

# Projeto Orientado a Objetos e U.M.L. <sup>1</sup>

Sérgio Luiz Tonsig <sup>2</sup>

## Resumo

O modelo Orientado a Objetos, já existe a quatro décadas e inicialmente estava aplicado às linguagens de programação. No início da década de 90, observou-se o surgimento de uma grande quantidade de métodos orientados a objetos, para o desenvolvimento de sistemas, muito embora, com grandes diferenças entre eles. Pela necessidade de mercado, a OMG (*Object Management Group*) apoiou a padronização de um meta-modelo que veio a ser chamado de UML (*Unified Modeling Language*). Com a padronização, a indústria do software passa a ter segurança em criar produtos para comercialização em larga escala, além de criar-se um ambiente favorável para a utilização do método, por parte do gerenciamento dos sistemas de informação das empresas.

## Palavras-Chave

Orientado a Objetos; Métodos Orientados a Objetos; OMG; UML; Padronização; Indústria do Software; Gerenciamento dos Sistemas de Informação.

## *Object-Oriented Project and U.M.L.*

### *Summary*

*The object-oriented model, it already exists to four decades and initially it was applied to the programming languages. In the beginning of the decade of 90, the appearance of a great amount of objects-oriented methods was observed, for the development of systems, very auspiciously, with great differences among them. For the market need, OMG (Object Management Group) it supported the standardization of a goal-model that came to be called UML (Unified Modeling Language). With the standardization, the software industry starts to have safety in creating products for commercialization in wide scale, besides creating a favorable atmosphere for the use of the method, on the part of the information systems management in the companies.*

### *Keywords*

*Object-oriented; Objects-oriented methods; OMG; UML; Standardization; Software Industry; Information Systems Management.*

---

<sup>1</sup> Artigo desenvolvido no Mestrado em Gerência de Sistemas de Informação da PUCCamp

<sup>2</sup> Analista de Sistemas, Especialista em Sistemas de Informação pela UFSCar

# 1. Paradigma da Orientação a Objetos

O paradigma da orientação a objetos, surgiu com a programação no final dos anos sessenta, com a linguagem SIMULA. Nos anos setenta, era parte importante da linguagem SMALLTALK, desenvolvida pela Xerox. Este paradigma só estreou na análise de sistemas no final da década de oitenta [RUMBAUGH, 1994].

A característica básica da aplicação do modelo orientado a objetos para o desenvolvimento de sistemas, é a façanha de terem unificado os formalismos utilizados na análise, projeto e programação.

## 1.1 Universo do Modelo Orientado a Objetos

O modelo apresenta alguns conceitos básicos que são apresentados abaixo [YOURDON & ARGILA, 1999]:

- a) Objeto é a representação abstrata de coisas do mundo real, que sob o ponto de vista do nosso problema, possuem atributos e métodos comuns.

Atributos representam as características do objeto, por exemplo, o objeto caneta, possui como atributos tamanho, cor, fabricante e modelo. Métodos são operações ou funções oferecidas pelo objeto, ou seja, aquilo que ele pode fazer. O objeto caneta pode ter um método chamado escrever.

- b) Os objetos também apresentam algumas propriedades.

Estado: Diz respeito a situação em que pode estar um determinado objeto. Por exemplo, o objeto caneta, pode estar no estado “com tinta” ou “sem tinta”. Podemos ainda atribuir a caneta outros estados “Integra para uso” ou “quebrada”. O estado, depende da natureza do objeto.

Comportamento: qualquer objeto apresenta um comportamento. O comportamento é o meio através do qual o objeto passa de um estado para o outro. Normalmente, isto se dá mediante uma condição/ação (a ação será sempre um método a ser executado).

- c) Todo objeto é identificável.

Embora se possa ter várias canetas de um mesmo tamanho, cor, fabricante e modelo, cada uma delas é uma caneta em particular, tem sua própria identidade, são portanto distinguíveis entre si.

- d) Uma Classe de Objetos representa um conjunto de objetos de mesma característica.

Suponha existir uma Classe de Objetos Avião. Ela será uma generalização de objetos avião. Todos os objetos desta classe terão identidade e serão distinguíveis. Dois aviões de mesma cor, tamanho e formato continuam sendo aviões distintos. Cada um deles é uma instância de objeto, dado a sua existência são diferenciados por sua identidade. Uma instância herda as características da classe a que pertence (por características entende-se os atributos e métodos).

e) Encapsulamento é a ocultação ou empacotamento de dados e procedimentos dentro do objeto.

Um objeto só permite o acesso a seus dados, mediante o acionamento de seus métodos, através de uma mensagem, para a qual, pode devolver uma resposta.

f) Aclopamento dinâmico, herança e polimorfismo.

Ao receber uma mensagem, o objeto verificará se há na classe a qual ele pertence, um serviço (método) que defina seu comportamento perante esta mensagem. Caso ele não encontre, verificará nas super-classes da classe (qualquer uma de onde tenha herdado algo). Este mecanismo de busca de serviços é chamado de aclopamento dinâmico.

Mensagens iguais, destinadas a objetos diferentes, podem gerar comportamentos diferentes [MARTIN & ODELL, 1996]. Para uma mesma mensagem, objetos diferentes podem responder ou agir de forma diferenciada; a isto, chamamos polimorfismo. Por exemplo, uma mensagem “print” para uma impressora, pode fazer com que a mesma imediatamente comece a imprimir. A mesma mensagem para uma outra impressora, pode ocasionar a apresentação de uma tela com opções de configuração da impressora. Portanto, temos a mesma mensagem enviada para objetos distintos os quais responderam de forma diferenciada.

## 2. UML

O Início dos anos 90 foi caracterizado pelo nascimento de uma grande quantidade de métodos e notações para suportar o desenvolvimento orientado a objetos, que tinha demonstrado ser um meio eficaz para produção de software.

Esta grande quantidade de métodos foi uma clara demonstração da aceitação do modelo orientado a objetos, mas gerou como inconveniente um grande número de metodologias e notações diferentes, de maneira que muitos usuários decidiram aguardar para saber qual delas, finalmente, se firmaria no mercado.

Por iniciativa da OMG (*Object Management Group*), foi aberto a proposta para apresentação de trabalhos de padronização de um modelo para desenvolvimento de sistemas que atendesse ao modelo orientado a objetos. A proposta vencedora, foi apresentada pela *Rational Software Corporation* e recebeu o nome de UML (*Unified Modeling Language*).

Esta proposta de padronização foi um esforço liderado por *Grady Booch*, *James Rumbaugh* e *Ivar Jacobson* que resultou na versão 1.0 da UML publicada em 13 de janeiro de 1997, e adotada como padrão pela OMG no mesmo ano. Aglutinou o que havia de melhor em vários métodos então existentes, tendo recebido a colaboração de vários metodologistas [FURLAN, 1998].

### 2.1 Componentes da UML

A Linguagem de Modelagem Unificada (UML), não é um método de desenvolvimento de sistemas. É uma linguagem de modelagem gráfica para descrever

um projeto de software. Ela simplifica o complexo processo de análise, projeto e construção de software criando visões do sistema que esta sendo construído [RATIONAL, 2000].

Um método pressupõe um modelo de linguagem e um processo. O modelo de linguagem é a notação que o método usa para descrever o projeto. O processo são os passos que devem ser seguidos para se construir o projeto.

O modelo de linguagem é uma parte muito importante do método. Corresponde ao ponto principal da comunicação. Se uma pessoa quer conversar com outra sobre o projeto, é através do modelo de linguagem que elas se entendem. Nessa hora, o processo não é utilizado.

A UML, como notação, possui um conjunto de artefatos gráficos, conforme segue:

a) Diagrama de Caso de Uso

É um diagrama usado para se identificar como o sistema se comporta em várias situações que podem ocorrer durante sua operação. Descreve o sistema, seu ambiente e a relação entre os dois. Os componentes deste diagrama são os “atores” e os “Use Case”, conforme mostra a figura 2. Uma visão da interação entre estes elementos pode ser visto na figura 3.



**Fig. 2 – Ator e Caso de Uso**

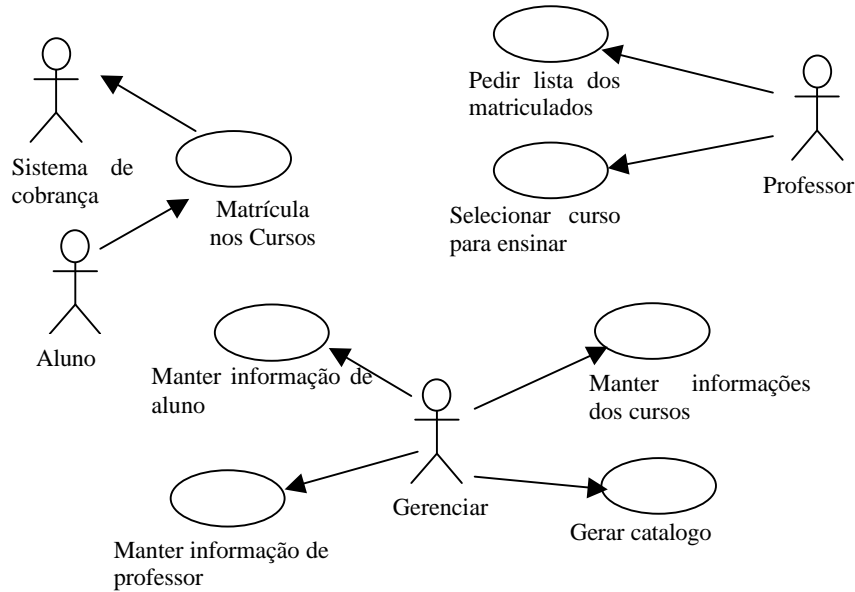
Um ator representa qualquer entidade que interage com o sistema, tendo as seguintes características:

- ator não é parte do sistema. Representa os papéis que o usuário do sistema pode desempenhar.
- ator pode interagir ativamente com o sistema.
- ator pode receber informações do sistema.
- ator pode representar um ser humano, uma máquina ou outro sistema.

Um caso de uso, é uma seqüência de ações que o sistema executa, revela um padrão de comportamento, acionado em geral por um ator e produz um resultado que contribui para os objetivos do sistema. Algumas de suas características são descritas abaixo:

- Um “Use Case” modela o diálogo entre atores e o sistema, ou mesmo entre casos de uso.
- Um “Use Case” é iniciado por um ator para invocar uma certa funcionalidade do sistema, ou pode ser acionado por um outro caso de uso.
- Um “Use Case” é fluxo de eventos completo e consistente.

- O conjunto de todos os “Use Case” representa todos as situações possíveis de utilização do sistema.



**Fig. 3 – Exemplo de um Use-Case**

Um diagrama de “Use Case”, deve ser rigorosamente avaliado e para tanto, aplica-se o conceito de cenário.

O cenário é uma instância de um “Use Case”. O “Use Case” deve ser descrito através de vários cenários. Devem ser construídos tantos cenários quantos forem necessários para se entender completamente todo o sistema. Podem ser considerados como teste informais para validação dos requisitos do sistema.

#### b) Diagrama de Classe

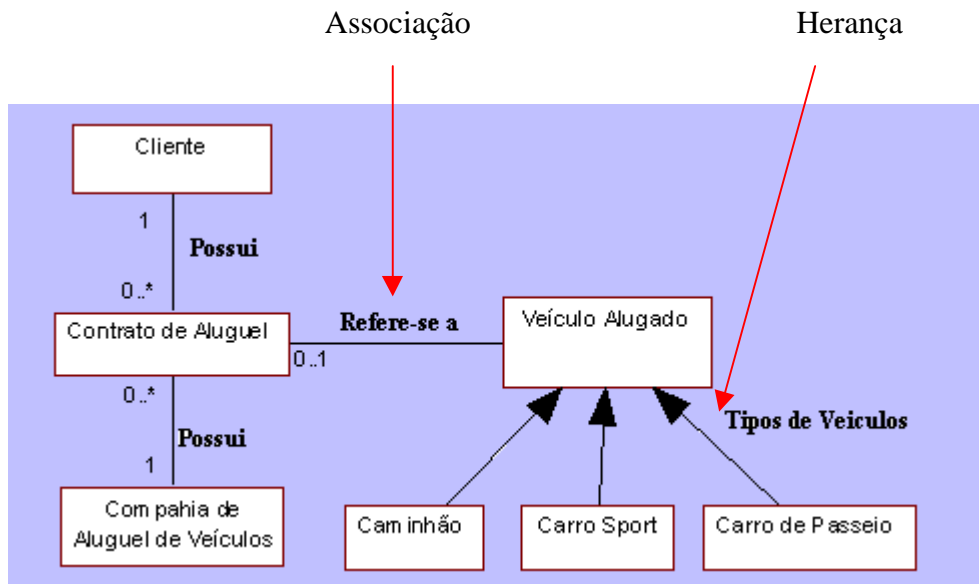
A notação usada pela UML para representar uma Classe de Objetos é:

Nome
Atributo
Métodos

A classe de objeto é representada por um retângulo, subdividido em três áreas. A primeira contém o nome da Classe, segunda contém seus atributos e a terceira contém seus métodos (funções/procedimentos).

#### c) Relações entre Classes

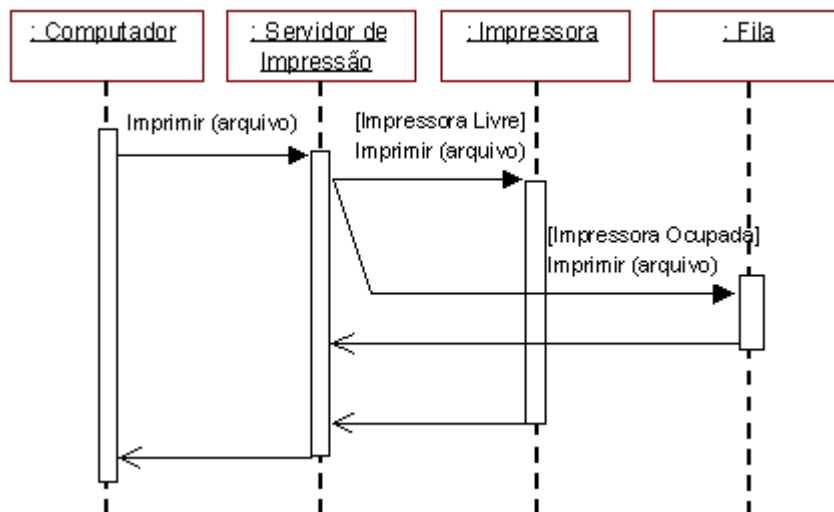
As classes podem apresentar três tipos de relações: associação, agregação e herança. Pela associação, sabe-se que há um relacionamento de dados entre as classes. Na agregação tem-se a visão de uma classe como sendo constituída pela agregação de outras e a relação de herança indica que a classe filha herda as características da classe pai (herda atributos e métodos).



**Figura 4 – Diagrama de Classe**

d) Diagrama de Sequência

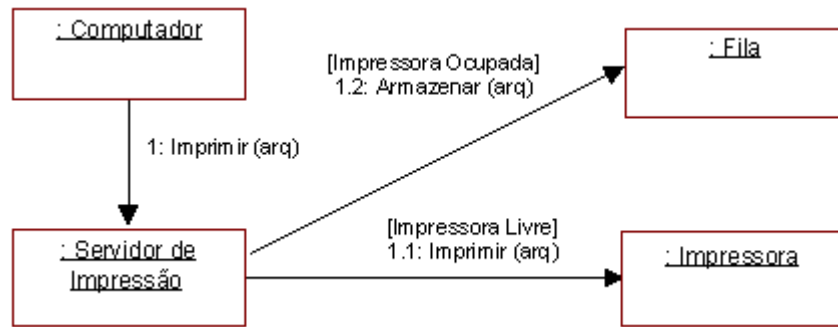
Mostra a interação entre objetos com a preocupação de documentar os métodos (funções/procedimentos) ao longo do tempo, conforme mostra a figura 5.



**Fig. 5 – Diagrama de Sequência**

e) Diagrama de Colaboração

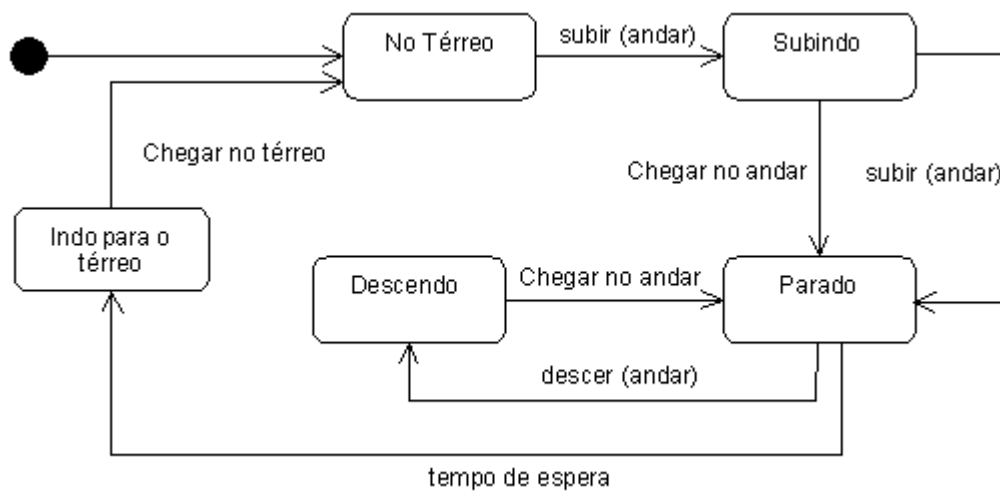
É um modo alternativo para representar a troca de mensagens entre um conjunto de Objetos, mostra a interação organizada entorno dos Objetos e suas ligações uns com os outros (Fig. 6).



**Fig. 6 – Diagrama de Colaboração**

f) Diagrama de Transição de Estados

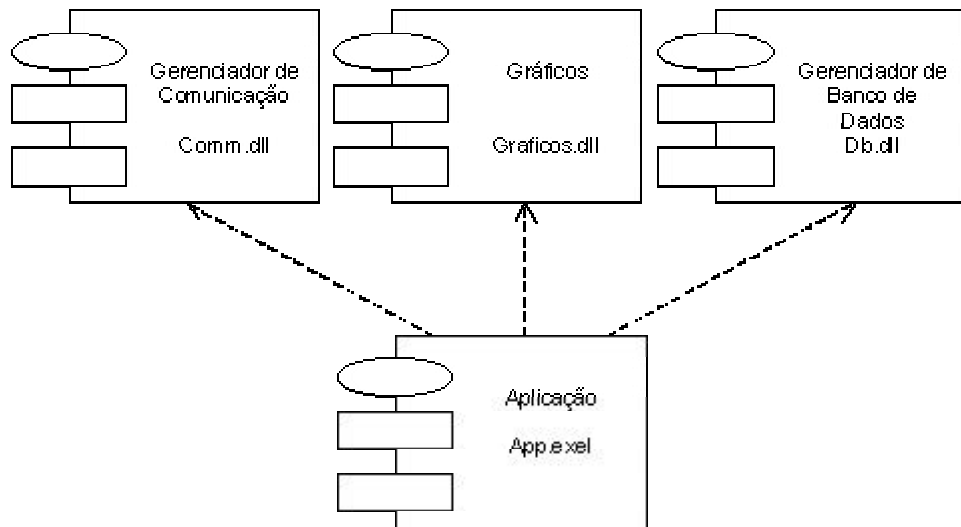
Com ele (fig. 7) pode-se fazer um mapeamento dos possíveis estados que pode se encontrar um objeto do sistema. Além dos estados, documenta-se a condição/ação para que o objeto saia de um determinado estado e vá para outro.



**Fig. 7 – Diagrama de Transição de Estados**

g) Diagrama de Componentes

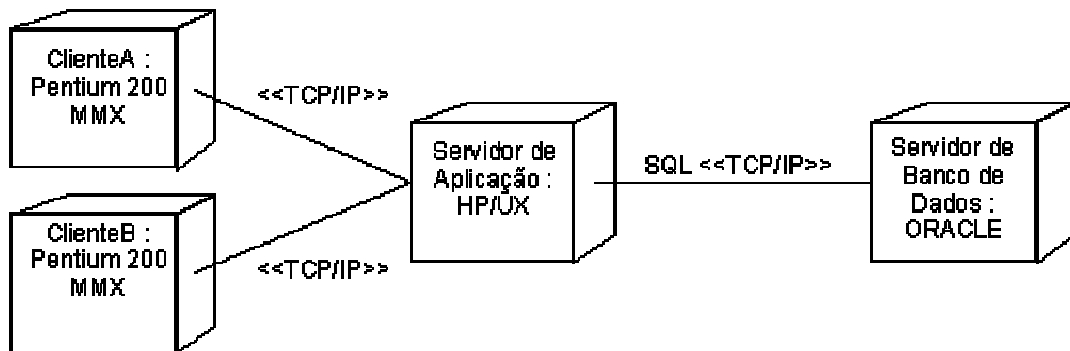
Documenta como deve ficar no mundo físico a organização dos componentes que constituem o sistema (fig. 8). Componente pode ser um código fonte, um executável ou uma biblioteca.



**Fig. 8 – Diagrama de Componentes**

h) Diagrama de Distribuição

Este diagrama (fig. 9) mostra a configuração dos elementos de processamento, visualizando a distribuição por toda a empresa.



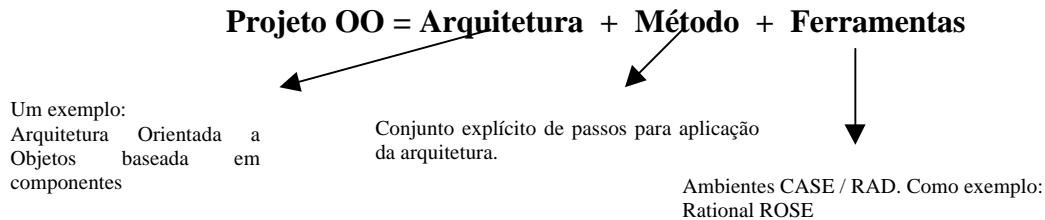
**Fig. 9 – Diagrama de Distribuição**

### 3. Projeto OO - Características Gerenciais e Comerciais

Por projeto <sup>3</sup> entende-se a implementação de todo artefato criado em tempo de análise, de acordo com o método de desenvolvimento escolhido.

Todo projeto de software bem sucedido ampara-se numa arquitetura forte bem exercitada [MATTIAZZI, 1998]. Para se aplicar eficazmente os conceitos dessa arquitetura, é necessário seguir um método bem definido. Tanto a arquitetura quanto o método precisam ser amparados por um conjunto de ferramentas que automatize parte do trabalho e que traga confiabilidade aos resultados obtidos.

<sup>3</sup> O termo 'projeto' não tem o mesmo sentido do termo *Project* do inglês. Nosso termo projeto refere-se ao *Design*. O Projeto propriamente dito (*Project*) leva em consideração fases anteriores ao nosso *design*, abordando tarefas de planejamento, supervisão e gerência e não somente o desenho do domínio do problema e sua implementação.



**Figura 10 - Projeto Orientado a Objetos**

Ao tratar-se de um projeto decorrente do modelo orientado a objetos (fig. 10), espera-se obter alguns benefícios, tais como: melhor gerenciamento da complexidade das funções pertinentes ao sistema, melhor mapeamento do mundo real versus mundo computacional e melhor qualidade de serviços decorrentes do sistema.

Além dos benefícios citados, outros fatores, ligados a produtividade, também são esperados. Deve haver uma maior reutilização do software, acarretando ganhos de performance na construção de novos sistemas. Todos estes aspectos conduzem à vantagens econômicas. A partir da década de 90, as aplicações se tornaram mais complexas, demorando mais tempo para serem executadas e apresentando um índice proporcionalmente maior de falhas. O tempo e o dinheiro necessários para a manutenção dessas aplicações também aumentaram substancialmente. A orientação a Objetos leva estes problemas específicos em consideração [AMBLER, 1997].

Então, por que 100% das empresas ainda não adotaram a tecnologia de objetos ? Os principais motivos apontados são [FURLAN, 1998]:

- Incerteza
- Falta de pessoas treinadas (*know-how*)
- Ferramentas Imaturas
- Investimentos significativos já realizados em ferramentas não Orientas a Objeto

Embora a tecnologia orientada a objetos não seja nova, o Projeto Orientado Objetos é uma abordagem inovadora para o desenvolvimento do software, que, se bem implementada, trará benefícios substanciais para o aumento de produtividade com qualidade, é necessário no entanto, que se consiga transpor as razões habituais de resistência para adoção de novas tecnologia.

## Referências

[AMBLER, 1997] AMBLER, SCOTT W. “*Análise e Projeto Orientados a Objeto*”. IBPI Press, RJ, 1997.

[FURLAN, 1998] FURLAN, JOSÉ DAVI. “*Modelagem de Objetos através da UML*”. Makron, SP, 1998.

[MARTIN & ODELL, 1996] MARTIN, JAMES E ODELL, JAMES J. “*Análise e Projeto Orientados a Objeto*”. Makron, SP, 1996.

- [**MATTIAZZI, 1998**] MATTIAZZI, LEONARDO DUARTE. “*Orientação a Objetos e a UML: Finalmente um rumo a seguir*”. Developers, Ano 3, número 26, Out/1998, p.26-29.
- [**RATIONAL, 2000**] RATIONAL SOFTWARE CORPORATION, Home Page “*UML Resource Center*”, <http://www.rational.com/>, consulta em 09/03/2000.
- [**RUMBAUGH, 1994**] RUMBAUGH, JAMES et alii. “*Modelagem e projetos baseados em objetos*”. Campus, Rio de Janeiro, 1994.
- [**YOURDON & ARGILA, 1999**] YOURDON, EDWARD E ARGILA, CARL. “*Análise e Projeto Orientados a Objetos*”. Makron, SP, 1999.