SEAMAIL

Monthly Newsletter from
Software Engineers Association

Volume 2, Number 10-12 October - December 1987

目　次

編集部から		1
マネジメント集中セミナー		2
レビューとウォークスルー	岡田正志／相沢圭一	2
ソフトウェア信頼性の管理	久保宏志／川北秀夫	6
IEEE ソフトウェア・メトリクス標準化試案検討会		11
討論記録		11
IEEE（試案）に基づく事例システムへの適例　検討シミュレーション	松村好高・佐藤貴司	25
IEEE 標準化試案の適用	野村　敏次	39
ソフトウェア品質メトリクスへの過大なる期待	佐藤　隆	42
アプリケーション開発から見たソフトウェア・メトリクスについての個人的感想	臼井　義美	44
ソフトウェア品質メトリクスフレームについての考察	山田　淳	47
第5回夏のプログラミング・ワークショップ		52
全体報告		52
グループA討論記録		56
グループB討論記録		59
グループC１討論記録		64
グループC２討論記録		69
グループD討論記録		73
グループE討論記録		78
ソフトウェア教育に関するワークショップ		82
全体報告		82
ソフトウェア技術者のための新入社員教育の現状と展望	篠崎　直二郎	85
フレキブルな国際企業を目指して	杉本　尚子	90
CD に思うこと	渡邊　雄一	92
キーボード・アレルギーについて思うこと	細野　広水	94
会員の声		96
ソフトウェア・シンポジウム '88 案内		103
10th ICSE 技術視察団案内		107
会員状況		109

ソフトウェア技術者協会（ＳＥＡ）は，ソフトウェア・エンジニアの，ソフトウェア・エンジニアによる，ソフトウェア・エンジニアのための団体であり，これまでに日本になかった新しいタイプのプロフェッショナル・ソサイエティたることを目指して，１９８５年１２月２０日に設立されました．

現在のソフトウェア技術が抱える最大の課題は，ソフトウェア・エンジニアリング研究の最前線（ステイト・オブ・アート）と，その実践状況（ステイト・オブ・プラクティス）との間に横たわる大きなギャップを埋めることだといわれています．ソフトウェア技術の特徴は，他の工学諸分野の技術にくらべて属人性がきわめて強い点にあります．したがって，そうしたテクノロジー・トランスファの成否の鍵は，研究者や技術者が，既存の社会組織の壁を越えて，相互の交流を効果的に行うためのメカニズムが確立できるか否かにかかっています．ＳＥＡは，ソフトウェア・ハウス，計算センタ，システム・ハウス，コンピュータ・メーカ，一般ユーザ，大学，研究所など，さまざまな職場で働く人々が，技術的・人間的交流を行うための自由な＜場＞であることをを目指しています．

ＳＥＡの具体的な活動としては，特定のテーマに関する研究分科会（ＳＩＧ）や地方支部の運営，月刊機関誌（ＳＥＡＭＡＩＬ）の発行，各種のセミナー，ワークショップ，シンポジウムなどのイベントの開催，既存の学会や業界団体の活動への協力，また，さなざまな国際交流の促進等があげられます．

なおＳＥＡは，個人参加を原則とする専門家団体です．その運営は，つねに中立かつ技術オリエンテッドな視点に立って行われ，特定の企業や組織あるいは業界の利益を代表することはありません．

代表幹事：　鈴木弘

常任幹事：　岸田孝一　長井剛一郎　盛田政敏　吉村鉄太郎

幹事：　稲田博　臼井義美　大木幹雄　岡本吉晴　落水浩一郎　柿下尚武　木村高志　久保宏志　熊谷章　斎藤信男　佐藤千明　芝原雄二　杉田義明　田中慎一郎　鳥居宏次　中園順三　西尾出　能登末之　針谷明　藤野晃延　松原友夫　丸尾浩一　水谷時雄　三浦信之　村井進

会計監事：　辻淳二　吉村成弘

常任委員長：　岸田孝一（会誌編集）　盛田政敏（企画総務）　吉村鉄太郎（技術研究）　杉田義明（セミナー・ワークショップ）

分科会世話人　環境分科会(SIGENV)：歌代和正　北村昌人　田中慎一郎

管理分科会(SIGMAN)：相沢圭一　芝原雄二　野々下幸治

教育分科会(SIGEDU)：大浦洋一　杉田義明　中園順三

再利用分科会(SIGREUSE)：青島茂　阿倍正平　村井進

ＡＩ分科会(SIGAI)：安倍昭敬　梅林信之　広川昭八　野辺良一　藤野晃延

ネットワーク分科会(SIGNET)：青島茂　鈴木弘　野中哲

法的保護分科会(SIGSPL)：能登末之

ＣＡＩ分科会(SIGCAI)：大木幹雄　寺嶋裕一　中谷多哉子　中西昌武

ドキュメント分科会(SIGDOC)：田中慎一郎　丸尾浩一

ＣＡＤ分科会(SIGCAD)：柿下尚武

支部世話人　関西支部：臼井義美　盛田政敏

横浜支部：熊谷章　林香　藤野晃延　松下和隆

長野支部：青柳和男　佐藤千明　細野広水

名古屋支部：岩田康　鈴木智　西村亨

ＳＥＡＭＡＩＬ編集グループ：大槻亮人　岸田孝一　佐原伸　芝原雄二　関崎邦夫　田中慎一郎　長井修治　野辺良一　藤野晃延　渡邊雄一

ＳＥＡＭＡＩＬ　Ｖｏｌ．２，　Ｎｏ．１０－１２　　昭和６２年１２月１日発行

編集人　岸田孝一

発行人　ソフトウェア技術者協会（ＳＥＡ）

〒１０２　東京都千代田区隼町２－１２　藤和半蔵門コープビル５０５

印刷所　サンビルト印刷株式会社　　〒１６２　東京都新宿区築地町８番地

定価　１，５００円

編集部から

1. はじめに

昨年末のアンケートで，1月の SEAMAIL を週刊誌にしたいという希望を表明しましたが，惜しくも及びませんでした．その代わりといっては何ですが，この合併号では，その名前に恥じないよう，3冊分の厚さに挑戦しました．

前号では，全ページ1段組みを試みましたが，この号では，1段と2段のミックス，しかも一部分はワープロのコピーを混ぜた形にしてみました．会員アンケートの結果では，紙面の整合性よりも定期発行をという声が圧倒的でしたので，これからもこうした形での編集を積極的にやっていこうと思います．

2. 七夕セミナー

7月7日に行われたマネジメント・セミナーのレポートが，ようやく2セッション分だけできあがりました．秋のセミナー・ウィークのテープには，まだほとんど手がついていませんが，アンケートで募集したボランティアの方々にお手伝いいただいて，順次記録をまとめて行く予定です．

3. ソフトウェア・メトリクス

8月末のソフトウェア・メトリクス・ワークショップについては，すでに報告書が出ていますが，読んでいない方も多いと思いますので，討論記録と何人かの参加者のポジション・ペーパを再録しました．

なお，これに引き続いて10月に大阪で行われた第6回信頼性シンポジウムの報告は，2～3ヶ月後に特集する予定です（時間的順序からいえば，その前に日中シンポジウムのレポートが出るはずです）．

4. 若手の会

第5回夏のプログラミング・ワークショップのレポートを，やはり，報告書からの抜粋・編集という形でお届けします．今年の第6回からは，会場を信州八ヶ岳に移し，スタイルを若干変えることになっていますので，盛岡での最後の若手の会です．ところで，盛岡はよい街ですので，何か別のイベントをという声も（特に中年の方から）多いのですが，どなたか企画を立てていただけませんか？

5. 教育ワークショップ

八ヶ岳の研修会場の下見を兼ねて11月に開かれた第1回教育ワークショップのレポートは，いま印刷中ですが，その中から，全体報告とポジション・ペーパ1編を収録しました．

6. 投稿

会員の方から寄せられたエッセイを硬軟とりまぜて3編のせました．今後，投稿はなるべくそのままコピー製版できるように，ワープロまたはコンピュータ出力でお寄せください．ちなみに，現在編集部手持ちの原稿は2～3編しかありません．みなさんの投稿をお待ちしております．

7. 会員の声・アンケート

アンケートへの御協力ありがとうございました．次号あたりで，何か特定のテーマに関するアンケートを試みたいと考えておりますので，よろしく．

8. シンポジウム & ICSE

ソフトウェア・シンポジウムのプログラムが決まりました．御覧のように，従来と比べて格段に充実した内容になっています．割引期間中の予約申込みをお忘れなく．また，ここ2～3ヶ月入会の伸びが鈍っているようですので，この機会に新入会員の勧誘もよろしく．

ICSE の Advance Program は，すでに別便で届いていると思います．共催団体の1つとして，SEA の会員も大勢活躍します．ツアーにも，すでに十数人の応募が集まっています．アジアでの2度目の開催を成功させるために，奮ってご参加ください．

9. 再見，上海！

長野での環境ワークショップは，中国・上海からのお客様2人を迎えて，成功裡に終わりました．そのさい，上海ソフトウェア・センターとの間でかねてから計画されていた第2回日中ソフトウェア・シンポジウムのスケジュールが，10月4～6日杭州市で行なうことに決まりました．ツアーは，10月3日出発，10日帰国の予定です．

討論テーマ，Call for Paper などの詳細は，次号で．

10. 次号の予告

ともあれ Vol. 2 は，今号でなんとか2ケ月遅れで発行できました． Vol. 3 は，編集部内規に従って，昨年のプロ野球のペナント制覇したチーム・カラーになるのですが，同じ色ではおもしろくないので，装いを新たにする意味でも，若干トーンを変えたいと思います．

Vol.3No.1の内容は，ヒミツです（へんしゅうちょう，それは原稿がないんですよ．「サッツ，サッツ」．さっそく原稿ありがとう）．というわけで，なんとか3月中にはお届けしたいと思います．

SEA マネジメント集中セミナー

レビューとウォースルー

講　師：　岡田　正志（日本電気ソフトウェア）
報告者：　相沢　圭一（ウチダコンピュータシステム）

１．はじめに

ソフトウェアの開発プロジェクトにおいては，ユーザーと開発者の最終成果物に対するイメージ，開発メンバー同志の様ざまなコミュニケーション等のギャップをいかに乗り越えるかが，成功か失敗かの分かれ目になる．メンバの士気を高め，プロジェクトを成功へと導くには，プロジェクト毎にカスタマイズされたチエック・リストを作成し，レビューやウォークスルーを行うことが有効である．

以上が当セッションの主旨であるが，時間枠３時間のうち２時間半にわたり，レビューの動機づけ，効率化，報告および記録の仕方，管理上のポイント，またレビューの効果的運用と諸問題について，ＯＨＰを多用して講義があった．

残りの３０分位で質疑応答がなされたが，このレポートでは，講義の部分をわたしなりにまとめてみた．

なお，テキストとして「レビューとウォークスルー」（ＯＨＰのコピーをまとめたもの）と，参考書として「ソフトウェア技術レビューハンドブック」（ダニエル　Ｐ．フリードマン・ジェラルド　Ｍ．ワインバーグ共著／岡田正志監訳・ＴＢＳ出版会）が配布された．

２．ソフトウェアプロジェクトを成功に導くために

ソフトウェアプロジェクトを成功裡に終結させるには，どのようなことがらが重要で，どのような心構えが必要なのか．講師は次のようにのべている．

「ひとつのものを見る場合，いろんな形で見ることができます（３次元の座標軸上に浮かぶ三角すいの図を示しながら）が，いろいろな観点，いろいろな方法によって見なければいけません．これが今日お話するレビューの本質になっていきます．

ソフトウェア・プロジェクトを考えるとき，そのプロジェクトが成功するためには，当然，技術的なセンスというものが必要ですが，その他に，心理学的なセンス，経営的なセンスというものが必要となってきます．

その他にも必要なものとして，分析力，文章力，プレゼンテーション能力，指導力，観察力，正確さなどがあげられます．つまり，ひとつのものを大局的に見ることが必要となってきます．

そこで，プロジェクトを成功させるということになると，そのプロジェクトを管理するという観点からも，同様な分析力やいろいろな力が必要となってきますし，それに加えて，実際の管理業務として，構成管理，工程管理，品質管理，進捗管理，委員管理といったものが発生してきます．

プロジェクトの成功というものを考えてみると，第１に，メンバーの士気が高揚していないといけない．第２に，専門家としての心構えといったものがなければいけない．第３に，これは参加しているメンバーにとって重要なことですが，自分のやった作業が，ひとつの製品に，あるいはその成果として反映されなければならない．第４に，信頼できる見積とスケジュール，第５に，管理者の役割の認識が必要となります．これは当然なことですが，プロジェクトのメンバーとメンバーのやっていることに対して，管理者や経営者が大きな関心をもっている，ということを示してあげるということが大事なことです．第６に，ひとつひとつの作業に対しては，（plan-do-check-action）というフェーズをとっていきながら正確性を期していくことが必要です．

プロジェクトを自発的に推進していけるというのは，理想的ですが，これにはどういうことが必要かというと，目標として，そのメンバーのスキルと，そのプロジェクトに合ったものが設定されている，役割が明確になっている，経営者や管理者がそのプロジェクトに非常に関心を持っている，ということをメンバーに示すことによって，使命感を高揚させていく，ということです．

また一人一人がリーダ・シップをとっていけるような人材の確保，チームの一員としての自覚を，チームの全員が持っているということも必要となります．メンバーがノウ

ハウ，スキルをもっていて，そういうものを人に伝達することによって，自分自身が向上するし，指導力がついてくるという意味で，技術力もプロジェクトの自発的推進には必要なことです．

レビューとウォークスルーについて説明するに当たっては，こういうバックグランドがあるということを，把握しておいてほしいと思います」

３．レビューの必要性

Boehm の「ソフトウェア・コスト属性－生産性変動範囲（図１）」によると，ソフトウェアの総コストに影響を与える最も大きな要素は，個人／チームの能力である．したがって，この部分を少しでも均一化し，生産性を上げるための手段として，レビューというものが有効となる．

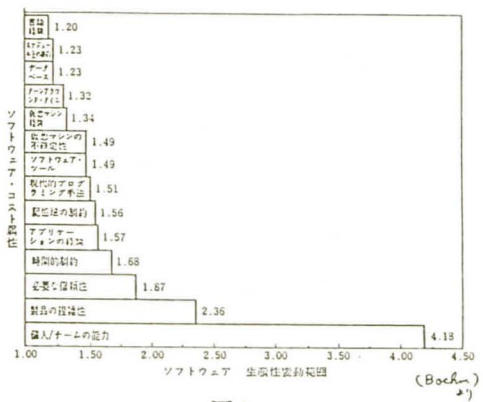


図１

４．レビューの動機づけ

レビューを動機づけ，現場に定着させていくためにはどうしたらよいのだろうか．講師は以下のように説明している．まずは格言を使ってみると．

「人は常に誤りをする」
「開発者は常に誤りをする」
「１００％はありえない」
「肯定と否定の両面から考えよ」
「ピンチのときにこそレビュー」
「代償は後になるほど高くつく」

さらに以下のような点から，レビューの必要性を強調している．

(1) レビューの重要性の認識
- 一般的分析からの認識
  （コンピュータ事故，ユーザの不満）
- 身近な事例からの認識
  （担当プロジェクトの現状分析，要因分析）

(2) レビューの側面
- 技術，管理，心理

(3) How よりも Why

(4) 不足の認識
- 技術不足，検討不足，調査不足，スキル不足

(5) 効果の実績の強調
  （例：テスト／保守コストの７０％削減）

(6) 自発の強制

(7) 小集団活動の利用

(8) ミーティングの活性化
- 失敗を公表できる雰囲気
- 目標の設定
- 責任感
- 管理者の理解
- 個人的摩擦の最小化→言葉使いの問題

(9) セルフ・チェックの認識

またよくある失敗例をあげて強調することもできる．ユーザ側と開発者側では，図２に示すようなギャップが存在する．これを解消するには，プロポーザル，仕様書，プロトタイプ等を徹底的に調査することが必要である．このようなものを調査する場合でも，いろいろな観点，少なくとも２，３の方法論を用いて行うべきであり，レビューが有効である．

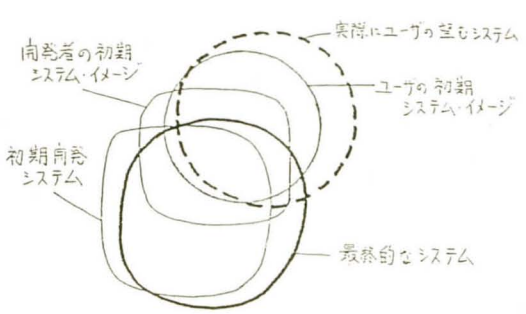


図２

５．レビューの効率化

プロジェクトの規模が大きい場合，工数面で，すべてに

ついて調査を行うことが不可能になる場合も発生する．このようなときには，以下のような点に留意してレビューを実施する必要がある．

- ソフトウェアのサイズを分割して，効率のよいレビューの規模とする．
- 関連する技術，設計技法に精通しておく．
- 技術上の問題点のみを扱う．推測をせずに，まず知っている人（ユーザ含む）に聞いてみることが必要．
- ある大きな欠点を発見したら，同程度の同じような欠点がないかどうかチェックする．
- ひとつのものをレビューするときには，関連資料を作らないで，必要なもの全てをひとつの資料に含んでおく．
- レビュー過程が正当であるか常にチェックする．
- 他のプロジェクトによらず，自分のところではどうするのかというカスタマイズが必要．

6．レビューの報告・記録

レビューが終了したあとは，かならず管理者への報告およびその記録をとる必要があるが，これについては以下の点に留意する必要がある．

- 抽象的でなく具体的に記述する．
- 簡潔性を重視する．
- どの部分について述べているのか明確にする．
- 関係する項目だけを明確にする．
- 項目外のものでも非常に有益なものは別の用紙に書く．
- 問題はその場では解決しないで提起するにとどめる．
- 他人の所為にしない．
- 理解できない不明な点は正直に書いておく．
- 作成経緯には言及しない．

7．管理上のポイント" 奇跡はおこらない"

レビューと管理する上でのポイントは以下のようである．

- 万事を自分でチェックはできない → 責任の明確化が必要．
- 責任は担当範囲全体に及ぶ．
- 誤りは安全な側に
- 技術的な情報に基づく．
- 推測により判断しない．
- 最大の問題点から解決する．
- 自発性を引き出す（本質的にコントロールされたくないという人間の一般的な性質を考慮する）．

そして，「重圧がかかればかかるほど，現在するものより望ましいものが見えてくる」ということを，肝に命じて管理をすることが大切である．また，最近では特に新人類的行動シンドロームといった現象もあるが，十分理解を示し，このような傾向を把握しておくことも重要である．

8．金曜チェック

ここで，文末に金曜チェック・リストを示したので，自分の参加しているプロジェクトにおけるレビューはどうであるのか，yes または no で答えてみてください．

以上１５問中１０問以上 yes であれば，そのプロジェクトはまず成功するはずである．

9．レビューの効果的運用と諸問題

レビューを効果的に運用するためには，以下のようなことに留意する必要がある．

- 成果物をいろいろな観点から見て判断する必要から，プロジェクト内だけでなく他の部門の人を加えて公式に行うと良い．
- レビューには，ウォークスルー，インスペクション，ラウンドロビン等のやり方があるが，観点を変えてみるにはインスペクション，平均的な人が集まり，リーダ的な人がいないときにはラウンドロビンがよいというように目的によって使い分けると有効である．
- レビューの教育という面からは，はじめはバグの少なそうなものから始めるとよい．
- リーダは人間関係を重視して運用する必要がある．レビューのおかげで人間関係が悪くなっては元も子もない．
- レビューによる技術移転ということを第１に考えていかなければならない．
- 小人数で効率的にできるだけ短時間で終了するようにする．
- その問題に対するスペシャリストが参加することが望ましい．
- 電子黒板やディスプレイ等，いろいろな道具を使う．
- レビューの資料はユーザと開発側とのギャップをうめるためにユーザにわかる言葉で書く等，コミュニケー

ションに最大の注意を払う．

・レビューの判定の基準をある程度設定しておく．

・事前にきちんと準備をしてからレビューに望む．

１０．おわりに

レビューやウォークスルーを現場に導入し，さらに定着させる仕事は，とても骨のおれる仕事である．チームのメンバーのみならず，経営者まで含めた全社的な協力が必要となり，これがなかなか困難なのである．にもかかわらず，当たり前の品質よりも魅力的な品質を望むならば，レビューとウォークスルーを有効に活用するより方法はなさそうである．ここをどうするかということについて，今回のセッションからはいろいろなヒントを得ることができた．ポイントはチェックリスト等の一般側を，いかに自分のプロジェクトへカスタマイズするかということだろう．しかし，それはプロジェクト毎に，そのプロジェクトに関わる人が考えなければならず，人材，教育の問題を含め，まだ非常に難しい問題が残されていると思われる．

問い		yes	no
Ａ１	どの工程のどの部分でレビューするか定めてますか？		
Ａ２	レビュー実施方法について推奨したり，具体的にどういう方法にするか定めてますか？		
Ａ３	ソフトウェアの開発技法やテスト技法を考慮したレビューを行っていますか？		
Ａ４	出来上がったシステムがユーザの要求と異なるような場合の防止対策をとっていますか？		
Ａ５	レビュー報告書の型式を定めていますか？		
Ｂ１	レビュー合格の判定基準を定めていますか？		
Ｂ２	ソフトウェア品質の合格基準または出荷基準を定めていますか？		
Ｂ３	プロジェクトが危機に陥らないような具体的なレビュー指針を持っていますか？		
Ｂ４	レビューによって明確にできる問題点とは何かが明確にされていますか？		
Ｂ５	当然の品質から魅力的な品質にするための施策を持っていますか？		
Ｃ１	エラーの状況としての原因に関するデータをとったことがありますか？		
Ｃ２	レビューの実績からチェックリストを作成したり整理したりしていますか？		
Ｃ３	レビューの負荷を軽くするための工夫をしていますか？		
Ｃ４	積極的かつ自発的なレビューにするための教育をしていますか？		
Ｃ５	プロジェクトのスケジュールに問題があるときレビューを最優先に考えますか？		

表１　金曜チェック・リスト

SEA マネージメント集中セミナー

ソフトウェア信頼性の管理

講　師：　久保　宏志（富士通）
報告者：　川北　秀夫（ジェー・エム・エー・システムズ）

１．はじめに

このセッションは，久保氏が実際の仕事や SEA の活動を通して，ほぼ確信するに至った，信頼性に対する考え方を軸に，次の３つの部分に分けて講義がすすめられた．

第１部－自己紹介，用語説明
第２部－本論
・久保氏の信頼性に関する歴史
・テンプレート
・ソフトウェアプロセスの現状
・パラダイム革新の例
・久保氏の抱負
第３部－討論(質疑応答)

このレポートでは，本論と討論を中心に報告したいと思う．なお，講演に先立って，久保氏が断りがきとして述べたことを，本論を考える上で必要と思うので，簡単に次に挙げておく．

(1) 前提

これからの話は，メインフレームのパッケージ・ソフトウェアを中心として考えてきたものであり，アプリケーションのカスタム・ソフトウェアなどの場合とは，若干異なるかも知れない．

(2) テーマ

今日のテーマは，信頼性向上にすぐ役立つ手法，技術ではなく，マネージャはどうあるべきかということについてである．信頼性マネジメントというよりは，むしろ，プロセス・マネジメントについて話す．

(3) 個人として考え方

今日の話は，富士通を代表して話すのではなく，むしろ，個人として，これまで考えたこと，また考えていることを話すことにする．

このレポートでは，便宜上，久保氏が述べたことがらを書く際に「私」を使っているが，これは，久保氏を現している．

２．本論

２．１　久保氏の信頼性に関する歴史

私は，製品計画の仕事に長い間携わっていて，その間は，信頼性の問題にはあまりかかわっていない．信頼性の問題に深くかかわりだしたのは，検査部門の長（昭和５８年から）になってからである．

私が検査部門にいる間に，さまざまなことを知ることができた．その中のいくつかを次にあげる．

・技術がどのように実践されているかその結果として，プロダクトがどのように作られているかを，一歩離れた所から見ることができた．
・品質管理の世界では，　PDCA(注1)のパラダイムが，非常に有効である，ということが確信できた．
・品質評価の技術(注2)が，検査部門のノウハウになっている．しかも非常に水準が高い．

ソフトウェア技術部門では，次のようなことを行い，そして学んだ．

・技術は，作る人と使う人という関係では，絶対にうまくいかない．作られた技術は，使われないでほこりをかぶるだけになる．必要な人が自分で作るのが，作った技術も活用されて一番よい．「技術開発は，自分でやれ」ということを確信するようになった．
・新しいパラダイムを作る現場に居合せることができた(注3)．
・ソフトウェア技術の全体の見取り図を作った．
・技術移転の問題に直面した．

(注1)Plan Do Check Action
(注2)現在のところ，品質評価の手法としては，試験

・現場には，それぞれの特性があり，その特性を反映した技術がある．
・各部門で持っている技術は，かなりの格差がある．
・技術交流は，知的生産のための有効な方法論である（SEAは，そのためにあるのではないか）．
・高度技術委員会に参加して，鳥居先生，岸田さん，寺本さんと知り合うことができ，視野を広げることができた．

Σプロジェクトへ約1年半出向している間では，次のことを経験した．

・言葉がメーカの人に伝わらなくて苦労した．要求仕様をメーカの人になかなか分かってもらえなかった．

２．２　テンプレート

ソフトウェアの信頼性を考えていくうえでは，ソフトウェアがどのように作られているか，そのプロセスを観察することが重要である．プロセスは，まだまだ不完全であり，バグが入り込み，信頼性が問題になる．この不完全なプロセスを革新することにより，ソフトウェアの信頼性を高めることができる．ここでは，プロセス革新の行動を見る際のテンプレート（観点）と，プロセスを観察する際のテンプレートについて述べる．

プロセスを革新する行動を，図１のようなモデルで考える．

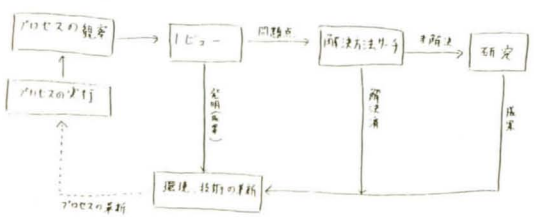


図１　プロセス革新のモデル

ここでは，プロセスを，環境（注4）のうえで，技術を働かせることの繰り返し，というモデルで考える．また，技術は，知的な生産のパラダイムと考える．このパラダイムは，大別すると，実行のパラダイムとコントロールのパラダイムとして考えることができる．

コントロールのパラダイムは，未熟な実行のパラダイムを補うためのもので，実行のパラダイムが完全ならば，本来必要のないパラダイムである．これらの考えから，プロセスを観察するためには，次のポイントをよく見ることが重要である．

・環境
・実行のパラダイム
・コントロールのパラダイム
・人間と技術(各パラダイム)と環境との関係

２．３　ソフトウェア・プロセスの現状

前述のテンプレートを使ってみたソフトウェア・プロセスの現状を参考文献からピックアップして述べる．

(1) 信頼性対策

信頼性対策を行うかどうかは，費用対効果で判断している．これは，実行のパラダイム（特にベリフィケーションの技術）が未熟で，コントロールのパラダイムでカバーしているために，信頼性対策に費用がかかるという状況のためである．

(2) 信頼性のための工程革新，工程定義

信頼性の観点からの工程(プロセス)の革新は，ほとんど行われていない．工程の定義についても，信頼性対策が鮮明に打ち出されていない．例えば，要求をどの程度満たしているか評価するバリデーション（Validation)）と，与えられた仕様を正確に実現しているかどうかを検査するベリフィケーション（Verification)）とは，異なるものだという意識すら明確になっていない．

また，PDCA のパラダイムにおける，目標と実際との対比をする目標管理もあまり行われていないようだ．

(3) ドキュメンテーションに対する信頼性の配慮

ドキュメンテーションについても，信頼性の配慮は，あまりされていないようだ．ドキュメンテーションの標準化は，よく行われている．しかし，これは，レビューが多用されるので，レビューしやすくするために標準化するという程度である．また，ドキュメンテーション支援ツールは，

を行っている．

(注3)後述のパラダイム革新の例としてあげている日野さんの仕事をさす．

(注4) ここでの環境は，開発支援環境のような狭い意味の環境ではなく，プロジェクトを取り巻く環境というような広い意味の環境を指す．

かなり作られているが，これも本来の意味での技術革新とは言いがたい．

ドキュメントは，本来，情報を作り出す，創造する，分析する，加工するなどのプロセスのアウトプットとして出てくるものである．つまり，ドキュメンテーションの信頼性を上げるためには，これらのプロセスのパラダイムを変えていく必要がある．質的な革新が必要なのである．残念ながら，ドキュメンテーション支援ツールには，このような質的な革新はあまりないようである．

(4) 言語の選択

開発言語選択の傾向は，高水準言語指向がはっきりとあらわれている．ただし，どんな高水準言語を使うかという議論は，あまり，なされていないようだ．

高水準言語の中のどんな種類の言語を使うか（例えば，抽象形データタイプのような機能を持った言語とか，オブジェクト指向の言語といった具合），それによって，信頼性にどんな影響があるのか，という議論は，あまりなされていない．

(5) 図形エディタ

図形エディタもよく使われているようだ．しかし，図形エディタを使うことによって，信頼性がどれだけ向上するかというと，あまり大きな期待はできない．手で図形を書く場合と，図形エディタで書く場合では，パラダイムの質的な違いはない．このため，信頼性の点では，あまり寄与しているとはいえない．

(6) 仕様化技術

仕様化技術と信頼性の問題を考える上では，バリデーションとベリフィケーションの二つの観点で考える必要がある．しかし現状では，この技術は，ほとんど何もないといえる．

(7) モジュール化

モジュール化の方法は，さまざまなものが考え出されているが，実際には，機能分割によるモジュール化しか行われていない．

(8) 信頼性における問題点

ソフトウェアの信頼性における問題点として，予測，計測，評価の困難さを挙げる人が多い．このことは，プロセスをコントロールしていないためと考えられる．また，プロセスを革新する新しいパラダイムを作るための手掛かりをつかみあぐねているようである．

２．４　パラダイム革新の例

(1) 日野さんの例

日野さんは，バグの分析からはじめて，現在は，設計言語を作ることを目標にしている．日野さんの足取りを表２に示す．

活動	成果
バグの分析	バグを未然に防ぐ「７つの原理」の導出
証明向き（注5）プログラムの書き方（パラダイム）の考察	WHEN 命令の発明
設計言語の作成	現在活動中

表２

(2) IBM の OS 開発部門の例

IBM の OS 開発部門では，「バグゼロのプロセスを作り出す」という目標を実現するための，プロセス・マネジメントのガイドラインを与えている．そのガイドラインを以下に簡単に述べる．

- 一貫性をもって，継続的に行うこと．
- プロセスは，分割してマネージすること．
- 分割したプロセスは， ETVX（注6）パラダイムでコントロールすること．
- プロセスの実行によって，得られたデータを集めて記録すること．
- 前項のデータを分析して，プロセスの改良に役立てること．
- 改良を記録すること．
- 改良の成果を評価すること

詳細は，参考文献[11]－[18]を参照していただきたい．

２．５　久保氏の抱負

プロセスをよくするためには，どんなことが必要か，また，私は，これからどんなことを実行していこうと考えているかを最後に述べて終わりにします．

（注5）証明向きとは，分かりやすいために，間違いが入り込みにくいということである．

（注6）PDCA パラダイムと同等

プロセスをよくするためには，次のことが必要だと考えている．

・作文能力
・PDCAパラダイムを実行する．
・テスト依存から証明指向への体質改善（そのためには，人間の論理的能力で証明できるようなプログラムを書くパラダイムが必要．例えば，日野さんのWHENなど）．
・統系学を適用するためのプロセス条件の理解．
・ソフトウェア・アーキテクチャ事例の研究を促す．
・プログラミング・パラダイム研究所を設立したい．

3．議論

以下には，当日行われた討論について要約した．討論は，終了時間を大幅に越えて行われるほど熱中したものだったが，ここでは紙面の関係上，筆者が興味深いと思ったものをピックアップして載せた．

問：信頼性と人間との関係については，どのように考えていますか？
答：今回の話をベースとして答えさせていただきたい．人間の要素を信頼性の考えの中に入れると，非常に複雑になるので，私は，あえて人間の要素を入れないで考えている．私は，技術を進歩させること，つまり，環境，パラダイムを進化させることにより，信頼性を高めるという立場で考えている．
問：テストは，実際の使用環境に即したテスト項目を使うのが一番効率がよいと思いますが，検査部門では，実際には使われないようなケースをテストして，バグ出しをしている．このようなことをなくし，効率化するために，テスト項目を開発部門と検査部門とで，共有化できないかということを考えているのですが，これに対して何か問題があるでしょうか．
答：検査のメリット（開発者と別人が検査をする）をどこで生かすかということが重要だと思います．検査本来の目的は，要求の実現度を評価する（バリデーション）ことであり，バグ出しの手伝いではないはずです．開発部門のテストは，ベリフィケーションであり，完全を目差すものです．だから，テスト項目を共有化することは，全く意味のないことです．沼津の検査部門では，デバックの手伝いからはじめ，しだいに検査で使っていたテスト・データ等は，開発部門に渡すようになった（ベリフィケーションの部分）．このような仕事のシフトをやりながら，評価に徹するようにした．
問：　検査部門が，ソフトウェアの仕様の理解にはいる時期はいつごろですか？
答：　富士通沼津の場合は，マニュアルが完成してから，それを見て理解する．完全にユーザと同じ立場として入る．機能仕様書のような，開発のためのドキュメントは見ない．
問：マニュアルは誰が作るのですか？
答：通常は，開発グループが作る．
問：標準化推進のための方法論やアドバイスは，何かないでしょうか？
答：結果としてどうなったかという意味でお答えします．標準化は，次の２つの意味で行われています．
・蓄積したデータに意味を持たせるため
・実行のパラダイムの標準化（あまりない．例－構造化プログラミング） IEEEでは，品質評価プロセスのパラダイムの標準化を行っている．
問：　プロセスをコントロール，実行するためのパラダイムは，どのように使うのか？
答：コントロールのパラダイムは，マネージャが使い，実行するパラダイムは，メンバーが使う．　IBMの標準化の例も実際には，コントロールの部分ではないかと思う．実行するパラダイムを蓄積するための環境を標準化している．環境の中のツールをドライブする部分などで，すべてコントロールの部分です．実行のパラダイムをどんどん新しいものに取り替えても困らないように，マネジメントのパラダイムを標準化していると考えられる．
問：テスト依存から証明指向への移行は，そう簡単にはいかないのではないか？
答：私のいた沼津では，テストの無力さは，かなり広く認識されていた．そのためテストの代わりにレビューを多用している．単価コスト当たりのバグ発見数は，圧倒的にレビューの方が多い．これをもう一歩進められないかということです．
問：開発者によるテストは，自分自身の思い込みや甘えなどのために，限界があるのではないか？
答：それを認めてしまっては，よくはならない．開発者の技が低いのだから，技を磨かせないといけない．第三者によるメリットは，そんな所に生かさなくてもいいのではないか．
問：製品によって，障害修正の多いものと少ないものがある．この原因は，人にあるのか，その製品を作るプロセス

にあるのか，また，うまくいっていない理由は，マネージャにあるのか，パラダイムにあるのか，久保さんは，どのように考えていますか？

答：検査では，製品による差別はしていない．それにもかかわらず差がでるのは，次のようなことが考えられる．

- テストデータのサンプリングがまずい．
- 検査の合否判定は，自ずから行ったテスト等のデータによる場合と，開発のプロセスのデータを集めて，それに基づく場合とがある．その差が悪い意味でででてきている．検査部が，顧客の代弁をしえていない．
- 検査部にいると，一歩離れた立場から見られるので，いろいろなことがよく見える．トラブルを起こしそうだと思った課長は，だいたい起こしてしまう．トラブルを起こす課長は，マネジメント力が弱い．データを取っていないため，ヒアリングをかけたときに答えられない．つまり，PDCAのパラダイムをまわしえていない．
- 検査部門は，悪い物を出荷しないのではなく，良い物を出荷するという立場の部門なので，いろいろなアクションをとり，なんとか出荷レベルに持ち上げるが，やはり出る．プロセスをみていて悪いやつは，どうしても外に出てしまう．

問： 他人が作ったプログラムの検査（消極的な検査，テスト）をすると，作った本人でないので，よく分からないため，盲判を押すような検査になりがちだと思います．この点を改善させるためのプロセス・コントロールの方法について，いいアイディアなど何かありませんでしょうか？

答：盲判を押してもいい所に問題がある．検査部が合格にした物に対してユーザー評価が悪い場合は，検査部の責任として，責任追及を行う検査部に出荷権限を与えているのだから，その位までしなくてはいけない．このような事件を起こしながら，そのフィードバックをかけることにより，技術を向上していくようにする．くやしい思いをさせて，プロセスの革新を考えさせる．マネジャの仕事は，このような問題（テーマ）を作ってあげることです．盲判を押しても良いということは，ユーザが甘いからで，もっと口うるさい人を付ける方がよい．

検査部は，１０数年かけて，開発部門と対等な地位を獲した．いろいろな問題を起こしながら，技術を向上していき，今ではものすごい技術を持っている．また，上の人の意識も重要で，開発部門と検査部門では，検査部門の意見をよく聞くなどの配慮が必要です．

問： パラダイムの革新を行う際に，技術の面が関与する割合と，人間的な面が関与する割合では，どちらが高いと思いますか？

答：技術を変えるというのと，人間を変えるというのが，同時進行的に進むと考えています．日野さんたちの場合は，皆，楽しみながら，技術を高めていたようです．

問： モラルを高めることにより，問題意識を持つようになり，技術の向上が進む，とは考えられないでしょうか？

答： 人間の問題は，意図的にはずしているのです．人間のせいにすると，何でも説明がついてしまう．人とか教育などは，よほど注意しないと成果がでない．問題が具体的ではないから，答らしきものが多く出て，結局何も変化しないことがおおい．それに対して，技術ははっきりしている．確実に実践すれば，それだけの成果が出る．マネジャは，失敗を飛躍のバネとして利用すべきだ．そして，失敗しても困らないように，マネジャが手をうっておく．つまり，コントロールの力を働かせる．

問： 個人のスキルアップは，今の環境では，限界があるのではないか．そのような環境で，なおかつ，信頼性を向上させるためのひとつのアプローチとして，設計パターンを定式化する手法があればよいと思うが，このようなアプローチは，行われているのでしょうか？

答： あると思います．標準化などが，その例になるでしょう．プロセスの作業幅を小さく切っていけば，品質はよくなると思います．ただし，邪道だと思います．品質はよくなるが，何かむなしい．できれば，人間の扱える世界にまで，問題を単純化するのがよい．それが，パラダイムの進歩だと思う．コントロールは必要だが，できるだけ少なくて済むようにしていきたい．

IEEE ソフトウェア・メトリクス標準化試案検討会

討 論 記 録

この検討会（ワークショップ）は，高信頼性研究会の主催によって，１９８７年８月末に，機械振興協会伊豆研修所で開催された．参加者は２９名．以下の討論記録は，すでに刊行されているレポートから抜粋し，編集部で若干手を加えたものである．

0. オープニング（８月２７日午後）

0.1 開会の挨拶

高信頼性研究会の主宰者である鳥居宏次氏（大阪大学）から，今回のワークショップ開催の経緯や目的について説明があった．要旨は以下の通り：

高信頼性ソフトウェア研究会は，私が主査をしている通産省工業技術院関係のある委員会のメンバーを中心に，ソフトウェア工学の実践的な問題に関して討論をする場として，これまで運営されてきた．具体的には，ソフトウェア信頼性に関するインフォーマルなシンポジウムを，すでにこれまでおよそ２年のあいだに，５回開いてきた．その意味では，かなり活動のボルテージは高いと自負している．

最初の２回は，幹事役の浦野さん（KDD）のお世話になって，東京で開催したが，昨年の秋からは，SEA 関西支部の臼井さん（JIP）の御協力を得て，私の本拠地である大阪で開催するようになった．

母体の委員会では，ソフトウェアの高信頼化に関する何らかの標準化案を策定すべく，現在作業中である．たまたま，この春第９回 ICSE のパーティで，岸田さん（SRA）の御紹介で，Naval Postgraduate School の N.Schneidwind 教授に会い，IEEE のワーキング・グループで，ソフトウェア品質メトリクスの標準化試案を検討中であることを知った．その後，手紙のやりとりの結果手に入ったのが，みなさんのお手元にある Draft Version-12 である（文末の表１にその目次を示す）．

この Draft は，まだ，文字通りの試案であり，未完成の状態にある．しかし，ソフトウェア品質の定量化という，きわめて間口の広い問題に対して，１つの統一的な概念上のフレームワークを与えることにより，さまざまに相異なる環境で仕事をしているソフトウェア技術者・研究者が，お互いの経験や知識を交流するための「仕掛け」を作ろうという意図は，高く評価できると思う．

この試案が提示している概念上のフレームワークは，図１に示すように，Factor - Criteria - Metrics という階層構造のモデルであり，いわゆる McCall のモデルに準拠している．付録 B に Factor および Criteria の例が出ているが，それらの間の関係はまだ明らかにされておらず，Metrics の例示もまだ欠けている．

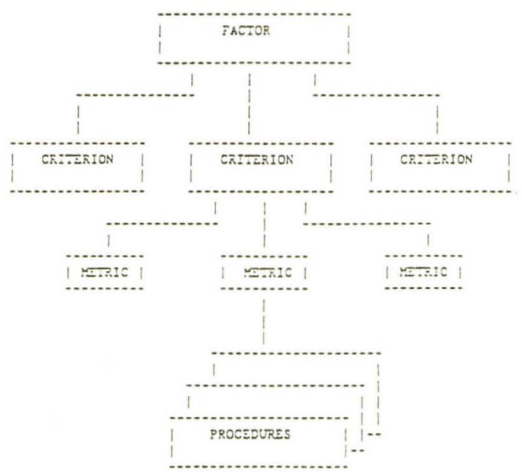


図１ IEEE 試案のフレームワーク

本文第 5 章には，実際のプロジェクトで，品質に関する要求を Establish（確立または設定）し，Metrics の計測を Implement（具体化または実践）し，最後にその有効性を Validate（検証または確認）するための，標準的な手続き（ガイドライン）が書かれているが，記述はまだ骨組みだけしかできあがっておらず，このままではとてもすぐ現場では使えそうにない．

付録 A に載せられた適用例も，まだ未完成だし，とりあげられた例題が DoD 関係のアプリケーションという，われわれにとってはなじみの薄いものが１つしかのっていない．

しかし，そうした不満ばかり述べていても始まらない．

こうした試案（Draft Proposal）に対する正しい態度は，その基本線を十分に理解し，内容をより現実的なものにするための貢献をなすべく，自分たちの方から積極的にアプローチすることであろう．私の委員会でも，この試案の枠組みをできるだけ活用して，ソフトウェア信頼性の定量的管理のための（日本的な？）ガイドライン作りを目指して行きたいと考えている．

ソフトウェアの信頼性という言葉は，広く社会のなかで使われているが，そこで話されている内容はまちまちであり，語り手によって話の切り口が異なっている．信頼性に関連するそれぞれの用語や概念について，個々の話の文脈にそって細かく分析してみなければ，お互い同じ土俵で議論ができないような状況にある．

信頼性とは，実際に数値的に計測されて，初めて意味を持つものだと考えられる．しかし，ソフトウェアの世界では，これまで余りそうしたアプローチが真剣に考えられなかったのが現状である．

－ ソフトウェアの信頼性を計測する，あるいは定量化する，ということを考えた場合，われわれは一体何をどうすればよいのか？

－ 現在の IEEE Draft Standard の姿は，現実への適用を考えた場合，非常に不十分であるが，記述の足りない部分をわれわれ自身の想像力で補いながら，それを身のまわりのプロジェクトに実際に当てはめてみたらどうなるのか？

今回のワークショップは，こうした課題をとりあげ，きわめて限定された方々（高信頼性研究会メンバー，前回の大阪信頼性シンポジウム参加者，ＳＥＡ幹事会メンバー）だけに声を掛けて，こうしてお集まりいただいた．あまり形式ばらず，つとめて本音だけを語りあうという，このグループの伝統にのっとって，議論をすすめて行きたい．

0.2 基調講演

東基衛氏（早稲田大学）が， ISO/TC97/SC7 における標準化活動について説明された．その要旨は，以下の通り：

(1) ISO/TC97/SC7 について

ISO とは，御承知のように，国際標準化機構であり，TC97/SC7 は，ソフトウェアの開発，システム・ドキュメントの標準化を扱っている分科会．そこには，いくつかの作業部会があるが，そのうちの WG3（ソフトウェア・デザインを担当するグループ）で，いま，ソフトウェア品質特性の標準化について，作業中である．

(2) 標準化のプロセス

ISO で国際標準が決まるまでのプロセスは，およそ次の通りである：

標準化作業項目案の提出
↓
投票による採否の決定
↓
Working Paper の作成（提案国）
↓
WG での合意後，SC でさらに審議
↓
Draft Proposal
↓
TC で討論，採否の投票
↓
Draft International Standard
↓
ISO の場で最終票決
↓
International Standard

ただし，標準化案としては否決されたものでも，内容が重要と認められたものは，Technical Document として出版されることもある．

(3) Quality Characteristics の標準化

現在，Criteria for Evaluation of Software が，第２次DPの段階で審議されている．これは，ソフトウェアの品質を，トップレベルで：

- Functionality
- Reliablity
- Ease of Use
- Efficiency
- Maitainability
- Portability

という６つの特性（Characteristics）に絞り込み，さらに下位の特性を Sub-Characteristics として，これらを補足する形で構造化しようというものである．

これは，もともと日本電気の SPEC-DOC をもとに考えられたもので，さらに SPEC-DOC 自体は，McCall のモデルを土台にしている．その意味では，IEEE 試案と根は同じだといえる．ただし，ISO での標準化は，どちらかといえばユーザの視点からアプローチしているのに対し，

IEEE の試案は，明らかに開発者の視点に立っている．

ISO で考えているソフトウェア評価のプロセスのモデルは，図2の通りである． IEEE でいう Factor という用語は，Characteristic に置き換えられている．

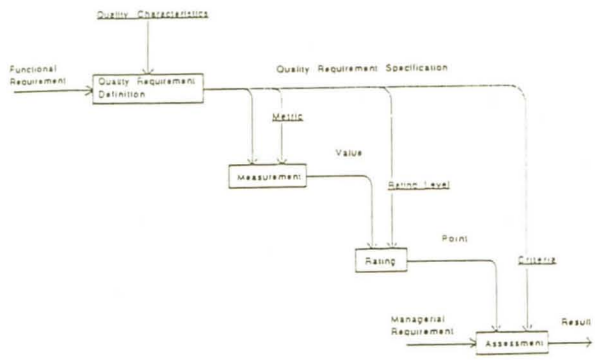


図2 ISO の評価プロセス・モデル

標準として扱うのは，Characteristics のみで，Rating Level 以降は，プロジェクトごとに変化するものと考えている．メトリックスについては，特に標準は設定しないが，テクニカル・レポートの形で，世の中で使われている実用的な指標を収集していきたいと考えている．

現在のところ，トップレベルのCharacteristicsについては，パリ会議（1987 年 6 月でほぼ合意がとれた．第2レベルのSub-Characteristicについては，日本の改訂案と，日本の第1提案に対するドイツの修正案の2つが持ち込まれ，並列に審議されている．合意は，次の国際会議以降になる見込み．

(4) IEEE 試案の状況について

本ドラフトは，IEEE の Software Engineering Standard Sub-commiteeで作業中である．まだ，投票の段階には，いたっていない．IEEE Standard のうちいくつかはANSI（アメリカ国内標準）となるが，ドラフト作成の段階から，ISO とはおたがいに連絡を取り合っている．

［質問］ 品質要因のうち， Availablity が Qualityから外されたのはなぜか？ ユーザから見た場合，ReliabilityはAvailabilityにつきるという見方もできると思うのだが．

［回答］ Availability は Functionality に含まれると考えられる．現在の ISO 案の狙いは，品質を構成する要因をはっきりさせることにより，いろいろな解釈の余地を残しながらも，標準としてとりあえず使えるものを定義しようということである．したがって，さまざまな意見を持つ人がいることを前提としつつ， Mass Market における消費者保護や，国際間でのソフトウェアの売買・発注・受注契約において，品質に関する事項がある場合の手助けとして．有効に使えるような標準作りをめざしている．

1. セッション1（8月27日夜）

チェアマン：春原 猛（三菱電機）
発表者 ：松本健一（大阪大学）
野村敏次（JIP）

1.1 松本健一（大阪大学）

IEEE 標準化試案の日本語訳作成をしての疑問点や，試案第 5章の Mesurement Methodology と，阪大での「エラー寿命に基づくプログラマーの能力に関するメトリックスの適用実験」の比較について，発表した．

(1) 日本語訳についての疑問点

・Aquisition とは何か？

－ 「購買」の意味であり，ソフトウェアの発注あるいは購入作業を意味する． したがって，社内開発の場合は使用していない．

－ Aquisition Manager は，欧米では，購入する製品の選択・決定権，さらには購入するか，自社開発でまかなうかを決定する権限まで与えられているのが普通であり，日本におけるイメージとはかなり異なる．

－ 試案中には，価値基準の要素として Aquisition の観点も含まれている．このように，開発者とは別個の観点も取り込まれていることを念頭におかねばならない．

－政府・軍関係では，「調達」という言葉もある．

・ Establish の訳語は「設定」か「確率」か？

－ 各プロジェクトの，あるいはその組織の特性を考慮して，システムの開発に当たり，最初に品質要求を設定するという意味では，「設定」であるが，次のプロジェクトでも同じ課程を繰り返すことにより，ついにはこの標準案の究極の目的である有用なメトリックスを確立するという意味では「確立」となる．

(2) 阪大での実験との比較

品質要求の設定（Establishing）に関する作業項目はない．学生実験という特殊性のため，コストについてはほとんど考慮する必要がなかった．むしろ，データ収集によって被験者の作業負荷が増大した（被験者への負荷も換算により，コストとしてとらえられるか？）．収集したデータが膨大になったため，保存・分析に苦労した．自分が提案したメトリックスの証明が難しい．

(3) 試案に対する感想・問題提起

全体的に抽象的すぎる．もっと具体的にしようとすると，対象分野，開発環境，開発プロセスを限定せざるをえなくなるのではないか？　分野にかかわりなくスタンダードとなりうるのは，どのレベルまでか？

被験者に対する負荷を考えるべきだ．コストへの換算をどうするか？　「余分なことはしたくない」というプログラマの心理をどう扱うか？

データを利用しやすく保存する方法を考える必要がある．データ量が膨大にならずに効率よく保存する方法はないか？　収集したデータを後日どのように用いるか？

メトリックスの妥当性を検証する手順についての指摘が必要である．

1.2 野村敏次（JIP）

試案の第５章にしたがってシミュレーションを行ったが，途中で挫折してしまった．その挫折の理由を分析しながら，試案に対する問題点を指摘する．

(1) 相互関係

Factor, Criteria, Metrics のあいだの相互関係がはっきりしない．これらの各項目は，それぞれの分野，環境からの要求により設定されるので，一様には決まらない．

マネージャのレベルからエンジニアのレベルへブレークダウンするためには，直接的なものでないにせよ，何か統一的なガイドラインがなければ，フレームワーク自体がおかしくなるのではないか？　ある程度標準的な相互関係の設定が必要ではないか？

(2) 定性的要因

定性的なものは，主観的要素（そのときの気分）が入り込みやすく，あいまいなものとなる．また，そのようなあいまいな基準がベースなのであれば，計測の結果もどこまで信用度があるか，わからなくなる．

定性的なものを，客観的かつ定量的に評価する方法がないか？　大きなシステムでは，定性的なものでも，関係者間の調整で客観的な意味付けができる．

(3) 計測のコスト

小さなシステムに対してコスト的に見合うか？　誰がコストを負担するのか？試案はシステムのサイズとしてどれくらいのものを想定しているのか？

「すべての開発に適用できる」と試案の前書きには書いてある．しかし，メトリックスのなかは，ある程度の規模以上でないとデータ収集が無意味になりかねないものがあり，メトリックスの選択にあたって，このことを考慮すべきである．

しかし，たとえ１人で開発するシステムでも，発注者はおり，マネージャ，エンジニアの役割は，誰かが担うことになるので，原則的にこの方法論は適用できると思われる．

(4) 定義の問題

Reliability, Maintainability, Testability, Expandability, Portability, Reusability, Responsivness 以外のファクタは，意味はとれるが，イメージがわかない．日本語に直して考えたとき，本来の意味とずれてしまう可能性がある．

1.3 一般的討論

(1) 目的

この試案のフレームワークの目的の一つは，ユーザ，マネージャ，エンジニアの間で，品質に関する議論を行なうための共通のプロトコルを設定することにある．

この試案では，ユーザ，マネージャに対しては優先順位を，マネージャ，エンジニアに対しては分担をそれぞれ割りあてている．なぜならば，それらの人々は各々めちゃくちゃな言葉を話しているため，共通のプロトコルを設定しなければ言葉が通じず，そのための変換のルールが，こうしたフレームワークを作ることの目的の一つだといえる．

しかし，現実的問題としては，ユーザの視点と開発者の視点とのギャップが存在していて，両者の視点の対応付けは，なかなかむずかしい．開発／購買という点でも，視点のギャップはある．

(2) 対象

このフレームワークは，どちらかというとマネージャよりであり，エンジニア向けの考慮が不足している．

ユーザ（マネージャでない）がどういうものを要求するかが Factor であり，それに対して，開発側が代用特性（Criteria，努力目標）を考え，Metrics　を設定するというふうに，善意に読み変えることもできる．

試案は全体的にトップダウン方式であり，ユーザ，マネ

ージャの各層に対して品質要求をいかに伝えるかということに力点を置いているが，それは，フレーム（プロトコルの変換）のルールに何が必要であるかを定義することこそが重要だと考えているからだろう．

そのあたりの枠組みが確定する以前に，Metrics の詳細について議論するのはあまり意味がない．そういう意味で，たしかに，エンジニアの立場での考慮が不足しているといえるかも知れない．

(3) Factor の位置づけ

ISO の Characteristics との相違は？

Factor はユーザレベルに近い概念ではあるが，ユーザレベルの用語ではなく，あくまで開発者サイドの用語である．Factor の上層として，さらにユーザレベルの用語があると考えられる．

ユーザの要求というものは，いわせておけばきりがないものであって，それをそのまま額面通り受け取るのは危険である．

ISO はユーザの視点から評価できる基準を決めようとしている．ISO の Characteristics は Factor に相当するが，Sub-Characteristicsは Criteria というよりは２次的な Factor である．

(4) 日本的 Q/C

試案は日本的 Q/Cに見られるような配慮が抜けている．日本的 Q/C では，エンジニアといえどもマネージャ，あるいはユーザの立場に立って考えることも必要であるし，それが動機づけにもなる．

このフレームワークは，ユーザ，マネージャ，エンジニアの各層を完全に分離しようとしている．こうした構造が日本的経営風土に合うかどうかは，問題である．

骨組みとしては，品質に対するアプローチが異なる点を除けば，品質機能展開と相通じるところがある．

(5) フレームワーク

Factor, Criteria, Metrics の３層分けをとるか？ Factor, Metrics の２層分けをとるか？ トップダウン，ボトムアップのいずれの手法をとるかによって，階層は変わって来る．ボトムアップの手法の場合，階層が多い方が実現しやすいと考えられる． この場合は，３層の方が妥当であろう．

制御系に近いソフトウェアの場合のように，特定の Factor, Metrics, Criteria の組み合わせがうまく構成できるものもあるが，事務系のソフトウェアではなかなかすっきりいかない場合が多いように思われる．周辺のシステムの場合，データの比較的単純なハンドリングをするようなプログラムは，評価にあたっての Metrics が用意しにくい．

極端な場合，ユーザの要求はただの一言であり，それをシステム化する階層については，一概に決められるものではない．それすら分からないのに，フレームワークの階層について論議することなど無理な話である．

制御系ソフトウェアの場合，極端にいえば Effciency を重視すればよく，いわば論理と数学の世界で論じられるのに対し，事務系ソフトウェアはそれほど Effecient である必要はなく，むしろ人間の行動や習慣についてのユーザ要求が，数多く入って来る．両者を同じ尺度では計れないであろう．

ハードウェアの延長線上のソフトとして，制御系のソフト（特に OS ）は，工業生産型の手法がとれるが，事務処理アプリケーションはユーザから見ると常に変わって行くため，セールス的なアプローチをとらねばならない．

試案には，もっと人間心理の層からのアプローチが必要なのではないか？

2. セッション２（８月２８日午前）

チェアマン：久保宏志（富士通）
発表者 ：松田茂広（MRI）
：永井光俊（JIP）
：松村好高（SRA）
：中野好則（日本 IBM）

2.1 松田茂広（MRI）

試案の Appendix Fの文献にそって，思考シミュレーションを行った．その際，品質要求は設定ずみであり，その中の１つとして信頼性という Factor および対応する Metrics があるという前提で考えてみた．シミュレーションの結果をもとに，現実のプロジェクトでの問題点を考えてみると，以下の通りである：

(1) 解釈のあいまいさ

品質要求を，AppendixAのような手順で設定する場合，「xx性」という表現は，人により解釈のばらつきがある．また，イメージがまとまらない．一見品質要求を合意したようでも，各自がブレークダウンしたものを持ち寄ると，解釈がまちまちになっていることがある．したがって，解釈のばらつきを小さくするような方法論が望まれる．

(2) マッピング

品質 Factor の解釈が１通りだとすれば，対応する Criteria へのマッピングは誰がやっても同じになるはずであ

るが，実際には「xx性」という形で議論されているために，Criteria の設定があいまいになってしまう．品質 Factor，Criteria を，それぞれどのような目的で，どのように使っていくか，区別がはっきりしない．

品質 Factor と Criteria の相違をはっきりさせることの意味は少ない．両者の区別をはっきりさせることにこだわる必要はない．

(3) 主観の混入

Metrics を計測することは，一見客観的に見えるが，実際にはかなり主観が入り込む余地がある（Metrics そのものの意味を解釈するのにも人間の主観は含まれる）．いったん数値化されると，統計上もっともらしく見えるが，ベースとなるデータはいいかげんな面があり，数字にだまされる可能性がある．

主観の入り込む余地をせばめるために，Varidationの中で方法論を作ろうとしているのではないか？

(4) 妥当性の検証

Metrics の妥当性検証にあたり，品質 Factor(F)の値を求める手順については，ほとんどふれられていないが，何か標準を示す必要があるのではないか？

「F は値をとりうる」という記述がある（5.4 Validation）．F の値は，日本的な発想であるが，Metrics が決まればでてくると思う．F=f（m1, m2, ...）の式の値を求めることは，現状ではかなりむずかしいだろう．

使い方が書いていない点がよくない．値に意味があるのは，Factor を何に使用するか目的がはっきりしている場合であるが，試案ではこの点は述べられていない．Factor という観点では，数値化された Criteria と，それ以外のものを総合して人間が判断することが必要になる．F の式を証明するだけでは不十分かも知れない．

関数の中では評価できない．したがって，分析的でない評価方法と，ボトムアップによる F の総合評価値により検証していく．

Varidation では Criteria がなくなり，Factor と Metrics の関係のみが残る．Factor と Criteria の妥当性も調べるべきではないのか？

品質 Factor(F) の値をMetrics(M) の値を用いずに求められるのでなければ，Metrics(M) の値の妥当性を検証できないのではないか？

Metrics(M) の値を使用しない方法としては，例えば，アンケート調査形式などがあるが，これでも検証は可能である．アンケート形式では，いくつかの Factor についてあいまいさが残ってしまい，結果的に Metrics の方は正しく測定できても，F の精度は低くなってしまう．

厳密な測定ができない場合でも，M の値を正規化することにより，ばらつきをなくし，F の精度を向上させることは可能だ．しかし，関数系は決められたとしても，そのストーリーの中で検証したのでは検証にならない．フィールドでは，結果系で判断するしかなく，プロセスに対しての検証は行えないという問題もある．

F の値を M を使用せずにできるものと，できないものがあるらしい．できるものについては，例を増やし，できないものについては，その理由について，多くの人と一緒に考え，互いに意見を交換することにしよう．

(5) 品質機能展開

試案と比べて，日本的な品質機能展開のほうが，各工程の中での目的がはっきりしているように思われる．試案では何に使用するのか漠然としているため，目的があいまいになってしまっている．手続きの１つのステップとして，Metrics の目的を設定するステップが必要ではないのか？

(6) 視野

試案は，McCall と Boehm の案に則していると考えられる．また，例題も Airborn システムの調達という特殊な事例である．このような形でスタンダードにするのは危険であるし，視野がせまくなりかねない．もっと他の広い視野を持ち，別の分野，別の開発の仕方のシステムに適用した上で，討論を詰めて標準化すべきではないか？

2.2 永井光俊（JIP）

事例によるシミュレーションの結果を踏まえて，およびソフトウェア・ハウスの経営という観点から，ソフトウェアの品質に対する考えを述べる．

(1) 要望

ソフトウェア・ハウスの経営という観点から考えると，あるシステム開発での経験を同様のシステムで生かして，生産性を上げることが必須となるため，拡張性，再利用性，保守性が，評価の尺度として重要視される．

これらは，本来，ユーザ要求からは出てこない Factor であるかも知れない．あくまで，作り手側の事情である．したがって，この標準化試案が完成した暁には，例題として，拡張性を軸としたケーススタディを載せるよう要望したい．

(2) モデル

数学的な記述でとらえられるモデルについては，信頼性

を無理に意識しないでも，直観や，何らかの Metrics を使って，それなりの評価は可能である．

数学的に表しにくいモデルについては，分析表の重要度の欄に全て○がつき，また，Metrics の割り付けが見えてこない場合がある．

さらに，Visual なもの，ユーザ・インターフェースの強い部分は，信頼性，正確さがユーザの目につきやすいため，アンケート等でもシビアなチェックを受けることが予想され，作り手側としては当初から信頼性，正確性に対する意識が強くなる．

Metrics の割り付けがつかないものについても，何らかの形で対応づける必要がある．

(3) 計測機構

外部評価でなく，システムそのものに Metrics を組み込むことを考えた場合，データ処理加工部分くらいでしか，とらえることができない．スタンダードとしては，結果を対象とするためモニタリングについてはここでは議論しない．

(4) ユーザ要求

ユーザ要求が非常にうつろいやすいものであるという事実を認めるとすれば，マネージャがユーザの要求をくみ出すに当たり，しっかりとした基準（判断能力）を持たねばならず，その人たちが Factor の重要度を結論づけるように思われる．付録の例題にしめされているように，まずユーザ要求から入るという方法論に対しては，やや懐疑的である．

実現可能性よりもユーザの要求の方が強く出るケースがある．ユーザ要求が強く出るということは，最初にユーザ要求があることには変わらないが，作り手側で実現可能性，開発コスト等を考えて，最終的に製品としてどう作るべきかという点を吟味するにあたり，それらの可能性を無視した形でユーザ要求が強いられることをいう（単なる仕様変更ではない．）．

Factor は Moving Target なので，それに耐えられる方法論でなければならない．それにプラス（前述したような）作り手側の事情もある．これに答えられる方法論を生み出さなければ，スタンダードとしては，確立されないであろう．

2.3 松村好高（SRA）

ソフトウェア・ハウスの立場にたって，実際に過去に開発した事務アプリケーションを例題として，試案の適用を実験的に行ってみた．

(1) システム・レベルでの要求

事務処理アプリケーションの場合，ハードウェア構成がすでに決まっていることが多く，システムのレベルで要求を考える機会は少ない．このような事情を考慮した例題が欲しい．

(2) 目標設定

Quality Goal の相互関係の判定基準は，対象ソフトウェア・システムごとに異なるように思える．そのため，判定基準の根拠が不明確になってしまう．

Quality Goalの相互関係は，個々のシステムにより変わるものではない．優先順位が変わるというのは，システムごとに設定が違うということではないか？ 人により考え方が変わるのではないか？いや，物の作り方，開発方法論に依存する属性だと考えるべきである．

各 Factor の概念が決まっているか，それをどんな組織がどんな方法論で行うか，により影響される． OS のような制御系のソフトと事務アプリケーションでは，構造そのものが違うため，OS では構造と保守性は反発するが，会計処理ではそれ程でもない．すべてについて統一的な見解など，存在しえない．

～bilityという表現は，かなり Static な感じがする．動的なプロセスの観点からとらえると，たとえば「xxによるTestablity」とか「xxに基づいたTestablity」という形にしなければ，意味がとらえにくい．

一般に，定量化の目標は，定義，方法論によって変わるのであって，対象システムによっては変わらないという考え方もある．

(3) 項目間の関係

Criteria とFactor，Metrics の関係は 1:1 ではなく，1:N ではないか？ というよりは N:N（ネットワーク構造）になるであろう． Metrics 単独では何を意味しているのか分からないため，それがどんな Factor と結び付いているかは，きわめて大切である．

全 Metrics を具体化するのは，現実的にコストがかかりすぎるため，数をもっと絞り込み得るような，有効な基準や方法論が望まれる．できるだけ少ない Metrics で Criteria を導ける方がよい．

(4) 定性的要因

定量化できない Metrics はどう扱えばよいのか？ それらも，何らかの形で定量化していくしかない．

(5) ユーザの理解

ユーザの品質に対する理解は，明確な形では出てこないように思われる．品質そのものについて，また品質を高めるのに必要なコストについて，ユーザの認識と協力を高めるにはどうしたらよいだろうか？

方法論の適用実践にあたっては，ユーザの理解があった方がやりやすい．社会そのものに対して，われわれの考えを受け入れてもらう努力をすべきであるし，また，ユーザの理解が得やすいようなスタンダードの作成が望まれる．

2.4 中野好則（日本 IBM）

IBMの品質管理パッケージSQUALASについての紹介，およびSQUALASとIEEE試案との相違について述べる．

IEEE試案の３つの概念：Factor, Criteria, Metricsは，SQUALAS における３つの概念：ユーザ要求品質，代用特性，管理項目に，ほぼ対応している．

SQUALAS では，数値化できるものとできないものにより，ユーザ要求品質と，ユーザ要求機能に分けて考えている．この点，IEEE の試案とは異なる．

［質問］ ユーザ要求はトータルに来るのであり，品質・機能に分かれて来ることはないのではないか？

［回答］ ユーザ要求を具体化するにあたって，品質と機能を分類するのであって，もともとの要求が分かれているという意味ではない．

数値化できない Factor についても，チェック・リスト的なものを使えば，測ることはできるだろう．代用特性，管理項目の設定については，最終的には，使用者側で決めてもらうことになるが，手助けとして，一般に知られているものをサンプリングして用意してあるので，それを選択して設定することもできる．

3 セッション３（８月２８日午後）

チェアマン：天池 学（沖電気）
発表者 ：山田 淳（東芝）
：佐藤 隆（ATR）

3.1 中間のまとめ 天池 学（沖電気）

チェアマンとして，いままでの討論の結果を「主観的」にまとめると，次のようになる．

IEEE 試案では，Metrics の目的があまり明確になっていないように感じられる．

対象ソフトウェアとしては，要求仕様が確立されたものを中心に考えているようにみえる．したがって，要求仕様からの展開が必要なソフトウェアは，品質機能展開を利用したほうがよいと思われる．

フレームワークについては，２階層で構成すべしという意見もあったが，２階層か３階層かは本質ではない．要は，Criteria をどう位置づけするか，である．

品質要求の設定についていえば，Factor と Criteriaは，明確に分けることはできない． Criteria は，Factor の代用特性（計測可能なもの）と位置づけられる（IBM 案）．Factd間の相関は，製品の属性の違いで生じるのではなく，製造の方法論によるものである．

Metrics の Implementation については，(1) 計測の具体化されたものがない，(2) Factor の選び方が不明，(3) F = f(Mi) であるが，定量化できるものとできないもの（定性的）とがある，といった問題がある．

Metrics の妥当性の確認については，試案で示している方法はたしかに有効であるが，具体性に欠ける． Validation という用語は，妥当性の「確認」と訳すことで合意が得られたように思われる．

3.2 山田淳（東芝）

ソフトウェア品質評価システムを例題として，その構築の際に必要な要素を分析し，机上シミュレーションを行ってみた．その結果，以下の点について，現在の試案には問題があると思われる．

List of Possible Requirement を作成するにあたり，どんな品質 Factor を重視するかは，システムあるいはソフトウェアの仕様がはっきり，またはある程度明らかになってからでないと決められない．しかし，実際には，両者が並行的に進行することが多い．

プライオリティをつけるにあたり，トレードオフや協調関係にある Factorをあらかじめ分析した表があるとよい．しかし，Factor レベルで書いてあるものでは，表自体をどこまで一般化できるか分からないので，Factor を詳細化した表現を使って，トレードオフや，協調関係の理由を示す表も用意する．

３層中２層くらいまでの各項目について設定できないだろうか？

Factor のうち基本的なもの（数個）はあらかじめ押さえておき，それと各プロジェクトまたは分野で，Criteria, Metrics を選択（改造）・変更して使用する．一番上の Factor を同じにしてしまえば，どの分野においても，どれを重要視したのか，またしようとするかのが，明確になってよい．

これまでの議論では，Factor は具体的には決めないという方針であったが，あらかじめ決めておいた方がよいのか？品質要求はあらかじめ決めておくが，Factor の重み付けを調整することにより個別の対処は可能である．この方法は SQUALAS と相通じるところがある．

実現方法としては，Metrics データベースに，あらかじめ Factor を登録しておき，各プロジェクトで必要なものをそこから取ってきて，開発方法論なども考慮した上で，Metrics の作成更新を行う． Criteria ぐらいまで，共通概念として扱い得ないだろうか？

Appendixにある，Factor，Criteria は，ユーザの言葉として，ISO の Characteristics に相当する．開発者の言葉として代用特性がある． Criteria は位置付けるとした場合，代用特性に相当するが，両者は 1:1 の対応関係にはないように思われる．

3.3 佐藤 隆（ATR）

Metrics の目的に的を絞って問題提起する．

システム品質 Metrics を適用するにあたっての（乗り越えるべき）障害は，品質 Factor を決定するにあたり，全関係者を一同に集めることが書かれているが，現実にはむずかしいという点である．

品質を測って何をするのか，目的が不明である．品質に関する目的は，それぞれの立場によって異なると思われる．

作る人
→ 品質を上げる方策をたてるために
管理する人
→ 品質を管理する方策をたてるために
品質を改善する人
→ 品質・工程の改善を工夫するために
使用する人
→ できた物を使用するために
品質を測る人
→ ？

われわれは，何のために，何を測ろうとしているのか？

試案の内容からでは，プロジェクト終了後，その Metrics の効用を他のプロジェクトにフィードバックすることが主目的であるように解釈できる．しかし，エンジニアの立場では，開発工程でリアルタイムに Metrics をとらえ，自分のプロジェクトにフィードバックすることが重要である．その点をもっと強調すべきである．

4 セッション4（8月28日夜）

チェアマン：保田勝通（日立製作所）
発表者 ：小林真二（沖電気）
：佐伯正夫（三菱電機）
：天池 学（沖電気）
：佐藤和夫（日本 IBM）
：小林正男（富士通）

4.1 小林真二（沖電気）

制御系のソフトを例にとって，システム再開に要する時間に関連して，信頼性の Criteria と Metrics を考えてみた．

Factor は，ユーザ要求である．それを，まずユーザの言葉でとらえ，次に開発側の立場で，関連するソフトウェア機能に分割する．

Criteria は，評価基準である．それは，ユーザが評価できる（観測できる）ものでなければならない．したがって，できるだけ数値化できるように努力する必要がある．

そして，Metrics とは，間接的に測定可能な測定データである．

システム再開を例にとって考えてみると，これは，一種の Criteria であるが，同時にまた，Metrics ないしは測定手段でもある．

システム再会時間が Metrics で，ステップ数がデータであろう． Metrics であるためには，データの善し悪しに関しての判定ができなければならない．目標とするステップ数と実際のステップ数の相違を，1つの metrics と考えられないか？

4.2 佐伯正夫（三菱電機）

OS や基本ソフトウェアの信頼性管理について，導入シミュレーションをしてみた．その際に，感じたこと，疑問点などをまとめた．

(1) 要求設定

品質要求の設定とは，製品の性能，用途，同種製品の実績などから，信頼性の対応目標レベルをランキングづけすることと考えられる．その場合に，信頼性については，他の品質要素とあまりトレードオフしたくない．ケースバイケースで，トレードオフが必要な状況もあるのではないか？

信頼性を絶対に高くしたいという要求があることが多いが，コストとの関係が漠然としていて，定量的な目標が設定しにくい．コストと信頼性の関係は，自動的にはつかめ

ないだろう．

(2) ユーザの視点

ユーザの視点からのニーズをとらえるという観点で，品質を考えたとき，常に，Factor または Characteristics に対して値を設定するのか？ 絶対的な値はあるのか？ （無理にでも）与えた方がよいのか？

品質展開の立場からは，各要求に対し，重要度に応じてそれぞれに重み付けを行っている．

絶対値としての値ではなく，相対的なコントラストに意味がある．それら（Factor や Characteristics）が，代用特性または Metrics のレベルに写像・展開されることは，目標値の設定・目標の管理がされる時点での相対的なコントラストとなって反映されることとなる．

品質 Factorの個数は，信頼性，機能，操作性，保守性，機能満足度（機能構成度）の６つ程度にまとめたい．

(3) Metrrics

Metrics の設定にあたっては，重複したものはなるべく削りたいが，少しでも別の角度から品質をとらえようとしているものであれば，残したいという気もする．また，プロセスとプロダクト，出荷前と出荷後の両面をカバーするように心掛ける．

製品固有の価値観（善し悪しのポイント）を考慮すると，品質機能展開などのアプローチにより，汎用ベーシックな特性項目を土台として，その上に製品固有のものを加えるようにしたほうがよい．

部分的な Metrics により，製品全体の品質を一刀両断的に是非を決めてしまうのは問題があるが，しかし，現実的には，計測コスト等を考えると，一刀両断的に評価できる Metrics があったほうがよい．

(4) 評価および妥当性の確認

品質の測定評価については，以下の問題が解決されないと実施は困難である．データベースの支援システムは，どうしても必要である．また，エンジニアに負担のかからないデータの収集方法を考えなければならない．現実的には，かなりうるさがられている．プロセスから自然に導出されるような形で，実現できることが望ましい．

正当性の確認については，例えば出荷後の誤り率などは，１つの Metrics において信憑性が高いと仮定できれば，正の相関が成り立っている他の Metrics においても信憑性が高くなると考えられる．

(5) 要望

ハウツーの部分を具体化したものをスタンダードとしてぜひに含めて欲しい．IEEE 試案では，手続きが４ステップくらいに分かれているが，導入，整備，検証という３ステップと，個々の年度の品質目標管理とか，プロジェクトの品質目標管理のサイクルに沿って行うべきステップとは異なる．そこは分けて書いた方が，ガイドブックとしては分かりやすい．

4.3 天池 学（沖電気）

品質評価基準の実施例（NTT ＋４社で調査，研究した結果）を紹介し，試案との比較を行った．

ここで用いた品質価値基準のフレームワークでは，７つの特性要因（Attribute）があり，Factorに相当している．あらかじめ特性要因の種類は決められている．評価項目は Criteria に似ているが，具体性と言った点でやや異なる．SQUALAS のように，定量的な測定ができるものとできないものに区別してはいない．測定項目は，Metrics に相当する．

このモデルは Tree 構造であり，クロス（N:N）の関係になることはない．このこと自体は，Metrics を導き易くするための措置である．そのために，１つの測定項目に対応する評価項目が複数ある場合は，重み付けをして絞り込んでいるが，このことの無理が結果にでてしまった．

Metrics の判定は，本来定性的なものをも評価できるように 0, 1, 2 の段階評価を行い，ついで評価点を積み上げて総合点を算出する．段階評価の仕方は，各点にそれぞれ与え方の基準を設け，人によるばらつきをなくすような尺度を設定している．総合点として得た値は，結果系と何らかの方法により結び付けられるはずであるが，今回は実施しなかった．

4.4 佐藤和夫（日本ＩＢＭ）

汎用製品の製品保証部門の立場から，試案を検討してみた．

Applicabilityは，”声なき声”の場合も適用できるように，市場の，特に汎用製品に対する品質要求に適用できる形に改善すべきだ（”声なき声”とは，例えば，パソコン・ソフトのユーザがそうであるように，開発者に対して要求を出すなど，直接つながりを持ってはいない集団のこと）．

［質問］ Developerに含まれるのではないか？

［回答］ Project Manager, Aquisition Managerに含まれる．市場調査部門に相当する部門がない会社の場合，そ

のような人が行っている．

［質問］ 受注型以外を考えると，Factor も変わってくるのでは？

［回答］ 誰が評価するかによって，評価項目は変わってくる．これまでの議論では，ユーザの立場から，また開発者の立場からの分類にふれてきたが，上記部門の分類も追加する必要があるのではないか？

Verification に関していえば，安心して使用できる Criteria，Metrics は，ほとんど存在しない．SQUALAS も例外ではない．

妥当性の確認に当たって，Factor, Criteria, Metrics の階層が消えているのはなぜか？ むしろ，Factor と Criteria の相互関係の妥当性を確認をすべきではないか？

SQUALAS では，ユーザ任せになっている． Factor, Criteria, Metrics の階層化は，それらを細分化していく段階では重要であるが，検証にあたっては Metrics により導かれる Factor の値のみが問題になる．

開発部門という立場で考える限りではこの考え方でもよいが，他の立場ではそうでない場合もある．

Factor と Criteria は，組合せ・選択により設定するのか？ 事前に決めておいて，その後重み付けによりFactor 間の調整をするか？ 重み付けをするということは，重み＝0の時，不用なものを除くということになり，これは選択になる．

4.5 小林正男（富士通）

自分の経験したプロジェクト（宇宙開発用ソフトウェア）を振り返ってみて，IEEE 試案の手法を導入した場合の利点について考えてみた．

設計フェーズに適用して，テストの前の信頼性を効率よく行えるようにするのがよいと思われる．われわれのプロジェクトでも，テスト・フェーズで品質の低さが顕在化したが，適切なMetricsを導入していれば回避できたと思う．

システムに非常に多くの機能をつけたため，操作性の低下を招いたが，要求フェーズにおいて，品質における操作性の優先順位を明確にしておけば，ユーザ満足度の高いシステムになったであろう．

設計・コーディング・フェーズで，成果物の進捗を定量的な品質で表すならば，状況を克明にでき，遅れを早い時期にとらえて対処できるであろう．

Metrics を適用するに際して，コスト（時間，人，金）の見積が問題になると思われる．

ソフトハウスへの一括発注した場合は，検収のさいに有効であろう．お互いに，コンセンサスをとらえることが容易になる．

5 クロージング（８月２９日午前）

チェアマン：岸田孝一（SRA）

発表者 ：大場 充（日本 IBM）

：松本健一（大阪大学）

：鳥居宏次（大阪大学）

5.1 大場 充（日本ＩＢＭ）

ソフトウェア工学の立場から，この IEEE Draft Standard について，若干のコメントを加える．また，ソフトウェアとハードウェアとでは，信頼性のとらえ方に相違があってしかるべきだと思う．この点についても，簡単な例を挙げて自分の考えを述べる．

(1) 視野

この試案は，DoD まわりの Contract-based のソフトウェア開発を中心に作成されているが，もっと幅の広いアプリケーション・ドメインを含めるべきだと思う．

(2) 用語の定義

（Appendix Bでは）Correctness, Reliability, Responsiveness のちがいがあまり明確でない． Responsiveness は，Reliabilityに近いように思われる． Evolution Stabilityは，概念的すぎて，該当する Criteria をさがしにくい．概念をもっとブレークダウンすべきである． Throughout Adquency は何のために必要か？ 特殊なアプリケーションに偏りすぎているように感じられる．

(3) ソフトウェア自身の問題

ソフトウェアの信頼性は，ソフトウェア自身に語らせるべきではないだろうか．

Resource Quality, Managerablility は，プロジェクトに関する要因であって，ソフトウェア自身の問題ではない．作り手の問題である． Schedule Adherence も同様に，スケジュールの問題であって，品質の問題ではない．さらに，Change Stability, Work Productivity, Preparedness はプロジェクトに関する Criteria である．

これらはいずれも間接的な影響は与えるかも知れないが，ソフトウェアの本質そのものではない．ソフトウェアの品質は，ソフトウェア自身で測定できる方がよく，プロセスやプロジェクトに関する要因は，ソフトウェアの要因とは別個にした方がよい．ただし，妥当性の確認では，両者を評価する必要がある．

Metrics のレベルでは，プロセスをソフトウェアから分けることはむずかしい．カバレージなどで，テストをする人のパフォーマンスを計る，というような形になる．

(4) ソフトとハード

ソフトウェアにハードウェアと同じ意味での信頼性はあるか？ ハードウェアの場合，初期故障と末期故障（ハード寿命）では故障は増大するが，定常状態では，故障率は一定の確率となる．ソフトウェアの場合，故障は，残っているバグの数の問題であって，これは減る一方である．

しかも，日数が経つにしたがって，故障が発見される可能性は高くなる（指数分布に推移する）．したがって，ソフトウェアの場合，定常状態のハードウェアにおいて MTTF が問題になるような意味で，信頼性を問うことはできない．

また，Reliability は Correctness の関数である．あるいは，Reliability を Factor とすれば，Correctness は Criteria の位置づけとなる．

(5) エラーと故障

エラーと故障とは分けて考えるべきではないか．

エラー現象が発生する確率と信頼性を結び付けるべきではない．厳密には，エラー（Error）とは，Fault をプログラムに入れた人間の行為であり，Fault とは Failre が起こる原因であり，Failure は，ユーザがエラー（Error）だと思う現象のことである．

通常，ソフトウェアでは，Fault をもって故障をカウントしている，また時間も密接に関係してはいるが，独立変数というわけではなく，たとえば，カバレージ，パス数などでも代用できる．

［質問］ 修正（メンテナンス）により構造が悪化することはないか？

［回答］ バグを直す課程で入りこむバグについては，機能の増加をプログラムのライン数の相対的増加という観点で見て行くと，メンテナンス直後はエラーが増加するが，その後は一定になることがわかっている．早い時期に一定にするためには，ソースを完全にすること， Restructuring を行うことが，効果的である．

［質問］ Reliability は Correctness と違うのでは？

［回答］ ReliabilityはFailureがないという定義である．Failureがないとしも，完成品がユーザのイメージと異なる場合に使われるのが Correctness であり，両者を同一の概念と見ることはできない．ユーザの不満を分析すると，たとえば，UserFriendlyでなければUsabilityと関係するし，機能として足りないのであれば，Expandability，Maintenability などと関係するであろう．

［質問］ Accuracy の問題はどうか？

［回答］ この問題は底が深いので，次回の大阪シンポジウムで議論しよう．

6.2 松本健一（大阪大学）

大阪大学では，プログラミング・パフォーマンスの定量化に関する実験を行ってきた．これまでにまとめた成果は，10月のCOMPSAC（東京）で論文発表する．これまでは，学生の演習時のデータだけを使ってきた．今後，同じような分析を企業の現場で得られたデータに対して試みたいので，御協力をお願いしたい．

(1) 収集データ

－ テキスト履歴（プログラムテキストの変更時刻，内容，理由）

－ 時間軸に用いるもの（端末使用時間 etc）

－ プログラム開発過程の概略をつかむためのもの（コマンド使用状況，プログラムサイズの変化）

(2)収集方法

－ ツールによる自動収集

(3)エラー寿命の算出方法

－ 手作業

・ プログラム完成終了後しか算出できない

・ 労力大

－ ツールを用いる

・ 近似値しか算出できない

・ プログラム開発時からリアルタイムで算出できる

・ 労力小

(4) データ収集における希望条件

－ プログラミング環境はＵＮＩＸ．

－ プログラマがデータ収集に協力的・好意的である．

－ ディスク容量に余裕がある．または，マシンの処理能力（速度）に余裕がある．または，マシンに大きな負荷がかかっても文句がでない．

－ 一定期間（ ex １か月）に作成・変更されたプログラムの，すべてのバージョンを貯えられる．

－ プログラム開発終了時には，これに加えて，１人あたり数百～数千KB のメモリーが割当てられる（２０００行のプログラムで 500KB）．

5.3 鳥居宏次（大阪大学）

お疲れさまでした．みなさんのご協力のおかげで，たいへん実り多い討論ができたと思います．

みなさんには，事前にご用意いただいたメモを，ワークショップでの討論の成果を踏まえて Re-Write していただき，それらを編集し，ワークショップでの討論の記録をまとめて，正式のレポートを作成しようと考えております．

以下の各項目のうち１つないしそれ以上についてお書きください：

- ＩＥＥＥ試案の現実のプロジェクトへの適用（机上シミュレーション）：今回提出したメモの Revision または新しい試み
- それによって明らかになった問題点の整理とその改善案（ＩＥＥＥ試案に対する修正または補足提案事項）
- Factor, Criteria, Metrics についての具体例の提案（文献／資料から，または組織／個人の考えにもとづく提案）
- ワークショップでの討論に関する個人的感想

原稿は，９月２１日（月）までに，今回のワークショップの世話人であるSRAの岸田さんあてにお送りください．なお，このレポートは，10 月 23 日（金）～ 24 日（土）に，大阪ガーデンパレスで開催を予定している第６回ソフトウェア信頼性シンポジウムのテキストとして使用します．

それでは，次は，大阪でお会いしましょう．

1.0 SCOPE AND REFERENCE

2.0 DEFINITION AND ACRONYMS

3.0 AUDIENCE AND ORGANIZATION OF STANDARD

4.0 CONCEPT OF SOFTWARE QUALITY METRIC
4.1 Introduction
4.2 Framework
4.3 Purpose
4.4 Applicability

5.0 MEASUREMENT METHODOLOGY
5.1 Establishing Software Quality Requirements For a System
5.1.1 Purpose
5.1.2 Identify a List of Possible Requirements
5.1.3 Order the Requirements
5.1.3.1 Survey All Parties
5.1.3.2 Perform an Engineering Analysis
5.1.3.3 Create a Single Consistent Set of Priorities
5.2 Establishing Quality Metrics
5.2.1 Identify Possible Metrics
5.2.2 Identify Measurment Costs
5.2.2.1 Measurment Costs
5.2.2.2 Software Development Process Change Costs
5.2.2.3 Organization Structure Change Costs
5.2.2.4 Special Equipment/ Training Costs
5.2.3 Identify Measurement Benefits
5.2.4 Commitment to the Measurement Plan
5.3 Implementation of Software Quality Measurments
5.3.1 Define and Document Data Collection Procedure and Responsibilites

5.3.2 Prototype the Measurment Process
5.3.3 Collect and Store Raw Process and Product Measurment Data
5.3.4 Compute the Metric Values From Collected Measurement Data
5.3.5 Apply Metrics Results to Software Products and Processes Evaluation
5.4 Validation of Software Quality Metrics
5.4.1 Puropose of Metrics Validation
5.4.1.1 Predictive Assessment
5.4.1.2 Retirement of Invalid Metrics
5.4.2 Identify the Quality Factors Data Sample
5.4.3 Identify the Metrics Data Sample
5.4.4 Perform a Statistical Analysis
5.4.4.1 Validation Procedure
5.4.4.2 Validity Criteria Identification

APPENDIX A. COMPLETE EXAMPLE OF USE THE METHODOLOGY
A.1 Example from the Dod Evnvironment
A.1.1 Introduction
A.1.2 Establish the Quality Requirements for a System
A.1.2.1 Identify a List of Possible Quality Factors
A.1.2.2 Order the Requirements
A.1.2.3 Prioritize the Set of Quality Factors
A.1.3 Establish Quality Metrics
A.1.4 Implementation of Quality Metrics
A.1.5 Validation of Quality Metrics

APPENDIX B. EXAMPLES OF FACTORS, CRITERIA AND METRICS, AND THE RELATIONSHIP BETWEEN THEM

APPENDIX C. SAMPLE METRICS DESCRIPTIONS

APPENDIX D. EDUCATION FOR USE OF STANDARD

APPENDIX E. RELATIONSHIP WHIT OTHER STANDARDS

APPENDIX F. BIBLIOGRAPHY

表1 Draft Version - 12 目次

IEEE（試案）に基づく事例システムへの適用検討シミュレーション

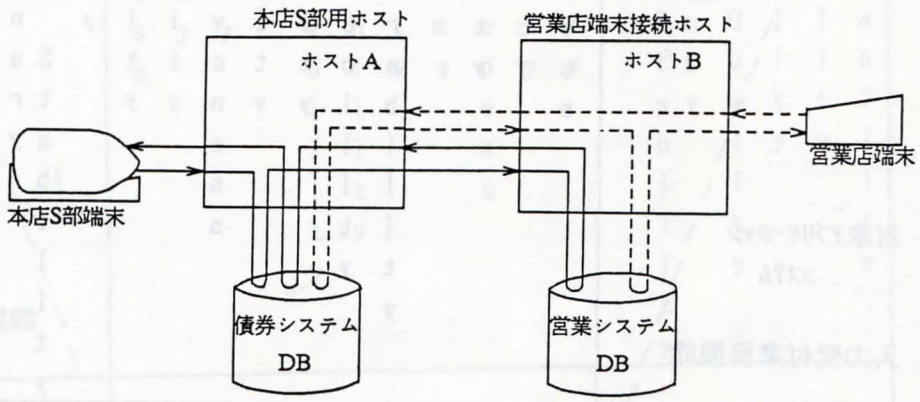
松村好高・佐藤貴司
ソフトウェア・リサーチ・アソシエイツ

1．ＩＥＥＥ（試案）方法論に沿った検討経緯

以下では、過去の開発システムに対し、試案に示された手順に沿ったソフトウエアの品質（信頼性）の定量的管理の導入の試行での検討経緯を述べる。　但し、当システムの実開発過程においては体系だった方法論の適用は行なっていない。

（1）事例対象システム

・ｼｽﾃﾑ概要……金融機関における公共債（国債、地方債、社債等）の売買業務に関する事務処理アプリケーション・システム
・ﾗｲﾌ・ｻｲｸﾙ……８～１２年
・ｺﾝﾋﾟｭｰﾃｨﾝｸﾞ・ｼｽﾃﾑ……ホスト分散型．　下図参照



・ｼｽﾃﾑ・ﾌｧﾝｸｼｮﾝ
ｵﾝﾗｲﾝ入力受付 …Ｓ部端末入力受付，営業店端末入力受付，ＤＢ自動復旧
業後ﾊﾞｯﾁ ……ＤＢ無矛盾ﾁｪｯｸ ，ＤＢ更新及び翌日ＤＢｾｯﾄｱｯﾌﾟ
還元資料作成 ，他ｼｽﾃﾑ・ﾃﾞｰﾀ引渡し及び受入れ
ﾎｽﾄ間ｺﾐｭﾆｹｰｼｮﾝ監視
・開発ﾌﾟﾛｼﾞｪｸﾄの全容
開発規模……当初の予想　大　500Kﾗｲﾝ
開発終了時更に大　約1,000Kﾗｲﾝ
開発期間……着手～最終ｶｯﾄｵｰﾊﾞ　20ｶ月（2回の中間ｶｯﾄｵｰﾊﾞ有）
着手～最終版安定稼働まで　24ｶ月
ｿﾌﾄｳｪｱ・ｼｽﾃﾑ ……複数ﾕｰｻﾞへの導入
の導入

（2）試案の方法論に沿った検討

■Section 5.1 Establishing Software Qaulity Requirements for a System

・5.1.1 Purpose（省略）

・5.1.2 Identify a List of Possible Requirements

前述「ｼｽﾃﾑ･ﾌｧﾝｸｼｮﾝ」および「開発ﾌﾟﾛｼﾞｪｸﾄの全容」を照らし合わせ、いくつかのｼｽﾃﾑ･ﾌｧﾝｸｼｮﾝに対して開発時当初のｼｽﾃﾑ要求項目より想定して作成を試みた。

表－１．品質要求リスト

Software Qaulity Factor ＼ 対象ｱﾌﾟﾘｹｰｼｮﾝｼｽﾃﾑ ＼ 入力受付業務機能	Correctness	Efficiency	Flexibility	Interoperability	Interchangeability	Maintainability	Portability	Reliability	Reusability	Robustness	Testability	Usability	Expandability	Reliability	Maintainability	Extendability	Interchangeability
ｵﾝﾗｲﾝ入力受付（ﾎｽﾄA側処理）																	
・S部端末入力受付	o		o	o		o	o	o	o	o				o	o	o	o
・営業店端末入力受付	o		o	o		o		o	o					o	o	o	o
ﾃﾞｰﾀﾁｪｯｸ，DB更新																	
検索																	
・DB自動復旧	o			o		o		o						o			o
・ﾒﾆｭｰｺﾝﾄﾛｰﾙ	o			o		o	o					o	o			o	
・共通ｻﾌﾞﾙｰﾁﾝ	o					o	o	o		o	o	o	o		o		o
計算，ﾁｪｯｸ																	

Intraoperability……ｼｽﾃﾑを構成するｿﾌﾄｳｪｱ構成要素間の通信性

Expandability ……ｿﾌﾄｳｪｱ機能の拡張可能性

・5.1.3 Order the Requirements

当ｼｽﾃﾑにおけるｼｽﾃﾑ・ﾚﾍﾞﾙ(ﾊｰﾄﾞｳｪｱを含)のQuality Requirementsは、ﾊｰﾄﾞｳｪｱ構成が固定的であったことからｿﾌﾄｳｪｱ・ﾚﾍﾞﾙも同様と考えられる。

以下の表では、当ｿﾌﾄｳｪｱ・ｼｽﾃﾑ の開発当初に既知であり、開発進行上必須かつ実施可能と思われたQuality Factorをﾋﾟｯｸｱｯﾌﾟしてみた。 その選択理由は、表の下に記した通りである。

表－２．当ｼｽﾃﾑで実施可能なSoftware Quality Factor

\ Software \ Qaulity \ Factor	Correctness	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Responsiveness	Reusability	Testability	Usability	Evolution	Resource	Manageability	Expandability	Interoperability
対象ｱﾌﾟﾘｹｰｼｮﾝ ｼｽﾃﾑ \ 入力受付業務機能 \																	
必須かつ実施可能			o			o						o				o	

Correctness およびReliability ……これらをﾋﾟｯｸｱｯﾌﾟしなかった理由は、ｶｯﾄｵｰﾊﾞ時期の遵守のほうの要求度が高く若干のﾊﾞｸﾞ存在が暗に許容されていた為である。

Integrity ……Correctness と同様の理由もあるが、ﾃﾞｰﾀﾁｪｯｸ・ﾛｼﾞｯｸ、DB更新手順、ｴﾗｰ 検出時処理等に関する同一構造化の遵守に解釈してあえてﾋﾟｯｸｱｯﾌﾟしなかった。

Flexibility およびExpandability ……最終ﾘﾘｰｽのｿﾌﾄｳｪｱが中間ﾘﾘｰｽ時のものに大幅な機能拡張を加えることが予め分っていたため、ﾋﾟｯｸｱｯﾌﾟした。

Maintainability ……ﾘﾘｰｽ後の制度改正にともなう修正の必然性のためﾋﾟｯｸｱｯﾌﾟした。

つぎに、表－１にあげたSoftware Quality Factor 個々のGoalの相互関係を当ｼｽﾃﾑ開発に当たってどのような関係にあったかを考えてみた。下記の表はその検討結果である。

表－３．Quality Goalsの相互関係

Quality Goals	Correctness	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Responsiveness	Reusability	Testability	Usability	Evolutionary Stability	Resource Quality	Manageability	Expandability	Intraoperability
Correctness																	
Efficiency	x																
Flexibility		x															
Integrity	o	x	x														
Interoperability	o	x		x													
Maintainability	o	x	o														
Portability		x			o	x											
Reliability	o		o			o											
Responsiveness		o	x	x	x	x	x										
Reusability	o	x	o			o	o	x	x								
Testability	o	x	o	x		o	o	o	x	o							
Usability	o	x	o	o		o		o	x		x						
Evolutionary Stability	o	x	o					o									
Resource Quality		x	x	x		o	o		x		x	x					
Manageability	o		o					o	x		o			o			
Expandability			o	x		o			x	o	x	x			o		
Intraoperability	o		o	x		o				o	x					o	

o…相互助長関係　x…相互排他関係　ﾌﾞﾗﾝｸ…無関係

さらに、Software Quality Factor の重要度決定の手順として、まず表－１にあげたFactor個々に対し重要度づけを行ない、この重要度に対して表－３のQuality Goals の相互関係によって調整を試みた。 下記の表はその検討結果である。

表－４．Software Quality Factor の重要度リスト……Quality Factorの最終Goal

Software Qaulity Factor / Software Unique Function	Correctness	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Reusability	Testability	Usability	Survivability	Reliability	Maintainability	Efficiency	Integrity
オンライン入力受付（ホストA側処理）																
・S部端末入力受付	E	N	E	G	N	E	E→G*	G	E→G*	E	N	N	N	E→G*	E	E
・営業店端末入力受付	E	N	E	G	N	E	N	E	E→G*	N	N	N	N	G	E	E
・DB自動復旧	E	N	N	E	N	E	N	E	N	N	N	N	N	E	N	E
・メニューコントロール	E	N	N	E	N	E	E→G*	E	N	N	N	E	E	N	N	E
・共通サブルーチン	E	N	N	N	N	E	E→G*	E	N	G	G	G	E	N	G	E

E…Excellent　G…Good　N…Not Applicable

前ページの表－４作成にあたり、表－３のQuality Goalの相互関係が排他関係にあるQuality Factorに関しては以下のものを採用した。

			採用 Factor
Flexibility	vs.	Resource Quality	Flexibility
Maitntainability	vs.	Portability	Maintainability
Usability	vs.	Portability	Usability
Expandability	vs.	Responsiveness	Expandability

■Section 5.2 Establishing Quality Metrics

・5.2.1 Identify Possible Metrics

まず、「表－２．当ｼｽﾃﾑで実施可能なSoftware Quality Factor」であげた各Factorに対し「表－４．Software Quality Factor の重要度リスト」を加味して、対象Factorを選定した。 その結果が下記の表である。

表－５．最終選定 Software Quality Factor

\ Software Qaulity Factor	Correctness	Efficiency	Flexibility	Integrity	Interoperability	Maintability	Portability	Reliability	Reusability	Reusability	Testability	Usability	Evolution	Resource Quality	Maintainability	Expandability	Integrity
対象ｱﾌﾟﾘｹｰｼｮﾝ ｼｽﾃﾑ																	
入力受付業務機能																	
必須かつ実施可能			o			o						o				o	
			↓			↓						↓				↓	
最終選定Factor			o			o						x				o	

o…選択　x…非選択

Usability……表－４の重要度は高いがﾒﾆｭｰｺﾝﾄﾛｰﾙに限定されているため、非選択とした。

つぎに、Software Quality Criteria の明確化の手順として、「表－５．最終選定 Software Quality Factor」の個々に対し試案のAppendix-Bに列挙されているQuality Criteriaの該当有無を考えた。 下記の表はその検討結果である。

表－６．Software Quality Criteria リスト

Software Quality Criteria ＼ Software Quality Factor	Flexibility	Maintainability	Expandability
1.Access Auditing	o	o	o
2.Access Control	o	o	o
3.Communications Commonality	o	o	o
4.Communicativeness			
5.Completeness			
6.Consistency	o	o	o
7.Data Commonality	o	o	o
8.Documentation Adequacy	o	o	o
9.Error Tolerance	o	o	
10.Execution Efficiency			
11.Instrumentation			
12.Machine Independence			o
13.Modularity	o	o	o
14.Operability			
15.Response Time Adequacy			
16.Self-Descriptiveness	o	o	o
17.Storage Efficiency			
18.Structural Simplicity	o	o	o
19.Test Adequacy			
20.Throughout Adequacy			
21.Traceability			
22.Fault Freeness	o	o	
23.Change Stability			
24.Work Productivity			
25.Technology Level	o	o	o
26.Standards Adherence	o	o	o
27.Schedule Adherence			
28.Preparedness			

試案によれば測定可能なCriteriaを明確にした後このCriteriaに対するMeasurementsおよびMetrics 設定の手順となっているが、現在の試案ではその過程が明らかになっていない。

従って、Metrics もしくはMetric Valueの導出方法を以下のように想定してみた。

Metric Value = F (score-1,score-2,・・,score-i,・・,score-n)

F………………関数

score-i ………Criteria-i番目の測定Measurement より算出されるある種の点数

また、score-iはつぎの式にて算出されるものとする。

score-i = G (meas-i1,meas-i2,・・,meas-ij,・・,meas-im)

G………………関数

meas-ij ………Criteria-i番目に対するMeasurement・item-jの測定値(下記Measurement Valueと同意語)

ここでは、Metric Value, 関数F, score-i, 関数G は明確にできないが、Criteriaに対するいくつかのMeasurement itemの洗い出しとその相互関係を考えてみた。 その結果を下記の表に記す。

表－7. Software Quality Criteria に対する Measurement item の例

Software Quality Criteria ＼ Measurement items	Acc. Audtng	Acc. Contrl	Comm. Commo.	Consistenc.	Data Commo.	Doc. Adequa.	Err. Tolera.	Mach. Indep.	Modulality	Self Descr	Stru. Simpl.	Fault Free.	Tech. Level	Sch. Adhere.
・DBセグメントのアクセス順序規則 (設計書, プログラム)	o	o								o				o
・DBアクセス手続きの採用規則 (設計書, プログラム)	o	o	o				o	o	o			o		o
・構成および構造規則 (設計書, プログラム)				o		o				o	o	o		o
・項目名称の継承規則 (プログラム)	o			o	o	o				o				o

上記の表におけるmeas-ij即ちCriteria-i番目に対するMeasurement・item-j番目の測定値は、つぎのように定義することができる。

meas-ij = 1 - (# of modules violated rules / # of total modules)

但し、ここでの対象はCriteria-iに対するMeasurement・item-jである。

また、Software Quality Criteria を「ソフトウェア品質(許容)基準」と日本語的に解釈すれば、個々のCriteriaごとにMetrics 値もしくはMeasure 値に対する受容値の設定が必要と思われるが、ここではそこまでの検討は行なっていない。

・5.2.2 Identify Measurement Costs

厳密な検討は行なっていないが、前述のMeasurement items を考えるに当たってはレビューもしくはコードインスペクション にて得られる程度の項目を対象にした。但し、測定実施に関する具体的な体制の検討はここでは行なっていない。

・5.2.3 Identify Measurement Benifits

当システムのリリースは中間リリースのものに大幅なエンハンス を加えて最終バージョン となることが予め分っていたため、試案の試行上では開発者の立場にたちこのようなアプローチ によるSoftware Qualityの明確な認識はエンハンス 時に生産性に大きく寄与するものと想定した。 但し、生産性の寄与率に関する厳密な検討はここでは行なっていない。

・5.2.4 Commitment to the Measurement Plan

未検討

以下の「Section 5.3 5.4」の試案の試行に関しては、当プロジェクトでは具体的に実施していないため未検討である。

■Section 5.3 Implementation of Software Quality Measurements

・5.3.1 Define and Document Data Collection Procedures and Responsibilities
・5.3.2 Prototype the Measurement Process
・5.3.3 Collect and Store Raw Process and Product Measurement Data
・5.3.4 Compute the Metric Values from Collected Measurement Data
・5.3.5 Apply Metrics Result to Software Products and Process Evaluation
・5.3.6 Explore Various Representations of the Metric Results

■Section 5.4 Validation of Software Quality Metrics

・5.4.1 Purpose of Metrics Validation
・5.4.2 Identify the Quality Factors Data Sample
・5.4.3 Identify the Metrics Data Sample
・5.4.4 Perform a Statistical Analysis

２．試案試行による感想・問題点・課題

■試案の試行実施における感想

・試案を読んだ後実際に試行しようとしたとき、まず感じたことはその難解さであった。　その理由としては、１．試案が英語で書かれていること（個人的英語力の不足）、２．個々の「××性」の具体的ｲﾒｰｼﾞの把握に労力を要したこと（個人的技量不足）、３．Appendix-AにおけるQuality Factorの中にAppendix-Bに記載されているQuality Criterion が出現していることによる混乱等、に起因していると思う。

■Quality Factor、Quality Criteriaおよびその対応関係

・試案にはFactorおよびCriteriaの一覧は記載されているが、なにをFactorに、またなにをCriteriaに選択するべきかの基準が明確に書かれていない。　上記３番目の感想のような混乱を避ける意味では、両者の選択基準を明確にしておいた方が良いと思う。

・FactorとCriteriaとの間には一般的な対応関係もしくは個別FactorごとのCriteriaの分類があると思われる(ﾜｰｸｼｮｯﾌﾟにて小林正男氏より資料提供)。　また、個別のｹｰｽ においてはこの対応関係を逸脱することも考えられる。　Standardでは、この対応関係や逸脱時の留意点に関しても記述しておいた方が良いと思う。（但し、本報告書の試行では上記の事柄を考慮に入れずに行なっている。）

■ｼｽﾃﾑ・ﾚﾍﾞﾙからSoftware Qualityの要求を定義する手順について

・一般的でないが、ｿﾌﾄｳｪｱ・ﾊｳｽにとってｼｽﾃﾑ・ﾚﾍﾞﾙからSoftware Qualityの要求を考える機会は少ないと思う。　その理由としては、ｿﾌﾄｳｪｱ・ｼｽﾃﾑ の開発着手時にはﾊｰﾄﾞｳｪｱ構成が定まっていることが多いからである。
特に、ｺﾝﾋﾟｭｰﾀ・ｼｽﾃﾑの既導入ﾕｰｻﾞ(特に大手ﾕｰｻﾞ)における事務処理ｱﾌﾟﾘｹｰｼｮﾝ分野のｿﾌﾄｳｪｱ・ｼｽﾃﾑ では上記のｹｰｽ が多い。

・試案提示手順に沿ってQuality を段階的に定義する過程を、全てのｿﾌﾄｳｪｱ・ｼｽﾃﾑ のｹｰｽ に適用するには負荷的に無理がある。　例えば、現実としては5Kｽﾃｯﾌﾟと500Kｽﾃｯﾌﾟのｼｽﾃﾑを同様に扱うことはできないし、同じ5Kｽﾃｯﾌﾟのｼｽﾃﾑであってもその難易度・重要度は異なるであろう。

・Standardの利用方法は使うものの判断に依存すると思うが、手順のｼｮｰﾄ・ｶｯﾄ方法や省略基準の定義はStandardに含んでいた方が良いと思う。

■Quality Goals 間の相互関係（ﾄﾚｰﾄﾞｵﾌ関係）

・Quality Goals 間の相互関係（ﾄﾚｰﾄﾞｵﾌ関係）を決定する際の判定基準は対象とするｿﾌﾄｳｪｱ･ｼｽﾃﾑ によって異なると思う。
その理由として、当報告書の表－３．ではMaintainability とReusability との関係を相互助長関係(o 印)にしており、それはあるﾌﾟﾛｸﾞﾗﾑ が当ｼｽﾃﾑ内における別の業務操作でのReusability を想定しているためであるが、異なった見方をすればそのﾌﾟﾛｸﾞﾗﾑ の当ｼｽﾃﾑ内におけるMaintainability を上げるためには汎用性を極力避けた結果、他の業務ｼｽﾃﾑでのReusability を下げることが想定されるためである。

・一般的な関係も存在すると思うが、その関係およびその考え方をStandardに記載しておいた方が良いと思う。

■Factors とCriteria、CriteriaとMeasurementsとの関係

・当試案におけるSoftware Quality FactorsとそのCriteria、およびCriteriaとそのMeasurementsもしくはMetric Valueとの関係は、１対１ではなくｍ対ｎと思う(ﾜｰｸｼｮｯﾌﾟにて指摘を受け、１対ｎをｍ対ｎに訂正)。

・試案ではFactors-Criteria-Metricsの関係およびMetric Value導出の考え方に関する記述が不足している。 Standardではこれらに関する言及があるべきと思う。

■Section 5.2 Establishing Quality Metrics について

・Criteriaの位置づけが「Section 4.2 Framework」では明確であったにもかかわらず、当Section では不鮮明になってしまっている。 Standardではもっと明確にしておく必要性を感じる。

・上記と重複するが、StandardではそのAppendixにMetrics の代表例を記載しておいた方が役立つと思う。

■定量化できないようなSoftware Quality Factor について

・当試案においては定量化可能なFactorのみの考え方についての記述と思えるが、定性的もしくは主観の判断を要する定量的Factor，Measurement(５段階評価法等) の扱い方もStandardでは触れておいた方が良いと思う。

■ﾕｰｻﾞのSoftware Qualityに関する認識について

・当試案に沿ったStandardを現場で実践するためには、品質そのものについての認識、品質維持のためのｺｽﾄ およびｽｹｼﾞｭｰﾙに対する認識等関するﾕｰｻﾞの意識改革が今後の課題となる。

■試案のFramework およびMethodology についての感想

・試行では受託ｿﾌﾄｳｪｱ・ｼｽﾃﾑ を対象にして行なってみたが、試案のFramework およびMethodology の適合性は概ねあるように感じた。

・また、試案のFramework およびMethodology は、ｿﾌﾄｳｪｱの要求定義段階においてﾕｰｻﾞから出される機能要求、品質要求を整理して考えるためのﾋﾞｭｰﾎﾟｲﾝﾄ を提供する点においても開発作業に有用であると思う。

3．ワークショップに参加しての感想

ワークショップに参加して、ＩＳＯ作業部会の動向や先輩諸兄の建設的提言を聞けたこと、またそれによって理解度がより鮮明になったことなど私個人にとって有用な機会であったことは言うまでもない。　但し、以下の２点に関してワークショップで言い足りなかったと感じたり、帰ってきてからも何度も疑問を感じたりしているので、繰り返しになるがそのことを以下に述べておきたい。

■Software Quality Factor はどの立場での表現でありその定義者は誰か

議論の最初に上記のテーマを扱ったかに記憶している。　そのとき、試案のFactorはﾕｰｻﾞが出す品質要求の表現か否か、また誰が品質要求の定義を行なうのかにつての議論がなされた。　これについての個人的見解を述べておきたい。

・品質要求を考える前に開発作業におけるｼｽﾃﾑ機能に関するﾕｰｻﾞ要求を考えてみると、ｼｽﾃﾑ機能のﾕｰｻﾞ要求はあいまいな形式・表現のものが多い。　このため、機能要求の明確な形式・表現化である機能要求定義は開発者側の人達が行ない、これをもとに開発の各ﾌｪｰｽﾞでﾕｰｻﾞとのﾚﾋﾞｭｰにて確認し合いながら確定していくのが現状作業の進め方である。
ｺﾝﾋﾟｭｰﾀ・ｼｽﾃﾑ・ﾋﾞｼﾞｮﾝの作成作業は（ｺﾝﾋﾟｭｰﾀ導入企業の情報ｼｽﾃﾑ部門に含まれているが）企業組織体の年間計画に沿ったものであり、ｼｽﾃﾑ開発作業自体には含まれないものと考えられる。　従って、開発者としての作業はｼｽﾃﾑ要求定義（機能および品質も含むがここでは機能が対象）がその第一歩であり、その機能要求定義作業の実施者は開発者側のﾏﾈｰｼﾞｬであると考えられる。　このﾏﾈｰｼﾞｬの解釈は以下のものである。

ﾏﾈｰｼﾞｬ ……要求定義作業において決定権を有し、これに基づいて開発されたｼｽﾃﾑ機能に対して責任を有する人達。

・これと同様にｼｽﾃﾑ品質に関するﾕｰｻﾞ要求を想定すると、ｼｽﾃﾑ品質のﾕｰｻﾞ要求もあいまいな形式・表現であることが予想され、品質要求の明確な形式・表現化である品質要求定義はやはり開発者側の人達が行なわざるをえないと思う。
この品質要求定義の確定化の過程は試案にも記述されているが、上記機能要求定義と同様の形態であると推定される。　また、試案においてはFactorはﾕｰｻﾞの品質要求ではなく品質要求定義であり、その作成者は開発者側のﾏﾈｰｼﾞｬであると解釈できるが、このことは上記機能要求定義を考えると妥当であると思う。

・但し、試案ではﾕｰｻﾞの品質要求に関して明記されていないが、Standardではﾕｰｻﾞの品質要求とFactorとの関係解釈・見解を記述しておくべきであると思う。

■Product かProcessか

議論の途中で、試案ではProduct とProcess とを扱っているがSoftware QualityにProcess が含まれるか否かの討論がなされたかに記憶している。 これについてはワークショップから帰ってきてからも何度も頭の中をよぎっている。 以下の記述は、この思考を繰り返している事柄に関するものである。

・ﾊｰﾄﾞｳｪｱ品質規格を考えてみると、ﾊｰﾄﾞｳｪｱ製品はある特性をもつ物質・物体から機械および人手を介して加工され新たな複合特性をもつ物質・物体として作成されるものであるため、製品即ちProduct に対して規定しているものであるとの印象を強くうける。

・ｿﾌﾄｳｪｱの品質を考えてみると、各段階で人の思考結果および人手を介在する多くの過程を経て作成されるため、Product に関することかProcess に関することかを明確に識別できない。
Measurmentもしくは Metricの具体例を考えてみると、その多くはProcessに関するものと思えてくる。

ﾃｽﾄ・ｶﾊﾞﾚｰｼﾞ …これは対象とするﾌﾟﾛｸﾞﾗﾑ 特性ではなく、そのﾌﾟﾛｸﾞﾗﾑ があるﾌﾟﾛｸﾞﾗﾏ 達によってﾃｽﾄ された作業結果を示す尺度である。従って、Product の品質特性ではなく、そのﾌﾟﾛｸﾞﾗﾑ のﾃｽﾄ Process の品質特性を示す尺度である。

ｴﾗｰ 寿命……あるﾌﾟﾛｸﾞﾗﾑ があるﾌﾟﾛｸﾞﾗﾏ によって作成される作業の特性を示す尺度と思える。

・ここではProduct にふれていないがProcess に関するものもあり、StandardとしてはProduct とProcess との両視点をもっているべきであると思う。

・ワークショップの後、当試案とは若干異なったアプローチ (Process 重点の？) の研究を知ったので紹介しておきます。

Ph. Victor R. Basili (University of Maryland)
「ｿﾌﾄｳｪｱの品質および生産性の定量化に関する研究」

最後に、当ワークショップにて各種提言および議論にて私どもの個人的理解度が深まったことに関して、参加者の方々に謝辞を述べておきたいと思います。

IEEE 標準化試案の適用

野村 敏次
日本電子計算

1．IEEE 試案の理解

試案の目的，枠組，階層，測定の方法論，について，以下のように理解し，また，付録の FACTOR, CRITERIA についても表にして，どんなものがあるかを理解した後，付録の例題に従って，マトリクスを作ろうと努力した．

(1) 標準化の目的

ここで作ろうとしている標準は，各組織においてソフトウェア品質メトリクスを活用するための枠組であり，これによって特定のメトリクスを規定するものではない．むしろ，メトリクスを明確にすることによって，マネジメントしていくための標準的な手続きを定義しようとしている．

(2) 品質メトリクスの枠組

ソフトウェア品質メトリクスは，次のような枠組を持つ．

- Establishment（設定／確立）
  - ソフトウェアシステムに対する品質要求の設定を行う．
  - 品質に関する属性の選択と，優先順位付けを行う．
- Implementation（実施）
  - ライフサイクル各段階における，品質属性の測定手続きの実施．
- Validation（妥当性の検証）
  - 品質を計る尺度の正当性を確認する．
  - ソフトウェア特性（エラー率，工数等）との関係を示す有力な証拠を発見する．

(3) 品質メトリクスの概念

品質のメトリクスは，次のような FACTOR, CRITERIA, METRICS の３つの概念によって，階層的に示される．

Factor：最上位の階層であり，システムの品質をマネージメントの観点から表したものである．マネージャや品質要求設定者にとって意味あるものである．

Criteria：Factor を技術的な概念に分解／変換したものであり，技術者にとって意味がある．これは，マネジャと技術者の間のコミュニケーションを容易にするものである．

Metrics：基本的なレベルの情報，またはデータであり，プロセスとプロダクトから得られる．あるものは，直接データとして得られ，あるものは何等かの手続きにより算出される．

(4) 測定の方法論

- 参考文献／経験／標準／規則／法律等々を参考として，品質要求をリストアップする．
- それらを，重要度に応じて重みづけをする．
- 明確にされた要求に対して，関係者で議論を行う．
- 要求をソフトウェア工学的に分析する．
- 関係者による議論の結果と，ソフトウェア工学的な分析結果との矛盾を解決する（この時，相互に排他的な要求は除去する必要がある．
- すべての関係者の同意を得る．
- 品質ファクタを明確にする．
- 各品質ファクタに対して，測定可能なクライテリアを明確にする．
- クライテリアに対するメトリクスを選択する．
- 測定（データ収集，分析，解釈，評価）に関するコストの算出を行う．
  - 開発プロセスの変化に対するコストの算出
  - 組織構成の変化に対するコストの算出
  - 特別なハードウェアや訓練に対するコストの算出
- 測定を行うことによって得られる，利益の明確化を行う．
- 測定を実施する．
- 妥当性の検証を行う．メトリクスの正当性を示すため，メトリクスと，それが表す品質ファクタとの関連が高いことを証明する．
- 但し，メトリクスの正当性の確認は，開発環境が不安定な状況では行うべきではない（ｅｘ．設計に用いる言語，

インプリメントに用いる言語，プログラム開発ツール等々が頻繁に変更されるような状況の時）

２．問題点の整理

マトリクスを作ろうとした一日半の努力は，結果として実らなかった．その理由としては，以下のようなものが上げられる．

(1) FACTOR，CRITERIA，METRICS の各項目の相関関係が不明であり，勝手に繋がりを持たせて良いものか否か迷う．

プロジェクトあるいはプロダクトの特性により，相関関係は一様には決らないが，標準は設定する必要があるであろう．各プロジェクトにおいては，運用段階でこれをアレンジして使用するのが望ましい．

(2) 定性的なものに対するプライオリティ付けが難しい．

定性的なものは，必要だと思えば必要に思われといった気分的なものに左右される．やっている本人は，こんないい加減なことで良いのだろうかと悩む．更に FACTORの数が多いので，時間を掛けて考える割には，結果はかなりあいまいなものとなる．

試案では２０弱の FACTORが挙げられているが，通常これらの内の幾つかを除き何れも重要なものである．しかし，数の多いことは使いにくさに繋がる．現実に適用する場合は，システムの規模に関係なく３つ以内（多くても５つ以内）に押えるべきである．その際，相互に矛盾するものについての検討は必要である．

(3) 小さなシステムでもコスト的に見合うのか？

例題に挙げられたシステムは大規模システムであって，私がシミュレートしたものとは全くレベルが合っていない．小さなシステムに対して，こうしたことを行うことは，果してコスト的に見合うものだろうかと悩む．

基本ソフトウェア等では，品質保証にかけた費用を製品に転嫁することも可能であろうが，アプリケーションソフトウェアの場合には，恐らく不可能であろう．しかしソフトウェアハウスの場合には，品質評価をシステム化していることを公にすることにより，プロダクトや企業そのものに対する信用を上げられるといったメリットは考えられる．

また，計算センタの場合は，システムの規模に関係なくこの方法論の適用は可能なので，これらの品質評価の実施によってリソースの削減，処理スピードの向上によってコスト・セービングを図ることは可能である．

(4) 定量的なメトリクスとの繋をどうするか？ 先が見えないのである．

FACTOR は Ability であり，CRITERIA は Extent であり，確かに CRITERIA の方が，少しだけブレークダウンされており，数字的なものに近づいているイメージはある．しかし，問題はこの CRITERIA と METRICS をどのように繋げるかである．これから必要な作業は，この両者の結合であろうと思われる．おそらく，その時，日本的な QC の品質評価のための測定項目が役に立ってくるであろう．

(5) 単語の定義の必要性．

英語の単語を「．．．性」と翻訳しただけでは，共通の認識は得られない．方法論や枠組の討論も必要であるが，全員が同じ土俵上で討論を行うためには，それと並行して，単語の定義を明確にするといった作業が必要となる．単語の日本語訳についての議論は余り生産的でない．

３．検討会に出席しての感想

検討会での討論は，非常に有意義であったが，討論の過程で気になった点を２つ程上げておきたい．

(1) 視点について

IEEEの試案が，マネジャの視点から多く見られており，ユーザの視点が欠けているという議論がなされ，ＳＱＵＡＬＡＳの例が取り上げられ，まずユーザの視点から見るべきだという意見が多かったように思う．これについて一言述べたい．

通常，ソフトウェア開発のプロセスは，ユーザの要求を分析し定義するところから始まる．当然のこととして最初にあるものはユーザ要求ではあるが，ユーザ要求そのものは，ソフトウェア開発のプロセスには入らない．同じ意味で，ユーザは品質要求を出すであろうが，ユーザの要求そのものは品質評価のプロセスに入れない方が良い．ユーザ要求を検討／分析し，品質要求定義を行うところからシステムとしての品質評価が始まるものと考える．

マネジャは，ユーザ要求の代弁者であり，技術者との橋渡し役であると考えるのが妥当である．更に過激なことを言えば，品質評価はユーザのためではなく，マネジャのた

めにあるとも言える．品質が悪く，ユーザに叱られ，かつ会社に損害を与えるようなことになって，一番困るのはマネジャなのである．その意味においてマネジャは，ユーザの代弁者にならざるを得ないのである．

従って，IEEE の試案でも述べているように，「マネージメントしていくための標準的な手続きを定義する」ことが重要であり，ここにおいては，ユーザの視点は入れる必要はない．

(2) 日本的 QC との整合性

試案を検討中に，わが国における QC の話が数多く出され，両者の整合性が議論の対象になった．私の感じでは，日本的な QC というものはボトムアップ的なアプローチをとっており，

IEEE 試案はトップダウン的なアプローチをとっているように思えた．

誤解があったら御容赦願いたいが，日本的な QC はソフトウェア開発過程において得られる数値的なものを出来るだけ多く測定し，出された結果を評価して品質特性に結びつけていく．従って，数値は測定されたが，それが何の役に立つのか解らない場合も出てくるようなイメージを受けた．これに対して IEEE 試案は，概念構成を定め，決められた方法論に基づいて品質特性をブレークダウンしていく方法を取っている．

両者をつなげるものは，おそらく測定項目であり，測定内容であり，測定データであろうと思われる．従って，方法論としては IEEE の試案を採用し， CRITERIA と METRICS の接点として，これらを用いるのが良いのではないかと考える．

ソフトウェア品質メトリクスへの過大なる期待

佐藤 隆
ATR 通信システム研究所

ソフトウェアの品質を測定評価するためのプロセスは、一般的に図1のようになる。なお、この図に示した4つのプロセスについての詳細な説明は、この論文集において他の人が行なっているので省略する。

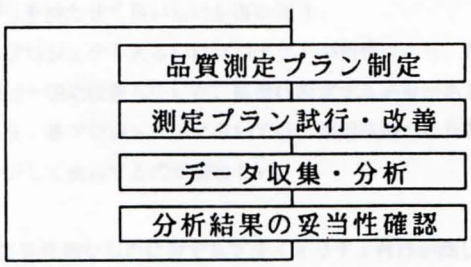


図1 ソフトウェア品質測定プロセス

このソフトウェア品質測定プロセスの内容は、測定するソフトウェアの性質や、測定する人の立場により異なってくる。大規模オンラインプログラムのようにソフトウェアが故障したときの社会的・経済的影響が大きければ信頼性や保守性が重視されるであろう。また、ワープロソフトなどのように多数の人に使われるソフトウェアであれば使用性や可搬性が重視されるであろう。

次に挙げる品質測定例は、机上のものである。実際の品質測定がこれらの例のようにでたらめに行なわれているわけではない。

たとえば、ワープロのソフトウェアを購入する際の品質測定プロセスを図1にしたがって行なうと、次のようになる。

(1) 品質測定プラン制定

自分の予算や利用目的により、品質要因やその測定項目・測定方法、分析・評価方法を決める。

○外部要因
- 予算は十万円程度である。
- ビジネス用文書を作成する。
- 簡単な図表を作成する。
- 私の持っているパソコンは××である。

○品質要因
- 機能性(Functionality)
- 使用性(Usability)
- 効率性(Efficiency)

○測定項目
- 金額
- 適用機種
- 必要記憶容量
- 会社での使用実績
- ビジネス用語登録数
- ユーザ単語最大登録数
- 図表作成機能

○測定方法
- パソコン情報誌などで調べる。
- 販売店で実際に操作する。

○分析・評価方法
- 各測定項目ごとに順位点をつけ、総合点で評価する。

(2) 測定プラン試行・改善

パソコン情報誌などを読んで情報を集め始めた。しかし、ビジネス用語登録数は調べられたが、自分の会社で使う単語が登録されているかどうかはわからない。また、販売店では小学生がゲームで遊んでいるのでパソコンが満足に使えない。店員の商品知識もあまり参考にならない。販売店での調査は時間がかかり、有益なデータも集まらないので行なわないことにした。

(3) データ収集・分析

販売店で試用することはあきらめて、パソコン情報誌だけにより調べた。その結果により、△社の☆☆を購入した。

(4) 分析結果の妥当性確認

☆☆を使っていて、次のような不満・不都合が生じた。
- ユーザ登録した単語が最後に選択される。
- 文書の切り貼りや流用の操作が面倒である。

・文章の変換が思ったとうりにならない。
・パソコンの機種が違うため、会社のワープロとデータ互換性がない。
・マニュアルが役に立たない。

次にワープロのソフトウェアを購入するときには次のような測定項目も必要であると反省した。（△社に対して苦情・提案を行なうことも消費者の行動パターンの一つである。しかし、ここでは品質測定の話に限ることにする。）

・学習機能、切り貼りなどの機能は、その有無だけでなく、操作性も品質評価の項目として取り入れるべきである。
・パソコン情報誌による情報だけでなく、典型的なビジネス文章を実際に作ってみることが必要である。

ソフトウェア品質測定の阻害要因として次のようなものがある。

(1) 品質要因・メトリクスが決定できない

品質要因・メトリクスを決定するための討議に関係者もしくは代表者を一堂に集めることができない。その理由として、以下の点が挙げられる。

・参加すべき関係者が多すぎる。
・討議する時間がもったいない。
・プロジェクトに未参加のメンバがいる。
・全員が参加すると意見がまとまらない。また、無理にまとめようとすると総花的なものとなってしまう。

(2) 結果がプロジェクトの役に立たない

品質メトリクスの測定・分析結果がプロジェクト終了後に提供される。プロジェクトの管理者や担当者にとっては、自分が参加しているプロジェクトの状況や、自分の作業の結果が品質にどう影響しているのかをリアルタイムに知りたいと望んでいる。しかし、現実のソフトウェア品質メトリクスおよび分析体制は、その要望に答えることができない。品質測定は自分達を支援・指導するためでなく、能力を評価するために行なうのではないかという疑問をいだく。（SRAの岸田氏によると次のプロジェクトには役に立つとのこと。）

(3) データが正確でない

正確なデータが集まらない。特に人手によるものはある程度の恣意が入り込む可能性がある。上司がうるさいから適当に提出してしまえ。自分にとってまずいデータがあるから適当に調整してしまう。

(4) コストが増加する

品質メトリクスを測定するのに人や時間、コストがかかる。計算機で自動的に集めるにしてもデータを集めるコストや、データを保存するコストなどが増加する。

(5) メトリクスに決定版がない

ソフトウェアの種類や開発体制により、品質メトリクスが一定ではない。特にユーザからみると、これさえ比較したら品質が判断できるというものがない。

(6) メトリクスに信頼性がない

測定した品質メトリクスが信頼性や保守性を的確に示しているかどうか判らない。品質メトリクスが社会的（業界的）に認められていないため説得力に欠けている。

これらのことから、ソフトウェア品質メトリクスに対して次のような性質を持つことを望む。

(1) ソフトウェア開発者が自分達が開発中のソフトウェア（仕様書やプログラム）の品質を評価できる。

(2) ソフトウェア開発者が自分達が開発したソフトウェアの品質を評価できる。

(3) プロジェクト管理者が現在および将来のソフトウェアの品質を知り、手遅れにならないうちに対策を施し得る。

(4) ソフトウェア購入者が購入前にその品質、性能を評価できる。

(5) 品質メトリクスだけでなく、その測定・分析体制についても考慮する。

(6) その品質メトリクスが社会的に認められる。

ソフトウェア品質メトリクスは、ソフトウェアを生業としている人、必要としている人にとっての拠り所となるものでなければならない。決して、学会発表のためだけの研究材料、管理しているというポーズをとるための道具としてはならない。

アプリケーション開発から見たソフトウェア・メトリクスについての個人的感想

臼井　義美
日本電子計算

我々ソフトウェアハウスの多くが開発を行っているビジネス・アプリケーションでは、定量的な信頼性に関する研究や実施は甚だ未熟な所が多く、今回の討論は非常に興味深く聞かせていただいた。ただ、アプリケーション・ソフトウェアの開発に携わっているものの立場からいくつか感じたことを述べてみたい。

我々が開発、販売しているアプリケーション・ソフトウェアは、ＯＳなどの基本ソフトと根本的な相違があることを先ず認識しておきたい。

すなわち、基本ソフトにおいては一般のアプリケーション・ソフトに比較して遥かに信頼性に関する比重が高いものである。（もちろん、アプリケーションにも絶対の信頼性を要求されるものもあるが。）それは、基本ソフトはソフトウェアというより、ハードウェアの一部分として捕らえるべきものだからであろう。例えば、音楽の再生装置を考えてみれば良い。そこで言うソフトウェアとは音楽そのものであって再生装置に組み込まれた各種のコントロール関係をつかさどるプログラムは、ハードの一部分と見なしているのが普通である。従って、工学的分析や評価に関する議論に対して違和感がない。つまりハードウェアと一体となって、機能性、信頼性、使用性、保守性などを云々するべきものであって、音楽に当る本当の意味での（？）ソフトウェアと同じ土俵でその信頼性を論議しても始まらないのである。

アプリケーション・ソフトにも大きく分類して２種類ある。ソフトウェア・ハウスやメーカが開発して販売しているソフトウェア・パッケージとユーザの要求に合わせて開発するソフトウェアである。

アプリケーション・ソフトにおいて、ユーザが求めるものは信頼性などのほか、楽しさ、飽きなさ、使い易さ、困らなさ、常識的さ、回復性の良さ、美しさ、十分さ、安心感、満足感など、様々な要求を持つものである。つまり、正確であると言うだけではソフトウェア・パッケージとしての商品価値は決定できないのである。理性よりも感性に訴える製品が求められる時代になってきたと言うことができる。たとえば自動車などの成熟商品はもはや品質によって消費者を引き付けることは難しく、付加価値やデザインなどの、感性を刺激するものによってシェアを競っている。これに比べてソフトウェアが未だに品質を云々しているのは、商品としての未熟さを物語っていると言えるであろう。

以上のような観点から見ると、標準化試案はハードウェアの延長線上でとらえた、ソフトウェア開発に対する品質の定量化をねらったものといえる。もともと信頼性とはハードウェアなど、アートでない世界にこそ有効な指標であると言えなくもない。しかし、理想的なアプリケーションに対して、品質は程度の差こそあれユーザの求める価値の一つにすぎない事を認識しておかねばならない。

一方、ユーザの要求に応じてオーダーメイドのソフトを開発する場合、制御用などのように、要求者と請け手の技術力の差が余り顕著でない場合は未だしも、ビジネス・アプリケーションなどでは、要求する方が全く素人の場合など、ユーザ要求を満たすための信頼性の確保と言うのは決定的に困難となる場合が多くある。従って、完全な要求仕様があって、その内容を満たすような設計をすれば良いという理想的な状況がととのっている場合は非常にまれで、プロトタイピングや完成後のメンテナンスで改善を繰り返しまともな物を作っていくことが多い。

これに対して基本ソフトは、通常メーカが独断的に決定した仕様をもとに開発し提供しているもので、外部のユーザの要求に合わせて開発したものではない。従って担当部門で決定された要求仕様を完全に満たす事が、品質の確保に直結している。

先に成熟商品の代表として挙げた自動車の信頼性も、一般利用者が希望する任意の仕様で１台１台作っていくなら多分ソフトウェアの信頼性とそれほど変わらないかもしれない。もちろん、ＯＳだってすべてユーザの要求に基づいて開発することになれば状況は同じであろう。

それでは、大量生産型商品の開発と１品型商品の開発においては、信頼性の基準にどのような差異があるのであろうか？　同じ基準が適用できるのであろうか？

＊（例１）ユーザが要求する信頼と、メーカが考えている信頼の食い違いは激しい！

彼女が「あなたを信頼しています。」と言った場合，彼の反応は？

ａ．何もしないことを信じています。

ｂ．してもいいです。責任さえ取ってくれれば

ｃ．必ずしてほしいのです。

このようにユーザの要求と言うものは、推し計り難いものなのである。従って信頼性を評価する前提は、要求が確定しているものについて行うと言うのは正しい。しかし、それでは一般のユーザからの要求は信頼性を保証しないことになるのだろうか？

＊（例２）ユーザが「システムを信頼しています。」と言った場合、
ａ．バグが無いことを確信しています。
ｂ．バグがあっても仕方がありません。責任さえ取ってくれれば
ｃ．バグは覚悟の上で使います。

このときａと解釈するユーザに提供するシステムは、そのユーザに完全な仕様を要求しなければ、期待された信頼性を達成できないであろう。ではｃと考えているユーザに対しての信頼性とは何か？

つまり、ユーザは果たしてあらゆる部分について高い信頼性を要求しているのかと言うことである。ソフトウェアの中には絶対正確でなければならない部分もあるが、間違っても大きな問題に発展しない部分もあり、できれば仕様通りであることに越したことはない程度の部分もあるはずである。
いま、要求に合致させなければ高信頼性は得られないと考えているが、逆に要求品質の程度についての定量化を考慮する必要があるのでは無かろうか？

次に、メトリクスによる定量化について考えてみたい。メトリクスには非常に少ないながらも客観的に数値化できるモデルを設定可能な場合と（正確であるかどうかはともかく共通の認識ができること）、多くの重み付けやｎ段階評価などに見られるように、主観的な数値化を行わなければならないときがある。この時、主観的な数値化を客観性のもとで取り扱えるかどうかが問題である。つまり、異なるソフトウェア、異なる評価委員、異なるユーザについて共通の認識を得られる数値化が可能かと言うことである。
もし、これらの評価の結果が相互に比較できなければ、定量的評価はどのような意義を持ち得るのであろうか？

この様な疑問がまだまだ出てきて、とても理解できたような気にならないのが現実である。今後の討論を通じてソフトウェア・メトリクスの意味を理解し、多くの方々の研究成果を聞かせていただきながら、少しずつ実際のアプリケーション開発の現場に適用してみたいと思っている。そして、メトリクスがアプリケーション・ソフトに対する評価基準としても十分役立つことを願っている。

ソフトウェア品質メトリクスフレームついての考察

山田　淳

現在、ソフトウェア生産の工業化をめざしたソフトウェア開発支援システムＩＭＡＰにおけるソフトウェア品質評価技術・品質評価環境構築に関する研究に従事している。

はじめに

ソフトウェア品質評価について以前からいだいていた問題点やＩＥＥＥメトリクスワークショップにおいて明確になった問題点について、筆者なりにまとめたものを以下に述べる。多くは問題点としてそのまま残っているが、解決のいとぐちが見えてきたものもある。

１．　品質メトリクスフレームの「ユーザー」

－　誰のための品質評価か　－

品質メトリクスフレームを用いて品質評価した結果が誰にとって意味をもつのかをはっきりさせる必要がある。品質についての関係者を分類すると、顧客（ユーザー）サイドと開発（メーカー）サイドとに大きく分けられる。さらにその中も分類すれば以下のようになる。

・顧客（ユーザー）サイド

1.マネージャー、システム導入についての決定権と責任をもつ人々

2.購入担当者、受入れ担当者

3.実際に使う人々、エンドユーザー、オペレータ

・開発（メーカー）サイド

1.マネージャー、システム納入、販売についての決定権と責任をもつ人々

2.セールス担当者、システム販売者、営業技術担当者

3.プロジェクト管理者、品質保証担当者、検査担当者

4.開発担当者、設計者、プログラマ、オペレータ

これらの関係者は種別が異なれば品質評価の活用方法もまた違う。マネージャーであれば、ユーザーサイドもメーカーサイドも特別に詳細な情報を必要とすることは少ない。使いやすいし、満足のゆく機能で業務の能率もあがった。信頼性も高い。このような抽象度の高い情報と、そう判断できる根拠が存在し、必要ならばいつでも取出してみせることができるという情報を必要としている。ユーザーサイドの受入れ担当者やメーカーサイドの検査担当者であれば、的を得た検査をしたいと考えるのが一般的であり、購入担当者であれば類似したシステムの比較を行いたいと考えるのが普通なので、いずれもメトリクスについての詳細な情報を必要とする。これに対してメーカーサイドの設計者や開発担当者、プログラマは、どのような仕様を満たすように作ればよいのか、いかに設計すべきか、どのようにすれば実現できるのかといった、品質を作り込むための指針になるようなメトリ

クスについてかなり詳細な情報を必要とする。

品質評価はこうした種々の要求に応えていかなくてはならない。したがって詳細なメトリクスを考えるときは、どの種別の人々に対しても意味を持たせるのかを明確にすることが重要である。また、いずれの種別の人々のためにも特性やメトリクスを準備することも必要である。

2. 品質メトリクスフレームの役割

－ 品質メトリクスフレームはなぜ必要か －

品質評価の活用の仕方の違う人々をいくつかの種別に分類することが可能で、それぞれの要求に合わせた品質特性やメトリクスが必要であるということを明らかにしたが、これら品質特性やメトリクスが全く別々に独立しているわけではない。マネージャーが要求している品質を判断できる根拠とは品質検査の結果や品質管理についての情報である。またプロジェクトで実現する品質についてプロジェクトリーダーが設計者と打合わせるときにも、要求に対する実現の方法を検討するために、互いの情報を解釈して交換する必要がある。このとき互いのメトリクスをよりあわせる枠組み（フレーム）があれば、意志の疎通がはかれる。

品質メトリクスフレームの中に出てくる特性あるいは要因といったものは品質関係者間のコミニューケションをはかるのための表現である。そしてフレームはコミニューケションを誘導する手順である。

ユーザーとメーカーとの間であれば、ユーザーの要求を特性でまず表現し、特性から開発者のためのメトリクスへとさらに翻訳して、開発者に伝えるのである。あるいは同じユーザサイドの内部あるいは開発サイドの内部であっても、互いに異なった種別の人々の間で、メトリクスから特性へ、特性からメトリクスへと翻訳することにより意図を伝えることができる。導入したシステムの社内の評判や現場の意見を聞くことの科学的アプローチともとれる。このように品質関係者間で品質について共通した認識の上で話をするために品質メトリクスフレームは不可欠である。

他方、品質評価の作業を適確にぬけなくかつ効率的に行う目的でも品質メトリクスフレームは不可欠である。予めメトリクスを準備しておくことにより、毎回個々のプロジェクトで考案する作業を選択や追加程度に大きく省力化できる。また特性やメトリクス間のトレードオフや協調関係も整理しておくことも可能であり、品質評価作業の能率を上げられる。このように評価の手順や結果の集計、弱点の発見などを支援するためにも品質メトリクスフレームは不可欠である。

3. 品質メトリクスフレームの対象範囲

－ どんなソフトウェアに適用するのか －

ＩＥＥＥドラフトで提案されていた品質メトリクスフレームは受注システムそれも高度の要求を満たすシステム（ＤＯＤが発注したシステム）を念頭においている。しかし、一般消費者向けから特定顧客からの受注品、あるいは商品から最先端技術システムまで含むシステム、ソフトウェアを対象範囲とすべきである。

4． 品質メトリクスフレームの構造

－ どのような構造が望ましいか －

共通の認識のもとにコミュニケーションをはかるを目的として考えると、品質フレームに望まれる構造とは次のようなものである。

フレームは階層化されており、上位レベルの抽象度の高い特性からはじまって、次第に詳細化し下位レベルへ進むにつれてメトリクスや具体的な尺度になる。そして上位レベルの特性は共通した基盤を構成するのに対して、下位レベルのメトリクスや尺度は品質評価情報の活用の仕方ごとに別れている。

実際の運用を考えると、いきなり数多くの特性や要因を列挙したのではとまどうばかりである。例えばＩＥＥＥドラフトは、一つのレベルに一度に２０個を越える要因を用意していることが、要因相互の違いを不明確にしていた。そのためＩＥＥＥドラフトをもとに作業を想定したときにも、要因相互の違いがどこにあるのかといった要因の意味を解釈することの方に作業の内容が費やされ、要因を評価する作業ははかどらなかった。

したがって一つのレベルに３～８個程度の特性を準備し、その下のレベルにそれぞれの特性を説明するサブ特性をやはり３～８個程度ずつ用意して、樹木状にする。そしてその下位レベルはマトリックス状に特性とメトリクスとを関係させる。そして最上位レベルから最下位レベルまでで３ないし５レベル程度の深さとする。このような構造が望ましい。

5． 品質メトリクスフレームの内容

－ どんなメトリクスが望ましいか

特性やメトリクスとして何をもってくればよいか。最上位レベルの特性はＩＳＯドラフトの機能性、信頼性、使用性、効率性、保守性、可搬性の６つ特性がある。これらは十分納得できるので、詳細かした特性についてもこれをもとに決められる。

問題はメトリクスおよびメトリクスと特性との関係である。メトリクスとしては要求どおりか確認する際に使うメトリクス、品質を作り込むのを助けるメトリクスというように、品質関係者の種別にあわせたメトリクスを考える必要がある。またメトリクスは理解しやすいこと、客観的なこと、測定しやすいことが求められる。さらに重要なことは妥当であることである。ひれはメトリクスの妥当性を検証する作業を必要とする。どのメトリクスも、その妥当性を証明するデータを呈示できる状態が望ましい。そしてどの工程で行うのかといったメトリクスを行うタイミングとかかる費用についても言及されているとよい。

標準化を考慮すると、既に使われているメトリクスからアイデアレベルのメトリクスまで広くメトリクスをとり揃えて提案する作業が必要である。

６．　品質評価の手順

－　品質メトリクスフレームをどう使うか　－

ここではＩＥＥＥドラフトおよびＩＳＯドラフトをもとに筆者自身の考えを加味して品質メトリクスフレームを使った品質評価の手順をシミュレートする。

品質メトリクスフレームにより品質特性からメトリクスまでを予め準備しておく。そしてユーザーサイドなら購入あるいは導入したいソフトウェアシステム、メーカーサイドならば受注したソフトウェアシステムの要求にあわせて品質の要求も明確化する。重視する品質特性から順に優先順位をつけるのである。これは品質の要求定義と言える。このとき数多くの品質特性の中からどれかを選択してメトリクスを行う方法もある。しかしそうすると正確性や信頼性とかにたいてい偏ってしまう。それで数種の品質特性の中から重要な品質特性を先頭にして優先順位を決めておき、メトリクスを優先度に応じた配分で全ての品質特性に対して行う方法のほうがよい。

こうして優先度の高い特性が決まれば、品質メトリクスフレームから自動的に優先度の高いメトリクスが決まる。ここでメトリクスのコストを勘案して実施するメトリクスを選定する。またソフトウェアシステムの要求や分野にあわせてメトリクスを追加したり選択してもよい。

次にメトリクスの評価基準を決める。例えば「画面一枚の表示にかかる時間の短さ」というメトリクスならば、「０．５秒以下ならＯＫ」という基準を決めるわけである。このあたりではソフトウェアシステムそのものの要求定義や仕様決定と歩調をあわせて決めることになると考えられる。

そしてメトリクスの持つ尺度で実際に測定を行う。「画面一枚の表示にかかる時間」を測定することになる。

測定結果と評価基準をもとにメトリクスの評価結果が決まる。全てのメトリクスの評価結果を品質メトリクスフレームの構造に沿って集計すれば品質特性ごとの評価も決まる。

この評価結果を書込んだ品質メトリクスフレームをもとに、品質改善の対策を練ることになる。したがってプロジェクトの工程ごとにメトリクスを行い品質評価をして評価結果を書込んだ品質メトリクスフレームを作り、プロジェクトの品質を現在進行形でモニターするのが望ましい。

７．　標準化

--どこまで標準化するのか－

標準化した品質メトリクスフレームを実際に運用することを考えると、がっちりと固めてしまうより、ある程度の自由度を持たせる必要がある。例えばどこの会社でも同じ社内でさえソフトウェア開発部門ごとに異なった文化を持っている。標準化が自由度を残すことにより、標準化というトップダウンとボトムアップが適度にかみあって、運用がうまく軌道に乗り、真に必要な品質の向上が円滑に進められると考えられる。

技術的には品質メトリクスフレームの上位２～３レベルの特性の定義を標準化し、メトリクスなどは選択や追加自由ということで数を並べて推薦するという形が良いのではないかと考える。もちろんいくつかの基本的なメトリクスがあれば、標準メトリクスとして、そのメトリクスについては最低限でもかならず調べることとしてもよい。

8. 結 論

品質評価や品質メトリクスフレームについて考えるところを述べ、以下の事項を明らかにした。

1.品質メトリクスフレームのユーザを意識する必要があること

2.品質メトリクスフレームは、品質について話をするために、また品質評価の作業を実際におこなうためにも重要かつ必要であること。

3.品質メトリクスフレームの構造は、階層化された、上位が特性を記載したツリー構造、下位レベルが特性とメトリクスとをつなぐマトリクス構造からなることが望ましい。

4.メトリクスは、妥当であること、品質メトリクスフレームのユーザに対応していること、理解しやすいこと、客観的なこと、測定しやすいこと、タイミングを示していること、コストを見積れることなどの性質を持つことが望ましい。

5.品質メトリクスフレームを用いた品質評価作業の手順

おわりに

整理は十分とは言えないが、列挙することは、これからどんな作業が必要かということを明らかにするために有効である考える。これからの研究や標準化の作業に役立てば幸いである。

第５回夏のプログラミング・ワークショップ

全体報告

藤野　晃延
富士ゼロックス情報システム

１．準備段階

前回のプログラム委員会を務められた松尾正敏さんが，第４回「夏のプログラミング・ワークショップ」報告書の全体報告を書かれ（Seamail Vol.1, No.10 に掲載されています），その文末でおっしゃられた氏の提言が，そもそも今回の，企画・運営をＯＢでやってみよう，という話の発端になったのではと思います．さらに下地として，’８３年に行われた第１回の時の参加メンバが何人か周りにいて，そのＯＢ会を開いた時にも，ＯＢによる企画・運営といった話題が取り沙汰されたこともあったりしました．こうした時を得たようにして引き受けてみたのですが，一体何から手を着けたらいいのか皆目見当もつかなくて，前途多難と思わざるを得ませんだした．

それでもプログラム委員の見知った顔ぶれに，まぁ何とかなるだろうと高をくくっていたのでしたが，１人も運営経験者のいないＯＢだけでは流石に心許ないので，前回のプログラム委員でもあり，盛岡が地元の野村行憲さんに，今回の実行委員長として参加して頂くことにして，第５回の運営スタッフが決まりました．

わざわざ盛岡から実行委員長にお出で頂き，２回のプログラム委員会の開催をへて，どうやら企画・運営は何とかなりそうだ，との実感を得たのは確か６月も半ば過ぎていたように覚えています．しかし肝心要のテーマについては，プログラム委員の意向がなかなかまとまらず，最後まで難航し，Seamail にワークショップの Call-for-Paper を載せる直前まで二転三転しました．ようやくたどり着いたのが，「バグのないソフトウェアの実現をめざして」という今回のテーマだったわけです．

２．ポジション・ペーパ募集と審査

テーマの設定になぜ時間がかかったかといいますと，ワークショップの第一の目的を，「広義でのテクノロジ・トランスファとインフラストラクチャの形成」と考えるプログラム委員会では，質の高い議論をするには，テーマも充分に吟味して設定すべきだろうということで，テーマ設定に議論百出，結局，

実行委員長	野村行憲	（岩手電子計算センタ）
プログラム委員長	藤野晃延	（富士ゼロックス情報システム）
プログラム委員	青島晴江	（株）高度情報システム
	佐久間まゆみ	（株）神戸コンピュータ・サービス
	佐原　伸	（株）ソフトウェア・リサーチ・アソシエイツ
	沢田寿実	（株）ソフトウェア・リサーチ・アソシエイツ
	月岡義範	中央システム（株）
	松下和隆	（株）三菱電気東部コンピュータ・システム
	村井　進	ムライ・テクニカル・ライテングCo.Ltd.
	渡辺雄一	（株）電力計算センター
事務局	中島千代子	ソフトウェア技術者協会

表　第５回夏のプログラミング・ワークショップ運営スタッフ

・テーマを絞って具体的にしたほうがいい
・ポジションが明確なら，テーマに固執しないほうがいい

という全く正反対の２つの意見に分かれてしまったことでしょうか．それでも最終的には，プログラミング経験者であるならば誰でも知っている「バグ」というものに注目し，これを多角的にとらえてディスカッションしてみようということで，どうやら今回のテーマが決まったのでした．

実はこのテーマをきめた背景には，プログラム委員長として，ひとつの思惑があったのです．それは，昨今のハード・ウェア／ソフトウェアの急速な進歩に伴い，これまで大学や研究レベルでしか話題にならなかった，形式的仕様記述によるアプローチ，あるいは数学的モデルに基づく厳密なソフトウェア開発の試みなどが，どの程度まで実用になりつつあるのか，あるいは導入されつつあるのか，といったことに興味があり，それをこの辺で確かめておくのも悪くない，そして，それに類するポジション・ペーパが応募されることを内心期待していたのです．しかし，残念ながらというか，当然というか，それに言及したペーパはわずか１件しかありませんでした．ソフトウェア開発の現場の状況を，はからずも再認識することになりました．

最初の Call-for-Paper がでたのが，確か４月ではなかったかと思いますが，既に５回目ということで定着していることと，「バグ」というテーマの馴染みやすさも効を奏したか，応募されたポジション・ペーパの総数は，募集定員をはるかに上回って７５編も寄せられ，プログラム委員諸氏は，一転して厳正な査読委員となって，慎重にポジション・ペーパの審査にあたらねばならなくなりました．

正直なところ，寄せられたポジション・ペーパの内容に関しては，それほど大きな差はなく，できることなら全員参加としたかったのですが，会場の関係などもあり，半数近い方々には涙をのんでいただかなくてはなりませんでした．それでも採用した数は，定員を１０名もオーバし，果たしてこれはワークショップの適性人数といえるだろうか，という疑問も残りましたが，不採用の決定という作業は，審査に当たったプログラム委員諸氏にはなかなか辛い作業でしたので，定員枠の拡張ということで対処することにしました．

ここで，ワークショップなどにポジション・ペーパを応募するために，審査に当たったプログラム委員諸氏からの経験上のアドバイスをいくつかを披露してみます．もし，応募したのに不採用になった方がいましたら，ご自分のペーパを見直してみるのも，バグ対策のひとつになるのではないでしょうか．というわけで，題して「成功するポジション・ペーパー傾向と対策」．

３．成功するポジション・ペーパー傾向と対策

(1) 体裁は重要だ

まずはポジション・ペーパの体裁です．体裁より内容だという考えは捨てる必要があります．

ワードプロセッサ等が普及したこともあり，応募の９割以上が，ワードプロセッサあるいはそれに類するもので作成され，手書きはきわめてわづかでした．文字の読み易さ，認識し易さという点で，ワードプロセッサ等により作成したペーパの方が明らかに有利です．特に今回のように応募数が多い場合にはなおさらです．まず査読者に読ませることが大切で，どんなに内容が優れていても，読み難いペーパでは，最初の段階で不利です．手書きでの応募は，この点を覚悟しておかないといけません．また，ただ文章を続けるだけでなく，適当な段階，見出し，図，といったものを有効に利用することも重要なポイントになってきます．全く同じ文章でも，適切な見出し，あるいは図を添えることで，説得力に大きな差が生じて来ますから，ぜひ工夫して欲しいと思います．たとえば，この原稿もほとんどが文章ですが，そういった意味ではそう読みやすいものではないと思います．

(2) 内容は当然重要－上滑りはだめ

さて，肝心のポジション・ペーパの内容です．

何人かの査読委員から，最新の技術あるいは話題に関する単語の羅列で終わっているようなのが多いのではないか，という指摘をたびたびされました．その理由として考えられるのは，昨今，情報源として日経*** などをはじめとして，コンピュータ関連雑誌などが広く購読されており，その結果，「ことば」としては知っていても，知識として充分に咀嚼されないままに終わっている情報が多いのではないでしょうか．したがって，引用が単に新技術の名前を列挙するだけに終わってしまい，具体的にその新技術をどう応用するか，といった自分なりの考察に欠けているように見受けられます．せめて，自分なりの考察を明示してもらえたら，と思わずにはいられません．

(3) 決め手は覇気

そして最後は「覇気」です．

査読委員に「おっ，コイツ，なかなか面白いことをいうな」と思わせればシメたもの．多少のバグはご愛敬で，かえって変に整ってしまうより受けがいいようです．ワークショップのポジション・ペーパとは，要はそれを読んだ相手に「もっとコイツの意見を聞きたい」，「もっと詳しくコイツと討論したい」，といった気持ちをおこささせることが肝要で，そういった意味での「覇気」を是非とも盛り込んで下さい．

４．開催－盛岡にて

現地では，おおむね予定通りの運営が行えたと思います．これはひとえに，実行委員長をはじめとして，地元から参加してくれた方々の協力におうところ大でした．本当にありがとうございました．

しかし何といっても，「盛岡」という場所が本当に素晴らしいですね．９月上旬という時期的にも過ごしやすく，天候も安定しており，屋内・屋外いずれでも討論に集中できる，素晴らしい環境だったと思います．議論に疲れた脳も，新鮮な空気に触れるとたちまちのうちにリフレッシュ，東京や首都圏ではこうは行かないでしょう．自然に有り難みを痛感した次第です（きっと，盛岡でつくられるソフトウェアは素晴らしいものにちがいないというのは，いいすぎですな）．

初日のキーノート・スピーチとして，山形より，ＹＣＣ情報システムの佐藤義明さんに，東北では情報処理産業関連の会社がある仙台から，日本システム・エンジニアリング仙台所長の山口忠義さんに来て頂き，それぞれ，地方におけるソフトウェア開発の実情とか，地方ユーザとの付き合いにおける苦心談（ノミニケーション）など，含蓄に富んだ話をしていただきました．

その後，各プログラム委員をパネラとするパネル・ディスカッションを行い，その主張に興味を持った人は適当にそのパネラがリーダとなるグループにエントリするという方法で，グループ分けを行いましたが，パネラとなった各プログラム委員の熱演で，初日早々，早くも夜１０時（！）まで，シッカリと議論の花を咲かせたのでした．

５．グループ別討論

いよいよ二日目からはグループ別討論です．特に意図したわけでもないのに，AI/ES，オブジェクト指向，プロトタイピング，形式的仕様，そして開発環境と，興味深いテーマ別に各グループが構成されていたのには驚きを禁じえませんでした．どのグループがどのようなアプローチを取ったのか，そして討論はどであったのかということは，グループ討論報告を読んでください．

グループ毎に適当な場所に分かれてのディスカッションは，期間中ずっと好天に恵まれたこともあり，私のグループを含めて，屋外に場所を定めたグループが多かったようです．懐深い盛岡の自然に包まれながらの討論は，普段忘れがちだった清々しさにあふれ，「考えを吐露し，意見を聞く」とうことを心の奥底から体験させてくれました．また一方では，会議室を占拠して一歩も屋外に出ることなく，ＯＨＰを使った討論に花を咲かせたグループもありました．

いずれにしろ，各グループ毎にさまざまな討論がおこなわれましたが，初日に若干みられた硬さもどこへやら，すっかり打ち解けた雰囲気に包まれ，最終日の全体発表にむけて，精力的な活動が続けられました．狭いホテルのシングル・ルームに集まり，深夜まで作業をしていたグループもあったと聞きます．

このグループ討論期間中の二日間，各グループのわがままをローカル・アレンジメントしてくださった，地元・盛岡から参加してくれた（株）岩手電子計算センタの方々に，ここにあらためてお礼を述べさせていただきます．ほんとうにありがとうございました．

６．最終日

最終日の発表は，ホテル側の手違いから（まさか「バグ」をテーマにしているからという配慮ではなかったと思いますが），当初の予定とは別の会場で行う羽目となりましたが，しかし災い転じて何とやら，なかなか明るく，スペース的にも余裕があり，全体発表にはおあつらえむきの会場となりました．また，ワークショップの最終日を飾るかのように，素晴らしい青空が広がり，雰囲気も嫌がうえにも盛り上がり，各グループの発表，それに続く質疑応答と活発な議論が次から次へと展開され，本当に時間が少な過ぎる！，と感じたのは，運営側だけではありませんでした．

各グループの発表は，全員が前に出てのリレー式発表スタイルを採りましたが，ＯＨＰを駆使しての各人各様の発表は，なかなか見応え聞き応えがありましたし，なんといってもある種の誇らしさ，満足感，そう，ちょうどひとつのプロジェクトを，苦労の末に完遂した時に見せる様を，発表者の顔に感じたのは私一人ではなかったことと思います．

７．振り返って

以上のように，今回のワークショップもまた成功裡に終えることができましたが，最後に若干の反省と，問題点の提起を行っておきたいと思います．

まず反省として，ＰＣのパネル・ディスカッションで，充分にお互いの主張について議論する時間がなかったことが挙げられます．これはパネラの数がやや多かったことと，発表時間の管理がルーズであったことが原因であったと思います．パネラの数を減らすか，発表時間の厳守を励行すればいいわけですが，一方では，グループ分けの指針として，グループ・リーダの主張を行うには，それなりに発表時間が必要なこと，また適正な人数のグループを構成する上で，参加者数に比例したグループ数，およびグループ・リーダ数が必要となるのは当然で，これにどう対処するかが，グループ分けの方法も含め，今後検討する必要があるでしょう．問題点の提起を含めて運営にあたって気がついたことは，

・宿泊施設以外に，時間外にも利用できるスペースを確保したい
・会議場の移動は極力避ける（備品・用具の運搬が大変）
・ＯＨＰなどはグループ毎に１台は用意したい
・懇談会は立食パーティ形式の方がいい

といったことです．

今回のワークショップでは，時間外にまとめや討論をホテルの自室（シングル・ルーム！）で行っていたグループもあり，このような場合に利用できるスペースを，宿泊施設とは別に確保しておけると便利なのですが，これは次回の課題としておきましょう．

８．新生「若手の会 in 八ヶ岳」

さて，すでにご承知と思いますが，「若手の会 in 盛岡」は今回をもって終了させていただき，次回の第６回「夏のプログラミング・ワークショップ」からは，場所を山梨県は八ヶ岳に移し，「若手の会 in 八ヶ岳」として新生する予定です．その詳細内容はまだ未定ですが，時間外のスペース確保は解消されるとか聞いています．テーマは，基本的には，現在の主要なプログラミング手法に的を絞り，ある共通の課題に対して，実際にそのプログラミング手法を適用し，そのケース・スタディを比較検討した実体験に基づく討論を行ってみようというものです．現在のところ，

・オブジェクト指向プログラミング
・形式的仕様記述によるプログラミング
・ルール・ベースド・プログラミング（エキスパート・システム）

等のプログラミング手法が対象としてあがっています．つまり，これらの手法を使って「プログラミング手法のケース・スタディ」を体験してみよう，というわけです．できれば，何台かワークステーションを運び込んでみたいとも考えています．

そこでお願いです．あなたが２９才以下でしたら「若手の会 in 八ヶ岳」に参加する資格があります，是非，第６回のこのワークショップに参加して下さい．また惜しくもあなたが３０才以上でしたら，運営スタッフとして参加するか，周囲の２９才以下の「若手」にこのことを教えてあげて，そして来年このワークショップに，是非，参加するよう勧めて下さい．

さて，第６回「夏のプログラミング・ワークショップ」は，これまでにない姿で展開しようとしています．この新生ワークショップの企画・運営スタッフを募集しておりますので，興味をお持ちの方は，ソフトウェア技術者協会・事務局まで連絡下さい．積極的な参加をお待ちしております．

というわけで，次会も多数の応募が予想されます．応募されるかたは，ポジション・ペーパの書き方を参考にして，是非，ソフトウェアの世界を担うべき討論の場でお会いしたいと思います．

次回の「夏のプログラミング・ワークショップ」に御期待ください．

第５回夏のプログラミング・ワークショップ

グループＡ：討論レポート

リーダ	野村　行憲	岩手電子計算センター
	沢田　寿実	ソフトウェア・リサーチ・アソシエイツ
メンバ	小笠原　宗弘	中央システム
	久永　勝雄	システムユニバース
	白田　正司	システムユニバース
	和田　伸一	日本情報開発
	清水　富雄	日本テクニカルシステム

１．はじめに

去る９月２日～９月５日までの４日間，我々は「バグのないソフトウェアを目指して」というテーマについて，あらゆる方向から意見を交わし，あらゆる方向に論議が発散し，収集のつかぬ議論をどうまとめるかに四苦八苦し，最終日の発表にこぎつけたのであった．その苦悩の４日間を以下に示したいと思う．

(1) 第１日目

キーノートスピーチ，および各ＰＣによるパネル・ディスカッションを拝聴し，グループ分けを行う（この日，Ａ組は４人であったが，その夜に１人引き込み，リーダを含めて７人で苦楽を共にすることとなった）．

(2) 第２日目

５人となったＡ組は，野村氏・沢田氏を頭に，討論は開始された．岩手公園→岩手展望台（ここでは，ニス塗り立てのイスに座り，バグ（蚊）と戦いながら『バグ』の偉大さについて語り合った）．

(3) 第３日目

小岩井農場で日陰の寒さに震えながら，言語と環境について語り合う．そしてその晩，深夜０時までかけて，発散してしまった議論をまとめたのであった（おかげでケーキを食べそこねてしまった）．

(4) 第４日目

成果発表（Ａ組は１番目であった）．

２．バグについて考える

まず我々は，この得体の知れない，しかし現実に存在している『バグ』について，不覚にも，真正面から真剣に取り組んでしまった．その結果，いかに『バグ』をなくすかという問題について討論しようとしたとたん，皆腕を組み，眉間にしわを寄せ，沈黙してしまうはめになってしまった．これでは盛岡まできた意味がない，押しても駄目なら引いてみな，という逆の発想から，究極の『バグ』作りへと話はすすむこととなった．

(1)『バグ』をいかに作り込むか

・伝染病的バグ

伝染病的な『バグ』ができないだろうか？　たとえば，一度発生すると，次々に他のプログラムに伝染し，最後にはシステム全体を破壊してしまうような『バグ』．

・忍者バグ

ある日突然出現して，その根源が跡形もなく消滅してしまう忍者のような『バグ』ができないだろうか．

・発見不可能なバグ

絶対に見つからないバグ．アセンブラまたはマシン語レベルだったら，可能性は大きいのではないか．

といったよう分類はできるのだが，作り方に関しては，これだといえる具体的な方法は見出せなかった．『バグ』を作り込むということは，並大抵のことではないことを思い知らされたわけだ．しかし，いかに作るかを考えてみると，現行のデバック方法もかなり効果があるのではないか，と思えてしまう．とはいえ『バグ』は存在している．そう考えてみると，『バグ』の偉大さに改めて脱帽してしまった．そこで，野村氏の「バグをお金に変える」というアイデアをヒントに，バグ神社の設立へと議論はすすんでんでいっ

た．

(2) こうなったらバグ神社

- ８月９日をバグの日とする（なぜ６月４日にしなかったか，考えてみてください）．
- バグ神社のお札の有効期間は，６ヶ月間とする．
- 有効期間中に『バグ』が発生した場合，お札の売上から保険金が支払われる．
- 信心の足りない者，お札の有効期間の過ぎている者には『バグ』が出ます．

３．『バグ』を減らす３つのアプローチ

３．１．プロトタイプ

ユーザとのインタフェイスから生じる『バグ』がある．その対策として有効と思われる手法にプロトタイプが上げられるが，その有効性として次のような点があげられる．

- ユーザの要望についての問題点を，ユーザ側に直接示すことができる点から、早期に確認ができる．
- 要望のあいまいなユーザから，明確な要望を引き出すのに有効（ユーザは目に見えるものしか把握出来ないため，処理上のことをどうこういってみても始まらない）．
- システム設計上，明確な目標となり，設計段階のバグの減少に役立つ．また，開発者のシステム・モデルの確認が正確に行える．
- 仕様書の廃止への道を切り開く（いつまでも現行の作業体系，要望→仕様書→コーディング→システム、というような形に縛られていては進歩がない）．

以上のような有効性があるが，現状ではまだ次のような問題点もある．

- とにかく早く作って，ユーザに提示出来なければならない．そのため，ツール等の開発が必要になる．
- 処理の仕様が明確になる前に，機能面を作成してしまうため，大変危険性が高い（こういう出力が欲しい，という要望だけで機能面を提示してしまい，後になって処理的に不可能だった，というような場合も有り得る）．
- まったく仕様書をなくしてしまうと，第三者がメンテナンスをする際，処理系の内部が把握出来ないという事態も有り得る．

以上の様な問題点もあげられるが，現在プロトタイプ的手法は，各自気がつかないうちに，何らかの形で実行しており，今後ますますプロトタイプ的手法は発展していくであろう．また，進化するプロトタイプということで，一旦基本的システムを納入してしまい，その後，機能の補充，あるいは追加を行っていく方法もある．大型コンピュータではほとんどこのタイプではないかと思う．

３．２．環境から見たバグ

開発環境の不備から『バグ』が発生するという意見がある．とすれば，開発環境が整備されれば，『バグ』が無くなるのではという推論が，当然でてくる．ところがそうでない場合がある，どうして？　こんな奴がいるからだ！

- カリアゲ君な奴
  とにかくへそ曲がり，俺には俺のやり方がある
- 星　飛雄馬な奴
  苦しみがなければ喜びが感じられない奴
- ニコちゃん大王な奴
  その環境が使いこなせない，または誤って理解しているなんだかよくわからない奴

ところで，もし理想的な環境があったら

- 出来ないという言い訳がひとつ減ってしまう．
- プログラマのブロイラー化につながる．

環境はよくすべきだが，完璧なものはよくない

プログラマーの存在価値がなくなってしまう

↓

プログラマの失業なんてことになってしまう

３．３．言語から見た『バグ』

(1) その１

現在の作業手順は，問題発生から計算機（マシン語）へと移行するまでの期間に，やたらと工程がありすぎて，その各工程がすすむ間に，『バグ』が潜む隙間が出来てしまう．そこで，図１に示すように，コードの抽象レベルを上げて，問題＝即コードというようにしてしまえば，工程間

に入り込む『バグ』は減少するのではないか．

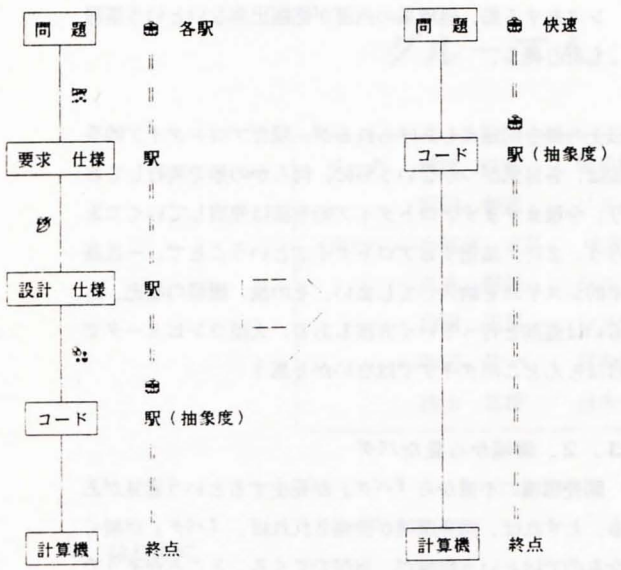


図1

(2) その2

現在使われている言語は，本来の機能を超越し，いろいろな機能へと拡張され過ぎているため，その分『バグ』が発生する率が高くなっているのではないか．その背景には，1つの問題（要求）に対し1つの言語の機能では表現出来なくなって来たことがあろうが，付け焼き刃で言語機能を拡張し，無理矢理表現してしまった結果である．

(a) 例えば COBOL

COBOL(Common Business Oriented Language)

現在最も多く使われている言語，ということで汎用性を持たせ過ぎている．そこで，頭の C を取ってしまい，COBOL 本来の姿に戻す．

OBOL(Only Business Oriented Language)

COBOL 本来の機能に戻ると，現在ではデータベースに置き換えることが可能である．

BOLO

かくして，COBOL の存在価値が無くなる．

(b) 他の言語を見てみると

やはり，その言語本来の目的以外に使われているものが多い．

(c) 言語は TPO によって使い分けるべきだ

しかし，現実の問題はそれほど単純ではない．1つの問題に対し1つの言語では無理がある．

(d) そこで，Multi Lingual Compiler

1つのソース上に，複数言語での記述が可能で，用途に合わせて言語を使い分けることになり，問題点に最もふさわしい言語が採用され，無理矢理拡張されたコーディングはしなくてよいことになる．

4．まとめ

今回初めてワークショップというスタイルの研修に参加したが，参加者のほとんどが同年代ということと，各PCの気さくなアドバイスのおかげで，リラックスした意見交換が出来ことは，大変有意義であったと思っています．

また，盛岡という土地柄のせいか，まるで時間が止まっていたかのように，4日間というものが瞬く間に過ぎ去っていた．普段都会の雑踏の中で仕事に追われている我々にとって，ストレスの解放になり，気分がリフレッシュされたようである．

最後に我がAグループのリーダ野村さん，沢田さんには，大変お世話になりました．メンバー一同に変わって心からお礼を申し上げます．

（**清水富雄**：日本テクニカルシステム）

第５回夏のプログラミング・ワークショップ

グループＢ：討論レポート

リーダ	青島　晴江	高度情報システム
	松下　和隆	三菱電機東部コンピュータサービス
メンバ	阿倍　一夫	東芝
	斎藤　久尚	松下システム・エンジニアリング
	波多　徹	インフォメーション・デベロップメント
	古川みちる	ソフトウェア・コントロール
	安本　秀夫	情報システム・サービス

１．はじめに

グループのメンバは，上記に示したおりである．

今回，バグのないソフトウェアを目指すために， AI はどのように役立つのかということで，対象システムの設定を行った後，エキスパート・システムについての仕様を作成した．

２．ソフトウェア・ライフサイクルとエキスパート・システム（ＥＳ）

ソフトウェア・ライフサイクルにおけるエキスパート・システムの活用を考えた場合，

要求仕様ＥＳ→基本設計ＥＳ→プログラム設計ＥＳ
→プログラム製造ＥＳ→テストＥＳ→運用ＥＳ

というように，個々のフェーズ毎に，何らかのエキスパート・システムが考えられ，各ザブＥＳが次のフェーズにリンクするような，トータル・システムとしてのエキスパート・システムが理想的であると考えられるが，ここでは，テスト・フェーズと運用フェーズにおけるエキスパート・システムについて討論した．

(1) テスト・フェーズにおけるエキスパート・システムの可能性

討論では，デバック時における新人とベテランの違いについての意見が出された．プログラムをデバックしていて，ある不具合事象が発生したとき，ベテランは，ほとんど瞬時（？）に不具合箇所を見付け修正してしまうが，新人は暗中模索の状態に陥る場合がある．

現在新人に与えられるデバック教育は，マニュアル等があっても，結局は徒弟制度的なところがあり，ＯＪＴでマンツーマンで教える場合が多い．そこで，こうした作業を支援するようなエキスパート・システムが作れないかどうかを議論した．まず，そのためには，知識ベースの作成が必要だが，ベテランが不具合箇所を見つける時，どのような思考過程をとって結論にたどりついたかは，本人も考えていないため，その獲得がむずかしい．

そこで，デバックしているベテランの隣に新人をＫＥとして置いて，いちいち聞き出す方法とか，これと同じことをエキスパート・システムで行い，ベテランの知識を獲得する手段などについても意見が出されたが，このような知識ベースを作るためには，適用業務毎に方法・手段が異なることから，ここでの討論としては，適当でないということで討論を打ち切った．

(2) 運用フェーズにおけるエキスパート・システムの可能性

運用フェーズでのエキスパート・システムについては，グループ・メンバの何人かは，業務上の問題点とある程度合致するところがあり，身近なテーマとしてとらえられた．それは，ユーザがシステムの使用中にトラブルが発生すると，夜中でも呼び出しがかかるという切実な問題を持っている人や，トラブルでユーザに行ってみたら，単なるオペレーション・ミスだったということが多いという人が，メンバに何人かいたからである．

いずれにしても，トラブル（？）即呼び出しというので

はなく，ユーザ自身である程度調査できるようなものがないかどうか，というのはメンバ共通のテーマとして考えやすいため，この運用フェーズに的を絞って，エキスパート・システムを考えてみることになった．

なお，期待できるような効果としては，ソフト作成，保守担当者の精神的ストレスからの解放による幸福な生活の実現等がある．

３．対象システムとその条件設定

まず，実際の検討に入る前に，各メンバがエキスパート・システムというものについて，どういうイメージを持っているか，またはどこまで知っているかを話してもらい，リーダの松下氏，青島氏とのＱ＆Ａにより全体のレベル合わせを行った．

次に，どのようなシステムの運用についてのエキスパート・システムを作成するかについての議論を行った．メンバ構成者の分野は，制御関係もいれば，事務処理関係もいて，計算機も，マイコン，ミニコンと範囲が広いため，同じ土俵で討論できるようにするために，「パソコン上で動作するソフトをユーザが購入したが動かなかった」，という状況を設定することにした．

４．システム基本思想

もともとの発想は，不用なトラブル対応を避けたいということなので，このシステムを討論する上で，次の点に重点を置いた．

・ユーザのオペレーションミスを指摘する．
・ユーザの性格，その時の精神状態を把握する入力装置を備えると共に，それに応じて，対話内容を変える．
・推論をすすめていって，ユーザのオペレーションミスではなく，バグと判定してもユーザには知らせずに，そのまま対話を続け，バグ調査用のデータ（メモリダンプや内部データ）を収集し，その後バグであったと報告し，電話かＦＡＸ，郵便でその出力データをシステム保守者に送る．

このような手段により，少しでもこの手の時間の無駄を排し，その分，他の仕事に時間が当てられるようにする．

５．エキスパート・システムのケース・スタディ

前出の基本思想に従い，エキスパート・システムの仕様作成にとりかかった．

５．１．システム概要

(1) 名称

本運用フェーズにおけるエキスパート・システムを，”ドラブルフィルター”と名づけた．

(2) 対象

パソコンを使用した一般的なシステム（ワープロ等）とする．

(3) 構成

本システムは，大きく２つの部に分けられる．

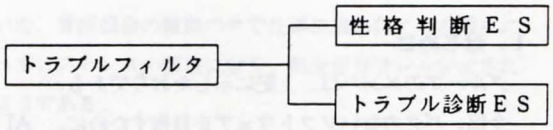


図１

(a) 性格判断ＥＳ

ユーザの手相・人相・血液型・血圧・脈拍を入力とし，その人の性格，現在の興奮状態を把握し，次のステップであるトラブル診断ＥＳの動作時の質問メッセージ，画面構成等が再構成され，マンマシン・インタフェイスに反映される．これでトラブル発生時のユーザの怒りを和らげ，冷静になってもらうのが目的である．

(b) トラブル診断ＥＳ

本システムの中心となる部分である．ユーザに事象を入力してもらい，その事象に対する原因の候補を知識ベースより順次検索し，質問していく．どれにもあてはまらなかったら，バグと判断するが，そのまま対話を続け，原因解明のための情報収集までを行うものである．

５．２．ハードウェア構成

一般的な通信可能なパソコンと同じような構成であるが，次の点が大きく異なる．

・画面は通常は鏡になっており，さらに人相入力用の画像処理部がある．
・キーボード上の右側には，原寸大の手形をした起動スイ

ッチがある．これを叩くことによりシステムが起動するが，この時，手相・指紋・血圧を入力する．

５．３．知識ベースの作成

あるトラブル事象が発生した時，それはバグではないという前提で，メンバの日頃の経験をもとに，その原因となるいろいろなオペレーション・ミスを出し合った．これで各々の過去の経験をもとに，多くの項目が上げられた．この時の項目を本文末の表１に示す．

５．４．マンマシン・インタフェイスの実際

まず，マンマシン・インタフェイスの基本となる性格の分類を次のように仮定した．

・ノンビリユーザ
・短気なユーザ
・パソコンマニアのユーザ
・プッツンユーザ

これらの分類により，以降のマンマシン・インタフェイスが展開する．

・システムを起動すると，画面にお辞儀している人が現れ，「いつもお世話になっております．ご不満の状況をお知らせ下さい」とメッセージを表示する．ただし，短気なユーザ，パソコンマニアのユーザに対しては，この画面は出さずにすぐ次に進む．
・トラブル事象を入力してもらい，知識データベースをもとに順次質問していく．この時，画面の右上に顔マークが表示されており，最初は赤色で，質問が進むにつれ，バグであると判定されると青く（まっ青の意味）なる．
・質問の途中でユーザのオペレーション・ミスとわかった時には，画面に笑顔の人が現れ，「いやあ，人間誰にでもまちがいはありますよ！」と，メッセージが出て終了する．もしバグと判定したときは，「恐れいりますが，もう一度実行してみて下さい」とメッセージを表示し，途中結果やメモリ・ダンプデータ収集を行い，最後に頭を下げた人の画面が出て，「うかがって，対処致します」とメッセージが出て診断が終了する．
・バグと判断した時は，付属のプリンタより，メモリ・ダンプデータと共に，「バグ認定証」なるものを出力する．これは次のように書かれている．「当トラブルをバグとして認定致します．バグ認定証が１０枚たまった場合，ハワイ一週間旅行に御招待致します．」

５．５．効果

このエキスパート・システムの効果としては，当初の目的であるユーザのオペレーション・ミスによる無駄な呼び出し防止の他，バグであった時の情報収集，さらにユーザは，このバグ認定証を収集しようとして，一生懸命バグつぶしに自発的に協力してくれる．つまり，短期間で高品質なプログラムを作成することができ，今回のワークショップのテーマと見事に通じているわけである．

６．終わりに

エキスパート・システムが，バグのないソフトウェアを目指すのにどう役立つのか，という漠然としたテーマに対し，最初メンバは，どう絞り込んでいったらよいのか，わからないまま時間ばかり経過していたが，リーダの御指導によりリラックスしてすすめられた．そのため，アイデアも出て，ここまでまとめることができた．それぞれ環境の異なる業務の人が集まり，一つのテーマについて討論したこと，リーダの方々から大いなる刺激を受けたことなど，短い期間に貴重な体験をさせていただきました．リーダの方々に深く感謝致します．

（阿倍　一夫：東芝）

ソフトが動かない

A．基本システムが立ち上がらない（コマンド入力待ち以前）

(1) ＣＲＴに何も出力されない

・画面ロック・キーが押下されている

・システムＦＤ（ＨＤ）が正しくない

－ＦＤ差込み方向

－ドライブNo

・輝度が暗い

・ＣＲＴの接続コネクタがはずれている

・節電モードでＣＲＴがＯＦＦ状態になっている

(2) ディスクにアクセスしない

・正常なＳＧがされていない

－バージョン（新規でない時

－バード構成（デバイス・メモリ）

－iPlのドライブNo.（ＯＳとバードがマッチ）

－システム定義ファイル(CONFiG.SYS等)

(3) ＯＳにシステム・ダウンした

・フレーク・キーを押下した.

・キー入力が早すぎる

・電源が瞬断した

・AUTO.EXECの内容が正しくない

・静電気・カミナリがあった（磁石）

(4) キー入力が何もできない

・画面ロック・キーを押下されている

・AUTO.EXECで処理

・キーボードのキーが不良（ロック・コーヒーをこぼした）

・キーボードの接続コネクタがはずれている

・ＣＲＴのセットアップが不良

(5) 応答メッセージがない

・画面ロック・キーを押下されている

・ECHOがOHFされている

・プリント，ＥＳＣキーが押下されている（プリンタの電源ＯＦＦ）

・しんぼうが足りない（待ち時間）

B．基本システムは立ち上がった（ユーザ・プログラムを起動した）

(1) プリンタ，ＣＲＴが暴走する

・終端コードを入力していない

・入力データの誤り（バイナリ・ファイルをプリント出力した）

・コントロール・キャラクタを入力した

・ＣＲＴの解像度がシステムと合わない

(2) 処理スピードが遅い

・ロード・アベレージが高い

・プライオリティが高い他のプログラムが動いている

(3) 意味不明のエラーメッセージが出力された

・エラー・メッセージを誤解している（理解が足りない）

(4) 機能が中途半端である

・ユーザがマンマシン・インタフェイスを思い込みで理解している

(5) コマンド・レスポンスがない

・A.5と同じ

(6) 計算の出力が期待値と違う

・入力データミス

・機能の誤解（四捨五入・・・）

表1　オペレーション・ミス

ＥＳがバグと結論する為の情報

○リトライして下さい（再現性の確認）→　再現したら他のサンプルパターン・データを入力してもらい，デバック情報を採取してもらう

○処理時間を測って下さい（処理が遅い）

第５回夏のプログラミング・ワークショップ

グループＣ１：討論レポート

リーダ	渡辺雄一	電力計算センター
メンバ	梅田秀一	ジェーエムエーシステムズ
	工藤勝雄	岩手電子計算センター
	斎藤直彦	ユニテック
	篠原勝美	ディーエム情報システム
	杉野秀美	横浜ファコムセンター
	高久和夫	三井銀ソフトウェアサービス
	徳堂康彦	ソフトウェア・リサーチ・アソシエイツ

１．はじめに

Ｃ１グループは当初，Ｃグループ総勢１４名として成立しましたが，”各自のおかれている環境がさまざまで討論にならないのでは？”という村井氏の意見と，「多勢では夜のフットワークが悪い」の原則にもとづいて，事務系（C1），制御系（C2）の２グループに分割することになりました．したがって，Ｃ１グループは，事務系アプリケーションのＳＥを中心として構成されています．

２．『バグ』とは？

ひとくちに『バグ』といっても，その奥は深いのでありました．

「ここちょっとおかしいんしゃない？」とユーザ，時にはプログラマからも指摘されて，「アッ，それ『バグ』だな」と判断する．つまり，「事象から『バグ』を認定する」というのが，我々ソフトウェア開発者の『バグ』との接し方ではないでしょうか？　では，「そもそも『バグ』ってなんなんだろう」と考えると，その対象は，あまりに広範囲にわたっていることがわかりました．それでまず，２日間という短い期間でもあるので，対象範囲をしぼるために，どのような時，「アッ，それ『バグ』だな」となるかを分類してみました．

表１がその結果ですが，ここで注目すべき点は，『バグ』の組み込まれる主な開発フェイズと，『バグ』によってそのシステムが受ける影響度の関係，つまり，開発初期の『バグ』は致命的であり，以降のフェイズに対しても広範囲に波及する，という事です．また明らかにミスと呼べるもの以外は，「ユーザ」からのクレームによるものである，ということもいえるようです．したがって，Ｃ１グループでは，

『ユーザ』の要求を満足できない

ことを『バグ』ととらえ，設計段階で組み込まれる『バグ』の減少を目指しての討論となりました．

３．システム開発現場の現状

ユーザとはわがままで優柔不断で”どうしようもない”人間の総称か？

「ったくあのユーザはしょうがないなぁ．あの時はこういってたのに」という話をよく耳にしますが，本当にユーザは”どうしようもない”のでしょうか？　「そうだ」の声がきこえてきそうですが，あえて，「ちがう！カモシレナイ」という観点から，現状のシステム開発が抱える問題点を，以下のように分析してみました．

「バグ（ミス）」の様々な発覚分類

事象	バグの組込まれる主な原因	バグの組込まれる開発PHASE	影響度
ｱﾚｯ! ｼｽﾃﾑがのらないぞ（開発ｼｽﾃﾑとﾀｰｹﾞｯﾄﾏｼｰﾝの不適合）	見積り違い ユーザ要求分析不足	設計（基本）	絶望
ｴｯ! このｼｽﾃﾑ構成ではそんなことできませんよ（ｼｽﾃﾑ構成とﾕｰｻﾞﾆｰｽﾞの不適合）	ユーザ要求未確立 ユーザ要求分析不足	設計（概要）	絶大
ここにあの項目が表示されないと使い物にならんナァー（結果物とﾕｰｻﾞﾆｰｽﾞの不適合）	ユーザ要求未確立 ユーザ要求分析不足	設計（詳細）	大
ｱｯ! 入力間違えたら落ちちゃった	プログラムミス	プログラミング	中
コンパイルエラー	ボカ	プログラミング	極少

表1　バグ（ミス）のさまざまな発覚分類

＊問題点1

システム設計の手段として，ユーザの要求をユーザ担当者が「口でしゃべり」，設計者が「耳で聞く」という方法をとっているため，相互の業務知識，電算知識の格差，年齢，性格の違いからくる「話し方」，「聞き方」の食い違いが発生する．

＊問題点2

各企業とも部署の専門化がすすんでいて，ユーザ担当者自身，他部署の関する知識が不足している．また，ユーザ側部署間での認識，要求の相違がある．

＊問題点3

現在のソフトウェア開発は，ユーザの新規業務に関するものが多く，ユーザ側で具体的な要求が確立されていない場合がある．

＊問題点4

ソフトウェア開発の期間が長期にわたるため，法律改正，情勢変化等によるユーザ要求の変更が起こる．

＊問題点5

ソフトウェア開発者の不足，著しい経済，社会情勢の変化に対する即時的な対応の要求，ソフトウェア産業間の競合等で，開発期間の絶対的不足からくる環境劣化．

これらの問題点を整理すると，「ユーザが悪い」という仮説はもろくもくずれさってしまいます．そこで我々は，問題点1～3を改善することにより，問題点4～5も必然的に改善されるであろう，という半ば強引ともいえる仮説をたて，

設計段階でくみこまれる『バグ』の減少は，いかにユーザの要求をひきだすかにかかっている．

という結論に達したのでした．そして，表2に示すような解決策を目指した方法論の暗中模索が始まりました．

問題点	解　決　策
1	ユーザ，システム開発者間インタフェイスの確立
2	ユーザ内部の要求の一元化
3	ユーザ要求分析結果の具体的なフィードバック

表2　問題点と解決策

4．大量メンテナンス時代の到来

大量システム開発は，とりも直さず大量メンテナンスの幕開けである．

システムの開発，この期間は，一般的には運用でのテストの完了をもって終わりますが，同時にメンテナンスの”長く暗い時代”のはじまりでもあります．そこで”メンテナンスという作業は新規開発よりも，むしろ困難な場合が多いものである”，という観点から，現状のメンテナンスが抱える問題点を，以下のように分析してみました．

＊問題点１

最良のドキュメントは，プログラム・ソースコードであるという現実．

＊問題点２

開発時のメンバーが，そのままメンテナンス・チームに残留することはまれで，蓄積された業務システムの知識が生かされない．

＊問題点３

メンテナンスの要求が外的要因で起こることが多く，緊急性をおびているため，ユーザのニーズを煮詰めている時間がない．

今後，各企業とも電算化はますますすすんでゆき，開発形態をかえないとすれば，システム開発者の作業は増えることはあれ，減ることはないでしょう．しかしそれは，新規開発ではなく，メンテナンスという形の作業になるはずです．であれば，これらの問題点の改善も，同時に目指さなければならないという結論に達しました．

5．レビューの効果（アプローチその１）

設計時のレビューは，有効な手段となっているか？（なりえるか？）

レビューの重要性についてはキーノート・スピーチで，山口氏がキーワードとして提示されていましたが，討論の席上でもこれを強調する意見が多くでました．しかし，「現状レビューはうまくいっているか？」という話題になると，メンバーの顔色は暗く沈んでゆくのでした．その理由として，「レビュー時のユーザの確認は儀式化している」という，なんとも悲壮感漂う意見が出ましたが，そうなってしまう理由は明確でありました．

システム開発者が提示するレビュー・シートをみても，ユーザはなんだかよくわからない．

これは決して「ユーザはアホだ！」といっているわけではなく，「ユーザの脳裏に具体的なイメージがわいてこない」ということのほかならないのです．この解決策として，出力のイメージをワープロ等を使い，ユーザのビジュアルな面への働きかけも行われているようですが，レビュー・シートへの文章のワープロ打ち，画面，帳票レイアウトシートの作成だけでも，かなりの工数がかかってしまう現状を考えると，その上何かを作る，というのは大変むずかしいと思われました．

レビューは重要，しかしその方法の抜本的な改善策が必要．

6．プロトタイピング（アプローチその２）

現在，システム開発はすべてプロトタイピングされている？？？

コーディネーターの渡辺さんのパネルの中にでてきたプロトタイピングという言葉は，メンバの頭の中で渦巻いていたようで，一人が口にすると全員がそちらを向いてしまいました．当初，メンバーのほぼ全員が「プロトタイピングという言葉を聞いたことはあったが，その意味をよく理解できない，まして実践したことはない」と考えていました．プロトタイプ＝原型，その意味するところが各自の中で固まっていく過程で，「そういうことなら実際にやったことがある」，「それはシステム開発者側のユーザを無視した先走りではないの？」，また「現状の開発では，ユーザの要求を完全に分析しきってから実作業にはいることはありえない，そういう意味ではすべてのシステムは，プロトタイピングでつくられているのでは？」等の意見がでましたが，いずれにしても，

ユーザの要求を具体的な形で引き出すことができる．

という点では，非常に有効な手段であるという考えが大勢をしめました．ただレビューの項でも述べたとおり，現状の開発形態を変えず（ユーザの要求を聞きながら），ユーザの要求の具現化のためにだけにこれを行うのは，工数面で非常に難しいと思われます．したがって，プロトタイ

プがユーザの要求を吸収し，最終的には完成品になるような方向へ持っていかないと，システム開発者側の作業効率は，かえって落ちてしまうということもありえます．であれば，

業種，適用業務毎のプロトタイプがユーザの要求発生以前に用意されている必要があるはずです．

７．我々に将来，用意されるべき環境を考える

アプリケーションＳＥとはわがままで”ムシのいい”人間の総称ある．

我々はこれまでのアプリケーションをもとに，「以下のような”システムの構築支援(?)ツール”の登場が，今後のソフトウェア産業には必要不可欠であろう」と考えます．

アプリケーション・ソフトウェア開発支援ツール

「MAKASYTE」（まかして！）*

＊仕様１：業務知識の体系化

・企業内での専門分野化が進んでいる現在，ＥＤＰの人間のみと開発を進めるのは危険である．そこで，各分野の専門家の知識を体系的に蓄積し，開発に携わるすべての人が参照することができれば，システム設計の効率が飛躍的に向上する．

・業務知識情報をユーザとともに構築（実際にはカストマイズ）していくことにより，ＳＥ・ユーザ間，ユーザ相互の知識，意識，用語表現の食い違いを解消できる．

＊仕様２：業務，適応業務毎のプロトタイプ（パッケージ）化

・現在，全く新分野の業種のシステム開発はほとんどない．ということは，これから開発しようとするシステムには必ずどこかに（社内であればなお，おいしい）プロトタイプとなるべきシステムが存在している．であれば我々はいちから開発する必要などどこにもなく，そのプロトタイプをもとにユーザの要求を具現化し，カストマイズしていくことで効率を上げればいい．

"APPLICATION SE"→"APPROPRIATION SE"

になってしまう！！

・新規ユーザに対しても，現状難しい”具体的な提案”が可能となる．また開発工数の把握が容易にでき，適正な見積り，プロジェクト管理にも役立つ．

＊仕様３：ユーザとの”息の長いお付き合い”

開発工程からメンテナンス履歴まですべてを管理，指導する”工程管理機能”をもたせ，ユーザをおりのなかにいれてしまう．

８．終わりに

今回のワークショップでのＣ１グループの雰囲気を表現すると，「はじめチョロチョロ，なかパッパ」とでもなるのでしょう．「チョロチョロ」段階では，「ユーザの要求分析はコミュニケーションの確立が不可欠である」と考えていたメンバ全員が，その実現の難しさに頭を痛めていましたが，古代よりの”コミュニケーション構築ツール”である『酒』によって，なんとか「パッパ」になったのは，”討論の活性化”と同時に，”議題の散乱”をも表現しているようです．そのことは，この討論レポートに如実にあらわれていると思います．

『バグ』のない，という言葉は，”システムの信頼性”という重要な意味とは別に，”開発の効率化”ひいてはシステム開発者の”生活の向上”にもつながるものであります．事務系ＳＥの特徴である「ツールは造るものではなく，期待するもの」という感覚をかなぐりすて，今後のシステム開発環境の改善に大いに努力したいと考えています．

最後にスロー・スタータがそろってしまったＣ１グループを，最後まで優しく見守っていただき，”牛乳まんじゅう”まで差し入れていただいたグループ・リーダの渡辺さん，そして今回のワークショップを運営していただいたプログラム委員の方々に，心より感謝の意を表し，Ｃ１グループの討論レポートを終わります．

（杉野　秀美：横浜ファコムセンター）

* Morioka Application Knowledge Administration Systematic Technology

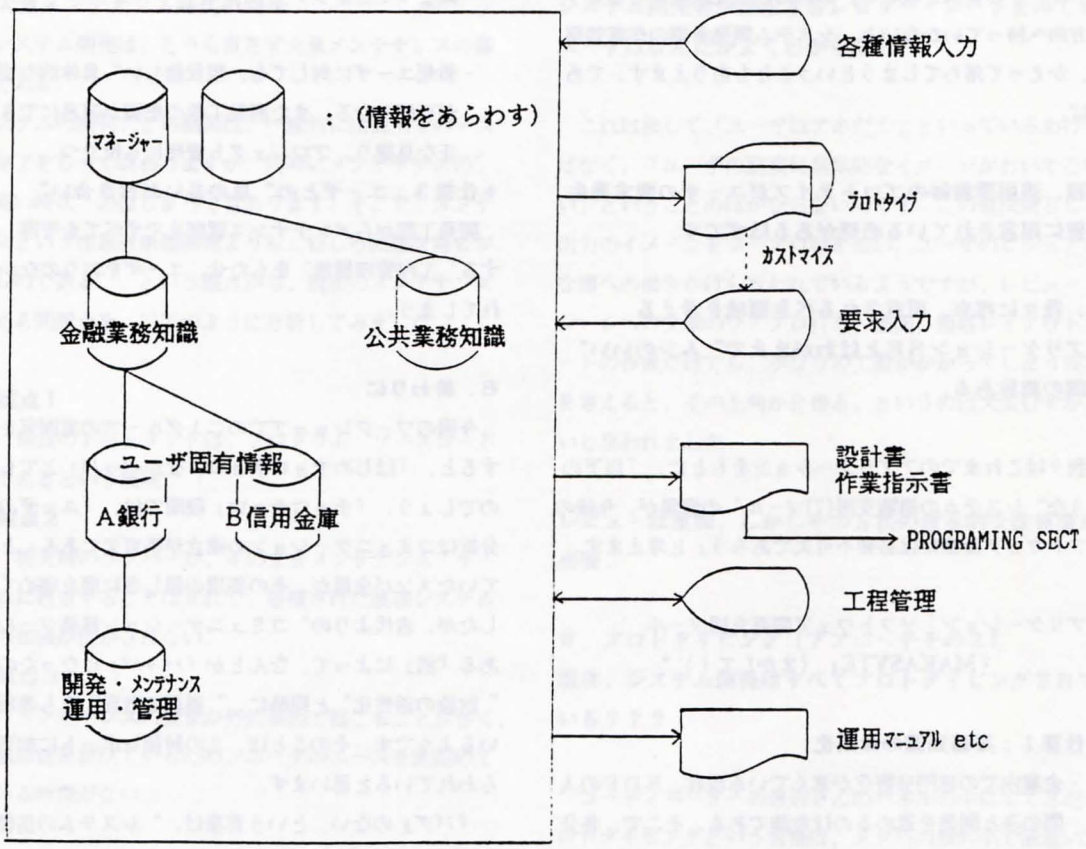


図1　MAKASETE 概要図

第５回夏のプログラミング・ワークショップ

グループＣ２：討論レポート

リーダ	村井　進	ムライ・テクニカル・ライティング
メンバ	石丸洋二	三菱電機コントロールソフトウェア
	上田宣子	セントラル・コンピュータ・サービス
	清水　太	セントラル・コンピュータ・サービス
	戸田倫史	ソフトウェアコントロール
	渡辺正雄	日本システム

はっきり言って我々は現実主義者である．我々は，日々，システムの開発に追われている．我々が現実主義者である由縁は，我々が，自ら開発環境を現実的に改善しようとしていることである．一部の人たちは，我々の考えるソフトウェア開発環境には思想がないと言う．しかし，古来から思想が本当に世の中を改善したことがあっただろうか．古代ギリシャの哲学は，人間の本質をとらえているらしい．しかし，未だに人々はそれを学び続けるに過ぎないのではないか．現実の生活に生かしている人が一体何人いるのだろうか．

フォーマルスッペクは美しい．オブジェクト指向も素晴らしい．我々が考える開発環境の現実的な泥臭さとは，その考える方向性において既に異なるものかも知れない．しかし，「フォーマルスペック，アーメン！！」とお祈りを唱えれば，天地創造のごとく，自動的にソフトウェアが出来上がるのだろうか．「オブジェクト！！オブジェクト！！」と木魚を叩き続ければ，バグという名の煩悩が消え去ってゆくのだろうか．

教会の神父さんだって人間である．たまには，女の人を好きになって苦しむこともある．また，お坊さんだって人である．誰もみていないところで，すき焼きを食べることもあるだろう．理想を説く人間ですら，理想への道からしばしば足を踏み外すのが人間の常である．本当に実現する理想などというものは，存在するだろうか．まるでメシアを迎えるユダヤ人のように，フォーマルスペックについて熱弁するのも良いだろう．しかし，その人たちも，きっと今ごろはLispやCでコーディングしていることだろう．まあ，いいや．フォーマルスペックやオブジェクト指向プログラムミングが，我々のような下々の衆生を救う日がきた時には，有り難く拍手大喝采を贈ることにしよう．

コンピュータは，ただ単に人間に使われるだけの機械ではない．人間の考えることのうち，単純な部分はかなり代行してくれそうである．我々は，今回のワークショップで，コンピュータが，ソフトウェア開発作業をどれくらい代行してくれ得るものなのか考えてみた．

ソフトウェア開発作業で，一体人間は何をやっているのだろうか．コンピュータ全盛の世の中であるから，何をしたかぐらいはユーザの側でも大体わかっているだろう．つまり要求は大体明確である．だけど，どうすれば最も良いかとなると，これはまた別問題である．何もないところからそれを考え出すのは，まさに人間の創造性発揮の場であり，あまりコンピュータのお世話にならなくてもよさそうである．しかし，人間が考えたことを，書いたり消したりして設計仕様書を作成する当たりは，理想的にはコンピュータが人間の脳波でも分析して自動的にやってほしい．現実は，ドキュメント化作業はかなりの部分人間が行っている．

人間が考えた詳細なロジックをプログラムに落とす当たりでは，やや自動化が進んでいる．しかし，相変わらず人間は非常に忙しい．コンピュータは暇である．テストに関しても，自動確認にはほど遠い．場合によっては御丁寧にテストデータまで人間が入力している．ここまでくると，人間はほとんどコンピュータの奴隷に近い．

これらの作業は，コンピュータを主体とする作業である．従って，やりようによっては，コンピュータがもっと自分でやってくれても良いことである．では逆に，プロジェクトチームの管理などの，主として人間を対象とした作業で

はどうだろうか．この点に関しては，コンピュータはほとんど手を貸してくれていない．

ところで，コンピュータが自分でやってくれそうな作業，あるいは手を貸してくれそうな作業を適当に分類して，次のようなマネージャー（支援機能群）にまとめてみた．

・要求分析マネージャ
・設計支援マネージャ
・インプリメント支援マネージャ
・テストデバック支援マネージャ
・対人関係マネージャ
・作業シナリオマネージャ

我々のグループはちょうど６人いるので，このマネージャの中から各自１つずつ選んで考えてみることにした．以下に記述するのがその結果である．

俺（平凡なエンジニア）が朝出社して考えることは，「今日はこれからどんな作業をどういう手順でしようかな」といった１日のスケジュールである（優秀なエンジニアは，既に電車の中で考えている．）俺は，おもむろにワークステーションのスイッチを入れる．すると，ＷＳは即座に「おはようございます」から挨拶を返してくれる．バイオリズムと星占いも自動的に表示してくれる．ついでに，彼女のバイオリズムも表示してくれる．つまらない感情的なスレ違いは俺の人生には似合わない．

今日の予定がスクリーンに現れてきた．俺はなんと多忙なのだ．しかしこれはひど過ぎる．課長の仕事まで紛れ込んでいるではないか．早速課長のスケジュールに組み込んでおこう．しかし，その前に昨夜の課長の帰宅時間を調べてみよう．あまり遅いようなら，かわいそうだから俺がやってやろう．さてと，スケジュールの調整が終わったところで，打ち合わせの予定を該当者に確認しておこう．この辺は，作業シナリオマネージャーが自動的にチェックしてくれるので楽である．おっと，仕事を始める前に，ゲームランドにエントリするのを忘れてはならない．今日は，３時ぐらいに将棋をやろうか．エントリしておけば，自動的に相手を捜してくれる．ついでに，夜は麻雀をやろう．デートするには，彼女のバイオリズムが悪過ぎる．さて，室温，換気，照明の具合いをチェックして仕事を始めるとするか．

今日は，例の家庭用パン焼き機を制御するシステムのアーキテクチャを設計しなければならない．これは，難物である．なんせ，何百万という家庭で使われる電気器具である．絶対に間違いは許されない．おまけに最近は生活レベルが上がったせいか，パンの焼け方に対する注文が非常にうるさいという．狐色に焦げたところの色あいまで指定してくる．パンが黒こげになったら，ソフトウェア・ハウスが責任を取る時代になってきたようだ．とにかく，要求定義マネージャを使って要求項目を確認してみよう．

打ち合わせは，イメージ・プロセッサを使って行った．従って，具体的なイメージは明確である．また，議事録プロセッサで作成した要求項目も，さすがにわかりやすい．とにかくよく喋る目立ちたがりやなおじさんの話は，ほとんど自動的にカットされている．中心点は，しなやかな長い髪の理知的な女性の話で構成されている．歳の頃は幾つだろうか．早く年齢シミュレータを開発してほしい．これを要求分析プロセッサで処理すれば，午前中には要求仕様はまとまるだろう．要求仕様の正当性の検証には，それほど時間はかかるまい．ついでにマニュアルプロッセサにかけて，マニュアルの骨組みも作っておこう．これからは，確実なマニュアル作りは，要求定義と設計のレビューの意味でも重要である．

要求定義が一段落したところで，ノウハウ・データベースを覗いてみよう．よくにたシステムのノウハウがごろごろ転がっている．このオートステーキ・ロースターなどは使えるかもしれない．ステーキをレア，ミディアム，ウェルダンに焼き分けるところの制御が素晴らしい．しかも，複合化設計に従った評価結果は８０点と出ている．設計履歴も十分に残っている．とにかくこれを使ってプロトタイプを作成してみるように星岡さんに頼んでおこう．プロトタイプを使えば，そんなに苦労しなくてもすむだろう．

と，ここまでやったところで，突然左側の画面にイメージェンシ・メッセンジが表示された．どうやら足音センサが課長の足音をキャッチしたらしい．対人マネージャが，「左後方３０°の方向から課長が接近してきます．この足音からは，感情的な興奮状態が想像されます．」と分析して警告している．やばい，と俺は思った．やはり，仕事を送り返したのがまずかったのだろうか．それにしても，ネットワークを使わないで，直接飛んで来るとは古い人だ．何だかんだ言っているうちにお昼になってしまった．右側のスクリーンに，近くの店の今日のお勧めランチメニュー

が映っている．

さて，それでは飯も食ったことだし，システム・アーキテクチャの設計を始めるとするか．ます，要求仕様書をアイデア・プロセッサにかけて，システムの論理的な動きを確認してみよう．それが解ければ，アーキテクチャの概要は決まってしまう．ただし，動的に変わる制約条件には注意しなければならない．焼き加減は，パン粉に含まれる水分まで考慮して制御する必要がある．

具体的な設計作業は杉下さんに任せてしまおう．レビューは，３日後に行うことにする．レビューの最重要課題は，要求仕様との整合性の確認，それに星岡さんのプロトタイプとの比較検討である．さらに，もし結果が良いようなら，早めにインプリメンテーションを行いたい．従って，共通部のモジュール化情報や，ロジック構造とデータ構造の選択に関する制約条件なども明確にしておいてもらおう．このあたりの具体的なプロジェクトの進め方は，設計支援マネージャ，それに作業シナリオ・マネージャに任せることにする．少々意見の食い違いが発生したところで，対人関係マネージャが，経験や性格など種々の要素を考慮して慎重に選んでくれた人たちである．間違っても，喧嘩別れにはならないだろう．

トイレの帰りに隣のマシンを覗いてみた．俺のマシンとは，まるで異なる世界が広がっている．個人の興味とはこうも異なるものなのだろうか．その中で，青海さんが楽しそうにコーディングをしている．さすがに，マルチランゲージの青海さんと言われただけあって，各部分部分に最も適したプログラミング言語を自由自在に使ってコーディングしている．これだけのプログラミング言語を使えるのに，どうして日本語しかしゃべれないのだろう．コーディングと言っても，かなりの部分は設計仕様から自動的に生成されてしまう．人間の作業は，本当に複雑で高度な部分に限られている．出来上がったプログラムは，ホログラフィを使って立体フローチャートにして眺めることが出来る．しかし，この機能は，プログラムのスパゲッテイ化を促進するだけだという批判も強い．あまり油を売っていないで，自分のマシンに帰って仕事をしよう．夜は楽しい麻雀が待っている．

なんだか疲れたなと思ったら，ゲームランドからメッセージが届いた．マシンが俺の疲れを自動的に察知して，将棋の相手を見つける優先度を上げてくれたらしい．「なんだ，相手は藤野原だ」．俺は一瞬ためらってしまった．こいつは強い．なんせ世の中の全ての物をクラスに分類して，その間の関係をフォーマルスペックで記述しようとしている奴だ．奴の頭の中では，将棋の駒とその配置は，お互いにフォーマルに記述できる関係を持った独立なオブジェクトとして定義されているに違いない．でも，よく考えてみれば，これは当たり前のことである．問題なのはフォーマルスペックで記述された将棋のルールや制約条件，それに実際の状況を判断して，最適なメッセージを特定の駒に送る機能である．きっと奴の頭の中には，推論エンジンが詰っているのだろう．それに対して，俺は直感しか持ち合わせていない．要するに，パターンマッチ人間である．従って，経験したことのないパターンが出てきたらお手上げに近い．なんだか，ソフトウェアハウスのたたき上げ管理職みたいで悲しい．ゲームランドからのメッセージによると，今夜の麻雀の面子も決まったようだ．そろそろ今日の作業の報告書を作成しておこう．作業シナリオマネージャが，今日の作業履歴をまとめてくれている．さらに，対人マネージャが，報告書の宛先毎に内容に修正を加えてくれる．特に，課長の報告には，「課長の御忠告に従って作業を進めたところ，」などという，いかにも課長が喜びそうな文章を，随所に自動的に入れてくれるのが非常にうれしい．このフレーズを自分で書くのは，精神衛生上好ましくない．また，ユーザに対する報告書では，知られてはまずいことは全て自動的にカットしてくれる．これは，過去，現在，未来にわたって，自分の発言の矛盾を防ぐ意味で重要な機能である．しかし，この対人関係マネージャーを使い過ぎると，通常の人間的な交際が出来なくなるので注意が必要である．

さて今日の事後ともそろそろ終わりだ．帰りに通りがかったら，田辺さんが，今日もこう惚の表情でデバックを行っていた．なんせマシンが恋人の彼である．デバック・マネージャとの二人三脚で，テストベッド，変更余波チェック機能，データフロー機能，アニメータ，修正情報自動作成機能などなど，ありとあらゆる機能を使いこなして，会社中のソフトのテストを行っている．彼の得意技は，要求仕様から設計仕様，ソースプログラムまでの，整合性のチェックを行うことである．場合によっては，「本当にこれで良いのか」と，ユーザに直接電話で確認するらしい．全くどっちの味方か解らない．しかし，きっと世の中には，こういう人が必要なのだろう．また，ダイヤルアクセルで実行速度を変えながらプロブラムを眺めていると，一発で

バグが解るというから神業である．それにしても，明日の朝のバグ・レポート・メイルが恐い．　「気を付けよう，注意一秒，バグ一生！！」

最後にハードウェアの仕様に触れておこう．外から見えたところは，ゲームセンタの富士スピードウェイを思い出してくれればよい．それを一回り大きくしたようなものである．前方上下，それに左右にはスクリーンが張り巡らしてある．ＣＰＵ？，メモリ？　話によると，常温ジョセフソン素子かなんか使っているらしいが，いくら使っているかは定かではない．このマシンに座っていると，不思議と肩こりが直ってしまう．どうも強力な磁気が発生しているらしい．

（村井純）

第５回夏のプログラミング・ワークショップ

グループＤ：討論レポート

リーダ	佐原　伸	ソフトウェア・リサーチ・アソシエイツ
	月岡義範	中央システム
	佐久間まゆみ	神戸コンピュータ・サービス
メンバ	日高康志	デンサン
	神代久稔	日本コンピュータ・システム
	笠川　満	富士通
	小田和敏	呉電子計算センター
	桜井　洋	郵船情報開発
	鈴木宏明	岩手電子計算センター
	越中一夫	中央システム

１．グループ討論の経験

(1) 第１日

この日行われた個性豊かなプログラム委員の方々のパネル発表を，緊張に緊張を重ねて聴き，中でも佐原氏の「オブジェクト指向的知性」，月岡氏の「仕様書不要論的バイタリティー」，佐久間女史の「遊園地的ドキュメント論」に引きよせられ，バグのように発生したのが，我がＤグループである．

(2) 第２日

前日の熱気も冷めやらぬまま，２日目を迎えた．まずは自己紹介からはじまり，それぞれが持っているバグ論の激突，と思いきや，我々Ｄグループは私も含めメンバ全員は，冷静沈着，かつ学習心旺盛のためか，ＰＣの話についつい耳を傾けっぱなしであった．がしかし，そこはワークショップ経験豊かなＰＣと，昼食のわんこそばの満腹感とに導かれ，除々にではあるが，バグについての打ちとけた話し合いとなった．当グループのメンバーは，ほとんどがCOBOL を中心とした事務計算ＡＰ開発に従事しているため，どうしても討論の矛先はユーザに向いてしまった．このままでは，日頃のグチで終わってしまうという危機感がメンバ全員を襲った．

そこで地元盛岡の雄，鈴木氏のはからいで環境再定義を試みることになった．環境設定を変えると，アウトプットも変わるという業界の迷信を信じ，ふだんからみんなが追い求めているよりよい環境へとタクシーを走らせたのである．そして，その夜全員が見た盛岡の夜景に，将来のエンジニア像をかさね合わせたのである．

(3) 第３日

まちにまった日が来た．今日が納期である．ホテルのロビーに集まったメンバの顔には，納期が迫ったエンジニア特有の真っ赤な目が光っていた．前日の環境設定が成功したのか，よりよい環境を求め続けるエンジニアの特性は，我々を小岩井農場へと導いたのである．木かげの下で討論している他のグループを横目に，まだ納期には時間があるというふだんの思考回路を確認しながら，よりよい環境を満喫するＤグループであった．

昼食後，よりよい環境は必ずあるというあくなき追求はとどまるところを知らず，同じ技術者の待つ手作り村へと向かったのである．そこでメンバが見たものは，完全に分業化，標準化された南部鉄生産システムであった．鉄というメッセージを，次から次へとオブジェクトに変えていく過程に，オブジェクト指向の将来を見たのである．

盛岡に来てよかったと，メンバ全員が確認しあったのはいうまでもない．そして会場へ戻った我々は，この日までに各メンバの中で確信を得たバグのないソフトウェアについてのまとまらないまとめを深夜まで行ったのである．

(4) 第４日

なんとか納期に間に合ったというあんど感のせいか，メンバの目はますます赤く光っていた．そして発表ははじまったのである．

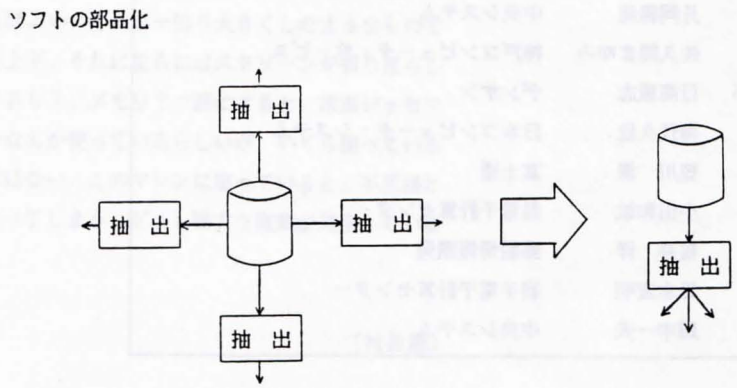


図1 1st Approach - Software

・パッケージでは対応できない類似業務の部品化
・ファイルの種類・定義の一元化
・ファイルアクセスのモジュール化
・ファイルアクセスにオブジェクト指向を取り込む

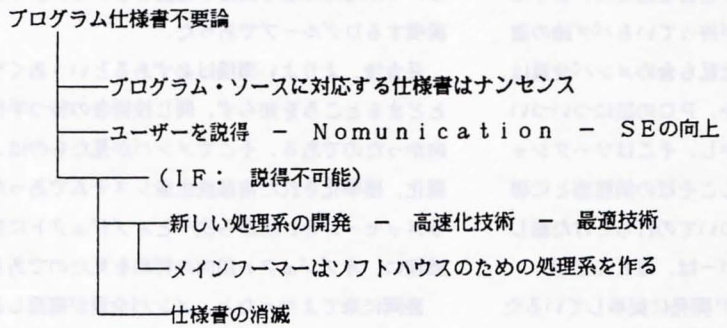
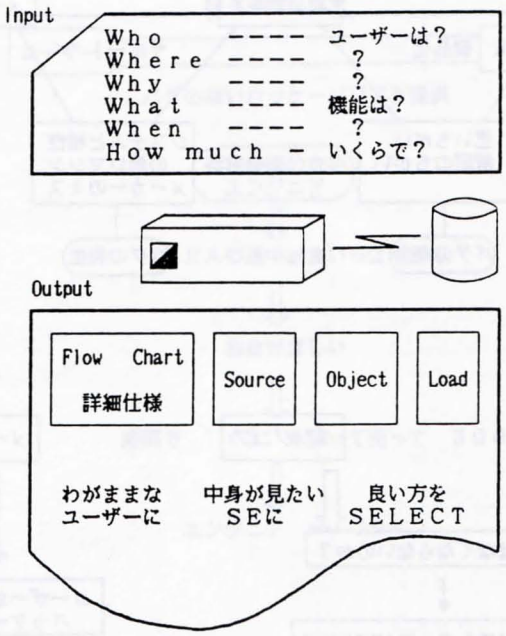


図2 2nd Approach - Documet

Tool Genarating Machine



こんなMachine ができればToolよりもAPそのものを作る！

Bug Learning Machine

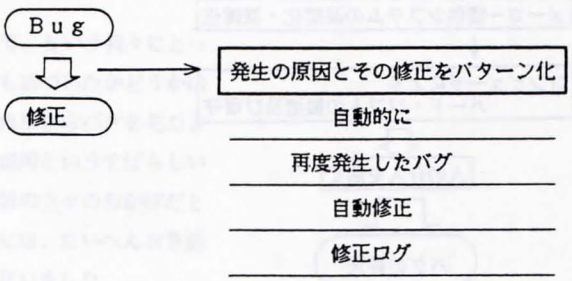


図3 3rd Approach - Hardware

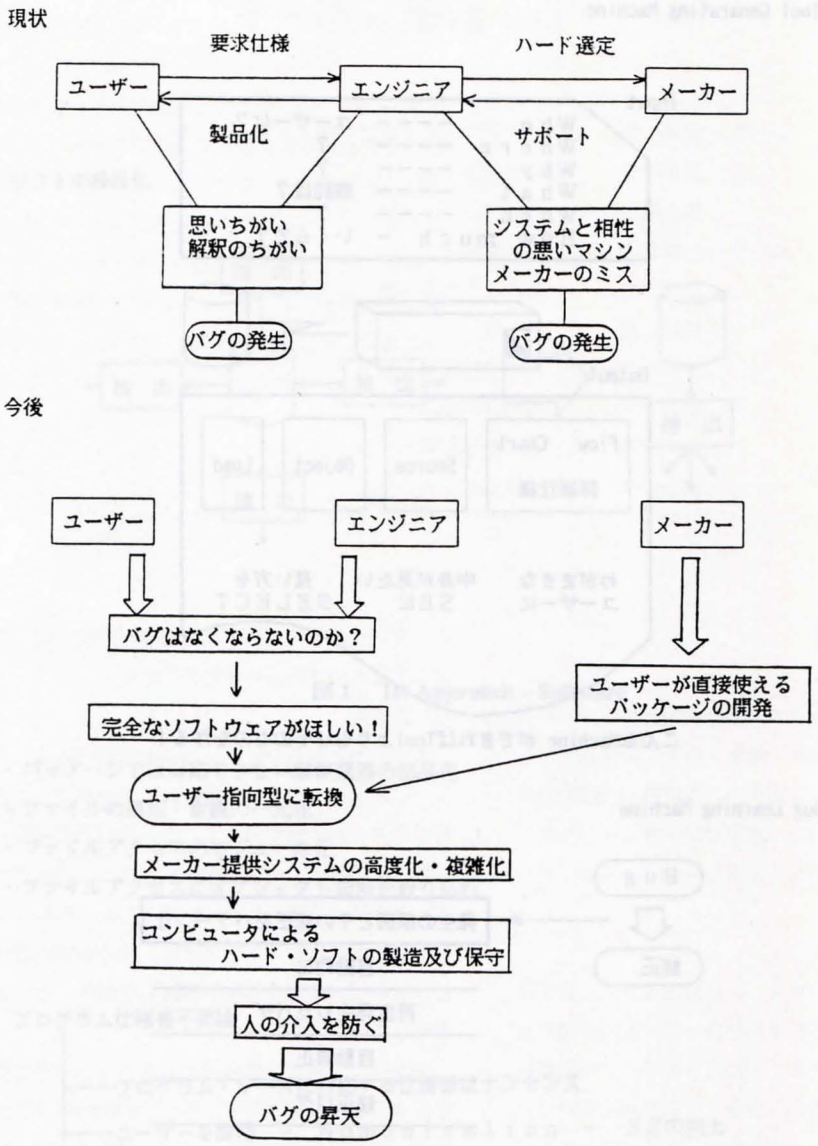


図4 Final Approach - Human

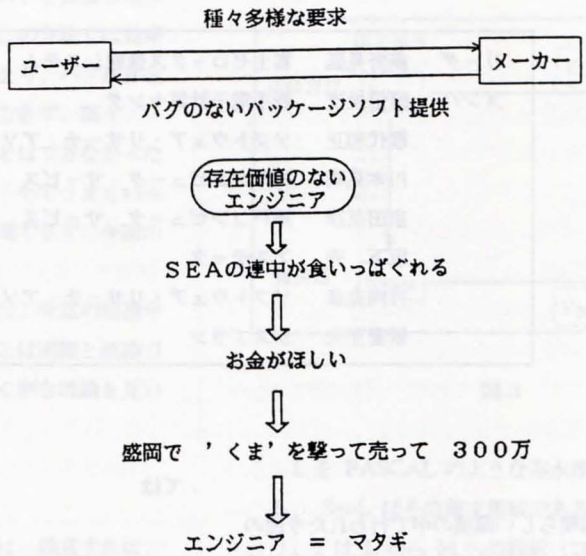


図５ 将来

２．終わりに

バグのないソフトウェアを目指して，という我々にとって永遠ともいえるテーマに，少しでも近づいたかどうかは疑問ですが，少なくともいろいろな角度からバグを見たような気がします．これもひとえに，盛岡というすばらしい環境と，実行委員会，プログラム委員の方々のおかげだと思います．また，地元盛岡の鈴木氏には，たいへんお世話になりました．どうもありがとうございました．

（越中一夫：中央システム）

第５回夏のプログラミング・ワークショップ

グループＥ：討論レポート

リーダ	藤野晃延	富士ゼロックス情報システム
メンバ	阿倍俊次	岩手電子計算センタ
	歌代和正	ソフトウェア・リサーチ・アソシエイツ
	川本晃司	神戸コンピュータ・サービス
	京田英彦	神戸コンピュータ・サービス
	坂下　秀	アステック
	竹内克彦	ソフトウェア・リサーチ・アソシエイツ
	新妻正夫	シスプラン

１．はじめに

岩手は盛岡という素晴らしい環境の中で行われた今回のワークショップの目的は，現場のソフトウェア技術者の最大の悩みともいえる「バグのないソフトウェアの実現」という命題に対して，その開発環境の構築および改善について，また，設計やプログラミング中に発生するバグの認識について，改めて考え直し，方向づけを行うことであった．

だが一口にバグといっても，たくさんの種類のバグがあり，また，開発環境やプログラミング言語によっても違ってくる．さらに現状は，開発しているシステムのほとんどがオーダ・メードであり，バグの発生する場所は不特定である．

このような困難な現状に対し，我々Ｅグループは，「本当にバグのないプログラムを作ることはできるか，そしてその方法は？」ということについて，理論的な側面から考えてみることにした．

２．アプローチ

先ず今回のワークショップのテーマをどうとらえるかを，バグに注目してプログラムの分類をすることにより明確にしようと試みた．

分類の仕方としては，まずプログラムを，

- バグのないプログラム
- バグのあるプログラム

の２つに大別し，さらに「バグのあるプログラム」については

- バグの多いプログラム
- バグの少ないプログラム

に分けることとした．この関係は図１のようになる．

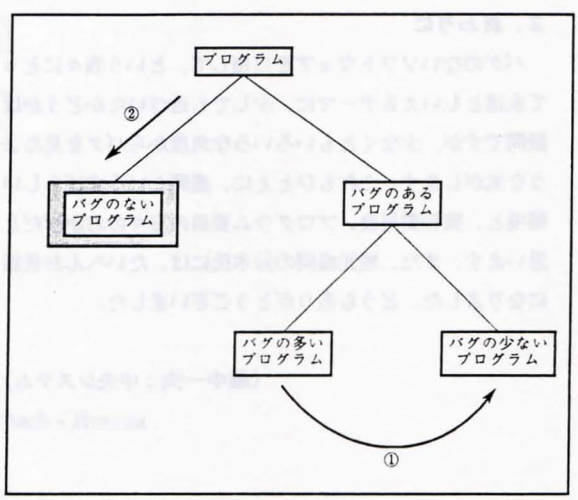


図１　バグとプログラムの関係

ここで述べられている「バグのあるプログラム」とは，ソフトウェア技術者により開発・テストされた，一般の（正常に動作するであろう）プログラム全てを指す総称である．その中には，バグの多いもの少ないもの，さまざまな

プログラムが含まれる．

ソフトウェア技術者達は，日夜（本当に夜も眠らず！！）ひたすら，ただ体力にものをいわせて，「バグの多いプログラム」から「バグの少ないプログラム」を目指した作業（図中□）に明け暮れている．しかしこの方法では効率も悪い上，プログラムが正しいこと，つまり，バグが存在していないということを証明することはできず，高々，「少なくともテストではバグを発見することはできなかった」といっているに過ぎない．このアプローチでうまくいっても「バグの少ないプログラム」しか実現できず，今回のテーマを満足することはできない．

Eグループでは，上記のような考察から，今迄の理論や作業方法では，今回のテーマを考えることは困難と結論づけ，図中□の作業を可能とするべく，全く別な理論を見いだすことをスローガンとした．

3．その理論とは！？

まず求める「プログラムの正しさ」とは，換言すれば，プログラムの意味的な正しさであり，このようなプログラムの意味に付いての理論としては，以下に示す3つの形式的意味論がよく知られている．

- 操作的意味論 (Operational Semantics)
- 表示的意味論 (Denotational Semantics)
- 公理的意味論 (Axiomatic Semantics)

A-calculus に基ずく表示的意味論は，スコットの提唱以来，さまざまな研究が続けられ進展を見せており，また適当な資料（中島玲二：著，「数理情報学入門」）も手もとにあることから，これを採用した．その目的とメリットを簡単に示すと次のようになる．

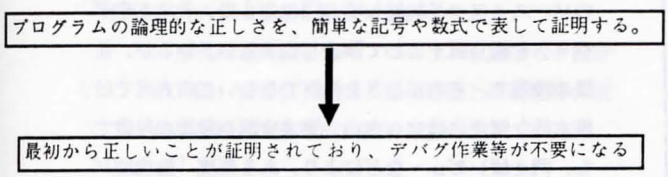


図2

3．1．形式的意味論とは

ここで参考までにプログラムの形式的意味論について，その概略を示しておく．詳細は資料1を参照されたい．

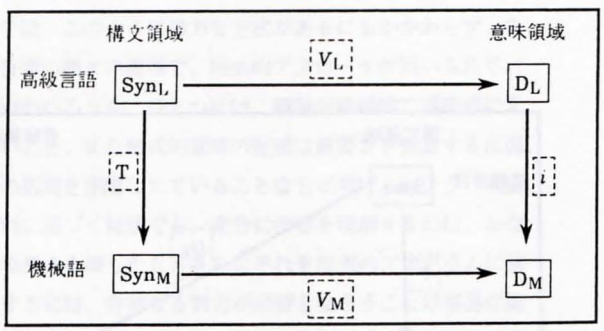


図3

Lを PASCAL のような高水準プログラム言語とする． SynL はその構文領域である． M は機械語であり，T は L から M への翻訳（コンパイル）である．数学の言葉では，T は SynL から SynM への関数として定義される． L の設計の意味論的側面は，L の意味を意味関数VLと意味領域 DL を与えることにより決まるとする． M に対してはその意味は VM と DM によってもともと与えられている． VM と DM は M のプログラムが実行される計算機システムにより決まる世界の記述である．つまり実際のプログラムの実行空間での動きである． DL から DM への関数 i は DL という抽象的領域を DM へ埋めこむことにより表現できる．

難しい言葉で述べたが，ここで言いたいのは，プログラムの意味（＝動き）を，数学的表現による意味論により表すことができるということである（スコット理論）．つまりバグのないプログラムとは，すなわち，作成者（あるいは仕様作成者）の意図する通りに動くプログラムのことであり，プログラムの動きを数学的な表現である関数としてとらえることで，その正しさを証明することが可能となる．

3．2．現状では

振り返って現状のソフトウェア開発を眺めてみると，明示的に意味領域を仮定し，プログラムに対して計算機システムから独立した意味記述を与えるようなこと

は一般的にはされていない．つまり DL が存在せず，下図のように VL と UL で置き換えているかのように見える．

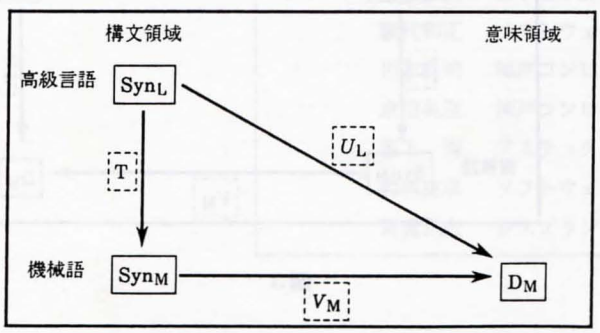


図4

しかしこれでは，ソフトウェア技術者がプログラムの意味も考えず，ただ単に形（構文）にのみ拘泥しているようで，少々誇張がある．またULでの実現では，個々の計算機システム DM への依存が大きくなり，もはやSynLを高級言語と呼ぶことさえ覚束なくなる．実際においては，Lに対しあまり明確ではないものの，なおかつある程度は，DM にも依存してしまうが，何らかの意味（図５中の’？’の部分）を想定しているものと考えるのが自然であろう．

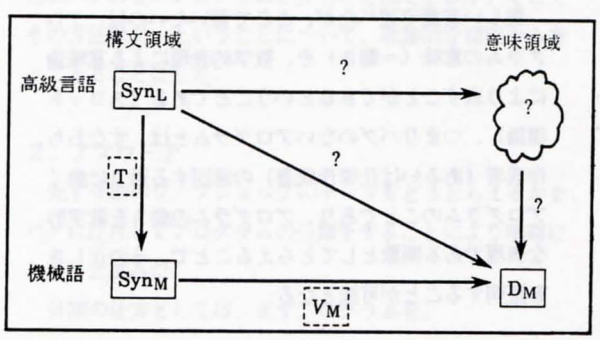


図5

つまり，プログラムの意味を頭の中で考えながら（設計概念），意図する動きを実際の計算機上で行わせよう（実現概念）として，プログラムを作成している状態である．プログラム言語のマニュアルに散見されるが，設計概念と実現概念が，互いに入り交じった記述がされていることがよくある．これは上のような状態を反映した典型的な例として挙げられよう．

３．３．バグの階層→形式的意味記述の位置

開発プロセスの階層の注目して，現状のソフトウェア開発で発生するバグの種類を，それが開発プロセスの階層のどこに起因するかにより分類してみると，図６のようになる．

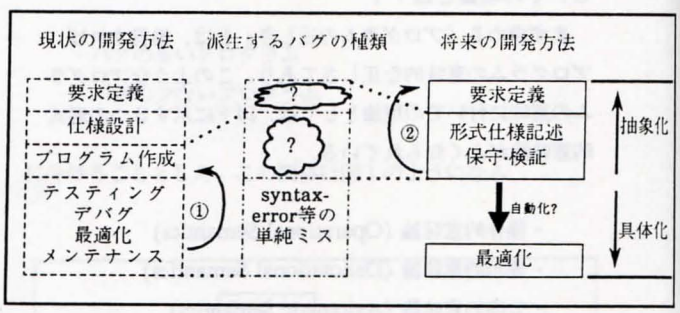


図6

図６に示すように，現状のソフトウェア開発では，要求分析・定義から保守にいたるまでの各段階で入り込むさまざまなバグを，ほとんどの場合，プログラムのソース・コードに対して，□のように，テストやデバックを繰り返すことで除去しようとしているが，仕様の段階で，その正しさを保証できないこの方式では，根本的な解決にはならない．要求定義や設計の段階でも，例えばレビューなどにより，ある程度「論理的バグ」を検出し除去することもできるが，それは仕様の「正しさ」を保証するわけではなく，また　syntax-error 等の単純ミスによるケアレス・バグの除去は，確かに開発環境の整備で解決できるが，プログラム作成以後の下流工程においてデバックでは，仕様段階以前に入り込んだ論理的バグの検出は非常に難しくなり，

コストもよりかかる．さらにその保守も，より一層困難になる．

つまり，仕様の段階で正しさが確認できないことが，全ての原因と考えられる．

一方，形式的仕様記述で行えば，その仕様を基に，半自動あるいは自動的に目的のマシン・コードまで変換することもできるであろうし，抽象度の高い仕様の段階で論理的な矛盾を検出でき，□のレベルでのループで矛盾解消を行える．このことは，要求や稼働環境などの動的変化に対し，仕様レベルで迅速な対応ができることをも意味する．

４．形式的意味論のメリット

形式的意味論のメリットについては，実際にその実用性を充分にチェックしてから評価することが，ここではとりあえずスローガン風に簡単にまとめておく．

(i) プログラム検証や仕様に数学的基礎を与え，正確な議論を可能にする．

プログラムの正しさとは，プログラムがその作者の意図通りに働くことであり，証明が可能となるためには，作者の意図（＝仕様）やプログラムの意味が，数学的議論の対象として取り扱えなければならない．これはプログラムの検証，デバック，合成等の自動化への前提でもある．

(ii) プログラム言語の設計者，実現者，使用者に共通の理解の場を与える．

形式的意味論により，実現が設計に忠実か否かの判断に基礎を与えることができ，またプログラムの保守や修正，マニュアルの記述等の正確さの保証や自動化への可能性を開く．

(iii) プログラム言語の設計に指針を与える．

形式的意味論をもつプログラム言語なら，設計の欠陥，妥当性，改良，他の言語との比較等，現象論を超えて合理的論拠を保証する．

５．結論

「バグのないソフトウェアの実現」に対し，形式的意味論を中心とするアプローチをとったEグループの討論は，盛岡の岩山展望台での円座を組んでの輪講を皮切りに，小岩井牧場の木陰での議論，角館は武家屋敷の黒塀前での沈思と，時空を超えて，従来のソフトウェア開発に対する我々の考えを恰も暴風雨のごとく粉微塵に吹き飛ばし，唯一，表示的意味論が進むべき道であることを顕示しつつ，Semantics-Engineer （！？）なる将来の花形職業の出現をも予感させた．

では，このように強力な方法があるにもかかわらず，なぜ日常の我々の環境で，形式的アプローチが用いられていないのだろうか．ひとつには，議論が超越的で現実感に乏しいこと，また形式的意味の記述は厳密さを保証する反面，その応用を困難にしていることなどが挙げられよう．実際，資料に基づく輪講でも，充分に内容を理解するには，かなりの努力を要したし，さらにそれを現実のプログラムに適用するには，非常なる努力が必要となろうことは容易に推察できる．あるいは困難かも知れないと思わせるほど「手強い」ことは確かである．

しかし最近になって，この分野は著しい進展を見せ，特に欧州での研究は盛んであり，例えば， VDM(Vienna Development Method) などは， Ada の formal-definition を与えるのに用いられ，実用的なレベルでの形式的意味を記述が可能であることを証明した．また米国では， Scheme（LISP の方言の一種）などにも表示的意味が与えられている．

何時までも，現在のように「正しいこと」を証明する術のない直感的なソフトウェア開発の手法に拘泥することなく，表示的意味論のような厳密なアプローチがあることにも充分注意し，その研究成果を積極的に取り込んでいうことが必要なのではないだろうか，少なくとも AI やオブジェクト指向と同様，注目に値するであろう．

最後に，現地での討論に適当な場所の確保等，ローカル・アレンジメントに奔走してくれた岩手電子計算センタの阿倍さん，また「数理情報入門」の輪講時，難しい理論に付いていけなくなりそうになる我々を， IOTA システムでの経験をもとに，なんとか理解できるよう，平易な言葉で解説してくれたアステックの坂下さん，本当にご苦労さまでした．

[References]

［１］「数理情報入門」，中島玲二，朝倉書店，１９８２

［２］"The Specification of Complex Systems",B.Cohen,W.T.Harwood,M.I.Jackson. Addison-Wesley,1986

（川本　晃司：神戸コンピュータ・サービス）

第１回ソフトウェア教育に関するワークショップ

全 体 報 告

君島 浩
富士通
杉田 義明
ソフトウェア・リサーチ・アソシエイツ

１．はじめに

念願であった，はじめてのソフトウェア教育に関するワークショップを，山梨県八ケ岳のふもとで，この１１月に開催することができました．事前の案内や準備不足もあって，予定していた参加者に集まっていただけるかどうか，不安で一杯の気持ちでしたが，第１線でソフトウェア教育に携わっている２２名の方々に参加をいただき，活発な意見交換や討論が，日夜を通して，フォーマルにまたはインフォーマルに行なわれ，充実したワークショップになったのではないかと思います．

もともと，ＳＥＡの教育分科会では，各社の教育担当や技術移転を担当されておられる方々が集まって，情報交換や関心あるテーマに関してのインフォーマル・セミナー等を，地味ながらも２年間に渡って，着実に行なってきています．そのような活動のなかで，もっと本格的な討論を行ない，いろいろな角度から活動を眺めてみるという狙いで，ワークショップを開催しようという提案が昨年ありました．それがようやく実現にこぎつけられた，ということです．

こうした状況の中で，記念すべき第１回のメイン・テーマとして，「実践的ソフトウェア技術者教育を考える」を取り上げました．これは，現在の企業などにおけるソフトウェア教育の諸活動を，さまざまな側面から見直し，より実践的で，かつ効果的な教育活動を実現するための指標を考えようとしたものです．

２．プログラム

ワークショップのプログラムは，５つのセッションで構成されており，各セッションのテーマと目的は以下に示す通りでした．

(1) セッション１

(a) テーマ

導入教育・新入社員教育

(b) 目的

どの企業においても重要視されている，新入社員やフレッシュマン・トレーニングなどの初期導入教育について，カリキュラム，教育体制，インストラクタ，スキル，運営方法，そのほかをとりあげ，さまざまな角度から討論をする．

(2) セッション２

(a) テーマ

教育の効果測定・キャリアパス

(b) 目的

古くて新しい問題である教育効果の測定について，ソフトウェア教育の現場ではどのようにとらえるのがよいか，現実的な解決策を考えます．また，キャリアパスの設定や関連してのスキル評価について，各社の経験や意見を交換します．

(3) セッション３

(a) テーマ

技術移転・教育ノウハウ蓄積

(b) 目的

生産性や品質改善を行うためのソフトウェア技術の開発現場への移転は，現在もっとも重要な課題です．そのため，各社においては事情に応じてさまざまな活動が，教育部門や技術スタッフを中心に行われています．技術マップを作成したり，エンドユーザを巻き込んでの技術向上運動などはその例です．ここでは，技術移転活動をいろいろな角度から考え，そのノウハウを，次のステップに活かすための具体的な方策を検討します．

(4) セッション４

(a) テーマ

採用・教育の諸問題

(b) 目的

教育部門で活動されている方は，何からの形で採用に関連されているようです．最近の慢性的な人材不足は，採用活動の問題へと直結しており，企業における教育の原点は，まず高度なスキル要員の採用であるとのスローガンは，現在の所むなしい状況です．ここでは，夜のセッションらしく自由な雰囲気のもとで，日頃の活動を全く別の立場で捕らえ，ザックバランに議論したいと思います．なお議論の内容は，このセッションに限りオフレコにします．

教育活動は，企業戦略と密接に連結する必要があることは当然であります．最近の技術動向や，企業を取り巻く経済環境から判断して，これから重要な検討項目がいくつか考えられています．ここでは，それらの課題をスピーカから提案していただき，これからの新しい教育の姿について討論することにします．

(5) セッション５

(a) テーマ

企業トレンドと教育

(b) 目的

教育活動は，企業戦略と密接に連結する必要があることは当然です．最近の技術動向や，企業を取り巻く経済環境から判断して，これから重要な検討項目がいくつか考えられています．ここでは，それらの課題をスピーカから提案していただき，これからの新しい教育の姿について討論することにします．

３．討論後

５つのセションを通じて結論するなら，先行企業に学ぶこと，教育部門が企業の各機構と直結すること，企業を変化させる教育部門であることが，大切だといえるのではないかと思います．小児科医がほとんどすべての病気を扱うように，教育という分野は，技術や経営の多くの分野に関係します．そのため，２泊３日のワークショップでの議論では，時間は全く足りませんでした．

日本の学会には，ソフトウェア教育に関する分科会はおそらくないと思います．今回のワークショップは，ソフトウェア教育を正面から（？）扱っているおそらく日本で唯一の活動グループであるＳＩＧＥＤＵ分科会の活発化に貢献したということだけでも，十分意義があったと考えます．

各セッションの発表，討論の要約と私感を述べてみます．

(1) セッション１

中小ソフトウェア・ハウスや，システム部門のソフトウェア導入教育が遅れていることは，さまざまなところで指摘されているが，この種の会合で，先行企業から，刺激とノウハウを受けるのが有効であると思う．

適性検査や情報処理技術者試験については，必要論，不要論，要改善論が出されたが，ワークショップ参加者の意見では，必要悪論が主流を占めた．また，情報処理試験合格の報奨が数十万円という例には，一同どよめいた．

プログラマ不要論が提示された．新人もＳＥであって，プログラマという職級は不要であり，ただしプログラミング教育は必要という意見が多かった．基礎的な導入教育「しか」できないとか，「さえ」できないという企業も多いだろう．本来は一人前の技術者とは一体何か，そのための中堅社員教育は何かという点が重要である．その問題の解決が，新入社員教育の反省にもつながると思う．

(2) セッション２

山口氏は，ＳＥに対する新しい分析を提示した．阿部氏は，ＯＪＴ目標管理や先輩・後輩の接触に力を入れた生涯教育体系を紹介した．加藤氏，企業教育や専門学校教育の経験に基づき，全く新しい企業教育体制を作るという計画を紹介した．

セッション１と同じく，プログラマ→ＳＥというキャリアパスはおかしいという声が強い．３０（３５）歳定年説については，ワークショップ参加者のほとんど「定年」を過ぎており，「人によっては第一線でやれる」，「終身雇用だから問題ない」という意見が多かった．しかし，「若年層の減少対策はどうするのか」という逆の見方には，有力な解答は出なかった．教育担当者には，当然若い人達もいるわけで，次回から若手の担当者たちにも是非とも参加してもらいたい．

(3) セッション３

大浦氏は，企業間・企業内の標準化不足の悩みを提示した．大久保氏は，「実務に役立つとして幅広い人間（エンドユーザ・ＳＥ・プログラマ，社内・外注），幅広い技術導入元（国外・国内，企業・大学）をどう関係づけたかという経験を紹介した．

このテーマについては，「トップは口では『教育が大切，技術が大切』というが，具体的な経営体制に何も反映してくれない」，という積年の問題をトップに改善させ，人本

主義，さらには技術本主義という，統合的経営体制を構築するべきであろう．もちろん，ユーザ・ニーズ優先という大原則と，技術本主義を両立させなければ意味はない．

(4) セッション4

オフレコ，アルコール入りということで，採用，面接，昇進，メンタルヘルスなど，話題は脱線・渋滞・大転換を続け，普通では聞けない話が披露された．

メンタルヘルスについては，すでに体制のある企業に学ぶべき時であろう．採用については経験に頼ることなく，企業イメージＰＲも含めて，系統的取組を導入すべきであろう．

(5) セッション5

河村氏は，専門学校におけるインフォメーション・エンジニア養成の新しい学科の設立主旨を紹介した．杉本氏は，柔軟な企業にふさわしいソフトウェア技術者像を分析的に究明する基礎研究を紹介した．

激しく変化する企業環境に適応するには，企業が柔軟であり，技術者も柔軟であり，そのために教育部門も柔軟でなければならない．プログラム言語とジョブ制御文の教育を至上のものと考え続けるか，企業を変化させるメカニズムになることを指向するかで，教育部門の将来は全く違ってくる．

５．これから

開催後の参加者アンケートなどを見るかぎりにおいては，結果はおおむね良好でしたが，運営面やプログラムの設定に関し，反省するものが多くあったように思います．こうした経験をもとに，今回のワークショップ参加者が中心になって，第２回，３回と引き続き開催されていくようになることを願っています．

最後に，今回の運営に参加していただいたプログラム委員の方，事務局，参加者の方々のご協力に厚く御礼を申し上げます．

第１回ソフトウェア教育に関するワークショップ

ソフトウェア技術者のための新入社員教育の現状と展望

篠崎　直二郎
日本電気ソフトウェア

１．はじめに

ソフトウェアを企業活動の主業務とする当社において、ソフトウェア技術者の育成は重要課題の一つである。

特に、キャリアパスを考慮したソフトウェア技術者の育成の立場においても、また、多数の新入社員を採用しいる現状を見ても、新入社員教育は重要な位置付けにある。

すなわち、新入社員を短期間でソフトウェア技術者、もしくは社会人として立ち上げ、実業務がスムーズにできるように教育訓練をする位置付けにある。

当社においては、新入社員教育でコンピュータの基礎技術は当然教育するが、それにも増して、コンピュータ技術者かつ社会人としての躾面を含む幅広い教育として、新入社員教育を考えている。

本編では、当社における新入社員教育の沿革を踏まえ、ソフトウェア技術者に対する新入社員教育の現状を解説する。すなわち、新入社員第一次教育だけでなく、入社後一年間の新入社員（これを研修員という）の育成に関して、新入社員第二次教育および指導員育成制度までを含め、ワークショップ参加に当たり長年わたる経験を基に、総合的に考察する。

２．新入社員教育の変遷

当社は、本年で創立１２周年を迎え、時流の波に乗り確実に成長を続けている。

新入社員教育においても、会社の発展と共に順次発展強化し、現在の形態に至っている。新入社員教育の変遷の様子を表１に示す。

以降に、当社における新入社員教育の変遷を順を追って示す。

表１　新入社員教育の変遷

項番	年度	項目	説明
1	昭和５１	ＮＥＣへ依頼	創立半年目に当たり、ＮＥＣの協力会社向け教育への参加
2	昭和５２	自社教育元年	全社を上げて新入社員の受け入れ教育を自主運営開始
3	昭和５３～昭和５６	揺籃期～発展期	教材の自主化、社外研修を始めカリキュラムの整備統合化
4	昭和５７	カリキュラム体系確立	教育検討プロジェクトによるカリキュラム体系確立とチェアマン制導入
5	昭和５８～昭和６１	教育部主導化への動き	教育部の強化に伴う教育ノウハウの蓄積と整合性への取り組み
6	昭和６２	配属先を意識した教育の開始	ＳＥ部門と開発部門の特性を考慮した実践的教育の開始

２．１　草創期（昭和５１年～５２年）

表１の項番１～２がこれに当たる。当社は創立以来、社是の一つとして「教育の充実」をかかげ早期より自社独自の新入社員教育の母体を作り上げた。

初年度に当たる昭和５１年度は、ＮＥＣの主催教育に参加する形を採った。しかし、昭和５２年度には、全社上げて自社独自教育へ取り組み、企画から実施に至るまで全てを独自に行うことに成功した。

当時の教育形態は、前半でコンピュータの基本教育（コンピュータ入門～プログラミング演習）を行い、後半で実業務に密着したコンピュータ別にクラス編成替えをして、実習訓練を採り入れた。この形式は、昭和５４年度まで継続した。

２．２　揺籃期～発展期（昭和５３年～５６年）

表１の項番３がこれに当たる。草創期においては、テキスト教材などは、市販本やマニュアル、また、ＮＥＣ版講習会テキストなどを用いており、カリキュラムの実施に重点が置かれた。しかし、昭和５２年末に、現在の教育部の母体となる技術管理部教育課が設置され、３名の専任スタッフが配置されてからは、新入社員教育の企画運営にも力を入れ始めた。草創期には技術教育一辺倒であったカリキュラム内容も、昭和５３年度からマナー教育やドキュメンテーション教育などの幅広い教育に心掛け、更に、昭和５４年度には、社外研修を外部団体（産能大）へ委託するなど、教育の充実を図った。

２．３　定着期（昭和５７年～６１年）

表１の項番４～５がこれに当たる。昭和５７年度新入社員教育の準備に際し、新入社員教育検討プロジェクトを組み、それまでの新入社員教育を抜本的に見直した。その結果、教育目標、カリキュラム目標レベルをより明確に打ち出すことができた。また、テキスト教材の自社独自版作成に向けて、大きく前進した。

昭和５７年度からは、教育課に専任のインストラクタが加わり、教育ノウハウの蓄積も急速に向上した。昭和５４年以来、外部委託していた社外研修も、昭和５７年度からは、自社オリジナルとして自主運営を開始した。また、以降で述べる、指導員育成制度が、昭和５８年度から開始した。

２．４　転換期（昭和６２年～）

表１の項番６がこれに当たる。昭和５２年度から自社主催教育を行ってきた新入社員教育も１０年を迎え、また、昭和５８年度に現体制に改変した教育部も確実に力を付けて、当社の新入社員教育の一つの型を築き上げた。しかし、職場ニーズの吸収や、より効果的な教育の実施という面から、昭和６２年度からＳＥ部門と開発部門の特性を意識したより実践的教育を開始した。

また、三か月間の新入社員第一次教育に続いて、業務に密着した、より高度な教育である新入社員第二次教育の実施を開始した。

３．カリキュラム体系

現在、教育部が考えている新入社員教育について、カリキュラム構成、カリキュラム内容、新入社員第二次教育および指導員育成制度について解説する。

３．１　新入社員第一次教育

新入社員教育におけるカリキュラム体系を図１に示す。新入社員第一次教育では、カリキュラム数で見ると、技術カリキュラムが４割、資質的要素を含む教養その他（以降、技術以外のものを非技術という）カリキュラムが６割となっている。ただし、講義日数でみると７割が技術カリキュラムで、３割が非技術カリキュラムとなっている。

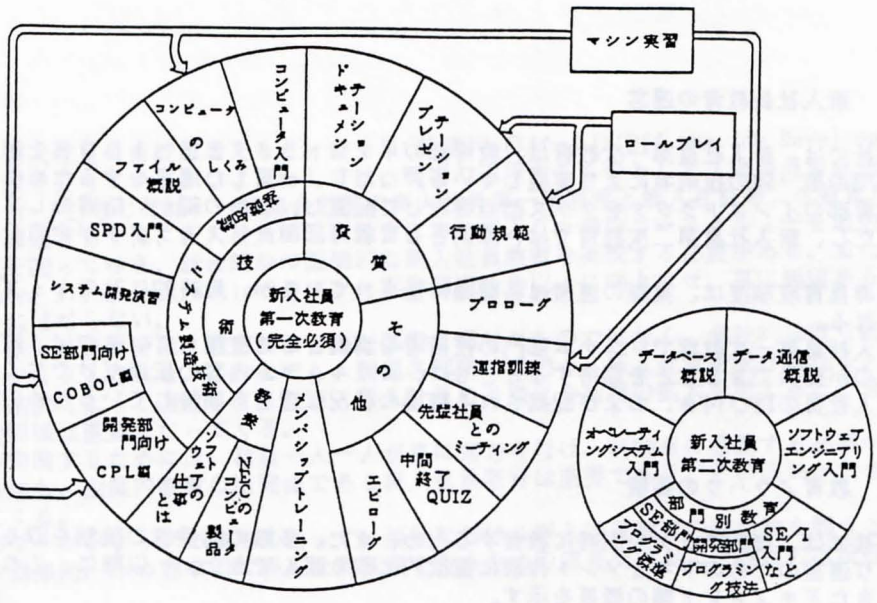


図１　新入社員教育　カリキュラム体系

新入社員第一次教育においては、技術的な項目は、基礎に主眼を置いている。すなわち、職場に密着した教育は、配属先で計画し実施されるが、この配属先の教育をスムーズに受け入れることができるレベルまで、新入社員を立ち上げることを狙いとしている。

新入社員第一次教育の構成としては、入門的な易しいものから、応用的なより実務に近いものへとカリキュラムを組み立てている。

３．２　新入社員第二次教育

図１に示す新入社員第二次教育のカリキュラムを研修員期間中に配置し、新入社員第一次教育の技術面を補強する。

完全必須性で運用している「プログラミング技法」と「プログラミング標準」は、部門の特性を考慮したコースである。また、選択で実施している、「オペレーティングシステム入門」、「データ通信概説」、「データベース概説」、「ソフトウェアエンジニアリング入門」は、Ｃ＆Ｃ技術の基幹をなすもので、大半の新入社員が受講している。

今後は、新入社員第二次教育のコースメニューを増加させ、より広い技術分野に対応できるように拡充強化していく予定である。

３．３　指導員育成制度

三か月の新入社員第一次教育が終了した後、新入社員は各職場に配属されるわけであるが、配属部署内では、ＯＪＤ（On the Job Development：仕事を通した能力開発）を計画的に実施している。これを「指導員育成制度」という。

指導員育成制度は、各部門に配属された新入社員に対してマンツーマンで先輩技術者が付き、指導育成を行う制度である。これは、具体的な育成目標を定め、育成レポートとしてまとめ、毎月評価していくものである。

４．教育内容と運営

新入社員第一次教育は、新規採用者が必ず履修するコースであり、ソフトウェア技術者の育成を考える上で一番基礎となる部分である。また、研修員の育成をも含めて、この章において、具体的に示す。

４．１　新入社員教育の運営

当社では、新入社員第一次教育は、教育部のインストラクタと延べ１００名を越えるシステム部門の第一線の技術者により実施している。ただし、一貫した運営をするために、全クラスに教育部のインストラクタをクラス担当者として配置し、きめの細かい指導をしている。

ただし、新入社員第二次教育では、原則として教育部のインストラクタが全てを担当している。

指導員育成制度は、実際の運営は各職場に任されているが、最終的に社のトップに報告書が出される。

新入社員第一次教育でシステム部門の技術者を講師として登用しているのは、技術的に最新のものが提供できることを期待するとともに、カリキュラム内容の伝達だけでなく、先輩社員と新入社員の結び付き、および配属される職場の状況伝達をも期待しているためである。

４．２　教育ノウハウの蓄積

当社では、複数クラスを同時に教育するため、また、多数の関係者の統制を取るために、当初より運営のためのドキュメント作成に意欲的に取り組んできた。　以降に、これまでに蓄積してきたドキュメント類の概要を示す。

(1) テキスト教材

昭和５８年度より進めてきた教材の活字化も昭和６０年度には完了し、読み易く信頼性の高い教材となっている。もともと、新入社員用として作成したテキストであるが、体系的に整備され、かつ、技術的な面で質が高いこともあり、各方面への流用と共に配属先においても３～５年は活用している状態である。

(2) 講師用資料

講義実施時受講者へ配布する資料類とは別に、カリキュラム単位に講師用資料としてインストラクターズガイドを作成している。これは、誰が講師となっても一定水準以上の講義および演習指導ができるようにと編集したものであり、俗に言う講師のための”虎の巻き”である。インストラクターズガイドは、一冊のガイドブックとして、次のものより構成している。①講義スケジュール、②カリキュラム記入用紙（カリキュラムの講義項目を詳細に解説したもの）、③標準ストーリノート（教案）、④演習指導要領である。演習指導要領は必要に応じて添付している。

(3) 運営資料

運営資料には、まず新入社員へ配布するものとして、研修中の注意事項、教育目標、各種作業指示などを記述した資料がある。これは、新入社員でも読めばわかるように詳しく記述しており、円滑な運営に寄与している。　次に、講師に関した資料であり、システム部門の講師選出文書、講師会議運営関連資料、講師への作業依頼書などがある。

最後に、新入社員教育をプロジェクトと考えて作成した、プロジェクト推進のための資料がある。これは、実施計画書、作業線表、起案、実施報告書などからなる。

以上で示したように、現在では新入社員第一次教育を実施するために、厚さ６０㎝の書類ファイルにして約３０巻におよぶドキュメントをノウハウとして蓄積している。

４．３　インストラクション技術

新入社員第一次教育を実施するに当たり、教育部インストラクタはもとより、各システム部門から選出された１００名を越える講師陣に対し、インストラクション技術の勉強会を実施してる。

特に、インストラクション技術は知識だけではだめなため、ロールプレイを取り入れた実習訓練として義務付けている。また、インストラクション技術を伝達するための教材も完備しており、指導ノウハウも充実している。

５．おわりに

現在の新入社員の育成を考えると、ＯＪＤ(前記)とOff-ＪＤ(Off the Job Development)の両立で成り立っている。ＯＪＤとしては、職場で運用するものとして「指導員育成制度」があり、Off-ＪＤとしては、一括集合教育の新入社員第一次教育と新入社員第二次教育がある。

今後の課題としては、新入社員第二次教育においてより一層のコースメニューを追加し、内容の充実を図ってゆき、総合的かつ効果的な新入社員教育を実現する必要がある。また、ＣＡＩ(Computer Aided Instruction)やマシン実習環境を今以上に向上させ、真に価値あるものとしてゆかねばならない。

なお、新入社員教育は、教育部などの一部署に閉じたものではなく、全社的な取り組みが必要なプロジェクトである。これまでも、関係各部門の協力と、新入社員を受け入れるシステム部門との連携によって運営してきたが、新入社員の多様化や業務の拡大を考えると、今後益々この協力関係は重要になってくる。

企業が発展するためには、社員一人一人が真に実力を付け、前向きに活動する必要がある。そのためにも、企業内教育の出発点である新入社員教育は重要であり、その成果が大きな意味を持ってくる。

今回のワークショップの参加に当って、当社における新入社員教育の妥当性を問うとともに、今後より効果的、かつ効率的な新入社員教育を行うための意見交換および各社の状況把握を目的とする。

フレキシブルな国際企業を目指して

杉本　尚子
フジコンピュータセンター

１．序文

企業は生き物であり，企業を大きくする要素は時代の要請に合った企業方針と異業種間交流及び先端技術を駆使した資源衛星・通信衛星等の国際土壌と超広角な視野を持った営業方針である．

いよいよ自由競争下での時期産業のビジネス・チャンス到来の時期である．このような時代背景を踏まえて欧米と比較して日本企業内の狭義な意味の問題点が３つあるが，国際企業への変貌と国際企業として発展してゆく為の新しい労働のあり方を企業理念に従い２．国際間の各種の摩擦問題について３．国内企業の抱える問題点，４．情報産業の産業構造のあり方，５．エンド・ユーザにおけるプロジェクトチームの編成方法について，概観し，私見として以下に述べる．

２．国際間の各種の摩擦問題についての私見

時代はスペシャリストを要請しているのか，時代はマネジメントを要請しているのか？　情報化の波は異業種の企業の中にも一抹の旋風を巻き起こしている．日本人企業の中で１時的な組織の空洞化が生じてもその組織構造の中での変革，すなわちどのような形で企業内構造変革は起こっていくのだろうか？　大組織の中では知識頭脳集団と呼ばれる組織が活躍することになるだろうが，その知識頭脳集団では日本人の群衆心理に基づく組織力の発揮ではなく，能力差が如実に現れるので欧米におけるスペシャリスト集団である個人主義が成立するかもしれないし，大組織集団の中でフレキシブルな構造体系を作る事が出来るかもしれない．

国際化社会の波は国際化企業に向けての日本企業形態の内部の変化に大きな影響を与えるだろう．日本の企業における評価とは，一律的に組織の力の結果が重要視され評価される点にある．しかし研究・開発等の業務内容によっては個人の専門外の仕事であっても習得された基礎学問や異業種の既存技術を応用して新しい発想による専門へのアプローチが既存の枠に捕らわれない新しい方法を生みだす要素を含みそこでの個人の独創力の発揮が出来るかも知れない．その技術的評価や価値が国際企業での生存競争の重要な要となり，欧米のビジネスマンに高い評価を得られる事を企業は知っているからであり，個人の技術力が企業をも動かすような影響力を持つならばそれは可能である．

欧米との新しいフェアな土俵での企業間の競争を欧米の人々に認めさせる為には質的物理的な量の数量的提示等が必要であり，日本流の集団組織による結果のごまかし，種々の摩擦の原因となっているあいまいな部分は透明にし，それらのカテゴリーを科学的に分析し提示し，国際人としての送受信を明確にする必要があるのではなかろうか？　特に根本的に人間関係に依存しない欧米においては集団理論の結果は通用しないし，それらの相違点は先端技術や経済摩擦等で実証されている．

国際的なビジネスは互いの習慣・言語・社会的風土や歴史・ビジネスの形態の相違・ポリシーの相違・国民性の相違等を前提として認めた上での高い次元の道徳感や人間性での交流の結果としてパートナーシップが誕生してくる．

これらの点で我々は欧米における評価の相違に気づいていない．批判的に言えば，組み合い運動に代表されるような広義の枠組みは流動的に変わるがその枠組みでしか生きられない習性，村社会に見られるような上下の階級落差による異色と呼ばれる者への排他や，同一基準ベクトルを捜し体裁ばかりに気を使うパターン化され定型化された人種，それぞれ階級や次元は異なるが社会の中で企業の中で若い古いを問わず日本人の大きなマイナスの特徴であり，能力や個性尊重や成功を素直に喜ぶ心にかけている．低迷化していた各種の問題点は時系列的に進行し，国際土壌で表面化されていま各種の摩擦の現象としてあらわれている．その結果なれ合いのごまかしでは通じなくなって社会・階級のどの部分に旧体制然とした問題点があるのかを分析しそれらを改革してゆかなければならない時代に来ていると言えよう．

批判する事はたやすいが実行する事はとても難しいので，実行は舵取りを間違わない次元や価値の理解できる時代の先駆者としての役割にふさわしい者でなくてはならないと思う．

３．国内企業の抱える問題点

組織内部の人脈や個人の企業理念の相違・生きざま・個人の技術力の格差に基づく仕事方法の相違等人間関係に依

存する部分が１つの大きな問題点であり，２点目は１企業枠を乗り越えて産・官・学や情報処理産業やコンサル等の自由な異業種間交流が相互の信頼関係により可能かどうかの問題点，最後に時代の要請すなわち急加速度的に直線的に進化する事に反発する組み合い組織のような内部の質や次元や体質の改善が３点目であり，これらの問題点をどのように透明化しそれらを乗り越え，フレキシブルな企業形態に移行するかが時代の先頭に立つリーダーシップ企業になり得るかどうかのキーになる．

広義な意味では企業を繁栄させる為には企業論理に基づく企業間の競争論理と紳士協定に基づく協調であり，送信・受信の明確な外資系のパートナーシップであり，企業間競争論理と異業種間論理と異業種間交流に基づく互いの進捗状況に合わせた企業間調和にある．

すなわち，時代の要請を素早くキャッチし成長する速度は各業種別企業やその企業規模・企業母体により土木・測量産業に代表されるように広く浅くの表層構造の企業構造を核にそれらの産業の特徴と格差をどのように関係づけるのかにより国際土壌での展開がなされる．その為には従来のカラーにとらわれない複合立体組織構造への変貌と企業の先行投資の研究産業への移行は必要不可欠な条件になるであろう．

４．日本の情報産業の産業構造のあり方

メディア産業ともてはやされた情報処理産業においても，残念ながらその実態は上記のような問題点をその内部に含んでいる．欧米の技術の加工技術の逆輸出としてのソフトウェア工学は企業論理の中で日本の風土・利用者側に立ったアプリケーションソフトウェアの商用化として進化しているが，今後の発展の核となるのは基礎研究に基づく独創技術にかかる．

一部の人々を除けば独創技術の歴史や土壌を持たない日本では新鮮なシンプルなシステムつまり単→複へのソフトウェア工学やコンピュータサイエンスが発展してゆくのは難しい．時代は進化しても以前旧体質としているのは時代に逆行する組み合いや人間だけで，先端技術の発展に欠かせない独創技術の土壌を潰す危険性をはらんでいる．ものの構造や調和や関係は人間工学的に分析しようと文章の美的描写や感性を重要視する側面から描写しようと同じであるが，（各分野により一概に断定できないが）芸術や文学と違って，サイエンス・アカデミックでは日進月歩の技術革命は試行錯誤の実験や環境や能力や努力や時期やあらゆる要素の結果にすぎないし，成功に導く事の方が難しい．

しかしその結果が高次元の技術的側面において学術レベルで時代を進化してきたと言え，我々はその恩恵に預かっているにすぎない．時代を推進する立場に立って上記のような様々なカテゴリーを考えると企業や社会への貢献度・知識水準に基づく各専門職種のあり方（憲法でも定義されているような底賃金・高賃金の区分け）・評価の基準は開かれたシステム体系としてフレキシブルに上への構造体系へ伸びてゆかなければならない．

企業の内部における　OA　化は完備されたがEWS(ENGINEERING WORK STATION) やCAD(COMPUTER AND GRAPHICS)化やパソコンのファイルサーバーの完備はなされていないし，第４次産業における国際土壌での世界間の通信技術も企業間で完備されていない．このような時代背景においてインテリジェントビルの中での新しい人間関係に依存しない質的・能力的・業務実績・学歴による賃金体系や労働環境が誕生してくると思われる．

時代の要請による企業の３資源の変革により，２１世紀にふさわしい労働の意味とは？　労働の価値とは？知識頭脳集団とは？　新しい企業論理に沿った新しい基準作り・カテゴリーを改めて考え直さなければならない時代・時代の過渡期に指しかかっている．

５．エンドユーザーにおけるプロジェクトチームの編成方法のあり方

国際企業を目指す企業において新産業の新規開発は，エンドユーザーのエキスパートである製造研究社の業務形態にも変革をもたらす．

それは閉じられたエンドユーザーのみの技術開発と言う枠を取り外した形で進行されパートナーシップでの企業を代表する高度な専門的な技術スキルを有する人々の活躍にかかっていると言えよう．それは各専門分野の専門家集団のプロジェクト制による実現に他ならない．国際企業移行のプロセスは，つまり研究産業全盛となる為にはエンドユーザーが閉ざされていた境界を自ら破り情報処理産業との境界・落差をなくす為現在の技術スキルを落とす事なく大海へ船出する努力が必要ではなかろうか？

その舵取りは財団法人がするのか，国の機関がするのかいずれにしても情報処理産業の国際複合立体情報産業への変貌さえも特にアプリケーションと言う分野ではエンドユーザーが鍵を握ると考えられる．

ＣＤに思うこと

渡邉 雄一
電力計算センター

１．バース（Verse）から

音楽にもいろいろなジャンルがありますが，みなさんはどんなものを好んでお聴きになるのでしょうか？　いや，聴くのではなくて演るほうが楽しくて，という方もおられるでしょう．私はもっぱら最近（ここ１５年くらい）は，ジャズばかりを聴いています．ほんとうはナマも聴きたいのですが，金欠病，多忙病，無休病の３重苦をわずらっている関係から，限られた休日に，わずかばかりのレコードのごく１部を，フルボリュームで聴くことを無二の楽しみにしています．

私もジャズといえばモダンあたりから入ったのですが，マイルスよりもモンクやロリンズを好んで聴いているうちに，バーカーをすっとばしてレスターにのめり込み，中間派に傾倒するようになりました．レスターのいい演奏を聴くうちに，メロディーのきれいなスウィング系の演奏にもしびれ，そうこうしているうちに，デキシー・スタイルの荒っぽいけれどものすごくエネルギッシュなものも，というようになっています．

なぜこのような状態におちいるかといいますと，エサ箱（レコード屋でコレクターが商品をあさっている図を想像して下さい）に安いレコードが並んでいると，「あっ，この人の名前は知っているけど，聴いたことないなぁと思って，つい食指が動くからです．それとて私の数少ないコレクション（６００枚程）では，集めたとて聴けるものはたかが知れています．ジャズのコレクターからすると，私のようなのはコレクターの部類に入らないようであります．だいたい４，０００枚くらい持っていて，伏目がちに「少々もっております」というのが相場のようです．個人的には，現在ハケットを中心に集めております．ハケットのように活動歴が長くかつ一般には超一流ではないけれども，信仰者の間では絶大な支持を得ているような人の物をコレクトするのは，かなり骨が折れます．それがまた，コレクターとしての気力を奮い起させる活力源になっているのかも知れませんが．

２．コレクターの悩み

ところでコレクターの１人として頭のいたいのは，そのコレクションの管理です．特にある人のレコードをコンプリートに集めようとすると，同じ曲でもテイクの違いはもとより，レコードのプレスにまでこだわるようになります（初回プレスの方が断然音がいい）．また，自分のあのレコードのＡ面の４曲目はキズが多少あたっていたけれど，このレコードはジャケットはいたんでいるけれど，レコードそのものはきれいだから買っておこう，となるともう病気です（私は一応慢性ですが，自覚症状があるので重度ではありません）．

レコーディングに関する情報が一切ないものは悲劇です．聴き比べてテイクなどの違いをチェックするしかありません．私の持っている例でいえば，１９５０年代の Capitol のレコードがこの種の代表格です．ジャッキーグリースンのオーケストラでハケットが吹いているものは（Capitol の再発ものだけでも），レコード番号が同じでテイクが異なっていたり，曲数が違っていたり，ジャケットのデザインが違っていたり，ステレオかモノラルかという調子で，１０インチも入れると５枚もあるというのがあります．それに他国の再発ものが加わるともう手が付けられない状態になります

ただでさえレコードの管理（情報の整理，保管場所の確保や重量の問題など）に困っているコレクターにとって，病気を悪化させるのが昨今のＣＤブームです．最近のＣＤでは，単にＬＰレコードの再発というだけでなく，オリジナル・マスタテープによる音おこしをしているものも多く，また収録時間が長くとれることから，商品としての体裁を整えるために，未発表のものも含めるという傾向があります．私もＣＤには冷たい目を向けていたのですが，昨年末に出たビリー・ホリディのセットものや，リーワイリーのものが発売されてくると，気分はおだやかではありません．

３．折角のＣＤだけれども

ＣＤについては，コンピュータの世界でもＣＤ－ＲＯＭということで，その応用が最近盛んなようです．私は，何でも新しいものは出来れば避けて通りたい「技術オジン」の１人として，ＣＤについてもひたすら無視を続けてきたのです．しかし幸いなことに，３月にあった春のセミナーで，ソニーの土井さんのプロジェクト管理にまつわる話しを聞いてから，講演の本来の意図とは外れますが，ＣＤに

興味を覚えるようになりました．細かい技術的なことはよくわかりませんが，現在のＣＤに対する不満な点は，その利点を生かしきっていないと思うからです．いくつか不満な点をあげると，

(1) ＣＤは傷みが容易にはわからない

これは中古市場で問題になってきます．アナログ・レコードの場合は，目視によるチェックや指によるトレース等により，問題箇所を特定して，その部分を試聴することで納得した買物ができていますが，ＣＤではそれができません．私は一度レコード屋で， recursive-call するＣＤを返品にきた人をみたことがあります．返品おことわりが原則の中古市場で（ＣＤの価格がまだ安くないだけに），今後のトラブルが少ないといいのですが．

(2) ジャケットに迫力がない

ＣＤのメリットの一つとして軽量・コンパクトであることがあげられます．しかしそれが，かえって逆にデメリットに作用しているのが，ジャケットの写真だと思います．つい最近の人（特に歌謡曲の人）では，ポスターとかで補うことも可能でしょうが，ジャズのものでは写真そのものが貴重で，それをみるのが楽しみのひとつでもある場合があります．画像もＣＤの中に含めて提供してもらえると助かります．

(3) なぜ，ライナーノーツがＣＤに入っていないのか

ＣＤにはアナログＬＰ１枚が，楽々と入ってしまう能力があるのに，なぜついでにライナーノーツや演奏記録等を入れてくれないのでしょか？　TEX や roff でというぜいたくはいいません．せめて曲名，演奏者名，母番号，収録時間，収録日時，場所等のディスコグラフィに相当する情報を ASCII コードでいいから，なぜいれてくれなかったのでしょうか（私はなにも松田聖子やピンクレディーのディスコグラフィをながめてにやにやする気など毛頭ありません）．

４．私にとって夢のＣＤとは

１人の音楽愛好者にとってＣＤは，今までレコードに満たされなかった音の特性の確保が，容易にできることは，すばらしいことだと思います．けれども，せっかくの文書や画像等の情報に対する配慮が，あわせてされていない点が，最大の欠点だと思います．音楽愛好者をパソコンの利用者として捲き込むことは，停滞している景気を高める効果ばかりでなく，ソフトウェアの利用理解の促進という面からも歓迎されるはずです．

また，コンピュータの外部記憶媒体としての機能も期待するのなら，利用者からの書込みもぜひ可能にする必要があるでしょう．そのような解決策はまったくないのでしょうか？　例えば，ＣＤ単独としては無理かもしれないので，フロッピー・ディスクと組み合せるなどが考えられます

ＣＤを媒介として，音楽の鑑賞のための環境がよりすばらしくなることは結構なのですが，せっかくのチャンスを潰さずに，よりよい付帯環境の構築・拡充が出来れば最高なのです．例えばこんなのはどうでしょうか？

【これからの音楽鑑賞】

- ＳＩＧＭＵＳＩＧセンターの jmnet （Japan Music NETwork）にログインして，例えばスウィング系ジャズのニュースを読みます．
- 有益な情報（新作情報，ディスコグラフィーの正誤表，発掘情報，再発ものの話題，コンサート情報等）に必要なものがあれば，それを記録してあるデータベースに，再度ログインします．
- もしそれが音楽そのものであれば，サンプル出荷の対象を自分のファイルにダウンロードして視聴します．
- もし気に入れば，それの購入手配を jmnet でします．
- 家に届いたＣＤを，まずチェックプログラムにかけて異常がないかを確認します．
- ＣＤプレーヤにかけて，自分のデータベース（ディスコグラフィー，購入記録，購入予定リスト等）の更新追加を行います．
- ライナーノーツを日本語翻訳システムにかけたあと，自分で訳しながら音楽に耳を傾けます．
- 音楽評を書いてＣＤの個人用記録領域に記録するとともに， jmnet のニュースに送ります．

コンピュータの理解が，一般大衆によりいい形で浸透していくためには，いままでの既成のメディアにこだわらずに，積極的にアピールを続けていくことが大切なのではないでしょか？　とくにネットワークのように，標準化の遅れているものでそれなりに設備投資を必要とするものは，なにか付帯効果がなければ，その需要はいつまでたっても遅々として進まないような気がします．私がＣＤプレーヤを買いたくなるときは，多分それに RS-232C のジャクが標準装備されている機種が，発売されたときのようです．

キーボード・アレルギーについて思うこと

細野　広水
エム・ケー・シー

１．はじめに

SEAMAIL Vol.2 No.8にある，宮田重明さんのレポート "ドキュメンテーション支援ツール作成の経験#を楽しく読ませて頂きました．"上司に相談しなければ"というプロセスが，再起呼び出しであったなどとは，考えてみたこともなく，思わず笑ってしまいました．そのとき，ついでに，ふと，あることを考えました．宮田さんのレポートの内容とは，なんら関わりはないのですが，たまたま，そこを読んでいたときに感じたので，引用させて頂いたまでです．

２．キーボード・アレルギーは存在するか？

キーボードアレルギーは，存在するか？　という問があったら，答えはイエスで，アレルギーは存在するだろう．しかし，それは，コンピュータの使いにくさをあらわす物差しにはならないとも思う．表１のようなアンケートとその答えから，キーボード・アレルギーについて考えてみました．

３．アンケートからの考察

表１をみていて，ひとつの結論が出ました．

よくコンピュータの使いにくさを表現するのに，　機種毎の取り扱いの違いをあげ，さらに比較のために，自動車の運転方法を並べる人がいます．確かに，現在の自動車は，それがアメリカのものであれ、ドイツのものであれ，ほとんど共通の運転方法で利用できます．でも，コンピュータの操作方法と自動車の運転方法を，単純に比較ができるでしょうか？

私は，「否」と思う．なぜなら，機能レベルでいえば，自動車は電卓以下だと思うのです．自動車の運転機能の主だったものといえば，人（もしくは荷物）をのせて，せいぜい前後左右の方向決定と加速減速ぐらいなもんでしょう．この程度なら，パワーアシストがあれば，電卓の「+-x=」ぐらいのキーで運転できるのです．

でも，自動車の運転は難しいとは思わない．なぜか？それは，全ての制御を運転手自身が行っているからです（電卓も同じです）．もし，それを自動車自身にやらせたら，どうなるでしょうか？　きっとタクシーの運転手のように，言葉を理解してくれる自動車が必要でしょう．そんな自動車はいつになったら現れるでしょうか？

なにしろ今の自動車は，一次元の速度を決定するだけのために，アクセル，ブレーキ，チェンジレバー，そして，スピードメータと，４つも使いこなさなければならないのです（オートマチック車だって，前進と後退の切り替えがあるのです）．メータを指で動かして，スピードを決めればそれで済むのに．

これに比較したら，コンピュータの機能の多いこと．そもそも，ひとつのコンピュータにいろいろやらせようと思うからいけない．もっと簡単に，たとえば，コピーしかできないパソコンとか，通信しかできないパソコンとか．同じ電池があるのに，写真が写せる自動車がありますか？同じ石油を使っているのに，月に行ける自動車がありますか？

４．結論にかえて

なんて，だんだんむちゃくちゃになってきました．この原稿は，パソコンに向かって，ワープロソフトを動かしながら，勝手気ままに考えながら打ち込んでいるものですから，２０行くらい前で何を考えたか分からなくなってしまって．

さて，最初の話に戻って．結論にかえていえば，小学校へ入学して以来（人によってはもっと前から），無理やり持たされて２０年もお付き合いのある鉛筆と，昨日今日初めて目の前においたコンピュータとを比較して，使いやすさを論じるというのは，無意味だと言いたいのです．

あるいは，高いお金を費やして，何十時間も教習所に通った自動車の運転と比較しても前提が全く違うのです．

ただし，今のコンピュータが，素人向けにできていないのは事実です．われわれに責任がないとは，思っていませんので，悪しからず．

問い	回答
Ｑ１：　あなたは，コンピュータの操作が，難しいと思ったことがありますか？	・コンピュータを使おうと思ったが，うまく使えなかった人　→　強い調子でＹＥＳ ・コンピュータをおもちゃのように扱っている人　→　ＹＥＳの人とＮＯの人がいる
Ｑ２：　操作が難しい(と思う)のは，設計が悪いからだと思いますか？	・大方の人　→　ＹＥＳ ・特にコンピュータを遠くから利用しながら，自分で触れたことがない人は　→　強い口調でＹＥＳ
Ｑ３：　あなたは，コンピュータをとりあえず結果が出るまで自分で操作したことがありますか？	・コンピュータは，難しいと思っている人，すぐに嫌になった人　→　強くＮＯ
Ｑ４：　ドキュメント類(帳票類)を作るのに，コンピュータ（ワープロ等)）と鉛筆とで，どちらが使いやすいですか？	・コンピュータを使ったことがない人，コンピュータに嫌われた人は　→　絶対に鉛筆
Ｑ５：　あなたは，コンピュータを使うために，勉強したことがありますか？	・ほとんどの人　→　ＮＯ
Ｑ６：　鉛筆で字や絵を書くことを何年やっていますか？	・ほとんどの人　→　２０年以上

表１

会員の声

Seamail Vol. 2 No. 9 と一緒に送った「SEAMAIL に関する会員アンケート」の集計結果と，一言メッセージを掲載します．アンケート総回答数は，95 名の会員の方々から頂きました．発送当時の会員が約 1,200 名ですから，7％の回答率ということになります．

1．設問A－C

設問A－Cの集計結果は，表1にまとめました．

回答結果にもとずいて，今号は，原稿をワープロ出力を切り貼りのまま使用したものを一部掲載しました．

Seamail は文章が多くて読みづらい，という意見は以前から寄せられていますが，寄せられる原稿そのものに図や表がほとんどはいっていません．編集部では，力のあるかぎり，文章から表や図を作成して読みやすいものにしていますが（表1のように），それにも限度があります．なるべく再利用できるものをよせていただけると助かります．

設問		回答項目		回答数
A-1	セミナー，フォーラム，ワークショップなどの記録（録音テープとＯＨＰのコピー）を原稿の形にまとめる仕事をお引受けいただけませんか？	Yes	（ぜひやりたい）	7
		Maybe	（暇なときならできるかもしれない）	36
		No	（むずかしい）	51
A-2	書評または内外の論文紹介あるいは新製品（ハード／ソフト）紹介のページ（それを担当するボランティア・グループ）を作りたいのですが	Yes	（ぜひやりたい）	7
		Maybe	（暇なときならできるかもしれない）	33
		No	（むずかしい）	53
A-3	創刊当初連載していた会員インタビューの欄を復活させたいのですが，インタビュア（もちろん原稿書きも）やっていただけませんか？	Yes	（ぜひやりたい）	6
		Maybe	（暇なときならできるかもしれない）	28
		No	（むずかしい）	60
B	現在の誌面は，Unix の ditroff 出力に統一されていますが，そのためには編集部サイドで Formatting や日本語の手直しに，かなりの労力を必要とします．そこで，今後は，会員からの投稿（機械出力のもの）は，そのままコピーして誌面を構成することも考えています．若干誌面が不統一で読みにくくなりますが，いかがでしょうか？	賛成	（読みやすさより定期発行のほうが大切）	76
		反対	（誌面の Quality を保つことのほうが大切）	11
		その他		6
C	当面の編集ボランティアの力では，毎月 32 ～ 40 ページの誌面を編集して行くのは，やや苦しいと予想されます．そこで，定期発行を第一義に考えて，とりあえず編集がすんだ分だけで出してしまうという案もあるのですが？	賛成	（定期発行されるのなら，ページ数の増減は問わない）	82
		反対	（若干発行が不定期になっても，一定のページ数が欲しい）	8
		その他		5

表1　アンケート回答結果表

2．設問Eの回答

この設問は，特定のテーマに関する「会員の声」アンケートとして，どのようなテーマがよいでしょうか？ というものでした．寄せられた回答から，類似のものは代表的なものに変えてあります．編集部では，タイムリーは話題を選んでアンケートを作成したいと思っていますが，みなさんも，是非，簡単なものでかまいませんので，原稿をお寄せ下さい．

・パソコン通信（ネットワーク一般を含む）
・ワークステーション（OS，管理を含む）
・ユーザ・インタフェイス
・ソフトウェア技術者と社会の関わり
・ハッカー
・ファミコン・インパクト
・在宅勤務
・外注について
・日本語システム
・中高年技術者の現状・将来
・UNIX（vs. MAC，メインフレーム・マシン）
・各種試験制度
・ソフトウェア再利用
・ソフトウェア技術者像
・技術者教育
・知的所有権，法的保護
・プログラム自動生成・合成
・ドキュメンテーション（DTP を含む）
・メンタルヘルス
・シグマ計画
・TRON
・開発環境

3．会員からの一言メッセージ

松村好高

SEAMAILの定期刊行について，若干のコメントをしたいと思います．

・形式について

62年度は，いままでの方針で編集し，新年度から，アンケート結果を踏まえた形態でのぞむのはどうでしょうか．

・記録の原稿化作業

行なった者のしか分からない苦労があると思います．個人的には，現在協力できない状況ですが，実施者の皆様には感謝します．

加藤　明

プログラマ定年を１０年も過ぎてからこの世界に本格的に関わることになり，分からないことばかり．セミナー等でも聞くだけで精一杯というあり様です．情報化社会に向かっているなら，もっとコンピューターが身近になるべきだと思いますが．自身喪失しています．

岸本剛一

企業組織とは独立したボランティア団体に魅力を感じて加入したものの，機関誌の通読もままならない落第会員です．現状では，contribute できる範囲は少ないのですが，パソコン通信で編集参加のようなことができれば，お役にたてるかもしれません．

洞地南郷

これまでに，関西支部の分科会に一度だけ参加し，楽しく拝聴しました．他の分科会等にも参加したいのですが，なかなかスケジュールに組み込めません．そこで，SEAMAIL を心待ちにし，セミナー，フォーラム等の記事（独断であろうと）だけを頼りに活動を知り，自分ももっと積極的に参加したいと考えています．みなさんも頑張って下さい．

北条正顕

昨年，肩に力を入れて，キーボードをたたいていました．その結果，今年は，１００ステップ肩になってしまいました．

後藤　浩

禁煙ならぬ禁コーディングの私です．

新妻正夫

もっと軽い読み物的なページがあるとおもしろいかもしれませんね．例えば，矢野徹さんの「ウィザードリィ日記」のような（依頼されれば書きますが）．

小林貞幸

小生，社員の技術教育を担当するセクションの責任者ですが，新入社員の社会人教育も含めて，研修スケジュール，

カリキュラム等（情報処理サービス業の社員教育はかくあるべき）について，支障のない方は，貴社の（貴方の）やり方，持論をお聞かせ下さい．

山口　均

開発方法を早くマスターしたいのですが，どうすれば良いですか？
ソフトウェア機器について知りたい．
自分の作ったプログラムが，ユーザでどれだけ使われているか知るには，どうすればよいか？

岡泉享祐

今月入会させていただいたばかりですが，もう少し早い時期から入会できていたらよかったと思っています．立場，環境の違ういろいろな方のお話をお聞きして，コンピュータがあらぬ方向にいかぬようにしたいと思っています．

山田二郎

私は，８７年４月にＵターンして，北海道に戻ってきました．東京集中型が改められて，地方の時代が来るよう，ここで努力したいと決意しています．

紺野富士利

他の人々（会員の方々）が，日頃どんな事で悩み，どんな解決策をとっているのか，知れるような場（SEAMAIL の中で）があるといい．

山岸英樹

私，日本電子計算（株）関西支社，白井様のご紹介で入会させていただきました．関西の一会員として，積極的に，できる事は協力させていただきたいと思っています．今後のソフトウェア技術者協会の発展に期待いたします．

南　慶一

最近気づいているのは，今後，将来，急増するのは，オフィスでの使い捨てソフトではないか，ということです．そのようなソフトを自動／半自動で生成するメタソフトが，（本当は）望まれているのではないか？様々な仕事にあわせた，使い捨てソフトをどう生成するかは，ユーザの（一種の）プログラミング，すなわち，これが世の中の「ソフト化」のひとつでしょう．

岡村　博

Macを買いました．　SEAネットは，どうなったんだろう？

小川昌邦

会費を値上げしてもいいですから，ボランティアの方には，薄謝を出すようにしてはどうですか．

中北たか子

SEAMAIL の船便シンドロームについては，編集部の方々がだいぶ憂慮されているようで，当方も胸がいたみます．でも，私など，　SEAMAIL の開封・読み取り処理を一括バッチ（２，３ヶ月に一度）で行っている不届き者ですから，船便でもラクダ便でも，手元に届くだけでもありがたいと思っています．どうぞ，編集部の皆様，ご自愛のほど．

藤原千尋

先日，ユーザーとして，新規システムの開発に携わっている友人と話し合ったときの結論．ユーザーの業務合理化のために，システムを開発した場合，それがユーザ側企業等の利潤拡大のみを生むのではなく，雇用者が余暇を与えられ，それを有効に用い，精神的向上を果たしてこそ，ＳＥとして社会の向上に寄与した満足が得られるのではないか．したがって，納品されるマニュアル類には，分厚い「システム操作法」と共に，「このシステムでできた余暇の使い方」なる一冊が含まれるべきであり，遊びの精神にたけ，暇を持て余す人々に，時間の楽しい使い方を指導できるというのが，望まれるＳＥの近未来像である？？

田村　整

SEA 発足時，ボランティア参加を考え，第一回の集まりに参加しましたが，地方に落ち着く身となり，SEAMAILが無くならない事を祈っています．開発環境が大きく変化して，その事の整理がついたら，　[D]-[3]のテーマ，又は，[D]-[2]のテーマで，何か投稿します．時期は約束できませんが。

清水保弘

現在，オブジェクト指向プログラミングに大変興味があります．業務，趣味ともに，オブジェクト指向言語を使える環境にありませんが，　1988年はその整備に努めたいと

思います．それから，どなたか「ゲーデル・エッシャー・バッハ」という本に，興味のある方はおられませんか？もし，いらっしゃったら，意見交換の場を作りませんか？

小須田正孝

新年を迎え，今の我々の仕事の存在を有り難く思うものです．情報処理産業がこれだけ，ニーズ，シーズが高まってきた歴史には，それなりの理由があると思うのですが，その理由を明確に挙げて見ると，人により，意見が千差万別であったりします．ニーズ，シーズが，それだけ多様化し続けているのですから，未だ未だ生きていけると思ってみています．ポジティブに視点を変えると，多様化の本質を見極め，さらに，仕事を掘り起こしていけば，充分歴史は今後更に開けると思うこの頃です．

中村昭雄

原稿はできるだけ手をかけずにする事を考えたほうが良いのでは．ワープロ等で編集する事を前提に，５インチ，MS-DOS でフロッピーをもらい，使用する等を行った方が早いと思います．ただし，プリンターは，レーザープリンターで！

中山照章

以前から，提案しているのですが，早く，SEANET を運用して，SEAMAILの原稿をネットワークで集めたら，いかがでしょうか．でも，原稿そのものがないと駄目か・・・．ランダムサンプリングした会員に，編集部から直接電話して，電話インタビューなんてのはどうでしょうか．

福田充利

SEA主催のNetwork（BBS）を構築するという話は，どうなっていますか？ NEC の PC-VAN の中に，寄生（CLOSED USER GROUPとして）すれば，（A）-（C）の問題は，半分くらい解決すると思いますが．

大沢章男

AI のマイコン故障診断システムを導入して効果を上げています． AI分科会で発表したいと思います．

荘司憲彦

横浜に在住しているせいか，毎月 YDOC のおしらせが送られてきます．ぜひ一度，出席してみたいと思っていますが，同じ横浜でも横浜の田舎（緑区）では，交通の便を考えると，つい，でそびれてしまいます．そこで，勤務先（東京.飯田橋）近辺で，会員の交流を第一目的とした，飲み会風集まりを開きたいのですが，賛同される方はご連絡下さい．

松本　豊

SEA も junet で，どんどん"声"をつのったらいかがですか？

成澤知子

入会したものの，フォーラムにも参加できないままに終わった’８７年でした．今年こそ，積極的に皆さんのお知恵拝借すべく，少しでも多く参加しようと決意しました．技術計算分野での，ソフトウェア生産性の向上のためには，まず，何をなすべきか，が現在のテーマであります．今年も宜しくお願いします．

北村幸雄

編集部の方が，会員の中から，毎月，ランダムに何名かを選んで，原稿を依頼してはいかがですか．甘えがあるのかも知れませんが，毎日，仕事に追われて，学研的な原稿はなかなか書けません．しかし，今考えている事ぐらいは書けると思います．

中友　宏

BBS を開局してみてはいかがでしょう．

野村行憲

編集長のご苦労は察するに余りありますが，　SEA会員の唯一の（オーバーかな？）楽しみです．声援だけはおもいっきり送りますので，何とか，定期航路に乗せて下さい．

田村和三

ＥＤＰ部門の在り方，かかわり方が，大きな問題となっています．場合によれば，存在すら云々されています．今後のあり方をよくよく考えねばならぬと思います．職人であってもいけない，専門家すぎても困る．オールマイティを要求されています．しかし，経営にありがちなマネージメントに貢献し，その貢献度がチェックできるようにしたいものである．

中西慎治

SEA会員自ら開発した，ソフトの紹介コーナーがあってもいいんじゃないのですか？　また，皆さんが，どんな仕事をしておられるか，たいへん興味があるので，その内容をＰＲも兼ねて，投稿しあうのもいいと思うのですが？

荻原秀和

パソコン全盛のこの時代，私は，一度もパソコンにさわったことがありません．パソコンって，そんなにいいものなのでしょうか？　知りたい．今年は，VAX11/780 も MICRO VAX2 も APOLLO DOMAIN も EWS4800 も忘れて，チャレンジしてみようかなと思っています．というのは，本当は，OS/2に非常に興味があるからなのです．

阪井　誠

「ハッカーには，女の子しか生まれない」という話を御存じですか？　CRT の影響で，染色体異常が起こるからだという話ですが，この度，無事に，私がハッカーでない事が証明されました．（誰ですか？テレタイプでハックしていただろうという人は）．

小林俊明

ちかごろ，UNIX に取りつかれています．昨年７月に導入したただ一台のSun-3を昼な夜なで使いまわしています．マルチウインドウは，一人で端末二台を使っている気分で最高．　TSS ターミナルよりはずっと早いよ．でも，この間，Diskがパンクした．まだ，Junetにつないでないので，誰かぶらさげて下さい．　public domain がほしいヨ！！

松原友夫

SEAMAILが，その名の通りであっても，受け取る方は，心待ちにしています．時々，一般の雑誌ではお目にかかれない，生の声に触れることができる，というのが，これの評判です．今までとぎれがちだったとしても，発行を続けてこられた，編集部の方々の努力には頭が下がります．私自身，色々な仕事を引き受け過ぎて，backlog がたまり，あまり貢献できそうにもないのが残念ですが，時には何かを寄稿したいと考えています．

日遙武彦

地方在住の会員にも，メリットがある SEA をめざして下さい．

玉井哲雄

日本語プログラミングについて調べています．開発経験や意見のある方，ご連絡下さい．

折田　豊

不正コピーは，ニセ札製造となんら変わらない．近いうち，どこかの大企業が著作権法違反で，罰せられるのではないかと，心配（本当は，ワクワク）している．いけないと知りつつやってしまう不正コピーの断ちかたは，ないのだろうか？

原田　峰

機関誌の遅延は，本会の非活性化につながります．なんとしても定期刊行にこぎつけてください．分科会の休会の多いこともいささか気になります．

臼井義美

SEA の会員の皆様は，本当にお忙しそうですね．原因はどこにあるのでしょうか．個人の問題，それとも環境？　組織？　企業？　ソフトウェア産業の宿命？　民族？　人間？　生命？　宇宙？　そもそもコンピュータは，人間の仕事を助けるために作ってるんじゃなかったっけ．

藤吉博章

何のお役にもたちませんで，編集部のみなさまには，本当に，申し上げる言葉もありません．　[C]には，”反対”しましたが，「定期」でかつ「一定の内容」であることは，同じくらい重要であると思います．どちらが欠けてもつい読まれないで過ごしがちだからです．ボランティアだけでなく，「有料」のスタッフも考えて良いのではないですか．

鈴木真二

SEAが開催するセミナーは，高い．もっと安くして，多くの参加者が得られるようにすべきだ．

尾平和聡

編集に携わる皆様のご苦労に感謝します．読者としては，益々の紙面の充実を願うのみであります．

小田七生

私のような地方（九州）の会員は，なかなかセミナーやフォーラムに参加できない．最近はやりの，パソコン通信

で，SEA専用の電子会議等（BBS＜SIG）ができるとありがたい．

大地英夫

最近考えている事ですが，納品物件の一つとして，「デザイナーズ・ノート」を作ろうと思っています．つまり，設計者として，設計上苦心した事や主張したい点に注目してもらい，アルゴリズム等を説明するものを残す．

岡部　仁

各地で研究会（分科会）が行われていますが，それぞれの紹介ページを設けてほしい．事務局の皆様のご努力を感謝しますと共に，今年度もどうぞよろしくお願い致します．

松沢好男

昨年より，ソフトハウスの淘汰が，頻繁に，取り上げられています．そんな中で中堅技術者は，より以上に自己啓発を行い，もっとアカデミックな視点で設計し，システムを評価していく姿勢が大事なのではと感じている昨今です．

藤本良子

九州地区の方，支部を作りましょう．ご意見，ご要望まってます．

金井美代子

限り無く多面的にソフトウェアにふれながら，一歩離れて見れば，やはり，時代の流れの中の，一要因であることを痛感！　ことコンピュータに関するならば，ケースに応じた傾向と対策に加えて，未来に向けて明確な方針づけを考えてゆきたいものですね．

咲間　繁

SEA主催の，** 研究会，** セミナー等に参加された場合は，一度，会後の飲み会に参加してみて下さい．本会よりおもしろくなるのはうけあいです．　SEにとって，ノミ（飲み）ケーションは，大切と感じております．

河村一樹

情報処理業界で，今，必要とされている人材は，ＳＥ，ＫＥ，そしてＩＥです．ＩＥとは，インフォメーションエンジニアの略であり，私が，提唱している職種です．そして，ＩＥ養成の為の新設科（情報コンサルティング科）を６３年度から，本校で開設する予定です．宜しくお願いいたします．以上，ＰＲをかねて。

藤野晃延

IEEE/Trans SE が，３－５年前の原稿を載せてひんしゅくをかっているが，　SEAMAIL も内容にこだわり過ぎて，時期を逸した感がある．ある程度，Timely にならないと，「会員」のために役立たないのではないか．また，文章のみ多く，一見して内容を理解する事がむずかしい．読者を引き込むような Layout も考えてほしい（両立は大変でも頑張ってほしい）．

茂木進一

編集長を含め，あちらこちらでパニックが起きているようですネ．私を含めまわりを見ると，同じような状態が続いています．ハード・ソフトの進歩した分は，我々にとってあまりメリットをもたらしていないようです．どうやったらヒマを作り出せるか，コンサルしてくれる AI は，どこかにないかなー．

牛島正人

情報特殊・一種などの情報処理技術者向けの，資格試験がありますが，これらの企業内での評価は，どうなのでしょうか．また，実力と資格には，どの程度の相関があるのでしょうか．

宮下高弘

私のように，ユーザーのＥＤＰ部門で働いていると，どうしても，外の情報が入りにくい．そこで，なるべく多くの情報紙を読むようにしている．今後は，セミナーやワークショップのレポートだけにとどまらず，誌上討論会や会員の皆様のシステム作成上の，苦労話，紹介など，面白い紙面作りをお願いしたい．そのためには我々会員も，積極的に投稿を行っていかなければ・・・しかし，仕事は忙しいですね．

豊田隆志

私の会社は，コンピュータなんぞにはとても縁遠い会社なので，「開発環境を考える必要がある」と言っても，誰も何も言ってくれません．今のところ一人で，どうすればよいか，試行錯誤しているところです．こんな私の助けとなるような企画も考えて下さい．私は，雑談の多いセミナ

ーが大好きです．ハードウェアと密接な関係にある，ソフトウェアが持つ問題について，意見交換をしませんか（例えば，リアルタイム制御と，リアルタイムモニタ，リアルタイムモニタが使えないリアルタイム系，ハードウェアがない状態でのディバック，スーパバイザ・モードでの実行プログラムの開発等）．ハードウェアとソフトウェア，両方の能力が要求される境界領域です．いろいろな意見を期待したいです．

石原由紀夫

あけましておめでとうございます．本年も良い年でありますよう願っております．今年も宜しくお願いいたします．みなさん，Busyな方々ばかりで，頑張って下さい．私は，日常こそ大事なのではないかと思ったりしています．一人平均，一日に，何人の人と会って，話をしているのでしょう？新しいソフトウェアのアイディアは，どんな時に生まれているのでしょう．仕事以外のホビーは，どんな事をしているのでしょう．もっとも大きいマーケットである事務に対して，どれだけ取り組んでいるんでしょう．

三谷保夫

ソフトウェアと言っても，分野が広いので，自分が精通していない問題，（例えば，既成ソフトの使い勝手，問題点など）について，他の会員のみなさんが，情報提供できるようなQ＆Aのコーナーを設けてはどうか．もちろん，アンサーは，１－２号遅れということで．（現実的には，パソコン通信で，やるべき話かもしれませんが・・・）

床尾光彦

はじめまして，新米の会員です． SUN に憧れつつ，商売の為，しかたなく 98 やっています．オンボロマシンのプログラマに愛の手を！

巽　和弘

unix は，ソフト環境開発として優れているとは思うが，ターゲットとしての，通常のＥＤＰの世界との関係がよくわからない． UNIX 世界の人の話は，クローズな話が多く，一般から見ると，よくわからないのでは．

野中　哲

SEAに入会して良かったのは，いろいろな世界の人と，知りあえるようになったこと．多くの人たちと話していると，世の中を少しずつでも動かしてゆけそうな，気がしてきます． SEAMAIL を読んでいるだけの人は，いろいろな催しにどんどん出てきてほしいと思います．

田中慎一郎

会員番号順にやってみるのもいいのではないでしょうか？ 数行であれば，強制するのも悪い事ではないと思いますし，それをきっかけに，自主的に何か書いてくれるようになるかもしれません．日本人は，でたがり屋が多い反面，引っ込み思案も多いので，一度ひっぱり出せば，自信がでて（？），意外と常連になってくれるかもしれません（新聞の投書欄みたいに）．とにかく，人をひっぱり出すのも SEA の仕事だと思います．何事にもきっかけが重要だから．

（順不同）

Advance Program

Software Symposium '88

1988年6月8日(水) ～9日(木)

虎ノ門パストラル (東京・港区)

併設チュートリアル: 6月7日 (火) 機械振興会館 (東京・港区)

ソフトウェア技術の実践的側面についての経験や知識を交流しあうための貴重な場として，80年代初めから毎年開催されてきたこのシンポジウムも，8回目を迎えます.

今回は，SEAを取り巻く多くの方々の御協力を得て，これまでになく強力なプログラム委員会が編成され，結果として，きわめて魅力的なプログラムを組むことができたと，関係者一同自負しております. ちなみに，応募論文数は約60篇を越えました. そのうちから20篇を採択し，それに若干の招待論文を加えた形で8つのPaper Sessionを構成しました. プログラムの最後は，アーキテクチャおよびマネジメントに関する最新の技術動向を論ずるパネルで締めくくってあります. ソフトウェアの国際化時代に対応して，海外からも2人の招待講演者を迎え，昨年から始まったツール展示も内容をより充実し，第3のトラックとしてツール・プレゼンテーションの枠も用意しました. また，初めての試みとして，シンポジウムの前日に4つのチュートリアル (各半日)も併設することになりました.

正式のパンフレットでの参加案内は4月になってからですが，3月末を期限として，割引価格での予約募集を行います. この機会にぜひ，奮ってお申込下さい.

主催

ソフトウェア技術者協会 (SEA)

(社) 情報サービス産業協会 (JISA)

6/8(水)

	鳳凰の間		
9:00	受付		
9:30-10:00	Opening　開会あいさつ：実行委員長 Symposium Orientation：Program 委員長		
10:00-12:00	**基調講演**　The USC System Factory Project *Speaker*　Walt Scacchi (University of Southern California) USCでは，1981年から，数十人の大学院生を組織したソフトウェア工場(System Factory)を作り，外部の企業から現実の開発業務を受託することによって，各種の先進的ツールを取り入れた環境の効用や，適切なマネジメントのスタイルなどについて，多角的な実験を積み重ねてきてます．Scacchi教授は，このプロジェクトの企画・運営の中心人物です．今回の基調講演では，そうした具体的な経験を踏まえて，今後のソフトウェア開発・管理のあるべき姿や，産学間における技術移転のアプローチなどについて，お話していただきます（逐次通訳つき）． *Coordinator*　岸田孝一（ソフトウェア・リサーチ・アソシエイツ）		
12:00-13:00	昼食		
	鳳凰の間	葵の間	梅の間
13:00-14:30	**Session-A1：部品化 / 再利用** *Chair*：飯沢　恒 （三菱電機東部コンピュータ・システム） 類似性の概念と類推による検索について：古宮誠一（IPA技術センター） ソフトウェア部品の一考察：渡邊雄一，高手山光雄 （電力計算センター） 未定（招待論文）：松本吉弘（東芝）	**Session-B1：環境** *Chair*：林　香 （ソフトウェア・リサーチ・アソシエイツ） ツール統合と環境改善：酒匂　寛 （ソフトウェア・リサーチ・アソシエイツ） ロボットコントローラ用ソフトウェア開発におけるテスト用シミュレータの作成：平尾一浩（ヒラタソフトウェアテクノロジー） Smalltalk-80による視覚的UNIX環境：青木　淳（富士ゼロックス情報システム）	**Tool Presentation-1** *Chair*：未定 自動走行ロボット『晴信』：山梨大学 Julia集合：京都芸術短期大学 日本語フォント処理：慶應大学
14:30-15:00	休憩		
15:00-16:30	**Session-A2：Software Documentation** *Chair*：佐原　伸 （ソフトウェア・リサーチ・アソシエイツ） DDBへのアプローチ：元治景朝 （神戸コンピューターサービス） 大規模システムのドキュメント作成支援環境：田中正則 （ソフトウェア・リサーチ・アソシエイツ） TEXによるソフトウェア・ドキュメンテーション：倉沢良一（アスキー）	**Session-B2：分散環境 / Network** *Chair*：野村敏次（日本電子計算） MELMUXを用いたノンストップ・ネットワーク・システム：小林　正 （三菱電機東部コンピュータ・システム） 知的分散型システム開発環境DEEN：小池幸徳（日本システムサイエンス） 広域ネットワークの管理ー junetの経験からー：加藤　朗（東京工業大学）	**Tool Presentation-2** *Chair*：未定 Key3：大阪大学 Personal Database：徳島大学 OBJ2：電子技術総合研究所
16:30-17:30	休憩		
	鳳凰の間		
17:30-20:00	**情報交換パーティ** 司会：未定 あいさつ：未定		

Tool展示（鳳凰の間）：6/8(12:00-17:30)および6/9(9:30-12:00)

(1)大阪大学（松浦敏雄）	**対話型グラフィックス支援ツール（Key3）** on Sony NEWS
(2)京都芸術短期大学（幸村真佐男）	**CGアート（Julia集合，他）** on Appollo Domain & PC9800
(3)慶應大学（斎藤信男）	**日本語フォント処理プログラム** on Sony NEWS
(4)静岡大学（落水浩一郎）	**Prolog-Shellによる開発プロセス記述** on Sony NEWS
(5)電子技術総合研究所（二木厚吉）	**代数的仕様記述言語（OBJ2）** on SUN3
(6)東京工業大学（片山卓也）	**階層的関数型言語支援環境（SAGE）** on SUN3
(7)徳島大学（山本米雄）	**A Personal Database System** on PC9800
(8)山梨大学（森　英雄）	**自動走行ロボット『晴信』**
(9)University of Hawaii（宮本勲）	**Prototyper's Workbench** on Textronix 4406 & Mac II

6/9(木)

	鳳凰の間	葵の間	梅の間
9:30-11:00	**Session-A3: 定量化 / 管理** *Chair*: 大場　充（日本アイ・ビー・エム） ソフトウェアの障害データに基づく信頼性の分析：天池　学，高原　実（沖電気工業） 定量的プロジェクト管理の実践と評価：渡辺昌司（三菱電機東部コンピュータ・システム） エラー寿命とプログラマの能力：松本健一（大阪大学）	**Session-B3: Expert System** *Chair*: 熊谷　章（PFU） 知識ベース・ソフトウェア開発環境REFINEによる設計とその評価：西川典子，伊野　誠（ソフトウェア・リサーチ・アソシエイツ） 性能解析エキスパートシステムの開発事例：田中　智（日本電気ソフトウェア） CSRLによる土木工学用エキスパート・システム：永井光俊（日本電子計算）	**Tool Presentation 3** *Chair*: 未定 SAGE: 東京工業大学 Prolog-Shell: 静岡大学 Prototyper's Workbench: University of Hawaii
11:00-11:30	休憩		
11:30-13:00	**Session-A4: 教育** *Chair*: 河村一樹（日本電子専門学校） ソフトウェア技術者教育の逆説：君島　浩（富士通） プロフェッショナルSEの養成：宮沢修二（情報処理技術者教育センター） 未定（招待論文）：山本米雄（徳島大学）	**Session-B4: Formal Approach** *Chair*: 藤野晃延（富士ゼロックス情報システム） 通信ソフトウェア要求仕様作成支援システムESCORT：若原　恭，角田良明，伊藤　篤（国際電信電話） ソフトウェア開発法の記述について：野村研仁（大阪大学） 階層的関数型言語体系によるSDA概念モデルの記述とプロトタイピング：片山卓也（東京工業大学）	**招待講演** *Chair*: 岸田孝一（ソフトウェア・リサーチ・アソシエイツ） **中国におけるソフトウェア工学の研究** **張　然（上海・復旦大学）** 近くて遠い国・中国のソフトウェア事情がどうなっているのか，残念ながら，われわれにはあまり手持ちの情報がありません．このセッションでは，去る1月の環境ワークショップにも来日された経験を持つ講師から，学界における研究開発の動向，ソフトウェア業界の現状，開発環境の実態や技術者の分布など，文革以後の中国のソフトウェア状況について，さまざまな角度からのお話を『日本語』で伺います．
13:00-14:00	昼食		
14:00-15:30	**Panel-1: これからのソフトウェア・マネジメント** *Chair*: 岡田正志（日本電気ソフトウェア） *Panelists*: 大場　充（日本アイ・ビー・エム） 片山禎昭（日本システム） 鳥居宏次（大阪大学）	**Panel-2: 近未来アーキテクチャのインパクト** *Chair*: 熊谷　章（PFU） *Panelists*: 坂村　健（東京大学）<交渉中> 中野秀男（大阪大学） 未定（富士通）	
15:30-16:00	鳳凰の間 **Closing**　最優秀論文（服部賞）発表 Software Symposium '89 実行委員長あいさつ		

Software Symposium '88 委員会

実行委員長
高田佳彦（日本電子計算）

プログラム委員長
藤野晃延（富士ゼロックス情報システム）
林　香（ソフトウェア・リサーチ・アソシエイツ）

プログラム委員
浅川新一（東京システム技研）
天池学（沖電気工業）
飯沢恒（三菱電機東部コンピュータシステム）
岩丸良明（三井銀行総合研究所）
臼井義美（日本電子計算）
浦野義頼（国際電信電話）
大場充（日本アイ・ビー・エム）
岡田正志（日本電気ソフトウェア）
落水浩一郎（静岡大学）
川崎好文（協同システム開発）
河村一樹（日本電子専門学校）
岸田孝一（ソフトウェア・リサーチ・アソシエイツ）
久保宏志（富士通）
熊谷章（PFU）
木間教大（日本ソフトウェア開発）
斎藤信男（慶應義塾大学）
佐原伸（ソフトウェア・リサーチ・アソシエイツ）
芝原雄二（沖ソフトウェア）
新美論（構造計画研究所）
寺本雅則（日本電気）
鳥居宏次（大阪大学）
中野秀男（大阪大学）
野村敏次（日本電子計算）
春原猛（三菱電機）
別役高明（ジェーエムエーシステムズ）
村井進（フリーランス・テクニカル・ライター）
村山公士（アカデミーオブビジネス専門学校）
盛田政敏（神戸コンピューターサービス）
山田淳（東芝）

●併設チュートリアル（開催日：６月７日（火），於　機械振興会館）

No		テーマ	講師
W	9:30-12:30	OSIの現状と動向	浦野　義頼　（国際電信電話）
		いよいよ本格的なネットワーク時代の到来を迎えて，その実用化および普及の動向がますます注目されています．OSI（開放型システム間相互接続）プロトコルをとりあげ，技術の概要から最新の国際動向までを，わかりやすく解説していただきます．	
X	9:30-12:30	技術レビューとウォークスルー	岡田　正志　（日本電気ソフトウェア）
		ソフトウェア開発の効率化と品質改善のための具体的な方策として，ウォークスルー技術が多くの期待を集めています．ここでは，G.ワインバーグの名著「ソフトウェア技術レビュー・ハンドブック」の訳者である講師をお招きし，効果的な位置づけや導入策について考えます．	
Y	13:30-16:30	プログラミング能力の定量的評価	鳥居　宏次　（大阪大学）
		プロジェクトを成功に導く重要な要因は，構成メンバーのスキルであるといわれています．ここでは，講師が大学および企業で収集した定量的データの分析結果をもとに，開発現場における評価やマネジメンとのあり方を，参加者との討論を通じて探ることにします．	
Z	13:30-16:30	AIとソフトウエア工学	玉井　哲雄　（三菱総合研究所）
		ソフトウエア工学とAIの研究成果の融合が期待されています．ここでは，ソフトウエア技術者がAIにどう取り組むべきか，AIの何が当面役に立つか，将来には何ができそうかなど，技術の最新動向を紹介しながら，これからの応用の可能性について考えてみることにします．	

---------- Software Symposium '88 参加予約募集要領 ----------

●参加費用

種　別 →	Symposium			併設チュートリアル		
↓申込期日	会員(SEA/JISA)	非会員	学生	会員(SEA/JISA)	非会員	学生
3月31日以前	**30,000**	**40,000**	**4,000**	**12,000**	**16,000**	**1,000**
4月1日以降	35,000	45,000	4,000	15,000	20,000	1,000

（Symposiumは1名あたり，併設チュートリアルは1名1セッションあたり:単位円）

●申し込み及び参加方法

所定の申し込み票にご記入の上，下記宛て郵便またはＦＡＸでお送り下さい．折り返し受講票，請求書をお送りします．なお，申込後のキャンセルはお断わりします．

〒102 東京都千代田区隼町２－１２　藤和半蔵門コープビル５０５
ソフトウエア技術者協会　（ＳＥＡ）
Symposium '88　係
FAX 03-234-9454 (TEL 03-234-9455)

Software Symposium '88 参加予約申込書（申込日：＿＿＿月＿＿＿日）

氏名：＿＿＿＿＿＿＿＿　種別（○印）：*SEA* 会員（*NO.*＿＿＿＿＿＿）　*JISA* 会員　非会員　学生

会社（学校）名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿　役職：＿＿＿＿＿＿＿＿＿＿＿＿

住所：（〒＿＿＿＿＿－＿＿＿＿）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

ＴＥＬ：＿＿＿＿＿－＿＿＿＿－＿＿＿＿＿＿＿＿（内線＿＿＿＿＿＿＿＿）

参加種別：（該当欄にチェックしてください）：

[ ]　*Symposium* のみ

[ ]　*Symposium* ＋チュートリアル（参加希望テーマに○印）[*W*　*X*　*Y*　*Z*]

申込総金額　：＿＿＿＿＿＿円

海外研修団・参加者募集

10th ICSE

10 TH International Conference on Software Engineering

（第１０回ソフトウェア工学国際会議）

於

シンガポール

ソフトウェア技術者協会（ＳＥＡ）

このたび，第１０回ソフトウェア工学国際会議（ＩＣＳＥ）が，今春４月１１日から４月１５日の予定で，シンガポールにおいて開催されることになりました．この会議は世界で最も権威のあるソフトウェア・コンファレンスであり，ＩＥＥＥ *Computer Society*，ＡＣＭ（米国コンピュータ協会）／*SIGSOFT* および各国コンピュータ／ソフトウェア協会の後援によって，１９７５年以降開催されてきており，アジア地区では第６回（１９８２年）東京につづいて２回目となります．

ＳＥＡは我が国における唯一の後援団体として認められており，会員は特別価格で参加する特典があります．この会議のプログラム委員に落水浩一郎（静岡大学），岸田孝一（ＳＲＡ），寺本雅則（日本電気），鳥居宏次（大阪大学），二木厚吉（電総研），松原友夫（日立ソフトウエアエンジニアリング）などＳＥＡ会員の各氏が任命されており，キイノートスピーカに岸田孝一（ＳＲＡ），論文発表に落水浩一郎（静岡大学），片山卓也（東京工業大学），岸田孝一（ＳＲＡ），斎藤信男（慶應義塾大学），鳥居宏次（大阪大学），中川中（ＳＲＡ），二木厚吉（電総研），松尾正敏（ＳＲＡ）など，ＳＥＡ会員が大活躍をしています．また，ツール展示には，慶應大学，静岡大学，東京工業大学，ＳＲＡからの出展が予定されています．

このような機会に世界のソフトウェア技術の最新動向に直接触れ，意見交換をすることは，きわめて重要だと思われます．ＳＥＡでは，同会議に出席し，技術調査をすると共に，併設のツールフェアを見学するための研修団を計画しました．研修を有意義なものにするために一般の単なる視察旅行とはちがった形で，会議関係者との事前ミーティングを開催し，最終的には簡単なレポートをまとめることを考えております．

これからの技術動向を捕らえる上ではよい機会です．多数の方々にご参加戴きたくご案内申し上げます

募集要項

１．目的：シンガポール開催される第１０回 *ICSE (International Conference on Software Engineering)* に出席し，発表や討論，ツールの展示を通じての技術調査と研修．

２．期間：昭和６３年４月１０日（日）より４月１６日（土）まで６泊７日．
事前ミーティングは３月２５日（金）国内で実施．

３．費用：ＳＥＡ会員価格１名当たり　２１８，０００円　（一般　２４８，０００円）
（航空運賃［エコノミークラス］，宿泊料金［２人１室］，バス料金，手荷物料金，団体保険料を含み，食事代，会議参加料は含まれていません．２０名を基準に設定しており，参加人数の増減により変更する可能性があります．）
会議（約ＵＳ＄３００），およびチュートリアル（１日あたり約ＵＳ＄２００）への参加は別途登録料が必要です．

４．募集人数：２０名

５．申込方法：３月１１日（金）まで，下記あてにＦＡＸ，または郵送にてお申し込みください，折り返し請求書・事前ミーティングの案内など必要な資料をお送りします．なお，定員に成り次第締め切らせていただきますので，あらかじめご承知おき願います．

〒102 東京都千代田区隼町２－１２　藤和半蔵門コーブビル５０５
ソフトウエア技術者協会　（ＳＥＡ）
TEL: 03-234-9455 FAX:03-234-9454

問い合わせ：*SEA*事務局・海外研修団*(03-234-9455)*までお願いします．

６．旅行取り扱い：近畿日本ツーリスト・新宿南口支店

SEA ICSE10 海外研修団・参加申込書　（申込日：　月　日）

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿，（ふりがな：＿＿＿＿＿＿＿＿＿＿），会員種別：（会員：*NO*＿＿＿＿，一般）

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿部門：＿＿＿＿＿＿＿＿＿＿＿＿役職：＿＿＿＿＿＿＿＿

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

ＴＥＬ：＿＿＿＿－＿＿＿＿－＿＿＿＿（内線＿＿＿＿）ＦＡＸ：＿＿＿＿－＿＿＿＿－＿＿＿＿
海外渡航の経験：無し，有り（　年　月頃），　喫煙の習慣：無し，有り

10th ICSE 海外研修団
日程表

1．事前ミーティング（東京：昭和６３年３月２５日（金））
2．海外研修　（昭和６３年４月１０日（日）－４月１６日（土）：６泊７日）

日次	日程	発着		内容	宿泊
1	４／１０（日）	東京（成田空港）	11:00 (SQ-007) 発		

		シンガポール	18:30 (SQ-007) 着	入国手続き後，専用バスにてホテルへ	シンガポール泊
2	４／１１（月）			ICSE Tutorial 第一日	シンガポール泊
3	４／１２（火）			ICSE Tutorial 第二日 10th ICSE 事前登録	シンガポール泊
4	４／１３（水）			10th ICSE Conference 第一日	シンガポール泊
5	４／１４（木）			10th ICSE Conference 第二日	シンガポール泊
6	４／１５（金）			10th ICSE Conference 第三日	シンガポール泊
7	４／１６（土）	シンガポール	10:45 (SQ-008) 発	専用バス，空路，帰国	

		東京（成田空港）	20:00 (SQ-008) 着	入国手続き後解散	

利用予定航空会社：
シンガポール航空：SQ-007,SQ-008（いづれも台北経由）

宿泊先：
Westin Plaza & Westin Stamford
2 Stamford Road, Singapore 0617
TEL:338-8585

3．事務局：

ソフトウエア技術者協会　（ＳＥＡ）
〒102 東京都千代田区隼町２－１２
藤和半蔵門コープビル５０５
TEL: 03-234-9455 FAX:03-234-9454

ＳＥＡ会員状況（昭和 63 年 2 月 13 日現在）

正会員	1226名（1月21日から15名増）
賛助会員	37社（1月21日から2社増）

＜正会員の勤務地および居住地域分布＞

	勤務地域	居住地域
北海道＝	9	9
山形　＝	1	1
宮城　＝	3	2
岩手　＝	6	6
福島　＝	1	1
秋田　＝	1	1
新潟　＝	6	6
栃木　＝	6	5
群馬　＝	1	1
茨城　＝	7	9
埼玉　＝	17	79
千葉　＝	15	80
東京　＝	706	434
神奈川＝	95	228
長野　＝	33	33
富山　＝	3	3
福井　＝	1	1
石川　＝	2	2
静岡　＝	17	15
岐阜　＝	2	6
愛知　＝	40	33
和歌山＝	2	1
三重　＝	1	2
滋賀　＝	3	4
京都　＝	14	21
大阪　＝	137	108
奈良　＝	3	8
兵庫　＝	27	43
愛媛　＝	2	2
徳島　＝	3	3
岡山　＝	1	1
広島　＝	4	4
鳥取　＝	0	1
福岡　＝	12	12
熊本　＝	12	13
大分　＝	4	4
佐賀　＝	1	1
宮崎　＝	1	1
鹿児島＝	3	3
沖縄　＝	1	1

＜男女分布＞

男　＝	1134
女　＝	64

＜年齢分布＞

20以下　＝	0
20_24　＝	60
25_29　＝	290
30_34　＝	331
35_39　＝	299
40_44　＝	143
45_49　＝	57
50_54　＝	21
55_59　＝	10
60以上　＝	7

＜血液型分布＞

A型　＝	487
O型　＝	353
B型　＝	264
AB型＝	123

＜賛助会員会社名＞

ＩＮ情報センター，旭化成工業，インターナショナル・データ・リサーチ，伊藤忠エレクトロニクス，ＳＢＣソフトウェア，エムテイシー，サン・ビルト印刷，情報システムサービス，ジェーエムエーシステムズ，セントラル・コンピュータ・サービス，ソフトウェア・リサーチ・アソシエイツ，ニッポンダイナミックシステムズ，マイクロキャビン，三菱電機，三菱電機セミコンダクトソフトウェア，ＰＦＵ，リクルート，リコーシステム開発，近畿日本ツーリスト，構造計画研究所，神戸コンピューターサービス，経調，辻システム計画事務所，東海クリエイト，東電ソフトウェア，日進ソフトウェア，日本システム，日本システムサイエンス，日本能率コンサルタント，日立製作所，日立ビジネス機器，富士ゼロックス情報システム，富士通，富士通ビジネスシステム

（アイウエオ順）

ソフトウェア技術者協会
〒102 東京都千代田区隼町2-12 藤和半蔵門コープビル505
TEL.03-234-9455 FAX.03-234-9454