



Isela Macía Bertrán

**Avaliação da Qualidade de Software
com Base em Modelos UML**

Dissertação de Mestrado

Dissertação apresentada como requisito parcial
para a obtenção do título de Mestre pelo Programa
de Pós-graduação em Informática da PUC-Rio.

Orientadores: Prof. Arndt von Staa
Cláudio Nogueira Sant'Anna

Rio de Janeiro
Março de 2009



Isela Macía Bertrán

**Avaliação da Qualidade de Software com Base
em Modelos UML**

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Arndt von Staa

Orientador

Departamento de Informática - PUC-Rio

Prof. Carlos José Pereira de Lucena

Departamento de Informática - PUC-Rio

Prof.^a Simone Diniz Junqueira Barbosa

Departamento de Informática - PUC-Rio

Prof. José Eugenio Leal

Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 25 de Março de 2009

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, da autora e do orientador.

Isela Macía Bertrán

Graduou-se em Ciência da Computação na Universidade de Havana (UH) em 2006. Atuou como desenvolvedor de software até 2007.

Ficha Catalográfica

Macía Bertrán, Isela

Avaliação da qualidade de software com base em modelos UML / Isela Macía Bertrán ; orientadores: Arndt Von Staa, Cláudio Nogueira Sant'Anna. – 2009.

131 f. : il. ; 30 cm

Dissertação (Mestrado em Informática)– Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.

Inclui bibliografia

1. Informática – Teses. 2. Design de software orientado a objetos. 3. Diagramas de classe. 4. Métricas de software. 5. Estratégias de detecção de bad smells. 6. Modelos da qualidade. 7. Engenharia de software experimental. I. Staa, Arndt von. II. Sant'Anna, Cláudio Nogueira. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Aos meus pais

Agradecimentos

Meus agradecimentos mais especiais vão para as pessoas que julgo mais responsáveis por tudo que alcancei até aqui: meus maravilhosos pais. Muito obrigada pelo amor, confiança, educação e uma vida inteira de dedicação que, sem dúvida, tem sido essenciais na minha formação e luta pelos meus objetivos. Ao resto de minha família, pelo carinho, apoio e momentos de alegria sempre importantes, ainda mais na distância: à minha irmã, avô, tias, tios e primos.

Ilimitados agradecimentos a Julio pela maneira única que me ama e compreende dia a dia. Por sua paciência e incentivo em meus momentos de dúvidas e temores. Sou muito grata e orgulhosa de tê-lo ao meu lado.

Agradeço especialmente ao meu orientador, professor Arndt von Staa pois, sem seu apoio, este trabalho não teria sido possível. Ele me deu liberdade para escolher o tema deste trabalho, indicando-me sempre o caminho correto. Agradeço por ter-me mostrado o mundo das métricas e da qualidade de software.

Ao meu co-orientador, Cláudio Sant’Anna, pela orientação, dicas, parceria e dedicação. Você foi fundamental para o meu trabalho, sempre fornecendo comentários e sugestões para a melhoria da qualidade do conteúdo e da redação de minha dissertação.

Aos colegas do LES e do departamento de informática, pelos momentos compartilhados e aprendizado dividido. Agradeço especialmente a Andrew por ter-me facilitado parte dos sistemas analisados nesta dissertação e a Camila por sua ajuda na confecção de minhas primeiras apresentações.

Aos amigos Francisco, Frank, Ilaria, Evelin e Dunieskys pelo companheirismo e momentos que passamos juntos.

A Merlin, Yuanersi, Yahima, Ivan, grandes amigos de longa data, e cuja amizade sempre se manteve forte apesar da distância.

Agradeço, finalmente, a CAPES pelo apoio financeiro.

Resumo

Macía, Isela. **Avaliação da Qualidade de Software com Base em Modelos UML**. Rio de Janeiro, 2009. 131p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Um dos objetivos da engenharia de software é a construção de software com um nível de qualidade elevado com o menor custo e no menor tempo possível. Nesse contexto, muitas técnicas para o controle da qualidade de *design* de software têm sido definidas. Além disso, mecanismos baseados em métricas para a detecção de problemas também têm sido definidos. A maioria dessas técnicas e mecanismos foca a análise do código fonte. Porém, para reduzir retrabalho inútil, é importante utilizar técnicas de análise da qualidade capazes de detectar problemas de *design* já desde os modelos dos sistemas. Esta dissertação propõe: (i) um conjunto de estratégias de detecção para identificar, em modelos UML, problemas de *design* específicos e recorrentes na literatura: *Long Parameter List*, *God Class*, *Data Class*, *Shotgun Surgery*, *Misplaced Class* e *God Package*, e (ii) a utilização do modelo da qualidade QMOOD para avaliar *design* de software a partir de seus diagramas de classes. Para automatizar a aplicação destes mecanismos foi implementada uma ferramenta: a QCDDTool. Os mecanismos desenvolvidos foram avaliados no contexto de dois estudos experimentais. O primeiro estudo avaliou a acurácia, precisão e *recall* das estratégias de detecção propostas. Esse estudo mostrou os benefícios e desvantagens da aplicação, em modelos, das estratégias de detecção propostas. O segundo estudo avaliou a utilidade da aplicação do modelo da qualidade QMOOD em diagramas UML. Esse estudo mostrou que foi possível identificar, em diagramas de classes, variações das propriedades de *design*, e, conseqüentemente, dos atributos da qualidade nos sistemas analisados.

Palavras-chave

Design de software orientado a objetos, diagramas de classe, métricas de software, estratégias de detecção de *bad smells*, modelos da qualidade, engenharia de software experimental.

Abstract

Macía, Isela. **Evaluation of Software Quality Based on UML Models**. Rio de Janeiro, 2009. 131p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

One of the goals of software engineering is the development of high quality software at a small cost and in a short period of time. In this context, several techniques have been defined for controlling the quality of software designs. Furthermore, many metrics-based mechanisms have been defined for detecting software design flaws. Most of these mechanisms and techniques focus on analyzing the source code. However, in order to reduce unnecessary rework it is important to use quality analysis techniques that allow the detection of design flaws earlier in the development cycle. We believe that these techniques should analyze design flaws starting from software models. This dissertation proposes: (i) a set of strategies to detect, in UML models, specific and recurrent design problems: Long Parameter List, God Class, Data Class, Shotgun Surgery, Misplaced Class and God Package; (ii) and the use of QMOOD quality model to analyze class diagrams. To automate the application of these mechanisms we implemented a tool: the QCDDTool. The detection strategies and QMOOD model were evaluated in the context of two experimental studies. The first study analyzed the accuracy, precision and recall of the proposed detection strategies. The second study analyzed the utility of use QMOOD quality model in the class diagrams. The results of the first study have shown the benefits and drawbacks of the application in class diagrams of some of the proposed detection strategies. The second study shows that it was possible to identify, based on class diagrams, variations of the design properties and consequently, of the quality attributes in the analyzed systems.

Keywords

Object oriented software design, class diagrams, software metrics, bad smells detection strategies, quality models, experimental software engineering.

Sumário

1 Introdução	16
2 Avaliação Quantitativa do <i>Design</i> de Software	19
2.1. <i>Design</i> de Software Orientado a Objetos	19
2.2. Medição de Software	22
2.3. Modelos de Controle da Qualidade de <i>Design</i> OO	25
2.3.1. Modelos de Qualidade	25
2.3.1.1. Modelo de Qualidade QMOOD	26
2.3.2. Estratégias para a Detecção de Problemas de <i>Design</i>	31
3 Trabalhos Relacionados	35
3.1. Métricas para Diagramas de Classes	36
3.2. Detecção de Problemas de <i>Design</i> no Código Fonte	38
3.3. Análise de <i>Design</i> em Diagramas UML	39
4 Detecção de Problemas de <i>Design</i>	42
4.1. Características dos Valores Limites	43
4.2. Problemas de <i>Design</i> para Classes	43
4.2.1. <i>Data Class</i>	43
4.2.1.1. Estratégia de Detecção para Código Fonte	44
4.2.1.2. Estratégia de Detecção para Modelo	46
4.2.2. <i>God Class</i>	47
4.2.2.1. Estratégia para Código Fonte	48
4.2.2.2. Estratégia de Detecção para Modelo	49
4.2.3. <i>Shotgun Surgery</i>	51
4.2.3.1. Estratégia de Detecção para Código Fonte	51
4.2.3.2. Estratégia de Detecção para Modelo	52
4.3. Problemas de <i>Design</i> em Pacotes	54
4.3.1. <i>God Package</i>	54
4.3.1.1. Estratégia de Detecção para Código Fonte	55

4.3.1.2. Estratégia de Detecção para Modelos	57
4.3.2. <i>Misplaced Class</i>	58
4.3.2.1. Estratégia de Detecção para Código Fonte	58
4.3.2.2. Estratégia de Detecção para Modelo	60
4.4. Problemas de <i>Design</i> para Métodos	61
4.4.1. <i>Long Parameter List</i>	61
 5 QCDDTool: Uma Ferramenta para Avaliar a Qualidade do <i>Design</i> em Modelos	 63
5.1. Visão Geral da Ferramenta	63
5.2. Processo de Funcionamento	64
5.3. Arquitetura e Implementação	65
5.3.1. Módulo de Importação de Diagramas de Classes	66
5.3.2. Módulo de Aplicação das Métricas	68
5.3.3. Aplicação das Estratégias de Detecção	69
5.3.4. Aplicação dos Modelos da Qualidade	73
5.3.5. GUI	75
5.3.6. Exportação dos Resultados	77
5.4. Pontos Fixos e Pontos de Extensão	78
5.5. Instanciação	79
5.5.1. Importação dos Diagramas	80
5.5.2. Definição de Métricas	81
5.5.3. Definição de Estratégias de Detecção	82
5.5.4. Definição de Modelos da Qualidade	83
5.5.5. Exportação dos Resultados da Análise	84
 6 Estudos Experimentais	 85
6.1. Avaliação das Estratégias de Detecção	86
6.1.1. Sistemas Envolvidos	86
6.1.2. Estrutura do Estudo	88
6.1.2.1. Definição do Estudo	88
6.1.2.2. Planejamento do Estudo	88
6.1.2.3. Operação	90
6.1.3. Resultados	91

6.1.3.1. Resultados para a Estratégia de Detecção <i>Data Class</i>	92
6.1.3.2. Resultados para a Estratégia de Detecção <i>God Class</i>	94
6.1.3.3. Resultados para a Estratégia de Detecção <i>Shotgun Surgery</i>	96
6.1.3.4. Resultados para a Estratégia de Detecção <i>God Package</i>	98
6.1.3.5. Resultados para a Estratégia de Detecção <i>Misplaced Class</i>	100
6.1.3.6. Resultados para a Estratégia de Detecção <i>Long Parameter List</i>	101
6.1.4. Discussões sobre os Resultados	102
6.2. Avaliação do modelo da qualidade QMOOD	103
6.2.1. O Formato do Estudo	104
6.2.2. Resultados das Propriedades do <i>Design</i> e Atributos da Qualidade Avaliados	105
6.2.3. Conclusões do Estudo	108
6.3. Ameaças à Validade dos Estudos Realizados	109
 7 Conclusões e Trabalhos Futuros	 111
7.1. Contribuições	111
7.2. Trabalhos Futuros	113
 8 Referências	 115
 Anexo A - Document Type Definition do arquivo XML utilizado por QCDDTool	 119
 Anexo B - Documento gerado por QCDDTool para Armazenar os Resultados das Estratégias de Detecção	 121
 Anexo C - Resultados da Aplicação das Estratégias de Detecção no Primeiro Estudo Experimental	 122

Lista de figuras

Figura 1 – Níveis e relações no QMOOD.	27
Figura 2 – Mecanismo de filtragem e composição.	33
Figura 3 – Estratégia de detecção de <i>Data Class</i> para código.	45
Figura 4 – Estratégia de detecção de <i>Data Class</i> para modelo.	47
Figura 5 – Estratégia de detecção de <i>God Class</i> para código.	48
Figura 6 – Estratégia de detecção de <i>God Class</i> para modelo.	50
Figura 7 – Estratégia de detecção de <i>Shotgun Surgery</i> para código.	51
Figura 8 – Estratégia de detecção de <i>Shotgun Surgery</i> para modelo.	53
Figura 9 – Estratégia de detecção de <i>God Package</i> para código.	55
Figura 10 – Estratégia de detecção de <i>God Package</i> para modelo.	57
Figura 11 – Estratégia de detecção de <i>Misplaced Class</i> para código.	59
Figura 12 – Estratégia de detecção de <i>Misplaced Class</i> para modelo.	60
Figura 13 – Estratégia de detecção de <i>Long Parameter List</i> .	62
Figura 14 – Processo de funcionamento da ferramenta.	64
Figura 15 – Diagrama de módulos da ferramenta.	65
Figura 16 – Atividades da importação dos diagramas de classes.	66
Figura 17 – Módulo de importação dos diagramas.	68
Figura 18 – Representação das métricas.	69
Figura 19 – Representação das estratégias de detecção.	70
Figura 20 – Representação do mecanismo de filtragem.	72
Figura 21 – Estrutura dos operadores.	73
Figura 22 – Modelos da qualidade.	75
Figura 23 – QDCTool: (a) treeview com classes e pacotes; (b) tabela com resultados das métricas associadas a uma componente; (c) tabela com resultados das propriedades de <i>design</i> .	76
Figura 24 – Relações de dependência, herança e generalização entre as classes.	76
Figura 25 – Componentes de <i>design</i> classificados como problemáticos.	77
Figura 26 – Estrutura do módulo de exportação dos resultados.	78
Figura 27 – Instanciação do módulo de importação dos diagramas.	80

Figura 28 – Instanciação das métricas.	82
Figura 29 – Instanciação das estratégias de detecção.	83
Figura 30 – Instanciação do modelo QMOOD.	84
Figura 31 – Instanciação do módulo de exportação.	84

Lista de tabelas

Tabela 1 – Métricas do modelo QMOOD.	28
Tabela 2 – Relações entre as métricas e as propriedades do <i>design</i> .	30
Tabela 3 – Classificação dos filtros para dados.	32
Tabela 4 – Métricas para diagramas de classes.	36
Tabela 5 – Instanciação da ferramenta.	79
Tabela 6 – Características dos sistemas envolvidos no estudo.	88
Tabela 7 – Resultados da estratégia de detecção <i>Data Class</i> .	93
Tabela 8 – Resultados para a estratégia de detecção <i>God Class</i> .	95
Tabela 9 – Resultados para a estratégia de detecção <i>Shotgun Surgery</i> .	96
Tabela 10 – Resultados para a estratégia de detecção <i>God Package</i> .	99
Tabela 11 – Resultados para a estratégia de detecção <i>Misplaced Class</i> .	100
Tabela 12 – Resultados para a estratégia de detecção de <i>Long Parameter List</i> .	101
Tabela 13 – Valores das propriedades de <i>design</i> .	105
Tabela 14 – Valores normalizados das propriedades de <i>design</i> .	106
Tabela 15 – Resultados da quantificação dos atributos da qualidade.	108
Tabela 16 – Resultados da estratégia de detecção <i>Long Parameter List</i>	122
Tabela 17 – Resultados das estratégias de detecção <i>Data Class</i>	124
Tabela 18 – Resultados da estratégia de detecção <i>God Class</i> .	126
Tabela 19 – Resultados da estratégia de detecção <i>Shotgun Surgery</i> .	127
Tabela 20 – Resultados da estratégia de detecção <i>God Package</i> .	130
Tabela 21 – Resultados da estratégia de detecção <i>Misplaced Class</i> .	131

Lista de Acrônimos e Abreviações

ATFD - Access To Foreign Data.
CA - Changing Attributes.
CAM - Coesion Among Methods.
CC - Client Class.
CL - Class Locality.
CM - Changing Methods.
DD - Dependency Dispersion.
DTD - Document Type Definition.
GQM - Goal-Question-Metric.
GUI - Graphical User Interface.
MOF - MetaObject Facility.
NOAM - Number Of Access Methods.
NOCC - Number Of Client Class.
NOCP - Number Of Client Packages.
NOED - Number Of External Dependencies.
NOP - Number Of Parameters.
NOPA - Number Of Public Attributes.
OMG - Object Management Group.
OO - Orientado a Objetos.
PC - Package Cohesion.
PS - Package Size.
QMOOD - Quality Model Object Oriented Design.
SAX - Simple API XML.
TCC - Tight Class Cohesion.
UML - Unified Modeling Language.
WMC - Weighted Method per Class.
WOC - Weight Of Class.
XMI - XML Metadata Interchange.
XML - Extensible Markup Language.