっかりしなければと思いました。
はっきり言って"ソフトウェア品質"など考えながら仕事をしたことがないというのが
事実ではないでしょうか。
今回のワークショップに参加して、今後新規業務やメンテナンスが入ってきたら
きっと盛岡で皆さんと話し合ったことが頭の中をかけめぐるのではないかと思います。

いろいろな人と会えた。
ワークショップに参加して全国のいろいろな人と仲良くなれてうれしいかったです。
ソフトウェア品質の定量的把握について皆さんと話し合ったわけですが、
数学のようにはっきりした答えはなかったかもしれないが、他社の人たちの苦労話し、
支援ツール、チェック方法などのいろいろな話しを聞けただけでも、今後の業務の中で
こんな方法があったなと思うときがあると思います。
そのときは皆さんにお聞きしたいと考えていますのでよろしくお願い致します。

最後に4日間ソフトウェア品質の定量的把握について考えてきたわけですが、
顧客、経営者、開発者に分けて考えた場合、開発者の立場は弱すぎると思いました。
顧客の満足度、納期のことばかりでなく、もっと開発者中心の基準があってもよいと
初めて感じました。
この4日間で今後の私の仕事に対する見方がかなり変わったと思います。

今回は意見も少ししか言えませんでしたが、次に参加するときはもっと知識を
蓄えてたくさん自分の考えをもって参加したいと思います。

4日間お世話になりました。

以 上

Position Statement

宮崎　祐（２８歳　Ｂ型）
日本電信電話（株）
ソフトウェア研究所　研究主任
〒108 港区港南１－９－１　ＮＴＴ品川ＴＷＩＮＳ
03-740-5717

経験：
プログラミング言語Ａｄａのコンパイル開発

ポジション・ステートメント：

１．ソフトウェアの開発方法

これまで私が行ってきたソフトウェア開発は，プログラム言語処理系の開発が中心である．品質管理に関しては，以下のようなことを行ってきた．

（１）　プログラムが３００Ｋ以上と大規模であるので，プログラムの構成を管理する専門のチームを作っている．この構成管理チームでは開発チームの各担当者が作成または改良したプログラムのファイル統合を行っている．専門のチームが各モジュールの統合をチェックすることにより，統合時のミスを防ぐような効果がある．

（２）　言語処理系の場合，各言語の文法を満足しているかどうかチェックしなければならずこの機能を満足しているかどうかをチェックするためには，多くのテストプログラムを走行しなければならない．そこで，テスト走行とトラブルの一次解析を行なう専門のテストチームを作成し，テスト走行の効率化を図っている．

（３）　機能仕様を満足しているかどうかをチェックするために，テストプログラムを４０００本使用し，チェックを行っている．これらのテストプログラムはチェックする機能別に分類されており，機能的に品質の悪い箇所をテストプログラムの本数で管理することができる．これらテストプログラムの走行及び，走行結果の管理はすべてツールにより自動化されている．

２．現状の問題点

（１）　大規模なプログラム開発においては，専門の構成管理チームを設けているにも関わらず，ソースコードのバージョン誤り等のファイル統合ミスが発生している．

（２）　品質確認のためのテストとしては，４０００本以上のテストプログラムを走行させているが，実際のユーザが使用するとかなりのバグが検出される．言語処理プログラムの場合，各ユーザで使用方法がまったく様々であるので，開発者側であまり考慮しなかった使い方をされるとバグが発生し易くなる．

（３）　現状の開発方法では，ユーザからみた品質（機能性，信頼性，使用性，効率性）に重点がおかれ，開発者側からみた品質（移植性，保守性）がおろそかになっている．

３．今後の対処

（１）　今回のワークショップのテーマでもある「品質定量化の枠組」を実際の開発においてどこまで取り込むことができ，また，どの程度効果があるのかを見極めて，効果がある場合は，開発に取り込んでいきたい．

平成元年９月２２日
ＮＴＴソフトウェア研究所
宮崎　祐

ポジションペーパ

１．ソフトウェアの開発方法

これまで私が行ってきたソフトウェア開発は、プログラム言語処理系の開発が中心である。品質管理に関しては、次のようなことを行ってきた。

機能仕様を満足しているかどうかをチェックするために、テストプログラムを４０００本使用し、チェックを行っている。これらのテストプログラムはチェックする機能別に分類されており、機能的に品質の悪い箇所をテストプログラムの本数で管理することができる。これらテストプログラムの走行及び、走行結果の管理はすべてツールにより自動化されている。

２．現状の問題点

品質確認のためのテストとしては、４０００本以上のテストプログラムを走行させているが、実際のユーザが使用するとかなりのバグが検出される。言語処理プログラムの場合、各ユーザで使用方法がまったく様々であるので、開発者側であまり考慮しなかった使い方をされるとバグが発生し易くなる。

３．今後の対処

今回のワークショップのテーマでもある「品質定量化の枠組」を実際の開発においてどこまで取り込むことができ、また、どの程度効果があるのかを見極めて、効果がある場合は、開発に取り込んでいきたい。

感想文

①ワークショップへの参加は初めてであったが、様々な立場のひとが集まっており、色々な意見を聞くことができ非常に有益であった。
②今回のテーマは、ソフトウェアの品質ということで、色々な立場のひとが共通的に討論することたテーマであったと思う。
③各立場のひとがそれぞれの意見を持っており、充実した討論をおこなうことができた。討論時間も十分であったと思う。
④最後の発表会は、発表だけで時間の大部分を費やしてしまったが、もう少し全体的な議論の時間があったらよかったと思う。
⑤私のグルーグは、様々な意見、問題点がだされ、品質管理の難しさを改めて認識したが、現状の問題点は、かなり整理することができた。また、人数的にも議論をまとめるのに丁度よい人数（８人）であったと思う。

Position Statement

荒井　憲一（２４歳　A型）
日立ソフトウェアエンジニアリング（株）
第２検査部第１検査課
〒231 横浜市中区尾上町６－８１
045-681-2111 内線(3415)

経験：
銀行（富士銀行，横浜銀行）フロント・エンド・システムの検査作業

ポジション・ステートメント：

ソフトウェア製品の品質把握ということに関する現状の問題点を、自分の担当するシステムを背景にして考えてみた。

１．上流工程での品質把握が不十分のため、後工程まで上流工程の品質が引き継がれてしまっている。

基本設計、詳細設計といった上流工程では、ドキュメント検査を実施している。このときの不良率（単位ページ当たりのドキュメント不良件数）と不良の内容（重要度や影響範囲）によってシステムの大体の品質は経験などを基にして予測できる。しかしこれは、「大体良さそうだ」「悪そうだ」というレベルで、精度は勿論悪い。かなりの経験があっても「何件不良が出る」とは予測出来ない筈である。
また、後工程での摘出バグを分析してみたところ、設計段階で作り込まれた不良が、約５０％程もあった。このことからも、上流工程での品質把握が重要であるといえる。

＜対策＞
上流工程での機能マトリクス（品質マトリクス）の作成による品質要素の明確化と、本マトリクス活用による開発工程一貫した品質管理の実施を検討している。

２．下流工程での残バグ予測は統計的手法を用いて実施しているが、精度は十分でない。また、問題機能と不良の発生状況との関連付けについても不十分である。

残バグの予測手法としては、探針（＊１）、ＦＲＣＳＴ（＊２）が用いられている。
探針とは、全検査項目から一定の割合をランダムにサンプリング（当社では１０％程度が良いとされている）して、製品検査同様の作業を実施、ここで摘出された不良件数と実施項目数から母不良率とその信頼限界（上限値、下限値）を求める。これによって残バグ数を予測し、品質を評価している。
また、ＦＲＣＳＴでは、不良摘出の推移がＧｏｍｐｅｒｚ曲線に近似出来ることから、工程途中までの累積バグ件数をこの曲線に当てはめる。これによって、デバッグ工程での不良摘出状況が収束しているかどうかを判断している。
しかし、上記のような手法を活用しているものの、摘出された不良の内容や修正ステップの量など様々な種類のバグを一律に一件として扱っているだけに、問題も多い。たとえば、探針で求めた残バグ数をクリアしていても、デバッグ工程の終盤で初期的な不良が摘出されたり、修正量の大きなバグが多発していれば、やはり品質が安定しているとは言い難い。

また、開発工程や開発方法などもプロジェクトによって異なる。例えば「単体テスト」→「組合せテスト」→「総合テスト」と行うところを、「組み合わせテスト」→「総合テスト」と行っていたり、開発方法も新規、改造（プログラムの流用を含む）など様々である。そういった違いによってバグの摘出状況は当然異なってくると思われるが、それに合わせた予測手法は確立されていない。

これら「不良の件数」以外の要素を加味してソフトウェアの品質を測るにはどうすれば良いかということが、現状の問題である。

＜対策＞

（1）機能マトリクス作成手法の確立と、本マトリクスをベースとした品質評価の実施。
（本マトリクスには他関連システムや運用／操作面など外部要因の取り込み要）

（2）開発方法、開発工程（期間含む）に合わせた予測手法の検討。

＊1　探針：抜取による先取り評価

＊2　ＦＲＣＳＴ（Forecasting Program）：累積不良件数予測

THE OTHER SIDE OF POSITION PAPER

～よく分からないこと・分かること～

荒井 雅一

はじめに、ポジションペーパーは前回の物をそのまま使用してください。それ自体には、何の変化もありません。

しかし、品質とか定量的とか把握とか普段検査という仕事をやっていながら今ひとつ馴染みの無かった言葉を４日間も使っていると、なんとなく愛着が沸いてきたというか実感が出てきたというか、そういった変化は出てきました。更に私達のグループでは、フィードバックというこれまた言葉の意味は分かっても実体のよく分からないテーマに無謀にも挑戦してしまいました。今思えばその討論も、よくまとまったのかまとまらなかったのかよく分かりません。ただ、フィードバックとはいつどこで誰がどうやってすれば良いのか、という謎が残ったことが大きな収穫だったのでしょう。・・・・・・グループの皆さん、すいません。グループ討論で、一応結論は出ていても、私にはまだまだ謎なのです。どうも私は何にでも謎を感じてしまう様です。だからこの感想文にも「よく分からない」という言葉が多いのかもしれません。（それもまた謎ですが）

以上が結論ですが、これではなんにも意味になってないと言われるのもなんなので、このワークショップでいろいろ思ったことをひとつづつ書いていきます。

まず、基調講演Ⅰではプログラマの能力測定という一見非人道的な話題に非常に興味を持ちました。だいたい動作測定とか標準時間の設定とかいう様な作業管理は、どうも人間味が無くて嫌いなんですが、これは面白かった。プログラマのチームの能力を、フィードバック情報に着目して測定するなんて、今まで理屈っぽく考えた事のない考え方でした。普段、なんとなく感じている事を理屈っぽく考えるというのは、これが学問なんだなと感じると共に、自分でも何か調べてみようかなという気にさせられました。また、この研究にはまだまだ続きがあるので、フィードバック効果について更に詳しい結果を出していただきたいと思いました。

そして、基調講演Ⅱではこんな事を感じました。ソフトウェアの品質なんて言う分かるのか分からないのかよく分からない物について話をするのだから、よっぽど自分でよく分かっていることを、分かりやすい言葉で伝えないと他人には伝わりません。その点この講演では、「ソフトウェアの品質を定量的に把握する、という事はどんぶり勘定をするということだ」という事がよくわかりました。実際、ソフトウェア品質について我々がよく分かっていることなんて、そんな程度なのかもしれません。「どんぶり勘定」ならよくわかりますから。この「どんぶり勘定」に一生懸命理屈を重ねると、どんどん話は分かりづらくなります。私を含めて参加者の皆さん、どうも話が難しくて、自分のポジションさえうまく説明出来なかった（と僕は感じた）のは、このへんに原因がありそうです。それと、

れは私の不勉強だとは思いますが、この業界には’ＹＯＫＯＭＯＪＩ’が多すぎると思ます。今に始まった事ではありませんが改めて感じました。どうにかならないのでしょか？

グループ討論については最初に述べた通りですが、いろいろな会社の人と話が出来た事（月並みながら）本当によかったと思います。

盛岡から帰って１０日。都会の雑踏にも再び慣れてしまいました。しかし、満員電車にられ、人混みの中を歩き、騒然とした環境で仕事をしていると、なんで盛岡はあんなにが少ないんだろうとか考えてしまいます。やはり盛岡は最高の環境でした。あんないい境の中でこんな難しい仕事はしたくありません。グループ発表では、環境がいい方がいとか言うのもありましたが、それでは脱出先がなくなってしまうではありませんか。

・・・・という、それこそ訳の分からない事を書いてしまった気もしますが、最後に。可は結構萎縮してしまいましたが、このつぎ機会があればそれなりの経験もひっさげてリトライしてみたいと思っています。どうかまたわたしを呼んでくださいませ。

Position Statement

河村　謙介（２９歳　Ｂ型）
ＮＴＴ九州技術開発センター
端末ソフト担当
〒860 熊本市桜町３番１号
096-321-3614

ポジション・ステートメント：

はじめに

私の担当する部門では，ネットワーク（電話回線等）を利用するシステムの開発を行っている．まだ設立されて約３年で部門としての経験も浅めである．

ソフトウェア品質と現状における問題点

・新しい部門ゆえ，手さぐりの状態である
・プログラマの蓄積が少ない
・ドキュメント等の作成において，様式等が確立されていない
（いい事が悪い事か？）
・組織としての問題（配置転換）
・開発期間

そこで，上記の問題を抱えながらどのように品質の定量的把握，品質の評価を行っていけばよりよい製品（ソフトウェア）を作ることができるのか模索している状態である．

・部品化による品質の向上
・環境の整備によるメンテナンス性の向上
（ワークステーション・ネットワークシステムの利用）

等，方法はいろいろ考えられるが，他のみなさんはどのように実践されているのが，討議の中から解決の糸口を見出したい．

ＮＴＴ九州技術開発センタ
端末ソフト担当　河村謙介

私の担当するセクションでは、主にネットワークを利用したＭＳ－ＤＯＳ上の社内向けＡＰを開発している。ＮＴＴ内部における電話交換機ソフト開発や、システム開発部門とは、少し事情が違うことを含みおきされたい。

ソフトウエアの品質をいう場合、通常は、
信頼性はどのくらい高いか
ユーザーの要求をどれだけ満たしているか
誰にでもすぐに使用できるか（操作性）
実行効率はよいか
ドキュメント類は整備されているか
保守は容易であるか
などで表せると思う。

品質管理において、その基準をどこに置くかは、非常に重要な部分である。ユーザー側から見た場合には、信頼性、要求充足度などが重要であろうし、開発する側からは保守性などが重視されるであろう。しかしソフトウェアが論理的であるが故、それらを定量的に表そうとするとむずかしいものがある。人間という曖昧な部品から作り出される物に定量的という物差しはあてづらい。

これらの品質に関する捉え方は、私の所属するセクションでは、現在、全くの開発担当者任せになっている。プロジェクトの完了は、チームリーダーの主観となっているのが現状である。リーダーが機能要求に対するテストを行い満足が行けばそれで良しとなっている。
それは、管理者の怠慢というわけではなく、そこに所属している人（約１０名程、１チーム３～４人）全てが手探りで、これからどうすれば良くなるかを模索している状態なのだ。何を隠そう私たちの組織は、つい３年ほど前に産声をあげたばかりなのである。そういった意味で、まだ体制的に確立した物がなく、データ取りなども行われていないのが現状である。

品質の把握に対する現状の問題点として、様式が確立されていないが故のドキュメント類の不備、他人が書いたドキュメントの見づらさや、開発担当者の技術的レベルの問題、経験不足などが上げられる。
それは、人事異動による担当者の交代とそれに伴う技術の流出も（当社においては２～３年で人が変わる。この場合、ノウハウもセクションに蓄積されずに、人と共に出ていってしまう場合が多い。又、新たに来る人間が、ソフトに対するノウハウを持ってやって来るとは限らない）見逃せないファクターとなっていると思われる。
また、このような状況でプロジェクトを引き継いだ場合、悲惨なものがある。

対策として、

様式の確立、それに伴う各種資料によるテストの実施（テスターは、担当者と違う人が望ましい）
ネットワークシステムの利用
各種ツールの利用と、コーディングスタイルの統一化

現在は、まだチーム自体もそれほど大きくなく、互いがそれぞれ意志疎通できる距離（行程が細分化されていない）にいる。しかし、今後、プロジェクトが大きくなって行けば、現在のようなやり方では、破綻が来るのは目に見えている。

もし、ソフトウェアの品質が定量的に把握できるならば、品質や生産性の向上という点において技術者や、管理者にとってすばらしいことである。

という訳で、うちの課長に、盛岡にその答えを捜しに行って来いと言われた私は、悩める小羊が群れのボスに放り出され、ただ一匹野に放たれたような心境でここに臨んでいるのである。

Position Statement

吉田　徹（２８歳　Ｏ型）
（株）岩手電子計算センター
企画・ソフト開発部ソフト第２課
〒020 盛岡市松尾町１７－８
0196-51-2626 内線(456)

経験：
金型ＣＡＤ，ＬＳＩ論理回路のシミュレーション

ポジション・ステートメント：

ソフトウェアの不良は，その開発工程，各フェーズで発生する種類が異なってくる．従って，おのずとそれぞれのフェーズでのテスト・評価も変わる．例えば，低階層でのテスト・評価は，プログラムのカバレッジを求める事で，数量的な把握が可能であろう．しかし，最終的にソフトウェアの品質は，その外部的な仕様をいかに満足しているか，という事にかかっている．それをどの程度テストできるか？テストの精度を高めるために，テスラーがどの程度仕様を理解しているか？という事になる．つまり，仕様を除々にブレイク・ダウンしていった時の内容をいかに把握でき，項目として洗い出せるかが，品質の管理をどこまでできるかという事になるのではないか？

今まで，バグの発生件数，プログラムのチェック・リストの消化項目数等で，プログラム不良の管理を行ってきた．テストの終了を判定する基準を，

（１）作業時間に対するバグの発生がどの程度収集したか？
（２）プログラムのチェックリストを１００％消化したか？

等によってきたが，これらは開発規模に対して，何件という目標を設定したもので，その精度に対してはあまり問われていない．

プログラムの細部を理解できる範囲は，１人数K Step だと言われている．様々な条件のもとで使われるソフトウェア等はそれらの条件を人手でいかに把握しきれるかが鍵となる．

ソフトウェアの開発の程んど全てを人手で行わなければならない現在，いかに客観的にさぼらず，各々のフェーズで発生する不良をできるだけ多く摘出できるようなテスト設計を行なう必要がある．

吉田　　徹
（株）岩手電子計算センター

プログラミング・ワークショップに参加して．．．

「ソフトウェアの品質」の定義について、あいまいな理解のまま臨んだワークショップでしたが

・当たり前の品質をいかに満足させるか
・魅力的な品質をいかに追求するか

ソフトウェアを製造する人たちが、普段の業務を離れて話し合うことが出来たのは、意義のあることだったと思います。

SEA 1990 － 91 年の主要イベント（実績と予定：1990.8.14 現在）

	***** これまで（1990 年 1 月～ 1990 年 8 月） *****	
1/11-13	第 2 回テクニカル マネジメント ワークショップ	北海道：函館市（参加者 14 名）
1/27	SIGENV-Forum「現場にとってソフトウェア開発環境とは？」	岩手：盛岡市（参加者 30 名）
1/31	SEA Forum「CASE ツールの現状と動向」	東京：機械振興会館（参加者 89 名）
2/1-3	第 2 回 ソフトウェア プロセス ワークショップ	静岡：伊東市（参加者 23 名）
2/27	SEA Forum「CASE ツールの導入と活用」	東京：機械振興会館（参加者 93 名）
3/2	九州支部設立 2 周年記念イベント	熊本：熊本市（参加者 57 名）
3/6-7	春の集中セミナー'90	東京：青年会議所会館（参加者延べ 165 名）
3/21-4/3	第 12 回 ICSE 研修ツアー	Nice（France）（参加者 17 名）
4/13	SEA Forum「ダウンサイジング !?」	東京：機械振興会館（参加者 70 名）
5/15（午後）	SEA Forum「ファジィ !?」	東京：機械振興会館（参加者 72 名）
5/15（夜）	SEA 1990 年度総会	東京：機械振興会館
5/23	CASE Forum in Sendai	宮城：仙台市農協会館（参加者 87 名）
6/5	第 3 回 日中ソフトウェア・シンポジウム	大阪：千里国際情報事業財団（参加者 51 名）
6/6	ソフトウェア シンポジウム '90 併設チュートリアル	京都：京都リサーチパーク（参加者 126 名）
6/7-8	ソフトウェア シンポジウム '90	京都：京都リサーチパーク（参加者 297 名）
7/13	SEA Forum「CASE の哲学を求めて」	東京：青年会議所会館（参加者 68 名）
7/17	CASE Forum in Nagano	長野：ホテル信濃路（参加者 62 名）
	***** これから（1990 年 8 月～ 1991 年 6 月） *****	
8/22-24	2nd International Symposium on Future Software Environment	Boulder(USA)（参加予定約 30 名）
9/5-8	第 8 回 夏のプログラミング ワークショップ（若手の会） 「CASE ツールの実際的応用」	岩手県：盛岡市（参加申込 45 名）
9/19-21	SEA 秋のセミナー ウィーク '90	東京：青年会議所会館（参加者募集中）
9/27	CASE Forum in Nagoya	愛知：名古屋会館（参加者募集中）
10（中旬）	CASE Forum in Kyushu	福岡：北九州市（企画中）
10/25-26	International Software Process Symposium	東京：会場未定（企画中）
10/29-31	6th International Software Process Workshop	北海道：函館市（参加予定 37 名）
11/1-2	第 11 回 ソフトウェア信頼性シンポジウム	大阪：阪大Σホール（発表者募集中）
11（中旬）	CASE Forum in Osaka	大阪：千里国際事業財団（企画中）
11/20-22	第 6 回 実践的開発環境ワークショップ	愛知：名古屋国際会議場（参加者募集中）
12（上旬）	4th SDE Symposium 研修ツアー	Irvine（USA）（企画中）
12（上旬）	第 4 回 教育ワークショップ	韓国にて開催の予定
1	第 3 回テクニカル マネジメント ワークショップ	北海道（予定）
1/31-2/1	第 3 回 ソフトウェア プロセス ワークショップ	静岡：伊東市（予定）
3	SEA 春のセミナー ウィーク '91	東京：青年会議所会館（予定）
6/10-12	ソフトウェア シンポジウム '91 および併設チュートリアル	愛知：名古屋国際会議場（論文募集中）

[1] 東京でのイベントを中心にリストアップしました.

[2] 90 年 9 月以降の SEA Forum は未定です.

ソフトウェア技術者協会（SEA）入会のおすすめ

ソフトウェア技術者協会 (SEA) は，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，それぞれ異なった環境に置かれているソフトウェア技術者または研究者が，そうした社会組織の壁を越えて，各自の経験や技術を自由に交流しあうための「場」として，1985 年 12 月に設立されました．

その主な活動は，機関誌 SEAMAIL の発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約 200 人にすぎなかった会員数もその後飛躍的に増加し，現在，北は北海道から南は沖縄まで，1800 名近くのメンバーを擁するにいたりました．法人賛助会員も約 80 社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

「現在のソフトウェア界における最大の課題は，技術移転の促進である」といわれています．これまでわが国には，そのための適切な社会的メカニズムが欠けていたように思われます．SEA は，そうした欠落を補うべく，これからますます活発な活動を展開して行きたいと考えています．いままで日本にはなかったこの新しいプロフェッショナル・ソサイエティの発展のために，ぜひとも，あなたのお力を貸してください．

個人正会員の入会金は 3,000 円，年会費は 7,000 円（入会時の払込は合計 10,000 円）です．法人賛助会員は，年会費 1 口 50,000 円です（入会金はなし）．会員には，毎月機関誌（A4 判約 30 ページ）が配布されます．また各種のイベントには，すべて会員価格で参加できます（法人賛助会員会社の社員は，会員扱いとなります）．

入会ご希望の方は，下記の申込書に必要事項をご記入の上，郵便または FAX で，事務局までお送り下さい．折り返し，会則，機関誌最新号，会費振込用紙などをお送りします．

申し込み先

〒 160　東京都新宿区四谷 3 - 12 丸正ビル 5F　ソフトウェア技術者協会

TEL 03 - 356 - 1077,　FAX 03 - 356 - 1072

.....

SEA 入会申込書（正会員）90-9

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）

年令 ＿＿才　性別（男　女）　血液型（ A　O　B　AB ）

勤務先名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

所属・役職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先住所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先 TEL：＿＿＿＿＿ - ＿＿＿＿＿ - ＿＿＿＿＿（内線 ＿＿＿＿＿）

勤務先 FAX：＿＿＿＿＿ - ＿＿＿＿＿ - ＿＿＿＿＿

自宅住所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

自宅 TEL：＿＿＿＿＿ - ＿＿＿＿＿ - ＿＿＿＿＿

連絡先（どちらかにチェック）　□勤務先　□自宅

.....

SEA 入会申込書（賛助会員）90-9

会社・団体名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

代表者氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）

連絡担当者：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）

所属・役職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：＿＿＿＿＿ - ＿＿＿＿＿ - ＿＿＿＿＿（内線 ＿＿＿＿＿）　FAX：＿＿＿＿＿ - ＿＿＿＿＿ - ＿＿＿＿＿

申込口数：＿＿＿＿＿口

ソフトウェア技術者協会
〒160 東京都新宿区四谷3-12 丸正ビル5F
TEL.03-356-1077　FAX.03-356-1072