QUALIDADE DE UMA FERRAMENTA DE APOIO E SIMULAÇÃO DE COMPOSTOS FERROMAGNÉTICOS

Ana Cláudia Martins de Souza

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharel em Ciência da Computação

Orientadora: Fernanda Claudia Alves Campos
Co-orientador: Marcelo Bernardes Vieira

Juiz de Fora, MG
Julho de 2009

QUALIDADE DE UMA FERRAMENTA DE APOIO E SIMULAÇÃO DE COMPOSTOS FERROMAGNÉTICOS

Ana Cláudia Martins de Souza

MONOGRAFIA SUBMETIDADA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada pela banca constituída pelos seguintes membros:

_____

Fernanda Claudia Alves Campos – orientadora
DSc. em Eng. de Sistemas e Computação, UFRJ/Brasil, 1999

_____

Marcelo Bernardes Vieira – co-orientador
DSc. em Ciência da Computação, ENSEA-UCP/France, 2002

_____

Marcelo Lobosco
DSc. em Eng. de Sistemas e Computação, UFRJ/Brasil, 2005

_____

Sócrates de Oliveira Dantas
DSc. em Ciências, UNICAMP/Brasil, 1994

Juiz de Fora, MG
Julho de 2009

Agradecimentos

A minha família por toda paciência e compreensão nos momentos em que não pude estar presente. Agradeço a eles também por sempre estarem ao meu lado me ajudando no que fosse possível e pelas várias orações feitas por mim.

A família do Roberto que me acolheu por todo este período me fazendo sentir um novo membro na família.

Ao Roberto pela colaboração na construção deste trabalho, pela paciência nos momentos de tensão, pelo companheirismo em me ajudar em tudo que eu precisasse.

Aos companheiros de trabalho pela compreensão nos momentos que precisei me ausentar.

A todos os professores envolvidos por toda dedicação e atenção a este trabalho.

Principalmente ao professor Marcelo Bernardes Vieira que me auxiliou durante a maior parte do desenvolvimento deste trabalho fazendo correções e sugestões que tornou possível a apresentação deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta parte do processo de construção de um compilador e as características de qualidade que o produto de software deverá ter, levantadas a partir da visão dos usuários. O levantamento de atributos do software teve como base o modelo de qualidades descrito pela Norma 9126-1, no qual foram focadas as características de funcionalidade e usabilidade. Foram utilizadas duas métricas para identificação dos atributos mais importantes do software, que serão utilizados para uma avaliação futura do mesmo. Conseguiu-se com este trabalho identificar onze características.

Palavras-Chave: qualidade de produto, modelo de qualidade, software para simulação científica.

Capítulo 1

Introdução

A Física da matéria condensada em sua abordagem teórica necessita de simuladores para construção e análise de objetos eletromagnéticos. Encontrar simuladores que transmitam este resultado por um baixo custo se torna um problema que, muitas vezes, impede ou atrasa trabalhos que podem resultar em novas descobertas.

1.1 Visão Geral

O compilador que será apresentado teve como base a linguagem sugerida em [Ferreira, 2009]. Todas as suas etapas foram construídas cautelosamente, visando principalmente os usuários finais da ferramenta, isto é, físicos e desenvolvedores de softwares que possam utilizar a ferramenta para construir outras ou apenas aprimorá-la.

Sabendo que pessoas com conhecimento restrito em programação utilizariam o software, as estruturas foram avaliadas e adequadas de forma a ser o mais natural possível. A proximidade com linguagens de programação já existentes foi uma estratégia utilizada. Nesse contexto, a linguagem C foi escolhida como principal exemplo para estruturas por ser a linguagem mais difundida no meio acadêmico.

Além disto, para facilitar a construção de qualquer expressão, foram inseridas à gramática todas as funções matemáticas contidas em uma biblioteca da linguagem de programação C denominada "*cmath*".

Antes de começar cada etapa do compilador, todos os dados necessários a ela foram levantados, estudados e estruturados de maneira que a etapa fosse construída corretamente e seus resultados fossem rápidos e concisos. Vários testes foram realizados e uma atenção maior foi dada às mensagens exibidas ao usuário.

Foi realizada uma entrevista com alguns atores envolvidos no projeto para fazer um levantamento de atributos de qualidade importantes ao software. Para realização desta entrevista os atores tiveram um breve esclarecimento sobre as definições das características funcionalidade e usabilidade, definidas pelo modelo de qualidade da norma ISO/IEC 9126-1 [ABNT, 2003].

Com os atributos citados pelos atores, uma ficha de validação foi construída. Esta ficha foi passada por uma mostra de seis atores envolvidos para que eles classificassem cada atributo como “Muito importante”, “Importante” e “Sem importância”. Após obter todas as fichas de validação preenchidas, todos as classificações foram contabilizadas.

Afim de selecionar apenas atributos importantes, duas métricas de classificação de dados foram selecionadas. Com as classificações de características prontas, foi realizada a interseção do desses conjuntos de forma a manter apenar atributos que sempre eram considerados importantes.

Obter uma ferramenta que atenda características dentro da usabilidade é algo realmente importante, principalmente considerando o fato de que,a Ferramenta de Apoio e Simulação de Compostos Ferromagnéticos será utilizada por físicos e por cientistas da computação. Todas as estruturas internas e externas da ferramenta foram avaliadas e ajustadas cautelosamente de forma que a ficasse realmente simples de construir e compreender as estruturas.

1.2 Objetivos

Esta monografia visa descrever o desenvolvimento e a definição de características de qualidade de um produto de software que tem o intuito de auxiliar os físicos na composição de objetos para pesquisa em Física da matéria condensada. Esse trabalho foi baseado no modelo de qualidade para características externas e internas definidos pela norma ISO/IEC 9126 [ABNT, 2003], e foram selecionadas as características relacionadas à funcionalidade e usabilidade.

Como objetivos secundários, pretende-se:

- documentar o desenvolvimento de uma ferramenta que atenda a necessidade de gerar compostos de objetos com base principal na construção de um compilador juntamente com o trabalho realizado em [Ferreira, 2009];
- validar com os futuros usuários e com as partes interessadas do software um conjunto de atributos de qualidade que o produto final deve ter. Para isto, serão realizadas entrevistas com os atores envolvidos no projeto com o objetivo de explicitar e validar essas características.

1.3 Estrutura geral da monografia

No capítulo 2 serão descritos detalhes do produto de software desenvolvido. O primeiro item será a linguagem utilizada, a qual foi elaborada para parecer com uma linguagem de programação comum, se tornando uma linguagem de aprendizado rápido, mesmo considerando usuários que não estão no universo da programação.

Em seguida serão apresentados detalhes do compilador desenvolvido para a linguagem descrita anteriormente. As suas etapas de construção serão apresentadas nas seguintes partes: Analise Léxica, Analise Sintática e Analise Semântica. Será exposto também o tratamento de erros e a construção da Árvore de Sintaxe Abstrata.

O capítulo 3 contém um breve tópico sobre qualidade, no qual serão definidos os aspectos de qualidade do produto. Ainda neste capítulo será apresentado o modelo de qualidade definido na norma ISO/IEC 9126 [ABNT, 2003] com suas características e sub-características. Modelo este utilizado como referência para o processo de avaliação da qualidade de produto de software. Será dado um foco maior em funcionalidade e usabilidade, pois estas características serão avaliadas no produto de software citado.

No capítulo 4 será apresentado o modelo do processo de validação dos atributos de qualidade do software. Essa seção terá o perfil dos atores entrevistados e as características de qualidade destacadas por cada um deles. O resultado de todo o processo de validação é então apresentado, resultando num conjunto específico de características.

Por fim, uma conclusão sobre o trabalho será apresentada.

Capítulo 2

O Gerador de Estruturas Ferromagnéticas

2.1 Introdução

A área acadêmica necessita de vários instrumentos tecnológicos para facilitar pesquisas e auxiliar nas metodologias de ensino. Os métodos de ensino devem acompanhar o avanço das tecnologias do mundo, o que muitas vezes não acontece por falta de incentivo e patrocínio.

Na área da Física é muito importante o uso de simuladores que transformem teoria em prática. Considerando o uso de objetos implícitos – compostos ferromagnéticos – temos que a demonstração da prática gera um entendimento mais aprofundado.

Foi construído um gerador de estruturas, no contexto do Núcleo do Simulador de Spins que está envolvido aos Projetos de Pesquisa dos professores Sócrates de Oliveira Dantas, Marcelo Bernardes Vieira e Marcelo Lobosco, por essa autora e por [Ferreira 2009], que visa auxiliar os professores e alunos na compreensão e nas pesquisas com compostos ferromagnéticos. A seguir estão descritos mais detalhes sobre este gerador de estruturas.

2.2 Produto

O produto gerador de estruturas foi desenvolvido com intuito de auxiliar os físicos na composição de objetos realmente interessantes para pesquisa em Física da matéria condensada. Essa composição de objetos é algo difícil de representar manualmente. O software facilita a criação e visualização dessas estruturas.

Para construção do gerador de estruturas foi utilizada a linguagem proposta por [Ferreira, 2009], que será detalhada a seguir.

2.2.1 Linguagem

Construir uma linguagem não é algo simples, pois deve levar em consideração todas as pessoas envolvidas no projeto.

Nesse projeto os primeiros a serem considerados envolvidos são os físicos, os quais são o foco do software desenvolvido. Portanto, a linguagem deve ser de fácil aprendizado. Mas, é necessário um bom conhecimento nas fórmulas matemáticas que geram determinadas estruturas.

O segundo grupo de envolvidos são os cientistas da computação, os quais continuarão este trabalho unindo este software a outros que serão desenvolvidos em sequência. Neste ponto, a linguagem construída teve como base a linguagem de programação C, mas com algumas particularidades.

Para facilitar a construção de formulas, todas as funções da biblioteca "*cmath*" da linguagem de programação C foram implementadas.

A gramática da linguagem está apresentada na figura 2.1 – Linguagem descrita por uma gramática LL(1).

Program	→	DefSolids | SpinPropertys | **lattice** "(" Exp "," Exp "," Exp "," Exp "," Exp "," Exp "," Exp "," Exp "," Exp ")" **begin** Stmt_List **end**
SpinPropertys	→	SpinProperty SpinPropertys | Epsilon
SpinProperty	→	**spin property** Type **id** "**=**" Value ";"
Type	→	**int** | **float** | **bool**
Value	→	**num** | **true** | **false**
DefSolids	→	DefSolid DefSolids | Epsilon
DefSolid	→	**solid id** Arguments Exp "**;**" | **composite id** Arguments **begin** Stmt_List **end**
Arguments	→	"(" Id_List ")" |

		Epsilon
Id_List	→	**id** Id_list_
Id_list_	→	"**,**" **id** Id_list_ |
		Epsilon
Exp	→	Exp1 Exp_
Exp_	→	"**||**" Exp1 Exp_ |
		Epsilon
Exp1	→	Exp2 Exp1_
Exp1_	→	"**&&**" Exp2 Exp1_ |
		Epsilon
Exp2	→	Exp3 Exp2_
Exp2_	→	"**==**" Exp3 Exp2_ |
		"**<>**" Exp3 Exp2_ |
		Epsilon
Exp3	→	Exp4 Exp3_
Exp3_	→	"**<**" Exp4 Exp3_ |
		"**<=**" Exp4 Exp3_ |
		"**>=**" Exp4 Exp3_ |
		"**>**" Exp4 Exp3_ |
		Epsilon
Exp4	→	Exp5 Exp4_
Exp4_	→	"**+**" Exp5 Exp4_ |
		"**-**" Exp5 Exp4_ |
		Epsilon
Exp5	→	Exp6 Exp5_
Exp5_	→	"*" Exp6 Exp5_ |
		"/" Exp6 Exp5_ |
		Epsilon
Exp6	→	**cos** "(" Exp ")" |
		sin "(" Exp ")" |
		tan "(" Exp ")" |
		acos "(" Exp ")" |
		asin "(" Exp ")" |
		atan "(" Exp ")" |
		atan2 "(" Exp "**,**" Exp ")" |
		cosh "(" Exp ")" |
		sinh "(" Exp ")" |
		tanh "(" Exp ")" |
		exp "(" Exp ")" |
		frexp "(" Exp "**,**" Exp ")" |
		ldexp "(" Exp "**,**" Exp ")" |
		log "(" Exp ")" |
		log10 "(" Exp ")" |
		modf "(" Exp "**,**" Exp ")" |
		pow "(" Exp "**,**" Exp ")" |
		sqrt "(" Exp ")" |
		ceil "(" Exp ")" |
		fabs "(" Exp ")" |
		floor "(" Exp ")" |
		fmod "(" Exp "**,**" Exp ")" |

```
                    Exp7

Exp7        →       union           "("  Exp  "," Exp ")" |
                    intersection    "("  Exp  "," Exp ")" |
                    difference      "("  Exp  "," Exp ")" |
                    not             "("  Exp  ")"  |
                    rotatex         "("  Exp  "," Exp ")" |
                    rotatey         "("  Exp  "," Exp ")" |
                    rotatez         "("  Exp  "," Exp ")" |
                    scale           "("  Exp  ","  Exp  ","  Exp  ","  Exp ")" |
                    shear           "("  Exp  ","  Exp  ","  Exp  ","  Exp ")" |
                    translate       "("  Exp  ","  Exp  ","  Exp  ","  Exp ")" |
                    Exp8

Exp8        →       "-"  Exp9  |
                    Exp9

Exp9        →       "(" Exp ")" |
                    Exp10

Exp10       →       num |
                    id  ExpProperty |
                    x |
                    y |
                    z

ExpProperty →       "(" Exp_list ")" |
                    Epsilon

Exp_list    →       Exp Exp_List_  |
                    Epsilon
Exp_list_   →       "," Exp Exp_List_|
                    Epsilon

Stmt_List   →       Statement   ";"  Stmt_List |
                    Epsilon

Statement   →       id  "=" Exp |
                    insert "(" Exp  PropertyList ")" |
                    while  "("  Exp  ")"
                     begin
                            Stmt_List
                     end

PropertyList →      "," PropertyAssign PropertyList |
                    Epsilon

PropertyAssign →    id "=" Value
```

Figura 2.1 – Linguagem descrita por uma gramática LL(1)

Com a linguagem definida por uma gramática pertencente à classe LL(1), o compilador pode ser construído. Um compilador é um programa que traduz um código fonte para uma linguagem de mais baixo nível.

A construção do estágio analítico (*front-end*) deste compilador teve três etapas bem definidas: analisador léxico, analisador sintático e analisador semântico. Os quais serão descritos a seguir.

2.2.2 Analisador Léxico

O analisador léxico é a primeiro estágio de um compilador. Ele é responsável por verificar se todas as palavras utilizadas como entrada são palavras aceitas pela linguagem e transformá-las em uma sequência de *tokens* que podem ser facilmente manipulados por um *parser*. Para isto, o usuário pode utilizar qualquer sentença definida como expressão regular na Tabela 1 – *Tokens* e expressões regulares.

Token	Expressões regulares	Atributo
SPIN	spin	
PROPERTY	property	
INT	int	
FLOAT	float	
BOOL	bool	
SOLID	solid	
COMPOSITE	composite	
BEGIN	begin	
END	end	
LATTICE	lattice	
WHILE	while	
INSERT	insert	
TRUE	true	
FALSE	false	
ASSING	=	
UNION	union	
INTERSECTION	intersection	
DIFFERENCE	difference	
NOT	not	
OPARENT	(	
CPARENT	)	
COLON	,	
SEMICOLON	;	
DOT	.	
OP_EQUAL	==	
OP_LESS	<	
OP_LESSEQUAL	<=	
OP_GREAT	>	
OP_GREATEQUAL	>=	
OP_DIFFERENT	<>	
OP_PLUS	+	
OP_MINUS	-	
OP_TIMES	*	
OP_DIVIDE	/	
OP_OR	||	
OP_AND	&&	

COS	cos	
SIN	sin	
TAN	tan	
ACOS	acos	
ASIN	asin	
ATAN	atan	
ATAN2	atan2	
COSH	cosh	
SINH	sinh	
TANH	tanh	
EXP	exp	
FREXP	frexp	
LDEXP	ldexp	
LOG	log	
LOG10	log10	
MODF	modf	
POW	pow	
SQRT	sqrt	
CEIL	ceil	
FABS	fabs	
FLOOR	floor	
FMOD	fmod	
ROTATEX	rotatex	
ROTATEY	rotatey	
ROTATEZ	rotatez	
SCALE	scale	
TRANSLATE	translate	
SHEAR	shear	
ID	letra(letraldígito)*	Atributo
NUM	dígito(dígito)* | dígito(dígito)*. (dígito)*	Atributo
X	x	
Y	y	
Z	z	
EOF	eof	

Tabela 2.1 – *Tokens* e expressões regulares

A análise léxica é uma forma de verificar se todos os caracteres de entrada pertencem ao alfabeto da linguagem definida. Caso algum caractere não pertença ao alfabeto, uma mensagem de erro deve ser exibida ao usuário.

2.2.3 Analisador Sintático

O analisador sintático tem como objetivo a verificação da estrutura sintática. Essa fase também é conhecida como *parsing*. Nesta análise, a estrutura de uma sequência de *tokens* é avaliada verificando se as estruturas gramaticais construídas estão de acordo com a gramática LL(1) apresentada na Figura 2.1.

Foi desenvolvido um analisador sintático descendente que é caracterizado por ser um algoritmo de análise sintática para um sub-conjunto de gramáticas livre de contexto. O "1" refere-se ao número de *tokens* antecipados para escolha da produção a ser expandida.

Ao longo da verificação sintática, uma Árvore de Sintaxe Abstrata é gerada para servir de interface com a parte sintética do compilador (*back-end*). A Tabela 2.2 mostra os nodos da árvore e suas possíveis formas.

	Nodo	**Formas possíveis**
1	PROGRAM	- (SOLID_LIST, SPIN_PROPERTYS, LATTICE, STATEMENT_LIST)
2	SOLID_LIST	- (DECL_SOLID, SOLID_LIST) - NULL
3	SPIN_PROPERTYS	- (SPIN_PROPERTY, SPIN_PROPERTYS) - NULL
4	LATTICE	- (EXP, EXP, EXP, EXP, EXP, EXP, EXP, EXP, EXP)
5	SPIN_PROPERTY	- (TYPE, ID, NUM) - (TYPE, ID, BOOL)
6	DECL_SOLID	- (ID, ARGUMENT_LIST, EXP) - (ID, ARGUMENT_LIST, STATEMENT_LIST)
7	ARGUMENT_LIST	- (ID, ARGUMENT_LIST) - NULL
8	STATEMENT_LIST	- (STATEMENT, STATEMENT_LIST) - NULL
9	STATEMENT	- (ASSIGN) - (INSERT) - (WHILE)
10	ASSIGN	- (ID, EXP)
11	INSERT	- (EXP, PROPERTY_LIST)
12	PROPERTY_LIST	- (PROPERTY_ASSIGN, PROPERTY_LIST) - NULL
13	PROPERTY_ASSIGN	- (ID, NUM) - (ID, BOOL)
14	WHILE	- (EXP, STATEMENT_LIST)
15	EXP	- (ID) - (NUM) - (x) - (y) - (z) - (OR_OP) - (AND_OP) - (RELATIONAL_OP) - (ADDITION_OP) - (MULTIPLICATION_OP) - (EQUAL_OP) - (UNARY_OP) - (FUNCTION_MATH) - (CSG_OP) - (NOT_OP) - (TRANSFORMATION_OP) - (OBJECT_CALL)
16	OR_OP	- (EXP, EXP)
17	AND_OP	- (EXP, EXP)

18	RELATIONAL_OP	- (OP_REL, EXP, EXP)
19	ADDITION_OP	- (OP_ADD, EXP, EXP)
20	MULTIPLICATION_OP	- (OP_MUL, EXP, EXP)
21	EQUAL_OP	- (OP_EQUAL, EXP, EXP)
22	UNARY_OP	- (EXP)
23	FUNCTION_MATH	- (FUNC_MATH, EXP) - (FUNC_MATH, EXP, EXP)
24	CSG_OP	- (CSG, EXP, EXP)
25	NOT_OP	- (EXP)
26	TRANSFORMATION_OP	- (TRANSFORMATION, EXP, EXP) - (TRANSFORMATION, EXP, EXP, EXP, EXP)
27	OBJECT_CALL	- (ID, EXPR_LIST)
28	EXPR_LIST	- (EXP, EXPR_LIST) - NULL

Tabela 2.2 – Árvore de sintaxe abstrata.

2.2.4 Analisador Semântico

O analisador semântico é responsável por avaliar se as estruturas de entrada estão semanticamente corretas. Em comparação com a análise sintática, que realiza uma análise livre de contexto, essa fase realiza uma verificação sensível ao contexto. Portanto, essa é a fase de verificação de todas as dependências de nomes e estruturas que não podem ser expressadas por uma linguagem livre de contexto.

Nesse trabalho, o analisador semântico foi implementado se utilizando o padrão de programação visitante (*visitor*).

Um visitante tem a intenção de representar uma operação a ser executada nos elementos de uma estrutura de objetos. Ele permite definir uma nova operação sem mudar as classes dos elementos os quais opera [Gamma, 2005]. Sendo assim, a operação a ser feita dependerá tanto do tipo de visitante quanto do tipo de elemento que o visitante visite.

A escolha por utilização deste padrão é pelo fato de se ter concentrado em um só local todos os métodos que irão visitar outras classes, o que torna fácil a manutenção e deixa o código mais organizado.

Capítulo 3

Qualidade de Software

3.1 Conceituação

Qualidade de software pode ser vista como um conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda as necessidades de seus usuários [Rocha, 2001]. Para a norma NBR ISO 8402, qualidade é a totalidade das características de uma entidade, que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas [NBR ISO 8402 in Rocha, 2001].

Para se obter a qualidade do produto é necessário avaliá-lo levantando os aspectos que são mais importantes para o software. O objetivo, ao se desenvolver um produto de software, não é alcançar a qualidade perfeita, mas sim a qualidade necessária e suficiente para o uso especificado, quando o produto for entregue e realmente utilizado pelos usuários [Rocha, 2001]. Qualidade custa caro e não é possível atingir todas as características de qualidade no mais alto grau [Boegh, 1993].

3.2 Qualidade do Produto

A qualidade de produto de software é baseada em normas que avaliam se o produto satisfaz o cliente e tem fácil manutenção. É o resultado direto das atividades realizadas no processo de desenvolvimento do software. Alguns exemplos de normas de qualidade de produto são as normas ISO/IEC 14598, 12119 e 9126 [Sodré, 2006].

A verdadeira melhoria da qualidade sempre se inicia, ao saber o que os clientes querem e, possivelmente, aquilo que precisam ou esperam. A principal responsabilidade do cliente é decidir sobre os requisitos da avaliação, pois estes oferecem o padrão para a medição da qualidade. Para tanto, é necessário que se faça uma grande interação com o usuário, uma vez que ele será o verdadeiro árbitro na avaliação da qualidade do produto [Campos, 2009].

A satisfação do usuário ou cliente está diretamente relacionada à qualidade e é percebida de formas diferentes. Qualidade de software é um conceito complexo, relativo

(depende sempre da perspectiva de quem está avaliando) e jamais se pode pensar em qualidade como sinônimo de perfeição [Lima, 2003].

A qualidade do software pode ser classificada em externa e interna. Tendo que, qualidade externa é visível aos usuários do software e qualidade interna é aquela pertinente aos desenvolvedores. Em geral, os usuários estão preocupados apenas com a qualidade externa, mas é a interna que nos leva a alcançar a qualidade externa.

A medição da qualidade pode ser realizada através do processo de engenharia de software e depois do software ter sido entregue ao usuário. Além disso, é necessário que ela seja avaliada através de várias características de qualidade. Caso este número de características seja insuficiente para definir qualidade em detalhes é necessário que cada característica de qualidade seja refinada em sub-características e estruturada hierarquicamente.

Para se obter a qualidade desejada de produtos de software, são necessário modelos que viabilizem a avaliação da qualidade desses produtos. Segundo a ISO, o principal propósito da avaliação de qualidade do software é fornecer resultados quantitativos referentes aos produtos de software, que sejam compreensíveis, aceitáveis e confiáveis por qualquer parte interessada. A satisfação dos usuários e o retorno econômico são, também, importantes considerações [Lima, 2003].

A seguir, será apresentada a norma ISO/IEC 9126-1 [ABNT, 2003] que possui um modelo de qualidade para produtos de software.

3.3 Norma ISO/IEC 9126-1

Comissões internacionais e nacionais para estudos da qualidade de software têm definido parâmetros para a qualidade de um software, como a SEI – *Software Engineering Institute* – e a ABNT – Associação Brasileira de Normas Técnicas.

A NBR ISO/IEC 9126 é uma norma elaborada pela ABNT sob o título geral "Engenharia de software - Qualidade do produto" e consiste nas seguintes partes:

- **Parte 1**: Modelo de qualidade;
- **Parte 2**: Métricas externas;
- **Parte 3**: Métricas internas;
- **Parte 4**: Métricas de qualidade em uso.

As características de qualidade e as métricas associadas podem ser úteis não só à avaliação de produto de software, mas também para a definição de requisitos de qualidade e outros usos. As características de qualidade do produto de software definidas na "Parte 1" da NBR ISO/IEC 9126 podem ser usadas para especificar requisitos funcionais e não-funcionais do cliente e do usuário [ABNT, 2003].

O modelo de qualidade do produto de software descrito pela NBR ISO/IEC 9126 [ABNT, 2003], é composto de duas partes:

- qualidade interna e qualidade externa;
- qualidade em uso.

A primeira parte do modelo especifica seis características para qualidade interna e externa, as quais são por sua vez subdivididas em sub-características. Estas sub-características são manifestadas externamente, quando o software é utilizado como parte de um sistema computacional, e são resultantes de atributos internos do software.

A segunda parte do modelo especifica quatro características de qualidade em uso, mas não apresenta o modelo de qualidade em uso além do nível de característica. Qualidade em uso é, para o usuário, o efeito combinado das seis características de qualidade do produto de software.

O modelo de qualidade definido na parte um da NBR ISO/IEC 9126 possui os seguintes exemplos de uso:

- validar a completitude de uma definição de requisitos;
- identificar requisitos de software;
- identificar objetivos de projeto de software;
- identificar objetivos para teste de software;
- identificar critérios para garantia de qualidade;
- identificar critérios de aceitação para produtos finais de software.

Existem diferentes visões da qualidade do produto e de suas métricas em diferentes estágios do ciclo de vida do software que podem ser vistos na Figura 3.1. Isto é, ciclo de vida de um software pode ser composto por várias etapas, e cada uma delas possui suas características e precisam ser avaliadas de uma maneira diferente.

Nos próximos parágrafos será definido a descrição de cada etapa do ciclo de vida do software, descrito na Figura 3.1.

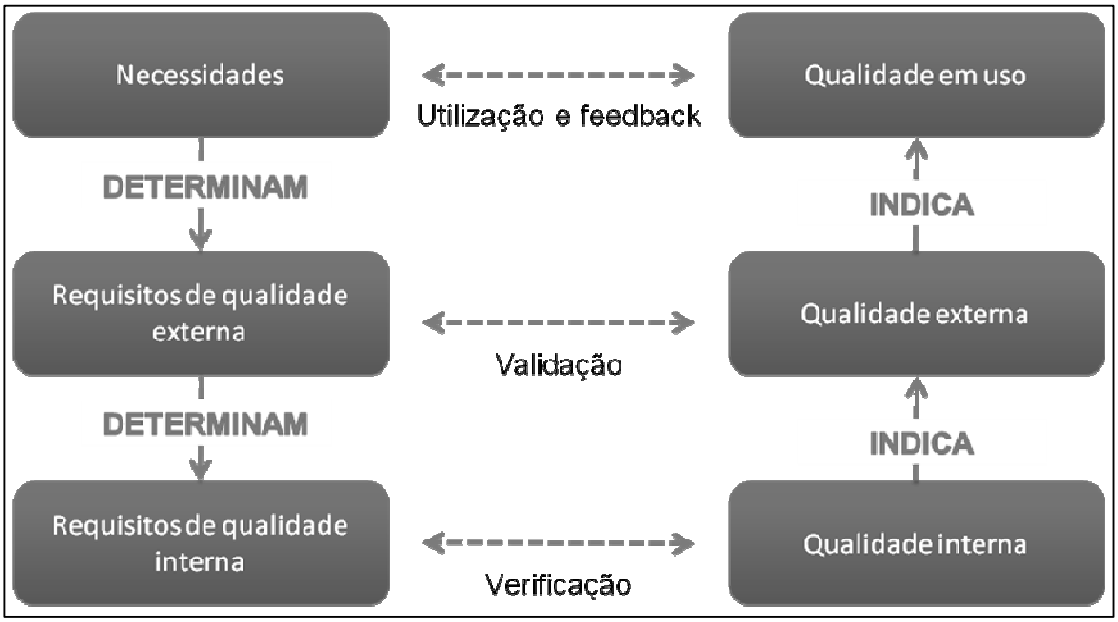

Figura 3.1 – Qualidade no ciclo de vida do software [ABNT, 2003]

As **necessidades** de qualidade do usuário podem ser especificadas como requisitos de qualidade pelas métricas de qualidade em uso, pelas métricas externas e algumas vezes por métricas internas. Convém que estes requisitos especificados por meio das métricas sejam utilizados como critérios quando um produto for validado. Obter um produto que satisfaça as necessidades do usuário normalmente requer uma abordagem iterativa para o desenvolvimento de software com *feedback* contínuo sob a perspectiva do usuário [ABNT, 2003].

Os **requisitos de qualidade externa** especificam o nível de qualidade requerido sob o ponto de vista externo. Eles incluem requisitos derivados das necessidades de qualidade dos usuários, incluindo os requisitos de qualidade em uso. Os requisitos de qualidade externa são usados como meta para validação em vários estágios de desenvolvimento. Convém que os requisitos de qualidade externa, abrangidos pelas características de qualidade definidas na parte um da NBR ISO/IEC 9126, sejam declarados na especificação de requisitos de qualidade usando métricas externas, bem como convém que sejam traduzidos em requisitos de qualidade interna e que também sejam usados como critérios quando da avaliação do produto [ABNT, 2003].

Os **requisitos de qualidade interna** especificam o nível de qualidade requerido sob o ponto de vista interno do produto. Os requisitos de qualidade interna são usados para especificar as propriedades dos produtos intermediários. Estes podem incluir modelos

estáticos e dinâmicos, outros documentos e código-fonte. Requisitos de qualidade interna podem ser usados como metas para validação em vários estágios de desenvolvimento. Eles também podem ser usados para definir estratégias de desenvolvimento e critérios de avaliação e de verificação durante o desenvolvimento. Isto pode incluir o uso de métricas adicionais (por exemplo, reusabilidade), as quais estão fora do escopo da NBR ISO/IEC 9126. Convém que requisitos de qualidade interna sejam definidos quantitativamente usando métricas internas [ABNT, 2003].

Qualidade interna é a totalidade das características do produto de software do ponto de vista interno. A qualidade interna é medida e avaliada com relação aos requisitos de qualidade interna. Detalhes da qualidade do produto de software podem ser melhorados durante a implementação do código, revisão e teste, mas a natureza fundamental da qualidade do produto de software representada pela qualidade interna mantém-se inalterada, a menos que seja reprojetada [ABNT, 2003].

Qualidade externa estimada (ou prevista) é a qualidade estimada ou prevista para o produto final de software, em cada estágio de desenvolvimento e para cada característica de qualidade, baseada no conhecimento da qualidade interna [ABNT, 2003].

Qualidade externa é a totalidade das características do produto de software do ponto de vista externo. É a qualidade quando o software é executado, o qual é tipicamente medido e avaliado enquanto está sendo testado num ambiente simulado, com dados simulados e usando métricas externas. Durante os testes, convém que a maioria dos defeitos seja descoberta e eliminada. Entretanto, alguns defeitos podem permanecer após o teste. Como é difícil corrigir a arquitetura do software ou outro aspecto básico do projeto do software, a base do projeto usualmente permanece inalterada ao longo do teste [ABNT, 2003].

Qualidade em uso estimada (ou prevista) é a qualidade estimada ou prevista para o produto final de software, em cada estágio de desenvolvimento e para cada característica de qualidade em uso, baseada no conhecimento da qualidade interna e externa [ABNT, 2003].

Qualidade em uso é a visão da qualidade do produto de software do ponto de vista do usuário, quando este produto é usado em um ambiente e um contexto de uso especificados. Ela mede o quanto usuários podem atingir seus objetivos num determinado ambiente e não as propriedades do software em si (qualidade em uso está definida na seção 7) [ABNT, 2003].

O nível de qualidade no ambiente do usuário pode ser diferente daquele no ambiente do desenvolvedor, por causa das diferenças entre necessidades e capacidades de diferentes usuários e diferenças entre os ambientes de hardware e de apoio. O usuário avalia somente aqueles atributos do software que são usados em sua tarefa. Algumas vezes, atributos de software especificados por um usuário final, durante a fase de análise de requisitos, não mais atendem aos requisitos do usuário quando o produto é usado, devido à mudança de requisitos do usuário e à dificuldade em especificar as necessidades implícitas [ABNT, 2003].

3.3 Modelo de qualidade para qualidade externa e interna

Esta seção define o modelo de qualidade externa e interna da Norma NBR ISO/IEC 9126. Ele categoriza os atributos de qualidade de software em seis características (funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade) as quais são, por sua vez, subdivididas em sub-características como pode ser visto na Figura 3.2. As sub-características podem ser medidas por meio de métricas externas e internas.

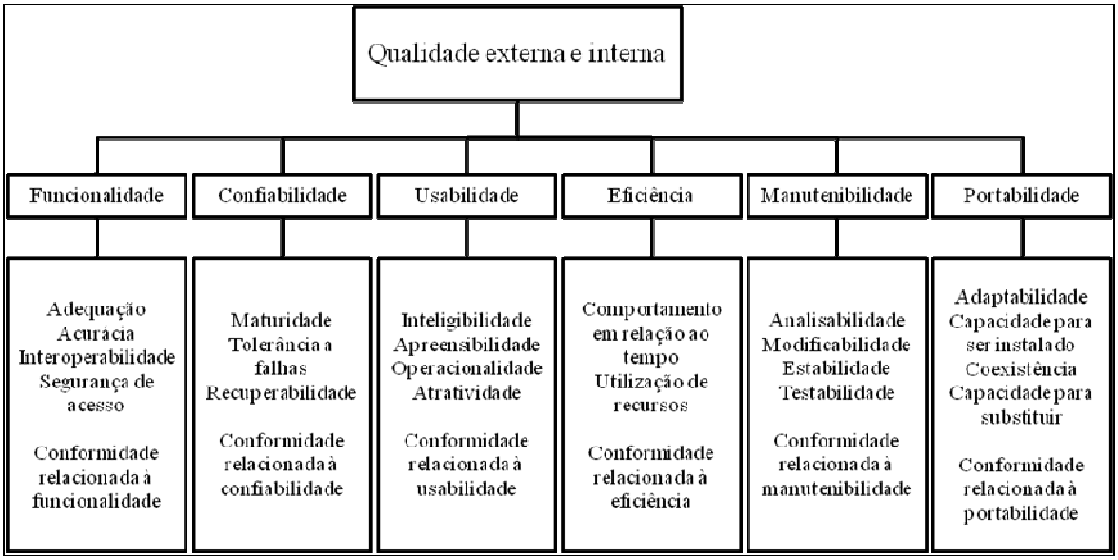

Figura 3.2 – Modelo de qualidade para qualidade externa e interna

Este trabalho teve o foco em apenas duas características: funcionalidade e usabilidade. Portanto, apenas estas serão definidas.

3.3.1 Funcionalidade

Defini-se funcionalidade por capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas [ABNT, 2003].

Ela é sub-dividida nas seguintes sub-características:

- **Adequação:** capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados.
- **Acurácia:** capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.
- **Interoperabilidade: c**apacidade do produto de software de interagir com um ou mais sistemas especificados.
- **Segurança de acesso:** capacidade do produto de software de proteger informações e dados, de forma que pessoas ou sistemas não autorizados não possam lê-los nem modificá-los e que não seja negado o acesso às pessoas ou sistemas autorizados.
- **Conformidade relacionada à funcionalidade:** capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações previstas em leis e prescrições similares relacionadas à funcionalidade.

3.3.2 Usabilidade

Defini-se usabilidade por capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas [ABNT, 2003].

Ela é sub-dividida nas seguintes sub-características:

- **Inteligibilidade:** capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.
- **Apreensibilidade:** capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.
- **Operacionalidade:** capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.

- **Atratividade:** capacidade do produto de software de ser atraente ao usuário.
- **Conformidade relacionada à usabilidade:** capacidade do produto de software de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade.

Capítulo 4

Qualidade do Gerador de Estruturas Ferromagnéticas

4.1 Introdução

A qualidade do produto final do Gerador de Estruturas Ferromagnéticas foi uma preocupação constante dos pesquisadores e dos desenvolvedores envolvidos. Foi definido então que seriam identificados os atributos de qualidade do produto, em um processo que envolveria todos os futuros usuários e outras pessoas relacionadas ao projeto (*stakeholders*).

Dessa forma foi desenvolvido um processo para identificar esses atributos de qualidade, baseados na Norma ISO/IEC 9126 [ABNT, 2003], com foco nas características de funcionalidade e usabilidade, já descritas no capítulo anterior.

Este estudo foi realizado a partir da perspectiva dos usuários finais e *stakeholders* do software, será conduzido dentro do contexto do Núcleo do Simulador de Spins. O universo dos participantes do estudo é pequeno, seis participantes, e, portanto, os resultados aqui produzidos não poderão ser generalizados. Contudo, o estudo é válido, pois se pretende utilizar as características sugeridas e validadas para elaborar um conjunto de atributos que comporão os requisitos de qualidade do software.

O modelo de avaliação proposto neste trabalho consiste em fazer um levantamento para futura avaliação dos principais artefatos do software de acordo com as características funcionalidade e usabilidade do modelo de qualidade, citado na norma ISO/IEC 9126-1 [ABNT, 2003]. Com a identificação destas característica espera-se também que todo o processo de desenvolvimento garanta que o produto final do software atenderá a esse conjunto de critérios de qualidade.

4.2 Planejamento e levantamento das características de qualidade

Para realizar o levantamento dos principais atributos relacionados à funcionalidade e à usabilidade do software desenvolvido, quatro atores envolvidos com o projeto foram

entrevistados. Os perfis destes atores e de outros dois que participaram da segunda parte da entrevista estão descritos na Tabela 4.1.

Ator	Título	Área de atuação 1	Tempo de atuação 1	Área de atuação 2	Tempo de atuação 2
Ator 1	D. Sc. em Ciência da Computação	Professor	6 anos	Pesquisador	5 anos
Ator 2	D. em Eng. de Sistemas e Computação	Professor	8 anos	Pesquisador	5 anos
Ator 3	Doutor em Ciências	Professor	17 anos	Pesquisador	11 anos
Ator 4	Graduando em Ciência da Computação	--	--	--	--
Ator 5	Graduando em Ciência da Computação	--	--	--	--
Ator 6	Mestrando em Modelagem de Sistemas Computacionais	--	--	--	--

Tabela 4.1 – Atores envolvidos.

A entrevista contou com duas etapas. Das entrevistas participaram os quatro primeiros atores descritos na Tabela 4.1. Considerando que os atores envolvidos não conheciam as características e sub-características do modelo de qualidade apresentado na norma ISO/IEC 9126 [ABNT, 2003] a definição destes conceitos foi incluída como primeira etapa. Na segunda etapa, cada entrevistado deveria citar características que considerasse importantes para o software gerador de estruturas ferromagnéticas. Com isso, foi possível levantar os principais atributos de qualidade sugeridos pelos atores.

Na primeira etapa, foi então apresentado ao entrevistado os conceitos de funcionalidade e de suas sub-características: adequação, acurácia, interoperabilidade, segurança de dados e conformidade em funcionalidade. A cada definição esclarecida o ator teve a possibilidade de identificar algo que achasse importante e adequado ao contexto. Neste ponto, todos concordaram que a sub-característica segurança de dados não se aplica neste momento, o que poderá ser avaliado no futuro é a utilização de criptografia com chaves publicas nos dados de saída.

Em seguida, foram apresentados os conceitos de usabilidade e de suas sub-características: inteligibilidade, apreensibilidade, operacionalidade, atratividade e conformidade em usabilidade. Da mesma forma que foi realizada na característica funcionalidade, o ator teve a possibilidade de levantar pontos importantes.

Após passar por todas as características e sub-características, os entrevistados tiveram a possibilidade de acrescentar novos atributos dentro do contexto sugerido. Além disto, os atores puderam explicitar suas idéias.

As entrevistas foram feitas pessoalmente e individualmente, o que fez com que um ator não influenciasse na resposta de outro. Este fato pode ser considerado um ponto importante, pois assim é possível saber o que todos pensam em comum e o que cada um considera importante e prioritário.

Neste levantamento foram identificados 45 atributos de qualidade. Os atributos identificados em cada entrevista estão relacionados no Apêndice A.

4.3 Validação das características de qualidade

Tendo como base as entrevistas realizadas e os atributos sugeridos, foi construída uma ficha de validação, os Tabela 4.2, contendo quarenta e cinco atributos. Para seleção dos atributos que devem compor o conjunto final de características de qualidade do software gerador de estruturas ferromagnéticas, a amostra dos seis atores descritos na Tabela 4.1 participaram do processo. Cada ator, selecionou de acordo com o seu julgamento, o grau de importância da característica listada, numa escala de três opções: muito importante, importante ou sem importância.

Nº	Característica – Sub-característica	Atributos	Muito importante	Importante	Sem importância
1	Funcionalidade – Adequação	Ter meios para modificar campos magnéticos em função do tempo.			
2	Funcionalidade – Adequação	Ter ambiente para geração de objetos ferromagnéticos arbitrários.			
3	Funcionalidade – Adequação	Ter meios de modificar o objeto em função do tempo.			
4	Funcionalidade – Adequação	Ser capaz de gerar estruturas de maneira correta e otimizada.			
5	Funcionalidade – Adequação	Ter uma linguagem capaz de gerar todas as estruturas possíveis.			
6	Funcionalidade – Adequação	Ter facilidade de gerar estruturas de diversas topologias simples e complexas (união e interseção, por exemplo).			
7	Funcionalidade – Adequação	Ter capacidade de gerar superestruturas (polímeros) a partir de uma estrutura simples.			
8	Funcionalidade – Acurácia	Exibir a estrutura desejada de forma precisa para não ter a necessidade de ficar verificando os pontos.			
9	Funcionalidade – Acurária	Exibir possíveis erros inseridos pelo usuário de uma maneira explicativa.			

10	Funcionalidade – Acurácia	Ser capaz de corretamente determinar o suporte geométrico dado a sua forma implícita.			
11	Funcionalidade – Acurácia	Ter informações de erros (*bugs*) de compilação.			
12	Funcionalidade – Interoperabilidade	Ter interface de entrada textual, mas com possibilidade de passagem remota do programa.			
13	Funcionalidade – Interoperabilidade	Ter interface de entrada com passagem direta (local) dentro do programa na linguagem C (utilizando uma API, por exemplo) que permitam programadores utilizar.			
14	Funcionalidade – Interoperabilidade	Permitir arquivos textos de entrada.			
15	Funcionalidade – Interoperabilidade	Ser capaz de entregar a estrutura geométrica dos objetos em diversos formatos: BSP, Octree, matriz tridimensional.			
16	Funcionalidade – Interoperabilidade	Ter uma saída de forma que outros softwares consigam ler.			
17	Funcionalidade – Interoperabilidade	Utilizar criptografia de chave publica para a saída.			
18	Funcionalidade – Interoperabilidade	Interagir com o Monte Carlo spins em tempo de utilização. (aceitando suas entradas e passando corretamente suas estruturas de saídas)			
19	Funcionalidade – Interoperabilidade	Ser extensível a outras tecnologias de grade.			
20	Funcionalidade – Interoperabilidade	O compilador (interpretador) e o particionador devem ser integrados.			
21	Funcionalidade – Conformidade	O compilador deve estar com em conformidade com a linguagem sugerida.			
22	Funcionalidade – Conformidade	Ter pré-requisitos mínimos de sistemas de simulação física.			
23	Funcionalidade – Conformidade	As grandezas físicas devem estar de acordo com sistema internacional de medidas.			
24	Funcionalidade – Conformidade	As grandezas físicas devem estar dentro das normas da ABNT de grandezas físicas.			
25	Usabilidade – Inteligibilidade	Ter manual básico da linguagem para os usuários conseguirem utilizar.			
26	Usabilidade – Inteligibilidade	Ter manual básico de erros que faça o usuário compreender melhor qual é o problema no código inserido.			
27	Usabilidade – Inteligibilidade	Ter uma linguagem fácil e próxima a linguagem comuns.			
28	Usabilidade – Inteligibilidade	Ter uma documentação de apresentação para que usuários saibam o que existe no software.			
29	Usabilidade – Inteligibilidade	Ter uma estrutura interna de componentes fácil de lidar para			

		programadores.			
30	Usabilidade – Inteligibilidade	Ter o código bem documentado para facilitar futuras expansões, manutenção e integrações.			
31	Usabilidade – Inteligibilidade	Ter uma documentação do gerador explicando como ele faz a transformação entre uma estrutura definida na linguagem e a estrutura gerada.			
32	Usabilidade – Apreensibilidade	Ter mensagens de erros intuitivas.			
33	Usabilidade – Apreensibilidade	Ter as construções da linguagem intuitivas ao usuário.			
34	Usabilidade – Apreensibilidade	Ter mais de uma forma (construção) de se obter um resultado.			
35	Usabilidade – Operacionalidade	Deve sempre retornar ou um erro ou a estrutura.			
36	Usabilidade – Operacionalidade	Ter mensagens de erro que é o mais próximo possível do que o usuário tentou fazer.			
37	Usabilidade – Operacionalidade	Deve compilar rapidamente.			
38	Usabilidade – Operacionalidade	Deve retornar a estrutura rapidamente.			
39	Usabilidade – Operacionalidade	Ser otimizado.			
40	Usabilidade – Atratividade	Ter características gerais para se tornar padrão no meio acadêmico.			
41	Usabilidade – Atratividade	Deve ser confiável nas operações de modelagem e simulação.			
42	Usabilidade – Atratividade	Ter como selecionar estruturas de forma visual, através de uma ferramenta gráfica.			
43	Usabilidade – Atratividade	Ter uma ajuda dentro do compilador de forma a auxiliar a utilização. (“?” abriria o a ajuda e “? comando” abriria a sintaxe do comando)			
44	Usabilidade – Atratividade	Levar em consideração formas clássicas de se definir objetos.			
45	Usabilidade – Atratividade	Levar em consideração formas clássicas de se determinar parâmetros de simulação.			

Tabela 4.2 – Ficha de validação.

Cada ator preencheu a ficha de validação individualmente. Todas as notas foram contabilizadas, e para realizar a seleção dos atributos mais importantes, foi feito um comparativo entre duas métricas utilizadas que serão descritas nos tópicos a seguir.

4.3.1 Métrica 1: utilização da média ponderada

A primeira métrica selecionada para selecionar as características mais importantes para os atores foi a média ponderada de cada atributo avaliado. Portanto, cada nota "Muito importante", "Importante" e "Sem importância" recebeu os seguintes pesos 4, 2 e 1, respectivamente. Estes pesos foram selecionados de forma que os artefatos considerados mais importantes fossem priorizados.

Considerando que "Muito importante" = "MI", "Importante" = "I" e "Sem importância" = "SI", utilizou-se a seguinte fórmula:

$$Metrica1 = \frac{(MI * 4) + (I * 2) + (SI * 1)}{(4 + 2 + 1)}$$

Após aplicar esta métrica em todos os pontos, obteve-se o gráfico presente na Figura 4.1.

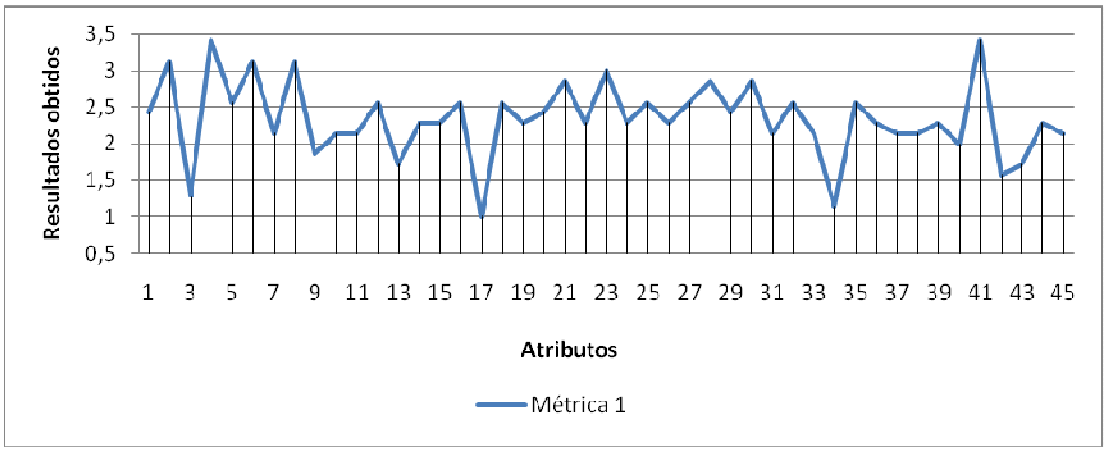

Figura 4.1 – Resultados da aplicação da métrica 1

Com esta métrica obteve-se a classificação dos vinte primeiros atributos com maiores pontuações, descrita na Tabela 4.3. As maiores notas foram selecionadas porque são as candidatas a atributos do software, por terem sido consideradas mais importantes que as outras.

Nº	Característica – Sub-característica	Atributos
4	Funcionalidade – Adequação	Ser capaz de gerar estruturas de maneira correta e otimizada.
41	Usabilidade – Atratividade	Deve ser confiável nas operações de modelagem e simulação.
2	Funcionalidade – Adequação	Ter ambiente para geração de objetos ferromagnéticos arbitrários.
6	Funcionalidade – Adequação	Ter facilidade de gerar estruturas de diversas topologias simples e complexas (união e interseção, por exemplo).
8	Funcionalidade – Acurácia	Exibir a estrutura desejada de forma precisa para não ter a necessidade de ficar verificando os pontos.
23	Funcionalidade – Conformidade	As grandezas físicas devem estar de acordo com sistema internacional de medidas.

21	Funcionalidade – Conformidade	O compilador deve estar com em conformidade com a linguagem sugerida.
28	Usabilidade – Inteligibilidade	Ter uma documentação de apresentação para que usuários saibam o que existe no software.
30	Usabilidade – Inteligibilidade	Ter o código bem documentado para facilitar futuras expansões, manutenção e integrações.
5	Funcionalidade – Adequação	Ter uma linguagem capaz de gerar todas as estruturas possíveis.
12	Funcionalidade – Interoperabilidade	Ter interface de entrada textual, mas com possibilidade de passagem remota do programa.
16	Funcionalidade – Interoperabilidade	Ter uma saída de forma que outros softwares consigam ler.
18	Funcionalidade – Interoperabilidade	Interagir com o Monte Carlo spins em tempo de utilização. (aceitando suas entradas e passando corretamente suas estruturas de saídas)
25	Usabilidade – Inteligibilidade	Ter manual básico da linguagem para os usuários conseguirem utilizar.
27	Usabilidade – Inteligibilidade	Ter uma linguagem fácil e próxima a linguagem comuns.
32	Usabilidade – Apreensibilidade	Ter mensagens de erros intuitivas.
35	Usabilidade – Operacionalidade	Deve sempre retornar ou um erro ou a estrutura.
1	Funcionalidade – Adequação	Ter meios para modificar campos magnéticos em função do tempo.
20	Funcionalidade – Interoperabilidade	O compilador (interpretador) e o particionador devem ser integrados.
29	Usabilidade – Inteligibilidade	Ter uma estrutura interna de componentes fácil de lidar para programadores.

Tabela 4.3 – Vinte primeiros atributos após a aplicação da métrica 1

4.3.2 Métrica 2: uso do decaimento quadrático

Nesta segunda métrica selecionada para selecionar as características mais importantes para os atores utilizou-se o decaimento quadrático das notas para "Sem importância". Com isto fica fácil visualizar quais atributos são mais importantes e quais possuem a maior probabilidade de sair da lista.

Considerando que "Muito importante" = "MI", "Importante" = "I" e "Sem importância" = "SI", utilizou-se a seguinte fórmula:

$$Metrica2 = \frac{(MI+1)*(I+1)}{SI^2+1}$$

Somou-se um a cada valor para garantir um melhor resultado.

Após aplicar esta métrica em todos os pontos, obteve-se o gráfico presente na Figura 4.2.

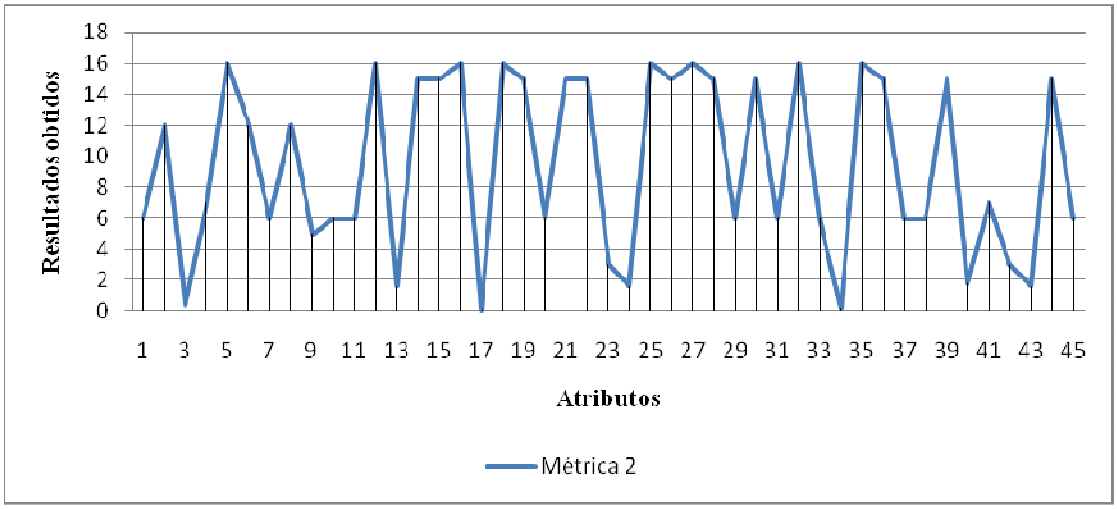

Figura 4.2 – Resultados da aplicação da métrica 2

Para esta métrica também foram extraídos os vinte primeiros atributos classificados com maiores pontuações, descrita na Tabela 4.4. Da mesma forma que a métrica anterior, as maiores notas foram selecionadas porque são as candidatas a atributos do software, por terem sido consideradas mais importantes que as outras.

Nº	Característica – Sub-característica	Atributos
5	Funcionalidade – Adequação	Ter uma linguagem capaz de gerar todas as estruturas possíveis.
12	Funcionalidade – Interoperabilidade	Ter interface de entrada textual, mas com possibilidade de passagem remota do programa.
16	Funcionalidade – Interoperabilidade	Ter uma saída de forma que outros softwares consigam ler.
18	Funcionalidade – Interoperabilidade	Interagir com o Monte Carlo spins em tempo de utilização. (aceitando suas entradas e passando corretamente suas estruturas de saídas)
25	Usabilidade – Inteligibilidade	Ter manual básico da linguagem para os usuários conseguirem utilizar.
27	Usabilidade – Inteligibilidade	Ter uma linguagem fácil e próxima a linguagem comuns.
32	Usabilidade – Apreensibilidade	Ter mensagens de erros intuitivas.
35	Usabilidade – Operacionalidade	Deve sempre retornar ou um erro ou a estrutura.
14	Funcionalidade – Interoperabilidade	Permitir arquivos textos de entrada.
15	Funcionalidade – Interoperabilidade	Ser capaz de entregar a estrutura geométrica dos objetos em diversos formatos: BSP, Octree, matriz tridimensional.
19	Funcionalidade – Interoperabilidade	Ser extensível a outras tecnologias de grade.
21	Funcionalidade – Conformidade	O compilador deve estar com em conformidade com a linguagem sugerida.
22	Funcionalidade – Conformidade	Ter pré-requisitos mínimos de sistemas de simulação física.
26	Usabilidade – Inteligibilidade	Ter manual básico de erros que faça o usuário compreender melhor qual é o problema no código inserido.
28	Usabilidade – Inteligibilidade	Ter uma documentação de apresentação para que usuários saibam o que existe no software.
30	Usabilidade – Inteligibilidade	Ter o código bem documentado para facilitar futuras

		expansões, manutenção e integrações.
36	Usabilidade – Operacionalidade	Ter mensagens de erro que é o mais próximo possível do que o usuário tentou fazer.
39	Usabilidade – Operacionalidade	Ser otimizado.
44	Usabilidade – Atratividade	Levar em consideração formas clássicas de se definir objetos.
2	Funcionalidade – Adequação	Ter ambiente para geração de objetos ferromagnéticos arbitrários.

Tabela 4.4 – Vinte primeiros atributos após a aplicação da métrica 2.

4.4 Características de qualidade do software gerador de estruturas ferromagnéticas

Após o processo de identificação e validação das características de qualidade para o produto de software gerador de estruturas ferromagnéticas, onde se priorizou as características de funcionalidade e usabilidade, a partir da Norma ISO/IEC 9126, foram então destacados os atributos descritos na Tabela 4.5.

Para chegar ao resultado dos principais características foi feita uma interseção entre os resultados das duas métricas aplicadas que resultou em onze atributos. Portanto, estes atributos podem ser considerados os principais atributos de qualidade que devem constar do software ao término da construção do mesmo..

Nº	Característica – Sub-característica	Atributos
2	Funcionalidade – Adequação	Ter ambiente para geração de objetos ferromagnéticos arbitrários.
5	Funcionalidade – Adequação	Ter uma linguagem capaz de gerar todas as estruturas possíveis.
12	Funcionalidade – Interoperabilidade	Ter interface de entrada textual, mas com possibilidade de passagem remota do programa.
16	Funcionalidade – Interoperabilidade	Ter uma saída de forma que outros softwares consigam ler.
18	Funcionalidade – Interoperabilidade	Interagir com o Monte Carlo spins em tempo de utilização. (aceitando suas entradas e passando corretamente suas estruturas de saídas)
21	Funcionalidade – Conformidade	O compilador deve estar com em conformidade com a linguagem sugerida.
27	Usabilidade – Inteligibilidade	Ter uma linguagem fácil e próxima a linguagem comuns.
28	Usabilidade – Inteligibilidade	Ter uma documentação de apresentação para que usuários saibam o que existe no software.
30	Usabilidade – Inteligibilidade	Ter o código bem documentado para facilitar futuras expansões, manutenção e integrações.
32	Usabilidade – Apreensibilidade	Ter mensagens de erros intuitivas.
35	Usabilidade – Operacionalidade	Deve sempre retornar ou um erro ou a estrutura.

Tabela 4.5 – Atributos de qualidade do gerador de estrutura

Com estes atributos será possível realizar a avaliação do produto de software ao longo do processo de desenvolvimento e após o término de sua construção.

4.5 Utilização dos atributos selecionados

A partir do momento em que se tem a lista dos principais atributos para o software, o mesmo pode ser construído com uma maior precisão, pois o desenvolvedor terá em mão todos os pontos que foram considerados como principais para a qualidade do produto.

Caso o levantamento de requisitos não tenha sido de forma correta, é necessário identificar os requisitos e retornar ao início do ciclo de vida do software validando todos os pontos de forma a verificar e validar se todos os atributos foram aplicados corretamente.

Caso algum atributo não tenha sido aplicado, o processo de produção voltará ao ponto em que este atributo possa ser inserido ao produto.

Ao chegar ao final do processo de produção será possível avaliar todos estes requisitos do software junto a equipe envolvida.

Capítulo 5

Considerações finais

Alcançar a qualidade do produto de software não é algo trivial. É necessário que todas as etapas de um processo de produção passem por uma avaliação para assim garantir a qualidade final do produto.

Este trabalho mostrou parte do produto a ser construído e todo o processo para se levantar atributos importantes para avaliá-lo ao fim da construção do software. Como o software está no processo final de construção, não foi possível realizar a sua avaliação, que fica aqui como sugestão de trabalho futuro.

Fazer o levantamento de requisitos requer envolvimento de todos os usuários e *stakeholders* do projeto. Portanto, existe uma enorme necessidade de realizar reuniões, analisar os dados, realizar pesquisas e entrevistas para conseguir chegar a um ponto em que o produto de software saia com os resultados esperados por todos.

Contudo, sempre existem pontos que não ficam conforme o esperado. E isto, faz com que a avaliação final do software seja necessária e imprescindível. Pois, a partir dela, é possível identificar pontos que não haviam sido analisados anteriormente.

Outras sugestões para trabalhos futuros:

- Realizar a avaliação do software gerador de estruturas ferromagnéticas utilizando os atributos explicitados por esta monografia;
- Fazer o levantamento de atributos do software gerador de estruturas ferromagnéticas das demais características do modelo de qualidade da Norma 9126-1 [ABNT, 2003];
- Analisar a qualidade em uso do software gerador de estruturas ferromagnéticas baseado na Norma 9126-1 [ABNT, 2003].

Apêndice A

Ator 1

Título: D. Sc. em Ciência da Computação, ENSEA-UCP/France, 2002

Profissão: Professor Universitário e Pesquisador

Tempo de atuação na área: 6 anos atuando como professor e 5 anos atuando como pesquisador

Questionário:

1. Funcionalidade:
    a. Adequação:
        i. Prover meios de modificar campos magnéticos em função do tempo;
        ii. Prover um ambiente de geração de objetos ferro magnéticos arbitrários;
        iii. Prover meios de modificar objetos em função do tempo.
    b. Acurácia:
        i. Deve ser capaz de corretamente determinar o suporte geométrico dado a sua forma implícita.
    c. Interoperabilidade:
        i. Entrada:
            1. Deve ter interface de entrada textual, mas com a possibilidade de passagem remota do programa;
            2. Passagem direta (local) dentro do programa na linguagem C, utilizando uma API, por exemplo, que permitam programadores utilizar;
            3. Deve permitir arquivos textos de entrada.
        ii. Saída:
            1. Deve ser capaz de entregar a estrutura geométrica do objeto em diversos formatos e estrutura de dados: BSP, Octree, matriz (grade) tridimensional.
    d. Segurança de acesso: (não se aplica)
    e. Conformidades:
        i. Pré-requisitos mínimos de sistemas de simulação física.

2. Usabilidade:
    a. Inteligibilidade:
        i. Necessidade de manual;
        ii. Uma linguagem fácil e próxima a linguagens comuns;
        iii. Estrutura de componentes fácil de lidar para programadores.
    b. Apreensibilidade:
        i. As construções da linguagem devem ser intuitivos suficientes para o físico utilizar.
    c. Operacionalidade:
        i. Deve-se compilar rapidamente;
        ii. Entregar a estrutura rapidamente;
        iii. Exibir mensagens de erros esclarecedoras.
    d. Atratividade:
        i. O software deve ter características gerais para se tornar padrão no meio acadêmico;
        ii. Deve ser confiável nas operações de modelagem e simulação.
    e. Conformidade:
        i. Deve-se levar em consideração formas clássicas de se definir objetos e de se determinar parâmetros de simulação.
3. Outras características:
    a. Tem que ser rápido;
    b. Tem que ser extensível a novas tecnologias de grade.

Ator 2

Título: D. em Eng. de Sistemas e Computação, UFRJ/Brasil, 2005

Profissão: Professor Universitário e Pesquisador

Tempo de atuação na área: 8 anos atuando como professor e 5 anos atuando como pesquisador

Questionário:

1. Funcionalidade:
    a. Adequação e acurácia:

i. Gerador de estrutura: Ser capaz de gerar as estruturas de maneira correta e otimizada;
ii. Compilador: linguagem ser capaz de expressar todas as estruturas possíveis, de maneira correta;
iii. Compilador: informações de erro de compilação;

b. Interoperabilidade:
i. O gerador de estrutura tem que interagir com o simulador de spins, de forma a passar as estruturas de forma adequada;

c. Segurança de acesso: (não se aplica)

d. Conformidades:
i. Compilador esteja em conformidade com a linguagem sugerida;

2. Usabilidade:

a. Inteligibilidade:
i. Manual de linguagem, para compreender e programar na linguagem.
ii. O código dos softwares devem estar bem documentados para desenvolvedores compreenderem facilmente o que cada código faz, para futuras expansões, manutenção, integrações;
iii. Documentação do gerador dizendo como ele faz a transformação entre uma estrutura definida na linguagem, estrutura que ele está representando.

b. Apreensibilidade:
i. Ter mensagens de erros intuitivas para o usuário aprender através dos erros.

c. Operacionalidade:
i. Mensagens de erro ser o mais próximo possível do erro;

d. Atratividade:
i. No futuro ter como selecionar as estruturas de forma visual, através de uma ferramenta gráfica, de forma que esta ferramenta passe as estruturas corretamente ao compilador;

e. Conformidade:
i. Operar de acordo com o que o usuário espera, sendo capaz de gerar as estruturas de acordo com o que o usuário está esperando;

3. Características:

- Processo de compilação e geração seja feito rápido e sem erros;

- Seja otimizado;
- Que todo o processo seja integrado, desde a estrutura até a geração e simulação de spins.

Ator 3

Título: Doutor em Ciências, UNICAMP/ Brasil, 1994

Profissão: Professor Universitário e Pesquisador

Tempo de atuação na área: 17 anos atuando como professor e 11 anos atuando como pesquisador

Questionário:

1. Funcionalidade
   a. Adequação:
      i. Ter a facilidade de gerar estruturas de diversas topologias (união e interseção, por exemplo) simples e complicadas;
      ii. Também gerar superestruturas (*polímeros*) a partir de uma estrutura simples.
   b. Acurácia:
      i. Ao gerar as estruturas deve ter confiança que a estrutura gerada está precisa. Não precisando verificar todos os pontos.
   c. Interoperabilidade:
      i. Sua saída tem que ser lida por outro software mesmo que for compactada por alguma criptografia de chave pública;
      ii. Deve interagir com o Monte Carlo Spins em tempo de navegação.
   d. Segurança de acesso: (não se aplica)
   e. Conformidades:
      i. As grandezas físicas (sistemas de medidas) devem estar dentro do padrão do sistema internacional de medidas;
      ii. Deve estar dentro das normas da ABNT de grandezas físicas (sistema internacional de medidas).
2. Usabilidade:
   a. Inteligibilidade:
      i. Ter uma documentação é fundamental para as pessoas saberem o que o software possui e como utilizar.

b. Apreensibilidade:
    i. Comandos devem ser intuitivos para o usuário saber e conseguir fazer;
    ii. Ter mais de uma forma para fazer a mesma tarefa para não dificultar o usuário.
c. Operacionalidade:
    i. Gerar as estruturas corretamente sem *bugs* internos.
d. Atratividade:
    i. Pode ter algo, mas não se adéqua;
    ii. Ter um help dentro do compilador para listar os comandos e os exemplos como "?" para listar help geral e "? comando" para exibir a sintaxe do comando.

Ator 4

Título: Graduando em Ciência da Computação, pela UFJF

Questionário:

1. Funcionalidade:
    a. Adequação:
        i. Lidar com estruturas.
    b. Acurácia:
        i. Exibição de possíveis erros do código inserido pelo usuário;
        ii. Exibição da estrutura desejada.
    c. Interoperabilidade: --
    d. Segurança de acesso: (não se aplica)
    e. Conformidades: --
2. Usabilidade:
    a. Inteligibilidade:
        i. Manual básico da linguagem
        ii. Manual básico de erros
    b. Apreensibilidade:
        i. Será necessário ter conhecimento da linguagem;
        ii. A linguagem é simples;
        iii. Para construção de estruturas complexas, é necessário ter conhecimento matemático.
    c. Operacionalidade:

i. Independente da entrada o programa deve reportar erros ou retornar a estrutura.

d. Atratividade:

i. Não possui estrutura atraente por se tratar de um software científico onde tudo tem como base a entrada de um código em um console.

e. Conformidade: --

3. Outras características:

a. Passos para gerar a estrutura:

i. Inserção de fórmulas;

ii. Verificação dos dados inseridos: exibe possíveis erros;

iii. Compilação dos dados: exibe possíveis erros;

iv. Visualização das estruturas geradas;

v. Exportação das estruturas.

Referências Bibliográficas

ABNT – ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 9126 Tecnologia de informação – Avaliação de produto de software**, Características de qualidade e diretrizes para o seu uso. Rio de Janeiro, 2003.

BOEGH, J. et al. ***A practitioners guide to evaluation of software*.** Publicado pelo *Software Engineering Standars Symposium.* Brighton, UK, 1993.

CAMPOS, F. C. A. **Notas de aula**. Juiz de Fora, 2009.

CAMPOS, A. M. **Implementação Paralela de Um Simulador de Spins em Uma Unidade de Processamento Gráfico**. Publicado por Universidade Federal de Juiz de Fora. Juiz de Fora, 2008.

FERREIRA, R. C. **Linguagem para geração de objetos implícitos para simulação de spins**. Publicado por Universidade Federal de Juiz de Fora. Juiz de Fora, 2009.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Padrões de Projeto: soluções reutilizáveis de software orientado a objetos**. – Publicado por Bookman. Porto Alegre, 2005.

PRESSMAN, R. S. **Engenharia de Software**. Edição: 6 – Publicado por McGraw-Hill Interamericana do Brasil Editora Ltda. São Paulo, 2006.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de Software – Teoria e Prática**. Publicado por Pearson Education. São Paulo, 2001.

SAMPAIO, A. R.; FILHO, J. B. S.; BELCHIOR, A. D. . **Qualidade Websites Bancários: um Estudo de Caso**. Publicado por Anais SBQS 2003 – II Simpósio Brasileiro de Qualidade de Software. Fortaleza, 2003.

SODRÉ, C. C. P. **Norma ISO/IEC 9126: Avaliação de Qualidade de Produtos de Software**. Publicado por Universidade Estadual de Londrina. Londrina, 2006.

SOMMERVILLE, I. **Engenharia de Software**. Edição: 8 – Publicado por Pearson Addison-Wesley, 2007.

VASCONCELOS, A. M. L.; MACIEL, T. M. M.; ROUILLER, A. C. **Introdução à Engenharia de Software e aos Princípios de Qualidade**. – Publicado por UFLA/FAEPE, 2004.