[登壇スライド]

プロダクトの品質に向き合うための知識と考え方

Shintaro Takahashi

Introduction

ソフトウェアプロダクトのテストや品質管理について、

知っている様で理解していないことや、テストや品質管理について考えるうえで知っておきたいことをまとめて紹介します。

割と初心者向けでそんなに深い内容話しません。

この発表を聞けば、明日から自分が関わるプロダクトのテストについて考えるスタートラインに立てるでしょう。

話さないこと

- 具体的なテスト手法の実践方法
- テストに関わる具体的なテクニック
- テストの書き方
- テストツール
- Haskellの布教
- ELDEN RING の DLC

Outline

- ソフトウェアの品質とは
- ソフトウェアテストとは
- 開発とテストに関わる基礎知識
- 押さえておきたい考え方
- まとめ

1. ソフトウェアの品質とは

ISOによる品質モデル

ISO(国際標準化機構)によってソフトウェアの品質モデルが**ISO/IEC SQuaRE シリーズ**として定義されている。

SQuaRE → " **S**ystem and Software **Qua**lity **R**equirements and **E**valuation "

- 25000 Guid to SQuaRE
- 25010 System and software quality models
- 25012 Data quality model
- 25030 Quality requirements framework

ISOによる品質モデル

ISO/IEC 25000 において以下のように用語定義されている

ソフトウェア品質 (software quality)

「明示された条件下で使用するとき、明示的ニーズ又は暗黙のニーズを満たすためのソフトウェア製品の能力。」

利用時の品質 (quality in use)

「特定の利用状況において、特定の利用者が特定の目標を達成するというニーズを満たすために、有効性、効率性、リスク回避性及び満足性に関して、ソフトウェア製品又はシステムを使用できる度合い。」

製品品質モデル(ISO25010)

ソフトウェア品質を示す品質特性の分類

- 機能的適合性
- 性能効率性
- 互換性
- 使用性
- 信頼性
- セキュリティ
- 保守性
- 移植性

製品品質モデル(ISO25010)

SOFTWARE PRODUCT QUALITY								
FUNCTIONAL SUITABILITY	**PERFORMANCE EFFICIENCY**	**COMPATIBILITY**	**INTERACTION CAPABILITY**	**RELIABILITY**	**SECURITY**	**MAINTAINABILITY**	**FLEXIBILITY**	**SAFETY**
FUNCTIONAL COMPLETENESS FUNCTIONAL CORRECTNESS FUNCTIONAL APPROPRIATENESS iso25000.com	TIME BEHAVIOUR RESOURCE UTILIZATION CAPACITY	CO-EXISTENCE INTEROPERABILITY	APPROPRIATENESS RECOGNIZABILITY LEARNABILITY OPERABILITY USER ERROR PROTECTION USER ENGAGEMENT INCLUSIVITY USER ASSISTANCE SELF-DESCRIPTIVENESS	FAULTLESSNESS AVAILABILITY FAULT TOLERANCE RECOVERABILITY	CONFIDENTIALITY INTEGRITY NON-REPUDIATION ACCOUNTABILITY AUTHENTICITY RESISTANCE	MODULARITY REUSABILITY ANALYSABILITY MODIFIABILITY TESTABILITY	ADAPTABILITY SCALABILITY INSTALLABILITY REPLACEABILITY	OPERATIONAL CONSTRAINT RISK IDENTIFICATION FAIL SAFE HAZARD WARNING SAFE INTEGRATION

利用時の品質モデル(ISO25010)

システムとの対話による結果(=製品の利用)に関係する品質特性

- 有効性
- 効率性
- 満足性
- リスク回避性
- 利用状況網羅性

品質モデルの対象

製品品質モデル

→ コンピュータシステム(ハードウェアを含む)とソフトウェア本体が品質評価の対象となる
「ソフトウェア製品の能力」

利用時の品質モデル

→ コンピュータシステム、ソフトウェア、データ、ネットワーク、利用環境、利用者等様々な要因が品質に影響を与える
「利用者が特定の目標を達成するというニーズを満たすために使用できる度合い」

狩野モデル

東京理科大学名誉教授の狩野紀昭（かのう のりあき）らによって考案された品質要素の分類および特徴づけの手法として開発されたモデル

品質要素	顧客の満足感・物理的充足状況
当たり前品質	充足されれば当たり前と受け止められるが、不充足であれば不満を引き起こす。
一元的品質	充足されれば満足を、不充足であれば不満を引き起こす。
魅力的品質	充足されれば満足を与えるが、不充足であっても仕方ないと受け取られる。
無関心品質	充足でも不充足でも、満足を与えず不満も引き起こさない。
逆品質	充足されているのに不満を引き起こしたり、不充足であるのに満足を与えたりする。

現在地情報

- ソフトウェアの品質とは
- ソフトウェアテストとは　　←**次ココ**
- 開発とテストに関わる基礎知識
- 押さえておきたい考え方
- まとめ

2. ソフトウェアテストとは

ソフトウェアテストとは

ソフトウェアの品質を高めるための検証
及びそれにかかわる諸活動

ソフトウェアテストとは

以上、おわり！

ソフトウェアテストとは

ソフトウェアのテストは
検証対象であるソフトウェアの品質と同じだけ様々な要素が関係し
多角的に検証する必要があるものです。
ソフトウェアの **テスト**が何であるかよりもテストしたい**品質**とは
何かを考えた方がより良いでしょう。

現在地情報

- ソフトウェアの品質とは
- ソフトウェアテストとは
- 開発とテストに関わる基礎知識　　←**次ココ**
- 押さえておきたい考え方
- まとめ

3. 開発とテストに関わる基礎知識

開発とテストに関わる基礎知識

開発とテストに関わる広く知られている知識をいくつか紹介します。
全て覚える必要は無く、どういった考え方で品質向上を目指すのかといった観点で聞いてください。

テストの種類

テストの種類

単体試験

Unit Test。 関数・メソッド等の小さな単位で行う

統合試験

Integration test, 結合テストとも呼ぶ。 単体テストが済んだものを組み合わせてテストする。

- 内部結合：独立したプログラムの内部で小さなモジュールを組み合わせた状態
- 外部結合：独立したプログラム同士を実際につなげた状態。システムテストと区別する場合は全てつなげた状態では無く1つずつ組み合わせた状態でテストを行うことが多い。

テストの種類

システムテスト
プログラムを単独ではなく他のプログラムやハードウェア、ネットワーク、データベースなどを組み合わせた状態で実施するテスト。統合試験と分けて考えない場合もある。

E2Eテスト
End to End test。ユーザーが実際にアプリケーションを使用するのと同様の環境で、ユーザーが実行するアクションを想定したテスト。 アプリケーションのユーザ体験全体をテストする

受入試験
検収テスト、User Acceptance Test (UAT) とも呼ぶ。製品をリリースする前にそれを受け入れるかどうか判定するための試験。
実際の製品の利用者が行う場合や、運用・保守担当、仕様策定者が行う場合がある。

テストの種類

リグレッションテスト
回帰試験、退行テストとも呼ぶ。プログラムに対し修正・変更を行った際に、以前通っていたテストを再度行う。
今回の修正・変更が他の意図しない機能に影響を与えていないかを検証する。(所謂デグレが起こって無いか)

性能試験
パフォーマンステスト。一度にどれくらいの数のリクエストに応答できるか、リクエストからレスポンスまでの速度はどれくらいか、画面表示の速度はどれくらいか、特定のサイズのデータを処理するのにどれくらい時間がかかるか、等システムの性能をテストする。

テストの種類

負荷テスト
システムに対して大量のリクエストやデータを送り、パフォーマンスと耐久性を評価を評価する。
パフォーマンステストが特定条件下での**性能**を測るのに対し、こちらはシステムの**限界**を測る。

セキュリティテスト
文字通りセキュリティに関するテスト。脆弱性が無いか、情報の機密性は保たれているか等。

テストの種類

ブラックボックステスト
プログラムの中身(ソースコード)を知らない状態で、**入力に対して期待する出力が得られるか**(要求仕様に合致するか)を検証するテスト。

ホワイトボックステスト
ブラックボックスとは逆に、プログラムの中身を知ったうえで**プログラムの論理構造が正しいか**を解析するテスト。

テストの種類

その他にもテスト対象やテスト方法によって紹介し切れないほどたくさんあります。 何をテストしたいか、どのようにテストするのが効果的か、を考えて適切なテスト手法を選択する必要があります。

※色々な観点をごちゃまぜにして紹介したので、横並びにすべきでないものも多々あります。

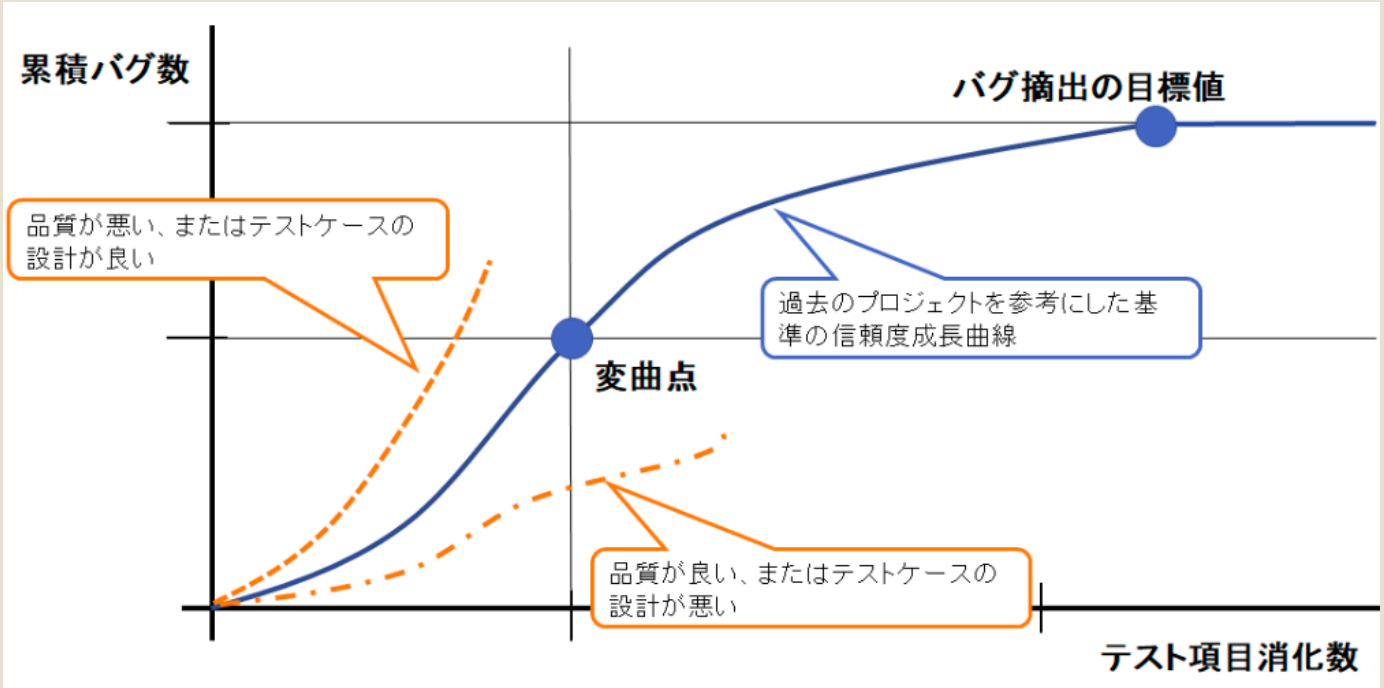
その他のテスト

- 探索的テスト
- モンキーテスト

テストの効果測定

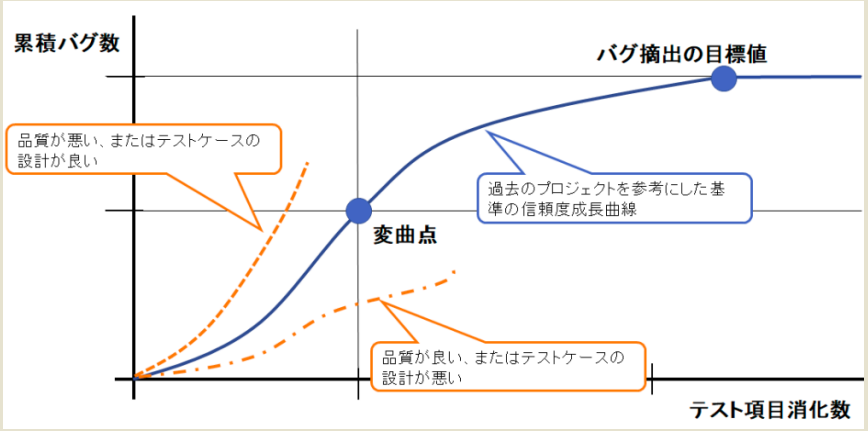
テストの効果測定

信頼度成長曲線



(出典: Qiitaの記事)

テストの効果測定 | 信頼度成長曲線



信頼度成長曲線とは

横軸にテスト消化数・テスト回数・経過時間(期間)等を取り、縦軸に検出したバグの数を取るグラフ。
テストと修正を繰り返した結果、現在の製品の品質やテスト手法の効果を測定する目的で使用される。

序盤はテスト手法が未熟なため傾斜が緩く、途中から成熟し伸びが良くなり、終盤になると潜在するバグ件数が減って緩やかになる。という形でS時のグラフになる事が多い。

テストの効果測定

その他にもテストの効果を測るための指標・手法は様々あります。

- カバレッジ(網羅率)
- ミューテーション解析
- テスト密度、欠陥密度
- etc

信頼度成長曲線からも分かるようにテスト活動の効果や品質を評価するためには
得られたメトリクスを用いて**継続的に分析する**ことが必要です。

Shift Left

Shift Left

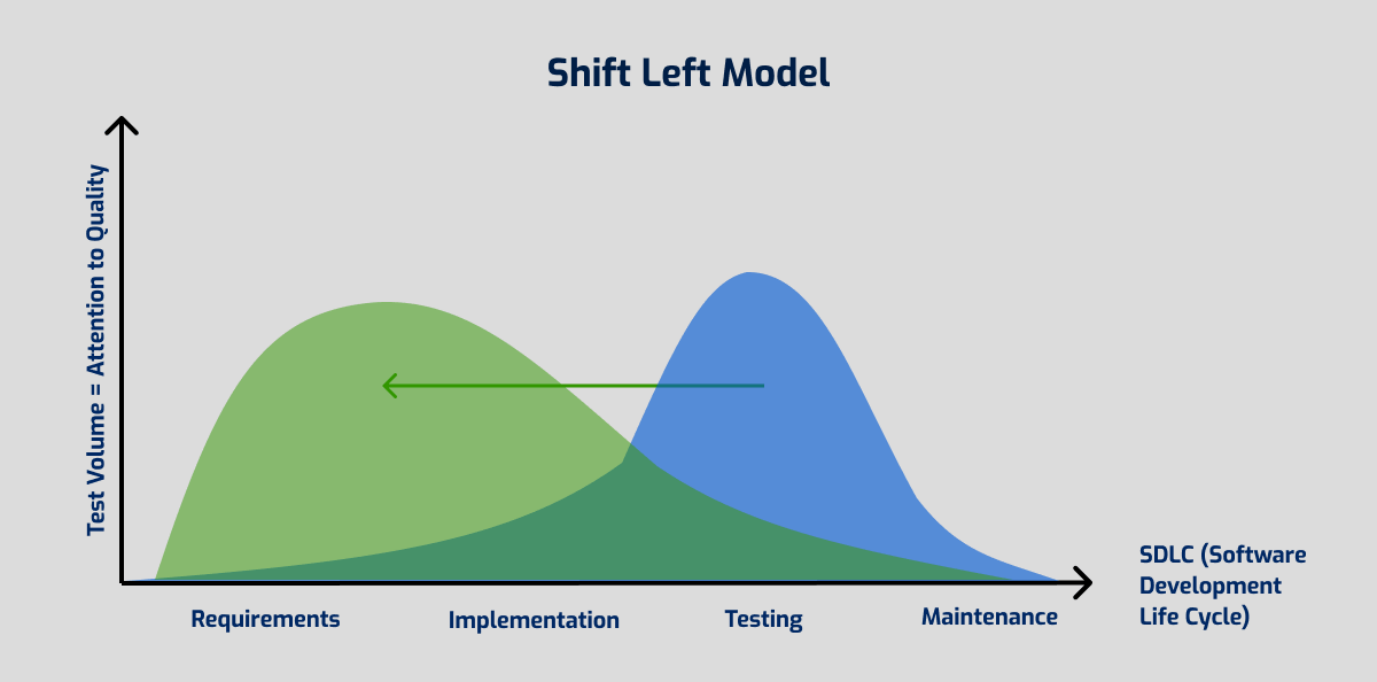
Shift Left

①要件定義→②計画→③設計→④開発→⑤テスト→⑥リリース

という開発の流れがあったとして

品質を高める活動を⑤から始めるのではなく、もっと左側(早い段階)から始めようという考え方。

Shift Left



(出典: Embracing the Shift Left Model in Software Testing)

Shift Left

Shift Leftに関するポイント

- バグは見つかるのが遅いほど直すのに時間がかかる
- 要求仕様を明確にする
- レビューをする
- コードベースの単体テストをする
- テストを効率化する

Shift Left

Shift Leftは単に「開発者がコードベースのテストを書こう」とか「自動テストをやろう」

という事では無く、早期から品質を高めるための活動を行っていくという考え方です。

最近ではDevOpsと絡めた開発のライフサイクルの最適化など色々な話の中に出てきます。

(時間と体力の都合で割愛します。。)

現在地情報

- ソフトウェアの品質とは
- ソフトウェアテストとは
- 開発とテストに関わる基礎知識
- 押さえておきたい考え方　←**ついにここまでキタ**
- まとめ

押さえておきたい考え方

押さえておきたい考え方

あらゆる工程からバグは発生する

押さえておきたい考え方

早期からパフォーマンステストと
セキュリティテストを意識する

押さえておきたい考え方

カバレッジは100％を目指さない

押さえておきたい考え方

どんなソフトウェアにもバグは存在する
バグを生まないプログラマもいない

押さえておきたい考え方

品質向上のための活動は続ける限り無限に続く

現在地情報

- ソフトウェアの品質とは
- ソフトウェアテストとは
- 開発とテストに関わる基礎知識
- 押さえておきたい考え方
- まとめ　　←**やっと終わるよ**

まとめ

- ソフトウェアの品質には様々な要素があるよ。
- ソフトウェアの品質には様々な視点で評価されるよ。
- テストは品質向上の活動。目的に応じて適切な方法を考えよう。
- バグは無限。上手く付き合っていこう。

参考文献

- Lisa Crispin, Janet Gregory(2009). 『実践アジャイルテスト』. 翔泳社
- 高橋 寿一(2013). 『知識ゼロから学ぶソフトウェアテスト【改訂版】』. 翔泳社
- 高橋 寿一(2021). 『ソフトウェア品質を高める開発者テスト ― アジャイル時代の実践的・効率的なテストのやり方』. 翔泳社
- 吉井 健文(2023). 『フロントエンド開発のためのテスト入門 ― 今からでも知っておきたい自動テスト戦略の必須知識』. 翔泳社