

テストについて

テストとは

テストとは、プログラムのバグを見つけ出す作業です。

正常に動作していると思われるプログラムを破綻させようとする、確信犯的でシステムチェックな作業といえるでしょう。

デバッグとは別物？

テストとデバッグは似ているように感じるかも知れませんが、別物です。

デバッグとは既にプログラムに問題があることがわかっているときに行います。テストは、正常に動いているように見えるプログラムに対して行います。

順序としては、テスト 問題発見 デバッグ、となります。

テストはなぜ必要？

テストを行う理由としては、次の点が上げられます。

潜在的なバグを見つけ出し、つぶす。

お客様へ「テスト仕様書」として提出する。

は当然ですが、 のお客様相手の場合、「テスト仕様書」として提出した内容と実際の動作内容が違った場合は、責任問題になりかねません。テスト仕様書は外向けの資料としても、非常に重要なモノの一つです。

テストを行うに当たって

系統的なテストを

プログラムのテストは系統的に実行することが肝心です。

自分が何をテストしようとしていて、そこからどんな結果を期待するのかを段階ごとに把握しなければなりません。テストは見落としがないように正しく実行する必要があり、またどこまでがテスト済みかわかるよう記録を取っておかなければなりません。

テストの例：マトリックス（行列）を作成する

どのような種類のテストを行うか、またどこまでテストを行ったかを明確にしておけるように、マトリックスを作成・使用することがあります。また、そこで発見されたバグを記録し、そのバグに対応したかどうかを記録する為に、バグ対応表を必ず作成しましょう。

 PCLの作成（マトリックス）

 バグ対応表の作成

 以上の二種類の表を作成する。

テストはインクリメンタルに

テスト作業はプログラムの作成と歩調を合わせて行う必要があります。

とりあえずプログラムを全部書き上げてから一気にテストする「ビッグバン」方式は、次にあげるインクリメンタル方式に比べてはるかに大変ですし時間もかかります。

プログラムのパーツを書いたらそれをテストし、またそれにコードを追加したらまたテスト・・・という風に、変更を加えるたびに関係する個所の動きをテストしていくようにしましょう。このように、一つの閉じた世界に対するテストのことを、『単体テスト』と呼びます。単体テストでいい結果が出る前に、そこを無視して別の場所の開発に移る、ということは絶対に避けましょう。

最終的なテスト

別々に記述された部分を通してテストすることを『結合テスト』と呼びます。これはプログラミングのクライマックスに行くことになるでしょう。各テストをこなしてきた人ならば、ここでヘンな結果がでて、どの辺りが原因か、アタリをつけることが出来るでしょう。また、個々に開発していた為に細かいことがそれぞれわからなくとも、現象と発生手順さえ提出すれば、その部分をコーディングした人がそれを見ればおのずと悪い場所がわかるものです。

テストの種類

正常テスト

正常に操作したときに、正常に結果が出るかを確認するテストです。一番基本的なテスト項目となるでしょう。

エラーテスト

エラーとなる扱いをしたときに、思い通りの振る舞いをするかを試すテストです。エンドユーザーとは、アプリケーションに対して、開発者が思いもよらない操作を行うものです。そういった操作をされた時に、容易にダウンしてしまうようでは、あまりいいシステムとはいえません。従って、誤った操作をされたときもしっかり対処してくれるかどうか確かめる、このようなテストが必要なのです。

境界条件テスト

条件分岐やループ回数が本当に正しいかどうかを確認するテスト。例えば「ユーザーの入力値が 9999 以下であれば～する」といったプログラムがあったとき、9999 を入力したときと 10000 を入力したときの挙動を確認する、といったテストです。

バグというのは、ほとんどこういった境界で起こります。従って、ある部分のコードが失敗するときには境界で失敗する可能性が高く、また逆に境界で正しく動作するプログラムは、他の場所でも正しく動く可能性が高いといえます。

事前・事後の状態テスト

何らかの動作を実行する前（事前）とした後（事後）で、期待される状態が保たれているかどうかを確認するテストです。

ストレステスト

膨大な量の入力を行ったり、膨大な時間、同じ処理をしつづけてシステムが破綻しないかを確認するテストです。例えば大きい数字を入力すると処理に時間がかかり過ぎないか、またサーバシステムなどで同じ処理を 1000000 回繰り返しても負荷が変わっていないか、などを調べます。