コンピュータ概論B - ソフトウェアを中心に -

#9 バグ・ソフトウェア工学

Yutaka Yasuda

ソフトウェア開発

- 願望

  正しく動作するソフトウェアをなるべく容易に安定して作成し、利用したい

- 問題

  バグ：正しく動作しない

  要求仕様を満たせない

  開発計画の見積もり失敗

“Software Factories に関する陳述”

- Jack Greenfield, Microsoft corp., July 2004
- The Standish Group. The Chaos Report. (1994) によれば、

  米国では毎年17 万 5000 のプロジェクトに約 2500 億ドルのソフトウェア開発費が投じられる

  僅か16%がスケジュールと予算の枠内に収まった

  31% が品質の問題によりキャンセル（810 億ドル相当）

  53% が予算超過（平均 189%）し、損失 590 億ドル

- ソフトウェア開発産業は工業化の段階としては未成熟と言わざるを得ない

ソフトウェアの安全性

- 事例

  みずほ銀行システムトラブル（2002/4, バグ）

  JR 東日本 Suica 改札トラブル（2006/12, バグ）

  NTT東西 ひかり電話接続不良（2006/9,10, 複合）

  ANA 運行システムトラブル（2007/5, バグ）

- ウィルス問題（各社, セキュリティホール等）

Software Update

- 事例：Microsoft のソフトウェア・アップデート

  ほとんどがトラブルまたは安全上の対策（バグ）

  減る気配なし

Software Update　(ex. Microsoft)

MS07-001 : Office の重要な更新 (921585) (2007/01/10)Office Update
MS07-002 : Excel の重要な更新 (927198) (2007/01/10)Office Update
MS07-003 : Outlook の重要な更新 (925938) (2007/01/10)Office Update
MS07-004 : Windows の重要な更新 (929969) (2007/01/10)Microsoft Update
MS07-005 : Windows の重要な更新 (923723) (2007/02/14)Microsoft Update
MS07-006 : Windows の重要な更新 (928255) (2007/02/14)Microsoft Update
MS07-007 : Windows の重要な更新 (927802) (2007/02/14)Microsoft Update
MS07-008 : Windows の重要な更新 (928843) (2007/02/14)Microsoft Update
MS07-009 : Windows の重要な更新 (927779) (2007/02/14)Microsoft Update
MS07-010 : マルウェア対策エンジンの重要な更新 (932135) (2007/02/14)
MS07-011 : Windows の重要な更新 (926436) (2007/02/14)Microsoft Update
MS07-012 : Windows の重要な更新 (924667) (2007/02/14)Microsoft Update
MS07-013 : Windows の重要な更新 (918118) (2007/02/14)Microsoft Update
MS07-014 : Word の重要な更新 (929434) (2007/02/14)Office Update
MS07-015 : Office の重要な更新 (932554) (2007/02/14)Office Update
MS07-016 : Internet Explorer の重要な更新 (928090) (2007/02/14)Microsoft Update

つづく… (資料は2007/3 末現在)

http://www.microsoft.com/japan/security/bulletins/visual_list.mspx
Microsoft, Inc. 絵で見るセキュリティ情報

バグ

- ソフトウェア障害の 3/4 が既知の初歩的なプログラミングの誤りによる - Eugene H. Spafford
- なぜバグは発生するのか？無くならないのか？

  「注意深く設計すれば良かった」だけか？

- コンピュータが本来持つ構造問題として理解していきましょう

ソフトウェアとバグの関係

バグ

- プログラムに含まれる「間違い」
- データは意味をもたない
- コンピュータは意味を扱わない
- バグ

  無意味な（矛盾した）処理でも指示通り動作する

  例えば「金利と残高を加算する」ところを間違えて
  「年齢と金額を加算」させてしまうかもしれない

- なぜ間違えた指示が最後まで？

プログラミング

- 「1から10までの数を足した結果を出せ」

  これはコンピュータにとって「難しすぎる」指示

- 手順の明記

  「Xを 1 から 10 まで変化させ、毎回 Y に繰り込め」

- プログラムとは何か

  目的に対して「何をどう処理するか」を詳述したもの

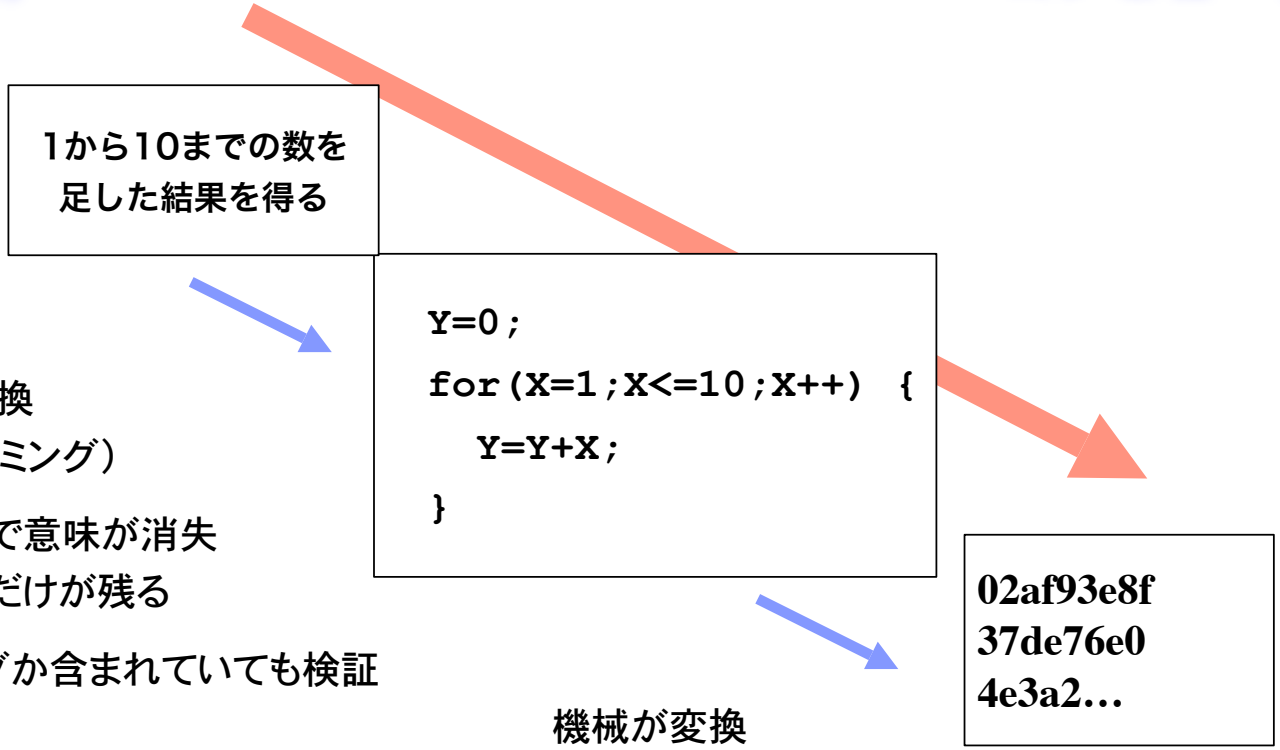
- 意味の消失

  「コンピュータはデータの意味を理解しないのと同様に、プログラムの意味も知らない」
  （ただ手順だけを知っている）

二つの変換過程

人間側

コンピュータ側



人間が変換
（プログラミング）

この時点で意味が消失
して手順だけが残る

つまりバグか含まれていても検証
できない

機械が変換

ここで実行されてはじめてバ
グが見つかる

失われれる意味

- プログラミングの過程で、プログラムから意味は失われる

  そのプログラムの目的を知るのはプログラマだけ

- コンピュータは情報ではなくデータだけを扱う

  データが表現する情報は入出力の前後にいる人間が扱う

- ソフトウェアは作業から意味を抜いた手順だけを扱う

  その意味（内容）は結果を受け取る人間だけが知る

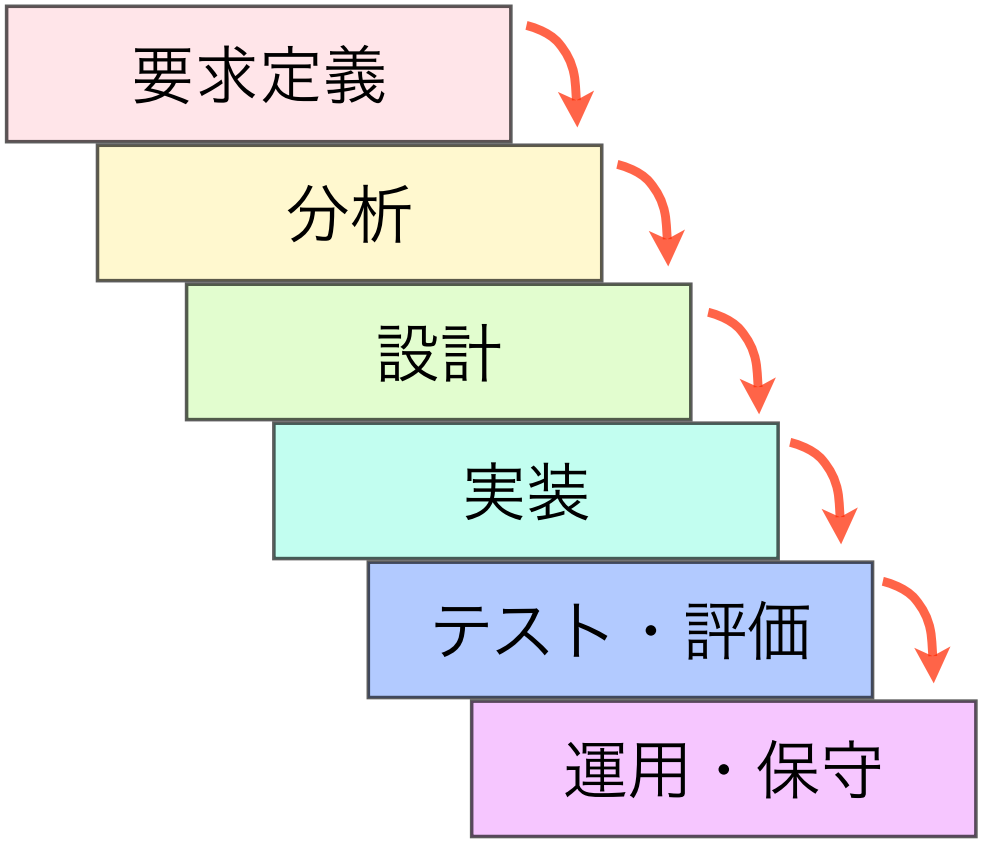
- バグ（意図しないプログラムの振る舞いをひきおこす不具合）が発生する根本的要因のひとつ

大規模システム開発の問題

Brooks に学ぶ

Water Fall model

- 古典的ソフトウェア開発手法
- ユーザの要求からはじめて段階を踏んで開発
- 各工程が完了すると次の行程に進む
- 家を建てる場合を想像せよ



人月の神話

- Brooks Jr., Frederick P.
  "The Mythical Man-Month: Essays on Software Engineering”, 1975
- IBM System 360 開発の経験から
- Water Fall model の限界
- 人月という考え方は適切でない

ブルックスの法則

- 人間と月（工数）は交換可能である、というのは幻想だ
- 遅れているプロジェクトへの要員追加はさらに進捗を遅らせるだけだ
- 優秀なプログラマは平均的なプログラマより１桁以上生産性が高い
- 工数はプログラムの規模の累乗になる
- プログラマがプログラミングとテストに費やす工数は、全工数の半分以下に過ぎない

「人月の神話」における主張

- ソフトウェア開発では作業員間のコミュニケーションの比重が（非常に）大きい
- 人数増はその比重を更に上げる
- 30年経ったいまでも予言として残っている
- エンジニアリングといっても社会工学的な側面であり、技術から離れている

銀の銃弾などない

- 1986年発表
- ソフトウェアの生産性をひとりでに高めるようなプログラミング手法は今後10年間登場しない
- ある程度は今も当てはまる（らしい）

まとめと対策

- なぜトラブル・バグはなくならないか？
- 情報とデータの相違

  コンピュータは情報でなくデータを扱う

  情報処理機械≠自動データ処理機械
- ソフトウェア工学の価値

  ソフトウェア開発に工学的手法を適用したい

  高度で安全なシステムを確実に開発できるように
- 脆弱なソフトウェア基盤からの脱却を

参考：Y2K 問題を思い出せ（or 調べてみよ）

- 坂東俊矢（京都産業大学法務研究科）
  「IT2001 なにが問題か」から

  「どこにあるかも知らないコンピュータの誤動作によるライフラインの断絶に備えて、半信半疑で風呂に水をため、当面の食料品を購入したりもしたが最後まで具体的な情報は消費者には伝わってこなかった」

  「消費者は情報社会が滑稽なほど不完全であって、それが自らの生活に直接の影響を与えるものであることだけは理解できたに違いない」

- 参照：首相官邸「コンピュータ西暦2000年問題」
  http://www.kantei.go.jp/jp/pc2000/

参考：様々なアプローチ

- 新しいプログラミング言語・環境・アイディア

  いかにして抽象化するか、また抽象化されたままプログラミングするか

  オブジェクト指向プログラミング
- XP (eXtreme Programming) / Kent Beck ら

  ペアプログラミング / 頻繁なリリース / 完成という概念を廃し、期限までに顧客の要求を最大限採り入れる

  小さなテストを繰り返し、修正を繰り返す
- より良いプログラミング、システム開発へのアプローチはまだまだ続く