テストスケジューリング手法による テスト＆デバッグの効率化に関する試み

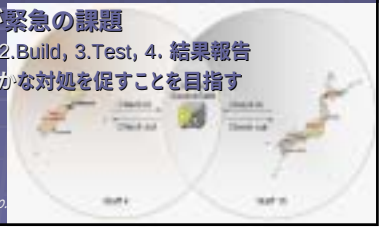
井口真一, 中尾憲二　(株)フォーラムエイト

Agenda

- 目的
  - デグレード対策
- 開発プロジェクトの概要
  - 開発プロジェクト
- 対策
  - デバッグ系の構築
  - テストの履歴
- まとめ
  - 効果, 反省と展開

背景と目的

- 弊社の製品UC-win/Series Frame (3D)開発問題
  - JPN staff ← 時差3，4時間 → NZL staff
- 意思伝達不足による詳細仕様の相違等
  - デグレード頻発
- デグレード対策が緊急の課題
  - 1.最新File取得, 2.Build, 3.Test, 4. 結果報告
  - 問題発生→速やかな対処を促すことを目指す



開発プロジェクトの紹介

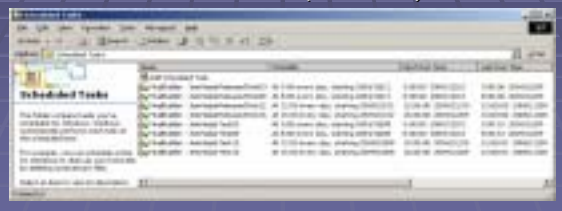
- 3次元骨組の構造解析
  - 2003年1月リリース
  - Ver.1.05.00開発中
- 開発スタッフ17名
  - うち4名NZ勤務

対策　製品コア部から最新版へ

ステップ	テスト	テストツール	対象テスター
第1段階	単体	DUnit	各自
	総合	---	全スタッフ
第2段階 Ver.1.01.02~	単体	DUnit	各自
	総合	WinRunner	ﾃｽﾄｸﾞﾙｰﾌﾟ & 全スタッフ
第3段階 Ver.1.02.00~	単体	DUnit	FinalBuilder
	総合	WinRunner	FinalBuilder

テストスケジューリング

- [タスク]でスケジューリング
- 毎日3、8、12、16時の4回実行(単体テスト)
- 毎日0、5、18時の3回実行(総合テスト)



デバッグ

- 実際の，Webブラウザによるバグの確認
  - Finial Builder 1
    - コンパイル
    - 単体
  - Finial Builder 2
    - テストスクリプト
  - BTS
    - バグあり　修正済み　確認済み

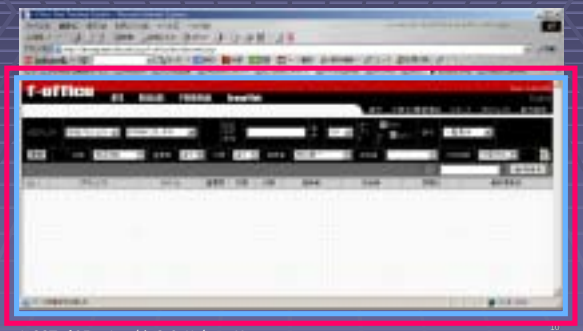
まとめ

- 効果
  - 統括的なテスト系の構築
  - 危機的バグの早急な除去
  - テスト作業の一元化による総作業時間の短縮
  - テスト項目は増加させることが可能
  - 自動テストの結果であるため，責任の所在が明確
- 反省
  - ユニットテスト，総合テストの更新作業

付録

- 以下リンク資料関係
- FinalBuilder:
  - http://www.atozedsoftware.com/finalbuilder/
- DUnit:
  - http://dunit.sourceforge.net/
- WinRunner:
  - http://www.mercury.co.jp/products/winrunner/

Bug Tracking System

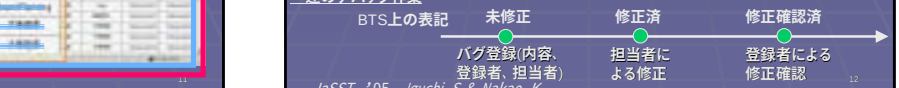


Bug Tracking System

対策　製品コア部から最新版へ

- 初期段階
  - ユニットテスト
    - 計算部　数値テスト
  - 手動テスト
    - 現行製品のコア部
      - 単純な計算モデルの作成と数値表示
    - Bug Tracking System (BTS)
      - バグや要求事項を適宜登録・状態と履歴を管理

一連のデバッグ作業



対策　製品の肥大化

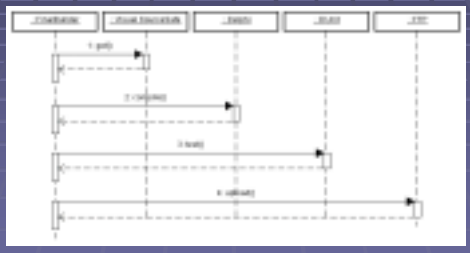
- 第2段階
  - バグ多発
- 自動テストの導入
  - ユニットテスト（各自）
    - 計算
  - 総合テスト
    - ベータ版の各ビルドに対し
      - 全体のコントロール，計算モデルを用いた数値テスト
        新規ビルド版に追いつけない

対策

- 第3段階
  - さらに肥大化し，デグレード危機
- スケジューリングでほぼ統括的な自動化
  - 保守以外のすべての作業をスケジューラーで管理
    → --- スケジューリングテストへ ---
    - 第3者的にテストを実行
    - 各4，5時間毎の単体･総合テスト

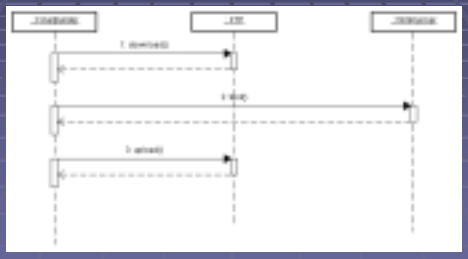
テスト

- 単体テストシーケンス FinalBuilder_1



テスト

- 総合テストシーケンス FinalBuilder_2



第3段階