# 7
# Conclusão

O trabalho proposto gerou bons resultados no que se refere à detecção de informações extras de *core concepts*, componentização e métricas de acoplamento e tamanho. Para projetos não complexos, o *AnalisadorUML* apresentou taxa de acerto de 95% e, para projeto complexos, a taxa de acerto caiu para 92%. Desta maneira, do ponto de vista de processos de software, o objetivo foi alcançado.

O inconveniente é que a componentização de projetos complexos, como necessita de um processamento maior, demora um tempo considerável. Foram necessárias duas horas de processamento para componentização do projeto *QaC* (projeto de maior complexidade do estudo). Ou seja, se o usuário modificar o número de grupos do projeto, serão necessárias por volta de outras duas horas, fato que atrasa a análise e pode impactar no desenvolvimento do projeto.

Outro inconveniente reside no leitor do XMI. Por mais que tenha sido desenvolvido um *parser* genérico, o usuário deverá construir um novo arquivo de transformação XMI-Metamodelo para cada ferramenta de edição de diagramas de classe que represente um metamodelo UML diferente.

## 7.1.
## Trabalhos Futuros

O trabalho futuro mais imediato, de alta relevância e longo prazo seria melhorar a componentização dos grupos. Só seriam expostas nas classes de interface do componente, as funções e propriedades que alguma classe de fora do componente utiliza.

Para desenvolver este tipo de verificação é necessário, além do diagrama de classes, o diagrama de sequência, que explica melhor como as funções são utilizadas entre as classes ao longo do tempo.

Outro trabalho futuro de curto prazo seria detectar tabelas mais importantes, grupos de tabela e métricas de tamanho e acoplamento para o Modelo Entidade-Relacionamento (MER). Os sete relacionamentos entre classes UML seriam substituídos por relacionamentos $1 \rightarrow n$ , $n \rightarrow 1$ , $n \rightarrow n$ , $1 \rightarrow 1$ . A

componentização e a detecção dos *core concepts* no MER seriam interessantes para validar o modelo, construir esquemas e desenvolver índices em tabelas altamente relacionadas. Uma facilidade estaria na engenharia reversa do modelo, visto que existem várias ferramentas que realizam este trabalho. Além disso, alguns Sistemas Gerenciadores de Banco de Dados dispõem consultas sobre meta-informações de tabelas e restrições (relacionamentos e chaves).

Por fim, outro trabalho relacionado, agora de médio prazo, seria a divisão de diagramas de classes em subdiagramas de classe através da leitura do XMI, transformação no grafo orientado de classes e obtenção dos pontos de corte do grafo. Estes pontos de corte são extraídos por algoritmo e, significam nós responsáveis por juntar diferentes regiões do diagrama de classes.

## 7.2. Resultados Parciais

Alguns resultados deste trabalho necessitam de uma base de amostras ainda maior para se ter uma aproximação mais verdadeira do seu comportamento.

O relacionamento de dependência cujo peso da aresta no grafo orientado de classes foi vinte e cinco pode ser melhor observado. Teoricamente, a dependência pode ser interpretada de diversas maneiras pelos arquitetos de software.

Como a injeção de dependência não costuma ser algo usual dentro do software, a base amostral só continha quatro instâncias deste tipo de relacionamento nos modelos de diagramas de classe gerados, um valor quase desprezível. O valor esperado para a dependência era para ser o menor entre os tipos de relacionamento UML. Para melhor estudar a dependência seria necessário incrementar a base amostral para este relacionamento.

Outro resultado que deveria ser melhorado era a estimativa de custo do projeto. Verificou-se com base nos projetos do trabalho que o custo do projeto é proporcional a complexidade do projeto, porém existem outras variáveis em questão. Uma destas variáveis é o grau de coesão do projeto e a ferramenta *AnalisadorUML* já possui este valor. Um trabalho futuro poderia observar uma métrica que combinasse estas e outras variáveis como número de métodos totais (como estimativa da complexidade de negócio) para convergir a um valor aproximado do custo.

## 7.3. Resultados Conquistados

Apesar da estipulação dos pesos dos relacionamentos ter sido feita com base em experimentação, o resultado das métricas de complexidade, coesão, tamanho e acoplamento funcionam para diagramas que não fizeram parte deste trabalho com bom grau de acerto. O único relacionamento que fica de fora disto é a dependência.

Do ponto de vista da engenharia de software, este trabalho demonstrou bons resultados quando existia a possibilidade de ser realizada uma engenharia reversa para gerar o modelo de acordo com o código existente. Este ponto foi debatido no uso de *sprints* de refatoração ou de correção de erros encontrados em metodologias ágeis como o *Scrum*.

Do ponto de vista de arquitetura de sistema, a visualização de classes desproporcionais e de má formação de componentes é observada de forma direta na ferramenta. A geração de classes com complexidade muito alta, a geração de componentes com muitas classes e de componentes pouco coesos realmente indicavam o uso de uma modelagem de software errônea. A detecção de falhas nos diagramas é observada e detectada com muita facilidade pelo *AnalisadorUML*.

Por último, se a estimativa de custo precisa ser melhor elaborada, este estudo conseguiu resultados na distribuição de trabalho entre desenvolvedores de acordo com os componentes para montar grupos de desenvolvedores e de acordo com a complexidade dos componentes e classes para expôr a necessidade do nível de qualidade do desenvolvedor e dos testes unitários e de contexto a serem realizados no software em questão.

A ferramenta *AnalisadorUML* está sendo utilizada em produção pelos laboratórios *ICA* e *LIRA* da PUC-Rio e pela *software house Druid Internet System* com bom *feedback* por parte de seus gestores.