Otimização de arquitetura de software utilizando Sistema de Colônia de Formigas

Mariane Affonso Medeiros
Orientador: Marco Aurélio Graciotto Silva
Coorientador: Filipe Roseiro Côgo

22 de junho de 2016

Introdução

- Problemas enfrentados pela arquitetura de software: **alto índice de mudanças** e a **grande dependência do ser humano;**
  - degradação e degeneração da arquitetura;
  - alta taxa de retrabalho;
  - desperdício de tempo;
- Dado que o design arquitetural possui critérios pré-estabelecidos que devem ser considerados durante a definição da arquitetura, podemos fazer uso de **métodos de busca** para definir a arquitetura de software;

Objetivo

- Otimização de arquitetura de software baseada em componentes utilizando a metaheuristica Sistema de Colônia de Formigas.

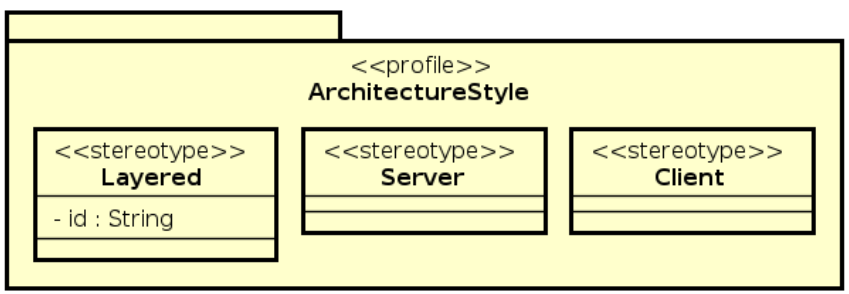
Representação do Modelo Arquitetural

- Modelo UML;
- API UML2 para processar o modelo UML;
- A partir do modelo extraímos as seguintes informações:
  - quantidade de classes,
  - pacotes,
  - relacionamentos entre classes e pacotes e
  - relacionamentos entre as classes da arquitetura.
- Recuperação do modelo UML;
- plugin Eclipse Modisco;

Representação do Estilo

Os estilos arquiteturais considerados neste trabalho foram estilo em camadas e cliente/servidor.

- O algoritmo permite a entrada de projetos com estilo arquitetural definido no modelo UML.
- O estilo arquitetural auxilia o ACO na busca pela solução mais adequada.
- Perfis (*Profiles*) e Estereótipos (*Stereotypes*) UML.



Representação do Problema

- Matriz representando os relacionamento classes e componentes;
- Matriz representando os relacionamentos entre as classes;

	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

	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

Métricas e Função Objetivo

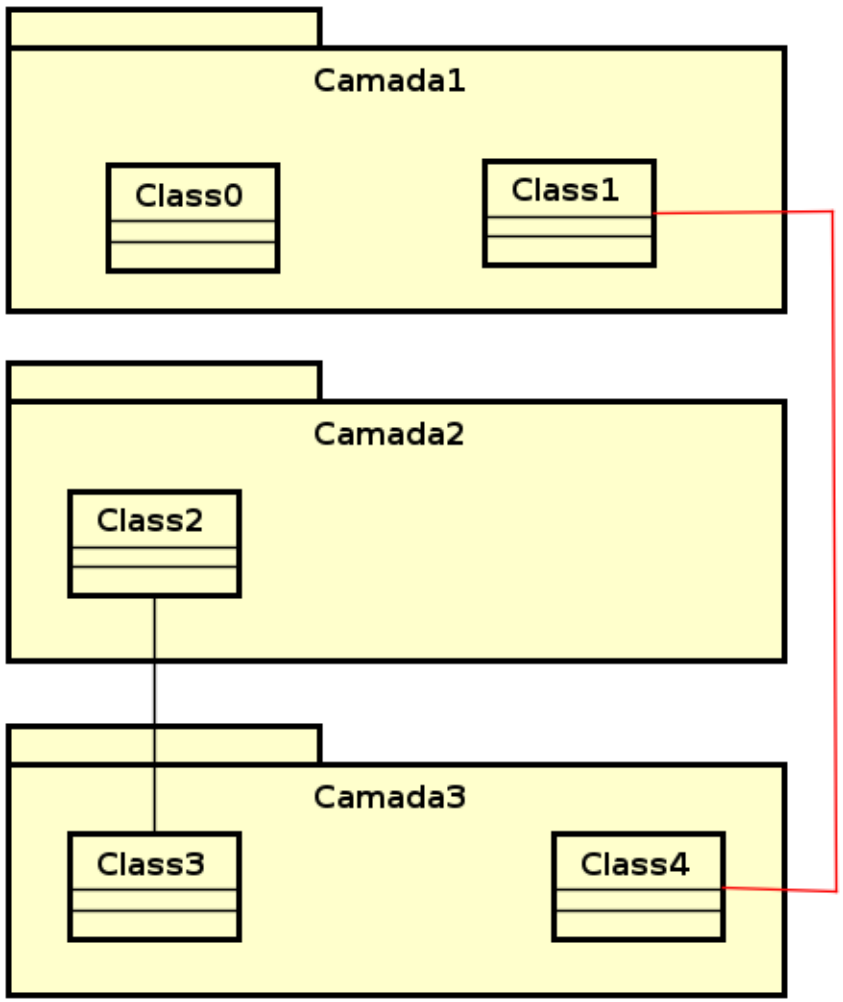
Métricas consideradas para avaliar a qualidade arquitetural:

- Coesão
- Acoplamento

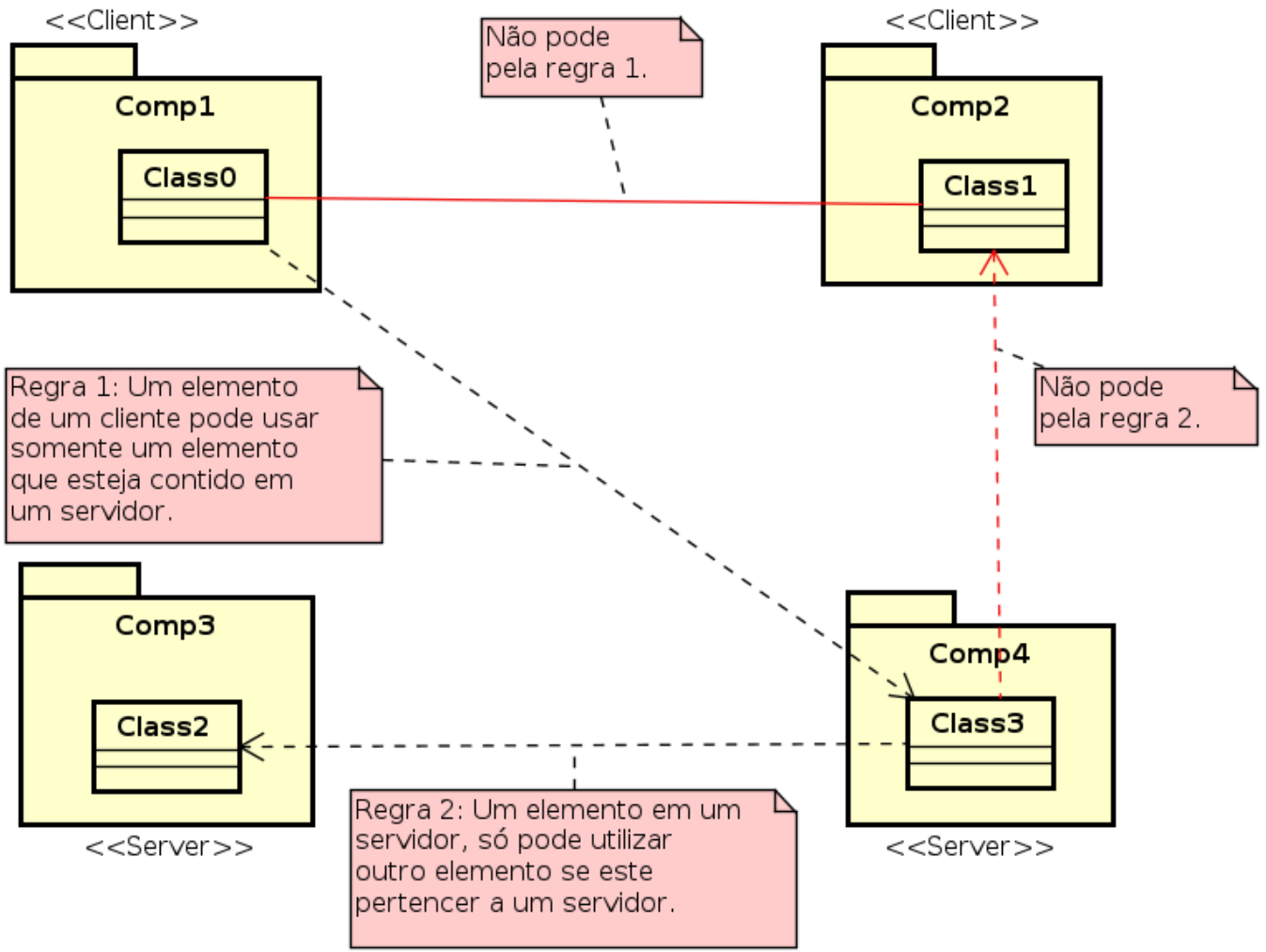
Estas métricas são encapsuladas em uma função objetivo chamada *Modularization Quality* (MQ);

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

Estilo Arquitetural Em Camada



Estilo Arquitetural Cliente/Servidor

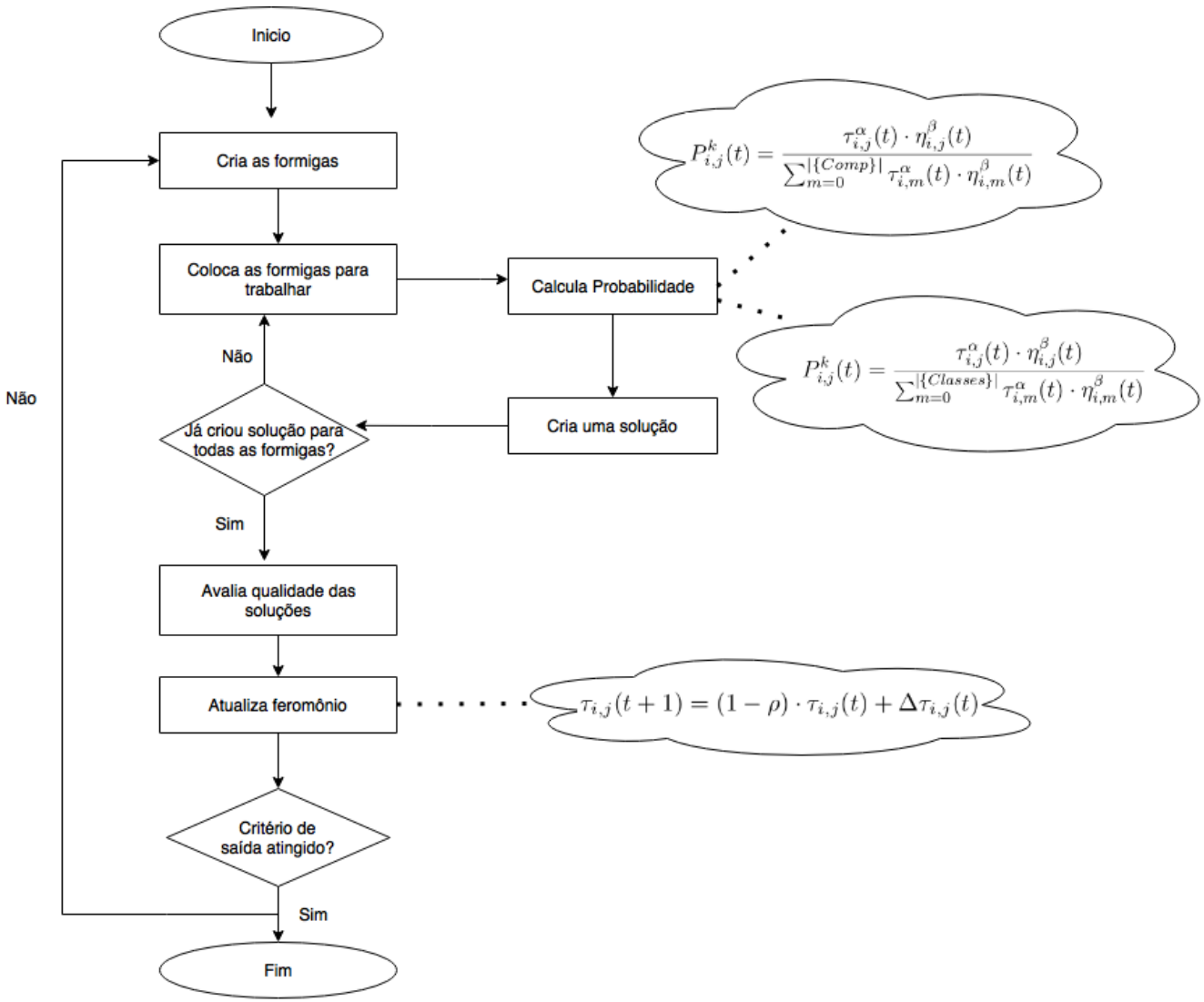


Estilo Arquitetural

- A métrica para verificação de estilo arquitetura é abordada como penalizações.
- Aplica-se penalizações a relacionamentos que infringem a regra do estilo arquitetura.

$$penalidade_{i,j} = \begin{cases} 1 - \frac{total_i + total_j}{totalGeralDeQuebrasDaRegra} & \text{se há estilo arquitetural} \\ 1 & \text{se não há estilo arquitetural} \end{cases} \quad (2)$$

ACO para Otimização Arquitetural



Calcula Penalidade

```
1:  function CALCULAPENALIDADE
2:      for i do 1 qtd_classes
3:          for j do 1 qtd_classes
4:              if estilo == LAYER then
5:                  verificaPenalidadeEstiloCamada(i,j, layer_i, layer_j)
6:              else if estilo == CLIENT/SERVER then
7:                  verificaPenalidadeEstiloClienteServidor(i,j, tipo_i, tipo_j)
8:              end if
9:          end for
10:     end for
11: end function
```

Experimento

- Versões 1.1.0 e 1.3.0 do software *Apache Ant*;
- Realizado 4 experimentos;
- Para cada teste feito, aplicamos várias combinações dos parâmetros de configuração (ρ, α, β, número de formigas e de iterações);
- Cada configuração foi executada 10 vezes.

Projeto	Classes	Componentes	Camadas	MQ
Apache Ant 1.1.0	97	7	6	0.55534
Apache Ant 1.3.0	274	26	20	0.51541

Resultados

Os resultados obtidos foram comparados com o modelo original da arquitetura

- MQ;
- Distribuição das classes nos componentes;
- Distribuição dos relacionamentos internos e externos.

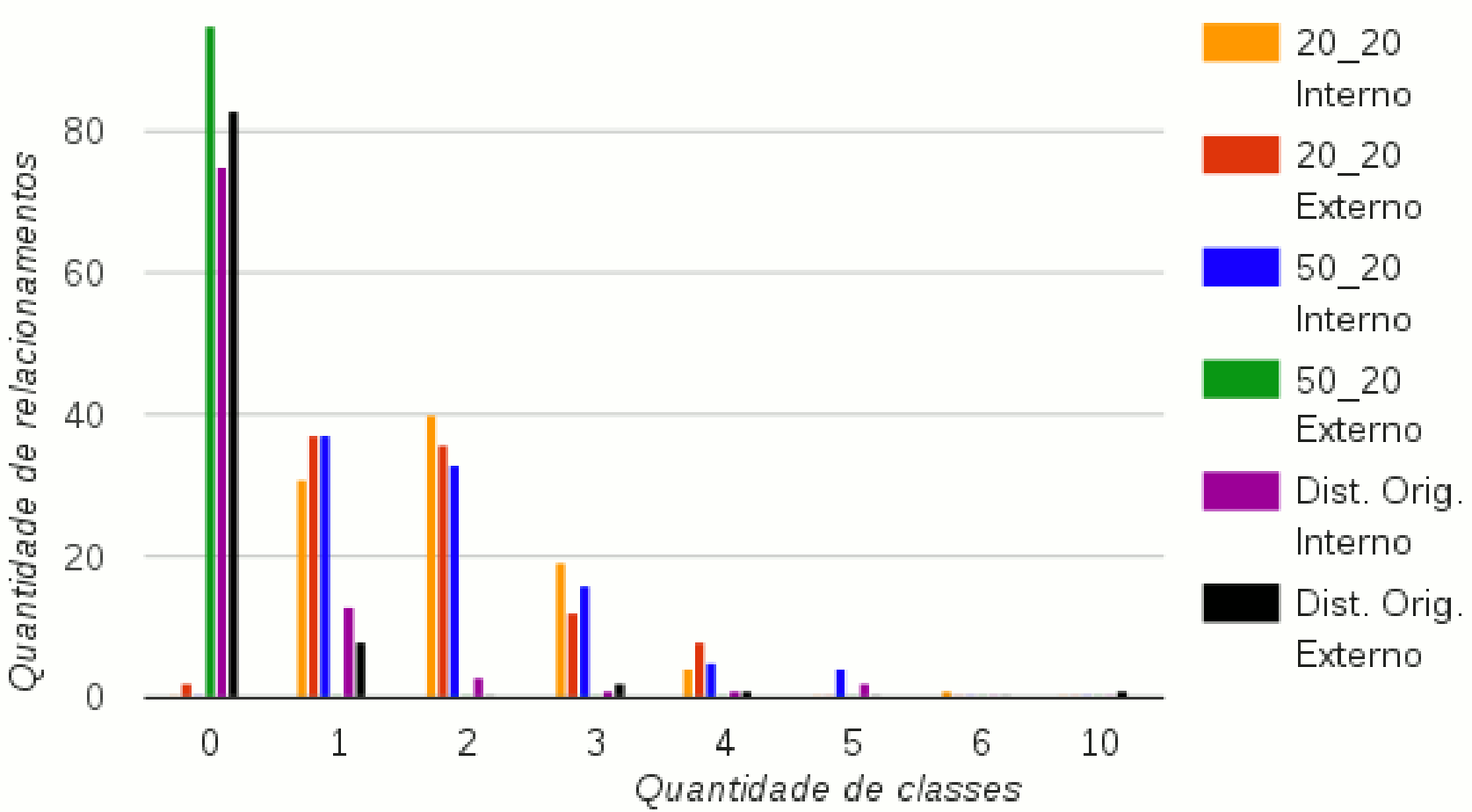
Resultados obtidos pelo ACO para versão 1.1.0 sem estilo arquitetural

Id	Iterações	Formigas	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

Resultados obtidos pelo ACO para versão 1.1.0 com estilo arquitetural

Id	Iterações	Formigas	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

Quantidade de classes e de relacionamentos internos e externos Apache Ant 1.1.0 com estilo arquitetural



Resultados obtidos para Apache 1.3.0

- Resultados obtidos pelo ACO para versão 1.3.0 sem estilo arquitetural

Id	Iterações	Formigas	ρ	α	β	MQ
1	20	500	0.7	0.9	0.0	0.60675
2	200	100	0.7	0.4	0.0	0.59478
3	100	50	0.0	0.4	0.0	0.56587

- Resultados obtidos pelo ACO para versão 1.3.0 com estilo arquitetural

Id	Iterações	Formigas	ρ	α	β	MQ	Rel. Penalizados
1	20	500	0.7	0.9	0.4	0.59489	0
2	100	50	0.7	0.4	0.4	0.58079	0

Tempo de Execução

Id	Versão	Estilo	Iterações	Formigas	ρ	α	β	Tempo De Execução
1	1.1.0	Não	100	15	0.2	0.4	0.2	00:04:32
2	1.1.0	Não	20	20	0.6	0.4	0.1	00:01:21
3	1.1.0	Sim	50	20	0.7	0.4	0.0	00:03:30
4	1.1.0	Sim	20	20	0.2	0.4	0.1	00:01:31
5	1.3.0	Não	20	500	0.7	0.9	0.0	13:58:58
6	1.3.0	Não	100	50	0.0	0.4	0.0	06:46:03
7	1.3.0	Sim	20	500	0.7	0.9	0.4	22:36:35
8	1.3.0	Sim	100	50	0.7	0.4	0.4	09:20:31

- ACO alcançou bons resultados considerando a métrica MQ;
- MQ torna a métrica de penalizações desnecessária;
- Tamanho da arquitetura, quantidade de iterações e formigas impactam no tempo de execução do algoritmo;

Ameaças a validade

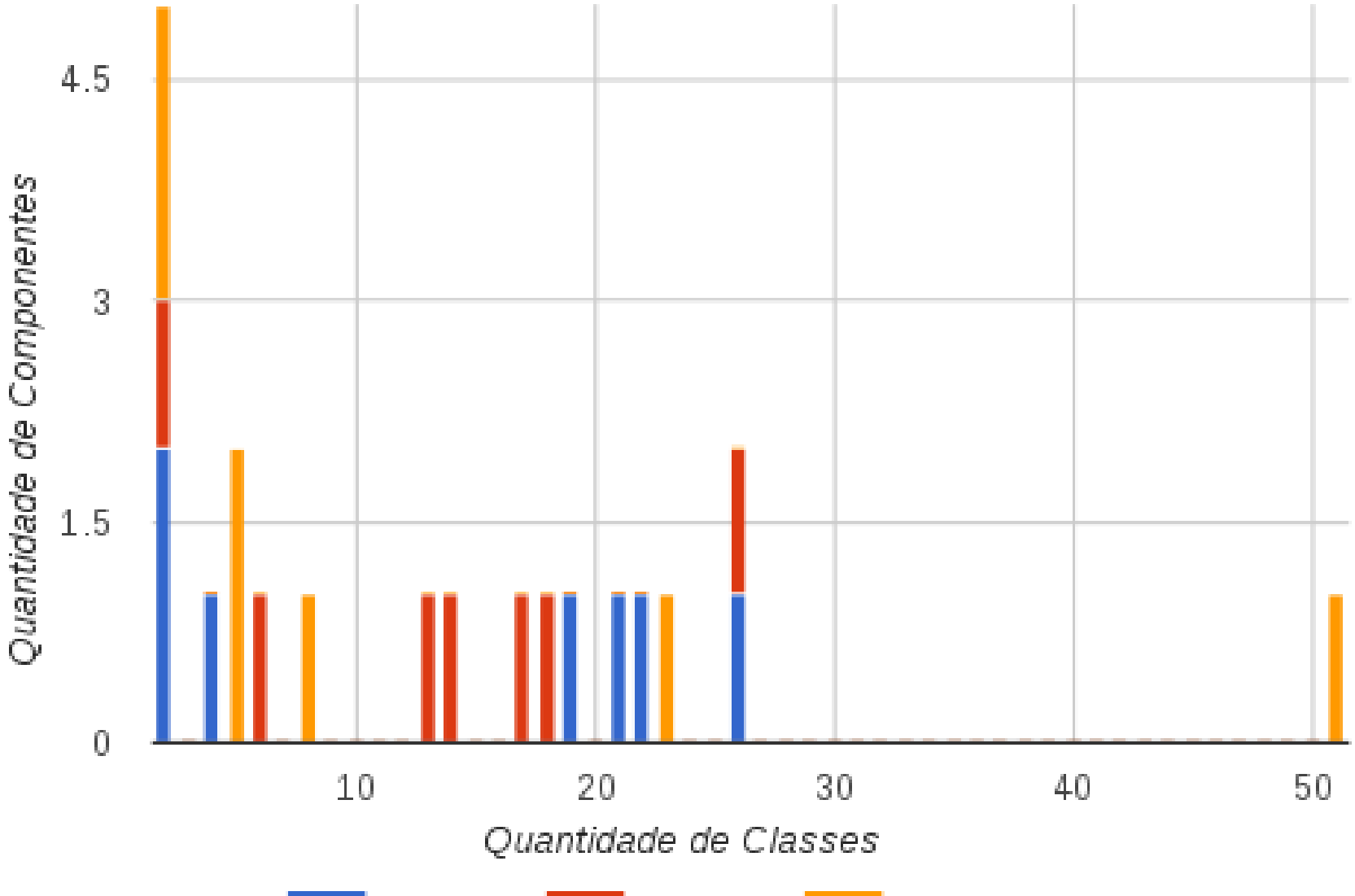
- Consideração apenas de elementos estáticos da arquitetura;
- Poucos casos de testes aplicados, necessidade de utilização do ACO em mais arquiteturas.

Trabalhos Futuros

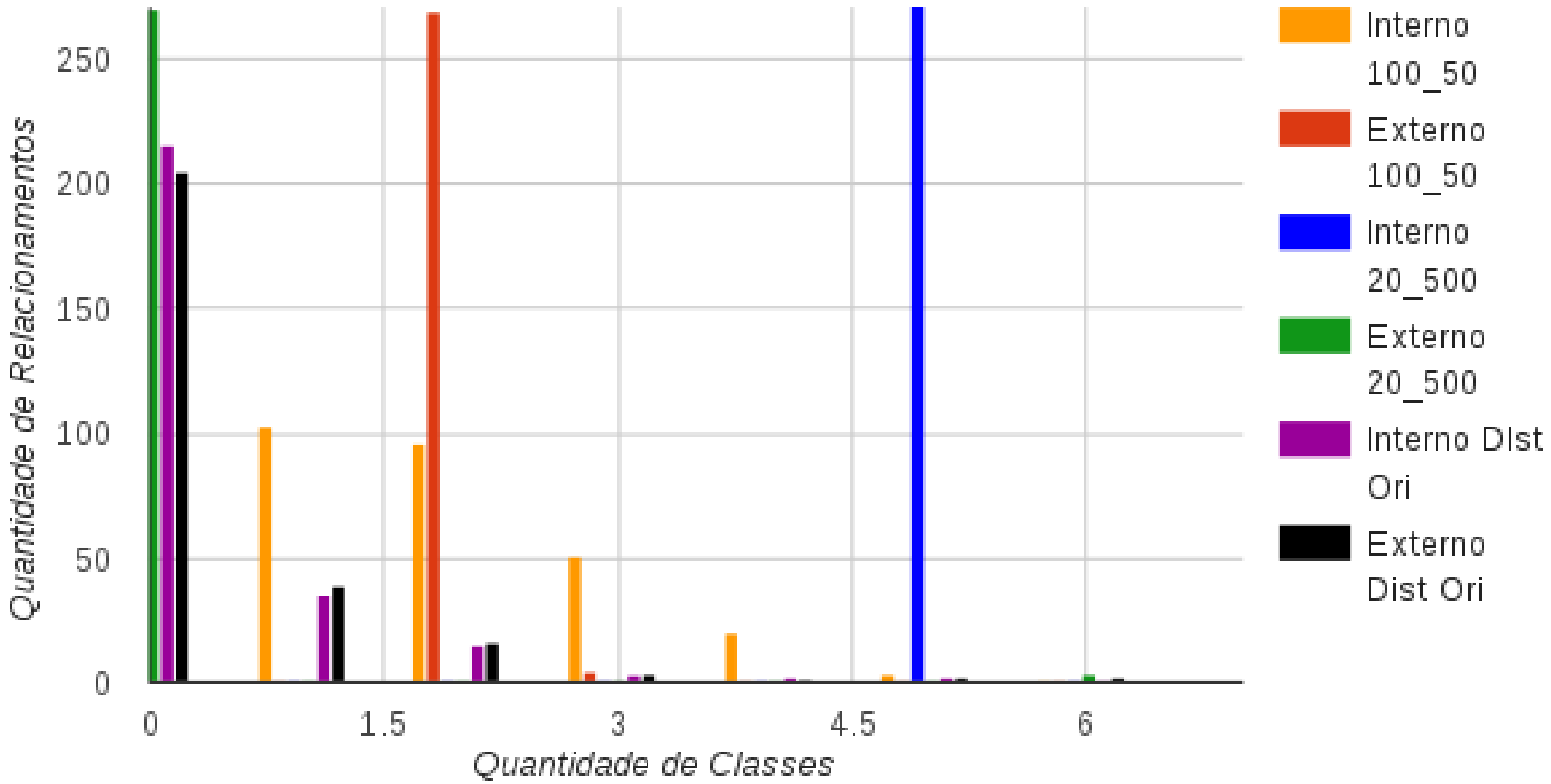
- Experimentos com outras metaheuristicas;
- Experimentos utilizando arquiteturas com estilo cliente/servidor;
- Implementação de outros estilos arquiteturais.
- Utilização de técnicas para melhoria do tempo de execução do algoritmo;
- Utilização de outras métricas de qualidade;

FIM

Distribuição das classes em seus respectivos componentes, versão 1.1.0 com estilo arquitetural



Quantidade de classes e relacionamentos internos e externos, versão 1.3.0 sem estilo arquitetural



Quantidade de classes distribuidas em cada componentes versão 1.3.0 sem estilo arquitetural