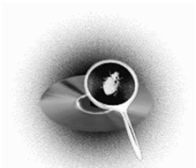
ソフトウェア・ハードウェア複合システムにおける
信頼性情報データベースのためのデータ構造の検討



情報システム学研究科シンポジウム
第9回「信頼性とシステム安全学」
2005/3/1（火）
電気通信大学 システム工学科
西 康晴・河野 哲也

© NISHI, Yasuharu

組込みソフトウェアの信頼性は重要

- 製品のインテリジェント化に伴い、ハードウェアに組み込まれたソフトウェアが製品の信頼性を担うようになってきた
  - 組込みシステム＝ソフトウェア・ハードウェア複合型システム
    - » 高級車の車載ECUは60以上／積載ソフトウェア規模は10年間で16倍
  - 半導体設計もC言語で行われるようになってきている
- 組込みシステムの信頼性は高いとは言えない
  - 品質事故が非常に多い
  - 成熟した開発を進めているとは言えない
    - » 技法もプロセスも規格も十分整備されているとは言えない
  - もともと信頼性要求の低かったものが、安全性を要求されるようになってしまう
    - » 携帯電話が財布になる
    - » カーナビがエンジンに制御信号を出す

© NISHI, Yasuharu

組込みシステムの品質事故の例

- 生命に関わる品質事故の例
  - パトリオット・ミサイルは計時システムのバグで28名の味方の命を奪った
  - 放射線治療器Therac-25は3名の患者の命を奪った
  - メルセデス・ベンツのブレーキシステムにバグがあり68万台がリコール
- 経済的損失を伴う品質事故の例
  - 300万行のうち1行余計だったせいで、AT&Tは11億ドルの損害
  - ARIANE5ロケットはオーバーフローによって爆発し、4億ドルの損害
  - 携帯電話は23万台が回収、42万台が無償交換、数十億円にのぼる損害
- 数え切れない品質事故が発生している
  - デジタル家電、カーナビ、ダム. . .

© NISHI, Yasuharu

信頼性・安全性に関する規格

- 包括的な規格はある
  - IEC61508-3
  - IEEE1228
- 分野ごとの規格が多いが、内容は似たり寄ったり
  - IEC62279（鉄道）
  - IEC60880（原子力発電所）
  - Def-Stan00-55（防衛）
  - IEC60601／FDA guidance（医療機器）
  - DO-178B（航空）
  - MISRA Guideline（車載機器）

信頼性を確保するための
「決め手」となる規格は存在しない

© NISHI, Yasuharu

固有技術による信頼性の確保

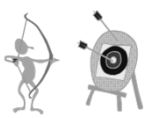
- ソフトウェアの信頼性を十分確保できる開発技術は現在のところ存在しない
  - オブジェクト指向、フォーマルメソッド…
  - 過去の不具合を再発させない開発を改善する仕組みが必要
- ソフトウェアを包括的かつ網羅的に検査することは、現実的には不可能である
  - 網羅的にテストすると膨大な量が必要になってしまう
    - » 携帯電話のテストは機能テスト以降で30万項目以上
  - 過去の不具合を再発させないようテストで検出する仕組みが必要

過去の不具合を水平展開して
未然防止する仕組みが必要

© NISHI, Yasuharu

不具合の水平展開による未然防止

- 開発における未然防止
  - 発生した不具合の情報を「信頼性情報データベース」に蓄積する
  - 過去の不具合と類似した不具合が、対象の設計に発生する可能性を「信頼性情報データベース」から検索する
  - 合致した不具合候補を発生させないよう設計で改善する
- テストにおける未然防止
  - 発生した不具合の情報を「信頼性情報データベース」に蓄積する
  - 過去の不具合と類似した不具合が、対象の設計に発生する可能性を「信頼性情報データベース」から検索する
  - 合致した不具合候補を検出するようテストを改善する
- メリットとデメリット
  - 改善がピンポイントに進むため、プロセスの重量化・硬直化を防止できる
  - 過去に発生した不具合と類似した不具合しか改善できず保証が難しい



© NISHI, Yasuharu

信頼性情報データベースによる不具合の水平展開

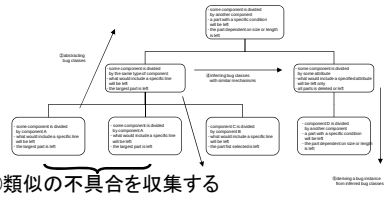
- 水平展開の手順
  - 類似の不具合を収集する
  - 抽象化して不具合の「クラス（分類カテゴリ）」を抽出する
  - 同じクラスに属する新たな不具合候補（インスタンス）を導出する
  - 類似のメカニズムを持つクラスを推測する
  - 推測されたクラスから不具合のインスタンスを導出する

不具合のクラス階層を構築して
より類似した不具合から検索する

© NISHI, Yasuharu

信頼性情報データベースによる不具合の水平展開

- 不具合のクラス階層の構築手順
  ①類似の不具合を収集する
  - » 不具合の事例をインスタンスとして収集する
  - » 不具合の記述を書き直す
  - » 似たような不具合に分類する



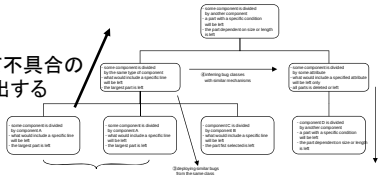
①類似の不具合を収集する

© NISHI, Yasuharu

信頼性情報データベースによる不具合の水平展開

- 不具合のクラス階層の構築手順
  ②抽象化して不具合のクラスを抽出する
  - » 似たようなバグとして分類されたバグを、本質的に共通な部分と異なる部分に区別する
  - » 共通な部分を残し異なる部分の記述を代名詞に置き換えることで、バグのクラスの記述を作成する

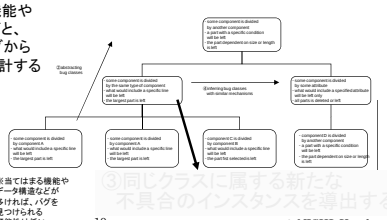
②抽象化して不具合の
クラスを抽出する



© NISHI, Yasuharu

信頼性情報データベースによる不具合の水平展開

- 不具合のクラス階層の構築手順
  ③同じクラスに属する新たな不具合のインスタンスを導出する
  - » クラスの記述に当てはまるテスト対象の機能やデータ構造を探す
  - » 当てはまったクラスの示すメカニズムや兆候を持つバグを具体化する
  - » 当てはまった機能やデータ構造などと、具体化したバグからテスト項目を設計する



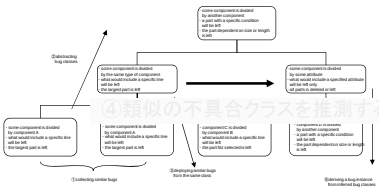
※当てはまる機能やデータ構造などが多ければ、バグを見つけられる可能性は低い

③同じクラスに属する新たな不具合のインスタンスを導出する

© NISHI, Yasuharu

信頼性情報データベースによる不具合の水平展開

- 不具合のクラス階層の構築手順
  ④類似の不具合クラスを推測する
  - » 親クラスを抽出し、その親クラスから新たに可能な子クラスを論理的に推測する

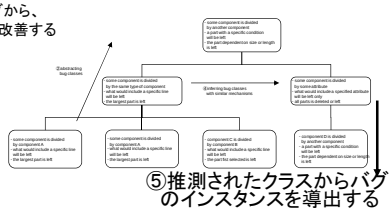


④類似の不具合クラスを推測する

© NISHI, Yasuharu

信頼性情報データベースによる不具合の水平展開

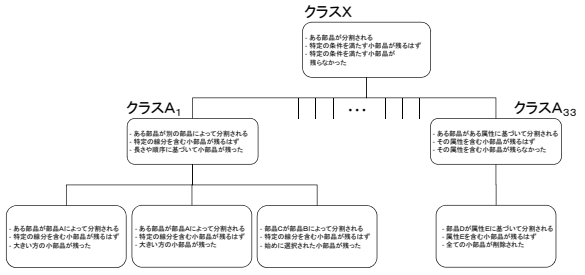
- 不具合のクラス階層の構築手順
  ⑤推測されたクラスからバグのインスタンスを導出する
  - » クラスの記述に当てはまるテスト対象の機能やデータ構造などを探す
  - » 当てはまったクラスの示すメカニズムや兆候を持つバグを具体化する
  - » 当てはまった機能やデータ構造などと、具体化したバグから、開発やテストを改善する



⑤推測されたクラスからバグのインスタンスを導出する

© NISHI, Yasuharu

CADソフトウェアの知識ツリー



© NISHI, Yasuharu

信頼性情報データベースのデータ構造

- 未然防止の精度は、信頼性情報データベースからいかに的確に検索できるかにかかっている
  - データ構造＝不具合のクラスを抽出する際の視点
    - » 質の高いデータ構造によって、精度の高い未然防止が可能になる
    - » 質の低いデータ構造では、未然防止できない改善を行う羽目になる
  - FMEAにおける故障モードのソフトウェア版
- 本研究では、信頼性情報データベースのデータ構造の質について検討した
  - キーワードに着目したデータ構造
  - ソフトウェアの機能に着目したデータ構造
  - ソフトウェアの構造に着目したデータ構造
  - 不具合のメカニズムに着目したデータ構造
  - ヒューマンエラーに着目したデータ構造
- 「日次平均値算出機能」を例に取る

© NISHI, Yasuharu

データ構造の検討：キーワード型

- キーワードに着目したデータ構造
  - 「日次」「平均値」「算出」の3つのキーワードで全文検索を行う
  - キーワードの類似語で全文検索を行う
    - » シソーラスなどを用いる
    - » 「月次」「年次」・・・
  - 構築は容易だが精度は低い
    - » ソフトウェアの内部構造に関する情報を必要としないため、蓄積は非常に容易である
    - » 全文検索のため、検索は非常に容易である
    - » 合致した不具合が、実際に発生するかどうかの保証は全くない

© NISHI, Yasuharu

データ構造の検討：機能型

- 機能に着目したデータ構造
  - 機能ツリーに沿って検索を行う
    - » 週次平均値算出機能や月次平均値算出機能
  - 構築は容易だが精度は低い
    - » ソフトウェアの内部構造に関する情報を必要とせず、機能名のみを必要とするため蓄積は非常に容易である
    - » ツリー検索のため、検索は非常に容易である
    - » 合致した不具合が、実際に発生するかどうかの保証は全くない
    - » ソフトウェア構造が理想的に機能分割されていれば、似たような不具合が発生するかもしれないが、現実的ではない

【機能ツリーの例】
・算出機能
　・平均値算出機能
　　・日次平均値算出機能
　　・週次平均値算出機能
　　・月次平均値算出機能
　・標準偏差算出機能
　　・・・

© NISHI, Yasuharu

データ構造の検討：ソフトウェア構造型

- ソフトウェアの構造に着目したデータ構造
  - 日次平均値算出機能が「スタック」を用いているとする
  - スタック構造を用いている設計を検索する
  - スタックに類似した構造を用いている設計を検索する
    - » キュー、配列
  - 構築はやや難しいが精度はそこそこ高い
    - » ソフトウェアの内部構造に関する情報を必要とする
    - » よく用いられる構造は、構造パターン名で検索できるが、合致数が非常に多いことが推測される
    - » よく用いられていない構造は、パターン認識が必要である
    - » 合致した不具合が、実際に発生する可能性は高い
    - » 不具合の原因は、その構造を採用しているから、ではない

© NISHI, Yasuharu

データ構造の検討：不具合メカニズム型

- 不具合のメカニズムに着目したデータ構造
  - 日次平均値算出機能がスタックを用いており、「オーバーフロー」で不具合が発生したとする
  - オーバフローが起こりうる構造を用いている設計を検索する
    - » 固定長構造を用いているスタックは合致する
    - » 可変長構造を用いているスタックは合致しない
    - » 固定長構造を用いているキューは合致する
  - オーバーフローに類似したメカニズムの不具合が起こりうる構造を用いている設計を検索する
    - » アンダーフロー
  - 構築はやや難しいが精度はそこそこ高い
    - » ソフトウェアの内部構造および不具合メカニズムに関する情報を必要とする
    - » パターン認識が必要なため、検索は容易ではない
    - » 合致した不具合が、実際に発生する可能性は高い
    - » 実はソフトウェア構造型と同じような精度なのでは？

© NISHI, Yasuharu

データ構造の検討:ヒューマンエラー型

- ヒューマンエラーに着目したデータ構造
  - 日次平均値算出機能がスタックを用いており、オーバーフローチェックの見落としによって不具合が発生したとする
  - オーバーフローチェックの見落としが起こりそうな設計を検索する
    - » なぜオーバーフローチェックの見落としが起こるのだろう?
  - 構築が難しく精度も低い
    - » ソフトウェアの内部構造、不具合メカニズム、ヒューマンエラーに関する情報を必要とする
    - » パターン認識が必要なため、検索は容易ではない
    - » ヒューマンエラーの原因が明らかにならない限り、実用的な検索にはならないだろう
    - » 不具合メカニズム型よりも本質を突いている気がする

© NISHI, Yasuharu

ヒューマンエラーの原因の分析

- なぜヒューマンエラーが起こるのだろう?
  - 組込みシステムのGUI部分に発生した不具合約100件について分析した
  - メニューのグレーアウトに起因する不具合が頻発しているので、なんとか未然防止したい
  - グレーアウトであれば必ず不具合が起きるわけではない
    - » 機能型では的確に検索できない
  - グレーアウトは全て同じようなソフトウェア構造を採用している
    - » 構造型でも的確に検索できない
  - どうも「例外的にグレーアウトする場合」のみ起こるようだ
- 「例外的」とは何だろう?

© NISHI, Yasuharu

プロダクト・アフォーダンス

- 「例外的」は、ヒューマンエラーの起きやすい開発対象物の属性である
  - 10数個あるメニューのうち、ごく少数の状況でグレーアウトする場合に発生する
  - 「多くのうち少数に例外的な開発事項が必要になる設計である」と抽象化したらどうだろうか
  - これを「プロダクト・アフォーダンス」と呼ぶことにする
    - » プロダクト・アフォーダンスとは、ヒューマンエラーの起きやすい開発対象物の属性である
    - » ソフトウェアは高度な知的生産物であるため、ハードウェアの物理化学法則はソフトウェアでは人間の認知法則に対応するだろう
    - » 設計と記法の両方にプロダクト・アフォーダンスがあるだろう

© NISHI, Yasuharu

プロダクト・アフォーダンス

- プロダクト・アフォーダンスに着目して信頼性情報データベースのデータ構造を構築すると精度の高い検索が可能になるという予想を行った
  - 設計上のプロダクト・アフォーダンス:
    - » 例外
    - » 複雑な組み合わせ
  - 記法上のプロダクト・アフォーダンス:C言語の場合
    - » 日常生活と違う記法を使うと不具合が起きやすい if (x=0) {・・・}
    - » 開発環境と異なる意味を持つ定義は不具合が起きやすい intの桁数
- 今後プロダクト・アフォーダンスに着目して信頼性情報データベースを構築し、精度を調査する必要がある

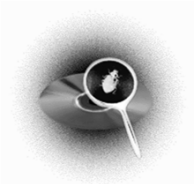
© NISHI, Yasuharu

まとめ

- ソフトウェア・ハードウェア複合型システム(組込みシステム)の信頼性は確保されていない
  - 品質事故が多発している
  - 決め手となる技術も規格も成熟していない
  - 過去の不具合を蓄積して未然防止する必要がある
- 未然防止のための信頼性情報データベースにおけるデータ構造について検討を行った
  - キーワード型や機能型は精度が低い
  - ソフトウェア構造型や不具合メカニズム型は精度がそこそこ高い
  - ヒューマンエラー型は本質的なはずだが実用的ではない
  - プロダクト・アフォーダンスに着目すると精度が高いだろうと予想した

© NISHI, Yasuharu

ご質問やご意見をお願い致します



電気通信大学 システム工学科
西 康晴(nishi@se.uec.ac.jp)

© NISHI, Yasuharu