EXPERIÊNCIAS NA UTILIZAÇÃO DA ORIENTAÇÃO A OBJETO NO DESENVOLVIMENTO DE SISTEMAS - PESQUISA BIBLIOGRÁFICA

Gilmar Jonas Deghi[*]

RESUMO

A utilização do paradigma da Orientação a Objeto promete revolucionar o desenvolvimento de *software* tornando-o uma atividade mais previsível, confiável e produtiva. Isto atende principalmente à alta demanda de *softwares* de complexidade crescente, exigido em tempo cada vez menor, e com uma alta taxa de mudanças no seu ciclo de vida devido principalmente ao ambiente competitivo em que estão mergulhadas as empresas. O artigo mostra resumidamente a necessidade de medidas para a melhor produção de *software*, a origem e a evolução histórica da Orientação a Objeto e na última parte, o seu uso e os pontos considerados importantes para a implementação em empresas dentro da bibliografia pesquisada. São apontados os principais fatores que levaram estas empresas a adotarem tal abordagem, seus pontos considerados positivos e negativos e os cuidados necessários para que se obtenha sucesso na utilização da Orientação a Objeto.

(*) Engenheiro elétrico pela FEI - Faculdade de Engenharia Industrial com especialização em Administração de Empresas pela FGV - Fundação Getúlio Vargas e mestrando pelo Programa de Pós-Graduação em Administração da FEA - Faculdade de Economia, Administração e Contabilidade da Universidade de São Paulo.

INTRODUÇÃO

O desenvolvimento de sistemas dentro da administração da informática nas empresas sempre foi, tradicionalmente, uma das mais difíceis questões a serem resolvidas. Seja pela dificuldade da métrica que auxilia na resolução do prazo de desenvolvimento, seja na metodologia utilizada e também nas linguagens de desenvolvimento utilizadas, o descontentamento com os sistemas de processamento de dados parece geral dentro das empresas. De acordo com Pressman (1995), "muitas pessoas e empresas ainda desenvolvem *software* de modo casual". A conseqüência disto se traduz em falhas de especificação junto ao usuário, prazos constantemente dilatados e qualidade baixa dos resultados. Coleman (1996), estrutura os principais problemas como :

- baixa capacidade de previsão: Os responsáveis tem sérios problemas em prever tempo e esforços necessários para produzir um sistema de acordo com os requisitos dos usuários;
- programas com baixa qualidade: Os requisitos originais levantados junto ao usuário estão incompletos ou contraditórios resultando muitas vezes em travamento, perda de dados e a não execução de tarefas. Tais problemas são geralmente descobertos nas etapas finais do processo de codificação;
- altos custos de manutenção: Quando o sistema não tiver sido construído de modo adequado, tanto a manutenção corretiva, que corrige erros existentes, com a evolutiva, que acrescenta recursos, são muito dispendiosas;
- duplicação de esforços: Os projetos normalmente são iniciados compartilhando pouco ou nenhum código de projetos anteriores. A replicação de códigos visando o seu aproveitamento em outros projetos é difícil dada a filosofia de trabalho utilizada e pelas linguagens tradicionais que dificultam a expressão das similaridades envolvidas.

No que diz respeito às metodologias e ferramentas disponíveis, Martin & Odell (1996) dizem que "nossa capacidade de criar *software* não está acompanhando a evolução do *hardware*". Portanto, a busca de ferramentas que permitam a criação mais ágil, confiável e de qualidade, passa por novas metodologias e até de novos paradigmas.

Além disso, em artigo recente da revista Business Week (04/08/97, p. 46), reproduzida pela gazeta Mercantil (08/08/97), "a revolução da informação está avançando à frente de sua matéria-prima vital : capacidade cerebral. À medida que explode a demanda de aplicações computadorizadas para tudo, desde comércio eletrônico na Internet até a resolução do problema do ano 2.000, as empresas estão se vendo carentes de programadores." O artigo ainda contabiliza a falta de pessoal qualificado em 190.000 empregos disponíveis somente nos EUA, e cujas vagas não são preenchidos. Em termos mundiais esta carência é da ordem de 400.000 posições em aberto, o que leva as indústrias de *software* a uma busca frenética de cérebros. Esta busca se dá em outros países, via Internet e qualquer meio possível. Assim, qualquer aumento de produtividade neste cenário, como a prometida pela orientação a objeto, seria muito bem vinda.

Apesar de relativamente recente, a Orientação a Objeto (OO), segundo Pressman (1995), ainda não dispõe de concordância universal sobre os "conceitos" que servem de base, mas um número limitado de idéias chave aparece repetidamente e já estão sendo utilizados em muitas empresas.

Apesar de todas as promessas, nem todos os autores vêem com entusiasmo o potencial da orientação a objeto. Existem sérias críticas em relação ao uso da tecnologia de orientação a objeto. Guthery (1989), questiona o desempenho dos programas orientados a objetos e a falta de dados concretos sustentando a reusabilidade (reutilização de trechos de códigos já prontos) e a eficácia da abordagem. Talvez, pelo artigo datar de perto de 10 anos atrás, o avanço da OO possa ter minimizado tais críticas, embora ainda hoje existe um receio do não retorno prometido pela abordagem orientada a objeto.

HISTÓRICO

Na década de 60 surge junto com o avanço dos primeiros computadores comerciais o desenvolvimento de sistemas, criando-se necessidades de processos automatizados de trabalho até então inexistentes. Tapscott (1995), acredita que a analogia do processo produtivo industrial com a produção de *software* é imperfeita. Neste último caso, a criatividade do profissional aliada ao conhecimento de novas ferramentas é essencial para que se adicione valor aos negócios.

No primeiro instante, sem metodologias a seguir, os analistas e o programadores, segundo Martin & McClure (1991), utilizavam meios mais ou menos intuitivos e empíricos para a análise e geração do código, fazendo com que a gestão também se tornasse igualmente empírica. "Os métodos de análise e programação precisavam deixar de ser amadorísticos e *ad hoc* para tornarem-se disciplinados e bem-pensados" (Martin & McClure - 1991). Uma primeira etapa na melhoria desta situação deu-se com a introdução por Chris Gane & Trish Sarson(1969), Edward Yourdon (1990), Tom DeMarco (1989) e outros, cada um com pequenas variantes, dos conceitos de *análise e projeto estruturado*. Nesta abordagem, baseada em processos e controles, pode-se obter produtos mais precisos e confiáveis. Apesar de ainda hoje bastante utilizada, a análise estruturada mostra indícios de que, face às novas ferramentas de informática, necessita ser revisada em seus conceitos gerais para que, segundo Martin (1996), possa acompanhar a velocidade cada vez maior necessária para a criação de *software*.

A vantagem de se ter um modelo conceitual dos processos referentes ao sistema (originado da abordagem estruturada), é que se pode trabalhar alterações antes de se chegar ao projeto físico e à implementação, como que, para construir um prédio, trabalhamos antes em sua planta. Existe entretanto uma série de desvantagens como o tempo consumido no projeto lógico e a insatisfação do usuário por não se obter resultados rápidos, fazendo com que muitas empresas não utilizem esta metodologia de modo pleno. Outra desvantagem é a alta volatilidade dos processos de um sistema, que devem acompanhar as cada vez maiores e constantes mudanças exigidas pelo ambiente interno e externo das empresas. Assim, modelos orientados a processos, como é o caso da análise estruturada, fiquem obsoletos rapidamente, exigindo-se um grande esforço de manutenção.

Uma outra técnica, hoje muito utilizada devido principalmente à maturidade dos Bancos de Dados Relacionais, é a de *análise de dados* proposta inicialmente por Peter Chen (1976). Tal abordagem, utiliza a modelagem dos dados, isto é, a criação de um modelo de dados departamental e/ou organizacional apropriado, que dará sustentação às funções que devem ser realizadas pelos sistemas. Esta abordagem baseia-se na maior estabilidade dos dados em relação aos processos, e são portanto, menos sujeitos a manutenção constante. Permitem adicionalmente, com o uso de ferramentas atuais (Bancos de Dados, Datawarehouse e outras), a recuperação desses dados de modo mais simples e eficaz.

Os seguidores de Chen (1976) argumentam que a partir de uma boa análise de dados - criando-se o chamado modelo entidade-relacionamento - consegue-se implementar bases de dados que espelham melhor a realidade do usuário e assim, as funções seriam simplesmente resultado da conveniente manipulação desses dados.

Pode-se adotar, e normalmente é isso que acontece hoje em muitas empresas, uma abordagem mista de funções e de dados, de modo a se aproveitar as melhores características de cada uma delas. Mesmo assim, a reutilização do código gerado na programação ainda é difícil sem muitas adaptações, ficando a produtividade muito baixa.

Apesar das técnicas de análise estruturada, essencial (uma variante da estruturada) e de modelagem de dados atenderem boa parte da atual demanda de desenvolvimento de sistemas, algumas barreiras que estão surgindo fazem com que se tenha necessidade de outras ferramentas. Estas barreiras, segundo Martin (1995), são devidas principalmente aos seguintes fatores:

- os sistemas atuais são diferentes dos anteriormente desenvolvidos, tanto na maior complexidade (dado que os mais simples já foram desenvolvidos), quanto às maiores exigências dos usuários em áreas ainda pouco exploradas. Além disso, o uso desses sistemas dentro da estratégia competitiva da empresa os

torna cada vez mais importantes e limitam drasticamente o tempo de desenvolvimento, em função da necessidade de respostas rápidas exigidas pelo mercado;

- o número de linhas de código de programação necessárias para implementar os atuais sistemas, a necessidade de uma constante integração entre eles e o alto grau de funções automatizadas, são também fatores que limitam a capacidade das abordagens tradicionais de desenvolvimento;
- a volatilidade desses sistemas é alta devido principalmente às exigências e mudanças do ambiente tecnológico e de mercado, acarretando o encurtamento da sua vida útil..

A necessidade de desenvolvimentos mais ágeis, mesmo em ambiente complexos, que permitam um aumento sensível de produtividade, leva a abordagens como a *prototipagem*, que possibilita a criação de modelos rápidos e de modo dinâmico, contando ainda com a participação efetiva do usuário. Apesar de incorporada em algumas metodologias tradicionais, parece ainda não ser a resposta definitiva ao problema.

A técnica de *orientação a objeto* pressupõem que se possa criar sistemas compondo-se objetos previamente criados reduzindo-se o tempo de desenvolvimento, com um significativo ganho de produtividade e qualidade. Tal realidade deve necessitar de um novo modelo de desenvolvimento de sistemas para que se possa obter o máximo proveito desta situação. Nadler (1985) diz que deve-se investigar procedimentos que visem a maximização da efetividade de todo o pessoal envolvido com a produção de *software* e também preparando profissionais para que possam explorar as novas possibilidades.

Martin (1995) define um objeto com “qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e os métodos que os manipulam”. Pode-se ainda ter objetos que são constituídos de outros objetos, assim como uma máquina é composta de componentes que por sua vez são constituídos de outros componentes. Portanto, pode-se construir objetos cada vez mais complexos através de objetos mais simples. Coleman (1996) encara os objetos como “átomos do processo de computação que trocam mensagens entre si. Estas mensagens resultam na ativação de métodos, os quais realizam as ações necessárias. O emissor da mensagem não precisa saber como o objeto receptor organiza o seu estado interno, mas apenas que este objeto responde a certas mensagens de maneira bem definida”. A Figura 1 mostra, de forma gráfica, o conceito fundamental de objeto onde os dados e os métodos estão contidos (encapsulados) neste objeto e recebem mensagens do meio em que está inserido, respondendo a esses estímulos de maneira adequada.

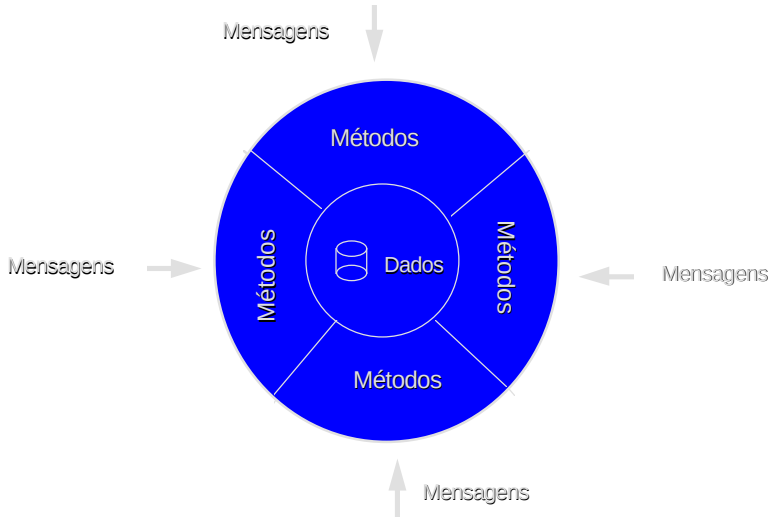

Figura 1 - Conceito Básico de um Objeto

Outro conceito fundamental na Orientação a Objeto é a de classes de objetos. Assim como na realidade concreta os objetos são associados em classes (por exemplo: automóveis) e subclasses (por exemplo: automóveis esportivos, de luxo, etc.), também pode-se fazer o mesmo na OO. Esses conceitos são ligados à propriedade de herança que diz que um tipo de objeto (classe) pode ter subtipos (subclasses). Segundo Martin (1995), "uma classe implementa o tipo de objeto. Uma subclasse herda as propriedades de sua classe-mãe; uma subsubclasse herda as propriedades das subclasses e assim por diante. Uma subclasse pode herdar a estrutura de dados e os métodos, ou alguns métodos, de sua superclasse. Ela também tem métodos e, às vezes, tipos de dados próprios."

Conforme Tapscott (1995), em um ambiente onde a OO esteja sendo plenamente utilizada, "o programador pode economizar tempo e esforço fazendo escolhas e, se necessário, modificando um determinado objeto ou uma combinação de objetos, para construir um novo aplicativo de *software*. Mediante a fusão de dois ou mais objetos já existentes, o desenvolvedor herda todas as funções e dados dos objetos; a seguir, esses objetos podem ser utilizados na construção de um novo aplicativo. Um fato muito importante é que o desenvolvedor herda toda a perícia daqueles que utilizaram e aperfeiçoaram o objeto no passado. Deste modo, passamos a poder reter experiências acumuladas e melhorar a qualidade do aplicativo novo resultante."

A orientação ao objeto como paradigma não é um conceito novo. Seus primeiros sinais apareceram na Noruega nos anos 60, através de uma linguagem chamada Simula-67, desenvolvida por Kristen Nygaard e Ole-Johan Dahl no Centro Norueguês de Computação. A linguagem Simula-67 apresentou pela primeira vez os conceitos de classes, rotinas correlatas e subclasses muito parecidos com as atuais linguagens orientadas a objeto. Algum tempo depois, em meados da década dos 70, cientistas do Xerox Palo Alto Research Center (Xerox PARC) criaram a linguagem Smalltalk, a primeira linguagem que pode ser chamada de orientada a objeto. Nesta linguagem, cada um dos seus elementos são implementados como um objeto . No desenvolvimento da Smalltalk, todos os aspectos da linguagem, o ambiente de programação e a cultura envolvida foram orientadas a objeto, e até hoje ela é considerada a mais pura destas linguagens.

A partir dos anos 80, a linguagem C torna-se uma linguagem popular de desenvolvimento, não só para microcomputadores como para outras arquiteturas e ambientes. No final da década de 80, Bjarne Stroustrup, da AT&T Bell Laboratories, expandiu a linguagem C, criando o C++, que suporta programação orientada a objeto. As subseqüentes versões comerciais e os incrementos nas ferramentas na linguagem C++, assim como um maior número de distribuidoras, ajudaram a chamar a atenção da comunidade de desenvolvimento de *software* para a programação orientada a objeto.

Segundo Tapscott (1995), a orientação a objeto nada mais é do que uma forma de padronização. Assim, segundo exemplo desse autor, "o desenvolvedor de *software*, da mesma forma que um construtor de casas, passa a poder escolher entre uma ampla variedade de portas, em vez de ter de projetar uma porta para cada casa nova. O tempo anteriormente gasto nesta fase freqüentemente redundante e improdutiva do desenvolvimento do aplicativo está agora sendo utilizado em novos requisitos de desenvolvimento e em melhores abordagens de projeto." A orientação a objeto permite programação com incrementos de alto nível de abstração - do objeto para a classe e para a biblioteca de classes - o que resulta em aplicações complexas executadas com alta produtividade.

A orientação a objeto se utilizada em toda a sua potencialidade deverá contribuir na qualidade técnica do projeto, satisfação do cliente e outros citados por Sbragia (1996). Apesar deste estudo ter mostrado que os fatores que se referem a metodologias não se mostrarem tão importantes, talvez sua influência em outros critérios seja maior do que parece. Particularmente, no paradigma de orientação a objeto, que necessita de um modo organizado de trabalho e no aumento de produtividade na criação de sistemas, a correta administração do processo pode ser a chave para o sucesso ou o fracasso de projetos.

A abordagem de orientação a objeto é tanto mais interessante e acrescenta maior produtividade e qualidade ao *software* quanto mais se implementa objetos e subclasses de objetos que possam ser reutilizáveis. Em um primeiro momento, a empresa que estiver desenvolvendo segundo esta abordagem

pode não sentir todos os ganhos de modo imediato mas a medida que novos objetos sejam criados e reutilizados os resultados começam a aparecer.

Para Martin (1995), a reusabilidade é um ato administrativo e os ambientes de desenvolvimento que queiram utilizar a orientação a objeto devem ser projetados e administrados convenientemente para se conseguir uma elevada reusabilidade, sob pena de não ter a vantagem oferecida por essa nova tecnologia.

Finalmente, segundo pesquisas recentes do Workgroup Technologies, Inc. citado na publicação da IBM "Transforming Your Business with Object Technology "(1993), a tecnologia de orientação a objeto terá atingido o seu desenvolvimento pleno a partir do ano de 1998 e seu crescimento a daí para frente será exponencial.

Uso da Orientação a Objeto nas Empresas - Pesquisa Bibliográfica

A bibliografia sobre desenvolvimento de sistemas utilizando o paradigma de orientação a objeto começa a ficar cada vez mais rica de casos. Até há alguns anos atrás, a orientação a objeto não passava de uma série de conceitos interessantes mas que careciam de exemplos concretos e que pudessem aprimorar métodos de trabalho. Hoje, conforme vai se aprofundando no assunto, descobre-se exemplos na literatura especializada, sejam livros ou revistas, na Internet e em dissertações e teses que apontam para um interesse crescente nesta área ainda com muitos conceitos a serem adotados em consenso.

Esta parte do trabalho procura selecionar alguns exemplos citados na bibliografia pesquisada pelo autor, destacando como seus pontos principais citados e comentários daqueles que já trabalharam com o paradigma de OO.

Alguns critérios apontados na literatura especializada indicam que a transição de uma metodologia convencional para um orientada a objeto deve ser feita de acordo com certos cuidados. Taurion (1997), diz que algumas armadilhas que podem comprometer esta transição, e que podem ser resumidas como :

1. subestimar o esforço de aprendizado, já que um simples treinamento em uma ferramenta OO pode não ser suficiente. É importante que se adquiram conceitos sólidos e que se respeite o tempo necessário dentro de uma curva de aprendizado, implicando em investimento razoáveis;
2. a escolha de ferramentas é importante pois, o *marketing* das mesmas pode levar a enganos. Algumas delas são vendidas como soluções OO e que permitem implementar soluções corporativas de missão crítica, o que na realidade pode se mostrar injustificado;
3. também com a escolha infeliz da consultoria pode-se ter graves problemas. Segundo Taurion, "nem sempre uma experiência profunda em C++ ou Smaltalk será suficiente para garantir o sucesso de um projeto OO. Um fator crítico é o gerenciamento do projeto". Ainda segundo o autor, esta consultoria deve ser escolhida não só pela sua experiência em gerenciamento, mas também através dos sólidos conceitos em OO, uso de ferramentas e conhecimento do negócio do cliente.

Mesmo dentro da OO, existe uma série de metodologias que podem ser aplicadas no desenvolvimento de sistemas. Pires & Faleiro (1996), mostram que uma metodologia OO é tanto mais robusta quanto maior o número de itens abaixo relacionados que sejam adotados. São eles :

1. "Está em uso por um período significativo de tempo - cinco anos ou mais - e durante este tempo existiu um processo de melhoria contínua da metodologia em função do uso.
2. Foi utilizada por um número significativo de projetos - em torno de 100 - com objetivos de implementação comercial de aplicações.
3. Um número significativo de referências na literatura técnica, estando documentada em pelo menos um livro disponível amplamente.
4. Suporta mais de uma ferramenta CASE disponível comercialmente

5. Foi utilizada com sucesso - no sentido amplo - na maioria dos projetos em que foi empregada."

Para se entender o uso da OO na comunidade de gerentes, analistas e programadores, uma primeira visão sobre a utilização da orientação a objeto é dada por Pires (1997) em pesquisa realizada para a revista Developers' Magazine. A empresa DRC Informações e Serviços Ltda., realizou uma pesquisa em um universo representativo de empresas brasileiras sobre a utilização da OO atualmente, e como elas estão planejando seu futuro.

Dentre as tabelas e linhas da pesquisa destacam-se alguns parâmetros de hoje que parecem significativos para se entender a dimensão deste novo paradigma.

- Qual o nível de aplicação da OO na sua empresa ?

Análise e projeto	27,8%
Programação	25,6%
Apenas projeto piloto	13,5%
Experimentação	15,8%
Apenas estudo	26,3%
Não estamos pensando	7,5%
Sem resposta	15,0%

- Quais as principais metodologias que estão sendo aplicadas ?

Coad/Yourdon	25,6%
Booch	4,5%
Rumbaugh (OMT)	15,0%
Jacobson (User case)	3,8%
CRC Card	0,8%
Unifield Method	3,0%
Shlaer/Mellor	3,0%
Fusion	4,5%
Outra	5,3%
Sem resposta	47,4%

- Quais as principais linguagens utilizadas ?

C++	15,0%
Smalltalk	0,8%
OOCOBOL	3,0%
Visual Basic	35,3%
PowerBuilder	9,0%
Delphi	33,1%
SQLWindows (Centura)	8,3%
Outra	17,3%
Sem resposta	24,1%

- Qual o escopo das aplicações desenvolvidas em OO ?

Departamental	37,6%
Corporativa	25,6%
Corporativa de missão Crítica	8,3%
Sem resposta	39,1%

- Quanto do orçamento para investimento em informática será aplicado em OO ?

Menos de 5%	31,6%
5 a 10%	12,8%
10 a 20%	6,8%
20 a 50%	8,3%
Acima de 50%	4,5%
Sem resposta	36,1%

- Há quanto tempo a OO está sendo aplicada na organização ?

Menos de 6 meses	38,3%
6 meses a 1 ano	21,1%
1 a 2 anos	12,8%
Mais de 2 anos	7,5%
Sem resposta	20,3%

- Quais as expectativas e benefícios esperados pela OO ?

Redução de custos	24,8%
Aumento de produtividade	66,2%
Redução de prazos	33,1%
Aumento de qualidade	56,4%
Desenvolv. de aplic. complexas	24,8%
Outras	3,0%
Sem resposta	12,8%

- Quais os receios e riscos esperados com a OO ?

Incap. de absorver novas tecnol.	14,3%

Dificuldade de aprendizado	31,6%
Falta de experiências práticas	43,6%
Imaturidade das tecnologias	33,8%
Custo para implementação	21,1%
Prazos de retorno demorados	24,8%
Outros	4,5%
Sem resposta	13,5%

- Qual o grau de satisfação com os resultados obtidos, até o momento, com OO ?

Totalmente satisfeito	20,3%
Parcialmente satisfeito	50,4%
Insatisfeito	6,0%
Sem resposta	23,3%

Casos Apresentados na Literatura Especializada

A seguir são expostos 6 casos do uso da OO que aparecem dentro da bibliografia pesquisada. Nesta apresentação não existe a preocupação de debater o que está descrito pelos autores mas sim transcrever as experiências vividas e registradas e apontar pontos chaves no gerenciamento de um ambiente OO.

ARS - Martin & Odell (1996) citam que "numerosas empresas já estão aplicando uma abordagem orientada a objeto ao planejamento, à análise, ao projeto e à implementação" e dentre elas, particularmente a ARS - Automation Research Systems, Limited - que utiliza a metodologia dos autores. A ARS descobriu que o desenvolvimento de sistemas em OO depende muito do método de aplicação e as lições aprendidas pela empresa foram quatro :

1. Definição do projeto da empresa - Quando se pensa em OO imagina-se somente uma rápida geração de código esquecendo-se de outros pontos importantes. Não se leva em conta os objetivos, políticas, estrutura, *layout* físico e assim por diante, comprometendo os resultados;
2. Metodologia - Deve-se modelar, além da estrutura e comportamento do objeto, os aspectos de negócio da organização - ambiente, objetivos, funções, pessoas, etc.;
3. Ferramentas - A empresa utiliza ferramentas CASE (Engenharia de *software* auxiliada por computador) acreditando que as mesmas são essenciais tanto para a qualidade do sistema, quanto para o desenvolvimento no tempo certo. Seus profissionais acreditam que a tecnologia CASE tem a capacidade de captar relações complexas que possam passar despercebidas pela mente humana, identificando e realçando incoerências para que recebam solução;
4. Treinamento - A empresa acredita que um método e uma ferramenta só são bons na medida da capacidade que o desenvolvedor de sistemas tem de usá-los.

A empresa em seu projeto piloto em OO conseguiu uma reusabilidade de código de 50 a 70 por cento entre três aplicações envolvidas. A melhoria foi nítida já que com os métodos convencionais o código utilizado era de 5 a 10 por cento. Também foi uma surpresa a velocidade com que modelos individuais podiam então ser desenvolvidos a partir de modelos básicos existentes.

Credit Suisse - Procurando aumentar a competitividade em escala global, o grupo Credit Suisse, fundado em 1856 em Zurique, partiu para uma proposta de flexibilização da Tecnologia da Informação. Com isso espera-se ganhar em rapidez no desenvolvimento de novas aplicações deixando-se a filosofia centralizada, fixa e monolítica dos sistemas de informação atuais para a filosofia de componentes. Tais componentes ou objetos, permitem a montagem rápida de soluções.
Segundo Smejda (1995), a filosofia OO permite que os desenvolvedores do Credit Suisse construam componentes que podem ser montados em múltiplas soluções de negócio. Gradualmente, até o ano de 2.005, a empresa terá modificada toda sua estrutura de Tecnologia de Informação. Hoje, os sistemas datam da década dos 70. Com esta nova estrutura a empresa espera ser 4 vezes melhor, com metade dos custos e com o tempo de

desenvolvimento de *software* também reduzido à metade, o que, segundo o artigo, é impossível com a atual tecnologia, fazendo-se necessário um novo paradigma.
O artigo mostra que graças a este novo paradigma surgiram novas formas de cooperação, especificamente grupos de trabalho (*teamwork*) muito eficientes. O *Credit Suisse*, segundo a autora, adotou grupos de trabalho segundo as teorias sistêmicas de hoje, com grupos reduzidos (8 a 15 pessoas), guiadas por um moderador. Este novo modelo de trabalho requer que se abandonem práticas tradicionais, adotando-se outras. Essas mudanças são :

- deixar o modelo de desenvolvimento linear (passo a passo), partindo-se para o desenvolvimento paralelo onde múltiplas partes de um mesmo problema são resolvidas simultaneamente;
- sair de uma estrutura de poder autocrática para outra democrática;
- abandonar os confrontos (sim, mas...) ficando com a integração (sim, e...);
- ir de uma estrutura com funções e responsabilidade fixas para outra de contribuição, onde se agregam conhecimentos e habilidades;
- de apresentações de mão única para formas interativas, de troca de informações no grupo orientadas à ação.

A autora finaliza o artigo com as palavras de um membro do grupo de desenvolvimento que diz que não se entende realmente o conceito de orientação a objeto até começar-se a trabalhar com eles e que, qualquer um que usou as antigas ferramentas de programação procedural deve aprender como enxergar objetos ao invés de funções e programas. A OO necessita mais do que uma nova maneira de pensar. Ela requer uma nova maneira de agir.

General Electric - Esta Aplicação de OO é descrita por Rumbagh *et alli* (1994). Trata-se do desenvolvimento para a General Electric de um sistema - chamado de *Compilador de Diagramas de Objetos* - que, a partir de uma encomenda de um cliente, deve gerar uma lista de material. Tal lista é uma árvore de peças diretas e indiretas que compõem uma montagem (equipamento), muitas vezes mencionada como um problema de explosão de peças.
O autor cita que o desenvolvimento dos modelos de objetos e funcionais, características da metodologia do próprio autor, foi muito importante, visto a grande quantidade de dados e passagens para tratá-los. O modelo dinâmico, outra característica da metodologia, foi relativamente pouco importante pois a interação do usuário com este sistema é pequena.
O projeto e a implementação desse sistema foram, a partir daí, rápidos, sendo só uma questão de detalhes mecânicos como preparar um pseudo-código, escrever o código real e depurá-lo. No final, o autor cita o excelente desempenho do sistema, a despeito dos requisitos pouco rígidos nesta área.

Companhia São Paulo de Petróleo - Para uma maior produtividade dos seus programadores e facilidades no processamento de notas fiscais, faturamento, cadastro de clientes e cadastros básicos, a empresa resolveu desenvolver dentro da OO. Para Carvalho (1994), o autor do artigo, o desenvolvimento próprio é necessário, pois o controle do governo na área tem uma série de particularidades que não justificam a compra de um pacote pronto.
A aplicação conta com aproximadamente 50 telas, envolvendo 70 tabelas. A equipe de desenvolvimento utiliza o conceito de classe de objetos, para permitir que, a partir da ocorrência de qualquer alteração relativa a um tema - pedido por exemplo - todo o contexto seja alterado. Na parte de faturamento, são incorporadas uma série de variáveis, até então processadas separadamente. A idéia é que o operador alimente o sistema em um menor espaço de tempo, com o mínimo de dados possível, gerando informações para a expedição de notas fiscais, alimentando outros sistemas como estoque e cobrança.
Na reportagem, o analista encarregado comenta que “a dificuldade que tivemos foi no entendimento da programação orientada a objeto”, pois para os programadores tradicionais fica mais di-

fícil entender os conceitos de OO, como por exemplo, que a ferramenta gera classes de forma que quando um fato ocorre, ela automaticamente solicita um procedimento.

DRM - Serviços de Consultoria - A empresa está no mercado desde 1988 e um dos serviços prestados é no desenvolvimento de aplicações. Recentemente a empresa repensou e redefiniu seu procedimentos rumando para a orientação a objeto, permitindo o rápido desenvolvimento de sistemas, a flexibilidade e a reusabilidade de código.
Os benefícios conseguidos pela empresa segundo Mellaci (1997), "não se encontram apenas restritos após a implementação de sistemas, mas inclusive do próprio desenvolvimento. Projetos já realizados, em regime misto, com instituições financeiras, envolvendo as fases de especificação, projeto, construção e homologação, foram realizados em cinco meses, gerando mais de 5000.000 linhas de código em C++, envolvendo 200 classes e 350 telas para interface com os usuários".
Ainda de acordo com o autor do artigo, foram registrados ganhos de 40% em relação a metodologias mais conservadoras. O autor não indica se tais ganhos foram financeiros ou de tempo, mas tudo indica que se trata de ganho de tempo. Para este projeto e para obter-se estes resultados, o número de profissionais utilizados gira em torno de 16 pessoas considerando os coordenadores de projeto, analistas de negócio, desenvolvedores de classes, documentadores, DBA (Administradores de Bancos de Dados) e suportes ao desenvolvimento (metodologia e arquitetura).
Finalizando o artigo, o autor registra que, além das características técnicas consideradas por este trabalho, um outro fator decisivo de sucesso é o correto e eficaz gerenciamento do projeto.

Mendes Júnior - A empresa passou a buscar uma plataforma de desenvolvimento com alta produtividade, robustez em aplicações de alta qualidade e baixo custo de desenvolvimento e manutenção. Isso deveria estar aliado à busca de atender a demanda reprimida das empresas do grupo, de forma ágil e competitiva. A orientação a objeto foi eleita como a metodologia que atendia a todos os itens. Segundo Maciel (1994), algumas diretrizes para a programação orientada a objetos foram então dadas :

- Flexibilidade - com a modularização proporcionada pelos objetos isolam-se alterações e minimiza-se o seu impacto. Uso dos mecanismos de herança e encapsulamento de dados e seus respectivos métodos, escondendo a complexidade;
- Agilidade - desenvolvimento mais rápido e concentração na análise do negócio e não nos detalhes de implementação;
- Qualidade - reutilização de códigos já testados e aprovados e possibilidade do uso da prototipação permitindo a participação dos usuários finais;
- Uso de interface gráfica MS Windows - permite maior facilidade de uso, integração com outros aplicativos MS Windows, sendo hoje um padrão da indústria;
- Integração com os principais Bancos de Dados Relacionais disponíveis - permite atender empresas com diferentes plataformas *back-end* de Bancos de Dados Relacionais, a partir de um mesmo conjunto de ferramentas *front-end* de desenvolvimento.
- Qualidade dos objetos nativos - disponibilidade imediata de linha completa de objetos GUI (Graphic User Interface) como *pushbottons*, *frames*, *listbox*, *combobox*, etc.;
- Facilidade das ferramentas - para a construção de telas, de relatórios, de consultas a Bancos de Dados, criação de classes e objetos e de aprendizagem, diminuindo o impacto da mudança de paradigma.

Apesar de ainda na fase piloto, na data do artigo, o autor identifica benefícios como : maior velocidade de desenvolvimento, facilidade de criação e manutenção de classes de objetos, facilidade de manutenção de programas, biblioteca de objetos robusta e confiável, alta qualidade dos sistemas implementados e boa performance, qualidade dos manuais e documentação técnica.

Para se tirar um bom proveito do ambiente de OO, o autor considera alguns aspectos que devem ser observados antes do desenvolvimento completo das aplicações. São eles :

- "Definir as metodologias de modelagem, análise e projeto.
- Personalizar as ferramentas de acordo com as metodologias.
- Definir os padrões para a criação de classes de objetos.
- Adaptar os objetos nativos e criar novos objetos.
- Estudar e entender com clareza a estrutura de herança e comportamento das classes de objetos nativos e criados.
- Divulgar os objetos e treinar a equipe de desenvolvimento no seu uso".

CONCLUSÃO

A evolução do paradigma de OO parece se dar de acordo com o esperado. Ainda falta a padronização de diversos conceitos, apesar dos esforços dos diversos grupos organizadores e normativos. A busca de uma maior produtividade em uma área crítica como o desenvolvimento de *software* e a falta de mão de obra qualificada tendem a acelerar a busca de novas ferramentas.

Existem também restrições quanto a metodologias (em grande número), gerenciamento, aprendizado e aceitação de um novo paradigma. O estudo de casos, tanto bem sucedidos quanto aqueles que resultaram em fracasso, certamente ajudarão a compor o talvez novo cenário cotidiano de desnvolvimento de *software* do próximo século.

REFERÊNCIAS BIBLIOGRÁFICAS

BEDEIAN, Arthur G. Management. Orlando,FL, Harcourt Brage Jovanovich College Publishers, 1993

BELLIN, Davis & **SUCHMAN** Susan. *Manual de desenvolvimento de sistemas estruturados*. São Paulo, Makron Books, 1993

BOOCH, Grady. Object Oriented Design - whit applications. Redwood City, CA, The Benjamin/Cummings Publishing Company,Inc., 1991

BROOKS, Frederick P. No Silver Bullet: Essence and Accidents of Softwre Engeneering. IEEE Computer, abr./87

BROWN, Bill. Leading a Successful Migation. *Object Magazine*, oct./96, pp. 38-43

CAMPOS FILHO, Maurício P. Os Sistemas de Informação e as Modernas Tendências da Tecnologia e dos Negócios. *RAE, Revista de Administração de Empresas*, FGV, São Paulo, vol. 34, n° 6, pp. 33 - 45, Nov./Dez. 1994

CARVALHO, Jackeline. *Pedidos e notas fiscais orientadas a objeto*. Computerworld, 28/11/94

CHASE, W. P. Management of System Engeneering. New York, John Wiley, 1974

CHEN, Peter P. The Entity-Relationship Model - toward a unifield view of data. ACM Transaction-Database Systems 1, Mar. 1976, pp. 9-36

COLEMAN, Derek et al. *Desenvolvimento Orientado a Objetos - O Método Fusion* -. Rio de Janeiro, Editora Campus, 1996

DEMARCO, Tom. *Análise Estruturada e Especificação de Sistema. Rio de Janeiro*, Editora Campus, 1989

ERICKSON, T. J. et alii. Managing Technology as a Business Strategy. *Sloan Management Review*. Spring . vol. 31, Number 3, 1990

FOURNIER, Roger. *Guia prático para desenvolvimento e manutenção de sistemas estruturados*. São Paulo, Makron Books, 1994

GAUSE, Donald C. & Weinberg, Gerald M. Explorando Requerimentos de Sistemas. São Paulo, Makron Books, 1991

GUPTA, A. K. and **WILEMON** D. Acelerating the Development of Technology-Based New Products. California Management Rev, Winter 1990

GUTHERY, Scott. Are the Emperor's New Clothes Object-Oriented. Dr. Dobb's Journal, december/1989

HendeRson-SELLERS, Brian. Book of Object-Oriented Knowledge. Englewood Ciffs, NJ, Prentice Hall, 1992

KEEN, Peter G. W. *Guia Gerencial para a Tecnologia da Informação*. Rio de Janeiro, Editora Campus, 1996

LITTLEWOOD, Bev & Strigini, Lorenzo. The Risk of *Software*. The Computer in the 21st Century - Scientific American, Special Issue, vol. 6, n°. 1, pp. 180 - 185, set/95

MACIEL, Paulo N. Estudo de Caso : A trajetória da Mendes Júnior na escolha de ferramenta para desenvolvimento de aplicações Cliente/Servidor com Orientação a Objeto. *Anais do Object Forum 94* - 1º Congresso Nacional sobre Orientação a Objeto, out/94

MARTIN, James & **MCCLURE**, Carma. *Técnicas Estruturadas e CASE*. São Paulo, Makron Books, 1991

MARTIN, James & **ODELL**, James J. *Análise e projeto orientados a objeto. São Paulo*, Makron Books, 1995

MARTIN, Merle P. Analysis and design of business information systems (2ª edition). New Jersey, Prentice-Hall Inc., 1995

MARTINS, Gilberto de A. *Manual para eleboração de monografias e dissertações* (2ª ed.). São Paulo, Editora Atlas, 1994

MELLACI, Fernando. Case Study : Gerando Sistemas Orientados a Objetos. *Developers' Magazine*, n° 12, pp. 44, agosto/97

MEYER, Bertrand. Object-Oriented *Software* Construction. Englewood Ciffs, NJ, Prentice Hall, 1988

MONARCHI, David E. & **PUHR**, Gretch I. *A Research Topology for Object-Oriented Analysis and Design. Communications of ACM*, vol. 35, n° 9, pp. 35, set/92

NaDler, Gerald. *Systems Methodology and Design. IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, n° 6, pp. 685 - 697, nov./dez. 1985

NERSON, Jean-Marc. **Applying Object-Oriented Analysis and Design. Communications of ACM**, vol. 35, n° 9, pp. 63, set/1992

OPTNER, S. L. System Analysis for Business Management. Englewood Ciffs, NJ, Prentice Hall, 1968

PIRES, Aldo. Pesquisa DevMag/DRC : O Uso da Orientação a Objetos no Brasil. *Developers' Magazine*, n°12, pp.12, agosto/97

PIRES, Aldo & **FALEIRO**, Jorge M. Um Comparativo para Metodologias de Desenvolvimento OO. *Developers' Magazine*, n.4, p.20, dezembro/96

PITTMAN, Matthew. Lessons Learned in Managing Object-Oriented Development. IEEE *Software*, vol 10, n° 1, pp. 43, jan/93

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

RUBIN, Kenneth S. & **GOLDBERG**, Adele. Object Behavior Analysis. *Communications of ACM*, vol. 48, n° 9, pp. 35, set/92

RUMBAUGH, James. Object-Oriented Modeling and Design. Prentice-Hall International Editions, New York, 1991

SBRAGIA, Roberto & **ROBIC**, André Ricardo. Sucesso em projetos de Informatização: Critérios de Avaliação e Fatores Condicionentes. *Caderno de Pesquisa em Administração FEA-USP*, São Paulo, vol.1, n° 2, pp. 1-12, 1º sem/96

SCHWARZ, Jim. Managing Object-Oriented Development Projects. Proceedings of *Software* Development, Santa Clara, CA, 1993

SETZER, Valdemar W. *Banco de Dados*. São Paulo, Editora Edgard Blucher Ltda, 1989

SMEDJA, Hellena. Credit Suisse has seen the future, and it's object oriented. *Software* Quaterly - *IBM's Magazine of Software Technologies.* vol. 2, n° 1, pp.53, 1995

TAPSCOTT, Don & **CASTON**, Art. Mudança de Paradigma. São Paulo, Makron Books, 1995

TAURION, Cezaron .Desafio: Como Migrar para Orientação a Objetos?. *Developers' Magazine*, Agosto/97

TAYLOR, David A. *Engenharia de Negócios com Tecnologia de Objetos*. Rio de Janeiro, Axcel Books, 1995

TAYLOR, David A.Object-Oriented Information Systems: Planning and Implementation. Reading, MA, Addison-Wesley, 1991

TAYLOR, David A.Object-Oriented Technology: A Manager's Guide. Reading, MA, Addison-Wesley, 1991

TORRES, Norberto A. *Manual de Planejamento de Informática Empresarial.* São Paulo, Makron Books, 1994

WALTON, Richard E. Tecnologia de Informação. São Paulo, Editora Atlas, 1993

WIRFS-BROCK, Rebecca et al. Design Object-Oriented *Software*. Englewood Ciffs, NJ, Prentice Hall, 1990

YIN, R. K. Case Study Research: Design and Methods. Sage Publications, EUA, 1989

YONG, Cu Shao. Tecnologia de Informação. Revista de Administração de Empresas FGV, São Paulo, pp. 78-87, Jan./Mar. 1992

YOURDON, Edward. Análise estruturada moderna. Rio de Janeiro, Editora Campus, 1990

YOURDON, Edward. Declínio e Queda dos Analistas e Programadores. São Paulo, Makron Books, 1995