日本と世界のソフトウェア ・テストの状況

ソフトウェアテストシンポジウム2003

2003年 3月 6日
日立ソフトウェア・エンジニアリング株式会社
山浦 恒央
yamaur_t@and.hitachi-sk.co.jp

第 1部 :日本のQAの現状

1.日立ソフトにおける品質データ:

開発フェーズごとの摘出バグのパーセンテージ

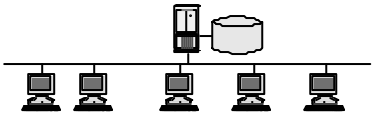
机上 デバッグ	単体 デバッグ	連動 デバッグ	統合 デバッグ	QAチーム のテスト	顧客 サイト
21.5%	35.1%	28.0%	6.1%	9.3%	0.02%

10,000 件のバグのうち、顧客サイトで出るのは 2件。

2.日立ソフトにおける QA方式

特別な方法は使っていない。以下のように 昔からの方法を踏襲。

(1) プログラマは、テスト項目を設計し、テスト項目を作成し、実行する。
(2) ソフトウェア製品のライフサイクルを通じて、バグの統計データを収集する。
(3) プロジェクトマネージャは、バグの統計データを分析し、プロジェクトを管理する。
(4) 設計部 と独立したQAチームが製品をテストする。



(1) プログラマは、テスト項目を設計し、テスト項目を作り、実行

a. プログラマがコーディングを終了すると、テスト項目の設計を開始。

b. テスト項目は 全てドキュメント化する。ドキュメント化をすると、以下のような利点がある。
- テスト項目 の設計 は、仕様を別の視点から見直すことと同じ。この見直しで、プログラム不良だけでなく、仕様関連の重要バグも見つかる。
- テスト自体の 品質を測定できる。
- 同じテストを何回も繰り返し実行できる。
- テストの実行は、別の 担当者にまかせられる(プロジェクトの日程遅延を回復するために投入された 素人エンジニア」にも、テストの実行可能)。

c. テストの自動実行化(テスト・スイート)を目指す。

(1) プログラマは、テスト項目を設計し、テスト項目を作り、実行 (続き)

テスト項目設計のガイドライン

a. 基本は、ソースコード 5 - 15行で テスト項目 1件
- 言語プロセッサ系 : 8-12 行で 1件
- オンライン・ネットワーク系 : 5-10 行で 1件
- バッチ系 :10-15 行で 1件

b. 正常ケース : 60% 以下
異常ケース : 15% 以上
限界・境界 : 10% 以上
環境(別OS等) : 15% 以上

(1) プログラマは、テスト項目を設計し、テスト項目を作り、実行 (続き)

マトリックスによるテスト項目の設計

入場料: 6歳以下 :無料, 7歳～12歳：500円,
13歳～18歳： 800円、 19歳以上： 1,000円

テスト項目 ID		AD001	AD002	AD003	AD004	AD005	AD006	AD007	AD008
年齢	ZZ	X							
	-1		X						
	0			X					
	6				X				
	12					X			
	18						X		
	19							X	
	999								X
入場料	0円			X	X				
	500円					X			
	800円						X		
	1000円							X	
	エラー	X	X						X
分類		異常	境界	正常	正常	正常	正常	正常	異常
第1テスト		5/05	5/05	5/05	5/05	5/05	5/05	5/05	5/05
マシンテスト		6/01	6/01	6/02	6/02	6/02	6/02	6/03	6/03

(1)プログラマは、テスト項目を設計し、テスト項目を作り、実行 (続き)

a. マトリックスによるテスト項目の利点
- 入力条件の組合せを網羅できる。
- 機械的にテスト設計が可能

b. マトリックスによるテスト項目の欠点
- テスト項目数が増える(特に、シンタックス・チェック)。
- 複雑な条件の記述が難しい(自然言語によるテスト項目記述も必要)。

(2) ソフトウェア製品のライフサイクルを通じて、バグの統計データを収集

プログラマは、全てのバグを文書でプロジェクト・マネージャに報告する。

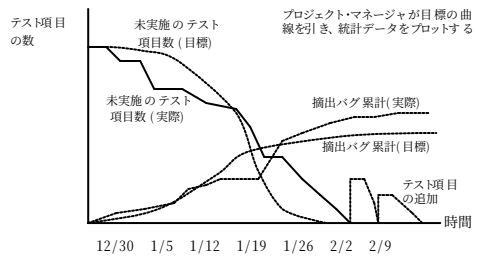
a. 報告開始時期
コーディングが終了し、コンパイル・エラーがなくなった時点(机上デバッグに入る条件が整ったとき)

b. 報告内容
- バグを摘出したとき
ID番号、バグの症状、バグの摘出者、摘出年月日、バグの重要度等

- バグを修正したとき
バグがあったモジュール名、バグの原因、修正者、テスト担当者、修正前コーディング、修正後コーディング等

(3) プロジェクトマネージャによるバグ データ分析

- **品質向上をするための最善の戦略を決定する。**
- **製品のリリース日を予測する。**



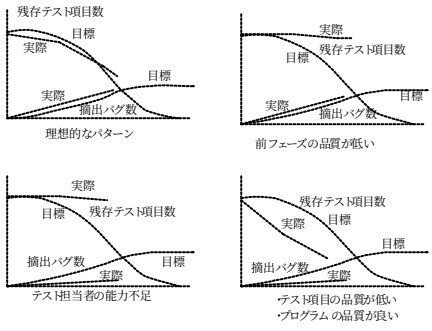
(3) プロジェクトマネージャによるバグ データ分析 (続き)

分析する内容

(1) モジュールごとの不良分析 (90%の不良は、10%のモジュールに集中する傾向にある)
(2) 機能固有の不良解析
(3) 個人ごとの不良分析 (個人の不良作りこみの癖を分析)

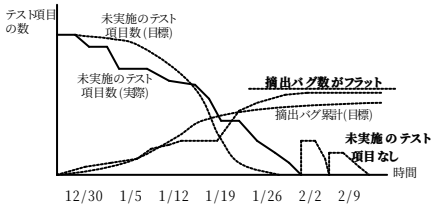
(3) プロジェクトマネージャによるバグ データ分析 (続き)

統計データのパターンによる問題点の分析



(3) プロジェクト・マネージャによるバグ データ分析 (続き)

製品のリリース(QA部への)条件
(1) 未実施のテスト項目がない。
(2) バグの摘出が頭打ちになった。
(3) バグが全件、修正されている。
(4) 48時間耐久テストに合格した。



(4) 独立したQAチームが製品をテスト

社内に、設計部と完全に独立したQA部がある。

Q1: QA部には、何人のQAエンジニアがいるか？
A1: 全エンジニアの約8%がQAエンジニア(4,000人中、330人)

Q2: QAエンジニアは、何をテストするか？
A2: 設計部が作成した物全て(仕様書、設計書、テスト項目、コード等)

Q3: QAエンジニアにどんな権限があるか?
A3: QA部の「合格印」がなければ、製品はリリースできない。QA部は、合格印を押した製品がリリースされると、その後の品質に関し、全責任を負う。

Q4: QAエンジニアは、どのようにテストをするのか？
A4: 顧客の視点でテストを実施(従って、ブラックボックス・テストが中心)。QA部は、設計部とは全く独立に、テスト項目を設計・実施する。

3. 品質のコスト (品質は無料ではない!)

仮定 : 10人のプログラマがいるプロジェクトで、12ヶ月にC言語で
100,000行のソフトウェアを開発する。

(1) 各開発フェーズに要する月数
- 仕様分析・仕様決定 : 2 ヶ月
- 設計 : 3 ヶ月
- コーディング : 2 ヶ月
- デバッグ : 3 ヶ月
- テスト(QA技術者による) : 2 ヶ月

(2) 3 ヶ月のデバッグ・フェーズの詳細
- 100,000行のソフトウェアに10,000 件のテスト項目が必要 (1,000件/プログラマ)
- 10,000件のテスト項目を設計するのに10日 (1日 1人あたり100件)
- 1,000件の机上デバッグに10日 (1日 1人あたり10件)
- 10,000件のマシン・デバッグに40日 (1日 1人あたり25件)

質問: 99.98% バグがない製品をなぜ開発できるのか？
他社より優れているためか？

答え: そうだと良いと思うが、実際の主な要因は以下の通り。
- このQA方式を30年以上も変えていない(方式に変化がない限り、毎年、良くなるのは当たり前)。
- レビューがうまく機能している。
- 退職率が低い。例えば、平均的なプロジェクト・マネージャは、勤続15年以上が普通(アメリカは、2〜3年)。

4. 本QA方式の異なる開発方式への適用性

(1) ウォーターフォール・モデル
本QA方式は、極めてうまく機能する。この理由として：
- 製品開発のライフサイクルが数ヶ月から数年に及ぶ(長い)。
- 各開発フェーズの、開始・終了が明確。
- 各開発フェーズに順序性がある。

(2) オブジェクト指向
- クラスやメソッドに本QA方式を適用(若干の変更あり)
- 本QA方式がオブジェクト指向での最良の方式かどうかは、現在、分析中。

(3) スパイラル・モデル
開発のライフサイクルが非常に短い場合(例えば、毎週、新しい製品をリリースする)、本QA方式は適当ではない。

第2部 :日本のQAと、アメリカ、インドのQA

質問: 日本式QAは、アメリカやインドでも適用可能か？

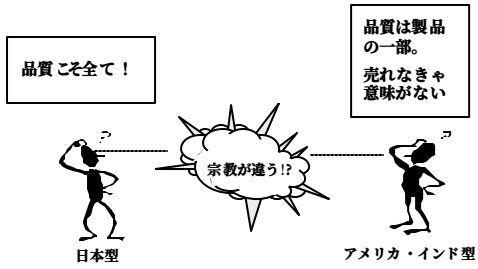
答え: もちろん、不可能

質問: なぜ、日本式 QAは、アメリカやインドで適用不可能か？

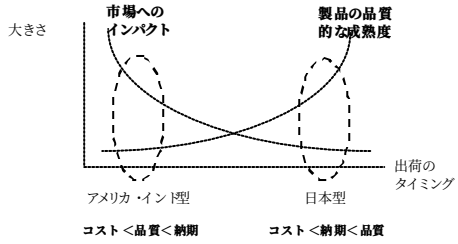
答え: 以下の理由による(日本式QAの課題？)。
(1) ソフトウェア開発での優先順位の違い
(2) ソフトウェア開発方式の変化
(3) 国民気質の違い

1. ソフトウェア開発における優先順位の違い

「日本型」対 「アメリカ・インド型」= 「品質」対 「市場性」

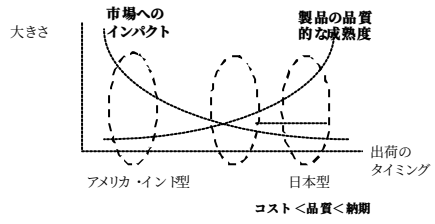


2. 日米の販売戦略の違い (従来の日本)



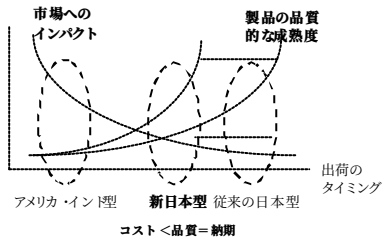
2. 日米の販売戦略の違い (これからの日本：その 1)

- マーケティング戦略の変化 :重厚長大から軽薄短小へ (数を売るには、販売タイミングが非常に重要。例えば、ボーナス商戦に間に合わないと売れないない等)
- 顧客の強い要求 (例 :会計年度に間に合わせる)



2. 日米の販売戦略の違い (これからの日本：その 2)

- リリース時期を早めつつ、品質を上げねばならない。



2. 日米の販売戦略の違い (これからの日本 :その 2)

質問 :では、なぜ、インドでCMMレベル 5を取得した会社が、世界で一番多いのか？ 品質が本当に良いためか？

答え : CMMは、 品質を向上させるプロセスがあるかどうか 」をチェック しているだけで、品質の絶対的な尺度は考 慮しない。例えば …

リトルリーグの野球チームでも、投手のローテーションに気を配り、定期的に選手の健康管理を実施していれば、CMMレベル 5を取得できる。一方、大リーグのワールド・シリーズ優勝チームでも、選手の管理を何もしていなければ、CMMのレベルは 1になる。

このように、CMMでは、チームの絶対的な投手力、打撃力 (絶対的な品質)と、チーム内の選手管理(プロセス改善能力)は、別物と考えている。

25

3. ソフトウェア開発方式の変化

・コスト削減、納期短縮のため、全て自社開発するのではなく 既に市場に存在する製品を組み込む。

→ 組み込む製品にバグがあっても、タイムリー に修正してもらえない。

→ 「バグ」を仕様としなければならない 場合が多い。

・技術の特殊化、特化により、他社に、製品 の一部の 開発を依頼せざるを得ない(例えば、携帯電話用のブラウザ)。

→ 製品の総合的な品質 は、他社発注製品の 品質次第。

→ 自社の品質保証方式を強制できない 強制 しても、やってくれない)。

26

4. ソフトウェア産業での、日米印の国民性 ・意識の差

日本のソフトウェア産業	アメリカ ・インドのソフトウェア産業
1. チーム全体の成功が 最重要 - 各個人から、 正直な」データを収集可。 - 統計データを元にした品質管理を適用できる。	1. チーム全体の成功より、個人の 成功が重要。 - 「バグを作りこんだ」 = 無能力」 - バグの 情報は、(無意識に)捏 造される。

27

4. ソフトウェア産業での、日米印の国民性 ・意識の差 (続き)

日本のソフトウェア産業	アメリカ ・インドのソフトウェア産業
2. 品質は、コストや納 期に優先	2. 早期リリースにより、マーケットの シェアを独占することは、品質より 優先

28

4. ソフトウェア産業での、日米印の国民性 ・意識の差 (続き)

日本のソフトウェア産業	アメリカ ・インドのソフトウェア産業
3. ソフトウェア開発では 「工業化」を重視 - キーワードは、 「安全サイド」 - 冒険をおかさない製品企画 - アメリカ等の斬新な技術の後追い	3. ソフトウェア開発では 「新規性」と、次世代の尖端技術」を重視 - キーワードは、 「挑戦」 - 斬新な(斬新過ぎる？)製品企画 - 画期的な技術を輩出

29

5. シェパードの分類による人間の行動パターン

感情的 (熱い)

* アメリカ ・インドのプログラマ*

Pタイプ (promotional:前進型)
- 自由を求める
- 常に、賞賛を期待

Fタイプ (facilitating: 協調型)
- 寛容を求める
- 常に、受容を期待

支配的 ― 謙虚

Cタイプ (controlling:制御型)
- 一貫性を求める
- 常に、達成を期待する

Aタイプ (analytical: 分析型)
- 正確さを求める
- 常に、安全性を期待する

* 日本のプログラマ*

抑圧的 (クール)

30

6. シェパードのモデルにおける法則

- 同じタイプに属する人同士は、うまくやれる。
- 違うタイプの人同士は、うまく行かないことがある。特に、対角線のタイプ同士は反発し合うことが多い(すなわち、PタイプとAタイプ、CタイプとFタイプ)。
- 同一人の中に、対角線の複数のタイプを併せ持っている人もいる。
  - CタイプとFタイプ —— 生まれながらのマネージャ型
  - PタイプとAタイプ —— アインシュタイン型

Pタイプ	Fタイプ
Cタイプ	Aタイプ

第3部： 昔からのQAの課題と解決案

1.現状の品質管理の課題

(1) ＱＡエンジニアの地位が低い？
- ・ＱＡは、創造的な仕事ではない？
- ・ＱＡエンジニアはプログラマより能力が低い？
- ・ＱＡエンジニアは、プログラマの尻拭い？
- ・ＱＡエンジニアは、システム開発能力がない？

(2) ＱＡエンジニアの士気が低い？

(3) 設計部とＱＡ部は、同じ製品のテストをするが、テスト項目は全く独立に設計・作成し、実行するので、時間と資源の無駄？

(4) 「早い検査」の実現

1.ＱＡに対する誤解？

・ＱＡは、創造的な仕事ではない？
・ＱＡエンジニアはプログラマより能力が低い？
・ＱＡエンジニアは、プログラマの尻拭い？
・ＱＡエンジニアは、システム開発能力がない？

⇩

例えば、原子力発電所を設計するのは簡単だが、安全性をテストするのは、極めて困難。この困難を可能にするのがＱＡエンジニア。

⇩

・プログラマより、はるかに高い能力が必要。
・非常に創造的な仕事。
・システムを容易に開発できる能力がないと、テストは無理。

2.「現状の品質管理の課題」の解決案

(1) ＱＡマネージャ制度の導入。

(2) 設計部とＱＡ部の資源共有・再利用。

(3) 経済産業省の国家資格として、「品質エンジニア」を新設。

3. 品質マネージャ制度の導入

テストの効率と独立性を上げるために、「品質マネージャ」をアサインする。

(1) テストの進化過程
第０ステージ：プログラマが自分でテストする。
第１ステージ：設計部内にテスト・チームがある。
第２ステージ：社内に、設計部とは独立したテスト・チームがある。
第３ステージ：「品質マネージャ」が開発プロジェクトの品質を全般的に指揮・制御。

3．品質マネージャ制度の導入（続き）

(2) 「品質マネージャ」の役割

・品質管理の最高責任者として、開発プロジェクトに関与。

・従来のＱＡエンジニアは、受身であったが、品質マネージャは品質確保に向けて、指揮権を発動し、積極的に関わる（ＱＡ関連の技術者のモラル、および、地位を向上させる）。

・プロジェクト・マネージャは製品開発の設計、工程管理、品質管理(デバッグ)を実施。品質マネージャは、プロジェクトの品質管理(テスト)の責任を持つ。

・品質マネージャは、品質に関し、プロジェクト・マネージャを指揮する(株式会社と経営コンサルタント、警察と検察の関係？)。

・品質マネージャは、他のプロジェクトを掛け持ちしてもよい。

4．設計とＱＡの資源共有・再利用

設計側のデバッグと、QA側のテストで、テスト項目、テスト・プログラムやテスト担当者を共通化。これにより、以下を実現する。

(1) テスト期間の短縮（全体の20％を占めるテスト期間の短縮化）

(2) テスト項目の設計・製造工数の大幅削減

(3) テスト要員の削減

⇩

資源の共有化、再利用のためには、「品質マネージャ」の指揮が必須

5．国家資格、「品質エンジニア」の新設

情報処理医術者の上級資格として、「品質エンジニア」を追加する

(1) テクニカル・エンジニア
　・エンベデッド・システム
　・システム管理
　・データベース
　・ネットワーク
(2) アプリケーション・エンジニア
(3) プロジェクト・マネージャ
(4) システム・アナリスト
(5) 品質エンジニア
　・入社10年目程度の経験と技術力
　・QA対象システムに最適のQA方式
　・システムの正当性、信頼性、機密性、安全性、可用性等を検証、分析。

5．国家資格、「品質エンジニア」の新設（続き）

新資格、「品質エンジニア」の追加により、期待できる効果

(1) 日本のQA技術者が、共通の用語や概念を共有。
(2) QA技術者の大幅な地位向上。
(3) QA技術者のプロ意識の育成・醸成。
(4) QA技術者の士気向上。
(5) QA技術者の技術力のスタンダードを設定。
(6) 会社や組織のQA能力の客観的な尺度。