UNIVERSIDADE FEDERAL DE LAVRAS

PRÓ-REITORIA DE PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO (PPGCC)

Linha de Pesquisa: Banco de Dados e Engenharia de Software

Detecting Code Smells in Software Product Lines

Ramon Abílio
ramon@posgrad.ufla.br
Mestrando

Eduardo Figueiredo
figueiredo@dcc.ufmg.br
Coorientador

Heitor Costa
heitor@dcc.ufla.br
Orientador

Agenda

1. Introduction
2. Background
3. Performed Work
4. Final Consideration

Agenda

1. **Introduction**
2. Background
3. Performed Work
4. Final Consideration

Introduction

- Software Quality
- Measure projects, process, and software
- Code Smell
- Measure-based Detection Strategy
- Software Product Lines (SPL)
- Feature-Oriented Programming (FOP)

Motivation

- Composition approach
  - *Different techniques: AHEAD, AspectJ, and CaesarJ*
  - *Based in: base code and delta*
- FOP source code may present symptoms of poor quality
- Code smells may exist in software written in every programming technique

Goals

The goal is to investigate the presence of ***code smells*** *in* ***FOP-based SPL*** *developed with* ***compositional approaches*** *and how we can* ***detect them*** *using* ***measure-based detection strategies***

Agenda

1. Introduction
2. **Background**
3. Performed Work
4. Final Consideration

Software Measures

- Projects, processes, and software properties
- Used to evaluate software properties in development or final phase
- Measures are often used to quantify:
  - *Coupling and cohesion*
  - *Modularity*
  - *Instability*
  - *Error-proneness*
  - *Identifying crosscutting concerns*

Traditional and Object-Oriented Measures

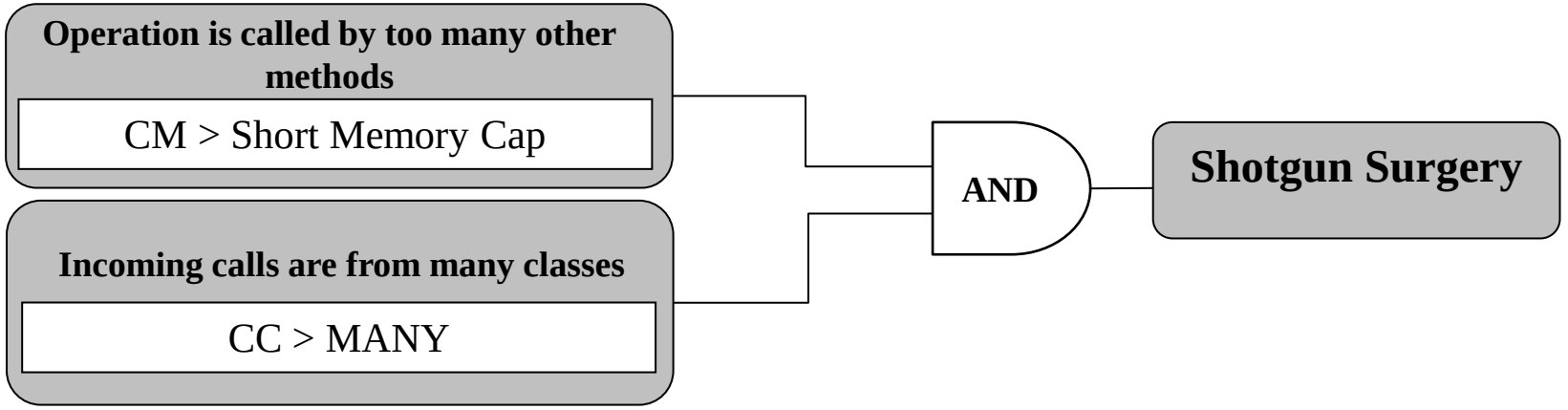
- Weighted Methods per Class (WMC)
- Coupling between Object Classes (CBO)
- Cyclomatic Complexity (Cyclo)
- Lines of Code (LoC)
- Method's Lines of Code (MLoC)
- Number of Attributes (NOA)
- Number of Methods (NOM)
- Number of Parameters (NP)
- Number of Operation Overrides (NOOr)
- Number of Overridden Operations (NOrO)

Code Smell

- Code smells aim to diagnose symptoms that may be indicative of something wrong in the design or an undesired source code property
- Code smells may be present in classes and methods
- In SPL, there are many smells which indicate potentially inadequate feature modeling or implementations (*variability smells)*
  - *Unused variability*
  - *Unused feature*
  - *Dead feature code*
  - *Duplicate code in alternative features*

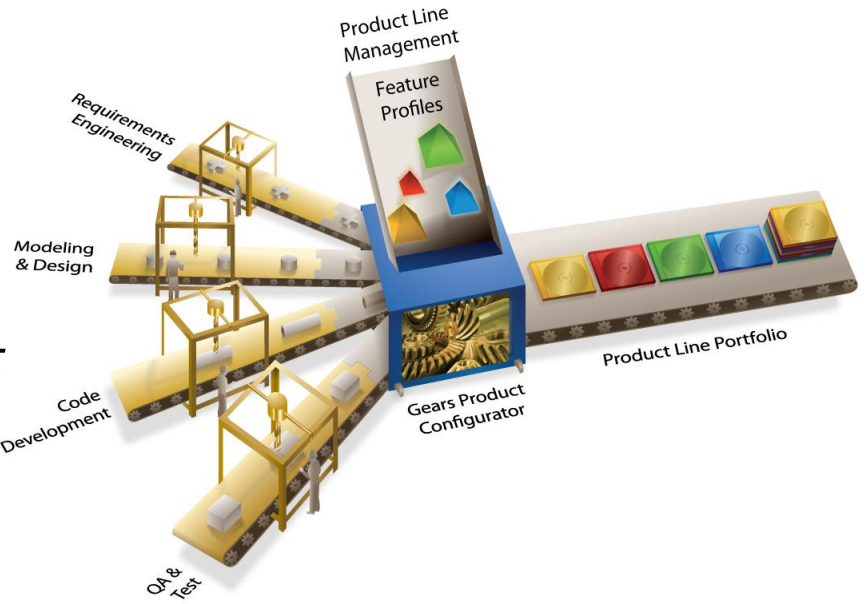
Software Measures and Code Smells Detection Strategies

- Measures are often fine grained
- Detection strategy



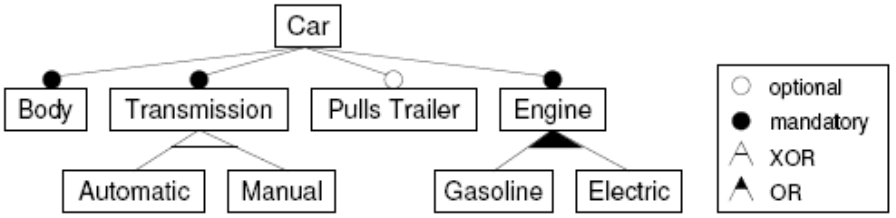
Software Product Lines (SPL)

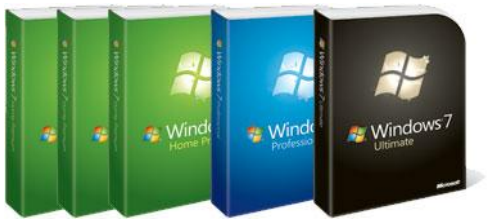
- The concept of product lines was fully introduced in the early 1990s
- Feature-Oriented Domain Analysis (FODA) method
- An SPL can be defined as:
  - *A set of systems that share common artifacts and are managed to meet specifics needs of clients and market*



Software Product Lines (SPL) (cont.)

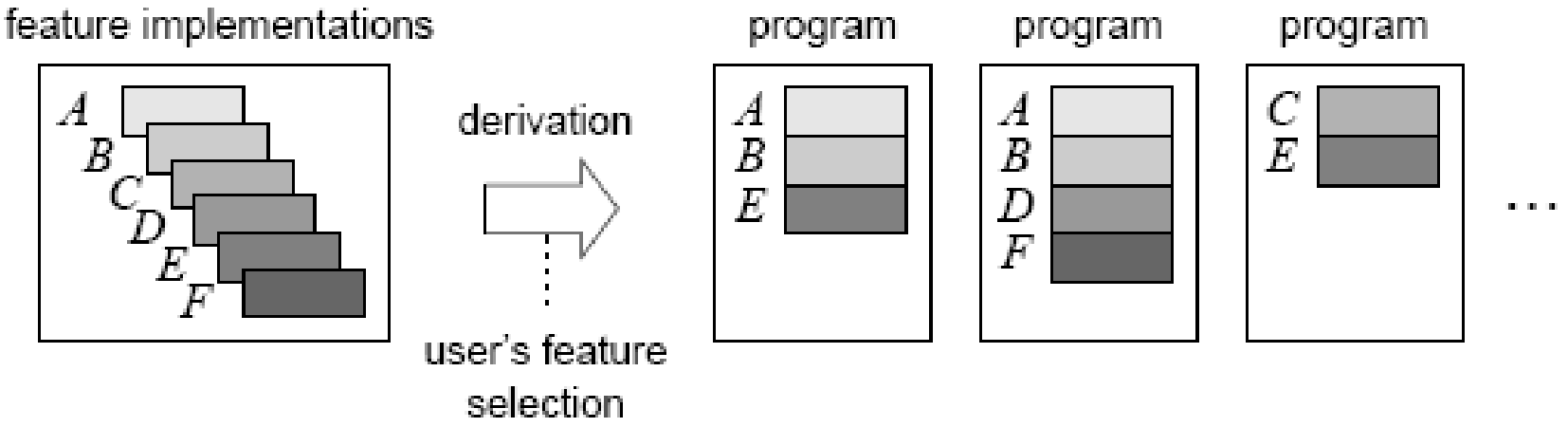
- Promotes large-scale and systematic reuse of components
- SPL Architecture:
  - *Common features → kernel*
  - *Other features → points of variation*





Feature-Oriented Programming

- Compositional approaches implement features as distinct (physically separated) code units
- To generate a product line member for a feature selection, the corresponding code units are determined and composed, usually at compile-time or deploy-time



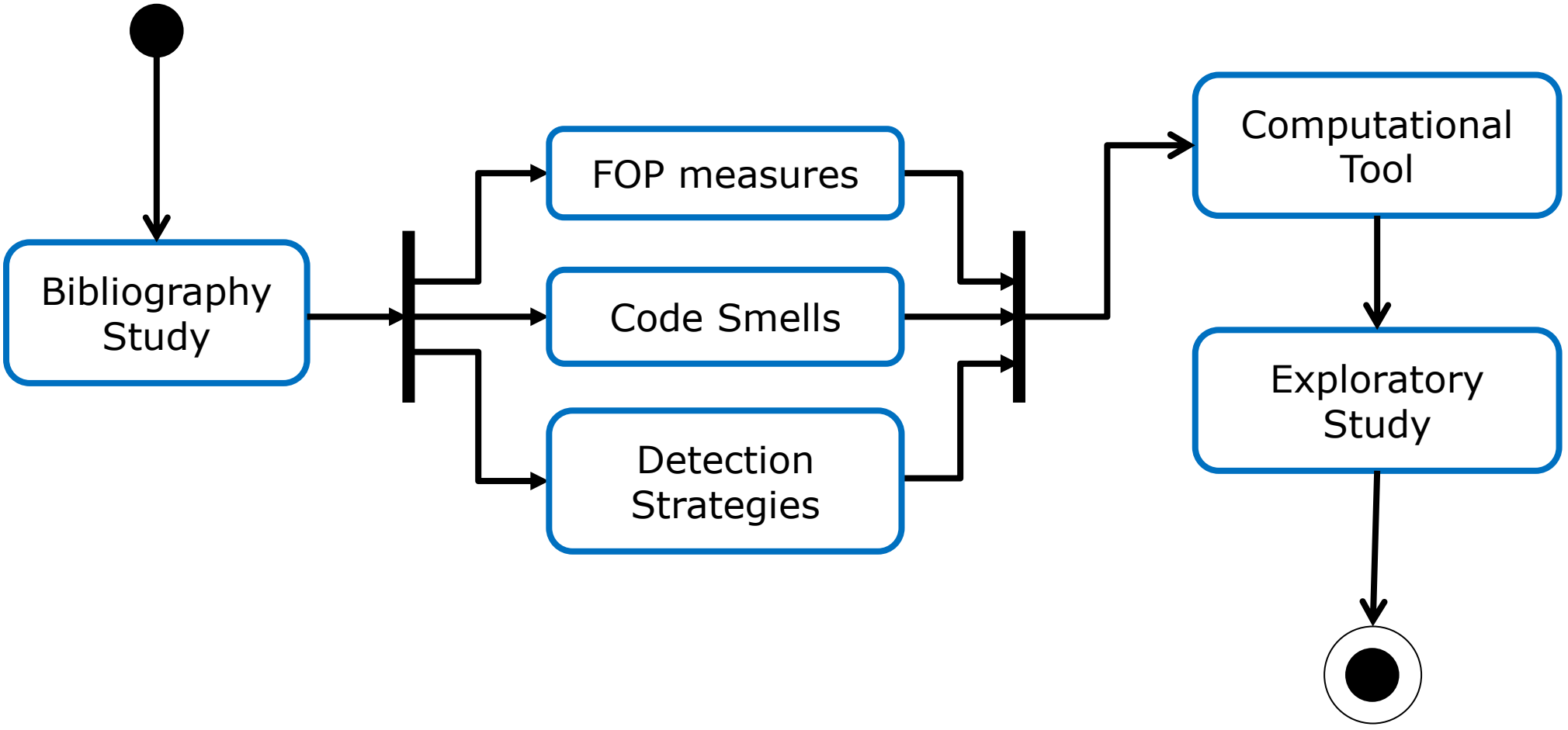
AHEAD

- Programs are defined as constants and features are added using refinement functions
  - *Classes implement basic functions of a system (constants)*
  - *Extensions, variations, and adaptations in these functions constitute the features (refinements)*
- Features
  - *Implemented in modules syntactically independent of the classes*
  - *Can insert or modify methods and attributes*

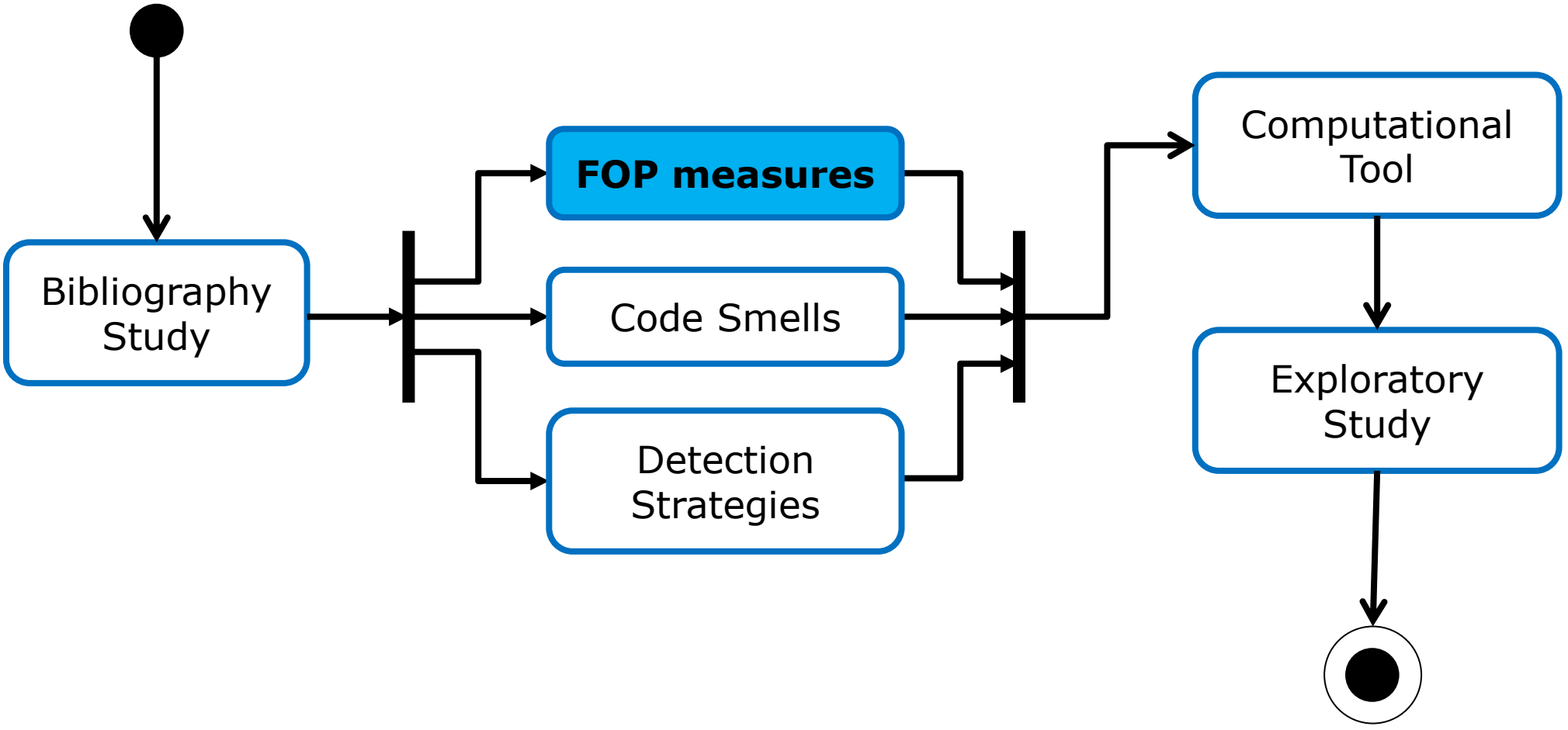
Agenda

1. Introduction
2. Background
3. **Performed Work**
4. Final Consideration

Methodological Procedures



Methodological Procedures

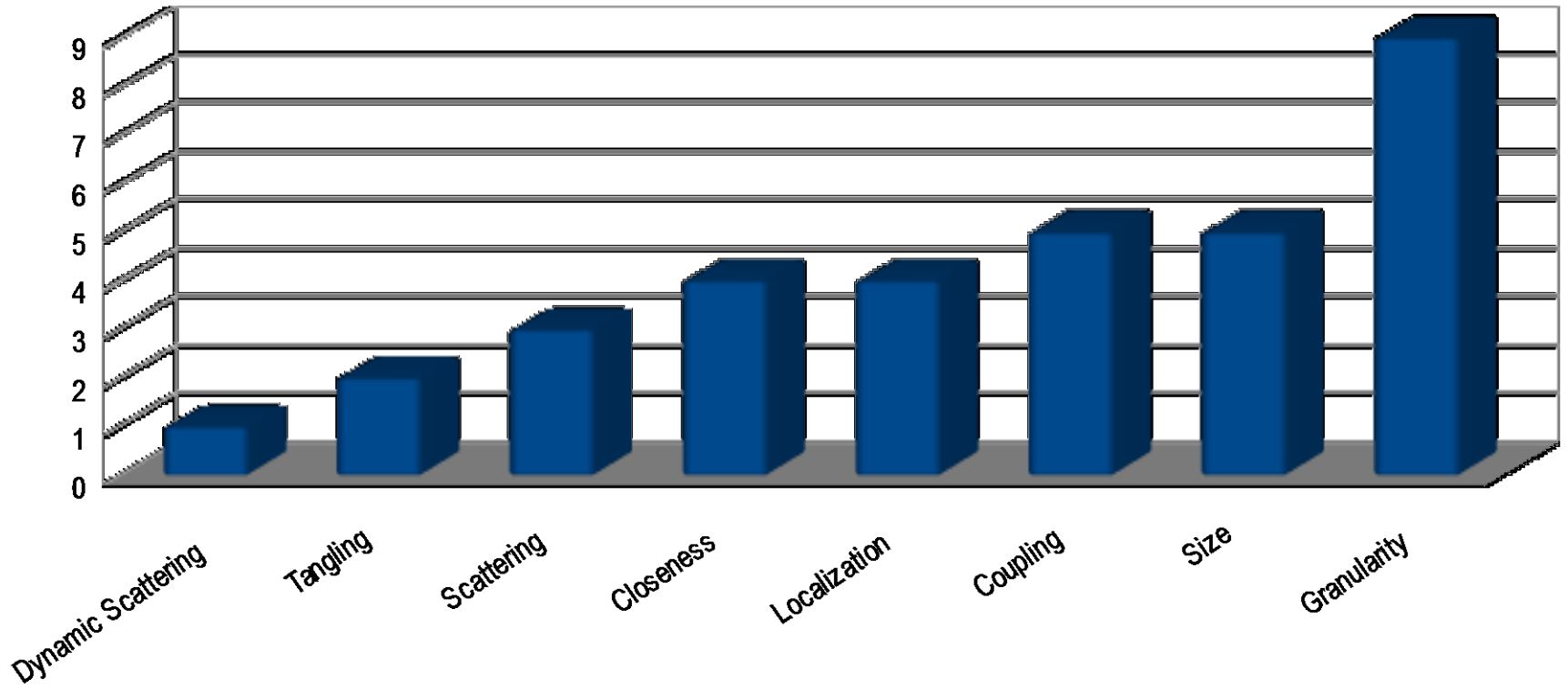


Identification and Analysis of FOP measures

- Systematic Review of Literature
  - *33 measures for Feature-Oriented Programming (FOP)*
  - *78 measures for Aspect-Oriented Programming (AOP)*
- We detailed 33 measures for FOP

30	Name: Number of Features (NOF)	[Figueiredo et al., 2008b]/ [Lopez-Herrejon; Apel, 2007]
Description: Counts the number of features in a program.		
Domain*: [0, N] where N is the number of all features.		
Property*: Size		Approach*: Independent
Value interpretation*: How close the value is to N, more features a program has, and its maintenance may be more complex.		

Distribution of the Measures among the Properties



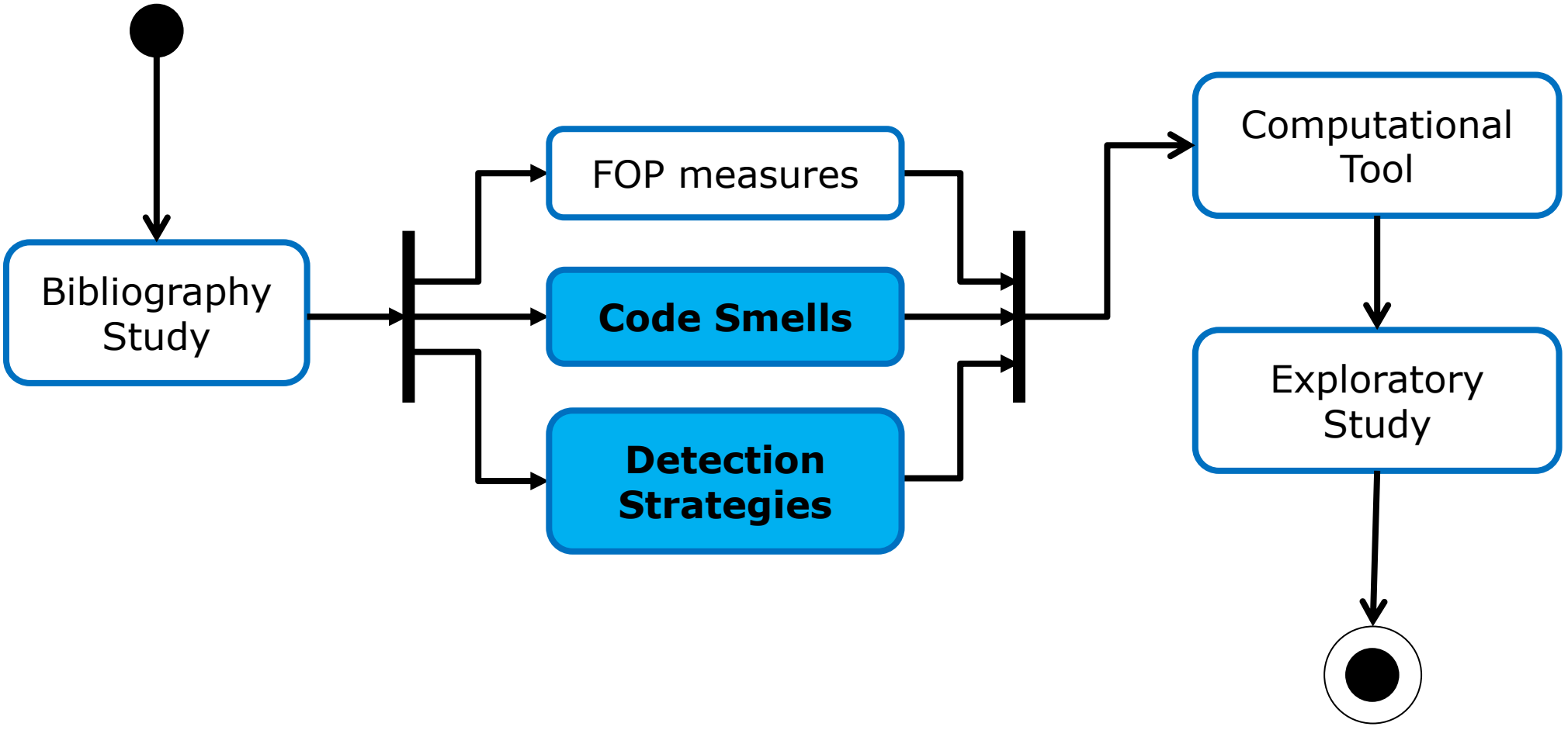
FOP measures per Approach

Annotative Approach	
Attribute	MethodBody
BeforeReturn	NestedStatement
Class	Number of Classes
ClassSignature	Number of Packages (NOP)
EndMethod	Packages
Expression	Scattering Degree (SD)
InterfaceMethod	StartMethod
Lines of Feature Code (LOF)	Statement
Method	Tangling Degree (TD)
Independent Approach	
Concentration	Hybrid Feature Coupling (HFC)
Dedication	Lines of Code (LOC)
Disparity	Maximum Textual Feature Coupling (TFCmax)
Distance Between Features (DIST)	Number of Features (NOF)
Dynamic Concern Diffusion over Components (dCDC)	Structural Feature Coupling (SFC)
Feature Scattering	Structural Feature Coupling Prime (SFC')
Feature Tangling	Textual Feature Coupling (TFC)
Aspect Approach	
Feature Crosscutting Degree (FCD)	

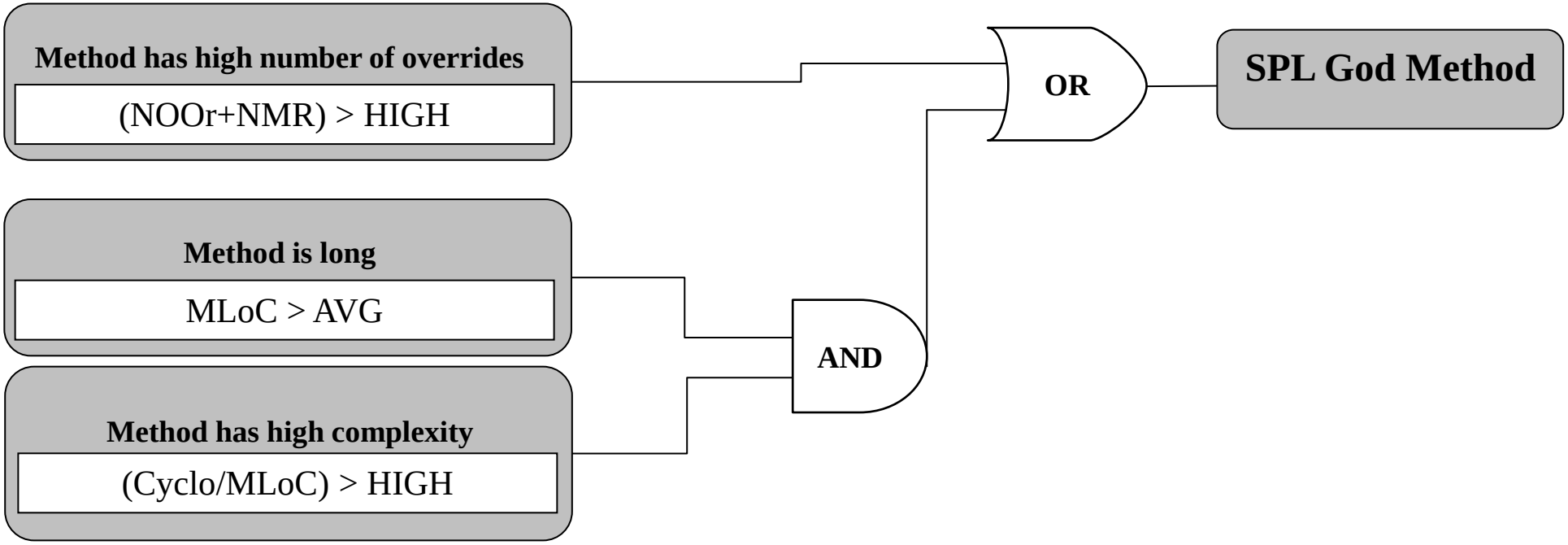
FOP measures for Compositional Approach

- Measures to address: constants and refinements
  - *Number of Method Refinements (NMR)*
  - *Number of Refined Methods (NRM)*
  - *Number of Constants (NOCt)*
  - *Number of Constant Refinements (NCR)*
  - *Number of Refinements (NOR)*
  - *Number of Components (NOC)*
  - *Number of Refined Constants (NRC)*
  - *Number of Features (NOF)*

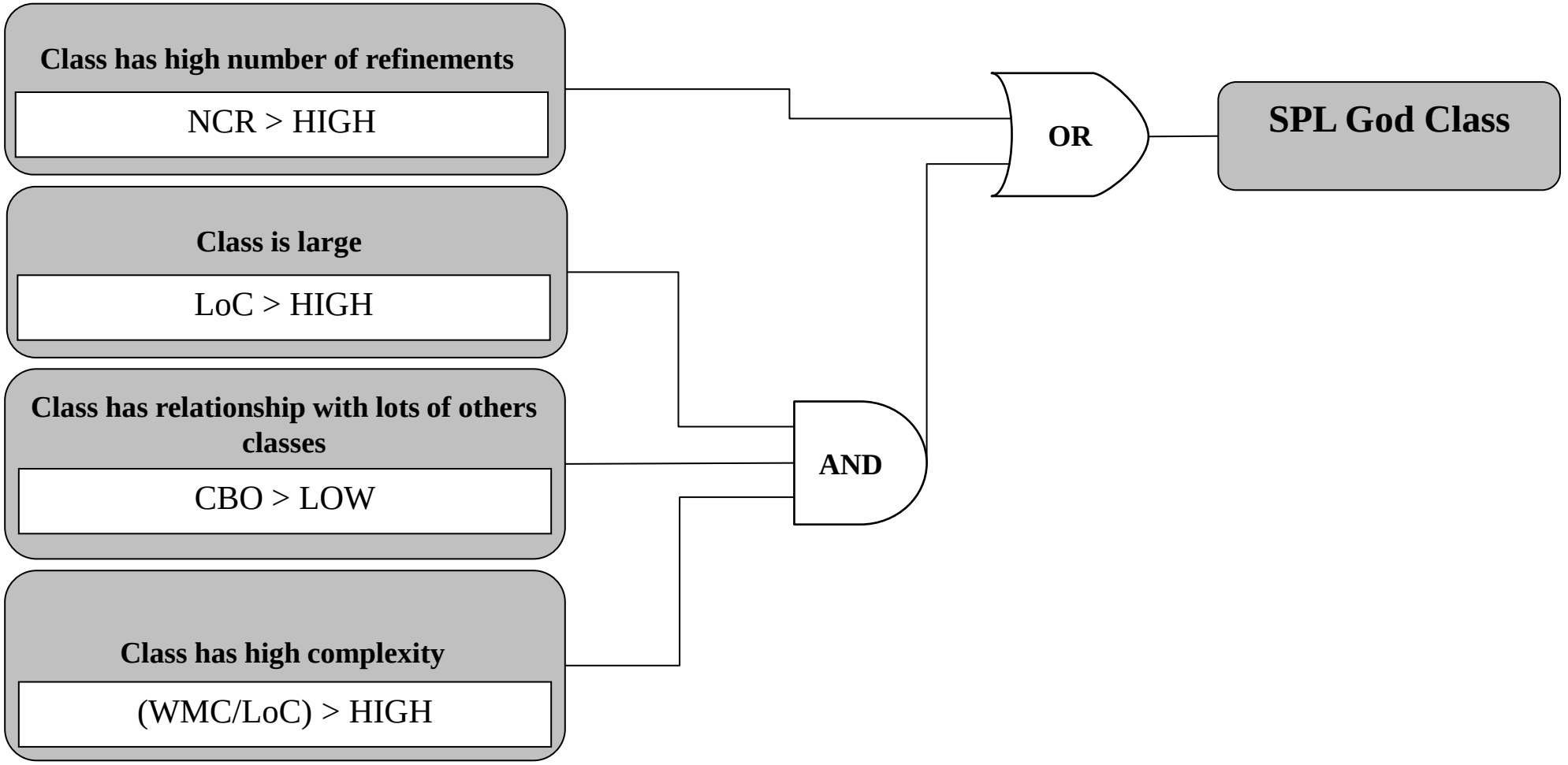
Methodological Procedures



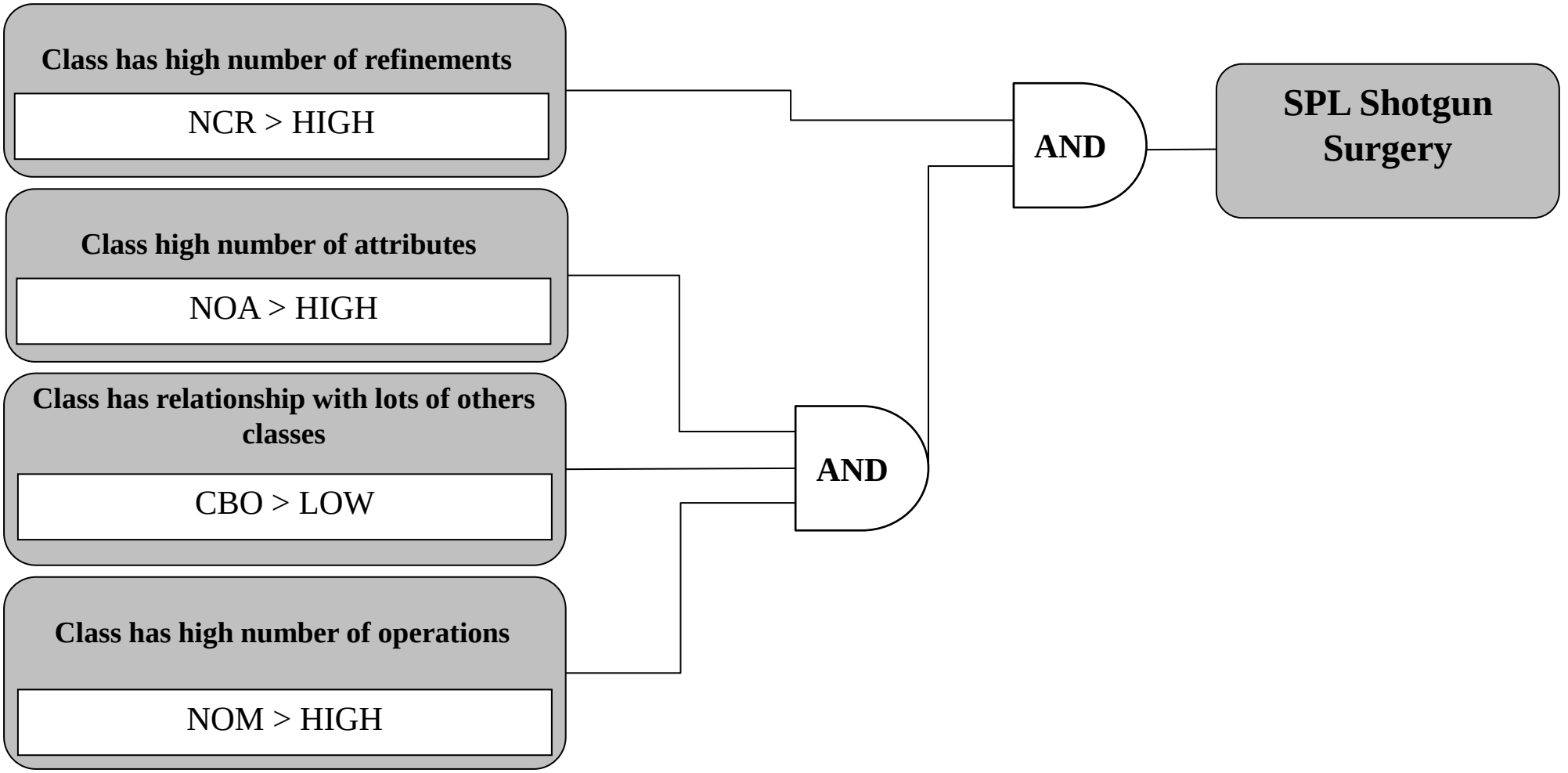
Measures-based Detection Strategy
SPL God Method



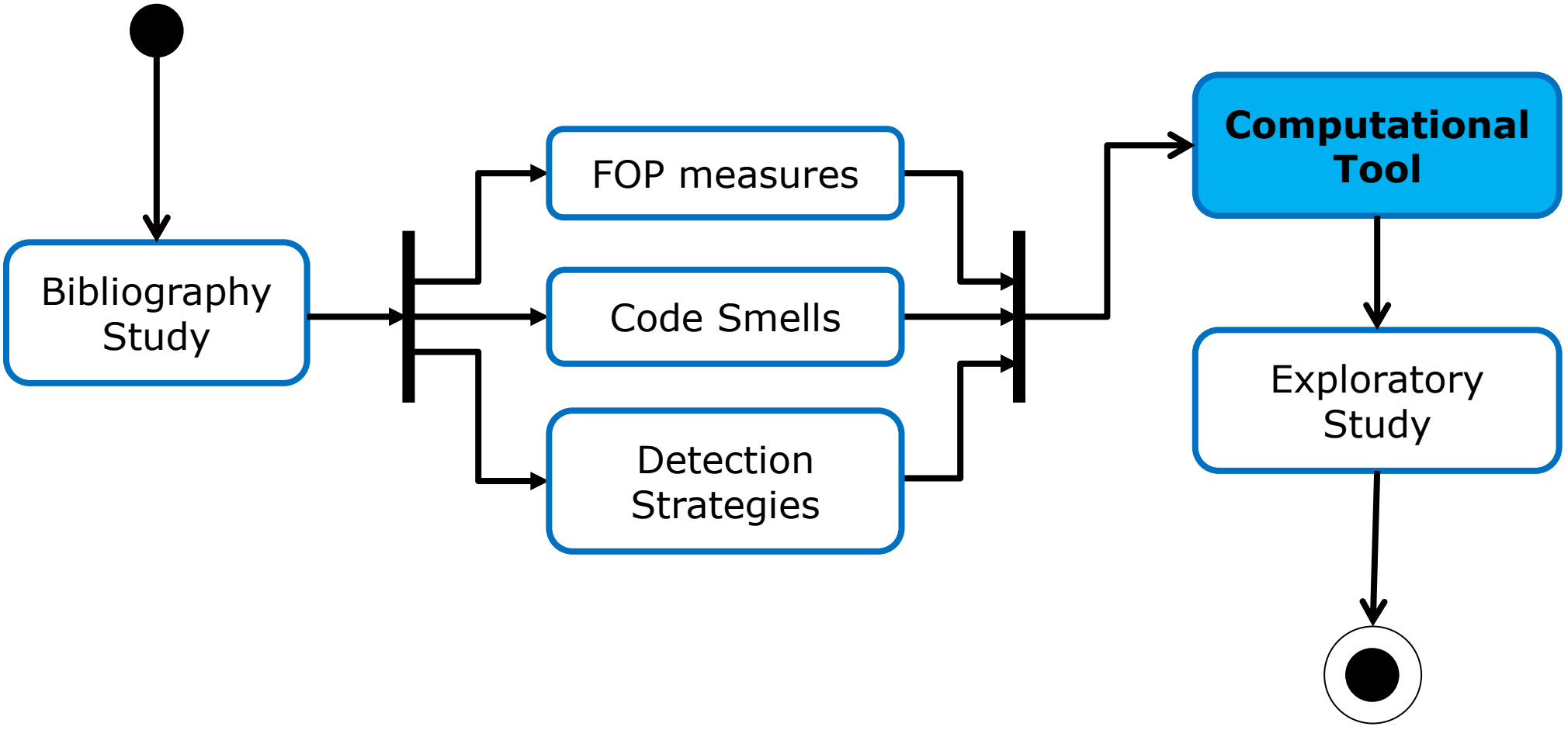
Measures-based Detection Strategy SPL God Class



Measures-based Detection Strategy
SPL Shotgun Surgery



Methodological Procedures



Computational Tool
Variability Smell Detection

- Variability Smell Detection (VSD)
  - *Eclipse IDE 4.3 (Kepler)*
  - *Java Development Tool (JDT)*
  - *Plug-in Development Environment (PDE)*
  - *FeatureIDE*
- VSD implements 19 primitive and 8 derived measures
  - *Primitive: traditional, OO, and FO*
  - *Derived: summarize some primitive measures*

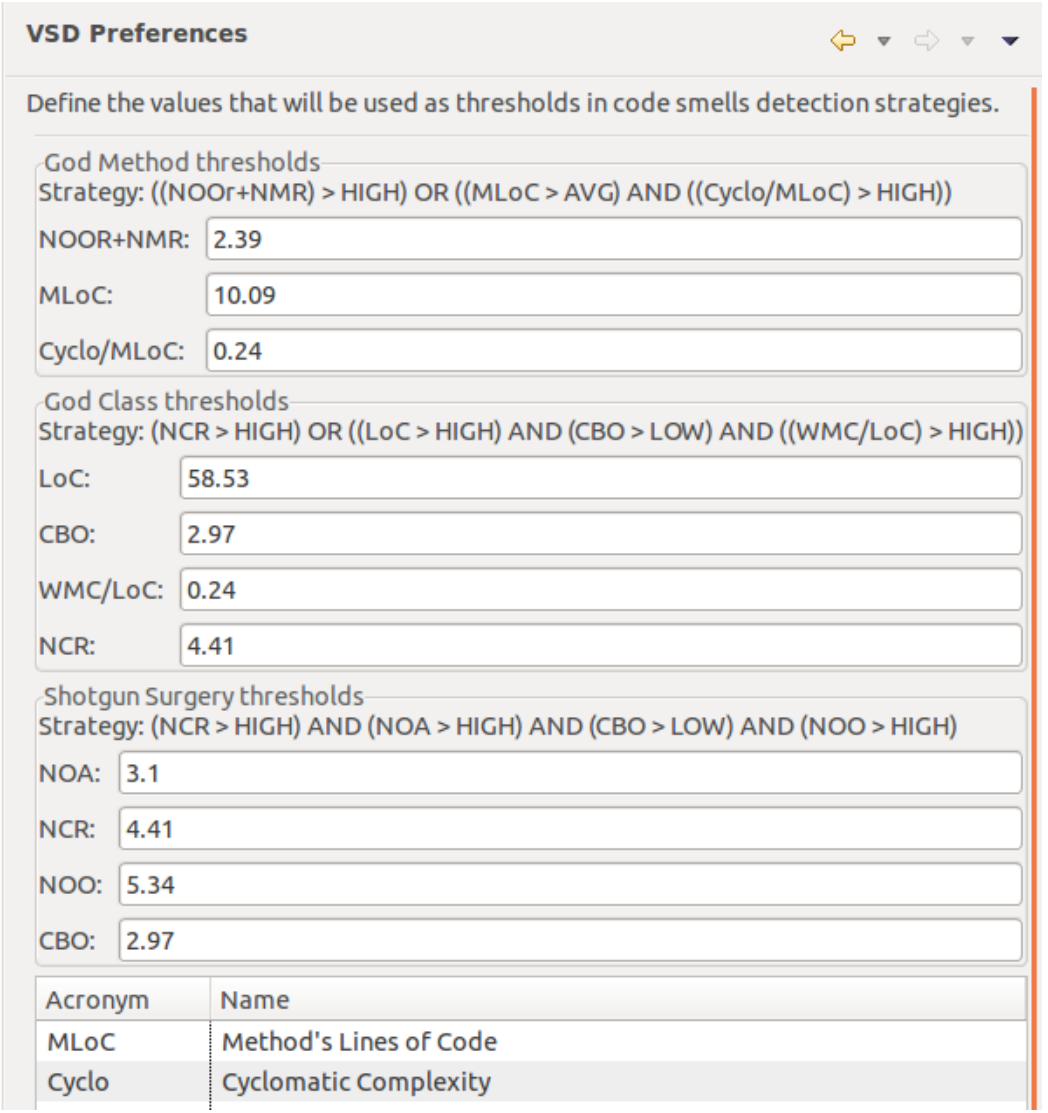
VSD Measures

Group	Measure	Primitive	Derived
Method	Method's Lines of Code (MLoC)	X	
	McCabe's Cyclomatic Complexity (Cyclo)	X	
	Number of Parameters (NP)	X	
	Number of Method Refinements (NMR)	X	
	Number of Operation Overrides (NOOr)	X	
Component	Lines of Code (LoC)	X	
	Number of Methods (NOM)	X	
	Number of Attributes (NOA)	X	
	Coupling between Objects (CBO)	X	
	Weighted Methods Per Class (WMC)		X
	Number of Constant Refinements (NCR)	X	
SPL	Number of Components (NOC)	X	
	Number of Constants (NOCt)	X	
	Number of Refinements (NOR)	X	
	Number of Refined Constants (NRC)	X	
	Total Number of Method Refinements (TNMR)	X	
	Number of Refined Methods (NRM)	X	
	Number of Overridden Operations (NOrO)	X	
	Number of Features (NOF)	X	
	Total Lines of Code (TLoC)		X
	Total Number of Methods (TNOM)		X
	Total Number of Attributes (TNOA)		X
	Total Cyclomatic Complexity (TCyclo)		X
	Total Coupling between Objects (TCBO)		X
	Total Number of Operation Overrides (TNOOr)		X

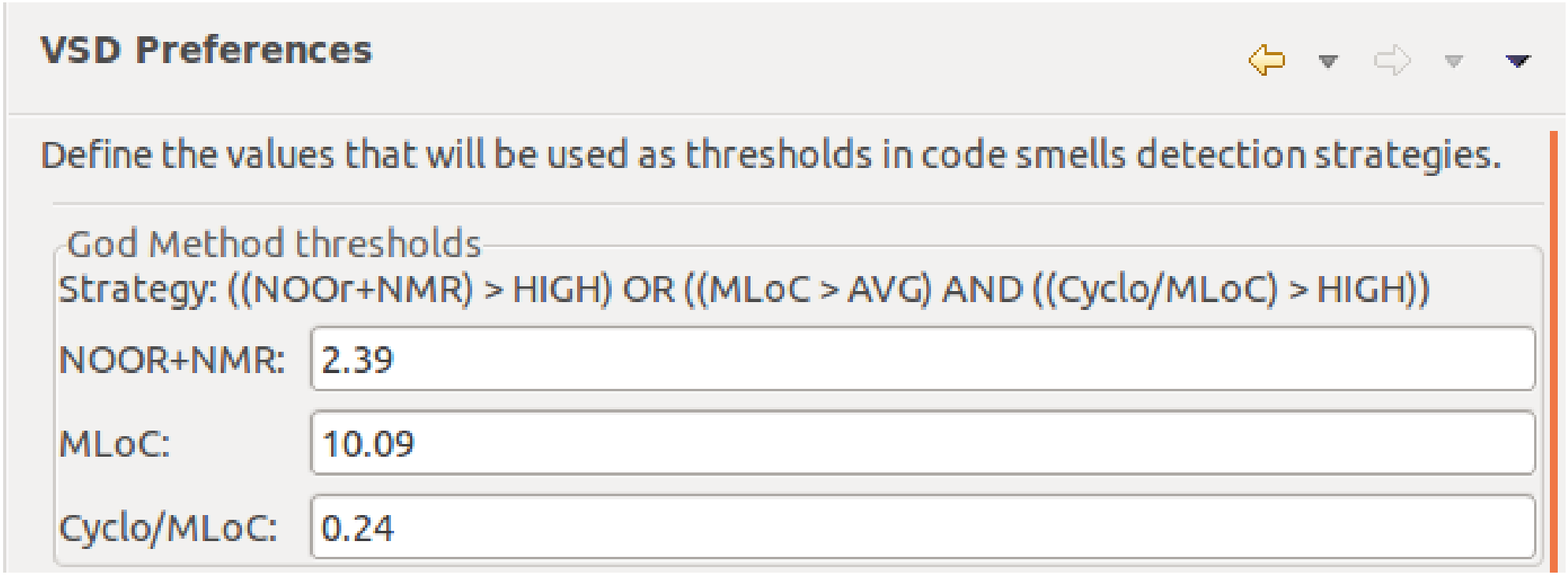
VSD Features

- VSD is accessed by a
  - *Right-click in a FeatureIDE AHEAD project*
- The available options are:
  - *Measure*
  - *Detect God Method*
  - *Detect God Class*
  - *Detect Shotgun Surgery*
- Each option has a specific view

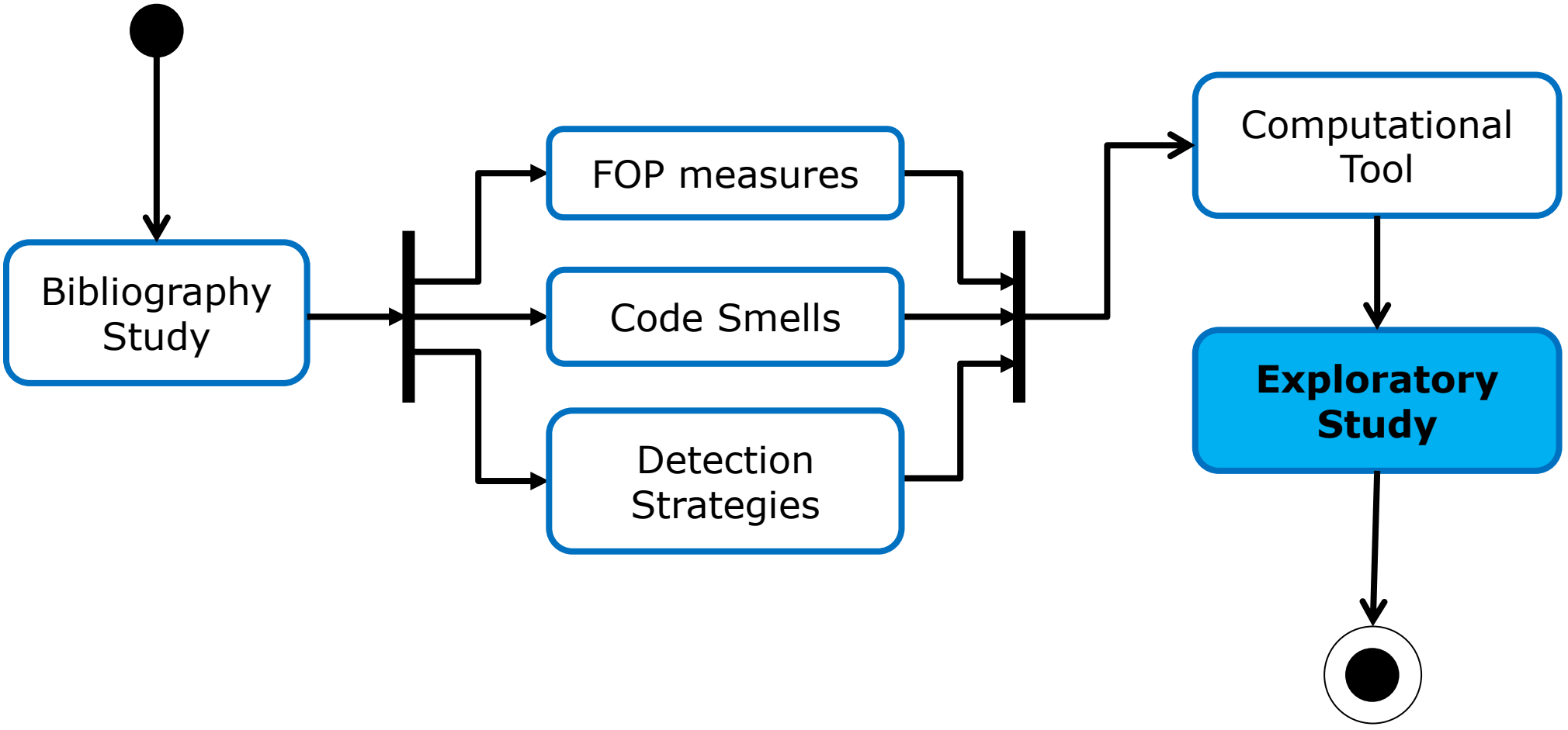
VSD Preference Page



VSD Preference Page (cont.)



Methodological Procedures



Exploratory Study

- Study Settings
  - *SPLs*
  - *Thresholds*
  - *Participants*
  - *Tasks*
  - *Analysis*
- Results and Discussion
- Threats to Validity

Analyzed SPLs

SPL	Domain	NOF	NOC	TLoC
AHEAD-Bali	Grammar tool	15	61	3,988
AHEAD-Guidsl	Graphical configuration tool	25	233	8,738
AHEAD-Java	Programming tool	106	963	16,719
DesktopSearcher	Local searcher tool	16	46	1,858
Devolution	E-mail and Instant Message client	11	68	3,913
EPL	Expression evaluation	12	21	98
GPL	Graph and algorithm library	36	75	1,824
TankWar	Game	30	88	4,669

Thresholds

- We measured the SPLs in three different granularities
  - *SPL, component, and method*
- The thresholds were based on statistical values
  - *Average ($\bar{X}$) and Standard Deviation (S)*
- We calculated the threshold values as follows:

Low	Average	High
$\bar{X}$ - S	$\bar{X}$	$\bar{X}$ + S

Calculated Thresholds

Derived Measures	Low	Avg	High
TLoC / NOC	16.53	37.53	58.53
TLoC / TNOM	6.00	10.09	14.19
TCyclo / TLoC	0.18	0.21	0.24
TNOA / NOC	0.58	1.84	3.10
TNOM / NOC	1.86	3.60	5.34
NOR / NOCt	0.50	2.46	4.41
(TNOOr + TNMR) / NOrO + NRM	1.45	1.92	2.39
TCBO / NOC	2.97	5.95	8.92

Participants

- This study involved 26 participants from two Universities
- Background Questionnaire
  - *Courses*
  - *Work Experience*
  - *Knowledge in:*
    - Java Programming (JP)
    - Software Measurement (SM)
    - Feature-Oriented Programming (FOP)
    - Software Product Lines (SPL)

Participants (cont.)

- Courses:
  - *96% of them took some course on OOP*
  - *92% took courses on Java Programming*
  - *77% took courses on Software Modeling*
- Work Experience:
  - *84% have some experience working in companies*
- Participants have:
  - *Basic knowledge in: SM, FOP, and SPL*
  - *Intermediate to advanced knowledge in: JP*

Tasks

- We divided the study in two 90-minute sessions:
  - *First session: training*
  - *Second session: inspection*

Feature	Component	Refinement	LoC	NOM	NOA	CBO	WMC	NCR	Has smell
Beschleunigung	Maler.jak	Y	5	1	0	2	1	0	
Beschleunigung	Tank.jak	Y	~~27~~	2	2	~~3~~	~~9~~	0	X
Beschleunigung	TankManager.jak	Y	5	1	0	1	1	0	
Beschleunigung	Tool.jak	Y	~~11~~	1	0	2	2	0	X
Bombe	Maler.jak	Y	5	1	0	2	1	0	
Bombe	Tank.jak	Y	~~22~~	1	0	2	~~8~~	0	X
Bombe	TankManager.jak	Y	5	1	0	1	1	0	

Analysis

- At first, we tabulated the
  - *Thresholds selected by the participants*
  - *Results of the participants' inspections*

- Hereafter, we
  - *Tabulated all participants' indications*
  - *Analyzed them seeking to track inspection failures*
  - *Performed statistical tests to verify the agreement*

Selected Thresholds

- SPL God Method
  - *AVERAGE: MLoC - 73%, Cyclo/MLoC - 64%, NMR + NOOr – 82%*
  - *Cyclo / MLoC: 64% - AVERAGE, 36% - HIGH*
- SPL God Class
  - *AVERAGE: selected by most participants for LoC, NOM, CBO, and NCR*
  - *NOA and WMC / LoC: lacks consensus among the participants*
- SPL Shotgun Surgery
  - *AVERAGE: selected by most participants for NOM, NOA, and NCR*
  - *CBO: participants are almost equally divided between LOW and AVERAGE*

SPL God Method

- From the 60 methods:
  - *62% were indicated*
- The disagreements occurred due to:
  - *Selected thresholds*
  - *Different operators used in the comparison*
  - *Failures in detection*
  - *Changes in thresholds*
- Detection strategy indicated
  - *52% with code smell*
- Strategy and participants agreed in 97%

SPL God Class

- From the 60 components:
  - *67% were indicated*
- The disagreements occurred due to:
  - *Selected thresholds*
  - *Failures in detection*
- Detection strategy indicated
  - *25% with code smell*
- Strategy and participants agreed in 97%

SPL Shotgun Surgery

- From the 60 components:
  - *50% were indicated*
- The disagreements occurred due to:
  - *Selected thresholds*
  - *Failures in detection*
- Detection strategy indicated
  - *7% with code smell*
- Strategy and participants agreed in 90%

Statistical Tests – Fleiss' Kappa

- Components and methods were divided in two categories:
  - *Indicated: components/methods with code smell marked with "1"*
  - *Not indicated: components/methods without code smell marked with "0"*
- More than two raters (participants)
  - Fleiss' Kappa
- P-value:

H0: There is no agreement among participants; i.e., Kappa = 0

H1: There is agreement among participants; i.e., Kappa > 0

Statistical Tests – Fleiss' Kappa (cont.)

- Kappa's Reference Values

Values of Kappa	Interpretation
< 0	No agreement
0 - 0.19	Poor agreement
0.20 - 0.39	Fair agreement
0.40 - 0.59	Moderate agreement
0.60 - 0.79	Substantial agreement
0.80 - 1.00	Almost perfect agreement

- Fleiss' Kappa Measures

	SPL God Method	SPL God Class	SPL Shotgun Surgery
Fleiss' Kappa	0.793	0.427	0.427
Confidence Interval - 95%	upper: 0.827 lower: 0.758	upper: 0.506 lower: 0.396	upper: 0.474 lower: 0.379
p-value	< 0.001	< 0.001	< 0.001
Interpretation	High Agreement	Moderate Agreement	Moderate Agreement

Statistical Tests – Binomial and Sign tests

- Agreement: detection strategy and majority of participants
- Detection strategy and participants agreed in:
  - *58 of the 60 inspections in SPL God Method and SPL God Class*
  - *54 of the 60 inspections in SPL Shotgun Surgery*
- Statistical test with 0.5 of "success" probability in each inspection
- The strategies can be used instead of manual inspection

Threats to Validity

- **Conclusion**
  - *Measures implementation*
  - *Comparison of agreement using descriptive analyses*

- **Internal**
  - *Documents descriptions*
  - *Number of participants in each group and how they were divided*
  - *Number of SPLs used to calculate the thresholds*

Threats to Validity (cont.)

- **Construct**
  - *A poorly planned study may conduct to poor or wrong results*
  - *The participants' behavior*

- **External**
  - *We used only one compositional approach (AHEAD)*
  - *The participants may not represent the population*

Agenda

1. Introduction
2. Background
3. Performed Work
4. **Final Consideration**

Related Work

- In regards to OO smells:
  - *Code smells and refactoring methods*
  - *Mechanism to detect and localize code smells in source code*
  - *Other code smells and presented their detection strategies*

- Concerning AO smells:
  - *Refactoring methods for AO code smells*
  - *Five code smells in software developed with AspectJ and algorithms to automatically detect them*
  - *Measures to identify the code smells*
  - *Detection strategies based on Marinescu's mechanisms*

Related Work (cont.)

- Recent researches have focused on
  - *Techniques*
  - *Feature cohesion*
  - *Code clones*
  - *Refactoring methods for code clones*
  - *Catalogued 14 smells and refactoring methods*

Related Work (cont.)

- This work:
  - *Three code smells to take FOP abstractions into account*
  - *Eight new measures and three detection strategies*
  - *Computational tool*
  - *AHEAD (most popular FOP language)*
  - *Exploratory study*

Conclusions

- AVERAGE value as a "secure" option
- Homogeneity of the participants
- Fleiss' Kappa
- Participants failed in their analysis
- The detection strategies can be used as a code smell predictor in FOP-based SPL

Contribution

- This work provides a code-perspective of variability smells
  - *New perspectives of God Method, God Class, and Shotgun Surgery*
  - *Three measures-based detection strategies*
  - *Eight measures to compositional approaches*
  - *Tool for measure and detect code smells*
- In addition, we
  - *Identified 33 FO and 78 AO measures*
  - *Detailed those 33 FO measures*
  - *Reported three repositories of SPLs*

Future Work

- Develop a tool to measure different source code based on compositional approaches
- Investigate new code smells and detection strategies
- Perform a study on code smells in evolving FOP-based SPL
- Study what specialists consider as a large and complex method/class
- Study refactoring methods for the proposed code smells
- Study correlation between participants' courses, experience, and knowledge and the selected thresholds

Publication Results

- We published preliminary results in a symposium and workshop:

Abilio, R.; Teles, P. P. R.; Costa, H. A. X.; Figueiredo, E. **A Systematic Review of Contemporary Metrics for Software Maintainability**. In: 6th Brazilian Symposium on Software Components Architectures and Reuse, pp.130-139, 2012.

Abilio, R; Figueiredo, E.; Costa, H. **Critérios e Diretrizes para o Desenvolvimento de Linhas de Produtos de Software Manuteníveis**. In: III Workshop de Teses e Dissertações em Engenharia de Software (WTDSoft), pp.40-44, 2013.

Detecting Code Smells in Software Product Lines

Ramon Abílio
ramon@posgrad.ufla.br
Mestrando

Eduardo Figueiredo
figueiredo@dcc.ufmg.br
Coorientador

Heitor Costa
heitor@dcc.ufla.br
Orientador

UNIVERSIDADE FEDERAL DE LAVRAS

UFMG

Grupo PqES

lab-soft software engineering laboratory

25/02/2014