

13th TOOLS Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2010)

The identification of anomalous code measures with conditioned interval metrics



Carlos López, Esperanza Manso, Yania Crespo

GIRO Research Group (Grupo de Investigación en Reutilización y Orientación a Objeto)

Universities of Valladolid and Burgos, Spain

clopezno@ubu.es

{manso,yania}@infor.uva.es



Outline



2/25

- Introduction
- Objectives
- Evaluating code tools
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- Future work

Introduction



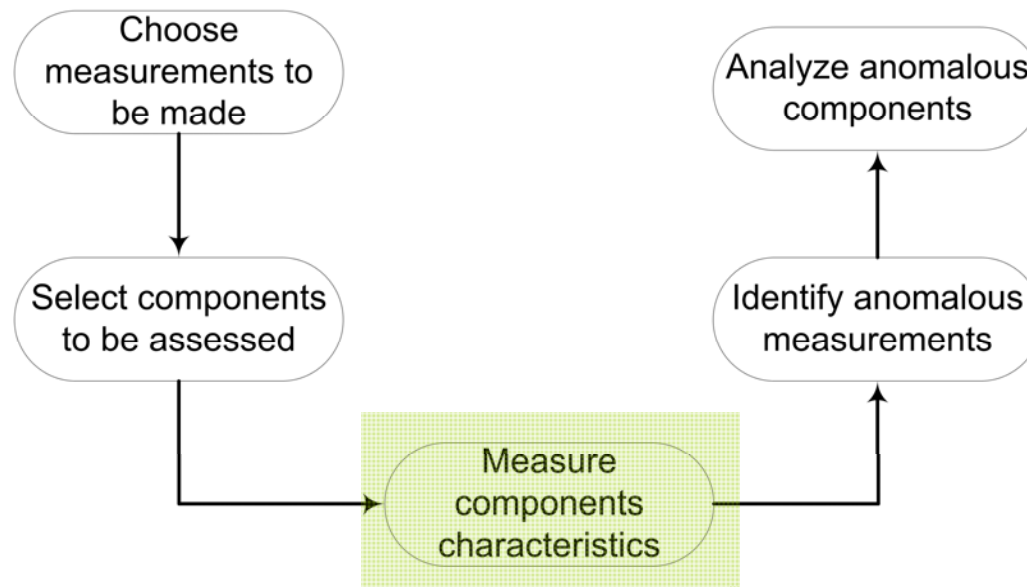
♦ Introduction

- ♦ Objectives
- ♦ Evaluating code tools
- ♦ Adaptation of the RefactorIt tool
- ♦ Use intervals for code entity
- ♦ Conclusions
- ♦ Future work

3/25

■ Measuring

- Consists of obtaining a numerical value for an attribute of a software product or process
- Part of a process
 - Measurement process defined by Sommerville



Introduction



◆ Introduction

- ◆ Objectives
- ◆ Evaluating code tools
- ◆ Adaptation of the RefactorIt tool
- ◆ Use intervals for code entity
- ◆ Conclusions
- ◆ Future work

4/25

- Identification anomalous measurements
 - Measurement is related with an **entity**
 - Measure uses a set of **software metrics**
 - Software metric definitions contain an interpretation of measured value
 - Provides a **interval** of preferred values
 - How do we identify anomalous measurements of entities?
 - Checking whether a particular measurement is within the range of preferred values

Introduction



♦ Introduction

- ♦ Objectives
- ♦ Evaluating code tools
- ♦ Adaptation of the RefactorIt tool
- ♦ Use intervals for code entity
- ♦ Conclusions
- ♦ Future work

5/25

- Our measurement context
 - Software product: Source code
 - Entities: packages, classes, methods
 - Software metrics: Code metrics
 - On packages: Robert Martin, Brito and Abreu.
 - On classes: Chidamber and Kemerer, Lorenz and Kid
 - On methods: McCabe
 - Interpretation of measured value
 - How do we interpret measures values?
 - What intervals of values we have to use?

Introduction



◆ Introduction

- ◆ Objectives
- ◆ Evaluating code tools
- ◆ Adaptation of the RefactorIt tool
- ◆ Use intervals for code entity
- ◆ Conclusions
- ◆ Future work

6/25

■ Problems of interpretation

- The intervals used to identify anomalous measurements, obtained through empirical experiments, are restricted to the measuring context/software application, thus limiting their use in other contexts/software applications
- Even recommended intervals, taken from past measurements in the same context, cannot be used for code entities from different categories (Exception, Test...)

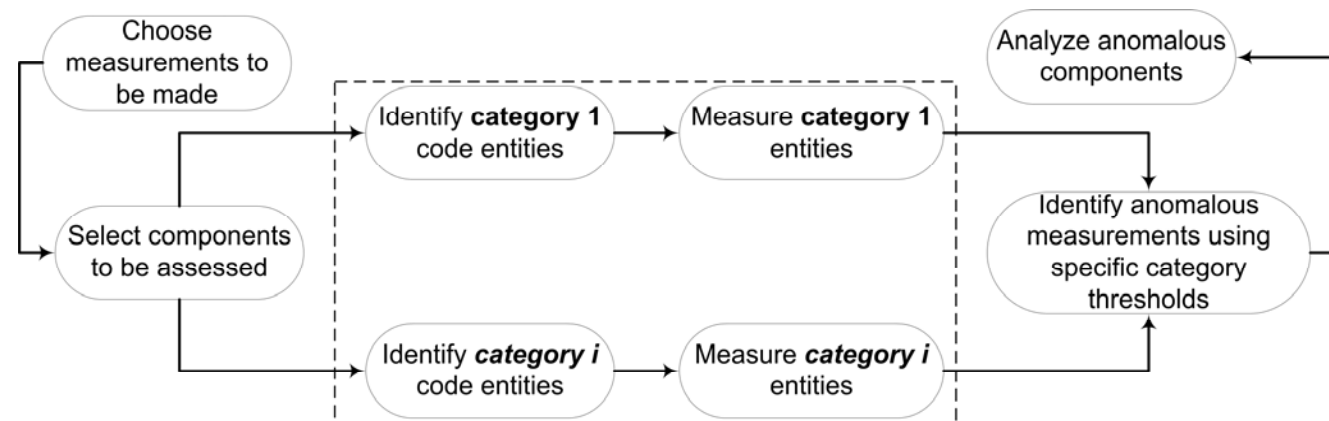
Objectives



- Introduction
- **Objectives**
- Evaluating code tools
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- Future work

- **Modify measurement process (continue)**
 - Add to the knowledge on dependency that the category of the code entity may have in identifying anomalous measurement intervals

7/25



Objectives



- Introduction
- **Objectives**
- Evaluating code tools
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- Future work

8/25

- **Modify measurement process**
 - Categories vs. UML stereotypes
 - exception, boundary (system, user and device interface), entity, control, test and utility
 - Define recommended intervals according to the entity category.
- **Adapt our process in a measurement code tool**
 - Evaluate code tools
 - Select a tool to adapt
 - Define a prototype

Evaluating code tools



- Introduction
- Objectives
- **Evaluating code tools**
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- Future work

9/25

- Measurement process needs tools
- Characteristics to evaluate tools
 - **C1.** Programming language on which the work is done.
 - **C2.** Input: binary or source files (binary/source/both).
 - **C3.** Number of metrics calculated (**C31** Chidamber and Kemerer, **C32** Lorenz and Kid, **C33** Robert Martin)
 - **C4.** Format for exporting results (html/txt/xml/xls).
 - **C5.** Graphic indicators or grouping and filtering techniques to analyze results (Yes/No).
 - **C6.** Configuration of metrics profiles.
 - **C7.** Automatic classification of code entities.
 - **C8.** Evaluation of multiple use intervals in the same evaluation.

Evaluating code tools



- Introduction
- Objectives
- **Evaluating code tools**
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- Future work

10/25

Tools	C1	C2	C3	C31	C32	C33	C4	C5	C6	C7	C8
Dependency Finder	java	binary	33	1	1	0	html, txt, xml	No	Yes	No	No
RefactorIt	java	sources	25	5	2	5	html, txt, xml	Yes	Yes	No	No
JDepend	java	binary	9	0	0	5	html, txt, xml	No	No	No	No
Eclipse Metrics - v1.3.6	java	sources	25	4	6	5	xml	No	Yes	No	No
NDepend	.NET	both	66	6	2	5	html, txt, xml, xls	Yes	Yes	No	No
SourceMonitor	java, C#, C++, VB	sources	14	0	0	0	txt, xml	Yes	Yes	No	No

Evaluating tools



- Introduction
- Objectives
- **Evaluating code tools**
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- Future work

11/25

- Criteria to select a tool
 - Open source
 - Measure Java source code
 - High number of metrics that implement each set considered
- These criteria leave RefactorIt and Eclipse Metrics as two possible candidates
- The final selection criterion is based on the
 - functionality offered RefactorIt on the metric profile management
 - user interface
 - provide a catalog of refactorings which assist in the maintenance process

Adaptation of the RefactorIt tool

- Introduction
- Objectives
- Evaluating code tools
- **Adaptation of the RefactorIt tool**
- Use intervals for code entity
- Conclusions
- Future work

12/25

■ Measurement process on RefactorIt

The screenshot shows the RefactorIt tool interface with the following components:

- Package Explorer:** Displays the project structure, including 'RefactorItLab' and 'java' packages.
- Code Editor:** Shows the 'Network.java' file with the following code:

```
/* This file is part of lanSimulation.
package ...

/**
 * A <em>Net...
 * The LAN n...
 */
public cl...
    Holds...
    Used...
    /**
    priva...
    /**
    Holds...
    Used...
    /**
    priva...
    /**
    Maps...
```
- 1. Select component to be assessed:** A dialog box showing the 'Metrics' list with 'Weighted Methods per Class (WMC)' selected. The 'Description' field states: 'Weighted Methods per Class (WMC): This calculates the sum of cyclomatic complexity metrics.' The 'Options' section shows 'Lower preferred limit: 1' and 'Upper preferred limit: 50'.
- 2. Define recommended thresholds:** A dialog box showing the 'WMC Selected metric' and the 'WMC recommended threshold' (50).
- 3. Identify anomalous measurement:** A table showing the results of the measurement process. The table has columns for 'Target', 'WMC', 'RFC', 'LCOM', 'DIT', and 'NOC'. The 'lanSimulation.internals' package is highlighted in yellow, indicating an anomalous measurement.

Target	WMC	RFC	LCOM	DIT	NOC
lanSimulation					
lanSimulation					
Network					
lanSimulation.internals	54	25	0,0	1	0
Node		28	0,0	1	0
Packet		0	0,0	1	0
lanSimulation.tests					

Adaptation of the RefactorIt tool



- Introduction
- Objectives
- Evaluating code tools
- **Adaptation of the RefactorIt tool**
- Use intervals for code entity
- Conclusions
- Future work

13/25

- The new requirements to adapt a new process
 - Open classification of code entities
 - Which categories of code entities do we want to consider?
 - Can we define new categories?
 - Classification of entities in the categories considered
 - How do we classify code entities?
 - Can any functionality of the tool assist to inspector?
 - Classification by entity groupings (by inspector)
 - Automatic classification (by algorithms)
- The adaptation of RefactorIt is named RefactorItUBU

Adaptation of the RefactorIt tool

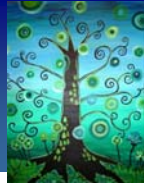


- Introduction
- Objectives
- Evaluating code tools
- **Adaptation of the RefactorIt tool**
- Use intervals for code entity
- Conclusions
- Future work

14/25

- RefactorItUBU open classification of code entities
 - Defining a configuration file from which the different categories under consideration are extracted (/refactorit_ubu/estereotipos.csv)
 - By defect standard UML stereotypes
 - *Unknown, Exception, Interface, Control, Entity, Test and Utility*
 - This classification will be used in two later activities
 - when the use/preferred interval of each metric is defined
 - when the measurement of the component is carried out

Adaptation of the RefactorIt tool

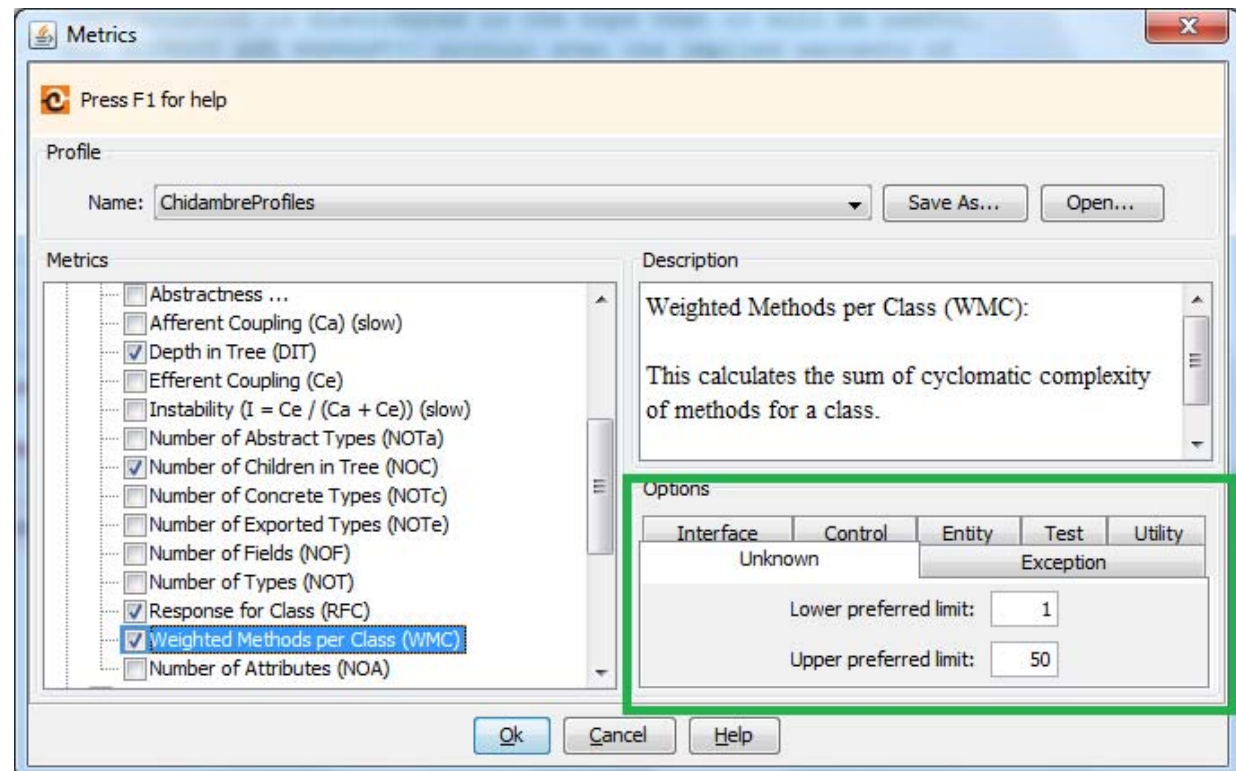


- Introduction
- Objectives
- Evaluating code tools
- **Adaptation of the RefactorIt tool**
- Use intervals for code entity
- Conclusions
- Future work

■ RefactorItUBU open classification of code entities

- when the use interval of each metric is defined

15/25



Adaptation of the RefactorIt tool



- Introduction
- Objectives
- Evaluating code tools
- **Adaptation of the RefactorIt tool**
- Use intervals for code entity
- Conclusions
- Future work

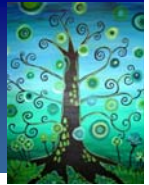
■ RefactorItUBU open classification of code entities

- when the measurement of the component is carried out

16/25

Target	STR	WMC	RFC	DIT	NOC	LCOM
Metrics	Unknown					
lanSimulation	Unknown					
LANSimulation	Unknown	9	25	1	0	0,0
Network	Unknown	54	28	1	0	0,6
lanSimulation.internals	Unknown					
Node	Unknown	2	0	1	0	0,0
Packet	Unknown	2	0	1	0	0,0
lanSimulation.tests	Test					
LANTests	Test	38	39	3	0	0,0
LANTests\$PreconditionViolationTestCase	Test	2	4	4	0	0,0

Adaptation of the RefactorIt tool



- Introduction
- Objectives
- Evaluating code tools
- **Adaptation of the RefactorIt tool**
- Use intervals for code entity
- Conclusions
- Future work

17/25

- RefactorItUbu - Classification by entity groupings by inspector
 - The physical grouping structures are
 - packages, which contain packages and classes
 - classes, which contain methods.
 - The application of a category on a grouping structure is propagated to the rest of the components
 - Example
 - An application with a logical grouping marked by a three-layered architecture could be classified by indicating the category of the three packages that contain the superior levels

Adaptation of the RefactorIt tool



- Introduction
- Objectives
- Evaluating code tools
- **Adaptation of the RefactorIt tool**
- Use intervals for code entity
- Conclusions
- Future work

18/25

- RefactorItUbu - Automatic classification by algorithms
 - Based on entities named conventions
 - For instance, the code entities whose name contains the literal strings "interface", "gui", "form", etc. usually belong to the category "*graphic interface*"
 - The inspector can define a configuration file where categories are associated a strings
 - Configuration file format
 - <package convention, category package, class convention, category class>
 - <"ui", "Interface", "listener", "Control">
 - <"test", "Test", "Exception", "Exception">

Use intervals for code entity



- Introduction
- Objectives
- Evaluating code tools
- Adaptation of the RefactorIt tool
- **Use intervals for code entity**
- Conclusions
- Future work

19/25

■ RefactorItUBU needs new data

■ Default categories considered: UML standard stereotypes

- e1 *exception*
- e2 *interface*
- e3 *entity*
- e4 *control*
- e5 *test*
- e6 *utility*
- **UnKnown**

■ For each metric in RefactorIT

- Define category use/preferred intervals
- Example: McCabe complexity metric VG
- VGe1 [1,2], VGe2 [1,2] , VGe3 [1,2] , VGe4 [1,3] , VGe5 [1,1], VGe6 [1,3]

Use intervals for code entity



- Introduction
- Objectives
- Evaluating code tools
- Adaptation of the RefactorIt tool
- **Use intervals for code entity**
- Conclusions
- Future work

20/25

- How we get category use/preferred intervals?
 - Empirical study
 - Select 10 projects from *SourceForge* (size=296.752 LOC)
 - Select and group metrics to evaluate from RefactorIt
 - package, class, method
 - size, documentation, coupling, inheritance, structural complexity, abstraction, cohesion and design principles
 - Measure the projects selected
 - Classify entity code in categories ($e_1, e_2, e_3 \dots$)
 - Create intervals per category
 - Analyze data/measurements
 - Calculate lower threshold as quartiles Q1 (percentile 25)
 - Calculate higher threshold as quartiles Q3 (percentile 75)

Use intervals for code entity



- Introduction
- Objectives
- Evaluating code tools
- Adaptation of the RefactorIt tool
- **Use intervals for code entity**
- Conclusions
- Future work

21/25

■ Some examples of intervals

- Weighted Methods per Class (WMC)
 - Chidamber and Kemerer
 - Class metric
 - Complexity metric
 - RefactorIt preferred/use intervals [1 , 50] ***
- Our empirical intervals (see figure)

Target	STR	WMC		
etrics	Unknown		e1 <i>exception</i>	[1.75 , 4.00]
lanSimulation	Unknown		e2 <i>interface</i>	[4.00 , 16.00]
+ LANSimulation	Unknown	9	e3 <i>entity</i>	[5.00 , 25.00]
+ Network	Unknown	54	e4 <i>control</i>	[3.00 , 16.25]
lanSimulation.internals	Unknown		e5 <i>test</i>	[2.00 , 8.25]
+ Node	Unknown	2	e6 <i>utility</i>	[4.00 , 22.00]
+ Packet	Unknown	2	UnKnown	[1 , 50] *** special category
lanSimulation.tests	Test			
+ LANTests	Test	38		
+ LANTests\$PreconditionViolationTestCase	Test	2		

Conclusions



- Introduction
- Objectives
- Evaluating code tools
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- **Conclusions**
- Future work

22/25

- Measurement process is modified
 - Incorporating the inspector's/evaluator's knowledge of the code entity classification
 - *Exception, Interface, Control, Entity, Test and Utility, Unknown*
- A use interval has been proposed for each metric and category of the classification
- The result of adapting the new code entity measurement process to the *RefactorIt* tool is presented
- Our conclusion is that it is necessary to pursue research into the field opened up by this case study

Future work



- Introduction
- Objectives
- Evaluating code tools
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- **Future work**

23/25

- Replicate the case study with other sets of projects
 - Refine and compare the proposed use intervals
- Validate the proposed measurement process
 - The use intervals of anomalous measurements are more accurate in the new scenario
- Validate, our measurement process proposal, with others sets of software metrics
- The labelling of code entities, as they sometimes do not belong to only one stereotype/category
 - we propose elaborating a new fuzzy classification

Future work



- Introduction
- Objectives
- Evaluating code tools
- Adaptation of the RefactorIt tool
- Use intervals for code entity
- Conclusions
- **Future work**

24/25

- The improvements achieved in this work depend on the new task of classification of entities according to the stereotypes/categories code considered. In this sense, it is necessary to carry out experiments that help to validate the consistency of classification by experts.

13th TOOLS Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2010)

The identification of anomalous code measures with conditioned interval metrics

Thank you for your attention



Carlos López, Esperanza Manso, Yania Crespo

GIRO Research Group (Grupo de Investigación en Reutilización y Orientación a Objeto)

Universities of Valladolid and Burgos, Spain

clopezno@ubu.es

{manso,yania}@infor.uva.es

