SEAMAIL

Newsletter from Software Engineers Association

Volume 5, Number 6-7 1990

目　次

第7回　夏のプログラミング・ワークショップ報告

編集者から		1
参加の呼びかけ		3
プログラム		5
参加者名簿		6
基調報告		7
ソフトウェア品質定量化枠組みが提案されるまでの経緯	松本健一	8
品質定量化の枠組みに基づく管理支援システム（SQUARE）の開発	松本健一	10
ソフトウェア品質は測定できるか？	大場　充	18
グループ討論の記録		21
ビジネス・グループ		22
七福神グループ		25
麻里ちゃんグループ		30
フィードフォワード・グループ		39
フィードバック・グループ		45
感想とポジション・ステートメント		49
プログラム委員		49
一般参加者		61

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

その主な活動は，機関誌 SEAMAIL の発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約200人にすぎなかった会員数もその後飛躍的に増加し，現在，北は北海道から南は沖縄まで，1700 名を越えるメンバーを擁するにいたりました．法人賛助会員も約 60 社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

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

代表幹事： 熊谷章

常任幹事： 落水浩一郎　岸田孝一　中野秀男　野村行憲　平尾一浩　深瀬弘恭　松原友夫

幹事： 青島茂　天池学　新井秀明　市川寛　臼井義美　江木康雄　岡田正志　大木統雄　大林正晴　片山禎昭　片山卓也　加藤重信　加藤重郎　窪田芳夫　杉田義明　田中一夫　玉井哲雄　玉川滋　鳥居宏次　中来田秀樹　中山照章　野中哲　野村敏次　林香　人見庸　藤野晃延　二木厚吉　北條正顕　松本省二　盛田政敏　山崎朝昭　山田淳　渡邊雄一

会計監事： 辻淳二　吉村成弘

分科会世話人 環境分科会(SIGENV)：田中慎一郎　渡邊雄一
管理分科会(SIGMAN)：大久保功　加藤重郎　野々下幸治
教育分科会(SIGEDU)：大浦洋一　杉田義明　中園順三
ネットワーク分科会(SIGNET)：青島茂　野中哲　久保宏志
法的保護分科会(SIGSPL)：能登末之

支部世話人 関西支部：臼井義美　中野秀男　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：小林貞幸　佐藤千明　細野広水
名古屋支部：岩田康　鈴木智
九州支部：植村正伸　小田七生　藤本良子　平尾一浩　松本初美　中島泰彦　後藤芳美

SEAMAIL Vol.5, No.6−7　平成2年8月31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）
〒160　新宿区四谷3−12　丸正ビル5F
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　1000円　（禁転載）

編集者から

第７回ソフトウェア品質の定量的把握ワークショップを「ソフトウェア品質定量化の枠組み - 実践的ソフトウェア・メトリクスの開発現場への定着を目指して - 」をテーマにしました。お集りになった方たちに、プログラム委員をつとめたメンバーから盛岡での討論の報告をしました。これも盛岡の行事が可能にしてくれたネットワーキングです。

二つは、第10回ソフトウェア信頼性シンポジウムです。1989.10.24 - 25 の二日間に大阪で行われました。盛岡のワークショップに参加してくださった方の中から４名の方にお願いして、それぞれのポジションのベースになっている現場体験を発表していただきました。つぎの４名の方です。

・鶴林　伸英さん（富士通九州システムエンジニアリング）
・前沢　伸一さん（日本システム）
・平松由美子さん（三菱電機東部コンピュータシステム）
・山崎　哲彰さん（ソフトウェア・コントロール）

４人の方の現場体験は多くの方に知っていただくに値すると判断し、特にお願いしたものです。４名の方は、そこでまた別の仲間に巡り合えたことになります。

４名の方へのお願いは思いつきで行った訳ではありません。ソフトウェア信頼性シンポジウムと、ソフトウェア品質の定量的把握の枠組みの間には深い関係があります。そのことも知って欲しいので若干説明します。定量的枠組みは、通産省工業技術院の事業のお手伝いをしている中から生れたものですが、この手伝いをしていた者たちの有志は、高信頼性ソフトウェア研究会なるものも生み出しました。そしてこの研究会が始めた活動がソフトウェア信頼性シンポジウムです。結果的にシンポジウムは枠組みを、産業界のソフトウェア開発現場に普及定着させる場の性格を強くもつようになりました。

ここいらの事情は、本報告書のために特に書き下ろしていただいた松本健一さんのレポート、「ソフトウェア品質の定量化フレームワークが提案されるまでの経緯」に、枠組みが生れ、発展してきた筋道の観点から書かれています。参考文献も示されています。収録したレポートの先頭にこれをおきましたので、ぜひお読みください。

ワークショップに参加することで、あるいはその報告書を手にすることで、ソフトウェア品質の定量的把握に枠組みを与え、そしてそれを現場の実践技術にしていくという、果てしなく続く長い長い道のりの営みに、皆さん一人ひとりが関与していることを自覚してください。松本さんのレポートは、皆さんにそれを呼びかけています。

歴史をつくる営みへの参画の仕方は様々でいいのですが、参画している人たちが分断されていては歴史をつくり動かす力は弱いものになります。盛岡に集ったように、あるいはまた大阪のシンポジウムにでかけて発言したように、機会をとらえてネットワークに連なる多くの人々との情報交換に努めてください。情報交換ネットワークは、情報の受け手だけでは機能しません。情報を発信する人が多くを得ることができるのがネットワークです。

盛岡でのワークショップが、盛岡市に居を構える岩手電子計算センターの皆さんのご尽力があって初めて実現しえたものであることを付加えておかなければなりません。ホテルや会場の手配、会食の場所の推薦、屋外での討論場所の推薦など、現地に疎い大部分の参加者のためにご尽力いただきました。記して感謝の意を表します。

この報告書が、ソフトウェア品質の問題に関心を有する方たちのネットワークを生み出し、成長させ、インタネットワーキングを媒介する触媒の一つとして機能することを願っています。

編集者の怠慢のために報告書の刊行が大幅に遅れたことを重ねてお詫びします。

プログラム委員長　　久保　宏志 @ ファコム・ハイタック㈱
TEL:　(03)264-1131(代表)
Internet:　kubo@kbhm.se.fujitsu.co.jp
kubo@plum.juice.or.jp
NIFTY-Serve:　PDG03202
日経MIX:　h.kubo
PC-VAN:　VMD83316

追伸：
実行委員長当時は、富士通㈱に所属していましたが、最近、ファコム・ハイタック㈱に転籍になりました。この紙面を借りてご連絡申上げます。富士通時代よりもはるかに深くソフトウェア品質の問題に関与することになりますので引続きお付き合いをお願いします。

第７回 夏のプログラミング・ワークショップ

ソフトウェア品質の定量的把握を目指して

参加者募集

主催：ソフトウェア技術者協会

後援：（社）情報サービス産業協会

早いものでこのワークショップ（若手の会）もすでに７回目を迎えてすっかり定着し，業界各社からも，次世代のエースを鍛える格好なトレーニングの「場」として高く評価されています．

昨年は武田信玄の亡霊に導かれて会場を八ヶ岳に移してみましたが，参加者およびスタッフの強い希望により，ふたたび北の都盛岡に戻って，「ソフトウエア品質の定量化」を話題に，実践的かつ具体的な議論を展開したいと考えています．

皆さんは，開発したソフトウェアの品質をどうやって評価し，プロジェクトの完了をどんな根拠にもとづいて判断していますか？　終わったはずが実はそうでなかったり，納期が来たためやむをえず仕事を打ち切ったとか，いろいろのことが起きていると思います．基本的な動作を確認した段階でとりあえず納品し，あとはユーザにバグを見つけてもらうなどといった笑えない状況も，現実には起こっているようです．そのような実際の事例を持ち寄って，現場での技術的実践を少しはましな状態に改善すべく，議論してみませんか？

聞くところによれば，「ソフトウエア品質定量化の枠組」とか称するものが，大阪大学の鳥居先生を囲むワーキング・グループによって提案され，世間の注目を集めているようです．そこで，今回のワークショップでは，この枠組の開発を実際に担当された方々を中心にプログラム委員会を構成し，実際に現場で苦労している若手技術者の目で，この枠組の実用性を点検するところまで，議論を進めてみたいと思います．

例年通り，意欲あふれる多くの若手技術者諸氏の積極的な参加をお待ちします．

開 催 要 領

１．期日： 1989 年 9 月 5 日（火）午後から 9 月 8 日（金）正午まで（3 泊 4 日）

２．会場： 盛岡市・ホテル東日本（盛岡市大通 3 丁目 3-18）
いわて共済ビル（盛岡市中央通 3 丁目 4-3）

３．定員： 35 名

４．参加資格： 年齢 30 歳以下，経験 2 年以上のソフトウェア技術者

５．費用： 60,000 円 (SEA 会員，JISA 会員)，80,000 円 (一般)
現地集合・現地解散とし，期間中の宿泊費（３食付き）および資料代を含みます．

６．スタッフ：

実行委員長：　野村行憲（岩手電子計算センター）

プログラム委員長：　久保宏志（富士通）

プログラム委員：　天池学（カシオ計算機）　飯沢恒（三菱電機東部コンピュータシステム）
熊谷章（PFU）　松村好高（SRA）　松本健一（大阪大学）　他

招待講演者（予定）：　大場充（IBM）

７．申込方法： 裏面の参加申込用紙に必要事項を記入の上，1989 年 7 月 20 日（木）までに，郵便または FAX で，SEA 事務局宛お送り下さい．申込者多数の場合には，プログラム委員会でポジション・ステートメントの内容を審査の上，参加者を選定させていただきますので，あらかじめ御了承ください

申込み先：

〒102　東京都千代田区隼町 2 - 12 - 505
ソフトウェア技術者協会
夏のプログラミング・ワークショップ係
Tel: 03 - 234 - 9455　Fax: 03 - 234 - 9454

***** 裏面に申込用紙があります *****

送付先:　　申込日付:　月　日

〒102　東京都千代田区隼町 2 - 12 - 505　ソフトウェア技術者協会
夏のプログラミング・ワークショップ係　Fax: 03 - 234 - 9454

第 7 回 夏のプログラミング・ワークショップ
参加申込用紙

氏　名: _____ (ふりがな) _____

年　齢: _____ (歳)　性　別: □ 男 □ 女　血液型: _____

会社名: _____

部　門: _____　役　職: _____

住　所: (〒　　) _____

TEL: (　　) − (　　) − (　　) 内線 (　　)

FAX: (　　) − (　　) − (　　)

これまでの主な開発経験:

ポジション・ステートメント:

ソフトウェア品質の定量的な把握や管理に関して，現状における主な問題点は何か，また今後どのような改善策を講じるべきかを，自分のこれまでの経験や知識に則して，なるべく具体的に書いてください (紙面が不足の場合は別紙を追加してかまいません).

第７回　夏のプログラミング・ワークショップ・プログラム

テーマ：ソフトウェア品質の定量的把握を目指して

９月５日(火)：
　基調報告(14:00 - 17:00)
　　松本健一：ソフトウェア品質の定量的把握に枠組みを与える
　情報交換パーティ(18:30 - 20:45)
　　自己紹介とポジション・ステートメントの発表
　グループ編制(20:45 - 21:00)
　　ポジションを層別して５グループに編制
　　プログラム委員もいずれかのグループに編入

９月６日(水)：
　基調報告(9:00 - 11:00)
　　大場　充：ソフトウェア品質を測ることはできる
　グループ討論(11:00 - 深夜)

９月７日(木)：
　グループ討論の中間発表(9:00 - 11:00)
　グループ討論(11:00 - 深夜)

９月８日(金)：
　グループ討論結果の発表(9:00 - 12:00)

第７回　夏のプログラミング・ワークショップ（若手の会）参加者名簿

［実行委員長］
野村　行憲（岩手電子計算センター）

［プログラム委員長］
久保　宏志（富士通（当時）、ファコム・ハイタック（現））

［プログラム委員］
天池　　学（カシオ計算機）
桜井　麻里（ＳＲＡ）
松村　好高（ＳＲＡ）
松本　健一（大阪大学）
山田　　淳（東芝）

［招待講演者］
大場　　充（日本アイビーエム）

［参加者（所属の後の数字は当時の年齢を表わします）］
岩井田浩章（電力計算センター，27）
白木　真二（立石電機，30）
白井　一好（三菱電機東部コンピュータシステム，27）
松田　正人（ソフトウェアコントロール，27）
根岸　　学（日立製作所，25）
平松由美子（三菱電機東部コンピュータシステム，24）
三宅　武司（日本電信電話，27）
吉山　徳博（富士通九州システムエンジニアリング，29）
鶴林　伸英（富士通九州システムエンジニアリング，27）
堀内　秀一（日本システム，30）
高田　義広（大阪大学，24）
前沢　伸一（日本システム，28）
後藤　　浩（シスプラン，24）
安喜　裕師（富士通徳島システムエンジニアリング，27）
山崎　哲彰（ソフトウェアコントロール，27）
山口　敬介（ケーシーエス，24）
湊　　幸雄（ケーシーエス，24）
伊佐　厚司（ケーシーエス，23）
根本　慶一（ＳＲＡ，26）
小池　　誠（ＳＲＡ，29）
岡田　正之（ＳＲＡ，27）
三田村英生（ＳＲＡ，28）
山城　英彦（ＳＲＡ，33）
門岡　春雄（ヒラタ・ソフトウェア・テクノロジー，24）
小森林正樹（岩手電子計算センター，29）
三浦　　博（岩手電子計算センター，28）
金沢美佐雄（岩手電子計算センター，29）
馬島　宗平（日本電信電話，29）
中川　明生（富士通，28）
高橋　良一（日本電子計算，28）
岩井　邦明（日本電子計算，22）
宮崎　　祐（日本電信電話，28）
荒井　雅一（日立ソフトウェアエンジニアリング，24）
河村　謙介（ＮＴＴ九州技術開発センタ，29）
吉田　　徹（岩手電子計算センター，28）

基 調 報 告

ソフトウェア品質の定量化フレームワークが提案されるまでの経緯

大阪大学　松本健一

今回のワークショップで議論の対象として取り上げた「ソフトウェア品質の定量化フレームワーク」は，通商産業省工業技術院の委託で日本規格協会が設けているソフトウェア専門委員会（主査：大阪大学 鳥居宏次）のメンバーが中心となって運営する，高信頼性ソフトウェア研究会での議論の結果の1つとして提案されたものである．以下では，「ソフトウェア品質の定量化フレームワーク」が提案され現在に至るまでの経緯を簡単に述べる．

フレームワークに関してはじめて議論が行われたのは，1987年8月に伊豆で開催されたIEEEソフトウェアメトリクス標準化試案検討会[3]である．この検討会では，IEEE Quality Metrics Standard Committeeが提案する“A draft standard for software quality metrics”[1]（以下，IEEE試案と呼ぶ）についてその有効性や妥当性が議論された．その結果，IEEE試案はソフトウェア品質の定量化に関して1つの統一的な概念上のフレームワークを与えるものであり，相異なる環境で仕事に携わるソフトウェア技術者，及び，研究者の間で，お互いの経験や知識を交換する上で重要な役割を果たすものであると評価された．但し，IEEE試案は文字通り試案であり，品質定量化の詳細については明確にされていない．また，アメリカの国防省に納入するソフトウェアを対象に検討されている試案を，日本のソフトウェア開発の現場にそのまま適用することは出来ないのではないかという意見も出された．そこで，日本のソフトウェア開発の実状にあった品質定量化のためのフレームワークとは一体どのようなものであるのか，IEEE試案をベースに検討を続けていくことが決定された．

1987年10月に行われた第6回ソフトウェア信頼性シンポジウムでは，IEEE試案が提案する品質定量化の手順についてより詳細に検討するためのパネル討論が行われた[4]．更に，1988年4月に行われた第7回ソフトウェア信頼性シンポジウムでは，企業におけるソフトウェアの開発現場にIEEE試案を適用した事例報告がなされ，フレームワーク適用上の問題点とその改善点について検討された．

開発現場への適用事例を検討する中で明らかになったIEEE試案の最大の問題点は，ソフトウェア品質をユーザ（カスタマ）の視点ではなく，管理者の視点で捉えている点であった．一般に日本の企業においては，ソフトウェアに対するユーザの満足の度合でソフトウェアの品質を評価すべきであると考えられている．ISO/TC97/SC7で作成されているソフトウェア品質定量化の試案[5]（以下，ISO試案と呼ぶ）では，こうした考え方が反映され，ソフトウェアの品質はユーザの視点から定義されている．また，ハードウェアの分野で広く利用されている品質機能展開[2][7]をソフトウェアに適用する試みにおいても，品質はまずユーザの視点で捉えられている．

1988年8月に八ヶ岳で行われたソフトウェア信頼性の定量化に関するワークショップでは，ソフトウェア品質をユーザの視点で捉えるISO試案，及び，品質機能展開の

考え方をIEEE試案に加味した新たなフレームワークが提案され，その有効性を検討するためのグループ討論が行われた．更に，1988年10月に行われた第8回ソフトウェア信頼性シンポジウムでは，企業におけるソフトウェアの開発現場に提案されたフレームワークを適用した事例報告がなされ，フレームワーク適用上の問題点とその改善点について検討された．

その後，検討結果に基づく改良が加えられたのが現在のフレームワークである．なお，1989年4月に行われた第9回ソフトウェア信頼性シンポジウムでは，フレームワークを適用することにより，ソフトウェア品質が効率よく実現できる事例報告がなされた．また，フレームワークに基づく品質表管理システムSQUARE[9]の開発も進められており，プロトタイプシステムはいくつかの企業において現在使用されている．

参考文献

[1]IEEE Quality Metrics Standard Committee, P1061: “A draft standard for software quality metrics”, Document number Ver.12 (1987).

[2]情報処理振興事業協会技術センター: “ソフトウェア品質評価モデルの調査報告書（その２）” (1987).

[3]鳥居 他: “IEEEソフトウェア・メトリックス標準化試案検討会”, SEAMAIL, 2, 10-12, pp.11-51 (1987).

[4]鳥居 他: “第6回ソフトウェア信頼性シンポジウム報告（その１）”, SEAMAIL, 3, 3-4, pp.2-17 (1988).

[5]日本規格協会: “マンマシンインターフェイス及びソフトウェア評価の標準化に関する調査研究報告書” (1988).

[6]鳥居宏次：“ソフトウェア・メトリックスの意義と方法論”, 昭和63年電気・情報関連学会連合大会, 29-1, pp.5.19-5.22 (1988).

[7]赤尾洋二編：“品質機能展開の実際”, 日本規格協会 (1988).

[8]鳥居 他：“ソフトウェア品質定量化のための枠組の検討報告”, 信学技報, R88-32, pp.57-64 (1988).

[9]高田 他：“ソフトウェア品質定量化のための枠組みと品質表管理システムSQUAREの開発”, 1989年電子情報通信学会秋季全国大会, p.6-155 (1989).

ソフトウェア品質定量化のための枠組みに基づく品質表管理支援システムの開発
- SQUAREシステムの開発 -

高田義広　松本健一　菊野　亨　鳥居宏次

大阪大学　基礎工学部
e-mail:takada@tori2.ics.osaka-u.junet

あらまし

ソフトウェア品質定量化のための枠組みSQUARE（Software Quality Assurance Based on Requirements）は，ソフトウェア信頼性シンポジウムにおける討論に基づいて提案されている枠組みである．この枠組みの特長は(1)ユーザ，管理者，技術者の３視点を導入していること，(2)品質表を利用して作業の効率化を目指すこと，である．しかし，枠組みSQUAREで利用する品質表の形式については確定しているが，そこに蓄積すべき内容については実用的なものは得られていないのが現状である．

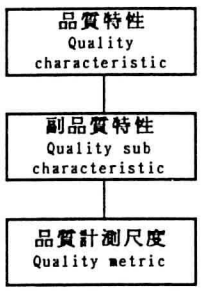
充実した品質表を得るためには，実際のソフトウェア開発に枠組みを適用して，段階的に品質表を改良する必要がある．そこで，品質表を管理するシステム（SQUAREシステムと呼ぶ）の設計を行ない，プロトタイプシステムの開発を行なった．

枠組みSQUAREに基づくソフトウェア開発手順は８ステップの基本操作から構成されている．開発したSQUAREシステムは，各ステップにおける基本操作の支援を行なうコンポーネントと，それらのコンポーネントを起動するユーザインタフェースなどから構成される．各コンポーネントに対しては，種々のソフトウェア開発現場に適合させるための調整，拡張の機能が組み込まれている．

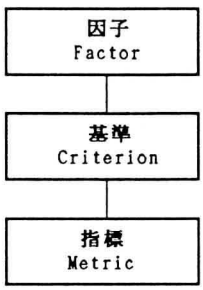
今後の課題としては，SQUAREシステムを利用して枠組みの適用事例を収集すること，その適用事例に基づいて充実した品質表を獲得すること，などがある．

枠組み-SQUARE-の概要

A．関連する枠組み　1

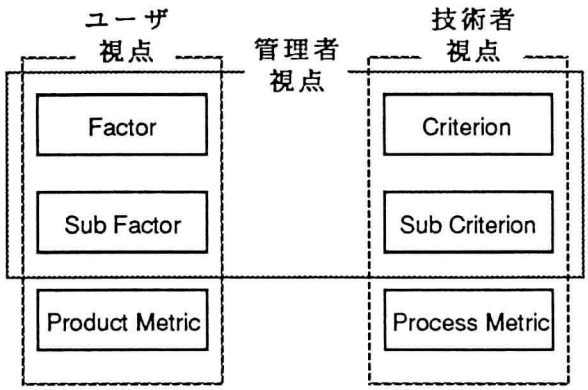


ISO試案
（ユーザ視点）



IEEE試案
（開発者視点）

B．枠組み-SQUARE-の３視点　2



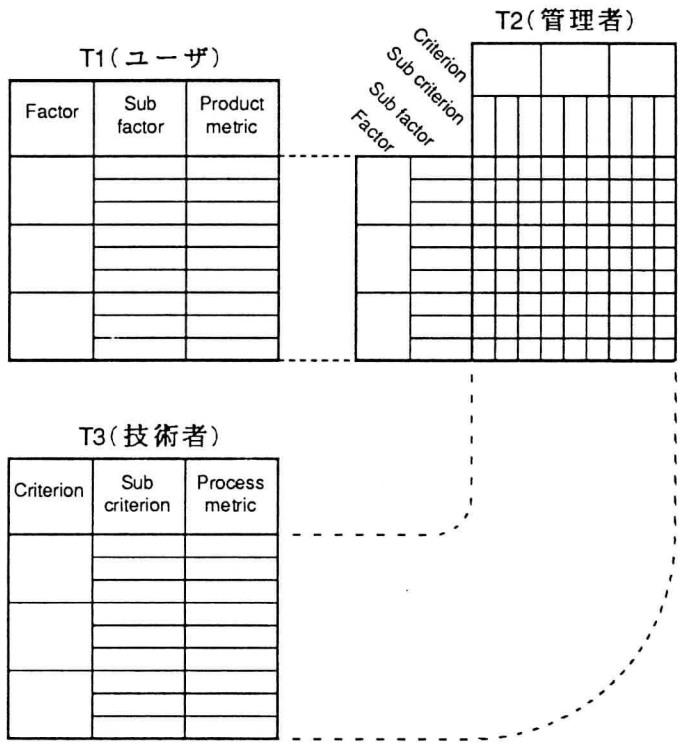
Factor. . . ユーザから見たソフトウェアの性質
Criterion. . . ソフトウェア開発技術
Sub factor. . . Factorを詳細化したもの
Sub criterion. . . Criterionを詳細化したもの
Product metric. . . Sub factorの評価尺度
Process metric. . . Sub criterionの評価尺度

C．品質表　3

	汎用	特定プロジェクト用
ユーザ	T1 Factor Sub factor Product metric 対応表	t1 Factor Sub factor Product metric 対応表
管理者	T2 Factor Criterion 対応表	t2 Factor Criterion 対応表
技術者	T3 Criterion Sub criterion Process metric 対応表	t3 Criterion Sub criterion Process metric 対応表

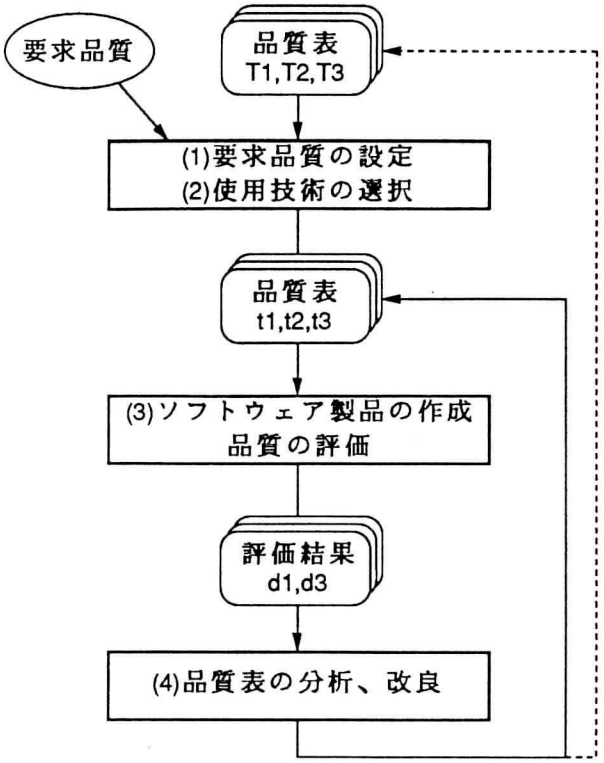
特定プロジェクト用の品質表は、汎用の品質表から必要な項目を抽出し、それに項目を追加、変更して作成する。

D．品質表（ユーザ、管理者、技術者）　4

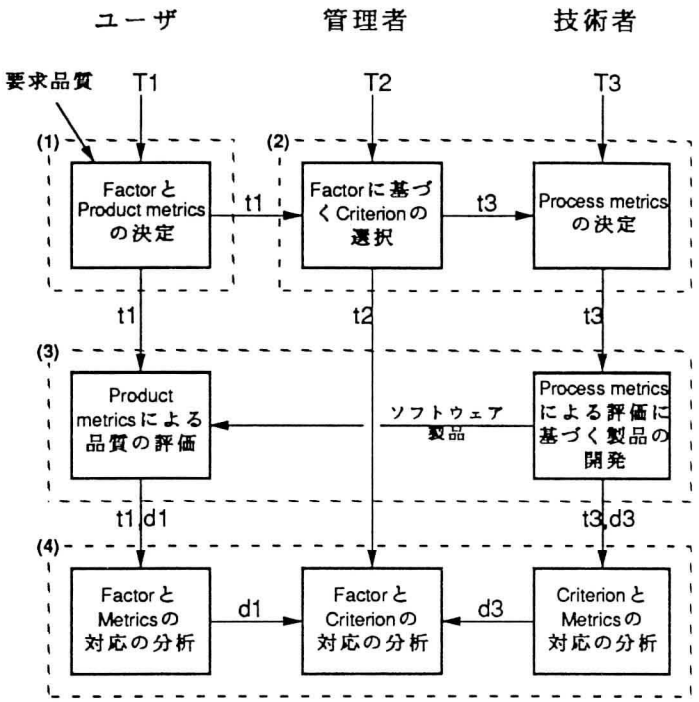


枠組み-SQUARE-の適用

A．ソフトウェア開発手順（概要） 5



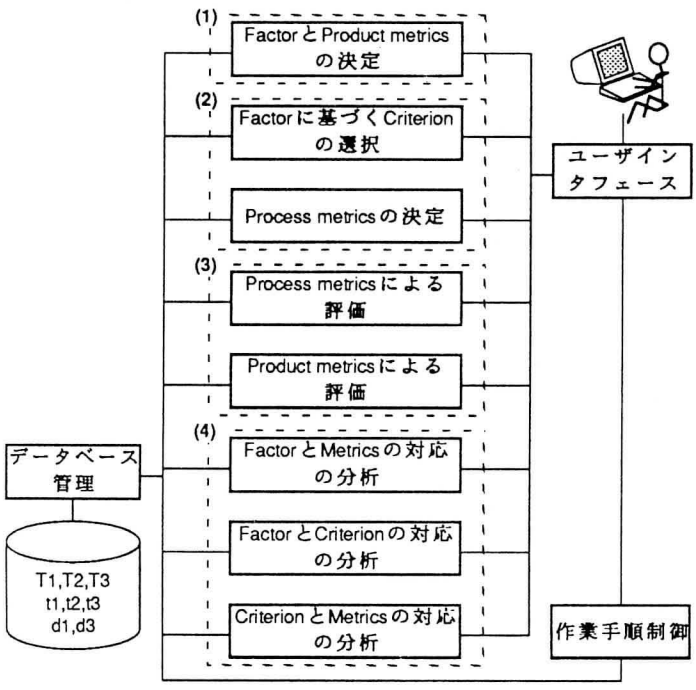
B．ソフトウェア開発手順（詳細） 6



品質表管理システム-SQUAREシステム-

A．システム構成 7

作成したSQUAREシステムプロトタイプは、ソフトウェア開発で利用する品質表の管理、改良を行うためのデータベースシステムである。

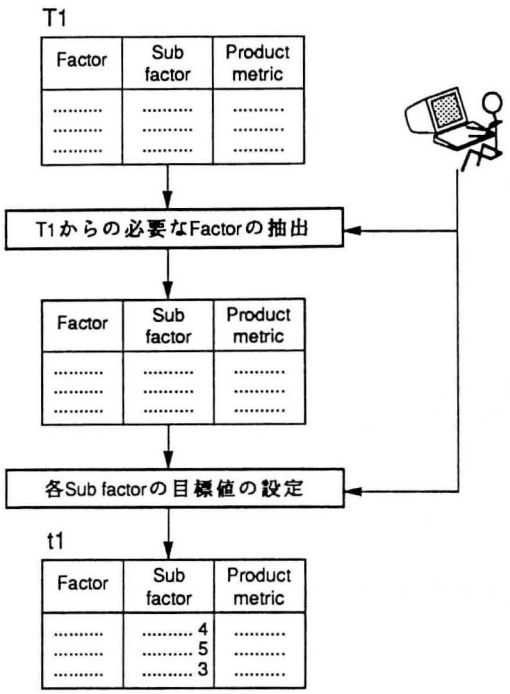


B．主な支援機能 8

(1) Factorと Product metrics の決定	ユーザの入力により、T1から必要なFactorを抽出し、t1を作成する。
(2) Factorに基づく Criterion の選択 Process metrics の決定	t1に基づいてT2より必要な項目を抽出し、t2を作成する。 t1,t2に基づいてt3を作成する。
(3) Process metrics による評価 Product metrics による評価	Metrics の評価値の入力により、Factorと Criterion の評価値を算出し、d1,d3を作成する。
(4) Factorと Metrics の対応の分析 Factorと Criterion の対応の分析 Criterion と Metrics の対応の分析	ユーザ、管理者、技術者の入力により、t1,t2,t3を変更する。 更に、必要ならばT1,T2,T3を変更する。

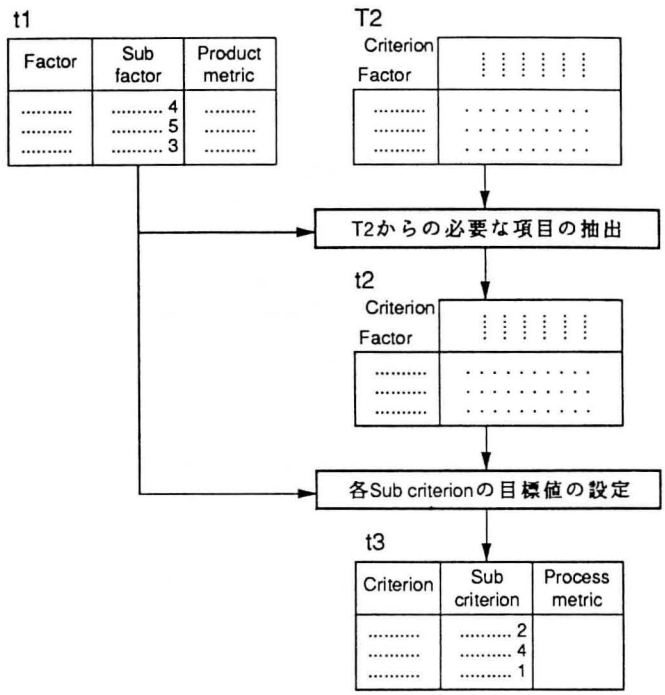
要求品質の設定
A . FactorとProduct metricsの決定

9



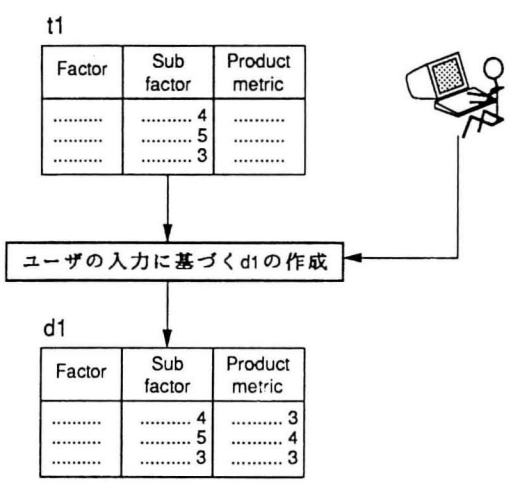
使用技術の選択
A . Factorに基づくCriterionの選択

10



品質の評価
A . Metricsによる評価

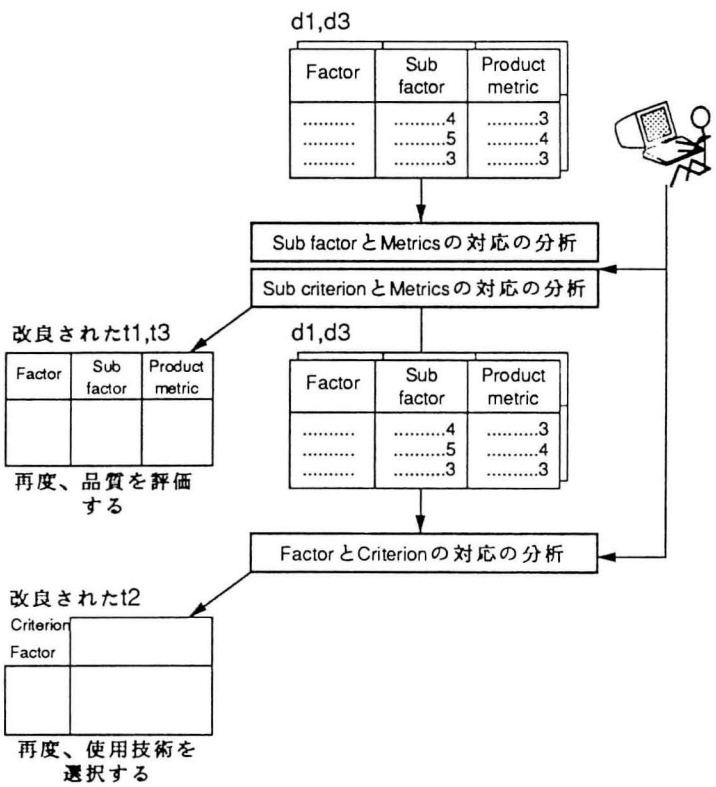
11

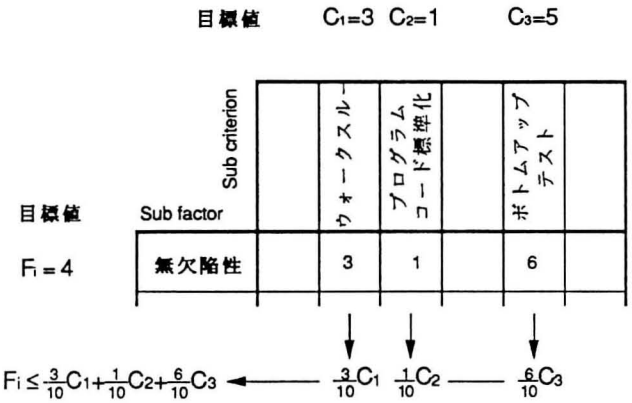


Sub factorとMetricsの対応は１対多なので、Sub factorの評価値はMetricsの評価値を加重平均して算出する。

品質表の分析、改良
A . FactorとCriterionとMetricsの対応の分析

12





解が複数個ある場合は、コスト計算を行ってコスト最小のものを選択する。

本システムは、種々のソフトウェア開発現場に適合させるための調整、拡張の機能を持つ。

(1)品質表T1,T2,T3の改良

(2)各コンポーネントの調整、拡張

例

・品質評価コンポーネントと評価ツールの結合による品質評価の自動化。

・品質表の分析、改良コンポーネントへの統計的手法の導入。

・作業手順制御部の変更による作業手順の調整。

品質表T1（ユーザ用）の例

Factor	Sub Factor	Product Metric
機能性	合目的性	ユーザ改良要求率
	正確性	操作説明書と実動作の合致度　　機能動作率
	セキュリティ	不正操作検出率　　機密保護率
	互換性	機能合致度　　データの処理可能率
	接続性	接続動作機能率
信頼性	無欠陥性	平均故障発生間隔
	誤り許容性	ダウン発生率　　誤入力,誤操作検出率
	可用性	稼働率
使用性	理解性	使用法誤解率
	習得容易性	オペレーション習熟時間
	操作性	使用準備容易化率　　タッチ回数　　収束作業時間 オペレーションミス平均時間間隔
	対話性	
効率性	実行効率性	ターンアラウンドタイム　　レスポンスタイム トランザクション処理件数　　ジョブ処理件数
	資源効率性	CPU使用率　　主記憶使用率　　ファイル使用率
保守性	修正容易性	
	拡張性	
	テスト容易性	
移植性	導入容易性	
	再利用性	
	ハードウェア独立性	
	ソフトウェア独立性	

品質表T2（管理者用）の例

Factor	Sub Factor	Criterion	設計技術								コーディング技術				テスト技術				ドキュメント作成技術	
		Sub Criterion	設計標準化	設計レビュー	プログラム再利用	データ暗号化	故障回避機能装備	故障診断機能装備	コマンド形式標準化	出力形式標準化	コード標準化	コードレビュー	プログラム構造簡単化	コード自動生成	テストデータ自動生成	機能テスト	ボトムアップテスト	エラーレポート作成	ドキュメント作成標準化	ドキュメント自動生成
機能性	合目的性		2	8																
	正確性		1		2											7				
	セキュリティ					10														
	互換性				10															
	接続性			3	7															
信頼性	無欠陥性										1	4				5				
	誤り許容性						10													
	可用性						2	8												
使用性	理解性								5	5										
	習得容易性								5	5										
	操作性			2					8											
	対話性			2						8										
効率性	実行効率性											5					5			
	資源効率性											5					5			
保守性	修正容易性										4								6	
	拡張性		5								3		2							
	テスト容易性										7								3	
移植性	ハードウェア独立性			5								5								
	ソフトウェア独立性			10																
	導入容易性																		10	
	再利用性				8								1						1	

品質表T3（技術者用）の例

Criterion	Sub Criterion	Process Metric
設計技術	設計標準化技術	設計標準化率　要求仕様展開率
	設計レビュー技術	設計レビューでのエラー発見率
	プログラム再利用技術	プログラム再利用率
	データ暗号化技術	データ暗号化率
	機密保護技術	機密保護率
	故障回避機能装備技術	故障回避機能装備率
	故障診断機能装備技術	故障診断機能装備率
	自習機能装備技術	自習機能装備率
	コマンド形式標準化技術	コマンド形式標準化率
コーディング技術	コード作成標準化技術	コメント記載率
	コードレビュー技術	コードレビューでのエラー発見率
	コード自動生成技術	コード自動生成率
	プログラム構造簡単化技術	プログラムの複雑度　平均モジュールサイズ
単体テスト技術	テストデータ自動生成技術	テストデータ自動生成率　テストデータ数
	仕様に基づくテストデータ作成技術	テストカバレッジ率　推定残存エラー数
	エラーレポート作成技術	エラーレポート作成率
結合テスト技術	テストデータ自動生成技術	テストデータ自動生成率　テストデータ数
	仕様に基づくテストデータ作成技術	テストカバレッジ率　推定残存エラー数
	エラーレポート作成技術	エラーレポート作成率
ドキュメント作成技術	ドキュメント作成標準化技術	エラーレポート記載率　図表記載率
	ドキュメント自動生成技術	ドキュメント自動生成率

ソフトウェア品質は測定できるか

日本ＩＢＭ　東京基礎研究所

大場　充

１．管理とは：一般論

一般に管理とは、組織の目的を達成するために必要となる活動を計画し、その計画を実施するために必要となる組織の資源を組織し、さらに活動の実施状況や成果を評価する総合的な行動である。すなわち、管理活動は、『計画』、『組織化』、『評価』の三段階の作業からなる。また、管理活動は、単にこれら三つの作業の単純な組合せではなく、評価の結果が次の計画に影響を与えるようなフィードバック・ループを形成している。従って、管理活動の軌跡を仮想的に考えれば、それは直線的ではなく、螺旋構造になる。『計画』に始まり、『評価』で終わる一つの管理周期を、『管理サイクル』と呼ぶ。

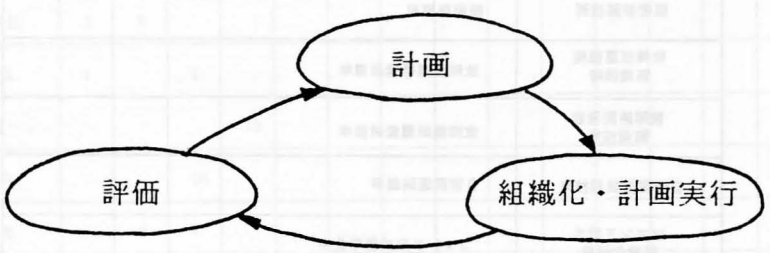


図１．管理サイクル

組織の活動を計画するためには、考えられるいくつかの代替案について、それらの予期され結果を事前に予測し検討しておくことが重要になる。また、実施された活動の進捗状況や最終的な成果を評価するためには、活動の成果を測定し、その測定結果に基づいて、客観的に評価を決定することが必要である。そのためには、活動の成果の品質と活動を実施するために投入した資源の量を計る基準と、計量のためのモデルが必要になる。

ソフトウェア産業以外の分野では、活動の成果物の品質や、その生産活動に費やされた資源の量を計る尺度や基準が決まっていることが多く、また、それらの測定も物理的に簡単である場合が多い。例えば、電気部品の生産ラインであれば、部品の仕様によってその設計品質は決まっており、生産ラインにおいては、その仕様に合致しているかどうかが、検査によって確認されている。検査の結果に基づいて、生産ラインでは不良率を計算し、生産ラインの状態を評価する。また、単位時間当たりの生産量によっても、生産ラインの状態を評価している。

電気部品の組み立てラインなどでは、ラインの中の個々の作業についても、それを細かな動作の組合せに分解することによって、過去の実績を参考に、作業時間を事前に予測し、そして実際の作業時間を計測することによって、ある程度科学的な管理が行なわれている。また、検査のための手順や道具も、標準的なものが開発されており、その利用が義務付けられている。

これらの科学的な管理の理論的な研究は、英国での産業革命以来、アダムスミスの労働価値説を基礎に、米国でのテイラーによる動作時間研究へと発展し、更に他の人間工学の分野における現在の、人間の動作に関するミクロな研究に至っている。興味深いことは、アダムスミスの労働価値説の研究に対して、コンピュータの元祖、バベージが大きく貢献していた点である。バベージ自身労働の価値に関する、数学的なモデルを提案した本を書いている。

ソフトウェア産業の外の世界では、このように人間の数百年にわたる英知の結晶が、

主としてハードウェアの生産の管理手段として、応用されてきている。ソフトウェアでは、その実体がよく理解されていない為、人間の歴史的資産である、管理のための技術に関する英知を、うまく利用できないまま、今日に至っているのではないか。

２．ソフトウェアのための管理の科学とは

ソフトウェアの開発における管理の問題も、他の産業における生産管理や品質管理の問題と共通点がある。しかし、ソフトウェアが物理的な構造をもたないため、他の産業で成功した例をそのまま適用してもうまくゆかないことが多い。また、ソフトウェア自身について言えば、ブルックスが『銀の弾丸』で指摘したように、今日のソフトウェアは、人類がこれまでに作り上げてきたものとは比較にならないほど複雑なものになってきている。

ソフトウェアの開発における管理の問題を考えるとき、最も重要なことは、この『目に見えない構造物をどうとらえるか』の問題と、『人間による複雑な問題の解決をどう管理すべきか』の問題についての、基本的な考え方（哲学）を確立することである。特に、この第一の問題は、現在までのところソフトウェアに特有な問題であると考えられている。第二の問題は、ソフトウェアだけでなく、複雑な構造物を設計する場合や、複雑な問題を解く場合に共通する問題である。この第二の問題に対しては、今日『分割統治』という考えかたが、一般的な解決法として知られている。

ソフトウェア固有の問題として重要なものに、『問題に対する物理的な制約がない』という性質も重要である。これもブルックスが『銀の弾丸』で指摘している問題であるが、従来からよく知られていた問題の一つである。すなわち、他の産業分野では、生産物はほとんど全てが、自然の、主として物理法則に従っている。従って、その規模や実現方法には、物理的な限界がある。ところが、ソフトウェアそれ自身は、論理的な構造物であるため、物理法則に制約されることはない。むしろ一般的には、人間がかってに決めた、社会のルールや法律に従わなければならない。しかし、これらの約束ごとは、自然の法則とは違って、人間がかってに変えることができる。従って、ソフトウェアには、その実現において、必ず従わなければならない絶対的法則が存在しない。選択の自由度は、無限にあるといえる。

このような性質をもつソフトウェアの開発を管理することの目的も、他の生産物の開発の管理のそれと、基本的に同じである。ソフトウェア開発管理ために、計画作業として重要なものは、『工数の見積り』である。次の組織化の作業として重要なものは、『工程の設計』である。そして、評価の作業として重要なものは、工程の成果物の『品質の評価』であり、またその生産工程の『生産性の評価』である。これらの管理の問題をむずかしくしている理由が、上述のソフトウェアの基本的な性質である。特に、それらの性質のうち、『見えない』ことと、『自然の法則に制約されない』ことの二つの基本的性質である。

ソフトウェアの開発をある程度科学的に管理するためには、上述した二つの基本的な性質に原因する問題を解決するための手段が必要となる。その手段として最もパワフルなものが、モデルとメトリクスである。メトリクス（計量）とは、ソフトウェアの大きさ、複雑さ、品質の良さ、生産に投入した資源の量などを計測する手段をである。また、モデルとは、ものの因果関係を科学的に客観的に説明するための道具である。ハードウェアの場合には、『目に見える』ために人間の直感をうまく利用することができ、その複雑さを管理できる程度に押えておくことが容易であった。そして、直感が正しく働かない場合には、自然の法則に反するため、それを実現できないことが誰の目にも明らかにわかった。ソフトウェアでは、『目に見えない』抽象性のため、人間の直感をうまく利用できない。従って、『目に見えない』実態を量で表現することにより、人間の直感を利用できるようにすることが必要になる。そのための手段がメトリクスである。また、モデルは、現在の状態から、将来の状態を予測するための、原因と結果の連鎖を生成するためのルールを記述したものである。このルールは、現状を分析するための道具としても使うことができる。すなわち、現在の状態を結果とした場合の、考えられる原因の連鎖を時間の経過とは逆方向に辿ることによって、一つの推論を構成することができる。

モデルのこのような時間の可逆性を利用することによって、現実の現象についての理解を深め、現象を制御（管理）するための手掛かりとすることができる。

３．ソフトウェア品質の測定

ソフトウェアの品質を測定するためには、ソフトウェアの品質の概念を測定可能な要素に分解し、それぞれの要素について尺度（測度）を設定することが必要になる。一般には、品質の概念を直接測定できるような要素に分解することは困難である。従って、品質の概念を、いくつかの中間的な概念に分解する。このようにして分解された、中間的な品質の概念を、『品質特性』と呼ぶ。その品質特性を直接測定する尺度を設定できる場合もあるが、そのような場合は一般的ではない。一般的には、更にその下位にあるより詳細な中間概念に分解することが必要になる。そのような概念を、『副品質特性』と呼ぶ。このような中間概念への分解を繰り返して、測定可能な尺度を設定できるレベルにまで落としてゆき、その最後のレベルで尺度を定義するのである。

そのようにして定義されたソフトウェアの品質概念の例として、現在ＩＳＯで議論されている『ソフトウェア品質特性』の体系がある。そのＩＳＯ試案では、ソフトウェアの品質を、機能性、信頼性、効率性、使いやすさ、保守性、移植性の６つの品質特性で構成している。各品質特性は、更に『副品質特性』に分解されている。その例として信頼性をとれば、成熟性、耐エラー性、利用可能性の３つの副品質特性に分解されている。この構造化が妥当なものかどうかについては、議論の残るところである。現時点では、一つの案と考えるべきであろう。モデルとして考えれば、説得力に乏しい。個人的な見解で言えば、大場・込山による、統合性、利用可能性、安全性の概念を副品質特性とする構造の方が説得力がある。

品質特性が定義され、更にそれが副品質特性に分解されて、それを直接測定できる尺度を設定できるレベルに到達すると、合理的な計測尺度（メトリクス）を決定することが必要になる。メトリクスの定義は、概念の定義から直接導出されることが望ましいが、それが不可能な場合には、概念を間接的に説明する尺度を定義することが必要になる。例えば、ある品質特性を説明する要因のひとつとして、プログラムの複雑度を計測する尺度が必要であるとする。この複雑度を直接測定する尺度を定義することは容易でないので、それを間接的に説明するプログラムの大きさをその代用とすることができる。そのような特性を特に『代用特性』と呼ぶこともある。実際にプログラムの大きさを測定する『コード行数』や『ステップ数』などは代用特性のメトリクスである。ここで重要なことは、メトリクスは概念を説明するものでなくてはならない。プログラムの大きさを測定する場合に、プログラム作成時間をとるなど、他の要因を含んで計測する尺度を採用することは、誤りである。互いに直交する要因を、独立して計測する尺度を選択することが重要である。

設定された尺度を用いて計測した後、ソフトウェアのある品質特性を評価する。このとき、尺度が絶対的なものであれば、その値によって、直接評価することができる。しかし、尺度が相対的なものである場合には、その値の解釈が重要になる。その値の解釈は、現在の状態から原因を推論して得られる場合と、逆に現在の状態を原因として、将来の状態を予測（推論）することによって得られる場合がある。このどちらの場合にも、モデルが必要になる。そのようなモデルは、現実を抽象して得られるものであり、現実を詳細に観察して、不必要な詳細を省いた後に、本質的な法則として残ったものである。単に現象をまねるだけのモデルは、本質的な法則が誤っている可能性もあり、つねに正しい結論に到達するとは限らない。モデルはその内容をよく吟味して利用することが重要である。『モデルは、魔術の道具ではなく、科学のどうぐである。』

以上

グループ討論の記録

ビジネスグループ（第７回　夏のプログラミングワークショップ）のまとめ

討論メンバー

	氏　名	会　社　名	地区
＊	松村　好高	ＳＲＡ	東京
＊	野村　行憲	岩手電子計算センター	岩手
＊	大場　充 （特別参加）	ＩＢＭ	東京
1	安喜　裕師	富士通徳島エンジニアリング	徳島
2	伊佐　厚司	ＫＣＳ	神戸
3	岩井　邦明	ＪＩＰ	東京
4	岩井田　浩章	電力計算センター	東京
5	門岡　春雄	ＨＳＴ	熊本
6	河村　謙介	ＮＴＴ九州技術センター	熊本
7	白井　一好	三菱電機東部コンピュータシステム	東京・神奈川
8	前沢　伸一	日本システム	東京
9	三田村　英生	ＳＲＡ	東京

I　討論テーマ

『ソフトウェア品質の定義としてなにを基準とするか』

II　討論結果

『ソフトウェアの品質には基準が必要である』

III 討論過程のまとめ

1　ソフトウェア品質の定量的把握（スタート）

2　開発現場からの品質への視点
- 品質の定量化は可能か
- システム利用者が素人女性
- 保守作業とソフトウェアの品質
- ソフトの品質＝仕様の達成度？
- 品質管理とコスト
- 保守における影響分析
- ユーザ＝開発者もしくは隣接者のとき、ドキュメント、標準化
- 定量化の目的・価値の設定は
- ドキュメントがないときのシステム分析　技術移転
- 品質 VS 納期
- ひな型、部分化と生産性、信頼性

3　討論テーマの設定
- ソフトウェア品質の定量化に関して品質の定義を行っては
- 自分達の作業の中で何が品質に該当仕様の（達成度，納期）するか
- 個々の立場によって異なると思うので
- 『品質の（定量化の）基準とその評価』を考えてみては
- いきなり品質の定量化に行けるのか
- 定量化にとらわれず、品質を考えてみてはどうか

↓

- 『ソフトウェア品質の定義としなにを基準とするか』（討論テーマ）

4　品質をどうとらえるか

1) 品質を決定する要因
- 仕事の継続性（ドキュメント）
- 検査部門の有無（分業）
- ユーザーとの密着度
- ユーザーの業種は
- 納期（開発時間）
- プログラムのパターン化
- 経営者からみた品質

ユーザーからみた品質

↑

我々（開発者）からみた品質

↑

ここにしぼる

- 品質の定義とお客様の満足度に関連があるのでは
  （例えば、客先での作業のため、働きぶりも評価される。）
- 『ユーザーの満足度＝ソフトウェアの品質』と考えてよさそうだ！？
- ソフトウェアを出荷するときの『判断基準』があれば（納期は除く）

↓

5　ソフトウェア品質の定義として何を基準とするのか

1) レビューローテーション
- 各開発者の成果物を第三者がレビューする
- すべての工程においてレビューは必要！
- しかし“ないがしろ”にされている。

2) テストカバレッジ
- カバレッジ・ツールの利用
- テストケースの網羅率を計る。
- 個々のパスの通過等を知ることができる。
- この率が高い程品質が高いといえるのでは？

3) テストデータ
- 生データを使う（例　徳島市民全員　２５万件）
- 現場テスト（オンラインの場合）－おばちゃんねえちゃんテスト
  (利点) 誤操作のテストはできる。
- 客先からデータをもらう。〔テスト結果のチェックも客にまかせる〕
  →理想の姿？
- 実開発者がテストデータを作る。

4) それぞれの問題点

レビュー	時間の制約
	１人ではできない
	ひんぱんな仕様変更に対応できない
	レビューアーへの説明が困難
カバレッジ	本来のテストの意味がうすらぐ
	全てのパスの組合せは測定できない
	タイミングテストができない
	インターフェースの確認ができない
テスト	チェック項目の洗い出しが完全にはできない
	テストデータが完全にそろわない
	運用データ 開発者が作成した試験データにもれがある

↓

6　『ソフトウェアの品質には基準が必要である』（討論結果）

IV　おわりに

終始、愚痴の言い合いや悲惨度チェックになりがちな当グループの討論を最後まで温かく見守り、また行き詰まった時ひとすじの希望の光を投げかけてくださった松村氏及び野村氏、さらに楽しい討論の場を与えて下さったワークショプを運営された方々に心より感謝致します。

第９回夏のプログラミング・ワークショップ　グループ討論報告書

テーマ：まず手始めに実行できそうな定量化手法を提案する
日時：1989年9月5日～8日
場所：盛岡市近郊
グループ名：七福人
参加者：金沢美佐雄（岩手電子計算センター）
小池 誠（SRA）
高田 義広（大阪大学）
鶴林 伸英（富士通九州システムエンジニアリング）
平松 由美子（三菱電機東部コンピュータシステム）
馬島 宗平（NTT）
山田 淳（東芝）　　（五十音順）

ワークショップの全体テーマ“ソフトウェア品質の定量的把握を目指して”に対して，我々のグループでは現状で実行できそうな定量化手法の提案を目指して討論を行った．先ず，定量化に関する現状の問題点について意見を出し合った．次に，仮想的な開発プロジェクトを想定して定量化手法を検討した．検討は，大阪大学から提案されている“定量化のための枠組み”に基づいて，(1)ユーザの立場から，(2)設計者，管理者，等の開発を行う者の立場から，の２つに分けて行った．討論の結果，実行できそうな定量化手法についていくつかの手がかりが得られた．

１．定量化に関する問題点

定量化に関する現状の問題点として次のものが挙げられた．

(1)データ収集の方法が複雑であり，収集に手数がかかる．

(2)用語の定義があいまいであるために，データ収集の結果に個人差が出る．…例えば，バグの定義があいまいであるために，どのようなエラーを１個のバグと数えてよいか分からないことが挙げられた．

(3)収集したデータが定量化したい事柄を反映していない場合がある．…　例とし，レビューにおける発見エラー数が挙げられた．発見エラー数が目標数を越えるま

でレビューを行ったが，発見されたエラーが単純なものばかりだったため，レビューが十分行われなっかたことが報告された．

2．ユーザのための定量化手法

先ず，ユーザのための定量化手法を検討するに当たって，定量化するべきソフトウェアの特性の洗いだしを，仮想プロジェクト（表１）について行った．特性の洗いだしはISOから提案されている分類に従って行った（表２）．次に，洗いだした各特性について定量化手法を検討した．主な検討結果として，(1)誤り許容性に対して被害率（故障した場合の予測される被害の大きさ），データのバックアップの頻度，(2)導入容易性に対してインストール回数，ユーザ数，等が利用できそうなことが分かった．また，洗い出された特性の中で定量化の方法が分からなかったものもいくつか残った．

表１　仮想プロジェクトの概要

種類	地理情報システム
機能	ある地域の，道路，建築物，住人，水道，等の詳細な情報の管理，検索，地図形式の表示，ポインティングデバイスによる編集
構成	データベース管理部，ユーザインタフェース部，等から成る．
ユーザ	市役所職員（機械の扱いを苦手とする．）

表2　仮想プロジェクトにおいて要求される特性

使用性	操作性　理解性
効率性	実行効率性
機能性	正確性　セキュリティ
信頼性	無欠陥性　誤り許容性

また，検討の結果，ソフトウェアの種類に依存せず必ず要求される特性があるらしいことが分かった（表4）．

表3　ソフトウェアの種類に依存せずに要求される特性

特性		説明
機能性	合目的性　セキュリティ	
信頼性	無欠陥性　誤り許容性	故障の影響の度合による．
使用性	理解性　習得容易性　操作性	MMIの有無による．
効率性	実行効率性	まれに，効率が不必要なことがある．
保守性	修正容易性　拡張性	使い捨てのソフトウェアでは，不必要．
移植性	導入容易性	ユーザの数による．

3．設計者，管理者のための定量化手法

設計者，管理者のための定量化手法を次の３つの事柄について検討した．

(1)ユーザ要件の把握の度合…　ユーザから提示される要件は潜在的な要件の一部に過ぎないため，上流工程でのユーザ要件の把握の度合の定量化は難しいことが分かった．ユーザ要件の検討（展開）の度合，ユーザのレベル，等に基づいて定量化できないかと言う意見が出た．

(2)設計の品質…　従来行われている設計書の枚数による定量化は，設計図の内容を無視しているため，ナンセンスであると言う意見が出た．設計書に含まれる図表数，表の項目数，等による定量化手法が提案された．また，より正確に定量化を行うためには作成される設計書を規定する必要が指摘された．さらに，データ収集を機械化すること，そのために設計を計算機上で進められるようにすること，の必要性が指摘された．

(3)レビューの品質…　従来行われている発見エラー数による定量化では，発見されたエラーの種類を無視しているため，うまく定量化できない場合があることが分かった．発見されたエラーを分類し，その比率と個数を併用する定量化手法が提案された．また，エラー分類方法の明確化が必要であることが指摘された．

4．まとめ

現状で実行できそうな定量化手法について次のような手がかりが得られた．

(1)誤り許容性に対して被害率（予想される被害の大きさ），データのバックアップの頻度，等を利用できそうである．

(2)導入容易性に対してインストール回数，ユーザ数，等が利用できそうである．

(3)ソフトウェアの種類に依存せず必ず要求される特性があるらしい．

(4)ユーザ要件の把握の程度に対してユーザ要件の検討（展開）の度合，ユーザのレベル，等に基づいて定量化できそうである．

(5)設計の品質に対して図表数，表の項目数，等による定量化手法が役立ちそうである．

(6)設計の品質の定量化を行うためには作成される設計書を規定する必要がある．

(7)設計の品質の定量化を容易にするためには，データ収集を機械化すること，設計を計算機上で進められるようにすることが必要である．

(8)レビューの品質に対して，発見されたエラーを分類し，その比率と個数を併用する定量化手法が役立ちそうである．

最後に，グループ全体としての感想をつけ加えさせて頂くと，少しは定量化を前進させられそうだと言うことであった．

’８９ＳＥＡ夏のプログラミングワークショップ　グループ討論レポート

＜＜　第４グループ　麻里ちゃんチーム　＞＞

グループメンバ

	［氏名］	［会社名］	［血液型］
(ﾘｰﾀﾞ)	桜井麻里	ＳＲＡ	B
	堀内秀一	日本システム	A
	後藤　浩	シスプラン	O
	山崎哲彰	ソフトウェアコントロール	A
	湊　幸雄	ケーシーエス	A
	根本慶一	ＳＲＡ	O
	小森林正樹	岩手電子計算センター	A
	宮崎　祐	ＮＴＴ	B
(ｵﾌﾞｻﾞｰﾊﾞ)	久保宏志	富士通	B

1．総論

①参加者の現場のプロフィールとポジションの交換
②ソフトウェア品質の定量的把握に関する、具体的な事例の研究
③研究した事例を、参加者の立場で議論する

以上のような流れで、討論を進めた。

結果的には「ソフトウェア品質の定量的把握を目指して」というよりも、「ソフトウェア品質の定量的把握の現場での導入を目指して」とでも題すべき、きわめて泥臭い議論になった。これはわれわれのチームに限ったことではなかったらしく、最終日の各チームの発表でも、現場サイドに密着した（せざるをえなかった）討論結果が見受けられた。

結論としては、

①ソフトウェア品質の定量的把握のためには、モデルを作成し、長期的にデータを取るしかない。
　モデルの妥当性、データの精度は、この際問わない。というのは、現実にデータを収集し、統計データを残そうとしている現場が、あまりにも少ないからである。
②品質把握の精度は、データを蓄積し、それを分析することにより上げていく
③とりあえず収集できるデータとしては、検出されるバグの数がある。
④データの収集が行われていない現場、もしくは組織で、データ収集を始めることは、
　・管理者の抵抗
　・技術者の抵抗
などのため、導入が難しいかもしれない。
　まずは、
　・必要性を認識させること
　・できることから始めること
である。
⑤品質把握だけでなく、品質向上のために、
　・標準化
　・レビュー
　・スキルアップ
を現場で実践していくことが不可欠。
⑥データ収集も含めて、ソフトウェア開発を機械化していくことが、効率化、品質向上のために望まれる。

おおむね、以上のような結果になった。

討論はけしてスムーズには進まなかった。思わぬ飛び越しや、割り込み、無限ループ等が絶えなかった。参加者の立場の違い、業務内容の違いから、共通認識を見つけることが非常に困難で、結論らしい結論は出ないままに終った感がある。しかし、考えようによってはこのまとまりのなさが、現場での対応の立ち遅れを反映しているとも言えるわけで、問題点に対する回答は、各参加者が、それぞれの職場で見つけていくしかないように思う。

では、討議の内容に入ろう。

具体的事例として選ばれたのは山崎（ソフトウェアコントロール＠東京）のポジションペーパである。内容については、ポジションペーパをお読み頂きたい。これは、山崎が現在担当しているプロジェクトにおいて、品質管理システムを適用し、データ収集を行ったときの問題点と対策をまとめたものである。定量的評価の手段となる帳票類や、評価尺度等については、松田（ソフトウェアコントロール＠大阪）のポジションペーパを参考にして頂きたい。

ポジションペーパの要点を整理すると、

前提
・何を測定するか・・・・・・バグ件数
・どうやって測定するか・・・バグ票による自己申告性

問題点
・モデルの精度が悪い
・測定の精度が悪い

対策
・モデルの精度を上げる・・・長期的にデータを蓄積して、統計データを得る
　プロジェクト進行にしたがって、予測値を修正する
・測定の精度を上げる・・・・意識の向上（バグ票の起票を徹底する）
　バグ件数測定の機械化

以上のようになる。

3．討論

討論は、取り上げた事例を各参加者が自分の立場から考え、成果、疑問点、問題点を上げることによって進められた。討論で提出された意見を構造化すると、以下のようになる。

- （１）実際の事例から品質を測るには...
  - その目的
    - ・なぜデータを取るのか
  - その方法
    - ・予測バグ数の設定
    - ・バグ票集計
    - ・検出率算出
  - 導入手段
    - ・どうやってデータを集めるか
- （２）事例から得られる問題点
  - 尺度の問題
    - ・何のデータを取るか
  - モデルの精度・妥当性の問題
    - ・プロジェクト規模
    - ・仕様変更
- （３）他の現場に導入しようとした時の問題点
  - ・技術者・管理者の不理解
  - ・コスト
- （４）定量的把握以外に品質向上のために必要なこと
  - ・レビュー
  - ・標準化
  - ・スキルアップ
- （５）機械化
  - 開発環境の機械化もまた、品質向上をバックアップする
- （６）進捗管理
  - 進捗管理においても定量的把握が重要
- （７）それ以外（根本的問題）
  - ウォータフォールモデルの限界

以下、それぞれのテーマで出された意見を紹介する。

（１）実際の事例から

①品質を測るには...

・データ収集が第一と考えた。しかし、データ収集にはそれなりのコストがかかる。そこで、品質管理部門を設置し、専門部隊による、データの収集と品質管理システムの構築を開始した。

・集めるデータを絞り込んだ。品質を測る尺度（速度、操作性等）をすべて計測しようとせず、バグのみに絞ってデータを集めることから始めた。

・結果的に非常に効果があった。今まで目に見えなかったバグ数の推移が、グラフ・表の形で見られるようになったからである。

②その目的

・データ収集は定量化の第一歩である。まず、データを取る。次に蓄積したデータから、統計的な分析をすることが可能となる。バグ票の収集・分析から、どこが弱いか、どんなバグの傾向があるかが分かるので、品質向上のために具体的なターゲットが設定できる。

・開発者の品質意識を高める効果がある。

・プログラマの質がプログラムの品質を決めてしまうという。しかし、プログラマ品質を把握することは非常に難しい。そこで、バグ票によって個人データを分析することで、プログラマの質を把握する一つの目安が得られるかもしれない。

・プロジェクト終了時に、バグ票の分析から、どの部分がバグが多かったか、どんな傾向のバグが多いかなど、保守のために重要なデータを引き継ぐことが可能。

③その方法

ソフトウェアに含まれる全バグ数の予測
↓
工程内の検出目標バグ数の設定
↓
バグ票の収集 ←
↓
検出率の算出
↓
次工程での検出目標バグ数の設定 ┘

・これは、いわゆる目標管理である。目標を設定することにより、バグを申告することに対する、技術者の抵抗をなくしていくことができる。検出バグ数を上げることが目標となっていくからだ。

・各工程の終わりに、予測バグ数の見直しが必要。

・予測精度を高めるためには、
　長期的にデータを蓄積し、統計データを出す
　開発作業の進行に沿って、軌道修正を行う
ことが有効。

④導入手段

・リーダがメンバにデータを取ることを徹底する。これが現場レベルではもっとも大事である。

・現場の負担を軽減するため、データの蓄積と分析を行う専門部隊を設置する。

・管理者レベルの理解が得られないなら、問題意識のある人が、自分のプロジェクトで、自分自身がデータを取ることから始める。

（２）事例から得られる問題点

①尺度の問題～何のデータを取るのか。

・バグ数だけでは把握できない品質がある。たとえば、

　要求仕様の品質

　レスポンスタイムなど性能面

　ドキュメント品質

・これに対して、対策とはならないが、

　ドキュメント枚数の測定

　という案が出ている。

・すべてのプログラムに応用できる定量的メトリクスを作るのは難しい。しかし、事務系、制御系、言語処理系など分野を絞れば、定量的把握の基準は作れそうである。

②モデルの精度・妥当性の問題

・プロジェクト規模

　１人プロジェクトや、小規模プロジェクトでは、大規模プロジェクトと同じ方法論で品質管理を進めることはできないのではないか。運用方法、目標設定とデータ分析の方法を、別に検討する必要があるかもしれない。

・仕様変更

　特に事務系アプリケーションを担当していると、ソフトウェア開発中に頻繁に仕様変更が入る。仕様変更を品質データに反映させる方法が必要である。

　仕様変更を回避する手段として実施しているのは、ユーザとレビューを行うことと、議事録を残すこと。

・いわゆるメンテナンス作業

　新規開発ではなく、改造、流用のシステムにどれだけのバグが混入するか。現在、ソフトウェア工学を始めとして、ソフトウェア品質管理の研究は、新規開発システムを対象にしたものばかりであり、メンテナンスを対象にした定量化の研究はない。メンテナンスによって、新規部分、流用部分にどれだけバグが混入するかは、データを取ってみなければ分からない。

（３）他の現場に導入しようとしたときの問題点

①技術者・管理者の不理解

・技術者は開発以外の作業が発生することから、管理者はデータ収集作業の分が工数増になることから、データ収集に抵抗を感じがちである。これを説得するには、

　データを収集することの必要性を説明する

　アピールを繰り返す

　根回しをし、理解者を増やしていく

　他社のデータ、実例を示す

以上のような手段が有効である。また管理者に対しては、顧客に働きかけ、顧客から圧力をかけてもらう方法もある。

②コスト

・データ収集にはコストがかかる。このコストを回収するには、データを収集するためのコストを、あらかじめ開発費の中に見込んでおくしかない。顧客に必要経費として認めてもらう努力が必要であろう。

③管理部門

・データの蓄積・分析のための専門部門が必要である。これには組織を動かすしかない。もし、会社として専門部門が作れないようなら、プロジェクト内にデータ管理の担当者を確保しておくことである。

（４）定量的把握以外に品質向上のために必要なこと

①レビュー

・タイミング ―― 各工程の終わり。

・内容 ―― 成果物を対象にしたレビュー。
→ユーザ参加レビューも必須。
レビューは開発担当者とレビューアによる共同作業である。

・問題点 ―― レビューアの負担が大きい。
→工数見積り時にレビュー工数を組み込んでおく。
→レビューの専門家を確保することも手である。

②標準化

・効果的な標準化を行うことによって、最低限の品質が保証できる。また、新人技術者の教育にも効果がある。

・標準化することにより、レビュー時のチェック項目が明確になる。

・ポイントは、
定量的な標準化を行うこと。これにより、品質の定量的把握が可能になる。
繰り返し作業を定型化すること。特にソフトウェアの世界は、大量生産がなく、一回限りの開発という場合が非常に多い。そのため、毎回異なる作業を行うことになり、標準化が困難と思われがちだが、必ず、同じ作業を行っている部分があるはずである。そこを見つけ標準化していくことが必要だ。

③スキルアップ

・プログラムの品質は、プログラマの質・チームの質に左右される。技術者のスキルアップは品質向上の有効な手段と考えるべきである。

・スキルアップのために、新人教育の強化、フォローアップ研修の実施などを行っている。

（５）機械化

・開発環境・試験環境を整備することが品質向上にもつながるはず。環境さえよければここまでできたのに、といった場面は結構見られるからである。

・生産物のヴァージョン管理、不良データなどの管理も、機械化することで効率化できる。

・理想的には、現在あるような設計支援、開発支援ツールに、品質データの収集、分析機能を付加した、高機能ワークベンチのようなものがほしい。

（６）進捗管理

・工程作業の終わりを何によって判断するか。

→工程開始時に設定した目標を達成しているかどうかレビューする。その際、目標設定は、検出バグ数、消化試験項目数など、生産物の品質を定量化できるものにしておくこと。

・ただし、数字はあまり厳密でない方がよい。少々ａｂｏｕｔなくらいが、現実的な管理ができる。

（７）ウォーターフォールモデルの限界

・開発過程をいくつかの工程に区切り、必ず前工程が終了してから次の工程に入るという、いわゆるウォーターフォールモデルは、現在多くの現場で、工程管理、品質管理に使われている。しかし、このモデルの問題点も、最近さかんに議論されている(*1)。設計品質の向上、ユーザーの満足度の向上のためには、プロトタイピングや段階的開発等の手法を取り入れていかないと、現実に対応できなくなる。

(*1)ウォーターフォールモデルは、元来管理の容易性から導入されたモデルである。大場充『ソフトウェアマネジメント』（ＨＢＪ出版局、1988）によれば、ウォーターフォールモデルには、以下のような問題点がある。

・要求定義、設計工程の作業内容の定義があいまいである。このことは、要求定義、設計工程での品質管理を困難にする。

・テストがある程度進むまで、設計の品質が評価できない。上流工程のレビューの対象になるのは、自然言語による仕様書であることがほとんどであるため、精度の高いレビューが期待できない。

4．結論

（１）

ソフトウェア品質の定量的把握のためには、まず、データを収集することである。しかし、開発現場でデータを収集することにはさまざまな障害がある。つまり、定量化は可能か、ということよりも、定量化を計ることが現場で可能か、というきわめて泥臭い問題がある。組織ぐるみ、会社ぐるみで実行しないと、データの蓄積はできない。

現在データ収集を行っていない現場でやるべきことは、

①データを取ること
②データを取ることの重要性をアピールすること
③データ収集・分析のための工数を確保すること（組織を作ってもらうことも一案）

ということになる。

（２）

データの収集を開始したとしても、データの精度、モデルの妥当性という問題は残されたままだ。これには、速効力のある対策はないが、長期的にデータを蓄積することと、プロジェクト規模や仕様変更などを考慮し、ケースに応じたモデルを作っていくことが必要である。

（３）

品質向上のためにはデータを取るだけではだめ。

・レビュー
・標準化
・スキルアップ

という３本柱を徹底すること。

また、品質の定量化を進捗管理の基準にし、各工程で生産物の品質をチェックすることも、開発ソフトの品質を上げていく有効な手段となる。

（４）

品質向上と品質把握の精度向上のために、品質データの管理も含めた作業環境の機械化が望まれる。

１９８９．９．１９　文責：山崎

第七回夏のプログラミングワークショップ(若手の会)

「ソフトウェア品質の定量的把握を目指して」

FFグループ

メンバー紹介:

松本健一　(大阪大学)

白木真二　(立石電機)

三浦　博　(岩手電子計算センター)

吉田　徹　(岩手電子計算センター)

高橋良一　(日本電子計算)

岡田正之　(SRA)

山口敬介　(ケーシーエス)

「ソフトウェア品質定量化のための枠組み」の特徴に１つは，ユーザから開発者に渡されるものが品質に対する要求だけでなく，その要求品質の実現に有効な開発技術と評価尺度の集合（開発計画）を含んでいる点である．

われわれフィードフォワード（ＦＦ）グループは，枠組みの持つこうした考え方がユーザと開発者の間だけでなく，ソフトウェア開発の各工程間にも当てはめることができると考えた．即ち，ソフトウェア開発の上流工程から下流工程に対しては，品質実現のために必要な要求を出すだけではなく，品質を効率よく実現するために有益な情報も合わせて渡してやる（フィードフォワードする）ことが重要であると考えたわけである．

そこでわれわれは，ソフトウェア開発の経験に基づいて，ソフトウェア開発の各工程間でどのような情報の受渡しが行なわれ，それらがそれぞれどのような意味を持つのか議論を行なった．議論においては，ソフトウェア開発を"仕様作成"，"プログラムデバッグ"，"テスト"の３つの工程に大別し，各工程を別々の会社（Ａ社，Ｂ社，Ｃ社）が担当する図１のようなモデルで考えた．特にここでは仕様作成を行なうＡ社からプログラムデバッグを行なうＢ社に対してフィードフォワードされる（されるべき）情報（フィードフォワード情報）について議論を行なった．

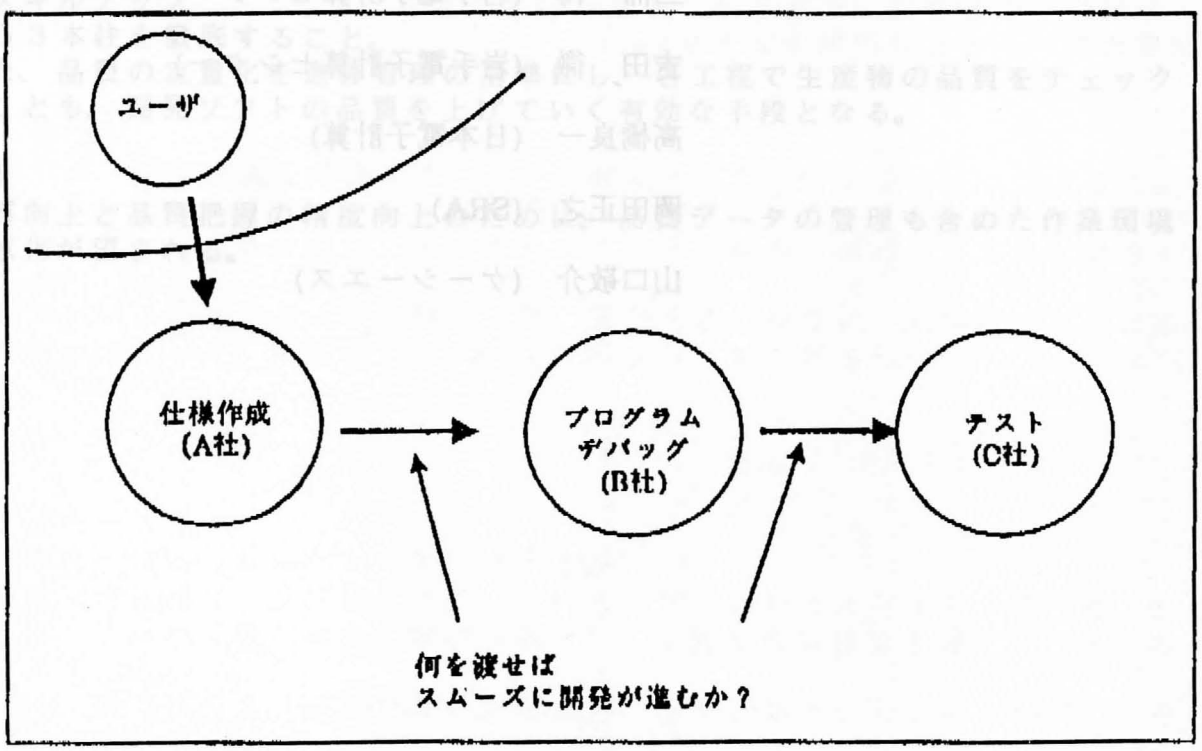


図１　ソフトウェア開発過程

フィードフォワード情報の洗い出しはＫＪ法を用いて行なった．洗い出されたキーワードを表１にまとめる．更に，表１のキーワードをグルーピングし，各グループに名前を付けたものを表２に，グループ間の関係を図２に示す．以下に，表２，及び，図２中の各グループについて簡単に述べる．

・開発制約…ソフトウェアを開発する上での物理的な制約．開発過程や開発環境に対する要求ともいえる．
・開発目的…ソフトウェア開発の目的．実現すべきソフトウェア品質を決定する基本となる．
・製品仕様…開発目的に基づいて決定されたソフトウェアの性能，及び，機能．開発目的を性能や機能として具体化したものといえる．
・仕様全般…製品使用に基づいて決定されたソフトウェア中の制御，及び，データの流れ，製品仕様を詳細化したものといえる．
・プログラム作成資料…実際にプログラムを作成する上で必要なドキュメント，ツール，及び，データ．プログラム作成を行なう上で有益な過去の経験や知識を含む．
・評価尺度…ソフトウェアとその開発過程を評価するための尺度．

ここで掲げたキーワードやグループがフィードフォワード情報として妥当なものであるかどうかの十分な議論は出来なかった．従って，あくまでも現時点での結論であるが，フィードフォワード情報は，"要求"，"評価"，"支援"の３つに分類される（表３参照）．またそれぞれはソフトウェア（プロダクト）とその開発過程（プロセス）に関するものに大別できる．

"要求"とはフィードフォワード情報の基本であり，現実のソフトウェア開発過程におけるフィードフォワード情報の多くはこれに当たると思われる．

"評価"は要求が実現されているか（されつつあるか）を確認するために必要である．「ソフトウェア品質定量化のための枠組み」においてもその必要性を指摘しているが，現実のソフトウェア開発においては明確にされない場合が多く，ましてやフィードフォワード情報とされている場合はほとんどない．

"要求"と"評価"が上流工程から下流工程に対するいわゆるムチであるとすれば，"支援"は下流工程に対するアメである．現実のソフトウェア開発においていくつかその例がみられるが，"支援"に対応する"要求"あるいは"評価"が明確になっていない場合が多いと思われる．

以上フィードフォワード（ＦＦ）グループにおいて行なわれた議論の内容について簡単にまとめた．最初に述べたように，ソフトウェア開発におけるフィードフォワードの考え方は，既に「ソフトウェア品質定量化のための枠組み」において取り入れられている．しかし，その詳細についてはまだ明確にされているとはいえず，表３に示したようなフィードフォワード情報の分類は，今後検討を重ねていくべき重要な課題の１つであると考える．

表1　ＫＪ法により洗い出されたキーワード

●納期･スケジュール

●大まかなプログラム容量

●各機能(モジュール)間の関係
(データ･制御の流れ)

●モジュール構造図

●開発方針(C, 85%以上とか)

●データ･フロー･ダイアグラム

●機能構造図

●テストデータ

●デバッグ仕様書

●プログラムID

●処理フロー

●ライブラリー

●サンプル
(似たようなテクニックを
用いたシート)

●各機能のI/F仕様

●ファイルの種類
(例えばISAM,VSAM)

●フォーマット

●ファイル説明(入出力)

●外部テーブル仕様

●ファイル仕様

●操作手順

●処理概要

●各機能及び各品質を実現するため
に必要な(もしくは使用してもらい
たい)開発技術･ツール

●処理内容の定版化(それ･あれ
という言葉をもちいない
　ブレークキー
　ソートキー
　マッチングキー)

●何のためのシステムか?
(何につかうのか)

●どういう人がどういう状態で?
(例;ネットワークか?)

●データフロー
(処理概要の表現手段として)

●プログラムの目的
(何をしたいのか)

●取扱い説明書(ユーザの観点から)

●入力･出力

●納期

●各機能および品質の実現に必要な
コスト(開発規模･日数)

●性能(処理速度とか)･機能

●ハードウェアの仕様

●システムの運用方法

●ユーザが望んでいるものは何か
(レスポンス、機能性、信頼性)

●用語説明

●各機能及び品質が計画通り実現さ
れたかどうかチェックするための
評価尺度･基準

表2　キーワードのグルーピング

・開発制約	納期
	コスト
	開発規模
・開発目的	開発方法
	目的
・製品仕様	性能
	機能
	ハードウェア仕様
	運用方法
	用語説明
・仕様全般	処理概要
	取扱説明書
・プログラム作成資料	入出力
	ファイル仕様
	フォーマット
	テーブル仕様
	プログラムＩＤ
	機能構造図
	モジュール構造図
	各機能のI/F仕様
	データフローダイアグラム
	サンプル
	ライブラリ
	開発技術
	ツール
・評価尺度	ソフトウェア品質の評価尺度
	開発過程の評価尺度

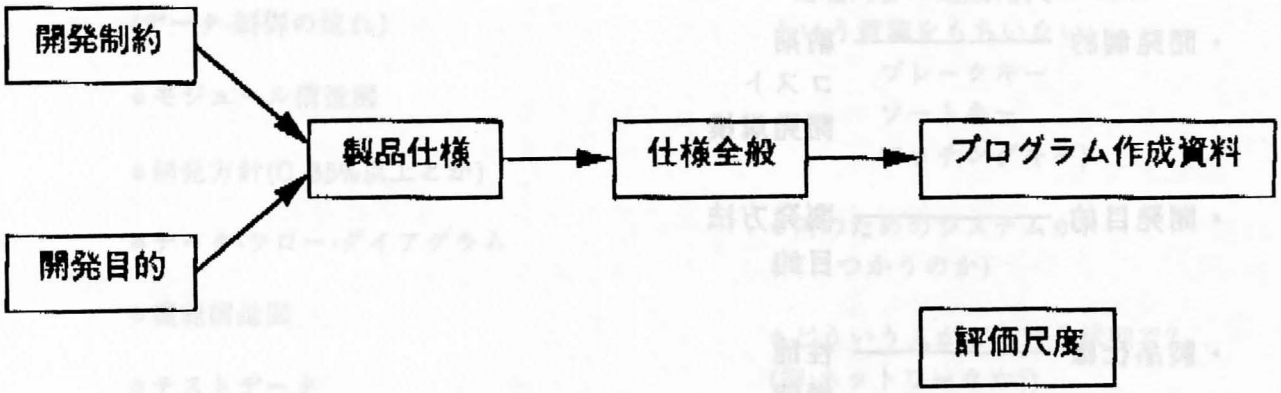


図2　グループ間の関係

表3　フィードフォワード情報

	プロダクト	プロセス
要求	開発目的 製品仕様 仕様全般	開発制約
評価	評価尺度 （プロダクトメトリック）（プロセスメトリック）	
支援	プログラム作成資料 （ツール）　（サンプル、ライブラリ）	

ワークショップ　サマリー　レポート

作成：カシオ計算機（株）
山城　英彦

1．テーマ

第５班のグループ討論は、以下の３項目にテーマを絞って討論を行なった。

1）Ｍｅｔｒｉｃｓの決定方法の検討。

2）品質評価基準の決定方法の検討。

3）開発作業へのフィードバック方法の検討。

2．参加者

1）オブザーバー：天池学　（カシオ計算機）
2）リーダー　：　中川明生（富士通）
3）書記　　：　山城英彦（カシオ計算機）
4）メンバー　：　松田正人（ソフトウエアコントロール）
根岸学　（日立製作所）
三宅武司（日本電信電話）
吉山徳博（富士通九州システムエンジニアリング）
荒井雅一（日立ソフトウエアエンジニアリング）

3．討論結果

1）概要

（1）Ｍｅｔｒｉｃｓの構成は、「設計部門と検査部門」及び、「検査部門とユーザ」の間で、各々設定する必要がある。

（2）作業環境にあったシステムを構築する為には、Ｍｅｔｒｉｃｓ及び評価基準を障害報告の分析結果を基にして追加／変更していくことが重要である。

（3）開発作業へのフィードバックは、障害報告の分析結果を基にして、実施効果のある開発作業に絞り込んで適用することが望ましい。

2）Ｍｅｔｒｉｃｓの構成

Ｍｅｔｒｉｃｓに関しては、

（1）Ｐｒｏｄｕｃｔ　Ｍｅｔｒｉｃ
ユーザにとっては、製品の品質を開発者に要求するための言葉である。
検査部門にとっては、ユーザからのＰｒｏｄｕｃｔ　Ｍｅｔｒｉｃのフィードバックにより、製品検査をどのような観点で行うべきかの指針を得ることが可能となるため、製品検査の質の向上を図ることができる。

（2）Ｐｒｏｃｅｓｓ　Ｍｅｔｒｉｃ
検査部門から設計部門に対しては、Ｐｒｏｃｅｓｓ　Ｍｅｔｒｉｃの形で、開発技術の向上を目的としたフィードバックが行える。
フィードバックの積み重ねにより、設計作業標準／検査作業標準として現場に適用が可能となる。

と言える。

この前提に基づき、ワークショップの討論会の結果から、開発者側のＭｅｔｒｉｃｓは、図１に示す構成をとることが現場への適用を考えた場合必要となると思われる。

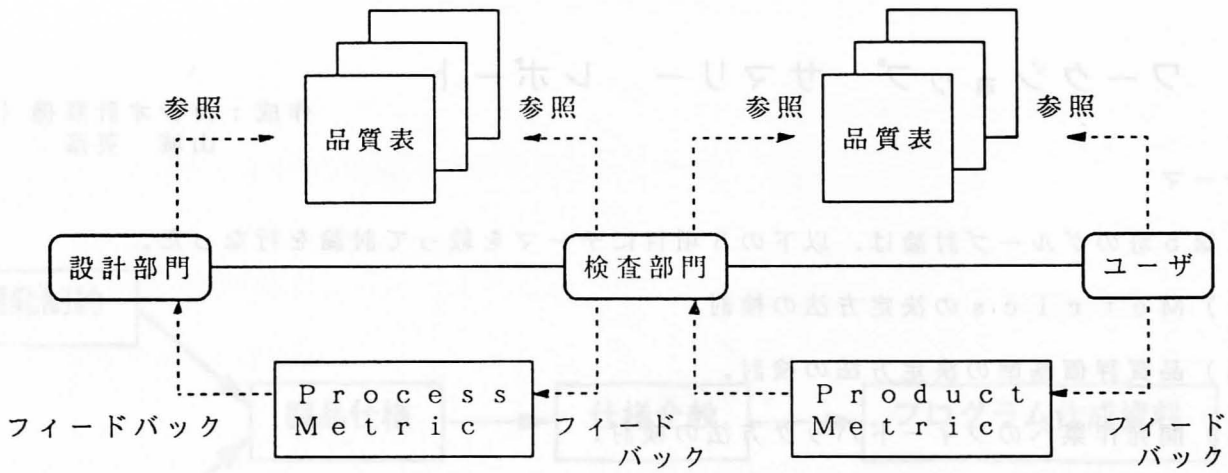


図１　Ｍｅｔｒｉｃｓの構成

３）開発作業への適用

開発作業への適用に際しては、図２の流れに基づいて、実施部門で適用方法を検討する必要がある。

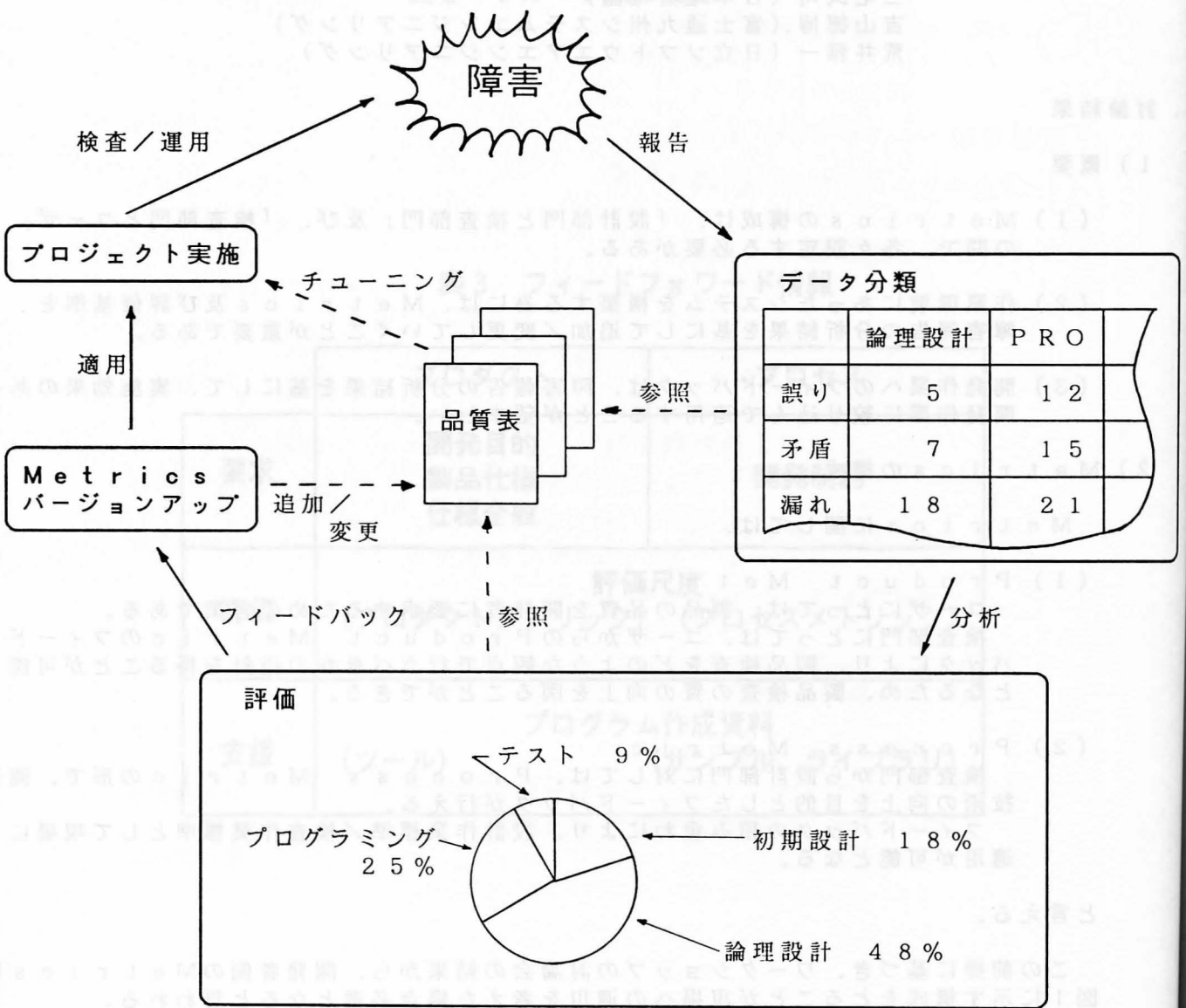


図２　開発の流れ

ポイント

1．プロジェクトに適用するために、品質表の各項目の細分化と、適用項目の絞り込む作業が必要となる。

2．重要な部分を計測するＭｅｔｒｉｃｓに対しては、より詳細なものにして、精度の高い計測を可能とすることが重要である。

3．障害データを体系的に分類し、バグの作り込み原因を明確化する必要がある。さらに、現場での適用を繰り返し行い、分類項目の見直し及び、品質表のブラシュアップを図ることが重要である。

4．おわりに

本レポートに、どれほどワークショップの成果を盛り込めたかは、討論に参加した方々の判断に委ねたいと思います。

しかし、久保委員長も言われているように、私としては、ワークショップを通じて皆さんと知り合えたことが最も大きな成果であると思っています。

今後とも、このつながりを大切にしていきたいと思っています。

なお、最後になりましたが、本レポート執筆に当たっては、ＮＴＴソフトウエア研究所の三宅武司氏にアドバイスを頂きました。

ここに、感謝の意を表します、ありがとうございました。

感想とポジション・ステートメント（プログラム委員）

野村行憲（岩手電子計算センター）
「ソフトウェア品質の定量的把握を目指して」.... の雑感

桜井麻里（SRA）
CASEツールをソフトウェア品質の定量的把握に役立てるために

松本健一（大阪大学基礎工学部情報工学科）
Position Paper改訂版

山田　淳（東芝）
ソフトウェア品質の定量的把握を目指してに参加して

第７回夏のプログラミング・ワークショップ
「ソフトウエア品質の定量的把握を目指して」
....の雑感

岩手電子計算センター
野村行憲

はじめに

盛岡での開催であり、プログラム委員の中で唯一の地元の人間ということで実行委員長を命ぜられ、どうなることかと不安であったが、プログラム委員長の久保さんが、ほとんどのことを処理していただいたので、あまり苦労もなくワークショップを運営することが出来ました。ここにあらためて感謝いたします。
私のやったことといえば、会場の手配や、情報交換パーティの準備、機器の手配、経費の精算など裏方の雑多なことです。実行委員長とは本来このようなものらしく、私のような実力の無いものでも十分に努まるもののようです。

1日目

松本（大阪大学）さんの基調講演については資料があるので省略して、夜の情報交換パーティの様子を紹介します。
18時３０分から21時までの2時間３０分のほとんどが、参加メンバーの自己紹介だけに費やされました。にもかかわらず、参加者が退屈しなかったのは参加者夫々が大変ユニークな方達で、紹介が楽しいものであったからだと思います。その証拠に、ホテルの担当者が翌日私を見つけて言うには、「最近のパーティはカラオケばかりでこれといった企画もなくマンネリ化していたが、昨日のパーティはトークだけであんなに盛り上がっていた。久しぶりに楽しい本当のパーティを見せていただきました。今日の朝礼で皆に参考にするよう話した。」とのことであった。

2日目

大場（日本IBM）さんの講演についても省略させていただいて、その後のグループ討論の様子を。
プログラム委員が5つに分れ、おのおのがグループをサポートするという、お馴染みのスタイルでグループ討論がスタートしました。私は松村（SRA）さんとペアで、第一グループ（別名ビジネスグループ）を担当することになりました。とはいっても前述の雑務に追われ、あまり討論には参加できませんでした。
この日の締めくくりとしてグループ討論の中間報告を行ない、スポーツのような夕食「わんこ蕎麦」へと繰り出しました。後半の給仕のおばさん（失礼！）と、挑戦者のかけひきはいつもながら会場を盛り上げました。

3日目
最も議論が集中する重要なこの日から、私もやっと議論に参加することが出来ました。小岩井農場で議論したためか、議論が迷える子羊のごとく揺れ動き、後半になって何とか方向性を見いだして帰路についた。会議場に戻り、夕食を後回しにして、明日の結果発表に向けて最後の追い込みにかかったが、タイムオーバーになってしまった。やむなく夕食を取り、ホテルの一室に集合して、やっとのことでOHPを書き上げたのが午前1時30分頃であった。

最終日
いよいよこのワークショップの成果を発表する日。憔悴しきった顔、サッパリした顔、アルコールがだいぶ残っている顔を前に、予想されたように我が第一グループからの発表であった。発表と質疑応答が繰り返され、予定時刻の12時に全てのプログラムが終了した。

感想
プログラムを造っている立場の人間が、最も重要な瞬間である「完成」を判断する基準が、以外にあいまいであることが 再確認された。
また、いわゆるバグ取りに対してはかなり神経を使っているにもかかわらず、それ以外の品質については以外に無頓着で、エンドユーザーが満足すればそれでOKというところが、受託開発には多いようである。
ともあれ、普段の仕事の中で何気なく見過ごしていた、ソフトウエアの品質を、あらためて考え直す良い機会であったと思う。また、このことをもっと積極的に研究し、実践している企業も有って、見習うべきものも有った。

おわりに
主催者側が勉強させられるような情無い状態であったにもかかわらず、一応の成功を納められたのは、久保委員長を始めとするプログラム委員の皆様と遠路はるばる参加していただいた参加者の御協力が有ったからに他ならず、ここにあらためて感謝いたします。また、主催者として何かと行き届かない点が多々有ったと思いますが、お許しいただけますようお願いいたします。
今回参加された皆さんに、このワークショップで得たことが、今後の仕事の上で役立つことのあることを願っています。

SEA主催 『'89 夏のプログラミングワークショップ in 盛岡』参加報告

==========
CASEツールをソフトウェア品質の定量的把握に役立てるために
==========

1989年10月14日
(株)SRA, ワークステーションＳＩ技術部
桜井麻里
mari@sra.co.jp

私は、9月上旬に盛岡で開催されたSEA主催の夏のプログラミングワークショップと、その報告を兼ねて9月26日に東京の機械振興会館で開かれたフォーラムに参加した。本稿は、私の参加報告であり、参加前と参加後の私の意識の変化を述べたものである。

私たちのグループでは、実際にデータ収集・解析をしはじめたという、(株)ソフトウェアコントロールの品質管理部門の山崎氏の発表を聞いて、それを中心に討論した。発表と討論の詳細は、山崎氏の報告に譲ることとしたい。

山崎氏のところでは、バグ数の目標設定から、バグ票の記入・収集・解析までを手作業で実施しているそうである。それを聞いて、UNIXを日常の作業環境として使用している私がすぐに考えたのが、「それだけやることがはっきりしているなら、比較的簡単にツールが作れそうなものである。」ということであった。しかしながら、山崎氏のところは、UNIX環境には馴染んでいないようであるし、仕様書類はすべて手書きという体制が確立しているようである。そしてツールには

- ツールは使いたい人が使いたい時にそろえるものである
- 普段の環境の中で自然と使えるものでない限り、ツールは忘れ去られる運命にある

という鉄則もあるので、今すぐに作り出すのは控えているのであるが、いつかは山崎氏をはじめデータ収集に励む人達のために、ツールの枠組くらいは提供したいと強く思うようになったのである。

ワークショップから帰ってきて、Tom DeMarco の 『CONTROLLING SOFTWARE PROJECTS』(邦題: 『品質と生産性を重視した ソフトウェア開発プロジェクト技法 － 見積り・設計・テストの効果的な構造化 －』渡辺純一訳)を読んだ。読んでみて、また、新たな発見をした。私の現在の仕事である、「構造化分析技法をベースにしたCASEツール」と、「ソフトウェア品質の定量的把握」がつながったのである。いまさらという感がなきにしもあらずであるが、これは、私にとっては、有意義な発見であった。

構造化分析をベースとしている現在のCASEツールは、システムの要求仕様を分析し、いわゆる上流工程の要求仕様書を作成するには役に立つ。しかし、せっかく作成した要求モデルを仕様書作成と関係者間のコミュニケーションの道具だけにしか使わないというのは何かもったいない気がする。もちろん、それだけでも、何もしないよりはずっといいのは明らかだと思うが、やはり、品質や生産性の定量化ができるようなプロジェクト管理を支援するところまでつないでみることの必要性を強く意識するようになった。そうすれば、CASEツールを実際の開発現場への普及を促進する可能性を高めることもできるであろう。

いくら、品質管理のために、データ収集が必要だからといっても、それだけのために、いわゆる力仕事をするのは、開発作業をしている人にとっては厄介であることは論をまたない。ワークショップの討論の中でも、現場でのデータ収集の困難さは、再三報告された。データ収集の困難さの要因はいろいろあるが、やはり、データ収集にコストと時間と手間がかかり過ぎるというおそれが、管理者ばかりでなく、開発作業をしている技術者の間でも蔓延しているというのが一番の問題のようであるということが、参加者の経験談からはっきりした。データ収集を効率的に行ない、データ収集の困難さを軽減するためにも、やはり、仕様書作成からプログラム作成・デバッグ・テスト・保守までの一連の作業の中で、自動的にデータが収集できるような仕掛けを作らなければならないであろう。データ収集が自動的にできれば、データの解析を機械的にできるようなしかけを作ることも、難しくないはずである。

そういった意味では、現在のCASEツールではまだまだ不足している部分が多い。しかし、現在のCASEツールでも、マシン可読な状態で、システムの要求モデルと設計モデルを一元管理しておき、必要に応じて、必要なデータを計測してみるというぐらいは、何とか実現できそうである。やってみれば、そこから、また新たな展望が開けるのではないかと考えている。幸い、私が扱っているツールは、新しい機能を加えることは比較的容易である。すでに、DeMarco が提唱している DFD や データ辞書が扱えるようになっている。見積りや品質の定量化に関する手法は、上記の本に記述されている。あとは、実現すればいいだけのような気がする。少なくとも、DFD やデータ構造の定義から、機械的に計算できることがらは、すぐにでも対応できそうである。近々、時間をつくって、実施してみたいと考えている。

私自身、ソフトウェア品質の定量的把握ということに関しては、組織的に管理されたことがなかったので、今まで、品質管理とかデータ収集とか作業の標準化といったことを、それほど重要視していなかったのであるが、このワークショップとフォーラムに参加して、私の意識も、大いに変化せざるを得なかった。ソフトウェア品質の定量的把握には非常に重要な意味があるということが良く理解できたように思われる。
また、ソフトウェア品質を定量的に把握するために、大学などの研究機関ばかりでなく、企業の開発の現場でも努力しており、成果も上がっているということがわかった。これは、私にとって、一つの発見である。さらに、ワークショップとフォーラムに参加しているうちに、私も、データ収集に関して、少しは努力してみようと考えるに至った。これも、参加した成果であろう。

結論として、私はワークショップに参加して、
「CASEツールは、ソフトウェア品質の定量的把握にも役立つに違いない」
と確信するに至ったのである。

以上

第7回夏のプログラミング・ワークショップ

Position Paper改訂版

松本健一（２７歳　ＡＢ型）
大阪大学基礎工学部情報工学科
〒560　大阪府豊中市待兼山町 1-1
06-844-1151（ex.4818)

今回のワークショップに参加した目的の１つは，企業の開発現場において品質がどの様に捉えられ実現されていくものなのか，その実態と問題点を明確にし，フレームワークに反映させることにありました．ではまず，ワークショップでの議論に基づいて，企業の開発現場の実態について簡単にまとめてみます．

1.ソフトウェアの品質とは信頼性のことである．

ソフトウェア品質の中で最も重要視されている特性は信頼性です．これは，ソフトウェアの品質の中で信頼性が最もわかりやすく，その評価方法，基準がいくつか提案されているためだと思われます．また，信頼性はユーザに対して最低限保証すべきものであると考えられているからかもしれません．

2.信頼性実現の決め手はレビューとテストである．

ソフトウェアの信頼性は，多くの時間と労力をレビューとテストに投入することにより実現されると考えられています．作り込むバグを如何に少なくするか，というよりも作り込んだバグを如何に多く発見し取り除くか，に重点が置かれていると言えます．

3.レビューとテストの良さは作成されるドキュメントで決まる

レビューやテストにおいては，チェックリストやバグレポート等，様々なドキュメントが作成されています．ドキュメント作成は，レビューやテストの目的（目標）を明確にし，レビューやテストの質を一定レベル以上に保つための工夫と言えます．作成されるドキュメントの量は多くなる傾向にあるため，標準化，計算機化が進められています．

以上は私の個人的な意見であり，間違った部分や見逃している点があるかもしれません．しかし，

ソフトウェア品質→信頼性の実現→レビューとテストの実施→ドキュメントの作成

という図式は，ソフトウェア品質の実現について考えれば誰でもが到達する比較的単純な答えの１つだと思います．

さて，こうした図式に従ってソフトウェアの開発が行われているにも関わらず，多く

の場合，品質の実現が困難であることもまた事実のようです．その理由の一つは，作成されたドキュメントの中身や評価に関して十分検討が行われていないためだと思います．

例えば，多くの場合，ソフトウェア開発の各工程で作成されるドキュメントの間のつながりについて明確にされていません．ソフトウェア開発のある工程のドキュメントは，その前の工程で作成されたドキュメントに必要な情報が新たに付加されたものと考えることが出来ます．当然，ドキュメント間で情報の表現方法が大きく異なることはあります．しかし，ドキュメントのある部分は，前工程で作成されたドキュメントの全体，もしくは，一部と対応関係が必ずあるはずです．この対応関係が明確にされていない，あるいは，対応関係を無視してドキュメントが作成されているとしたら，各工程で実現された品質は次の工程に引き継がれず，最終的なソフトウェアにおける品質の実現は困難になると考えられます．従って，ドキュメント間の対応関係を明確にし，対応関係に基づいたドキュメントの定量的評価を行い，工程間で確実に品質が引き継がれて行くようにすることが，ソフトウェア品質を効率よく実現するためには必要と考えます．

現在提案されているフレームワークでも，ソフトウェア開発過程において作成されるドキュメントはどうあるべきか，また，どう評価されるべきか，といったことは明確にされていません．ここで述べたドキュメント間の対応関係の明確化と，対応関係に基づくドキュメントの定量的評価の必要性は，今後フレームワークをより現実的なものとして行く上で，検討すべき重要な課題と考えられます．

13th Oct. 1989

7th PROGRAMMING WORKSHOP
ソフトウェア品質の定量的把握を目指して
に参加して

山田　淳

東芝　システム・ソフトウェア技術研究所

1．はじめに

今回のプログラミング・ワークショップは、ソフトウェアの品質に関する議論が活発で、やや長い目かと思われた４日間という期間も、ふたをあければあっという間の短かさであった。あれだけ集中して討議するという機会は、なかなかないのではないだろうか。

せっかくの議論をそのままにしておく手はないので、これを筆者なりにまとめて振り返ってみたい。

2．品質定量化の枠組みの効用

品質定量化の枠組みを討論の材料にすることにより、議論の焦点を集中させることができたと感じている。

筆者のグループの討議では品質定量化の枠組みをもとに、地理情報システムを例にして、ユーザの観点と設計者の観点の双方から分析してみた。

その結果、ユーザの観点からは、地理情報システムの例を通して、アプリケーション分野にかかわらずに重要な品質特性項目を検討することができた。

また、枠組みの中で提案されている品質特性の分類やメトリクスをもとにユーザのための（観点の）メトリクスを考え、設計者の観点からは、皆が日常抱いている定量化や品質の問題を検討したりすることもできた。

このように枠組みを活用して、実際のソフトウェア開発の初期段階（要求定義段階）で検討会を行う際に、品質特性の枠組みを材料に討論することで、ユーザの観点と開発設計者の観点の両方から分析することができると思われる。

3. メトリクスについて

やはりメトリクスを設定するのは容易ではないが、それでも次に示すようないくつかの要点はありそうである。

(1)簡明であること、ある程度納得できる理由づけがあること。

(2)ユーザ観点のメトリクスを考えること。使用性や信頼性などのメトリクスを考える際に特に有効と思われる。
これは、ユーザの要求をはっきりさせる助けになるばかりでなく、メーカ側では設計目標にしたり、出荷試験での確認項目に使えると考えられる。

(3)設計のためのメトリクスを考えること。これは、設計や開発作業の問題点を再考する機会にもなり、あるいは開発過程で実際に計測できそうなメトリクスを設定することをとおして、開発中の品質確認の計画を立てることにも役立つと思われる。

こうした点を配慮しながら、まず手始めに使えそうなメトリクスを考え出すことを基本姿勢として、メトリクスの設定を行うのが現実的なようである。

4. 定量化の問題点とは

最後にソフトウェア品質の定量化を阻害する問題点を整理してみたい。
しかし、最終日に行われた各グループ毎の討論結果の発表を振り返ると、定量化の問題に限らず、それを含んだ品質向上を阻害する問題点の提起もかなりあったように思う。

そこでソフトウェア品質定量化および品質向上の問題点と解決策という観点でまとめてみたものが以下である。（必ずしも解決策が見いだされたわけではないが。）

(1)時間がないという問題点
納期優先になりがち。特に仕様変更の多いもの（ＯＡなど）。
機械化、ツール化

(2)担当者の確保が困難。
検査担当を別に設ける場合には、検査担当が最初の設計段階から参加していないと、効果がでない。

(3)テストの問題
テストカバレッジを使用するのが一般的。
テストデータの作成が困難。
もれのないテスト項目の作成は困難。

(4)心理的問題

バグ原因を設計担当者に求めてしまうとモラル低下。

「バグを憎んで人を憎まず」

バグ発見目標を設定すると、目標を達成しても

再テストに至る上限値以上は発見しない。

設計者自身がテストするのでなくテスト担当者が必要。

しかし単体テストは設計者がやる以外ないので、動機づけ必要。

(5)効果的レビューの問題

レビュー目標として

ソースコード１ｋｓ当りのレビュー時間（　１０Ｈ／ＫＳくらい）

レビュー当りの指摘件数

を設定しているが、目標達成してもなお不具合の多い場合がある。

設計ロジックや仕様の内容を調べるメトリクスが欲しいところ。

制御構造の複雑さを示すＭｃＣａｂｅのＣｙｃｌｏｍａｔｉｃ Ｎｕｍｂｅｒ

などで、集中レビューすべき範囲を求めるなど。

(6)不具合データ収集の問題

発見バグのカウント方法が個人で異なるので信頼できる程度に限界あり。

簡単明瞭な定義

使い方に応じた定義

例　現象でカウント　→　テスト時のバグ曲線作成

ファイル修正数、量でカウント　→　設計時の不具合発生多発部分の抽出

レビュー時の発見不具合の分類があいまい。個人で異なるので信頼できる程度に限界あり。

簡単明瞭な分類定義。

少々妙な分類でもころころ変えずに、２，３年集めて分類する方がよい。

(7)環境の問題

より客観的かつ手間取らない測定のためには機械化する必要あり。

環境がまだ整っていない。

(8)メトリクス設定の問題

定量化するための計測尺度であるメトリクスの設定が容易ではない。

計測し易く、かつ理由あるいはモデルに納得できるものが必要。

(9)ユーザの観点を見落としがち

どんな品質特性を確保すべきかを決定する、メトリクスを設定する、

さらにメトリクスで計測したときの目標値を設定するためには
ユーザの使い方、運用のされ方についての十分な分析が必要。

(10)発注ソフトの問題
発注業務での効果的な開発には、機能仕様に加えて評価方法も発注先へ示す。

(11)動機づけと活用の問題
定量化したものは、設計基準値の形にして、
設計へのフィードバックを行っていく。

5. おわりに

参加者の問題意識が高く、非常に興味深い討論ができてよい気分です。
また、おもしろい発想もよく飛び出したのもよかった。
これは討論場所が自由裁量だったことによる環境の効果が出たものと思います。
問題意識を持ったメンバーが、場所を転々としながら議論しあう、その熱いこと。
すこし秋の涼しい風が吹き始めた時期でしたが、まさに盛岡の熱き夏のワークショップといったところだったでしょうか。

E N D

感想とポジション・ステートメント（一般参加者）

岩井田浩章（電力計算センター）
白木　真二（立石電機）
白井　一好（三菱電機東部コンピュータシステム）
松田　正人（ソフトウェアコントロール）
根岸　　学（日立製作所）
平松由美子（三菱電機東部コンピュータシステム）
三宅　武司（日本電信電話）
吉山　徳博（富士通九州システムエンジニアリング）
鶴林　伸英（富士通九州システムエンジニアリング）
堀内　秀一（日本システム）
高田　義広（大阪大学）
前沢　伸一（日本システム）
後藤　　浩（シスプラン）
安喜　裕師（富士通徳島システムエンジニアリング）
山崎　哲彰（ソフトウェアコントロール）
山口　敬介（ケーシーエス）
湊　　幸雄（ケーシーエス）
伊佐　厚司（ケーシーエス）
根本　慶一（ＳＲＡ）
小池　　誠（ＳＲＡ）
岡田　正之（ＳＲＡ）
三田村英生（ＳＲＡ）
山城　英彦（ＳＲＡ）
門岡　春雄（ヒラタ・ソフトウェア・テクノロジー）
小森林正樹（岩手電子計算センター）
三浦　　博（岩手電子計算センター）
金沢美佐雄（岩手電子計算センター）
馬島　宗平（日本電信電話）
中川　明生（富士通）
高橋　良一（日本電子計算）
岩井　邦明（日本電子計算）
宮崎　　祐（日本電信電話）
荒井　雅一（日立ソフトウェアエンジニアリング）
河村　謙介（ＮＴＴ九州技術開発センタ）
吉田　　徹（岩手電子計算センター）

第７回夏のプログラミングワーク・ショップ

氏　名：岩井田　浩章　　ふりがな：いわいだ　ひろあき
年　齢：２７歳　　性　別：男　　血液型：Ａ型
会社名：株式会社　電力計算センター
部　門：大手町事業部
住　所：〒１００　東京都千代田区大手町１－６－１
ＴＥＬ：０３－２１５－６４８１（直通）
ＦＡＸ：０３－２８７－２８８０

ポジション・ステートメント

主に事務処理の機械化業務として、オフコンにおいて設計・開発されたシステムの安定した運用及びその改良に日夜励んでいる。当システムは経理・購買や図書管理業務を中心とした構成になっていて、それらの設計や初期開発は他社の手によって行われた。

そこで、システムに対して客観的な立場である運用担当者としての数年間の作業経験より、ソフトウェアの品質や管理に関する主な問題点は次の３つが考えられる。

- トラブルに対応しやすいように、機能単位に分割して作成されているか。
- 運用システムを修正・改良する場合、その開発環境の作成が容易に行えるか。
- あるサブシステムを改良する場合、システム全体に及ぼす影響を把握できるか。

現時点においてこれらの諸問題は解決されていない。しかし、設計や開発に参加していない上に、仕様書やプログラムのソース等が運用環境と一致しないような最悪な状況の中で、以下に示すような対応を行っている。

- プログラムや画面定義体等のソースの管理を明確化し、それ自体をドキュメントとして扱う。
- 各種ドキュメントをテキストデータとして扱い、オフコンだけではなくパソコンから大型計算機までを駆使してその解析を行う。
- 解析するために各種の技法（ＭＳ－ＤＯＳのツールコマンドの使用等）を試み、ソースプログラムの書き方等を見易いものから解析し易いものにする。

これらは確立されたものではなく改善策に近いものでありかつ開発サイドの立場からの考えであるため、システムの全体的な品質を把握するには不十分に思える。

そこでシステムにおいて『ユーザ仕様の達成度』＝『品質』と定義した場合、それを向上させるため目安として次に示す項目の定量化をはかり総合的に品質を判断したい。

- システムの仕様書をもとにしたレビューの充実度。
- 上記レビュー後の仕様と完成品（納品物）の状態における仕様との違い。
- テスト段階で発見されたバグのシステムに与える影響度と個数。
- テスト時に使用する実運用データの網羅度。

ここで述べたものは暗中模索中のものばかりではあるが、これらを含めソフトウェア品質の定量的な把握や管理の技法の確立をめざしたい。

岩井田　浩章　　株式会社　電力計算センター

感想文（第７回夏のプログラミングワークショップ）

このようなワークショップに初めての参加ですが、連日の討論における色々な方との意見交換や情報収拾により、充実した時間を過ことができました。といっても終わってみればそう思えるわけで、当初『ソフトウェア品質の定量的把握』に関して自分の環境をだぶらせて考えた時、定量的把握とはこうだというはっきりしたイメージが浮かばず、どうしたものかと悩んでいました。そこで背伸びをしてもしょうがないといった開き直りの心境で今回のワークショップに臨んだわけです。

しかし、何やら難しい（自分としては）パネルや基調講演の連続で開き直りも通用しないかと落ち込んでいたのですが、ビジネス関係のグループに入りそこで行われたメンバの自己紹介で、自分と同じような環境で仕事をしている方が多数いるのに気付き、けっこう感激してしまいました。

その後討論に入ったわけですが、最初は環境が似ているためかお互いの愚痴の聞き合いになってしまい、色々と社会勉強をさせてもらいました。討論場所を会議場から小岩井農場などに移し、熱がだんだと入って来るにしたがって一つ強く感じたことがありました。それは、今回のワークショップのようにひとつずつでもいいから自分のイメージを明確にしていけば、とっつきにくいことでも「どうにかなりそう」だと思えるようになったことです。自分はとっつきにくいことは「どうにもならない」と考えるタイプですので、これは自分自身にとって大きな変化でした。

この感想の中で『ソフトウェア品質の定量的把握』とはなんぞやといった問題にほとんど触れてはおりませんが、それはいまだにイメージがはっきりしていないからではなく、はっきりさせようとしている途中であるからだと思っています。

したがって、このワークショップで得た「どうにかなりそう」指向を有効に使い、またまたどっぷり浸かってしまった泥沼環境から一日でも早く抜け出し、その勢いで今回のテーマをより深く継続的に考えていきます。

Position Statement

白木　真二（３０歳　ＡＢ型）
立石電機（株）長岡事業所
産機統括事業部メカトロ事業部開発室
〒617 京都府長岡京市下海印寺
075-951-5111 内線(387)

ポジション・ステートメント：

問題点：　現状では，ソフト品質として，単位ステップ当たりのバグ数，単位ステップ当たりのテスト数，バグ成長曲線による残バグ数等の信頼性の側面からしか判断できていない．その為，保守性，モジュール独立性等に問題がある．又，品質に対する開発者の意識が低い．

改善策：　現在，ソフト品質ワーキンググループを社内で作り，新しい品質の基準作りを行っている．その１つとして，品質機能展開をソフトウェアにも適用してもよいと判断（個人ベースだが）している．しかし，この品質機能展開では，前工程に多大な時間を要する．従って，これ以外にもっとよい方法（ツール）がないか，検討中．

■ 作成者　　白木真二

■ 業務内容
ロボットを制御するコントローラのソフト開発

■ コントローラの概要
○ 作業を記述する表面言語としてＢＡＳＩＣ準拠のロボット言語を採用。
○ コントローラだけで、モータの位置制御が可能。

■ ソフト開発に使用している言語
○ Ｃ言語
○ アセンブリ言語（処理速度を要求される所だけ）
［補足］　Ｃ言語とアセンブリ言語の比率　１：１　（ソースコードレベル）

■ 自分のポジション
○ アセンブリ言語で作成しなければならない機能の機能設計・構造設計・外部仕様作成（Ａ）とプログラミング・テスト作業（Ｂ）を行っている。
比率　Ａ：Ｂ　＝　８：２

■ 現状の問題点
○ ソフト開発ステップの局面
・ 機能設計・構造設計の各ステップの区切りをきっちりとレビューできていない（レビュー率３割程度）。
ここで、総バグの３～４割が作り込まれている。
ここで作り込まれるバグは、
要求性能を満たしていない
要求機能の１部がない
要求機能以外の機能がある
のように、要求仕様に直接影響を与える致命的なバグが多い。
［理由］とにかく、すぐにプログラミングすることに注力してしまう。

○ 仕様書（機能設計、構造設計、外部仕様、その他）作成局面
・ 開発スタート時に、各種用語を明確に定義していなかったため、Ａ氏の作成した仕様書とＢ氏の作成した仕様書で、同じことを表現しているにもかかわらず、プログラマが勘違いをして、異なった動きをするルーチンを作成してしまった。
これにより、仕様書の修正・ソフト修正に２日かかった。
・ マニュアル作成において、ユーザーサイドに立った書き方をしていなかったため、工場からの指摘で再度書き直した。これにより、２月遅れとなった。

○ ソフトのメンテ局面
・ 処理速度を重視する余り、テクニックの塊みたいなソフトウェアとなった。
このソフトウェアの場合、担当者に直接聞くしか引き継げなかった。

○ リリース局面
・ ソフトのリリースは、バグ成長曲線から残バグ３、検出率０．１以下でありながら、市場バグが多数（１０こ程度）発生。
これは、テスト方法に問題があったためと判断する。
機能の一部だけを集中的に行い、他の部分の十分ではなかった。
時に、リアルタイムで走るルーチンは、テスト方法が難しく、億劫になりがちである。

白木　真二
立石電機（株）長岡事業所　ＣＫ課

ワークショップの感想

ワークショップに参加するに当って

・ソフトウェアの品質を測定できる何かを得る

ことを目的としていた。
その結果、２つの情報が得られたことは有意義であった。
その他、ワークショップについて感じたことを列挙する。

・会社から３泊４日の休み（？）をもらい、交通費もいただいたからには、必ず何かを得て帰ろう、意気込んでいたが、完全に空振りしてしまった。
ワークショップの進め方が、これまで経験していない方法（わいわい、わきわき、遊びを意識しながら、楽しく、重くならないように）であったため、最初は戸惑いと、来なければよかったかな、という感じであった。
このような進め方もひとつの発想の転換、と割り切り、早く遊ぶために、密度の濃い議論を行うような姿勢へと変化して行った。
そして、以前に受けたレクチャーの一つを思い出した。
「外注さんにソフト開発を依頼するとき、これだけの期間でやって下さい。ただし、予定期間よりはやく開発が終了しても、お金は払います。そして、残った期間は、何を行っても（遊んでも勉強しても）かまいません。」とソフト依頼した場合、予定より早く、よい品質のソフトが上がってくる。
これも、目的（遊びたい）がはっきりしているための効果のような気がする。

・各種の業務の方々と議論したが、アプリケーションレベル、ＯＳレベル、制御レベルによって要求される品質のレベルが異なっているのを感じた。
特に、アプリケーションレベルでは、業務の大半が機能拡張・移植作業であるため、流用性が最も重要な要素であるようだ。
この辺りについて、もっと議論をしても良かったかなと感じた。

・ワークショップの開催場所が、遠い盛岡であり、ハネムーン気分で左脳ではなく、右脳を使用できたように思う。
盛岡に向う飛行機の中で、速読の本を読んだが、この本の中でおもしろいことが書かれてあった。
日本人は、小・中・高校・大学で記憶作業ばかり行うため、左脳しか使用していない。従って、非常によい発想が生れない。ところがアメリカでは、大学の講義の中に速読法があり、ここで右脳を訓練している。そのため、よい発想ができる。
今後の日本のソフト開発を考えるとき、右脳を刺激・開発するようなシステムがでてくる必要があるのではないか。

［Ｐｏｓｉｔｉｏｎ　Ｓｔａｔｅｍｅｎｔと感想文］　　８９／９／１８

白井　一好
三菱電機東部コンピュータシステム（株）ビジネスシステム第２部第２課
〒１０５　港区海岸１－９－１５竹芝ビル１１Ｆ（ 03-434-9990）

Ｉ Ｐｏｓｉｔｉｏｎ Ｓｔａｔｅｍｅｎｔ

今まで，バグ検出件数，試験項目数などの集計に気を取られ過ぎていた。というより，集計をしたことの満足感のみに満足していた。集計した結果をどうフィードバックし，次期プロジェクトに生かしていくのかを最大の課題として盛岡に乗り込んだつもりだったが，結果としてその答えは見付からなかったようだ。

ただ，次の２点を実施することにより，答えを自分の力で発見していく気にはなった。

①出荷判断基準はこうとらえる

Ｓ／Ｗの品質の定量化，つまり，出荷の判断基準（あくまでもビジネスグループのとらえ方ですが・・）をどう設定するのかという問題に対し，試験項目を漏れなく洗いだし，すべてを試験する（合格する）ことで，品質をとらえていこうと思う。

１００試験項目を設定し，７５件試験完了していたら，７５％の品質。あと２５試験行う。（実際には進捗管理用に終わりそうだが・・）

具体的には，

１）１試験項目の重みを統一する。－－＞これは実現可能

２）試験計画（試験項目設定）を綿密にたてる。－－＞試験項目数＝基準だから

３）レビューを今以上に充実する。－－＞レビューした結果，漏れがあったら注意力，技術力不足と諦める。

②品質表を充実していく

現状，総合試験計画を立てるとき，品質表なるものを活用している。いろんな角度から漏れがないように，おおざっぱに試験項目を挙げたものである。

- 正常データ試験
- 異状データ試験
- 重複処理試験
- 容量試験
- Ｈ／Ｗ環境試験
- ＪＯＢ関連試験
- 性能試験
- リカバリー試験
- 操作性試験

基本仕様が決まった段階に，それぞれの試験で思い付くまま項目を挙げておき，大きな抜けを防ぐ。プログラム作成が終わる頃，詳細な試験項目を書いていく・・といった具合である。

まだ漠然としすぎている品質表だが，プロジェクトを何回となく進めていくうちに充実したものを作っていこうと考えている。

具体的には，

１）出荷後のバグを分析する。－－＞この試験をしたら，このバグは救えたという考え方に立ち，品質表に追加，修正していく

２）とりあえず，汎用的なものは目指さず，そのシステムに密着したものを作る。－－＞まずは，身近な所から・・

Ⅱ 感 想 文

盛岡では大変お世話になりました。

非常に貴重な体験をさせていただき，今後の仕事に役立てることができそうです。

そこで，一言感想・・・・

①学界，他メーカーとも，まだ品質の定量的把握の手法は確立されておらず，技術力，レビューなどに依存している。

②品質の定量的把握は，統計的な要素が強い。

③わんこそばは，体によくない。

以上

Position Statement

松田　正人（２７歳　Ａ型）
（株）ソフトウェアコントロール
〒550 大阪市西区江戸堀１－２－２　大同生命肥後橋ビル８Ｆ
06-448-3772

経験：
アセンブラ（4/8 ビット）及び C 言語による家電関係のアプリケーションプログラム

ポジション・ステートメント：ソフトウェアの品質の定量的把握について

現状と問題点：

最近ではソフトウェアが身近な存在になってきて直接機械を操作して使用する人が増え，家電製品においてはマイコンを組み込んだ製品が当たり前となって，それと同時に高品質なソフトウェアが要求されてきた．

確かにここ数年来よりソフトウェアの品質は向上しているように思われるが，いまだ高品質なソフトウェアを定義することができないままである．

ユーザというものはソフトウェアを評価するにあたって品質の良さをどこに求めているかがまちまちなのでとうしても定量的に評価できないことが多い．

そのため品質の保証をするためにはいくつかのキーワードがあって，それらを全てクリアすればいいとなってしまい，数値をつかった定量的評価をしていないと考えられる．

それでは製作する側の立場としてはソフトウェアの品質をどう評価しているのだろうか．またそれを定量的に評価しているのだろうか．

製作する側もソフトウェアの品質を評価するのにいろいろポイントがあって把握しきることは難しいと思われるのでここでは「バグがないこと」にポイントをしぼって考えてみる．

ソフトウェアに「バグがない」ということを証明することはまず不可能で，逆にバグはつきものであるという認めていう人も少なくはなかろう．

私がこの３年間経験してきた中でバグがないことを評価するための構成はテスト成績と考えていた．

テストで品質を評価するためにはテキストケースの内容，量が正しいかどうかを決定しなければならない．

テストケースがユーザの要求仕様の満たされているのは当然であるが，時にはテストする環境が実現できないとか，テストケースが無数にあってテストしきれないといったケースもあるのでユーザとレビューを行い判断することにしている．

後はテスト仕様書どおりの結果が得られればバグがないと判断できるわけである．

しかしテストを行った結果，バグはなかったことがイコール高品質なソフトウェアと判断するのは早計

Position Statement

で，もしかするとバグがでてこなかったと考えないとあとでとんでもない結果を招いてしまうこともあるだろう．

だからそれに基ずいた範囲をテストしてもバグは残っていると考えられ，テストの成果が高品質とはいいきれい．

このことから品質を評価するためにはテストのやり方や内容ではなく，テストだけに頼ってしまっていること自体に問題があると思われる．

対策：

品質を良くするためにはできるだけバグの数を減らさなければならない．

そのためにはまずテストの過程だけに頼るのではなくもっと前工程からバグをひきだすほうが良いと考えられる．

さてその工程毎に評価する方法だが，定量的に評価するために何かにデータを用いて品質を保証するための目標値を設定する．

まず，元データは作成しようとするソフトウェアの予想ステップ数とし，目標値となるのはテストケースの数や，バグの数である．

これらが推測できればこのデータをもとに製作るればソフトウェアの品質は定量的に把握できろと考えていいのではないでしょうか．

ただしこの方法は一朝一夕にできるものではなく推測するためのデータも必要である．

私どもでは今期より推測するためのデータを作成し，各個人で実践を繰り返している．

これを使用した結果は全て出たわけではないが，テストに至る工程までに出せるだけのバグは出しているので，組合わせテストでしか出ないバグ以外はほとんどないと予想している．

これが予想と変わらずにすすんでいればその工程での品質は報奨できるわけであり，ソフトウェアを定量的に把握するためにはかなり有効であると思われる．

ソフトウェアの品質の定量的把握においての問題点

前回のポジションペーパーの結論で述べたようにソフトウェアの品質を把握するために、各工程別に行なうべきであるという点で遭遇した問題を考えてみる。

今後品質管理をしていくにあたって　どの工程でどのような品質があれば最終的に高品質なソフトウェアと評価できるだろうか

開発環境が良く、開発期間も余裕があるようなプロジェクトの場合ならできる限りの品質管理を行なうこともできよう。

またユーザ側との見積り段階で品質管理をするための時間をあたえてくれる場合も同様である。

ところが今までの経験上　時間、工数、納期に限界がある場合　どうやって品質管理をしていくか考えていなかった。

しかしそれではいつまでたっても品質はよくならない。

そのためにどうすれば品質管理ができるかを考えてみれば、少なからずとも各工程のポイントとなる部分だけを押さえる必要がある。

そこで現在　表１，２をもとに定量項目、定性項目を選択して数値化し、実績値で評価する方法をとっているが、選択された項目の評価までされていないのが現状である。

今後　品質管理をしていく上で、開発環境にかかわらずどの工程でどの品質が評価できればいいのかを考えなければならない。

例　こんな開発プロジェクトの時　どうしますか

- 開発期間が短い。
- プログラムが分かりにくい。
- 作って見ないとわからない部分がある。

表１　評価尺度と工程

	明示度	使用容易度	資源活用度	環境独立度	仕様確定度	無矛盾度	作り込み度
ｼｽﾃﾑ設計	○	○			○	○	○
ﾌﾟﾛｸﾞﾗﾑ設計	○	○	○	○	○	○	○
詳細設計	○		○	○	○	○	○
ｺｰﾃﾞｨﾝｸﾞ	○		○	○		○	○
単体試験	○						○
組合試験	○						○
総合試験	○						○

表２　評価尺度の定量項目と定性項目

	定量項目	定性項目
明示度	・ドキュメント量　・コメント率 ・レビュー回数　・外部参照データ数 ・試験項目数　・未解決不良件数 ・モジュール分割数 ・モジュール毎の平均ステップ数	・ドキュメントの分かりやすさ ・リストの分かりやすさ ・試験結果の確認のしやすさ ・標準化遵守度合
使用容易度	・レビュー回数　・質問件数 ・クレーム件数　・再現不能不良件数 ・工程別工数比　・ＭＴＴＲ ・打ち合せ回数	・操作のしやすさ ・応答の速さ ・便利さ ・転用度合
資源活用度	・プログラムサイズ ・ファイルサイズ ・実行速度 ・メモリ容量（静的・動的） ・ファイル容量	・無駄のなさ（ﾒﾓﾘ・ﾌｧｲﾙ・ﾃﾞﾊﾞｲｽ） ・ゆとり度
環境独立度	・外部参照データ数 ・使用ファイル数 ・その他リソース数	・装置に対する依存度合 （ﾒﾓﾘ・ﾌｧｲﾙ・ﾃﾞﾊﾞｲｽ） ・言語特性に対する依存度合 ・その他制約状況
仕様確定度	・ペンディング件数　・打ち合せ回数 ・ドキュメント不良件数 ・修正回数　・レビュー回数 ・仕様変更件数　・仕様確定遅延率 ・質問回数	・設計技術力 ・満足度合
無矛盾度	・試験項目数 ・不良件数（ドキュメント・プログラム） ・修正回数 ・レビュー回数	・ドキュメントとソースコードの対応 ・変更管理の徹底度合 ・標準化遵守度合 ・製作技術力　・設計技術力
作り込み度	・不良件数（ドキュメント・プログラム） ・不良収束率　・未解決不良件数 ・試験項目数　・再現不能不良件数 ・試験項目消化率　・再利用率 ・レビュー回数　・工程別工数比 ・工程遅延率　・レビュー工数比	・試験項目の妥当性 ・レビュー内容の妥当性 ・再利用物件の使い込み度合 ・工程作業の安定度合

第7回夏のプログラミング・ワークショップ参加報告書

（株）ソフトウェアコントロール　　松田　正人

今期からわが社でも品質管理を行うようになりましたが、ワークショップに参加するまではこれを行うことの重要性を理解していなかったのが現状でした。

ですからポジションペーパーを見て頂いてもわかるように”バグのないプログラムをいかにして判定するか”という問題に対して各工程ごとに数値（目標値）を設定して実績値と評価するのが１つの方法であると述べる程度でした。

ワークショップに参加することで品質管理の重要性も少しは理解でき、また今後の開発過程でもうまく利用して定着させていこうと考えています。

また参加前の自分のポジションは大きく変わっていませんが、各工程ごとに品質を管理するために設定の仕方や評価の仕方は検討しなければならないと思われます。

現在　わが社の品質管理方法がＩＳＯの品質管理モデルに基ずく品質管理方法に似た形にできるかどうか自分なりに検討中です。

今後　このようなワークショップがあれば是非参加したいと思っています。

以上

<ポジション・ステートメント>　　平松由美子

1．”定量的な把握”に関する問題点！

～　現状　～

現在担当しているプロジェクトでは、品質の定量的な把握をするため次のような事を行った。

①単体試験時の各モジュール毎の障害件数の収集
②単体試験後の机上デバッグ時の各モジュール毎の障害件数の収集
③組合試験時の障害記録・障害管理台帳・障害管理グラフの作成
④デザインレビューの実施時間・指摘項目・対象仕様書枚数・参加人数の収集
⑤容量の推移

②のデータは、組合試験工程の妥当性を見るために利用された。
③の障害記録・障害管理台帳は、処置・確認の確実なフォローと処置順位のプライオリティ付けに利用された。
③の障害管理グラフは状況把握の他に、追加試験を行うか否かの判断に利用された。
①④⑤はプロジェクト遂行中は特に利用されていない。

②の”単体試験後の机上デバッグ”は通常のプロジェクトでは行わない。しかし、組合試験を行ったところバグが多すぎて工程遅れが生じたため、単体試験後机上デバッグを実施したモジュールから組合試験を行うことにした。（①から立てた対策ではない）

～　問題点　～

①”品質の定量的な把握”＝”障害件数”と、なってしまい試験フェーズにならないと、定量的な把握が出来ないこと。

②把握したデータから、うまく品質が読み取れないこと。
つまり、データを取っただけとなってしまうこと。特に、デザインレビューのデータは”これだけやりました”というものにすぎなくなってしまうこと。

③最大の問題点は、システム設計でのバグが、最後の試験フェーズにならないと発見できないことである。つまり、設計の品質の定量的な把握が、設計完了時に出来ないことである。

④”ソフトウェア品質が良い”とは、ユーザが”役に立つ！使いやすい！と思うこと”であると思う。極端にいえば、多少バグがあっても、ユーザが”役に立つ！使いやすい！”と思い利用してもらえればよいのではないだろうか？
その、ソフトウェアの役に立つ！使いやすい！というのは、性能値以外にどのように定量的に把握すればよいの不明であること。

～　解決策　～

①設計・制作時のデザインレビューでの、指摘件数に目標値をつくる。

②設計・制作時のデザインレビューも、指摘件数の管理グラフを作成し、収束するまでデザインレビューを実施する。

2．”管理”に関する問題点！

～　現状　及び　問題点～

①障害管理者は存在しても、予測をしたりや対策打ったり出来ないことが多い。
（データを読み切れない、データの取り方が悪い、時間が取れないなど）

②デザインレビューの指摘件数や、容量の変化など、それをいつも管理し予測する人がいない。

③担当者が管理者も行うとき、工程が遅れてくると管理がおろそかになる。

～　解決策　～

①管理だけする人が必要。管理の時間を十分にとる必要がある。
しかし、”十分な時間”とはどれくらいかは、まだ不明である。

②管理者のレベルによりデータを読み切れない、データの取り方が悪いなどと言うことのないようにしなければならない。デザインレビューの指摘件数や、容量の変化などのデータを読み方は、まだルール化されたものは無い。

3．盛岡ワークショップで得たこと

盛岡ワークショップで得たことは、大変たくさんあったが、次のことが実務で大変役に立つと思われる。

①ソフトウェア品質には、”ユーザのための品質と管理者・設計者のための品質がある”ということがわかった。

ユーザための品質とは、ユーザがソフトウェアを買うか、買わないか判断基準となる品質である。ゆえに、品質をユーザが判断できる何らかの方法で完成したソフトウェアを測定し、”このソフトウェアはこれだけの品質がある”ということを提示しなければならないだろう。
（裏をかえせば、ソフトウェアを設計・製作するときの品質の最終目標値となる）

管理者・設計者のための品質とは、コストや納期確保の判断基準となる品質である。
設計・製作の各段階で品質を測定し、どれほど製品の品質目標値に近付づいているか、不良が早期発見されているかを示し、コストダウンと納期確保がどれほど可能か示さなければならない。

②設計・制作時のデザインレビューでの、指摘件数に目標値をつくる場合、目標値を達成するとそれ以上発見できない事があるので、単に何件以上というのではなく指摘しやすい単純ミスなどは除き、処理抜けなど大事な指摘が何件という目標値を設定する。
また、その処理抜けなど大事な指摘がX件以下・Y件以上であれば再レビューする様にする。
X件、Y件は機能・処理は異なると思う。また、どれ程にしたら、適切なのかはまだ不明。

試験の障害件数を指摘件数と置き換えて考えることも可能だと思う。

4．感想

盛岡ワークショップに参加して、ふと考えたことがある。それは、ソフトウェアの価値（値段）についてである。世の中に流通しているものの値段はなににによるのか？ソフトウェアの価値はなにで決まるのか？
例えば、メロンは味と見掛けの美しさという品質評価により値段が決まり、人工は余り関係ないのではないか？絵画（芸術品）は、人が美しい・すばらしいう品質評価により値段が決まり、材料費は関係ないのではないか？
ソフトウェア品質で、ユーザにとって”役に立つ！””使いやすい！”ということは、非常に測定しにくいものである。メロンの味や見掛けの美しさ、絵画（芸術品）の、美しさ・すばらさも非常に測定しにくいものという意味では同じである。（つまり、人の感覚）
”ソフトウェア品質が良い”とは、ユーザが”役に立つ！使いやすい！と思うことだとしたら、ソフトウェアも品質で価値が決まる時代がくるのだろうか？
それとも、お菓子や薬など添加物表示や成分表示と同様に、ソフトウェアの品質表示が義務づけられるようになるのだろうか？
定量化把握とは後者のことを言うのだろうけれど・・・！

Position Statement

三宅　武司（２７歳　Ａ型）
日本電信電話（株）
ソフトウェア研究所
〒108 港区港南１－９－１　ＮＴＴ品川ＴＷＩＮＳ
03-740-5720

経験：
パソコン上のメトリクス支援ツール（Ｃ言語，約４０ｋ）
但し，設計のみ．

ポジション・ステートメント：

品質のとらえ方

ISO や，IEEE，さらに阪大の鳥居研でそれぞれ提唱している様に，ソフトウェアの品質フレームワークがつくられ，ユーザ視点に立って，ソフトウェアを検証しようという気運になってきたが，あくまで，研究レベルの話であり，現場では，ソフトの品質＝信頼性であり，品質を広い概念で捉えることに抵抗がある．（そんなところまで，ユーザの言うことをいちいち聞いていられない）．

品質の定量的把握

品質＝信頼性という概念は，唯一（？）発生バグ数が定量的な品質を表すメトリクスだからではないか？

品質を定量的に把握するためには，定量的なメトリクスを導出してやる必要があるが，使用性の様な定性的な尺度をどの様に取り扱うべきか，導出したメトリクスの妥当性の問題等，解決しなければならない．

体制的な問題

ユーザの立場に立った品質の保証という意味からは，設計・製造部門からは独立した品質保証部門の設置が必要．しかし，この様な部門の設置，権限の持たせ方（品質と納期とどちらを優先させるか？）についてのコンセンサスを社内で得ることが困難．

☆

ソフトウェアのもつ品質の意味を，広く理解させることが，まず，必要ではないでしょうか．そして，もう一つ，品質の定量的把握の基礎となる．定量的（？）品質メトリクスを導出するための手順（方法）を明確にすること．

Position Statement

NTT ソフトウェア研究所
三宅 武司

まず、" 品質" について議論する前に！
従来、ソフトウェアの品質と言った場合、信頼性（バグ摘出率および、ダウン率）を指していた（これらの数値が湯唯一定量化可能なメトリクスであったから？）．しかしながら、ソフトウェアの適用範囲の拡大と、社会に与える影響が大きくなるにつけソフトの品質の範囲が拡大してきた．ソフトウェアの品質モデルについては、従来B.BoehmeやMcCallのモデルが有名であったが、近年ではISOおよびIEEEによる標準化活動および、阪大鳥居研からソフトウェアのフレームワークについての提案がされている．
これらのモデルではソフトの品質を次のように定義している．

・ソフトウェアの品質：ユーザー要求に対して、ソフトウェア（含むマニュアル）がいかに満足しているかの度合

以降この定義にしたがって、ソフトウェアの品質について議論して行きたい．

現在、実際のプロジェクトを対象に、ソフトウェア・メトリクスの定量化およびこれを用いた品質保証プロセスモデルの試行実験および研究を進めている．ここでの経験から次のような問題があった．

『問題点』

・（１）品質の要求
先に記述したように、ソフトの品質はいかにユーザーの要求に適合しているかの度合と定義した．したがって、各ユーザーによって要求する品質が異なる．しかしながら、一般にユーザーすべての品質特性を選択してしまい、要求品質のしぼり込がむずかしい．

・（２）上流工程での品質の定量化
品質保証の効果を高めかつ、確実なものとするためには上流工程において確実に品質の保証を行わなければならない．この段階でソフトウェアの品質を把握するためには、代用特性（つまり内部特性）による品質の把握が必要となる．しかしながら、上流工程で最終製品の品質を確実に説明できる代用特性を導出することが困難である（メトリクスを設定しても本当に最終製品の品質を反映しているのか最終製品が完成するまで解らない）．

・（３）品質の把握
現状では、開発者は開発途中の信頼性？（単体試験摘出バグ数、結合試験摘出バグ数）でソフトウェアの品質を把握しているのみ．ましてや、ユーザーリリース後の［使用性］などは考えていない．もしくは、考える余裕がない．実際に、品質＝信頼性（バグ摘出率）という概念が根強く残っており、ユーザーの要求をいちいち聞いていられないということと、定量化のためのオーバーヘッドを受け入れにくいという問題がある．

『問題点に対しての現在の対策』

品質の定義および、上記（１）（２）（３）の問題点をまとめると、
・（１）品質を広く捉えるために
・（２）品質の定量的把握
を解決しなければならない．以下に、現在試みている対策を述べる．

・（１）品質を広く捉えるために
「米国RADCの調査の例を持ち出すまでもなく、開発されたソフトウェアのうち過半数が満足に使われないまま、闇に葬り去られている（自分の周りを見てご覧なさい！）．これらのソフトは、「ユーザーの要求する機能を満足していない」、「MMIが酷い」、「保守出来ない」等の問題を抱えている．だから、信頼性だけにとらわれず、広い視点からソフトを見ないととんでもないめに逢うよ」と機会ある毎にプロジェクトを脅かしたり、実際に品質メトリクスを開発担当者に見せたりして意識を高めている．
実際に「MMI」や「保守性」については、開発担当者もその重要性を認識しており（それだけ痛いめにあっている？）、しかも、それが、下流工程になってからでは手に負えないことも解って来ている．しかし、ソフトウェアの品質特性の中でも特に「信頼性」の重要度が高いので理屈では解ってもらえても実際には？

・（２）品質の定量的把握
前記のとおり、品質メトリクスを用いた品質保証モデルの適用を行っている．
既存品質メトリクスを用い、ターゲットプロジェクトにチューンさせ使用している．将来は、メタメトリクス？を設定し、ソフトウェアとして最小限確保しておかなければならない品質特性に対するメトリクスを考えている（機能性、信頼性、保守性などの一部の副特性に対するメトリクスになってしまう？）実際には、このメトリクスを用いて品質計測作業を行うわけであるが、現場に適用させる際には、計測作業によるオーバーヘッドが問題になる．しかし、経験上計測によるオーバーヘッドに比べて、後工程における工数削減効果が高いため、プロジェクトリーダーを口説き落とせば（もちろん品質に対する広い理解が必要）実質上は問題とならないはずである（今の生贄プロジェクトは半強制的にやらせていますが）．

（１）、（２）をまとめて
高品質ソフトウェアを作るためには、上流工程から定量的に品質を把握し、品質の作り込をおこなわなくてはならない．これを現場に適用させるには、品質保証のプロセスモデルを構築する必要がある．このプロセスモデルには、「ユーザー要求の把握」・「ユーザー要求の定量化すなわち品質メトリクスの導出」・「品質の把握（計測）」・「計測値の評価」等の技術を確立しなければならない．

盛岡ではこれらの問題点のうち１つでも糸口が見えれば最高と考えています:-)

Position Statement

三宅 武司（２８歳 A型）
日本電信電話（株）
ソフトウェア研究所
〒１０８港区港南１－９－１ NTT品川TWINS
０３－７４０－５７２０
E-mail: miyake@ntt-twins.ntt.jp

ポジションステートメント：

基本的にはポジションに変化は無かったのですが、若干の追加を行いたいと思います。

まず、〝品質〟について議論するまえに！
従来、ソフトウェアの品質と言った場合、信頼性（バグ摘出率および、ＭＴＴＦ等のダウンに関する尺度）を指していた。これは、唯一定量化可能なメトリクスであったからと思われる。しかしながら、ソフトウェアの適用範囲の拡大と社会に与える影響が大きくなるにつれてソフトの品質の範囲が拡大してきた。ソフトウェアの品質モデルについては、従来B.BoehmeやMcCallのモデルが有名であったが、近年ではISOおよびIEEEによる標準化活動によるものや、阪大鳥居研からソフトウェアの品質フレームワークについての提案がされている。
これらのモデルではソフトの品質を次のように定義している。

☆ソフトウェアの品質：ユーザー要求に対してソフトウェア（含むマニュアル）が如何に満足しているかの度合

以降この定義にしたがって、ソフトウェアの品質の定量化について議論 していく。

『ソフトウェアの品質の定量化の問題点』
「ソフトウェアの定量化の必要性」
ソフトウェアも工業製品である以上（もう家内工業であった時代は昔話です）品質の管理、コストの管理、進捗管理等について、経営者・プロジェクト管理者・担当者そしてユーザーたるお客にとっても何らかの指標というものが必要となっている。
・たとえば、適正な価格なのか現在のところユーザーは勿論、誰にも判断がつかない。
・工程終了判断等は「感」と「度胸」と「経験？」がまかりとおっている（感の鋭さがプロジェクトリーダーの力量？）

「上流工程における品質の定量化」
現在のようなウオーターフォール・モデルの開発パラダイムでは、上流工程の品質管理が重要であることは周知の事実である。しかしながら、まだ出来ぬソフトの品質を設計書等のドキュメントから把握することは困難である。設計書の品質チェックというと、えてして‘てにおは’や誤字のチェックになりがちで、本質的なところまでは計測不可能ではないだろうか？。現在のところ、品質メトリクスを用いて設計書類をチェックする方法が手順化されているが（SQMAT等）、メトリクスを設定するのが一苦労である。ソフトウェアはそれぞれ個々に要求が異なり、当然これを測定するためのメトリクスはソフト毎に設定すべきである。しかも、最終的な品質はソフトが完成しなければ本当の品質が確認できないので、設定したメトリクスの妥当性の検証は少なくとも当該プロジェクトが走っている間には不可能と思う。
いよいよ品質の定量化（品質メトリクス）を考慮した新たなソフト開発のパルダイムを検討しなければならないかも？

『品質の定量化へのアプローチ』
ソフト開発現場に品質の定量化を浸透させるために
（1）ソフトの〝品質〟の考え
（2）品質の定量的把握実践
を明らかにしなければならないと考える。

（1）ソフトの品質とは？
米国RADCの調査の例を持ち出すまでもなく、開発されたソフトウェアのうち過半数が満足に使われないまま、闇に葬り去られている（自分の周りをよく見てご覧なさい！）。これらのソフトは、「ユーザーの要求する機能を満足していない」、「Ｍ／Ｍ／Ｉが最低」、「保守できない」等の問題を抱えている。だから、信頼性だけにとらわれず、広い視野にたってソフトを見ないととんでもない事になるヨ！。と機会のある度に教育している。たしかに、信頼性はソフトウェアにとって重要な品質特性の１つであり、これに固執している人もいらっしゃる様子だが。
要は「如何に現場の開発担当者にソフトウェアの品質に関心を持たせるか」である。今の日本の品質保証技術には目を見張るものがあるが（ハードウェア）、これはＱＣ活動によるものが大きいと思う。この活動の特徴はトップダウンによるものではなく、担当者レベルからのボトムアップにある。つまり開発担当者自が常に品質に対する意識を持ち続けることが重要であり、ソフトの開発者も同様に品質に対する意識向上を図らなくてはならないと考えている。

（2）品質の定量的把握の実践
過去に品質メトリクスを用いた品質定量的把握を試みたことがある。
既存の品質メトリクスを参考に、ターゲットプロジェクトにチューンさせて使用した。今後適用範囲を拡大させメトリクスをブラッシュアップさせ、とりあえず最終的にはメタメトリクスを設定してソフトウェアとして最小限確保しなければならない特性をチェックできるようなものを考えている。しかし、現場に適用させる際に工数の問題が出てくる。現在のところどの程度のオーバーヘッドであれば現場で適用してもらえるのか不明であるが、経験的にソフト開発の全工程を通してみれば上流工程で厳格な品質保証を行えば、テストなどの後工程の工数を大幅に削減可能であるので、要は現場の人達の意識の問題と思う。

いずれにせよ盛岡で議論したように、開発過程のメトリクスはあくまでレビューのチェックポイントを与えるものである。そしてメトリクスの設定はバグの分析／整理を体系的に行い、バグを作り込んだ原因を明らかにすることで手順化が可能と考える。つまりバグを作り込まないための方策もしくは、設計技術の１つとしてメトリクスを設定できると思う。そして、このメトリクスを洗練させるために幅広く適用／分析を繰り返し行わなければならない。そして、最終的には上記で述べたようなメタメトリクスのようなものが導出できるものと思う。そして、このメトリクスは開発者にとっては、プログラム作成のガイドラインとして利用可能なものとなるはずである。

「第７回夏のプログラミング・ワークショップに参加して」
私は品質の定量化を業務としているので、特にソフトウェアの定量化および品質の概念について目から鱗が落ちるようなことは期待していなかったのですが、研究所におりますと現場のことがよく分からない（こんなことではこまります！）という理由を無理やりでっち上げ、ワークショップに参加される方々との議論を楽しみにしておりました。
その結果、会社／分野／立場等の違いにより、開発に携わっている人達のソフトウェアの品質／定量化への意識の違いを改めて認識できました。「ソフトウェアの定量化はできるのか？」との出問が出されたときには、唖然としてしまいました（当方ソフトウェアの定量化を研究しておりますので、こんなことは考えてもみませんでしたし、出来ないとなるとメシの食い上げになってしまいます・・・もっとも、だから研究の対象なのでしょうが）。また、開発者と管理者共にソフトの定量化を望んでいるものと解釈していましたが、定量化が開発者個人の管理の道具に使われることを危惧している事も初めて知ることができました。その他、参加者の方々と議論を重ねる上で、開発現場の状況を聞かせていただいて、現場では品質の把握等ソフトの定量化に関して混乱状態であること、そして、それがソフトにかかわっているすべての会社組織の共通な問題であることを確認出来ました。
最後に、久保さんがおっしゃっていたように、ヒューマンネットワークを築くことが今回のワークショップの一番の目的であると思います。どだい定量化をテーマに２日や３日で議論を尽くせたら世話がありません。定量化は１年２年で結果が出てくるようなものではないと思います。今後、参加した人達と連絡を取り合って、どの様に意識が変化していくのか、また、社内でどの様にアプローチしていくのか皆さんの活躍を楽しみにしています。

Position Statement

吉山　徳博（２９歳　Ａ型）
（株）富士通九州システムエンジニアリング
第四システム部第二システム課
〒812 福岡市博多区博多駅前１－５－１　朝日生命ビル１０Ｆ
092-472-6611

経験：
電気ＣＡＤパッケージ開発

ポジション・ステートメント：

現状の問題点：

・　ユーザの要求に迅速に対応する必要があり，開発から提供までの期間が短く（３～６ケ月），品質基準の判断（どこまでチェックすればよいか）が難しい．
・　ユーザ先で発生したトラブルは，緊急を要するものが多く，この対応に追われて，品質管理でなく障害発生状況一覧作成で終始している．

今後の改善策：

・　開発時のチェック機構をルール化し，品質チェックを強化する．
・　トラブル発生に対して原因分析会を行い，開発メンバーに品質の重要性を意識付ける．また，同時にスキルアップを図る．

第7回　夏のプログラミング・ワークショップ（若手の会）

「ソフトウェア品質の定量化把握を目指して」に参加して

氏名　：　吉山　徳博（２９歳　A型）

㈱　富士通九州システムエンジニアリング
第四システム部第二システム課

経験　：　電気ＣＡＤパッケージ開発

ハードウエアに関する品質測定は、既に確立され実施されているが、ソフトウエアに関しては、未だ確立されておらず、試行錯誤で行われているのが現状である。しかし、今後、ソフトウエアもハードウエアと同様に品質測定が重要であり、ハードウエアが通った轍を急速なスピードで進む必要がある。これは、社会的な動きであり、ソストウエアが社会の枝に成りつつある状況では、決して逃げられない道と思う。

現在、ソフトウエアの品質測定の一方法として、ＩＳＯが提唱しているＳＱＭ（Ｓｏｆｔｗａｒｅ　Ｑｕａｌｉｔｙ　Ｍｅａｓｕｒｅｍｅｎｔ）法があり、多くの企業でトライアルがなされている。この手法は、ユーザから発生する要求と、開発時の設計基準を組み合わせた品質表によって、ソフトウエアの品質を測定するというものである。この方法に関しても、現場に適用して品質を向上させるためには、現場にあった品質表を作成することが重要である。また、品質表の精度は、各開発時に発生した問題（例えば障害）を分析し、それを事前に防ぐまたは推測するためには、何を開発時にチェックすべきかを検討し、品質表に反映することによって高めることができると確信した。つまり、開発→評価・分析→開発の循環によって初めて、開発の品質は高まるわけである。

ところが、現実は、このフェードバックの循環が十分に行われていない。今回、このワークショップに参加したことをきっかけにして、品質表という観点で、分析・評価を行い開発作業にフェードバックすることを行いたいと思う。

また、今回のような、社外セミナーで、同世代の同じ立場の人達と、ディスカッションできたことは、大変良い経験になったと思う。ソフト業界も、今後、グローバルな意味でオープンになって行くことが必要と感じた。

以上

Position Statement

鶴林　伸英（２７歳　Ｂ型）
富士通九州システムエンジニアリング
第四システム部第一システム課
〒812 福岡市博多区博多駅前１－５－１　朝日生命ビル１０Ｆ
092-472-6611 内線(624)

経験：
ICAD/SDS　MMF2 プログラム開発

ポジション・ステートメント：

障害件数０件をめざしているが達成できていないのが，ソフトウェア品質の現状である．

私達は，品質管理の方法として，「各工程（PG，IT，ST）別での障害検出目標値の設定」「障害検出状況推移のグラフ化」「障害の原因分析」を行ってきたが，障害原因として明確で，且つ具体的対策が施されていないものとして，以下の点を挙げることができた．

・プロジェクト内における各メンバ間の意識のずれ
・ユーザ影響度の考慮もれ
・思い込み
・チェックシートの不備

そこで上記の未解決項目のつぶし，品質向上の対策として以下のことを実施した．

・作業分担を工夫し全体を把握しているチーフが確実にソースレビューを行える体制を作る．
・開発中に発見した障害で擬似的ユーザ障害対応を行いユーザに対する認識を向上させる．
・方式や処理を決定するまでのプロセスを重視し第三者を交えて最終決定を下す．
・プログラム設計書をソースレビュー時のチェックシートとして有効に活用する．

第7回　夏のプログラミング・ワークショップ（若手の会）

『ソフトウェア品質の定量化把握を目指して』に参加して

氏名　：　鶴林　伸英（２７歳　B型）

㈱　富士通九州システムエンジニアリング
第四システム部　第一システム課

経験　：　ＩＣＡＤ／ＳＤＳパッケージ開発

ソフトウエア品質定量化のための枠組みＳＱＵＡＲＥ（Software Quality Assurance Based on Requirements）は，ソフトウエア信頼性シンポジウムにおける討論に基ずいて提案されている枠組みであり，その特徴として

① ユーザ，管理者，開発者の３視点を導入していること

② 品質表を利用して作業の効率化を目指していること　が挙げられる。

しかし、ＳＱＵＡＲＥを利用する品質表の形式については確定しているが，そこに蓄積すべき内容については実用的なＭｅｔｒｉｃが得られていないのが現状である。

今回のグループ討論で品質表に蓄積すべきＭｅｔｒｉｃの検証を行ったが，漠然としたものしか出ず具体的なものの検証／検出はできなかった。

しかし，ユーザ視点でのＭｅｔｒｉｃ，開発者視点でのＭｅｔｒｉｃを検証していくうちに，開発者視点でしか捕らえていなかった自分の視野の狭さを痛感した。

今までは，『品質が良い＝障害が無い』と思っていたものが『品質が良い＝機能性，信頼性，使用性，効率性，保守性，移植性が充実している』という具合に自己の意識革命になったと思う。

ただ，『ソフトウエアの品質はソースレビュで決まる』という自分の信念は確固たるものになったと思う。

また、同世代の同じ立場の人達と、ディスカッションできたことは自分自身を活性化でき大変良かったと思った。

今後は，今回のワークショップで得たものを活かして，開発作業に従事していく所存である。

Ｐｏｓｉｔｉｏｎ　Ｓｔａｔｅｍｅｎｔ

堀内　秀一　（３０歳　Ａ型）
日本システム（株）
第２システム技術部
〒１９０　立川市曙町１－１８－２
０４２５－２６－５９１１　内線（６６０）

経験：
東京電力（株）東部系統給電指令所　自動給電システム

ポジション・ステートメント：

Ｂｏｅｈｍらの研究に見られるように、ソフトウェア品質にはいろいろな要素がある。その中で特に（ソフトウェアを作る側の人間にとって）重要な要素は信頼性である。

信頼性を計る為にソフトウェア信頼度成長曲線やテスト・カバレージ、テスト項目消化率などが考えられているが、（御存知の通り）いずれもそれ１つで信頼性を計るのは無理がある。また、他の要素も１つのメジャーでは計れないだろう。

しかし、要素を細かく分割し（分割してみろと言われると困るれど）、１つ１つの要素についてメジャーを捜していけば、その組み合わせで品質を定量的に把握できるかもしれない。かなり、将来のことに成りそうだ。

ソフトウェア品質の定量化は、ソフトウェア技術者にとって有用であるばかりでなく、ユーザーにとっても目に見えないソフトウェアを評価する為に有用である。（辛い面も有るが）ソフトウェア品質を定量的に把握することは必要だ。

第７回　夏のプログラミング・ワークショップの感想

日本システム　堀内秀一

私は今回初めてこのワークショップに参加しました。立場や環境が違う人達と会って、話を聞くうちに貴重なカルチャー・ショックをたくさん受けました。この体験は今後の私の活動に役立つことでしょう。普段、社内の人達と話していても解決しない問題のヒントを得たこともありました。

会社の後輩達にも、この貴重な体験をしてほしいと思っています。

Position Statement

高田　義広（２４歳　Ａ型）
大阪大学基礎工学部情報工学科
〒560 豊中市待兼山町１－１
06-844-1151 内線(4816)

ポジション・ステートメント：

現在，提案されている枠組みに基づいて，ソフトウェアに対する品質要求の実現を支援するシステムを開発しています．

この枠組みでは，ソフトウェアに対する品質要求を効率よく実現するには，（１）要求を明確にする，（２）要求を実現するのみ適切な開発プランを立てる，（３）開発プランを十分に実行する，（４）実現された製品の品質を確認する，ことが必要と考えています．また，そのために２種類の評価尺度が必要と考えています．１つは，品質を確認するための製品の評価尺度です．システムでは，枠組みの適用を容易にするために品質表を利用します．また，品質要求から開発プランを作成する機能を持ちます．

現状では，枠組みやシステムは実際のソフトウェア開発には適用されていません．今後，実際の適用事例を増やして行き，枠組みやシステムの有効性や問題点を明らかにできればと考えています．