



Laboratoire  
d'Informatique  
de Robotique  
et de Microélectronique  
de Montpellier

LIRMM



um2  
UNIVERSITÉ MONTPELLIER 2  
SCIENCES ET HUMAINES



cnrs

# Mining Reusable Software Components from Object-Oriented Source Code of a Set of Similar Software

Anas Shatnawi  
Abdelhak-Djamel Seriali

1

LIRMM Laboratory, Montpellier, France

## Outlines

- Introduction
- Background: the ROMANTIC Approach.
  - [From Object to Component: the Mapping Model.](#)
  - From Object to Component: the Quality Measurement Model.
- The Proposed Approach
  - Identifying Potential Components.
  - Identifying Similar Components.
  - [Reusable Component Mining from Similar Potential Ones.](#)
  - Identifying structure of the reusable components.
  - Documentation of Components.
- Experimental Results
- Conclusions
- Future Directions

2

## Outlines

- Introduction
- Background: the ROMANTIC Approach
  - [From Object to Component: the Mapping Model.](#)
  - From Object to Component: the Quality Measurement Model.
- The Proposed Approach.
  - Identifying Potential Components.
  - Identifying Similar Components.
  - [Reusable Component Mining from Similar Potential Ones.](#)
  - Identifying structure of the reusable components.
  - Documentation of Components.
- Experimental Results.
- Conclusions.
- Future Directions.

3

## Introduction

- [One of the most important approaches](#) supporting [software](#) reuse is Component Based Software Engineering (CBSE).
- The lack of component libraries is one of the major limitations against widely use of CBSE [in](#) the industry.
- Also, software components are admitted as more reusable entities than object-oriented ones.

4

## Introduction (cont.)

- Thus,
  - Many approaches have been proposed to identify components from existing object-oriented software.
- Nevertheless, these approaches mines components by analyzing single software.
  - Thus, the mined components may be useless in other software and, consequently, their reusability is not guaranteed.

5

## Introduction (cont.)

- In many cases, companies developed many software systems
  - In the same domain, but with functional or technical variations.
  - Adding some variations to an existing software to meet the requirements of a new need.
- We propose an approach to mine reusable components from a set of similar object-oriented software
  - E.g. product variants.

6

## Introduction (cont.)

- The goal is to analyse the source code of these software to identify pieces of code that may form reusable components
  - Which will be more useful (reusable) for the development of new software than those mined from singular ones.

7

## Outlines

- Introduction
- Background: the ROMANTIC Approach
  - [From Object to Component: the Mapping Model.](#)
  - From Object to Component: the Quality Measurement Model.
- The Proposed Approach.
  - Identifying Potential Components.
  - Identifying Similar Components.
  - [Reusable Component Mining from Similar Potential Ones.](#)
  - Identifying structure of the reusable components.
  - Documentation of Components.
- Experimental Results.
- Conclusions.
- Future Directions.

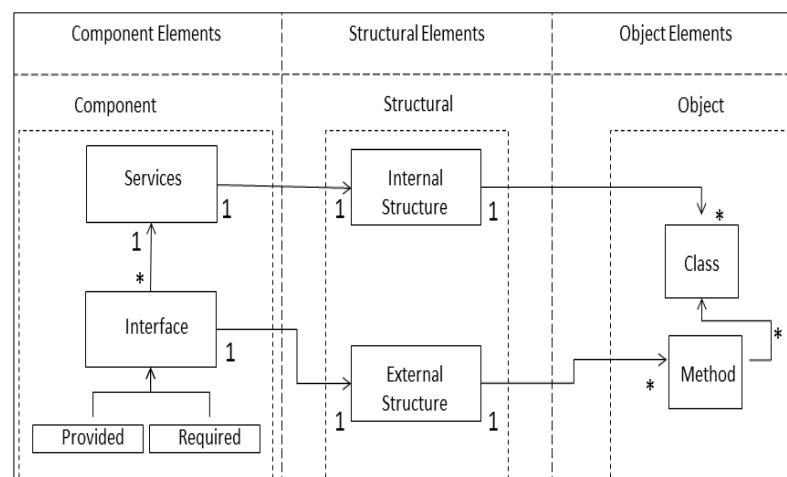
8

## The ROMANTIC Approach

- In our previous works, we have proposed the *ROMANTIC* approach
  - To extract a component-based architecture from an object-oriented software.
- ROMANTIC is mainly based on two models:
  - Mapping model.
  - Quality measurement model.
- We rely on these two models to mine reusable components from similar software.

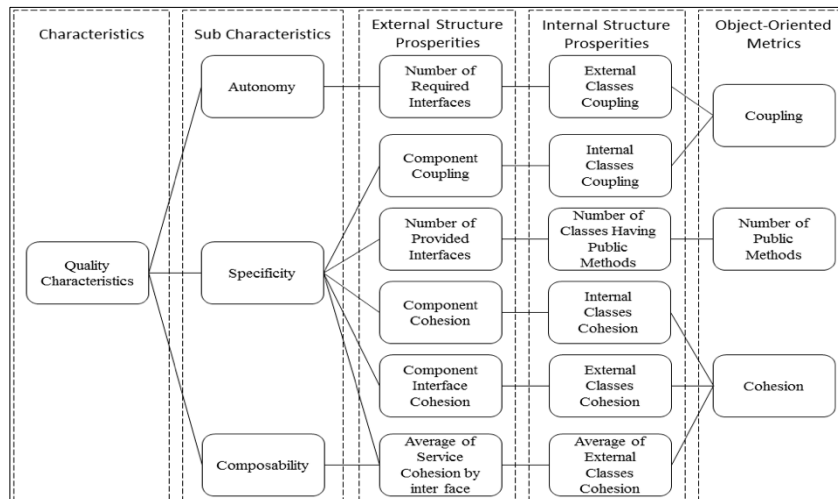
9

## From Object to Component: the Mapping Model



10

## From Object to Component: the Quality Measurement Model



11

## From Object to Component: the Quality Measurement Model (cont.)

- $Q(E) = 1/\sum i \uparrow \lambda_i (\lambda_1 * S(E) + \lambda_2 * A(E) + \lambda_3 * C(E))$
- Where:
  - $E$  is an object-oriented component composed of a group of classes.
  - $S(E)$ ,  $A(E)$  and  $C(E)$  refer to the specificity, autonomy, and composability of  $E$  respectively.
  - $\lambda_1$ ,  $\lambda_2$ ,  $\lambda_3$  are weight values, situated in  $[0-1]$ . These are used by the architect to weight each characteristic as needed.

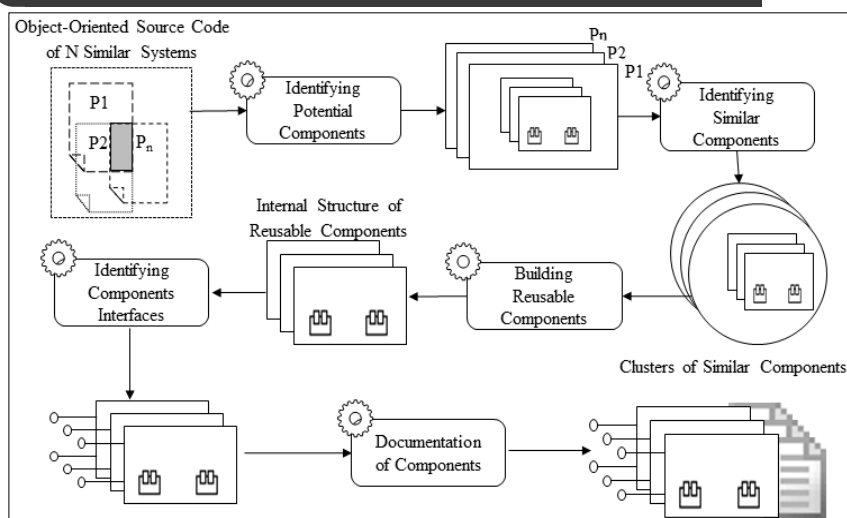
12

## Outlines

- Introduction.
- Background: the ROMANTIC Approach.
  - From Object to Component: the Mapping Model.
  - From Object to Component: the Quality Measurement Model
- The Proposed Approach
  - Identifying Potential Components.
  - Identifying Similar Components.
  - Reusable Component Mining from Similar Potential Ones.
  - Identifying structure of the reusable components.
  - Documentation of Components.
- Experimental Results.
- Conclusions.
- Future Directions.

13

## The Proposed Approach



14

## Identifying Potential Components

- A potential component is a group of classes that are gradually formed starting from an object-oriented core class.
- Each class can be considered as a potential core class to form a component.
- Other classes are identified based on incrementally fusion process.

15

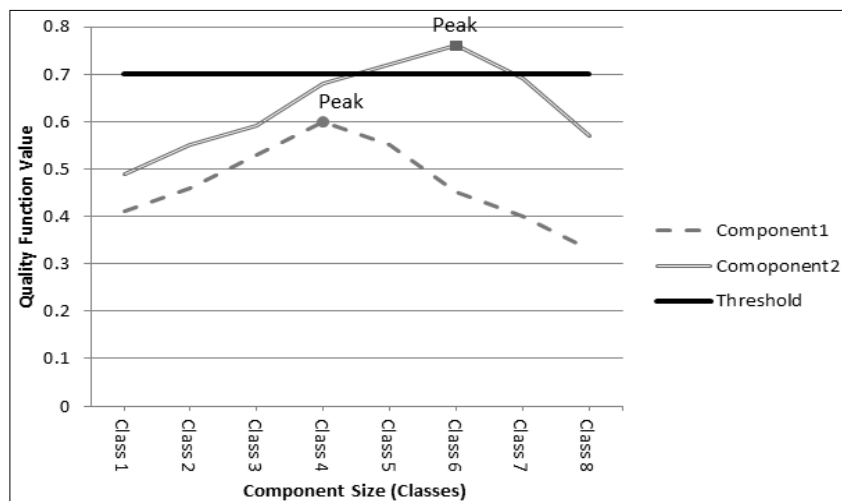
## Identifying Potential Components (cont.)

- The selection of the class to be added at each step is decided based on the value of the quality of the resulted component.
  - The class obtaining the highest quality value is selected to extend the current group.
  - We do this until all classes are grouped into a single group.
- Some classes of this group will be excluded.
  - Classes that are added after the quality function reaches the peak value.

16



## Identifying Potential Components (cont.)



17

## Identifying Similar Components

- Potential components are mined from similar systems
  - Thus, some of them may be similar.
- Similar components are those share the majority of its classes and differ considering few ones.
- These components may be considered as variants of one common component
  - Which is considered more reusable.

18

## Identifying Similar Components (cont.)

- Thus, similar components are gathered into groups.
- Building groups of similar components are based on lexical similarity metrics
  - The strength of similarity links between classes composing each component.
  - Each component is considered as a text document where the content is composed of a list of component classes' names.
  - Cosine similarity metric.

19

## Identifying Similar Components (cont.)

- Hierarchical clustering algorithm to gather similar components into groups.
  - Individual components as initial leaf nodes in a binary tree.
  - The two most similar nodes are grouped into a new one.
  - This is continued until all nodes are grouped.
  - Depth first search algorithm is used to find the cut-off points.
    - A node has a similarity value exceeding the average similarity value of its children.

20

## **Reusable Component Mining from Similar Potential Ones**

- One common component is extracted from each group of similar components.
- A collection of classes that is used to form the reusable component is composed of
  - All shared classes.
  - Some of non-shared.

21

## **Reusable Component Mining from Similar Potential Ones (cont.)**

- Shared classes form the core of the reusable component.
  - But considering only the shared classes may not form a correct component following our quality measurement model.
- Selecting a non-shared class
  - The density of non-shared class.
  - The quality of the component.

22

## Reusable Component Mining from Similar Potential Ones

- **Steps:**

- Extracting all candidate subsets among non-shared classes of the group.
- Subsets that reach a predefined density threshold are only taken into consideration.
- Evaluating the quality of the subsets.
- The subset that maximizes the quality value is grouped with the core classes to form the reusable component.

23

## Identifying structure of the reusable components

- A component is used based on its provided and required interfaces.
  - Provided interfaces are composed of the public methods of classes that compose its external structure.
  - Required interfaces are composed of the methods that are used from the other components
    - i.e. the provided interfaces of the other components.

24

## Identifying structure of the reusable components (cont.)

- We rely on the following heuristics:
  - A group of methods belongs to the same object-oriented interface may belong to the same component's interface.
  - Cohesive and lexically similar methods have high probability to belong to the same interface.
  - When methods are called many times together, this is an indicator of a high correlation of use.

25

## Identifying structure of the reusable components (cont.)

- $Interface(M) = 1 / \sum_i \lambda_i ( \lambda_1 * SI(M) + \lambda_2 * SM(M) + \lambda_3 * CU(M) + \lambda_4 * CI(M) )$
- Where:
  - $M$ : a set of methods.
  - $SI$ : measures how much a set of methods  $M$  belongs to the same object-oriented interface.
  - $SM$ : measures how much a set of methods  $M$  is similar using cosine and cohesion (LCC) metrics.
  - $CU$ : measures how many times a set of methods  $M$  has been called together by the same component.
  - $CI$ : measures how many times a set of methods  $M$  is invoked together.

26

## Identifying structure of the reusable components (cont.)

- This function is used as a fitness function in a hierarchical clustering algorithm to partition a set of public methods into a set of clusters
  - Where each cluster is a component's interface.

27

## Documentation of Components

- The documentation of a component helps the developers to find a component that meets their needs.
- The description of the component functionalities forms an important part of its documentation.
- Thus, we propose to identify for each mined component its main functionalities.

28

## Documentation of Components (cont.)

### 1- Identifying the component functionalities

- The specificity of a component refers to the functionalities that are provided
  - Number of public methods is proportional to the number of functionalities.
  - Classes providing the same functionalities must be cohesive.
  - Classes participating in the same functionality must have a high cohesion with themselves and low coupling with other parts in the component.

29

## Documentation of Components (cont.)

### 1- Identifying the component functionalities

- $S(E) = 1/5 * (1/|I| * \sum_{i \in I} LCC(i) + LCC(I) + LCC(E) + Coupl(E) + noPub(I))$
- We use this equation as a fitness function in a hierarchical clustering algorithm to decompose component classes into partitions
  - Where each one represents one of the functionality of the analyzed component.

30

## Documentation of Components (cont.)

### 2- Generation of the functionality description

- The description consists of the most frequent words in the partition classes' names.
- A class name is often a set of nouns concatenated by the camel-case notation.
  - These nouns are representing a meaningful name for the main purpose of the class.
  - The first noun in a class name holds the main goal of the class, and so on.

31

## Documentation of Components (cont.)

### 2- Generation of the functionality description

- We propose the following three steps.
  1. Tokens are extracted by separating the classes names according to the camel-case syntax
    - E.g. MediaController is divided into Media, and Controller.
  2. A weight is affected to each extracted token
    - The tokens which are the first word of a class name are given a large weight. Other tokens are given a small weight.
  3. Tokens which have the highest weight is used to construct the functionality description in an orderly manner.

32



## Documentation of Components (cont.)

### 2- Generation of the functionality description

- $Weight(w) = 1 / \sum_{i=1}^4 N_i * (1 * N_1 + 0.75 * N_2 + 0.50 * N_3 + 0.25 * N_4)$
- Where:
  - $W$ : refers to a word.
  - $N_i$  refers to the number of occurrence of the word  $w$  in the position  $i$ .

33

## Outlines

- Introduction.
- Background: the ROMANTIC Approach.
  - [From Object to Component: the Mapping Model.](#)
  - From Object to Component: the Quality Measurement Model.
- The Proposed Approach.
  - Identifying Potential Components.
  - Identifying Similar Components.
  - [Reusable Component Mining from Similar Potential Ones.](#)
  - Identifying structure of the reusable components.
  - Documentation of Components.
- Experimental Results.
- Conclusions.
- Future Directions.

34

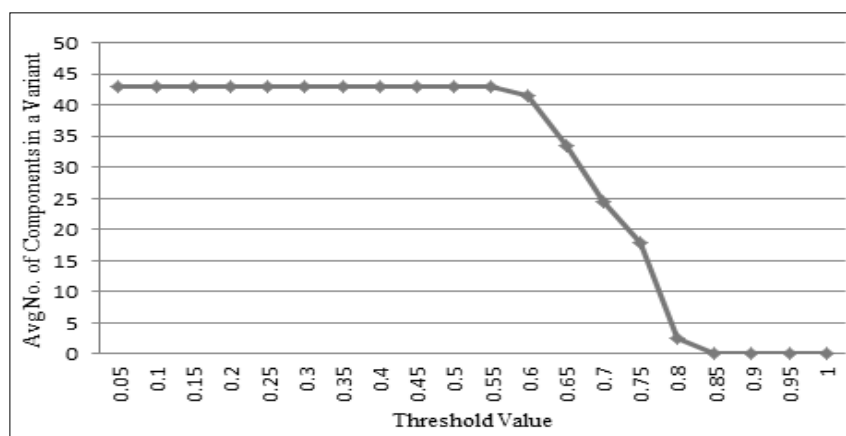
## Experimental Results

- We have applied it onto two open source Software Product Line Java applications of different sizes
  - Mobile Media.
  - ArgoUML-SPL.
- To consider that a group of classes forms a component, its quality function value should exceed a predefined quality threshold.
- We tested the quality threshold value from 0 up to 1 by incrementing it 0.05 in each run.

35

## Experimental Results (cont.)

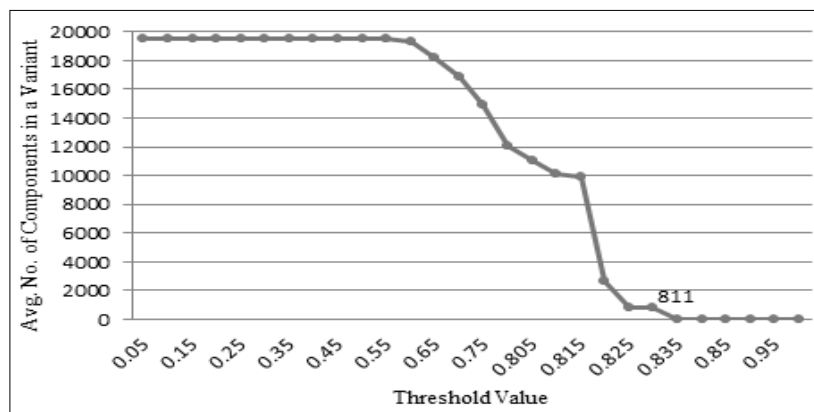
- Changing the Quality threshold value to extract all potential components in Mobile Media.



36

## Experimental Results (cont.)

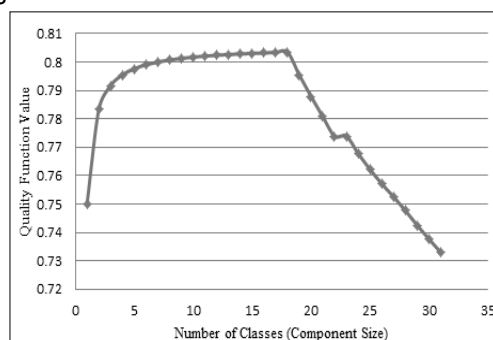
- Changing the Quality threshold value to extract all potential components in ArgoUML-SPL.



37

## Experimental Results (cont.)

- An instance of a potential component extracted from ArgoUML-SPL.
  - GoClassToNavigableClass as the core class.
  - the 18 first classes form this potential component.
  - The remaining classes are rejected.



38

## Experimental Results (cont.)

- The results of potential components extraction, when 0.70 and 0.83 are assigned as threshold value respectively for Mobile Media and ArgoUML.

Product Name	AVG # of potential components in all variants	AVG component size (classes)	AVG Spe	AVG Aut	AVG Com
Mobile Media	24.5	6.45	0.56	0.71	0.83
ArgoUML-SPL	811	11.38	0.64	0.83	0.89

39

## Experimental Results (cont.)

- The results of component's clustering

Product	# of clusters	AVG number of components in a cluster	AVG number of shared classes in a cluster
Mobile Media	42	5.38	5.04
ArgoUML-SPL	325	5.26	8.67

40

## Experimental Results (cont.)

- The final set of mined components, when we assign 0.50 to the density threshold value.

Product	# of mined reusable components	AVG component size	AVG Spe	AVG Aut	AVG Com
Mobile Media	39	5.61	0.58	0.74	0.90
ArgoUML-SPL	324	9.77	0.61	0.84	0.84

41

## Experimental Results (cont.)

- Some components from Mobile Media.

Description of the functionalities	# of variants that contains this component	Size (class)	Spe.	Aut.	Cop.
New Constants Screen Album Image	6	6	0.59	0.75	0.94
Add Constants Photo Album	8	10	0.57	0.75	0.89
Count Software Splash Down Screen					
Base Image Constants Album Screen Accessor List	6	9	0.67	0.50	0.85
Controller Image Interface Thread					

42

## Reusability Validation

- To validate the reusability of our results
  - Comparing them with ones that are mined from singular system.
- The reusability of a component
  - The ratio between the number of systems that can reuse this component to the number of all systems.

43

## Reusability Validation (cont.)

- K-fold cross validation method
  - Validate the results of the mining model.
  - Partitioning the data set into two parts
    - Train data: to learn the mining model.
    - Test data: to validate the mining model.
  - Divide the data set into K parts
    - K-1 parts as train data.
    - The other one as test data.

44

## Reusability Validation (cont.)

K	Similar Systems	Singular System
2	32%	28%
4	18%	15%
8	09%	07%

45

## Reusability Validation (cont.)

- The slight difference between the reusability results comes from the nature of our case studies
  - Where these case studies are very similar.
  - Consequently, the resulting components are closely similar
    - i.e. there are many groups of similar components containing exactly the same classes which resulted the same reusable component.
  - Therefore, there is very small difference in the results.

46

## Outlines

- Introduction.
- Background: the ROMANTIC Approach.
  - From Object to Component: the Mapping Model.
  - From Object to Component: the Quality Measurement Model.
- The Proposed Approach.
  - Identifying Potential Components.
  - Identifying Similar Components.
  - Reusable Component Mining from Similar Potential Ones.
  - Identifying structure of the reusable components.
  - Documentation of Components.
- Experimental Results.
- Conclusions.
- Future Directions.

47

## Conclusions

- Mining components from similar software provides more guarantees for the reusability of the mined components rather than depending on single software.
- An approach is proposed to mine reusable components from a set of similar object-oriented systems.

48



## Conclusions (cont.)

- The results show that
  - There are components that are shared in many systems.
  - These ones are more reusable.
- There are two aspects to be considered regarding the hypothesis of our approach
  - We consider that the variability between software is in the class level.
  - Forming a component by adding a non-shared class to the core ones may cause a dead code.

49

## Outlines

- Introduction.
- Background: the ROMANTIC Approach.
  - [From Object to Component: the Mapping Model.](#)
  - From Object to Component: the Quality Measurement Model.
- The Proposed Approach.
  - Identifying Potential Components.
  - Identifying Similar Components.
  - [Reusable Component Mining from Similar Potential Ones.](#)
  - Identifying structure of the reusable components.
  - Documentation of Components.
- Experimental Results.
- Conclusions.
- Future Directions.

50

## Future Directions

- The future directions will focus on migrating similar software into component based software product line.