バグのパターンを用いたテストの提案

Japan Symposium on Software Testing 2006
2006年1月30日(月)
電気通信大学　河野 哲也・西 康晴

発表の流れ

- 研究の背景
- 導入
- パターンを用いたテストの提案
- 適用
- 考察
- まとめ

発表の流れ

- **研究の背景**
- 導入
- パターンを用いたテストの提案
- 適用
- 考察
- まとめ

組み込みソフトウェアの不具合が多発

- ソフトウェアの品質の現状
  - 公開されている不具合の事例
    - 自動車のエンジン制御プログラムに不具合
    - 新幹線の自動列車制御装置のプログラムミス
    - 自動車のバッテリーコントロールに不具合
  - 十分なテストが行えていない
    - 普通に動くようにするだけで精一杯

ソフトウェアテストの悲惨な現状

- 機能数の増加に伴い莫大な時間がかかる
  - 機能の組み合わせでテストしようとすると大変
    - 例.携帯電話
      通話中にアドレスを閲覧しメールする
    - 携帯電話では50万件以上の
      テストケースを実施している事例もある
- 品質を確保するための重要な工程である
  - 開発の最終工程であるがゆえに
    打ち切られる可能性も高い

バグを狙ってテストする手法が必要

- より少ない工数で多くのバグを検出する
  - 網羅ではなく、ピンポイントでテストする
- 経験豊富なテスト技術者は勘と経験で
  狙ったテストを行っている
  - バグの傾向をつかみピンポイントでテストしている
    - ノウハウではなく形式知にしなければならない

ソフトウェア開発の悩ましい問題

- 失敗を繰り返す
  - 過去に作り込んだバグをまた作り込む
    - 同じようなバグ、同じ原因のバグが頻発
- バグの情報はあるが誰も見ない、使わない
  - バグの情報はデータベース化はされている
    - 数千の莫大なバグの情報から探すのは困難である
    - 情報をそのまま使うのは無理がある

バグの情報は開発の大きな資産である

- 再利用性を高める必要がある
  - 再利用するにはパターン化が必要
    - 同じようなバグをパターン化する
    - パターンを蓄積する
    - パターンをテストへフィードバックする
- 情報を再利用しテストを改善する

バグのパターンを用いたテストの提案

- バグの傾向をパターン化し、それを使ってテストする
  - パターンをテスト工程へフィードバックする
    - バグの多そうな部分がわかる
  - バグを狙ってテストする
    - テスト工程の改善が行える

発表の流れ

- 研究の背景
- **導入**
- パターンを用いたテストの提案
- 適用
- 適用結果の検討
- まとめ

導入

- 日常的な例でプロセスを説明する
  - 例.風邪を予防する
  - 目的:風邪の事例から風邪を予防したい
- 2つのプロセス
  - パターン蓄積プロセス
  - パターン使用プロセス

風邪をひいた原因の具体例

- 風邪をひいた原因の事例の列挙
  - 雨にぬれた
  - 毎日残業だった
  - カップラーメンばかり食べた
  - 風呂あがりに外出した
  - 運動後着替えなかった
  - 夜更しをした
  - 薄着で外出した

パターン蓄積プロセス

- パターン抽出
  - “似ている原因”に分類する

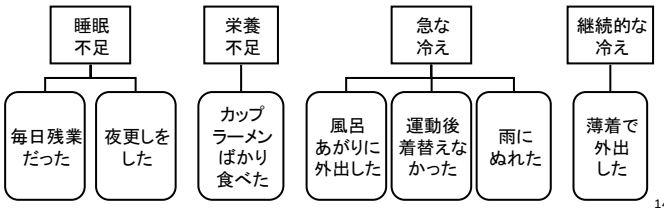
- 雨にぬれた
- 毎日残業だった
- カップラーメンばかり食べた
- 風呂あがりに外出した
- 運動後着替えなかった
- 夜更しをした
- 薄着で外出した

毎日残業だった　夜更しをした　カップラーメンばかり食べた　風呂あがりに外出した　運動後着替えなかった　雨にぬれた　薄着で外出した

パターン蓄積プロセス

- パターン抽出
  - 何が似ているのか分析をする
  - 抽象化を行い
    パターンを抽出する

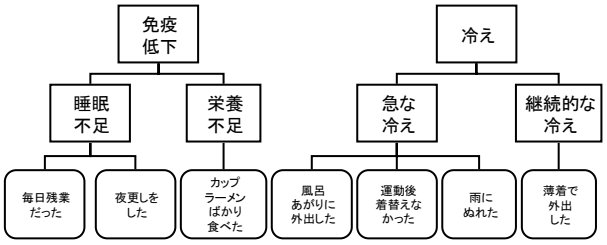
- 雨にぬれた
- 毎日残業だった
- カップラーメンばかり食べた
- 風呂あがりに外出した
- 運動後着替えなかった
- 夜更しをした
- 薄着で外出した



パターン蓄積プロセス

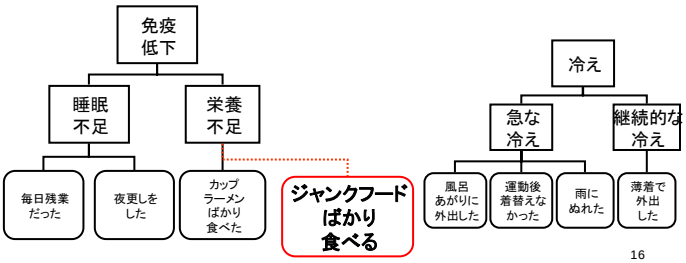
- パターンの構造化
  - 似ているパターンに分類する
  - 親パターンを抽出する
  - パターンツリーを構築する

- 睡眠不足
- 栄養不足
- 急な冷え
- 継続的な冷え



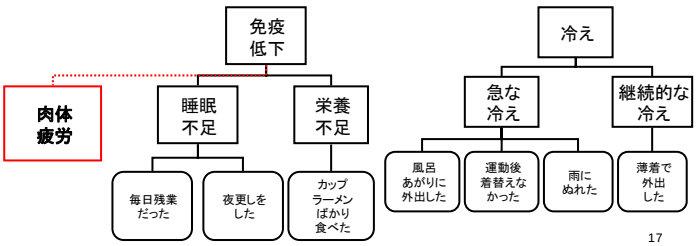
パターン使用プロセス

- パターンより風邪をひきそうな原因が予測できる
  - パターン“栄養不足”から
    ”ジャンクフードばかり食べる”と
    風邪をひくことが予測できる



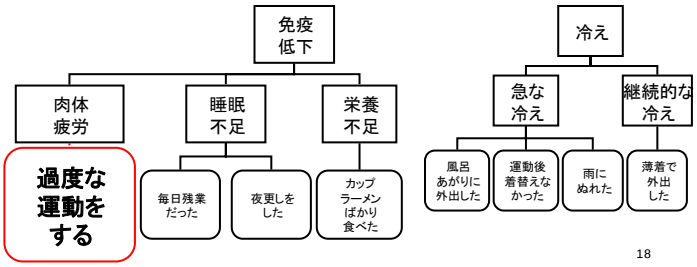
パターン使用プロセス

- パターンツリーより新たなパターンが予測できる
  - 親パターン“免疫低下”から
    子パターン”肉体疲労”が予測できる



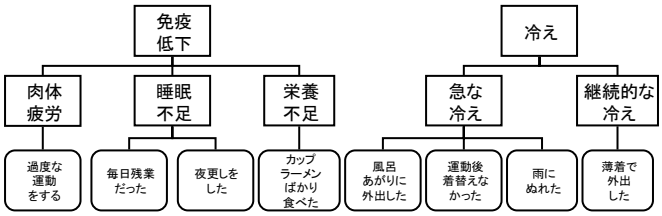
パターン使用プロセス

- パターンより風邪をひきそうな原因が予測できる
  - パターン”肉体疲労”から”過度な運動をする”と
    風邪をひくことが予測できる



パターン使用プロセス

- パターンツリーを観点に風邪の予防ができる
  - 疲れないようにしましょう！
  - 栄養が不足しないようにしましょう！
  - 急に冷えないようにしましょう！



発表の流れ

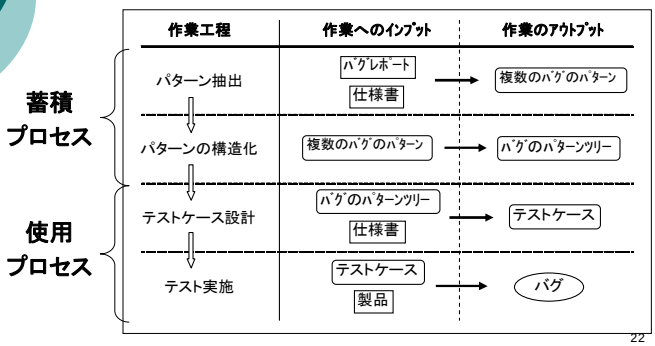
- 研究の背景
- 導入
- **パターンを用いたテストの提案**
- 適用
- 考察
- まとめ

バグのパターンとは

- 発見したバグに既視感(デジャブ)を感じたことはありませんか？
  - このバグは前に見たことあるなぁ
  - この遷移の仕方のバグは前に見たバグと同じだなぁ
    - 決して、バグの事象が同じなわけではない
- バグのパターンが存在している
  - 同じ間違え方、似たような間違え方をしている

バグのパターンを用いたテストの提案

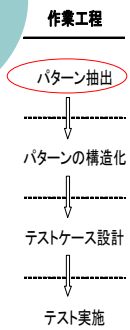
- バグのパターンを用いたテストプロセス



パターン抽出に用いる情報

- バグレポート
  - テスト技術者がバグを検出した時にデータベースに登録する
    - バグの概要
    - 再現手順
    - 修正コメント
- 仕様書
  - テスト設計の時に用いる
    - 製品の機能
    - ユーザーの使い方(ユースケース)
    - 画面デザイン

バグのパターン抽出(パターン蓄積プロセス)



① どのようにしてバグが作り込まれたのかのメカニズムを明らかにする

② バグを似たようなメカニズムに分類する

③ 何が似ているのか分析し抽象化しパターンを抽出する
  - 修正コメントの記述を揃えると抽象化が容易になる

バグのパターンの構造化(パターン蓄積プロセス)

作業工程
パターン抽出
パターンの構造化
テストケース設計
テスト実施

① 似ているパターン毎に分類する
- 似ているパターンもあれば
  全く異なるパターンもある

② 何が似ているのか分析し
抽象化し親パターンを抽出する
- 子のパターン群から親のパターンを抽出

③ 親パターンを似ているパターン毎に分類し
更に新たなパターンを抽出する

テストケース設計(パターン使用プロセス)

作業工程
パターン抽出
パターンの構造化
テストケース設計
テスト実施

① バグのパターンツリーをテストの観点とする
② 仕様書の中からバグを作り込んでるであろう部分を検索する
③ それをテストするテストケースを設計する

テスト実施(パターン使用プロセス)

作業工程
パターン抽出
パターンの構造化
テストケース設計
テスト実施

① テストケースに従い、製品をテストする
② バグが検出されればバグレポートを登録する
- 通常のテスト実施と同様である

発表の流れ

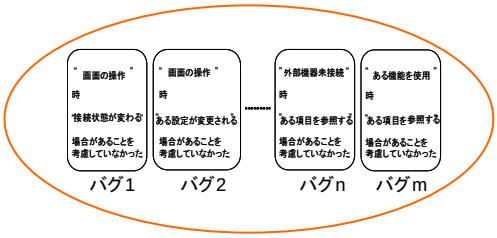
- 研究の背景
- 導入
- パターンを用いたテストの提案
- **適用**
- 考察
- まとめ

実際のソフトウェアのバグへの適用

- 適用対象
  - 組み込みソフトウェア
    - GUI部分であるメニュー機能での約100件のバグ
    - 例.文字化け、文言の間違い、画面遷移せず

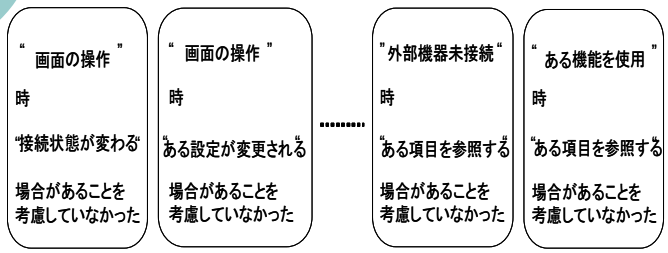
バグのパターン抽出(パターン蓄積プロセス)

① バグが作り込まれたメカニズムを明らかにした
② 数件が同じようなメカニズムとして分類できた



バグのパターン抽出(パターン蓄積プロセス)

- バグのレポートの修正コメントの記述を揃えた



バグのパターンの抽出(パターン蓄積プロセス)

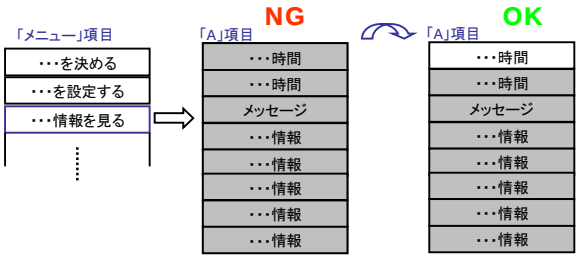
③ 何が似ているのか分析し
抽象化を行いパターンを抽出する
- 「~の時・・・の場合があることを考慮していなかった」と書き換えることができた
- 抽象化を行った
「~する場合がある」のだから通常は「~する」とは違う

- つまり「~する場合がある」とは「例外」のときである

パターン「例外」が抽出できた

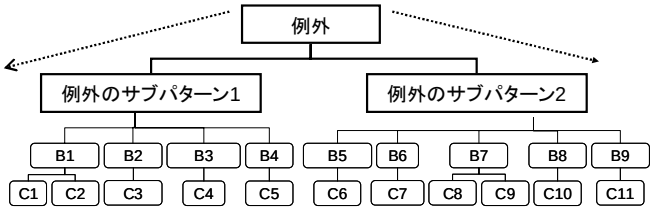
パターン「例外」の具体例

- あるハードウェアを未接続時
「A」項目の一番上以外をすべてグレーアウトする仕様だが
すべてをグレーアウトしてしまうバグ



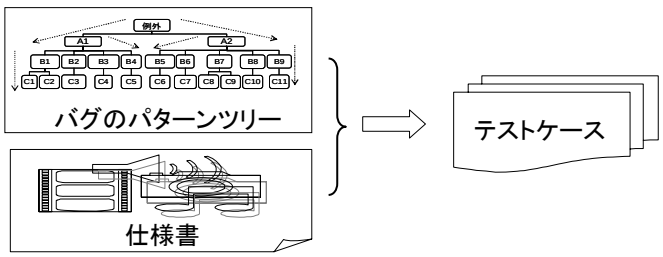
バグのパターンの構造化(パターン蓄積プロセス)

- 「例外」は親のパターンであった
  - 子のパターンを導出した
  - バグのパターンツリーを構築した



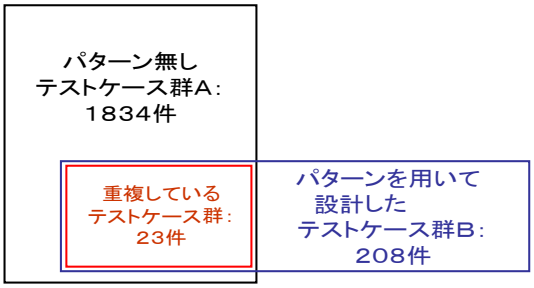
テストケース設計(パターン使用プロセス)

- バグのパターンツリーを観点としてテストケース設計を行った



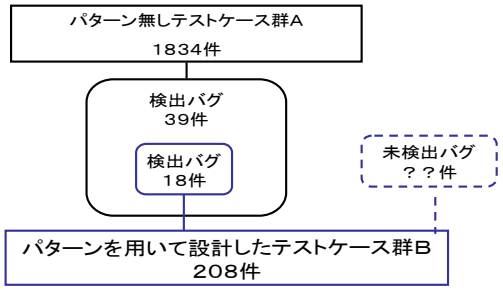
テストケースレビュー(パターン使用プロセス)

- テストケースの比較
  - パターン無しテストケース群Aと
パターンを用いて設計したテストケース群Bを比較した



テストケースレビュー(パターン使用プロセス)

- テストケースと検出バグの関係
  - テストケース群Bで検出できるであろうバグを調査した



テストケースレビュー(パターン使用プロセス)

- テスト精度の評価
  - Aの(パターン無し)テストケース群　:1834件
  - Bの(パターンあり)テストケース群　:208件
  - 重複しているテストケース群　:23件
  - Aで検出されたバグ数:
    39件(39/1834=2.1%)
  - Aで検出されたバグのうち
    Bで検出されるはずのバグ数:
    18件(18/208 = 8.7%)

約4倍のテスト精度向上となった

発表の流れ

- 研究の背景
- 導入
- パターンを用いたテストの提案
- 適用
- **考察**
- まとめ

ところで、、、

- バグのパターン抽出
  1. どのようにしてバグが作り込まれたのかの
     メカニズムを明らかにする
  2. バグを似たようなメカニズムに
     分類する
  3. 何が似ているのか分析し
     抽象化しパターンを抽出する

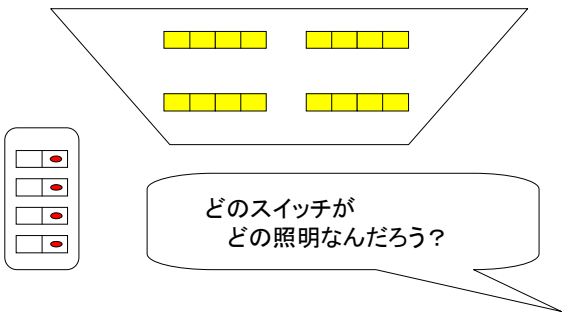
何を観点に分類するの？

プロダクト･アフォーダンス

- 人は誰でも間違える
  - 「例外」で作り込まれたバグは
    仕様書で抜けていたわけではない
  - 考慮できなかった人が悪いのか?
- アフォーダンス
  - モノをどう使うかは
    そのものが情報をアフォードしている(与えている)
    - 例.椅子は座ることを与えている
      床は立つこと歩くことを与えている

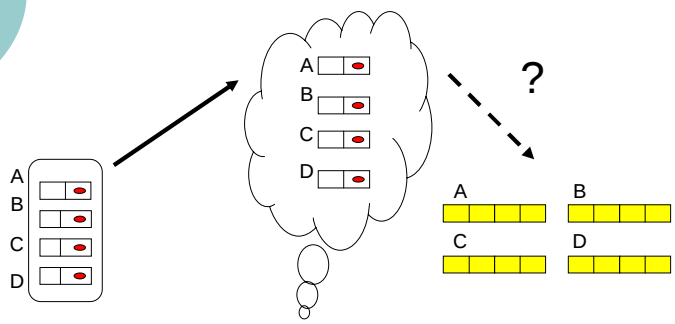
アフォーダンスの悪い例

- 混乱させる会議室の照明のスイッチ



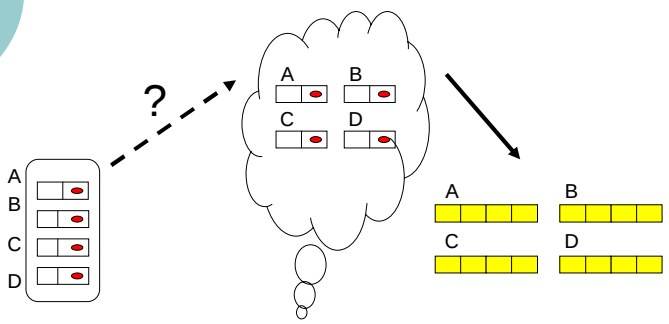
アフォーダンスの悪い例

- 照明を消したい!!



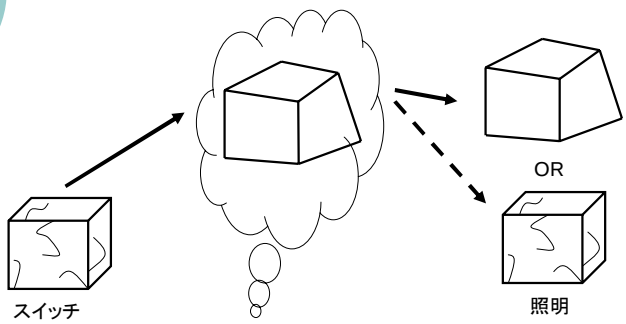
アフォーダンスの悪い例

- 照明を消したい!!



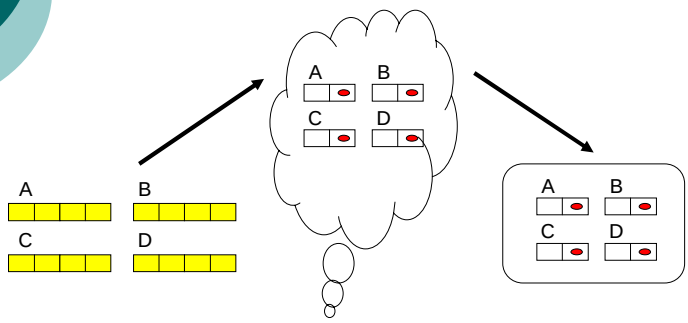
アフォーダンスが悪いと間違える

- アフォーダンスの悪さはヒューマンエラーの温床



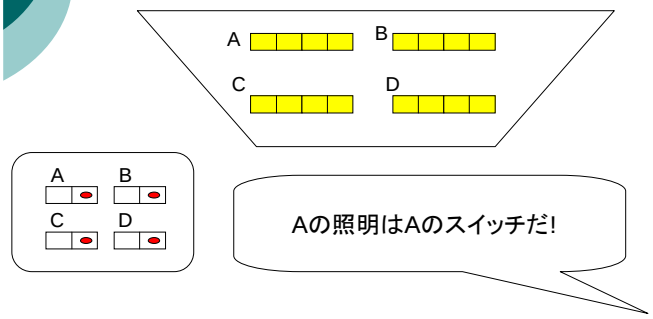
アフォーダンスが良いと間違えない

- 照明を消したい!!



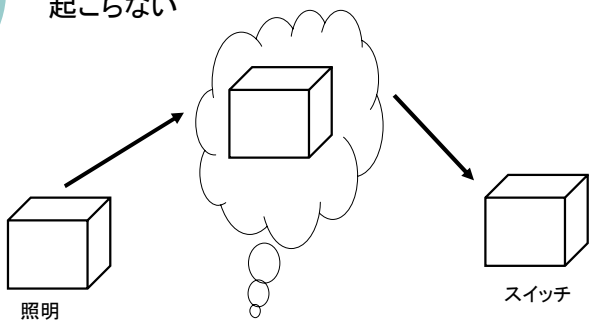
アフォーダンスが良いと間違えない

- 混乱しない会議室の照明のスイッチ



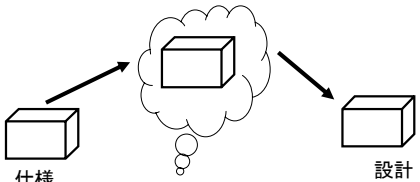
アフォーダンスが良いと間違えない

- アフォーダンスの良いとヒューマンエラーは起こらない



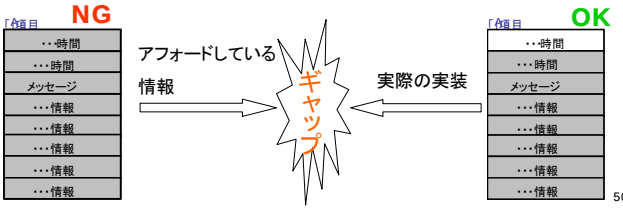
プロダクト・アフォーダンス

- 「どのように実装するかは
  仕様自身が情報を与えている」
  - 仕様書がアフォードしている情報と
    実際に実装すべき情報との間にギャップが生じると
    バグを作り込んでしまう



「例外」とは？

- プロダクト･アフォーダンスでみた「例外」
  - 「たくさんの同じ処理を実装すること」
    をアフォードしている
  - 「1つだけ例外的に違う処理を実装すること」
    はアフォードされない
  - バグを作り込んでしまう



プロダクト･アフォーダンスという観点

- プロダクト･アフォーダンスを観点に
  バグを分類する
  - 「プロダクト･アフォーダンスの悪さ」を見つけよう
  - きっとバグが見つかるはず

発表の流れ

- 研究の背景
- 導入
- パターンを用いたテストの提案
- 適用
- 考察
- **まとめ**

まとめ

- バグのパターンを用いたテストを提案した
- ソフトウェアのバグに適用し
  パターン抽出・パターンの構造化・テストケース設計
  を行った
- プロダクト・アフォーダンスという観点を提案した

今後の課題

- 適用で得られたテストケースの実施
  - どれくらいのバグが検出できるの?
- 実施結果で得られたデータについての検討
  - バグの検出率はどれくらいなの?
- そのほかのパターンの抽出
  - 「例外」以外のパターンは何があるの?
- 違う機能でのバグに対しての適用
  - メニュー機能以外のバグに適用できるの?

ご清聴ありがとうございました

電気通信大学　河野 哲也
kouno@se.uec.ac.jp