

## 研究の概要

### 背景：ソフトウェアシステムのビジネスルール

ソフトウェアシステムが現実世界で解決する問題は多様  
 - 問題ごとに異なる文脈とルールがある

本を借りているのは常に図書館会員  
 ダブルブッキングが起こらない  
 A 部門の人が読めるファイルにファイル X の内容がコピーされることはない

きちんと考えないと、正しく作ったつもりでも望まぬ結果に...  
 - 文脈やルールの設計を中心とした開発手法が注目されている

### 課題：表に表れないビジネス上の制約

開発者が記述するのはソフトの振舞い  
 現場の担当者に重要なのはビジネス上の制約

「本を借りているのは常に図書館会員」かな？  
 運用してみた限り、とりあえず OK そう？でも本当に大丈夫かな...

暗黙的な制約は、開発者が意識しない → テストされない  
 暗黙的な制約について説明されない

1 year later...  
 パージョンアップします！  
 退会処理を変更しよう... (未返却本チェックを誤って無効に)

新しい開発者  
 図書館システム v2  
 その可能性見落としてた！  
 リリース後の修正すごく大変なのに...

本を借りたまま退会した人が出ちゃった！

### 切り拓く未来

「本を借りているのは常に図書館会員」ってこと言えますよね？  
 ツールが隠れた制約をあぶり出す

提案ツール  
 利用者への説明にもなる  
 それなら安心！

未返却本がない場合、退会処理を受け付ける  
 貸し出し時には会員チェック

図書館システム

検査項目 (テストケースなど) に反映しよう  
 検査項目に入っているから、後の開発でもバグにすぐ気付ける！

検査項目をより豊かに → バグを予防

振舞いだけでなく制約も明示的に扱う開発を促進 + ビジネスルールの説明可能性を向上  
 利用者が安心できる高信頼なソフトウェア社会へ！

### 成果

ソフトの振舞いの形式仕様での表現に対し  
 振舞いが常に満たす制約 (不変条件) を自動で獲得する手法

- 変異解析・自動修正問題として定式化
- 論理式の性質を利用した変異操作を定義
- 実装・事例研究

プログラムコードと形式仕様の接続手法構築の試み

ソフトの振舞いの本質を論理式で抽出したもの  
 ビジネスルールの本質に焦点を当てる  
 ソフトウェア工学分野でホットな自動修正のアプローチ

## 成果の詳細

### 基本方針：変異解析を応用した不変条件の解析

振舞い仕様  
 変異した (欠陥を埋め込まれた) 振舞い仕様たち  
 検査項目 (不変条件)

発火条件  $f(x,y) \ \&\& \ g(y,z)$   
 状態変化  $x := \dots, y := \dots$

発火条件  $true \ \&\& \ g(y,z)$   
 状態変化  $x \ \&\& \ f(x,y) \ \&\& \ g(y,z)$

検査対象をわざと壊し、それを見つけられるか？検査項目を検査 (変異解析はもとはテストコードの解析技術)

どれだけ/どの変異を発見できる？

### 式の強化操作・弱化操作

形式仕様記述言語 Event-B の全ての述語と式について操作を定義  
 - 述語を **確実に** 強化 (弱化)

- $P \vee Q$  を強化 ...  $\perp, P, Q, P \wedge Q, (強化版) P \vee Q, \dots$
- $S \subseteq T$  を弱化 ...  $T, (S \text{ の subset}) \subseteq T, S \subseteq (T \text{ の superset}), \dots$
- 集合の式を **確実に** 拡張 (縮小)
- $S$  を拡張 ... (全体集合)  $\cup, Su$  (仕様中出现する同じ型の式), ...
- $S$  から  $T$  への関数を縮小 ...  $\emptyset, (S \text{ の subset})$  から  $T$  への関数,  $S$  から  $(T \text{ の subset})$  への関数,  $S$  から  $T$  への全単射関数, ...

既存の変異解析手法よりも体系立った変異

### 実装・事例研究

形式仕様記述言語 Event-B の開発環境 Rodin のプラグインとして実装

- フラッシュメモリ用のファイルシステムの仕様
- 水星探査プロジェクト BepiColombo の X 線分光計ソフトの仕様
- Ada プログラムコードから抽出された、衛星の姿勢・軌道制御システムの仕様

対象の不変条件と振舞いを選択  
 自動で強化された不変条件を出力

### 振舞い仕様の変異と検出・不変条件強化の定式化

不変条件を満たす状態の集合 (まだ弱い)  
 仕様にある振舞いの到達可能状態の集合

不変条件が弱いので変異を検出できなかった...  
 変異した振舞いの到達可能状態の集合

不変条件を強めてこの変異を検出できた！  
 変異した振舞いの到達可能状態の集合

検出問題：振舞いの到達可能状態が不変条件からはみ出る？ (論理式として定式化)

振舞いの到達可能状態は複雑で把握しづらい  
 不変条件：振舞いの到達可能状態なら必ず満たす制約 → 振舞いを説明する性質

到達可能状態が広がるような欠陥 (発火条件弱化・状態変化拡張) を振舞いの仕様に埋め込む

元の振舞いの到達可能状態の満たす制約がより詳しく判明 → 振舞いの「学習」が進んだ！

最終的には振舞いにとっても近く、しかも簡潔な制約の獲得を目指す

ポイントを突いて制約を強化・振舞いを説明

振舞いに隠れていた制約「本を借りているのは常に図書館会員」をあぶり出せた！

true に変異 (発火条件弱化)  
 元の不变条件  $i$   
 貸出情報 ∈ 蔵書 → 人  
 強化された不变条件  $i'$   
 貸出情報 ∈ 蔵書 → 会員

ソルバ  
 変異後は  $i$  に収まる？ yes  
 $i$  はまだ弱い... 強化して  $i'$  を作ろう  
 変異後は  $i'$  に収まる？ no

### プログラムコードと形式仕様との接続の試み (layered architecture の場合)

「ドメイン駆動設計」で開発されたコードと Event-B 形式仕様との接続を模索  
 ドメイン駆動設計のパターンと仕様上のパターンの対応を分析

- アプリケーションサービス
- ドメインサービス

エンジニアから注目されている、実用的かつビジネスルールを明示的に扱う設計手法

UI  
 アプリケーション  
 ドメイン  
 インフラ