間隔

内容

箇条書きのテスト記事です。

1. チェック 1
2. チェック 2
3. アドホックテストのタイプ
    - バディテスト：2人のチームメンバー (通常は開発者とテスター) が協力して特定のモジュールをテストし、異なる視点を提供します。
    - ペアテスト：2人のテスターが同じマシン上で協力してアプリケーションのレビューとテストを行い、突発的なアイデアを生み出します。
    - Monkey Testing:非構造化の状態が非常に多いこのアプローチでは、ランダムなデータを入力するか、ランダムなアクションを実行して、ソフトウェアの動作を確認し、クラッシュが発生しないことを確認します。

4. 画像の後に行頭文字

長所と短所
賛成論

- 時間の節約：正式なテストケースを作成する時間のかかるプロセスを省略できるため、厳しい納期に最適です。
- 正式なテストを補完：構造化されたテストスイートでは見逃されてしまうような、より深いバグや不明瞭なバグを見つけます。
- 柔軟性：要件が不完全または変化した場合に完全に適応します。

1. アドホックテストで使用される手法

アドホック検査法は、事前定義された検査例を用いずに不具合を迅速に同定するために検査者が用いる非公式の検査法である。これらの手法は、アプリケーションの経験、直感、およびランダムな探索に依存しています。

- 誤りの推測：不具合が発生しやすい領域を予測するテスターの経験に基づき、一般的な間違いやリスクのあるモジュールに焦点を当てます。
- ランダム入力テスト：予期しないデータまたは無効なデータを使用してシステムの動作をチェックし、アプリケーションが異常な入力を適切に処理することを確認します。
- 境界テスト（非公式）:入力制限のエッジでエラーを特定するために入力の最小値と最大値をテストします。
- 探索的テストアプローチ：事前に定義されたテストケースを使用せずにアプリケーションを同時に学習およびテストすることで、不具合を動的に検出します。
- Monkey Testing:システムの安定性をチェックし、クラッシュを特定するロジックなしで、ランダムなアクションを実行します。
- セッションベースのテスト：特定のモジュールを効率的にカバーするために、短いセッションに重点を置いてテストを実施します。
- ネガティブテスト：誤った入力や無効な入力を使用して、システムがエラーと検証を正しく処理していることを確認します。
- ユーザシナリオテスト：実際のユーザの動作をシミュレートして、実際の状況でアプリケーションが正しく動作することを確認し、ユーザビリティの問題を特定します。

例：モバイルアプリ

テスターは、モバイルアプリケーションに対してランダムなアクションを実行します。

- 画面をすばやく切り替える
- デバイスを頻繁に回転する
- インターネット接続が少ないアプリを使用する
- 一度に複数の機能を開く

これにより、パフォーマンスの問題、UIのエラー、またはアプリケーションのクラッシュが明らかになる場合があります。

アドホックテストに最適な条件

時間が限られている場合に最も効果的なのは、テスターが各分野に関する豊富な知識を持っていることです。隠れた不具合を見つけ出すには、迅速な調査が必要です。

- 試験時間が限られている場合。
- 正式なテストが完了したら、
- 機能またはモジュールの迅速な検証が必要な場合。
- 詳細なテストケースが利用できない場合。
- 開発がほぼ完了したとき。
- 高リスクまたはエラーが発生しやすい領域をテストする場合

アドホックテストの制限

文書化と構造が不十分なため、テストの再現と不具合全体の保証が困難

- 適切なドキュメントがないため、後で不具合を追跡することが困難です。
- テストのカバレッジは、非構造化特性のため保証されません。

- テスターの経験と知識に大きく依存します。
- 見つかった問題は再現が困難な場合があります。
- 正式かつ体系的なテスト方法に代わるものではありません。

アドホックテストの利点

予期しない不具合を正式なテストケースなしで迅速に特定し、時間を節約してテストカバレッジ全体を向上させます。

- 書面によるテストケースでは見つからなかった可能性があるバグを特定します。
- 特に締め切りが厳しい場合に短時間で実施できます。
- 創造的な思考を促し、新しいテストシナリオの作成を支援します。
- 予期しない問題を発見することで、製品の品質を向上させます。
- ソフトウェア開発ライフサイクルプロセス(SDLC)のどの段階でも実施できます。

アドホックテストを実施するためのベストプラクティス

- 製品に関する豊富な知識：アプリケーションとその機能を明確に理解します。
- エラーが発生しやすい領域の特定：不具合が発生しやすいモジュールに重点を置きます。
- 重要な機能の優先順位付け：重要でリスクの高い領域を最初にテストします。
- 基本レベルでの計画：正式なテストケースがなくても、テスト対象の大まかなアイデアを維持します。
- 適切なツールの使用：デバッグおよびテストツールを使用して効率を改善します。

1. https://www.youtube.com/
2. https://fdk-stage.cisco.com/c/en/us/support/docs/licensing/ask-licensing/cda/device-management/lic219871-this-is-the-test-article-to-check-the.html
3. https://www.geeksforgeeks.org/software-engineering/adhoc-testing-in-software/

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。