O IMPACTO DA UTILIZAÇÃO DE DESIGN PATTERNS NAS MÉTRICAS E ESTIMATIVAS DE PROJETOS DE SOFTWARE: a utilização de padrões tem alguma influência nas estimativas?

Ricardo Alexsandro de Medeiros Valentim[1]
Plácido Antônio de Souza Neto[2]

RESUMO

O presente artigo é uma pesquisa exploratória e investigativa que abordou alguns estudos que permitiram avaliar os impactos dos Designs Patterns sob as métricas e estimativas de sistemas orientados a objetos.

Palavras-chave: Desing Patterns. Desenvolvimento de Softwares – métricas. Engenharia de Software.

THE USE IMPACT OF DESIGN PATTERNS ON THE METRICS AND ESTIMATES OF SOFTWARE PROJECTS: "does the use of patterns influence the estimates?"

ABSTRACT

This paper aims to evaluate, through an exploratory and investigating research, the impacts of Design Patterns on metrics and estimates of object-oriented systems.

Keywords: Design Patterns. Software Development– metrics. Software Engineering.

[1] Bacharel em Sistemas de Informação (FARN) e Professor do CEFET-RN. E-mail: valentim@cefetrn.br
[2] Professor do CEFET-RN. E-mail: placidoneto@cefetrn.br

1 INTRODUÇÃO

A indústria de sistemas utilizando-se da Engenharia de Software tem se desenvolvido bastante nos últimos tempos. Este crescimento se justifica devido a crescente demanda por sistema que venham a resolver problemas da atual sociedade. Diante deste contexto, os indivíduos envolvidos na produção de software perceberam que muitos dos problemas de projetos de sistemas eram recorrentes. Este foi um fato importante, pois a partir deste ponto surgiram os Designs Patterns (Padrões de Projetos) que são aplicados a problemas clássicos na produção de softwares.

Verificando a importância e o crescimento da produção de sistemas computacionais, a Engenharia de Software cria técnicas com vistas a medir o tamanho de um sistema. O objetivo destas medições é encontrar métricas que possibilitem estimar os recursos e os custos necessários para o desenvolvimento de softwares.

Diante do contexto exposto, este trabalho teve como objeto de estudo, a realização de uma pesquisa onde o foco foi investigar o que tem sido estudado quanto à aplicação de métricas em projetos de desenvolvimento de softwares que utilizam designs patterns. Assim, possibilitando verificar qual o impacto que a adoção de designs patterns ocasiona sobre as métricas de um sistema, e conseqüente sobre suas estimativas.

O presente artigo utilizou uma metodologia de pesquisa exploratória e investigativa. Com isso, identificando alguns estudos que permitiram avaliar o relacionamento entre a utilização de designs patterns, métricas e estimativas de sistemas. Sendo este um aspecto substancial, pois pode contribuir para que arquitetos e engenheiros de softwares tenham dados de ponderação, quando estes optem pela à aplicação de Designs Patterns em seus projetos, pois poderão adequar às métricas e estimativas.

As seções dois e três tratam da fundamentação teórica deste artigo, abordando as temáticas pertinentes ao seu objeto de estudo. A seção quatro trata do desenvolvimento de um estudo de caso referente às métricas aplicadas a Design Patterns. Por fim, a seção cinco trata das considerações finais e dos possíveis trabalhos em perspectiva.

2 ENGENHARIA DE SOFTWARE

A engenharia de software é um ramo da engenharia de sistemas preocupada com desenvolvimento de software grandes e complexos (FINKELSTEIN; KRAMER, 2000, p. 4). Estudando além de técnicas de desenvolvimento de software, a administração de projetos de software, criação de ferramentas, teorias e métodos que dêem suporte a criação de sistemas computacionais.

Segundo Finkelstein; Kramer (2000, p. 4): "A Engenharia de Software tem obtido uma maior maturidade, e suas pesquisas tem mudado, pois têm sido orientadas a problemas reais da indústria."

Há muitas definições propostas referente à Engenharia de Software, sempre enfatizando a sua importância no desenvolvimento de sistemas, destacando os métodos, as ferramentas e os procedimentos como elementos fundamentais.

Com base em Presman (1995, p. 31):

- **Método:** Os métodos proporcionam os detalhes de com fazer para construir um software. Os métodos permeiam uma série de atividades como: planejamento, estimativa, análise de requisitos de sistemas, codificação, teste, manutenção;
- **Ferramentas:** São criadas para apoiar de maneira automatizada ou semi-automatizada os métodos. Atualmente já existem varias ferramentas bastantes populares (*freeware* e proprietárias);
- **Procedimentos:** Realizam a coesão entre as ferramentas e os métodos, assim provendo o desenvolvimento racional e oportuno dos sistemas computacionais.

A Engenharia de Software assim como as demais engenharias, tem desenvolvido processos de produção que viabilizam a qualidade do produto. Para Sommerville (2004, p.36) "Processos de Software é um conjunto de atividades e resultados associados que levam a produção de um produto de software".

Norteado por Sommerville (2004, p. 36): apesar dos diversos processos de software, existem atividades fundamentais comuns à maioria deles, tais como:

- **Especificação de software:** definição das funcionalidades do software e restrições em sua operação;
- **Projeto e implementação de software**: deve ser produzido o software de acordo com sua especificação;
- **Validação de software**: deve ser validado para garantir a conformidade dos requisitos solicitados pelo cliente;
- **Evolução de Software:** é necessária a evolução do software para atender as necessidades mutáveis do cliente.

Os processos de software agregaram ainda mais importância a Engenharia de Software, uma vez que estes contribuem para o desenvolvimento de sistemas de maneira fabril, tal como nas outras engenharias já tradicionais. Porém existe um outro fator relevante, que é a medição de sistemas – como medir o tamanho de um software? Para Pressman (1995, p. 59) "medir é fundamental em qualquer disciplina de engenharia, e a Engenharia de Software também não é uma exceção". Neste sentido, a medição de um software é um fator essencial em projetos, pois torna possível estimar custos e esforços que deverão ser empregados no desenvolvimento de sistema. A Engenharia de software disponibiliza vários métodos de métricas e estimativas nos quais serão discutidos na seção subseqüente.

2.1 MÉTRICAS E ESTIMATIVAS

Medir ou tentar estimar o tamanho de um software é uma das partes essenciais do planejamento na construção de sistemas, isso porque, através de

medições ou de estimativas pode-se aferir o tempo, o esforço e os recursos necessários para o desenvolvimento de softwares.

Os autores Fenton; Neil (2002, p.360) fazem um questionamento clássico, no qual ilustra bem a importância de se medir um sistema na fase de planejamento:

> O modelo prediz que necessito de quatro pessoas trabalhando durante dois anos para construir um sistema deste tipo e deste tamanho. Mas somente posso financiar três pessoas trabalhando por um ano. Se não puder sacrificar a qualidade, quão bom deve ser a equipe de funcionários do projeto para construir o sistema com os recursos limitados?

Tomando por base os aspectos supramencionados por Fenton; Neil (2002), pode-se verificar que as métricas de um sistema, podem influir em questões de tomadas decisões, onde será possível visualizar cenários de maior ou de menor risco. Por exemplo, para o mesmo questionamento, a resposta talvez fosse que em um ano, mesmo com um quadro funcional excepcional existiria uma grande possibilidade de ultrapassar o prazo inicial previsto para entrega do sistema.

Este contexto remete a um ponto chave no processo de medição de tamanho de software e consequentemente ao de planejamento, que é o de identificar um modelo de métrica que permita estimar de forma mais consistente o tempo e o esforço necessário à produção de um dado sistema. Na subseção seguinte são discutidas métricas utilizadas na construção de sistemas orientados a objetos.

2.2 MÉTRICAS PARA SISTEMAS ORIENTADOS A OBJETOS

Existem atualmente duas visões referentes a métricas para projetos de software, uma aplicada aos projetos de softwares tradicionais, ou estruturados e uma outra visão direcionada a projetos de softwares orientados a objetos.

Segundo Chidamber; Kemerer (1994, p.476-477) existem dois tipos gerais de visões que podem ser aplicados atualmente em métricas de software. A primeira categoria tem uma base teórica voltada a softwares estruturados, ou seja, projetos de software não orientado a objetos. A segunda categoria é mais específica ao desenvolvimento de projetos orientados a objetos (OO). Sendo estes projetos OO uma abordagem que tem como foco uma implementação baseada em um modelo do mundo real em termos de seus objetos. Portanto, sendo uma antítese as abordagens mais tradicionais que são orientadas a funções, deste modo separando dados e procedimentos.

O contexto dos Designs Patterns envolve especificamente a orientação a objetos, desta forma, o presente trabalho será balizado nas metodologias de métricas que são especificas a OO. Efetivamente, considerando a série de métricas propostas por Chidamber; Kemerer (1994) em seu artigo "*A Metrics Suite for Object Oriented Design*".

2.2.1 Métricas fundamentadas na orientação objetos

Existem muitas metodologias para projetos orientados a objetos. Chidamber; Kemerer (1994, p. 477) fazem menção a quatro etapas as quais consideram essenciais na construção de projetos orientados a objetos:

- Identificação das classes e dos objetos;
- Identificar a semântica das classes e objetos;
- Identificar a semântica entre as classes e os objetos;
- Implementação das classes e objetos.

Estes preceitos da orientação a objetos fizeram com que surgissem tipos de métricas especificas a OO, os autores Chidamber; Kemerer (1994, p. 482-486), sugerem as seguintes métricas:

- **WMC** (*Weighted Methods Per Class*): quantidade de métodos de uma determinada classe, considerando-se que as complexidades dos métodos são iguais;
- **DIT** (*Depth of Inheritance Tree):* profundidade da árvore de herança, isto é, quantas classes ancestrais podem afetar potencialmente uma determinada classe (Se o seu valor for igual a um, afirma-se que os testes nas classes não são considerados complexos e o potencial de reuso dos métodos herdados é baixo);
- **NOC** (*Number of Children*): quantidade de subclasses imediatamente subordinadas à classe na hierarquia, isto é, quantas subclasses estão herdando os métodos da super classe. Quanto menor o seu valor, menor é o potencial de reuso das subclasses;
- **CBO (***Coupling between object classes*): quantidade de outras classes a que uma determinada classe está acoplada.

O Quadro 1 ilustra um exemplo onde são medidos 3 sistemas fictícios onde foram aplicados valores aleatórios que servirão como hipóteses para descrever melhor o raciocino empregado.

Métricas	Sistema 1	Sistema 2	Sistema 3
WMC	214	229	256
CBO	3	6	6
DIT	10	23	1
NOC	40	80	4

Quadro 1 – Valores das métricas aplicadas a três sistemas OO
Fonte: Dados da Pesquisa

Observando apenas as métricas DIT e NOC é possível identificar o potencial de reuso dos métodos para os três sistemas, o sistema que tem maior potencial de reuso é sistema dois, pois tem o maior valor para NOC o que também é confirmado através do valor da métrica DIT. Um outro ponto a ser considerado no Quadro 1 é o sistema três que tem um baixo potencial de reuso, o que é indicado pelo valor da métrica NOC e confirmado pelo valor da métrica DIT. No caso do Quadro 1, usado como hipótese é possível identificar que a métrica de software para projetos orientados a objetos não apenas pode ser utilizados para medir o tamanho, mas também para determinar se alguns requisitos de qualidade do sistema serão atingidos. Por exemplo, para o sistema três poderia ser afirmado que não esta atendendo ao quesito de reusabilidade o que pode gerar outros impactos, por exemplo, reduzir a manutenibilidade.

3 DESIGN PATTERNS

Os indivíduos envolvidos na produção de software perceberam que muitos dos problemas de projetos de sistemas eram recorrentes. Este foi um dos fatores que motivou o surgimento dos Designs Patterns ou Padrões de Projetos, os quais são aplicados a problemas clássicos na produção de softwares.

"Os padrões descrevem problemas do nosso ambiente e o cerne de sua solução, de modo que se possa aplicar essa solução muitas vezes, sem nunca fazê-lo da mesma maneira." (GAMMA et al, 2000, p.19). Um fator é que estes padrões serão expressos em termos de objetos e interfaces, de modo a serem aplicados a problemas de um determinado contexto.

Os Designs Patterns desempenham um relevante papel na construção de projetos, pois foram desenvolvidos para serem reutilizados. Assim, subsidiando aos arquitetos de software uma ferramenta que irá evitar que os problemas sejam resolvidos a partir de princípios elementares, ou seja, fazendo tudo desde o inicio.

Segundo Gamma et al (2000, p. 18): "Uma coisa que os bons projetistas sabem que não devem fazer é resolver cada problema a partir de princípios elementares ou do zero. Em vez disso, eles utilizam soluções que já funcionaram no passado".

Geralmente os fatores ligados à reutilização envolvem projetos orientados a objetos e as conseqüências em utilizar um padrão ou um conjunto de padrões incluem o seu impacto sobre a flexibilidade, a extensibilidade, ou a portabilidade de um sistema. Fazer uma relação entre estes fatores irá ajudar a entender e verificar a importância da aplicação de Designs Patterns.

Existem atualmente 23 (vinte e três) padrões de projetos bem popularizados, porém, alguns autores os classificam de forma diferente, como pode ser visto, nos quadros 2 e 3.

Objetivo	Padrões
Interfaces	Adapter, Façade, Composite, Bridge
Responsabilidade	Singleton, Observer, Mediator, Proxy, Chain of Responsability, Flyweight
Construção	Builder, Factory Method, Abstract Factory, Prototype, Memento
Operação	Template Method, State, Strategy, Command, Interpreter
Extensões	Decorator, Iterator, Visitor

Quadro 2 - Classificação de padrões considerando seus objetivos
Fonte: Metsker (2004, p.22).

Propósito	Padrões
Criação	Adapter, Façade, Composite, Bridge
Estruturais	Singleton, Observer, Mediator, Proxy, Chain of Responsability, Flyweight
Comportamentais	Builder, Factory Method, Abstract Factory, Prototype, Memento
Operação	Template Method, State, Strategy, Command, Interpreter
Extensões	Decorator, Iterator, Visitor

Quadro 3 - Classificação de padrões considerando seus propósitos
Fonte: Gamma, et al (2000, p.45).

A utilização de Designs Patterns no desenvolvimento de sistemas orientados a objetos é um fator substancial, pois aponta como devem ser utilizadas técnicas fundamentais da orientação a objeto, o que favorece a reutilização.

"Acreditamos que os nossos padrões de projetos sejam uma peça importante que tem faltado nos métodos de projetos orientados a objetos. Os padrões mostram como utilizar técnicas básicas, tais como objetos, herança, polimorfismo" (GAMMA et al, 2000, p. 325).

Efetivamente, os Designs Patterns podem contribuir para suavizar a transição do modelo de análise para o modelo de implementação, haja vista, se balizam em tipos recorrentes de problemas de implementação em projetos de software. Assim, podendo ser vistos como um modelo de "micro arquitetura" aplicado a um dado problema, o que estabelece uma linguagem comum entre a construção de aplicações que permeiam um mesmo contexto.

Para Antoniol; Fiutem; Cristoforetti (1998, p. 2):

> [...] a descrição da solução tenta capturar a introspecção essencial que o padrão personifica, de modo que outros possam aprender sobre ele, e o empregar em situações similares: os Designs Patterns ajudam a criar uma linguagem compartilhada onde é possível permear experiências, comunicando-se sobre determinados problemas e suas soluções.

A Figura 1 ilustra o padrão Adapter, que segundo Gamma et al (2000, p.140), "tem o objetivo de converter a interface de uma classe em outra interface esperada pelos clientes. Adapter permite a comunicação entre classes que não poderiam trabalhar juntas devido à incompatibilidade de suas interfaces.".

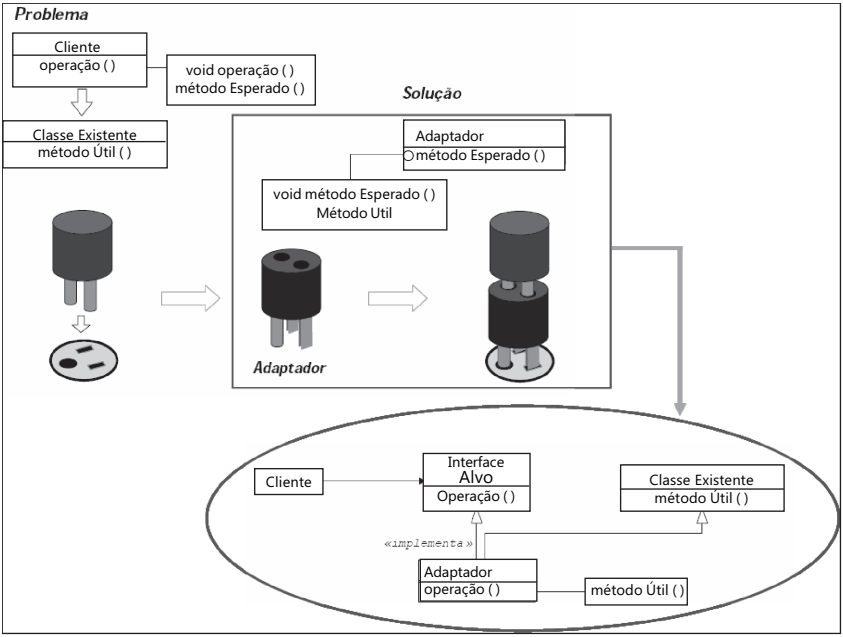

Figura 1 - Padrão Adapter
Fonte: Baseado em Gamma et al (2000, p. 142).

Através da própria fisiologia do padrão Adapter é possível identificar o problema e também verificar a sua aplicabilidade, no qual é descrita na solução proposta por Gamma et al (2000) na Figura 1. Deste modo, confirmado o que Antoniol; Fiutem; Cristoforetti (1998, p. 2) afirmam em seu artigo.

4 MÉTRICAS APLICADAS A DESIGN PATTERNS: UM ESTUDO DE CASO

O estudo de caso realizado nesta seção se fundamenta na pesquisa de Antoniol; Fiutem; Cristoforetti (1998), onde foram aplicadas métricas de softwares para a detecção de padrões de projetos. Neste contexto, os autores aplicaram um processo totalmente automatizado para identificação de padrões (ANTONIOL; FIUTEM; CRISTOFORETTI, 1998, p. 6), como pode ser observado pela Figura 2.

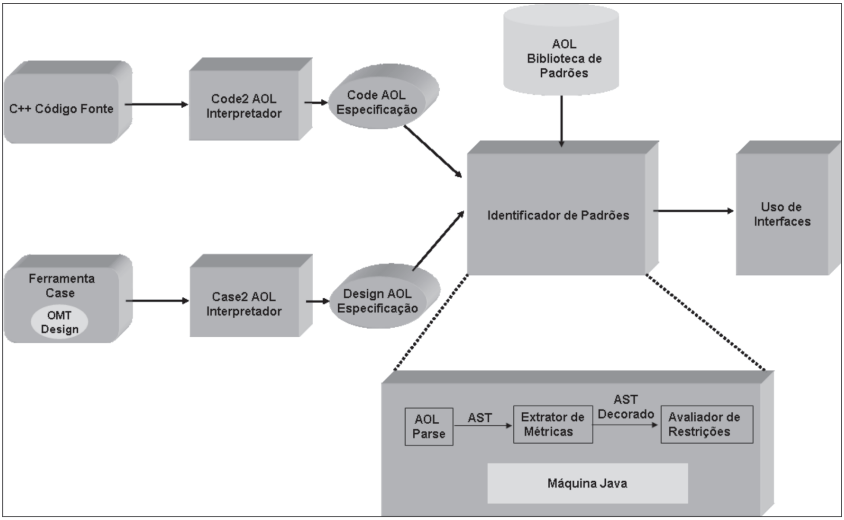

Figura 2 - Arquitetura que representa o processo do extrator de padrões
Fonte: Antoniol; Fiutem; Cristoforetti (1998)

O processo ilustrado na Figura 2 pode ser iniciado através de um código fonte em C++, e/ou por um modelo baseado em UML (*Unified Modeling Language*); logo após sendo interpretado e passado para uma definição em uma especificação AOL (*Abstract Object Language*) com intuito de manter a independência da linguagem de programação e/ou das ferramentas de modelagem adotadas. Portanto, o extrator de padrões pode ser aplicado tanto a um código quanto a um design.

A fim de avaliar o quanto os designs patterns podem influenciar nas métricas de um sistema e consequentemente nas estimativas, o presente artigo realizou um estudo nos resultados obtidos através dos sistemas abordados na pesquisa de Antoniol; Fiutem; Cristoforetti (1998). Para tanto, foram verificados os resultados descritos em dois sistemas de domínio público, levando-se em consideração os conceitos das métricas de softwares orientados a objetos, proposta por Chidamber; Kemerer (1994).

No Quadro 4 são colocados dados referentes a dois sistemas de domínio público groff e LEDA. Segundo Antoniol; Fiutem; Cristoforetti (1998, p.9) aplicando a sua metodologia:

- No groff não foi identificado à presença de Design Patterns;
- No LEDA foi identificado a presença do padrão Adapter.

Características	groff	LEDA
LOC	127762	115882
Classes	25	1857
Atributos	89	426
Operação	173	4332
Agregação	4	166
Associação	26	334
Herança	4	85

Quadro 4 - Análise dos Sistemas groff e LEDA segundo suas características
Fonte: Antoniol; Fiutem; Cristoforetti (1998, p.9)

Remetendo a proposta de Chidamber; Kemerer (1994) é possível com base nas métricas descritas por estes autores realizar algumas inferências relacionando o número de classes e heranças, tais como:

- No sistema groff com 127762 (cento e vinte sete mil setecentos e sessenta e dois) linhas de código existem apenas 25 (vinte e cinco) classes, isso é um indicativo de que existe pouca modularidade. Um outro fator é o baixo índice de heranças, apenas 4 (quatro), o que remete ao NOC das métricas aplicadas a projetos de softwares OO, indicando que este sistema tem um baixo potencial de reutilização;
- No sistema LEDA, onde foi detectado a presença de Design Pattern é possível observar que este tem um maior índice de modularidade relacionando o número de linhas de código por classes (se comparado ao groff). O seu potencial de reutilização também é maior com base no NOC.

Para o caso abordado no Quadro 4, o uso de Design Pattern cria um impacto positivo sobre as métricas do sistema. Isso porque, ao analisar os softwares através dos dados coletados pela pesquisa Antoniol; Fiutem; Cristoforetti (1998), e ao aferi-lo com base na métrica NOC estabelecida por Chidamber; Kemerer (1994) foi identificado que o sistema LEDA que apresentou a utilização do padrão Adapter tem um maior potencial de reuso. Portanto, indicando para efeito de estimativa uma redução no esforço.

Os resultados coletados não utilizaram todas as métricas propostas por Chidamber; Kemerer (1994), isso porque, estas não puderam ser aferidas mediante as características expostas no Quadro 4, que remete ao artigo de Antoniol; Fiutem; Cristoforetti (1998). Com tudo, a métrica NOC foi identificada, assim, viabilizando analisar as potencialidades de reuso de alguns sistemas. Sendo, portanto, uma referência para verificar que o uso de padrões de projetos causa de fato algum impacto sobre as métricas, assim refletindo sobre as estimativas do sistema.

5 CONSIDERAÇÕES FINAIS

Aplicar métricas de projetos orientados a objetos em sistemas que fazem utilização de Design Pattern pode acrescentar uma ponderação substancial às estimativas dos projetos. Isso porque, com base no estudo realizado neste artigo, foi possível identificar que o projeto onde se denotou a presença do uso de Design Pattern tem os seus artefatos de código mais propensos ao reuso, desta maneira refletindo sob as estimativas.

Um fator importante também observado é que as métricas a serem aplicadas a projetos de software que fazem uso de Design Pattern devem ser fundamentalmente dirigidas a objetos, isso porque estes tipos de métricas devem contemplar as características de software orientado a objetos. Portanto, utilizar métricas de sistemas estruturados não é conveniente uma vez que estas não prevêem pontos intrínsecos da OO, tais como: profundidade da árvore de herança, número de filhos de uma classe, etc.

O presente artigo também despertou como trabalho futuro a utilização de métricas orientadas a objetos como ponto de validação de arquiteturas baseadas e Design Pattern. A meta deste trabalho será validar modelos que utilizam padrões de projetos em sua arquitetura. Deste modo, esperando-se avaliar se determinado padrão de projeto está sendo bem empregado na arquitetura do sistema, para tanto levando-se em consideração os níveis de reuso e desacoplamento provido pelo padrão. Acredita-se que este trabalho é totalmente factível, haja vista que o conjunto de métricas estudadas no presente artigo provê subsidio para tal.

REFERÊNCIAS

ANTONIOL, G.; FIUTEM, R.; CRISTOFORETTI, L. Using Metrics to Identify Design Patterns in Object-Oriented Software. In: SOFTWARE METRICS SYMPOSIUM, 5.,1998. **Proceedings...** [S.l.]: IEEE, 1998.

CHIDAMBER, S. R.; KEMERER, C. F. A Metrics Suite for Object Oriented Design. **IEEE Transactions on Software Engineering**, v. 20, n. 6, June 1994.

FENTON, N.; NEIL, M. Software metrics: roadmap. In: FINKELSTEIN, A. (ed.). **The Future of Software Engineering**. [S.l.]: ACM Press, 2002.

FINKELSTEIN, A.; KRAMER J. Software Engineering: A Roadmap. In: CONFERENCE ON SOFTWARE ENGINEERING, 2000. **Proceedings**... [S. l.]: ACM Press, 2000.

GAMMA, E. et al. **Padrões de Projeto**. Porto Alegre: Bookman, 2000.

METSKER, S. J. **Padrões de Projeto em Java**. Porto Alegre: Bookman, 2004.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

SOMMERVILLE, Ian. **Engenharia de Software**. Rio de Janeiro: Prentice-Hall, 2004.