特集論文

コードレビュー手法の改善による コード品質確保の効率化

井元　薫*
細谷泰夫*

Efficiency of Code Quality Assurance by Improving a Method of Code Review

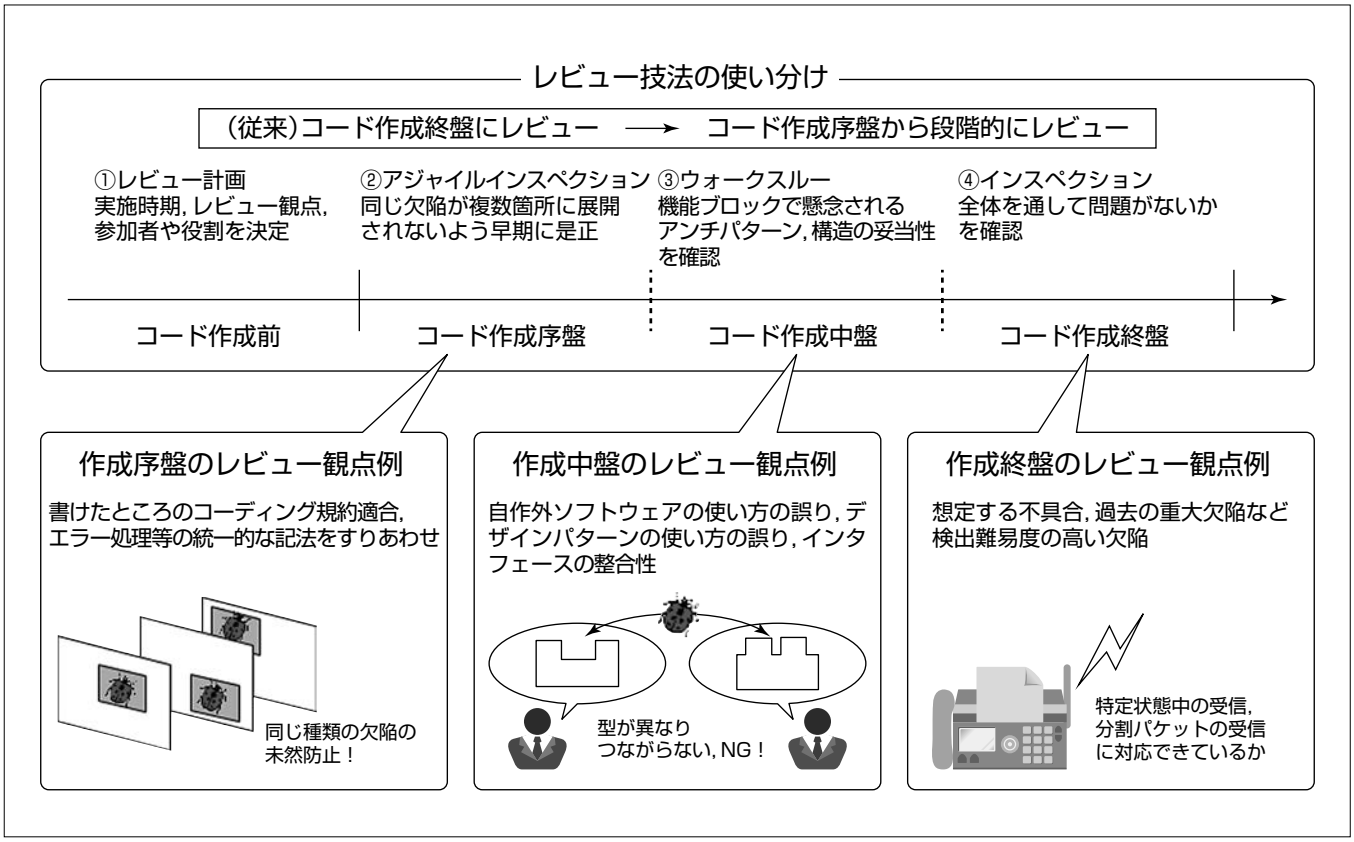
Kaoru Imoto, Yasuo Hosotani

要　旨

近年，製品開発でソフトウェアの高機能化と短期開発が進んでいる。この状況下でソフトウェアのQCD(Quality Cost Delivery)を達成することが求められる。

ソフトウェア開発ではレビューとテストを組み合わせた品質確保が一般的である。ソースコードのレビュー(以下“コードレビュー”という。)では，ソフトウェアの誤動作の要因となる欠陥や，ソフトウェア構造崩れといった将来の保守性を阻害する要因等本質的な欠陥の除去が重要である。しかし，大規模化したコードに対し，限られたコストと期間でのレビューでは，指摘が軽微な欠陥に偏った場合に，先に述べた本質的な欠陥が除去しきれないままレビューが完了してしまうという課題があった。

指摘が軽微な欠陥に偏る原因を分析した結果，レビュー観点が曖昧であること，レビューアのスキル不足が主な原因であることが分かった。そこで，コードレビューで検出する欠陥を分類し，検出したい欠陥の種類に応じたレビュー観点を設定した。さらに，レビュー観点に応じた技法，タイミングを明確化した。レビュー計画で，必要な観点，技法，タイミングを計画し，適切なレビューアをアサインすることによって，レビューアのスキル不足やレビュー観点の重複回避が可能となった。これらの方策によって，限られた期間とコストで効果的なコードレビューの実施が可能となった。



レビュー技法，レビュー観点，レビュータイミングの使い分けによるコードレビュー計画

コード作成フェーズによって，レビュー技法，レビュー観点，レビュータイミングを使い分けることで，軽微な欠陥の指摘は早期に，検出難易度の高い指摘は後期に集中させる。これによって，コード品質を確保する。

*設計システム技術センター

1. まえがき

近年，製品開発でソフトウェアの高機能化と短期開発が進んでいる。この状況下でソフトウェアのQCDを達成することが求められる。

ソフトウェア開発ではレビューとテストを組み合わせた品質確保が一般的である。ソースコードのレビューでは，ソフトウェアの誤動作の要因となる欠陥や，ソフトウェア構造崩れといった将来の保守性を阻害する要因等本質的な欠陥の除去が重要である。しかし大規模化したコードに対し，限られたコストと期間によるレビューでは，指摘が軽微な欠陥に偏り，先に述べた本質的な欠陥が除去しきれないという課題があった。

そこで，コードレビューで検出する欠陥を分類し，検出したい欠陥の種類に応じたレビュータイミング，レビュー技法，レビュー観点を設定することで，本質的な欠陥を効果的に検出可能なレビュー手法を開発した。

本稿では，コードレビューが抱える課題に対し，開発したレビュー手法とその効果について述べる。

2. コードレビューの課題

2.1 軽微な欠陥指摘への偏り抑制(レビューの質の問題)

ソフトウェア開発の大規模化，短納期化が進んでおり，コードレビューの時間が十分に取れないために欠陥が流出するリスクが増加している。そのため，コードレビューで効率的に欠陥を検出する必要がある[1]。従来，デザインレビューでは，セルフレビュー，ピアレビュー，書面レビュー，及び会議レビューを段階的に組み合わせて行っていた。しかし，次の２つの原因によって，軽微な欠陥への指摘に偏ってしまい，本質的な欠陥の除去が困難となる。

(1) レビュー観点が曖昧で，狙いが曖昧なまま目についた欠陥を指摘する。

(2) レビューアのスキル不足で保守性や移植性といった内部品質に関する指摘や本質的な欠陥が見つけられない。

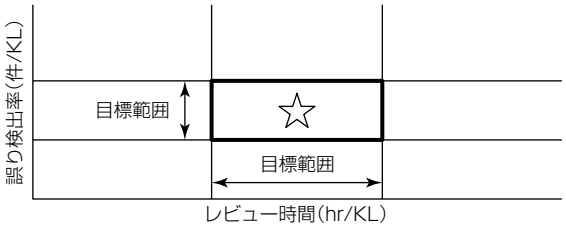


図１．コードレビューのゾーン判定

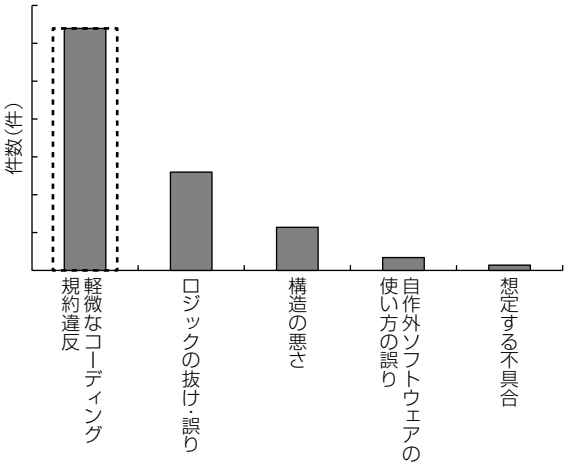


図２．コードレビューの欠陥種別

2.2 従来のレビュー品質評価方法の問題点

品質を評価する手法の１つとして，ゾーン判定[2]がある。ゾーン判定は，**図１**のように表す。レビュー時間の基準値と，誤り検出率の目標範囲をともに満たしている場合に，品質が良いと評価される。図の星印のように，レビュー時間，誤り検出率がともに目標範囲内であり，かつ，**図２**のように，コーディング規約違反のような軽微な欠陥の指摘に偏ってしまっている場合は，ゾーン判定だけでは，コードレビューが十分に実施されていると判定されてしまう。このような場合に，重大な欠陥を見逃すリスクが大きくなる。

3. 対　策

3.1 レビュー観点の体系化

(1) 欠陥の種類をレビュー観点として体系化

コードレビューで検出する狙いを定めやすくするために，次の観点で指摘すべき欠陥全体を**図３**のとおり４象限に分割した。

(縦軸)粒度大／小

大：複数箇所のコードを見て指摘できるもの。

小：１箇所のコードだけ見て指摘できるもの。

(横軸)誤り／誤りでない

誤り：不具合動作を起こす原因となるもの。

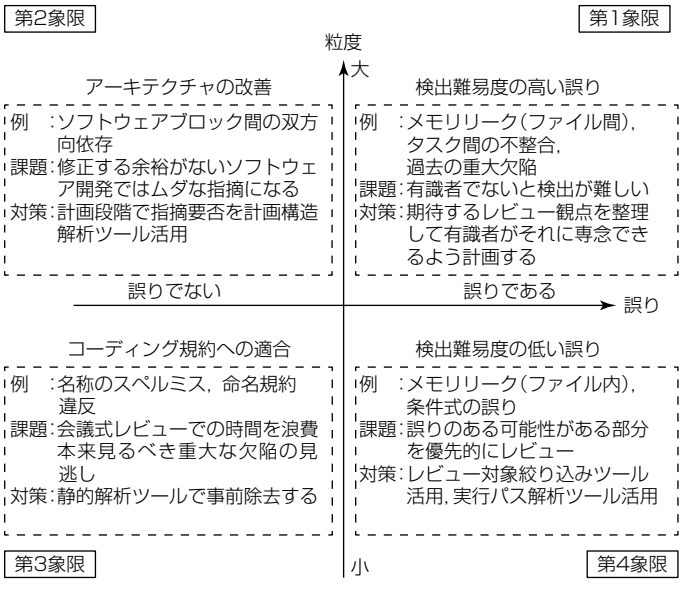


図３．指摘すべき欠陥全体の４象限

誤りでない：改善提案，書き方に関する指摘，質問等。
正常動作するが，コーディング規約適合，保守性悪化等の影響があるもの。
例）命名規則違反，スペルミス，複雑なソフトウェア構造

さらに，それぞれの象限に対応したレビュー観点を体系化した(**表１**)。例えば，構造の悪さは，複雑さ等構造に関する要件に対し，コードの適合を確認する観点でレビューを行う。レビューアには，構造の良しあしを判断できるスキルが必要となるため，難易度は高くなる。また，構造の悪さは，複数のファイルにまたがる観点と，単一のファイル内に収まる観点があるため，対応する象限は第２象限，第３象限となる。

⑵　レビューアごとの狙う欠陥を明確化

表２のようにレビューアのスキルを考慮し，レビュー観点を割り当て，レビューの狙いを明確化する。

構造の悪さ等難易度の高いものは，ベテランをレビューアとし，ロジックの抜け・誤り等難易度の低いものは，中堅や若手をレビューアとする。

3.2　レビュー観点によるレビュー計画の適正化

⑴　段階的なレビュー計画

図４のように，コード作成前にレビューを計画し，コード作成序盤，中盤，終盤それぞれのレビューで，アジャイルインスペクション，ウォークスルー，インスペクションとレビュー技法を使い分ける。各段階でのレビュー観点を明確にすることによって，軽微な指摘への偏りを防止できる。

また，**表３**のように，レビュー観点をレビュー実施時期によって，使い分ける。第１ステップでは，軽微な欠陥を早期に指摘することで，教育効果によって同種の欠陥増加を防止する。第２ステップ以降では“構造の悪さ”等の検出難易度の高い欠陥指摘に集中する。

⑵　レビュー完了基準の設定

対象となるレビュー観点に基づいてコードレビューを実施した際，欠陥割合が基準値を超える場合に，レビュー完了とする。完了基準に未達の場合，同じ観点で再レビューを実施する。レビューごとに**表３**のように対象となるレビュー観点を定める。

3.3　ツール活用によるレビューの効率化

次の用途にツールを活用することによって，コードレビューを効率化できる。

⑴　レビュー対象の絞り込み

レビュー観点に合致するレビュー対象を絞り込むことで，限られた工数とリソースを用いて効果的なレビューを実施できる。

⑵　ロジックの抜け・誤りの自動検出

ロジックの抜け(if文のelse抜け等)，誤り(配列外アクセス等)の可能性がある箇所を自動検出することでレビューを効率化できる。

⑶　レビュー対象の構造の理解

レビュー対象の静的構造に関する情報をレビューアに提示することで，レビュー対象を理解しやすくし，目視によるレビューを効率的，効果的に実施可能にする。

表１．レビュー観点の体系化

No.	レビュー観点		難易度	象限
1	構造の悪さ	複雑さなど構造に関する要件に対し，コードの適合を確認する。	高	2，3
2	想定する不具合	製品不具合事例に対し，原因となりうるコードでの対策を確認する。	高	1
3	自作外ソフトウェアの使い方の誤り	自作外ソフトウェアの不具合事例に対し，原因となりうるコードでの対策を確認する。	高	1，4
4	設計ポリシー違反	SPL等設計ポリシーに対し，コードの適合を確認する。	中	2
5	仕様との不整合	仕様書とコードの不整合がないことを確認する。	中	1，4
6	変更影響箇所の漏れ・誤り	ソフトウェア構造を基に変更箇所，変更方法が妥当であることを確認する。	中	1，2，3，4
7	ロジックの抜け・誤り	未初期化などコーディング誤りパターンに合致するものがないことを確認する。	低	1，4
8	コーディング規約違反	コーディング規約への適合を確認する。	低	3

SPL：Software Product Line

表２．レビュー観点とレビューアの対応例

No.	レビュー観点	難易度	レビューア
1	構造の悪さ	高	ベテランA
2	想定する不具合	高	ベテランB ベテランC
3	自作外ソフトウェアの使い方の誤り	高	ベテランC
4	設計ポリシー違反	中	中堅A
5	仕様との不整合	中	中堅A 中堅B
6	変更影響箇所の漏れ・誤り	中	中堅C
7	ロジックの抜け・誤り	低	中堅A 若手A
8	コーディング規約違反	低	中堅B 若手B

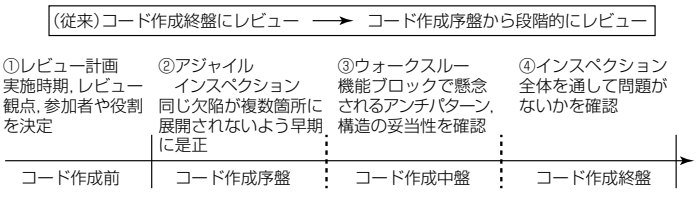


図４．レビュー技法の使い分け

表３．コードレビュー計画例

No.	レビュー観点	難易度	レビュー＃１		レビュー＃２		レビュー＃３	
			実施	レビューア	実施	レビューア	実施	レビューア
1	構造の悪さ	高					○	ベテランA
2	想定する不具合	高					○	ベテランB ベテランC
3	自作外ソフトウェアの使い方の誤り	高					○	ベテランC
4	設計ポリシー違反	中			○	中堅A		
5	仕様との不整合	中			○	中堅A 中堅B		
6	変更影響箇所の漏れ・誤り	中			○	中堅C		
7	ロジックの抜け・誤り	低	○	中堅A 若手A				
8	コーディング規約違反	低	○	中堅B 若手B				

表4．コードレビューに活用可能なツールの例

ツールの用途	ツール名	ツール概要
レビュー対象の絞り込み	CCFinder	クローン検出ツール。ソースコードをトークン単位で直接比較することでコードクローンを検出する。
レビュー対象の絞り込み レビュー対象の構造の理解	Lattix	アーキテクチャ分析ツール。ソフトウェアアーキテクチャの構造分析を行い，サブシステム間の依存関係を簡潔な表形式(マトリックス)で表現する。
ロジックの抜け・誤りの自動検出	Klocwork	実行パス解析技術の活用によって危険箇所を効率的，網羅的に検出する。

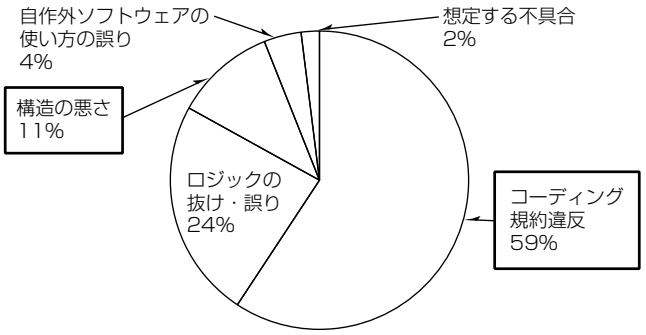


図5．改善前のコードレビューの欠陥分類

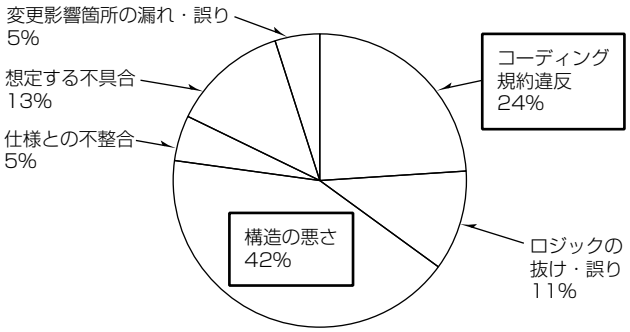


図6．改善後のコードレビューの欠陥分類

コードレビューに活用可能なツールの例を**表4**に示す。

4. 効　果

レビュー観点を体系化することによって，レビュー観点の偏りが減少し，本質的な欠陥除去の効果が増加した。開発期間約8か月の通信シミュレータの開発に適用し，**図5**，**図6**のように，検出難易度の高い欠陥検出(構造の悪さ)の割合が11%から42%へ増加し，軽微な欠陥検出(コーディング規約違反)の割合が59%から24%へ減少した。

また，コードレビューごとのレビュー観点設定(**表5**)によるレビュー計画の適正化によって，効率良く品質を確保することができた。**図7**のとおり，第2ステップで対象としていた構造の悪さとロジックの抜け・誤りの観点について，欠陥割合が30%となった。完了基準(50%)未達となったため，再レビューを実施したことによって，狙っていた欠陥を検出することができた。

表5．コードレビューごとのレビュー観点例

レビュー	対象レビュー観点
#1	コーディング規約違反 仕様との不整合
#2	構造の悪さ ロジックの抜け・誤り
#3 再レビュー	構造の悪さ ロジックの抜け・誤り

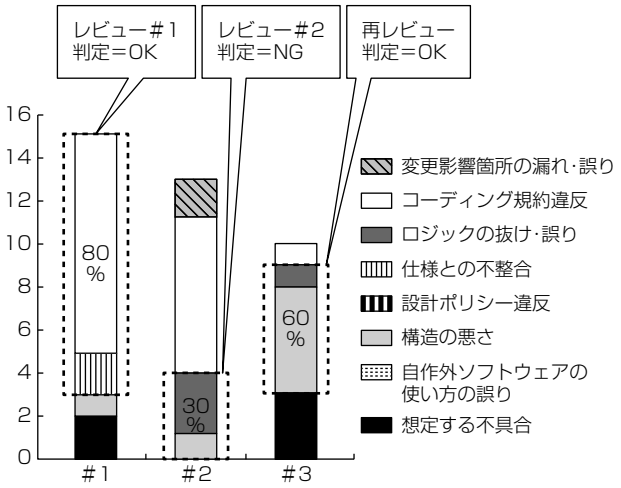


図7．コードレビュー実施結果例

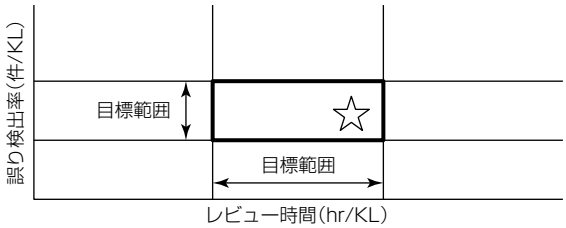


図8．再レビュー後のゾーン判定

再レビュー後のゾーン判定は**図8**の星印となり，レビュー時間，誤り検出率ともに目標範囲内となった。

5. む　す　び

製品開発で，ソフトウェアの高機能化と短期開発が進んでいる。そこで，限られたコストで効果的に欠陥を検出するために，開発したレビュー手法とその効果について述べた。

今後は，レビュー手法を更に進化させ，品質確保の効率化を推進していく。

参 考 文 献

(1) (独)情報処理推進機構　ソフトウェア・エンジニアリング・センター：組込みソフトウェア開発における品質向上の勧め(コーディング編)，翔泳社，31(2005)
(2) (独)情報処理推進機構　ソフトウェア・エンジニアリング・センター：定量的品質予測のススメ，オーム社，58(2008)