GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY

JaSST'07 Kansai 基調講演

実践的なテストに向けた試み
ー 21世紀に求められる信頼性のために ー

(1) 故障に耐える (始まりはFTC)
(2) 開発環境の変化
(3) 攻撃にも耐える (20世紀のDC)
(4) ものすごい変化
(5) 市民社会を支える (21世紀のDC)

菊野　亨
大阪大学大学院情報科学研究科

OSAKA UNIVERSITY

岩国: 錦帯橋

3

Edinburg: Forth Rail Bridge

4

(1) **始まりは**FTC

- FTCとは
- 1975年当時の故障
- 故障→エラー→障害
- 有望な技術

始まりはFTC

FTC(フォールトトレラントコンピューティング)とは

- システム中に 故障 があった (そして，それが表面化してきた) としても，システムが正常に動くようにする技術

- 注目すべき視点
  故障 の範囲

  システムが正常に動く で求められている内

始まりはFTC

1975年当時の故障 (自然発生的な故障)

	定常的	過渡的
物理的	製造工程の欠陥 素子の劣化	環境の悪化 素子の劣化
概念的	仕様のミス 設計のミス ソフトウェアのミス	オペレータのミス 入力の過負荷

- 基本的には，自然現象による脅威(natural threat)と人間活動(設計，製造，運用)におけるミス (human error)を含んでいる.

始まりはFTC

故障(fault)→ エラー(error)→ 障害(failure)

例

- プログラムのコーディング中にバグ(故障)を作り込んでしまった．(テストで検出，修正されないままで出荷された)
- あるシステムに実装されたプログラムはそのバグを含む命令を実行されない限り，問題とならない．
- さらに，仮に実行されたとしてもその結果(エラー)が外部に見えてこない限り，やはり問題とならない．
- しかし，いずれそのバグを含む命令が実行され，システムが仕様に反する反応をするとき障害がユーザに見えてくる．

始まりはFTC

有望な技術

- 主なシステム技術
  - 冗長化(符号，データ，プロセッサ)
  - 診断→隔離→回復
  - 再構成
  - チェックポイント・リスタート
  - フェイルセーフ
- 支援ツール
  - フォーマルメソッド (検証)
  - テストベッド
  - シミュレーション

(2) 開発環境の変化

- システムが意味するもの
- 品質が意味するもの
- ライフサイクルが意味するもの

開発環境の変化
システムが意味するもの

制御の複雑さ

単独システム　S

↘ 大規模化　c x S

↘ ネットワーク化　S^c

↘ モバイル化　c^S

開発環境の変化

品質が意味するもの

機能・性能の実現

↘

限定された期間，機能，性能を
若干の劣化を許して維持する．

↘

QoSの保証
曖昧で気ままだが，
要求レベルは厳しく

開発環境の変化

ライフサイクルが意味するもの

特定目的のために，システムの
全てを1回限りで開発

↘

評判の良いシステムは
長く使い続けられる

↘

想定外の目的に利用される
ことも起こりうる

(3) 攻撃にも耐える(20世紀のDC)

- DCとは
- 1995年当時の故障
- DCの現実
- 有望な技術

攻撃にも耐える(20世紀のDC)
DC(ディペンダブルコンピューティング)とは

- 様々な アクシデント(例えば，故障，バグ，システムへの悪意ある妨害など) があったとしても，システムの提供する サービスをユーザの満足するレベルで維持する ようにする技術

攻撃にも耐える(20世紀のDC)

1995年当時の故障 (意図的な故障)

	定常的	過渡的
物理的	破壊 妨害	妨害
概念的	データの盗用 データの改ざん 妨害	

- 基本的には悪意ある攻撃による脅威(human attack)を含んでいる．これ以外に，システム同士あるいはシステムと人間のインタフェースに起こる不具合なども問題になりつつある．

攻撃にも耐える(20世紀のDC)

通信システムのダウン

- 1990年1月
  AT&T社の全国規模での通話及びデータ通信が9時間も途絶した．
- 1991年6月，7月
  東海岸と西海岸の主な都市で電話網がダウンした．
- 原因
  200万行のプログラム中の3行のミス

攻撃にも耐える(20世紀のDC)

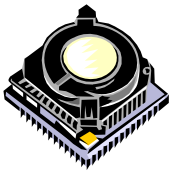
不正侵入，クラッキング

- 1986年
  - スタンフォード大学の60台のコンピュータ
  - シリコンバレーの15の企業
  - クラッカーの侵入
- 1988年
  - コーネル大学の大学院生によるワームプログラムの作成
- 「The Cuckoo's Egg」 Clifford Stoll
- 犯罪としての不正侵入
- 原因
  - インターネットの脆弱性
  - セキュリティの弱点

攻撃にも耐える(20世紀のDC)

企業責任

- Intel社のPentiumチップの浮動小数の除算にバグがあった.
  - 強烈な反対広告
  - インターネットでの論争



- Walt Disney社のDVD "Lion King"にバグがあった.
  - 数千件の苦情

問題点が判明したときの誠実さと
顧客サービスこそがビジネスとして有用である.

攻撃にも耐える(20世紀のDC)

有望な技術

- 主なシステム技術
  - セキュリティ
  - 分散化
  - 高信頼プロトコル
  - IV & V (Independent Verification and Validation)
- 支援ツール
  - モデル検査
  - フォールトインジェクション
- 実践的なテスト技法
  - All-pairs --- テストケースの絞込み
  - Fault-prone prediction --- テスト対象の絞込み

実践的なテスト技法を求めて (1)

― All-pairs ―

テストケースの絞り込み

All-pairs

背景

- ソフトウェアの機能増大は，あまりにも多くのテストケースを生み出している．
  - 全数テストは100年かけてもできない状況．
- フォールト（不具合）の多くは，少数のパラメータ値の組み合わせによって顕在化する
  - Kuhn等の研究
    - R.D.Kuhn et al. “Software Fault Interactions and Implications for Software Testing,” *IEEE Transactions on Software Engineering*, 30(6), June 2004

All-pairs

背景

- N個のパラメータにより健在化したフォールトの割合(%)
  - フォールトに関係しているパラメータ数(N)

N	組込み	Web browser	HTTP server	DB
1	66	29	42	68
2	31	47	28	25
3	2	19	19	5
4	1	2	7	2
5		2	0	
6		1	4	

1,000～10,000 Lines of Code　　100,000 LoC

All-pairs

All-pairs法

- Kuhn等の結果
  - 大部分のフォールトは，少数のパラメータの値の組み合わせによって，顕在化する
- 2つのパラメータの値のペアを中心にテストすることで，テスト効率を上げられる！
- All-pairs
  - パラメータの全てのペアについて，パラメータ値の組み合わせのすべてをテストする

All-pairs

テストケース例

- Webベースのアプリケーションのテスト
  - ブラウザ(NN, IE, Opera)
  - OS (Win, Lin, Mac)
  - 接続 (LAN, ISDN, PPP)
  - プラグイン（RealPlayer, MediaPlayer, QuickTime）

$3^4 = 81$ ➡ **9**

全数テスト　　**All-pairs**

	Browser	OS	Conn	Plugin
テスト1	NN	Win	LAN	Real
テスト2	IE	Lin	ISDN	Real
テスト3	Opera	Mac	PPP	Real
テスト4	NN	Lin	PPP	Media
テスト5	IE	Mac	LAN	Media
テスト6	Opera	Win	ISDN	Media
テスト7	NN	Mac	ISDN	QT
テスト8	IE	Win	PPP	QT
テスト9	Opera	Lin	LAN	QT

All-pairs
全数テストとAll-pairs

- パラメータの数に対して必要なテストケース数
  - 各パラメータのテスト値の数が3個のとき

パラメータ数	全数テスト	All-pairs
2	$3^2 = 9$	9
3	$3^3 = 27$	9
4	$3^4 = 81$	9
5	$3^5 = 243$	11
6	$3^6 = 729$	12
13	$3^{13} = 1594323$	15
16	$3^{16} = 43046721$	17

All-pairs
事例(JaSST'06東京)

- All-pairsを利用した評価に関する報告
  - 出典
    - All-pair法を利用した携帯電話組込み用モバイルFeliCa ICチップファームウェア評価に関する報告、太田 豊一（フェリカネットワークス）ら、JaSST' 06東京
  - 背景
    - 携帯電話組込み用モバイルFeliCa ICチップファームウェアの開発
  - 従来手法とAll-pairsとの比較実験

All-pairs
実験概要(JaSST'06)

- 実験手順
  - 特定モジュールに対するテスト項目を作成
  - 各手法によりテストケースを生成
  - コード実行率(カバレッジ)を評価
    - コード実行率とフォールト発見率に相関があるから
- 比較対象
  - 従来手法
    - 仕様に精通した担当者が、仕様を評価するテストケースとして作成
  - All-pairs

All-pairs
評価結果(JaSST'06)

- All-pairsの利点
  - テスト項目数が半分程度と少なく、効率的である
  - コード実行率が優れている

	従来手法	All-pairs
テスト項目数	75個	30個
A 関数	100%	100%
D 関数	90%	100%
F 関数	76%	91%
H 関数	96%	98%
I 関数	96%	96%

コード実行率

※一部抜粋

All-pairs

事例の考察

- All-pairs
  - コード実行率が高いため、高いバグ発見能力が期待できる
  - テストケース数を効率的に削減できる
  - 全ペアを網羅する
  - しかし、最小のテストケースの生成は難しい

All-pairs

研究内容

- グリーディ法
  - 選択基準に基づいて、最も良いテストケースを1つずつ生成していく手法
- 新たな選択基準
  - テスト環境の設定にかかるコスト

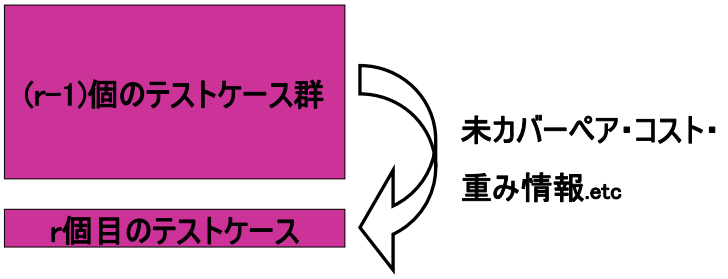
All-pairs
グリーディ法

- テストケース選択アルゴリズム
  - アルゴリズム自体はNP困難*
  - そこで、一度に生成するのではなく、グリーディ法を用いてヒューリスティックに求める
- グリーディ法
  - 局所的に最適のものを選択し、解を求める手法

*C.J. Colbourn, Combinatorial aspects of covering arrays, Le Matematiche (Catania) 58 (2004), 121-167.

All-pairs
テストケース選択アルゴリズム

- 良いテストケースを一つ一つ生成
  - 良い＝未カバーペアを多く含む
  - 全てのペアを網羅するまで



All-pairs

グリーディ法の利点

- 生成されるテストケースの性質
  - 良いテストケースほど，早く生成される
    - テスト時間に制約がある場合に有効
- テストケース生成アルゴリズムの性質
  - アルゴリズムの設計が容易
  - 計算時間が小さい
  - 禁則処理への対応が容易

All-pairs

参考文献

- 英文のサーベイ
  - Colburn, “Combinatorial Aspects of Covering Arrays”, Le Matematiche (Catania) 58, pp. 121-167, 2004.
  - M. Grindal, J. Offutt, S. F. Andler. “Combination Testing Strategies: A Survey,” Journal of Software Testing, Verification and Reliability, vol.15, No.3, pp.167-199, 2005.
- 日本語で読めるもの
  - コープランド，はじめて学ぶソフトウェアの技法，日経BP, 2005.
  - 秋山, “直交表による組み合わせテスト入門”，ソフトウェアプレス, vol.2, pp. 89-107, 2005.

実践的なテスト技法を求めて (2)

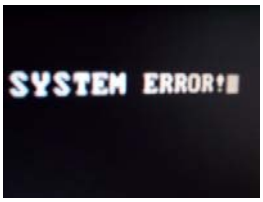
－ Fault-prone Filtering －

テスト対象の絞り込み

Fault-prone Filtering

Fault prone prediction

- Fault prone (FP:不具合のありそうな)モジュール
  - Fault proneモジュールにテストを集中させれば，テスト効率が向上する事を期待できる
  - KhoshgoftaarらはFault-proneモジュールを予測する効果として，テストコストが半減すると主張.
  - 1990年代以降，
    数多くの研究がなされてきた.



Fault-prone Filtering

従来の研究：概説

- FP予測は長い歴史を持つ分野
  - 多様な予測手法、メトリクス、データセットでの実験
- しかし，工数予測におけるCOCOMOのようにデファクトスタンダードとなった手法は存在しない.
- FP予測手法の主なアプローチ
  - ロジスティック回帰
  - 分類木
  - その他

Fault-prone Filtering

アプローチ例: ロジスティック回帰

- 変数の統計的回帰モデルの一種
  - ある事象が発生する確率を直接予測する
- ソフトウェアモジュールが不具合を含む確率を計算
  - 説明変数: ソフトウェアメトリクス
  - 目的変数: 不具合の有無(又は，不具合を含む確率)

Fault-prone Filtering

ロジスティック回帰の例

- YがFPとなる確率

$$p(Y = FP) = \frac{e^{Linear}}{1 + e^{Linear}}$$

Linear = $C_0+C_1X_1+C_2X_2+\cdots+C_nX_n$

X_i:説明変数, C_i:係数

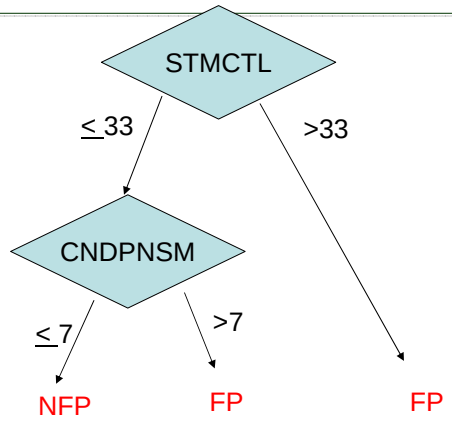
- 例:
  - X_1: NIP (継承関係に無いポリモルフィズムの数)
  - X_2: OCMIC (exportにおけるクラスとメソッドのカップリング)
  - X_3: OCMEC (inportにおけるクラスとメソッドのカップリング)

L.C. Briand, W.L. Melo, and J. Wust. Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans. on Software Engineering, 28(7):706-720, 2002.

Fault-prone Filtering

アプローチ例: 分類木

- 葉が分類を表し、枝のラベルがその分類に至るまでの条件を表す



STMCTL:制御文の数,
CNDPNSM:分岐からの辺の総数

T. M. Khoshgoftaar and E. B. Allen. Controlling overfitting in classification tree models of software quality. Empirical Software Engineering, 6(1):59--79, 2001.

Fault-prone Filtering

ソフトウェアメトリクス

- Halstead metrics:プログラムの大きさ
  - オペレータ
    - 制御構造，演算子，関数
  - オペランド
    - 変数，定数
- McCabe metrics：手続きの複雑さ
  - 制御の基本パス数(分岐の数+1)
- コード行数
- オブジェクト指向メトリクス

Fault-prone Filtering

過去の研究で使用されたデータ

- 公開されているもの
  - NASA’s Metrics Data Project
  - PROMISE project
- オープンソース (自力で収集)
  - Apache, Xpose, Jwriter, Eclipse
- 独自のデータ (企業などからの提供)
  - 大規模な通信系組み込みシステム
  - 医療系システム

Fault-prone Filtering

Fault-prone予測の考察

- FP Predictionの分野は成熟している
  - 長年にわたり，多様な手法が試みられてきた.
- なぜデファクトスタンダードとなる手法が無いのか
  - それほど精度が高いものではない
    - 不具合モジュールの6,7割程度が予測できる程度
  - 予測には多くのメトリクスが必要
    - COCOMOで測るものとは別
- 効果の割に手間がかかると敬遠されているのでは.

Fault-prone Filtering

我々の提案 (Fault-Prone Filtering[1])

- 手間を削減:
  ソースコードのみを使用してFPを予測する
  - 従来使用していたメトリクスを使用しない
  - **スパムフィルタを利用する → 適用労力の削減**
  - ある意味，「ものぐさ」な手法
- 精度評価
  - 一致度は 80%前後 (従来手法よりやや良)
  - 再現率 (本物の不具合を不具合と予測できる率) が高い

Mizuno and Kikuno, ``Training on Errors Experiment to Detect Fault-Prone Software Modules by Spam Filter,'' In The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (**ESEC/FSE2007**), (to appear).

Fault-prone Filtering

基本的なアイディア

- SPAMメールのフィルタリング
  - 電子メール内の単語の頻度情報をBayesianフィルタで処理し，これまでに観測されたSPAM, non-SPAMグループのどちらに属するかを計算
- Fault-prone フィルタリング
  - ソフトウェアモジュール内の単語の頻度情報をBayesianフィルタで処理し，これまでに観測されたfault-prone, non-fault-proneのどちらに属するかを計算

Fault-prone Filtering

手法の概要: 評価

- 評価基準
  - 一致度: 全体的な精度 $(N_1+N_4) / \Sigma N_i$
  - 再現率: $N_4 / (N_3 + N_4)$
    - 実不具合を予測が網羅した率
  - 第II種の過誤率: $N_3 / (N_2+N_4)$
    - NFPとした予測が外れる率

		予測	
		NFP	FP
実測	NFP	N_1	N_2
	FP	N_3	N_4

手法	一致度	再現率	第II種の過誤率
回帰分類木(CART)	0.699	--	0.149
ロジスティック回帰	0.906	0.682	---
PCA+ロジスティック回帰	0.840	0.483	0.727
FP Filtering	**0.835**	**0.839**	**0.044**

＊一致度はデータの偏りなどに大きな影響を受けるので目安にはなるが評価には使えない.

Fault-prone Filtering

Fault-prone Filtering: まとめ

- ソースコードからFPを予測する手法を提案
  - 精度は過去の研究に劣らない
  - 使用する材料はソースコードのみ
- 参考文献

T. M. Khoshgoftaar and M. Seliya “Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study” Empirical Software Engineering,9,pp.229-257,2004

L.C. Briand, W.L. Melo, and J. Wust. Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans. on Software Engineering, 28(7):706-720, 2002.

O. Mizuno and T. Kikuno, ``Training on Errors Experiment to Detect Fault-Prone Software Modules by Spam Filter,'' In The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE2007), (to appear).

(4) ものすごい変化

- 21世紀市民社会におけるシステム開発の課題
- 人間系を含めた評価を前提とする
  環境に優しいシステム作り

ものすごい変化

キーワード

- CS (顧客満足度)と 環境 (もったいない)

ヒント: BSE問題，無農薬リンゴ

1. どの牧場で，誰が生産したか
2. 加工，流通プロセスの安心・安全
3. コスト高を納得の上で選択できる
4. 説明責任*，トレーサビリティ
5. 地球温暖化への配慮

*市民に理解できるように説明する

ものすごい変化

CSを得るために

1. どの牧場で，誰が生産したか
   - 開発プロセスのレベル (CMMIなど)
   - 開発担当者のモラール
   - 利用したモジュールの品質，性能
2. 加工，流通プロセスの安心・安全
   - 運用，管理プロセス
3. コスト高を納得の上で選択できる
   - 20世紀のDC
   - コストーリスク評価による選択
4. 説明責任*，トレーサビリティ
   - わかりやすい状況説明
   - チェックポイント
5. 地球温暖化への配慮
   - 低消費電力化
   - ライフサイクル

(5)市民社会を支える(21世紀のDC)

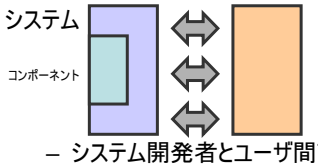
- 振り返ってみると
- 求められる保証

21世紀のDC

振り返ってみると

- 1975年 (FTCの時代)

  - コンポーネント開発者とユーザ間での品質，性能保証(FTC)
- 1995年 (DCの時代)

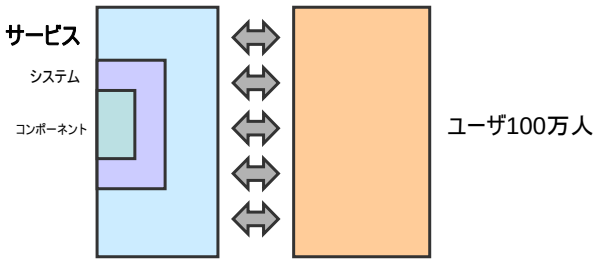


  - システム開発者とユーザ間でのサービス維持についての保証(DC)
    (但し，サービスを開発者視点で評価)

21世紀のDC

振り返ってみると

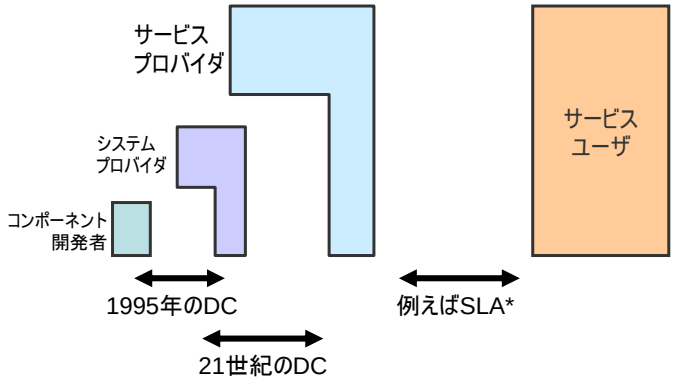
- 2010年



- 開発者とユーザ間でのサービス保証
  (但し，サービスをユーザ視点で評価)

21世紀のDC

振り返ってみると



*Service Level Agreement

サービス提供側とサービス利用者側の間でサービスレベルに関する取り決めをする.
サービスレベルとコストは一般には対立関係にある.

21世紀のDC

求められる保証

- 来るべきユビキタス社会あるいはアンビエント社会の中で市民が安心・安全な生活，活動を楽しめることを保証する．

より具体的には

- 運用段階に入っているシステムの提供するサービスが，合意されている，あるいは常識的な判断の下で，ユーザの満足するレベルに常に維持されること．また，システムがそうした特性を有することがユーザに明示的に提示されている．さらにそれがいわゆるブランド力となり，システム(あるいはサービス)を選ぶ上でのユーザの判断材料となる．

Thanks

- 参考図書

(1)「IT社会の法と倫理」
(A Gift of Fire: Social Legal and Ethical Issues in Computing) サラ・バーズ著，日本情報倫理協会訳 (2002年)

(2)「フォールトトレラントシステム論」
当麻喜弘編著，電子情報通信学会 (1990年)

(3)「情報化社会の安全と信頼を担保する情報技術の構築 ―ニュー・ディペンダビリティを求めて―」
CRDS-FY2006-SP-07、科学技術振興機構　研究開発戦略センター