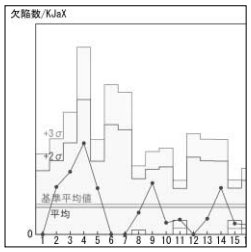


統計的品質管理によるソフトウェアの品質改善

中　塚　康　介*・中　村　伸　裕

Software Quality Improvement by Statistical Quality Control —— by Kosuke Nakatsuka and Nobuhiro Nakamura ——
For quality assurance during software development, preventing and removing code errors is important. While strict quality control is conducted during production of industrial products, not enough statistical quality control is carried out in software development. Because of this lack of statistical control of quality in software development, it is extremely difficult to assure that all errors have been removed. Sumitomo Electric has defined measurement parameters such as those that reduce variation in scale and quality, and applies statistical quality control methods such as use of control charts that are popular in industrial production. Sumitomo Electric also estimates potential defects based on performance data. Through these activities, Sumitomo Electric has succeeded in improving software quality.

1. 緒　　言

徹底した品質管理がなされている工業製品に対し、ソフトウェアの開発では統計的品質管理が十分には実践されていない。ソフトウェアの品質問題が多発して30％のユーザーがシステムに何らかの不満を持っている[(1)]のが現状であり、統計的な品質管理の重要性が高まっている。

ソフトウェアの品質が問題となる中で、それを改善するためにCMMI[(2)]などのプロセス改善のためのモデルが構築され、多くの企業で適用されている。それらのモデルでは、開発プロセスの品質を予測し、問題点を定量的に判別して改善するための各種のメトリクスに基づく統計的な管理が重要視されている。

本稿では、当社における統計的品質管理による基幹系ソフトウェアの品質改善の取り組みについて記述する。

2. 楽々Framework®Ⅱによる基幹系ソフトウェア開発

当社では基幹系ソフトウェア開発に独自に開発した楽々Framework®Ⅱを用いている。楽々Framework®Ⅱはデータ中心設計（Data Oriented Analysis ； DOA）に基づき、高い生産性をもってJavaによるWebシステムを開発するためのアプリケーションフレームワークである[(3)]。**写真１**に開発されたソフトウェアのスクリーンショットを例示する。

楽々Framework®Ⅱでは表示画面、画面遷移、データ入出力などが部品化されており、部品を組合せてソフトウェアを開発する組立型開発を行う。部品を組合せるだけで多くの機能が実現できるため、プログラムの記述量が少なくて済み、欠陥数も少なくなる特徴がある。欠陥は組立部分とプログラム部分に作り込まれる可能性がある。

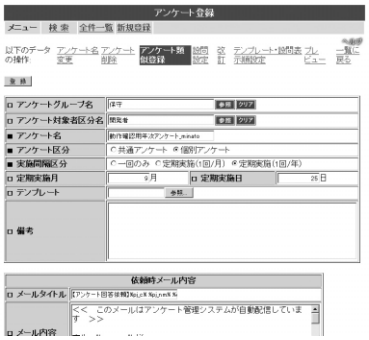


写真１　楽々Framework®Ⅱによるソフトウェア例

このような楽々Framework®Ⅱの特徴から、一般的な欠陥の作り込み防止・欠陥除去の手法に加えて、プログラムの規模の測定手法など、部品化に対応した手法を適用する必要がある。本稿では、このような楽々Framework®Ⅱ固有の背景を踏まえながら、当社の基幹系ソフトウェア開発の統計的品質管理について記述する。

3. 欠陥の作り込み防止と除去

ソフトウェアの品質を定めるものとして欠陥がある。本稿では、プログラムが設計通りの機能を果たさない不具合を欠陥とする。出荷後のソフトウェア中の欠陥が少なければ少ないほど品質が良いと言えるため、欠陥数を減らすことが必要となる。このためには

・プログラム製造工程で欠陥を作り込まないこと

・作り込まれてしまった欠陥をテスト工程で確実に除去すること

の2点を行わなければならない。プログラムが大きく複雑であるほど欠陥が多く含まれるため、規模と複雑さの観点から欠陥防止と除去を行う必要がある。当社では、これらの問題に対して、**図1**に示すようなアプローチで解決を図った。

・複雑度の制御：プログラムの複雑度の上限を定め、複雑なプログラムを作らないことにより欠陥を少なくする。
・規模指数の決定：組立型プログラミングに対応したプログラム規模の測定を行い、管理図を使ったテスト工程の管理を適切に行えるようにする。
・欠陥数の予測：IF文の数と欠陥数の相関からプログラムに含まれる潜在的な欠陥数を見積る。これにより、テストの十分性を評価可能とする。
・工程毎の欠陥数の可視化：開発の各工程で流出した欠陥数を監視し、適時の対策を可能とする。

また、これらの制御を行うための測定を行い、開発されるプログラムソースおよび開発関連文書を管理するためのツールを開発した。

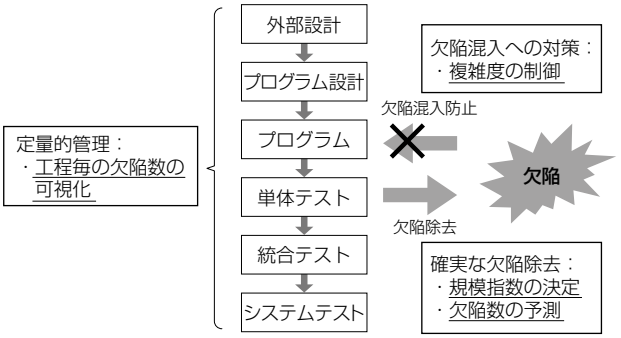


図1　欠陥の防止・除去のアプローチ

3－1　複雑度の制御　ソフトウェア開発時に欠陥を作り込まないようにするためには、ロジックを簡潔にし、複雑なプログラムを作らないことが必要となる。プログラムの複雑度を定める指標としては、Halstead複雑度、McCabeサイクロマティック複雑度、保守性指標などが提案されている(4)が、当社ではMcCabeのサイクロマティック複雑度※1を採用している。

サイクロマティック複雑度はプログラム中の分岐数を元にした複雑度の指標であり、10以下であればよい構造、30を越える場合には問題があるとされる(5)。当社では複雑度が20を越えるものは重点的なコードインスペクションを行い、プログラムの分割やロジックの修正が求められる。

McCabeのサイクロマティック複雑度の上限を定めることでプログラムの品質を管理できるのは、楽々Framework®Ⅱでのプログラミングの特徴による。**図2**に開発内容の内訳を示す。画面遷移やデータベース処理などは共通化された部品として開発されており、この部品が開発内容の80％を占めている。新たにプログラムしなければならないのは残りの20％の部分を占める業務ロジック部分となる。この業務ロジック部分では業務に必要な部分のみが記述され、それ以外でMcCabeのサイクロマティック複雑度が上昇することがない。本質的な業務ロジックの部分のみがMcCabeのサイクロマティック複雑度に反映されることが、効果的な導入につながっている。

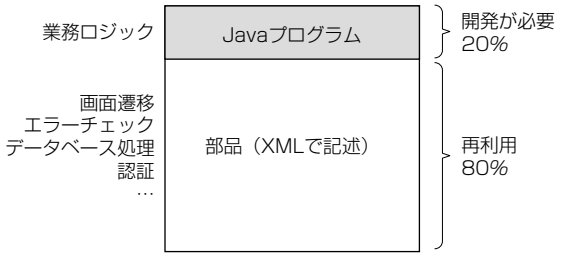


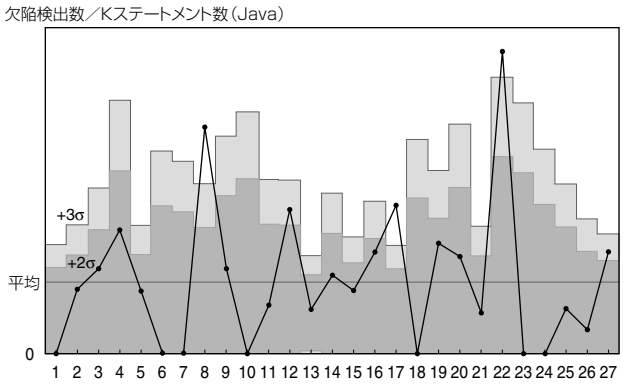
図2　楽々Framework®Ⅱにおける開発内容の内訳

McCabeのサイクロマティック複雑度は、開発環境のEclipse※2上で動作するプラグインによってリアルタイムに、かつ、自動計測される。これにより、プログラム作成時にプログラマ自身によって確認でき、簡潔なプログラム作成を意識させることができる。

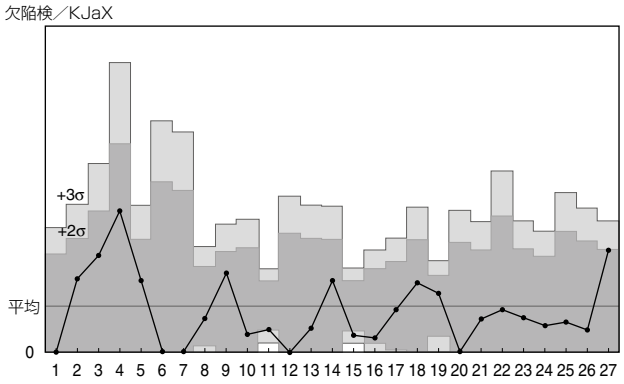
3－2　規模指数の決定と欠陥数評価　楽々Framework®Ⅱのプログラミングでは、再利用されるプログラム部品の構成をXMLにより記述し、残りの業務ロジックをJavaで実装する。楽々Framework®Ⅱではプログラム部品の再利用率が高く、Javaの実装量が少なくなることから、Javaソースコードの行数だけではプログラムの規模を決定できない。規模が決定できなければ、工数や欠陥数の定量的管理ができなくなる。例えば、1本のプログラムの欠陥数が、想定される量よりも多いのか少ないのかといった判断ができない。このため、XMLによる記述とJavaによる記述をあわせたプログラムの記述全体からプログラムの規模を測定する指数JaXを決定した。

・プログラム中の各メトリクスから、規模への寄与が高いJavaの行数、および、XMLのタグ数を選択した。
・Javaの行数とXMLのタグ数を変数とし、工数を求める直線を回帰分析により求める。
・CART（Classification and Regression Trees）によりJaXによる規模を分割し、開発者が利用しやすい閾値を定める。

規模指数にJaXを用いることで、管理図上の規模を原因とするばらつきを抑えることができるため、欠陥のあるプログラムの監視が行いやすくなる。**図3（a）**に、横軸に各



（a）Javaステートメント数に対する欠陥数



（b）Javaステートメント数に対する欠陥数とJaXに対する欠陥数の対比

図３　Javaステートメント数に対する欠陥数とJaXに対する欠陥数の対比

プログラムを取り、Javaソースコードの1,000行数あたりの欠陥数をプロットしたu管理図を示す。同様に、**図３（b）**に1,000JaXあたりの欠陥数をプロットしたu管理図を示す。

図３（a）のようにJavaステートメント数で管理図をプロットした場合、Javaの記述量が少ないために規模が正しく測定されず、正常なプログラムが管理限界を外れたものとして検出されてしまう。これに対して、**図３（b）**のようにJaXを用いると規模が正しく測定され、適切な管理が行える。

３－３　IF文の数による欠陥数予測　ソフトウェアの品質を高めるためには、プログラム開発時にプログラマが欠陥を作り込まないことに加えて、テスト担当者がテストを確実に行い、欠陥を除去することが必要となる。しかしながら、現実的には除去による欠陥の数が少なくなるにつれて、新たな欠陥を見つけるためのコストや工期が大きくなる。コストと品質を最適にするためには、どこまで欠陥を除去するのかを決定しなければならないため、プログラム中にどれだけの欠陥が含まれているかを予測する必要がある。当社の基幹系ソフトウェアでは、プログラム中のIF文の数がプログラムに含まれる欠陥数と相関があることが回帰分析により得られた。**図４**にIF文と欠陥数の相関の例を示す。

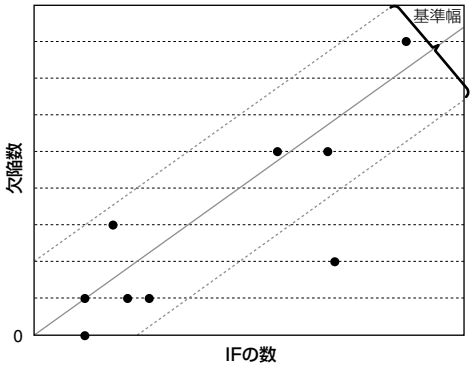


図４　プログラム中のIF文の数と除去欠陥数の相関

図４は横軸をIF文の数、縦軸を欠陥数にとり、各プログラムをプロットしている。図中の実線は回帰分析により得られた直線となり、点線が基準幅となる。テストによって摘出された欠陥数がこの基準幅に収まっていることがテスト完了の基準として用いられる。点が基準幅より右下に含まれていれば、IF文の数に対して摘出された欠陥数が少ないことを表し、もともとのプログラムが高品質だった、あるいは、テスト不足が疑われる。この時、テストとは別にプログラムとテストの分析が行われ、別プロジェクトから再利用されたプログラムであるなど、問題が無いことが確認されればテスト完了となり、確認されなければ再テストが実施される。一方、点が基準幅よりも左上に含まれていれば、過剰に欠陥が摘出されていることを表し、プログラムの品質に問題がある。この時、プログラムを作成したプログラマに問題点の対応についての指導が行われ、欠陥の再発を防止する。

３－４　工程毎の欠陥数の可視化　各工程ではu管理図などを用いることで欠陥数の監視を行うことができる。一方、開発プロジェクトを通して欠陥数を監視する場合、工程毎の欠陥数の移り変りを見ることが必要となってくる。当社では、各プロジェクトで工程ごとの欠陥の増減を工程を通してプロットする欠陥推移図と呼ばれるグラフを作成し、欠陥数の監視を行っている。**図５**に欠陥推移図を示す。

図５の上向きの矢印が欠陥が作り込まれた数、下向きの矢印が欠陥が取り除かれた数となっている。外部設計、基本設計、詳細設計の工程では、設計とレビューが対になって行われ、その工程での欠陥除去が行われる。PG開発ではまずコードインスペクション（CI）によって欠陥が取り除かれる。その後、単体テストで外部仕様書、プログラム仕様書との整合性が精査される。最後に各プログラムを統合した統合テストを行い、プログラム間の欠陥を除去する。

欠陥推移図により、ある工程の除去欠陥数が少なければ、次工程での目標除去欠陥数を増加させるなどの判断が可能と

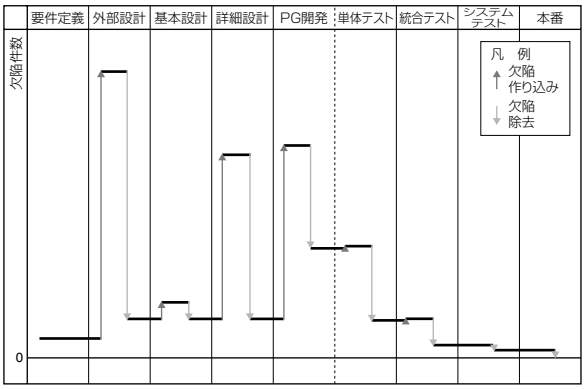


図５　欠陥推移図

なる。また、一般に開発プロジェクトチームの成熟度が高ければ下流工程での欠陥数が少なくなると言われるが、欠陥推移図にプロットした場合、ピーク位置が上流工程に移動することを意味しており、欠陥数の大小だけでなく欠陥推移図の形状自身も開発チームの品質管理能力の目安となる。

4.　管理ツールの開発

統計的管理を全仕様書、全プログラムに対して確実に行うためには、リビジョン管理などの構成管理、測定を行うツールが必要となる。当社情報システム部では、isdocと呼ばれるシステムを開発し、開発文書の構成管理と統計的管理を実現している。**写真２**にスクリーンショットを示す。

写真２　構成管理・統計的管理ツールisdoc

isdocは以下の機能を持つ。

・仕様書やプログラムソースの版管理や、各文書間の紐付けなどの構成管理機能

・仕様書やプログラムに代表される成果物の行数や工数、欠陥数などの各種測定、データ収集を行う品質測定機能

・u管理図などの各種グラフをプロットするグラフ機能

これらの機能は一般に個別のソフトウェアとして販売されていることが多く、開発プロセスにあわせて連携・カスタマイズさせることが困難である。上記の機能を持つ統一ツールを自社開発することで、楽々Framework®Ⅱを用いる当社の開発プロセスにあわせた構成管理、品質管理を可能としている。

5.　評　　価

本稿で述べた定量的管理によって、欠陥の作り込みの防止や欠陥の確実な除去が行われると、単体テスト以後に流出する欠陥数が減少する。

図６に本稿の取り組みを実際に基幹系ソフトウェア開発プロジェクトに適用する前後での統合テスト時に摘出される欠陥数の推移を示す。横軸には各プロジェクトを完了日順に並べ、縦軸には統合テストで摘出された欠陥数をプロジェクト全体の全てのプログラムをあわせた機能数（1,000ファンクションポイント単位）で割った値を示す。

図の中程を境として、摘出欠陥数がおよそ半分となっている。これは、統合テストより前工程のプログラム開発や単体テストで、欠陥の作り込みの防止や欠陥の確実な除去が行われた結果であると考えられる。また、グラフの上下のばらつきも減少しており、欠陥防止、および、欠陥除去のプロセスが安定していると見られる。

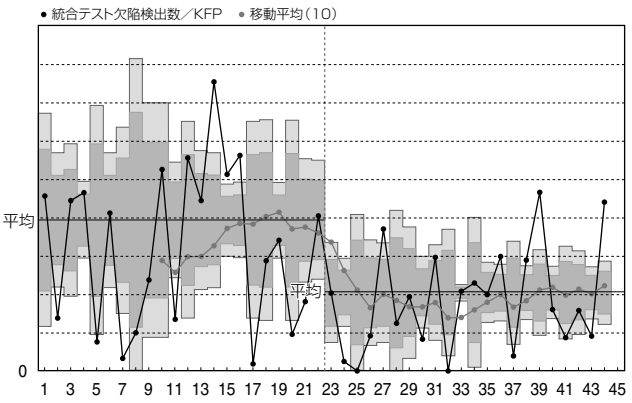


図６　統合テストの摘出欠陥数の推移

6. 結　言

本稿では、当社の基幹系ソフトウェア開発における統計的品質管理の取り組みについて記述した。高品質なソフトウェアを開発するためには、欠陥を作り込まない、また、作り込んだ欠陥を確実に除去することが必要であり、そのためには統計的な数値の制御が必要となる。当社では、McCabeのサイクロマティック複雑度などの制御、欠陥数の予測、規模指数の決定、工程間の欠陥数の可視化などにより、基幹系ソフトウェア開発の品質の見える化と統計的制御に取り組んできた。この取り組みは基幹系ソフトウェア開発の開発標準として導入され、単体テスト以後に流出する欠陥数を半減することができた。

今後は、楽々Framework®Ⅱの開発に対応した各メトリクスを用いた見積モデルの構築や、プロジェクトメンバが各人の作業の品質測定を行い改善していくPSP(6)の導入などに取り組み、さらなる品質改善を目指す。

用 語 集

※1　McCabeのサイクロマティック複雑度

プログラム中で通りうる処理の流れの数を合計したもの。プログラム中で行っている処理の複雑さを表しており、大きければ大きいほど、テストや保守が困難になる。

※2　Eclipse

ソフトウェアの開発に必要な複数のツールをまとめ、同一のインタフェースで開発を行えるようにした統合開発環境の一つ。ソースコードが公開され、規約を守った上で無償で利用することのできるオープンソースとして公開されている。

参 考 文 献

(1) 日本情報システム・ユーザー協会、「ユーザー企業 ソフトウェアメトリックス調査報告書 2008年版」(2008)
(2) Carnegie Mellon University：“Software Engineering Institute. CMMI”, http://www.sei.cmu.edu/cmmi/index.html
(3) 池田和壽、「DOAによるモデル駆動型のシステム開発－楽々FrameworkⅡの開発－」、SEIテクニカルレビュー、住友電気工業株式会社、no. 165, p. 33-37 (2004)
(4) Spinellis, Diomidis, Code Quality.、鵜飼文敏、後藤正徳、平林俊一、まつもとゆきひろ監修、トップスタジオ訳、毎日コミュニケーションズ (2007)
(5) Jones, Capers「ソフトウェア開発の定量化手法」、鶴保征城、富野寿訳、第2版、共立出版 (1998)
(6) Humphrey, Watts S.、「PSPガイドブック」、秋山義博監訳、JASPIC TSP研究会訳、飛翔社 (2007)

執　筆　者

中塚　康介*：情報システム部　博士（情報学）
ソフトウェアエンジニアリングの推進、新規開発技術の導入に従事

中村　伸裕：情報システム部　グループ長

＊主執筆者