トップエスイー修了制作

静的コード解析ツールの指摘における複雑度を用いた確認優先度付け手法の提案

日本電気株式会社　渋谷 健介　k-shibuya@bq.jp.nec.com

開発における問題点

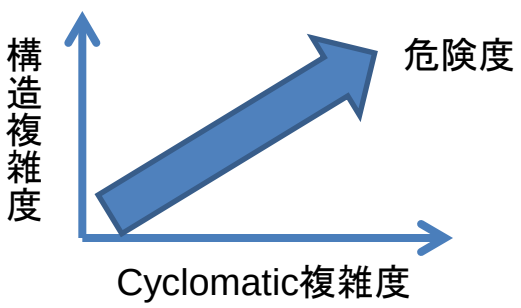
ソフトウェアの品質向上のために静的コード解析ツールを使用することは有用であるが，指摘が大量に検出されることが多々あり，全ての指摘を確認する工数がない，多数の指摘をどれから見るべきかわからない，といった課題が挙がっている．このため，危険度の高い指摘から順に確認できる手法が求められている．

手法・ツールの提案による解決

複雑度メトリクスの高いファイルに対する指摘から優先的に確認する手法を提案する．
複雑度の測定には『Cyclomatic複雑度』と『構造複雑度』の2つのメトリクスを用い，双方が共に高い指摘から順に確認することで，危険度の高い指摘から優先的に確認することが可能となると考える．

提案手法概要

- 提案手法
  - 既存研究にて有効性が示されている『Cyclomatic複雑度』と『構造複雑度』の2つのメトリクスを用い，双方が共に高い指摘から優先的に確認



Cyclomatic 複雑度（循環的複雑度）
- プログラムを有向グラフとして表現したときの，線形独立な経路の数
- 複雑度が大きいほどバグを作りこみやすいと言われている
- 4つのカテゴリに分類される

カテゴリ	Cyclomatic複雑度
Very High	51～
High	21～50
Medium	11～20
Low	1～10

構造複雑度
- 参照数（Visibility Fan Out: VFO）と被参照数（Visibility Fan In: VFI）の2つの値を組み合わせ，4つのカテゴリに分類する
- 複雑度が大きいほど修正時の影響範囲が広くリスクが大きい

カテゴリ	VFO	VFI
Core	High	High
Control	High	Low
Utility	Low	High
Peripheral	Low	Low

評価結果

- 評価概要
  - 自社製品開発16プロジェクトにて静的コード解析ツールにおけるバグ指摘と提案手法で用いる2つの複雑度との相関を調査

- 評価結果とまとめ
  - どちらの複雑度とも相関関係があり，組み合わせるとより高まる
  - 複雑度の高いファイルに対する指摘から優先的に確認することは効果的であると考えられる

バグ比率	Peripheral	Utility	Control	Core	計
Very High	0.44	1.36	0.56	0.66	0.63
High	0.17	0.32	0.18	0.24	0.20
Medium	0.09	0.17	0.12	0.09	0.10
Low	0.02	0.04	0.04	0.07	0.03
計	0.02	0.14	0.14	0.22	0.09

※バグ比率＝各カテゴリにおける「バグ数」/「ファイル数」
※使用した静的コード解析ツール：C++test
※使用した複雑度測定ツール：Understand, Lattix