A study about Ant Colony to component-based software architecture optimization

Mariane Affonso Medeiros ,
Filipe Roseiro Côgo, Marco Aurélio Graciotto Silva

marianeaffonsomedeiros@gmail.com,
{filiper,magsilva}@utfpr.edu.br

21 de setembro de 2016

Introduction

Some works that propose optimization of component-based software architecture.

- [**Hussain-etal:2015**]
  - Particle Swarm Optimization - PSO;
- [**Ramirez-etal:2015**]
  - Genetic Algorithm;
- [**Mueller:2014**]
  - Ant Colony Optimization (ACO);

Introduction

Some works that propose optimization of component-based software architecture.

- [**Hussain-etal:2015**]
  - Particle Swarm Optimization - PSO;
- [**Ramirez-etal:2015**]
  - Genetic Algorithm;
- [**Mueller:2014**]
  - Ant Colony Optimization (ACO);

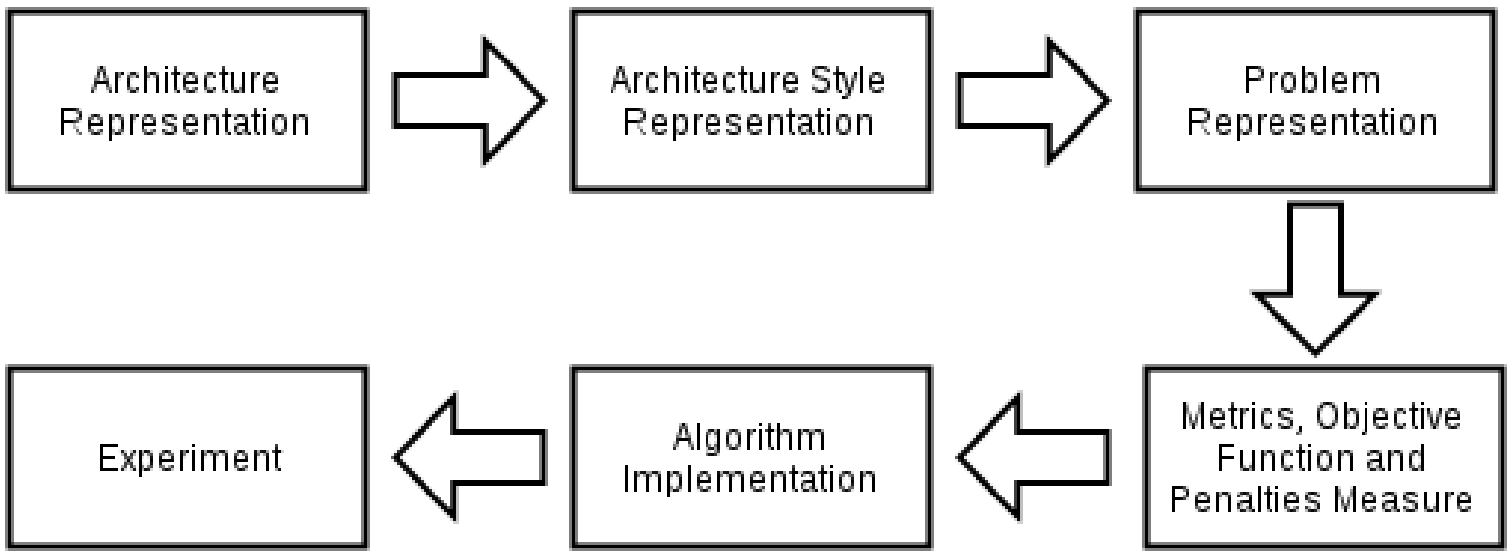
Lack of approaches that consider architectural style during the optimization.

Objective

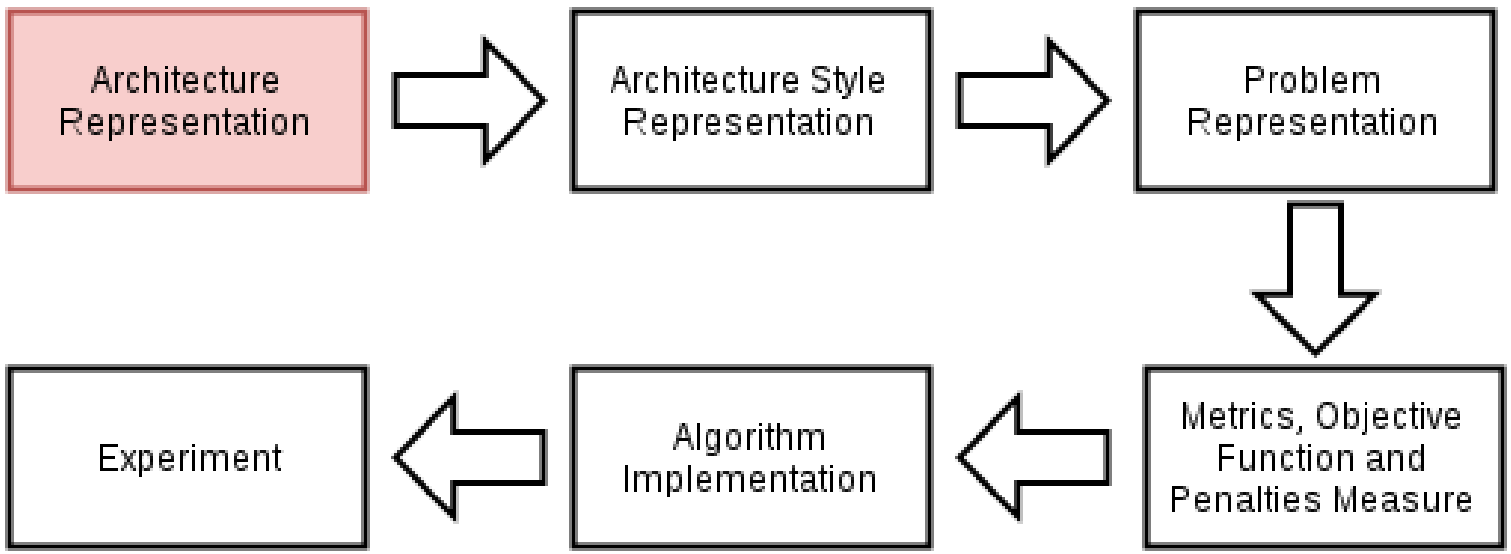
The aim of this study is to observe the behavior of Ant Colony to optimization of component-based software architecture, considering coesion, coupling and preservation of the architectural style.

- Comparison of proposed solutions by algorithm with the original architecture;
- Observe how is the evolution of metric value in relation to values assigned to the parameters of metaheuristic.

Method


Architecture Representation
Architecture Style Representation
Problem Representation
Metrics, Objective Function and Penalties Measure
Algorithm Implementation
Experiment

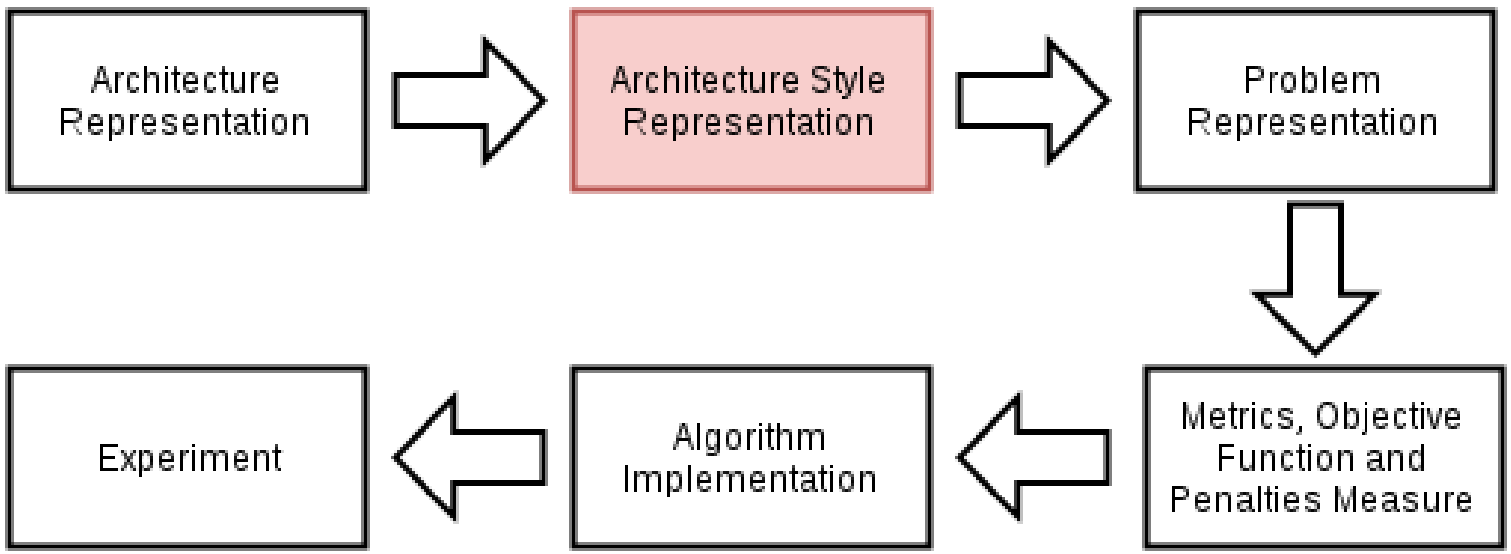
Method


Architecture Representation
Architecture Style Representation
Problem Representation
Metrics, Objective Function and Penalties Measure
Algorithm Implementation
Experiment

Representation of Architecture Model

- UML Model;
- API UML2 to process the UML model;
- From UML model we extract the informations:
  - Number of classes;
  - Number of packages
  - Relationships between classes and packages;
  - Relationships between classes of the architecture.
- Recovery UML Model;
- plugin Eclipse Modisco;

Method

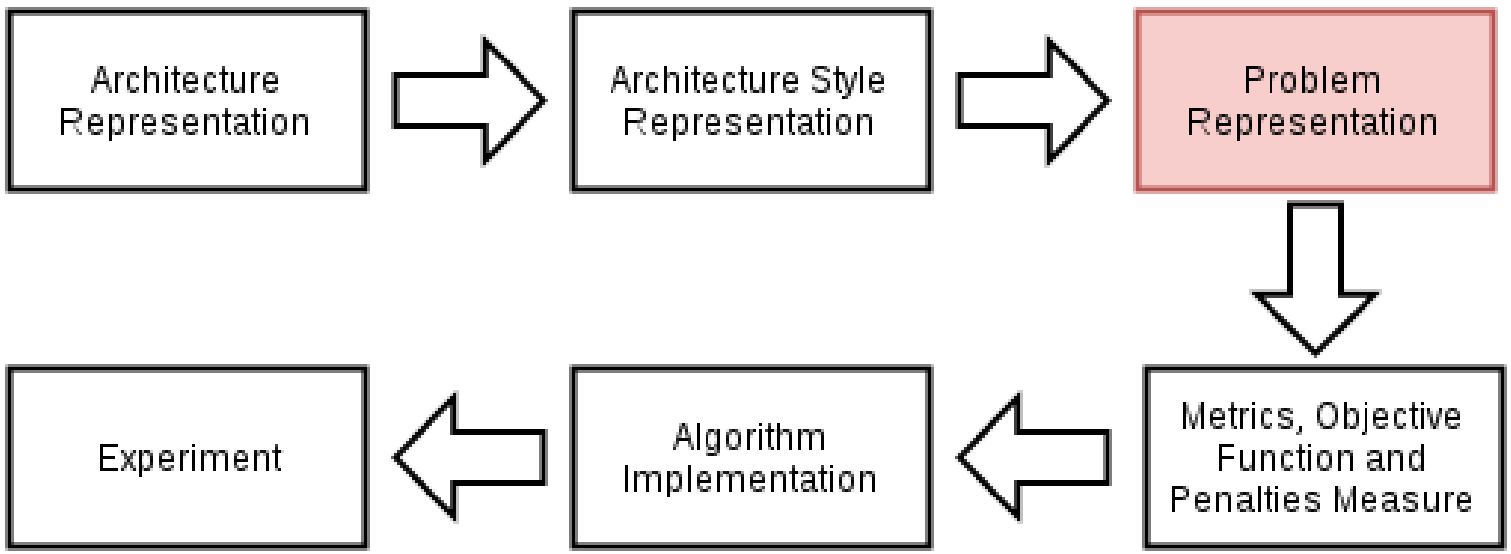

Architecture Representation
Architecture Style Representation
Problem Representation
Metrics, Objective Function and Penalties Measure
Algorithm Implementation
Experiment

Style Representation

Architecture style considered:

- Layered Architecture;
- Profiles and Stereotypes UML.
- Profile defined: **ArchitectureStyle**;
- Stereotype defined to layered architecture: **Layered**

Method


Architecture Representation
Architecture Style Representation
Problem Representation
Metrics, Objective Function and Penalties Measure
Algorithm Implementation
Experiment

Problem Representation

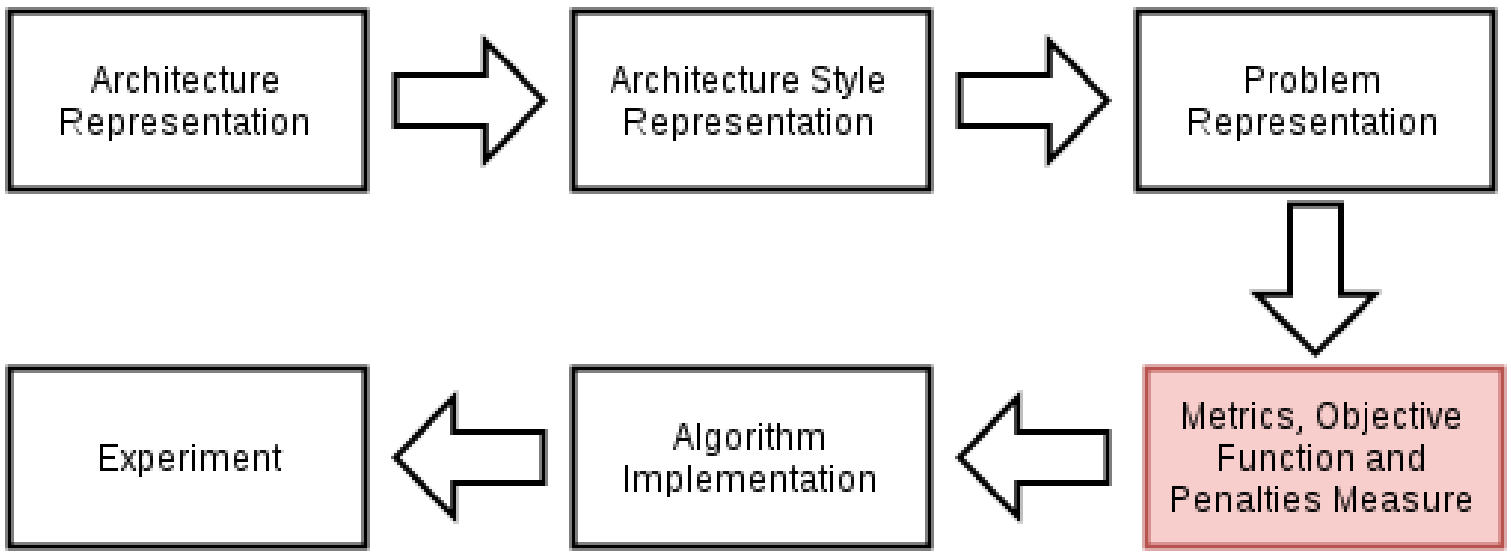
- Matrix that represents relationships between classes and components;

	Comp1	Comp2	Comp3	N
C1	0.5	0.5	0.7	0.5
C2	0.5	0.7	0.5	0.5
C3	0.5	0.5	0.5	0.5
C4	0.5	0.8	0.7	0.5

- Matrix that represents relationships between classes;

	C1	C2	C3	C4	N
C1	0.5	0.6	0.5	0.5	0.5
C2	0.5	0.5	0.5	0.5	0.5
C3	0.5	0.5	0.5	0.5	0.5
C4	0.5	0.5	0.5	0.5	0.5

Method


Architecture Representation
Architecture Style Representation
Problem Representation
Metrics, Objective Function and Penalties Measure
Algorithm Implementation
Experiment

Metric and Objective Function

Metrics to evaluate architecture quality:

- Coesion;
- Coupling.

This two metrics are encapsulated in a objective function called:
Modularization Quality (MQ)

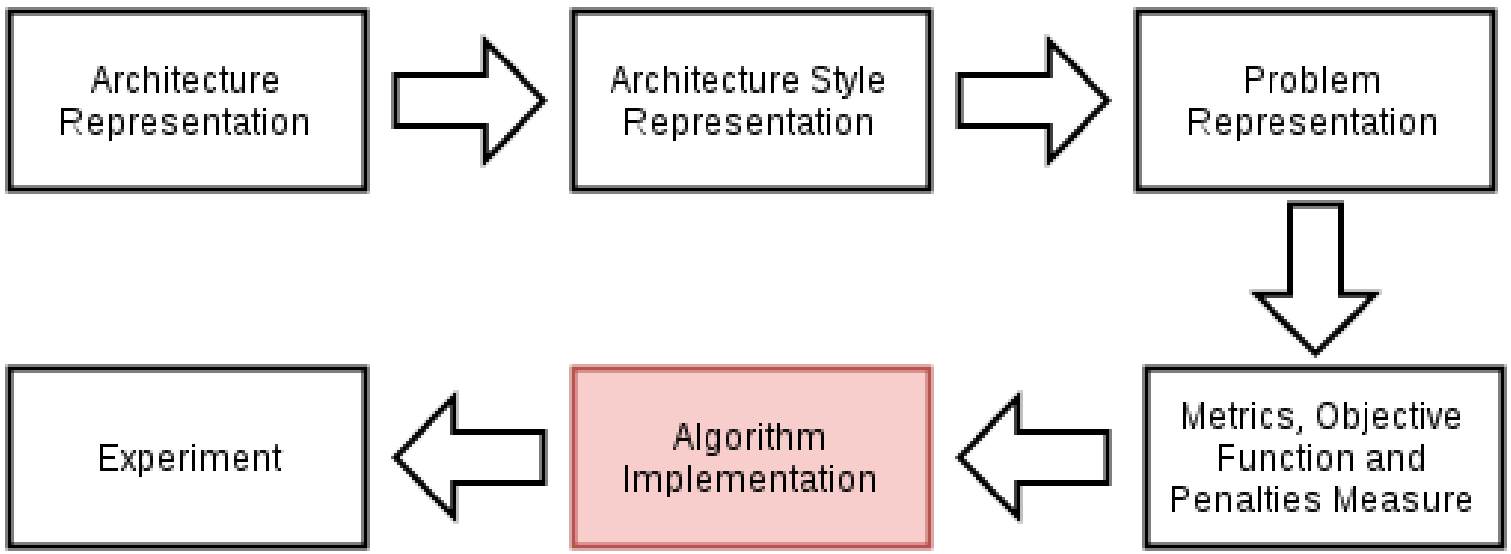
$$MQ = \begin{cases} \frac{\sum_{i=1}^{k} A_i}{k} - \frac{\sum_{i,j=1}^{k} E_{i,j}}{\frac{k \cdot (k-1)}{2}} & \text{se k} > 1 \\ A_1 & \text{se k} = 1 \end{cases} \quad (1)$$

Architecture Style

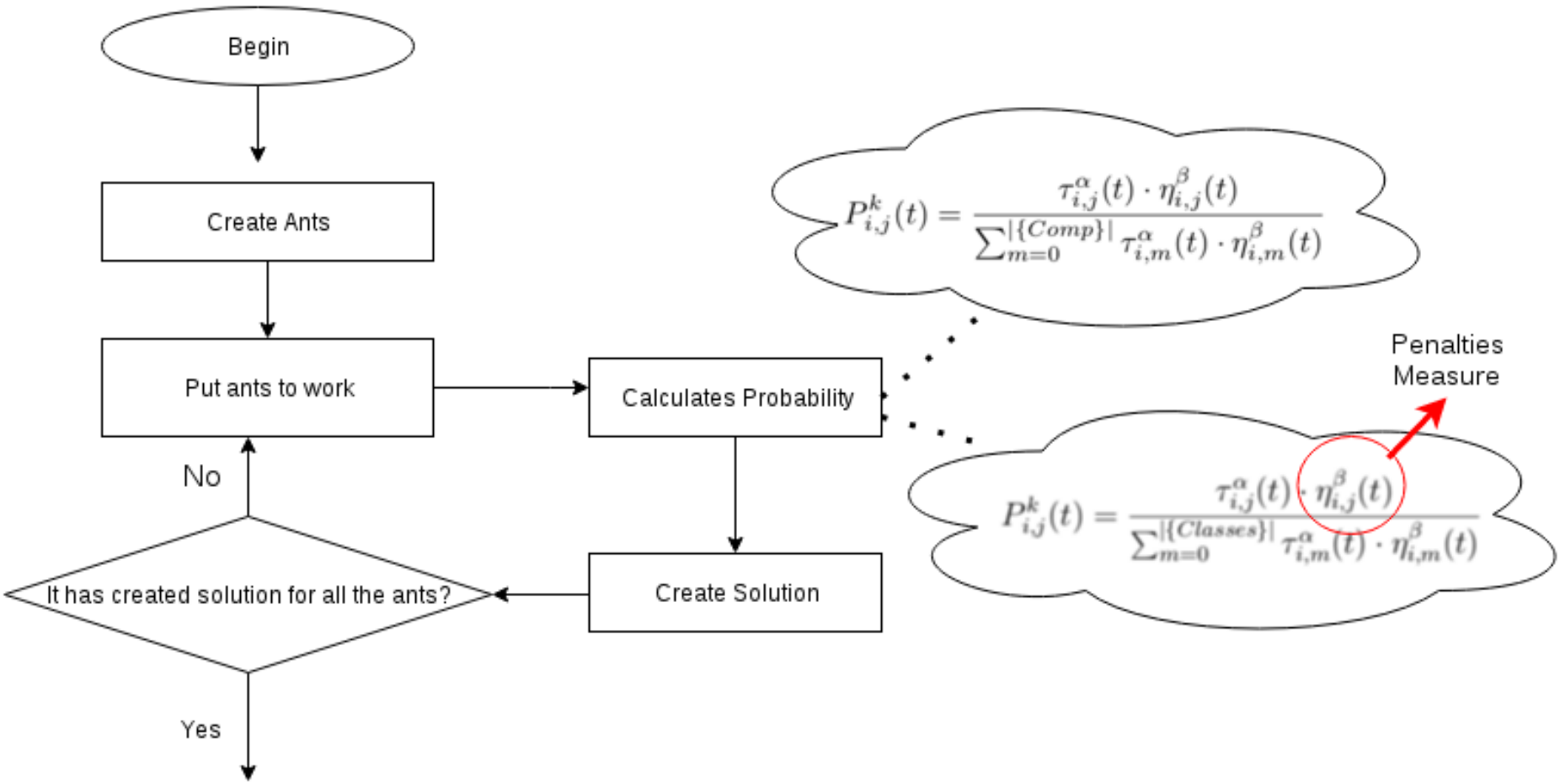
- To check if architecture style is violated we use penalizations;
- We apply penalizations to the relationships that break the rules of architecture style;

$$penalidade_{i,j} = \begin{cases} 1 - \frac{total_i + total_j}{totalGeralDeQuebrasDaRegra} & \text{if there is style} \\ 1 & \text{if there is no style} \end{cases} \quad (2)$$

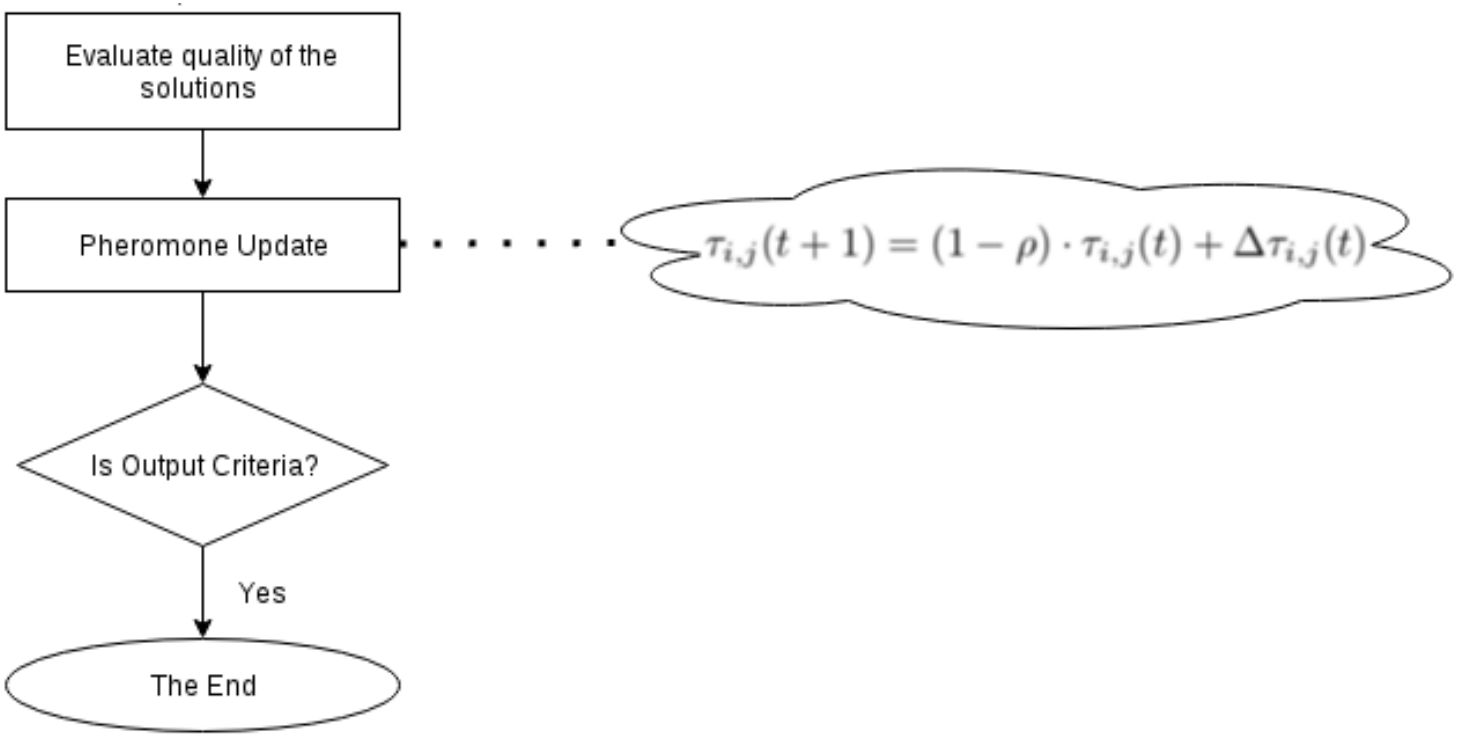
Method


Architecture Representation
Architecture Style Representation
Problem Representation
Metrics, Objective Function and Penalties Measure
Algorithm Implementation
Experiment

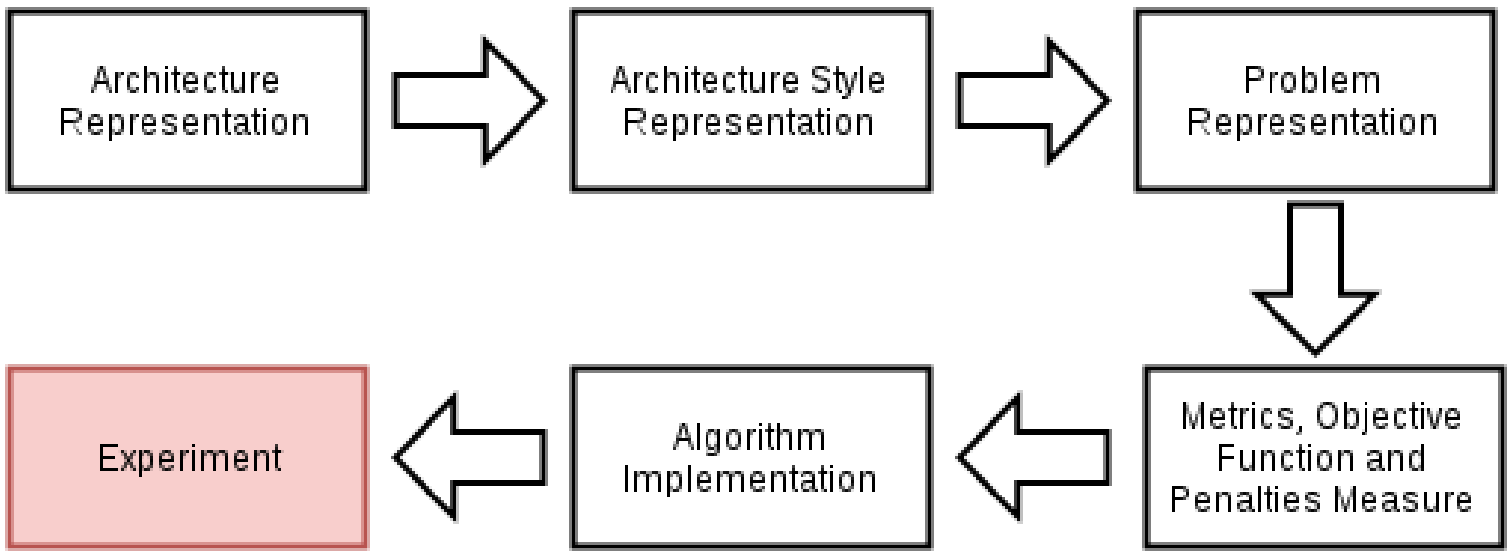
ACO to Architecture Optimization



ACO to Architecture Optimization



Method

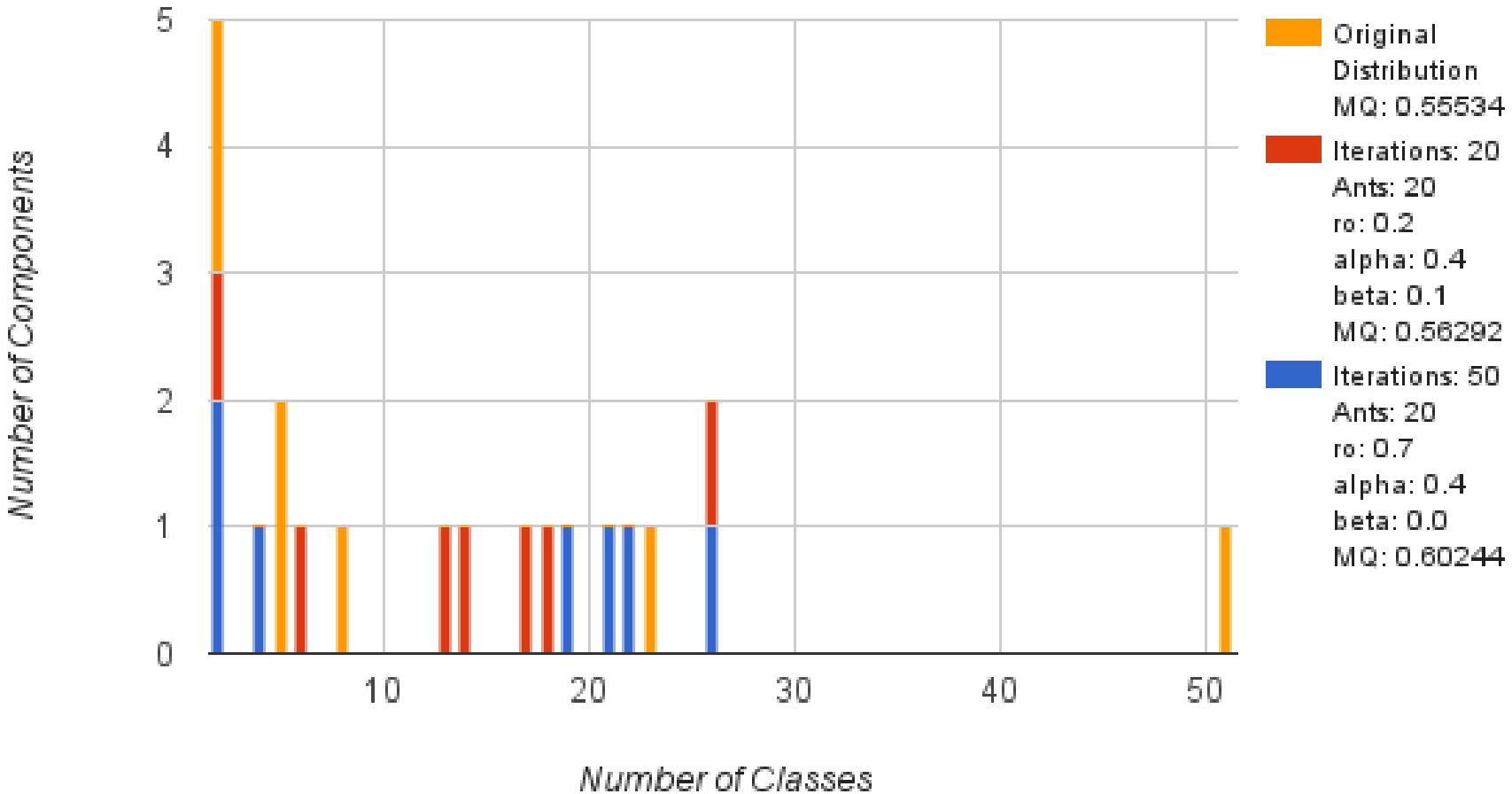

Architecture Representation
Architecture Style Representation
Problem Representation
Metrics, Objective Function and Penalties Measure
Algorithm Implementation
Experiment

Experiment

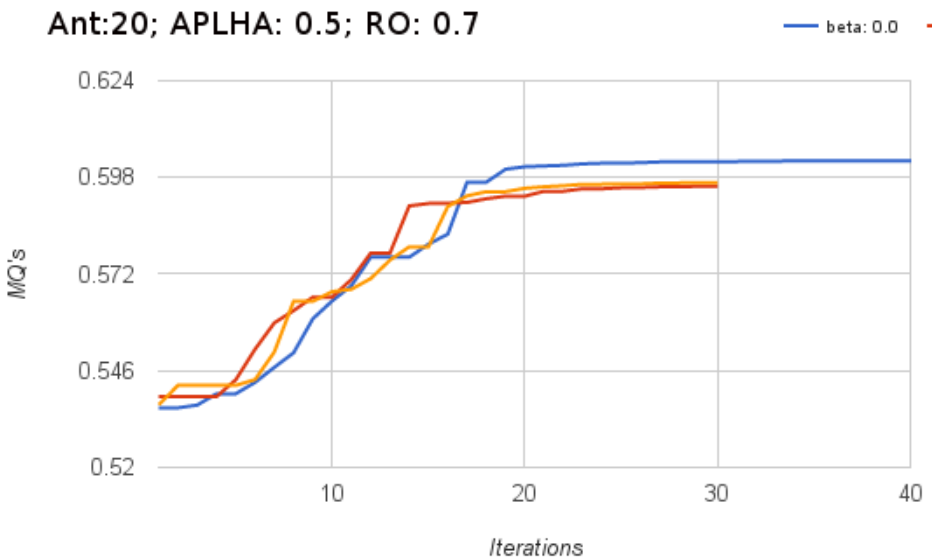
- Version 1.1.0 of software Apache Ant;
- Realized 1 experiment;
- To each test done, we apply many combinations of the parameters configuration (ρ, α, β, number of ant and iterations);
- Each configuration was performed 10 times;

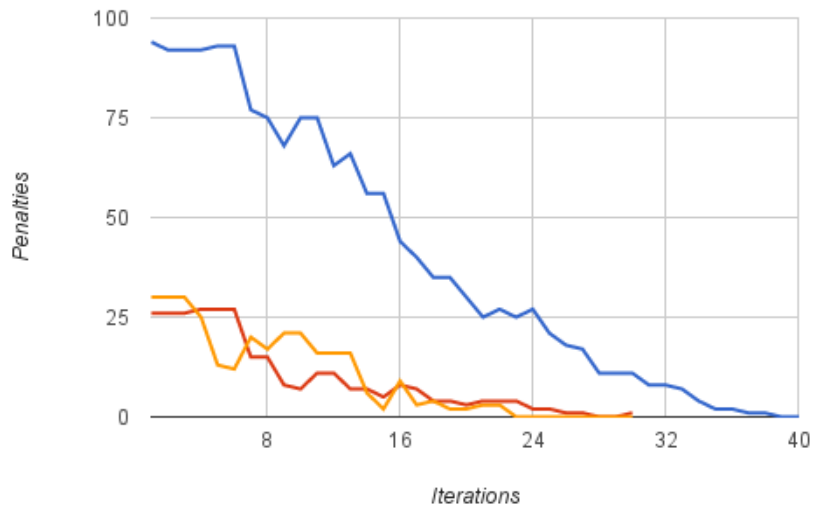
Project	Classes	Components	Layers	MQ
Apache Ant 1.1.0	97	7	6	0.55534

Distribution of classes in components



Evolution of MQ and Penalties





Conclusion

- ACO has achieved value of MQ 11% higher than original architecture;
- The MQ metric tends bring to zero the number of external relationships, causing that the layered architecture style do not be contravened;
- However, if consider fewer iterations we observed that the number of penalties, to solutions that have values attributed to beta, is smaller than the solutions that were not attributed beta values.

Future Works

- Apply more experiments, with other projects;
- Evaluate others architecture styles;
- Do the multi-objective metaheuristic;

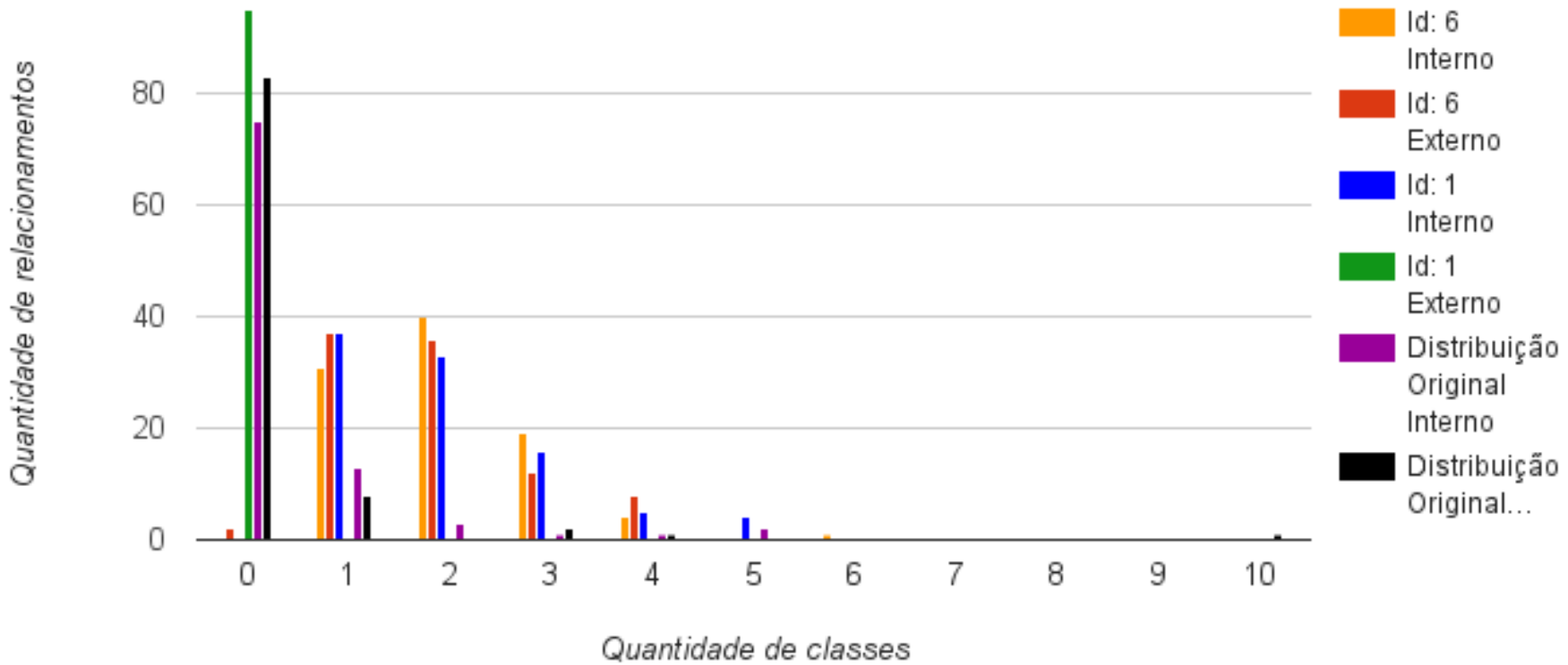
A study about Ant Colony to component-based software architecture optimization

Mariane Affonso Medeiros ,
Filipe Roseiro Côgo, Marco Aurélio Graciotto Silva

marianeaffonsomedeiros@gmail.com,
{filiper,magsilva}@utfpr.edu.br

21 de setembro de 2016

Number of classes and internal and external relationships of Apache Ant 1.1.0 with architectural style



Results obtained by ACO to version 1.1.0 without architectural style

Id	Iterations	Ants	ρ	α	β	MQ
1	100	15	0.2	0.4	0.2	0.61802
2	100	5	0.4	0.2	0.2	0.61765
3	100	15	0.4	0.2	0.4	0.61033
4	100	5	0.1	0.9	0.8	0.59660
5	50	20	0.7	0.4	0.9	0.59499
6	30	20	0.4	0.6	0.9	0.59274
7	20	20	0.6	0.4	0.1	0.59023

Results obtained by the ACO to version 1.1.0 with architectural style

Id	Iterations	Ants	ρ	α	β	MQ	Rel. Penalizados
1	50	20	0.7	0.4	0.0	0.60244	0
2	100	15	0.8	0.2	0.4	0.60019	0
3	30	20	0.7	0.4	0.3	0.59660	3
4	30	20	0.7	0.4	0.7	0.59563	0
5	20	20	0.5	0.4	0.1	0.58182	8
6	20	20	0.2	0.4	0.1	0.56292	16