Volume

2

MANUAIS TÉCNICOS

Cursos de Formação Profissional

Análise Essencial de Sistemas com Orientação a Objetos

PLENA CONSULTORIA E PLANEJAMENTO EM INFORMÁTICA LTDA

Análise Essencial de Sistemas com Orientação a Objetos

Rua da Consolação, 2697 - 5° andar - São Paulo - SP - 01416-001
Telefone (011) 852-1333 • Fax (011) 852-0013
e-mail plena@ibm.net

Índice analítico

Capítulo 1

Sistemas

Os sistemas com os quais nós trabalhamos podem ser classificados como "Sistemas de Resposta Planejada" e possuem as seguintes características:

- Há uma necessidade ou oportunidade de negócio que precisa ser atendida no mundo real; o sistema surge como resposta a essa necessidade ou oportunidade.

- Todo sistema tem uma finalidade específica: o que o sistema faz, e armazena, está diretamente relacionado com essa finalidade.

- Os sistemas têm de fornecer respostas previsíveis: há um conjunto de regras que o sistema deve obedecer, no sentido de responder a eventos externos ao sistema, de acordo com sua finalidade.

- Os sistema são transferíveis: as regras que os regem precisam ser escritas de modo que os sistemas possam ser transferidos de um processador para outro, quer ele seja automatizado ou humano.

- A fronteira do sistema é caracterizada pelas entradas/origens dos dados e pelas saídas/destinos das informações; a determinação da "fronteira" de um sistema é uma boa ferramenta para caracterizá-lo.

- O interior do sistema consiste de atividades e de dados armazenados, que são as engrenagens que respondem a eventos externos, de acordo com as finalidades do sistema.

- A essência do sistema independe da tecnologia utilizada para implementá-lo. Por exemplo: o sistema de contas correntes do Banco do Brasil é o mesmo desde sua fundação há mais de um século. O que tem mudado, muito rapidamente, é a tecnologia de implementação do sistema.

A Captação da Essência

O trabalho de análise de um sistema consiste em conseguir captar a sua essência: os fatos que ocorrem fora do sistema que causarão uma ação do sistema como resposta. Esse trabalho deve procurar filtrar todas as condições e características de implementação, em busca da sua essência. Essencial é aquilo que o sistema deve ter. Se não tiver, o sistema não conseguirá atingir seus objetivos.

A Formação do Pessoal de Sistemas

Em geral, os profissionais de informática são formados em Economia, Administração de Empresas, Engenharia, etc.. Em seus cursos de graduação tiveram contato com a informática em temas como Cálculo Numérico.

A formação efetiva desses profissionais dá-se por "osmose" dentro das empresas, por troca de "experiências" com profissionais mais tarimbados. Quando há treinamento, este é executado pelo fabricante do equipamento utilizado e em ferramentas de software existentes e utilizadas na instalação. Dessa forma, depois de alguns anos, o profissional é promovido à categoria "senior" e possui vasto conhecimento em softwares básico, habitualmente denominados por três ou quatro letras. As técnicas de análise de sistemas, de levantamento de dados, de projeto de sistemas, de codificação e de testes praticadas são aquelas que os outros já praticavam.

Aliado a esse fato, existe outro: o profissional de informática, em geral, é fortemente reacionário e resistente a novas técnicas que causem qualquer mudança em sua rotina diária e em seu comportamento pessoal.

Não podemos esquecer que a informática é uma ciência nova, que está em constante e rápida evolução.

Como promotor da entrada da tecnologia na empresa, o profissional de informática não deve ser ele próprio reativo às novas tecnologias.

A Cultura do Usuário

Os usuários têm a formação de acordo com sua área de atuação. Assim, é claro que o Departamento Jurídico está repleto de advogados, a Contabilidade é formada por contadores e o Departamento Financeiro é comandado por economistas.

Os usuários, ao longo dos anos, têm tido algumas más experiências com a informática. Sistemas desenvolvidos para eles que não funcionaram adequadamente, sistemas que nunca ficaram prontos, sistemas que não se adaptam às mudanças em sua área, sistemas que acabam acrescentando trabalho ao seu dia-a-dia. Apesar disso, ele ainda acredita na informática e quer, cada vez mais, apoio de sistemas em suas atividades. O difícil é conseguir que o pessoal lá da informática o atenda. Por isso vemos, com tanta freqüência, usuários montando verdadeiros "CPDs" paralelos para terem um mínimo de apoio informatizado em seus negócios.

A Comunicação Analista X Usuário

Os usuários conhecem o problema, os negócios, muito melhor que os analistas. Os analistas conhecem técnicas para resolver os problemas. O trabalho de análise deve produzir uma solução para o problema. Geralmente a linguagem do usuário é impregnada de termos do vocabulário técnico do assunto do problema que os analistas não entendem completamente. Por outro lado, ao expor a solução, os analistas utilizam uma linguagem técnica derivada do "informatiquês" que o usuário desconhece.

A qualidade da solução é fortemente influenciada pela qualidade de sua validação pelos usuários. Será que nós entendemos corretamente o problema? Será que a solução é viável?

É importante que o trabalho seja feito em conjunto e que se utilize de vocabulário próprio, comum a todos os participantes. Analistas e usuários trabalhando no mesmo time, em busca da melhor solução. Trabalhar junto leva ao entendimento, à melhor comunicação e à sinergia.

A Documentação

Para possibilitar essa comunicação há a necessidade de documentarmos o entendimento obtido. Essa documentação servirá de base, num momento posterior, ao trabalho de manutenção do sistema já instalado.

Para que a documentação seja útil, é importante que ela seja produzida durante o trabalho, que seja completa e que esteja atualizada. Uma documentação desatualizada não serve para nada.

Sistemas com documentação incompleta ou desatualizada precisam de "arqueólogos", especialistas que vasculham fragmentos de código, arquivos e "dumps" com o objetivo de construir hipóteses a respeito de "pra que serve isso?" no sistema. Além dos "arqueólogos", o sistema precisará ser atendido por "pediatras", profissionais que são chamados a qualquer hora, principalmente de madrugada ou aos fins de semana, toda vez que o sistema "passa mal".

Modelos

Um modelo é uma representação abstrata de algo real. O modelo tem o objetivo de imitar a realidade para possibilitar que esta seja estudada quanto ao seu comportamento.

Tradicionalmente, a engenharia tem empregado modelos em escala, plantas, desenhos de circuitos, protótipos.

O uso de modelos torna o estudo mais barato e seguro: é muito mais rápido e barato construir um modelo que construir a "coisa real".

O objetivo do modelo é mostrar como será o sistema, para permitir sua inspeção e verificação, de modo a poder receber alterações e adaptações antes de ficar pronto.

Qualquer modelo realça certos aspectos do que está sendo modelado em detrimento de outros.

A ferramenta que usamos para modelar influi diretamente na forma como pensamos sobre a realidade e determina quais aspectos serão mostrados e quais ficarão escondidos.

Ferramentas de Modelagem

Para modelarmos sistemas, necessitamos de ferramentas com as seguintes características:

- **Gráficas**, com apoio de textos, como um mapa: legíveis, concisas e padronizadas. A parte gráfica particiona o sistema em componentes interrelacionados, sem as restrições de um texto linear. A parte textual descreve os componentes e suas interconexões;

- **Particionáveis**, do geral para o específico, como um atlas geográfico, tornando o modelo mais fácil de entender. Precisamos ser capazes de ter a visão geral do sistema sem nos preocuparmos com detalhes e, termos a visão do detalhe sem perdermos a noção do todo;

- **Rigorosas**, como as escalas de uma planta. O modelo do sistema deve ser preciso, sem erros, incompletudes e redundâncias. Mas deve ser flexível, fácil de mudar, pois o sistema está sempre em constante evolução;

- **Capazes de predizer o comportamento** do sistema, como um protótipo. O modelo deve ser capaz de nos dizer o que é bom e o que é ruim no sistema, em termos objetivos, antes que o sistema seja implementado. O modelo deve nos permitir simular o comportamento do sistema.

- **Capazes de mostrar os aspectos críticos** do sistema, como fotos de vários ângulos, com simplicidade. Cada modelo deve ser uma projeção do sistema em um espaço simplificado.

Ferramentas de Modelagem Tradicionais

As ferramentas tradicionais de modelagem de sistemas levam a modelos redundantes, contraditórios e difíceis de checar quanto à sua completude e precisão. É necessário que utilizemos ferramentas de modelagem substancialmente diferentes daquelas anteriormente utilizadas pelos métodos tradicionais.

Fases do Desenvolvimento de Sistemas

O Desenvolvimento de sistemas com Análise Essencial se dá sempre de maneira a se obter mais conhecimento e qualidade a cada passo, através de refinamentos sucessivos. O primeiro passo é construir o **Modelo do Ambiente** do sistema, composto pelo Diagrama de Contexto e pela Lista de Eventos Externos. A finalidade desse modelo é determinar a abrangência do sistema pela especificação dos eventos que motivam a sua ação, das entradas e suas origens e das saídas e seus destinos.

A partir da Lista de Eventos Externos, do Modelo do Ambiente, deve ser construído o **Modelo do Comportamento** do sistema, que especifica de que forma o sistema é estruturado em termos de dados e funções para atingir aos objetivos estabelecidos e para responder aos eventos do **Modelo do Ambiente**. A Figura SDS 1 - O desenvolvimento de sistemas com Análise Essencial, mostra o contexto do desenvolvimento de sistemas visto, ele próprio, como um sistema.

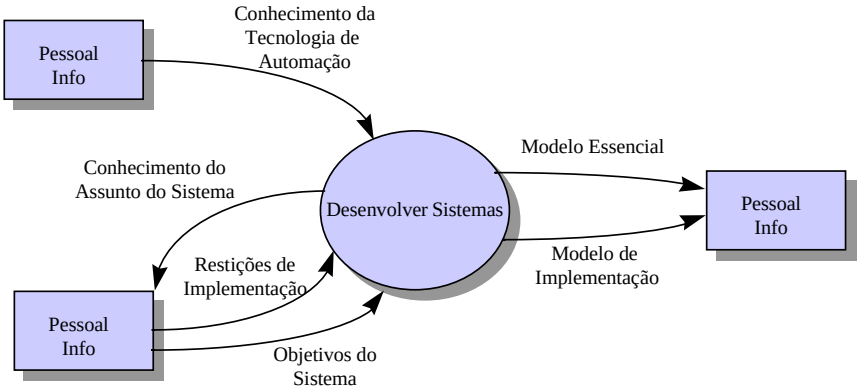


Figura SDS 1 - O desenvolvimento de sistemas com Análise Essencial

O desenvolvimento de sistemas interage com o pessoal de informática, que detém o conhecimento da tecnologia de automação; com o usuário, responsável pela definição dos objetivos e restrições do sistema e profundo conhecedor do assunto e do ambiente do sistema. A empresa é a beneficiária do trabalho, recebendo o produto do desenvolvimento: o sistema.

A Figura 0 - Desenvolver Sistemas, mostra os dois principais processos do desenvolvimento de sistemas.

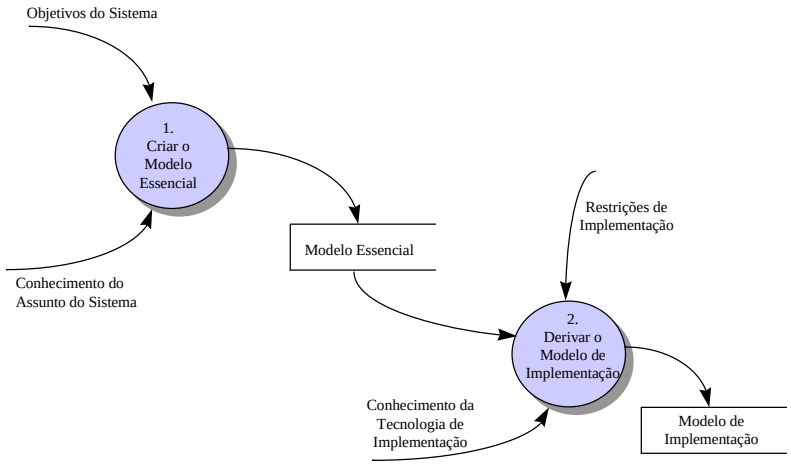


Figura 0 - Desenvolver Sistemas

A Figura 1 - Criar o Modelo Essencial detalha o processo de modelagem essencial. Em primeiro lugar fazemos a modelagem do Ambiente do Sistema: a quais eventos externos o sistema deve responder; quais os fluxos de dados fornecidos pelo ambiente

ao sistema; de onde vêm esses dados; quais as informações que o sistema deve produzir; quem são os usuários dessas informações.

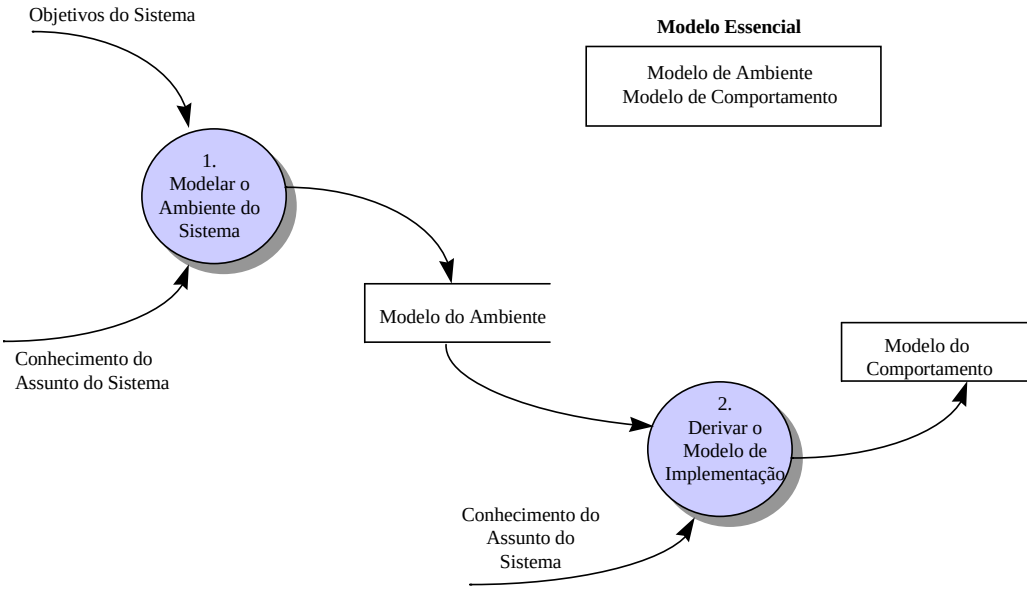


Figura 1 - Criar o Modelo Essencial

A partir do Modelo do Ambiente é derivado o Comportamento do Sistema: quais são as estruturas de dados, os objetos de negócio, quais são os processos internos do sistema necessários para garantir que o sistema atinja seus objetivos, através da ativação dos métodos dos objetos de negócio.

A Figura 1.1 - Modelar o Ambiente do Sistema, mostra que essa tarefa é feita pela determinação do Contexto do Sistema e pela identificação de Eventos Externos. Essas tarefas são executadas em conjunto, uma subsidiando a outra.

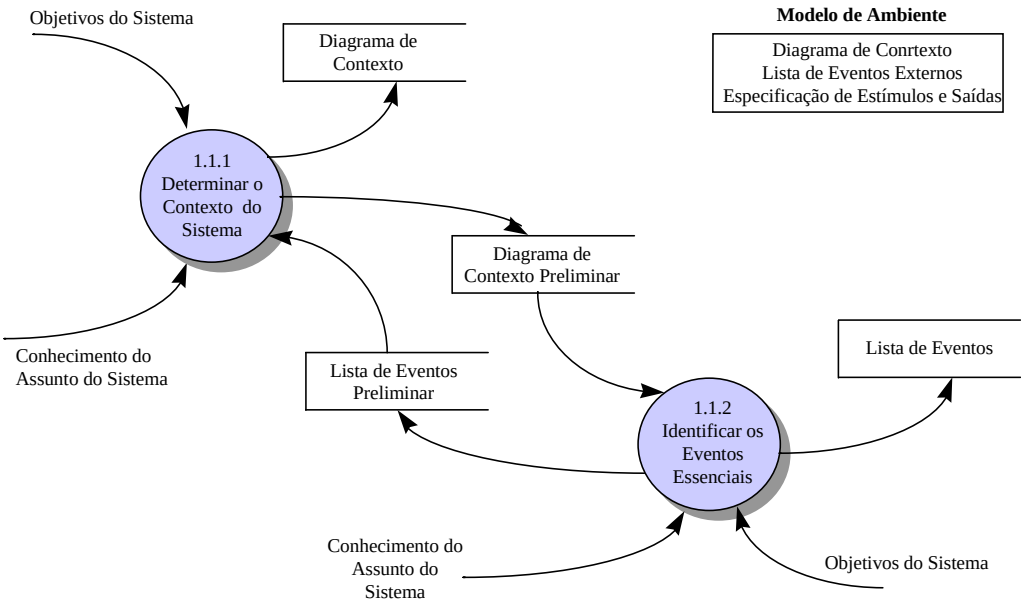


Figura 1.1 - Modelar o Ambiente do Sistema

A figura 1.2 - Derivar o Comportamento do Sistema - ilustra que essa tarefa é constituída de quatro partes: A modelagem dos dados, a modelagem das atividades essenciais, a modelagem dos objetos de negócio e a especificação dos detalhes em um dicionário de dados.

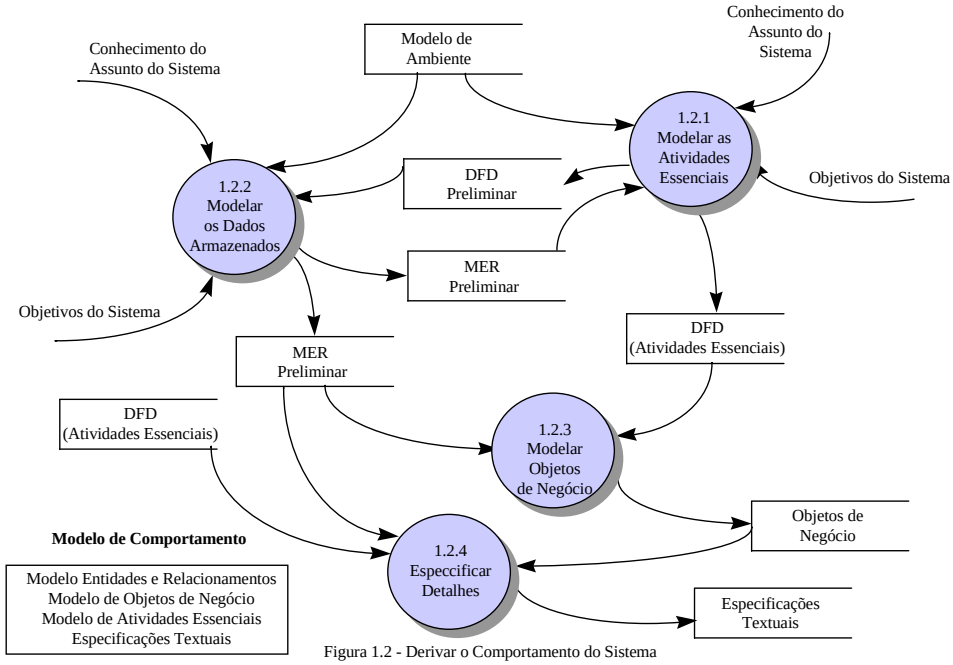


Figura 1.2 - Derivar o Comportamento do Sistema

A Figura 2 - Derivar o Modelo de Implementação, consiste, em primeiro lugar, na determinação do Modelo de Processadores, onde as atividades essenciais são alocadas a processadores, que são os agentes que executarão cada uma das atividades implementadas, fisicamente(entenda-se, aqui, por exemplo, computadores e pessoas); na derivação do Modelo de Tarefas, onde as atividades são organizadas, por processador, em seqüências ou grupos de operação que fazem trabalho útil para o usuário (tarefas), e na derivação do Modelo de Arquitetura de Código, onde as mesmas atividades são divididas em objetos ou módulos funcionais (dependendo do ambiente de hardware), com o máximo de coesão interna e o mínimo de interdependência entre módulos.

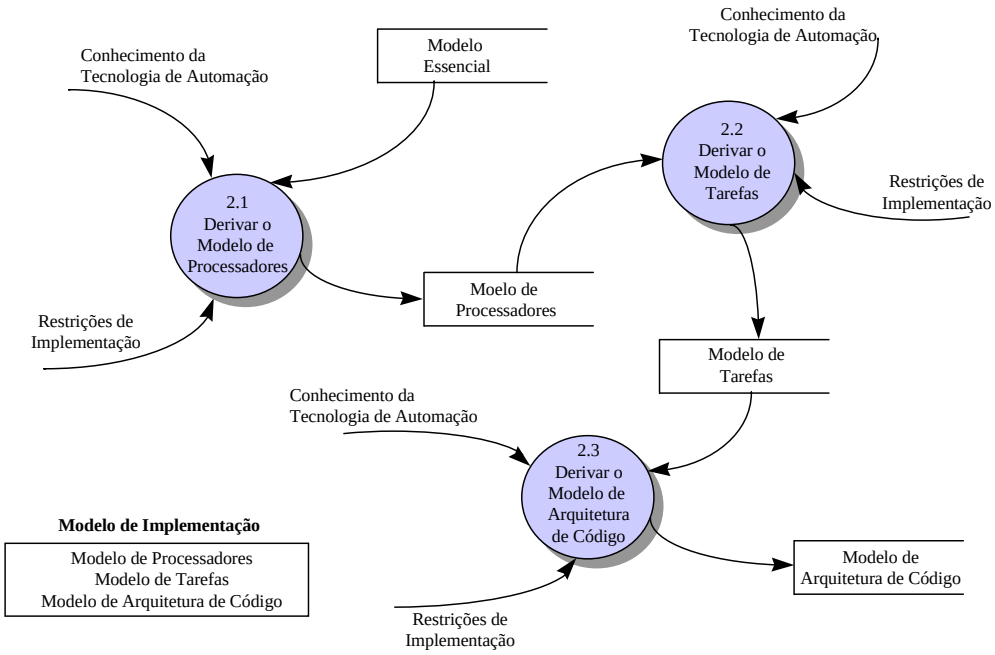


Figura 2 - Derivar o Modelo de Implementação

Tabela de Produtos

Usando o Modelo de Processos do Sistema de Desenvolvimento de Sistemas (SDS), representamos abaixo os produtos conseguidos com as técnicas.

Modelo de Sistemas	Modelo Essencial	Modelo de Ambiente	. Lista de Eventos . Diagrama de Contexto . Especificação Estímulos/Saídas
		Modelo de Comportamento	. Modelo de Dados DER + DD . Mod. Atividades DFD + Minispecs
	Modelo de Implementação	Modelo de Processadores	. DFD de Processadores . Diálogos . Protótipos
		Modelo de Tarefas	. Jobs . Procedures . Transações p/ Tarefas Automatizadas
		Modelo de Arquitetura de Código	. Projeto de Objetos

Capítulo 2

Modelagem de Eventos

A construção do Modelo Ambiental se dá através da Definição do Contexto do Sistema e da Modelagem da Lista de Eventos Externos.

Um **evento externo** é algo que ocorre no ambiente (fora dos limites do sistema) e que provoca uma resposta do sistema, como conseqüência. Um evento pode ser sinalizado por um fluxo de dados ou ser localizado em um determinado instante no tempo.

Analisar eventos é, basicamente, identificar fatos que ocorrem no meio ambiente que interage com o sistema e que exigem uma resposta do mesmo. Esta resposta pode ser, por exemplo, o armazenamento de uma informação ou a produção de um resultado.

Algoritmo para Modelagem de Eventos

Analise o ambiente do sistema e ponha no papel todo fato que, a princípio, pareça determinar uma ação do sistema. Cada evento deve ser escrito em uma oração. Lembre-se que uma oração é uma construção gramatical que expressa uma idéia completa. Deve possuir um sujeito, um verbo e um objeto.

Eventos Relacionados

Para cada evento identificado, responda as perguntas abaixo. A resposta a essas perguntas levará à identificação de outros eventos. Para cada um desses novos eventos, responda novamente as perguntas. Repita esse processo exaustivamente, até que o ciclo se feche.

- Existe algum evento que seja uma variação significativa do evento identificado? Normalmente esses eventos podem ser escritos com as mesmas palavras do evento original, trocando-se apenas o verbo.

- Existe algum evento oposto ao evento identificado? Por exemplo: oposto a vender é comprar; oposto a pagar é receber (antônimos).

- Há algum evento negativo do evento identificado? Consiste na negação do evento. Negativo de pagar é não pagar.

- Existe algum evento que deve preceder o evento identificado? Normalmente esses eventos são pré-requisitos do evento em questão. Por exemplo, para que seja feito um pagamento, é necessário que tenha sido feita uma compra.

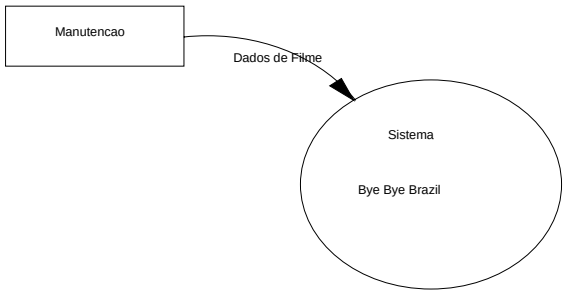
- Há algum evento que seja conseqüência do evento identificado? Estes eventos devem acontecer, como conseqüência, após o evento em questão .

Determinando o Contexto do Sistema

O **Diagrama de Contexto** é a representação gráfica do **Ambiente do Sistema**. Nele, o sistema como um todo é representado por um círculo, com o seu nome. Os usuários (fornecedores e receptores de informações) são representados por retângulos, rotulados pelo nome do agente externo e as informações trocadas são as setas identificadas pelo nome do pacote de informações que fluem entre o sistema e seus usuários.

Eventos Externos e Eventos Internos

Na modelagem do ambiente, preocupamo-nos apenas com os eventos externos, isto é, aqueles que ocorrem fora do sistema. Eventos internos são representados por orações onde o sujeito é o sistema ou por respostas do sistema a eventos externos.



Fluxos de Dados e Eventos

Nem todo evento é sinalizado por um fluxo de dados. Os eventos temporais, por exemplo, não possuem um fluxo de dados de entrada como estímulo.

Nem todo evento possui um fluxo de saída correspondente. Um evento pode causar, apenas, que algum dado seja armazenado.

Eventos diferentes podem estar associados a um mesmo fluxo de dados.

Documentando o Modelo Ambiental

O **Modelo Ambiental** deve ser documentado através das seguintes especificações:

- **Objetivos do Sistema**. É uma lista de objetivos que o sistema deve atender. Essa lista será a "baliza" que norteará o conceito de essencial ou não, onde essencial é tudo aquilo que é necessário para atingir um determinado objetivo.

- **Lista de Problemas**. Essa lista ajudará na construção da **Lista de Objetivos**. Os objetivos de um sistema podem ser de duas categorias: os que são ligados às necessidades futuras do usuário e os que são ligados à solução de problemas que o usuário enfrenta atualmente.

- **Lista de Eventos Externos**. Constará da especificação, para cada evento, do fluxo de estímulo e sua origem, da resposta que o sistema deve dar ao evento e do fluxo de saída produzido pelo sistema, com seu destino.

- **Diagrama de Contexto**. Consiste na representação gráfica dos elementos identificados na Lista de Eventos Externos, representando as origens e

seus estímulos, aos destinos e suas saídas, além do sistema que os recebe / produz, sem nenhuma especificação de detalhes.

- **Dicionário de Dados**. Uma entrada para cada dado identificado como fluxo de estímulo ou de saída do sistema (Ver Capítulo "Especificações Textuais").

Capítulo 3

Após a determinação do **Modelo de Ambient**e, deve ser derivado o **Modelo de Dados do Sistema**, através da técnica de **Modelagem de Entidades e Relacionamentos** (ver manual específico).

A modelagem dos dados determina os depósitos de dados, que o sistema utilizará para custodiar as informações que lhe são essenciais.

Os dados serão derivados do Modelo de Ambiente e, portanto, para passar ao próximo item, tenha-o à mão.

Produzindo o MER a partir da Lista de Eventos

Para cada evento do **Modelo de Ambiente**, produza o **MER** conforme a seqüência abaixo:

- **Evento** = o sujeito e o objeto do evento são candidatos a *entidades* e o verbo, candidato a *relacionamento*.
- **Estímulo** = verificar se os dados devem ser armazenados e onde (sujeito, objeto ou outra entidade).
- **Ação / Resposta** = conhecimento do sistema (regras); quais Entidades ou Relacionamento são criados, usados, modificados ou removidos
- **Saída** = verificar onde (Entidade ou Relacionameto) os dados devem ser obtidos.

Obs.: Lembrar que Eventos Temporais não têm estímulo e nem sempre são escritos com *sujeito, verbo e objeto*.

Eventos Custodiais

Os **Eventos Essenciais** são divididos em dois tipos:

- **Fundamentais** - têm a ver com o *negócio do sistema*, o dia-a-dia comum do(s) usuário(s);
- **Custodiais** - tomam conta da base de informações da organização, garantindo o funcionamento das respostas planejadas para os Eventos Fundamentais.

O **Modelo de Dados** foi derivado do Modelo Ambiental mais o conhecimento do assunto, porém, o Modelo Ambiental não atende totalmente o Modelo de Dados.

Para que esse problema seja resolvido, nós precisamos refinar o Modelo Ambiental, isto é, *adicionar novos eventos* à lista, usando a seguinte regra:

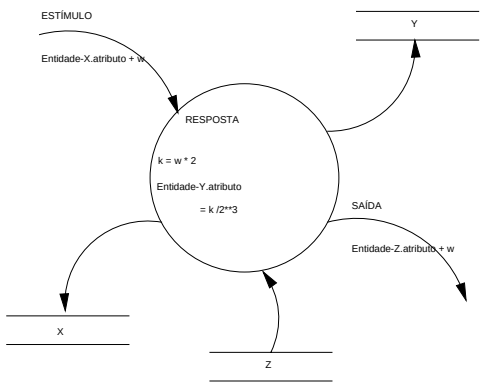
Verificar se existem e são significativos, eventos que:

- criam;
- modificam;
- removem e
- usam

cada objeto do Modelo de Dados, isto é, *entidade, relacionamento* e *atributo significativo* (atributo que muda o estado da entidade).

Obs.: Lembre-se que refinar o Modelo Ambiental é, além de adicionar os eventos, completá-lo com os estímulos e saídas na Lista de Eventos e no Diagrama de Contexto e com a resposta planejada para cada evento.

Anatomia do Processo Essencial



Derivando o Modelo de Processos

Com o Modelo de Ambiente e o Modelo de Dados à mão, faça os seguintes passos, para cada evento e respectivos desenhos em folha única.

- Passo 1: **Identifique os Processos** = para cada evento, **crie** um **Processo** (representado por uma bolha no DFD) rotulado com o nome da resposta que o sistema dá ao evento.

- Passo 2: **Adicione os Fluxos de Dados** = adicione o **Fluxo de Estímulo**, representando-o como uma **seta entrando no Processo** e adicione o(s) **Fluxo(s) de Saída(s)**, representando-o(s) como seta(s) saindo do Processo.

- Passo 3: **Adicione os Depósitos de Dados** = considere a lógica do Processo e **adicione** os **Depósitos de Dados criados, removidos** ou **atualizados** pelo processo, representando-os por duas linhas horizontais, com o seu nome tirado do Modelo de Dados e por uma seta saindo do processo e entrando no referido depósito. ***Adicione*** os **Depósitos de Dados lidos** pelo processo, representando-os por duas linhas horizontais com o seu nome tirado do Modelo de Dados e por uma seta saindo do depósito e entrando no processo.

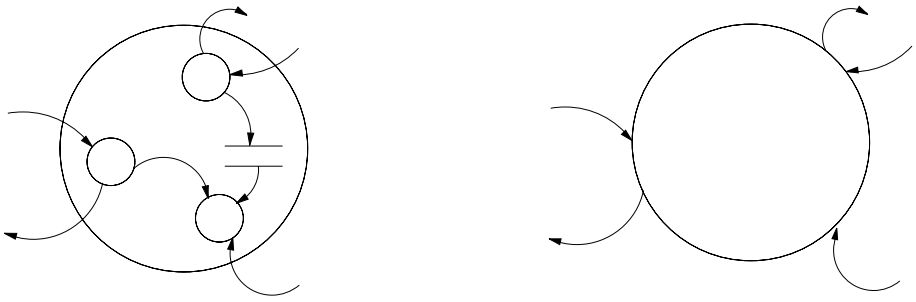
O modelo produzido terá tantos processos quantos forem os eventos. É óbvio que esse modelo não é bom para apresentação, validação e compreensão do sistema. Por isso, ele deverá ser nivelado.

Nivelando o Modelo de Processos

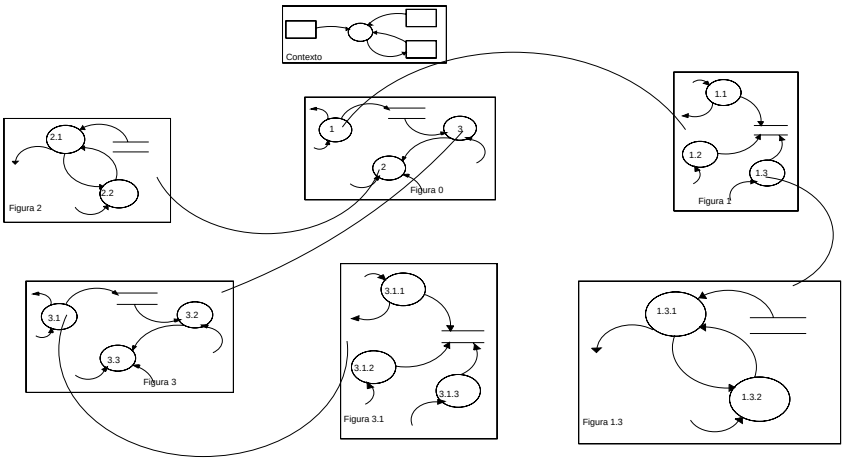
Nivelar o **Modelo de Processos** consiste em organizá-lo em vários DFD's, onde cada um deles apresenta um determinado nível de detalhe. Assim, como num atlas, haverá um modelo que representa o sistema e outro que representa os subsistemas. Cada processo do DFD nível 1 (aquele que representa os subsistemas) pode ser detalhado em outro DFD que o considera como se aquele processo fosse o sistema. Este nivelamento deve ser feito até que todos os processos sejam representados em seu mais baixo nível, ou seja, seu detalhamento é feito através de especificação de lógica de processo, ao invés de por outro DFD.

O Conceito de Nivelamento

Um DFD, que representa um conjunto de transformações graficamente conectadas, pode ser desenhado através de uma única transformação, com o mesmo conjunto de entradas e saídas.



Várias transformações podem ser juntadas em uma única, num nível mais alto. Da mesma maneira, uma transformação pode ser dividida em um grupo de transformações, num nível mais baixo.



O menor nível de transformações determina o modelo completo do sistema. As transformações de mais alto nível servem apenas como abreviações e nos ajudam a lidar com a complexidade dos diagramas de mais baixo nível.

Desde que a única diferença entre um nível e outro é o grau de detalhe mostrado, só escrevemos especificações textuais para as transformações de mais baixo nível: as **primitivas funcionais**.

Balanceando o Modelo

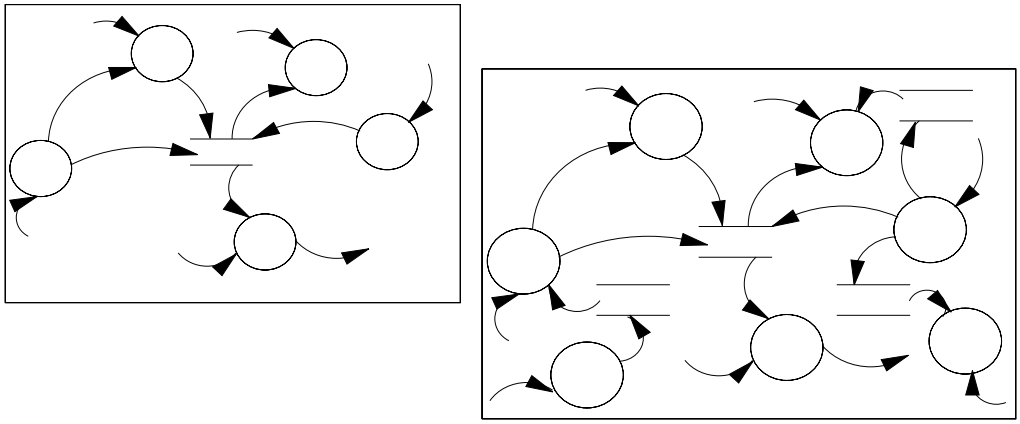
A organização do modelo gráfico é muito mais do que uma questão de convivência.

Um DFD ou DER complexo pode sobrecarregar o leitor pela apresentação de mais detalhes do que sua mente pode absorver.

Desde que a correção do modelo depende de sua compreensibilidade, o processo de balanceamento e revisão é essencial para assegurar a qualidade do processo de desenvolvimento de sistemas.

Pesquisas sobre a memória e percepção humanas sugerem que um diagrama que contenha 7 elementos, com uma variação de mais ou menos 2, irá sobrecarregar a maioria dos leitores.

Uma **regra objetiva** é: um ***DFD inteligível*** é um DFD com poucos fluxos de dados, que ***não deve ter mais que meia dúzia*** de transformações.



Balanceando entre Projeções

O modelo de processos e o modelo de dados estão balanceados quando:

- cada elemento de dados, de cada entidade no DER, é usado por uma ou mais transformações ou é requerido por consultas *ad-hoc*;
- cada relacionamento no DER é usado por uma ou mais transformações ou é requerido por consultas *ad-hoc*;
- cada elemento de dados, de cada entidade ou entidade associativa no DER, é carregado por uma ou mais transformações e
- cada relacionamento do DER é criado por uma ou mais transformações.

Balanceamento entre Níveis

Os modelos de dados e de processos devem fornecer um modelo único e consistente do sistema.

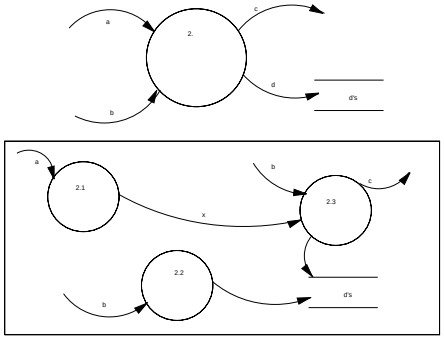
Precisamos assegurar que cada nível do modelo é exatamente eqüivalente aos demais níveis.

Regra de balanceamento

Cada processo "pai" deve ter exatamente as mesmas entradas e saídas que o diagrama "filho", que está no nível imediatamente abaixo.

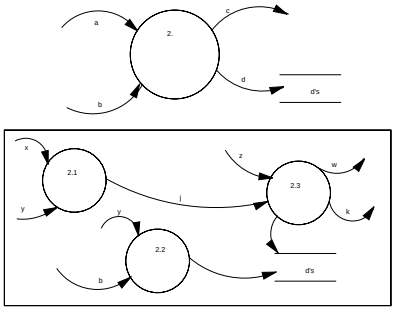
Balanceamento visual de fluxos

Se nós não nivelarmos os fluxos, o balanceamento será fácil: a bolha "pai" e o diagrama "filho" terão exatamente os mesmos nomes de fluxos.



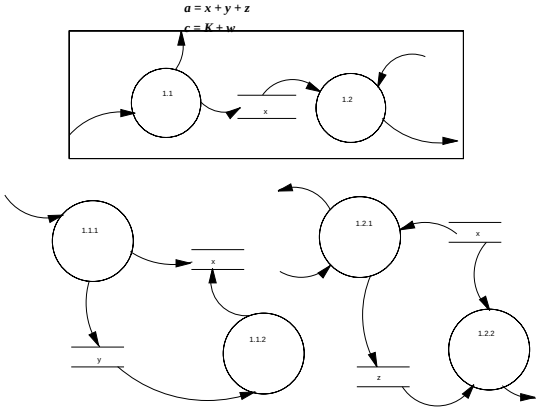
Balanceamento via especificações textuais

Quando nivelamos fluxos, usamos as especificações textuais (**Dicionário de Dados**) para estabelecer a equivalência dos "fluxos pai" e "fluxos filhos".

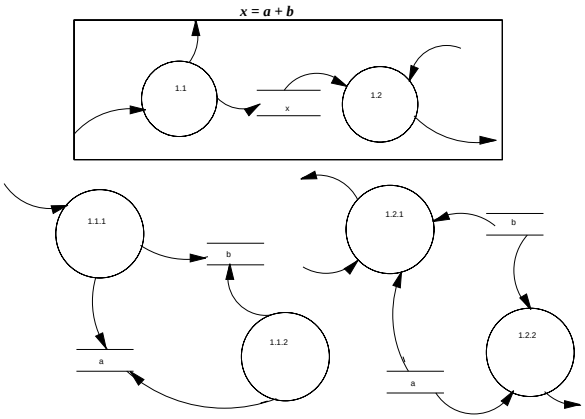


Balanceando Depósitos de Dados

Um depósito de dados aparece pela primeira vez, no nível em que ele é compartilhado entre dois processos. Abaixo desse nível, o depósito aparece sempre que for referenciado.

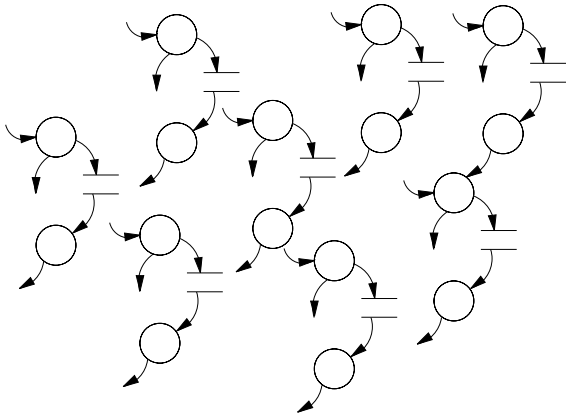


Depósitos de dados podem ser nivelados da mesma forma que processos e fluxos.

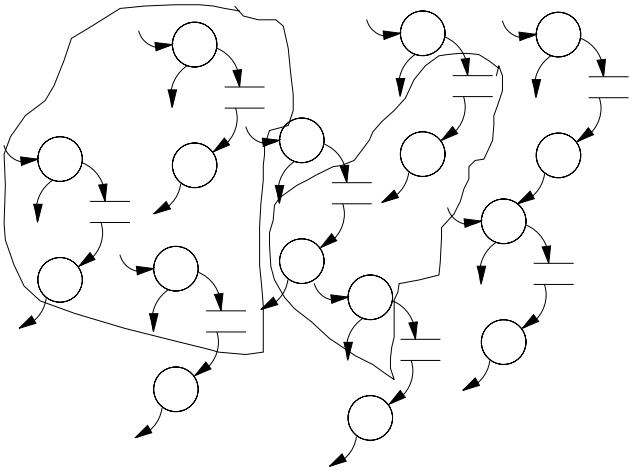


Nivelando o Modelo Essencial

A derivação das projeções gráficas, a partir da lista de eventos, produz um modelo que não é bem particionado.

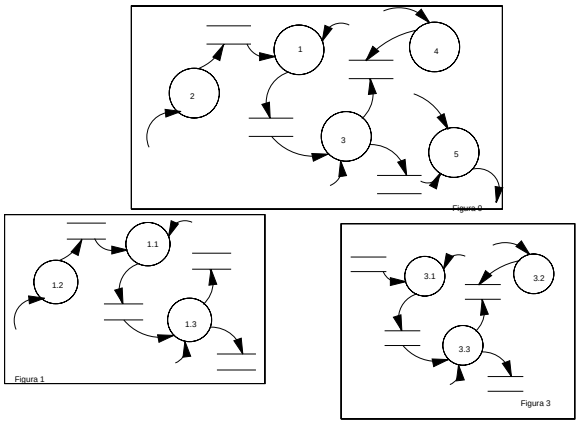


Para reorganizar o modelo, junte os processos de acordo com os depósitos de dados referenciados por eles.



Nivelando para cima

Represente cada grupo como uma transformação em um DFD de mais alto nível e junte os detalhes em diagramas filho". **Particione de forma a minimizar interfaces**.



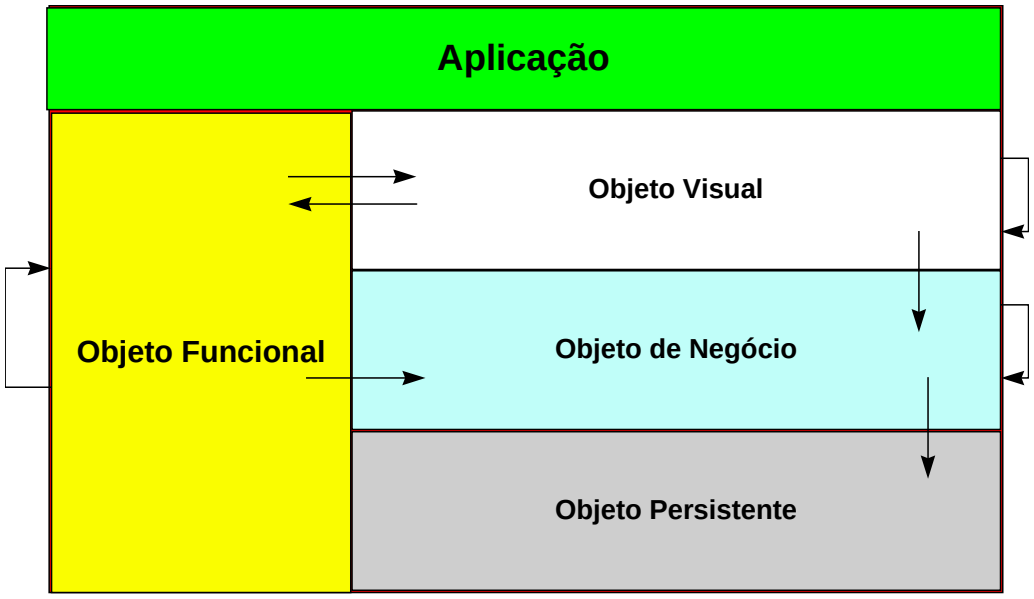
Capítulo 4

Orientação a Objetos

Objetos são estruturas conceituais que juntam dados e processos. A organização do modelo do sistema em torno dos objetos simplifica sua implementação e viabiliza a reutilização de código, tornando possível o conceito de fábrica de software.

Estrutura em camadas

A implementação do sistema será organizada em camadas de objetos com a finalidade de isolar a tecnologia da essência do sistema.



Durante a modelagem essencial do sistema, temos condições de projetar os objetos que são independentes de tecnologia, que são os objetos funcionais (que contém a lógica da resposta dos eventos) e os objetos de negócio, onde os dados e métodos são distribuídos. Os objetos visuais e os objetos persistentes serão projetados depois de definida a tecnologia, durante a modelagem de implementação.

Conceitos básicos

Objetos são empacotamentos que contém os **procedimentos** e os **dados relacionados**. Na maioria dos casos correspondem aos **objetos de negócio** do mundo real tais como: pedidos, faturas, produtos, máquinas, departamentos, pessoas.

Alguns princípios são fundamentais para se compreender os objetos:

- **Abstração**: um tipo de objeto representa um conjunto de características compartilhadas por outros

- **Encapsulamento**: empacotamento das operações e dados em um tipo de objeto só acessíveis através da interface do objeto. Disponibiliza publica e globalmente os acesso aos dados e operações: protocolo do objeto

- **Reusabilidade**: utilização de tipos de objetos já definidos no projeto e classes de objetos na implementação

- **Especialização**: herda operações, tipos de atributos e relacionamentos de um ou mais supertipos. Generalização de supertipos a partir de coisas comuns

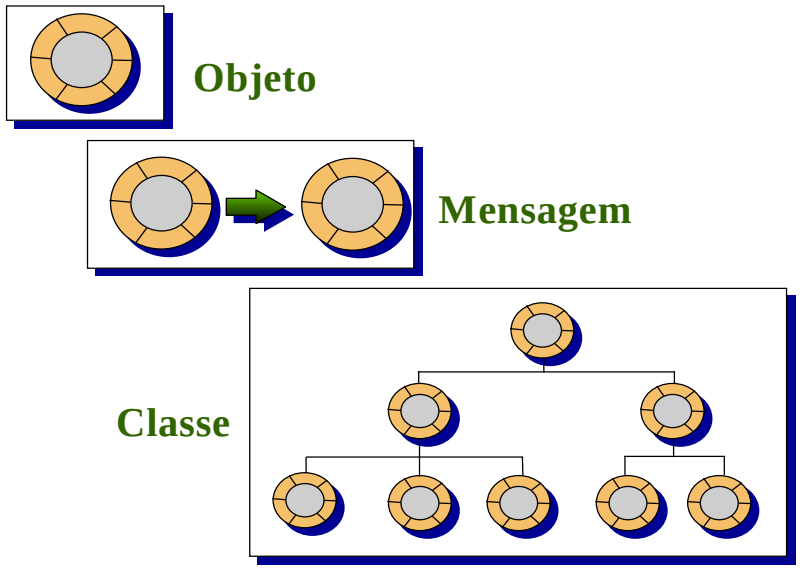
- **Comunicação**: solicitações a outros objetos através de mensagens ou requisições (ou chamadas) de operações

- **Polimorfismo**: quando uma mesma operação pode ser aplicada a diversos tipos de objetos diferentes

- ***Mensagens*** são os meios de comunicação entre objetos. Na essência, objetos solicitam **serviços** a outros objetos, para que juntos realizem uma operação essencial de negócio.

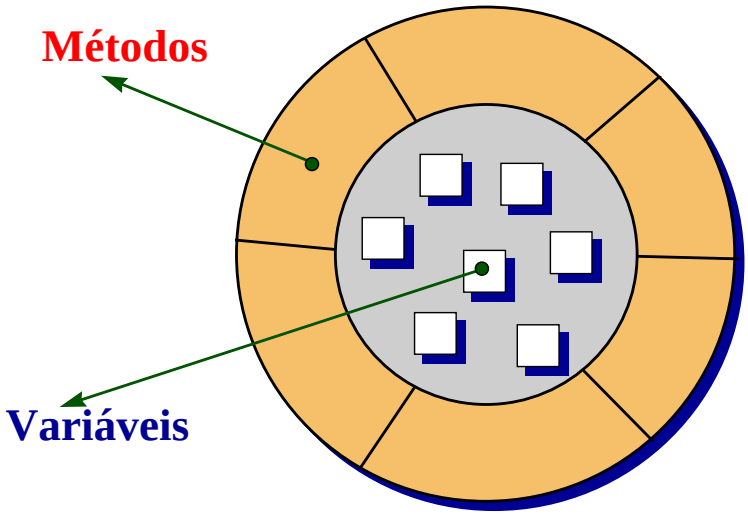
- ***Classes*** são como gabaritos para definição de tipos de objetos. Por exemplo, todo objeto Nota Fiscal poderia ser descrito como uma única classe de Notas Fiscais, onde estariam especificados todas as propriedades ou os atributos de todas as notas, assim como os procedimentos ou serviços possíveis de serem oferecidos por essas notas.

A ***idéia básica da tecnologia*** é a de ***replicar*** a forma como são construídos os equipamentos de *hardware*. Nesta analogia, o objeto se comporta como uma "caixa preta", equivalente aos circuitos integrados utilizados na construção dos referidos equipamentos.



- Ocultação da informação: Apenas o próprio objeto é capaz de ver, manipular e atualizar seus atributos;

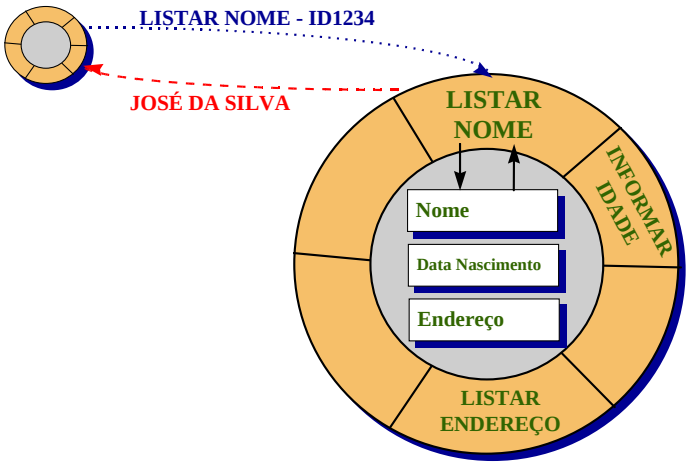
- Hereditariedade: As características e comportamentos do objeto são determinados pela herança adquirida de suas classes ancestrais.



Procedimentos ou Serviços são codificados em Métodos do objeto. Dados são armazenados em variáveis, atributos, propriedades ou características. Somente os métodos acessam as variáveis do objeto. Um objeto solicita a execução de um método

do outro objeto via mensagem, eventualmente, com parâmetros. O objeto solicitado responde com o dado contido na variável acessada pelo método.

Objetos Compostos.

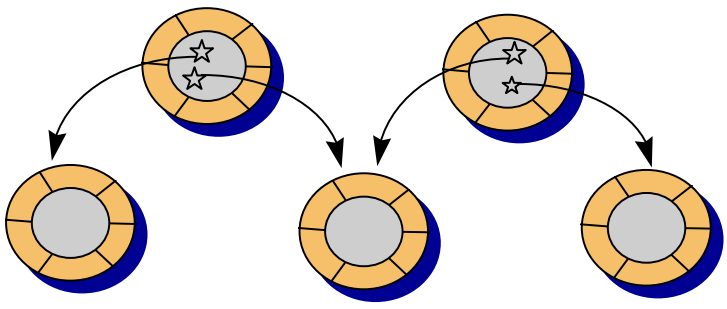


Os objetos podem "conter" outros objetos. Normalmente, a composição é implementada fazendo referência aos objetos contidos usando os *identificadores dos objetos*. Um *identificador* ou *handle* do objeto o torna único, mesmo que os valores de seus atributos sejam idênticos. Desta forma:

- Os objetos "contidos" podem mudar de tamanho e composição sem afetar o objeto que os contém. A manutenção de sistemas complexos fica muito mais simples do que de outra forma.

- Os objetos "contidos" podem participar de quaisquer objetos compostos permitindo uma forma vital de capturar informações do mundo real.

Uma ***mensagem*** é uma solicitação feita a partir de um objeto (*transmissor*) a outro



objeto (*receptor*) para que ele execute algum de seus métodos.

Uma mensagem é composta de três partes:

- A identificação do receptor da mensagem
- O método que está sendo solicitado ao receptor
- Informações adicionais que o receptor eventualmente necessite para a execução do método solicitado

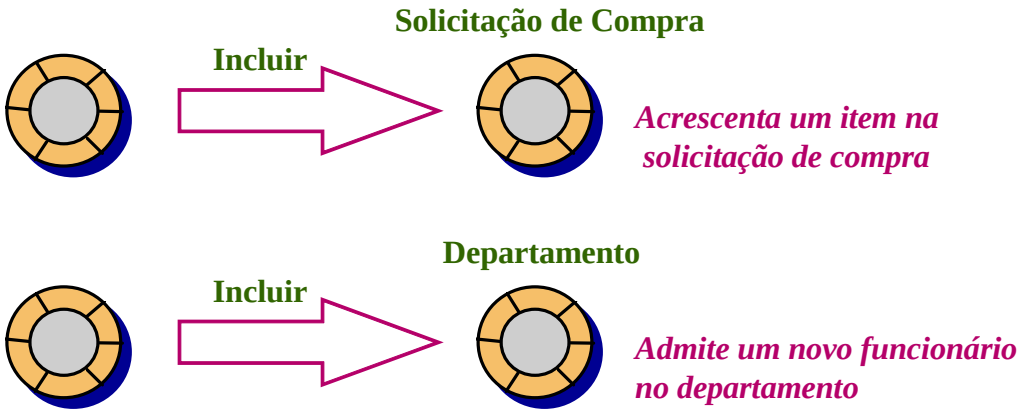
Polimorfismo

Como os objetos são definidos independentes dos outros, um ***mesmo nome*** pode ser utilizado para solicitar a execução de vários métodos diferentes de objetos diferentes.

Nenhum parâmetro de mensagem pode ser usado como ***argumento de controle***.

O polimorfismo ***elimina o uso*** de ***CASE*** (caso) ou ***IFs*** (se) encadeados.

Novos "casos" são criados pela construção de novos métodos de novos objetos.



Critérios para encontrar objetos de negócio

Objetos podem ser identificados a partir do Modelo Essencial do Sistema e são validados com os seguintes critérios:

- Podem ser vistos como "coisas" (substantivos), mesmo que descritos por verbos
- Contêm informação de alguma espécie
- Possuem procedimentos associados a eles
- Têm "casos" especiais ou poderão tê-los no futuro
- Compartilham propriedades com outros objetos em potencial

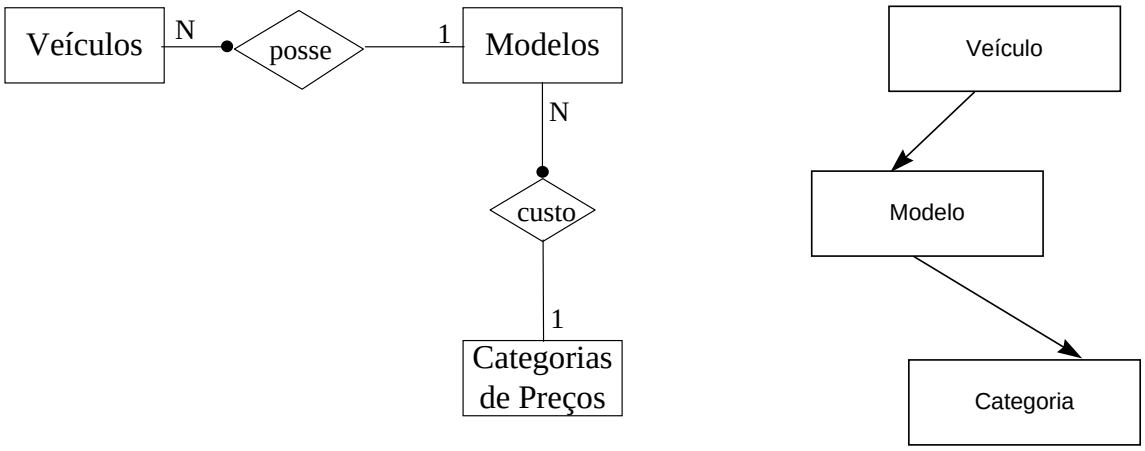
- Contêm "coisas" já definidas como objetos

Objetos de Negócio a partir do MER

Análise dos assuntos contidos no diagrama entidade relacionamento do evento.

Consideram-se:

- as cardinalidades
- as totalidades
- a obtenção de uma única instância do objeto



O modelo hierárquico do objeto de negócio mostra que Veículo é composto pelo objeto Modelo, que por sua vez é constituído pelo objeto Categoria.

O objeto funcional (lógica do evento) que lida com o objeto de negócio veículo deve fazer referência apenas ao Veículo. Modelo por sua vez é invocado por um método de Veículo e Categoria é instanciada por algum método de Modelo. Desta forma, a complexidade do objeto de negócio é escondida através de sua estrutura de composição, o que tornará a especificação muito mais simples para se especificar e entender.

Capítulo 5

Nossa intenção é escrever o mínimo de texto, para especificarmos o sistema de maneira formal e rigorosa.

Para isso, vamos, em um Dicionário de Dados, especificar:

- Composição de Dados (Grupos de Dados);
- Elementos de Dados;
- Entidades;
- Relacionamentos e
- Processos.

Especificação de Composição de Dados

Objetivo

- Definir sintaticamente a composição de dados não elementares.

Ocorrências

- Uma para cada depósito de dados e fluxo não elementar em um DFD e uma para cada entidade em um DER.

Exemplo

- Sócios = número_sócio + endereço_sócio + (fone_sócio) + (fone_comercial) + CPF_sócio + RG_sócio
- Dados_artísticos = nome_ator + {/participações/ nome_filme + gênero_filme + nome_personagem + ano_produção}
- Preferência = [cod_filme |ator | gênero | título]
- Dados_pessoais_alterados = <endereço_sócio | fone_sócio | fone_comercial>

Sintaxe

Símbolo	Significado
=	é composto de
+	e
{ }	várias ocorrências de
[ ]	apenas um dentre
<>	qualquer conjunto dentre
|	ou
/ /	nome (rótulo) de grupo repetitivo
( )	opcional
**	comentário
@	identificador

Especificação de Elemento de Dados

Objetivos

- documentar o significado de cada elemento;
- especificar o significado do item;
- especificar o domínio do elemento pela enumeração ou especificação de seus limites;
- especificar se o elemento é discreto ou contínuo e
- especificar a implementação física adotada em tempo de implementação.

Ocorrências

- uma para cada fluxo elementar no DFD, e
- uma para cada elemento em uma especificação de composição de dados.

Especificação de Entidade

Objetivos

- definir o significado da entidade através da descrição de seu papel no sistema;

- a especificação deve explicitar o critério de inclusão, ou seja, qual é a regra ou conceito que faz com que um determinado objeto seja considerado como pertinente a este conjunto;

- deve especificar, também, o critério de exclusão: o que faz com que um objeto deixe de pertencer a este conjunto e

- adicionalmente, se adequado, definir também os estados que uma ocorrência da entidade pode assumir, através dos eventos que a colocam nesses estados.

Ocorrências

- uma por entidade no DER.

Especificação de Relacionamento

Objetivos

- descrever o significado do relacionamento e

- especificar a cardinalidade e a totalidade do relacionamento.

Ocorrências

- uma por relacionamento no DER.

Especificação de Objetos

Especificação de Processo

Objetivo

- documentar a a composição dos objetos

Ocorrências

- uma para cada objeto.

Gabarito

Nome do Objeto

Atributos	**Serviços**
atributo1: tipo atributo2: tipo	**nomeDoMétodo1(argumento1: tipo,...) : tipo** minispec do método **nomeDoMétodo2(argumento1: tipo,...) : tipo** minispec do método

Exemplo

Categoria

Atributos	Serviços
categoria: Id valorHora: Valor valorMes: Valor cancelar: Booleano	**encontrar(categoria: Id) : Booleano** SE ENCONTRAR Categorias. COM .categoria=categoria RETORNE Verdadeiro SENAO RETORNE Falso **criar(categoria: Id, valorHora: Valor, valorMes: Valor) : Nulo** CRIE Categorias .codigo := categoria .valorHora := valorHora .valorMes := valorMes **excluir(categoria: Id) : Nulo** REMOVA Categorias

Considerações sobre Especificação de Processo

Fonte : Rápida e Limpa, FG&A

Cada processo deve ser completamente especificado pela ação executada para:

- manipular os fluxos de dados de entrada;
- produzir os fluxos de dados de saída e
- atualizar a memória do sistema.
- Para produzir especificações consistentes e sem ambigüidades, você pode usar NRDE's (elementos de dados não armazenados) e atributos. Por exemplo:
  - cliente.nome,
  - .endereço
  - fornecedor_código

Sempre que você usar um elemento de dados dentro de uma especificação de processo, ele será considerado **importado** pelo processo. Um atributo também é considerado **importado** se fizer parte do dicionário de dados.

De qualquer maneira, existem casos em que um elemento de dados é usado para manter algum resultado intermediário exclusivamente dentro do processo. Tais elementos de dados são chamados de **elementos de dados locais**. Como sugestão, inicie o nome de todos os elementos de dados locais com uma arroba (**@**). Por exemplo:

- @@e:= produto.valor * .quantidade
- produto.valor = @preco_por_unidade * pedido.quantidade

Expressões e Operadores

As expressões são escritas usando a notação pré-fixada convencional de linguagem de programação. Por exemplo:

- x+=y x := x + y

- x-=y x := x - y
- x*=y x := x * y
- x/=y x := x / y

Sempre que você atribuir um valor a um elemento de dados, ele é considerado **exportado** pelo processo. Um atributo também é **exportado** se ele fizer parte de algum fluxo de dados recebido pelo processo.

Para comparar valores pode-se usar os operadores **=**, **>>**, **<<**, **>>=**, **<<=** e **<<>>**, e, para as operações lógicas, os operadores **and**, **or** e **not**. Você pode (e deve) usar parênteses para agrupar sub-expressões.

Funções

Você pode usar funções usando um nome seguido por parênteses. Se houver argumentos, estes devem estar entre parênteses e separados por vírgula. Por exemplo:

- novo_saldo:=ajuste(saldo_anterior, fator_classificador)

Não existe pré-definição de funções. Quando se determina uma nova função, esta deve ser, necessariamente, definida como um processo.

Recebendo e Enviando Fluxos de Dados

A construção **receba** (*receive*) é usada para receber um fluxo de dados. Por exemplo:

- **receba** pedido_usuário

Você pode também receber parte de um fluxo de dados. Por exemplo:

- **receba** item **de** pedido

Neste caso, item deve ser rótulo de um grupo de dados.

A construção **produza** (*produce*) é usada para enviar fluxos de dados de saída ou parte deles. Por exemplo:

- **produza** nota_de_autorização e
- **produza** item **de** fatura

Criando e Removendo Ocorrências de Objeto

A construção **crie** (*create*) cria novas ocorrências de um objeto. Por exemplo:

- **crie** cliente.

Como os objetos são inter-relacionados, existe uma forma especial de utilizar a construção **crie,** que, da mesma maneira, especifica os atributos apontadores para alguns seletores de ocorrência. Por exemplo:

- **crie** fatura. **de** cliente., vendedor.

é exatamente igual a:

- **crie** fatura.
- fatura.cliente := cliente.
- fatura.vendedor:= vendedor.

A construção **remova** (*remove*) é usada para remover a ocorrência do objeto corrente. Por exemplo:

- **remova** fatura.

Encontrando uma Ocorrência do Objeto

Use a construção **encontre** (*find*) para encontrar uma ocorrência particular de um objeto. Por exemplo:

- **encontre** cliente. **com** .nome **=** nome_pedido

Você pode usar uma forma especial da construção **encontre,** para encontrar a ocorrência do objeto apontada por um outro:

- **encontre** cliente. **com** cliente. **=** fatura.cliente

Seleção

A construção **se** (*if*) permite especificar uma seleção entre alternativas:

- **se** cliente.saldo **>>** 100000
  crédito:="aprovado"
  f_se

- **se** cliente.saldo **<<**100000
  crédito := "rejeitado"
  se_não
  crédito := "aprovado"
  f_se

- **se** cliente.tipo = "especial"
  crédito := "aprovado"
  se_não_se cliente.saldo **>>** 5000

```
      crédito := "aprovado"
se_não_se cliente.saldo >> 1000
      crédito := "revisão"
se_não
    crédito:="rejeitado"
f_se
```

Construções de Repetição

A única construção de iteração permitida é a **para_cada** (*for_each*), que pode ser aplicada em diversas situações. O uso mais comum do **para_cada** é para especificar um grupo de ações, que deve ser executado para cada tipo do objeto:

- **para_cada** cliente.
  classificação := cliente.classe
  produza item do relatório
  f_para

Podemos especificar um critério de classificação:

- **para_cada** cliente. **com** .tipo = "revendedor"
  ou .tipo = "distribuidor"
  em_ordem_de cliente.cep ascendente,
  .nome descendente
  classificação:= cliente.classe
  produza item do relatorio
  f_para

Existe ainda uma forma de classificação manual para objetos inter-relacionados. Por exemplo:

- **para_cada** fatura. **de** cliente.
  .....
  f_para

- **para_cada** fatura. **de** cliente. **com** fatura.estado **=** "pendente"
  .....
  f_para

Finalmente, existe uma forma de manipular itens repetitivos em fluxos de dados de entrada:

- **para_cada** item **em** fornecedor_pedido
  .....
  f_para

Neste caso, item deve ser um rótulo de um grupo de dados.

Capítulo 5

Bibliografia

1. Análise Essencial de Sistemas
    - Stephen M. McMenamim / J. Palmer
2. Análise Estruturada Moderna
    - Edward Yourdon
3. Análise Estruturada e Especificação de Sistemas
    - Tom de Marco
4. Desenvolvendo Sistemas sem Complicação
    - Paul T. Ward
5. Análise Estruturada de Sistemas
    - Chris Gane / Trish Sarson