賢くソフトウェアをつつく・直す

探索・学習によるソフトウェア工学

ERATO-MMSDプロジェクト グループ３「インテリジェンス協働形式手法」

どんな問題？

ソフトウェアが扱う領域はますます増えており、システムの不具合検出やその修正などの様々なタスクは非常に難しく人手で太刀打ちできないものになりつつあります。

どんな研究？

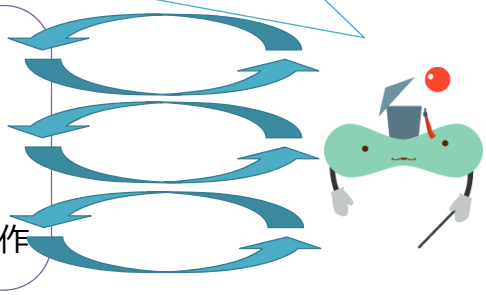
ソフトウェア工学における様々な問題を、探索・学習の技術を活用し、賢くポイントを絞り込んだり、傾向を学習したりすることによって効果的・効率的に解くことを目指しています。

探索・学習技術の活用

例： 自動ブレーキソフトウェアの不具合発見

設定１でやってみたら歩行者まで最小８m，
設定２では３mだったぞ
設定２を少しいじるともっと危ないケースが出るかな？

シミュレーションによりテストするときの多数のパラメータ・設定項目
- 初期速度
- 先行車や歩行者の位置
- 路面状況・カーブや坂
- ドライバーのアクセル・ブレーキ・ハンドル操作



ポイント絞り込みや傾向の把握により「欲しいもの」にどんどん近づけていく！**あぶり出す！**

※ 遺伝子の進化を模倣した進化計算などメタヒューリスティックの考え方を適用

様々な応用
- 危険なケースなど要件に違反するテストケースを生成する
- バージョンアップで挙動が大きく変わってしまうテストケースを生成する
- 網羅性や多様性が高いテストケースリスト（テストスイート）を生成する
- 正しさの基準を満たすような設計やコードの断片や修正を生成する
- ・・・

理論も活用してスピードアップも！ → ポスターA14

機械学習システムのテスティング

by 石川 冬樹

新しい難しさがある！
→ ポスターA10

＜例＞
- 各入力に対する出力の正解を決めるのが困難または不可能で，正解と照らし合わせられない
- 少し入力（例：画像）をいじるだけで，思いもしない形で出力が変わることがある
- 原則不完全（100%になりえない）ので，どこまで制度を追い求めればよいかわからない

「あぶり出し」テストケース生成を活用！
- 前のバージョンと大きく結果が変わるような入力を見つける
- 小さな入力の変化で大きく出力が変わる場合を見つける
- システム全体での問題（事故など）につながるような誤検出に絞って見つける

強化学習によるテスティング

by 加藤 亙貴

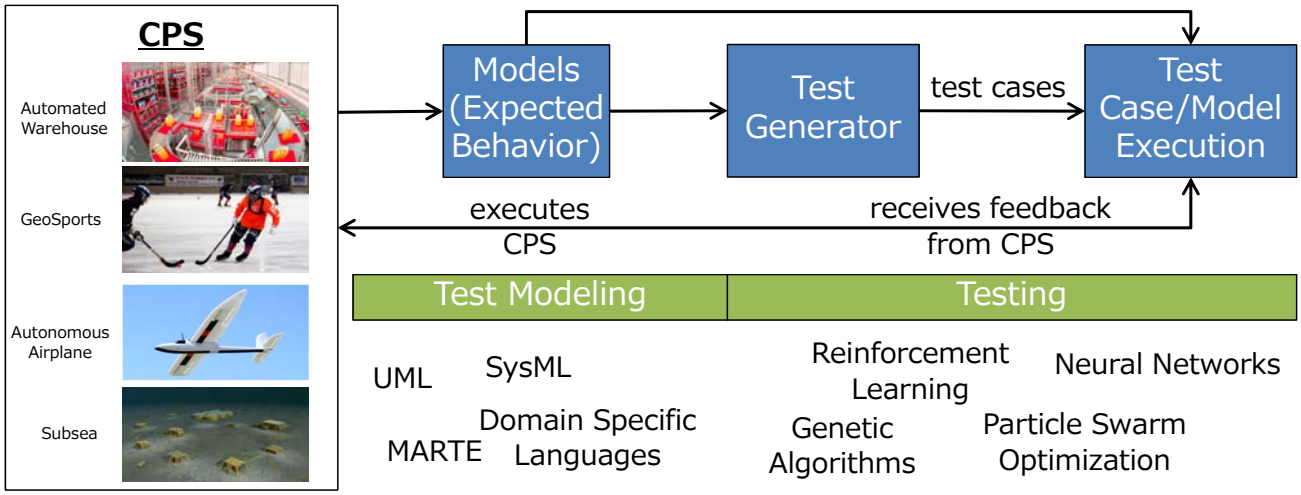
危ないケースなどの「あぶり出し方」を訓練し
少し環境・製品が変わってもさっと検査

バリエーション
- 製品ファミリー
- 設計時のパラメータ等の試行錯誤
- 想定環境

Testing Cyber-Physical Systems (CPS)

by Shaukat Ali

- **Application Domains:** Automotive, Avionics, Maritime, Oil&Gas, etc.
- **Objective:** Ensure dependable operation/behavior in real environment , e.g., safety, security, correctness
- **Challenges:** Unpredictable physical environment, black-boxes, interactions among software, hardware, human, information networks, …
- **Methods:** Model-based Testing, Search-based Testing, Machine Learning



Using Mutation for Testing and Repairing Code/Models

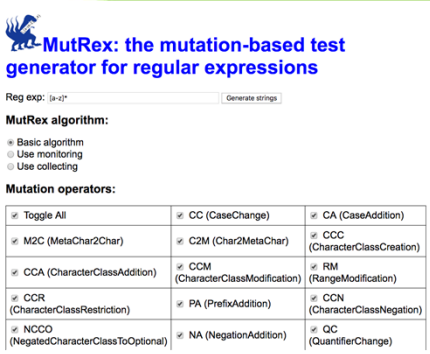
by Paolo Arcaini

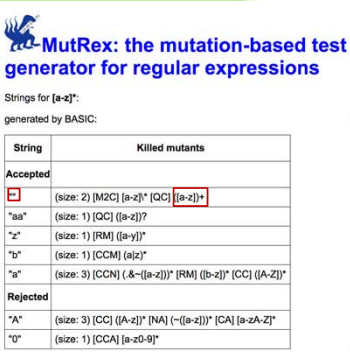
Fault-based testing

- Some faults are more frequent than others. E.g.:
  - developer wrote **if**(var < 5) instead of **if**(var <= 5)
  - developer wrote **[a-z]*** instead of **[a-z]+**
- *Fault-based testing* is used for two purposes:
  - checking if a test suite detects all the faults
  - **generating** a test suite for detecting all the faults
- Fault-based testing is applied to: code, formal models, **regular expressions**, **feature models**

Fault-based test generation for regular expressions

- We identified typical faults in regular expressions (e.g., wrong multiplicity operator)
- The tool **MutRex** builds tests for capturing such faults in a regular expression under test
- The user is the *oracle*



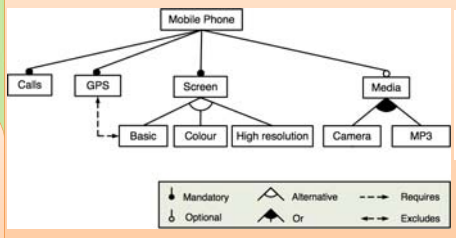


Automatic repair

- Automatic repair aims at **removing faults** from SW artefacts

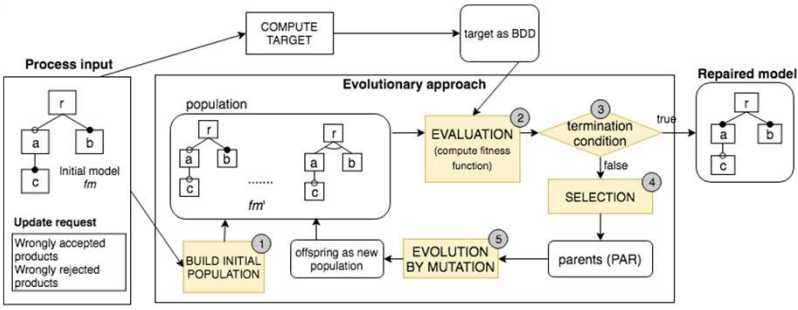
Automatic repair of feature models

- We consider **feature models** and suppose to have a set of desired **update requests** (from failing tests)



Failing tests
- Products with *Camera* and *Basic* are wrongly accepted
- Products without *GPS* are wrongly rejected

- **Evolutionary approach for automatic repair**
  - In mutation testing, **mutants** are used to detect faults
  - If a SW artefact contains a fault, the corresponding mutant does not: **Let's use it for repairing the feature model!**



NII
連絡先： 石川 冬樹 （国立情報学研究所 アーキテクチャ科学研究系 准教授）
Email： f-ishikawa@nii.ac.jp Web : http://research.nii.ac.jp/~f-ishikawa/