Vol.89 No.03 298-299

Professional Report

ソフトウェア開発への統計的プロセス制御の適用

Application of Statistical Process Control to Software Development

小室　睦　Mutsumi Komuro

ソフトウェア開発では，(1)創造的で人への依存が大きい，(2)複数の共通原因が混在しうる，(3)同質で大量のデータを得るのが難しいといった適用への課題がある。これらの課題を克服するには，成果物中心からプロセス中心の考え方に切り替えること，また，改善活動では実施者の心理も考慮し，自主・自律的な改善を実現していくことが重要となる。

ここでは，統計的プロセス制御のソフトウェア開発への適用について，日立ソフトウェアエンジニアリング株式会社での経験に基づいて述べる。統計的プロセス制御がソフトウェア開発においても有用であることを，ピアレビューに対する統計的プロセス制御による適用事例を通して示すとともに，特に品質向上および原価低減に大きな効果を持つことを示す。

小室　睦
1985年日立ソフトウェアエンジニアリング株式会社入社
技術開発本部 プロセス改善技術センタ所属
現在，同社の全社プロセス改善に従事
ACM会員，IEEE会員
日本ソフトウェア科学会会員
情報処理学会会員
プロジェクトマネジメント学会会員
日本数学会会員

1　はじめに

わが国では1950年代以降，QC(Quality Control：品質管理)活動が製造業で大きな成功を収め，日本製品の品質向上と競争力強化に大きく貢献した[1), 2)]。しかし，ここで使われている統計的プロセス制御の手法をソフトウェア開発に適用することは，現在のところあまり広く行われていないように思われる。

ソフトウェア開発に対するプロセス改善のモデルには，ソフトウェアCMM(Capability Maturity Model：能力成熟度モデル)※)，およびその発展であるCMMI(Capability Maturity Model Integration：能力成熟度モデル統合)※)があり，そこでは統計的プロセス制御の手法は，レベル4，5といった，いわゆる高成熟度レベルに位置づけられている[3)]。このことは，統計的プロセス制御手法の重要性を示すと同時に，ソフトウェア開発に適用する際の難しさをも表していると考えられる。例えば，CMMI(v.1.2)にはフルスペックで22個のプロセス領域があるが，段階表現でレベル4，5に分類されるのはそのうちわずか四つに過ぎず，残りの18個は高成熟度レベルに到達するための基礎固めや準備にあたると見られる。

ソフトウェアCMMやCMMIに基づくプロセス改善運動では，SEPG(Software Engineering Process Group)※)あるいはEPG(Engineering Process Group)※)と呼ばれる改善の取りまとめ役が組織としてのプロセス改善を展開していく仕組みを想定している[4)]。このような体制をとる場合，組織側からの押しつけを避け，開発現場の要望や実態に基づき，開発現場が益を実感できる改善策を展開していくことが大切である。統計的プロセス制御の実現においては，この観点がとりわけ重要となる。

ここでは，日立ソフトウェアエンジニアリング株式会社(以下，日立ソフトと言う。)における統計的プロセス制御手法のソフトウェア開発への適用，特に，ピアレビュープロセスへの適用と改善による品質向上や原価低減への効果，ソフトウェア開発プロセスの特徴とその統計的プロセス制御への影響，および実際の適用から得られた教訓について述べる。

2　プロセス改善の背景

日立ソフトは，ソフトウェアシステムの開発，システムインテグレーションを中心としたサービス提供を行っているソフトウェア企業である。創業以来，9次にわたる品質向上運動，11次にわたる生産性向上運動を中心に改善運動を実施しており，高品質のシステム構築能力を強みとして業績を伸ばしてきた。

一方，冷戦の終了を契機として始まった価格破壊とグロー

バル化の波は，日本のソフトウェア産業にも押し寄せてきており，品質に関する強みを保持しながら，飛躍的な生産性向上を実現することがビジネス上の大きな課題となっている。そのためには，これまでの運動に加えて，世界にも通用する新たな視点を採用することが重要と考えた。そこで，プロセス改善におけるデファクトスタンダードであるCMMIに基づいたプロセス改善運動を，全社規模で，2001年から推進している。

このプロセス改善運動を展開するにあたって以下の方針を設定した。

(1) 全社の組織的強化
(2) 各組織(各事業部，あるいはその下の本部)の実態を反映した改善の実施
(3) 迅速な改善の実現
(4) 改善体制の整備と人材育成

これらの方針の実現方法についてはすでに発表済み[5), 6), 7)]なので，ここでは詳説を避けるが，(1)と(4)を実現するため，全社の各事業部にSEPGを設け，各事業部の実態に応じた改善活動を実施するようにした。また，(2)と(3)を実現するためには，各組織の現状をなるべく正確かつ客観的に把握することが肝要と考え，改善の初期にCMMIの正式アプレイザル(成熟度判定)であるSCAMPIアプレイザルを実施し，ここで指摘された改善点を中心に改善活動を展開した。これらの施策は有効に機能し，全社すべての事業部でレベル3を達成した。また，レベル3までの主要な改善事項であったプロセスに対する品質保証活動が，統計的に有意な品質向上効果を持つことを確かめた[8)]（**表1**参照)。

しかし，レベル3はプロセス改善のための組織的な仕組みが整い，その後の改善のための基礎ができたという状態であり，ビジネス的な観点から見ても十分に満足できるだけの効果を生むとは限らない。主要事業部の一つである産業システム事業部では，レベル3達成に至る以前から，事業部長の主導の下，「コードインスペクション運動」を展開していた。これはソースコードに対するピアレビューを徹底しようという運動である。同事業部は産業系の組込みシステムの構築などを手がけており，事業的に今後拡大が期待される一方，比較的新しい分野であるため，プロセスを徹底することで品質向上，原価低減できる余地がまだかなりあると考えられていたことが背景としてある。

表1　日立ソフトのCMMIレベル達成組織一覧
全社で改善に取り組み，全事業部でCMMIレベル3以上を達成している。

事業部	本部	レベル	達成時期
公共社会システム事業部	公共システム本部	レベル3	2002年1月
金融システム事業部	——	レベル3	2002年6月
開発事業部	開発本部	レベル3	2002年7月
	ネットワークシステム本部	レベル3	2003年12月
産業システム事業部	——	レベル3 レベル4 レベル5	2002年10月 2004年2月 2004年10月

このコードインスペクション運動では，対象言語に関する知識やレビュー経験に基づいてレビューアの認定制度を設け，ソースコードのピアレビューを行う際には，認定されたレビューアの参加が事業部内規で義務づけられている。

同事業部におけるピアレビューの改善を中心に，統計的プロセス制御の適用経験について以下に述べる。

3　統計的プロセス制御とは

3.1　SQC，TQMと統計的プロセス制御

統計的プロセス制御とは統計的手法を用いたプロセス管理手法であり，プロセスの定義，測定，制御，そして改善を含んでいる[9)]。日本では統計的プロセス制御の技法はSQC(Statistical Quality Control：統計的品質管理)の名で知られ，その活動はTQM(Total Quality Management：総合的品質管理)として体系化され[1), 2)]，製造業を中心に広く普及している。TQMでよく利用されるツールは「七つ道具」と呼ばれている。そのうちでも，最も特徴的なツールが「管理図」である。「品質管理は管理図に始まり，管理図に終わる。」とも言われる。統計的プロセス制御の典型的な活動は次のようなものである。

(1) 管理限界や管理図に関する変動の判定ルールなどを用いて，管理されていない状態を判定する。
(2) 管理されていない状態に対しては，それを引き起こした特殊原因を見極める。
(3) 特殊原因を取り除いて，プロセスを安定させる。

3.2　管理図と統計分析

管理図とは，通常，時系列にデータをプロットした折れ線グラフ(ランチャート)で，管理限界を示す横線が入ったものである。プロセス実績を管理図にプロットしていき，管理限界と比較することにより，プロセスの実施状況を随時，視覚的にチェックして制御することができる。データの群分けやデータ分布に対する仮定などによって，使用する管理図が異なってくる。

XmR管理図は隣り合う二つのデータを一つの群とする群分けに基づく管理図で，各群の範囲の変動を表すmR(moving Range：移動範囲)図と個別データの変動を表すX図の二つのグラフから成る。管理限界はmR図のデータを基に統計的手法で決定する。Z管理図はポアソン分布を仮定した管理図であり，ソフトウェア開発では作業成果物の欠陥密度(レビューでの指摘密度，テストでのバグ密度など)の

管理などによく用いられる。管理限界はポアソン分布の統計的性質を利用して算出する。X-bar-S管理図は対象データを幾つかの群に分け，各群のそれぞれの標準偏差を計算し，これらのばらつき具合を表すS図と，各群の平均値のばらつき具合を表すX-bar図の二つから成る。管理限界はS図のデータを基に統計的手法によって決定する。X-bar-S管理図は群分けされたデータが多数入手できるときに有用な管理図である。群と群の間に統計的に有意な差異があるかどうかを視覚的に示すことができるので，改善効果の確認にも用いることができる。

管理図では管理限界の設定が重要であるが，これは群ごとの統計量を用いて推定した母標準偏差をσとして，平均値から±3σの位置に設定される。管理図を用いたプロセス実績のチェックとしては，この±3σの管理限界を超すデータがあるかどうかを見るのが基本である。プロセスが安定して実施されていれば，±3σを超すようなデータが出現する確率はきわめて小さいので，もしこのようなデータが観測されれば何かプロセス上の特殊原因によって引き起こされたものと考え，その原因を探る。日立ソフトでは，これを含めて以下のテスト1～4をチェックしている。正規分布を仮定すると，テスト2～4もテスト1と同程度に小さな出現確率しか持たないことが，これらのチェックを行う根拠である。

テスト1：±3σの管理限界を超えるデータがある。
テスト2：連続した三つのデータのうち2個以上が中心線（平均値）から見て，同じ側にあり，隔たりが2σを超えている。
テスト3：連続した五つのデータのうち4個以上が，中心線から見て同じ側にあり，隔たりが1σを超えている。
テスト4：連続した九つのデータが，中心線から見て同じ側にある。

ただし，X-bar-S管理図のS図で各群のサイズが小さい場合，およびXmR管理図のmR図ではテスト2～4は適用しない。これは，これらの図にプロットされるデータの分布が正規分布からかけ離れており，確率計算の前提を満足しないからである。なお，テスト1～4は，Florac，Carleton[9]の記述を主に参考にして定めたが，テスト4の条件は，JIS規格[10]に合わせて「連続した8点」から「連続した9点」に変更して用いている。

管理図は本来プロセス制御のために考案されたツールであるが，幾つかの留意点を守って適切に使用すれば統計分析にも有効に用いることができる[9]。留意すべき点は次の2点にまとめられる。

（1）合理的な群分けの原則を順守する。
（2）プロセスごとのパフォーマンスを評価し，必要ならプロセスの分離を実施する。

群分けは管理図の根底にある基本的な考え方であり，採用プロセス，環境など実施状況が類似したデータを同じ群として群分けすることにより，標準偏差の推定値σの算出に対する特殊原因の影響を最小化しようという手法である。ソフトウェア開発プロセスでは人的要因による変動が入りやすいので，この原則を守ることは特に重要となる。

また，プロセスが異なればパフォーマンスも異なってくるので，異なるプロセスに由来するデータは区別して扱うべきだという当然のことである。層別の考え方[1]の特別な場合と言ってもよい。統計処理では多数のデータが必要だと妄信して，複数のプロセスを一緒に扱って，結局無意味な結果しか出てこないというケースを時折見かける。実際には安定したデータの分布を得ることのほうが，むやみにデータ数を増やすことよりずっと重要で価値がある[9]。

4 ソフトウェア開発プロセスへの統計的プロセス制御の適用

4.1 適用の難しさ

進捗（ちょく）管理，ピアレビュー，テストプロセスなど，幾つかのプロセスに対しては，統計的プロセス制御がソフトウェア開発においても有効であるという報告がすでにある[11), 12), 13)]。しかし，統計的プロセス制御がソフトウェア産業全般で広く認められ，活用されているという状況には，まだ程遠いように思われる。これは一つには，統計的プロセス制御が伝統的に製造業で用いられ，そこで大きな成功を収めたため，統計的プロセス制御は製造業のプロセスのような成果物中心のプロセスにのみ有効であるといった先入観があったのではないかと推測される。

日立ソフトでもTQMから派生した小集団活動や考案改善は実施されてきたが，その中で管理図の使用法が伝えられたことはないように思われる。また，テスト工程のように，成果物が明確に見えてきた段階では，バグ率やバグ死滅曲線などを用いた定量的管理の手法が確立されているが，より上流で人の関与する割合が高いプロセスに対する定量的管理手法は必ずしも確立されているとは言えない。

一般に，ソフトウェア開発は人や環境に依存する部分が大きく，よく言えば知的で創造的，悪く言えば労働集約的である。このため，成果物中心よりもプロセス中心のアプローチが重要となるが，プロセスを安定化させるのはいっそう難しい。例えば，Kanはソフトウェア開発への統計的プロセス制御の直接的な適用は，プロセスが複雑で高いレベルの創造性と精神活動を伴うため困難であると述べている[14]。

4.2 ソフトウェア開発プロセスの特徴

統計的プロセス制御をソフトウェアプロセスに適用する際

の困難は，以下に示すソフトウェア開発プロセスの特徴に由来するものと考えられる。

(1) 創造的で人への依存が大きいため，ソフトウェア開発プロセスは，大きなばらつきを持ち，不安定となりやすい。

(2) ツール，開発技法，開発ソフトウェアの種類，開発環境などのさまざまな要因がプロセス実績に影響を及ぼしうる。

(3) 同質で大量のデータを得ることが困難である。製造業では日々多くの製品や部品がラインから生産され，一度に多くの実績データが得られる。これに対して，ソフトウェア開発で一度に生産される成果物の量は限られており，さらに，同じ工程の成果物であっても顧客やプロジェクトに応じて特有な部分があり，同質でないことが多い。

しかし，統計的プロセス制御はソフトウェア開発においても重要かつ有用である。統計的プロセス制御がプロセスを安定化し改善するための指針を与えること，改善効果を実証するための基礎となることは後述する。

このような困難を乗り越える鍵は，従来以上にプロセスに力点を置いた活動をすることにあるように思われる。例えば，成果物の実績データに加えて，プロセス実績のデータを扱うことで利用可能なデータの量を増やすことができる。Florac, Carleton[9]が提唱しているようにプロセスの定義の中に人，ツール，環境などの要因も含ませて考えることにするなら，統計的プロセス制御は人の活動を安定化させることにも使えるし，複数の共通原因を特定し，分離する助けにもなる。

5　日立ソフトにおける適用経験

統計的プロセス制御をソフトウェア開発プロセスにどう適用するかというやり方にも，前述した困難は影響する。ここではこの点も含めて，日立ソフトにおける適用経験について述べる。

統計的プロセス制御は有用な技法であるが，プロセスに対して，その実績を定量化して分析・制御・改善を実施していくわけであるから，相応のオーバーヘッドが伴う。すべてのプロセスに対し統計的プロセス制御を実施するわけにはいかない。十分な効果が期待されるプロセスを選択して実施すべきである。この選択には当然，事業的な戦略や目標が反映されなければならない。

5.1 適用プロセスの選択と事業的効果

ソフトウェア開発プロセスの中で，統計的プロセス制御の適用対象としてまず考えられるのはテストプロセスであろう。製造業での適用との類似を追及する観点からは，これが自然な選択であるように思われる。

日立ソフトでは，テストプロセスにすでに幾つかの尺度と定量的管理手法を導入していた。代表的な尺度として，バグ率と前倒し摘出率，定量的管理手法としてバグ死滅曲線の利用が挙げられる。バグ率はテスト工程における欠陥密度のことで，キロステップ当たりに摘出したバグの件数を全社で測定している。前倒し摘出率はバグの総件数のうち，単体デバッグ工程以前に摘出したものの割合のことである。バグ死滅曲線とはバグの累積件数と未消化のPCL（Printer Control Language）件数を一つのグラフにプロットしてテスト工程の進み具合と品質状況を見るものである。

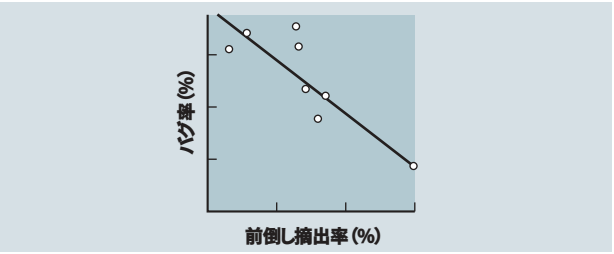

図1　前倒し摘出率とバグ率の相関
欠陥を上流で摘出する割合が高いほど品質も高くなる。

このようにテストプロセスは定量化手法がある程度確立されており，さらに品質保証部門による強いサポートも実施されているため，統計的プロセス制御を導入しても，改善を実施できる余地は比較的少ないように思われた。また，品質向上のためには上流工程での品質作り込みが重要であることがTQMの教えでもあり，下流のテストプロセスのみを改善しても得られる事業上の効果は限定されていると見られる。

改善の初期の段階で，統計的プロセス制御の適用対象検討の一環として，前倒し摘出率とバグ率の相関を調べた。その結果得られた散布図と回帰直線を**図1**に示す。

この図からわかるように，前倒し摘出率が高くなるとバグ率も下がるという負の相関がある。これは，より上流で欠陥を除くほうが，より大きな品質向上効果があることを示している。前倒し摘出率の算出には単体テストで除いたバグのほか，プログラム作成者自身が行う机上レビューやプロジェクト内部で他の技術者とレビューするいわゆる「内部レビュー」で摘出された欠陥の数も一部計上されている。品質向上にはこれらのレビューの貢献が重要と考えられる。

日立ソフトで高成熟度レベルの改善を本格的に検討しだしたのは，主要事業部がレベル3を達成した2002年以降であるが，この時点で産業システム事業部ではコードインスペクション運動，すなわちソースコードに対するピアレビューを徹底する活動を開始しており，ピアレビューに関する実績データがかなり蓄積されていた。しかし，ピアレビューに対する定量的な管理はまだなされておらず，改善の余地がかなりあるものと考えられた。また，より上流に位置し，品質および生産性への寄与も大きいと期待された。以上のような考察から，ピアレビュープロセスを統計的プロセス制御の適用

対象として選択した。

5.2 ピアレビュープロセス，指標と管理図

ピアレビューとは欠陥摘出と改善点の特定を目的に実施する作業成果物に対するレビューのことである[15), 16), 17)]。ピア（peer）とは作者と対等な同僚のことを指し，管理者は出席しないのが普通とされる。代表的なレビュー方法としてはフェイガンインスペクション[18)]やウォークスルーがある。

管理者が原則として出席しないのは，ピアレビューの実績データを人の評価に使うような誤用を避けるためである。作者はその作業成果物に関して最も詳しい技術者であり，作者が自ら進んで欠陥の早期摘出に努めることで，大きなレビュー効果が得られる。レビューの場を作者が欠陥を指摘しづらくなるような雰囲気にすることは，ピアレビューにおいては厳に慎むべきことである。もちろん，レビューの中には管理者が主催し，管理的視点からチェックをかけるものもあり，そういったレビューにはまた別の意義がある。ピアレビューはこのような管理者レビューとは区別すべきものだということである。

日立ソフトの場合，仕様書や設計書を含む各種ドキュメントとソースコードに対するピアレビューが実施されている。この中にはすでに述べた内部レビューや机上レビューも含まれる。レビューが品質向上，また，手戻り防止による原価低減に効果のあることは一般的によく知られていたが，上述のようなピアレビューと管理者レビューとの区別は，あまり明確には意識されていなかった。また，ピアレビュー実施によって具体的にどの程度の効果が上がるのか，定量的に把握することはできていなかった。

現在，日立ソフトで用いているピアレビューに対する主な指標とその定義を**表2**に示す。レビュー速度は単位時間当たりにどれだけの規模の成果物をレビューしたかにより，レビューを進める速さを表す。指摘密度はレビューに関する欠陥密度であり，単位規模当たり何件の指摘があったかで定義される。レビュー前倒し摘出率は前述の前倒し摘出率のレビュー版で，ソースコードに対して摘出された総欠陥数に対して，レビュー指摘で摘出した欠陥数の割合で定義される。レビュー効率は工数当たりの指摘件数を算出したもので，レビューがどの程度の効率で実施されているかを示している。

表2　ピアレビューに対する指標
レビュー速度と指摘密度は，プロジェクトがピアレビュープロセスを制御するのに用いる。レビュー前倒し摘出率とレビュー効率は，組織レベルでの目標値設定やベストプラクティスの発掘に使う。

指標名	定義式
レビュー速度	（レビュー対象規模）/（レビュー時間）
指摘密度	（指摘件数）/（レビュー対象規模）
レビュー前倒し摘出率	（ソースコードレビューでの指摘数合計）/（レビューおよびテストで摘出した総欠陥数）
レビュー効率	（指摘件数）/（レビューにかけた工数）

レビュー速度はXmR管理図，指摘密度はZ管理図を用いて，毎回のレビュー実績を監視・制御している。レビュー速度が速くなればなるほど，欠陥の見落としが増えることが知られているので，レビュー速度があまり大きな値にならないよう，つまりじっくりレビューするように毎回のレビューを管理する。指摘密度は品質指標であり，レビュープロセスが安定しているという前提の下では，この値が小さければ品質が良いと判断される。レビュープロセスが不安定となるケースでよくあるのは，急ぎすぎて見落としが増える場合である。そこで，レビュー速度と併用して判断するのが有効である。このほかにも，レビューのやり方に何か違いがなかったか，準備状況や環境に変化がなかったか，レビュー対象物の難易度はどうかなどを総合的に見て判断する必要がある。

レビュー前倒し摘出率は各プロジェクト，あるいは組織のピアレビュー活動の進展の具合を測る指標で，組織およびプロジェクトでこの指標の目標値を決めて管理している。この率が高いほど，テストからレビューに活動の力点が移ってきている度合いが大きいといえる。ハンフリーの提唱しているPSP（Personal Software Process）では同様の指標が「yield」という名で用いられており，yieldの上昇とともに品質が飛躍的に向上し，テスト工程の工数を大幅に削減できることが報告されている[20)]。

レビュー効率はピアレビューのやり方が効率的かどうかを判断するのに用いることができる。ただし，効率を最初から強調しすぎると，改善が間違った方向に向かう危険があるので注意が必要である。実際，品質向上を考えずにレビューにかかる工数だけを気にするのなら，レビューなどしないほうがよいという結論になってしまう。

指標の使用目的から見て，これらの四つの指標は2種類に大別できる。レビュー速度と指摘密度の二つはプロジェクトレベルで毎回のレビュー実績を監視・制御し，品質作り込みを実現するためのものであり，管理図を用いてプロセス制御を行っている。他の二つの指標は毎回の監視というよりは，ある程度データがそろった時点で，改善の進展度合いを確認したり，次の改善のための教訓を得るためのものである。

また，レビュー速度と指摘密度の二つは「制御のための指標」，他の二つは「分析のための指標」と呼ぶこともできる。制御のための指標はプロジェクトレベルでプロセス制御に利用されるが，分析のための指標は組織レベルでの利用が基本である。レビュー効率の分析はXmR管理図で実施するが，この場合の管理図の利用は制御というより，分析のためである。

ソフトウェア開発プロセスでは人の関与が大きく，指標を定めるとそれを最適化することをノルマのように感じる心理

的効果が生じるので，このような使い分けは明確にしておいたほうがよい。レビュー効率は上手に使うと，よいレビュー方法の特定ができ有用な指標であるが，上述のような誤解も生じやすい。このため，組織レベルの分析でのみ用いる副指標という扱いにして，プロジェクトレベルではこの指標をあまり意識させないようにしている。

「制御のための指標」と「分析のための指標」の区別を述べたが，これと同様の注意が管理図のチェックルールにも当てはまる。チェックルールとしてテスト1〜4を前述したが，JISの管理図の規格ではこれらを含む八つのルールを定めている。また，管理図を使い慣れてくると，ルールには抵触せずとも，何か変だと異常を感じとれる場合もある。

管理図によるチェックは一種の仮説検定であり，シューハートが管理限界を3σという大きな値に設定したのは，第一種の誤り，すなわち偽の警告を少なくしたかったためであると言われている。しかし，3σによる限界チェックのみだと，今度は第二種の誤りが多くなる，すなわち，異常の検出能力が小さいことになる。そこで，テスト1〜4を含む幾つかのルールが提案されてきた。日立ソフトではチェックルールを少なく抑える方針をとり，組織のルールとして定まっているテスト1〜4だけとしている。ただし，管理図を使うプロジェクトが自主的にチェックルールを付け加えたり，違和感のある部分を調べてレビュープロセスを改善していくことは禁じておらず，むしろ奨励している。これは組織レベルのルールとして定めると，指摘されたときに怒られているような心理状態に陥りがちなので，ルール数を増やしたくなかったこと，および，異常の指摘とその原因分析からプロセスに対する改善点が浮き彫りになることから，その可能性をできるだけ明白にしたかったからである。指標の分類との類似で言えば，組織レベルのルールであるテスト1〜4は「制御のためのルール」，プロジェクトが自主的に付け加えるチェックは「改善のためのルール」であると言うことができる。

5.3 プロセス制御の例 —— 個別データの扱い

管理図の基本的な使い方は，変動に対する特殊原因を見つけ，プロセスを安定化することである。図2はドキュメントレビューに対するレビュー速度をプロットしたXmR管理図である。一つのプロジェクトで実施した同一ドキュメントに対する複数のレビューデータを時系列に並べている。丸で囲んだ部分で，テスト1に抵触する異常が1件見つかっている。

図3は同じレビューデータに対する指摘密度をプロットしたZ管理図である。図で丸をつけたのと同じレビューの指摘密度の点を丸で囲んである。この値自体は異常値ではないが，前後のレビューに比べて指摘密度は落ちている。二つの図を併用すると，丸をつけたレビューは急ぎすぎて指摘が

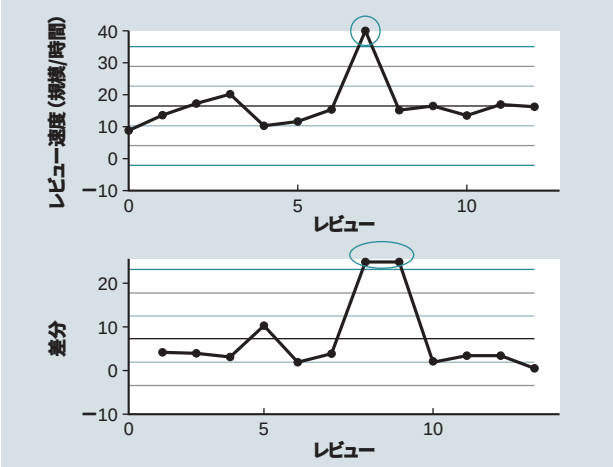

図2　レビュー速度に対するXmR 管理図
丸をつけたレビューデータはレビュー速度が速すぎる異常値である。

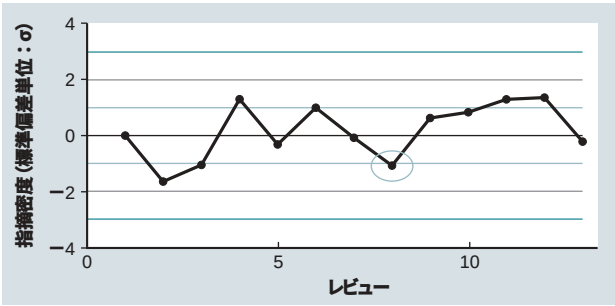

図3　指摘密度に対するZ 管理図
レビュー速度が速すぎたレビュー（丸印）では指摘密度が前後に比べて低い。

相対的に少なくなっており，欠陥の見落としが多発している可能性がある。レビュー記録に戻って調べてみると，このレビューでは他のレビューの平均の約3倍の規模の成果物を一度のレビューにかけようとしていたことがわかった。

もちろん，より正確な判断をするためにはこのレビューでどんな成果物をレビューしたのか，レビュー方法はどんなやり方をとったのか，参加者の構成に特別なことはなかったのかなど，レビュープロセスに関するさまざまな情報を検討する必要がある。しかし，この実績値から見る限り，1回のレビュー対象規模をもう少し小さくして，何回かのレビューに分割して再レビューすることが求められる。

このような監視・制御を逐次実施していき，必要に応じて是正措置をとることで，ピアレビュープロセスを安定化していくことができる。レビュープロセスは人への依存が大きいプロセスであるため，各回のレビューごとに特有な部分があっても不思議ではない。各回のデータを個別に調べていき，是正措置を早めにとることが重要である。特殊原因を一つ一つ解決していくことで，プロセスを安定化させ，レビュー実績のベースラインを確立することができる。

6　ピアレビュープロセスの改善

日立ソフトで実施したピアレビュープロセスの改善につい

て述べる。

6.1 効率分析 —— 複数の共通原因への対処

高成熟度に向けた改善活動の初期に実施した組織レベルの分析を例として述べる。初期段階ではレビュー記録のフォーマットに不備があり，レビュー規模が不明な場合があった。このため，レビュー速度や指摘密度の分析は行えなかった。そこで，次善の策として，レビュー規模がなくても算出できるレビュー効率に関して分析を行い，異なるレビュー方法を特定した。

この分析ではXmR管理図を用いた。XmRでは隣り合うデータが一つの群として扱われることから，分析する観点から見て類似のデータが隣り合っていることが肝要となる。通常のプロジェクトレベルの制御では，時系列にデータを順に入力していく。これは実施時期が近いデータは類似性が高いと仮定していることになる。この分析では複数のプロジェクトのデータを扱ったため，まずプロジェクトごとにデータを整理し，各プロジェクト内で時系列に並べた。また，プロジェクトの並びも同じ部に属する業務が類似したプロジェクトを隣り合わせるようにした。

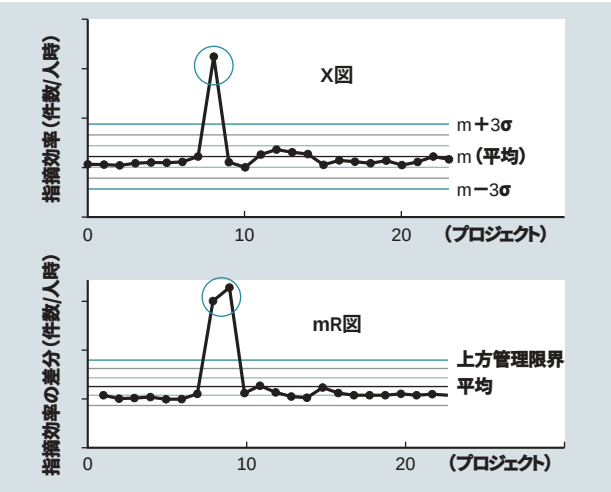

図4　レビュー効率の最初のプロット結果
丸をつけたデータは著しい異常値である。

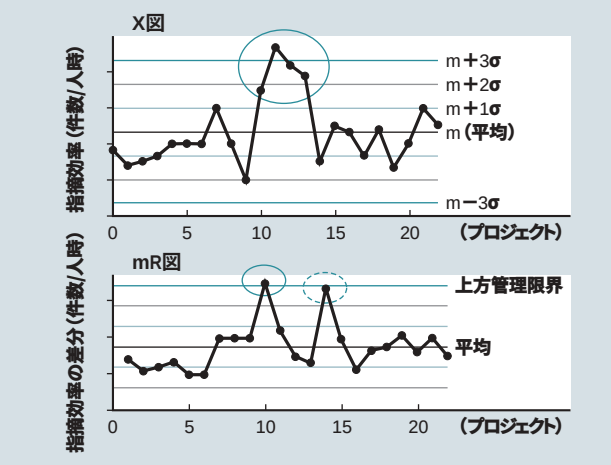

図5　再プロット結果
上図で丸をつけた4件のデータは，他の部分より指摘効率が高い。下図ではこれら４件のデータとの境界で異常値もしくは異常値に高い値になっている。

実際のXmR管理図を図4に示す。丸をつけたデータは他のデータから桁（けた）が違うほどかけ離れた明らかな異常値である。

このデータのレビュー記録を調べてみると，他のレビューとは明らかに異なる次のような特徴があった。

（1）認定されたレビューアがモデレータ（取りまとめ者）の役割を果たしていた。

（2）事前準備を実施し，そこでレビューでチェックする観点について合意を取っていた。

（3）定められた観点，具体的にはメモリの割り当てと開放だけに集中して，大量のソースコードの該当個所だけをレビューしていた。

（4）レビュー時には欠陥の摘出だけに専念し，修正方法など，他の議論は避けるようにしていた。

これらの特徴はWheeler[16]によるピアレビューの分類では，特定観点レビュー（Selected Aspect Review），インスペクションなどと共通するものとなっている。

ただし，図のプロット結果では準備作業に要した工数は考慮に入れていない。また，レビュー記録から見ると実際には複数回実施したレビューの結果を最後にまとめて記録されたような節もあり，記録されたレビュー時間の精度に問題があるのではないかという疑念もある。したがって，図4に表れている異常がすべて上記のレビューの特徴に起因するものかどうかは不明である。

次にこの異常値を外して，他のデータを再度XmR図にプロットし直した。図5が異常値を外した後のXmR図で，丸で囲んだ部分に新しい異常が出てきた。

このmR図では実線の丸で囲んだ管理限界を超したデータ以外に，破線の丸で囲んだデータもほとんど管理限界に近い値を示しており，移動範囲に不安定な要因があることがうかがえる。次に，X図を見ると，大きな丸で囲んだ4件のデータの部分でテスト1，2，3にすべて引っかかる激しい異常が観測されている。レビュー効率の値をヒストグラムで図6に示す。山が二つ以上に分かれており，左側で大きな山を作っているデータから離れたデータが4件あることが見て取れる。これらが丸をつけた4件のデータである。

レビュー記録を調べると，この4件のデータに対応するレビューは，同一のプロジェクト，同一の認定レビューアが指導して実施したものであった。レビュー方法は通常のウォークスルーで，レビュー方法に際立って特徴的な点は見いだせなかった。ただ，このプロジェクトは組込みシステム用のある特定のプラットフォームでの開発を繰り返しており，熟練者がそろっていた。

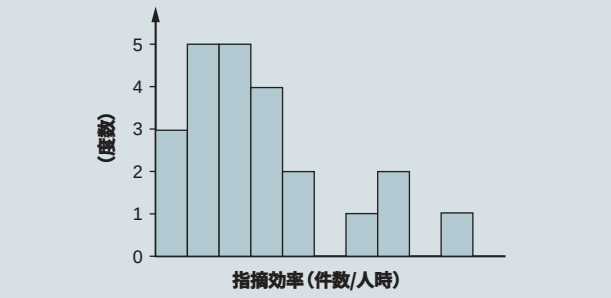

図6　レビュー効率のヒストグラム
2つ以上の群に分かれていることが読み取れる。

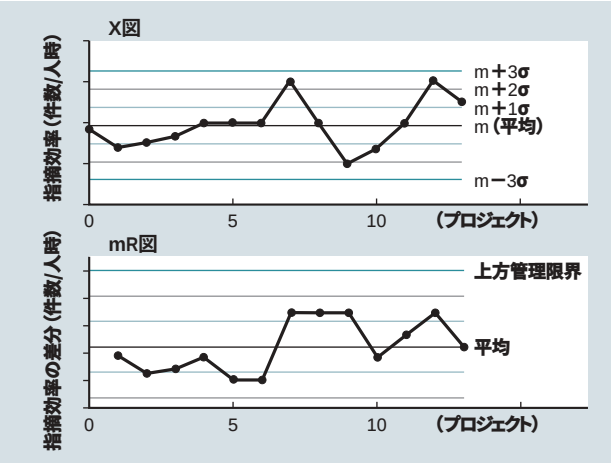

図7　特殊原因を除いた後
安定したデータ分布となっている。

レビュー方法に関する差異ではないが，プロセスの実施環境が違い，パフォーマンス的にも明らかに異なることから，以後このプロジェクトでのレビューは他のプロジェクトのレビューとは別のカテゴリとして扱うことにした。

このプロジェクトのデータを除いたデータをプロットし直すと，図7に示すとおり安定した範囲に収まった。レビュー方法としては，これらはすべてウォークスルーに分類されるレビューであった。

この後，新しいデータが追加されたため，以上と同様の分析を繰り返した。その結果，特殊原因の分析から次のような現象が観察された。

(1) 効率の高かったレビュー方法
　(a) レビュー準備を行った場合
　(b) 2人によるレビューの場合
　(c) 3人によるレビューで，指導する認定レビューアが該当レビューの業務内容にも精通している場合
(2) 効率の低かったレビュー方法
　(a) レビューに時間をかけすぎている場合

効率の低かったレビューでは，工程の最後のほうで，関係者を集めてほぼ1日かけてレビューをしていた例があった。興味深いのは，そのプロジェクトでは慣習的にこのようなレビュー方法をとっており，定量的に測定・比較したこともなかったため，レビュー実施者が自分たちのプロセスの効率の低さに気がついていなかったことである。

すでに述べたようにピアレビューの実施方法や測定，記録方法に不備があったため，以上の分析結果は完全に信頼できるものとは言えない。しかし，分析結果としてウォークスルーのパフォーマンスがほぼ把握できた。また，異常値の特殊原因を調べていくことにより上述のようなピアレビューのベストプラクティスを特定した。特に，レビューの仕方によってパフォーマンスがかなり異なることと，広く実施されているウォークスルー型のレビューはパフォーマンスが低いことから，ピアレビューの実施方法に大きな改善余地があることが明白となった。

なお，効率のよいピアレビュー方法と，それ以外の通常のウォークスルーのレビュー効率の比はおおむね4〜5倍程度であった。

6.2 初期分析時の効果分析

欠陥を上流で除く割合が多いほど，品質が向上することはすでに述べた。これは，ピアレビューの品質向上効果をも示している。ここではさらに，工数削減にもつながることを示す。

ウォークスルー型のレビューに関しては，指摘効率，すなわち工数当たりの指摘件数がほぼわかるようになった。これの逆数をとれば，欠陥を1件指摘するのにかかる工数が計算できる。これにさらに欠陥を実際に修正するのに要する工数およびレビューの準備工数を加えて，欠陥1件を摘出して修正するのに要する工数の推定値を算出した。

一方，欠陥を発見して除くには，テストを用いる方法もある。テスト工程での工数のかかり方を求めて，ピアレビューと比較することを考えた。テスト工程では組織レベルで，プロジェクトごとの摘出バグ数，工数が記録されていた。ただ，残念ながらこの時点では個々のバグに対する摘出工数，修正工数などの細かな記録がなかったため，バグ数と工数の間の相関を回帰分析により推定した。すなわち，バグの数が大きいほど，テスト工程でかかる工数も大きくなるという正の相関があるので，この相関係数として，バグが1件増えたときに増加する工数を算出した。この値とウォークスルーの場合のパフォーマンスの平均値との比を計算すると，約2.3となり，テスト工程のほうが余計に工数がかかるという結果になった(図8参照)。テスト工程で摘出しているバグを，ピアレビューによって摘出するようにすれば，工数を削減できることが同図からわかる。

さらに，この分析で用いたウォークスルーのデータは改善当初に実施されていたそれほど効率のよくないレビュー方法に基づくものであるから，レビュー方法に工夫をすれば削減効果はさらに大きくなることがわかる。

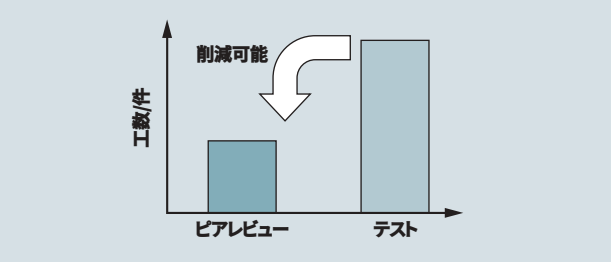

図8　ピアレビューとテストの工数比較
ピアレビューはテストの半分以下の工数で欠陥を除去できる。

6.3 改善項目と施策 —— 自律的改善に向けて

初期分析の結果は以下のようになる。

(1) プロセスに対する測定，記録の不備がわかったので，当面の改善点が明らかになった。

(2) パフォーマンス分析により，ベストプラクティス，ワーストプラクティスがわかり，その後の改善点や改善の方向づけができた。

(3) 精密な分析ではないが，ピアレビューの改善が品質および原価低減に効果を持つことがはっきりとした。

(2)で見いだしたベストプラクティスとワーストプラクティスはすでに多くの文献で指摘されていた内容と一致している[15), 16), 17), 18), 19)]。また，(3)の改善効果についても同様の内容の報告がすでにある[14)]。したがって，これらは新しい結果というわけではないが，自分たちのデータで結果が示されたということに意味があり，現場に展開していく際に大きな説得力を持った。例えば，自分たちの経験と文献の内容が符合したことはNAH（Not Applicable Here）症候群を乗り越える助けとなり，フェイガンインスペクションなどすでに提唱されているレビュー方法を導入する動機づけにもなった。

(1)と(2)に対応する改善点を実現するためにとった代表的な施策を次に述べる。

6.3.1　指標とその測定法の明確化，ルール化

(1) レビュー記録のフォーマットを改訂して，準備状況，レビュー対象規模など必要事項が記載されるようにした。

(2) ピアレビューに対する指標として，レビュー速度，指摘密度，レビュー前倒し摘出率の三つを定めた。さらに，最初の二つについてはそれぞれ XmR， Z 管理図でプロセス制御することとした。

6.3.2　ビジネスゴールとの関連づけ

日立ソフトとしてのビジネスゴールは高い品質の維持と競争力強化のための原価低減の2点にある。ピアレビューはこのどちらにも効果のあることがわかっており，初期分析の結果から定量的な効果予測も可能となった。そこで組織的にビジネスゴールに関連づけた目標設定を行うことにした。具体的には品質向上，原価低減の目標値を満足できるようにレビュー前倒し摘出率の事業部としての目標値を定め，これを受け，各プロジェクトも目標値を決めることとした。

6.3.3　ベストプラクティスの明文化

効果の高いレビュー方法の特徴をまとめ，ガイドラインとして明文化した。典型的な内容は以下のとおりである。

(1)レビュー時間，参加人数を絞って集中レビューする，ガイドラインとしてレビュー時間は2時間以内，参加者は4〜5名までとした。

(2)事前にレビュー観点を決めて共有する。このレビュー観点にはシステム目標，プロジェクト目標が反映される。

(3)成果物を事前配布し，参加者は目を通しておく。

(4)管理図を用いてレビュー実績を常にチェックして，安定化を図る。特に，レビュー速度を監視して，急ぎすぎたレビューにならないようにする。

(5) 成果物がすべて完成するのを待つようなことはせず，レビューできる状態になったものから順次レビューにかける。

6.3.4　ピアレビューに関する教育

二つの講座を立ち上げ，産業システム事業部内の設計者に教育を実施した。どちらの講座も100名以上が受講し，レビューの中心となる開発リーダー層をほぼ網羅している。

(1) ピアレビュー入門講座

ピアレビューの定義，意義，同事業部での実績に基づく効果の説明，上述のベストラインとガイドライン，管理図の見方と使い方などを解説する。

(2) フェイガンインスペクション講座

代表的なピアレビュー手法であり，欠陥摘出効果が最も高いとされるフェイガンインスペクションについてその手順を解説する。

6.3.5　ツールの提供

改善をうまく進めるには現場が自分で効果を実感できることが大切である。また，プロセスの安定化を図るためにも，現場で実績を測定・分析して即座にフィードバックしていける仕組みが求められる。そこで，プロジェクトが自分たちで管理図をプロットして異常値をチェックできるようなツールを開発し配布した。このツールでは管理図の種類や用途は完全にピアレビューに限定して，レビュー速度分析用のXmR管理図，指摘密度分析用のZ管理図だけをサポートするようにした。

6.3.6　分析サービス

統計的プロセス制御は新しい試みなので，ツールの配布と並行して，プロジェクトからデータを送ってもらって分析を実施し，結果を返却するサービスも実施した。結果はなるべくプロジェクトメンバーに直接説明するようにした。

6.3.7　静的解析ツールの適用サービス

ソースコードのピアレビューの助けとなるために，コードを静的に解析するツールを用意し，ピアレビューの事前準備

の一つとして活用してもらうようにした。

以上のような改善策において，特に注意をしたのは，改善活動がプロジェクトの自主的，自律的な活動として実施されるようにサポートしていくことであった。このような考えから，指標定義や管理図のチェックルールでも，組織からの押しつけと受け取られ得る内容は極力避けるようにした。

改善内容が浸透した事例として，次のようなプロジェクトがあった。そこではピアレビューを何回か実施していくうちに，レビュー時間が順次長くなり，管理図で見ると管理アウトにはなっていないものの指摘密度がしだいに落ちてきていた。そこで，プロジェクト内で原因分析の話し合いを行い，対策としてレビュー時に欠陥の修正方法の議論を避け，欠陥のたたき出しに専念するようにした。この結果，レビュー時間がガイドラインどおりの2時間になり，指摘密度もアップした。

このプロジェクトで実施されていたのはウォークスルー型のピアレビューであったが，たたき出しに専念するというインスペクションの特徴を取り入れて成功したわけである。ここで，特筆すべきなのは，この改善をプロジェクト自らが話し合って自主的に実施したということである。これが，正に今回の改善のねらいとしたところであった。

6.4　レビューデータの分析再訪と効果確認

6.4.1　フェイガンインスペクションの効果

前述したような施策を実施し，レビュー速度および指摘密度のパフォーマンスベースラインを組織レベルで確立した。この分析の過程で，まずフェイガンインスペクションの指摘密度がウォークスルーのそれよりも統計的有意に高いことが確認され，分離した別のベースラインを作成することとした。ここで，有意水準は管理図による統計的プロセス制御の設定に合わせて3σを超えるか，それと同程度に小さな確率とする。ここで，σは想定している分布の標準偏差を表す。

6.4.2　静的解析ツール利用の効果

次に，レビュー対象規模と指摘密度の関係を調べた。すでに，レビュー速度のベースラインを確立したことにより，レビュー方法およびレビュー条件（カテゴリなど）ごとに許容されるレビュー速度の範囲，特に速度の上限値は決まっている。しかし，レビュー速度は割り算をする点がやや間接的であるし，ベースラインの範囲であったとしても，ビジネスゴールを満足するとは限らない。そこで，直接的に把握できるレビュー規模の大きさによってビジネスゴールの実現具合を把握することを試みた。

レビュープロセスが安定している範囲で見ると，レビュー対象規模を大きくとることにより，レビュー速度が速くなり，指摘密度が落ちてくるという強い相関が観察された。ビジネスゴールはレビュー前倒し摘出率の目標値として表現されているが，品質指標としての欠陥密度の実績とレビュー前倒し摘出率を掛け算すれば，レビューで最低限必要な指摘密度がわかる。先の相関から，レビュー対象規模がある程度の大きさになると，指摘密度がこの下限値を下回ってしまうことが想定される。このことから，レビュー対象規模の上限値 L を決めることができる。

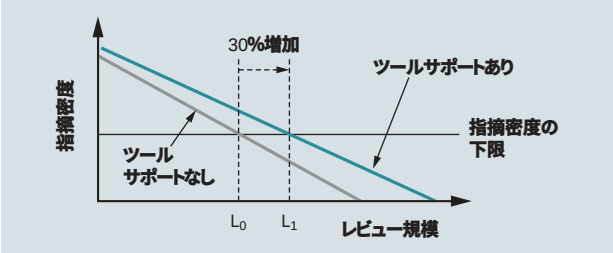

図9　ツールによる準備作業の効果
静的解析ツールをレビューの準備に用いると，1回のレビューでチェックするソースコードを30%増やしても，同程度の指摘密度を確保できる。

この手続きを，ウォークスルーのパフォーマンスについて，静的解析ツールを準備に使わなかった場合と使った場合に分けて，それぞれの上限値L0，L1を求めた(図9参照)。ツールを使用した場合，指摘密度の下がり方がゆるやかとなり，L1はL0の約1.3倍の値となった。なお，レビュー対象規模がL0以下の範囲では，ツールサポートを用いたか否かで指摘密度に有意な差は見られなかったが，L0とL1の間の範囲では統計的に有意な差が見いだされた。

この分析結果を受けて，事業部内規を次のように改訂した。「ソースコードレビューでは1回のレビュー規模を極力 L0 以下に抑えること，もし，それ以上の規模を対象にする場合には，静的解析ツールによる事前チェックを義務づけ，その場合でも L1を超す規模のピアレビューは認めない。」

6.4.3　レビュー効率の向上

初期分析の段階では，レビュー対象規模の記録が不十分であったこともあり，レビュー効率を分析し，ウォークスルー型のレビューのデータの安定部分を取り出した。しかし，その後のレビュー方法の改善活動ではレビュー効率については意図的にあまり強調しないようにし，指標としてもレビュー速度と指摘密度の二つをプロジェクトレベルの指標とした。これは，効率をメインの目標に掲げると，じっくりとレビューせずに，ともかく指摘件数を増やすという方向に走ってしまうのではないかと恐れたためである。改善としては，レビュー速度が速くなりすぎないように，ゆっくりじっくりレビューすることをめざした。これによって品質が向上し，結果的にレビュー効率も向上することを期待した。

改善前後のウォークスルー型レビューのパフォーマンスベースラインの平均値で比較すると，静的解析ツールを用いない場合で約4.8倍，用いた場合には約5.0倍にレビュー

効率が向上していた。この4〜5倍という数字は，初期分析のときに見いだしたベストプラクティスのパフォーマンスにほぼ等しい。改善運動の内容がプロジェクトに受け入れられ，ベストプラクティスが実際に実施されるようになった証左と考えられる。

7　統計的プロセス制御適用から得られた教訓

日立ソフトにおける適用経験から得られた教訓を幾つか挙げる。

(1) 成果物に関するデータだけでなくプロセスに関するデータを活用する。

ソフトウェア開発では，人に依存する創造的活動が多いため，プロセスを安定化することが特に大事である。さらに，事業的観点から価値の高い改善は上流工程に関するものであることが多い。この場合，明確に定義された形式の成果物は得られない場合も多いので，プロセスに着目することがきわめて重要となる。

(2) 個別データの扱いが，より重要である。

製造業の場合と異なり，同質で大量のデータが得られることは少ない。このため，個別データを分析し，きめ細かい改善を図っていくことが重要となる。

(3) データの量を増やすことより，データの安定性を重視する。

安定化したプロセスを得るためにデータを分離すること(層別)が必要な場合がある。統計処理には大量のデータが必要という先入観から，データの分離を嫌う向きもあるが，統計的プロセス制御ではプロセスの安定化こそ重要である。複数の共通原因が混在しているようなシステムでは分離を行わずにベースラインを確立することはできない。

(4) 改善活動における心理的要因も考慮する。

その改善活動が実施する人にどういう心理的影響を与えるかまで考慮する必要がある。人のやる気を損なうような押しつけは避け，自主的，自律的な改善をサポートしていくことが大切である。改善は本来楽しいものであり，管理図のように改善効果を可視化する工夫は大事なことである。

8　おわりに

ここでは，日立ソフトウェアエンジニアリング株式会社の事例を基に，統計的プロセス制御のソフトウェア開発プロセスへの適用について述べた。

統計的プロセス制御はソフトウェア開発においても有用であり，例えば，ピアレビューに対する統計的プロセス制御が，品質向上および原価低減に大きな効果を持つことを示す実証データが得られている。適用に関してはさまざまな課題があるが，それを克服する鍵は成果物中心の活動からプロセス中心の活動に頭を切り替えていくことにあると思われる。また，改善活動においては実施する人への心理的影響も考慮し，やる気を出させるようサポートしていくことが肝要である。

プロセス改善活動は事業目標に沿った形で行われなければならない。統計的プロセス制御はどの方向に改善活動を進めれば，事業目標を実現する効果が得られるのか，定量的な形で答えを出してくれる優れたナビゲーションシステムとなり得る。

参考文献など

1) 石川：品質管理入門，財団法人日本科学技術連盟(1989)
2) 鐵：品質管理のための統計的方法入門，財団法人日本科学技術連盟(2000)
3) M. B. Chrissis et al.：CMMI® Guidelines for Process Integration and Product Improvement，Addison-Wesley(2003)
4) W. S. Humphrey：Managing the Software Process，Addison-Wesley(1989)
5) 小室，外：開発現場の実態に基づいたピアレビュー手法改善と改善効果の定量的分析，No.4，Vol. 1，SEC Journal(2005)
6) M. Komuro：Experiences of Applying SPC Techniques to Software Development Processes，Proceedings of ICSE(2006)
7) M. Komuro et al.：Effective Use of PIIDs makes Process Improvement Easier，SEI SEPG Conference(2004)
8) 小室，外：プロセス改善活動の定量的評価，プロジェクトマネジメント学会春季研究発表会，pp.133-138(2004)
9) W.A.Florac et al.：Measuring the Software Process，Addison-Wesley(1999)
10) 日本規格協会編：JISハンドブック57　品質管理2001，財団法人日本規格協会(2001)
11) E.F. Weller：Practical Applications of Statistical Process Control，pp.48-55，IEEE Software，May/June(2000)
12) J. W. Cangussu et al.：Monitoring the Software Test Process Using Statistical Process Control， A Logarithmic Approach， Proc. of the 9th European software engineering conference，pp. 253-265，(2003.9)
13) M.A. Lantzy：Application of statistical process control to the software process， pp. 113 - 123， Proc. of the ninth Washington Ada symposium(1992)
14) S.H. Kan：Metrics and Models in Software Quality Engineering，Addison-Wesley(2003)
15) K. E Wiegers：Peer Reviews in Software: a practical guide，Addison-Wesley(2002)
16) D. A.Wheeler et al.：Software Peer Reviews，pp.454-469 in R. Thayer (editor)，Software Engineering Project Management，IEEE Computer Society Press(1997)
17) T. Gilb et al.：Software Inspection，Addison-Wesley(1993)
18) M.E.Fagan：Design and Code Inspections to Reduce Errors in Program Development，IBM Systems Journal(1976)
19) D.Bisant et al.： A Two-Person Inspection Method to Improve Programming Productivity，IEEE Transactions on Software Engineering，Vol. 15，No. 10，Oct(1989.10)
20) W.S. Humphrey：A Discipline for Software Engineering，Addison-Wesley(1995)