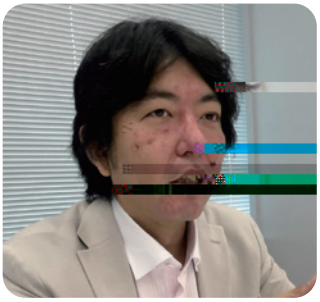
1万8000件のエラー検出で発想を変えた

ソフトウェア品質向上のために「見える化」が大切な理由

ソースコードの品質向上は時代を問わず開発者にとって大きな課題だ。さまざまな手法やメトリクス、ツールが提供されてきたが、選定、導入はどのように行えばいいのだろうか。「奇跡の更改」と称されたプロジェクトを率いた担当者に聞いた。

NTT DATA



NTTデータ 公共・社会基盤事業推進部
茂呂範氏

プロジェクトの問題化を未然に防ぐには「品質の見える化」が不可欠

ソフトウェア品質の高さを強みに、公共部門をはじめ、さまざまな分野でエンタープライズシステムの開発、構築や運用を支援してきた NTT データ。同社は 2018 年、ソフトウェアのさらなる品質向上や上流段階からの品質確保を目指して、静的解析ツールを活用したソースコードの客観的、定量的な評価に取り組み始めた。

その背景について、NTT データ 公共・社会基盤事業推進部の茂呂範氏は次のように話す。「近年、2 ～ 3 年でフレームワークやミドルウェア、フロントエンドの流行が著しく変化する中で、スピーディーな開発と高い品質の確保の両方が求められるようになってきており、開発プロジェクトの難易度が非常に高まっています」

茂呂氏は、自身で大型プロジェクトを率いる傍ら、社内での高難易度プロジェクトの支援もたびたび経験してきた。そのたびに痛感したのは、慢性的なエンジニア不足だ。特に、ソースコードレビューで難易度の高いバグを検出したりシステムアーキテクチャ全体を設計したりできるスキルの高いエンジニアが不足していた。

その状況下でソフトウェア品質を高めるには、まずは「見える化」が必要だ――茂呂氏は高難易度プロジェクト支援の経験を通じてそう考えるようになった。問題が起きてから駆け付けるのではなく、もっと早い段階で品質を評価し「この案件は注意が必要」といった予測ができれば、効果的に対処できる。「限られた優秀なエンジニアをどのプロジェクトに重点的に配置するか」というマネジメントの観点からも、品質の「見える化」が有効ではないかと考えたのだ。

では、具体的にどの段階で品質を見える化すればいいのだろうか。

「私の経験則から言うと、結合テストで問題が見えてきます。その前のコーディングと単体テストをアウトソースしているからです。アウトソースしていると、どうしても品質が見えなくなってしまいます。いろいろなところで出来上がったものをつなげて試験を始めてみると『ここも動かない、そこも動かない』と初めて問題が判明するのです。テストより前の段階で品質を俯瞰（ふかん）できれば、このような問題の低減にもつながるはずです」（茂呂氏）

複数の静的解析ツールを横並びで評価、最も多く故障を見つけたのは……

2016 年、茂呂氏はある高難易度プロジェクトの支援要員として携わることになった。そこで「品質を上げるためにできることは全てやれ」というミッションを与えられたことを機に、以前から考えていた静的解析ツールの導入による「品質の見える化」を実施することにした。

まず試したのはオープンソースソフトウェア（OSS）の「SonarQube」だ。過去に静的分析 OSS の「FindBugs」を利用した際の「チェック不要なコードが問題として検出した。

「昔のツテをたどって、10 案件のソースコードを『ユニットテスト完了後』『結合試験完了後』『サービス開始後』の 3 パターンで入手し、各ツールで検出した故障とプロジェクト中の試験で検出した故障をひも付けました。その結果、われわれの試験で検出した故障を一番効率的かつ多く見つけたのが『Coverity』でした」（茂呂氏）

Coverity を採用した決め手の一つは、ユニットテストで検出できるプログラミングエラーはもちろん、コピー＆ペースト時の修正ミスなどの検出を通じて、結合テストで初めて見つかるような業務ロジックのエラーを発見できたことだ。将来的な全社展開を考えると、C 言語や Java はもちろん、Python など幅広い言語に対応していることもポイントだった。

PoCで実証、プログラム品質向上に寄与したCoverityの費用対効果

一方で、大規模な更改プロジェクトでの導入となると、費用対効果を示さなければ周囲の納得は得にくい。そこで茂呂氏は「C 言語で書かれたプログラムの品質をどれだけ高められるか」を示すため、PoC（概念実証）を実施することにした。

「C 言語で書かれたプログラムを新しい環境に移行するとき最も恐ろしいのが、CPU アーキテクチャの変更や OS のバージョンアップによってメモリ回りの挙動が変わり、これまで動作していたプログラムが止まってしまうことだ。この種の問題をどれくらい検出できるのか、Coverity で試すことにした」（茂呂氏）

10 個のバッチプログラムを対象に PoC を実施した結果、10 個のうち 6 個のバッチプログラムがメモリ違反で停止してしまった。Coverity で解析したところ、メモリ違反の原因となっている箇所を全て検出できたという。「対象プログラムが 10 個だけなら人手による試験でも見つけられますが、これが 100 個、1000 個と増えていっても迅速に検出するには、Coverity のようなツールが必要です」と費用対効果を示し、導入にこぎ着けた。

茂呂氏が率いる更改プロジェクトの開発環境は、パッケージリポジトリ管理ソフトウェア「Nexus」や開発者間のコミュニケーショ

SYNOPSYS®

https://www.synopsys.com/jp/software

日本シノプシス合同会社ソフトウェア インテグリティグループ
〒158-0094 東京都世田谷区玉川2-21-1二子玉川ライズオフィス
TEL: 03-6746-3600 Email: sig-japan@synopsys.com