目次

カバレッジ100%の難しさ

カバレッジ分析を使用しても、未テスト箇所を完全に無くすのは困難

・テスト漏れ箇所
・デッドコード
・例外処理コード

→カバレッジ100%という目標値を設定できないために、テストを徹底出来ない。

カバレッジを100%に出来ないと

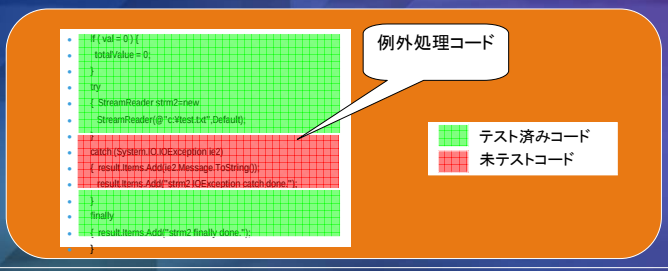
・ソースコード中に未テストの箇所を残したまま次工程に進んでしまい、品質が低下

・後工程のチェックでバグが発見され、手戻りによって工数が増加

品質の低下と工数増加による、プロジェクトのコスト増・納期遅れ

未テストのコードが残る（1）

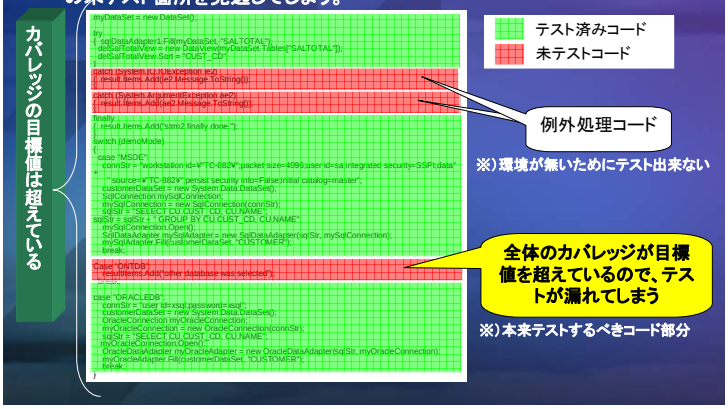
・例外処理のコード部分が、テストできずに残ってしまう。



・例外処理のテストには、ハードウェアによる環境の用意が必要
・テスト環境の準備に十分な工数、予算はとられていない
・例外処理のテストには、高度な技術力も必要

未テストのコードが残る（2）

・カバレッジの目標値を80～90%等にせざるを得ないため、例外処理以外の未テスト箇所を見逃してしまう。



CompuwareJapan

Page 6

チェックと修正にかかる工数が増大

・手戻り作業によって、作業工数が増大してしまう
・運用段階で例外処理に問題が発生すると、システムダウンの怖れ
→システム修復に莫大な工数・コストがかかる

CompuwareJapan

Page 7

解決策の御提案

コンピュウェア製品の活用による
カバレッジ100%の実現

CompuwareJapan

Page 8

カバレッジ100%を実現するDevPartner

DevPartner Studio ：
カバレッジ分析によるテスト網羅率測定

例外処理コード

```
try
{ StreamReader strm2=new
  StreamReader(@"c:¥test.txt",Default);
}
catch (System.IO.IOException ie2)
{ result.Items.Add(ie2.Message.ToString());
  result.Items.Add("strm2 IOException catch done.");
}
finally
{ result.Items.Add("strm2 finally done.");
}
```

テスト済みコード
未テストコード

```
if ( val = 0 ) {
  totalValue = 0;
}
try
{ StreamReader strm2=new
  StreamReader(@"c:¥test.txt",Default);
}
catch (System.IO.IOException ie2)
{ result.Items.Add(ie2.Message.ToString());
  result.Items.Add("strm2 IOException catch done.");
}
finally
{ result.Items.Add("strm2 finally done.");
}
```

DevPartner Fault Simulator ：
例外をシミュレートし、例外処理部分のテストを実現

CompuwareJapan

Page 9

まとめ

・例外部分のテストによって、カバレッジを徹底
→後工程への持ち越しを防ぎ、手戻りによる工数増加を防ぐ
→例外処理のテストを徹底することで、それ以外（通常処理）の未テスト箇所も明確に
→100%という明確な目標を持つことで、テストを徹底出来る

システム品質の向上および、
作業工数の増加防止を実現

COMPUWARE

www.compuware.co.jp