



- ► Introdução/Contextualização
- Funcionalidades
- Implementação
- App Demo
- Conclusões/Trabalhos Futuros

Introdução Contextualizaçã o Métricas → Compreender questões pertinentes sobre o código desenvolvido.

Smells → Dificultam a compreensão Refactoring → Resolve problemas de compreensibilidade Recurso a ANTLR (Árvore sintática) e TOM (para travessia da mesma)

Funcionalidades

- Medição de métricas sintáticas (Linhas de código,...)
- Medição de métricas de complexidade (Halstead e McCabe)
- ▶ Deteção de smells (Negação de Condições, Excesso de Linhas...)
- Refactoring (Troca de Instruções, Eliminação de variáveis)
- GUI baseada em Swing



- Extração de métricas segue uma estratégia global que preenche as estruturas de dados(Programa e Função)
- Smells partem da comparação dos valores obtidos com valores de referência
- Valores de referência : Valoração pesada (Repositório e Comunidade)
- Refactoring segue uma estratégia por medida implementada.

```
While(_,_,_,_,_) -> {
   auxFunc.incLines(1);
   auxFunc.incWhiles();
   auxFunc.adicionaOperador("While");
   auxFunc.adicionaOperador(")");
   auxFunc.adicionaOperador("(");
For(_,_,_,_,_,_, -> {
   auxFunc.incLines(1);
   auxFunc.incFors();
   auxFunc.adicionaOperador("For");
   auxFunc.adicionaOperador(")");
   auxFunc.adicionaOperador("(");
            private static int contaArgumentos(Argumentos args) {
                %match(args) {
                    ListaArgumentos(argl, argsTail*) -> {
                        return `contaArgumentos(arg1) + `contaArgumentos(argsTail*);
```

Imp.

Extração

Métricas

```
Argumento( , , ,id, ) -> {
    auxFunc.adicionaOperando('id);
```

return 0;

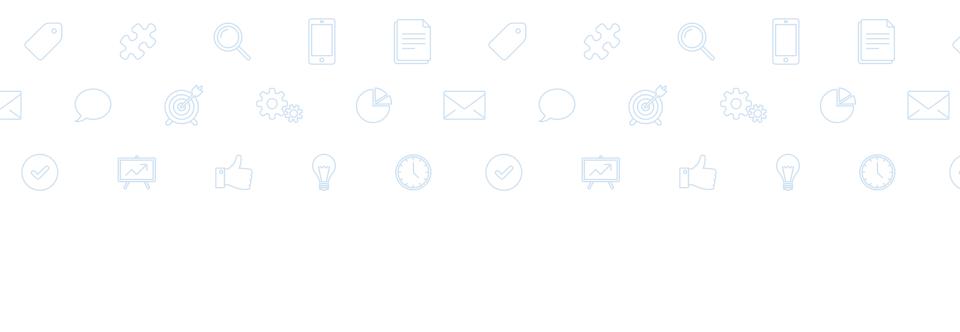
return 1:

auxFunc.incLocalVars('id);

visit Instrucao { $If(c1,c2,c3,Nao(e),c4,c5,then,els) \rightarrow {$ if('els != 'Exp(Empty())) { return `If(c1,c2,c3,e,c4,c5,els,then);} else return `If(c1,c2,c3,Nao(e),c4,c5,then,els); Imp. Refactoring public static Declaracoes testeRemoveVars(List<String> unusedVars, Declaracoes decls) { %match(decls) { ListaDecl(decl, declss*) -> { %match(decl){ d@Decl(id, , , ,) -> { if (unusedVars.contains('id)){ return testeRemoveVars(unusedVars, `declss); else{ return `ListaDecl(d, testeRemoveVars(unusedVars, declss)); return decls:

%strategy refactCondNeg() extends Identity(){

Conclusões e Trabalhos Futuros Utilização de Tom permitiu a diminuição de esforço programático, em oposição a escrita explícita na gramática ANTLR GUI permite uma fácil e intuitiva visualização dos dados Aumentar o catálogo de métricas, smells e refactoring Rever a modularização do programa (Estratégias)



App Demo

