

3章： プログラムを失敗させる

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

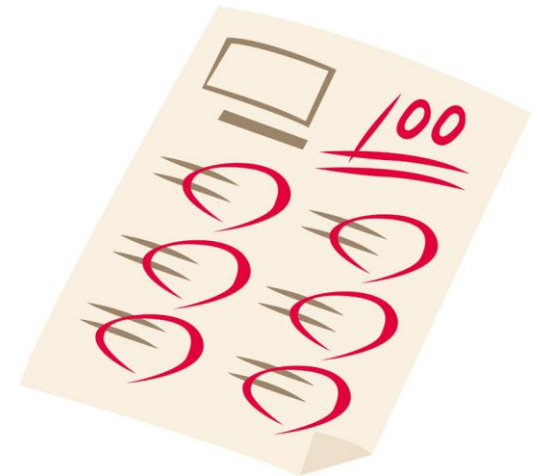
氏名 吉富 楓雅

目次

1. デバッグ用テストの必要性
2. プログラムの制御
3. 3種類の層とテスト
4. まとめ

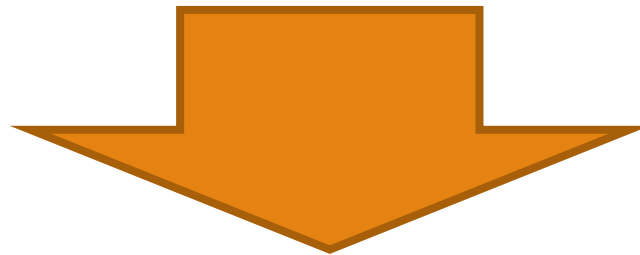
本章では自動化と
問題の分離手法に
焦点を絞りテスト
方法を確認する

1. デバッグ用テストの必要性



プログラムの問題を知る

- ・ユーザからの報告だけがプログラムの問題を知る手段ではない
- ・ほとんどの問題は開発者側のテストで見つかる



テストによってであれユーザからの指摘であれ問題が見つからなければデバッグはできない

妥当性確認テスト

妥当性確認テスト:

- ・テストで問題を見つけてから欠陥を突き止めてそれを修正
- ・未知の問題を検出することが目的



反対に既知の問題を発見することを目的とするテスト
はデバッグ用テスト

テストを行うことの利点

- ・既存のテストを再利用できる
- ・手動で実施するのが難しいまたは不可能なテストを実行できる
- ・テストを繰り返し実行できる
- ・ソフトウェアの信頼度が上がる

自動デバッグの利点

- テストケースの単純化が自動的に行われる
- 以下に関する障害の原因の自動的な分離が可能
 - 障害を引き起こす入力
 - 障害を引き起こすコード修正
 - 障害を引き起こすスレッドスケジューリング
 - 障害を引き起こすプログラムの状態

2. プログラムの制御



障害報告の実例

次の操作を実行するといつも私のマシンのMozillaがクラッシュしてしまいます

- > Mozillaを起動する.
- > bugzilla.mozilla.orgにアクセスする.
- > 「search for bug」を選択する.
- > 画面をファイルに出力する. 下と右の余白は.50に設定する.
(出力先ファイルには, /var/tmp/netscape.psファイルを使用)
- > 一度その操作が終わったら, **もう一度まったく同じことを同じファイル(/var/tmp/netscape.psファイル)に対して行う.**
- > ブラウザはセグメンテーションフォルトでクラッシュする.

自動テストを作成する

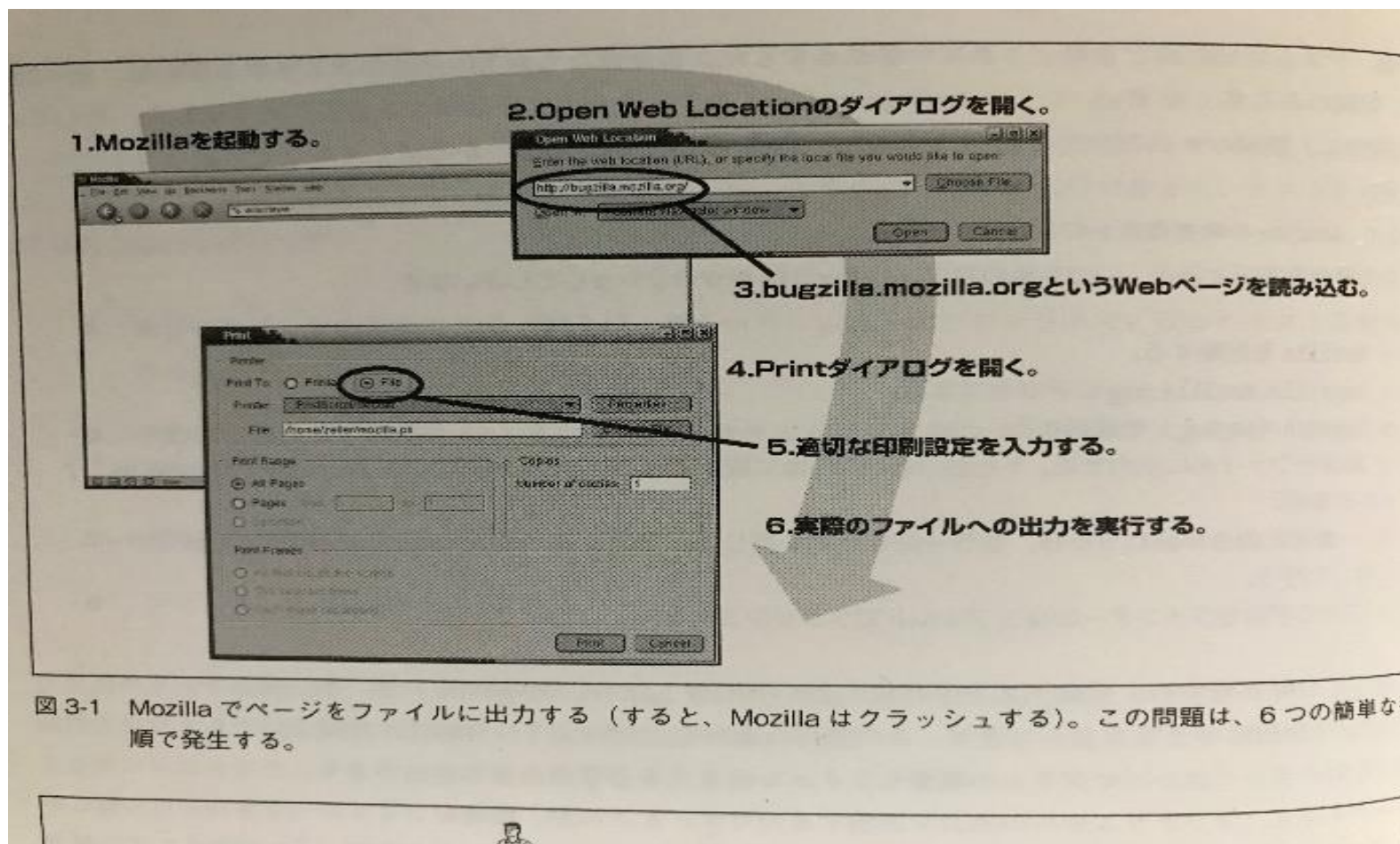
一般的に自動テストではプログラムの環境をシミュレートする必要がある

しかし環境をシミュレートするのは難しい...

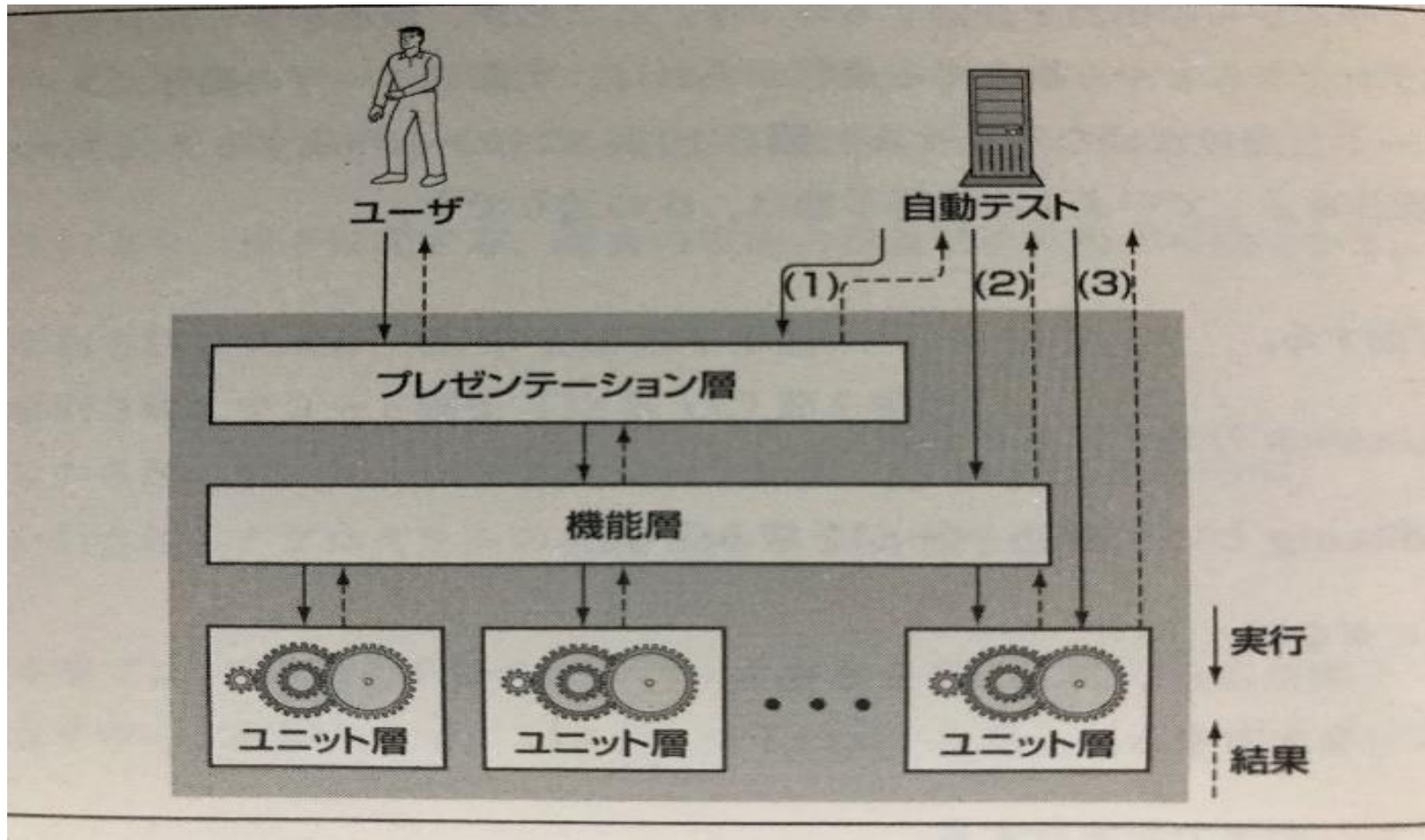
実際のユーザの操作をシミュレートしなければならない



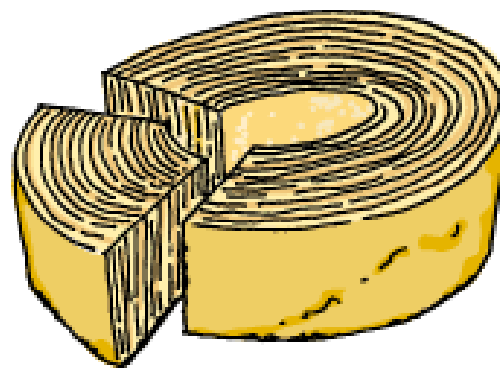
今回の障害のシミュレーション



典型的な3つの層に分類



3. 3種類の層でのテスト



プレゼンテーション層のテスト

プログラムへの入力をシミュレートしプログラムからの出力を監視

利点：

- ・プレゼンテーション層のテストではどのようなプログラムが対象であっても、ユーザの振る舞いをシミュレートし自動化することができる

機能層のテスト

自動化用に設計されたインタフェースがプログラムで提供されていれば便利

利点:

- ・一般にテスト結果の収集と評価が簡単に行える
- ・人間向けに設計されたプレゼンテーション層でのテストが難しい場合でも問題はない

ユニット層のテスト

ここではプログラム全体ではなく特定のユニットだけを自動実行することを考える

利点:

- ・簡単に自動化できる

最近はこのユニット層でのテストが注目されている

外科手術に例えると...

プレゼンテーション層のテスト:

実際に外科医が行う手術を見てテストする



機能層のテスト:

外科医の手術過程のマニュアルについてテストする

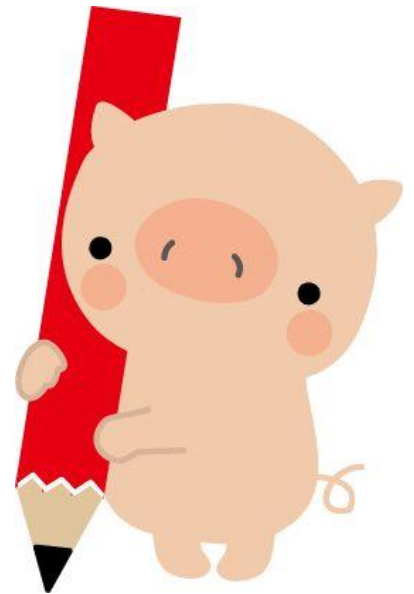


ユニット層のテスト:

手術過程を分離させそれぞれについてテストする



4. まとめ



まとめ

- ・既知の問題を発見するためにデバッグ用テストが必要
- ・自動テストは実際のユーザの操作をシミュレートしなければならないため難しい
- ・プログラムを3つの層に分類しそれぞれにおいてテストするという方法がある