31SQiP 研究会　特別講義　レポート

作 成 日：　2016 年　2 月 5 日
書記氏名：　加藤　蔵次

日時	2016 年 1 月 15 日（金）　10:00 ～12:00
会場	(財) 日本科学技術連盟・東高円寺ビル　地下 1 階　講堂
テーマ	「レビューを失敗させるコツ」
講師名・所属	原　佑貴子　氏（日本アイ・ビー・エム株式会社　グローバルビジネスサービス）
司会者	中谷　一樹　氏（TIS 株式会社　生産革新本部）
アジェンダ	1. レビューを実施する理由 2. レビューを失敗させるコツ 3. これからのレビュー
アブストラクト	レビューを実施することは必要なことだと理解していると思います。それにも関わらずレビューが嫌いだと思っている人、有効なレビューができていないと困っている人が大多数です。レビューは４０年以上もの実績がある手法です。ここで説明する７つのコツで、レビューが開発のように面白くて楽しいものになって欲しいと思います。最後に、これまで人間がマニュアルで実施してきたレビューがこれから先どう変わっていくのか考えてみます。

第 8 回の例会では、「レビューを失敗させるコツ」と題して日本アイ・ビー・エム株式会社の原さんにご講演を頂きました。「失敗させる」という興味を引くタイトル、７つのコツを７番目から順番にカウントダウンしていく構成、聴講者を巻き込んで考えるなど最後まで楽しく、そして原さんのレビューに対する熱い思いが十分に伝わった講演でした。原さん、ありがとうございました。

－－

1．レビューを実施する理由

レビューを実施することは必要なことだと皆さん理解していると思いますが、それにも関わらずレビューが嫌いな人が大多数です。その理由は、バグを好んで入れ込む人は決していないにも関わらず、知らず知らずに入れ込んでしまったことを責められている感じがするからだと思います。また「レビューをすると品質があがる」と思われるかもしれませんが、それは単なる迷信です。レビューを実施するだけでは単にバグが検出され品質が悪いことがわかるだけです。検出されたバグを利用して改善（フィードバック）することによって、始めて品質が向上するのです。

また、有効なレビューが実施できていないと困っている人が大多数です。その理由として「効率的なレビューができていない」・「重要な欠陥が検出できない」といったことが挙げられます。突き詰めていくと「レビュー時間を確保できない」・「軽微な欠陥（誤字・脱字）しか検出できない」といったことなどが理由となります。

こういった状況に陥ってしまう理由をこれから共有していきたいと思います。

2．レビューを失敗させるコツ

これから７つのコツを説明します。カウントダウン方式になっていて７番から始まって最後は１番となっています。

7） ７つのステップ

フォーマル・インスペクションは IBM　Kingston 研究所の Michael E. Fagan らが 1960 年代前半にテスト前にソースコードからバグを除去したいという思いで考案し、徐々にブラッシュアップして上流の仕様書にも適用できるようにした手法で 1976 年に公に発表しました。ソフトウェア工学の方から、どんなに良い手法であっても儲けが確保できなければその手法は定着しないと言われたことがあります。その点からするとフォーマル・インスペクションは 40 年以上もの間使い続けられている手法であり、正しく使えば必ず儲かるハズです。

レビューには７つの手順（開始基準、プラン、準備、ミーティング、リワーク、フォローアップ、完了基準）があります。この手順の中でのポイントを述べます。

・開始基準を満たしているか(※１)

レビューに値する成果物かどうか、例えばゲートのレビューで成果物に次のような言葉がある時は闇雲にレビューを始めてはいけません。(To Do、要検討、～のハズ、～かも、といったような言葉)

・レビューの計画を立てる

テストに計画があるように、レビューにもいつからいつまでの段階で、誰が、どのような見方でレビューするか計画を立てる必要があります。会議室に集まってミーティングをすれば良いというものではありません。

・レビューの前に個人で事前チェックをする

レビューの前に対象の資料を配布しレビューまでに個人で事前にチェックします。軽微な欠陥（誤字・脱字、フォーマット、細かい指摘、等）をオフラインで担当者に伝えて事前に修正しておきます。レビューのミーティングでは重大な欠陥のみを持ち寄って実施するのが本来のレビューの形です。

(※１)イテレーション開発のように徐々に開発する場合（質疑応答から）

プロジェクトの状況・前提を事前に確認してから実施して下さい。どこまでできているかを前提としてレビューして下さい。第三者的には、レビューで検出した結果が問題なのか、まだ取り掛かっていないだけなのか判断できないので要確認事項として挙げた上でプロジェクトの前提をもとに最終的に判断して下さい。

6） ６手法の使い方

レビューには代表的な６つの手法があります。厳密性の高い手法から、フォーマルインスペクション、ウォークスルー、パスアラウンド、Ｎフォールドレビュー、２パーソンインスペクション、個人レビュー・セルフチェッ

クとなります。レビューを選択する基準としては欠陥検出精度、コスト負荷、時間負荷、検査の厳密性、チームワーク・組織醸成、教育効果・技術移転、専門教育（トレーニング）の必要性、記録・フォロー活動追跡、管理の難易度が考えられます。
プロジェクトのリスクや制約事項（状況やアサインできるメンバ等）に応じてレビュー手法を選択します。その選択の仕方には考え方がありますので、いくつかの事例を紹介します。

例1］プロジェクトに時間制約が多く、レビューする時間がとれない場合
フォーマルインスペクションまたは2パーソン・インスペクションで、フェーズの立ち上がり時期にレビューします。1回目は、開発者が最初の成果物を書きあげた時期にレビューします。仕様書であれば記述（書き方）の属人性を確認し書き方のバラツキを開発の初期に抑えます。2回目は、機能性の欠如や、時間がない時は正常系の仕様に専念しがちなので例外処理の抜けがないかレビューします。このようにしておけば後のフェーズが楽になります。

例2］品質を最優先する場合
特に人の命が関わっているようなソフトには、Nフォールド・レビュー(※2)を使います。人数と回数を複数に分けてレビューしていきます。

例3］プロジェクトメンバの技術力の差が大きい・経験差がばらつく場合
例えば初めてレビューに参加して何をチェックしてよいかわからない場合、ベテランと二人一組で観点を教えてもらいながら同一の対象をレビューします。2パーソン・インスペクションまたはウォークスルーで、ベテランが持っている見方・考え方を継承していきます。

例4］集合レビューができない場合
パスアラウンドでレビューします。対象の仕様書をまず最初の人がチェックした結果を記入して、次の人に渡します。次の人は、その前の人のレビューした結果の上に更にチェックした結果を記入します。このようにして、どんどんレビューした結果を追記していきます。一番最後の人が腕の立つ人でないと大変ですが、これも前の人の考え方とかが学べるので良いトレーニングとなり技術の継承ができます。

(※2)Nフォールド・レビューとは？（質疑応答から）
多重度を高めてレビューする方法です。多重度には、チーム編成が多重（複数のチーム）、あるいはチームで視点を変えながら複数回レビューを繰り返すといった方法が挙げられます。いずれにしろ冗長性を高めてレビューを実施します。フォーマルメソッドとは違った厳密なルールがあり、トレーニングが必要です。
参考文献
「ピアレビュー」 Karl E. Wiegers 著　日経BP 社

5）5人以上でレビューすると破綻する
5人以上でレビューしてもレビューの結果に対する議論が発散しますし、バグの検出傾向もそれほど変わりません。検出結果も重なりがちとなりますし、いくら上手くリードしても結局何が悪いのかまとめることが大変となります。

4）4つのロール
フォーマル・インスペクションでは、4つの役割を定義しています。レビューをリードする Moderator、仕様書やソースコードを読むReader、仕様の矛盾や結果の指摘をするValidator（検証者）と作成者であるAuthor の4人です。
このレビューの会議でのポイントは作成者であるAuthor に説明させないことです。書かれたことが他の人にどのように解釈されるかを検証するのがレビューですので、読み手であるReader が読み上げます。ここでの一番のキーポイントは「言い換え（paraphrase)」をすることです。Reader は仕様書をただ読み上げるのではなく、仕様書やソースコードに書かれている事を自分の言葉で言い換えます。言い換えることにより作成者の意図とあっているか

確認します。Reader と Author のやり取りと、Validator のチェック結果をもとに作成者が欠陥なのか要確認項目なのか意図の相違なのかを判断していきます。

もう一つのポイントは会議の中では欠陥の検出に集中することです。原因や対策の議論はしません、むしろしてはいけません。ミーティングの時間が無駄に延びてしまいます。

欠陥検出に集中して議論が白熱すると、これも欠陥ではないかとか新たな見方とか出てくることがあり、これをファントム効果と言います。こういった経験をするとレビューが楽しくなってくると思います。

3）仏の顔も三度まで

いろいろなプロジェクトでレビューを実施していくと、ある組織体とかある企業とかで検出されるバグの傾向が似てきます。プロジェクトが扱うソフトウェアの性質や開発メンバのスキルセットの差とはあまり関係なく、組織や企業の教育方法によってバグの傾向が変わってきます。組織や企業の方針や教え方によって間違い方が異なります。

何度指摘してもバグの傾向が変わらない場合には、その組織体そのものに改善するようお願いしていきます。

2）思わず二度見

怪しい言葉であるNGワード（To Do、～のはず、～かも、「？」、といったような言葉）がフェーズの完了時に仕様書やソースコードのコメントに残っている場合は、フェーズ移行は許可できません。仕様書とかソースコードで出会ったこういったNGワードを蓄えておけば、この先のレビューで出てきた場合に必ず検出することができます。そのほか、暫定、対応、とりあえず、一旦、保留、別途、検討、といった言葉がNGワードに該当します。

このようなNGワードがどこで使われているか調べることにより、どういったところができていないのか、どのあたりが問題なのか、どういったところで困っているのか仮説を立てることができます。しかしながら、こういったNGワードが書いてあること自体が悪いわけではありません。記録してあるからこそ、後から直すことができます。担当者がどういったことで苦労しているのか、その実態をマネジメント層が理解できているかということが重要です。

1）一発勝負レビュー

フェーズの最後に８割ぐらいできた段階でまとめて１回でレビューするべきではありません。レビューの対象が多ければ多くなるほど指摘件数が多くなります。指摘に対応する工数が確保できず、次フェーズに繰り越しになってしまいます。

一度に実施したい気持ちもわかりますが、成果物は「こまめに」レビューするべきです。バグの含有率も減少しますし、重大なバグの検出率も上がります。例えば、フェーズの立ち上がり・中間・最終（８割程度完成）の３段階で実施するだけでもバグの傾向を含めて状況が変わります。実施してみると、最初は現場から抵抗がありましたが、実際には現場の負担感も減るようです。フェーズの立ち上がりでバラツキを抑えて、中間で維持されているかチェックするとともに進捗の確認や傾向を分析し、８割程度できた段階で全部通してどうかというように見方も変えてレビューします。

3．これからのレビュー

開発手法がいろいろ進歩していく中で、人間がマニュアルで実施してきたレビューがこの先どう変わっていくのか考えてみたいと思います。

レビューで検出できるバグにはレベルの階層があると考えています。誰でも検出できる誤字脱字といったものから、抜け漏れとか保守性といった難易度の高いものまであります。標準に則っていないといった簡単なものから、表現が不統一とか矛盾しているといった検出しやすいもの、一見して読みにくいとか一意性が欠如しているといった難易度の低いバグがあります。そういった難易度の低いバグを取り除いた後でやっと難易度の高いバグを検出することができるのが実際です。多くのレビューでは、軽微なバグがノイズになって重大なバグを埋もれさせてしまうという面があります。

軽微なバグの抽出に人間の労力を使わずに、そういったところは自動化して検出できるのではと考えています。自動化ツールを使って軽微で作業的なドキュメントの品質検証を実施し、人間の思考をもっと高度な検証に使います。チェックリストに書かれていて人間が実施しなくても一定の作業でできるようなチェック項目は、自動化

していくことができると考えています。夜中にこのようなツールを実行して翌朝には結果が出力されているようになれば、レビューの負担を軽減することができますし、早い段階でバグを撲滅することができるようになります。バグの潜在期間が短いうちにバグを取り去ることができるようになり、スピードアップができるのではないでしょうか。

さてここで、みなさんに問いかけたいと思います。「レビューは作業（頭を使わずに手だけ動かす）でしょうか？」。レビューはメトリクスの計測をしたり分析したり仮説を立てたり、いろいろ試行錯誤するクリエイティブな仕事だと思います。しかし、テストに比べて自由度が高いために、それゆえに扱いの難しさが先行し楽しさから遠ざかっているのではないかと思います。本来レビューは辛いものである必要はなく、また決して堅苦しいものではありません。人をあざ笑うものではなく、バグがどこからきたのかこのバグは何者なのか、バグを楽しんでもらいたいと思います。

もう１つ。「レビューを楽しいものにしていないのは誰でしょう？」答えは敢えてここでは言いませんので、みなさん自身で考えてみて下さい。レビューで人に何か言われたから面白くなくなってしまったのであれば、レビューする側になった時には、そういう風に思わせないようにして下さい。決してレビューは不必要な手法ではないので、レビューで見てもらいたいとまた思うように、開発のように面白さ・楽しさをどこかで感じてもらいたいと思います。

レビューを失敗するコツを話してきましたが、最後に、レビューをうまくやるコツについてお話しします。成果物を見ているだけではレビューが上手くいくとは思っていません。バグそのものを突き詰めていけば、それは人が作り込むものです。レビューでは、その人自身とか、その人を取り巻く環境に目を向けて欲しいと思います。そういうところにレビューで仮説を立てるヒントがあると思います。