

Mark Alan Junho Song

Mecanização do Processo de Projeto e
Implementação de Classes em Ambientes Orientados
por Objetos

Mecanização do Processo de Projeto e Implementação de Classes
em Ambientes Orientados por Objetos

MARK ALAN JUNHO SONG

Dissertação apresentada ao Departamento
de Ciência da Computação do Instituto de
Ciências Exatas da Universidade Federal de
Minas Gerais, como requisito parcial para a
obtenção do grau de Mestre em Ciência da
Computação.

Prof. Dr. João Eduardo de Almeida Dantas - Orientador
DCC - ICEx - UFMG

Marina S. P. R. P. - Coorientadora
Prof. Dr. Maria Augusta da Silva Bragança
DCC - ICEx - UFMG

Belo Horizonte

20 de abril de 1996

Prof. Dr. João Eduardo de Almeida Dantas
DCC - ICEx - UFMG

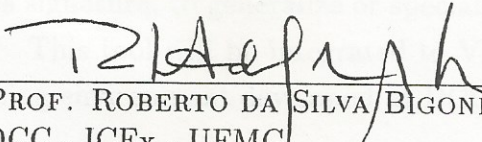
Belo Horizonte, 10 de abril de 1996

FOLHA DE APROVAÇÃO

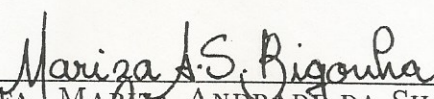
Mecanização do Processo de Projeto e Implementação de Classes em Ambientes Orientados por Objetos

MARK ALAN JUNHO SONG

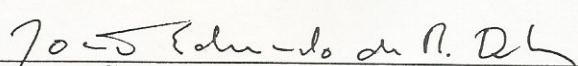
Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:



PROF. ROBERTO DA SILVA BIGONHA - Orientador
DCC - ICE_x - UFMG



PROFA. MARIZA ANDRADA DA SILVA BIGONHA
DCC - ICE_x - UFMG



PROF. JOÃO EDUARDO DE REZENDE DANTAS
DCC - ICE_x - UFMG

Belo Horizonte, 18 de abril de 1996.

Abstract

As program organization is based on types it manipulates and the style of object-oriented programming encourages code reuse in the development of programs, the knowledge and retrieval of existing types in the library are an important aspect of the design and implementation of new classes.

The purpose of this work is to design and develop a tool that support part of the design and implementation of classes process helping the recovery of software components from libraries.

The tool provides operations to manipulate classes, data member, function member, etc through a user friendly interface. It is possible to search and recover classes based on its signature, to generalize or specialize classes and visualize existing hierarchy.

This tool will be integrated to Visual Ita [RC95, MT95], an object-oriented programming environment developed at DCC-UFMG.

Agradecimento

Resumo

Como o estilo da programação orientada por objetos encoraja o desenvolvimento de programas a partir da reutilização de códigos, o reconhecimento dos tipos existentes na biblioteca constitui um aspecto essencial do projeto e implementação das novas classes: o princípio de organização de um programa é baseado nos tipos que este manipula.

Propõe-se projetar e implementar uma ferramenta que automatize parte do processo de projeto e implementação de classes auxiliando o programador durante a etapa de consulta à biblioteca. Oferece-se um conjunto de operações que permitem a manipulação de classes a partir da biblioteca ou da especificação fornecida como entrada.

Tais operações possibilitarão a manipulação das várias classes e métodos, a pesquisa e recuperação através de sua assinatura, a elaboração de hierarquias por especializações e/ou generalizações além de permitir a visualização das diversas hierarquias existentes.

Esta ferramenta funcionará de forma integrada ao sistema de programação orientado por objetos Visual Ita [RC95], baseado na linguagem Ita [MT95], desenvolvidos no DCC-UFMG.

Aos amigos, meus familiares e minha esposa Luciana em especial, pela companhia e apoio.

Agradecimento

Ao professor Roberto da Silva Bigonha, pelo constante incentivo, ajuda na solução de diversos problemas e orientação.

Aos amigos, meus familiares e minha esposa Loredana em especial, pela companhia e apoio.

Sumário

1	Introdução	1
1.1	Definição do Problema	1
1.2	Objetivo da Dissertação	3
1.3	Organização da Dissertação	4
2	Projeto Orientado por Objetos	5
2.1	Técnicas de Projeto	5
2.1.1	Determinando os Objetos	5
2.1.2	Definindo as Classes	6
2.1.3	Identificando os Relacionamentos entre Classes	7
2.1.4	Definindo as Hierarquias	8
2.1.5	Conclusão	9
2.2	Ferramentas	9
3	Ferramenta Proposta	12
3.1	O Menu Principal	13
3.1.1	O Item File	14
3.1.2	O Item View	16
3.1.3	O Item Window	18
3.1.4	O Item Help	20
3.2	A Barra de Ferramentas	20
3.2.1	Controles de Documentos	20
3.2.2	Controles de Classes	20
3.2.3	Generalizações	24
3.2.4	Controles de Manipulação de Membros	25
3.2.5	Controles de Compilação	28
3.2.6	Edição de Classes	29
3.3	Acesso à Biblioteca de Classes	31
3.3.1	Visões da Hierarquia	31
3.3.2	Recuperação de Componentes	33
3.4	Considerações Finais	34
4	Um Exemplo de Aplicação	35
4.1	Especificando o Problema	35
4.1.1	Determinando os Objetos/Classes da Aplicação	35
4.1.2	Determinando os Membros das Classes	35
4.1.3	Revendo a Especificação	37

4.2	Coleções	38
4.3	Implementações	40
4.3.1	Controlando Definições	40
5	Projeto da Ferramenta	46
5.1	Protótipos de Membros	46
5.2	Protótipos de Classes	47
5.3	Assinaturas de Membros	48
5.4	Assinaturas de Classes	48
5.5	Equivalência de Classes	49
5.6	Propriedades	50
5.7	Recuperação de Componentes	51
5.7.1	Recuperando Classes pela Assinatura	52
5.7.2	Recuperando Componentes por Propriedades	53
5.8	Generalizações	54
5.9	Geração de Protótipos de Módulos	54
5.10	Um Analisador para Declarações de Ita	56
5.11	Compilação	57
5.12	Implementação Dependente do Compilador	57
5.13	Integração com o Ambiente	58
5.14	Considerações Finais	59
6	Conclusões	60
	Bibliografia	62
A	Uma Gramática para Declarações de Ita	64
A.1	A Gramática	64
A.2	O Analisador	66
B	Implementação Dependente do Compilador	74
B.1	Estruturas de dados	74
B.2	Operações	76
B.3	Implementações	77