DETECTION STRATEGIES: METRICS-BASED RULES FOR DETECTING DESIGN FLAWS

INTRODUÇÃO

- Projetos de softwares OO estão sujeitos a erros
- Ex. Falta de flexibilidade e dificuldade de manutenção
- Necessidade na identificação dos problemas do projeto

INTRODUÇÃO

- A única forma de entender o projeto é o quantificando.
- *"É possível expressar boas regras de um projeto de um modo quantificável ?"*

DEFINIÇÃO DO PROBLEMA

- Há um grande incentivo para identificação e utilização de métricas OO
- Porém a utilização destas métricas criaram problemas:
  - A definição de métricas são algumas vezes imprecisas, confusas ou incompletas
  - A interpretação dos resultados não é muito precisa, pouco empírica, o que gera desconfiança
  - Interpretação individual das métricas. Uma métrica isolada por apontar uma anomalia, mas nem sempre indica a causa da anomalia.

DEFINIÇÃO DO PROBLEMA

- Portanto, o objetivo do artigo é propor mecanismos que permitam, através de estratégias de detecção, definir resultados mais palpáveis extraídos das métricas coletadas

ESTRATÉGIAS DE DETECÇÃO

- *"Estratégia de detecção é uma expressão quantificável de uma regra, aonde podemos verificar se fragmentos do código-fonte estão em conformidade com esta regra definida."*
- As métricas permitem essa quantificação.

MECANISMOS DE FILTRAGEM

- Filtros possibilitam a redução dos dados
- Permite avaliar aspectos isolados
- Definição de intervalos de valores aceitáveis

MECANISMOS DE FILTRAGEM

- Existem dois tipos de filtros:
  - Marginal: é definido uma margem para divisão do conjunto de dados
  - Intervalo: é definido um limite inferior e outro superior para o conjunto de dados

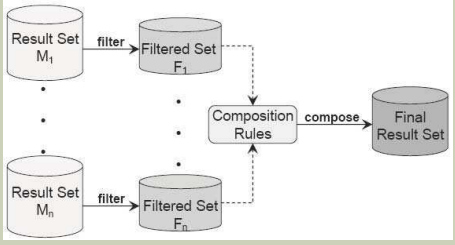
MECANISMOS DE FILTRAGEM

Type of Data Filter	Limit Specifiers		Filter Example
Marginal	Semantical	*Relative*	• `TopValues(10)`, • `BottomValues(5%)`
		Absolute	• `HigherThan(20)` • `LowerThan(6)`
	Statistical		• `Box-Plot`
Type of Data Filter	**Specification**		**Filter Example**
Interval	Composition of two marginal filters, with semantical limit specifiers of opposite polarities		`Between(20,30) := HigherThan(20) ∧ LowerThan(30)`

MECANISMOS DE COMPOSIÇÃO

- Mesclar métricas através de operações.
- Ponto de vista lógico: permite aplicar operadores lógicos às métricas
- Ponto de vista de conjuntos: permite agrupar as métricas em conjuntos

MECANISMOS DE COMPOSIÇÃO



DEFINIÇÃO DA ESTRATÉGIA DE DETECÇÃO

- Utilização do *"God Class"* como exemplo
- O ponto de partida é dado por uma ou mais regras informais.
- As métricas podem contemplar uma ou mais regras

METODOLOGIA

- Primeiramente, as regras definidas para *"God Class"* tratam de encontrar uma classe com alta complexidade, baixa coesão e um alto nível de acoplamento.

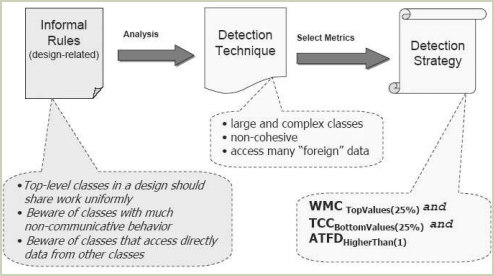
METODOLOGIA

- O segundo passo é definir métricas que contemplem as três regras anteriores:
  - *Weighted Method Count* (WMC): soma de complexidade dos métodos
  - *Tight Class Cohesion* (TCC): é o número de métodos conectados diretamente
  - *Access to Foreign Data* (ATFD): número de acessos à atributos de classes externas

METODOLOGIA

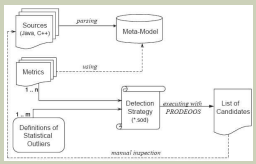
- O próximo passo é definir o mecanismo de filtragem.
- Neste caso o autor definiu TopValues(25%) para algumas métricas e *HigherThan*(1) para outras.
- Por fim, agora todos esses elementos devem ser relacionados utilizando os mecanismos de composição.

METODOLOGIA



IMPLEMENTAÇÃO DAS ESTRATÉGIAS DE DETECÇÃO

- Para automatizar a detecção destas estratégias, devemos expressar o que queremos em formato computacional



IMPLEMENTAÇÃO DAS ESTRATÉGIAS DE DETECÇÃO

$$GodClass(S) = S' \left| \begin{array}{l} S' \subseteq S, \forall C \in S' \\ (WMC(C), TopValues(25\%)) \wedge (ATFD(C), HigherThan(1)) \wedge (TCC, BottomValues(25\%)) \end{array} \right.$$

$$GodClass(S) = S' \left| \begin{array}{l} S' \subseteq S, \forall C \in S' \\ (ATFD(C), HigherThan(1)) \wedge ((WMC(C), TopValues(25\%)) \vee (TCC, BottomValues(25\%))) \end{array} \right.$$

FERRAMENTA DESENVOLVIDA

- Uma ferramenta foi desenvolvida onde as métricas são coletadas automaticamente, assim como as falhas do projeto definidas na ferramenta do autor
- Os resultados devem ser analisados manualmente

CASO DE ESTUDOS

Design Flaw	Suspects in SV1	Suspects in SV2	False Positives	Special Cases	Real Flaws	Accuracy Rate
Feature Envy	40	15	11	4	25	63%
God Method	4	4	1	3	3	75%
Shotgun Surgery	15	7	6	1	9	60%
Refused Bequest	22	6	4	2	18	81%
God Class	5	2	2	0	3	60%
Data Class	3	2	1	1	2	66%
God Package	2	1	1	0	1	50%
Misplaced Class	4	2	1	1	3	75%
Wide Subsys. Interface	5	1	1	0	4	80%

CONCLUSÃO

- Identifica informações reais relacionadas à falhas em fragmentos do código-fonte do software
- Permite quantificar um bom desing de softwares OO

TRABALHOS FUTUROS

- Fazer um estudo mais apurado para escolha dos valores limites
- Fazer a validação da ferramenta em outros softwares
- Estudar outros métodos diferentes das estratégias de detecção afim de estender a ferramenta