快速 脱!休日出勤 デバッグ技法

.NETアプリケーション障害解析の極意

第8回 ソースコードレビュー －UT工程－

株式会社NTTデータ
飯山 教史
IIYAMA, Takashi

テストの前にやるべきこと

この連載では、ここまで開発工程に沿ってコーディングからデバッグまで説明を進めてきた。「デバッグが終了すれば次はテスト」である。開発したコードをすぐにテストしたい気持ちはわかるが、その前に「ソースコードレビュー」が必要である。昔と異なりプログラムの実行にそれほどコストがかからなくなった現在では、ソースコードレビューの重要さが減り「面倒くさい」といった理由から実施されること自体少なくなってきている。

しかし、ソースコードレビューをしなかったりレビュー時に手を抜いたりすると、バグが後工程にすり抜けるためバグ解決のコストは増加してしまう。後工程になればなるほどバグの修正コストが増大するからだ。このためソースコードレビューはその後行なわれるテスト同様、“必ず”実施しなくてはならない作業なのである。

そこで今回は、軽視されがちだが重要な作業であるソースコードレビューを取り上げ、開発者の視点に立ってどのようにレビューを行なうべきかを解説する。

レベル >>> Level
1 2 3 4 5

言語 >>> Language
- Visual Basic

ツール >>> Tool
- Visual Studio 2005 Team Edition for Software Testers

なぜソースコードレビューが必要か？

冒頭でも書いたように、ソースコードレビューは面倒な作業である。コードを書いた開発者以外の誰かにコードを読んでもらうので、当然の結果としてより多くの時間とコストがかかるようになる。そこまでして、なぜソースコードレビューを実施しなければならないのだろうか？　個人的な経験からいえば、レビューを実施する理由として以下のメリットが挙げられる。

①手戻りを減らす
②後工程を楽にする
③知識を横展開する

まず①だが、レビューにより第三者の目でコードがチェックされるので、ベストプラクティスではない実装方法や仕様どおりではない実装部分がないかどうかが検討される。その結果、問題が顕在化する前の段階で指摘され、成果物（この場合はコード）の後工程になってからの手戻りが減る。開発者なら後工程になればなるほどコードの修正作業のコストが増加することがわかってもらえるだろう。レビューにより手戻りの数が減るので、結果として後工程のコストは削減される可能性が高いのである。

次の②は①と関連している。冒頭でも触れたが、レビューにより後工程へ

とすり抜けるバグの数が減少する。後工程になると、バグが検出されたときに実行されるプログラムの数が増えるので、検出されたバグの原因特定が困難になるのが容易に想像できるだろう。レビューにより後工程で対処するバグの数が減るので、トータルでみた場合、後工程の作業量が減るのである。

③は即効性はないが重要なポイントである。なぜ開発者はバグのあるコードを書いてしまうのか？　それはそのコードにバグがあることを理解していないからだ。つまりコードを書いた人の知識ではそのコードが問題を起こすこと、あるいは間違っていることがわからないのである。そのためコードの記述者が問題を起こすコードであることを理解しないと、ずっと同じミスを繰り返してしまう。そこでレビューでより多くの人にコードを見てもらえば、自分が書いたコードの問題がわかり、レビューア全員のノウハウを吸収できるのである。

何度も繰り返すが、レビューを行なえば稼動時間およびコストは増大する。しかしそれが後にもたらす影響は、そこで消費された稼動／コストを十分にカバーできることも忘れてはならない。一見すると面倒なレビューも大局的に見れば全体の作業量を減らし、後工程で楽ができるようになる。積極的にソースコードレビューに取り組んでもらいたい。

ソースコードレビューの進め方

では、どのようにソースコードレビューを行なえばよいのだろうか？　そこで、以降では、レビューの参加者と進め方についてそれぞれ説明する。

開発中によく行なわれる有識者への質問といった非公式なものであればそれほどかしこまってやる必要はないが、システム開発の正式なタスクとして実施するのであれば、以下のように進めてもらいたい。

●参加者

司会者

レビュー全体の進捗を管理すると共に、レビュー対象のコードの準備やレビュー開催の周知などソースコードレビュー全体の作業管理を行なう。ソースコードレビューで、より多くのバグが指摘されるようにレビューを運営しなくてはならない。技術的にはレビュー対象のコードの専門家である必要はない。

レビューア

レビュー対象のコードの専門家レベルの知識を有し、バグを指摘する。レビュー対象のコードを書いた開発者以外の開発者、もしくは上級のアーキテクトレベルの開発者が担当する。

書記

レビューで指摘されたバグを記録する。正確に指摘内容を記録するため、別作業と兼務してはならない。

開発者

ソースコードレビューのコードを書いた開発者。レビュー中の議論で、コードの意図や背景などを説明し、レビューアと議論を行なう。

●進め方

①計画

司会者がレビューを行なう日時、場所、レビュー対象のコードを指定する。

②準備

司会者は以下の準備を行なう。

・レビュー対象コードの閲覧方法を決める
・設計書などの資料をレビューアに渡す
・（もしテンプレートがあれば）チェックシートを作成し、配布する

レビューアは以下の準備を行なう。

・設計書を読み、仕様を理解する。
・レビュー対象コードをレビューし、結果を記録する。

③レビュー実施

司会者がレビュー作業を取りまとめ、議論を進める。

書記は指摘されたエラーを記録する。

レビューアと開発者はレビュー内容について議論を行なう。このとき、解決策まで指摘すると議論が発散する可能性が高くなるので、あくまでもバグの指摘／検出に集中すること。

④レポート作成

書記はレビューで指摘されたバグをすべて記録したレポートをレビュー直後に作成し、司会者へ提出する。

司会者はレポートをレビューの関連