テストマネージャの決断

2018/3/8

モデレータ　町田　欣史（NTTデータ）

パネリスト　長谷川　聡（ベリサーブ）
柿崎　憲（サイバード）
湯本　剛（ASTER）

セッション概要

- 本セッションでは、テストマネジメントの経験が豊富な３名のエンジニアをお迎えします。
- 現場でよくあるいくつかの状況において、彼らがどのようにテストマネージャとしての決断を行うのかをお見せします。
- ３名はそれぞれ、エンタープライズ系のテスト、組み込み系のテスト、Web系のテストを専門としており、経験や背景が異なります。
- 経験や背景の異なる３名が、それぞれの状況で、決断に至るために集める情報、決断のタイミング、決断に至るまでの振る舞いをディスカッションします。
- 概念的なテストマネジメントの知識から、具体的なテストマネージャの立ち回りについて学びたい方にオススメします。

本セッションの進め方

- オープニング
- テストマネージャとしての決断
  - 状況１（ポジショントーク含む）
  - 状況２
  - 状況３
- クロージング

それぞれの状況に関するディスカッションは質疑を含めて
２５分程度を予定しています。

状況１：計画値を超えるバグが見つかった

あなたがシステムテストのテストマネージャを担当しているシステムは、5年前にファーストリリースをしました。それ以来、継続的にバージョンアップ（機能追加）を行い、計10回の市場リリースをしました。

これらの機能追加開発では、テストレベルごとにテスト計画時にバグ数を見積もり、実績との差を管理してきました。この差は開発を重ねるごとに安定してきており、6回目の機能追加開発以降では、システムテストでのバグ数の計画値と実績値の差は10%以内に収まってきました。

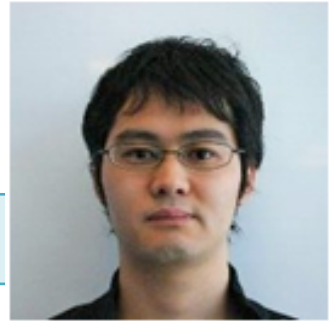
今回（11回目）の機能追加開発では、システムテストでのバグ数を50件と計画しました。ところが、約30%のテストケースを実行した時点で既に計画値である50件のバグが検出されてしまいました。

この状況において、あなたはテストマネージャとして何をしますか？

コンテキストシート(長谷川)

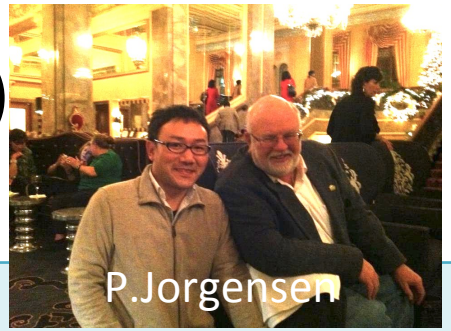
システム形態	組込み
開発規模	100万～300万STEP程度
開発期間	派生開発で9ヶ月～1年半
プロジェクトメンバ数	開発：細かい人数は不明も200～400名 テスト：システムテスト以降で100名程度
テストチームメンバ数	システムテストで100名
体制 (テストチームの位置づけ)	コンポーネント～統合テストまで開発チームが実施 システムテスト以降は開発チームから独立
責任範囲(対象テストレベル)	システムテスト、受け入れテスト
開発モデル	ウォーターフォール
内製／外注	ほぼ外注
システム利用者	コンシューマ
その他の特徴	・開発者は近くにいない ・本当に開発している人が誰かがわからない ・運用を変えることがものすごく大変(契約構造等)

コンテキストシート(柿崎)



システム形態	Web／スマートフォンアプリ
開発規模	スマートフォンアプリの場合 クライアント側35万STEP、サーバ側45万STEP
開発期間	新規開発で6ヵ月～1年くらい
プロジェクトメンバ数	10～60名前後(テスター込み)×プロダクト数分
テストチームメンバ数	1～5名前後×プロダクト数分
体制 (テストチームの位置づけ)	できるだけ開発チームと一体化させる
責任範囲 (対象テストレベル)	開発チーム → コンポーネント、統合（前半） テストチーム → 統合（後半）、システム
開発モデル	アジャイルでありたい
内製／外注	ほぼ内製
システム利用者	コンシューマ
その他の特徴	複数のプロダクトを管理する。開発者と近い

コンテキストシート(湯本)



システム形態	エンタープライズ
開発規模	さまざま（自社にある数十のシステム）
開発期間	100程度の変更案件を1年かけて順次リリース
プロジェクトメンバ数	1システムあたり，1名～300名程度
テストチームメンバ数	テストマネジメントするテストチームが10名
体制 (テストチームの位置づけ)	コンポーネント～システムテストまで開発チーム ユーザ受け入れテスト以降はビジネス側チーム それらのテストをマネジメントするテストチーム
責任範囲(対象テストレベル)	統合テスト、システムテスト、ユーザ受け入れテスト
開発モデル	ウォーターフォール？
内製／外注	数人は社員，あとは協力会社
システム利用者	コンシューマ／社内
その他の特徴	・メインフレーム、Web、パッケージなどさまざま ・いろいろな国の人が開発に関わる

長谷川さん	柿崎さん	湯本さん
組込み	Web／スマホアプリ	エンプラ
100万～300万STEP	80万STEP（スマホアプリ）	さまざま
9か月～1年半	半年～1年	1年
プロジェクト 500名 テストチーム 100名	プロジェクト 10～60名 テストチーム 1～5名 （×プロダクト数）	プロジェクト 1～300名 テストチーム 10名 （テストマネジメント）
統合テストまで：開発チーム システムテスト：テストチーム	統合（前半）まで：開発チーム 統合（後半）から：テストチーム	システムテストまで：開発チーム 受け入れテストから：ビジネス側 テストチームがマネジメント
システムテスト、受け入れテスト	すべて（開発チームとテストチームが一体化）	統合テスト、システムテスト、受け入れテスト
ウォーターフォール	アジャイル	ウォーターフォール？
ほぼ外注	ほぼ内製	ほぼ外注
コンシューマ向け	コンシューマ向け	コンシューマ／社内向け
• 開発者との距離が遠い • 運用を変えるのが大変（契約構造等）	• 複数のプロダクトを管理 • 開発者との距離が近い	• メインフレーム、Web、パッケージなどさまざま • 外国人が開発に関わる

状況１：計画値を超えるバグが見つかった

パネリストの主な回答

- 騒がない
- 50件が本当にバグかどうか確認する
  （重複バグ、仕様書の問題、本来見つけるべき工程など）
- 開発者の様子を見に行く
- リリースの時期やリリースする機能を調整する
- バグに一番詳しい人（テストチーム）が対処する
- この状況を生み出さないためにバグの状況をリアルタイムで追いかける

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.6 テストメトリクスの定義および使用

欠陥に関連するメトリクスには、次のようなものがある。

（中略）

- 次のように分類した、欠陥の数または割合：
  - 特定のテストアイテムまたはコンポーネント
  - 根本原因
  - 欠陥の起源（たとえば、要求仕様、新しいフィーチャ、リグレッションなど）
  - テストリリース
  - 混入／検出／除去されたフェーズ
  - 優先度／重要度
  - 拒否されたレポートまたは重複レポート

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.6 テストメトリクスの定義および使用

テストコントロールは、テストにより生成された情報と、プロジェクトまたはテストの状況の変化に対応する必要がある。たとえば、動的テストにより、多くの欠陥を含むとは予測しなかった領域で大量の欠陥を検出した場合、または、テスト開始の遅延によりテスト実行期間を短縮した場合、リスク分析と計画を見直す必要がある。これにより、テストの優先度の見直しや、残りのテスト実行活動の再割り当てが発生することがある。

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.6 テストメトリクスの定義および使用

テスト進捗レポートによりテスト計画からの逸脱を発見した場合、テストコントロールを実行する必要がある。テストコントロールの目的は、プロジェクトまたはテストが成功に向かうよう軌道修正することである。テスト結果を使用してプロジェクトのコントロール活動に影響を与えるか、活動を測定する際に、次のオプションを考慮する必要がある。

- 品質リスク分析、テストの優先度、およびテスト計画の見直し
- リソースの追加、またはプロジェクト活動やテスト活動の追加
- リリース日の延期
- テスト終了基準の緩和または厳格化
- プロジェクトの対象範囲（機能および非機能）の変更

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

4.2.2 無効および重複した欠陥レポートのマネジメント

特定の状況では、不正が、欠陥の兆候としてではなく、テスト環境、テストデータ、テストウェアの他の要素、またはテスト担当者自身の誤解により発生することがある。テスト担当者が欠陥レポートをオープンし、その後、そのレポートはテストでの成果物の欠陥と関連しないことが判明した場合、偽陽性結果をもたらす。このようなレポートは、無効な欠陥レポートとして、キャンセルするか、クローズする。さらに、1つの欠陥が異なる兆候を示し、テスト担当者にとってまったく関連していないように見える場合がある。2つまたはそれ以上の欠陥レポートが登録され、その後、同じ根本原因に関連していることが判明した場合、通常、それらの欠陥レポートの1つのみを維持し、他の欠陥レポートは重複欠陥レポートとしてクローズする。

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

4.3 欠陥レポート情報

収集された欠陥データは、テスト進捗のモニタリング、コントロール、および終了基準の評価に使用できる。たとえば、欠陥情報は、欠陥密度分析、検出し解決した欠陥の傾向分析、欠陥の検出から解決までの平均時間、および故障強度（たとえば、MTBF分析など）に使用できる。

収集される欠陥データは、次のアイテムを含む。

（省略）

状況２：不十分なテストベース＋複雑な開発体制

あなたは、初めて経験するプロダクトを開発するプロジェクトに、統合テスト以降のテストマネージャとしてアサインされました。

このプロダクトは、大分すると５つの大きな機能に分類されます。それらの５つの大きな機能ごとに、それぞれ別の開発組織（会社）が基本設計から単体テストまでを担当することになっています。

現在、開発は基本設計まで完了し、詳細設計を行っている段階です。

テストマネージャのあなたは統合テスト以降のテスト計画を立てるため、テストベースとなる基本設計書を確認していました。ところが、基本設計書が存在しない機能があったり、基本設計書はあっても記述内容や記述粒度がバラバラであることが判明しました。

この状況において、あなたはテストマネージャとして何をしますか？

状況２：不十分なテストベース＋複雑な開発体制

パネリストの主な回答

- 自分の足で調べに行く
- 解決のためのタスクフォースを作って事態を収拾する
- 組織を再整理する
- テストの責任ではない
- 仕様書の問題としてアラートをあげる
- テストケースのレビューを強化する
- 開発標準と比べる
- テストチームを使って仕様を整理する

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

1.3 テスト分析

次の状況では、詳細なテスト条件の仕様が、特に効果的な可能性がある。

- 開発ライフサイクル、コストや時間の制約、またはその他の要因に対応するために、チェックリストなどの簡易なテスト設計文書化方式を使用している。
- 公式な要件、またはそれ以外の開発成果物が、テストベースとして、ほとんど、あるいはまったく使用できない。
- プロジェクトが大規模、複雑、または高リスクであり、単純にテストケースを開発成果物に関連付けることでは提供できないモニタリングおよびコントロールのレベルを必要としている。

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.2.3 テストを選択するためのその他の技法

要件ベースドテストに対する一般的な障害は、多くの場合、要求仕様が曖昧、検証不能、不完全、または存在しないことである。すべての組織がこれらの問題の解決に意欲的であるわけではないので、このような状況に直面したテスト担当者は、別のテスト戦略を選択する必要がある。

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.3.3 テストを選択するためのその他の技法

最後に、一般的なもう1つの方法は、対処的アプローチを使用することである。対処的アプローチでは、テスト実行の前には、テストの分析作業、設計作業、または実装作業をほとんど行わない。テストチームは、実際に提供されたプロダクトに対する対応に焦点を当てる。バグの偏在を検出すると、そこでさらなるテストを実施する。優先度付けと割り当ては、完全に動的に行う。対処的アプローチは他のアプローチの補完として機能するが、単独で採用した場合は、重要ではあるが少数のバグしか発生していないアプリケーションの主要な領域を見逃す傾向がある。

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.8 分散テスト、アウトソーステスト、およびインソーステスト

多くの場合、テスト活動の一部もしくはすべてが、他の会社の従業員やプロジェクトチームとは別の人々よって、異なった地域で行われる。テスト活動が複数の地域で行われる場合、そのテスト活動を「分散テスト」と呼ぶ。テスト活動を、同じ企業の従業員でなく、プロジェクトチームと同じ地域にいない人たちによって、1つの地域または複数の地域で実行する場合、そのテスト活動を「アウトソーステスト」と呼ぶ。テスト活動を、プロジェクトチームと同じ地域に存在するが同僚の従業員でない人々によって実行する場合、そのテスト活動を「インソーステスト」と呼ぶ。

次ページにつづく

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.8 分散テスト、アウトソーステスト、およびインソーステスト

このようなテスト活動のすべてにおいて、共通して明確なコミュニケーションチャネルの他、使命、業務、提出書類に対する要望を十分にまとめておくことが必要になる。

（中略）

また、このようなテスト活動のすべてにおいて、方法論の調整が共通して必要になる。

（中略）

分散テストの場合、テスト作業を複数の地域に分割することを明示し、これを合理的に決定しなければならない。

状況３：リリース直前の仕様変更

あなたは、２年の開発期間を予定しているプロジェクトでテストマネージャとして活動しています。３ヵ月後のリリースに向けて、現在システムテストを行っています。システムテスト終了後はユーザ受け入れテストを行った後に市場リリースする計画となっています。現在、やや計画から遅れてはいるものの、３ヵ月後のリリースには何とか間に合いそうだと判断しています。

そんな中、ビジネスアナリストから仕様変更の依頼がありました。実は、つい最近ライバル会社が類似のシステムを１ヵ月後に市場に投入する発表をしました。そのため、現在の機能のままリリースしても差別化が行えず、かつ後発になってしまうので、あなたの会社のビジネスに大きなインパクトがあるという判断がされました。

そこで、プロジェクトマネージャがテストマネージャのあなたに相談してきました。というのも、この仕様変更は、システムに対して新たなコードを追加するだけで対応できるものではなく、既存部分にも手を入れる必要があります。そのため、コンポーネントテストや統合テストを実施済みのモジュールに対しても、大規模な回帰テストが必要になると考えられます。

この状況において、あなたはテストマネージャとして何をしますか？

状況３：リリース直前の仕様変更

パネリストの回答

- 受けたくない
- 段階リリースをする
- リスクベースで対応する
- サポートチームに体制を作ってもらい、連携する
- 途中でやり方を変えたくない
  （リスクベースにもしたくない）
- 受け入れないとは言えない
- Web系なら1か月後よりも前にリリースする方法を考える
- 顧客価値を最優先に考える
- 限定されたリリースをする

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.2.1 テストステークホルダについて

テストマネージャは、プロジェクトまたはプログラムに関する特定のテストステークホルダを識別する必要がある。また、ステークホルダとテストとの関係の本質、およびテストチームがステークホルダのニーズに応える方法について理解する必要がある。

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.2.2 その他のソフトウェア開発ライフサイクル活動と成果物

テストマネージャは、テストアナリストおよびテクニカルテストアナリストと協力し、スケジュール要件とリソース要件をプロジェクトマネージャに伝える必要がある。テストマネージャは、プロジェクトマネージャと協力し、プロジェクト計画の変更を理解し、テストコントロールアクションを実行して、これらの変更に対応する必要がある。

【参考】JSTQB Advanced Levelシラバス（テストマネージャ）より

2.3.4 テストプロセスにおけるテストの優先度付けと工数の割り当て

たとえば、リスクベースドテストでは、テストを実行して欠陥を検出すると、テスト担当者は残存する未解決のリスクのレベルを調べることができる。これは、リスクベースドテストを使用して、正しいリリース時期を決定するのに役立つ。テストレポートでは、実現された利点と実現されていない利点の他、対応済みのリスクと未対応のリスクを掲載する必要がある。リスクカバレッジに基づいたテスト結果レポートの例については、[Black03]を参照されたい。