PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UM AMBIENTE PARA AVALIAÇÃO DE PROJETO DE SOFTWARE ORIENTADO A OBJETO.

Sebastião Eustáquio de Jesus

Belo Horizonte

Maio de 2008

SEBASTIÃO EUSTÁQUIO DE JESUS

UM AMBIENTE PARA AVALIAÇÃO DE PROJETO DE SOFTWARE ORIENTADO A OBJETO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Informática na área de concentração Sistemas de Informação.

Orientador: Prof. Dr. Mark Alan Junho Song

Belo Horizonte

Maio de 2008

PUC Minas
Programa de Pós-graduação em Informática

FOLHA DE APROVAÇÃO

" Um ambiente para avaliação de projeto de software orientado a objeto "

Sebastião Eustáquio de Jesus

Dissertação defendida e aprovada pela seguinte banca examinadora:

Prof. Mark Alan Junho Song - Orientador (PUC Minas)
Prof. Sérgio Vale Aguiar Campos (UFMG)
Prof. Marco Túlio de Oliveira Valente (PUC Minas)

Prof. Mark Alan Junho Song - Orientador (PUC Minas)
Doutor em Ciência da Computação - UFMG

Prof. Sérgio Vale Aguiar Campos (UFMG)
Doutor em Ciência da Computação - CMU, Estados Unidos

Prof. Marco Túlio de Oliveira Valente - (PUC Minas)
Doutor em Ciência da Computação - UFMG

Belo Horizonte, 30 de maio de 2008.

Anel Rodoviário – KM 23,5, Rua Walter Ianni, 255 – São Gabriel / Belo Horizonte – MG / CEP: 31980-110
Fone: (31) 3439-5204 – Fax: (31) 3439-5204 – e-mail: mestradoinfsec@pucminas.br

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Jesus, Sebastião Eustáquio
J58a Um ambiente para avaliação de projeto de software orientado a objeto
Sebastião Eustáquio Jesus. Belo Horizonte, 2008.
67f. : Il.

Orientador: Mark Alan Junho Song
Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais.
Programa de Pós-Graduação em Informática

1. Engenharia de software. 2. Programação orientada a objetos (Computação).
3. Software – Controle de qualidade. 4. UML (Computação). 5. XML
(Linguagem de marcação de documento). I. Song, Mark Alan Junho. II.
Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação
em Informática. III. Título.

CDU: 681.3.03

Dedicatória

Este trabalho é dedicado à memória de minha mãe Jandira.

Agradecimentos

Agradeço aos mestres pela paciência ao nos mostrar o caminho e aos amigos que incentivaram a realização desta caminhada e colaboraram de diferentes formas para que ela chegasse ao final.

Resumo

A construção de software é um processo que exige a aplicação de diferentes conceitos e técnicas da Engenharia de Software - especificamente quando se deseja gerar um produto com qualidade. Dentre as diversas abordagens para garantia da qualidade, o uso de métricas tem se tornado relevante por permitir a caracterização e mensuração dos diferentes aspectos de projeto como tamanho, complexidade, reutilização de código, facilidade de manutenção e outros.

Embora importante, são poucos os relatos sobre o uso de métricas relacionado à avaliação da qualidade do projeto quando em andamento. Em geral, a avaliação se faz após sua construção/implantação concentrando-se apenas nos aspectos funcionais. Neste caso, espera-se que a experiência do desenvolvedor resulte em um produto com qualidade.

Este estudo apresenta um ambiente computacional para a avaliação de projetos orientados a objetos baseado em métricas qualitativas e quantitativas. Obtidas a partir de elementos do próprio projeto tais métricas possibilitam uma contínua avaliação da qualidade do mesmo. A identificação antecipada de possíveis erros, durante as fases iniciais, evita que se propaguem para fases posteriores do ciclo de desenvolvimento como, por exemplo, a de geração de código.

Palavras-Chave: Engenharia de Software, Orientação a Objetos, Métricas, Qualidade de Software, UML, XMI.

Abstract

The software system development process applies different software engineering concepts and techniques to generate high quality products. Metrics are one among such techniques. Although the use of metrics is very important, there are few reports relating their use before the code generation phase. For this reason, the software developers need objective and valid measures for uses in the evaluation and improvement of product quality from the initial stages of the development.

This work presents a tool to evaluate object-oriented software design based on qualitative and quantitative metrics. The earlier the design fault detection occurs in a project life cycle, the less is the possibility of error propagation to later phases.

Keywords: software engineering, metrics, software quality UML, XMI.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

APOO - Ambiente para avaliação de projeto de software Orientado a Objeto
API – Aplication Program Interface
CASE - Computer Aided Software Engineering
GPL - General Public Licence
IDE – Integrated Development Environment
ISO - International Standard Organization
JVM - Java Virtual Machine
LOC – Lines Of Code
MDA - Model Driven Architecture
MOF - Meta object facility
MOO - Model object oriented
OMG - Object Management Group
OO - Orientação a objetos
OOD – Object Oriented Design
PF – Pontos de Função
SI - Sistema de Informação
SIBC - Sistema de Informação baseado em computador
SGA – Sistema de Gestão Acadêmica
SWSC - Space and Warning Systems Center
SQL - structured query language
UML - Unified Modeling Language
USAF – United States Air Force
XMI - XML Metadata Interchange
XML - Extensible Markup Language

CONTEÚDO

1. Introdução

1.1 Medições em Engenharia de Software

A produção de Sistemas de Informação (SI), com qualidade é hoje um imperativo no mercado de produtos de software. Mensurar esta qualidade torna-se uma questão central em qualquer sistema que será desenvolvido [GEN05].

Medidas de software são classificadas como diretas e indiretas. É denominada medição direta aquela que se refere a um único atributo auto contido, ou seja, independe dos demais. O tamanho do código em bytes, a quantidade de erros produzidos após implantação, o tempo de execução ou o tempo de desenvolvimento do projeto são exemplos de medições diretas. Medições indiretas, por sua vez, envolvem atributos correlacionados. A relação entre o número de erros e o número de linhas de código de um programa é um exemplo - esta relação pode sugerir, por exemplo, uma melhor modularização do software.

Importa também a distinção entre atributos internos e externos - interno é aquele medido puramente em termos do produto em si, como por exemplo, o número de métodos de uma classe. Diz-se externo o atributo mensurável a partir dos relacionamentos com as entidades de seu ambiente como, por exemplo, a facilidade de uso.

Gerentes de software, em geral, estão interessados nos aspectos externos do processo de desenvolvimento, como facilidade de manutenção e confiabilidade. Malgrado serem as métricas Orientadas a Objeto (OO) quase sempre baseadas em atributos internos é evidente a influência das mesmas nos aspectos externos de um projeto [HAR98].

O padrão ISO [ISO01], por exemplo, define medidas que se baseiam em atributos externos como funcionalidade, usabilidade, confiabilidade, eficiência, manutenibilidade, portabilidade e eficiência para avaliação da qualidade do produto.

Existe forte ligação entre qualidade interna e externa. Por exemplo, a qualidade das estruturas de dados e a eficiência dos algoritmos (aspectos internos definidos durante a fase de projeto e implementados na fase de construção), certamente influenciam a eficiência do produto final (aspecto externo).

O uso de métricas possibilita a avaliação da qualidade dos artefatos e modelos produzidos em cada fase do projeto - concepção, elaboração, construção e implantação [PAT03, WAK05]. Este fato permite que correções, quando necessárias, sejam feitas no decorrer do desenvolvimento

evitando o desperdício de tempo e de custos desnecessários que poderiam ocorrer após o produto entrar em operação. Tome como exemplo, o Space and Warning Systems Center (SWSC da Força Aérea Americana – USAF) utiliza métricas de complexidade, acoplamento, etc, aplicadas aos modelos de software partindo da hipótese de que, quanto melhor for a modelagem, melhor será o sistema produzido e menor será o número de falhas durante as fases posteriores do desenvolvimento [SWS96].

Mesmo assim, não existe um consenso para o que seja a qualidade do software, uma definição única aceita por toda comunidade de desenvolvedores. Neste trabalho adota-se a definição proposta em [IEE94]:

"1-Grau pelo qual um sistema componente ou processo atende aos requisitos especificados",
"2-Grau pelo qual um sistema componente ou processo atende às necessidades ou expectativas do cliente ou do usuário".

1.2 Motivação

É fato, na engenharia de software, que a qualidade de um sistema precisa ser assegurada desde a fase inicial do seu ciclo de vida [GEN05, LUI05]. Recentes paradigmas como Arquitetura Orientada a Modelos – MDA [OMG07] e Desenvolvimento Orientado a Modelos - MDD [OMG08] têm salientado a importância da construção dos modelos desde a especificação do projeto.

No ambiente profissional, o projeto de software assume extrema importância, pois, um projeto consistente certamente resulta em um produto de software com a devida qualidade, facilita e reduz o número de manutenções no produto e atende, principalmente, aos requisitos funcionais especificados pelo usuário.

Desta forma, são imperativas as pesquisas focadas em medidas de avaliação de projeto que vislumbrem melhorias de qualidade. Métricas de processo e de produto podem e devem servir como suporte às atividades de gerenciamento, acompanhamento de cronogramas, redução de custos e controle em geral. Além disso, permitem mensurar as atividades de análise, projeto, codificação, documentação e teste [BRI96].

É mister destacar que os termos medida e métrica são muitas vezes sinônimos. Este trabalho, entretanto, os diferencia. Medida se refere a um valor obtido para uma dada característica de um produto. Métrica é a característica em si a ser avaliada. Por exemplo, o tamanho do software é uma métrica enquanto o valor em bytes é uma medida.

Formalmente uma métrica é uma função que associa um número ou um símbolo a uma entidade para caracterizar um atributo ou um grupo de atributos, sendo o valor da métrica denominado de medida [PUR03].

A qualidade do software no paradigma OO vem sendo discutida e experimentada por diversos pesquisadores através do uso de diferentes modelos de métricas para a avaliação dos artefatos produzidos [BAN02, BRI96, CHI94, GEN01, GEN02, QUA07]. Conforme citado anteriormente, é fato que a medição da qualidade destes artefatos visa garantir que o produto tenha a qualidade mínima estabelecida na especificação dos requisitos, proporcionando economia de recursos no desenvolvimento e redução de manutenções corretivas do produto final.

Entretanto, um problema que surge ao se trabalhar com produtos de projeto é a coleta de dados e o cálculo das métricas diretamente a partir dos elementos produzidos. Poucas são, por exemplo, as ferramentas que possibilitam a coleta e a avaliação da qualidade dos mesmos - as existentes são proprietárias, de código fechado e, indisponíveis à maioria dos desenvolvedores [SDM06, LUI05].

Convém lembrar que verificar a qualidade do software nas fases iniciais e intermediárias do processo de desenvolvimento tem sido uma busca constante da comunidade acadêmica através de diferentes modelos baseados em métricas [GEN05, PAT03, WAK05].

1.3 Objetivo

A pesquisa apresentada neste trabalho identificou problemas recorrentes em projetos de software que usam a UML como linguagem de modelagem nas fases iniciais do desenvolvimento:

1. O projeto, quando decorrido o prazo estabelecido para seu término, não satisfaz todos os requisitos funcionais definidos na proposição inicial;
2. Quem desenvolve não tem como comparar a qualidade do seu projeto e corrigir erros porventura encontrados. Não há uma referência para tal comparação;
3. Existe dificuldade para avaliar com adequação a qualidade do software e a documentação do projeto em curto espaço de tempo.

Tais problemas levam à reflexão sobre a qualidade final do projeto e a forma como o mesmo é conduzido e avaliado. Uma possível alternativa é determinar e mensurar as principais características e qualidades do produto em desenvolvimento com o auxílio de ferramentas de modelagem que permitam a extração automática de diferentes métricas a partir dos artefatos gerados.

Dentre estes, certamente, o diagrama de classes oferece a possibilidade de avaliação das mais

diversas medidas envolvendo os principais elementos de projeto, tais como: o número de classes, de atributos, de métodos, de subclasses, profundidade da árvore de herança e outros.

Neste trabalho implementa-se uma ferramenta de avaliação da qualidade de projetos de software OO baseado em métricas de classe, denominado APOO. Esta recebe como entrada um modelo de classes UML no formato XMI extraindo diferentes métricas orientadas a objeto. Tais métricas são comparadas a valores catalogados sejam, da literatura, às de um projeto padrão pré-estabelecido, ou, ainda, às obtidas de um conjunto de projetos relacionados, fornecendo um indicativo da qualidade do produto em questão.

1.4 Organização

Este trabalho está organizado em 5 capítulos. No Capítulo 2 são abordados os principais trabalhos relacionados com o tema. O Capítulo 3 descreve a arquitetura do ambiente APOO, as bases para construção da ferramenta e o modelo de avaliação de projetos orientados a objetos centrados em UML. O Capítulo 4 apresenta estudos de casos para teste e validação do ambiente proposto. O Capítulo 5 conclui esta dissertação relacionando os trabalhos futuros.

2. Revisão da literatura

2.1. Introdução

Grande parte dos custos dos sistemas de informação está, atualmente, no desenvolvimento do software. Enquanto o custo do hardware decresce rapidamente, a produtividade em software melhora lentamente fazendo com que o custo deste cresça em relação àquele.

Um dos pontos importantes nos custos destes sistemas é a manutenção. Diferentes soluções podem ser adotadas para o problema como, por exemplo, o controle através da utilização de métricas de software durante a fase de desenvolvimento que são utilizadas como indicadores de qualidade auxiliando a identificação de potenciais problemas [BRE07].

Uma das primeiras atividades de desenvolvimento é certamente a construção de modelos que orientem o projeto e a construção dos respectivos artefatos. Entretanto, para o entendimento do problema real são exigidas diferentes técnicas que conduzem à elaboração de diferentes modelos [ATK03].

A idéia de modelagem está diretamente relacionada à necessidade de construção de produtos com qualidade. Esta qualidade é, em geral, medida por características visíveis pelo usuário (qualidade externa) que por sua vez depende de detalhes de projeto e construção (qualidade interna). A avaliação desta qualidade é um dos aspectos estudados pela indústria de software e pela academia como apresentado ao longo deste capítulo.

2.2. Modelagem de software

Por modelo de software, entende-se a abstração construída pelo arquiteto de software e por engenheiro de software visando mostrar uma faceta específica do produto em desenvolvimento. Pode-se, por exemplo, abstrair tanto os aspectos funcionais do sistema como a arquitetura operacional do mesmo. O principal objetivo é a simplificação que permite um entendimento do todo através da análise das partes.

A modelagem de software, ao longo dos anos, tem lançado mão de diferentes abordagens batizadas por alguns autores como modelo ou paradigma. Sendo denominados modelos estruturados e orientados a objetos de acordo com as características da linguagem de programação usada na construção do software.

Paradigmas incorporando novas características, principalmente em função da evolução

tecnológica do hardware e do software básico, têm surgido. Mais recentemente paradigmas como MDD - Model Driven Development [ATK03] e MDA - Model Driven Architecture [OMG06] têm enfatizado a necessidade da correta escolha de modelos desde o início do ciclo de vida de desenvolvimento.

2.3. Modelagem Orientada a Objeto

A modelagem orientada a objeto (MOO) vem se mostrando popular nos ambientes de desenvolvimento de software [NAS98a] sendo que o principal bloco de construção é a classe de objetos e não o algoritmo como nos programas construídos de forma tradicional segundo o paradigma procedimental.

Entende-se por objeto o elemento estruturado a partir do vocabulário do espaço do problema e/ou do espaço da solução podendo representar elementos concretos ou abstratos.

Por classe, a descrição de um conjunto de objetos onde todos têm os mesmos atributos e as mesmas operações. Um atributo é uma abstração que define uma propriedade de um objeto podendo assumir diferentes valores. Uma operação é uma seqüência de declarações e comandos executáveis encapsulados dentro da classe como um pequeno programa independente para manipular os valores dos atributos [SCH99].

2.4. Ferramentas para Modelagem

A construção de modelos, representando as diferentes visões do software é, normalmente desenvolvida com o auxílio de ferramentas CASE. O objetivo destas é melhorar a produtividade das tarefas envolvidas.

Diversas ferramentas estão disponíveis, sejam estas de uso livre como ArgoUML, Umbrello, JUDE, OMONDO, ou proprietárias como Rose Enterprise e Artisan Real-Time Studio [ARG07, WIK07, DMO07].

O principal papel das ferramentas CASE e dos ambientes integrados de desenvolvimento (IDE’s) é possibilitar ao projetista a especificação das características desejadas do modelo de forma rápida e interativa.

Através de descrições textuais ou gráficas elaboradas nos ambientes obtêm-se os scripts de construção, por exemplo, de tabelas em um banco de dados, protótipos de interface com usuário ou mesmo a geração de parte do código. Recentemente têm surgido ambientes IDE, de uso livre tais como Eclipse, Netbeans e outros que incorporam plugins para modelagem de software.

Embora necessárias tais ferramentas apresentam inconvenientes: as produzidas por grandes empresas de software, enquanto completas e cobrindo todo o ciclo de desenvolvimento, desde especificação de requisitos à geração de código, são complexas e caras. Outras, de uso livre, suprimem algumas facilidades que o desenvolvedor necessita como, por exemplo, importar ou exportar modelos para diferentes ferramentas ou mesmo suportar diferentes diagramas UML.

Um importante aspecto a ser considerado em quaisquer ferramentas de modelagem é o formato utilizado pelas mesmas para armazenar internamente ou para importar e exportar modelos construídos durante a fase de projeto. É indispensável à manipulação de um mesmo modelo, em diferentes ferramentas, sem nenhum tipo de restrição a fim de simplificar a manutenção, conversão e adaptação do software [LUI05, PAT03, SEI08].

Destarte, o intercâmbio de modelos é um aspecto fundamental no desenvolvimento de aplicações. Este é um problema que interessa diretamente às indústrias de software, pois a maioria das ferramentas não permite tal intercâmbio.

Uma solução para o problema é o uso de padrões como, por exemplo, o XMI para intercâmbio de metadados de modelos baseados em MOF (Meta Object Facility), na UML e em XML (Extended Markup Language) [OMG02].

2.5. A Qualidade do Software

É certo que o software construído hoje com a melhor técnica vigente, com o passar dos anos, será conhecido como software legado e poderá estar fora dos padrões futuros de qualidade. Este é um problema inevitável resultante da evolução tecnológica que irá exigir do mesmo ser então adaptado, aperfeiçoado, ampliado ou redesenhado.

A crescente demanda por software tem criado como conseqüência, necessidade de pessoal especializado em engenharia de software, qualidade e gerenciamento de desenvolvimento que adote métodos e ferramentas adequados. Desta forma, a qualidade ganha cada vez mais importância no processo de desenvolvimento [BHA05].

Para os profissionais da área de software, como os analistas, programadores e engenheiros, interessam tanto as características externas do produto, quanto, e principalmente, as internas do mesmo - como os padrões arquiteturais de desenvolvimento, os algoritmos e as estruturas de dados.

Importante, também, é a qualidade do processo no que se refere às atividades que afetam o produto final. Fato este motivador de diferentes modelos de qualidade de processos como o Modelo de Maturidade e Capacidade (CMM) [SEI07] e ISO9000 [ISO01].

Nota-se, portanto, que avaliar a qualidade de software não é uma tarefa simples devido à quantidade de fatores envolvidos. Entretanto, quanto melhor a qualidade do projeto no presente maior será a facilidade de se fazer uma nova engenharia neste produto a qualquer momento.

É certo que no ambiente de negócio a qualidade está intimamente associada à relação custo-benefício, ou seja, a relação entre a qualidade técnica do software e o seu valor comercial. A qualidade dos produtos desenvolvidos está associada à imagem da empresa podendo decidir seu sucesso no mercado. Segundo Glass [GLA98] tal qualidade pode ser expressa pela equação:

"Satisfação do usuário = produto adequado + qualidade máxima + entrega do produto dentro do prazo e dentro do cronograma".

Da mesma forma que na indústria de bens duráveis a qualidade do produto é avaliada através de ensaios, testes e medições associados ao controle estatístico, na Engenharia de Software busca-se definir novas formas de medir e avaliar a qualidade do software durante e após sua construção através de métricas. A seção seguinte descreve alguns dos principais modelos encontrados na literatura.

2.6. Modelos de Métricas

Neste trabalho utiliza-se a definição de métrica estabelecida pelo IEEE Standard Glossary of Software Engineering Terms [IEE94]:

"Medida quantitativa do grau em que um sistema, componente ou processo possui determinado atributo"

Note que esta definição se aplica a toda métrica, independente de sua classificação como interna ou externa, de produto ou de processo. Desta forma, o principal objetivo de um conjunto de métricas é permitir a avaliação de diferentes dimensões do software.

Estas dimensões possibilitam medir e quantificar itens de qualidade como acoplamento, coesão, complexidade, herança, reutilização e tamanho que poderão fornecer indicativos da existência de falhas de projeto. Modelos de métricas têm sido propostos na tentativa buscar atributos do processo e do produto de software que permitam definir e mensurar sua qualidade tanto de forma direta quanto indireta. As seções seguintes descrevem os principais modelos de métricas.

2.6.1. Modelo de Chidamber e Kemerer (CK)

O modelo proposto por Chidamber e Kemerer, denominado MOOSE (Metrics for Object Oriented Software Engineering), é considerado precursor e serve como referência para diversos estudos sobre métricas OO [CHI94]. Validado empiricamente [BAS96, OLA07], é utilizado na obtenção de indicadores de qualidade e de falhas em projetos de classes.

Basili, Briand e Melo coletaram dados de desenvolvimento de oito projetos de um sistema de gerenciamento de informações de porte médio (duração de 4 meses, em linguagem C++, oito grupos de três estudantes de graduação como desenvolvedores, usando banco de dados relacional e técnicas de desenvolvimento OO) e versando sobre as mesmas funcionalidades.

Através de testes estatísticos de aceitação verificou-se que as métricas propostas poderiam ser usadas como indicadores de qualidade, com a recomendação de serem elaboradas mais pesquisas e refinamentos dos estudos; o que vem ocorrendo até a presente data [BAS96]. A Tabela 1 apresenta as métricas deste modelo.

Item	*Métrica*	Descrição
1	WMC	Weighted Methods per Class (métodos ponderados por classe). É a soma das complexidades de todos os métodos de uma classe. WMC corresponde ao número de métodos da classe quando todas as complexidades são assumidas como unitárias.
2	RFC	Response For a Class (resposta de uma classe). É o conjunto de métodos que podem ser potencialmente executados em resposta a uma mensagem recebida.
3	LCOM	Lack of COhesion in Methods (falta de coesão em métodos). É o número de métodos que tem acesso a um ou mais atributos da mesma classe (variáveis de instância).
4	CBO	Coupling Between Objects Class (acoplamento entre classes de objeto).
5	DIT	Depth of Inheritance Tree (profundidade da árvore de herança). É o comprimento máximo do caminho de um nó até a raiz da árvore de herança.
6	NOC	Number Of Children (número de filhos de uma classe). É a contagem das classes diretamente subordinadas a uma classe ancestral.

Tabela 1 – Métricas Chidamber e Kemerer.

2.6.2. Modelo MOOD

O modelo MOOD (Metrics for Object Oriented Design) é composto por seis métricas. A ênfase está nos aspectos de projeto como herança, encapsulamento e acoplamento (Brito e Carapuça, 1994 in El Wakil, 2004). A Tabela 2 apresenta as métricas propostas e uma descrição segundo [HAR97].

Item	Métrica	Definição
1	MHF	Method Hiding Factor (fator de ocultamento de método). É a razão entre o número de métodos ocultos (métodos protegidos e métodos privados) e o total de métodos. É proposta como medida de encapsulamento.
2	AHF	Attribute Hiding Factor (fator de ocultamento de atributo). É a razão entre o número de atributos ocultos (atributos protegidos e privados) e o total de atributos.
3	MIF	Method Inheritance Factor (fator de herança de métodos). É a razão entre o número de métodos herdados e o total de métodos.
4	PF	Polymorphism Factor (Fator de polimorfismo). É a razão entre o número de métodos herdados redefinidos e o número total de métodos possíveis de redefinição no sistema.
5	CF	Coupling Factor (Fator de acoplamento). É o número de comunicações entre classes. Cada comunicação entre uma classe cliente e uma classe servidora é contada.
6	AIF	Attribute Inheritance Factor (fator de herança de atributos). É a razão entre o número de atributos herdados e o total de atributos.

Tabela 2 – Métricas de Brito e Carapuça.

2.6.3. Modelo de Lorenz e Kidd (LK)

Lorenz e Kidd propõem um conjunto de métricas para avaliar as características do projeto de software OO, agrupadas em tamanho, herança, aspectos internos e externos da classe [LOR94].

Basicamente, métricas orientadas a tamanho focalizam a contagem de atributos e métodos para uma classe individual e valores médios para todo o sistema. As relativas à herança focalizam o modo pelo qual, operações são reutilizadas ao longo da hierarquia de classes. Métricas para aspectos internos tratam da coesão e as de aspectos externos examinam acoplamento e reuso.

Embora criticado por não fazer parte de um modelo propriamente dito, as métricas propostas apresentam uma definição clara, além de serem facilmente coletadas e aplicadas nas fases iniciais de um projeto [WAK04].

A Tabela 3 relaciona nove das métricas deste modelo conforme apresentado em [HAR97]. Onde:

- HNL - nível de profundidade na hierarquia. Corresponde à métrica DIT em [CHI94].
- TMP - número de parâmetros dos métodos da classe.
- TNO - número total de métodos da classe.

Item	Classificação	Métrica	Definição
1	Tamanho	PM	Number of Public Methods (Número de métodos públicos da classe).
2		NM	Number of Methods (Número de métodos da classe. Inclui métodos públicos, privados e protegidos).
3		NV	Number of Variables per class (Número de variáveis de instância da classe. Inclui variáveis públicas, privadas e protegidas).
4		NPV	Number of Public Variables per class (Número de variáveis públicas da classe).
5	Herança	NMO	Number of Methods Overridden by a subclass (Número de métodos redefinidos por uma subclasse).
6		NMI	Number of Methods Inherited by a subclass (Número de métodos herdados por uma subclasse).
7		NMA	Number of Methods Added by a subclass (Número de métodos adicionados pela subclasse).
8		SIX	Specialization Index (Índice de especialização = (NMO*HNL)/TNO).
9	Interno	APPM	Average Parameter per Method (Número médio de parâmetros por método = (TMP/TNO)).

Tabela 3 – Métricas de Lorenz e Kidd.

2.6.4. Modelo de Métricas Genero et al

Genero, Piattini e Calero estudam diversos modelos de métricas a partir dos quais derivam um modelo próprio [GEN00, GEN02]. Dividem, conforme Tabela 4 e 5, as métricas em dois grandes grupos: as obtidas no contexto de um diagrama de classes (aplicável ao diagrama como um todo) e, as observadas no escopo da classe (aplicáveis a uma classe do modelo).

Item	Métrica	Definição
01	Nassoc	Number of Association (Total de relacionamentos de associação no diagrama de classe).
02	Nagg	Number of Aggregation (Total de relacionamentos de agregação no diagrama de classe).
03	Ndep	Number of Dependencies (Total de relacionamentos de dependência no diagrama de classe).
04	Ngen	Number of Generalization (Total de relacionamentos de generalização no diagrama classe). Cada par pai-filho é contado.
05	NgenH	Number of Generalization Hierarchies (Total de hierarquias de generalização no diagrama de classe).
06	NaggH	Number of Aggregation Hierarchies (Total de hierarquias de agregação no diagrama de classe).
07	MaxDIT	Maximum DIT (Máximo dos DIT, conforme [CHI94], do diagrama de classe). É o maior dos DIT das classes do diagrama.
08	MaxHAgg	Maximum HAgg (Máximo HAgg do diagrama). HAgg está na Tabela 5.

Tabela 4 – Métricas no contexto de diagrama de classes.

Segundo os autores a crescente demanda por qualidade de software tem colocado o tema *qualidade* como um *diferencial* entre produtos. Por esta razão os desenvolvedores necessitam

medidas objetivas para utilizarem na avaliação e melhoria do produto a partir dos estágios iniciais do desenvolvimento.

Na abordagem proposta, o diagrama de classes é de grande importância, pois serve de base para todo o trabalho de projeto e implementação. Além disso, focar na qualidade do diagrama de classes contribui significativamente para a melhoria dos sistemas [GEN05].

Item	Métrica	Definição
1	NAssocC	Number of Association (Total de relacionamentos de associação que uma classe tem com outra ou consigo mesma).
2	Hagg	Heigth of a class within aggregation hierarchy (é o comprimento do maior caminho de uma classe até suas folhas).
3	NODP	Number of Direct Parts (total de "partes diretas" que compõem uma classe composta).
4	NP	Number of Parts (total de classes *parte*[1], diretas ou indiretas, de uma classe *todo*). Considera a relação *todo-parte*.
5	NW	Number of Wholes (total de classes *todo*, diretas ou indiretas, de uma classe *parte*). Considera a relação todo-parte.
6	Magg	Multiple Aggregation (total de classes *todo* diretas do qual uma classe é parte-de, dentro de uma hierarquia de agregação).
7	NdepIn	Number of Dependencies In (total de classes que dependem de uma classe).
8	NDepOut	Number of Dependencies Out (total de classes que tem classes dependentes).

Tabela 5 – Métricas observadas no escopo de classe.

2.6.5. Modelo QMOOD

QMOOD (*Quality Model for Object Oriented Design*) é o modelo proposto por Bansiya e Davis [BAN02]. O enfoque está na avaliação das propriedades estruturais e comportamentais dos projetos de classes tais como encapsulamento, modularização e coesão, conforme apresentado na Tabela 6.

2.7. Aplicações

Diversas ferramentas e pesquisas, que incorporam métricas para avaliar a qualidade do software, têm surgido. Por exemplo, a ferramenta OOMeter desenvolvida para processar código fonte Java e C#, assim como, diagramas UML efetuando o cálculo de métricas Chidamber [ALG05]. Outro

[1] O relacionamento *todo-parte* é considerado em diferentes variações. Este relacionamento expressa uma dependência entre uma classe e outra de forma diferente de uma associação e considera as relações diretas ou indiretas (Tabela 5). Um exemplo deste relacionamento é: Polígono-Segmento de reta onde "polígono" é *todo* e segmento é *parte*.

exemplo é a utilização destas para a avaliação da qualidade de sistemas multimídia, industriais e editores de música [DAR05].

Item	Métrica	Definição
1	DAM	Data Access Metric (Métrica de Acesso a Dados). Razão entre o total de atributos privados e protegidos e o total de todos os atributos da classe.
2	DCC	Direct Class Coupling (Acoplamento Direto da Classe). Contagem do número de diferentes classes a que uma classe está diretamente relacionada.
3	CAM	Cohesion Among Methods of Class. (Coesão Entre os Métodos da Classe). Calcula a ligação entre os métodos da classe baseado na lista de parâmetros dos métodos.
4	MOA	Measure Of Aggregation (Medida de Agregação). Número de declarações de dados cujos tipos são classes definidas pelo usuário.
5	MFA	Measure of Functional Abstraction. (Medida de Abstração Funcional). É o razão entre o número de métodos herdados pela classe e o número total de métodos accessíveis pelos métodos membro da classe.
6	DSC	Design Size in Classes (Tamanho do projeto em classes). É o número total de classes no projeto.
7	NOH	Number Of Hierarchies (Número de hierarquias). É o número de hierarquias de classes no projeto.
8	ANA	Average Number of Ancestors (Número Médio de Ancestrais). Considera o número de classes em todos os caminhos da classe raiz até às classes folhas.
9	NOP	Number Of Polymorphic Methods (Número de Métodos Polimórficos). Contagem do número de métodos que podem exibir comportamento polimórfico.
10	CIS	Class Interface Size (Tamanho da Interface da Classe). É o número de métodos públicos na classe.
11	NOM	Number Of Methods (Número de Métodos). É o número de métodos da classe.

Tabela 6 – Métricas de Bansiya e Davis.

Convém ressaltar que a maioria das ferramentas disponíveis efetua as medições processando códigos fontes escritos, principalmente, na linguagem C++ ou Java (Tabela 7). Apesar dos avanços, os autores consideram a relevância de novos estudos para a avaliação de métricas a partir de modelos ou artefatos gerados nas fases iniciais de projeto [DAR05, GEN01].

Ferramenta	Vendedor	Linguagem	URL	Métricas cobertas [CHI94]
Krakatau Metrics	Power Software	C++ e Java	www.powersoftware.com	TODAS
Jstyle	Codework	Java	www.codework.com	WMC, DIT, RFC, LCOM
Project Analyzer	Aivosto	Visual Basic	www.aivosto.com	TODAS
RSMestrics	M Squared Technologies	C++ e Java	www.msquaredtechnologies.com	DIT, NOC
SDMetrics	SDMetrics	C++ e Java	www.sdmetrics.com	WMC, DIT, NOC, CBO, RFC
Software Metrics	McCabe & Associates	C++ e Java	www.mccabe.com	WMC, DIT, NOC, RFC, LCOM
Understand	Scientific Toolworks	C++ e Java	www.scitools.com	TODAS

Tabela 7 - Ferramentas Comerciais para Coleta de Métricas.

In et al [KIM03] apresentam a ferramenta OSMAT (Ontology based Software Metrics Analysis Tool) que, a partir de diagramas UML, disponibiliza o cálculo de diversas métricas classificadas em cinco grupos conforme Tabela 8.

Grupo	Abreviatura	Nome da métrica
Primitivo	TNC	Total Number of Class (Nº total de classes)
	TNO	Total Number of Operation (Nº total de operações)
	TNP	Total Number of Parameters (Nº total de parâmetros)
	TNCA	Total Number of Class Attributes (Nº total de atributos de classe)
	TNIR	Total Number of Inheritance relationships (Nº total de relacionamentos de herança)
	TNRR	Total Number of Realization Relationships (Nº total de relacionamentos de realização).
	TNUR	Total Number of Use Relationships (Nº total de relacionamentos de uso)
	TNA	Total Number of Associations (Nº total de associações da classe).
	TNR	Total Number of Roles (Nº total de papeis)
Tendência de falha	WMC	Weighted Method per Class (Métodos ponderados por classe)
	NOC	Number of Children per Class (Nº de filhos por classe)
	DIT	Depth of Inheritance Tree (Profundidade da árvore de herança)
Medição de qualidade	MHF	Method Hiding Factor (Fator de ocultamento de método)
	AHF	Attribute Hiding Factor (Fator de ocultamento de atributo)
	MIF	Method Inheritance Factor (Fator de herança de método)
	AIF	Attribute Inheritance Factor (Fator de herança de atributo)
Acoplamento	OLC	Object Level Coupling (Nível de acoplamento de objetos)
	CLC	Class Level Coupling (Nível de acoplamento de classes)
	PLC	Package Level Coupling (Nível de acoplamento de pacotes)
Caso de uso	NOA	Number of Actor (Nº de atores)
	NOUC	Number of Use Cases (Nº de casos de uso)
	NOUCA	Number of Use Cases per Actor (Nº de casos de uso por ator)

Tabela 8 - Métricas avaliadas em OSMAT.

Cada grupo reúne características que indicam o tamanho do projeto (número de classes, métodos e número de casos de uso), estrutura (número de filhos por classe ou profundidade da árvore de herança), qualidade (em termos de ocultamento ou herança de atributos e métodos) e acoplamento.

A ferramenta recebe como entrada modelos no formato mdl (formato proprietário - Rational Rose©) produzindo saídas no formato XML. Pode-se observar que a ferramenta proposta se baseia principalmente nas métricas apresentadas em [CHI94, LOR94, BRI96, BAN02].

Convém notar que alguns estudos procuram mostrar que inexistem dados estatísticos suficientes para provar o significado do valor de uma métrica – este fato é apenas um indicativo que grandes diferenças, entre valores, devam ser investigadas [NAS98a]. Por exemplo a métrica DIT mede a profundidade de uma classes na árvore de herança . A classe raiz tem valor de DIT igual a 0

(zero). Um valor situado entre 1 e 4 adotado como ideal está baseado nos diferentes estudos empíricos existentes como em [BAS96]. Uma classe com profundidade alta pode ter um grande numero de métodos herdados e aumentar a complexidade de entendimento do seu comportamento embora seu potencial de reutilização em outros sistemas seja aumentado. Isto exige maior esforço de manutenção já que funcionalidades da classe poderão estar dispersas em outras classes da hierarquia. Além disto uma alteração na classe superior da hierarquia requer o teste de todas as classes descendentes. Valores sugeridos para as métricas de classe são mostrados na Tabela 9.

PMR	Public Method Ratio (razão entre métodos públicos). É o total de métodos públicos do modelo dividido pelo total de métodos (públicos, privados, protegidos). $5\% \leq PMR \leq 50\%$
CR	Class Responsability (responsabilidade da classe). CR = (PCC + POC) dividido por (2 x NOM) onde PCC = número de métodos que implementam pré-condições e POC = número de métodos que implementam pós-condições. $20\% \leq CR \leq 75\%$.
CBO	$1 \leq CBO \leq 4$
DIT	$1 \leq DIT \leq 4$
NOC	$1 \leq NOC \leq 4$
NOM	$3 \leq NOM \leq 7$. NOM = número médio de métodos por classe.
NOA	$2 \leq NOA \leq 5$. NOA = número médio de atributos por classe.

Tabela 9 – Valores Sugeridos.

Por fim, cita-se o trabalho realizado na NASA-SATC [NAS98b] que analisa dados de métricas para diferentes linguagens de programação (Tabela 10). São utilizados documentos de requisitos de projeto, código de programa, planos de teste e outros.

Métrica	Valor limite	Comentários
Número de métodos (NOM)	≤ 20: desejável, ≤ 40 é aceitável.	Inclui métodos construtores e métodos destrutores. Ideal: ~ 10.
Métodos ponderados por classe (WMC)	≤ 100: aceitável.	Fornece uma melhor idéia da complexidade e dá um indicativo do tempo e esforço para desenvolver e manter o software
Resposta para a classe (RFC)	≤ 100	Quanto maior o número de métodos que pode ser chamado de uma classe maior a complexidade da classe exigindo maior esforço de testes e depuração
RFC/NOM	≤ 5 (C++) e ≤ 10 (Java)	Esta métrica faz um bom trabalho de selecionar as classes que necessitam teste exaustivo. Como em Java tudo é classe os valores são mais altos.
Acoplamento entre objetos (CBO)	≤ 5	Valores altos indicam classes com dificuldade de entendimento, reuso e manutenção. Baixo valor torna a classe mais fácil de entender e menos sujeita a erros.
Profundidade na árvore (DIT)	Entre 2 e 3	Maior a profundidade maior o grau de reuso. Abaixo de 2 pode indicar um projeto que não aproveita a herança. Acima de 5 aumenta muito a complexidade.

Tabela 10 - Valores de Métricas – NASA-SATC.

2.8. Métricas e a Avaliação de Projetos de Software

Interessa às empresas definir programas de métricas como forma de compreender e modelar a engenharia de software em relação aos processos e produtos, e também como meio de auxílio ao gerenciamento e obtenção de melhorias na aplicação dos mesmos [SCO04].

Há muito se reconhece que a melhoria do processo de software requer medições. A experiência tem mostrado que são poucas as métricas universais e que, as ferramentas mais eficientes são especializadas, seja em aspectos do processo ou do domínio do software [DAR05].

Estudos têm documentado a relação entre métricas OO e gerenciamento de projeto englobando produtividade, esforço de projeto, reutilização, defeitos, falhas, facilidade de manutenção e redução de custo [CHI98**].** Destes resultam as seguintes observações:

- Métricas OO têm sido aplicadas com sucesso em vários domínios e linguagens de programação;
- Há uma forte relação com fatores de qualidade tais como custos, defeitos, reutilização e facilidade de manutenção - relações estas que ultrapassam o aspecto de tamanho do software;
- Um conjunto de métricas envolvendo tamanho (medido por SLOC ou por variações de WMC), acoplamento (medido por CBO ou RFC) e coesão (medida por LCOM) são úteis em modelos de previsão;
- Herança (medida por DIT ou NOC) é aparentemente pouco usada e sua relação com o projeto é menos precisa, e
- Espera-se que relações exatas entre métricas e previsões de interesse gerencial sejam calibradas pelas influências locais do ambiente onde ocorre o desenvolvimento.

Não é surpresa para El Wakil [WAK05], que, o principal pré-requisito para se obter um sistema de informação com qualidade é avaliá-la continua e antecipadamente dentro do processo de desenvolvimento evitando, assim, omissões ou falhas e desvios em relação ao desejado.

Outro aspecto importante do projeto é a adequação de métricas relacionadas ao diagrama de classe para a avaliação de atributos de qualidade [GEN05] permitindo ao desenvolvedor de software:

- Identificar pontos fracos quando o custo ainda é baixo evitando correção em fases posteriores – quando o custo pode, inclusive, se tornar proibitivo;
- Escolher alternativas de projeto de forma objetiva e;

- Fazer previsão de características da qualidade externa tais como facilidade de manutenção, reutilização, bem como melhor alocação de recursos com base em previsões.

Embora o uso de métricas permita realizar medidas objetivas, evitando em muitos casos, avaliações subjetivas, existem críticas quanto à validade de algumas propostas. Mayer e Hall [MAY99] criticam o uso de métricas e a terminologia utilizada, como propostas no modelo de Chidamber e Kemerer [CHI94]. Afirmam que o termo complexidade, por exemplo, é utilizado sem uma definição clara em relação aos sistemas OO. Afirmam, também, que muitos dos estudos ligados ao uso de métricas não reconhecem os aspectos particulares de projetos - estão simplesmente assumindo uma extensão das técnicas da programação estruturada.

Voas afirma que: ***"Métricas fornecem uma orientação; elas não são recipientes absolutos para alcance da qualidade"*** [VOA99]. O autor acredita que uma informação numérica indicando se um software é bom ou ruim constitui um mito da qualidade de software, uma vez que uma métrica não mede aspectos semânticos do projeto. Entretanto, concorda que uma métrica é capaz de avaliar o processo de desenvolvimento da organização permitindo assim melhorias.

2.9. Considerações Finais

O número de métricas OO é extenso o que demonstra uma preocupação dos pesquisadores com o assunto. A Tabela 11 apresenta uma lista de métricas propostas [CHI94, LOR94, BRI96, GEN05]. Uma relação mais abrangente pode ser vista em Xenos et al. [XEN00].

Algumas das métricas propostas foram criadas antes da existência de linguagens para modelagem e se baseiam exclusivamente em código de programa. Este fato impede o cálculo das métricas a partir de modelos de classes representados, por exemplo, em UML, uma vez que o código fonte inexiste.

Importa destacar que diversos autores propõem métricas conceitualmente iguais com diferentes nomes como, por exemplo, DIT [CHI94] e MaxDIT [GEN02]. Outro aspecto a considerar: no campo da medição de software OO a utilização de medidas aplicadas nos estágios iniciais do processo de desenvolvimento é ainda restrita - os trabalhos são normalmente focados na medição baseada no código do programa e não em métricas derivadas de elementos de projeto.

Além disso, embora o uso de métricas para a avaliação de projetos de software seja comum, um problema ainda persistente é a escassez de ferramentas que automatizem o processo de coleta e avaliação de dados a partir de modelos descritos em UML [GEN05].

Com o objetivo de abordar este problema o capítulo seguinte descreve um ambiente de apoio

para utilização de métricas na avaliação da qualidade de projetos de software.

Nº	Métrica	Descrição	Referência
1.	CBO	Acoplamento entre classes	[CHI94]
2.	DIT	Profundidade da árvore de herança	[CHI94]
3.	LCOM	Falta de coesão em métodos	[CHI94]
4.	NOC	Número de filhos por classe	[CHI94]
5.	RFC	Resposta para classe	[CHI94]
6.	WMC	Métodos ponderados por classe	[CHI94]
7.	Nagg	Número de agregações no diagrama de classe	[GEN02]
8.	Nassoc	Número de associações no diagrama de classe.	[GEN02]
9.	Ndep	Número de dependências no diagrama de classe.	[GEN02]
10	Ngen	Número de generalizações no diagrama classe.	[GEN02]
11	NgenH	Total de hierarquias de generalização no diagrama de classe.	[GEN02]
12	NaggH	Total de hierarquias de agregação no diagrama.	[GEN02]
13	MaxDIT	Máximo entre os DIT [CHI94] do diagrama de classe.	[GEN02]
14	MaxHAgg	Máximo comprimento dos caminhos desde a classe de agregação até suas folhas no diagrama.	[GEN02]
15	APPM	Número médio de parâmetros por método (TMP/TNO)	[LOR94]
16	NCM	Número de métodos da classe (métodos globais)	[LOR94]
17	NCV	Número de variáveis da classe	[LOR94]
18	NIM	Número de métodos da classe	[LOR94]
19	NIV	Número de variáveis de instância da classe	[LOR94]
20	NMA	Número de métodos na subclasse	[LOR94]
21	NMI	Número de métodos herdados pela subclasse	[LOR94]
22	NMO	Número de métodos sobrecarregados da subclasse	[LOR94]
23	PIM	Número de métodos públicos da instância	[LOR94]
24	SIX	Índice de especialização (NMO*HNL)/TNO	[LOR94]
25	AHF	Fator de ocultamento de atributo	[BRI96]
26	AIF	Fator de herança de atributo	[BRI96]
27	CF	Fator de acoplamento	[BRI96]
28	CLF	Fator de clusterização	[BRI96]
29	MHF	Fator de ocultamento de método	[BRI96]
30	MIF	Fator de herança de método	[BRI96]
31	PF	Fator de polimorfismo	[BRI96]
32	RF	Fator de reutilização	[BRI96]
33	DAM	Acesso a dados: razão entre o total de atributos privados e protegidos e o total de atributos da classe	[BAN02]
34	DSC	Contagem do número de classes no projeto	[BAN02]
35	NOH	Contagem do número de hierarquias de classes no projeto	[BAN02]

Tabela 11 - Métricas OO.

3. O Ambiente APOO

3.1. Introdução

Como discutido anteriormente, o diagrama de classes é o artefato base para todo trabalho de projeto e desenvolvimento de software. A avaliação antecipada de sua qualidade tende a auxiliar o projetista na construção das aplicações evitando revisões desnecessárias em etapas posteriores.

A disponibilidade de medidas de qualidade do software de forma objetiva evita polêmicas sobre questões subjetivas ligadas ao processo de avaliação. Além disso, como a tarefa de manutenção consome e continuará a consumir a maior parte dos recursos do ciclo de vida, este fato reforça ainda mais a preocupação com a qualidade [GEN05].

Este capítulo descreve os aspectos relacionados à construção do ambiente APOO centrado na importância da modelagem e medição tanto para o ambiente profissional quanto acadêmico.

Há de se considerar o fato que muitas das ferramentas existentes são aplicáveis sobre código fonte e somente geram informações úteis no estágio final do ciclo de desenvolvimento após implementações.

Certamente, o uso de um ambiente de extração e avaliação de métricas propicia aos desenvolvedores os seguintes benefícios:

- Incentiva a cultura de medição no processo de desenvolvimento de software;
- Colabora para desenvolvimento de projetos e construção de aplicações com qualidade, e
- Contribui para o desenvolvimento de novas ferramentas automatizadas que permitam a avaliação da qualidade do projeto em suas fases iniciais.

As seções seguintes descrevem a arquitetura do ambiente proposto.

3.2. Métricas APOO

Na atual versão do ambiente são consideradas apenas as métricas que podem ser obtidas nos estágios iniciais da modelagem, ou seja, medições relacionadas ao diagrama de classes da UML. Dentre as diversas categorias apresentadas no Capítulo 2 são tratadas: tamanho [LOR94], encapsulamento [BRI96], herança [LOR94; BRI96], acoplamento [CHI94, BRI96], relacionamento [GEN02, GEN05] e aspectos internos [LOR94].

Estas categorias avaliam as seguintes características de um projeto: número de classes, de

subclasses, métodos, e atributos. É contemplado o número de descendentes de uma classe (subclasses), a profundidade da árvore de herança, o acoplamento entre as classes, o ocultamento de métodos e atributos e outras. As métricas avaliadas são relacionadas na Tabela 12. O Apêndice A1 as descreve detalhadamente.

Item	Sigla	Nome
1.	NC	Número de classes
2.	NCPu	Número de classes públicas
3.	NCPv	Número de classes privadas
4.	NCPt	Número de classes protegidas
5.	NCPc	Número de classes de pacote
6.	NCIn	Número de classes de interface
7.	NCAb	Número de classes abstratas
8.	NCAs	Número de classes de associação
9.	NRAss	Número de relacionamentos de associação
10.	NRAgr	Número de relacionamentos de agregação
11.	NRCom	Número de relacionamentos de composição
12.	NRDep	Número de relacionamentos de dependência
13.	NRHer	Número de relacionamentos de herança
14.	NRImp	Número de relacionamentos de implementação
15.	NA	Número de atributos
16.	NAPu	Número de atributos públicos
17.	NAPv	Número de atributos privados
18.	NAPt	Número de atributos protegidos
19.	NAPc	Número de atributos de pacote
20.	NM	Número de métodos
21.	NMPu	Número de métodos públicos
22.	NMPv	Número de métodos privados
23.	NMPt	Número de métodos protegidos
24.	NMAb	Número de métodos abstratos
25.	MA	NA/NC - média de atributos por classe
26.	MAPu	NAPu/NC - média de atributos públicos por classe
27.	MAPv	NAPv/NC - média de atributos privados por classe
28.	MAPt	NAPt/NC - média de atributos protegidos por classe
29.	MAPc	NAPc/NC - média de atributos de pacote por classe
30.	TAP	NAPu/NA - Taxa de atributos públicos
31.	MM	NM/NC - média de métodos por classe
32.	MMPu	NMPu/NC – média de métodos públicos por classe
33.	MMPV	NMPv/NC – média de métodos privados por classe
34.	MMPt	NMPt/NC - média de métodos protegidos por classe
35.	MMAb	NMAb/NC – média de métodos abstratos por classe
36.	TMP	NMPu/NM – Taxa de Métodos Públicos
37.	FOM	Fator de ocultamento de método
38.	FOA	Fator de Ocultamento de Atributo
39.	maxNF	Número máximo de filhos de uma classe
40.	CBOh	CBO por herança – acoplamento entre objetos
41.	CBOp	CBO por passagem de parâmetros
42.	maxPAH	Profundidade máxima da árvore de herança

Tabela 12 – Métricas APOO.

3.3. Bases

Na construção do ambiente alguns pontos básicos foram estabelecidos:

- Utilização de ferramentas de modelagem de software com livre licença de uso;
- Escolha do formato XMI para representação de modelos UML permitindo a portabilidade entre diferentes ferramentas de modelagem;
- Seleção de métricas de classe passíveis de serem extraídas nas fases iniciais de projeto;
- Extração automática baseada em informações de diferentes projetos e diretamente de diagramas UML e cálculo das métricas sem intervenção do desenvolvedor.

Da representação XMI extraem-se os elementos de modelagem utilizando-se o parser DOM [OMG06]. Os resultados, correspondentes aos artefatos da modelagem UML, são disponibilizados para cálculo dos valores das métricas selecionadas em APOO.

3.4. Arquitetura do Ambiente

A arquitetura de APOO está baseada na plataforma Java possibilitando processamento em diferentes ambientes computacionais - a versão atual é executada como um aplicativo isolado em qualquer plataforma desktop que possua uma JVM (Java Virtual Machine).

As funcionalidades da ferramenta são exibidas na Figura 1: recebe como entrada um projeto modelado em UML e exportado no formato XMI, fornecendo como saída os valores de métricas indicados na Tabela 12.

Basicamente, há um módulo ***Projeto*** responsável pela leitura das classes no formato XMI. Neste processo há a intervenção do parser que extrai os elementos necessários para a análise das respectivas métricas (módulo ***Análise***). Este módulo se encarrega da identificação e validação dos elementos. A seguir, efetuam-se os cálculos das métricas envolvidas (módulo ***Cálculo***).

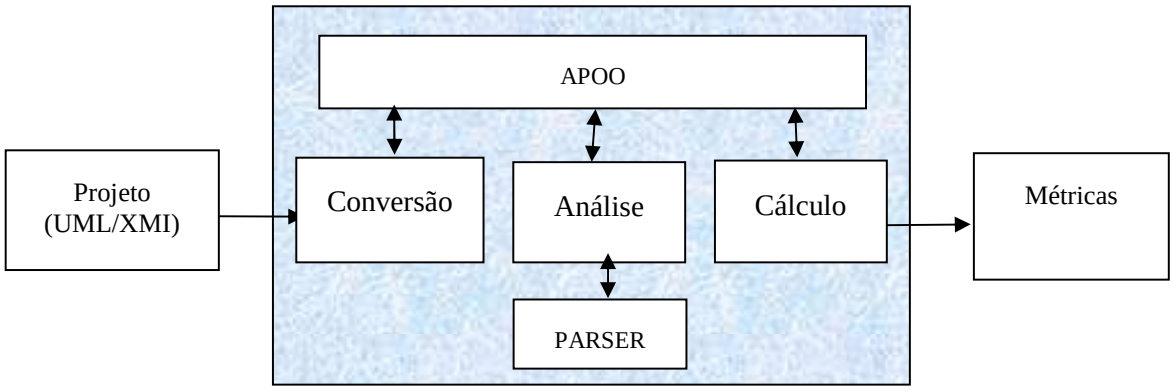


Figura 1 - Módulos APOO.

As medições são analisadas conjuntamente com outros dados de projeto para avaliar o grau de qualidade do mesmo. Pode-se, por exemplo, comparar o resultado obtido com os de um projeto padrão ou mesmo com valores experimentais encontrados na literatura. A Figura 3 exibe, como exemplo, a execução de APOO a partir do diagrama de classes definido na Figura 2.

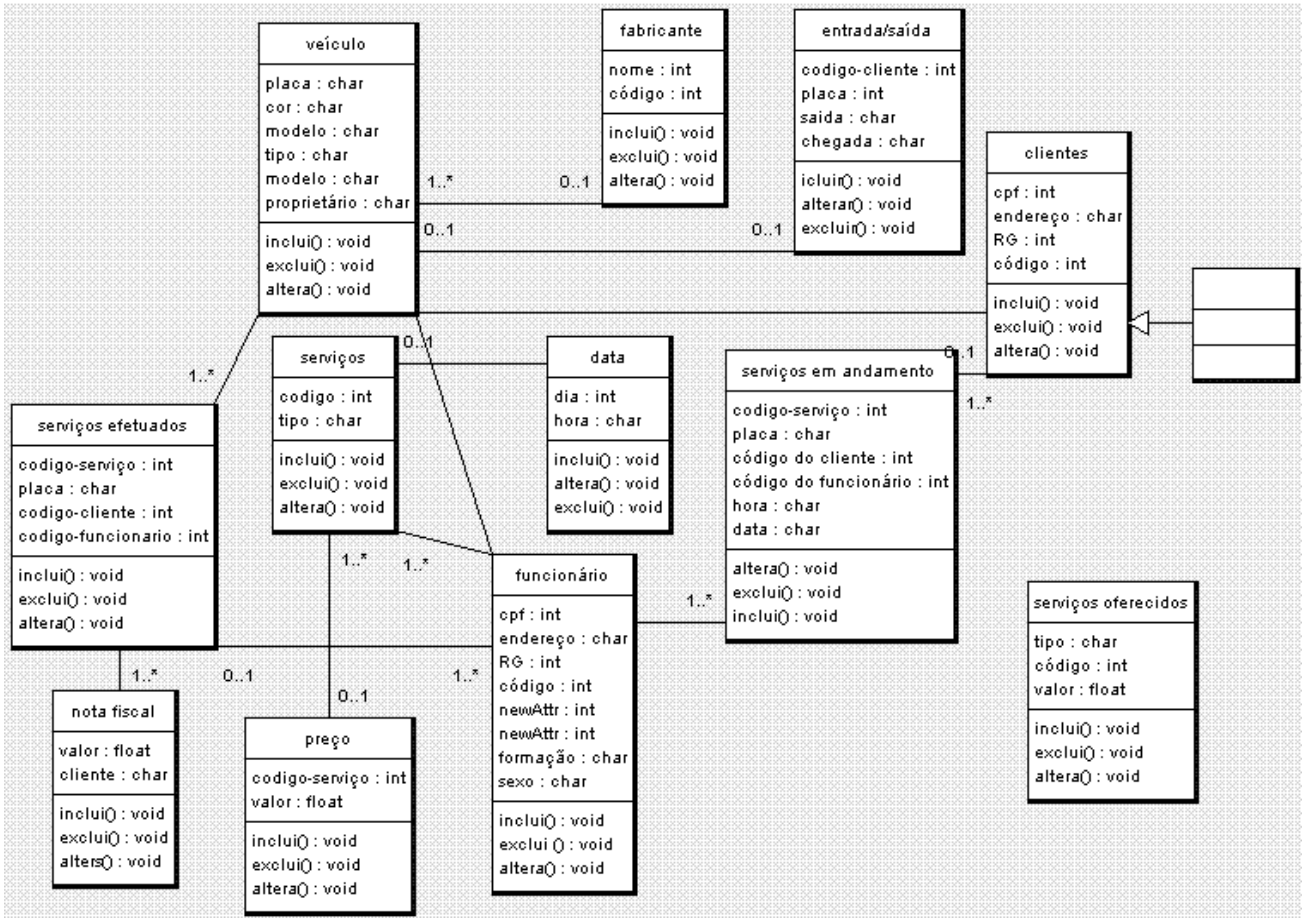


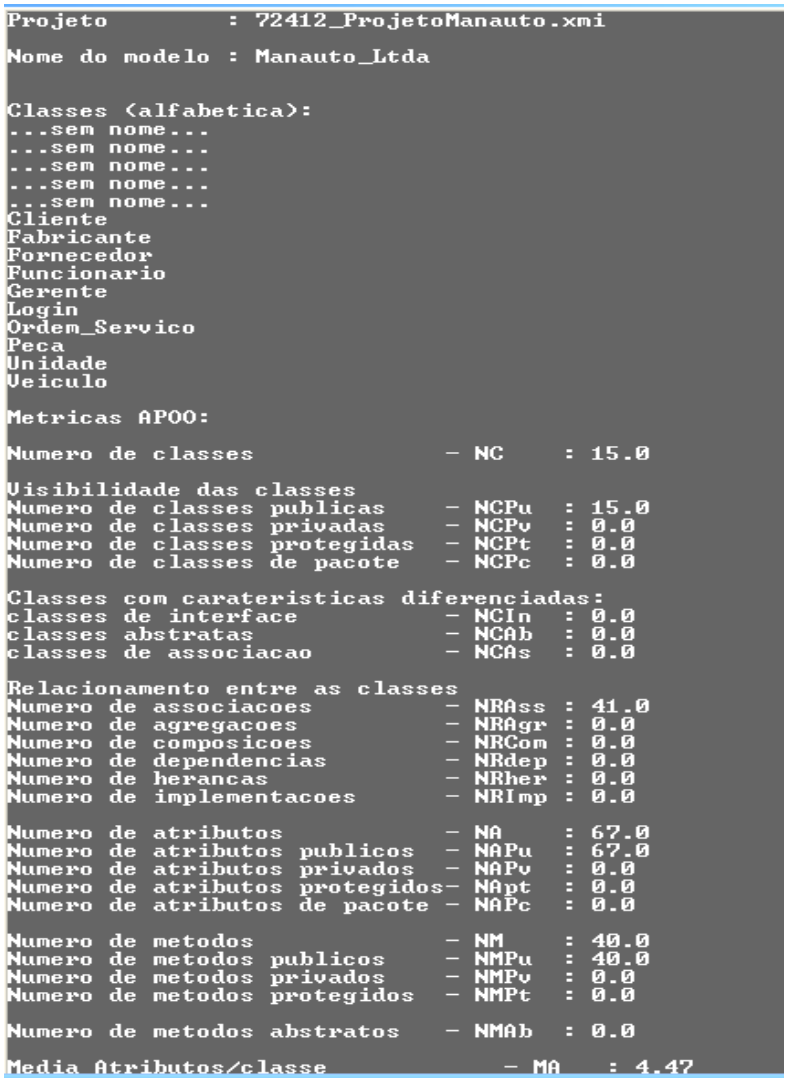
Figura 2 – Exemplo de Diagrama de Classes.

O diagrama de classes da Figura 2 foi gerado em razão da proposta de desenvolvimento de um software para auxílio no controle de serviços de manutenção em veículos em uma rede de oficinas de uma empresa que atende automóveis, utilitários e motos. O software deveria permitir registrar o veículo, o cliente e os serviços feitos quando de uma manutenção. O cliente poderia consultar via web o andamento dos serviços. Outros detalhes da proposta podem ser vistos no Apêndice A2. Observa-se diretamente no desenho que diversas falhas de projeto foram cometidas tais como classes sem associação, atributos sem indicação de visibilidade e métodos simples, genéricos e sem nenhum parâmetro. Estas falhas, certamente, provocarão impactos negativos na avaliação final do projeto.

3.5. Escopo

A versão atual da ferramenta está centrada no uso de métricas de classes. Entretanto, classes não são os únicos elementos de construção que podem ser obtidos através dos modelos UML.

Os diagramas de caso de uso, de seqüência e de estados são também importantes fontes de informações e também estão sujeitas às definições de métricas. Como APOO lida apenas com classes, torna-se uma ferramenta, a princípio, restrita a análise destes diagramas. Espera-se, em trabalhos futuros, a ampliação de seu domínio para o suporte às métricas envolvidas nos demais modelos.



```
Projeto         : 72412_ProjetoManauto.xmi

Nome do modelo : Manauto_Ltda


Classes (alfabetica):
...sem nome...
...sem nome...
...sem nome...
...sem nome...
...sem nome...
Cliente
Fabricante
Fornecedor
Funcionario
Gerente
Login
Ordem_Servico
Peca
Unidade
Veiculo

Metricas APOO:

Numero de classes               - NC    : 15.0

Visibilidade das classes
Numero de classes publicas      - NCPu  : 15.0
Numero de classes privadas      - NCPv  : 0.0
Numero de classes protegidas    - NCPt  : 0.0
Numero de classes de pacote     - NCPc  : 0.0

Classes com caracteristicas diferenciadas:
classes de interface            - NCIn  : 0.0
classes abstratas               - NCAb  : 0.0
classes de associacao           - NCAs  : 0.0

Relacionamento entre as classes
Numero de associacoes           - NRAss : 41.0
Numero de agregacoes            - NRAgr : 0.0
Numero de composicoes           - NRCom : 0.0
Numero de dependencias          - NRdep : 0.0
Numero de herancas              - NRher : 0.0
Numero de implementacoes        - NRImp : 0.0

Numero de atributos             - NA    : 67.0
Numero de atributos publicos    - NAPu  : 67.0
Numero de atributos privados    - NAPv  : 0.0
Numero de atributos protegidos- NApt  : 0.0
Numero de atributos de pacote - NAPc  : 0.0

Numero de metodos               - NM    : 40.0
Numero de metodos publicos      - NMPu  : 40.0
Numero de metodos privados      - NMPv  : 0.0
Numero de metodos protegidos    - NMPt  : 0.0

Numero de metodos abstratos     - NMAb  : 0.0

Media Atributos/classe              - MA    : 4.47
```

Figura 3 – Relatório Parcial com Métricas Obtido via APOO.

3.6. Avaliação via APOO

O processo de avaliação implica no uso de diferentes modelos de medições aplicados as métricas. Três modelos experimentais foram propostos conforme descrito nas subseções seguintes.

3.6.1. Avaliação Comparativa - Medidas de Tendência Central

Neste modelo os projetos produzidos pelos diferentes desenvolvedores são submetidos a APOO resultando em um conjunto de métricas para cada projeto. Nesta forma de avaliação comparam-se os resultados de todos os projetos com valores médios do próprio grupo, obtendo-se então informações relativas aos projetos a partir de desvios em relação estes valores.

Para cada métrica obtém-se a média e desvio padrão, considerando a distribuição estatística pela curva normal dos valores [FON94]. Neste tipo de avaliação há o pressuposto de que todos os desenvolvedores foram expostos ao mesmo tipo de conhecimento e, ao ser tomada uma amostra aleatória da população, pode ser aplicada uma distribuição normal para associar um valor a este conhecimento [SVI99].

Desta forma, cada métrica é então comparada com o valor médio observando a distribuição compreendida entre a média (μ) e o desvio padrão (σ) - região de avaliação.

Formalmente: $\mu - \sigma \leq métrica \leq \mu + \sigma$.

A cada métrica associa-se um único peso que permite definir a relevância da mesma no projeto avaliado. A qualidade deste resulta da soma dos pontos comparados com os valores das médias obtidas.

O resultado é um percentual do valor final a ser atribuído. Desta forma, cada desenvolvedor tem uma avaliação do respectivo projeto relativo ao perfil do grupo ao qual pertence.

3.6.2. Avaliação Comparativa - Projeto de Referência

Neste modelo as métricas de um projeto são comparadas com as de outro tomado como referência. Associa-se a cada métrica então um respectivo intervalo de aceitação.

Para cada projeto avaliado verifica-se a adequação das métricas em relação ao intervalo tomado por referência. Cada métrica é avaliada a partir de um desvio (peso de 0 a 100%) dos respectivos valores de referência.

Este modelo permite aplicar maior rigor na avaliação dos resultados – no modelo anterior, por exemplo, há forte variação nos valores médios e desvios padrões se medidas destoantes (baixas

ou altas) forem encontradas. A qualidade do projeto é definida como um percentual dos valores obtidos em relação aos definidos no projeto de referência.

3.6.3. Avaliação Comparativa - Valores Catalogados na Literatura

Neste modelo cada projeto tem a métrica comparada com valores oriundos da literatura conforme descrito no Capítulo 2. Os valores são definidos experimentalmente através de pesquisas realizadas por diferentes grupos em ambientes controlados [GEN02, HAR98, KIM03].

Convém ressaltar que tais valores foram validados estatisticamente com testes de hipótese, coeficientes de correlação e estudos de variabilidade, fornecendo em alguns casos valores mínimos e máximos aceitáveis [BRI96, GEN01, GYI05, HAR98b, KIM03, KAN04, OLA07].

Este modelo possibilita comparar projetos de uma mesma natureza com valores já referendados. O principal problema, neste caso, é a obtenção das medições visto que nem todas as métricas apresentam dados catalogados. Os valores considerados são exibidos na Tabela 13.

Métrica	LI	LS		Métrica	LI	LS
NC	2	29		NAPc	0	56
NRAss	1	20		NM	1	20
NRAgr	0	9		NMPu	1	20
NRDep	0	4		NMPv	1	20
NRHer	0	24		NMPt	1	20
NRHer	0	24		NMAb	1	20
FOA	44%	68%		maxNF	1	4
NA	4	56		CBOh	1	4
NAPu	0	56		CBOp	1	4
NAPv	0	56		maxPAH	1	4
NAPt	0	56		FOM	8%	25%

Tabela 13 – Valores Catalogados para APOO.

3.7. Considerações Finais

Este capítulo descreveu a ferramenta APOO, sua arquitetura e modelos de avaliação de métricas. Para a validação da mesma, o capítulo seguinte apresenta estudos de casos. Diferentes projetos de diferentes grupos de desenvolvedores são testados com este objetivo.

4. Estudo de Caso

4.1. Introdução

Os modelos de avaliação, com o uso do ambiente APOO, foram discutidos no Capítulo 3. Neste capítulo são apresentados os resultados obtidos a partir de projetos elaborados na disciplina de Engenharia de Software.

O estudo analisa diferentes projetos de software para diferentes grupos de projetistas com prévio conhecimento dos conceitos de programação, análise e projeto orientado por objeto. Além disso, um pressuposto básico para a avaliação é a existência de domínio dos métodos, técnicas e ferramentas de modelagem de software usando UML.

A partir do segundo semestre de 2006, durante três semestres consecutivos, foram propostos temas de projetos a serem desenvolvidos:

- 2º/2006 - sistema de controle de manutenção de veículos;
- 1º/2007 - sistema de controle de locação de equipamentos para construção civil;
- 2º/2007 - sistema de controle de agendamento em posto de saúde publica.

As seguintes etapas foram aplicadas: execuções de APOO para coleta de dados (diretamente dos diagramas UML), cálculo e geração dos valores de métricas referentes a cada projeto, e análise dos resultados obtidos.

Dentre as diversas ferramentas CASE disponíveis para modelagem em UML, como ROSE©, POSEIDON©, ArgoUML e JUDE© sugeriu-se o uso de ArgoUML [ARG06] por ser de uso livre, código aberto, e possibilitar a exportação de dados no formato XMI [OMG06] - formato este adotado por diversos desenvolvedores de software no mercado.

Este capítulo apresenta os resultados do estudo de caso relativo ao desenvolvimento da aplicação de controle de manutenção de veículos, no modelo cliente-servidor, acessados via rede local/web.

4.2. Resultados

Nove projetos foram utilizados para o primeiro teste de APOO. A Tabela 14 apresenta as métricas obtidas. Exibe também os valores de um projeto de referência (PR) fornecido pelo avaliador.

Características	Tema: Sistema de Controle de Manutenção de Veículos										
	Métricas	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	PR
Classes do projeto	NC	15	13	32	11	23	7	8	25	17	8
Encapsulamento de classes	NCPu	15	13	32	11	23	7	8	25	17	8
	NCPv	0	0	0	0	0	0	0	0	0	0
	NCPt	0	0	0	0	0	0	0	0	0	0
	NCPc	0	0	0	0	0	0	0	0	0	0
Associações de classes	NCAs	0	0	0	0	0	2	0	0	0	1
	NCIn	0	1	0	0	0	0	0	0	0	0
	NCAb	0	0	0	0	0	0	0	0	0	0
Relacionamentos de classes	NRAss	41	43	38	38	11	33	17	26	22	5
	NRAgre	0	0	0	0	0	1	0	0	0	0
	NRCom	0	0	0	0	0	1	0	0	0	0
	NRDep	0	0	0	0	1	3	0	0	0	0
	NRHer	0	1	0	0	10	1	2	0	6	0
	NRImp	0	0	0	0	0	0	0	0	0	0
Atributos do projeto	NA	67	45	56	40	42	14	26	67	48	37
	NAPu	67	44	56	37	41	14	26	60	48	8
	NAPv	0	1	0	3	1	0	0	7	0	0
	NAPt	0	0	0	0	0	0	0	0	0	29
	NAPt	0	0	0	0	0	0	0	0	0	0
Métodos do projeto	NM	40	36	40	40	32	0	0	36	8	34
	NMPu	40	36	40	40	31	0	0	36	8	27
	NMPv	0	0	0	0	1	0	0	0	0	0
	NMPt	0	0	0	0	0	0	0	0	0	7
	NMAb	0	0	0	0	0	0	0	0	0	0
Herança	maxNF	0	1	0	0	3	0	2	0	3	0
	MaxPAH		1			1		1		1	0
Acoplamento	CBOh	0	1	0	0	10	1	2	0	6	0
	CBOp								54	0	3
Ocultamento	FOM	0%	0%	0%	0%	3%	0%	0%	0%	0	21%
	FOA	0%	2%	0%	8%	2%	0%	0%	10%	0	78%

Tabela 14 – Métricas Obtidas para o Sistema de Controle de Manutenção de Veículos.

A seguir são apresentadas análises sobre algumas das métricas propostas classificadas por categorias. Outras métricas podem ser obtidas no apêndice A2.

4.2.1 Métricas relativas a tamanho e complexidade

Estas métricas fornecem informações básicas do tamanho e complexidade do projeto. São denominadas ***primitivas***, pois derivam diretamente da contagem dos elementos presentes. Além

disso, servem de base para o cálculo de outras ***derivada***s [KIM03].

A. Número de classes (NC) - esta métrica indica o tamanho do modelo identificado pelos desenvolvedores e suas percepções relativas ao escopo do projeto e conseqüentemente o esforço de construção do software.

A variação nos valores encontrados para a métrica NC pode ser observada na Figura 4. Era de se esperar uma maior uniformidade, pois os desenvolvedores possuem um mesmo nível de conhecimento. Note que distorções podem ser indicativas de dificuldades no entendimento do escopo do projeto.

Conforme valores obtidos, apenas três projetos se aproximam do padrão de referência (PR). Outro fato observado é a presença de classes sem identificação, o que eleva o número de classes indicando descuido de modelagem.

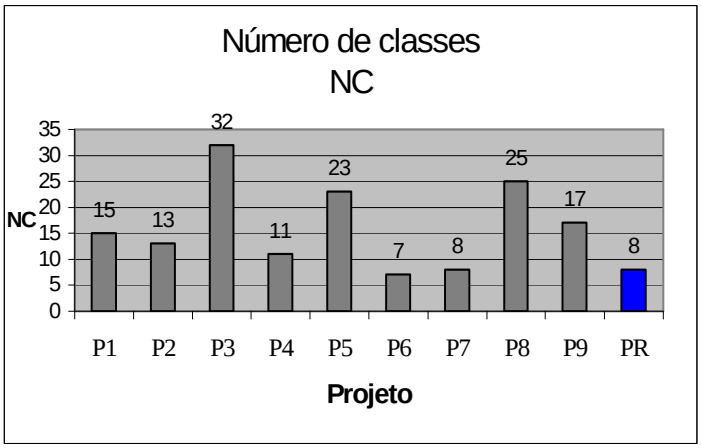


Figura 4. Métrica NC.

B. Número de atributos (NA) - esta métrica (Figura 5) é essencial para a verificação de fatoração das classes além de aspectos de ocultação de informação. Observa-se uma grande variação nos valores encontrados - mínimo de 14, máximo de 67, com amplitude 53. É de se esperar que o aumento no número de classes implique em um aumento no número de atributos.

Entretanto, uma relação NA/NC muito alta é um indicativo que o desenvolvedor não fatora/particiona as classes para melhor reuso via herança ou associação. No caso de uma relação baixa é um indicativo de classes subutilizadas. É importante para o aperfeiçoamento das abstrações.

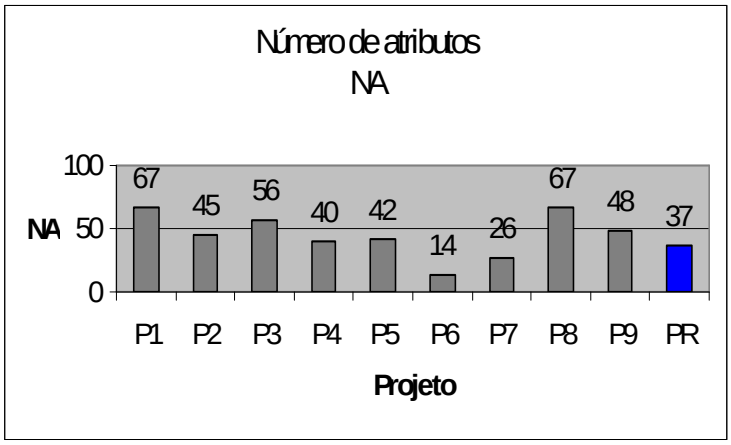


Figura 5. Métrica NA.

C. Número de métodos (NM) - Não ocorre grande variação nos valores encontrados com exceção do projeto P_9 (Figura 6). Os projetos P_6 e P_7 são casos típicos de descuido na modelagem, uma vez que, nenhum método é disponibilizado nas classes de negócio. Outro aspecto a ser considerado é a relação NM/NC. Pode-se observar a existência de classes sem métodos quando a relação é menor que um (1), como no projeto P_9.

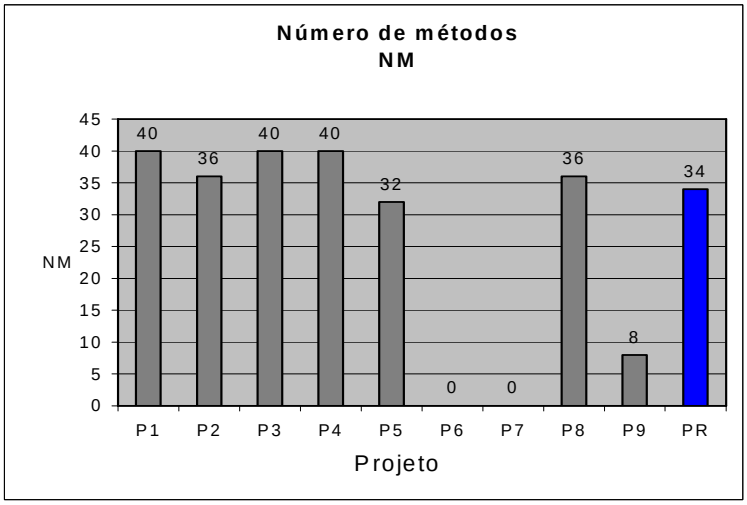


Figura 6. Métrica NM.

4.2.2. Métricas relativas à herança

A. Número máximo de filhos (maxNF): indica um grau de reuso desejável. Define a divisão em subclasses. Um valor muito alto pode indicar excesso de classes especializadas. Além disso, pode indicar centralização de responsabilidades em poucas classes. Pode ser observado (Figura 7) que apenas quatro projetos (P_2, P_5, P_7 e P_9) consideram a especialização para modelagem do

problema, isto é, número de filhos maior que 0 (zero), tirando benefícios da reutilização de classes.

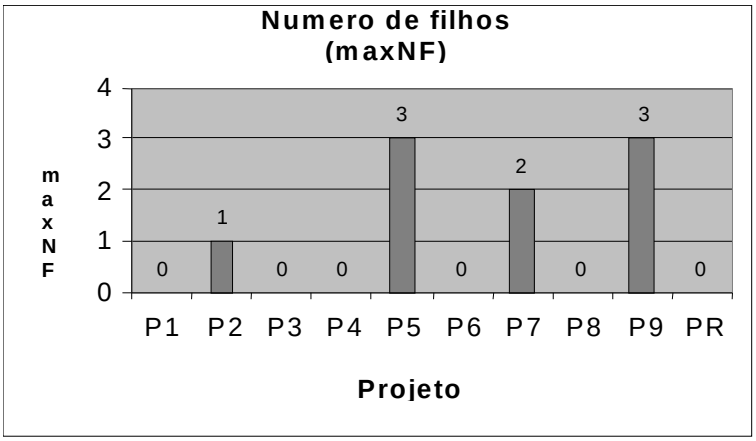


Figura 7. Métrica maxNF

Um valor alto de maxNF indica que uma das classes do projeto apresenta um excessivo número de filhos indicando uma possível falha de particionamento na definição das subclasses. Pela Figura 7 observa-se que apenas quatro projetos consideraram os aspectos de especialização de classes, fato que sugere algum tipo de dificuldade de entendimento pelo grupo e merece uma análise mais detalhada.

B. Profundidade máxima da árvore de herança (maxPAH): esta métrica permite verificar o grau de reutilização de classes no projeto. Quanto maior a profundidade da classe na hierarquia, maior a complexidade para a correta implementação e testes.

Um maior esforço deve ser despendido para o entendimento dos componentes disponibilizados pelas classes ancestrais. Observa-se pela Figura 8 que apenas quatro projetos estabeleceram maxPAH com valores aceitáveis (capítulo 2, seção 2.6). Os demais não consideraram ou não identificaram a necessidade de especializações.

É possível notar a relação entre maxPAH e maxNF. Valores altos para maxNF indicam um diagrama com grande amplitude o que sugere uso inadequado de especialização – uma única classe com um número elevado de descendentes diretos. Valores altos para maxPAH indicam maxPAH.uso excessivo de herança – um número elevado de descendentes.

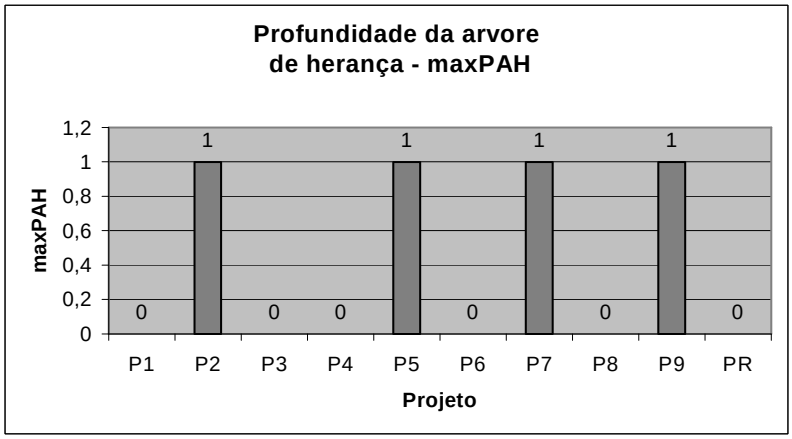


Figura 8 – Métrica

4.2.3 Métricas relativas a acoplamento

As métricas de acoplamento indicam o grau de dependência entre classes. Aqui a preocupação deve ser a de manter o nível de acoplamento baixo.

O acoplamento acontece por herança, por passagem de classes como parâmetro numa mensagem, por utilização de métodos ou de atributos de outra classe. No diagrama de classes o acoplamento se manifesta via herança e assinatura dos métodos. [CHI94, HAR98a, OLA07]

A. Acoplamento por herança - CBOh: é obtido pela contagem do número de especializações no diagrama. Pode-se observar (Figura 9) que o projeto P_5 apresenta um alto valor para CBOh. Tal fato facilita a reutilização, mas deve ser analisado com cuidado para se evitar particionamento excessivo de classes criadas desnecessariamente.

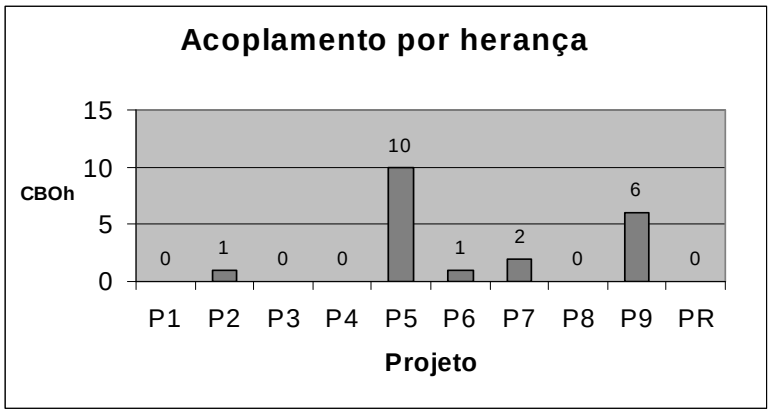


Figura 9 - Métrica CBOh

CBOh mede o número de subclasses existentes em todo o diagrama. O valor 10 para o projeto P_5 indica que das 23 (vinte e três classes) desenvolvidas, métrica NC da Figura 4, 10 (dez)

são subclasses, resultando em aproximadamente 43% de especialização. Este valor em termos absolutos pode não ser representativo, mas se comparado com os valores dos outros projetos sugere que uma investigação deve ser realizada junto a todo o grupo.

B. Acoplamento por parâmetro - CBOp: é obtido pela contagem das classes presentes na assinatura dos métodos. Nenhum projeto apresenta acoplamento por parâmetro (Figura 10), este fato indica a existência de dificuldades na identificação de passagem classes como parâmetro.

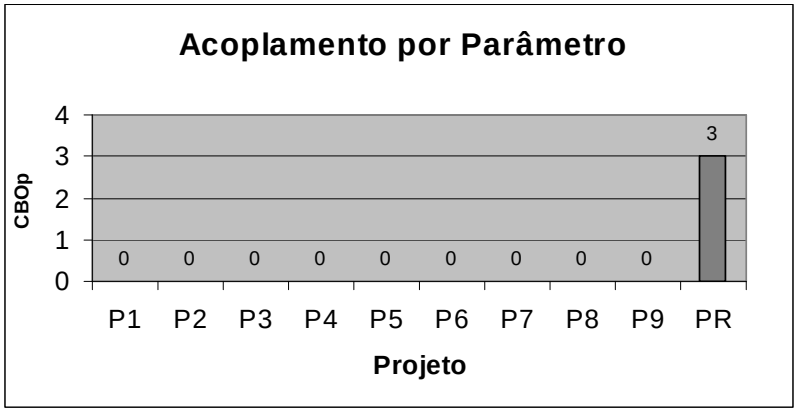


Figura 10 – Métrica CBOp

4.2.4 Métricas relativas a encapsulamento

As métricas de encapsulamento indicam se o desenvolvedor se preocupa em definir atributos e métodos visíveis, exportados pela classe. Fornece, portanto uma medida do ocultamento de atributos e métodos [BRI96, HAR98a, OLA07].

Recomenda-se (Capitulo 2, seção 2.6) que os fatores de ocultamento de atributos sejam altos, fato este que indica a necessidade de serem declarados privados ou protegidos. Em contra partida, como os métodos devem ser compartilhados precisam ser declarados públicos implicando em um fator de ocultamento baixo.

A. Fator de Ocultamento de atributo-FOA: indica o grau de ocultamento de atributos de todo o projeto. É a relação entre atributos ocultos (privados ou protegidos) e o total de atributos do projeto.

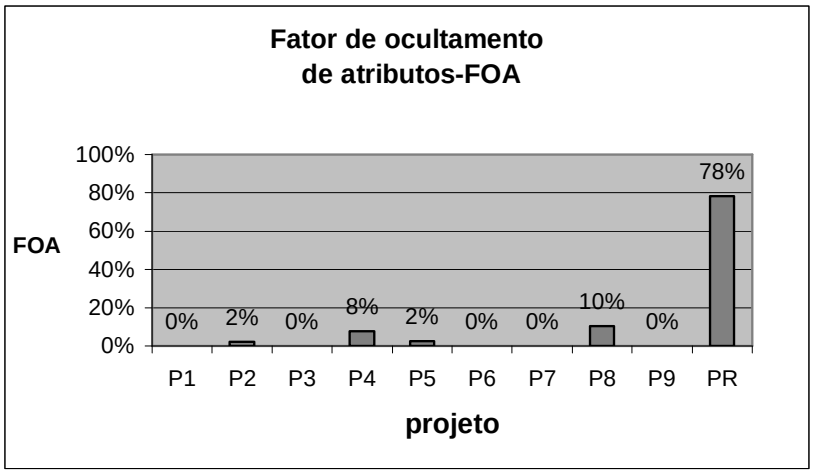


Figura 11 – Métrica FOA

Observa-se pela Figura 11 que não houve, durante o projeto, uma análise mais profunda do aspecto de ocultamento de atributos (o maior valor é 10%) e cinco dos projetos consideraram todos os atributos públicos. Neste caso, tal fato indica uma percepção incorreta dos conceitos de abstração – não se ocultam detalhes de implementação. O acesso aos dados permite a existência de objetos inconsistentes.

B. Fator de Ocultamento de Método - FOM: indica o grau de ocultamento de métodos para todo o projeto. É a relação entre métodos ocultos (privados e protegidos) e o total de métodos.

Verifica-se, pela Figura 12, que apenas um projeto considerou tal aspecto. Ora, é de se esperar que classes disponibilizem serviços através de métodos públicos, o que implica em um baixo valor para FOM. Valores próximos do limite superior (100%) certamente indicam um projeto de baixa qualidade.

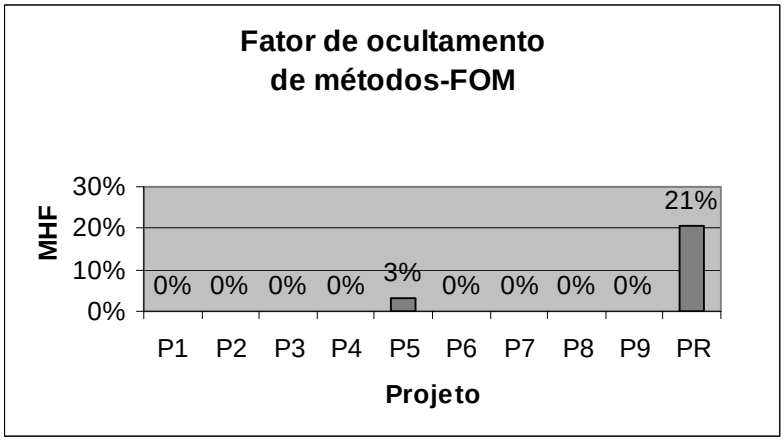


Figura 12 – Métrica FOM

4.3 Avaliações dos Projetos

APOO possibilita a avaliação automática dos projetos submetidos efetuando comparações com medidas de tendência central, ou em relação a um projeto de referência ou, então, em relação aos valores previamente catalogados na literatura (Capítulo 2) considerando todas as métricas com igual peso (peso 1). As subseções seguintes apresentam os resultados obtidos.

4.3.1 Avaliação com Medidas de Tendência Central

Para esta forma de avaliação são realizados cálculos estatísticos para obtenção dos valores de referência tomando-se, como universo, todos os projetos apresentados. São obtidas medidas de tendência central e erro de estimativa (Tabela 15).

Utiliza-se uma Distribuição Normal centrada na média e desvio-padrão do grupo. Foi estabelecido com um desvio padrão em torno da média, um intervalo de aceitação com limite inferior (LI) e superior (LS). Avalia-se, comparando o valor de cada métrica do projeto com os valores do intervalo. O padrão é a atribuição de pesos iguais para todas as métricas avaliadas. A pontuação final é a soma ponderada das métricas, com peso 1 (um) para as que se situarem no intervalo e 0 (zero) para aquelas fora do mesmo.

Estatística						
Métricas	Mediana	Média	Desvio	Erro	LI	LS
NC	15	17	8	3	9	25
NCPu	15	17	8	3	9	25
NCPv	0	0	0	0	0	0
NCPt	0	0	0	0	0	0
NCPc	0	0	0	0	0	0
NCAs	0	0	1	0	0	1
NCInb	0	0	0	0	0	0
NCAb	0	0	0	0	0	0
NRAss	33	30	11	4	19	41
NRAgr	0	0	0	0	0	0
NRCom	0	0	0	0	0	0
NRDep	0	0	1	0	0	1
NRHer	1	2	3	1	0	5
NRImp	0	0	0	0	0	0
NA	45	45	18	6	27	63
NAPu	44	44	17	6	27	61
NAPv	0	1	2	1	0	3
NAPt	0	0	0	0	0	0
NAPc	0	0	0	0	0	0
NM	36	26	18	6	8	44
NMPu	36	26	18	6	8	44
NMPv	0	0	0	0	0	0
NMPt	0	0	0	0	0	0
NMAb	0	0	0	0	0	0
MaxNOC	0	1	1	0	0	2
CBOh	1	2	3	1	0	5
CBOp	0	0	0	0	0	0
MaxDIT	0	0	1	0	0	1
FOM	0	0%	1%	0	0%	1%
FOA	0	3%	4%	0	0%	6%

Tabela 15 – Medidas Estatísticas.

A Tabela 16 apresenta os resultados da avaliação. A nota final corresponde ao percentual dos acertos obtidos tomando como base os pontos médios de cada métrica.

Pontos:								
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
7	15	7	11	13	3	4	8	11
Notas								
4,4	9,4	4,4	6,9	8,1	1,9	2,5	5	6,9

Tabela 16 – Avaliação dos Projetos

Na Tabela 17 são exibidas as notas atribuídas pelo avaliador sem o uso de APOO. Observam-se diferenças nas notas do avaliador em 4 dos 9 projetos. P_2 e P5 obtêm notas menores e P7 e P8 maiores.

Projeto	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
Nota do avaliador	5	5	6	7	5	3	7	9	5
Nota APOO	4,4	9,4	4,4	6,9	8,1	1,9	2,5	5	6,9

Tabela 17 – Comparação: Avaliador e APOO

Além disso, este modelo de avaliação está sujeito à ocorrência de valores extremos, muito altos ou muito baixos, que acabam por influenciar a média e o desvio padrão. Observa-se também que a mesma ponderação para todas as métricas pode não ser a ideal. Tal fato pode ser usado para avaliações parciais de modo a uniformizar o entendimento dos grupos de projeto acerca de conceitos ainda não consolidados. Por exemplo, de classe, de herança, de acoplamento, de associação e outros, podendo-se assim reduzir a influência de valores extremos obtendo-se um desvio menor em torno da média.

4.3.2 Avaliação usando o Padrão de Referência

Para esta forma de avaliação cada métrica é comparada com a de um ***projeto padrão*** de referência fornecido ao APOO pelo avaliador. Este define uma faixa de tolerância em torno do valor considerado padrão. São consideradas, para efeito de avaliação, apenas as que se situam dentro do intervalo de tolerância. A nota é o percentual das métricas no intervalo. A Tabela 18 mostra os valores utilizados para o projeto de referência.

Valores do PR							
Métrica	PR	LI	LS	Métrica	PR	LI	LS
NC	8	3	13	NAPu	8	3	13
NCPu	8	3	13	NAPv	0	0	5
NCPv	0	0	5	NAPt	29	24	34
NCPt	0	0	5	NAPc	0	0	5
NCPc	0	0	5	NM	34	29	39
NCAs	1	0	6	NMPu	27	22	32
NCIn	0	0	5	NMPv	0	0	5
NCAb	0	0	5	NMPt	7	2	12
NRAss	5	0	10	NMAb	0	0	5
NRAgr	0	0	5	maxNF	0	0	5
NRAss	0	0	5	CBOh	0	0	5
NRDep	0	0	5	CBOp	3	0	8
NRHer	0	0	5	maxPAH	0	0	5
NRImp	0	0	5	FOM	21%	0%	23%
NA	37	32	42	FOA	78%	0%	86%

Tabela 18 – Valores do projeto de Referência

A tabela 19 exibe nova avaliação dos projetos frente ao modelo de referência. Neste caso os resultados demonstram que a avaliação, com o projeto de referência torna o processo mais rígido em relação à média. O uso de um padrão permite ao Avaliador calibrar o intervalo considerando diferentes aspectos relevantes ao projeto - amplia (ou reduz) o intervalo de tolerância de acordo com considerações obtidas pelo próprio grupo.

A mesma ponderação para todas as métricas, aqui também, pode não ser a ideal. A diferenciação de pesos poderia ser introduzida em estágios mais avançados do projeto exigindo-se maior rigor na modelagem das classes.

As notas obtidas com o padrão de referência foram, em geral, menores que ao se usar a média dos projetos. A avaliação com projeto de referência estabelece faixas de aceitação mais estreitas em relação à média. Entretanto, P_6 e $P_{7,}$ quando avaliados com o projeto de referência obtêm notas maiores. A medida de tendência central não possibilita a avaliação de métricas cujos valores são destoantes da média, mas em um projeto de referência estas mesmas métricas podem estar dentro da faixa de aceitação.

Pontos dos projetos usando o Padrão de Referência									
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	PR
2	7	2	7	7	5	5	4	2	16
Nota do Avaliador									
5	5	6	7	5	3	7	9	5	
Nota APOO usando a média									
4,4	9,4	4,4	6,9	8,1	1,9	2,5	5	6,9	
Nota APOO usando o padrão de referência									
3	6	3	6	6	4	4	4	3	10

Tabela 19 – Avaliação de Tendência Central x Projeto de Referência

4.3.3 Avaliação com Valores da Literatura

Nesta avaliação o valor de cada métrica do projeto é comparado com valor mínimo e máximo recomendados por diferentes pesquisadores. No caso de ocorrência de diferentes valores para a mesma métrica, utiliza-se o de maior amplitude por permitir flexibilidade na avaliação do projeto.

Os resultados são obtidos a partir do intervalo de aceitação. Como o limite inferior, para algumas métricas, não é determinado assume-se o valor zero. Nesta avaliação, os valores de métricas são aceitos se pertencentes ao intervalo determinado por um limite superior obtido na

literatura. Os resultados são mostrados na Tabela 20.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	PR
Pontos usando valores da literatura:									
8	14	9	11	15	12	15	8	16	15
Nota do Avaliador									
5	5	6	7	5	3	7	9	5	
Nota APOO com valores da literatura									
3	6	4	5	6	5	6	3	7	6

Tabela 20 – Notas usando valores da literatura

Observa-se que a avaliação final dos projetos, novamente, não coincide com as do avaliador. Entretanto, os valores obtidos pela avaliação da ferramenta estão com distribuição uniforme. Este fato certamente deriva da maior amplitude da faixa de aceitação obtida na literatura como no caso das métricas envolvendo numero de classes e de atributos. A avaliação utiliza os limites inferiores e superiores constantes da tabela 21.

Valores da Literatura					
Métrica	LI	LS	Métrica	LI	LS
NC	2	29	NAPu	0	56
NCPu	0	0	NAPv	0	56
NCPv	0	0	NAPt	0	56
NCPt	0	0	NAPc	0	56
NCPc	0	0	NM	1	20
NCAs	0	0	NMPu	1	20
NCIn	0	0	NMPv	1	20
NCAb	0	0	NMPt	1	20
NRAss	1	20	NMAb	1	20
NRAgr	0	9	maxNF	1	4
NRAss	0	0	CBOh	1	4
NRDep	0	4	CBOp	1	4
NRHer	0	24	maxPAH	1	4
NRImp	0	0	FOM	8%	25%
NA	4	56	FOA	44%	68%

Tabela 21 – Valores de Referência da Literatura

A avaliação com o uso de valores da literatura é uma tentativa de verificar se um projeto,

classificado por tamanho (classes, atributos e métodos), apresenta valores de métricas aceitáveis frente a diferentes projetos já avaliados.

Uma comparação entre os resultados obtidos pelos três critérios anteriores é mostrada na Figura 13. Com o uso de um projeto de referência (PR) a avaliação se apresenta conservadora conforme discutido anteriormente.

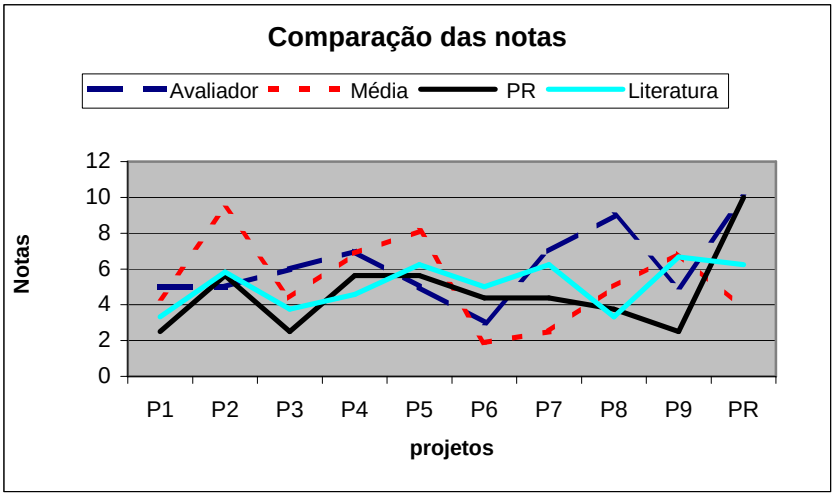


Figura 13 – Comparação dos resultados

Esse aspecto permite ao avaliador rever os critérios adotados na definição do intervalo de tolerância estabelecendo um maior ou menor rigor no desenvolvimento do projeto.

A avaliação pela média é mais flexível e pode não ser a ideal se o grupo se mostra heterogêneo, pois há uma grande variação nos intervalos de tolerância. Entretanto, permite orientar o grupo quanto a distorções detectadas pela ferramenta. Este fato possibilita o avaliador reforçar conceitos ainda não consolidados.

Interessante observar que os pontos de inflexão demonstram que, salvo as exceções, um projeto bem avaliado por determinado modelo tende a seguir o mesmo processo frente aos demais.

4.4 Considerações finais

Este capítulo apresentou um estudo de caso para validação do ambiente APOO onde foram coletados automaticamente os dados de nove projetos elaborados em um período de 4 (quatro) meses por grupos de estudantes de um urso de engenharia de software.

A partir destes dados foram extraídas e calculadas métricas baseadas nos diferentes modelos apresentados no Capítulo 2 para avaliar os projetos de três diferentes formas: comparação com a

média do grupo, comparação com um projeto de referência e comparação com valores da literatura. Os resultados, embora preliminares indicam que esta área de estudo é interessante e bastante controvertida, na opinião de alguns pesquisadores [VOA99, MAY99] porque:

- Medidas empíricas em qualidade de software vêm sendo sugeridas;
- A qualidade de software orientado a objeto é sempre demandada, mas não existe um consenso quanto à melhor forma de medi-la, daí a busca de modelos mais adequados;
- Os ambientes de desenvolvimento sempre apresentam demandas de estimativas de prazo, esforço em pessoas-mês, e custo de projetos logo no seu início o que leva à busca de modelos de estimativa adequados a cada ambiente.

5. Conclusão

Nesta pesquisa foram levantados os relevantes trabalhos na área de medição e da qualidade de desenvolvimento e de produto de software orientado a objeto e proposto um ambiente de avaliação de projetos utilizando classes modeladas com uso de UML e exportadas no padrão de intercâmbio de dados XMI com o objetivo de avaliar aspectos de qualidade de sistemas de gerenciamento de informações através de medidas quantitativas.

A qualidade dos Sistemas de Informação depende fortemente da qualidade do software que lhe dá suporte, este depende da qualidade do projeto. Daí a importância de se dedicar estudos visando melhorias na qualidade dos artefatos de projeto com metodologias centradas em métricas de produto e de processo.

O uso de ferramentas que permitam efetuar medições e extrair métricas, nas diferentes etapas do desenvolvimento, certamente auxilia na avaliação da qualidade do projeto e do produto de software.

Entretanto, a maioria das ferramentas comerciais para avaliação de projetos, baseadas em métricas, são centradas em código fonte. Poucas são as que utilizam modelos de projeto como entrada.

Neste trabalho foi apresentado um estudo dos atuais modelos de métricas utilizados na avaliação da qualidade de software. Além disso, implementou-se o ambiente APOO, baseando-se em métricas, para a avaliação quantitativa de modelos de classes UML exportados no formato XMI.

Os testes realizados demonstraram que a abordagem proposta é viável auxiliando, por exemplo, a avaliação de aprendizado nas disciplinas de engenharia de software. A extração de métricas diretamente dos modelos de classe oferece flexibilidade e rapidez na avaliação de algumas características do projeto, permitindo aos instrutores acompanhar, com maior detalhe, a correta construção do produto de software.

5.1 Trabalhos Futuros

O modelo de classes é apenas um dos componentes de um projeto de software. Diferentes características podem ser descritas por variados modelos como: diagramas de seqüência, caso de uso, estado, distribuição, atividades e outros. Desta forma, como possíveis trabalhos futuros cita-se:

- portar o ambiente para uma plataforma WebServer onde as avaliações de projeto são efetuadas remotamente;

- incorporar métricas para utilização dos demais diagramas UML;
- introduzir recursos que possibilitem desenvolvimentos colaborativos, e
- ampliar o ambiente para suporte à modelos de estimativas de esforço e custo para o desenvolvimento do software.

Referências

[ALG05] ALGHANDI, J. S. RUFAI, R. A.; KHAN, S. M.. OOMeter: a software Quality Assurance Tool, CSMR, pp. 190-191, Proceedings of Ninth European Conference on Software Maintenance and Reengineering (CSMR'05), 2005.

[ARG07] ArgoUML disponível em www.tigris.org acesso em 19/03/2007

[ATK03] Atkinson C., KühneT.. Model-Driven Development: A metamodeling Foundation, IEEE software, vol 20, no. 5, pp. 36-41, 2003.

[BAN02] BANSIYA, J.; DAVIS, C. G.. A Hierarchical Model for Object_Oriented Design Quality Assessment. IEEE Transaction on Software Engineering, January 2002, vol. 28, No. 1.

[BAS96] Basili, V., Briand, L., Melo W.. A Validation of Object-Oriented Design Quality Metrics as Quality Indicators, IEEE Transactions on Software Engineering, vol. 22, no. 10, pp. 751-761, 1996.

[BHA05] BHATTI, Shahid N.. Why Quality? ISO 9126 Software Quality Metrics (Functionality) Support by UML Suite. ACM Sigsoft Software Engineering Notes March 2005 volume 30 Number 2.

[BLA04] BLAYA, Carolina. Processo de Avaliação. Prática Educativa – Textos, artigos e reflexões. Disponível em www.ufrgs.br/transe/med/textos/2004_07_20_text.htm acesso em 09/08/2007.

[BRE07] BREESAM, Kadhim M.. Metrics for Object-Oriented Design Focusing on Class Inheritance Metrics. Dependability on Computer Systems, 2007. DepCoS-Relcomex'07 2nd International Conference 14-16 june 2007 page(s): 231-237.

[BRI94] BRITO e ABREU, F., CARAPUÇA, R. Object Software Engineering: Measuring and Controlling the Development Process. 4th International Conference on Software Quality, Mc Lean, Va, USA, 1994.

[BRI96] BRITO e ABREU, F.; MELO, W., Evaluating the Impact of OO Design on Software Quality. Third International Software Metrics Symposium. Berlin, 1996.

[CHI94] CHIDAMBER, Shyam R., KEMERER, Chris F.. A metrics Suite for object Oriented. IEEE Transactions on Software Engineering, vol. 20, No. 6, June 1994, pp. 476-493.

[CHI98] CHIDAMBER, Shyam R.; DARCI, David P.; KEMERER, Chris F. Managerial use of Metrics for Object-Oriented Software: An Exploratory Analysis. IEEE Transactions on Software Engineering, vol. 24, No. 8, August 1998, pp. 629-639.

[DAR05] DARCY, D. P. KEMERER F. C. OO Metrics in Practice. IEEE Software Nov/Dec 2005, pp 17-19.

[DMO07] Dmoz open source project. Disponível em http://dmoz.org/Computers/Programming/Methodologies/Modeling_languages/Unified_Modeling_language acesso em 20/06/2007.

[FON92] FONSECA, J. S.; MARTINS, G. A..Curso de Estatística. 5ª ed., São Paulo, Atlas, 1994.

[GEN01] GENERO, M.; PIATTINI M.; JIMÉNES L.. Empirical Validation of Class Diagram Complexity Metrics. Computer Science Society, 2001. Proceedings. XXI International Conference of the Chilean. November 2001, pp. 95-104.

[GEN02] GENERO, M.; PIATTINI M.; CALERO C.. Empirical Validation of Class Diagram Metrics. Proceedings of the 2002 International Symposium on Empirical Software Engineering (ISESE'02)

[GEN05] GENERO, M.; PIATTINI M.; CALERO C.. A Survey of Metrics for UML Class Diagrams. Journal of Object Technology, vol. 4 , No 9, November-December, 2005.

[GLA98] GLASS, R., Defining Quality Intuitively, IEEE software, May 1998, pp. 103-104, 107.

[GYI05] Gyimóthy, T.; FERENC, R; SIKET, I.. Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. IEEE Transactions on Software Engineering, vol. 31, NO. 10, October 2005, pp. 897-910.

[HAR97] HARRISON, R., COUNSELL, S., NITHI, R.. An Overview of Object-Oriented Metrics, IEEE Transactions on Software Engineering, 2/1997 pp. 230-235.

[HAR98a] HARRISON, R., COUNSELL, S.. An Evaluation of the MOOD Set of Object-Oriented Software Metrics, IEEE Transactions on Software Engineering, vol. 24, NO. 6, JUNE 1998 pp. 491-496.

[HAR98b] HARRISON, R., COUNSELL, S., NITHI, R.. Coupling Metrics for Object-Oriented Design, Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International. November 1998,pp. 150-157.

[IEE90] IEEE Software Engineering Standards, Standard 610.12-1990, pp. 47-48.

[IEE94] IEEE Standard Glossary of Software Engineering Terms.

[ISO01] ISO/IEC 9126-1: "Information Technology-Software Product Quality - part 1: Quality Model", 2001

[JAC99] JACOBSON, Ivar, BOOCH, Grady, RUMBAUGH, James. The Unified Software Development Process. 1ª ed. Addison-Wesley,1999.

[KAN04] KANMANI, S.; RHYMEND, U.V.; SANKARANARAYANAN, V.; THAMBIDURAI, P.. Investigation into the Explotation of Object-Oriented Features. Software Engineering Notes, vol 29, NO 2, MARCH 2004, pp. 1-9.

[KIM03] IN, Peter, KIM, SangEun, BARRY, Mattew.. UML-Based Object Oriented Metrics for Architecture Complexity Analysis, 28-3-2003. Texas A&M University. Disponível em http://faculty.cs.tamu.edu/hohin acesso em 01/10/2006 e http://sunset.usc.edu/GSAW/gsaw2003/s8e/in.pdf acesso em 13/03/2007.

[LOR94] LORENZ, M., KIDD, J.. Object-Oriented Software Metrics, Prentice Hall, 1994.

[LUI05] LAVAZZA, L., AGOSTINI, A.. Automated Measurement of UML Models: an open toolset approach, in Journal of Object Technology, vol 4, no. 4, MAY-June 2005,115-134.

[MAY99] MAYER, Tobias. HALL, Tracy. A critical Analysis of Current OO Design Metrics. Software Quality Journal, 8, 97-110,1999.

[NAS98a] NASA-Software Assurance Technology Center, ROSEMBERG, L. H.. Applying and Interpreting Object Oriented Metrics, 1998. http://satc.gsfc.nasa.gov/support/STC_APR98/apply_oo/apply_oo.html acesso em 13/03/2007

[NAS98b] NASA-Software Assurance Technolgy Center http://satc.gsfc.nasa.gov/metrics/codemetrics/oo/thresholds/index.html acesso em 29/12/2006

[OLA07] OLAGUE, H. M.; ETZKORN, L. H.; GHOLSTON, S.; QUATTLEBAUM, S.. Empirical Validation of three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes.

IEEE Transactions on Software Engineering, vol. 33, No. 6, June 2007, pp 402-419.

Developed Using Highly Iterative or Agile Software Development Process.

[OMG02] OMG, XML Metadata Interchange (XMI) Specification. Version 1.2, January 2002.

[OMG06] OMG, em www.omg.org acesso em 30/11/2006.

[OMG07] OMG, em www.omg.org/mda acesso em 06/01/2007.

[OMG08] OMG, em www.omg.org/mda/mda_files/GC34-2485-03_LoRes.pdf acesso em 17/06/2008

[PAT03] PATERSON, T.,RUSSEL,C.R., DEWAR, R.G.. Object-Oriented Software design Metrics from XMI, Heriot Watt University, Technical Report HW-MAC TR008, November 2003.

[PUR03] PURAO, Sandeep; VAISHNAVI, Vijay. Product Metrics for Object-Oriented Systems. 2003, ACM Computing Surveys, Vol 35, No 2, June 2003, pp. 191-221.

[QUA07] QUAH, Tong-Seng. An Implementation of Metrics Extraction and Analyses Tool. International Journal on Artificial Inteligence Tools, vol 16, No 1, February 2007, pp. 121-128.

[SCH99] SCHROEDER, Mark. A practical Guide to Object-Oriented Metrics. IT Pro November-December 1999.

[SCO04] SCOTTO, Marco, SILLITI, Alberto, SUCCI, Giancarlo, VERNAZZA, Tullio. A relational approach to software metrics. 2004 ACM Symposium on Applied Computing.

[SDM06] SDMetrics em http://www.sdmetrics.com acesso em 19/12/2006

[SEI07] Software Engineering Institute - Carnegie Mellon disponível em http://www.sei.cmu.edu/cmm acesso em 30/03/2007.

[SEI08] Software Engineering Institute - Carnegie Mellon. Model Driven Architecture (MDA) disponível em http://www.sei.cmu.edu/isis/guide/technologies/mda.htm em 06/03/2008

[SVI03] SVINICKI, M. D.. Evaluating and Grading Students. IN Teachers and Students A Sourcebook for UT-Austin faculty. Center for Teaching Effectiveness, The University of Texas, at Austin, 2003. Versão online de Teachers and Students, 1999. disponível em www.utexas.edu./academic/cte/sourcebook/ acesso em 05/02/2008.

[SWS96] SWSC, Object-Oriented Model Metrics, version 0.5, 1 September 1996 em www.cin.ufpe.br/~inspector/relacionados/Object-oriente%20Model%20Metrics%20Document.htm acesso em 24/01/2007

[VOA99] VOAS, Jeffrey. Software Quality's Eight Greatest Myths. IEEE Software, September/October 1999.

[WAK05] EL-WAKIL, M., EL-BASTAWISI, A., BOSHRA, M., FAHMY, A.. Object-Oriented Design Quality Models A survey and Comparison. Faculty of Computers and Information, Cairo University, Cairo, Egito, 2005.

[WIK07] Wikimedia Foundation. Wikipedia, the Free Encyclopedia. Disponível em http://en.wikipedia.org/wiki/List_of_UML_tools acesso em 20/06/2007.

[XEN00] XENOS, M., STAVRINOUDIS, D., ZIKOULI, K., CHRISTODOULALIS, D. Object-Oriented Metrics – A Survey. Proceedings of the FESMA 2000, Federation of European Software Measurement Associations, Madrid, Spain, 2000.

Apêndice

A1 – Descrição das métricas implementadas no APOO

Neste apêndice apresentamos as métricas propostas para APOO tomando como base os modelos propostos pelos diferentes autores considerando nesta fase do estudo aquelas que podem ser levantadas a partir da modelagem de classes.

1. Número de classes - NC
    a) Descrição: É a contagem das classes existentes no projeto.
    b) Importância: é uma métrica indicativa do tamanho do projeto. Quanto maior o número do NC maior o tamanho do projeto, indicando necessidade de maior esforço de desenvolvimento. O projeto pode apresentar maior número de relacionamentos (associação, agregação, composição, generalização, especialização) entre classes. Auxiliado por valores históricos de custo e esforço pode auxiliar nas estimativas de custo de desenvolvimento ou manutenção. Permite o cálculo de outras métricas ou indicadores que usem o número de classes em sua fórmula, principalmente, no cálculo de valores médios.
    c) Referencia: [GEN02], [BAN02], [KIM03]

2. Número de classes públicas - NCPu
    a) Descrição: É a contagem de todas as classes públicas (visíveis em todo projeto) Importância: estabelece uma indicação de como o ocultamento de classes está projetado.
    b) Referencia: [GEN02], [BAN02], [KIM03]

3. Número de classes privadas - NCPv
    a) Descrição: É a contagem de todas as classes privadas do projeto.
    b) Importância: estabelece uma indicação de como o ocultamento de classes está projetado..
    c) Referencia: [GEN02], [BAN02], [KIM03]

4. Número de classes protegidas - NCPt
    a) Descrição: É a contagem de todas as classes protegidas do projeto.
    b) Importância: estabelece uma indicação de como o ocultamento de classes ocorre.
    c) Referencia: [GEN02], [BAN02], [KIM03]

5. Número de classes de interface - NCIn
    a) Descrição: É a contagem das classes que implementam classes abstratas do projeto.
    b) Importância: estabelece uma indicação de da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

6. Número de classes abstratas - NCAb
    a) Descrição: É a contagem das classes que definem um padrão a ser implementado.
    b) Importância: estabelece uma indicação de da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

7. Número de classes de associação -NCAs
    a) Descrição: É a contagem das classes que estabelecem uma associação entre duas classes quando da multiplicidade muitos para muitos.
    b) Importância: estabelece uma indicação de da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

8. Número de relacionamentos de associação - NRAss
    a) Sigla: Descrição: É a contagem dos relacionamentos de associação entre as classes
    b) Importância: estabelece uma indicação da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

9. Número de relacionamentos de agregação - NRAgr
    a) Descrição: É a contagem dos relacionamentos de agregação entre as classes
    b) Importância: estabelece uma indicação da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

10. Número de relacionamentos de composição - NRCom
    a) Descrição: É a contagem dos relacionamentos de composição entre as classes
    b) Importância: estabelece uma indicação da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

11. Número de relacionamentos de dependência - NRDep
    a) Descrição: É a contagem dos relacionamentos de dependência(é dependente de) entre as classes.
    b) Importância: estabelece uma indicação da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

12. Número de relacionamentos de herança - NRHer
    a) Descrição: É a contagem dos relacionamentos de herança entre as classes
    b) Importância: estabelece uma indicação da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

13. Número de relacionamentos de implementação - NRImp
    a) Descrição: É a contagem dos relacionamentos de implementação entre as classes
    b) Importância: estabelece uma indicação da estrutura do projeto de classes.
    c) Referencia: [GEN02], [BAN02], [KIM03]

14. Número de Atributos - NA
    a) Descrição: contagem de todos os atributos existentes em um projeto ou classe. Inclui todos os tipos de atributos (públicos, privados e protegidos) da classe sem considerar aspectos de herança e associações.
    b) Importância: fornecer uma indicação básica do tamanho, complexidade e esforço de desenvolvimento de um projeto ou de uma classe. Um número muito alto pode indicar coesão ruim e exigir a decomposição em novas classes. Zero atributo pode indicar que é necessária uma melhor análise podendo indicar que a classe é utilitária. É uma métrica que deve ser usada com os atributos classificados por tipo para verificação da taxa de ocultamento em um projeto ou uma classe.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]
    d) Valores encontrados:
        - Domínio: 0 <= NA
        - Desejável: 2 <= NA <= 5

15. Número de Atributos Públicos - NAPu
    a) Descrição: É a contagem dos atributos públicos existentes no projeto ou em uma classe.
    b) Importância: fornecer uma indicação do tamanho da classe e seu grau de relacionamento com outras classes. Se a classe possui um número alto de atributos públicos pode ser um indicativo de que possui muito relacionamento com outros objetos e ser um ponto central ou ponto chave do projeto.
    c) Referência: [LOR94]
    d) Valores encontrados:
        - Domínio: 0 <= NAPu
        - Desejável: 2 <= NAPu <= 5

16. Número de atributos privados - NAPv
    a) Descrição: Todos os atributos privados existentes em um projeto ou classe são contados.
    b) Importância: fornecer uma indicação do ocultamento de atributos da classe. Permite o cálculo da taxa de atributos privados.
    c) Referência: [LOR94]

17. Número de Atributos Protegidos - NAPt
    a) Descrição: Todos os atributos privados existentes em um projeto ou classe são contados.
    b) Importância: fornecer uma indicação do ocultamento de atributos da classe. Permite o cálculo da taxa de atributos protegidos.
    c) Referência: [LOR94]

18. Número de Atributos de Pacote - NAPc
    a) Descrição: É a contagem dos atributos visíveis apenas, ao nível de pacote do diagrama.
    b) Importância: fornecer uma indicação do ocultamento de atributos da classe em pacotes.
    c) Referência: [LOR94]

19. Número de Métodos - NM
    a) Descrição: E a contagem dos métodos existentes no projeto. Estão incluídos os métodos públicos, privados, protegidos, estáticos e abstratos.
    b) Importância: : fornecer uma indicação do tamanho do projeto. Não são observados aspectos de herança e portanto os métodos herdados não são contados. O número de métodos junto com suas complexidades são determinantes no tempo de desenvolvimento e manutenção da

de um projeto ou de uma classe. Quando uma classe isoladamente possui um grande número de métodos ela se torna mais genérica, fornecendo muitas funcionalidades e dificultando seu reuso. Esta métrica é básica para o calculo de outras métricas de projeto consideradas como derivadas.

c) Referência: [LOR94].

d) Valores encontrados:

- Domínio: maior que zero
- Sugerido: 3 <= NM <=7.

20. Número de Métodos Públicos - NMPu.

a) Descrição: E a contagem dos métodos públicos existentes no projeto.

b) Importância: É um indicativo do ocultamento dos serviços disponíveis para outras classes. Se seu valor é alto e igual ou próximo de NM é um indicativo de mau uso do ocultamento de método no projeto. Os serviços de uma classe prestados às outras são necessáriamente públicos mas serviços próprios não necessitam ser.

c) Referência: [LOR94].

d) Valores encontrados:

- Domínio: n/e.
- Sugerido: n/e.

21. Número de Métodos Privados - NMPv

a) Descrição: Todos os métodos privados existentes em uma classe são contados.

b) Importância: É um indicativo do ocultamento dos serviços da classe.

c) Referência: [LOR94].

d) Valores encontrados:

- Domínio: n/e.
- Sugerido: n/e.

22. Número de Métodos Protegidos - NMPt

a) Descrição: Todos os métodos protegidos existentes em uma classe são contados..

b) Importância: É um indicativo do ocultamento dos serviços da classe.

c) Referência: [LOR94].

d) Valores encontrados:

- Domínio: n/e.

- Sugerido: n/e.

23. Número de Métodos abstratos - NMAb
    a) Descrição: E a contagem dos métodos abstratos existentes nas classes abstratas.
    b) Importância: Indicação da estrutura do projeto.
    c) Referência: [LOR94].

24. Média de Atributos por classe - MA
    a) Descrição: Obtém-se dividindo o total de atributos do diagrama pelo total de classes (NA/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MA alto indica maior complexidade dos testes do projeto.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03].

25. Média de Atributos Públicos por classe - MAPu
    a) Descrição: Obtém-se dividindo o total de atributos públicos do diagrama pelo total de classes (NAPu/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MAPu alto indica maior complexidade dos testes do projeto. . Juntamente com o desvio padrão pode indicar a existência de classes que merecem mais atenção.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

26. Média de Atributos Privados por classe - MAPv
    a) Descrição: Obtém-se dividindo o total de atributos privados do diagrama pelo total de classes (NAPv/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MAPv alto indica maior complexidade dos testes do projeto.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

27. Média de Atributos Protegidos por classe - MAPt
    a) Descrição: Obtém-se dividindo o total de atributos protegidos do diagrama pelo total de classes (NAPt/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MAPt alto que maioria das classes tem ocultamento de atributos
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03].

28. Média de Atributos de Pacote por classe - MAPc
    a) Descrição: Obtém-se dividindo o total de atributos de pacotes no diagrama pelo total de classes (NAPc/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MAPc alto que maioria das classes tem ocultamento de pacote
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

29. Taxa de Atributos Públicos - TAP
    a) Descrição: Obtém-se dividindo o total de atributos públicos do diagrama pelo total de atributos (NAPu/NA)
    b) Importância: fornecer uma indicação do ocultamento de atributos do projeto. Um TAP alto indica que a maioria dos atributos está oculto e serve somente às próprias classes.
    c) Referência: [GEN02]

30. Média de métodos por classe - MM
    a) Descrição: Obtém-se dividindo o total de métodos do diagrama pelo total de classes (NM/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MM alto indica que a maioria das classes possui muitos métodos o que pode indicar uma falta de coesão.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

31. Média de Métodos Públicos por classe - MMPu
    a) Descrição: Obtém-se dividindo o total de métodos públicos do diagrama pelo total de classes (NMPu/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MMPu alto indica que a maioria das classes possui métodos públicos.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

32. Média de Métodos Privados por classe - MMPv
    a) Descrição: Obtém-se dividindo o total de métodos privados do diagrama pelo total de classes (NMPv/NC)

b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MMPv alto indica que a maioria das classes possui métodos privados.
c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

33. Média de Métodos Protegidos por classe - MMPt
    a) Descrição: Obtém-se dividindo o total de métodos protegidos do diagrama pelo total de classes (NMPt/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MMPt alto indica que a maioria das classes possui métodos protegidos.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

34. Média de Métodos Abstratos por classe - MMAb
    a) Descrição: Obtém-se dividindo o total de métodos abstratos do diagrama pelo total de classes (NMAb/NC)
    b) Importância: fornecer uma indicação do tamanho médio e da complexidade do software. Um MMAb alto indica que a maioria das classes do projeto possui métodos abstratos.
    c) Referência: [GEN02], [LOR94], [SWS96], [KIM03]

35. Taxa de Métodos Públicos - TMP
    a) Descrição: Obtém-se dividindo o total de métodos públicos do diagrama pelo total de métodos (NMPu/NM)
    b) Importância: fornecer uma indicação do ocultamento de métodos do projeto. Um TMP alto indica que a maioria dos métodos está disponível para todas as classes facilitando o reuso.
    c) Referência: [GEN02]

36. Fator de ocultamento de método - FOM
    a) Descrição: É a relação entre os métodos ocultos (privados e protegidos) e o total de métodos.
    b) Importância: fornecer uma indicação do grau do ocultamento de métodos da classe (1 ou 100% indica ocultamento total).
    c) Referência: [LOR94]

37. Fator de ocultamento de atributo - FOA
    a) Descrição: É a relação entre os atributos ocultos (privados e protegidos) de uma classe e o total de atributos.

b) Importância: fornecer uma indicação do grau do ocultamento de atributos da classe em pacotes. (1 ou 100% indica ocultamento total)

c) Referência: [LOR94]

38. Número máximo de filhos - maxNF

a) Descrição: É a contagem das subclasses imediatamente subordinadas a uma classe.

b) Importância: Mostra o grau de generalização de uma classe. O número de filhos permite avaliar a influência potencial da classe no projeto. Um valor de NOC alto, indicando muitos filhos para uma classe implica em maior esforço de teste das classes embora o uso de herança seja alto. Fornece uma indicação rápida da complexidade de desenvolvimento de uma ou mais classes em função do grau de generalização. Alto numero de filhos para uma classe identifica problemas no particionamento.

c) Referência: [CHI94], [SWS96].

d) Valores encontrados:

- Domínio: 0 <= maxNF
- Desejável: 1 <= maxNF <=4

39. Acoplamento por herança - CBOh

a) Descrição: Verifica o número de especializações existentes no diagrama.

b) Importância: Mostra o grau de acoplamento de uma classe com seus ancestrais através da herança. Um valor de CBOh alto, indica um uso excessivo de herança e possivelmente um mau particionamento das classes, embora ao reuso seja um fator de melhoria da produtividade. Um valor alto é considerado indesejável.

c) Referência: [CHI94]

d) Valores encontrados:

- Domínio: 0 <= CBOh
- Desejável: 1 <= CBOh <=4

40. Acoplamento por passagem de parâmetro - CBOp

a) Descrição: É a contagem de todas as classes que são passadas como parâmetro em uma chamada de métodos de outra classe.

b) Importância: Mostra o grau de acoplamento de uma classe com outra através de parâmetros. Um valor de CBOp alto, indica um mau projeto pois aumenta a dependência entre as classes e possibilidade de falhas caso haja alguma alteração na classe acoplada.

c) Referência: [CHI94]

d) Valores encontrados:

- Domínio: 0 <= CBOp
- Desejável: 1 <= CBOp <=4

41. Profundidade máxima na árvore de herança - maxPAH

a) Descrição: É o comprimento do caminho desde a classe até a raiz da árvore de herança. A raiz é considerada nível 0 (zero).

b) Importância: fornecer uma indicação básica e rápida da complexidade do projeto. Quanto mais profunda a classe estiver na árvore de herança mais difícil se torna prever o seu comportamento e maior é sua complexidade no desenvolvimento, embora seu potencial de reutilização seja maior pelo fato de herdar das classes ancestrais. O maior DIT do projeto identifica a classe que tem maior herança e que solicitará mais em termos de reutilização ou indica uma classe muito especializada.

c) Referência: [CHI94], [GEN02], [NAS98b]

d) Valores encontrados:

- Domínio: >= 2
- Desejável: 2<= maxPAH <= 5

A2 - Estudo de caso – Controle de Manutenção de Veículos

A2.1 Especificação preliminar de requisitos

Engenharia de Software
2º semestre de 2006

Projeto de software

1. Proposta de trabalho

   Projetar e construir um software para controle do processo de manutenção de veículos em oficina mecânica conforme especificações preliminares abaixo:
   A Manauto Ltda é uma empresa de manutenção de veículos multimarca que opera com três unidades em diferentes locais da cidade: unidade de veículos de passeio, unidade de veículos de trabalho (utilitários, caminhões leves e médios), e unidade de motocicletas. O escritório administrativo funciona junto à unidade de veículos de passeio.
   Deseja-se montar um sistema integrado de controle de serviços de manutenção de veículos que atenda às diversas unidades e com as seguintes funcionalidades:

   - Cadastro dos fabricantes dos veículos com os quais a empresa trabalha;
   - Cadastro de marca, modelo, tipo do veiculo, etc.;
   - Cadastro dos funcionários da empresa responsáveis por serviços de oficina;
   - Cadastro de serviços e respectivos preços;
   - Registro dos serviços efetuados nos veículos dos clientes;
   - Emissão de informações de andamento dos serviços nos veículos dos clientes incluindo a possibilidade de consulta via Web.

2. Condições
   - O desenvolvimento deverá ser feito usando modelagem UML na ferramenta CASE ArgoUML e feita a implementação da parte cliente-servidor em ambiente Java;
   - Deverão ser postadas no Learnloop, versões preliminares do diagrama de classes do projeto no formato XMI de acordo com o cronograma do projeto a ser distribuído no laboratório de engenharia de software.
   - O acompanhamento do trabalho será feito semanalmente com entrega de relatório de atividades realizadas na semana
   - Dúvidas serão esclarecidas durante o desenvolvimento;
   - O projeto está aberto a novas idéias e sugestões
   - O trabalho deverá ser feito preferencialmente em duplas

3. Avaliação
   O valor total do trabalho é de 20 pontos distribuídos conforme discriminado abaixo:

   a) 10 pontos pela apresentação do software no laboratório onde serão observados:
      - Atendimento dos requisitos mínimos propostos, avaliação da qualidade da interface com usuário, originalidade (cópias não serão pontuadas) e criatividade;
      - Entrega de CD-ROM contendo todo o projeto (fontes, executáveis, diagramas UML, manual do usuário, scripts de tabelas e textos de marcação referente ao projeto WEB);
   b) 10 pontos pela documentação do projeto com UML. Entregar em papel A4 e observando normas de redação, de acordo com as normas de redação de trabalho científico recomendado ABNT e disponível no sitio da Biblioteca.

c) Trabalhos fora do prazo terão penalidade de 5% por dia de atraso até o limite máximo de dois dias.

A2.2 Resultados obtidos

Nove projetos foram utilizados para o primeiro teste de APOO. A Tabela A2.1 apresenta as métricas obtidas. Exibe também os valores de um projeto de referência (PR) fornecido pelo avaliador.

Características	Tema: Sistema de Controle de Manutenção de Veículos										
	Métricas	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	PR
Classes do projeto	NC	15	13	32	11	23	7	8	25	17	8
Encapsulamento de classes	NCPu	15	13	32	11	23	7	8	25	17	8
	NCPv	0	0	0	0	0	0	0	0	0	0
	NCPt	0	0	0	0	0	0	0	0	0	0
	NCPc	0	0	0	0	0	0	0	0	0	0
Aassociações de classes	NCAs	0	0	0	0	0	2	0	0	0	1
	NCIn	0	1	0	0	0	0	0	0	0	0
	NCAb	0	0	0	0	0	0	0	0	0	0
Relacionamentos de classes	NRAss	41	43	38	38	11	33	17	26	22	5
	NRAgre	0	0	0	0	0	1	0	0	0	0
	NRCom	0	0	0	0	0	1	0	0	0	0
	NRDep	0	0	0	0	1	3	0	0	0	0
	NRHer	0	1	0	0	10	1	2	0	6	0
	NRImp	0	0	0	0	0	0	0	0	0	0
Atributos do projeto	NA	67	45	56	40	42	14	26	67	48	37
	NAPu	67	44	56	37	41	14	26	60	48	8
	NAPv	0	1	0	3	1	0	0	7	0	0
	NAPt	0	0	0	0	0	0	0	0	0	29
	NAPt	0	0	0	0	0	0	0	0	0	0
Métodos do projeto	NM	40	36	40	40	32	0	0	36	8	34
	NMPu	40	36	40	40	31	0	0	36	8	27
	NMPv	0	0	0	0	1	0	0	0	0	0
	NMPt	0	0	0	0	0	0	0	0	0	7
	NMAb	0	0	0	0	0	0	0	0	0	0
Herança	maxNF	0	1	0	0	3	0	2	0	3	0
	MaxPAH		1			1		1		1	0
Acoplamento	CBOh	0	1	0	0	10	1	2	0	6	0
	CBOp								54	0	3
Ocultamento	FOM	0%	0%	0%	0%	3%	0%	0%	0%	0	21%
	FOA	0%	2%	0%	8%	2%	0%	0%	10%	0	78%

Tabela A2.1 – Métricas Obtidas para o Sistema de Controle de Manutenção de Veículos.

A seguir são apresentadas análises sobre algumas das métricas propostas classificadas por

categorias.

A2.2.1 Métricas relativas a tamanho e complexidade

Estas métricas fornecem informação básica do tamanho e complexidade do projeto. São denominadas ***primitivas***, pois derivam diretamente da contagem dos elementos presentes. Elas servem de bases, para o cálculo de outras métricas denominadas ***derivada***s e usadas para avaliação de qualidade e obtenção de valores médios [KIM03].

A. Número de classes (NC) - esta métrica indica o tamanho do modelo identificado pelos desenvolvedores e suas percepções relativas ao escopo do projeto e conseqüentemente o esforço de construção do software.

A variação nos valores encontrados para a métrica NC pode ser observada na Figura A2.1. Era de se esperar uma maior uniformidade, pois os desenvolvedores possuem um mesmo nível de conhecimento. Note que distorções podem ser indicativas de dificuldades no entendimento do escopo do projeto.

Conforme valores obtidos, apenas três projetos se aproximam do padrão de referência (PR). Outro fato observado é a presença de classes sem identificação, o que eleva o número de classes indicando descuido de modelagem.

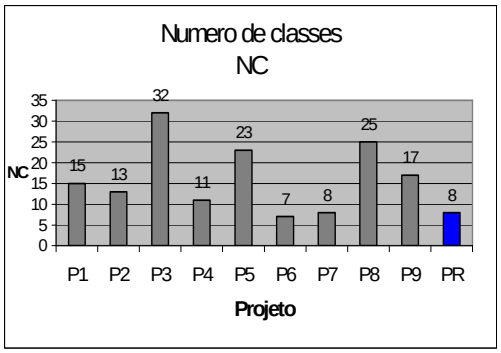


Figura A2.1 - Métrica NC.

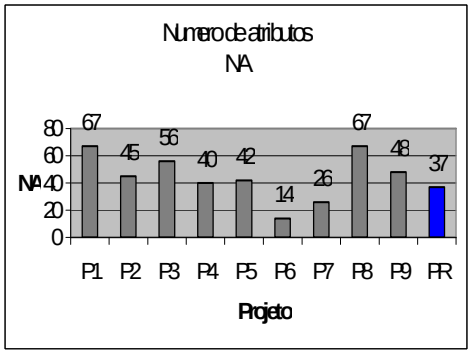


Figura A2.2 - Métrica NA.

B. Número de atributos (NA) - esta métrica (Figura A2.2) é essencial para a verificação de fatoração das classes além de aspectos de ocultação de informação. Observa-se uma grande variação nos valores encontrados - mínimo de 14, máximo de 67, com amplitude 53. É de se esperar que o aumento no número de classes implique em um aumento no número de atributos. Entretanto, uma relação NA/NC muito alta é um indicativo que o desenvolvedor não fatora/particiona as classes para melhor reuso via herança ou associação. No caso de uma relação baixa é um indicativo de

classes subutilizadas. É importante para o aperfeiçoamento das abstrações.

C. Número de métodos (NM) - Não ocorre grande variação nos valores encontrados com exceção do projeto P9 (Figura A2.3). Os projetos P6 e P7 são casos típicos de descuido na modelagem, uma vez que nenhum método é disponibilizado nas classes de negócio. Outro aspecto a ser considerado é a relação NM/NC. Pode-se observar a existência de classes sem métodos quando a relação é menor que um (1), como no projeto P_9.

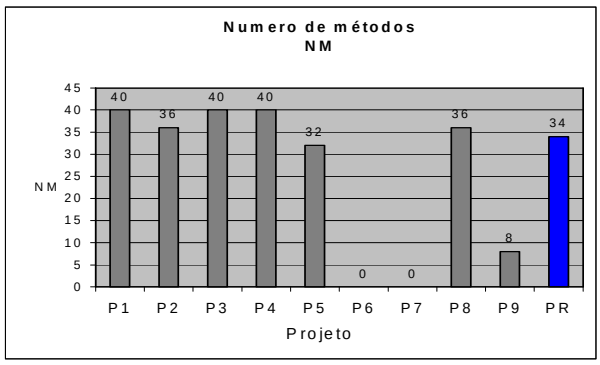


Figura A2.3 - Métrica NM.

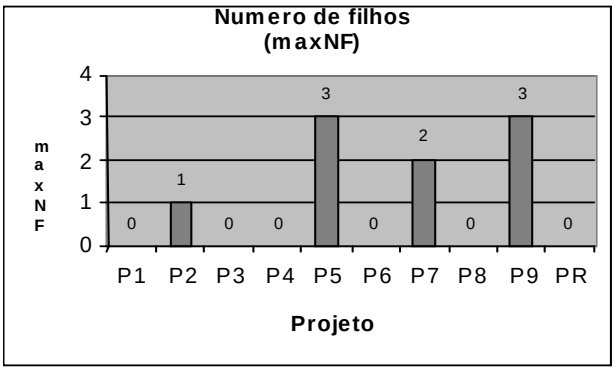


Figura A2.4 - Métrica maxNF

A2.2.2. Métricas relativas à herança

D. Número máximo de filhos (maxNF): indica um grau de reuso desejável. Define a divisão em subclasses. Um valor muito alto pode indicar excesso de classes especializadas. Além disso, pode indicar centralização de responsabilidades em poucas classes. Pode ser observado (Figura A2.4) que apenas quatro projetos (P_2, P_5, P_7 e P_9) consideram a especialização para modelagem do problema, isto é, número de filhos maior que 0 (zero), tirando benefícios da reutilização de classes.

Profundidade máxima da árvore de herança (maxPAH): esta métrica permite verificar o grau de reutilização de classes no projeto. Quanto maior a profundidade da classe na hierarquia, maior a complexidade para a correta implementação e testes. Um maior esforço deve ser despendido para o entendimento dos componentes disponibilizados pelas classes ancestrais.

Observa-se pela Figura A2.5 que apenas quatro projetos estabeleceram maxPAH com valores aceitáveis (capítulo 2 seção 2.6). Os demais não consideraram ou não identificaram a necessidade de especializações.

É possível notar a relação entre maxPAH e maxNF. Valores altos para maxNF indicam um diagrama com grande amplitude o que sugere uso inadequado de especialização – uma única classe com um número elevado de descendentes diretos. Valores altos para maxPAH indicam uso excessivo de herança – um número elevado de descendentes.

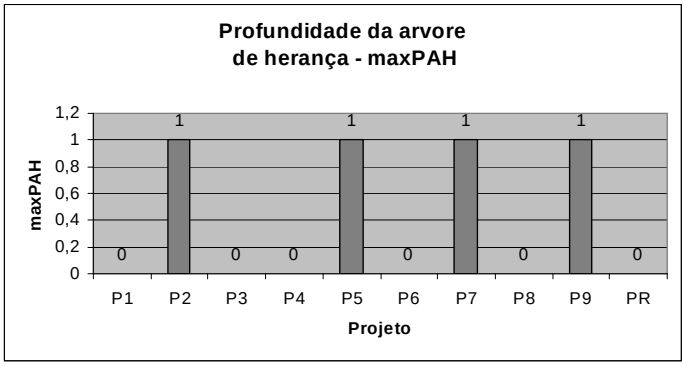


Figura A2.5 – Métrica maxPAH.

A2.2.3 Métricas relativas a acoplamento

As métricas de acoplamento indicam o grau de dependência entre classes. Aqui a preocupação deve ser a de manter o nível de acoplamento baixo.

O acoplamento acontece por herança, por passagem de classes como parâmetro numa mensagem, por utilização de métodos ou de atributos de outra classe. No diagrama de classes o acoplamento se manifesta via herança e assinatura dos métodos. [CHI94, HAR98a, OLA07]

Acoplamento por herança - CBOh: é obtido pela contagem do número de especializações no diagrama.

Pode-se observar (Figura A2.6) que o projeto P_5 apresenta um alto valor para CBOh. Tal fato facilita a reutilização, mas deve ser analisado com cuidado para se evitar particionamento excessivo de classes criadas desnecessariamente.

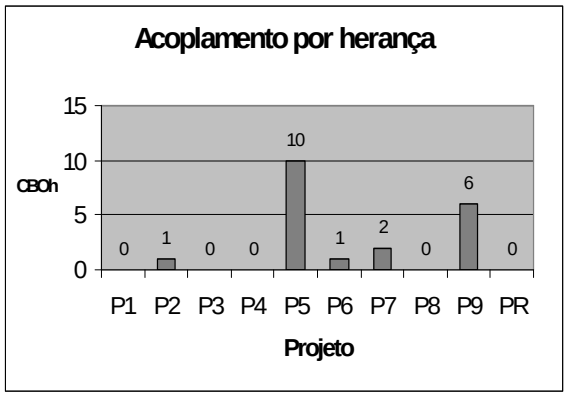


Figura A2.6 –Métrica CBOh

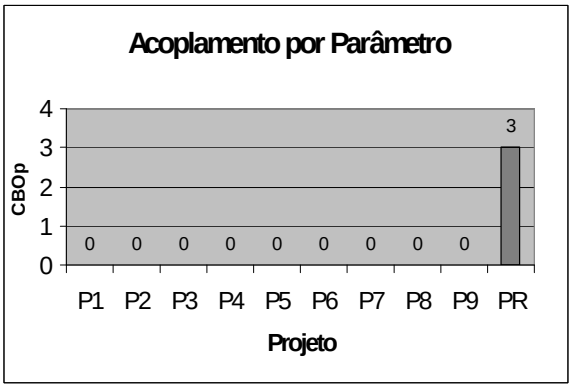


Figura A2.7 – Métrica CBOp

Acoplamento por parâmetro - CBOp: é obtido pela contagem das classes presentes na assinatura dos métodos. Nenhum projeto apresenta acoplamento por parâmetro (Figura A2.7), este fato indica a existência de dificuldades na identificação de passagem classes como parâmetro

A2.2.4 Métricas relativas a encapsulamento

As métricas de encapsulamento indicam se o desenvolvedor teve a preocupação de definir quais atributos e métodos podem ser visíveis dentro e fora da classe. Fornece, portanto uma medida do ocultamento de atributos e métodos. [BRI96, HAR98a, OLA07].

Recomenda-se que os fatores de ocultamento de atributos sejam altos indicando que os atributos devem ser privados ou protegidos. Já os métodos devem ser compartilhados com outras classes e, portanto, devem ser públicos implicando em fatores com baixos valores.

Fator de Ocultamento de atributo-FOA: indica o grau de ocultamento de atributos de todo o projeto. É a relação entre atributos ocultos para as outras classes (privados e protegidos) e o total de atributos do projeto.Observa-se pela Figura A2.8 que não houve, durante o projeto, uma análise mais profunda do aspecto de ocultamento de atributos (o maior valor é igual a 10%) e cinco dos projetos consideraram todos atributos como públicos (valor igual a 0 %).

Fator de Ocultamento de Método - FOM: indica o valor do fator de ocultamento de métodos para todo o projeto. É a relação entre métodos ocultos (privados e protegidos) e o total de métodos.

Verifica-se pela Figura A2.9 que apenas um projeto considerou ocultamento de método. A

recomendação é de que os métodos de uma classe atendam às outras classes do projeto. Um valor alto de FOM significa que a maioria dos métodos é do tipo público. FOM igual a 0% para todas as classes do projeto indica uma má qualidade.

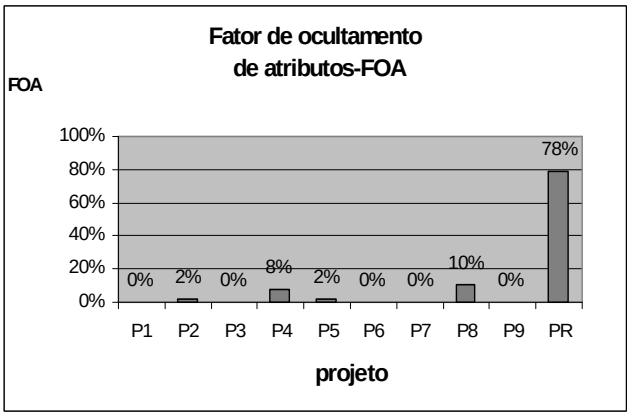


Figura A2.8 – Métrica FOA

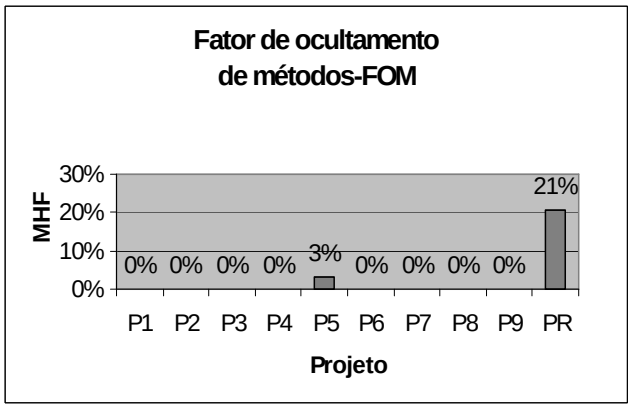


Figura A2.9 – Métrica FOM

A2.3 Métricas derivadas

Estas métricas são derivadas das métricas primitivas e permitem avaliar a relação entre os diversos elementos através de valores médios ou uma taxa obtida pelo relacionamento de duas ou mais métricas. Nas Figuras A2.10 e A2.11 observa-se que a maioria dos projetos apresenta atributos de classe como públicos não levando em consideração aspectos de ocultamento.

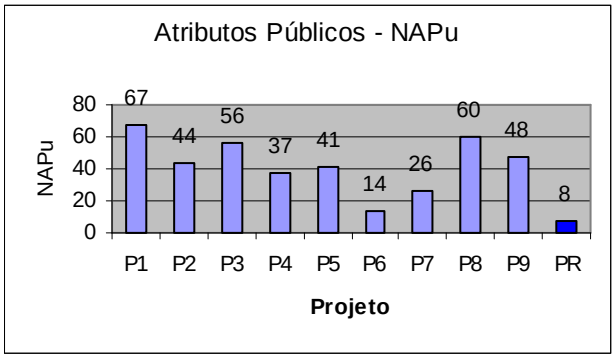


Figura A2.10 – Métrica NAPu

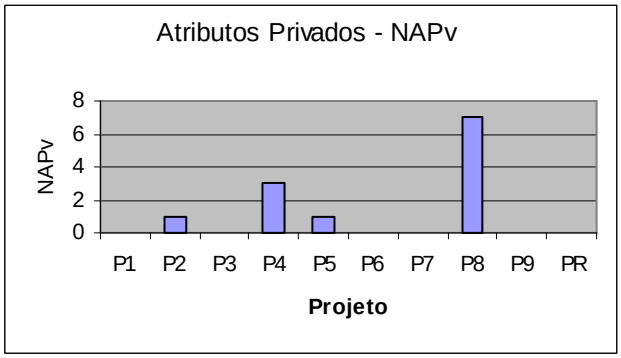


Figura A2.11 – Métrica NAPv

Pela Figura A2.12 o valor médio dos métodos das classes encontra-se abaixo do padrão e abaixo do valor 4 que seria o mínimo para as classes de persistência do negócio.

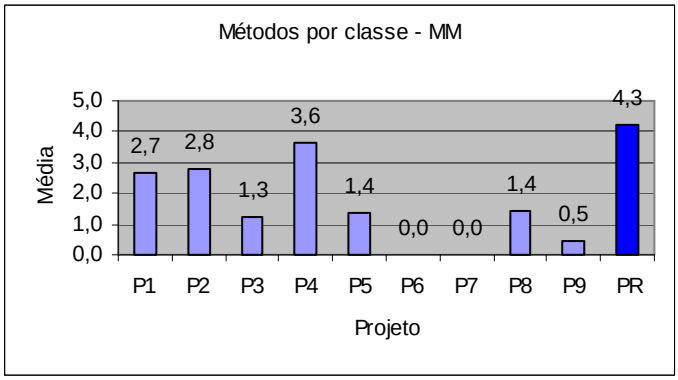


FIGURA A2.12 – Métrica MM

A2.4 Avaliações dos Projetos

APOO possibilita a avaliação automática dos projetos submetidos efetuando comparações com medidas de tendência central, ou em relação a um projeto de referência ou, então, em relação aos valores previamente catalogados na literatura (Capítulo 2) considerando todas as métricas com igual peso (peso 1). As subseções seguintes apresentam os resultados obtidos.

A2.4.1 Avaliação com Medidas de Tendência Central

Para esta forma de avaliação são realizados cálculos estatísticos para obtenção dos valores de referência tomando-se, como universo, todos os projetos apresentados. São obtidas medidas de tendência central e erro de estimativa (Tabela A2.2).

	Estatística					
Métricas	Mediana	Média	Desvio	Erro	LI	LS
NC	15	17	8	3	9	25
NCPu	15	17	8	3	9	25
NCPv	0	0	0	0	0	0
NCPt	0	0	0	0	0	0
NCPc	0	0	0	0	0	0
NCAs	0	0	1	0	0	1
NCInb	0	0	0	0	0	0
NCAb	0	0	0	0	0	0
NRAss	33	30	11	4	19	41
NRAgr	0	0	0	0	0	0
NRCom	0	0	0	0	0	0
NRDep	0	0	1	0	0	1
NRHer	1	2	3	1	0	5
NRImp	0	0	0	0	0	0
NA	45	45	18	6	27	63
NAPu	44	44	17	6	27	61
NAPv	0	1	2	1	0	3
NAPt	0	0	0	0	0	0
NAPc	0	0	0	0	0	0
NM	36	26	18	6	8	44
NMPu	36	26	18	6	8	44
NMPv	0	0	0	0	0	0
NMPt	0	0	0	0	0	0
NMAb	0	0	0	0	0	0
MaxNOC	0	1	1	0	0	2
CBOh	1	2	3	1	0	5
CBOp	0	0	0	0	0	0
MaxDIT	0	0	1	0	0	1
FOM	0	0%	1%	0	0%	1%
FOA	0	3%	4%	0	0%	6%

Tabela A2.2 – Medidas Estatísticas.

A Tabela A2.3 apresenta os resultados da avaliação. A nota final corresponde ao percentual dos acertos obtidos tomando como base os pontos médios de cada métrica.

Pontos:								
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
7	15	7	11	13	3	4	8	11
Notas								
4,4	9,4	4,4	6,9	8,1	1,9	2,5	5	6,9

Tabela A2.3 – Avaliação dos Projetos

Utiliza-se uma Distribuição Normal centrada na média e desvio-padrão do grupo. Foi estabelecido com um desvio padrão em torno da média, um intervalo de aceitação com limite inferior (LI) e superior (LS). Avalia-se, comparando o valor de cada métrica do projeto com os valores do intervalo. O padrão é a atribuição de pesos iguais para todas as métricas avaliadas. A pontuação final é a soma ponderada das métricas, com peso 1 (um) para as que se situarem no intervalo e 0 (zero) para aquelas fora do mesmo.

Na Tabela A2.4 são exibidas as notas atribuídas pelo avaliador sem o uso de APOO. Observam-se diferenças nas notas do avaliador em 4 dos 9 projetos. P_2 e P_5 obtêm notas menores e P_7 e P_8 maiores. Neste caso, a nota do avaliador não levou em consideração os resultados obtidos pela ferramenta pois sua avaliação foi feita manualmente e considerou outros aspectos do projeto como a documentação, implementação e apresentação, servindo apenas para verificação da nota de APOO em relação `situação real.

Projeto	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
Nota do avaliador	5	5	6	7	5	3	7	9	5
Nota APOO	4,4	9,4	4,4	6,9	8,1	1,9	2,5	5	6,9

Tabela A2.4 – Comparação Avaliador X APOO

Além disso, este modelo de avaliação está sujeito à ocorrência de valores extremos, muito altos ou muito baixos, que acabam por influenciar a média e o desvio padrão. Observa-se também que a mesma ponderação para todas as métricas pode não ser a ideal. Tal fato pode ser usado para avaliações parciais de modo a uniformizar o entendimento dos grupos de projeto acerca de conceitos ainda não consolidados. Por exemplo, de classe, de herança, de acoplamento, de associação e outros, podendo-se assim reduzir a influência de valores extremos obtendo-se um desvio menor em torno da média.

A2.4.2 Avaliação usando o Padrão de Referência

Para esta forma de avaliação cada métrica é comparada com a de um ***projeto padrão*** de referência fornecido ao APOO pelo avaliador. Este define uma faixa de tolerância em torno do valor considerado padrão. São consideradas, para efeito de avaliação, apenas as que se situam dentro do intervalo de tolerância. A nota é o percentual das métricas no intervalo. A Tabela A2.5 mostra os valores utilizados para o projeto de referência.

Valores do PR							
Métrica	PR	LI	LS	Métrica	PR	LI	LS
NC	8	3	13	NAPu	8	3	13
NCPu	8	3	13	NAPv	0	0	5
NCPv	0	0	5	NAPt	29	24	34
NCPt	0	0	5	NAPc	0	0	5
NCPc	0	0	5	NM	34	29	39
NCAs	1	0	6	NMPu	27	22	32
NCIn	0	0	5	NMPv	0	0	5
NCAb	0	0	5	NMPt	7	2	12
NRAss	5	0	10	NMAb	0	0	5
NRAgr	0	0	5	maxNF	0	0	5
NRAss	0	0	5	CBOh	0	0	5
NRDep	0	0	5	CBOp	3	0	8
NRHer	0	0	5	maxPAH	0	0	5
NRImp	0	0	5	FOM	21%	0%	23%
NA	37	32	42	FOA	78%	0%	86%

Tabela A2.5 – Valores do Projeto de Referência

A tabela A2.6 exibe nova avaliação dos projetos frente ao modelo de referência. Neste caso os resultados demonstram que a avaliação, com o projeto de referência torna o processo mais rígido em relação à média. O uso de um padrão permite ao Avaliador calibrar o intervalo considerando diferentes aspectos relevantes ao projeto - amplia (ou reduz) o intervalo de tolerância de acordo com considerações obtidas pelo próprio grupo.

A mesma ponderação para todas as métricas, aqui também, pode não ser a ideal. A diferenciação de pesos poderia ser introduzida em estágios mais avançados do projeto exigindo-se maior rigor na modelagem das classes.

As notas obtidas com o padrão de referência foram, em geral, menores que ao se usar a média dos projetos. A avaliação com projeto de referência estabelece faixas de aceitação mais

estreitas em relação à média. Entretanto, P_6 e P_7, quando avaliados com o projeto de referência obtêm notas maiores. A medida de tendência central não possibilita a avaliação de métricas cujos valores são destoantes da média, mas em um projeto de referência estas mesmas métricas podem estar dentro da faixa de aceitação.

Pontos dos projetos usando o Padrão de Referência									
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	PR
2	7	2	7	7	5	5	4	2	16
Nota do Avaliador									
5	5	6	7	5	3	7	9	5	
Nota APOO usando a média									
4,4	9,4	4,4	6,9	8,1	1,9	2,5	5	6,9	
Nota APOO usando o padrão de referência									
3	6	3	6	6	4	4	4	3	10

Tabela A2.6 – Avaliação de Tendência Central x Projeto de Referência

A2.4.3 Avaliação com Valores da Literatura

Nesta avaliação o valor de cada métrica do projeto é comparado com valores mínimo e máximo recomendados por diferentes pesquisadores. No caso de ocorrência de diferentes valores para a mesma métrica, utiliza-se o de maior amplitude por permitir flexibilidade na avaliação do projeto.

Os resultados são obtidos a partir do intervalo de aceitação. Como o limite inferior, para algumas métricas, não é determinado assume-se o valor zero. Nesta avaliação, os valores de métricas são aceitos se pertencentes ao intervalo estipulado – determinado por um limite superior obtido na literatura. Os resultados são mostrados na Tabela A2.7.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	PR
Pontos usando valores da literatura:									
8	14	9	11	15	12	15	8	16	15
Nota do Avaliador									
5	5	6	7	5	3	7	9	5	
Nota APOO com valores da literatura									
3	6	4	5	6	5	6	3	7	6

Tabela A2.7 – Notas usando valores da literatura

Observa-se que a avaliação final dos projetos, novamente, não coincide com as do avaliador. Entretanto, os valores obtidos pela avaliação da ferramenta estão com distribuição uniforme. Este

fato certamente deriva da maior amplitude da faixa de aceitação obtida na literatura. A avaliação utiliza os limites inferiores e superiores constantes da tabela A2.8.

Valores da Literatura					
Métrica	LI	LS	Métrica	LI	LS
NC	2	29	NAPu	0	56
NCPu	0	0	NAPv	0	56
NCPv	0	0	NAPt	0	56
NCPt	0	0	NAPc	0	56
NCPc	0	0	NM	1	20
NCAs	0	0	NMPu	1	20
NCIn	0	0	NMPv	1	20
NCAb	0	0	NMPt	1	20
NRAss	1	20	NMAb	1	20
NRAgr	0	9	maxNF	1	4
NRAss	0	0	CBOh	1	4
NRDep	0	4	CBOp	1	4
NRHer	0	24	maxPAH	1	4
NRImp	0	0	FOM	8%	25%
NA	4	56	FOA	44%	68%

Tabela A2.8 – Valores de Referência da Literatura

A avaliação com o uso de valores da literatura é uma tentativa de se verificar se um projeto, classificado por tamanho (classes, atributos e métodos), apresenta valores de métricas aceitáveis frente a diferentes projetos já avaliados.

Uma comparação entre os resultados obtidos pelos três critérios anteriores é mostrada na Figura A2.13. Com o uso de um projeto de referência (PR) a avaliação se apresenta conservadora conforme discutido anteriormente.

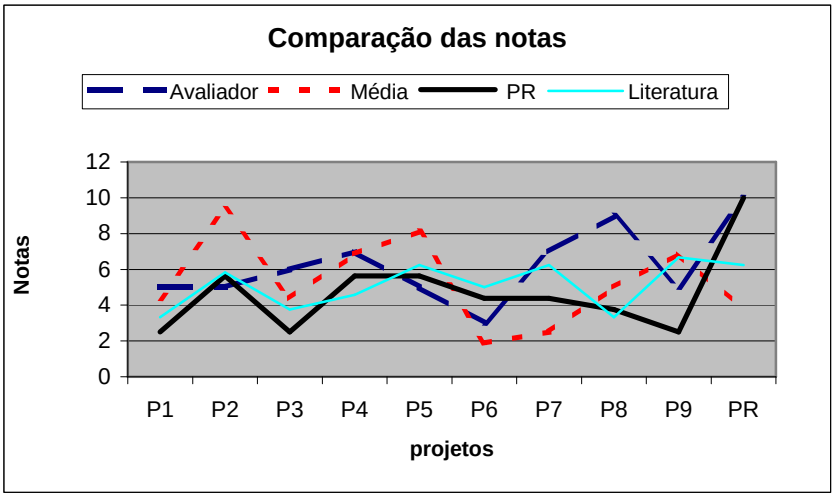


Figura A2.13 – Comparação dos resultados

Esse aspecto permite ao avaliador rever os critérios adotados na definição do intervalo de tolerância estabelecendo um maior ou menor rigor no desenvolvimento do projeto.

A avaliação pela média é mais flexível e pode não ser a ideal se o grupo se mostra heterogêneo, pois há uma grande variação nos intervalos de tolerância. Entretanto, permite orientar o grupo quanto a distorções detectadas pela ferramenta. Este fato possibilita o avaliador reforçar conceitos ainda não consolidados

Interessante observar que os pontos de inflexão demonstram que, salvo as exceções, um projeto bem avaliado por determinado modelo tende a seg uir o mesmo processo frente aos demais.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo