デバッグ/仕様変更/バージョンアップ…
オープン・ソースの大規模開発用環境で
不毛な繰り返しテストとはオサラバ

マイコンでトライ!

C/C++自動テスト・ツールで高品質プログラムをつくる

第3回 単体テスト&静的解析の自動化

会田 圭司

今回は，過去2回の連載にて紹介した単体テストと静的解析を，継続的インテグレーションに組み込んで実行するコツを紹介します．

● おさらい…継続的インテグレーションCI

現在のソフトウェア開発では，差分・保守・派生開発が全体の50%を占めます[1]．しかも，最近は多数のソフトウェアを並行して開発することが多くなっています．限られた時間の中でソフトウェア開発を円滑に進め品質を保つためのしくみが継続的インテグレーション(CI；Continuous Integration)です．

● ビルドやテストを自動で行えば開発効率がよい

インテグレーションとはビルドや単体テストなどの作業です．CIの目的は，バグなどの問題を早期に検出し，フィードバックのサイクルを短くすることで，ソフトウェア開発の品質と生産性を向上させることです．この手法はアジャイル・ソフトウェア開発方法論の一つであるXP(エクストリーム・プログラミング)から生まれました．

継続的インテグレーションでは，**図1**のようにビルドやテストの実施→検出された問題のフィードバック→問題への対応を一連の流れとして行える，専用のソフトウェアを利用します．代表的なソフトウェアとしてはオープン・ソースのJenkinsがあるほか，Team Foundation Server(マイクロソフト)など商用の構成管理ツールでも，サポートされつつあります．

継続的インテグレーションを実施する際には，

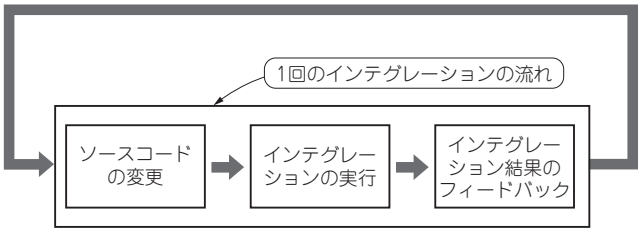

図1 継続的インテグレーションのサイクル

(1)「いつ」インテグレーションを実施するのか？
(2)「何を」インテグレーション対象とするのか？

のポイントを押さえます．

その1…単体テストの自動化

●単体テストを簡単におさらい

連載第1回(2013年2月号)にて取り上げた単体テストの技法に，ブラックボックス・テストとホワイトボックス・テストがありました．それぞれの特徴を振り返ってみましょう．

- ブラックボックス・テスト…プログラムが仕様通りに動作するかを確認する
- ホワイトボックス・テスト…内部構造に着目し網羅的にプログラムの動作を確認する

これらの技法はお互いを補完する関係にあるため，両方の技法による単体テストを行うことが，高品質なプログラムをつくる近道です．一般には以下のような手順で単体テストを進めます．

(1) ブラックボックス・テストを実施する
(2) ブラックボックス・テストのカバレッジを確認する
(3) ホワイトボックス・テストにてブラックボックス・テストでは実施されていない命令や分岐をテストする
(4) ホワイトボックス・テストのテスト・ケースをレビューする

● 効率よく単体テストしたいなら対象をひたすら細かく！

単体テストの**図2**のような手順で実行します．単体テストのテスト対象は，実装した関数(ソース・コード)のみで行うことが理想です．テスト対象をプログラム内で小さな単位に分割し，細かくテストを実施します．他の開発者が修正したコードと一緒にテストを実施してしまっては，テ