解釈を変更可能なプログラムソースコード保守性評価ツールの実現と適用

早稲田大学大学院　基幹理工学研究科
高田 正樹　鷲崎 弘宜　遠藤 匠
株式会社小松製作所　開発本部　ICT開発センタ
佐藤 雅宏　杉村 俊輔　関 洋平

1

Agenda

- 1. 背景
- 2. 品質評価ツールとGQM
- 3. 品質評価ツールの問題点
- 4. 問題の解決法
- 5. 使用したGQMモデル
- 6. 作成したツールの説明
- 7. 実験とその評価対象
- 8. 結果と考察
- 9. 今後のアプローチと課題

2

背景

- ソフトウェア開発が大規模化かつ派生開発へ移行
  - ソフトウェアを1から開発するより効率が良い
- 派生開発を行うにあたって、品質評価が重要
  - 保守性・移植性の評価が高いと、派生開発を行いやすい
- ソフトウェア品質評価ツールの使用
  - ソースコードを静的解析する
  - GQM法に基づいて品質を評価

3

品質評価ツールとGQM

オージス総研のAdqua (アドクア)

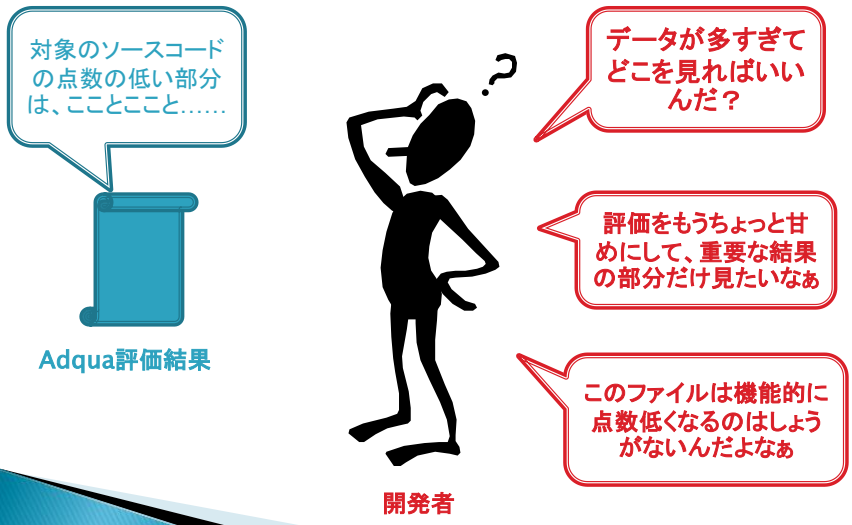
- GQMを得点化・75点未満を赤く表示する

解析性の詳細

Question	Sub Question	Metric		得点
Q041000 不要なコードをそのまま残していないか (71.72点)	Q041001 意味のない式・演算・ステートメントを残していないか (71.72点)	MFI093	重複インクルード数	65.97
		MFI097	未使用の実引数の数	44.58
		MFI098	未使用のラベルの数	100.00
		MFI127	参照していないヘッダをincludeしている数	7.28
		MFI136	副作用のない文の数	100.00
		MFI297	冗長な*演算子の数	100.00
		MFI334	使用しない関数の数	75.77
		MFI464	未使用の定数の数	
		MFn007	到達しないステートメントの数	62.91
		MFn041	使用・再使用されない変数の数	57.34
Q042000 記述場所が適切か (81.58点)	Q042001 ヘッダファイルに存在すべき宣言をソースファイルに記述していないか (75.14点)	MFI152	ソースファイルで宣言された外部結合を持つ変数や関数の数	64.26
		MFI331	ヘッダファイル内に存在するべきtypedef宣言の数	100.00
	Q042002 ソースファイルに存在すべき定義をヘッダファイルに記述していないか (72.85点)	MFI189	ヘッダファイル内に存在する、外部結合を持つオブジェクト・関数の定義の数	72.85
	Q042003 ファイル中での記述場所は適切か (100.00点)	MFI327	ブロックスコープで宣言された関数の数	100.00
		MFI328	ブロックスコープで宣言された結合をもつオブジェクトの数	100.00
		MFI329	ブロック内で定義された#defineまたは#undefの数	100.00

4

ソフトウェア品質評価ツールの問題点



5

問題の解決法

- 測定データが多すぎる

→評価したいクエスチョンだけに絞り、クエスチョンの追加・変更・削除を自由に行える

- 重要な結果だけを見たい

→メトリクス評価の閾値を自由に変更できる

- 開発者の意図にそぐわない

→開発者がクエスチョン毎に評価式を設定できる

6

使用する拡張GQMモデル

- 解釈を伴うGQMモデルの例

Goal 品質	Question 指針	Metric メトリクス	Interpretation 解釈	Result 評価結果
G1: ・・・	Q1: ・・・	M1: ・・・ M2: ・・・	M1> 0 or (M1 =0 and M2 > 0)	NG
	Q2: ・・・	M3: ・・・	M3 >= XX	NG

- 実際に使用した拡張GQMモデル（Question2のみ）

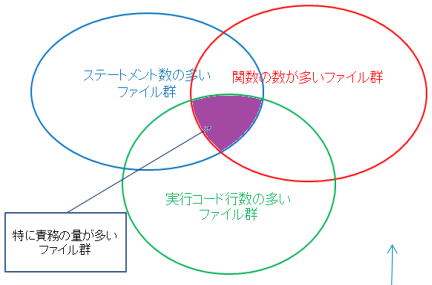
ID	Question	Metrics
2	1ファイルの責務の量は適切か	MFI004 ステートメント数
		MFI019 関数の数
		MFI477 実行コード行数
	(MFI004 =< 100 || MFI019 =< 10 || MFI477 =< 150) && (MFI004 < 300 && MFI019 < 25 && MFI477 < 300)	

7

使用したQuestion例

Q2　責務の量は適切か？

- MFI004 ステートメント数
- MFI019 関数の数
- MFI477 実行コード行数

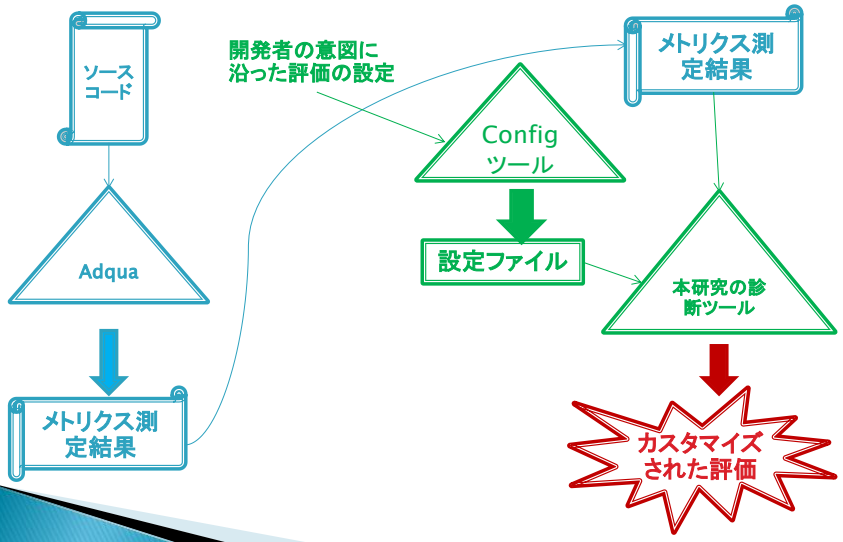


- クエスチョンがOKと評価されるための条件式

(MFI004＜300&&MFI019＜25&&MFI477＜300)&&
(MFI004≦100 || MFI019≦10 || MFI477≦150)

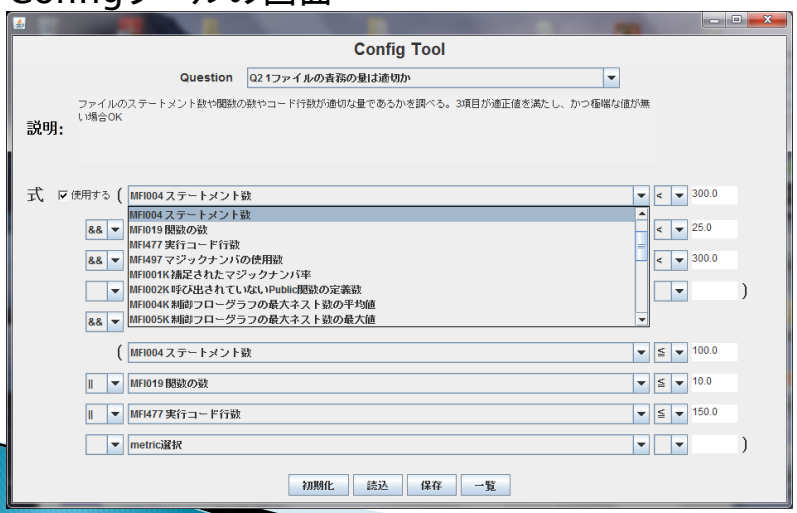
8

作成したツール概要



Configツール

- Configツールの画面



10

診断ツール

- 診断ツールの出力結果例

SystemReport

レポート作成日時
2013年 月 日 時 分 秒
プロジェクト名:

Qestion名

Q1:不要な引数をそのまま残していないか
Q2:1ファイルの責務の量は適切か
Q3:マジックナンバを不必要に用いてないか
Q4:Publicでの定義は適切であるか
Q5:関数の処理が複雑すぎないか
Q6:演算の順序をわかりやすくしているか
Q7:コードクローンは多すぎないか
Q8:外部結合の数が異常でないか、また外部結合である必要あるか
Q9:対象プログラムに対して依存している外部の要素は限定されているか
Q10:対象プログラムが依存している外部の要素は限定されているか

File名	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
sample1.cpp	NG	OK	NG	OK	OK	OK	OK	OK	OK	OK
sample2.cpp	NG	NG	OK	NG	NG	NG	NG	OK	OK	OK
sample3.cpp	NG	OK	OK	OK	OK	OK	OK	NG	OK	OK
sample4.cpp	NG	OK	OK	OK	OK	OK	OK	OK	OK	OK
sample5.cpp	NG	OK	OK	OK	OK	OK	OK	OK	OK	OK
sample6.cpp	NG	OK	OK	OK	OK	OK	OK	OK	OK	OK

11

診断ツール

- 診断ツールの出力結果例

Question2 Report

Question2
1ファイルの責務の量は適切か

使用している条件式
(MFI004 < 300 && MFI019 < 25 && MFI477 < 300)
&&(MFI004 ≦ 100 || MFI019 ≦ 10 || MFI477 ≦ 150)

メトリクス
MFI004:ステートメント数
MFI019:関数の数
MFI477:実行コード行数

デフォルト条件式
(MFI004 < Y1 && MFI019 < Y2 && MFI477 < Y3)
&& (MFI004 < X1 || MFI019 < X2 || MFI477 < X3)
X:1つでも満たせばよい境界値
Y:すべてが満たされていなければいけない境界値

赤字は直接クエスチョンをNGとしているメトリクス
青字はメトリクスの条件を満たしてはいないが直接クエスチョンをNGとしていないメトリクス

File名	MFI004<300	MFI019<25	MFI477<300	MFI004≦100	MFI019≦10	MFI477≦150	改善ポイント
sample2.cpp	false	true	false	false	false	false	S1 S3 S2
sample9.cpp	false	false	true	false	false	false	S1 S2 S3

system reportへ

12

実験とその評価対象

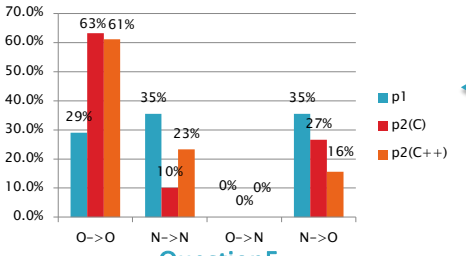
- 今回、Adquaの評価結果と診断ツールの評価結果を比較する実験を行った
- Adquaは診断ツールで使用したメトリクスの得点の平均値を使用し、75点以上ならOK、それ以外はNG

プロジェクト	p1	p2
開発言語	Cのみ	C,C++
ソースコード行数	約5万	C:約2.5万,C++:約15万
総ファイル数	100以下	C:100以下 ,C++:500以下

13

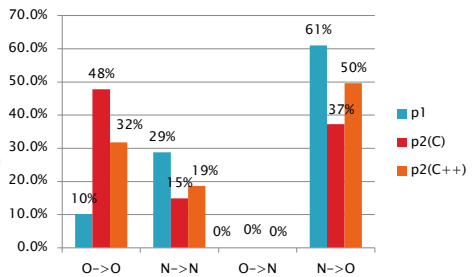
診断結果

- 結果の一部としてQ2,Q5についてのグラフを示す
- Adqua→本ツールでの結果の変化を示している



Question5
関数の処理が複雑すぎないか

Question2
1ファイルの責務の量は適切か



14

診断結果

Q2　責務の量は適切か？

- MFI004 ステートメント数
- MFI019 関数の数
- MFI477 実行コード行数

ファイル名	MFI004	MFI019	MFI477	平均点	MFI004<300	MFI019<25	MFI477<300	MFI004≦100	MFI019≦10	MFI477≦150	Adqua	診断ツール
Sample1.cpp	73.89	18.89	88.57	60.45	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	NG	OK
Sample2.cpp	100	100	100	100	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	OK	OK
Sample3.cpp	100	53.59	100	84.53	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	OK	OK
Sample4.cpp	9.96	0	33.63	14.53	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	NG	NG
Sample5.cpp	100	100	100	100	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	OK	OK
Sample6.cpp	100	91.3	100	97.1	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	OK	OK
Sample7.cpp	53.43	40.13	76.36	56.64	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	NG	OK
Sample8.cpp	100	100	100	100	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	OK	OK

「関数の数が増えるからといって、クラスやファイルを分割してしまうと、ファイルの役割が曖昧となりコードの可読性は損なわれてしまう」という意図が汲まれている

15

ヒアリングと考察

- 閾値の変更による影響が大きかったので、メトリクスの組み合わせの部分ももっと考慮してほしい
- ファイル単位ではなく関数単位レベルで行えると、より適切な粒度で評価できる
- 改善要求があるファイルがNGと診断されていた
- AdquaでNGとなっていたファイルを、解釈によって減らすことができていた
- 派生開発における差分の無いコードについては、修正できない問題がある

→時間、リスクなど

16

今後のアプローチ

①現状の視覚化
②評価結果のレビュー
- 開発者同士での共有
- 妥当性検証

- 3つの対処方法
  - A　直ちに対処へ
  - B　次世代開発時に対処へ
  - C　対処しない（Question、評価式の見直し）
- 対処判断材料
  - 致命度
  - リスク
  - 今後の派生への影響度

17

今後のアプローチ

- 評価結果のレビュー方法例

ファイル名	Question	致命度	リスク	影響度	対処方法	コメント
xxx.c	Q1	L	H	H	B	
xxx.c	Q2	H	L	H	A	
yyy.c	Q1	H	H	H	A	
yyy.c	Q2	L	H	L	C	
yyy.c	Q3	H	L	L	A	
zzz.c	Q2	H	H	L	B	

  - A　直ちに対処へ
  - B　次世代開発時に対処へ
  - C　対処しない

  - H＝High
  - L＝Low

18

今後の課題

- ツールとその適用方法は完成した
→改善の余地はまだまだ存在する
- Questionの数が少ない
  - Adqua以外のメトリクスも考慮したい
  - 開発者の意図をもっと表現していきたい
  - 評価式の決定プロセスも明文化したい
- 今後のアプローチで示した表をより具体的に

- より多くの場面でツールを使ってもらいたい

19

ご静聴ありがとうございました

20