ソフトウェア工学 112－3
（1996．11．15）

効率的レビュープロセスの設計方法について

猪野 仁　　平山雅之
株式会社東芝 システム・ソフトウェア生産技術研究所
〒210 川崎市幸区柳町 70 番地
masashi@ssel.toshiba.co.jp

ソフトウェア開発におけるレビューの重要性は広く認識されており、古くよりさまざまなソフトウェアレビューの方法論が提案されている。しかし、現状のリソースや納期の制約のもとで十分に効果的なレビューが行われているとはいえない場合が多い。本稿では、レビュープロセスを構成する主要な活動項目（レビュー対象を選定する、レビュータイミングを定める、エラーを検出するなど）に対して、プロジェクトの制約や対象ソフトの特徴をもとに、活動項目毎の具体的な実施方法を定めていくことによって効率的なレビュープロセスを設計する方法を提案する。

Designing a High Efficient Review Process

Masashi Ino,　Masayuki Hirayama
Systems & Software Engineering lab., R & D Center, Toshiba Corp.
70 yanagi-cho, Saiwai-ku, Kawasaki, 210, Japan

Importance of a review in software development is recognized widely, and various review methods have been proposed. However under constrains of time for delivery and development resources, reviews are not put into effect in many actual projects. In this paper, we proposed a method for designing an efficient review process, by defining each activity of review process components (selecting parts of software, defining review timing, detecting errors, and so on), considering projects' constrains.

1.はじめに

ソフトウェアレビューに関して、これまでにさまざまなレビュー手法が提案されてきている。1970年代より、フォーマルレビュー、ウォークスルー、インスペクションなどが提案され、特にインスペクション[1][2][3]は現在でもレビューの代表的手法としてよく使われている。1990年代前後には、異なるチェック観点にレビュー担当者を分けて並行にチェックを行うN-Foldインスペクション[4]や、チェック観点を分解し段階的にレビューを行なうフェーズドインスペクション[5]が提案され、またIBMのクリーンルーム手法[7]の検証レビューは、方法論全体の中で重要な要素となっており、詳細化という限られた局面の正当性をチーム全員の理解のもとに保証する。

さまざまなレビュー手法は特定のレビューの進め方やレビュー対象のチェック方法を定めているが、実際のプロジェクトはそのシステムの特徴、規模、開発プロセスやメンバー構成、メンバーのスキル分布等が多岐に渡っており、必ずしもどれか一つのレビュー手法を適用してうまくいくとは限らない。また、ソフトウェア開発におけるレビューの重要性は十分認識されている一方、実際の現場では、「短納期のため時間的に圧迫されている」、「レビューをしても十分なエラーを検出できずあまり効果が上がらない」、「レビューできる人が少ない」、「レビュー対象量が多くて全部見きれない」ということなどから十分レビューされていない場合も多々ある。この現状に対して、そのそれぞれのプロジェクトの現状に適合した無理のない形で、より少ない時間とレビュー回数で効率的に十分なエラーを見つけられるようなレビュープロセスを設計できることが、レビューの効果をあげるために必要である。

本稿では、レビュープロセスを構成する「レビュー対象を選定する」、「レビュータイミングを定める」、「エラーを検出する」など9つの主要な活動項目に対して、プロジェクトの特徴をもとに活動項目毎の具体的な実施方法を定めていくことによってレビュープロセスを設計する方法を提案する。各活動項目に対してはさまざまな実施方法とその適用基準からなる実施方法選定ガイドラインを用意しており、容易にレビュープロセスを設計していくことができる。また各活動項目に対してはその評価メトリクスと評価方法ガイドラインを用意しており、設計されたレビュープロセスの妥当性を評価し次の開発にフィードバックすることができる。あるプロジェクトに対して、このレビュープロセスの設計および評価の方法を用いて、レビュープロセスを設計し、そのレビュープロセスに基づいてレビューを実施し、そのレビュープロセスを評価した適用事例を示し、提案する方法の評価を行う。

2.「レビュープロセスの設計および評価手法」の提案

図1にレビュープロセスの設計および評価の手順を示し、各ステップの内容について説明していく。

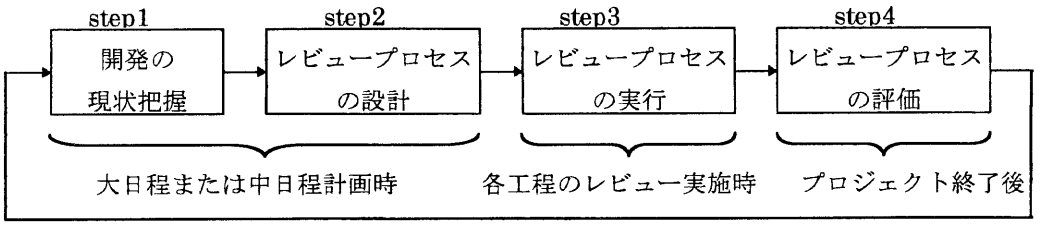


図1．レビュープロセス設計および評価手順

step1. 開発の現状を把握する。システムの規模や特徴、開発体制（スケジュール、人員構成、開発プロセス）、以前のレビュープロセスの評価結果、レビュー導入の容易さなどを明確にする。

step2. レビュープロセスを設計する。レビュープロセスを構成するいくつかの活動項目に対して、開発の現状をもとに実施方法設定ガイドラインを用いて具体的な実施方法を与えていく。この実施方法の連続がそのプロジェクトのレビュープロセスとなる。step1,2 はプロジェクトの計画工程で行い、各工程でどのようにレビューを行うかの手順を定める。

step3. レビュープロセスを実行する。設計、コーディングなどの各工程では、それぞれの工程に対して設計されたレビュープロセス（どのようにレビューを行うかの手順）に沿って、レビューの実施を繰り返す。またレビュープロセスを評価するために必要なメトリクスに沿ったデータを記録する。

step4. プロジェクトの終了後に、レビュープロセスを評価する。活動項目毎に設定されたメトリクスとその評価方法ガイドラインを用いレビュープロセス（各活動項目の実施方法）を評価する。

一連のステップが終わると step1 へ戻り、step4 での評価結果と新たな開発の現状をもとに再び手順を繰り返す。これによってさらにレビューの効率化、プロジェクトへの適合化を進めていくことができる。

以下、2.1 節で step2 で用いる活動項目の定義と、レビュープロセス選定ガイドラインの例を示し、2.2 節で step4 で用いる活動項目に対応するメトリクスの定義と、その評価ガイドラインの例を示す。

2.1.レビュープロセスを構成する活動項目の定義

step2 でのレビュープロセスの設計は、表１に示す９つの活動項目に対して、その実施方法を選択、カスタマイズ、または新たな実施方法を定めることである。それぞれの活動項目に対しては、さまざまな実施方法とその適用基準からなる実施方法設定ガイドラインを用意している。活動項目「(2)レビュー対象を選定する」と「(7)エラーを検出する」に対する実施方法設定ガイドラインの例を表２に示す。

表１．活動項目と実施方法例

活動項目	活動項目に対する実施方法の例
(1)目的を定める	使い易さの確認、仕様との整合・他システムとの整合、保守性・安全性のチェック、問題の解決、品質のモニタ、教育
(2)レビュー対象を選定する	全部、問題のあるもの、作成者のスキル・対象の複雑さ・重要度に応じて選定する
(3)メンバーを選定する	役割を決める、役割に割り当てる
(4)実施タイミングを定める	作成完了時、問題があるとき、段階的・定期的に、単体テスト終了時
(5)単位レビュー量を定める	システム・作成者・レビュー者に応じて推奨値を決める、機能単位を単位レビュー量とする
(6)事前準備をする	概要説明する、事前にエラー検出する
(7)エラーを検出する	ツール・技法・規約を利用する、検出項目を限定・分担・段階化する、設計書との対応をみる
(8)会議を進行する	エラーの検出と確認に集中、問題の解決方法に焦点、解決担当の割り当て、合否の判定
(9)エラーを修正･確認する	対応の有無・解決法を作成者が決定する、リーダが決定する、修正を確認する

表2．活動項目に対する実施方法設定ガイドラインの例

	活動項目に対する実施方法	適用の条件
(2) 対象を選定する	機能の重要度に基づいて選択する	再利用される部分／高安全性、高信頼性が要求される部分／複雑な処理の部分が明確である
	機能量、複雑性などの測定ツールの結果に応じて選択する	複雑性、メンバーのスキルに明らかな差がない、または不明である
	サンプリング(人や機能毎に開発対象の一部分を抜き出して検査)により選択する	
	スキルに応じて選択する	開発システムや開発に使う言語の経験が浅い人、過去にバグが多かった人
(7) エラーを検出する	エラー検出ツールを利用する（静的コード解析ツール、動的メモリ操作エラーチェックツールなど)	費用、マシン環境、教育、習熟性などを考慮してペイする
	エラー検出技法を用いる（パストレース法、トランザクションフローテスト法など)	システムの特徴に技法が適合している、教育・習熟コストに見合う
	チェックリスト、規約、標準を利用する	指摘が不十分になりがち、開発規模が比較的大きい、保守が重要な位置を占めている
	チェック項目を段階的にチェックする（簡単な問題と専門的な問題を検出する段階、人を分ける)	専門性の高いレビュー者が少ない（またはリソースが限られている）、専門性の低いレビュー者は用意できる
	チェック項目を重要なものに限定する（1つまたはいくつかの項目に限定して重点的にチェック)	

2.2.レビュープロセスの評価メトリクスの定義

設計されたレビュープロセスに従ってレビューが実施されるが、その妥当性を評価するために、レビュープロセスの評価メトリクスを設定する。メトリクスはレビュープロセスのそれぞれの活動項目に対して設定され、プロジェクト終了時にそれぞれの活動項目を評価する。表1の各活動項目に対するメトリクスを表3に示す。これらのデータは、レビュー記録用紙を用意して各回のレビューで記録する。

表3．メトリクスの定義と関連する活動項目

メトリクス	メトリクスの説明	関連活動項目
(a)対象量	頁(ドキュメント)、ＬＯＣ(=Lines of Code)(ソースコード)	(2)(5)
(b)準備工数	人数×準備時間	(6)
(c)会議工数	人数×会議時間	(6)(8)
(d)エラー件数	指摘エラー件数	(7)
(e)エラー内容	エラー種別、エラー内容	(1)～(9)
(f)エラー密度	エラー件数／対象量	(2)(5)
(g)エラー対応率	対応エラー件数／指摘エラー件数	(9)
(h)レビュー工数	準備工数＋会議工数	(2)(4)
(i)エラー検出効率	エラー件数／レビュー工数	(1)～(8)
(j)削減工数	Σ(エラー件数×エラー当たりの削減工数)－レビュー工数	(1)～(9)

活動項目毎に設定されたメトリクスによって、どのように評価するかを定めた表4のような評価ガイドラインを用い、レビュープロセスの評価を行う。

表4．活動項目に対する評価方法ガイドラインの例

活動項目	メトリクス	評価方法
(5)単位レビュー量を定める	(f)エラー検出効率	今回の単位レビュー量の設定により前回よりエラー検出効率が高くなったか?
	(e)エラー内容	単位レビュー量の設定によって指摘されるエラー内容の内訳はどう変化したか?

3.「レビュープロセスの設計および評価手法」の適用事例

活動項目のすべてに対し、その活動項目の実施の有無も含めて実施方法を定めることによって、そのプロジェクトに応じたレビュープロセスが設計できる。この節では前節で提案したレビュープロセスの設計および評価手法をあるプロジェクト(project-A)に適用した事例を示す。

3.1.開発の現状把握 (step1)

はじめにproject-Aに対する現状把握を行い、以下のような結果となった。

(1)レビュー対象工程　プログラム作成工程　**(2)開発対象システム**　制御用システムのサブシステム

(3)規模　大規模(140KLOC)　**(4)開発体制**　１８人で開発、１人３～４機能を担当

(5)スキル　開発経験が浅い開発者が多い、レビューできる人が少ない。

(6)システムの特徴　本システムは標準ソフトであり、客先毎にインデント対応して出荷する。インデント時には、データの設定や変更、ソフトの変更等がある。基本的にシステムは停止せずに運用されるため、長時間経過時のレアケースの十分な考慮が必要。機能間で複雑性にあまり差はない。

(7)前回開発システムの評価

① 図2はレビューで指摘されたエラーを修正した（対応した）率を示している。「指摘されたエラーに対する対応率が低く担当者によってばらつきが大きい」ということがわかる。

② 図3はエラーの種別を分類したものを示している。保守に関するエラー指摘の割合がかなり高くなっているが、対応されないエラーの多くは保守性エラーであった。またレビュータイミングが単体テスト終了後でコードが固まった後であったこともあり、直接システムの動作に影響しないものは納期の影響で後回しにされがちであった。しかしシステムの特徴からわかるように保守工程での品質および生産性向上のために保守性の向上が必要である。

図2．エラー対応率（前回）　平均50．8%

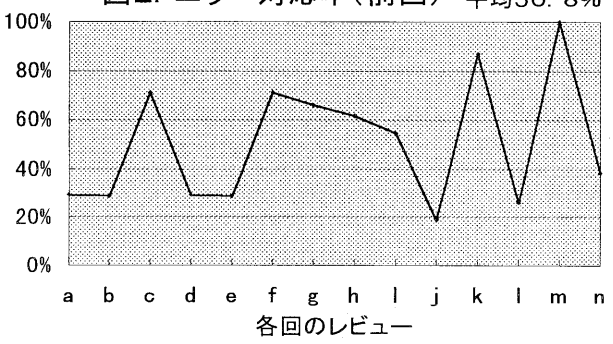
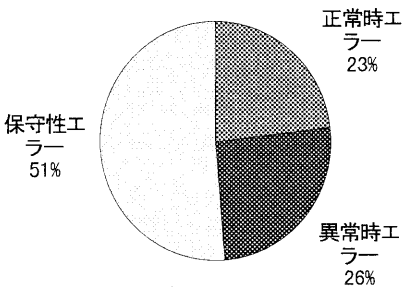


図3．エラーの分類（前回）



3.2.レビュープロセスの設計 (step2)

次に、表2に示すような実施方法設定ガイドラインに上記の開発の現状を適用条件として与え、表1で定義した各活動項目に対する実施方法を定めた。これを表5に示す。

表5　project-A のプログラム作成工程のレビュープロセス

活動項目	実施方法
(1)目的を定める	保守性、異常系のチェックを主目的とする。
(2)対象を選定する	①全体的に開発者の経験が浅く、開発対象の重要度や難易度には差異がないので、初めに作った機能を全員が1人1回ずつレビューされるように選択する。 ②一回目のレビューの結果、エラー密度が大きかった対象の作成者は次の機能についてもレビュー対象として選定する。
(3)メンバーを選定する	開発者は全員一人一回ずつレビューされるようにする。レビュー者は3人のベテランが持ち回りで行う。
(4)実施タイミングを定める	早いフィードバックをかけるため作成完了時にレビューする。
(5)単位レビュー量を定める	一回のレビューで一機能分のレビューを行う。
(6)事前準備をする	事前にレビューを行い、問題点リストを作成する。
(7)エラーを検出する	①保守性エラーはプロジェクト全体で必ず守るように取り決めし、規約（表6参照）として用意した。また異常系エラーもレビュー者によって指摘がばらつくため、これも規約として用意する。 ②保守性エラーや異常系エラーのいくつかを自動検出するツール（静的コード解析ツール、動的メモリエラーチェックツール）を利用する。本ツールはレビュー時にレビュー者が指摘の補助として使ってもよいが、ここでは、開発者が利用しレビュー前に検出・修正を終えていることとした。
(8)会議を進行する	エラーの確認に集中する。必要な指導は行う。
(9)エラーを修正・確認する	守るべき規約を明らかにし、それには確実に対応させる。

また、「(7)エラーを検出する」で用意することとした規約の概略を表6に示す。

表6．用意した規約（抜粋）

保守性エラー	①コメントと実態の不一致・コメントのスタイル　②コーディングスタイル（スペーシング、ネーミング）　③長すぎる関数、複雑すぎるロジック　④マジックナンバー、データの埋め込み　⑤エラーの扱いの統一化　⑥型の不整合，小さな型への変換　⑦配列領域オーバー　⑧初期化抜け
異常系エラー	①ログがあふれる、データストレージがあふれる　②メモリが不足する　③異常ケースの考慮抜け　④異常時の回復処理　⑤過負荷時の性能

3.3.レビュープロセスの実行と評価 (step3, step4)

レビュープロセスの評価では、各活動項目に対して設定されたメトリクスを用い、活動項目のそれぞれを評価することで行う。ここでは、「(2)レビュー対象を選定する」、「(7)エラーを検出する」、「(9)エラーを修正する」の3つの活動項目を取り上げ評価結果を示す。

3.3.1.「(2)レビュー対象を選定する」に対するの評価

表3に示した「(2)レビュー対象を選定する」に関連するメトリクスのうち、(a)対象量、(f)エラー密度を用いて実施方法を評価する。レビュー実施の際は、全員（18人）に対して初めに作った機能を1人

1回ずつレビューした。また2回目以降に作成した機能については、レビュー対象の選定効率を評価するため、平均的なスキルの人を時期をずらして3人選び3回だけレビューを行った。

その結果、表4に示すように、レビューした量は全体の3割であるが、その3割についての平均エラー密度は11.5件/KLOCであるのに対し、レビューしなかった量（全体の7割）については、2回目以降作成の機能に対して行った3回のレビューから見積もられる平均エラー密度は3.8件/KLOCであった。1回目の平均エラー密度は2回目以降の約3倍であり、高エラー密度の対象を選択することができ、限られたレビューリソースを効率的に活用することができた。1回目に比べ2回目以降は、①対象知識が増える、②同種のプログラムのため一度早期に指摘されたエラーを2機能目以降は作り込まなくなる、等の効果があったためと考えられる。

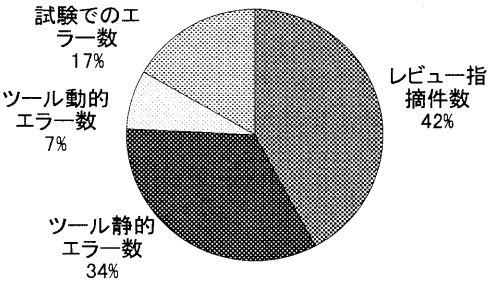
表7．「(2)レビュー対象を選定する」の評価結果

評価メトリクス	レビュー量	レビュー回数	平均エラー密度
初めに作成した機能 (選択したもの)	31.5KLOC (29%)	18回	11.5件/KLOC
2回目以降作成の機能 (選択しなかったもの)	107KLOC (71%)	3回 (見積り実施)	3.8件/KLOC (見積もり値)

3.3.2.「(7)エラーを検出する」に対する評価

表3で示した「(7)エラーを検出する」に関するメトリクスのうち、(e)エラー内容と(d)エラー件数に注目して評価を行った。エラー検出方法別の件数の割合を図4に示す。エラー検出ツールを利用した場合、平均4割のエラーをレビュー前に除去し、レビュー工数を減らし、エラー検出効率を向上させることができた。レビューの場ではより高度なエラーに集中し効率を上げることができた。

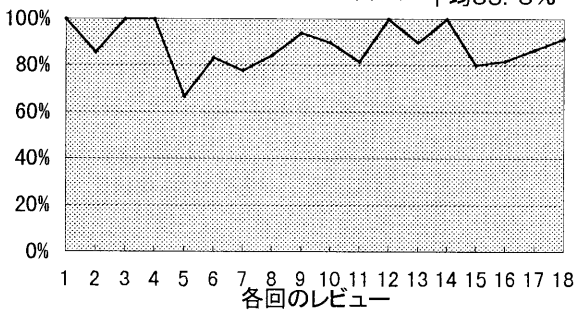
図4．ツールによるエラー検出効果



3.3.3.「(9)エラーを修正する」に対する評価

「(9)エラーを修正する」に関するメトリクスのうち(g)エラー対応率に注目して評価を行った。エラー対応率は、前回（図3参照）は平均50.8%だったが今回（図5参照）は平均85.8%に上がり、また人によるばらつきも少なくなった。これにより検出されても対応されないエラーの割合を大幅に減らし、かけた工数に対する品質向上の効率を改善することができた。

図5．エラー対応率（今回） 平均85．8%



4．「レビュープロセスの設計および評価手法」の評価

対象プロジェクトの性質に応じたレビュープロセス設計手法を用いることによって以下のような効果があった。

① レビュープロセスの設計手法（設計手順、活動項目、実施方法選定ガイドラインなど）により、プロジェクトの現状に適合した効率的なレビュープロセスが容易に設計できるようになった。

② レビュープロセスを評価する方法（活動項目毎のメトリクスと評価基準）を与えることにより、レビュー活動項目のどこがよくどこが悪いかが明らかになり、次のプロジェクトでの具体的な改善（レビュープロセスの設計）が容易になった。

③ 設計されたレビュープロセスはプロジェクトで実行可能でかつ効率のよいものであり、レビューの形骸化や縮小傾向に歯止めをかけられるようになった。

④ これら一連の開発条件、設計、評価のデータは設計方法や評価方法のガイドラインの形で蓄積されるようになり、他のさまざまなプロジェクトでのより効率的なレビュープロセスの設計が可能になった。

5.まとめ

本論文ではレビュープロセスを構成する９つの活動項目に対して、プロジェクトの特徴をもとに活動項目毎の具体的な実施方法を定めていくことによってレビュープロセスを設計する方法を示した。また活動項目に関連するメトリクスを設定し、レビュープロセスの効率・妥当性を評価する方法を併せて示した。適用事例では、「エラー密度の高い対象を選択し限られたレビューリソースを効率的に活用できた。」、「検出されても対応されないエラー割合を大幅に減らし、品質向上のための費用対効果を上げることができた。」などの結果が得られ、これら活動項目毎の効率化、改善により全体としてのレビュープロセスをより効率的なものとしていくことができた。このように本手法を用いて効率的なレビュープロセスが設計ができた。

今後は、多くの事例に本手法を適用し、さまざまな性質を持った開発プロジェクトに対しての適用可能性を検証していくとともに、レビュー技術の調査、研究を継続し、活動項目に対する実施方法選定ガイドライン、メトリクスを用いた評価ガイドラインを充実させていきたい。

参考文献

[1] Michael E. Fagan, Design and code inspections to reduce errors in program development, IBM System Journal, Volume 15, Number 3 , 1976

[2] Robert G. Ebenau and Susan H. Strauss, Software Inspection Process, Systems Design & Implementation Series, 1993

[3] Tom Gilb and Dorothy Graham, Software Inspection, 1993

[4] Schneider,G.M.,Martin,J. and Tsai,W.T. An experimental study of fault detection in user requirements documents. ACM Trans. Software Engineering Method 1,2, 1992

[5] John C. Knight and E. Ann Myers, An Improved Inspection Technique, COMMUNICATIONS OF THE ACM, Volume 36, Number 11, 1993

[6] Robert Grandy and Tom Van Slack, Key Lessons in Achieving Widespread Inspection Use, IEEE Software, Volume 11, Number 4, 1994

[7] 佐藤、無故障ソフトウェアを開発するためのクリーンルーム手法紹介、情報処理、３５巻９号、1994

[8] Daniel P. Freedman and Gerald M. Weinberg 著、岡田監訳、ｿﾌﾄｳｪｱ技術ﾚﾋﾞｭｰﾊﾝﾄﾞﾌﾞｯｸ、1987

[9] E.ヨードン著　国友義久・千田正彦共訳、ソフトウェアの構造化ウォークスルー、1981

[10] 菅野文友監修、ソフトウェア・デザインレビュー、日科技連、1982

[11] 佐藤、ソフトウェア・デザイン・レビューの実践技法、ソフト・リサーチ・センター、1990

[12] ソフトウェアプロセス成熟度の改善、Watts S. Humphrey 著、藤野監訳、日科技連、1991