特集論文

定量的品質管理の効率化によるソフトウェア設計品質の向上

鈴木　博*　　高橋洋一***
塚本裕嗣**　　徳本修一***
矢野雅嗣***

Improvement of Software Design Quality by Effective Quantitative Quality Management

Hiroshi Suzuki, Hiroshi Tsukamoto, Masatsugu Yano, Yoichi Takahashi, Shuichi Tokumoto

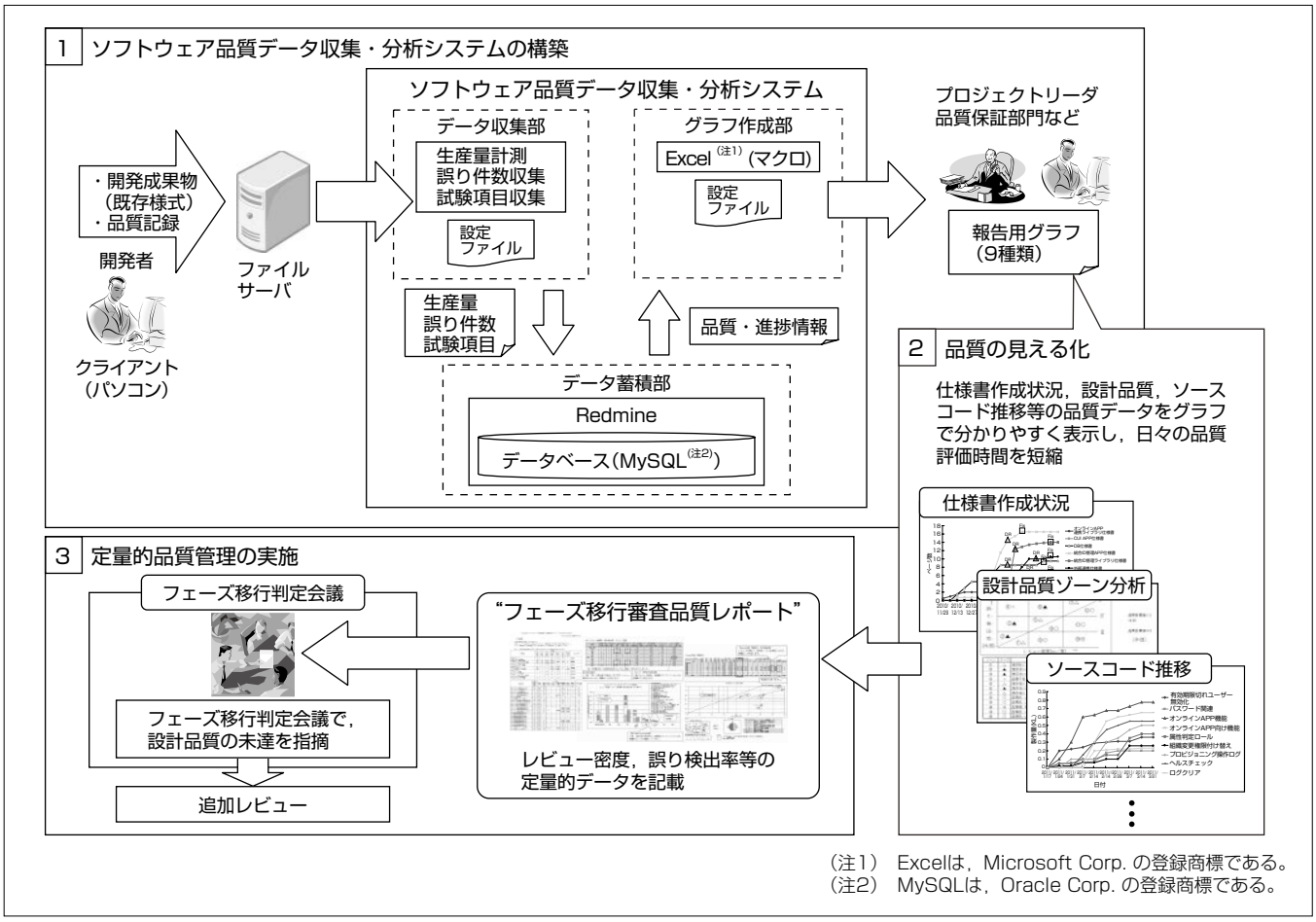
要　旨

近年，ソフトウェアが大規模・複雑化する中，計画されたQCD（Quality（品質），Cost（費用），Delivery（納期））を達成することはソフトウェア開発における大きな課題の一つとなっている。計画QCDを達成するためには，設計段階で品質を作りこみ，試験段階ではその品質を確認するという開発プロセスを短期間で効率的に実施することが重要である。そのためには，設計時に作成する仕様書などの品質を定量的に把握・分析し，弱点を早期に抽出して設計品質を確保する必要がある。

品質の定量的把握・分析に必要なソフトウェア品質データの収集・分析は，これまで人手によって表計算ソフトウェアなどのツールを使って行われていたため，データ収集・分析に時間がかかり開発のホールドポイントで定量的な品質の把握が十分にできないという課題があった。

この課題を解決するため，効率的な定量的品質管理の実現を支援する“ソフトウェア品質データ収集・分析システム”を開発し，ソフトウェアの設計品質向上を実現した。

このシステムは開発者が作成する仕様書などの開発成果物，及び仕様書のレビュー結果等の品質記録を入力とし，仕様書作成状況などを表す複数のグラフを自動生成する。既存の仕様書様式を変更することなく品質状況を表すグラフの作成が可能であり，品質管理のための開発者への負担を最小限に抑え，開発フェーズごとに必要な品質情報を短時間で提供することで，品質向上を実現した。



（注1） Excelは，Microsoft Corp. の登録商標である。
（注2） MySQLは，Oracle Corp. の登録商標である。

“ソフトウェア品質データ収集・分析システム”を活用した定量的品質管理

開発者は作成した仕様書，及びそれらに関わる品質記録をサーバに格納する。このシステムはこれらを入力として，仕様書作成状況などの品質に関わる複数のグラフを自動生成する。開発フェーズ移行判定会議ではこれらのグラフを用いて作成した品質レポートを基に移行判定を行う。

*三菱電機㈱ 設計システム技術センター　**三菱電機エンジニアリング㈱　***三菱電機㈱ 情報技術総合研究所

1. まえがき

近年，ソフトウェアが大規模・複雑化する中，計画されたQCDを達成することはソフトウェア開発における大きな課題の一つとなっている。計画QCDを達成するためには，設計段階で品質を作りこみ，試験段階ではその品質を確認するという開発プロセスを短期間で効率的に実施することが重要である。そのためには，設計時に作成する仕様書などの品質を定量的に把握・分析し，弱点を早期に抽出して設計品質を確保する必要がある[1]。

品質の定量的把握・分析に必要なソフトウェア品質データの収集・分析は，これまで人手によって表計算ソフトウェアなどのツールを使って行われていたため，データ収集・分析に時間がかかり開発のホールドポイントで定量的な品質の把握が十分にできないという課題があった。

この課題を解決するため，効率的な定量的品質管理の実現を支援する“ソフトウェア品質データ収集・分析システム”を開発し，実開発へ適用してソフトウェアの設計品質を向上させた。

本稿では，このシステムの概要，機能，及びそれらの機能によって解決される問題点を中心に述べる。

2. ソフトウェア品質データ収集・分析システム

2.1 システム構成

このシステムは**図1**に示すように，“データ収集部”“データ蓄積部”“グラフ作成部”の3つの要素で構成した。

開発者は設計仕様書などの開発成果物をクライアントパソコンで作成し，サーバに格納する。また，仕様書のレビュー結果などの品質記録も合わせてサーバに格納する。これらの開発成果物及び品質記録をこのシステムへの入力とする。

2.1.1 データ収集部

データ収集部では，サーバに格納された仕様書のページ数などの生産量を計測する。また，品質記録データから誤り件数や誤り種別などの情報を取り出し，生産量と併せてCSV(Comma Separated Values)ファイルを生成する。このCSVファイルはRedmine[2](データ蓄積部で使用しているプロジェクト管理ソフトウェア)の管理単位である1チケット分の情報に相当する。

ファイルサーバ上の品質記録は通常表計算ソフトウェアのデータとして格納され，次のような状況が想定される。

(1) 開発プロジェクトごとに書式やファイル名称，ファイル数が異なる可能性がある。
(2) 開発プロジェクトごとに，ファイルの存在するサーバ上のディレクトリが異なる可能性がある。
(3) 開発フェーズによって書式が異なる可能性がある。

そのため，データ収集部では，その設定ファイルでデータの抽出元を品質データファイルのセル単位まで指定可能としている。さらに，開発プロジェクトごとに別の設定ファイルを用意することで，プロジェクトによる仕様書などの書式の違いに対応し，複数プロジェクトでの品質データの同時計測も可能としている。

2.1.2 データ蓄積部

データ蓄積部では，データ収集部で生成した品質関連のCSVデータをRedmineで処理可能なチケット形式に変換して，データベースに格納する。

Redmineはオープンソースソフトウェアであり，三菱電機での使用実績もあることから，このシステムの開発期間短縮及び開発コスト削減のために採用した。また，Redmineを使用することで障害管理や進捗管理などのRedmineの持つ各種機能を直接使用することも可能となり，プロジェクト管理・品質管理方法に柔軟性を持たせることができた[3]。

2.1.3 グラフ作成部

グラフ作成部ではデータ蓄積部に格納されている品質データを基に，仕様書作成状況や仕様書の品質評価状況を表

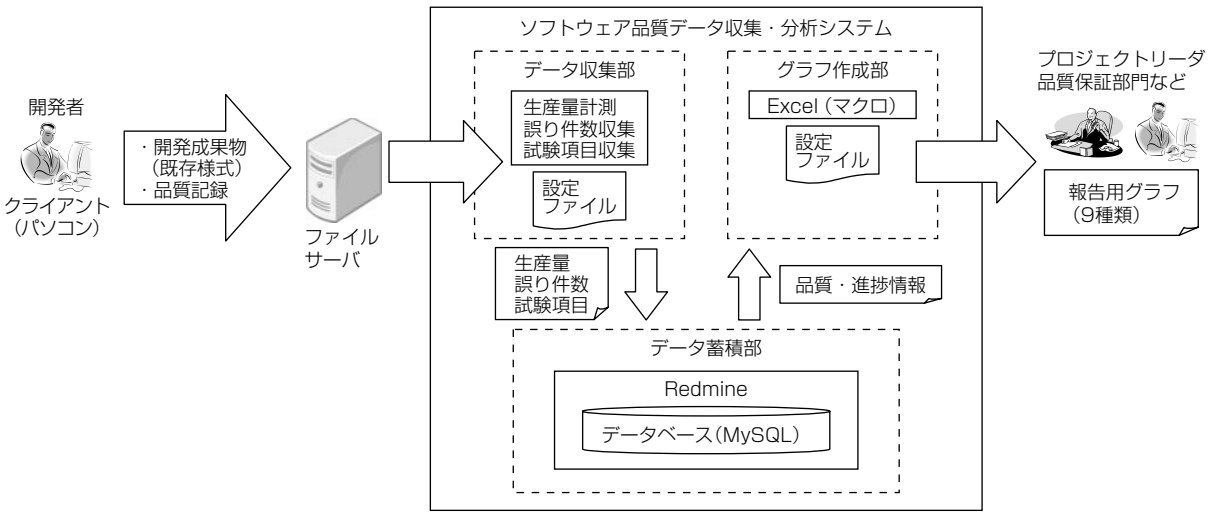


図1．ソフトウェア品質データ収集・分析システム

すグラフ(**表 1**)を作成する。グラフ作成部の設定ファイルにどのグラフを作成するかを指定することで，プロジェクトリーダーや品質保証部門が必要とするグラフを作成することができる。また，将来プロジェクトマネージャーがブラウザ経由でリアルタイムに品質状況を確認できるようにするため，イメージファイル出力機能も実現した。

2.2 機　　能

2.2.1 既存様式の仕様書から品質データを自動収集

異なるプロジェクトでは，使用する品質データの記録様式などが異なる場合が多く，品質状況を表すグラフを生成する場合は，システム仕様に合わせて入力ファイルを作成し直す必要がある。しかしながら，プロジェクトごとに入力ファイルを作成し直すことは，開発工程上大きなオーバヘッドとなり，工程遅延を引き起こす要因にもなりかねない。そのため，このシステムでは品質データを収集する際に設定ファイルに各種情報を設定することで，既存様式で作られた仕様書などを変更することなく，品質データの自動収集を実現している。

設定ファイルの記載例を**図 2** に示す。この例では，“仕様書作成開始日”“仕様書作成完了日”“仕様書ページ数”等のデータが“ファイル名：ソフトウェア仕様書”“シート名：品質基礎データ”に格納されており，それらの位置を示すデータを“データ収集部設定ファイル”に設定することで，対応する値を参照している。このように，設定ファイルでデータの所在を指定することで，異なる様式の仕様書にも柔軟に対応することができる。

また，“仕様書作成開始日”などの追加属性はRedmineの標準チケットのカスタムフィールドを使用して拡張し，一つのチケットで品質情報まで含めて一元管理した。

2.2.2 品質状況をグラフで分かりやすく表示

既存様式で作成された仕様書から自動収集したデータを基に複数のグラフを自動生成することで，短時間で品質状況を見える化することができる。このシステムで作成するグラフは，設計段階だけでなく製造・試験段階での品質確認も考慮して作成した。今回，設計段階で実開発へ適用した主なグラフは次のとおりである。

(1) 仕様書作成状況

サーバに格納されている仕様書のページ数を自動的にカウントし，日次・週次の仕様書の作成状況をグラフ化する(**図 3**)。図から，作成開始日から作成完了予定日までの計画線と比較することで，例えば仕様書Cは作成当初は計画より遅延していたものの，後半は計画通りの進捗に挽回(ばんかい)していることが分かる。このようにして，作成作業が遅延している仕様書を把握し，改善を図ることができる。

(2) 設計品質ゾーン分析

仕様書のレビュー結果を基に，1ページあたりのレビュー密度(時間/ページ)と1ページあたりの誤り検出率(件/ページ)を指標として，仕様書の品質状況を見える化している(**図 4**)。図から，“仕様書A第3章”はレビュー密度が低く誤り検出率が高いため，品質不良であり，再レビュー

表 1．作成グラフ一覧

	グラフ名称	内容
1	仕様書作成状況	日次・週次での仕様書作成状況
2	設計品質ゾーン分析	仕様書の品質状況
3	ソースコード推移	日次・週次でのソースコード作成状況
4	試験進捗管理	試験項目数(計画・実績)等の推移
5	機能別誤り発生推移	日次・週次での機能モジュールごとの誤り発生推移
6	試験品質ゾーン分析	機能モジュールごとの品質状況
7	開発フェーズ別誤り推移	開発フェーズごとの誤り発生推移
8	累積工数推移	作業工数(計画・実績)推移
9	残存誤り推定	開発フェーズごとの残存誤り推移

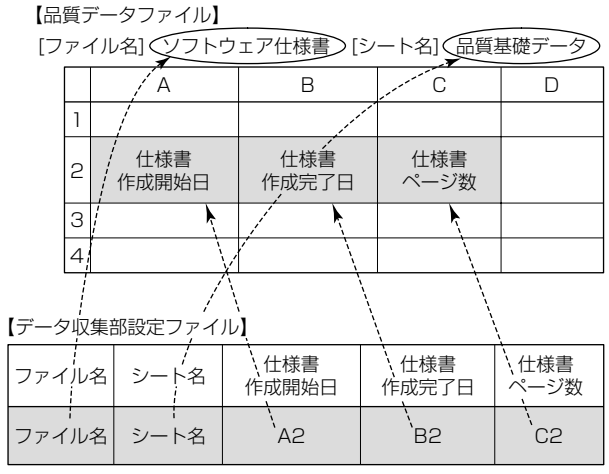


図 2．データ収集部の設定ファイル記載例

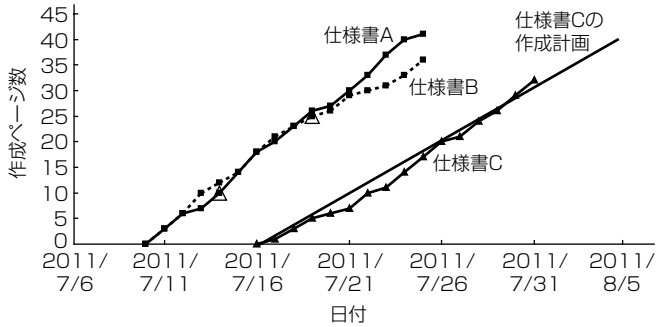


図 3．仕様書作成状況

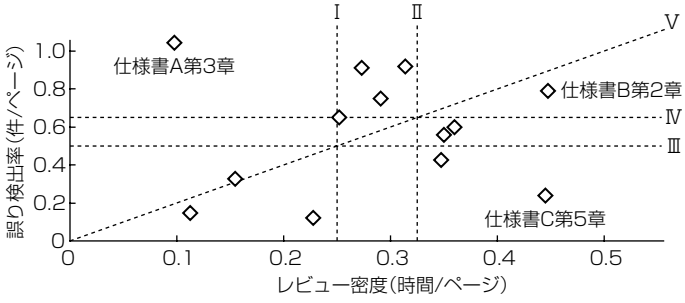


図 4．設計品質ゾーン分析

による改善が必要であることが分かる。また，“仕様書C第5章”はレビュー密度が高く，誤り検出率が低いことから，品質は良好であると判断できる。このようにして，レビューが不足している仕様書(又は仕様書内の章)が明らかになり，再レビューを行うことで設計品質を向上させることができる。

(3) ソースコード推移

仕様書作成状況と同様に，サーバに格納されたソースコードからそのライン数を自動的にカウントし，ソースコードの作成状況をグラフ化する(**図5**)。図から，作成開始日から作成完了日までの計画線と比較することで，例えばモジュールCは作成当初は計画を上回る進捗であったが，後半は計画より遅延しており，対策が必要であることが分かる。このようにして，作成遅延が発生しているモジュールの把握が可能となる。

3. 定量データに基づく品質確認

このシステムを実開発に適用し，出力される各種グラフを活用して，定量的品質管理を実践した。具体的な取組みを次に示す。

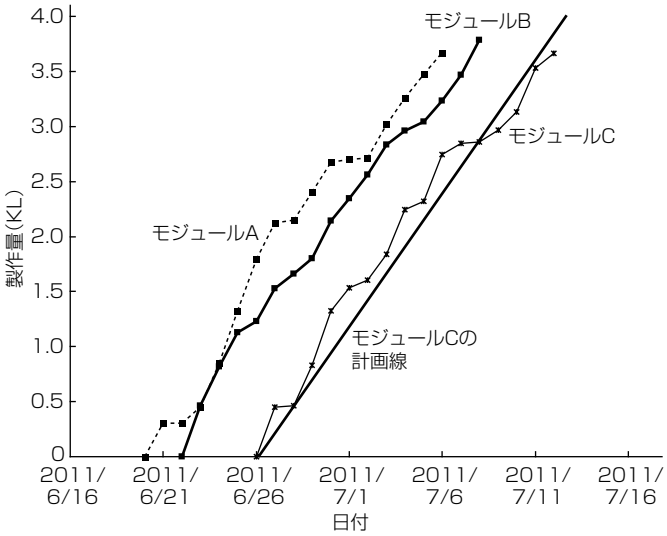


図5．ソースコード推移

3.1 日次変化量による品質確認

“仕様書作成状況”“ソースコード推移”等を日次変化量報告書としてまとめ，仕様書及びソースコードの日々の変化量を監視した(**図6**)。仕様書・ソースコードの作成量が急減に変化している場合は，異なる仕様書・ソースコードの誤った登録や仕様書・ソースコードを誤って削除したなどの可能性が考えられる。また，ソースコードの正常な統合によっても異常な変化量を示す場合もあるため，開発担当者に異常な変化量の理由を聞くことで品質上の問題有無を確認した。問題があった場合は原因分析を行い，是正処置を立案・実施することで品質を確保するとともに，誤操作などの再発を防止した。

3.2 週次品質レポートによる品質確認

仕様書の章ごとの“レビュー指摘率”や“誤り検出率”，及び“設計品質ゾーン評価”等を基に，週次品質レポートを作成し，品質状況が一目で確認できるようにした(**図7**)。品質確認の結果，“レビュー密度”が不足している場合は仕様書の追加レビューを実施して設計品質を向上させた。また，“誤り検出率”が目標値を大幅に上回っている場合は誤りが残存している可能性が大きく，誤りの内容確認などの品質再評価を実施して弱点部分を明確にし，追加レビューを実施した。

3.3 開発フェーズ移行判定会議による品質確認

日次・週次での品質確認に加え，このシステムから出力される各種グラフを基に作成した資料に基づき，マネジメント及び品証部門による開発ホールドポイントでの品質評価を“開発フェーズ移行判定会議”として実施した(**図8**)。人手で品質に関するレポートを作成していた際には，品質確認のための資料作成に時間がかかり移行判定会議に定量的データが提示できない場合もあった。このシステムはその問題を解決し，品質確認対象の仕様書が多い場合でも定量的品質データの短時間での提供が可能となり，次の項目を実施することで，設計段階での品質を確保することができた。

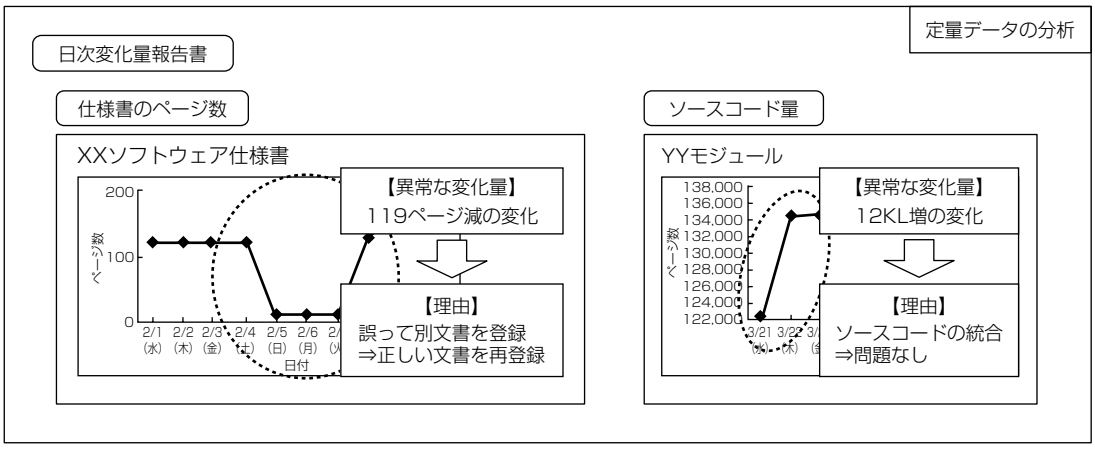
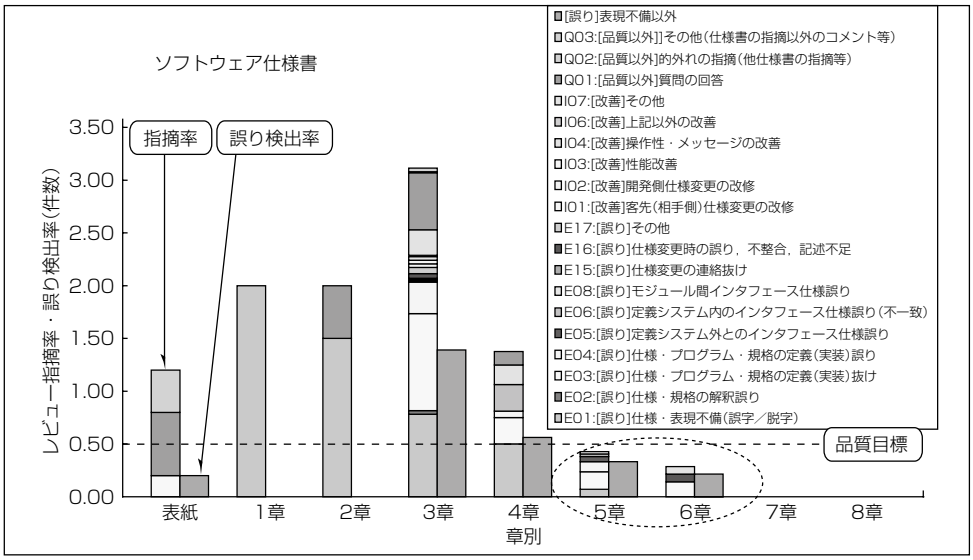


図6．日次変化量による品質確認

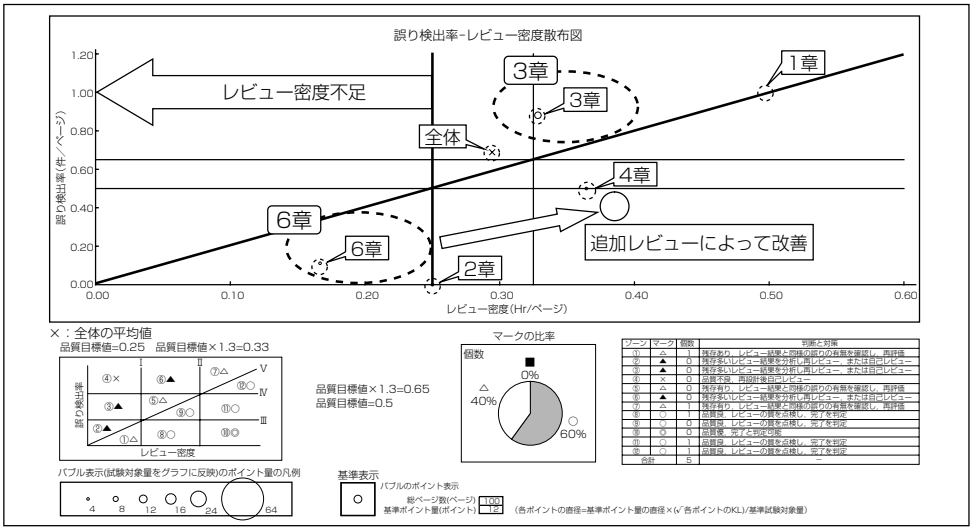
週次品質レポート

章別の“指摘率”“誤り検出率”



5章／6章の“指摘率”“誤り検出率”が品質目標未達

設計品質ゾーン評価



・6章の“レビュー密度”不足
・3章の“レビュー密度”は品質目標達成であるが，“誤り検出率”も高い⇒残存誤りあり，再評価が必要

図7．週次品質レポートによる品質確認

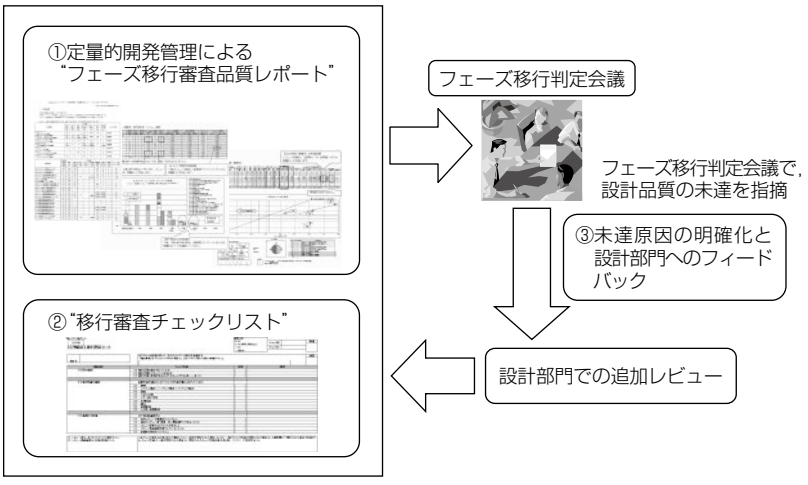


図8．開発フェーズ移行判定会議による品質確認

(1) “フェーズ移行審査品質レポート”による定量的品質評価
(2) “移行審査チェックリスト”による移行条件の確認
(3) 品質目標が未達の場合には，その原因を明確化し設計部門へフィードバック

4. むすび

“ソフトウェア品質データ収集・分析システム”を開発して実開発に適用したことで，品質管理のための開発者の負荷を最小限に抑え“品質の見える化”を実現した。グラフによる分かりやすい品質データを提供することで弱点部分を短期間で明確化し，設計部門による改善を実施することで，設計段階での品質作りこみを可能とした。

今後は，このシステムの適用を複数のソフトウェア開発に拡大するとともに，設計段階だけでなく，実装・試験段階での品質確認・改善にも適用し，製品品質の向上を図っていく。

参考文献

(1) 山下昭裕，ほか：ソフトウェア開発環境の現状と展望，三菱電機技報，**84**，No.5，266～270（2010）
(2) Redmineホームページ：http://redmine.jp/
(3) 小川明彦，ほか：Redmineによるタスクマネジメント実践技法，翔泳社（2010）