IT-105

UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

Tema: Análise Comparativa entre Metodologias Estruturadas e Orientadas a Objectos

FASE DE ANÁLISE DO CICLO DE VIDA DE DESENVOLVIMENTO DE SISTEMA DE INFORMAÇÃO

TRABALHO DE LICENCIATURA

Autor:
José Vasconcelos

Maputo, Maio de 2002

UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

Tema: Análise Comparativa entre Metodologias Estruturadas e Orientadas a Objectos

FASE DE ANÁLISE DO CICLO DE VIDA DE DESENVOLVIMENTO DE SISTEMA DE INFORMAÇÃO

TRABALHO DE LICENCIATURA

Supervisores:
Dr.ª Esselina Macome e
Dr. Carlos Cumbane

Autor:
José Vasconcelos

Maputo, Maio de 2002

Agradecimentos

- Aos Meus supervisores, dr. Carlos Cumbana e Drª. Esselina Macome, pela força que imprimiram na orientação para que se tornasse possível a realização deste trabalho;
- À TDM, pela disponibilização dos recursos necessários para a execução deste trabalho;
- À minha família, que sempre soube apoiar-me em momentos mais difíceis e pela compreensão concedida;
- Aos meus colegas da Faculdade e do serviço e a todos que directa ou indirectamente contribuíram decisivamente para que este sonho se tornasse uma realidade;
- os meus melhores agradecimentos.

Dedicatória

Ao meu pai Vasconcelos e à minha mãe Ana. À minha esposa, Rabeca e aos meus filhos, agradecimentos

Declaração de Honra

Declaro por minha honra que este trabalho nunca foi submetido para a obtenção de qualquer grau e é resultado das minhas pesquisas e cujas as fontes utilizadas contém da página da bibliografia.

(José Vasconcelos)

Maputo, 28 de Maio de 2002

Resumo

A análise de sistemas é uma actividade que envolve o desenvolvimento de diferentes tipos de modelos sistemas. Para isso, é necessário o uso de técnicas adequadas. Actualmente se dispõe de dois paradigmas para a análise de sistemas: a análise estruturada e a análise orientada a objectos, esta última modelada na base da AOO (UML).

Esta dissertação apresenta um estudo comparativo entre os dois paradigmas, na fase de análise e especificação de requisitos no ciclo de vida de desenvolvimento de sistemas (CVDS). analisando os seus recursos voltados a modelos de dados e modelo funcional.

Este trabalho compreende nove capítulos. O capítulo 1 dedica-se essencialmente à introdução onde faz-se a definição do problema, objectivos bem como a metodologia e material de estudo usados; o capítulo 2 faz uma resenha do desenvolvimento e evolução das metodologias e suas limitações, assim como o surgimento de novas metodologias. O capitulo 3 aborda exclusivamente a fase de análise de requisitos que constitui a área de estudo.

O capítulo 4 apresenta as ferramentas necessárias para a análise e comparação das metodologias estruturadas e orientadas a objectos. Enquanto que o capitulo 5 faz a análise e comparação das ferramentas anteriormente apresentadas no capitulo 4; o capitulo 6 apresenta uma discussão sobre este estudo se a UML vem substituir ou complementar a metodologia estruturada; o capitulo 7 é a conclusão do trabalho; o capítulo 8 são as recomendações; o capitulo 9 apresenta a bibliografia utilizada e por fim apresenta-se o anexo.

Neste trabalho tomou-se como caso de estudo, uma agência imobiliária que se dedica ao aluguer de imóveis, onde se analisa e compara o modo como cada uma das metodologias identifica e representa o modelo funcional e o modelo de dados.

ÍNDICE

Acrónimos

AE	- Análise Estruturada
AOO	- Análise Orientada a Objectos
BDOO	- *Bases de Dados Orientadas a Objecto*
CASE	- *Computer Aided Software Engineering*
CVDS	- Ciclo de Vida de Desenvolvimento de Sistema de Informação
DP	- Descrição dos Processos
DD	- Dicionário de Dados
DC	- Diagrama de Contexto
DE	- Diagrama de Estrutura
DFD	- Digrama de Fluxo de Dados
DEA	- Diagrama de Entidades e Associações
DED	- Diagrama de Estrutura de Dados
ETHIC	- *Efective Technical and Human Implementation of Computer*
IE	- *Information Engineering*
ISAC	- *Information System work and Analysis of Changes*
MDSI	- Metodologias de Desenvolvimento de Sistemas de Informação
MSF	- *Microsoft Solutions Fra*mework
JSD	- *Jackson System Development*
YSM	- *Yourdon Sustem Method*
OO	- Orientada a Objectos
OO-BOOCH	- *Object Oriented segundo Booch*
OSSAD	- *Office Suport Analysis and Design*
OMT	- *Object Modeling Technique*
OOPSLA	- *Object Oriented Programming System languages and Aplications*
OMG	- *Object Management Group*
OOSE	- Object Oriented Software Engineering e Objectory
SI	- Sistemas de Informação
SSADM	- *Strutured System analysis and Design Method*
SSM	- *Soft System Methodology*
STRADIS	- *Strutured analysis and Design and Implementation of Information s*ystem
RAD	- *Rapid Aplication Development*
TI	- Tecnologias de Informação
UML	- *Unified Modeling Language*

1. Introdução

A divulgação das TI, genericamente designadas por informática, nas organizações, aconteceu segundo um processo gradual ditado, por um lado, pelos sucessivos avanços tecnológicos a nível de hardware e software e, por outro pelo, crescente reconhecimento das suas potencialidades no tratamento de informação.

A análise e desenho de sistemas é uma actividade importante e necessária nos dias de hoje pois envolve o estudo das interacções entre as pessoas, organizações, computadores "e por sua vez frustrante, repleta de relacionamentos entre pessoas, indefinida e difícil", [1].

A questão que se coloca é: Qual é a metodologia ou o processo a seguir para o desenvolvimento dos diferentes tipos de sistemas, dos quais o produto ou saída da análise de sistemas é que se torna entrada para o projecto da arquitectura geral do *hardware* e do *software*?

Muitas técnicas foram desenvolvidas e utilizadas ao longo dos tempos, é possível citar o "modelo - entidade-relacionamento" [2], que se preocupou com a diagramação dos modelos de dados, independentemente dos processos executados sobre os dados. Por outro lado, "a modelagem destes processos (funções) foi alvo de vários estudos através do DFD, [3]. Esta é uma ferramenta de modelagem que permite que se imagine um sistema como uma rede de processos funcionais interligados por tubos e tanques de armazenamento de dados".

Estes estudos foram a base da análise estrutura clássica. Entretanto, após alguns anos de experiência prática, foi identificada a necessidade de algumas alterações ou extensões. Estas alterações/extensões foram feitas e

[1] Booch, 1999
[2] Pereira, 1997
[3] Yourdon, 1992

resultaram no processo conhecido como análise estruturada moderna, que se tornou um padrão mundialmente aceite para análise de sistemas.

A análise e desenho de sistemas de informação está intimamente ligado ao aparecimento e evolução de computadores tendo como arma fundamental o uso de metodologias. Para facilitar o trabalho do pessoal envolvido e no desenvolvimento de sistemas de informação. Metodologias que vem simplesmente como meio de auxílio ao analista de sistemas e mais recentemente ao desenhador de sistemas na tarefa de desenvolver um sistema de informação que se pretende acima de tudo que seja fiável aos utilizadores, [4].

As metodologias orientadas a objecto surgem como desenvolvimento de metodologias estruturadas para responder as novas exigências de sistemas, isto porque as metodologias estruturadas já não eram capazes de corresponder, eficazmente, às necessidades de novos sistemas, mais complexos que representam diferentes situações da vida das organizações.

O maior desafio neste campo terá sido a criação de metodologias estruturadas e orientadas a objectos. Estas duas metodologias tem sido vulgarmente usadas nos últimos anos devido essencialmente ao seu poder técnico de representação diagramática.

Dada a diferenciabilidade dos princípios das metodologias é muito importante antes de se iniciar o desenvolvimento de qualquer sistema informático identificar alguns aspectos tais como a natureza do sistema, características e especificidades, recursos materiais e humanos e os objectivos a atingir.

Estes critérios poderão ajudar os desenvolvedores de sistemas a determinar qual é a metodologia a ser usada para o sistema que se pretende desenvolver. Neste trabalho vai-se fazer uma análise comparativa entre as duas metodologias na fase de análise de sistema do CVDS.

[4] Yourdon, 1992

A razão primordial que origina o desenvolvimento deste assunto é que com a revolução das tecnologias de informação existe uma gama de soluções alternativas para satisfazer os projectos de sistemas de Informação. Contudo, a maior dificuldade nos sistemas de informação persiste na escolha da alternativa para a solução do problema colocado ainda que devidamente colocado. No campo das tecnologias para o desenvolvimento de sistemas de informação estão disponíveis várias metodologias para o efeito, [5].

A pergunta que se faz é: O que usar nos dias de hoje? Análise estruturada ou análise orientada a objectos? Para melhor responder a estas perguntas é importante que se comparem os seus recursos e se identifique em que situações melhor se aplica um ou outro paradigma.

1.1 - Definição do problema

Presentemente vários estudos estão sendo feitos com o propósito de avaliar o estado actual das metodologias usadas no desenvolvimento de sistemas informáticos de modo a verificar quais são os avanços alcançados de forma a responder cabalmente as exigências dos sistemas de Informação [6]. Actualmente se dispõe de dois paradigmas para a análise de sistemas: AE e AOO.

A análise estruturada de sistemas compõe-se de um conjunto de técnicas e ferramentas, em constante evolução, surgidos do sucesso da programação e do projecto estruturado. Seu conceito fundamental é a construção de um modelo lógico, já que fornece uma visão lógica de como o sistema vai funcionar.

O paradigma orientado a objectos tornou-se um dos mais importantes na modelação de sistemas devido ao emprego dos conceitos, abstracções,

[5] Comolo, 1997
[6] Roque, 1998

encapsulamento e também à reusabilidade em grande escala dos componentes modelados , [7].

Mesmo nos dias actuais a OO produziu um forte impacto sobre diversas tecnologias tais como: sistemas distribuídos, sistemas *real-Time* Integrados, sistemas de informação, sistemas de negócio, sistemas técnicos, etc, [8]. Como se pode depreender com a AOO, consegue-se representar o comportamento e acções reais que são muito complexos e difíceis de ser representados em AE dadas as suas limitações. É por esse motivo que as metodologias orientadas a objectos surgem para colmatar essas limitações.

Como se sabe actualmente existem muitas metodologias de desenvolvimento de sistemas de informação e, dentre estas, importa saber seleccionar qual é a metodologia a ser usada para a solução do problema, se será necessário usar os dois paradigmas ou uma das metodologias.

Sendo assim, é importante identificar as diferenças e semelhanças existentes entre as duas abordagens de modo a auxiliar os desenvolvedores de sistemas de informação na selecção da metodologia apropriada para cada tipo de Sistema de Informação que se pretenda desenvolver isto devido à complexidade das organizações e sistemas de informação.

1.2 - Objectivo geral

O Objectivo geral deste trabalho é analisar comparativamente as metodologias estruturada e orientada a objectos na fase de análise do sistema.

[7] Yourdon, 1991
[8] Andrade, 1998

1.3 - Objectivos específicos

- Identificar os aspectos comuns entre as metodologias estruturadas e orientadas a objectos;
- Identificar as diferenças entre as metodologias estruturadas e orientadas a objectos;
- Avaliar as metodologias estruturadas e orientadas a objectos na perspectiva de encontrar os pontos em que as metodologias se poderão complementar;
- Comparar os requisitos das metodologias estruturadas e orientadas a objecto na fase de análise e especificação de requisitos;
- Analisar e avaliar até que ponto as metodologias orientadas a objectos vem substituir as estruturadas.

1.4 – Metodologia e fontes utilizadas

Para a alcançar os objectivos traçados neste trabalho foi feita uma abordagem descritiva e comparativa entre as duas metodologias que se baseia na compreensão das diferentes fases do ciclo de vida do desenvolvimento de sistemas.

Para a obtenção dos dados necessários foram usadas as fontes bibliográficas que constam deste trabalho, consulta a artigos e trabalhos relacionados com o tema na *Internet*. O presente trabalho é de carácter investigativo Para a escrita do relatório final foi usado um computador *pentium* II, 32 Mb de *Ram*, *Windows* 98 e *Office* 97.

2. Modelos de desenvolvimento de sistemas de informação

2.1 - Conceito de metodologia

Metodologia é um conjunto recomendado de filosofias, fases, procedimentos, técnicas, regras, ferramentas, documentação, gestão e treinamento para o desenvolvimento de um sistema de informação. Verifica-se neste conceito a

inclusão, entre outros, de filosofias que são as teorias e crenças que norteiam os objectivos e procedimentos de uma metodologia. Esta definição é uma tentativa de generalização, [9].

2.2 - A necessidade de metodologias

Os primeiros sistemas de Informação desenvolvidos, na década de 60, foram largamente implementados sem o auxílio de uma metodologia explícita de desenvolvimento de sistemas de Informação. Nesses anos, as pessoas que implementavam sistemas de informação eram programadores que não tinham, necessariamente, habilidades adequadas de comunicação ou compreensão das necessidades dos usuários.

Na realidade havia a preocupação com as habilidades técnicas dos programadores, deixando-se as demais para um segundo plano. Adicionalmente, os sistemas de Informação desenvolvidos geralmente custavam muito mais como também demoravam além do esperado para serem colocados em uso, [10].

Poucos programadores seguiam algum tipo de metodologia baseando-se, em sua maioria, na própria experiência. Modificações no sistema em desenvolvimento, devido a novas necessidades de seus usuários ou a uma deficiência na especificação inicial destas necessidades, levavam a efeitos indesejáveis e inesperados nas demais partes do sistema, [11]. Adicionalmente, o uso crescente das TI associado a necessidade de gestão por sistemas apropriados levou à situação em que se tornou necessário um método capaz de orientar o desenvolvimento do SI.
Por consequência na década seguinte verificaram as seguintes mudanças:

[9] Fitzgerald, 1988
[10] Roque, 1998
[11] Roque, 1998

1. O reconhecimento crescente de que parte do desenvolvimento de sistemas envolve análise, projecto e construção, existindo, portanto, funções distintas de analista de sistema e de programador;

2. A conscielização de que as organizações estavam crescendo em tamanho e complexidade, sendo mais desejável os sistemas de informações integrados do que soluções específicas para os problemas de cada processo organizacional.

Como forma de apoiar os gestores e analistas no desenvolvimento de sistemas de informação foram sintetizadas metodologias formais para o planeamento e coordenação de esforços, quer num dado projecto, quer integrando os múltiplos projectos realizados numa organização de modo a rentabilizar os esforços.

As metodologias tem por objectivo a definição de directrizes gerais que englobam:

- Descrição dos elementos chave dos quais as aplicações dependem e sobre as quais são construídas;
- Descrição das inter-relações entre os elementos chave;
- Documentação do estado actual das necessidades de Informação;
- Definição de planos futuros.
- Uma metodologia deve satisfazer as condições seguintes:
- Fornecer uma abordagem integral com preocupações sobre o planeamento estratégico do ambiente global do sistema de Informação em coerência com a estratégia de negócio da organização; a identificação dos requisitos a ter em conta; a concepção do sistema para a satisfação dos requisitos, a programação, teste e instalação de novos sistemas ou alterações de sistemas já existentes , [12].
- Facilitar e padronizar a adaptação do sistema de Informação às estratégias da organização, assegurando a minimização de riscos de investimento em tecnologias de Informação e fornecer referências e

[12] Roque, 1998

modelos normalizados; fornecer processos de estimar, planear e controlar projectos de desenvolvimento de sistemas; recomendar as técnicas de engenharia de software mais adequadas e garantir a comunicação eficiente entre os diferentes intervenientes nos projectos, [13].

Em resposta ao cenário formado na década de 70, passou a ser utilizado um modelo metodológico para desenvolvimento de sistemas de informação: o modelo *Waterfall,* [14]. Também conhecido como análise de sistemas convencional ou ciclo de vida do desenvolvimento de sistemas, este modelo teve e tem um papel fundamental na área de sistemas de informação, sendo a base sobre a qual foram criadas a maioria das metodologias existentes.

2.3 - O Modelo *Waterfall*

Desenvolvido no final da década de 1960 e princípios da década de 1970, o modelo *Waterfall* é ainda hoje a abordagem mais praticada no desenvolvimento de sistemas de informação. Esta abordagem assume que um sistema de informação tem um ciclo de vida semelhante ao de qualquer produto, com início, meio e fim, [15]. Cada etapa do ciclo de vida de desenvolvimento de sistemas pressupõe actividades que devem ser complementadas antes do início da seguinte etapa.

Existem várias versões do modelo *Waterfall* segundo os vários autores das metodologias de desenvolvimento de sistemas de informação. Este é também conhecido como modelo convencional e contemporâneo ou modelo cascata, e, a versão tradicional do CVDS apresenta a divisão mostrada na figura 2.1.

Existem muitas variantes deste modelo propostas pelos diferentes pesquisadores ou desenvolvedores de *software* e adaptadas a diferentes tipos de *software*. A característica comum é um fluxo linear e sequencial de actividades semelhantes.

[13] Roque, 1998
[14] Avison, 1997
[15] Avison 1997

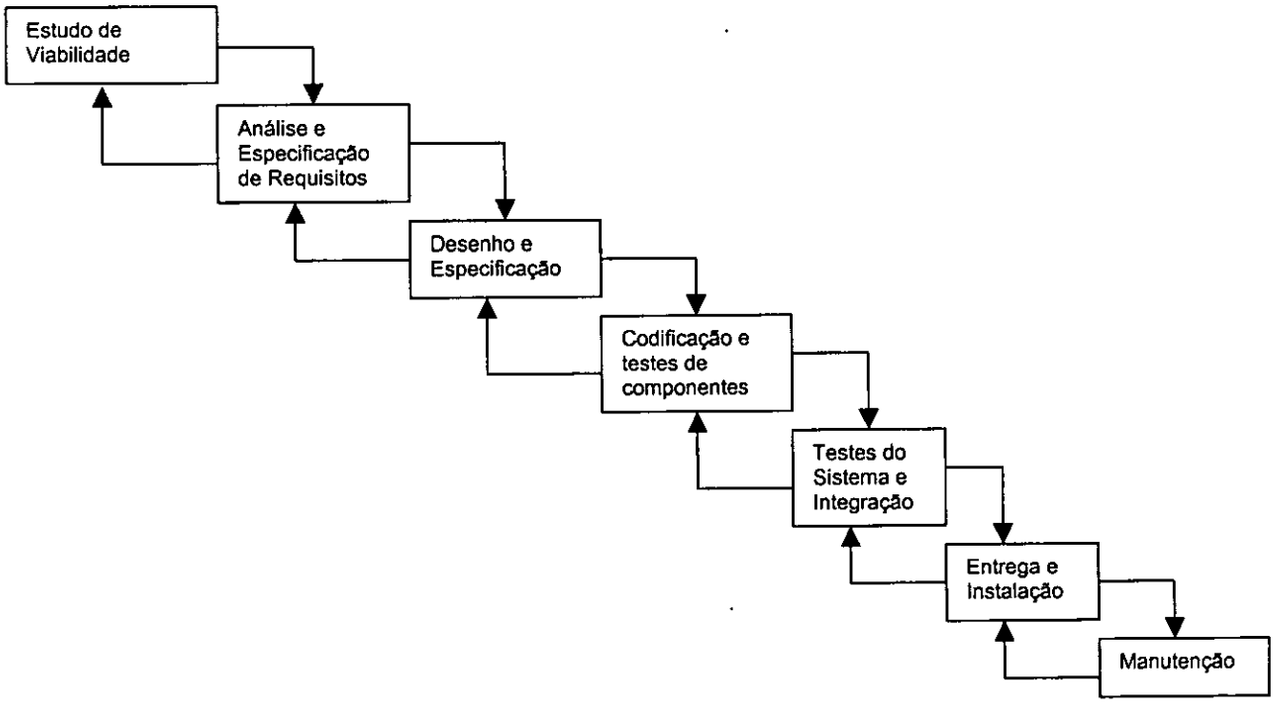

Fig 2.1

Outros autores, apresentam apenas quatro etapas: análise, projecto, codificação e teste. Nesta situação, algumas etapas englobam actividades que distribuíram em duas, como por exemplo, a etapa da análise consistiria da definição do projecto e estudo do sistema. Esta diferença encontrada nas actividades relacionadas a cada estágio em nada modifica o modelo *Waterfall,* [16].

Independentemente das diferenças observadas entre autores, o modelo *Waterfall*, possui características que o recomenda. Primeiramente este modelo já foi experimentado e testado extensivamente. Verificou-se, ao final de cada fase, a oportunidade dos analistas e usuários avaliarem o progresso obtido até aquele momento. Da mesma forma, a divisão do desenvolvimento de um sistema em fases permitiu um controle maior sobre este desenvolvimento, fato que pode, teoricamente, representar melhores resultados - melhores sistemas de informação. O modelo de desenvolvimento "*Waterfall*", é considerado pilar para o desenvolvimento de outros modelos de desenvolvimento de sistemas de informação.

[16] Roque, 1998

2.4 - Limitações do modelo *Waterfall*

Mesmo sendo o modelo referencial da grande maioria das metodologias existentes, o modelo *Waterfall* apresenta algumas limitações que devem ser ressaltadas, [17]:

1. Uma vez que considera que uma etapa deve ser iniciada após a conclusão das actividades da etapa anterior, é gasto uma quantidade razoável de tempo e esforços levantando informações, especificando e documentando-as a cada etapa para sua posterior utilização; este facto pode levar à demora na instalação do sistema tornando-o, muitas vezes, obsoleto quando efectivamente colocado em operação. Os resultados que são observados na etapa pós-implementação são demorados, isto é, não ocorrerão até que muitos passos tenham sido completados. A maioria das implementações do ciclo de vida em cascata apoia-se em fases sequenciais, o que significa que meses ou anos podem se passar antes que os usuários vejam qualquer evidência tangível de progresso. A exigência de uma formalidade volumosa, baseada em papel, leva a maioria das organizações e a maioria dos profissionais da área, que não têm tempo nem disposição, a praticar o ciclo de vida tradicional de um modo menos rigoroso e formal;

2. Devido ao seu grau de formalidade que exige especificações e documentações para cada processo que o sistema de Informação executa, alterações são inibidas tornando este processo, muitas vezes, inflexível à mudanças. O próprio processo de detecção de erros no ciclo de vida em cascata clássico é reservado à fase de teste formal do projecto. Neste estágio, a pressão nas actividades finais de desenvolvimento do sistema como detecção de erros de análise e

[17] Roque, 1998

projecto, levam a situações onde torna-se difícil a correcção destes tendo em vista o custo associado à eliminação dos erros;

3. Os processos de tomada de decisão exigem, na maioria das vezes, actividades não estruturadas, que não possuem procedimentos bem definidos. Esta realidade, dentro de uma abordagem tradicional - e formal - dificulta a definição das especificações do sistema, dependendo de requisitos correctos e estáveis. No ciclo de vida em cascata a qualidade da codificação depende da qualidade do projecto, e a qualidade do projecto depende do esforço de análise. Se os requisitos do usuário tiverem sido mal interpretados ou mal entendidos, ou se o usuário alterar os requisitos durante a fase de projecto e implementação subsequente, o ciclo de vida poderá não produzir resultados para o real problema determinado;

4. Uso da implementação *bottom-up*. A implementação *bottom-up* inicia o seu trabalho testando os módulos do sistema, depois subsistemas e finalmente o sistema. Assim, os erros mais sérios (integridade do sistema) são encontrados no final e não no início da fase de testes [18].

2.5 - Abordagens conservadoras versus abordagens radicais

A maioria das metodologias baseadas no modelo *Waterfall* presume que uma actividade deve ser concluída antes que a próxima se inicie. Devido as limitações anteriormente citadas, começou-se a pensar na utilização mais flexível deste modelo. Assim, uma situação extrema, existiria todas as actividades do ciclo de vida em cascata desenvolvendo-se simultaneamente e, no outro extremo, a utilização da abordagem sequencial, na qual há a conclusão de todas as actividades de uma etapa antes de começar a seguinte.

[18] Roque, 1998

Uma abordagem radical ao ciclo de vida em cascata seria aquela em que todas as actividades se desenvolvem paralelamente desde o início do projecto. Em contraste, numa abordagem conservadora toda actividade N é completada antes que a actividade N + 1 se inicie.

Há um número infinito de opções entre os extremos radicais e conservadores. Como exemplo de opção entre a abordagem conservadora e a abordagem radical, existe a conclusão completa de algumas etapas antes do início da etapa seguinte (abordagem conservadora) e o início de outras etapas sem a conclusão completa das etapas anteriores (abordagem radical) [19], a escolha entre as diversas possibilidades disponíveis depende:

- Nível de estabilidade do usuário: Quando o usuário caracteriza-se por sua instabilidade ou inexperiência na definição de suas reais necessidades, situação que levaria a criação de sucessivas especificações, uma abordagem mais radical faz-se necessária;

- Nível de urgência na produção de resultados tangíveis e imediatos: Se por questões políticas ou outras pressões externas, foram necessário que o sistema esteja a funcionar em determinada data, então, uma abordagem radical é justificada. Nestas situações parece ser mais importante ter 90% do sistema completo na data especificada do que 100% das etapas de análise e projecto;

- Exigências de se produzir cronogramas, orçamentos, estimativas, etc. A maior parte dos grandes projectos necessita de estimativas de pessoal, recursos de computação e outros relativamente detalhados, o que só é possível depois de um levantamento, análise e projecto razoavelmente detalhados. Assim, quanto mais detalhadas e precisas as estimativas tiverem de ser, mais provavelmente uma abordagem conservadora se fará necessária, [20].

[19] Yourdon, 1990
[20] Roque, 1998

2.6 - Modelos alternativos

Tendo em conta as limitações da abordagem tradicional, o desenvolvimento de sistemas de informação poderia ser administrado numa série de incrementos. Assim, poderia haver uma série de ciclos de vida tradicionais para cada incremento. Este modelo alternativo é conhecido como modelo incremental.

Modelo de desenvolvimento incremental

Seguindo a mesma linha, foi proposto o modelo - *Evolutionary spiral process*. Este modelo baseia-se em quatro principais actividades:

1. Determinação dos objectivos, alternativas e restrições;
2. Análise de risco e prototipagem;
3. Validação e verificação;
4. Planeamento da fase seguinte.

Esta concepção tende a criar um roteiro de actividades e etapas para que se alcance uma maturidade do processo evolutivo de desenvolvimento de sistemas complexos e obter se no fim do processo, um produto na sua forma mais completa possível.

Modelo de desenvolvimento de protótipos

Este modelo é também usualmente denominado por prototificação, subdividida em dois tipos: rápida prototificação e prototificação estruturada. A rápida prototificação caracteriza-se por ser um método de desenvolvimento incremental de sistemas. O seu objectivo é desenvolver rapidamente as partes de um sistema e de uma forma contínua obter-se retorno dos resultados de desenvolvimento para a avaliação pelos usuários.

Modelo de desenvolvimento evolucionário

O modelo caracteriza-se pelo levantamento de uma série de esforços de desenvolvimento que conduzem o processo para a obtenção dos produtos que providenciam alguma capacidade operacional necessária, enquanto ao mesmo tempo vai-se gerando os requisitos do sistema.

Modelo de desenvolvimento espiral

O modelo de desenvolvimento espiral engloba os modelos incremental, protótipos e evolucionário. Cada fase deste modelo é baseado nas respectivas fases de cada um dos modelos integrantes. Uma maneira muito simples de analisar este modelo é considerá-lo como um modelo cascata onde cada fase é precedida por uma análise de risco e a sua execução é feita evolucionariamente (Incrementalmente).

Modelo de desenvolvimento de transformação

O modelo de transformação tem as suas raízes na abordagem de métodos formais para o desenvolvimento de *software*: A ideia é que o desenvolvimento deve ser visto como uma sequência de passos que gradualmente transforma uma especificação formal num programa. O passo inicial é transformar os requisitos informais numa especificação funcional formal. Esta descrição formal é então transformada numa outra descrição formal mais detalhada, e assim, sucessivamente, até chegar ao ponto em que se tenha componentes de programas que satisfaçam a especificação.

Durante este processo de transformações sucessivas – chamado de optimização – as especificações formais produzidas podem ser executadas por um processador abstracto, permitindo uma verificação formal da sua validação antes mesmo que o programa venha a ser implementado. Antes de serem transformadas cada especificação formal deve ser verificada em relação às expectativas dos usuários.

O Modelo de desenvolvimento constrói e conserta

O produto é construído sem qualquer especificação ou projecto. O produto é de novo trabalhado quantas vezes for necessário para satisfazer o cliente. Este modelo pode funcionar razoavelmente para sistemas de pequeno porte.

2.7 - Pontos essenciais para comparação de metodologias

Com este capitulo pretende-se analisar-se as delimitações do conjunto das matérias do estudo para a modalidade de comparação e fornecimento de ênfase sobre os aspectos essenciais que são tratados neste trabalho. Cada matéria é retirada do seu conjunto específico mediante as razões apontadas em cada caso, cuja a ideia fulcral é atingir as metas finais traçadas para este estudo. É por isso que se destaca o CVDS entre os vários modelos de desenvolvimento de sistemas de informação, sendo a fase de análise do CVDS uma das fases que compõem o CVDS dos modelos.

2.8 - Uma classificação das metodologias

Independentemente do modelo utilizado como referência metodológica, verifica-se que um sistema de informações possui três dimensões de complexidade que precisam de ser modeladas: funções, dados e o comportamento dependente do tempo.

2.9 - Complexidade das metodologias

As metodologias actualmente utilizadas podem ser classificadas pelo nível de enfoque dado a cada uma dessas dimensões. Dessa forma, existem as metodologias com enfoque nas funções do sistema, as metodologias com enfoque nos dados do sistema e as metodologias com enfoque no comportamento do sistema, representado pelas funções e dados.

Cronologicamente as primeiras metodologias desenvolvidas voltavam-se para as funções do sistema - processos - e eram chamadas de estruturadas. Posteriormente surgiu a engenharia da informação, com ênfase nos dados e mais recentemente as metodologias orientadas a objecto. Algumas metodologias, no entanto, extrapolam os enfoques considerados nesta classificação sendo por isto agrupadas como um grupo especial - outras metodologias - e analisados individualmente.

Observa-se que há um ciclo típico para revoluções em todos os campos da ciência, inclusive no que se refere a metodologia de desenvolvimento de sistemas de Informação. Dessa forma, quando uma nova metodologia é criada isto decorre da necessidade de solução de problemas até então não resolvidos e que naturalmente irá substituir progressivamente as metodologias mais antigas, [21]. Assim, a medida que é aplicada a problemas maiores e mais complexos, esta nova metodologia passa a enfrentar problemas até então desconhecidos e, após inúmeros ajustes e alterações, uma nova metodologia se torna necessária.

Cabe ressaltar que, baseando-se nesta classificação das MDSI - estruturada, engenharia de informação, orientação a objectos e outras - há de considerar a própria evolução de cada classe de metodologias. Assim há diferentes níveis evolutivos entre as metodologias estruturadas criando-se a classificação de metodologias estruturadas primitivas.

Metodologias estruturadas média e metodologias estruturadas avançadas. Esta subclassificação permite compreender a tendência, entre aqueles que utilizam metodologias estruturadas primitivas, que evoluem inicialmente para metodologias estruturadas modernas e posteriormente para metodologias de engenharia da informação ou orientadas a objectos, [22].

Outro factor relevante, no que se refere à mudança de metodologias, refere-se ao fato de, ao se começar utilizar uma nova metodologia há uma razoável

[21] Davies, 1993

demanda de tempo na aprendizagem desta e das técnicas e ferramentas a ela associadas. Assim, mudanças se tornam justificadas, quando a antiga metodologia não consegue lidar facilmente com problemas existentes.
Considerando a curva de experiência das metodologias estruturadas, engenharia da informação e orientada a objectos, pode-se verificar que:

- As metodologias estruturadas encontram-se no estágio de maturidade;
- As metodologias de engenharia da informação estão no estágio de ascensão e
- As metodologias orientadas a objectos encontram-se num estágio de expansão inicial.

2.10 - A evolução das metodologias

No que se refere aos modelos referenciais, a maioria das metodologias, não importa a classe, baseia-se no modelo *Waterfall*. Cabe ressaltar, porém, que estas metodologias, principalmente as mais recentes, procuram evitar as limitações do modelo anteriormente focados, utilizando abordagens menos conservadoras.

Baseando-se na classificação das metodologias de acordo com o enfoque na modelagem do sistema, este estudo vai analisar as seguintes metodologias: Metodologias estruturadas *(STRADIS*, metodologia estruturada moderna, *YSM, SSADM, Merise*), engenharia da informação, metodologia orientada a objectos, *ISAC*, *ETHICS*, *Multiview*, *RAD*, *OSSAD* e *Microsoft Solutions Framework* apresentados algumas metodologias de destaque dentro de cada classe - estruturada, engenharia da informação, orientação a objectos e outras, [23].

[22] Davies, 1993
[23] Roque, 1998

2.11 - Interacção entre as metodologias e o CVDS

Para a execução de um sistema de informação é imprescindível a escolha de uma metodologia, esta metodologia deve obrigatoriamente seguir um CVDS: Existem várias MDSI, o que torna-se muito difícil para muitos desenvolvedor de sistemas é escolher que tipo de metodologia deve usar para o desenvolvimento de um dado sistema. Isto depende de muitos factores tais como organizacionais, tipo de sistema que se pretende, volume de informação que vai manusear, experiência da equipa de desenvolvimento e outros factores indeterminados.

De modo a facilitar a análise comparativa das metodologias, importa neste momento mencionar as metodologias que serão alvo de estudo.
Metodologias estruturadas: *STRADIS – Structured Analysis, Design and Implementation of Information, YSM – Yourdon Systems Method, SSADM – Structured Systems Analysis and Design Methodology, Merise*, Metodologia estruturada Moderna, Engenharia de Informação.

Metodologias orientadas a objectos: *ISAC – Information Systems Work and Analysis of Changes, ETHICS – Effective Technical and Human Implementation of Computer-Based System, Mutiview, Rad – Rapid Application Development, Ossad – Office Support Analysis and Design, MSF – Microsoft Solutions Framework e SSM – Soft System Methodology* [24].

Importa referir que cada metodologia possui a sua própria filosofia, técnicas, ferramentas e algumas tem o propósito para certas áreas de aplicação. [25] E, cada metodologia adequa-se melhor a determinadas fases do CVDS segundo suas características. A *STRADIS* é uma metodologia baseada na decomposição funcional, enquanto que a metodologia estruturada moderna, possui a decomposição funcional e o projecto *top-down*, através dos quais

[24] Comolo, 1998
[25] Davis, 1993

um problema é sucessivamente decomposto em unidades geríveis, [26]. A *YSM* cobre as actividades de organização e sistema.

A *ETHIC* é uma metodologia participativa, *SSM* é evolutiva, *Multiview* é híbrida. Uma característica também importante a ter em conta na distinção das metodologias é a sua capacidade de resolver problemas. Há metodologias para a construção de sistemas complexos que são facilmente identificáveis, por exemplo *SSM* e outras para sistemas claramente definidos, [27]. As metodologias estruturadas são as mais antigas tradicionalmente usadas.

A interacção dos dois paradigmas nos CVDS é bastante diferenciada, isto porque as metodologias estruturadas possuem muitas fases no CVDS, enquanto as OO tentam sempre juntar as primeiras fases formando só uma. Isto quer dizer que muitas actividades de várias fases são acopladas numa, e outras fases são quase que as mesmas com as metodologias estruturadas e se acentuam significativamente existindo pouca diferença. Para melhor orientar este estudo, é conveniente trabalhar com modelos unificados orientados a objectos.

2.12 - Benefícios do uso de uma metodologia

A escolha de uma metodologia para o desenvolvimento de sistema de informação constitui um dos passos mais importantes no processo de desenvolvimento de sistemas, é através dele que as equipas e seus membros mantém uma linguagem comum durante todo o ciclo de vida do desenvolvimento. Os principais benefícios na adopção de uma metodologia são:

- Permitir a compartilha da mesma filosofia de trabalho entre os desenvolvedores de sistemas de Informação;
- Evitar trabalho desnecessário;

[26] Yourdon, 1989

- Estabelecer pontos de verificação para acompanhamento e controle de execução do projecto do sistema em desenvolvimento;
- Aumentar a produtividade;
- Facilitar o envolvimento do usuário no projecto;
- Providenciar uma notação e linguagem comum, [28].

2.13 - Surgimento de UML

Nos últimos anos, uma nova visão de análise de sistemas começou a surgir: a de Análise Orientada a objectos (AOO). A primeira metodologia foi publicada por *Shlaer/Mellor* em 1988 e, desde então, cerca de sessenta metodologias AOO foram disponibilizadas, sendo que, cada uma, possui notação, semântica e ferramentas próprias, sendo incapazes de se comunicarem, [29].

Em setembro de 1993, os metodologistas iniciaram um acordo para a selecção de metodologias orientadas a objecto, que ficou mais evidenciado na *OOPSLA*, em 1994. Dois pontos foram ponderados:

a) Grande número de metodologias disponíveis trazia dificuldades de avaliação, além do temer que algumas sobrevivessem apenas por um curto espaço de tempo;

b) Cada metodologia aparecia baseada num modelo diferente de objectos.
O caminho para estes problemas era a tentativa de uma unificação. Em junho de 1995 o *OMG – Object Management Group* estabeleceu o trabalho de um grupo força tarefa para a análise e projecto OO visando a padronização do núcleo das metodologias OO. A comunidade OO foi convocada a apresentar propostas para a unificação de metodologias. Este objectivo começou a se concretizar com a união de *James Rumbaugh*, autor da *OMT – Object Modeling Technique, e de Grady*

[27] Comolo, 1998
[28] Andrade at All, 1998
[29] Tonsig, 1998

Booch, desenvolvido pelo Método de Booch. Em Outubro de 1995, na *OOPSLA* foi apresentada a especificação 0.8 da *UML – Unified Modeling Language*. Esta notação não dava ênfase à área dedicada a modelagem de negócios e nem a levantamento de requisitos, lacuna preenchida com a participação de *Ivar Jacobson*, desenvolvedor das metodologias OOSE – *Object Oriented Software Engineering* e *Objectory*. Assim, em junho de 1996, era lançada a especificação 0.9 da *UML*. A especificação 1.0 foi apresentada em janeiro de 1997, e o *OMG* adoptou a especificação 1.1 da *UML* em novembro de 1997. A última especificação (1.3) da *UML* foi lançada em abril de 1999, [30]. Eis as fases do Desenvolvimento de um sistema em *UML*.

- Análise de Requisitos;
- Análise;
- *Desenho* (Projecto);
- Programação;
- Teste.

As cinco fases acima mencionadas não devem ser executadas na ordem descrita, mas concomitantemente de forma que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas anteriormente de maneira que o resultado global gere um produto de alta qualidade e performance. Para o caso especifico deste trabalho, interessa falar exclusivamente da fase de análise e especificação de requisitos, [31].

3. Fase de análise

3.1 - Motivações para escolha da fase de análise

A fase de análise constitui a base deste trabalho, nesta fase vou comparar as técnicas de metodologias estruturadas e metodologias orientadas a objectos,

[30] Booch, 1999
[31] Barros, 1998

fundamentalmente sobre as ferramentas para a análise, documentação e especificação dos requisitos do sistema.

Porém, para a efectivação deste estudo, escolhi a fase de análise e especificação de requisitos, porque esta fase é a fase mais sensível para o sucesso do sistema em termos de satisfação dos problemas dos usuários.

Mesmo assim, as metodologias actuais preocupam-se muito com a análise porque é nesta fase onde se estudam todos os requisitos que o usuário apresenta aos analistas, e torna-se mais difícil ainda enquadrar todos esses requisitos quando se transita para a fase de desenho.

Esta tarefa é muito delicada. É por isso que muitos sistemas mesmo após a sua conclusão não satisfazem completamente as reais necessidades dos usuários, por um lado porque o analista não fez conforme o usuário desejava, por outro porque o usuário não consegue dizer cabalmente o que quer que o sistema faça ou ainda porque é difícil entender os requisitos do usuário.

3.2 - Análise e especificação de requisitos

A análise e especificação de requisitos de sistemas é uma tarefa muito importante a ser considerada quando se pretende desenvolver um sistema, independentemente da metodologia que se pretende usar. Isto justifica-se pelo facto de envolver os gestores da organização, usuários, e os analistas de sistemas. Esta fase procura determinar quais os objectivos do sistema e as fronteiras associadas a este.

Geralmente esta fase começa com a análise do sistema, e pode alongar se até à elaboração de um documento das especificações do sistema e inicia-se o plano de actividades para o desenvolvimento, depois de serem bem refinados os requisitos. Neste contexto é preciso notar que existe uma grande dependência entre a análise e a especificação por serem actividades que devem ser realizadas conjuntamente.

A análise sempre procura fazer o levantamento e observação dos elementos do domínio do sistema a ser desenvolvido, isto exige a identificação das pessoas envolvidas, as informações do domínio do sistema. As actividades e pessoas que não estão no domínio do sistema serão consideradas entidades externas.

A especificação notabiliza-se pelo facto de apresentar as propostas de soluções de como os problemas levantados na fase de análise serão resolvidos pelo sistema a ser desenvolvido, [32]. O objectivo principal da especificação é mostrar de maneira sistemática as propriedades funcionais que são importantes para resolver o problema em causa.

Um outro elemento importante na análise são os requisitos, cuja sua tarefa fundamental é identificar o que o sistema deverá fazer e definir os critérios de avaliação que serão usados para avaliar se o sistema responde o que foi previamente definido ou não. Desenvolver um sistema de Informação é uma tarefa complexa, como tal, é imperioso definir os requisitos que tornam o sistema útil.

Requisitos são os objectivos ou restrições estabelecidos por clientes e usuários do sistema que definem as diversas propriedades do sistema. Os requisitos de sistemas são, obviamente, aqueles dentre os requisitos de sistema que dizem respeito à propriedades do sistema, [33].
Normalmente encontramos dois tipos de requisitos a saber: Requisitos funcionais e não-funcionais.

3.3 - Requisitos funcionais

Este tipo de requisito mostra as diversas funções que os clientes e usuários pretendem que o software ofereça. Praticamente eles definem a funcionalidade que se deseja do sistema, isto é as operações a serem realizadas pelo sistema, pode ser através de comandos de usuários e ou

[32] Leite, 2000

ocorrência de eventos externos ou internos do sistema.

Ex:

- "O sistema deve possibilitar o cálculo de depósitos e levantamentos diários dos clientes no fim de cada dia";
- "O sistema deve ser capaz de emitir relatórios diários dos depósitos e levantamentos" , etc.

Sendo assim os requisitos funcionais devem determinar o que o sistema deve fazer e não se preocupar de como ele pode fazer, isto na tentativa de satisfazer àquilo que os usuários querem.

3.4 - Requisitos não – funcionais

Garantem a qualidade geral que um sistema deve oferecer, tais como o desempenho, manutenção, o uso do sistema custos e outras. Estes requisitos são descritos de maneira informal, maneira controversa e as vezes difíceis de validar.

Ex:

- " A base de dados deve ser protegida para acesso apenas de usuários autorizados";
- ." O sistema operativo deve ser *Unix*";
- "O tempo de desenvolvimento não deve ultrapassar seis meses", *etc.*

A necessidade de estabelecer os requisitos de forma precisa é muito importante porque a medida que a complexidade do sistema aumenta, os requisitos exercem uma influência uns sobre os outros.

3.5 - Engenharia de requisitos

Os diferentes tipos de relacionamentos e restrições que os requisitos exercem uns sobre os outros são difíceis de controlar, se considerarmos que algumas decisões ao serem tomadas podem afectar outras fases ou devem ser tomada nas seguintes. Desta maneira os requisitos precisam de ser

[33] Pressman, 1995

devidamente controlados durante toda a fase de ciclo de vida.

Dada a importância e complexidade desta fase originou o aparecimento de engenharia de requisitos, cujo objectivo é salientar que definir requisitos de sistemas é uma actividade fundamental e independentemente das outras fases de desenvolvimento de sistemas, [34]. Este princípio encontra a fundamentação nos próprios processos que devem ser planificados e geridos ao longo de todo o ciclo. Eis os objectivos da engenharia de requisitos:

- Investigação de objectivos, funções e restrições de uma aplicação de sistema;
- Especificar o sistema;
- Gerir a evolução do Sistema.

A engenharia de requisitos deve abarcar a documentação das funções, desempenho, interfaces externas e internos e padrões de qualidade do sistema.

3.6 - Benefícios da engenharia de requisitos

- Concordância entre analistas e usuários sobre o trabalho a ser feito e quais os critérios de aceitação do sistema;
- Uma base precisa para a estimativa dos recursos (Custo, pessoal, ferramentas e equipamentos);
- Melhoria na manutenção, usabilidade e qualidades do sistema;
- Atingir os objectivos previamente definidos.

3.7 - Técnicas de comunicação com os usuários

Um dos grandes objectivos da engenharia de requisitos é facilitar a comunicação entre analistas e usuários para dar lugar à captura de requisitos que pretendem ser modelados.

[34] Pressman, 1995

Eis as técnicas mais importantes para a comunicação com o usuário:

- Entrevistas;
- Observação *in loco*;
- Encontros;
- Leitura da Documentação existente.

A entrevista é normalmente a primeira técnica utilizada. Analistas entrevistam clientes para definir os objectivos gerais e restrições que o *software* deverá ter. A entrevista deve ser feita de forma objectiva visando obter o máximo de informações do cliente. Diversas secções de entrevistas podem ser marcadas.

Na observação *in loco* os analista devem estar enquadrados nas rotinas de trabalho da organização tentando entender e descrever as principais actividades que são realizadas. Na observação devem ser identificadas quais as actividades que podem ser informatizadas, quem são os potenciais usuários, quais são as tarefas que eles querem realizar com a ajuda do novo sistema, etc. A observação deve ser complementada com entrevistas específicas com os futuros usuários.

Os encontros são reuniões envolvendo os usuários e a equipa de desenvolvimento e têm como objectivo fazer o levantamento de informações, descrição do problema e metas para o futuro, podem ser realizados dentro ou fora da organização e normalmente duram alguns dias. Nestes encontros vão ser analisadas e sumarizadas e depois validadas pelos usuários e no fim é feito um relatório.

4. Ferramentas necessárias para comparação

Considera-se que a fase de análise é a mais difícil de ser automatizada e portanto deve-se contar com ferramentas imprescindíveis para o processo de comparação dos dois paradigmas. Sabe-se de antemão que existem muitas ferramentas, mas dada as limitações do próprio trabalho é necessário seleccionar parte dessa ferramenta adequada a fase em estudo.

Nas metodologias estruturadas de sistemas destacam-se as seguintes ferramentas: DC, DFD, DD, Entidades, DEA, Normalização, DE, DP, DED, Tabelas de Decisão e Árvore de Decisão.

Nas metodologias Orientadas ao Objecto (UML) temos: Actores, Uses Cases, Objectos, Classe, Diagrama *Use-Case*, Diagrama de Classes, Diagrama de Objectos, Diagrama de Estado, Diagrama de Sequência, Diagrama de Colaboração, Diagrama de Actividade, Diagrama de Componente, Diagrama de Execução.

4.1 - Análise das ferramentas de comparação

O Princípio de análise duma componente baseia-se principalmente no conhecimento do sistema por parte de quem faz a comparação. Este tipo de princípio consiste na observação cuidadosamente das características de uma componente, por dentro e em volta da sua ferramenta de análise. Consoante o exame que se faz analisam-se as características especificas dessas ferramentas para mais tarde adoptar-se um princípio de sua caracterização obedecendo a certas condições em sua volta.

4.2 - Ferramentas da AE

Para o comparação das metodologias serão analisado os DFD e DEA pelas razões apresentadas no capitulo I. O DFD é composto por entidades externas, processos, arquivos de dados e fluxos de dados. E por sua vez o DEA contém entidades e relações. Uma entidade externa que pode ser gerido pelo sistema, convertido num arquivo de dado. A entidade externa que não é gerível pelo sistema é que fornece Informação ao sistema, sendo assim, a entidade externa tirada do DEA tem o carácter de um arquivo de dado do DFD. Por isso, estas três componentes são conjugados em apenas uma única componente – entidade.

Análise das entidades

A entidade deve ser caracterizada pela sua denominação e pelo seu tipo de acordo com a origem, regularmente a entidade externa, arquivo de dados e entidade. Os primeiros dois tipos são retirados do DFD e o último tipo é retirado do DEA. Se for uma entidade externa, deve ser aquela entidade que seja controlada pelo sistema através do arquivo de dados correspondente, no caso de ser uma fonte externa, por não ter reflexo nos componentes; das ferramentas da AOO (UML), não interessa ser especificada. Para o âmbito do comparação, a designação de entidade será formada como um substantivo impróprio e é preferível no singular.

Análise dos processos

O processo é caracterizado pela sua denominação e pela respectiva entidade associada. No processo extraído do DFD observa-se qual é o arquivo de dado no DFD sobre o qual, directamente, executa a operação de escrita. Este arquivo de dado é que será a entidade associada. Se o processo estiver associado a mais de uma entidade, então, cada associação deve ser caracterizada em linhas separadas, isto é, o princípio fornece um determinado processo a uma entidade de cada vez. Se for o caso de muitos processos associados com a mesma entidade, a caracterização é feita numa linha de especificação. A denominação da entidade segue o princípio de análise das entidades, e a denominação do processo é formado por um verbo no infinitivo com o respectivo complemento directo que não se identifica completamente com a denominação da entidade ligada ao processo.

Análise de fluxos de dados

Um fluxo de dados deve ser caracterizado por denominação do processo que envia o fluxo (processo solicitante), da respectiva entidade associada na ocorrência do envio do fluxo de dado, do processo que recebe o fluxo (processo solicitado), da respectiva entidade associada na ocorrência da recepção do fluxo de dados e do conjunto dos dados que formam o fluxo. O conjunto dos dados deve referir-se apenas aqueles dados que são úteis, alvo

das operações no processo solicitado. Para o âmbito do processo de comparação, são analisados os fluxos de dados que mantém as relações entre processos que solicitam e os processos solicitados cujas entidades associadas sejam diferentes. A denominação dos processos e das entidades segue a análise dos processos.

Análise dos eventos

Os eventos constituem a parte fundamental de um sistema. Por isso, o primeiro passo na especificação de um sistema é identificar a que eventos do mundo exterior ele deverá responder. Isto, por si só, já ajuda a delimitar as fronteiras do problema de que estamos a tratar.
Ex: O cliente formula o pedido de material.

Análise dos atributos

Um atributo é qualquer propriedade de uma entidade que assume valores individuais num domínio que seja do interesse do sistema alvo. Os atributos são retirados das tabelas do DEA expandido, sem alteração da sua denominação. Todos os atributos de uma tabela são tomados numa única linha de entidade correspondente e essa tabela, cuja a designação segue a análise das entidades.

Análise das relações

Pode-se distinguir dois tipos de relações a saber: relação de conjunto e relação de associação. A caracterização duma relação deve conter a denominação da entidade de partida na leitura da relação no DEA, a designação da relação inerente à participação da entidade de partida na relação, os limites superior e inferior da participação da entidade de partida na relação, a denominação da entidade de chegada na leitura da relação, a denominação da relação referente à participação da entidade de chegada na relação e os limites superior e inferior. Esta caracterização serve perfeitamente para uma relação de associação entre as entidades.

Para a relação do conjunto que se subdivide em relação de dependência e de subconjunto a denominação da relação deverá ter como valores todo ou parte correspondendo a "pai" ou "filho" respectivamente, de acordo com a participação da entidade na relação de dependência, e como valores generalização ou especialização correspondendo a "pai" ou "filho" respectivamente, de acordo com a participação na relação de subconjuntos. Para esta última relação não tem importância a indicação dos limites superior e inferior.

Análise de DC

Este diagrama realça diversas características importantes do sistema. Como por exemplo: as pessoas, as organizações ou os sistemas com os quais o sistema em estudo se comunica (entidades usuárias do sistema). Identifica também os dados recebidos do mundo exterior, os dados produzidos pelo sistema e enviados ao mundo exterior, os dicionários de dados compartilhados pelo sistema e as entidades usuárias e, finalmente, as fronteiras entre o sistema e o resto do mundo.

Análise de DFD

Um aspecto importante num sistema de Informação são as funções. Todo o sistema de Informação armazena e transforma os dados de entrada em dados de saída. Estas transformações são realizadas pelas suas funções. Todo o modelo funcional de um sistema é formado por uma representação gráfica – uma rede de funções ou processos interligados – acompanhada de uma descrição de cada função e suas interfaces. A representação gráfica do modelo funcional é feita através de um diagrama chamado de DFD.

O DFD é um modelo lógico que mostra a interdependência de funções que compõem um sistema, apresentando fluxos de dados que serão manuseados no sistema a modelar. As componentes básicas de um DFD são: Fontes ou destinos de dados (entidades externas), depósitos de dados (dados retidos

para atender consulta ou necessidade posterior), fluxo de dados (dados que fluem entre processos, entidades e depósitos).

O DFD é uma ferramenta gráfica que pode ser muito valiosa durante a análise de requisitos de sistemas. O DFD descreve o fluxo de Informação sem uma representação explicita da lógica das funções.

No entanto, a notação básica usada no DFD não é suficiente para descrever os requisitos de um sistema. Por exemplo, o conteúdo de um fluxo de dados não é detalhado no diagrama; da mesma forma, o conteúdo de um depósito de dados não é exposto no DFD. Por isso, o DFD necessariamente deve ser complentamentado por um dicionário que fornece estes dados. Da mesma forma os processos não são detalhados no DFD, mas sim na descrição dos processos.

Análise de DEA

Modelagem de dados é um método de análise de sistemas que procura especificar, a partir do factos relevantes que estejam associados ao domínio do conhecimento analisado, a perspectiva dos dados, permitindo organizá-los em estruturas bem definidas; estabelecer as regras de dependência entre eles, produzindo um modelo expresso por uma representação, ao mesmo tempo, descritiva e diagramática. O DEA corresponde aos arquivos de dados extraídos no DFD.

Os dados associados a um sistema de informação são pesquisados; os tipos de dados e suas propriedades são definidas e o modelo de dados é construído, definindo-se também os relacionamentos entre os tipos de dados. Quando isto é feito correctamente, com forte envolvimento dos usuários, os dados necessários ao contexto analisado são encontrados e definidos.

O modelo é normalizado, isto é, as redundâncias são removidas, os relacionamentos necessário são estabelecidos, viabilizando, assim, um projecto de base de dados livre de anomalias de actualização. Além disso,

geralmente com a definição dos dados, podem ser definidos os processos para criar, modificar e excluir ocorrências de dados.

Coloca-se pergunta chave que o analista deve responder. "Quais são as entidades que são necessárias para satisfazer o objectivo do sistema e de que precisamos de armazenar dados a seu respeito"?

O relacionamento é uma interdependência de duplo sentido entre duas entidades. Uma entidade "A" está relacionada com uma entidade "B", se e somente se a chave de uma delas residir na tupla (linha) de outra como estrangeira.

4.3 - Ferramentas da AOO (UML)

As ferramentas da AOO (*UML*) são constituídos por visão "use case", diagrama de *use case*, diagrama de classes e diagramas de objectos. O diagrama de use case é composto por use case, actores externos e o sistema modelado. O diagrama de classes é constituído por classes, as relações que se estabelecem entre as classes bem como a dependência entre as classes. O diagrama de objectos é composto por instâncias das classes modeladas.

Análise da visão *use case*

Esta visão descreve de uma maneira geral o funcionamento do sistema executado pelos actores externos ao sistema (usuários). A visão *use case* é central, já que o seu conteúdo constitui a base do desenvolvimento das outras visões do sistema.

Análise de actores

Os actores representam o papel de uma entidade externa ao sistema como um usuário, um *hardware*, ou outro sistema que interage com o sistema modelado, um fornecedor de Informação ou ainda o receptor de Informação.

Em suma, um actor representa qualquer entidade que interage com o sistema. Os actores iniciam a comunicação com o sistema através dos *uses cases*. Um actor é conectado a um ou mais *use-cases* através de associações. Os actores podem possuir relacionamentos de generalização que definem um comportamento comum de herança. Por outro lado, um actor modela qualquer coisa que precise trocar Informação com o sistema. Um ponto de partida para identificar os actores é verificar por que é que o sistema deve ser desenvolvido.

Análise de *use case*

As use cases representam as funcionalidades das *use cases* que lidam com o armazenamento e manipulação de Informação. São mapeadas directamente para classes e objectos na análise de modelos e mostram também, o funcionamento que é directamente dependente do ambiente do sistema, isto é as funções requeridas pelo interface do sistema.

O *use cases* em colaborações é muito importante, onde estas são a descrição de um contexto mostrando classes/objectos, seu relacionamento e sua interacção exemplificando como as classes/objectos interagem para executar uma actividade especifica no sistema. Uma colaboração é descrita por diagramas de actividades e colaboração.

Quando um use case é implementado, a responsabilidade de cada passo da execução deve ser associada às classes que participam na colaboração, especificando as operações necessárias dentro destas classes juntamente com a definição de como elas vão interagir.

Quando usar os *use case*?

Os *use case* são muito úteis para auxiliar na análise de requisitos do sistema a ser desenvolvido e para facilitar o planeamento e controlo dos respectivos iterativos.
Normalmente começa-se com um conjunto não exaustivo de *use cases* e vai-se identificando outros adicionais. Cada *use case* corresponde um a potencial

requisito. Importa ter isso em mente porque um requisito que não se consegue estabelecer é um requisito que não se pode considerar no sistema a ser desenvolvido.

Alguns analistas preferem criar *uses cases* de grande granularidade. Outros preferem criar *uses cases* pequenos. Esta solução é geralmente melhor, mas corre-se o risco de obter um número muito elevado e difícil de gerir.

Análise dos cenários

Um cenário é uma instância de um caso de uso. Descreve a sequência de eventos que ocorrem durante a execução do sistema. Os cenários mostram as principais interacções, formatos externos de exibição e intercâmbio de interacções, [35].

Cada caso de uso possui os cenários primários e secundários. Os primários apresentam a forma de como o sistema deveria funcionar normalmente. Enquanto que os secundários apresentam as excepções dos cenários primários.
Os cenários devem ser elaborados em linguagem simples e com os usuários, para se obter informações precisas sobre o comportamento do caso de uso (diagrama de sequência).

O número de cenários a serem elaborados depende da complexidade do sistema. Geralmente são elaborados os cenários necessários à compreensão do sistema em desenvolvimento, os de alto risco e os de especial interesse.

Análise das classes

Uma classe é a descrição de um tipo de objecto. Todos os objectos são exemplares das classes, onde a classe descreve as propriedades e comportamentos dos mesmos. As classes são utilizadas para classificar os objectos identificados no mundo real, sendo assim, elas devem ser retiradas

[35] andrade, at all, 1998

do domínio do problema e serem nomeadas pelo que elas representam no sistema. Esta actividade deve ser feita por alguém com bastante conhecimento do domínio do problema.

As classes encontram-se inseridas no diagrama de Classes onde elas mostram a estrutura estática de um sistema. Uma classe tem um nome, atributos e operações que podem ser executadas sobre a ela.

Classe

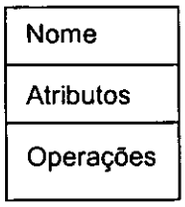

Fig. 4.1

Onde:

Nome: irá conter o nome da classe modelada;
Atributos (estrutura): irá conter a relação de atributos que a classe possui na sua estrutura interna;
Operações (Comportamento): irá conter os métodos de manipulação de dados e de comunicação de uma classe com outras.

Análise dos objectos

Um objecto existe no mundo real e é um elemento que se pode manipular, acompanhar seu comportamento, criar, destruir, etc. Existem também objectos que não se encontram no mundo real, mas que podem ser derivados da estrutura de outros objectos do mundo real.
EX: Comportamento de sistema informático.

Os objectos e as instâncias das classes são mostradas no diagrama de objectos como se fossem o perfil do sistema num certo momento da sua execução. Um objecto deve ser caracterizado pela sua designação e tipo.

Análise dos atributos

No caso mais geral, a notação para um atributo específico é o nome e o tipo. Os atributos têm sempre um valor único de cada vez. Os diagramas não indicam se um atributo é obrigatório ou opcional.
Um atributo é caracterizado pelo objecto que o encapsula. A denominação de um serviço é correspondente ao objecto sem a sua alteração.

Análise de dados e comportamentos

Ao contrário das técnicas estruturadas, em AOO (UML), os dados de um objecto estão ligados ao seu comportamento e rotinas que manipulam estes dados directamente, e vice-versa. Desta forma é possível manter um controle mais aprimorado das operações do sistema e objectos sobre os dados. Cada objecto têm responsabilidade directa na manipulação dos seus dados.

Análise de compartilhamento e reutilização

A AOO (UML) promove o compartilhamento em diversos níveis. A herança da estrutura de dados e do seu comportamento permite que a estrutura comum seja compartilhada por diversas subclasses semelhantes sem redudâncias. O compartilhamento de código, com utilização de herança, é uma das principais armas da AOO (UML). Sendo assim a AOO permite também a reutilização de informações, internamente num sistema, bem como nos sistemas e projectos futuros.

Análise das operações

As operações correspondem aos métodos de manipulação de dados e de comunicação de uma classe com as outras no sistema. Isto é, uma operação refere-se à unidade de processamento que pode ser solicitada. O procedimento é implementado com um método que é a especificação de como a operação é executada.

È útil distinguir entre as operações que alteram, e as que não alteram, o estado de uma classe. Uma interrogação é uma operação que obtém o valor de uma classe sem alterar o seu estado observável. As operações que alteram o estado observável de uma classe são denominadas modificadoras. As interrogações podem ser executadas por qualquer ordem, mas os modificadores exigem cuidados com a sua sequenciação.

Análise das relações e associações

As relações e associações indicam as ligações de um objecto com outro. Uma relação deve ser caracterizada pela denominação de partida na leitura de uma relação nos diagramas de classes e de objectos. A relação é relevante na participação da classe ou objecto de partida da relação. Deve possuir os seus limites inferior e superior da participação do objecto de partida na relação, pela denominação do objecto de chegada na relação bem como os seus limites superiores e inferiores designadas nessa relação.

No relacionamento entre as classes usa-se um losango com traços nas extremidades que ligam as classes que se relacionam.
A cardinalidade mapeia as restrições no relacionamento. Um asterisco (*) determina zero ou mais, um sinal de adição (+) determina um ou mais, uma linha sem cardinalidade tem o mesmo significado que o (*), um intervalo é determinado da seguinte maneira: (2..4) e uma cardinalidade fixa é determinada pelo número explicito desta cardinalidade, por exemplo: 2,5.

Esta visão é adequada para uma relação de associação entre classes e, para a relação de estrutura que se subdivide em relação de escala e de herança, são chamados de : "todo", "parte" para estrutura por escala e "generalização" e "especialização" para a relação de estrutura por herança.

Análise das restrições

Os conceitos de associação, de atributo ou de generalização são, afinal, formas de especificar restrições. No entanto, os conceitos chaves dos diagramas de classes não permitem exprimir todas as restrições.

Há restrições que têm que ser expressas de forma explícita porque não caem em nenhuma das categorias previstas nos diagramas de classes. Usam-se Chavetas para exprimir o que deve ser restringido.

Análise das mensagens

A mensagem informa a um objecto o que ele deve fazer. A operação é nomeada na mensagem, mas o que é feito está na verdade oculto (encapsulado) no objecto que recebe a mensagem.

Sendo assim, para fazer com que um objecto realize alguma acção, é necessário enviar uma solicitação. Esta faz com que uma operação seja invocada. A operação executa o método apropriado e, opcionalmente, retorna a resposta. A mensagem que constitui a solicitação contém o nome do objecto, o nome da operação, e às vezes, um grupo de parâmetros.

No diagrama de colaboração, mostra-se a troca de mensagens entre objectos. A interacção de mensagens é mostrada em ambos os diagramas. Sua ênfase do diagrama for o decorrer do tempo, é melhor escolher o diagrama de sequência, mas se a ênfase for o contexto do sistema, é melhor dar prioridade ao diagrama de colaboração.

Tanto o diagrama de classes como o de objectos possuem mensagens que são implicitamente discriminadas, estas mensagens carregam parâmetros para um serviço num determinado objecto partindo de fontes externas ou

mesmo dos serviços do objecto ou dos registos desse objecto por via dos seus atributos.

Para a localização destas mensagens em cada objecto receptor, tira-se os argumentos da operação solicitada e procura-se agrupá-los, primeiro, em função dos parâmetros associados aos tais argumentos, segundo, em função dos instantes em que os parâmetros forem passados para o serviço, por meio de argumentos.

O conjunto dos parâmetros para o serviço, por meio de argumentos suficientes e capazes de accionar completamente o serviço solicitado e examina-se a resposta deste serviço. A resposta será um conjunto dos parâmetros passados para um certo destino ou fonte externa .

Análise de diagrama de use-*case*

A modelagem de um diagrama *use-case* é uma técnica usada para descrever e definir os requisitos funcionais de um sistema. Eles são escritos em termos de actores externos *use-cases* e o sistema a ser desenvolvido. Os actores podem também ser *hardware* ou outro sistema que interage com o sistema modelado.

Os *use cases* representam individualmente os requisitos funcionais. Os actores iniciam a comunicação com o sistema através dos *use-cases*, onde estes representam uma sequência de acções executadas pelo sistema e recebem do actor que lhes utiliza dados tangíveis de um tipo ou formato já conhecido. Tudo isso é definido juntamente com o *use-case* através do texto de documentação.

Os actores e *use-cases* são classes. Um actor é conectado a um ou mais *use-case* através de associações, e tanto actores quanto *use-case* podem possuir relacionamentos de generalização que definem um comportamento comum de herança em superclasses especializadas em subclasses.

Análise do diagrama de classes

O Diagrama de classes mostra a estrutura estática das classes de um sistema onde estas representam as "coisas" que são controladas pelo sistema modelado. As classes relacionam-se com outras através de diversas maneiras: Associações, dependência (uma classe depende ou usa outra classe), especialização (uma classe é especialização de outra classe), ou classes agrupadas por características similares. Todos estes relacionamentos são mostrados no diagrama de classe juntamente com as suas estruturas internas que são atributos e operações. Este diagrama é considerado estático já que a sua estrutura descrita é sempre válido em qualquer fase de ciclo de vida do sistema.

Uma classe num diagrama pode ser directamente implementada utilizando-se uma linguagem de programação orientada a objectos que tenham suporte directo para construção de classes. Para criar um diagrama de classes, as classes tem que estar identificadas, descritas e relacionadas entre si.

Análise do digrama de objectos

O diagrama de objectos é uma variação do diagrama de classes e utiliza quase a mesma notação. A diferença é que o diagrama de objectos mostra os objectos que forma instanciados das classes. O diagrama de objectos é como se fosse o perfil do sistema num certo momento da sua execução. A mesma notação do diagrama de classes é utilizada com duas execuções: Os objectos são escritos com os seus nomes sublinhados e todas as instâncias num relacionamento são mostradas. Os diagramas de objectos não são tão importantes como os diagramas de classes, mas eles são muito úteis para exemplificar diagramas complexos de classes ajudando muito em sua compreensão. Diagramas de objectos também são usados como parte dos diagramas de colaboração, onde a colaboração dinâmica entre os objectos do sistema é mostrada na fig. 4.2.

Diagrama de objectos

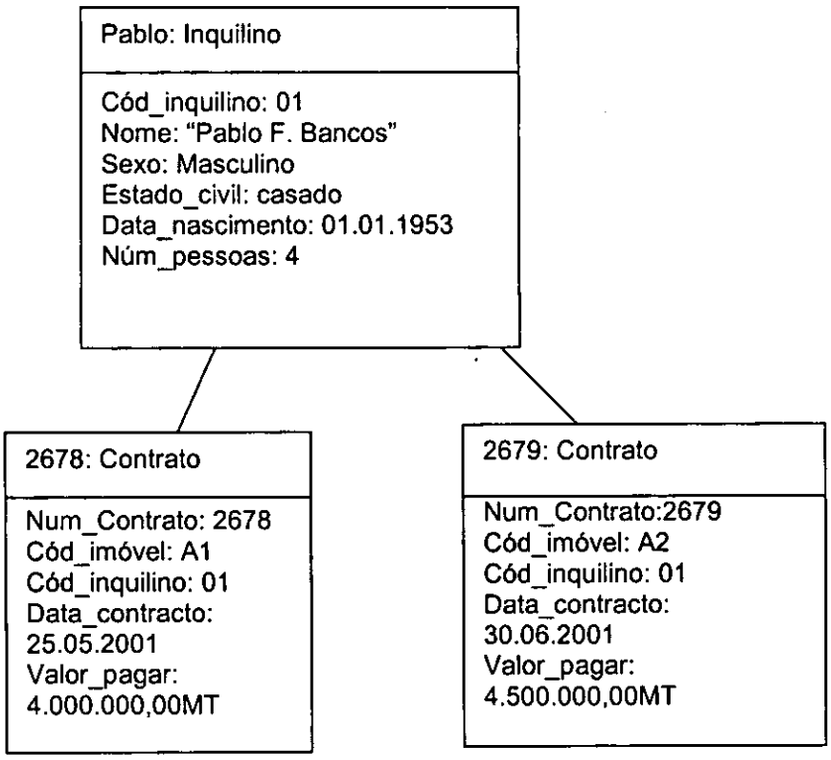

Fig. 4..2

Análise de digrama de colaboração

Um diagrama de colaboração mostra de maneira semelhante ao diagrama de sequência, a colaboração entre os objectos. Normalmente pode-se escolher entre utilizar o diagrama de colaboração ou diagrama de sequência.

Análise de diagrama de actividade

Os diagramas de actividades capturam acções e seus resultados. Eles focam o trabalho executado na implementação de uma operação, e suas actividades numa instância de um objecto. O diagrama de actividade é uma variação do diagrama de estado e possui um objecto um pouco diferente do diagrama de estado, que é o de capturar acções (trabalho e actividades que serão executados) e os seus resultados em termos das mudanças de estado dos objectos.

Um diagrama de actividade é uma maneira alternativa de se mostrar interacções, com a possibilidade de expressar como as acções são executadas, o que elas fazem (mudança de estado dos objectos), quando elas são executadas (Sequência das acções) e onde elas acontecem. Um diagrama de actividade funciona como um diagrama de fluxo de controle.

Análise de diagrama de componentes

O diagrama de componentes e o de execução são diagramas que mostram o sistema por um lado funcional, expondo as relações entre seus componentes e a organização dos seus módulos durante a execução.

O diagrama de componentes descreve as componentes de *software* e suas dependências. Representando a estrutura do código gerado. As componentes são a implementação na arquitectura física dos conceitos e da funcionalidade definidos na arquitectura lógica (classes, objectos e seus relacionamentos). Eles são tipicamente os ficheiros implementados no ambiente de desenvolvimento.

Uma componente, é mostrada em AOO (UML), em forma de um rectângulo com uma elipse e dois rectângulos menores do seu lado esquerdo. O nome do componente é escrito abaixo ou dentro do seu símbolo.

Análise de diagrama de execução

O diagrama de execução mostra a arquitectura física do *hardware* e do *software* no sistema. Pode mostrar os actuais computadores e periféricos, juntamente com as conexões que eles estabelecem entre si e pode mostrar também os tipos de conexões entre esses computadores e periféricos. É a ultima descrição física da topologia do sistema, descrevendo a estrutura de *hardware* e *software* que executam em cada unidade.

4.4 - Aspectos comuns das duas abordagens

As metodologias utilizadas para a análise, planeamento, desenvolvimento e implementação de sistemas de informações deveriam, tendo em vista esta realidade, possibilitar uma maior flexibilidade que permitisse à organização adaptam-se facilmente às mudanças.

É preciso realçar que o objectivo de um sistema de informação deveria ser o de resolver os problemas da organização, capitalizando em oportunidades de negócios, redução de custos, obtenção de vantagens estratégicas, entre outros. No entanto, importa referir que, primordialmente, o sistema de informação deve ir ao encontro das expectativas e objectivos dos usuários finais, pertencentes a organização. Neste sentido, importa chamar a atenção aos chamados aspectos humanos inerentes a todo o processo de análise, planeamento, desenvolvimento e implementação de sistemas de informações.

Durante a fase de análise, nas duas metodologias várias actividades são desenvolvidas, só que é do conhecimento geral a dificuldade de se identificar a necessidade de dados para um sistema. Identificar entidades ou classes de um sistema pode ser complicado, e deve ser feito por *experts* do domínio do problema do sistema que está a ser modelado. As entidades ou classes devem ser retiradas do domínio do problema e serem classificadas pelo que elas representam no sistema.

Não se encontra qualquer indicador de como identificar e especificar as necessidades do usuário. Alguma referência pode ser encontrada em vários livros antigos que tratam da eliminação de requisitos e que indicam as entrevistas, questionários, observação *in loco*; encontros, leitura da documentação existente e outros.

Aproveitando a difusão do uso de bancos de dado relacionais, o modelo de dados normalmente utilizado nos Sistemas analisados, através da

metodologia de análise estruturada, é criado apartir de processo de normalização, que procura transformar visões de dados em assuntos singulares. A normalização é um processo formal que examina os atributos das entidades com o intuito de minimizar a redundância.

Em relação a este aspecto não se encontram muitas orientações para a AOO (UML). Isto se deve, principalmente, ao facto de que as bases de dados utilizadas continuam sendo relacionais. As BDOO ainda são objecto de estudos académicos, por isso pouca coisa existe a nível comercial. É fundamental apresentar uma interessante e apropriada "normalização do modelo de classes", bastante similar ao processo utilizado na análise estruturada. A normalização do modelo apresenta a proposta seguinte:

- A Primeira forma normal (remoção de grupos repetitivos) é obtida através dos seguintes passos: Verificar se a classe analisada possui atributos que são dependentes de atributos dentro da classe [A] analisada; destacar os atributos repetitivos e suas respectivas operações, criando uma nova classe [B] que observará esses itens; estabelecer a associação de agregação regular e multiplicidade ou número de ocorrências entre as classes [A] e [B].

- A segunda forma normal (remoção de dependências parciais) é obtida através dos seguintes passos: verificar se a classe analisada possui atributos que são dependentes de atributos que fazem parte do identificador composto (e não de todo identificador); destacar os atributos com dependência parcial e suas respectivas operações, incorporando-os a uma nova classe, que terá como identificador o atributo que possuía as dependências.

- A terceira forma normal (remoção de dependências transitivas) é obtida através dos seguintes passos: verificar se a classe analisada possui atributos que são dependentes de outros atributos nela contidos; destacar os atributos com dependência transitiva e suas respectivas

operações, incorporando-os a uma nova classe, que terá como identificador o atributo que possuía as dependências; eliminar os atributos obtidos por cálculo apartir de outros atributos da classe analisada.

- Para que seja verificada a quarta forma normal (remoção de dependências multivaloradas) deve haver pelo menos três classes envolvidas gerando uma classe de associação (associação ternária). Para que a quarta forma normal seja aplicada, o relacionamento das classes A, B e C deverá ter como hipótese: a) A e B são relacionadas; b) A e C são relacionadas; c) mas B e C são independentes. Portanto, aplica-se a quarta forma normal para remover a dependência multivalorada entre B e C transformando uma associação ternária em duas associações binárias associativas.

Como vimos a trás nas ferramentas de comparação, em ambas as abordagens temos os atributos, os relacionamentos, as associações bem como o grau de uma associação, assim como a herança. Na análise estruturada encontramos isto no diagrama de entidade e associações, já em UML estes factos são verificados no diagrama de classes e de objectos [36].

Ambas metodologias são muito semelhantes em várias formas, embora cada uma encara o sistema de um ponto de nota diferente. Alguns autores argumentam que já que a maioria dos instrumentos e técnicas usadas nas duas metodologias são comuns, elas próprias devem ter um grau de semelhança. Por outro lado as duas abordagens tiveram as suas raízes nas técnicas estruturadas de programação.

Muitos autores, [37]. Destacam a importância do plano da mudança gerada na organização com desenvolvimento e implementação de um sistema de Informação.

[36] Bertolini at all, 1998

Para estudar as metodologias apresentadas procurou-se definir áreas que necessitam de ser avaliadas numa metodologia. As cinco áreas propostas basearem-se:

5- Estudo comparativo

5.1 - Modelo funcional: como identificar as funções

O modelo funcional visa descrever todos os procedimentos (não funções) necessários para que o sistema possa alcançar os seus objectivos. Na análise estruturada não existem regras claras de como identificar as funções necessárias. As poucas orientações, [38], indicam que deve existir funções para a manutenção das informações pertinentes no sistema, bem como, funções para suprir as necessidades de respostas, através da emissão de relatórios e documentos.

Encontram-se também alguns indicadores para auxiliar nesta identificação e que se reportam ao tempo em que os procedimentos devem correr, como por exemplo, identificar procedimentos que devem ser realizados diariamente, mensalmente, anualmente ou esporadicamente.

Por exemplo: Para o sistema de controle de aluguer de imóveis, algumas funções seriam: Manter os dados de imóveis, efectuar o contrato, manter as informações sobre as pessoas, identificar os inquilinos com pagamentos atrasados, controlar pagamentos, emitir os relatórios mensais e anuais de pagamentos.

A AOO (*UML)* já possui um certo formalismo para resolver esta questão – o diagrama de *use case* – que mostra um número de actores externos e sua conexão com o sistema. Um diagrama *use case* descreve um conjunto de sequências, no qual cada sequência representa a interacção de funções com o próprio sistema, (seus actores) e suas abstracções principais.

[37] Bertolini at all, 1998

A modelagem de um diagrama de *use case* é uma técnica usada para descrever e definir os requisitos funcionais de um sistema. *Os use case* são descritos somente como são vistos externamente pelo actor e não descrevem como a funcionalidade é realizada dentro do sistema. Um diagrama de use case é um diagrama que mostra um conjunto de *use case*, actores e seus relacionamentos de dependência, generalização e associação. A figura 5.1 mostra um diagrama de *use case* para o sistema de controle de aluguer de imóveis, [39].

Caso em estudo

Agência imobiliária

Os proprietários disponibilizam os seus imóveis para serem alugados – é feito um contrato definindo as regras para esta prestação de serviços por parte da imobiliária.
A agência dedica-se ao aluguer de imóveis a inquilinos. O proprietário do imóvel dirige-se à agência a fim de pôr o seu imóvel em aluguer. A agência celebra o contrato com os inquilinos que pretendem as casas.

Os inquilinos que pretendem alugar o imóvel, dirigem-se à agência solicitando um imóvel para aluguer. Se se chegar à conclusão de que o inquilino está em condições de alugar o, celebra-se o contrato de aluguer com o inquilino. Sobre o este pretende-se registar o cod_inquilino, nome, estado civil, número de pessoas que vão morar, data de nascimento e sexo.

Sobre o contrato a agência regista o número_contrato, cod_imovel, cod_inquilino, data_contrato e valor_apagar. Sobre os imóveis é preciso saber o cod_imóvel, descrição, cod_cidade, cod_proprietário, tipo , endereço, cidade, estado de conservação e proprietário do imóvel. Como se sabe os imóveis localizam-se numa determinada cidade e podem ser moradias ou apartamentos. Cada apartamento ou moradia tem um cod_imovel.

[38] bertolini at all, 1998

Como se sabe, cada imóvel pertence a um determinado proprietário, sobre o proprietário a agência regista o cod_proprietário, cod_imovel, nome, estado civil, local de trabalho, telefone e quantia a receber por mês e por ano. Os proprietários podem ser pessoas singulares ou empresas. Para cada apartamento ou vivenda regista-se o cod_Imóvel e cod_proprietário.
Num determinado dia do mês a imobiliária repassa os valores devidos aos proprietários. Anualmente, a imobiliária deve fornecer aos proprietários uma declaração do que eles receberam relativo a cada imóvel para fins de declaração de imposto de renda.

Diagrama de *use case*

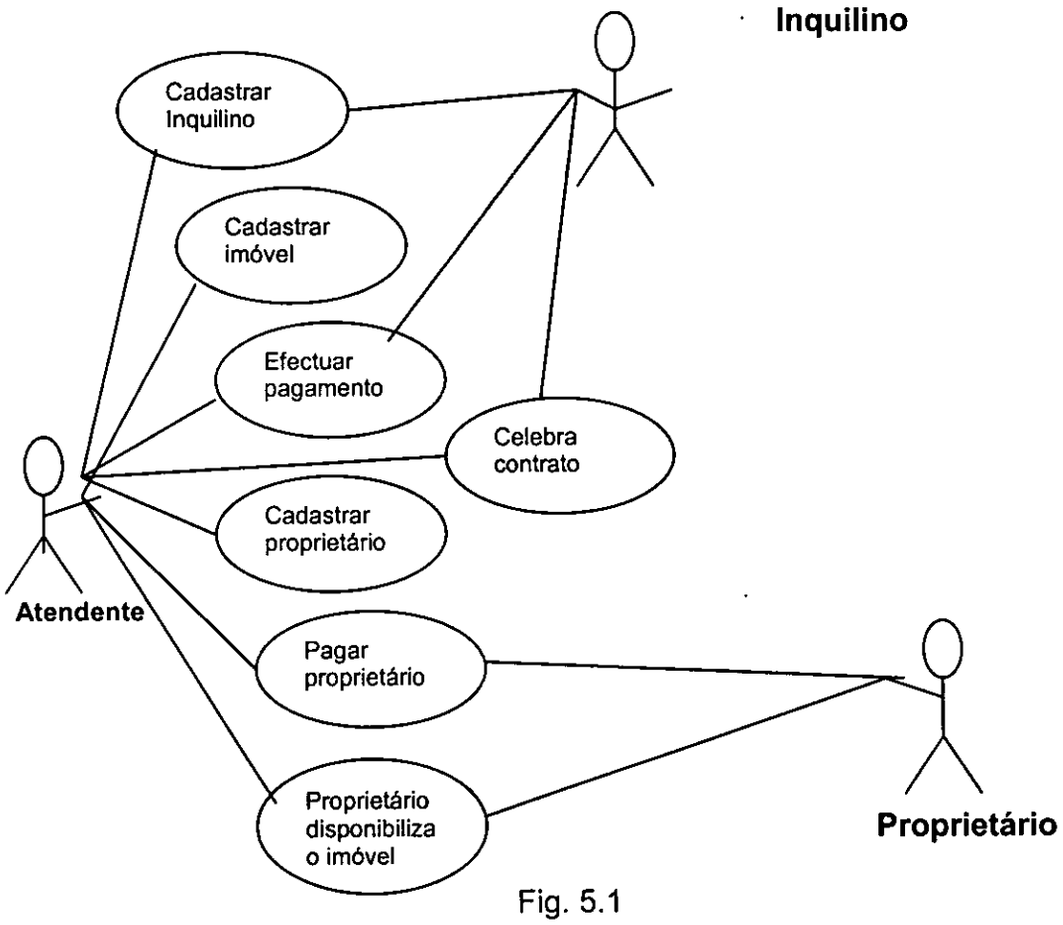

Fig. 5.1

[39] Bertolini at all, 1998

5.2 - Modelo funcional: como representar

A representação das funções, na análise estruturada, é feita através do DFD, que é uma ferramenta simples mas poderosa para modelagem das funções de um sistema. As componentes de um DFD típico são: O processo, o fluxo, o arquivo e a entidade externa. A figura 5.2 mostra uma parte do DFD que modela as funções identificadas anteriormente para o sistema da Agência Imobiliária.

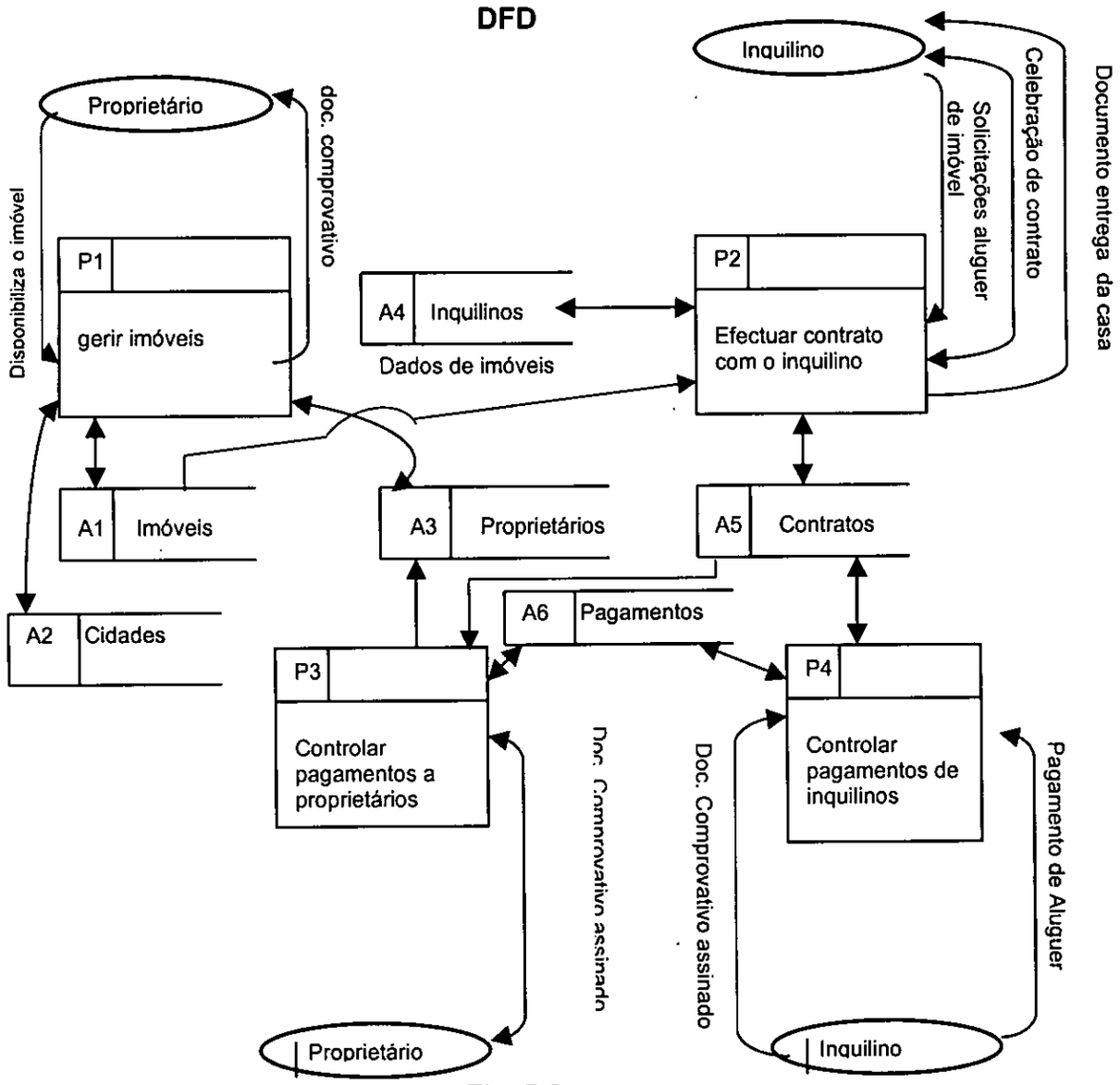

Fig. 5.2

Numa visão mais ampla, o diagrama de *use case* da AOO (*UML*) seria a representação do modelo funcional do sistema, entretanto ele mostra a realização das use case e as possíveis sequências de interacção que ocorrem neles. Quando se deseja visualizar o comportamento de vários objectos dentro de um único use case, a partir das mensagens que são passadas entre eles, é necessário usar um tipo de diagrama de interacção, chamado de diagrama de sequência. Um diagrama de sequência mostra as interacções de objectos organizados numa sequência de tempo e de mensagens trocadas.

5.3 - Modelo de dados: como representar

Para a representação do modelo de dados utilizado na AE e na AOO (*UML*), far-se-á o uso de uma modelagem simples, baseada no funcionamento de uma agência Imobiliária que tem sob sua responsabilidade a locação de imóveis de terceiros, conforme a descrição do caso em estudo, que anteriormente foi apresentada.

Da posse destas informações do caso em estudo, foi realizado o processo de normalização tendo sido criado o modelo de dados a seguir apresentado:

inquilino
Cod_Inquilino
Nome
Estado_Civil
Data_nascimento
Número_pessoas
Sexo

Contrato
Numero_contrato
Cod_imovel
Cod_inquilino
Data_Contrato
Valor_Apagar

Moradia
Mun_Mor
Cod_imóvel

Imóvel
Cod_imóvel
Cod_Proprietário
Descrição
Cod_Cidade
Tipo
Endereço
Estado_Conservação

Apartamento
Num_Apar
Cod_Imóvel

Cidade
Cod_Cidade
Nome_Cidade

Proprietário	**Singulares**
Cod_proprietário	Estado_Civil Cod_proprietário Local_trabalho
Nome	
Telefone	**Empresas**
Quantia_Receber	Sigla
Tipo	Cod_proprietário

Na AE esta modelagem é representada num Modelo Entidade-Relacionamento, conforme a figura abaixo. As entidades estão representadas por rectângulos e os relacionamentos fazem o uso das linhas que representam as referências lógicas existentes entre diversas entidades e, associadas a elas, apresentam-se marcações de opcionalidade/obrigatoriedade, marcação de cardinalidade e marcações de indicação de regra de exclusão.

DEA

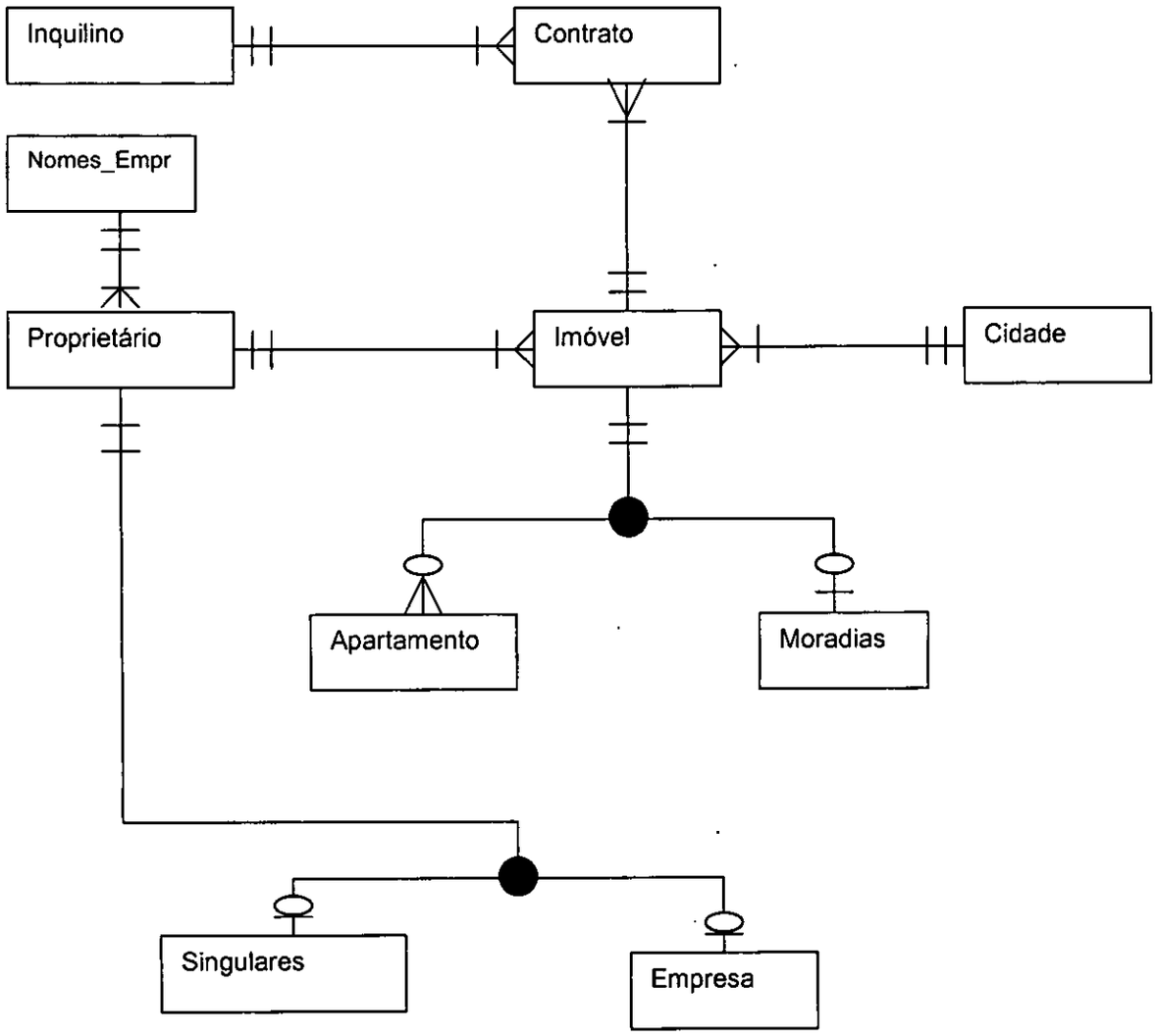

Fig. 5.3

Para se fazer a representação deste modelo usando AOO (UML) utiliza-se o Diagrama de Classes, A figura 4.5 que se segue mostra a estrutura estática das classes no sistema. O diagrama é considerado estático, pois a estrutura descrita é sempre válida, em qualquer ponto do ciclo de vida do sistema.

Em AOO (*UML*) as classes são representadas por um rectângulo dividido em três compartimentos: o compartimento de nome, que contém apenas o nome da classe modelada; o de atributos, que possui a relação de atributos que a classe possuía na sua estrutura interna e o compartimento de operações, que serão os métodos de manipulação de dados e de comunicação entre as classes do sistema. Os relacionamentos entre as classes são desenhados

como caminhos conectando os rectângulos representativos das classes. Existem diferentes tipos de relacionamentos.

Diagrama de classes

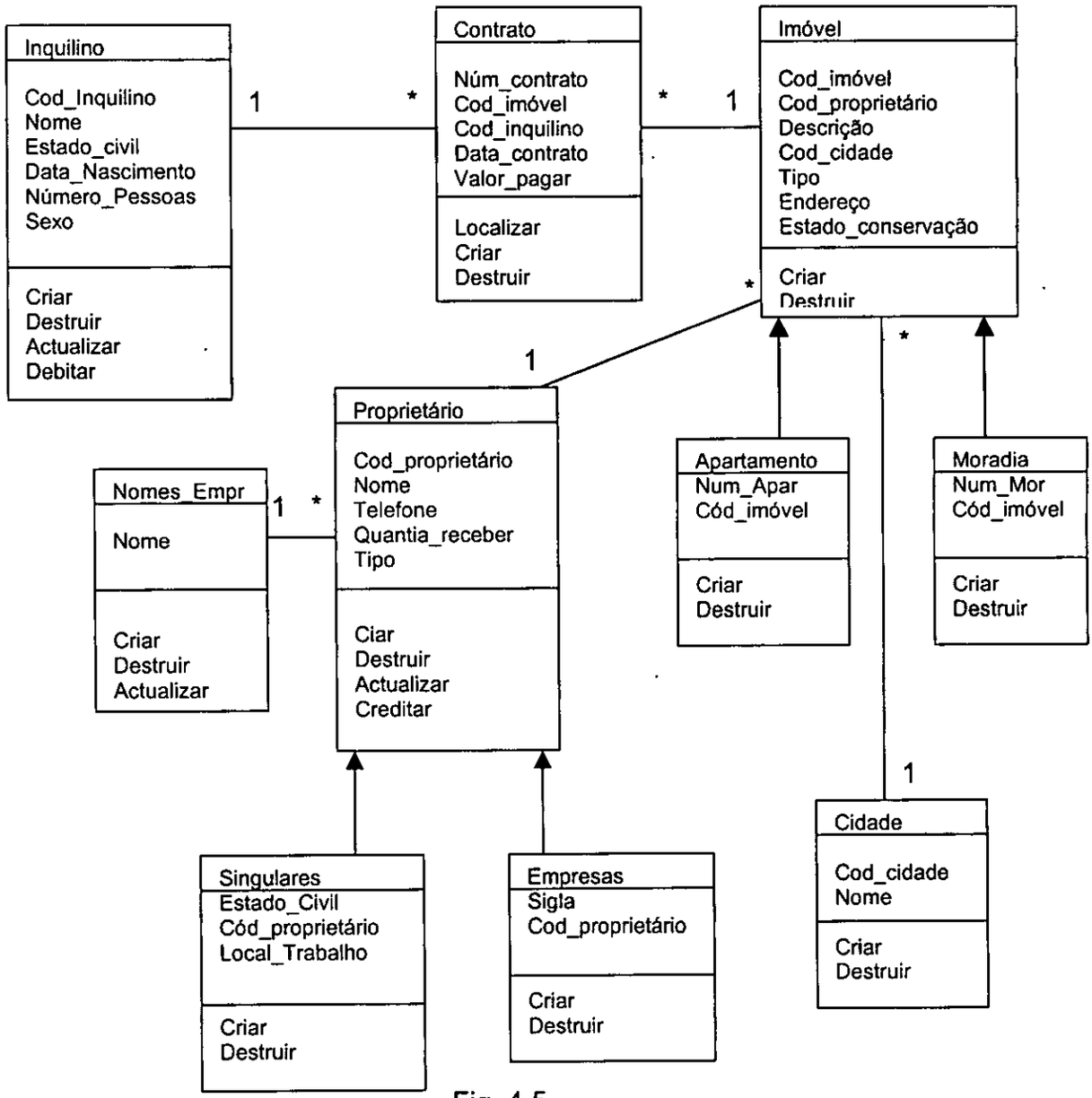

Fig. 4.5

Em AOO (UML) também é possível fazer-se o uso de um diagrama de objectos, que é uma variante detalhado do diagrama de classes e que usa uma notação idêntica. A diferença entre os dois é que o diagrama de objectos mostra as instâncias das classes, ao invés da classe. A mesma notação de diagrama de classes é usada, com duas excepções: Os objectos são escritos

com o seu nome sublinhado e todas as instâncias de um relacionamento são mostradas, [40].

6 – Discussão

No capítulo 2 foi definido o conceito de metodologia para o desenvolvimento de sistemas de informação como "um conjunto recomendado de filosofias, fases, procedimentos, técnicas, regras, ferramentas, documentação, gestão e treinamento para a actividade de desenvolvimento de um sistema de Informação", [41].

Entende-se por filosofia, neste contexto, como sendo o conjunto de princípios que norteiam uma metodologia. De modo geral pode-se afirmar que todas as metodologias são baseadas numa filosofia comum que sugerem que as metodologias devem aprimorar o desenvolvimento de sistemas de informação, [42].

Um assunto importante no desenvolvimento de sistemas de informação é a análise do problema e a análise e especificação de requisitos produzidos apartir da Informação fornecida pelos utilizadores. Durante a análise do problema um grande volume de informação relacionada com as perspectivas de diferentes pessoas deve ser organizada.

Nesta ordem, a facilidade de comunicação entre os analistas do sistema e os utilizadores do sistema é um assunto importante no acto da análise do problema e requisitos. Considera-se primordial o uso de um número de questões para discutir e contrastar as duas metodologias, [43].

A tabela 1 que se segue mostra as diferenças de nomenclatura existentes entre as duas abordagens. O conceito de cada componente é o mesmo em ambas as abordagens. O que justifica a mudança na nomenclatura em AOO

[40] Bertolini at all, 1998
[41] Roque, 1998
[42] Roque, 1998
[43] Bertolini at all, 1998

através da *UML*, é o facto deste paradigma tentar aproximar-se muito de uma visão do mundo real na modelagem de sistemas. Sendo assim, são apresentados alguns critérios para discutir e contrastar as metodologias, [44].

AE	AOO
Entidade	Classe
Atributo	Propriedade
Chave Primária	Identidade
Relacionamento	Associação

Tabela 1: Diferença de Nomenclatura

a) **Facilitar a comunicação** – A AE focaliza no processamento de dados. Este modelo encaixa com muitas aplicações de negócio. Muitos processos de negócios podem ser descritos com exactidão em termos de dados e processamento. Se a AE é aplicado através do uso da linguagem e terminologia do utilizador, deverá ser um instrumento de comunicação útil com os utilizadores. A AOO (UML) facilita a comunicação durante a análise porque usa classes e objectos que circundam o problema que está descrito na linguagem do utilizador;

b) **Providência dos meios de definir as fronteiras do sistema**. – O DFD começa com o nível denominado DC que define a visão do sistema no nível mais alto, focalizando a interacção entre o sistema e o mundo externo. Isto origina uma boa definição da fronteira do sistema. A AOO *(UML)* está enfraquecido relativamente a este assunto, porque o método usado para seleccionar classes e objectos é o sistema ou o ambiente, e contém um objecto especifico e seus serviços;

c) **Providência dos meio de definir as partições, abstracções e projecções** – O multinível dos DFD define o sistema apartir do nível mais

[44] Mazer, 1999

alto para o nível mais baixo de programação. A tal análise de processo fornece um meio para definir partições, abstracções e projecções. Em AOO (*UML*) a providência de partições e abstracções das classes é feito por meio de generalização e especialização e da sua construção interna;

d) **Encoraja o analista a pensar em termos do problema (aplicação)** – Como oposto o alto nível do DFD pode certamente encorajar os analistas a pensar em documentar em termos de requisitos como oposto à solução. Mesmo assim o DFD pode apresentar soluções bem claras para o desenho e ainda pode não existir uma distinção entre a análise dos requisitos e desenho. Já a AOO (*UML*) aparece um pouco discreto para assegurar tais objectos, atributos e conexões e o remanescente das estruturas no âmbito dos requisitos. Mesmo assim, às vezes pode ser difícil executar a análise do problema e manter objectos no âmbito do problema sem cair no desenho;

e) **Facilidade de os analistas a modificar a estrutura do conhecimento** – A modificação da Informação da análise e especificação dos requisitos está relacionada a orientação do utilizador de ferramentas automatizadas. Existem ferramentas automáticas para a AE e AOO;

f) **A Forma como encaram os dados** – A AE encara o âmbito do problema como um número de funções com os dados a serem secundários. Enquanto que a AOO (*UML*) encara os dados como primários e as funções como secundários e usa os objectos que se preocupam tanto com os dados e os métodos com os quais os dados são manipulados;

g) **Modo de encarar a solução do problema** – A AOO (UML) congrega todos os processos, dados, processos de controle e armazenamento de dados dentro de uma classe, enquanto que a AE separa estes elementos, isto é, *Top Down*, onde cada alto nível de função encapsula todos os detalhes das funções do nível mais baixo. Mesmo assim, a análise estruturada separa os dados das funções e a AOO (UML) encapsula a maioria do conhecimento do sistema quando possível dentro dos aspectos do sistema.

A tabela 2 enfatiza a diferença na filosofia entre as duas metodologias, mesmo assim, não está evidente qual é a metodologia para certo sistema. Alguns obcecados da *AOO (UML)* clamam que ela envolve a forma natural de pensar do que a metodologia funcional. Alguns proponentes das técnicas estruturadas insistem em afirmar que é tão natural pensar sobre as funções assim como sobre os objectos, [45].

AE	AOO (UML)
Atributos descritos em elipse fora da entidade.	Propriedades descritas no interior da classe.
Existência do conceito de relacionamento total e parcial, indicado por um circulo preenchido no canto do relacionamento.	A associação "total" é indicada pela colocação do número 1 como o menor número aceite de cardinalidade num dos lados da associação.
Apenas atributos são descritos no modelo.	As propriedades e acções da classe são descritos no modelo.
Ausência de polimorfismo	Presença de polimorfismo
Comportamento do sistema são as funções.	Comportamento do sistema são os objectos.
Caracteriza-se pela Estrutura Hierárquica.	Caracteriza-se pela Herança.

Tabela 2: Diferenças nos Modelos

As metodologias estruturadas de desenvolvimento de sistemas de informação apresentam muitas limitações que introduzem pontos de descontinuidade no processo de desenvolvimento de sistemas de Informação. Estas limitações ocorrem, principalmente, por serem utilizados diferentes diagramas em cada fase do ciclo de vida de um sistema, acarretando perda de informação e inconsistências na transição de um diagrama para o outro.

[45] Betolini at all, 1998

A AOO (*UML*) utiliza um modelo único, o qual é utilizado em todas as fases do ciclo de vida de um sistema. Dessa forma, o processo de desenvolvimento de sistemas é simplificado, interactivo e controlável . Cada interação acrescenta uma característica ao modelo, de forma a haver menores possibilidades de inconsistências e erros.

O Desenvolvimento de sistemas usando a AOO (*UML*) concentra a maior parte do esforço na fase de análise de requisitos. Este esforço adicional é compensado nas fases mais adiante como a implementação e mais fácil se torna a manutenção devido ao encapsulamento e reuso.

A tabela 3, a seguir, mostra que o que constitui desvantagem para a AE, é vantagem para *AOO* vantagem dada a sua evolução apartir da análise estruturada.

AE	AOO
A "poluição" Visual obtida pelo excesso de setas para ligação dos atributos às entidades.	Diminuição da poluição visual, colocando-se as propriedades dentro das classe.
A dificuldade de automatização dos modelos.	Com a vantagem conseguida ao se colocar as propriedades dentro das classes, torna-se possível o processo de automatização dos modelos em AOO (UML).
Não é possível representar as acções exercidas pelas entidades.	É possível descrever quais as acções as classes podem exercer sobre seus dados.

Tabela 3: Vantagens e desvantagens

6.1 – A AOO (*UML*) vem substituir ou complementar a AE?

Este é um assunto muito polémico pelo facto de até este momento estarem a decorrer estudos se a AOO (*UML*) vem substituir ou complementar a

abordagem estruturada. Mesmo assim, as duas metodologias acabam complementando-se porque muitas das vezes, quando de desenvolve um sistema faz-se análise usando a AOO (*UML*) e a implementação utiliza-se o modelo relacional. Este modelo é uma ferramenta da análise estruturada. É por isso que é normal falar se de bases de dados relacionais-objectos que usam tipo de dados complexos, linguagem de consulta poderosa e alta protecção, [46].

Em diversas aplicações cientificas, as informações estão fortemente relacionadas a determinadas funções, ou seja, os dados e suas operações estão fortemente interligados, [47]. A AOO permite a manipulação do par atributo-serviço e incorpora mecanismos que armazenam os obstâculos ao longo do processo de desenvolvimento de sistemas de informação.

7 - Conclusão

Nos últimos anos, verificou-se uma verdadeira revolução no campo da TI. Inicialmente voltada apenas para os aspectos técnicos envolvidos, as preocupações desta área passaram também a invocar as organizações e as pessoas que dela participam.

As metodologias de desenvolvimento de SI, da mesma forma, têm evoluído e um conjunto variado de técnicas e ferramentas associadas têm sido desenvolvidos e utilizado nos últimos anos.

Na actualidade, as metodologias procuram cada vez mais uma maior flexibilidade, condição básica para que atendam as mudanças vividas no campo das TI. As mudanças nos processos e estratégias de negócio, a necessidade de ciclos de vida mais curtos no desenvolvimento de SI, a adequação dos SI às necessidades das organizações, são preocupações que exigem maior flexibilidade.

[46] Mazer, 1999
[47] Mazer, 1999

Não existem grandes diferenças entre a AE e a AOO, não obstante, as diferenças fundamentais ocorrem quando se trata do modelo funcional, mas, mesmo assim, não se pode afirmar que um paradigma seja melhor que o outro. Certamente deve ser utilizado aquele que melhor responde as reais necessidades de desenvolvimento de cada sistema.

A AE, está consolidada como padrão e dificilmente será substituída a curto prazo, [48]. Sua utilização tem sido feita por analistas de diferentes tipos de SI, desde sistemas de pequeno porte até sistemas de segurança crítica.

A AOO, através da *UML*, é uma abordagem bastante nova, o que dificulta dar pareceres mais conclusivos sobre a sua utilização. A UML vai além de uma simples padronização a procura de uma notação unificada, [49]. A AOO (*UML*) recebeu influência das técnicas de modelagem de dados (modelo entidade-relacionamento), modelagem de negócio, modelagem de objectos e componentes e incorporou ideias.

Certamente que o aspecto mais importante da OO é a característica de encapsulamento dos dados, métodos e operações, enquanto na modelagem AE e linguagens de programação não OO) os dados são manipulados por qualquer processo; na OO, cada método que trata os dados pertencentes a uma classe, está encapsulado e sob controle de uma determinada classe e só ele faz operações necessárias sobre os dados. Todo o acesso a um dado de uma classe será feito por métodos associados àquela classe. Esta caracterização certamente garante uma maior confiabilidade e integridade dos dados.

A AOO (*UML*) é uma técnica muito importante para decompor o problema complexo em objectos ou conceitos. Por sua vez a análise estruturada decompõe o problema pela função ou processo e fluxos de dados, resultando no desmantelamento hierárquico dos processos compostos por sub-

[48] Betolini at all, 1998
[49] Furlan, 1998

processos. Esta técnica pode enquadrar-se bem em muitas aplicações de negócio baseadas em dados.

A AOO, nos últimos anos, tem tentado amadurecer a fase de análise e especificação de requisitos com o intuito de identificar as responsabilidades do sistema a ser implementado; é preciso um esforço conjunto entre os analistas de sistemas e os especialistas da área do problema, para definir o que o sistema deve fazer e não como.

Durante a fase de análise deve-se modelar o sistema como parte do mundo real, evitando criar uma representação estática e limitada do domínio do problema. Os requisitos de um sistema estão sempre a evoluir e provocando mudanças no modelo inicial.

A AOO (*UML)* focaliza a decomposição do espaço do problema em objecto do que função. A motivação primária para a orientação do objecto é de que o sistema evolui e as suas funções tendem a modificar, mas os seus objectos permanecem inalteráveis. Por isso que um sistema desenvolvido usando a OO poderá ser inerentemente conservável do que o sistema construído usando as metodologias funcionais, [50].

Dada as semelhanças e diferenças existentes entre a abordagem estruturada e a OO é ainda sobejamente debatido o caso de que uma metodologia é melhor que a outra. Um factor importante que pode ser usado na determinação de qual é a metodologia a ser usada é o tipo de sistema a ser desenvolvido.

[50] Davies, 1993

8 - Recomendações

Quando se pretende desenvolver um sistema de informação é aconselhável analisar as características do sistema de Informação, se for um SI de pequeno porte, centralizado nos dados, nas acções e acontecimentos é recomendável o uso de metodologia funcional.

Por vezes é importante usar as duas abordagens, isto é, fazer a análise usando a abordagem orientada a objecto e a implementação pode-se usar a metodologia funcional. Contudo quando se está perante um sistema bastante complexo, onde se manipulam vários tipos de informação recomenda-se o uso de metodologia orientada a objecto pois esta além do processamento de transações, integra os aspectos comportamentais do mundo real.

Mesmo assim a AOO (UML) está a começar a ser bem sucedida entre as metodologias já que pode ser usado para quase qualquer tipo de sistema de informação e fornece bons resultados.

9 – Bibliografia

- ANDRADE, L. P. e RAMOS, G. C. R. 1998. Método de Desenvolvimento de Sistemas Orientado a Objectos Utilizando a UML, Brasília, DF
- *AVISON, D.E e FITZGERALD, G.* 1997. *Information Systems Development Methodologies, Techniques and Tools. 2 ed. London: McGraw-Hill.*
- BLAY, G. L. & COLLINS, G. 1982. *Structured Systems Developmment Techniques*, editora *Great Britain,*.
- *BERTOLINI, C, MOURA, R. C, BORTOLI, L. A e COLOSSI* J. C. 1998, Análise Estruturada e Análise Orientada a Objectos: Estudo Comparativo, Universidade de Passo Fundo – Instituto de Ciências Exatas e Geociências, Bairro São José.
- *BOOCH*, G. 1994. *Object-oriented analysis and Design with Applications, Addison-wesley. Publishing* co.
- BARROS, P. 1998. (UML-Linguagem de Modelagem Unificada), Brasíl. 21/02/2001, ftp.sj.univali.br/prof/Westrupp/2162/UML-em-Portugues/page12.html.
- *COMOLO*, F.R. J. 1997. Trabalho de Licenciatura : Processo de Transicção entre as Metodologias Estruturadas e Orientadas a Objectos, 1ª edição, Maputo, UEM.
- *CASPERS*, J. 1994. *Object-Oriented Programming, 1st Edition.*
- *YOURDON*, E. & *CONSTANTINE*, L. L. 1992. Projecto Estruturado de Sistemas, Rio de Janeiro.
- *DAVIES*, P. B. 1993. *Information System Development, By the Macmillan Press Ltda*, Inglaterra *Second Edition*, 263.
- *DATE*, C. J. 1984. Bancos de dados, Rio de Janeiro editora *campus*,
- *FURLAN*, J. D. 1998. Modelagem de Objectos Através da UML, Makron *Books.*
- *HERLEA*, D. 1999. *Structured and Object-Oriented Analysis and Design.*
- ILTEC - Instituto de Linguistica Teórica e Computacional, 1993. Dicionário de Termos Informáticos, 1ª edição, Lisboa, Edições Cosmos.

- *JAMES*, R. , *Ivar*, J. and *Grady*, B. 1999. *The Unified Modeling Language User Guide*, *Addison Wesley Longman, Inc.*
- *JAIR C LEITE, J.C. 2000, 23/05/2001,* http://www.dimap.ufrn.br/~jair/ES/c3.html.
- *MICHALSKU*, H. 1994. Object Oriented Design: Terms and defitions.
- *MARTIN*, J & *Gdell*, J. J. 1995. Análise e Projecto Orientado a Objecto Rio de janeiro, Editora *Makros Books.*
- *MAZER* A. Jr. 1999. Bancos de Dados Objecto-Relacional, Universidade Estadual de Ponta Grossa, Brasil.
- *MARTYNA*, B. & *Elbert*, B. 1994. Client/Server Computing, Artech. House INC.
- Pereira, J. L. 1997. Tecnologia de Bases de Dados, Lisboa, editora de Informática LTD.
- *PRESSMAN*, R. S. 1995. Engenharia de *Software*. Brasil, *Makron Books.*
- RAMALHO, J. A. 1999. Microsoft SQL, Server 7, Markon Books, São Paulo.
- ROQUE, R. F. 1998. Estudo Comparativo de Metodologias de Desenvolvimento de Sistemas de Informação Utilizando a Técnica Delphi, Universidade Federal de Santa Catarina, Florianópolis. http://www.eps.ufsc.br/disserta98/ruth/ 12/04/2001.
- *STEVENS* W. P. 1981. Projecto Estruturado de Sistemas, 3ª edição, editora *campus.*
- *SARSON*, T e *Gane* C. 1983. Análise Estruturada de Sistemas, Brazilia, editora Livros técnicos e científicos Ltd.
- *SILBERSCHATZ*, A. *KORTH*, H. F. e *SUDARSHAN*, S. 1999. Sistema de Banco de Dados, 3ª edição, São Paulo, *Makron Books.*
- *TONSIG*, S.L, (UML - Linguagem de Modelagem Unificada), 10/09/2001, http://www.geocities.com/Athens/Olympus/1307/manual6.zip,
- *TONSIG S. L 1998.* Métodos Orientados a Objectos, Brasil.
- *YOURDON, E. & Cood*, P. 1991. Object-Oriented Analysis, second edition, editora PTR *Practice Hall Building.*

- *YOURDON*, E. and *Edward*, E. 1989. *Modern Structured Analysis*, *Englewood Cliffs*, N.J: *Prentice Hall*.
- *YOURDON*, E. 1990. Análise Estruturada Moderna, Editora *Campus* Ltda Série *Yourdon Press*.

Anexo

Conceitos básicos

Objecto – é a representação informativa a mais aproximada de um objecto rela, que tem características que o distingue de outros objectos e pode ser manipulado através do seu "interface"

Abstracção de dados (*data abstraction*) – deixa de haver um conjunto de tipos de dados específicos e operações específicas para esses dados e passamos a ter dados genéricos e operações genéricas definidas no objecto.

Classe (*Class*) – é o esqueleto de um objecto em que se definem os tipos de dados e os métodos que agem sobre esses dados. Os objectos são instâncias de uma classe.

Classe abstracta (*abstract Class*) – uma classe que não tem instâncias e sim subclasses que, por sua vez, podem ter instâncias.

Ocultar informação (*Information hidding*) – trata-se de ocultar os pormenores de implementação e da estrutura de um objecto. Assim, temos um meio de protecção e de restrição de acesso aos dados do objectos.

Encapsulamento (*encapsulation*) – este termo refere duas propriedades dos objectos: o facto dos dados e do código para manipular formarem um objecto, e o facto dos clientes de um módulo deverem depender somente do interface do módulo, não necessitando de conhecer a estrutura interna do módulo para o utilizarem.

Hierarquia de classes (*Class hierarchy*) – quando se derivam classes forma-se uma estrutura hierárquica em árvore, sendo a relação entre elas uma relação de especialização (superclasse, subclasse).

Herança (*inheritance*) – a herança entre classes permite que uma classe herde uma ou várias outras classes. Por seu lado, uma subclasse herda de uma ou mais superclasses os seus atributos e comportamento. As subclasses dizem-se especializações da superclasse pois estas adicionam outros atributos e comportamentos ao que herdam e, por vezes, alteram comportamentos herdados. As superclasses dizem-se generalizações das suas subclasses. Há uma característica importante da herança que reside no facto desta ser recursiva: uma classe herda de uma superclasse que herdou de outra superclasse, etc.

Herança múltipla (*Multiple inheritance*) – quando um objecto herda características de mais do que um objecto.

Polimorfismo (*polymorphism*) – quando se usa a mesma mensagem (método) para comunicar com vários objectos, mas cada um deles interpreta-a a seu modo.

Objecto composto (*compound object*) – objecto composto por um conjunto de objectos relacionados entre si.

Identificador do objecto (*object identifier OID*) – identificador unívoco de um objecto. Uma base de dados verdadeiramente orientada para objectos garante que o uso deste identificador seja transparente para os clientes. Mesmo que o objecto não esteja em memória, a base de dados encarrega-se de o carregar para a memória sem intervenção explicita do programa ou utilizador.

Metadata – dados que descrevem dados. Os modelos são metadata visto que descrevem dados, relações e até modelos.