デブサミ2006 Developers Summit

.NET開発における品質改善
– Visual Studio 2005 Team Systemによる継続的インテグレーション –

10-C-5

黒石高広
NAgile.com管理人
http://www.nagile.com/

本セッション目的

- (.NET開発において)ソフトウェアの品質を改善／向上させるために必要なことを解説します
  - 内容的には.NETを知らない方でも大丈夫です
- 継続的インテグレーションの考え方とどのように実践すればよいか解説します

Agenda

- 品質って何？
- 品質の測定
- テストって何?
- テスト戦略重要
- 継続的インテグレーション
- テスト駆動開発
- VSTSの利用
- まずは基礎をしっかりと固める
- 開発者とテスト担当者の協調
- テストのある生活をはじめませんか？
- まとめ
- Appendix.

品質って何？

- 品質属性
  - 可用性,効率性,柔軟性(拡張性),完全性(安全性),相互接続性,保守性,移植性,信頼性,再利用性,堅牢性,試験性(検証可能性),使用性(ユーザビリティ) …
    - 詳細はAppendix. 参考情報2を参照
- 一言で品質と言っても多くの指標が存在する
- 品質属性にはトレードオフがある

品質の測定

- ソフトウェア・メトリクス（評価指標）
  - ソフトウェアの品質をある指標から測定し、可視化するツール
  - 結合度や凝集度の測定
- 正しく動作するソフトウェアの品質を測定することで意味のあるデータを採取できる

テストって何？

- 一般的なテスト = 検査
- テスティング（テスト）の定義
  - 「テスティングは、テスト集合を入力することによってプログラムが起こすエラーを可視化するための行為である」（参考情報1）

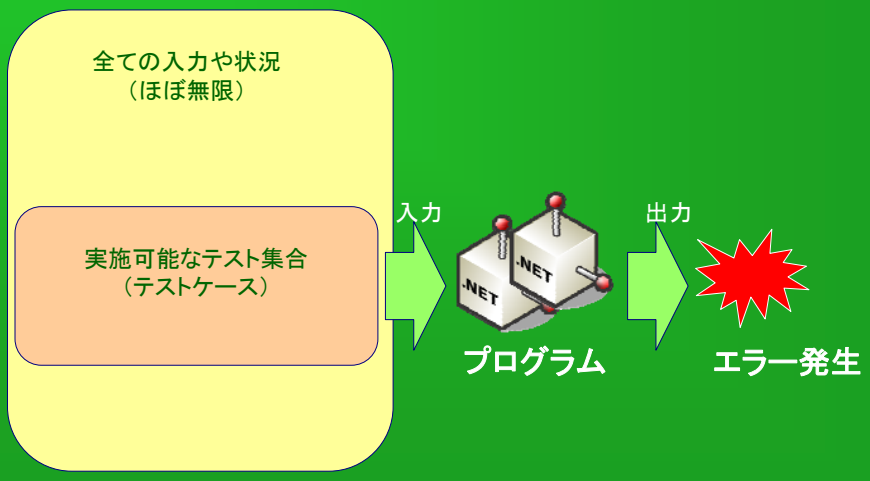
テストの種類

- ユニットテスト（単体テスト）
- 回帰テスト
- 統合テスト（結合テスト）
- システムテスト
- 受け入れ／能力認定テスト
- 据付けテスト
- パフォーマンステスト
- …など

テストに関する2つの事実

- テストにはコストがかかる
- テストでバグが完全に無いことは証明できない

バグが完全に無いことは証明できない



テスト戦略重要

- 我々は様々な制約の中でソフトウェア開発を行っています
  - 時間の制約
  - 人員の制約
  - 予算の制約
- テストを効率的、効果的に実施すること
- まずは機能面でのテストをきっちりと行うこと
  - まともに動かないソフトウェアの品質を測定してもあまり意味がない

出口の検査を厳しくするだけでは駄目

- ソフトウェア開発は同じ商品を大量に製造するものではない
  - 出口で不良品をはじくという考え方では品質は改善されない
- 問題を素早く発見し、早期解決することが最も重要
  - 開発サイクルの中に品質改善の仕組みを組み込む
- そこで、「継続的インテグレーション」の登場

継続的インテグレーション

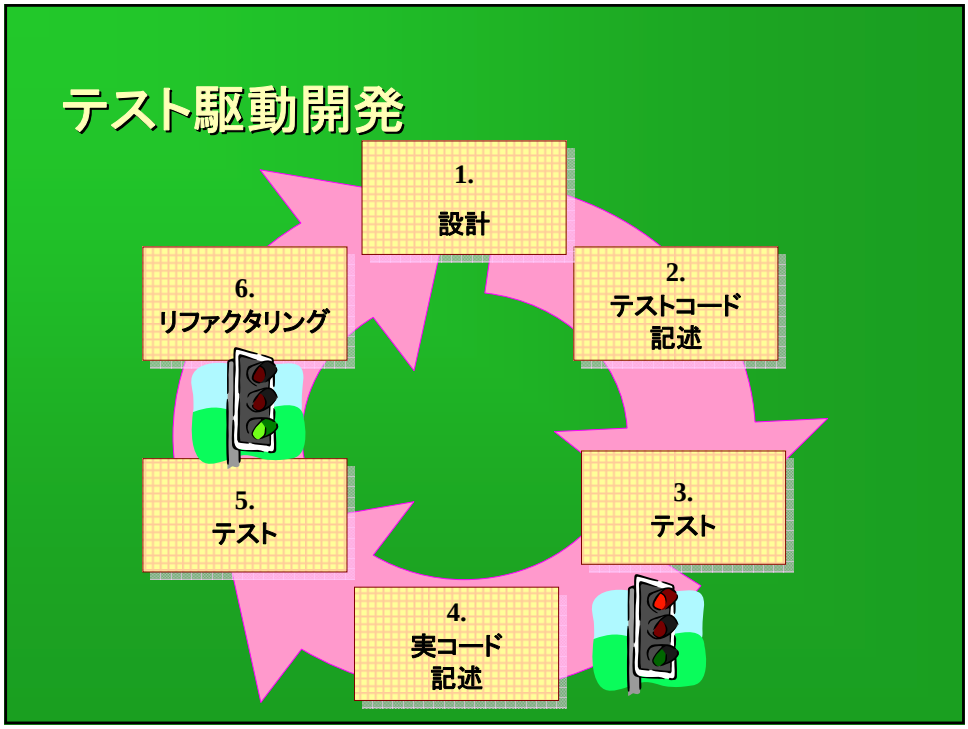
- インテグレーション＝統合テスト
  - 結合テストとも呼ばれる
- 継続的に毎日、何回も統合テストを実施すること
  - Extreme Programmingのプラクティスの1つ
- 日次ビルドからが取り組みやすい

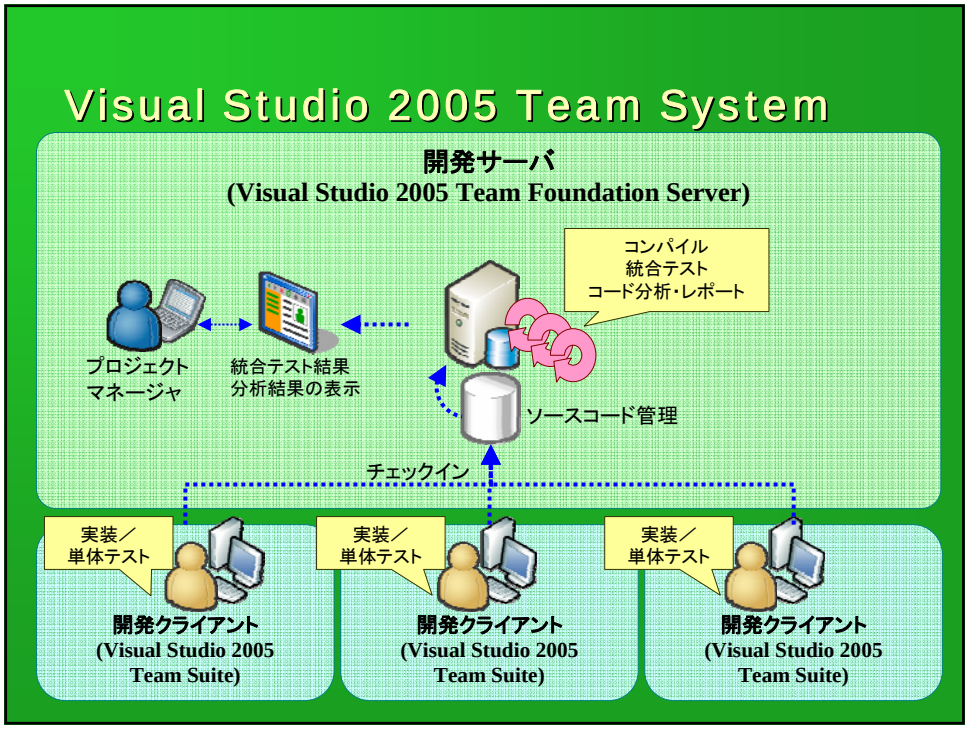
何故、継続的インテグレーション？

- 多くのソフトウェア開発は複数の開発者が同時並行の作業で開発を行っている
- 継続的インテグレーションを効率よく行うには？
  - テストの自動化・可視化
  - テスト駆動開発と組み合わせることで効果大

テスト駆動開発（テストファースト）

- テストファーストはXP(Extreme Programming)のプラクティスのひとつ
- テストコードに記述した振る舞い（仕様）どおりにプログラムが動作するか確認しながら開発を行う
  - テストコード = 動く仕様書、実行可能な仕様書
  - 最近では、「Behaviour Driven Development」という言葉も生まれている（参考情報2）
  - 単体テストの側面も持つ
- テストの自動化・可視化


テスト駆動開発
1.
設計
2.
テストコード
記述
3.
テスト
4.
実コード
記述
5.
テスト
6.
リファクタリング


Visual Studio 2005 Team System
開発サーバ
(Visual Studio 2005 Team Foundation Server)
コンパイル
統合テスト
コード分析・レポート
プロジェクト
マネージャ
統合テスト結果
分析結果の表示
ソースコード管理
チェックイン
実装／
単体テスト
開発クライアント
(Visual Studio 2005
Team Suite)
実装／
単体テスト
開発クライアント
(Visual Studio 2005
Team Suite)
実装／
単体テスト
開発クライアント
(Visual Studio 2005
Team Suite)

Visual Studio 2005 Team System

- 開発クライアント(Visual Studio 2005 Team Suites)から開発サーバや分析機能までを含む包括的な開発環境
  - ソースコードバージョン管理
  - タスク管理
  - ドキュメント管理
  - 分析・レポート
- 品質向上のための様々な機能
  - テスト駆動開発や継続的インテグレーションもサポート
  - コーディング規約の検証やパフォーマンスツールなど
- 関連セッション
  - 【9-D-6】Team Foundation Server によるチーム開発コトハジメ

継続的インテグレーションDemo

- Visual Studio 2005 Team Systemを利用した継続的インテグレーション

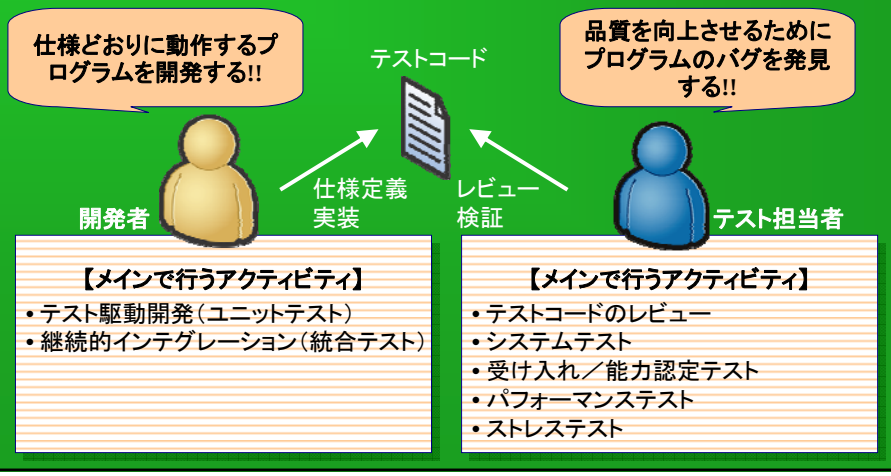
まずは基礎をしっかりと固める

- テスト駆動開発・継続的インテグレーションはソフトウェアの機能が仕様どおりに正しく動作することを確認するためのもの
- 仕様どおりに正しく動作しないソフトウェアの品質をとやかく言ってもあまり意味がない
- その後、パフォーマンス、セキュリティ、信頼性などに関するテストを加え、さらに品質を向上させる

テストコードを資産として残す

開発者とテスト担当者の協調

- 開発者とテスト担当者間の“良い”協調関係構築



テストのある生活をはじめませんか？

- プログラムの修正を行った後、テストコードを実行しプログラムが仕様どおりに動いていることを確認
- 正月休みなどの長い休み明けから戻ってきた時、テストコードを実行しプログラムが仕様どおりに動いていることを確認
- 顧客の環境へプログラムを持って行った時、テストコードを実行し異なる環境でもプログラムが仕様どおりに動いていることを確認

- テストのある生活とは、安心のある生活（ソフトウェア開発）です

まとめ

- ソフトウェアの品質を改善・向上させるには開発サイクルの中にいかに品質を向上させる仕組みを組み込めるかが鍵
- テスト駆動開発と継続的インテグレーションを行うことで、開発サイクルの中に品質向上の仕組みを組み込むことが出来ます
- 更にコードレビューやパフォーマンステストなどと組み合わせて品質を向上させましょう

Cruise Control .NET

- 継続的インテグレーションを実践するためのオープンソースのツール
  - 米国 ThoghtWorks社が開発
  - 最新版：1.0
- 利用方法やサンプルコード、日本語化モジュールはNAgile.comで公開中
  - http://www.nagile.com

Appendix. 品質属性

- 可用性
  - システムを利用することができる計画された動作可能時間の指標
- 効率性
  - システムがプロセッサやディスク容量、ネットワークなどをどのくらい効率的に使用するかを示す指標
- 柔軟性（拡張性）
  - システムがどのくらい容易に新機能を追加可能かを示す指標、拡張性と同意
- 完全性（安全性）
  - システムがどのくらい完全に（安全に）保たれているかを示す指標、安全性、セキュリティ要求と同意
- 相互接続性
  - システムが他システムとどのくらい容易にデータやサービスなどを連携できるかを示す指標
- 保守性
  - システム運用中に欠陥の修正やソフトウェアの変更をどのくらい容易に行うことができるかを示す指標

Appendix. 品質属性（続き）

- 移植性
  - システムをある稼働環境から別の稼働環境へどのくらい容易に移行させることができるかを示す指標
- 信頼性
  - システムの操作がどのくらいの割合で正常に完了したかなどを示す指標、可用性と関連
- 再利用性
  - システムがどのくらいサービス化（モジュール化）され他システムから再利用が可能かを示す指標
- 堅牢性
  - システムが無効な入力などの予想外の稼動条件でどのくらい正しく稼動し続けることができるかを示す指標
- 試験性（検証可能性）
  - システムのコンポーネントやソフトウェアの欠陥をどのくらい容易に検出することができるかを示す指標
- 使用性（ユーザビリティ）
  - システムの使いやすさを示す指標

Appendix. 品質属性トレードオフ

	可用性	効率性	柔軟性	完全性	相互接続性	保守性	移植性	信頼性	再利用性	堅牢性	試験性	使用性	備考
可用性								▲		▲			
効率性			▼		▼	▼	▼	▼		▼	▼	▼	
柔軟性		▼		▼		▲	▲	▲			▲		
完全性		▼			▼				▼		▼	▼	
相互接続性		▼	▲	▼			▲						
保守性	▲	▼	▲					▲			▲		
移植性		▼	▲		▲	▼			▲		▲	▼	
信頼性	▲	▼	▲			▲				▲	▲	▲	
再利用性		▼	▲	▼	▲	▲	▲	▼			▲		
堅牢性	▲	▼						▲				▲	
試験性	▲	▼	▲			▲		▲				▲	
使用性		▼								▲	▼		

※ 横軸の属性を高めると他の属性も高まる場合は▲、他の属性が低くなる場合は▼

参考情報

1. 書籍：「ソフトウェア開発へのSWEBOKの適用」
   - 松本 吉弘 (著)
   - 出版社: オーム社
   - ISBN: 4274500373 (2005/05)
2. 書籍：ソフトウェア要求
   - Karl.E.Wiegers (著); 渡部 洋子 (翻訳)
   - 出版社: 日経BP社 ;
   - ISBN: 4891003545 ; (2003/07/10)
3. Web : Behaviour Driven Development
   - http://log.giantech.jp/BDDIntro-ja.html
     - A NEW LOOK AT TEST-DRIVEN DEVELOPMENT
     - Author: Dave Astels; 永和システム　懸田さん和訳
4. Web : NAgile.com
   - http://www.nagile.com/
     - 継続的インテグレーション、CCNETに関する資料など

powered by