

2章: 問題の管理

ANDREAS ZELLER 著 中田秀基 監訳

デバッグの理論と実践

-なぜプログラムはうまく動かないのか-

学籍番号 60206299

氏名 吉富 楓雅

目次

1. たくさんの問題たち
2. 問題の報告
3. 問題の管理と分類
4. 問題の処理
5. 問題管理システムのメンテナンス
6. 重複を管理する
7. まとめ

たくさん問題達



欠陥の特定

理論的： 欠陥はプログラマがその部分のコードを書いた瞬間に作られる

実際： 欠陥もしくは欠陥の何らかの影響を見つけた瞬間から始まる

**ユーザの問題を解決することは単なるデバッグ
で済む問題ではない**

自分たちで対処



修正したコードの配布



手順の整理と管理



pixta.jp - 2656170

ソフトウェア問題のライフサイクル

ユーザが問題についてベンダに知らせる

ベンダが問題を再現する

ベンダが問題の環境を特定する

ベンダが問題の場所を特定し手元で修正する

ベンダがユーザに修正を配布する

ライフサイクルの整理

- ・ どの問題が現在未解決なのか？

未解決な問題とはソフトウェアに修正しなければならない欠陥があることを意味する。

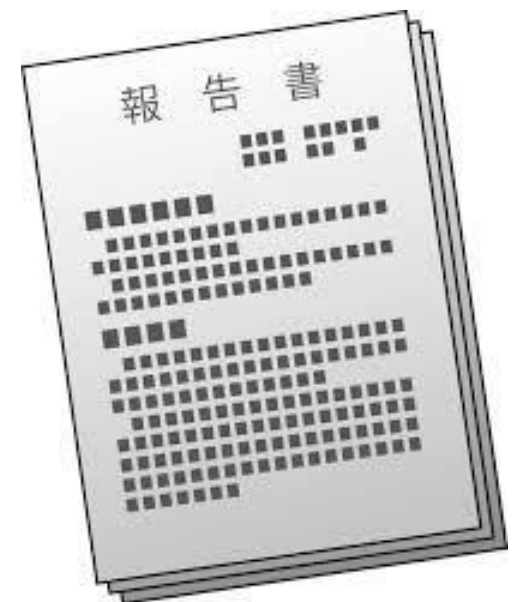
- ・ 最も深刻な問題はどれか？

一般的には最も深刻な問題とは最初に修正されるべきものである。

- ・ 似たような問題が過去に発生しているか？

同様の問題が過去にあれば解決策がすでにあって修正を配布すれば済むだけかもしれない。

問題の報告



問題の報告

開発者は問題を再現



- ・その問題について詳細を解明する手段となる
- ・その問題が修正されたかどうか確認する手段となる

この作業に必要な情報は障害報告(Problem Report: PR)
と呼ばれる

問題の事実

2008年 ザーランド大学

よいバグレポートはどういったものか

Apache, Eclipse, Mozillaの経験豊かな開発者156人を対象

- ・開発者が最も役に立つと考えるものとユーザが地味するものとの間の乖離がある
- ・開発者にとって問題についての事実がなによりも重要

プログラムによって問題が発生するときの診断情報

- ・プログラムによっては実行中に診断のための情報を収集することがある



pixta.jp - 14584470

- ・プログラムがクラッシュすると多くの場合OSは**スタックトレース**を生成する

スタックトレース:

プログラムがクラッシュした瞬間にアクティブであった関数の一覧

スタックトレース

```
01 public class test01 {  
02     public static void func01(){  
03         func02();  
04     }  
05  
06     public static void func02(){  
07         func03();  
08     }  
09  
10     public static void func03(){  
11         //Exception発生  
12         Integer a=10/0;  
13     }  
14  
15     public static void main(String[] args){  
16         func01();  
17     }  
18 }
```

```
01 Exception in thread "main" java.lang.  
ArithmeticException: / by zero  
02     at test01.func03(test01.java:12)  
03     at test01.func02(test01.java:7)  
04     at test01.func01(test01.java:3)  
05     at test01.main(test01.java:16)
```

問題の管理と分類



問題の管理

システムが抱える現在の問題:

単一の「問題リスト」文書を使って管理

- ・ このリストには今までの**すべての**未解決の問題あるいは解決されていない問題が記載されている

問題を管理する別の方法:

問題データベースに**すべての**情報(障害報告)を格納

- ・ 問題データベースは開発者やユーザ問題の数が多くなっても対応できる

問題の重要度

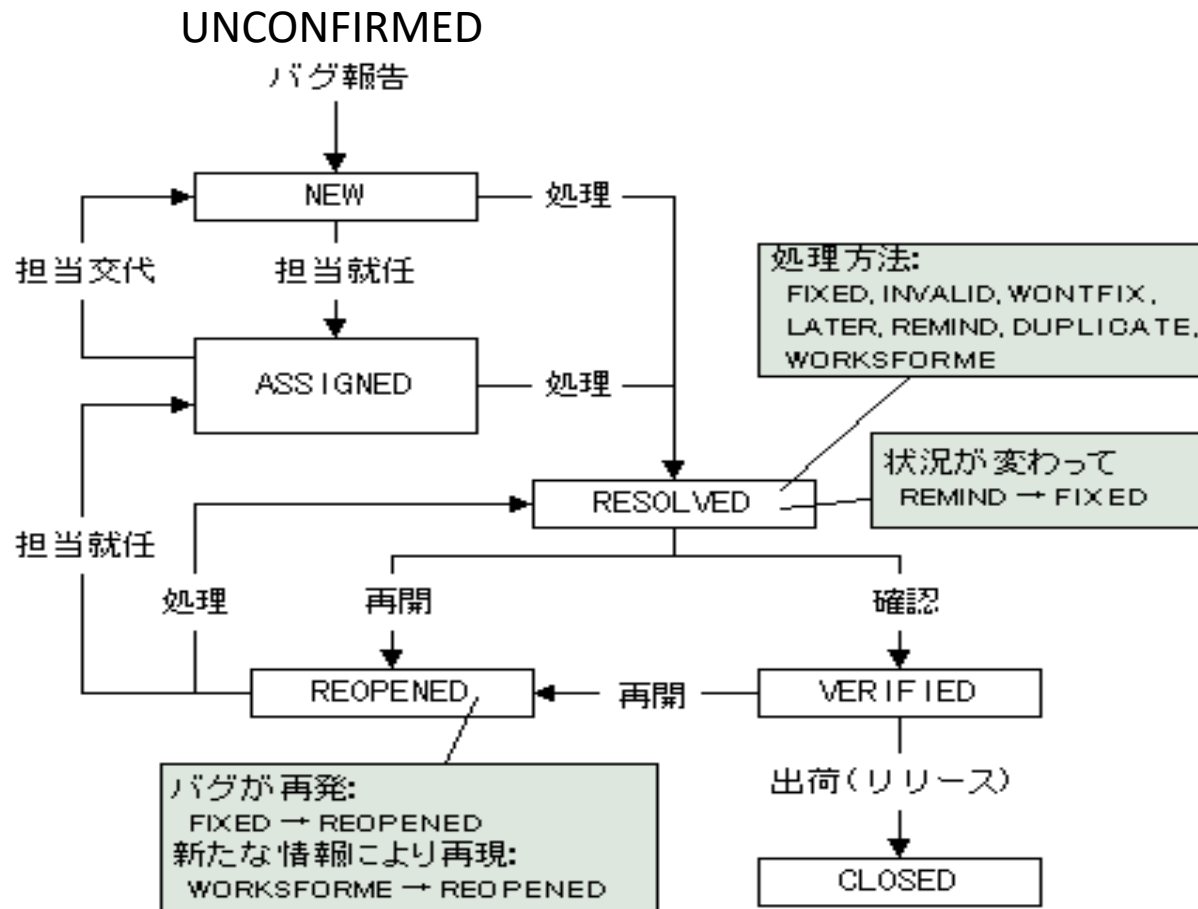
Bugzillaの重要度の分類

- **ブロッカー**: 開発やテスト作業を中断するようなバグ
- **クリティカル**: クラッシュ, データの消失, 深刻なメモリワーク
- **メジャー**: 機能の大きな損失
- **ノーマル**: 「標準的な」問題
- **マイナー**: 機能の小さな損失あるいは簡単な回避策が存在するような問題
- **トリビアル**: 誤字やテキストのズレといった表面的な問題
- **拡張**: 拡張の要求. 障害ではなく機能の要求

問題の処理



問題の処理



問題管理システムのメンテナンス



pixta.jp - 7580946

問題のメンテナンス

だれがメンテナンスするのか？

- ・障害報告の登録
- ・障害報告の分類
- ・優先度の決定
- ・問題の面倒
- ・問題のクローズ

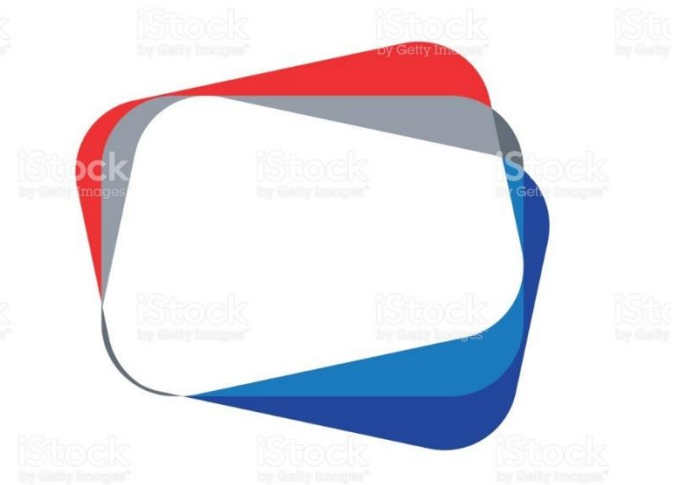


決定

pixta.jp - 2199742



重複を管理する



重複を管理する

WebページAを開いたらクラッシュした！



WebページBを開いたらクラッシュした！



WebページCを開いたらクラッシュした！



重複の解決策

この問題の解決策: **単純化**

障害報告を簡単にしてそのような関係する事実だけを残すようにする

しかし...

- ・**時代遅れの障害報告は明示的に検索されたときに表示されるべき**
- ・Bugzillaでは今後修正されない問題には**WONTFIX(修正しない)**が割り当てられる

まとめ



まとめ

- 問題のライフサイクル
- 問題の重要度分類と処理手順
- 重複を避けるために「単純化」を行う