テストファーストの導入と運用

テストファースト導入方法と統合開発環境eclipseを
利用したプロジェクトマネジメントについて

株式会社エスプランニング　福西　伸康

はじめに

なぜテストファースト導入をテーマとしたのか？

理由：いまいち使われていない

アジャイルだスパイラルだと、なんやかんや言っても結局現場で使われているのはウォーターフォールだったりする。

結論：いまいち理解されていない

プロジェクトマネージャーがウォーターフォールしか知らない

or

予算を握る人がウォーターフォールしか知らない

or

ノウハウのない新しい事はやりたくない人ばかりだ

上流の人の理解不足で、ソフトウェアプロセスの選択肢が狭いプロジェクトが多いのではないか？

本日のレジュメ

我らにソフトウェアプロセス選択の自由を！
と言う事で、まずはテストファーストを布教しよう！

1．テストファーストって何だ？
そもそもテストファーストって何ですか？テスト第一ですか？

2．誰を狙う？
狙うべき単体試験工程のプロジェクトマネジメントを理解する。

3．どうやって布教（洗脳）する？
テストファースト導入の利点と問題点を考える。

4．ネタを挙げて裏を取るべし
プロジェクトマネジメントに使う指標と指針を提案する。

5．うまくやってもらうべし
ツールを使って楽に、そして精度の高いマネジメントを行ってもらう。

テストファーストって何だ？

テストファーストとは？

設計でメソッド名なり引数なりが決まった時点で以下の手順を踏む手法。

1. コードにはメソッドを空実装。
2. 対応するテストコードを先に完成。
3. そのテストをパスする様にコード実装を行う。

中身（コードの実装）より先に、外側（メソッドの振る舞い等）を作るのがテストファースト。

XP（エクストリーム・プログラミング）などで強く推奨されており
ウォーターフォールのプロジェクトでも導入は比較的容易。

誰を狙う？

単体試験工程のプロジェクトマネジメントを理解する

単体試験工程では何を求められているか？

- 単体試験の質・量が品質を担保できるものであるかどうか？

  PMBOKでは品質マネジメント項目で、「評価尺度の定義」と「バグ予実管理の実施」が規定されており、テストファーストの新規導入では、この評価尺度をどの様に定義するかが問題となる。バグの予実管理（バグ密度など）は単体試験工程では意味をなさなくなる。

- 単体試験の進捗具合はスケジュールにあっているかどうか？

  PMBOKではタイムマネジメント項目で「作業計画」と「進捗の予実管理の実施」が規定されており、進捗具合の把握をどの様に行うかが問題となる。

管理のためには定量的なデータが必要であり、テストファーストを導入する場合、利用するデータを見直す必要がでてしまう人がいませんか？それは誰でしょう？

どうやって布教(洗脳)する？

まずは、プロジェクトマネジメントの観点から見たメリットデメリットを提示する。

・メリット

早い段階で設計考慮漏れの発見がされやすく、手戻り工数が削減される。
試験漏れが起こりにくく、保守工数が削減される。
クラス設計スキルが向上することで開発・保守工数が削減される。

・デメリット

必要工数の算出方法や妥当性判断、スケジュールの把握が難しい。
明確な品質の評価尺度を提示しにくい。
前提として、設計者にある程度のクラス設計スキルを要求する。

デメリットの解決方法も提案しなければダメでは？

ネタを挙げて裏を取るべし（１）

メトリクス計測による定量データを使って、できるだけシンプルに運用する方法はないものか。

管理工数も削減できれば皆幸せなのではないか？

・・・というかここにトラップを仕掛けることでテストファースト導入がうまくいくんじゃ？

単体試験工程で指標として使っていた過去の定量データ達

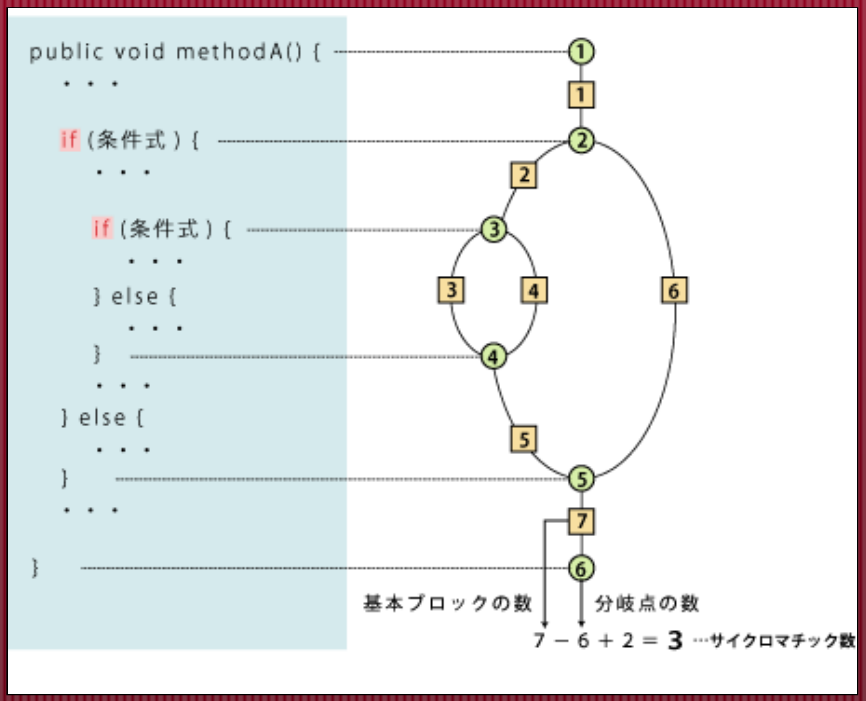
・バグ密度・試験密度

・・・品質マネジメントで利用。コード量あたり（1000行とか）のバグ件数や試験項目数を算出するある意味乱暴な指標。その乱暴さゆえ、開発者が意図的に目標値に近づける努力を行う事がある。

⇒そもそも、かなり長期に渡って統計を取らないと意味をなさない。

ネタを挙げて裏を取るべし（２）

着目する指標　マッケーブのサイクロマチック数



プログラムの複雑さを示します。
分岐が多いほど大きな数字となり
テスト項目数が必要になります。

製造・テストの重み・量・難易度・
クラス設計熟練度の判断に
ある程度利用可能。

計測には後述のメトリクス計測ツー

ルを使用する事で容易に計測。

この指標（だけ）を使って品質を管理しましょう。（と甘く囁く）

ネタを挙げて裏を取るべし（３）

品質のマネジメント指針（単体試験工程）

品質のマネジメント指標として、マッケーブのサイクロマチック数を利用し、
単体試験の工数的重みの判断や、単体試験項目数の妥当性判断を行う。

・１メソッドのサイクロマチック数が大きければ難易度が高く、保守性が低いと
判断できる。
⇒サイクロマチック数は原則10以下となる様にプロジェクトで規定すべき。

・１メソッドあたりの試験項目数は、そのメソッドのサイクロマチック数よりも少ない場合
なんらかの理由があると判断できる。
⇒サイクロマチック数が10以下であれば最低試験項目数は+２～３で収まる。

・バグ件数の予実管理は行わない。

ネタを挙げて裏を取るべし（４）

進捗のマネジメント指針（製造・単体試験工程）

進捗のマネジメント指針として、製造工程と単体試験工程を

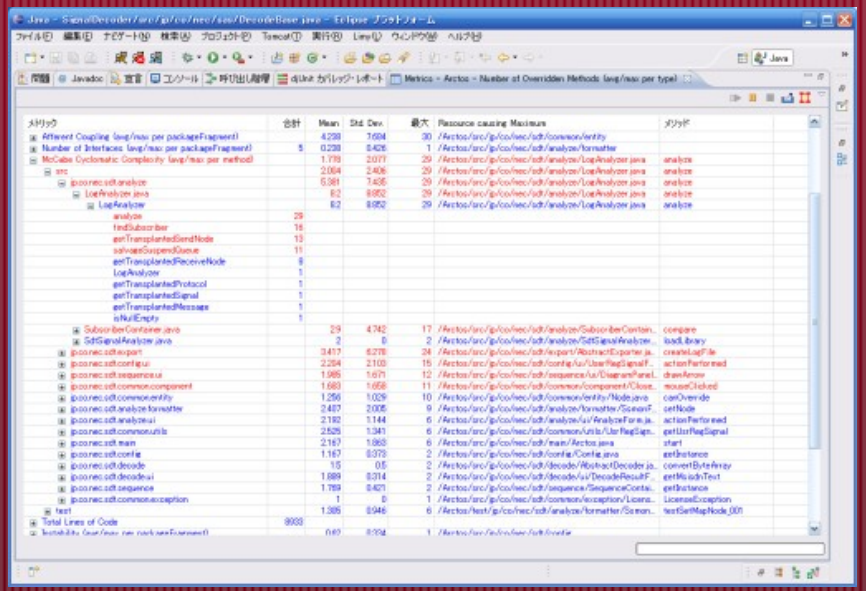
同一工程

として扱う。

・スケジュールはJUnitなどの単体テストフレームワークで試験OKとなる時期と数で管理を行う。
　⇒単体テストフレームワークを使用できないものについては別途協議。
　⇒進捗の予実管理は必ずS字曲線で管理。
　（工程初期はテストコード作成がメインになるため）

うまくやってもらうべし（１）

メトリクス計測ツール
Eclipse Metrics Plugin（Frank Sauer）



統合開発環境Eclipseのプラグインとして使え、クラス数・メソッド数・実コード量・マッケーブのサイクロマチック数・依存関係などの計測が可能なフリーウェア。

計測結果はXMLファイルに出力可能。閾値を設定することでワーニングとすることも可能で、保守性向上のコーディングガイドラインを設定し、運用することも可能

うまくやってもらうべし（２）

運用について

テストファーストの導入には、単体試験の難易度を下げるため、そして保守性を高めるために、特にクラス設計者のスキルアップが欠かせません。

それは、サイクロマチック数を下げる設計＝オブジェクト指向の原則である

- SRP（単一責任の原則）
- OCP（オープン・クローズドの原則）
- DIP（依存関係逆転の原則）

を基本とした設計スキルを養うことです。

サイクロマチック数縛りをプロジェクトにかけることで、否応にも意識しなければ設計できなくなりますので品質と工数削減にとても貢献するかも知れません。（良い事ずくめですね！）

ご清聴ありがとうございます。

危険なプロジェクトもサルベージ
㈱エスプランニングでお送りしました。