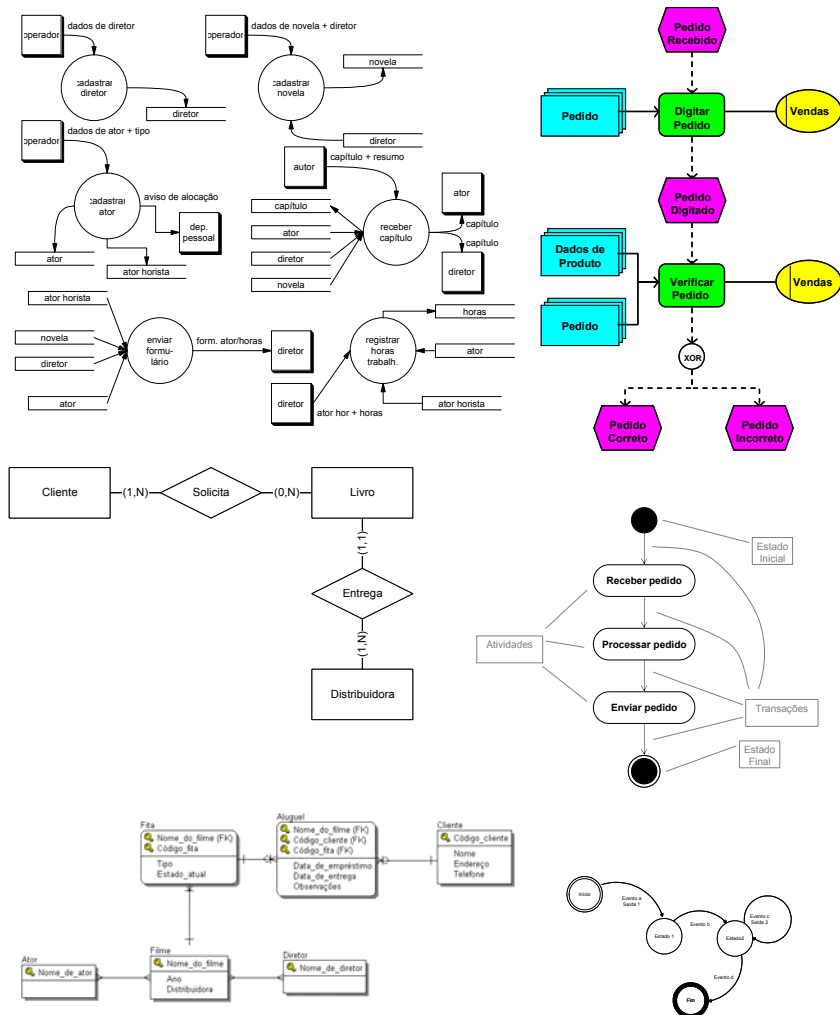


# Modelagem de Sistemas de Informação

## Análise Essencial Moderna

Edição 2004/1º Semestre

Geraldo Xexéo



## **Modelagem de Sistemas de Informação: Análise Essencial Moderna**

Geraldo Xexéo.

Copyright © 2004 Geraldo Xexéo

Todos os direitos reservados e protegidos pela lei 5988 de 14/12/73.

Nenhuma parte deste livro, sem autorização prévia por escrito dos autores, poderá ser reproduzida ou transmitida sejam quais forem os meios empregados: eletrônicos, mecânicos, fotográficos, gravação ou quaisquer outros.

Todo o esforço foi feito para fornecer a mais completa e adequada informação. Contudo, o autor não assume responsabilidade pelos resultados e uso da informação fornecida.

e-mail de contato: [xexeo@ufrj.br](mailto:xexeo@ufrj.br), [xexeo@acm.org](mailto:xexeo@acm.org)

# Sumário

<b>CAPÍTULO I. INTRODUÇÃO .....</b>	<b>1</b>
<b>CAPÍTULO II. SISTEMAS DE INFORMAÇÃO.....</b>	<b>4</b>
II.1    INTRODUÇÃO .....	4
II.2    DADOS, INFORMAÇÃO, CONHECIMENTO .....	5
II.3    SISTEMAS DE INFORMAÇÃO E A ORGANIZAÇÃO .....	6
II.3.1    Sistemas de Informações Típicos.....	8
II.4    TIPOS DE PROJETOS DE SISTEMAS DE INFORMAÇÃO.....	8
II.4.1    Porque são feitos projetos de SI.....	9
II.5    EXERCÍCIOS .....	10
<b>CAPÍTULO III. O DESENVOLVIMENTO DE SOFTWARE .....</b>	<b>12</b>
III.1    ANÁLISE .....	13
III.1.1    Modelos de Análise.....	13
III.1.2    Ferramentas da Análise.....	14
III.1.3    A Tecnologia.....	14
III.1.4    O Analista de Sistemas.....	14
III.2    CICLO DE VIDA E PROCESSO DE DESENVOLVIMENTO .....	15
III.2.1    A Necessidade de Garantir a Qualidade .....	15
III.2.2    Modelos de Processo mais Comuns.....	16
III.3    A EQUIPE DE DESENVOLVIMENTO .....	21
III.4    EXERCÍCIOS .....	23
III.4.1    Trabalho em Grupo (Projeto de Cadeira) .....	23
<b>CAPÍTULO IV. USUÁRIOS E REQUISITOS .....</b>	<b>25</b>
IV.1    STAKEHOLDERS E USUÁRIOS .....	25
IV.1.1    Perspectivas dos Usuários .....	27
IV.2    TIPOS DE REQUISITOS .....	28
IV.2.1    Requisitos Funcionais e de Informação .....	28
IV.2.2    Requisitos não funcionais.....	29
IV.2.3    Outros tipos de Requisitos.....	30
IV.3    REQUISITOS VERDADEIROS E FALSOS.....	31
IV.3.1    O Usuário não Sabe Tudo.....	32
IV.4    DESCREVENDO REQUISITOS.....	32
IV.4.1    Dependência de Requisitos .....	33
IV.4.2    Priorizando Requisitos.....	33
IV.5    MÉTODOS DE ELICITAÇÃO DE REQUISITOS .....	34
IV.5.1    A entrevista .....	36

IV.5.2	JAD .....	44
IV.6	EXERCÍCIOS .....	46
IV.6.1	Projeto 1: Livraria ABC.....	46
IV.6.2	Projeto de Curso .....	46
<b>CAPÍTULO V. UMA PROPOSTA INICIAL.....</b>		<b>47</b>
V.1	O PRIMEIRO CONTATO E OS PRIMEIROS RESULTADOS .....	47
V.2	A SOLICITAÇÃO DO CLIENTE .....	48
V.3	OBJETIVO DO SISTEMA.....	48
V.4	PROBLEMAS .....	49
V.5	OPORTUNIDADES .....	50
V.6	METAS E SUAS MÉTRICAS.....	51
V.6.1	Metas subjetivas .....	52
V.6.2	Levantando Objetivos e Metas .....	53
V.7	OS REQUISITOS PRELIMINARES.....	53
V.7.1	Definindo o Escopo .....	53
V.8	O SISTEMA ATUAL.....	54
V.8.1	Problema do sistema atual .....	54
V.9	VISÃO DO NOVO SISTEMA .....	55
V.9.1	Oportunidades para o novo sistema.....	55
V.9.2	Pontos Críticos ou Pontos Chave.....	55
V.9.3	Restrições .....	56
V.10	GERÊNCIA DE RISCOS .....	56
V.11	CONSTRUINDO UM GLOSSÁRIO .....	58
V.12	ENTENDENDO O QUE SERÁ FEITO .....	59
V.13	PROPOSTA COMERCIAL.....	59
V.13.1	Calculando Preço e Custo.....	59
V.14	O RESUMO EXECUTIVO.....	60
V.15	ESTRUTURA DA PROPOSTA INICIAL.....	61
V.16	EXERCÍCIO .....	62
V.16.1	Projeto 1: Livraria ABC.....	62
V.16.2	Projeto de curso .....	62
<b>CAPÍTULO VI. MODELO DE NEGÓCIO .....</b>		<b>63</b>
VI.1	O ORGANOGRAMA.....	64
VI.2	MODELAGEM DE FUNÇÕES DE NEGÓCIO (IDEF0) .....	65
VI.3	PROCESSOS DE NEGÓCIO.....	74
VI.3.1	Fluxogramas .....	75
VI.3.2	EPC .....	75

VI.3.3	<i>Diagramas de Atividade</i> .....	81
VI.3.4	<i>Diagramas de Raias</i> .....	83
VI.4	REGRAS DE NEGÓCIO.....	84
VI.4.1	<i>Declarações Estruturais</i> .....	85
VI.4.2	<i>Declarações de Ações (ou Restrições)</i> .....	86
VI.4.3	<i>Derivações</i> .....	86
VI.4.4	<i>Descrevendo Regras de Negócio</i> .....	87
VI.5	OUTRAS FERRAMENTAS DE MODELAGEM.....	90
VI.5.1	<i>Responsáveis por decisão (Processo x Organização)</i> .....	90
VI.5.2	<i>Dados x Processos (CRUD)</i> .....	91
VI.5.3	<i>Corrente/Planejado</i> .....	92
VI.6	RESUMO DA MODELAGEM DE NEGÓCIO.....	92
VI.7	EXERCÍCIOS .....	92
VI.7.1	<i>Projeto 1: Livraria ABC</i> .....	92
<b>CAPÍTULO VII. MODELO CONCEITUAL DE DADOS .....</b>		<b>93</b>
VII.1	MODELOS E ABSTRAÇÕES.....	94
VII.1.1	<i>Classificação (é um membro de)</i> .....	94
VII.1.2	<i>Composição ou Agregação (é feito de)</i> .....	95
VII.1.3	<i>Generalização (é um, é como)</i> .....	95
VII.1.4	<i>Identificação (é identificado por)</i> .....	95
VII.1.5	<i>Trabalhando com as abstrações</i> .....	96
VII.2	A MEMÓRIA DO SISTEMA.....	96
VII.2.1	<i>Modelo Conceitual (MC)</i> .....	96
VII.2.2	<i>Modelo Lógico</i> .....	97
VII.2.3	<i>Modelo Físico</i> .....	97
VII.3	MODELO DE ENTIDADES E RELACIONAMENTOS.....	98
VII.4	O DIAGRAMA DE ENTIDADES E RELACIONAMENTOS .....	100
VII.4.1	<i>Exemplo de Modelo E-R</i> .....	102
VII.5	DESENVOLVENDO O MODELO CONCEITUAL .....	103
VII.6	ENTIDADES .....	104
VII.6.1	<i>Onde encontrá-las</i> .....	106
VII.6.2	<i>Descrevendo Entidades</i> .....	107
VII.7	RELACIONAMENTOS.....	108
VII.7.1	<i>Cardinalidade</i> .....	111
VII.7.2	<i>Descrevendo Relacionamentos</i> .....	116
VII.8	ATRIBUTOS .....	117
VII.8.1	<i>Descrevendo Atributos</i> .....	117

VII.9	IDENTIFICANDO ENTIDADES .....	118
VII.9.1	<i>Atributos Identificadores (Chaves Candidatas e Chaves Primárias)</i> .....	118
VII.9.2	<i>Relacionamentos Identificadores</i> .....	119
VII.10	DESCRIÇÃO GRÁFICA DO MODELO .....	120
VII.10.1	<i>Exemplos de notações</i> .....	122
VII.10.2	<i>Notação adotada</i> .....	125
VII.11	TÉCNICAS DE DESENVOLVIMENTO DO MODELO .....	125
VII.11.1	<i>Técnica Top-Down</i> .....	125
VII.11.2	<i>Técnica Bottom-Up</i> .....	129
VII.11.3	<i>Técnica Inside-Out</i> .....	129
VII.11.4	<i>Técnica Mista</i> .....	130
VII.11.5	<i>Que técnica usar</i> .....	130
VII.11.6	<i>Equivalência de modelos</i> .....	130
VII.12	REPRESENTANDO O ASPECTO TEMPORAL .....	130
VII.13	FORMAS NORMAIS .....	131
VII.13.1	<i>Primeira Forma Normal (1FN)</i> .....	131
VII.13.2	<i>Algumas Anomalias Resolvidas pelas Formas Normais</i> .....	133
VII.13.3	<i>Segunda Forma Normal (2FN)</i> .....	134
VII.13.4	<i>Terceira Forma Normal</i> .....	134
VII.13.5	<i>Outras formas normais</i> .....	135
VII.13.6	<i>Formas Normais e o Modelo E-R</i> .....	136
VII.14	OUTROS TERMOS.....	136
VII.15	VERIFICANDO O MODELO .....	136
VII.16	LEITURA COMPLEMENTAR .....	137
<b>CAPÍTULO VIII. MODELO FUNCIONAL ESSENCIAL .....</b>		<b>138</b>
VIII.1	PERSPECTIVA HISTÓRICA .....	138
VIII.2	A ANÁLISE ESSENCIAL.....	139
VIII.3	OS PRINCÍPIOS DA MODELAGEM ESSENCIAL .....	139
VIII.3.1	<i>O Orçamento para a Complexidade</i> .....	140
VIII.3.2	<i>A Neutralidade Tecnológica</i> .....	140
VIII.3.3	<i>A Tecnologia Interna Perfeita</i> .....	141
VIII.3.4	<i>O Modelo Essencial Mínimo</i> .....	141
VIII.3.5	<i>O Princípio da Ausência de Surpresas</i> .....	142
VIII.4	A ESSÊNCIA .....	142
VIII.5	AGENTES EXTERNOS.....	144
VIII.6	EVENTOS ESSENCIAIS .....	145
VIII.6.1	<i>Tipos de Eventos</i> .....	147

VIII.6.2	<i>Um Exemplo Esclarecedor</i> .....	149
VIII.6.3	<i>Habilitando Eventos</i> .....	150
VIII.7	DESCREVENDO EVENTOS ESSENCIAIS .....	150
VIII.8	LEVANTANDO OS EVENTOS ESSENCIAIS .....	152
VIII.8.1	<i>Classificando os Eventos</i> .....	153
VIII.8.2	<i>Encontrando Eventos</i> .....	154
VIII.8.3	<i>Simplificando Eventos</i> .....	155
VIII.9	AS RESPOSTAS AOS EVENTOS .....	155
VIII.9.1	<i>Confundindo eventos e respostas</i> .....	155
VIII.10	A MEMÓRIA DO SISTEMA .....	156
VIII.10.1	<i>Eventos x Entidades (Matriz CRUD)</i> .....	156
VIII.10.2	<i>Verificando a consistência</i> .....	157
VIII.10.3	<i>Manipulando a Matriz CRUD</i> .....	157
VIII.10.4	<i>Extensões da Matriz CRUD</i> .....	158
VIII.11	ENTENDENDO UM EVENTO .....	159
VIII.12	O DICIONÁRIO DE EVENTOS .....	160
VIII.13	O DIAGRAMA DE FLUXO DE DADOS .....	162
VIII.13.1	<i>Algumas regras sintáticas</i> .....	165
VIII.13.2	<i>Entendendo os Fluxos de Dados</i> .....	165
VIII.13.3	<i>Entendendo as Memórias</i> .....	166
VIII.13.4	<i>Tipos de DFD</i> .....	166
VIII.13.5	<i>Criando o DFD Particionado</i> .....	167
VIII.13.6	<i>Criando o DFD hierárquico</i> .....	167
VIII.14	ESPECIFICANDO PROCESSOS .....	168
VIII.14.1	<i>Especificação do Tipo Caixa Preta</i> .....	169
VIII.14.2	<i>Especificação do Tipo Caixa Branca</i> .....	169
VIII.14.3	<i>Mini-especificações</i> .....	169
VIII.14.4	<i>O Diagrama de Transição de Estados</i> .....	171
VIII.14.5	<i>Tabelas de Decisão</i> .....	172
VIII.14.6	<i>Pré-condições e pós-condições</i> .....	173
VIII.15	DICIONÁRIO DE DADOS .....	173
VIII.16	EXERCÍCIO .....	174
VIII.16.1	<i>Rede Bobo de Televisão</i> .....	174
VIII.17	MODELOS ADICIONAIS .....	176
VIII.17.1	<i>Finalizando a Análise</i> .....	176
<b>CAPÍTULO IX. MODELO DE INTERFACE</b> .....		<b>177</b>
IX.1	A INTERAÇÃO HOMEM COMPUTADOR (IHC) .....	177

IX.2	PROJETANDO A INTERFACE COM O USUÁRIO .....	178
IX.2.1	<i>Introdução ao Modelo Cognitivo usado em IHC</i> .....	179
IX.3	COLETANDO INFORMAÇÕES SOBRE O USUÁRIO .....	179
IX.3.1	<i>Questionário para identificação de usuários</i> .....	179
IX.4	PROTÓTIPOS .....	180
IX.4.1	<i>Protótipos de Baixa Fidelidade &amp; StoryBoarding</i> .....	182
IX.5	MODELOS DE NAVEGAÇÃO .....	186
IX.5.1	<i>O Diagrama de Navegação de Telas</i> .....	187
<b>CAPÍTULO X. QUAL O TAMANHO DO SOFTWARE .....</b>		<b>189</b>
X.1	QUAL O TAMANHO DO SISTEMA? .....	189
X.2	PREÇO E CUSTO .....	190
X.3	ESFORÇO E TEMPO DE DESENVOLVIMENTO .....	190
X.4	O TAMANHO DO SOFTWARE .....	191
X.5	UMA VISÃO REDUZIDA DO MODELO COCOMO II.....	191
X.5.1	<i>Distribuição do Esforço</i> .....	193
X.6	ANÁLISE DE PONTOS DE FUNÇÃO .....	193
X.6.1	<i>Procedimento de Contagem</i> .....	194
X.6.2	<i>Identificando Funções de Negócio</i> .....	195
X.6.3	<i>Determinando a Complexidade</i> .....	199
X.6.4	<i>As Perguntas</i> .....	200
X.6.5	<i>Cálculo dos Pfs Finais</i> .....	202
X.6.6	<i>Contando PFs na Análise</i> .....	202
X.6.7	<i>Inflação de PFs ao decorrer do projeto</i> .....	204
X.6.8	<i>Utilizando Pfs para previsões</i> .....	204
X.7	LIGANDO COCOMO II E PONTOS DE FUNÇÃO .....	205
X.8	ESTIMANDO O TAMANHO.....	207
X.8.1	<i>A Técnica de Delfos</i> .....	207
X.8.2	<i>Cenários</i> .....	208
X.9	VERIFICANDO A SANIDADE DA ESTIMATIVA .....	208
<b>CAPÍTULO XI. PROJETO 1: LIVRARIA ABC.....</b>		<b>209</b>
XI.1	ENTREVISTA 1.....	209
XI.2	ENTREVISTA 2.....	209
XI.3	ENTREVISTA 3.....	210
<b>CAPÍTULO XII. PROJETO 2: EMPRESA DE CLIPPING CLIPTUDO .....</b>		<b>212</b>
<b>CAPÍTULO XIII. BIBLIOGRAFIA .....</b>		<b>213</b>
<b>CAPÍTULO XIV. ÍNDICE .....</b>		<b>216</b>



# Capítulo I. Introdução

---

Este livro é sobre a Modelagem de Sistemas de Informação seguindo uma forma bastante atualizada da Análise Essencial, unificada com outros métodos importantes no dia a dia do desenvolvedor de software, como o Modelo de Entidades e Relacionamentos.

Ele é fruto da experiência de 10 anos ensinando Análise de Sistemas para graduação, primeiro na Universidade Estácio de Sá e depois na Universidade Federal do Rio de Janeiro, já como professor Adjunto. O texto foi criado, alterado, cortado e aumentado de forma a atender as necessidades do curso, do mercado e dos alunos. Muito do conteúdo aqui apresentado é resultado direto de dúvidas e dos erros mais comuns que os alunos apresentam durante o curso. A matéria apresentada também é resultado de 15 anos de atuação como consultor, envolvido direta ou indiretamente na análise e implementação de sistemas em diferentes projetos, sobre temas variados como administração pública, gerência de satélites, controle de caminhões, etc.

O livro foi construído com dois propósitos. O primeiro é apoiar um primeiro curso de modelagem de sistemas de informação, fundamentado na análise essencial, no nível de graduação ou extensão, visando formar um analista de sistemas. O segundo é fornecer uma base de sustentação para o analista no seu dia a dia no trabalho. Não apresentamos um método único de análise ou especificação de sistemas, mas sim um conjunto de ferramentas que podem ser usadas tanto em conjunto como em separado, porém baseadas em uma filosofia comum. Este livro não trata de modelagem orientada a objeto, que é normalmente assunto de um segundo curso de modelagem de sistemas.

Algumas premissas foram importantes na construção do texto. Não é um texto com foco teórico, mas sim aplicado. Durante os 10 anos em que o curso já foi dado, todas as provas foram práticas, apresentando problemas para serem analisados e modelados, e com consulta.

Neste livro, seguindo a análise essencial, estamos interessados em sistemas de informação que produzem **respostas planejadas**, isto é, que executam ações pré-programadas em função de mudanças reconhecíveis e previsíveis no ambiente externo. Chamamos essas mudanças no estado do ambiente de **eventos essenciais**. Não estamos interessados em respostas *ad-hoc*, isto é, respostas que tem que ser praticamente inventadas caso a caso, mas sim em eventos que podem ser previstos e para os quais podemos programar respostas em um computador digital.

É importante deixar claro que a análise essencial, unificada com o modelo de entidades e relacionamento, é uma ferramenta de grande qualidade para o desenvolvimento de sistemas de informação.

Quando um cliente solicita um sistema de informação, pensa em automação de algum processo, na modernização de um processo já automatizado ou na criação de um novo processo em sua empresa. O cliente então imagina um sistema que executa algumas funções, gerencia alguns dados e fornece alguns relatórios. Esse sistema imaginado pelo cliente, porém, raramente descreve de forma clara o que ele realmente precisa, ou que precisará no dia que o sistema ficar pronto. Cabe ao analista ajudar ao cliente a descobrir como é o sistema realmente necessário.

O segundo capítulo faz uma pequena introdução aos Sistemas de Informação, principalmente aqueles que encontramos dentro de organizações. O terceiro capítulo discute o desenvolvimento de software. Ambos os capítulos são introdutórios e têm como finalidade nivelar conhecimento, sendo que o ideal é que o aluno envolvido nesse estudo tenha feito antes desse curso cadeiras equivalentes a Sistemas de Informação e Introdução a Engenharia de Software.

O quarto capítulo discute o levantamento de requisitos do sistema. Apresenta ao leitor dois conceitos importantes: o *stakeholder*, palavra que significa “aquele que tem algum interesse (aposta)”, e que traduzimos de forma liberal para interessado, e uma classificação de tipos de requisitos de um sistema, onde se sobressaem os requisitos funcionais e não funcionais. O capítulo também apresenta uma leve introdução a métodos de elicitação de requisitos, discutindo os métodos tradicionais e mais comuns, a entrevista e o JAD, detalhadamente.

O quinto capítulo apresenta a proposta inicial. O objetivo é permitir ao desenvolvedor compreender de forma geral o que será feito no projeto de desenvolvimento, dando margem para a criação de uma proposta comercial. Nesse capítulo ainda tratamos o desenvolvimento de sistemas como uma atividade informal, a nível de negócios.

O sexto capítulo trata da modelagem de negócios, partindo de modelos de alto grau de abstração, como o IDEF0, para modelos detalhados do comportamento da organização, como EPC e regras de negócio. Esses métodos podem substituir na sua totalidade o uso de DFDs para modelar a encarnação de um sistema, sendo na prática mais adequados para o mapeamento do funcionamento real de uma organização.

O sétimo capítulo trata da modelagem conceitual de dados, baseado no modelo de entidades e relacionamentos, necessidade primordial para uma boa análise de sistemas. O oitavo capítulo trata da modelagem funcional essencial. Juntos, esses dois modelos definem o funcionamento esperado do novo sistema. Esses modelos são completados com um modelo da interface do sistema, preferivelmente na forma de um protótipo, que é discutido no capítulo nove.

Finalmente o capítulo dez trata de formas de prever o esforço necessário para desenvolver um sistema de software, usando como fator de determinação os documentos previamente gerados.

Além disso, apresentamos ao final alguns projetos para os alunos usarem nos exercícios.

Recomenda-se que os capítulos sejam apresentados na ordem em que estão no livro. Os capítulos de análise funcional e modelagem de dados já foram dados em diferentes ordens, tanto um quanto o outro na frente, porém a análise de dados independente da análise funcional e o inverso não é verdade, o que recomenda que seja a ordem adotada. Além disso, a modelagem de dados pode ser vista como uma forma de detalhar, a nível do sistema, as regras de negócio, o que apresenta uma continuidade ao curso. O livro suporta bem um período de 15 semanas, com 4 horas por semana, incluindo aulas teóricas e exercícios, duas provas e um trabalho por equipe que tem em média 15

eventos e 10 entidades, e que deve modelar de forma completa um sistema, de acordo com todo material apresentado.

# Capítulo II. Sistemas de Informação

---

Sistemas de Informação
Dados, Informação, Conhecimento
SI e a Organização
ERP

## II.1 Introdução

Sistemas de Informação são utilizados em empresas e outros tipos de organizações para planejamento, monitoração, comunicação e controle das suas atividades, por meio da manipulação e guarda de informações. Segundo o Dicionário Aurélio, a palavra sistema significa, entre outras coisas, um “Conjunto particular de instrumentos e convenções adotados com o fim de dar uma informação”. Os instrumentos são as ferramentas que utilizamos para fazê-los funcionar, como: programas de computador, relatórios, formulários, etc. As convenções são as suas regras de utilização. Apesar de sistemas de informação não necessitarem de computadores para existir, hoje em dia é comum associar o termo imediatamente a uma implementação usando software, hardware e redes.

Um exemplo típico de sistema de informação é um sistema de aluguel para uma vídeo-locadora. Entre suas várias finalidades, a principal é certamente controlar o aluguel das fitas, informando quem está com qual fita em um determinado momento (quando), e quanto deve pagar por isso. Além disso, o sistema permite outras atividades, como a gerência do estoque de fitas, a monitoração das fitas mais e menos alugadas, etc.

É importante entender que sistemas de informação são **sistemas interativos e reativos**. Isso significa que interagem com um ambiente externo, normalmente sendo utilizados por pessoas, reagindo a mudanças que podemos identificar como relevantes ao seu funcionamento. A metodologia de desenvolvimento apresentada nesse livro é feita sob medida para esse tipo de sistema.

<p><b>Um Sistema de Informação é um conjunto de componentes inter-relacionados que coleta dados no ambiente em que opera, usando recursos de sensoriamento e telecomunicações (entrada), analisa esses dados (processamento) e finalmente apresenta o produto como informação útil (saída), sendo construído de forma a atender os interesses de uma organização, de seus clientes internos e externos e de todos aqueles atingidos direta ou indiretamente pelo novo produto<sup>1</sup>.</b></p>
--

---

<sup>1</sup> Xexéo, J.A. “Tese de Doutorado” XXX

## II.2 Dados, Informação, Conhecimento

Antes de entender o que é um Sistema de Informação, é preciso entender melhor o que significa a palavra **Informação**.

Novamente, vamos recorrer a dicionários para ter uma definição inicial. Segundo o American Heritage, informação é o dado quando processado, guardado ou transmitido. Já no dicionário Aurélio, informação, entre outros significados, pode ser “Conhecimento amplo e bem fundamentado, resultante da análise e combinação de vários informes”, “Coleção de fatos ou de outros dados fornecidos à máquina, a fim de se objetivar um processamento” ou ainda “Segundo a teoria da informação, medida da redução da incerteza, sobre um determinado estado de coisas, por intermédio de uma mensagem”. Apesar de não estarmos diretamente envolvidos com a teoria da informação, não podemos deixar de notar a importância da definição que diz que **a informação reduz a incerteza por meio de uma mensagem**.

Estamos interessados em criar uma diferenciação entre dados, informação e conhecimento, mesmo que as palavras possam ser consideradas sinônimas em muitos contextos. Apesar de serem normalmente confundidas ou utilizadas de forma intercambiável, elas podem ser mais bem entendidas e utilizadas se analisadas como representando conceitos diferentes.

**Dados** são apenas os símbolos que usamos para representar a informação, o registro de diferentes aspectos de um fato ou fenômeno. Os números que guardamos em um banco de dados são, como diz o nome, “dados”. Dados não são interpretados, eles existem, são adquiridos de alguma forma, guardados de outra forma e, possivelmente, apresentados em uma terceira. O computador é uma máquina que manipula dados.

Por outro lado, **informação** é o dado com significado, normalmente processado de forma a ser útil. Uma informação deve permitir responder perguntas como “quando”, “quanto”, “quem”, “qual” e “onde”<sup>2</sup> sobre alguma coisa.

$\text{Informação} = \text{Dado} + \text{Significado}$
--

É necessário fazer um mapeamento entre dados e informação. Esse mapeamento pode ser simples ou complexo, dependendo de várias variáveis envolvidas, que vão desde decisões arbitrárias tomadas pelo desenvolvedor até padrões internacionais. Por exemplo, em muitos sistemas é preciso ter a informação do sexo de uma pessoa (masculino ou feminino). Para isso, guardados um número (1 ou 0) ou uma letra (M ou F) que é o dado que faz a indicação da informação.

**Conhecimento** é a aplicação da informação. Podemos dizer que permite responder a pergunta “como”, pois envolve argumentos, explicações e justificativas. Entre as três palavras que estamos analisando, conhecimento é a que tem significado mais geral na língua portuguesa, podendo ser usada no dia a dia como sinônimo de informação. Em informática, porém, normalmente chamamos de conhecimento algo que pode ser escrito na

---

<sup>2</sup> What, where, when, who, How much

forma de regras (como em “se for maior de 18 anos, então pode tirar carteira de motorista”).

Além disso, ainda podemos analisar as palavras compreensão (ou entendimento) e sabedoria. A compreensão permite responder a pergunta “por que”, enquanto a sabedoria é um processo complexo de compreensão e avaliação do entendimento.

Bellinger et. al. Afirnam que com o aumento da compreensão e da capacidade de fazer conexões entre partes (dados, informações, etc.), passamos do trabalho direto com o dado em si para informação, então para o conhecimento e finalmente para a sabedoria.

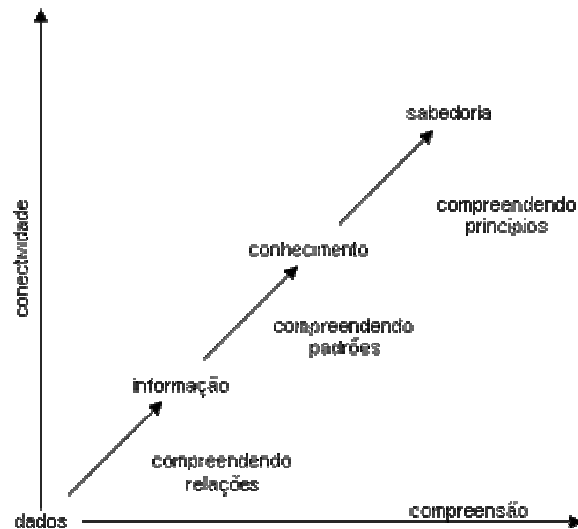


Figura 1. Entendimento das relações entre dados, informação, conhecimento e sabedoria, segundo Bellinger et al.[1]

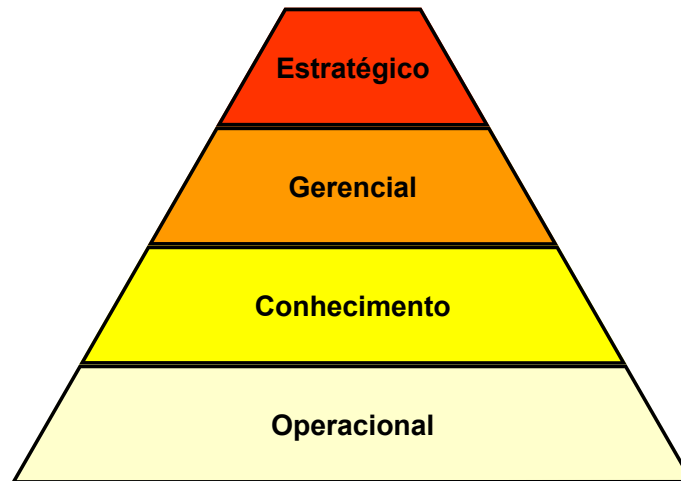
## II.3 Sistemas de Informação e a Organização

Sistemas de informação atualmente servem em todas as áreas e níveis das organizações, sendo considerados como ferramenta essencial para o sucesso de suas atividades. Dessa forma, são usados por diferentes tipos de funcionários de uma empresa. Isso nos permite classificá-los de acordo com a responsabilidade assumida por seus usuários dentro da organização em quatro tipos principais, como sugerido por Laudon [2]:

- **Sistemas de nível operacional**, que tratam da execução, acompanhamento e registro da operação diária da empresa, sendo geralmente sistemas fortemente transacionais. Exemplos são sistemas de vendas, folha de pagamento, etc.
- **Sistemas de nível de conhecimento**, que suportam as pessoas que trabalham com dados e conhecimento dentro da organização. Exemplos simples de sistemas desse tipo são os processadores de texto e as planilhas eletrônicas.
- **Sistemas de nível gerencial**, que utilizam dados da operação e outros dados inseridos nesses sistemas para permitir a obtenção de informações

que permitam a gerência da empresa, suportando a tomada de decisões, o controle e o monitoramento.

- **Sistemas de nível estratégico**, que são sistemas destinados a decisões de mais alto nível (efeito estratégico) e utilizam dados de todos os sistemas anteriores, normalmente de forma agregada e processada, sendo utilizados pela alta gerência.



**Figura 2. Níveis dos sistemas de informação dentro de uma organização**

Ainda segundo Laudon, a cada nível de sistemas de informação podemos associar um ou mais tipos de sistemas.

- **Sistemas de Suporte Executivo (SSE)**, encontrados no nível estratégico, são destinados a apoiar a alta gerência em tarefas estratégicas, como o planejamento de longo prazo. Usam dados fortemente agregados, internos e externos a organização e são capazes de responder perguntas específicas ou ainda fazer projeções. Podem ser capazes de fazer simulações e ter uma interface interativa.
- **Sistemas de Apoio a Decisão (SAD)**, encontrados no nível gerencial, são utilizados pelos vários níveis de gerência e utilizam como entrada dados em pequeno volume (agregações) ou ainda bases massivas de dados previamente preparadas para permiti-la atividades de análise de dados. Como resposta, devem fornecer relatórios específicos, análises e decisões e respostas a perguntas *ad-hoc*.
- **Sistemas de Informação Gerencial (SIG)**, também encontrados no nível gerencial, são utilizados pelos vários níveis de gerência. Utilizam grande volume de dados ou sumários de transações e modelos simples para obter relatórios sumários (agregados) e de exceções.
- **Sistemas de Trabalho com Conhecimento (STC)**, encontrados no nível de conhecimento, utilizam projetos, especificações e bases de

conhecimento em geral para produzir modelos e gráficos. Normalmente são utilizados por profissionais com nível superior.

- **Sistemas de Escritório (SE)**, encontrados no nível de conhecimento, tem como objetivo aumentar a produtividade na manipulação de dados em um escritório. Permitem a manipulação de documentos, correio eletrônico e agendas.
- **Sistemas Processamento de Transações (SPT)**, encontrados no nível operacional, tratam eventos e transações e fornecem relatórios detalhados, listas e sumários, utilizados pelos gerentes, além de documentos específicos para a transação em que são utilizados.

Os SPT suportam não só a operação diária da empresa, mas também criam os dados que são mais tarde utilizados pelos outros tipos de sistemas.

### **II.3.1 Sistemas de Informações Típicos**

Atualmente já consideremos que vários sistemas de informações típicos de uma empresa são necessidades básicas que podem ser atendidas de uma só vez. Esses sistemas constroem o que é comumente chamado de ERPs – de Enterprise Resource Planning – ou Sistemas de Gestão Empresarial em português – mas que na prática não são sistemas de planejamento (ou de recursos), mas sim de controle e administração de uma empresa.

Entre as características encontradas em ERPs podemos citar a integração das atividades da empresa e o uso de um banco de dados único. O líder mundial do mercado é a SAP AG, com o produto SAP R/3. O custo de implantação de um ERP de grande porte pode chegar até 300 milhões de dólares. No Brasil, existem produtos menos ambiciosos e mais baratos.

Os sistemas de ERP atuais contêm módulos representando os mais típicos sistemas de informações necessários em uma empresa, tais como: Contabilidade Fiscal, Contabilidade Gerencial, Orçamento e Execução Orçamentária, Ativo Fixo, Caixa e bancos, Fluxo de Caixa, Aplicações e Empréstimos, Contas a Receber, Contas a Pagar, Controle de Viagens, Controle de Inadimplência, Administração dos preços de venda, Compras, Controle de fretes, Controle de contratos, Controle de investimentos, Cotações de vendas, Estoque, Exportação, Faturamento, Gerenciamento de armazéns, Importação, Obrigações fiscais, Pedidos, Previsão de vendas, Recebimento, Gestão de informação de RH, Pagadoria, Treinamento, RH scorecard, Planejamento de RH, Planejamento de produção, Planejamento da capacidade, Custos industriais, Controle de chão de fábrica, Controle da produção, Configurador de produtos, Planejamento de Manutenção, Acompanhamento de Manutenção e ainda muitos outros...

## **II.4 Tipos de Projetos de Sistemas de Informação**

Existem três tipos de projeto de sistemas de informação: manual, manual para automático e re-automação [3]. Os processos de re-automação ainda podem se dividir em recodificação, re-projeto e re-desenvolvimento, melhoria ou manutenção.



Todos esses tipos de projeto apresentam ao analista de sistemas o mesmo desafio: descobrir o que deve ser feito. Porém, cada tipo apresenta certas particularidades que facilitam ou dificultam esse trabalho de análise.

O trabalho do analista em sistemas manuais é mais relacionado à formalização, por meio de documentação e padrões, de processos já adotados, a criação de novos processos e a transformação de processos existentes tendo em vista otimizá-los ou possibilitar que atendam novas necessidades da organização. Esses processos podem ser bastante complexos e convolutos em alguns casos, o que exige do analista uma boa capacidade de compreensão e modelagem. Porém, como não serão transformados em programas de computador, o analista pode trabalhar com ferramentais mais informais e mais próximas ao dia a dia do usuário.

Os projetos que apresentam maior dificuldade são os de passagem do processo manual para o automático. Isso acontece porque normalmente esses projetos exigem todo o trabalho feito no tipo anterior, e, de forma adicional, a criação de um modelo computacional e com certo grau de formalidade, que possa ser usado pelos desenvolvedores. Não há, a princípio, uma guia que indique a adequação da automação ou que novos resultados podem ser obtidos. O usuário, por não ter acesso a sistemas de informação que executem a mesma atividade, tem pouco conhecimento sobre o que é possível fazer, ou não tem idéia de qual o custo de produzir certos resultados.

Já os projetos de re-desenvolvimento apresentam a vantagem de possuir uma base que pode ser utilizado como referência do que deve ser feito (repetido), do que não deve ser mantido (eliminações) e das novas atividades necessárias. O usuário, acostumado e experiente com um sistema existente, pode fornecer informações mais adequadas sobre o que espera do novo sistema, ou da manutenção ou melhoria sendo feita.

#### **II.4.1 Porque são feitos projetos de SI**

Muitos são os motivos que influenciam o início de um projeto de desenvolvimento de um Sistema de Informação. Em geral, usando um raciocínio econômico, podemos dizer que um projeto é iniciado quando o **retorno esperado** supera o **custo do projeto**. O problema é que não é fácil converter esses valores em números normalmente.

Vários motivos podem ser analisados como benefícios esperados de um projeto. Um deles é modernização de um sistema. Com o tempo a tecnologia de um sistema vai se tornando superada. Isso faz com que o risco e o custo de manter o sistema funcionando naquela tecnologia aumentem, aumentando gradativamente o interesse de se transportar o sistema para uma nova plataforma. Simultaneamente, novas tecnologias apresentam novas oportunidades, como desempenho superior ou facilidade de aprendizado, aumentando também com o tempo o interesse nessa atualização. Chega um momento então que passa a valer a pena o investimento em modernização.

Outro motivo importante é a mudança de premissas básicas do negócio, causada pela atuação da firma no mercado. Essas mudanças tanto podem de dentro da empresa quanto podem ser provocadas por mudanças na legislação ou por ação dos concorrentes. Por exemplo, há alguns anos atrás, no Brasil, foram feitas várias modificações nos sistemas financeiros das empresas para aceitar a mudança de moedas e o convívio com moedas diferentes simultaneamente. Em outro exemplo, com a invenção e grande

aceitação dos sistemas de premiação por viagens ou por milhas, as companhias aéreas tiveram que desenvolver sistemas específicos, interagindo com seus sistemas de passagens, para tratar do assunto. Muito comum também é a mudança de uma atividade da empresa, seja por um processo contínuo, como o de qualidade total, quanto por processos radicais como os de reengenharia e *downsizing*.

Os sistemas de informação também são importantes por oferecerem as empresas uma capacidade maior de competição. Com a informação correta e com os processos corretos de tratamento da informação uma empresa pode ter um diferencial de qualidade no mercado. Por outro lado, se todo um mercado já adotou um tipo de sistema, ou se pelo menos um concorrente já o fez, a empresa que não tem um sistema equivalente fica prejudicada na competição. Esse tipo de efeito foi visto quando as companhias aéreas passaram a vender passagens via Internet. No início era mais uma propaganda, depois passou a ser um diferencial positivo. Atualmente todas as companhias aéreas possuem formas de vender passagens diretamente via Internet.

Hoje em dia um grande motivador de novos projetos e a busca por melhor tratamento da informação que já existe em sistemas de tipo operacional, como a criação de Sistemas Suporte Executivo.

## **II.5 Exercícios**

- 1) Escolha um tipo de negócio de pequeno porte, como uma agência de viagens, e descubra (ou imagine) quais os principais sistemas de informação que ela necessita ou pode usar.
- 2) Classifique os sistemas anteriores quanto ao seu nível na organização.
- 3) Classifique os sistemas anteriores quanto ao seu tipo.
- 4) Imagine que esse negócio se torna um grande negócio, por exemplo, uma grande cadeia de agências de viagens, e descubra (ou imagine) que novos sistemas podem ser necessários.
- 5) Que sistemas de informação fazem parte de seu dia a dia? Que papel você assume ao utilizar esses sistemas?
- 6) Que sistemas de informação você pode se lembrar que contém informações importantes sobre sua vida pessoal ou profissional?
- 7) Imagine uma empresa de plano de saúde que possui um sistema de nível operacional que registra e permite a aprovação pela pessoa responsável de exames e consultas. Que sistemas de informação de outros níveis podem ser feitos para utilizar essa informação? Que outros sistemas de informação podem fornecer informação para o sistema de aprovação?
- 8) Defina, para um sistema de informação escolhido por você, as informações necessárias, que dados às descrevem e que conhecimento pode ser obtido a partir delas.
- 9) Muitas lojas no Rio de Janeiro ainda utilizam sistemas de informação feitos em MS-DOS. Faça uma análise dos custos e benefícios para mudar um sistema desse

tipo para Windows ou de mantê-lo como está. Qual a melhor opção atualmente?  
Qual a melhor opção nos próximos 2, 5 e 10 anos?

- 10) Que tipos de sistemas podem interessar a Livraria ABC?
- 11) Que tipos de sistemas podem interessar a Empresa de Clipping ClipTudo?

# Capítulo III. O Desenvolvimento de Software

---

*Análise: ... 3. Exame de cada parte de um todo, tendo em vista conhecer sua natureza, suas proporções, suas funções, suas relações, etc.*

*Novo Dic. Aurélio*

Análise de Sistemas
Analista de Sistemas
Ferramentas
Ciclo de Vida
Equipe de Desenvolvimento

Desenvolver software é a atividade, de longo prazo, de criar um ou mais programas de computador, um sistema, de forma a atender necessidades específicas de um cliente ou grupo de clientes.

No desenvolvimento de software realizamos as atividades de descoberta das necessidades e de criação do produto de software propriamente dito. Podemos dividir as atividades do processo de desenvolvimento em alguns grandes grupos:

- Atividades de Análise, cuja finalidade é descobrir “o que” deve ser feito.
- Atividades de Projeto, cuja finalidade é descobrir “como” o software deve ser feito.
- Atividades de Implementação, cuja finalidade é produzir o produto de software de acordo com as especificações produzidas nas fases anteriores.
- Atividades de Controle de Qualidade, onde se incluem todas as atividades com objetivo de garantir a qualidade do produto, como testes e verificações.

De acordo com o processo de desenvolvimento escolhido, cada uma dessas atividades pode ser dividida em várias outras sub-atividades ou tarefas. Elas podem ser executadas de diferentes formas, em diferentes ordens. Também é possível que as atividades de análise e projeto sejam feitas de forma implícita, por exemplo, quando desenvolvemos o software unicamente por meio de protótipos.

Dependendo do grau de formalidade do processo de desenvolvimento, que deve ser escolhido em função de um grupo de variáveis, como a complexidade do projeto, prazo e características da equipe. Enquanto um produto para um único usuário pode ser feito por meio de protótipos, sem nenhuma fase bem-definida, produtos muito grandes, como um sistema de informações para toda uma empresa, necessitam ser realizados em passos muito bem planejados.

Esse livro trata de algumas metodologias usadas no processo desenvolvimento de software. Dependendo da fonte que utilizamos, esse processo pode envolver uma miríade de atividades. Na norma ISO 12207 são citadas, por exemplo: análise de requisitos,

projeto, codificação, integração, testes, instalação e aceitação. Outras referências podem apresentar divisões distintas dessas atividades, ou incluir atividades novas (como testes de integração e testes de unidades). Nesse livro estamos principalmente interessados em conhecer ferramentas para realizar a análise de um sistema de informação.

### **III.1 Análise**

Por **análise** entendemos a tarefa de levantar e descrever os requisitos de um sistema, definindo de que forma deve funcionar para atender as expectativas de todos que nele possuem algum interesse. Normalmente se diz que a análise define “o que” o sistema deve fazer sem especificar “como” fará. Durante a análise devem ser explicitadas que tarefas o sistema deve executar e que dados deve manter em memória. Para isso, criamos um ou mais modelos do sistema, descrevendo-o com diferentes perspectivas e graus de detalhe.

É a partir da análise de desenvolvemos um sistema. Ela é, simultaneamente, um acordo entre os desenvolvedores e seus clientes e um mecanismo de comunicação entre os desenvolvedores. Em ambos os casos, a análise define que serviços devem ser fornecidos pelo sistema a ser implementado e, por consequência, que serviços não estão no escopo do sistema.

Segundo Pressman [4], “todos os métodos de análise devem ser capazes de suportar cinco atividades:

1. Representar e entender o domínio da informação;
2. Definir as funções que o software deve executar;
3. Representar o comportamento do software em função dos eventos externos;
4. Separar os modelos de informação, função e comportamento de maneira a apresentar os detalhes de forma hierárquica, e,
5. Prover a informação essencial em direção à determinação dos detalhes de implementação.”

Dessa definição, é possível deduzir que para a análise de um sistema ser útil e de qualidade, não basta entender “o que” deve ser feito, mas também desenvolver uma representação que permita documentar e comunicar essa informação.

#### **III.1.1 Modelos de Análise**

Vários autores fizeram propostas de modelos para descrever a análise de sistemas. Dentro do conceito de análise, apresentaremos os seguintes modelos:

- Modelo de Negócio, que descrevem como funciona o negócio onde o sistema está inserido.
- Modelo de Dados, que descreve os dados guardados pela memória do sistema, na forma de um Modelo Conceitual e segundo o método de entidades e relacionamentos.

- Modelo Funcional, que descreve a funcionalidade essencial do sistema, utilizando diagramas de fluxo de dados.

Incorporamos também, em nossa fase de análise, a modelagem por pontos de função, que nos permitirá definir um tamanho para nosso sistema.

### **III.1.2 Ferramentas da Análise**

A maior parte do trabalho de análise é feita da comunicação entre pessoas. Muito dessa comunicação é feita na linguagem natal dessas pessoas, como o português. Um grande problema dessa comunicação é que línguas como o Português e o Inglês permitem a construção de sentenças ambíguas.

No desenvolvimento de sistemas temos que evitar ao máximo as ambigüidades. Para isso, temos que restringir a linguagem utilizada de forma que uma sentença só tenha uma interpretação possível (ou mais provável). Para isso foram criadas várias linguagens, algumas gráficas, que tem o poder de restringir as interpretações possíveis do que queremos dizer. Veremos no decorrer desse livro uma visão geral de algumas dessas linguagens.

### **III.1.3 A Tecnologia**

A grande maioria dos autores advoga que não devemos levar em conta a tecnologia que a ser empregada no desenvolvimento durante a análise de sistemas. A análise deve se preocupar com o “o que fazer” e nunca com o “como fazer”.

Como estaremos seguindo os princípios da análise essencial, esta questão ficará inicialmente ainda mais restrita, pois o modelo essencial não pode conter nenhuma referência à tecnologia, sob o risco de produzir requisitos falsos.

Isso não quer dizer que o analista não possua as informações sobre a tecnologia, apenas que não as usa enquanto faz o trabalho de análise. Na prática, estamos sempre limitados por escolhas como linguagens, sistemas gerenciadores de bancos de dados, arquiteturas de rede e de computador. O que devemos fazer é não deixar que essas limitações influenciem nossas decisões sobre “o que” deve fazer o sistema<sup>3</sup>.

Da mesma forma, não queremos dizer que o analista deve ignorar essas questões ao trabalhar. Ele deve anotá-las, pensar em seus impactos, porém não deve trazê-las para o modelo criado na análise.

### **III.1.4 O Analista de Sistemas**

O Analista de Sistemas é o responsável por levantar os requisitos do sistema e transformá-los em uma especificação do que deve ser feito, i.e., a Análise do Sistema. Para isso, assume vários papéis [3]: repórter, detetive, consultor, diagnosticador, investigador, organizador, solucionador de problemas, avaliador, simplificador, artista, escultor, arquiteto, auditor, especialista de organização e métodos, especialista do domínio da aplicação, gerente, etc. O porquê dessa longa relação de papéis é o fato de o analista realmente ter que assumi-los ao longo do processo. Ele entrevista pessoas em busca de

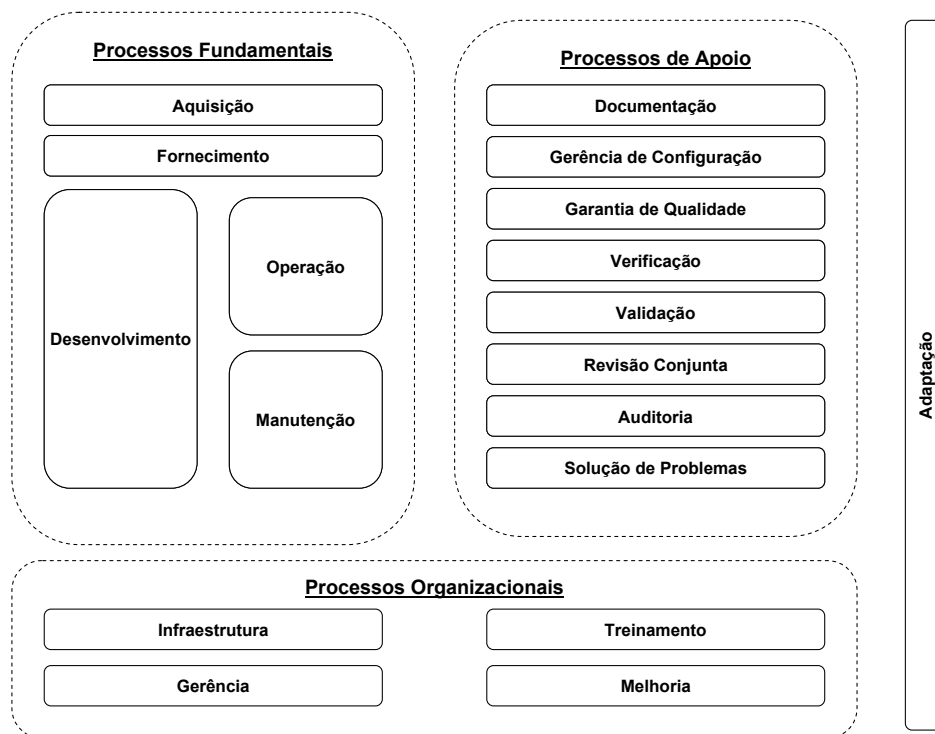
---

<sup>3</sup> Ou seja, essas decisões, muitas vezes tomadas antes de se iniciar a análise de um sistema, devem influenciar apenas a forma de funcionamento, não a razão desse funcionamento.

fatos e detalhes, descobre fatos escondidos (intencionalmente ou não), muitas vezes por meio de inferências ou pistas encontradas, propõe soluções mais adequadas para problemas atuais e futuros, a partir de diagnósticos, planeja sistemas abstratos a partir de diagramas, e ainda muitas outras atividades.

## III.2 Ciclo de Vida e Processo de Desenvolvimento

A norma ISO 12207 fornece um *framework* para compreender as atividades e processos do ciclo de vida do software da sua concepção até o fim do seu uso, que podemos indicar pela figura a seguir. É importante perceber a quantidade de processos envolvidos no ciclo de vida do software, pois eles nos mostram que muitas vezes subestimamos as verdadeiras necessidades para implantar e manter um sistema de informação.



**Figura 3. Framework fornecido pela norma ISO 12207, adaptado de Machado [5].**

### III.2.1 A Necessidade de Garantir a Qualidade

Desenvolver software é um processo de transformação de uma necessidade do cliente em uma seqüência de produtos de software (análise, projeto, protótipo, manuais) que tem em seu fim um programa de computador. Essas transformações são imperfeitas, devidos a problemas de comunicação entre o usuário e o desenvolvedor e falhas nas técnicas utilizadas pelo desenvolvedor para garantir que nenhuma informação é perdida ou inserida de forma espúria no sistema.

Para garantir que o sistema faz o que o usuário deseja, utilizamos duas técnicas: a verificação e a validação. Verificar significa analisar se o produto de uma fase do processo

de desenvolvimento está de acordo com sua especificação. Validar significa analisar se o produto de uma fase do processo de desenvolvimento está de acordo com as expectativas do cliente.

Precisamos ter claro em nossa mente a diferença entre as duas atividades. Quando transformamos um algoritmo em português para pascal, por exemplo, podemos fazer isso de forma perfeita e, ao mesmo tempo, fazer algo que o cliente não deseja. Quando validamos um programa com o cliente e o aprovamos, não necessariamente o que fizemos foi o que estava escrito na especificação do programa.

A tarefa mais importante, na verdade, é a validação, já que devemos atender o cliente. A validação, porém, é geralmente mais informal e mais custosa que a verificação. Assim, verificando que cada passo dado durante o processo de desenvolvimento esta conforme o passo anterior previu podemos economizar na validação.

### III.2.2 Modelos de Processo mais Comuns



**Figura 4. “Start Coding” não é considerado um dos melhores modelos de processo que você pode escolher.**

#### III.2.2.1 Processo em Cascata

Também conhecido como Linear Sequencial. Nesse processo, assumimos que as atividades de análise, projeto e implementação podem ser feitas de forma sequencial, sem que sejam necessárias interações entre as fases.

Um processo em cascata típico contaria com as seguintes fases:

1. Modelagem do Sistema, onde são estabelecidos os requisitos do sistema do qual o software sendo realizado participa, incluindo os requisitos de informação e de negócios.
2. Análise de Requisitos, onde são modelados os requisitos de informação, funcionais, comportamentais, de desempenho e de interface do software..
3. Projeto, onde são planejadas as estruturas de dados, a arquitetura do sistema e o comportamento é mapeado em procedimentos.

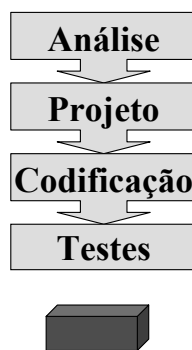


4. Codificação, onde o projeto é transformado em uma linguagem inteligível pelo computador.
5. Testes, onde verificamos e validamos o software.
6. Manutenção, onde garantimos a usabilidade do software.

Defendido inicialmente como a forma correta de desenvolver software, é basicamente impossível de ser realizado. Podemos encontrar alguns exemplos de sucesso em casos onde o produto sendo desenvolvido e o ambiente de desenvolvimento são muito bem conhecidos.

Devido à divisão radical entre as fases, é dada grande ênfase a documentação. Inicialmente assumia-se que cada fase seria executada por uma equipe distinta de especialistas, fato que está se tornando mais raro hoje em dia. Também havia discussões sobre até que ponto deveria ir o projeto e onde começava codificação. De acordo com a complexidade do sistema, muitas vezes as fases de codificação e testes eram divididas em “codificação e teste de unidades” e “integração e testes de integração”.

Entre seus principais defeitos está o fato que o cliente só vê o produto final no último dia do desenvolvimento. Assim, é impossível detectar falhas ou atender demandas imediatas do cliente. Além disso, a participação do usuário é muito baixa.



**Figura 5. O Modelo de Processo em Cascata ou Seqüencial**

#### **III.2.2.2 Prototipagem**

No processo de Prototipagem (pura) o desenvolvedor interage diretamente com o usuário, escutando seus pedidos e desenvolvendo, imediatamente, um protótipo do produto desejado. O usuário, então, utiliza esse protótipo e fornece ao desenvolvedor novas informações que o levam a modificar o protótipo, de maneira a atender todas as necessidades do usuário.

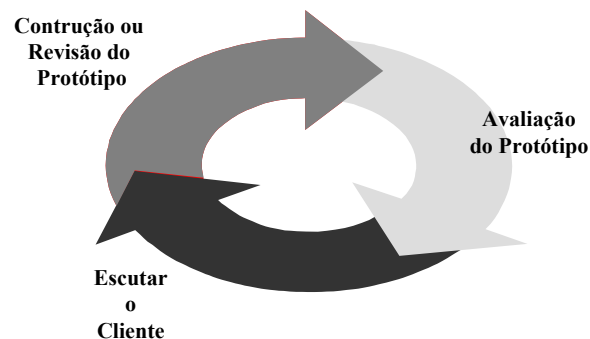
É claramente um processo de desenvolvimento baseado em um ciclo de realimentação de informações, com alta participação do usuário.

Não existe uma fase formal de análise ou projeto. Isso pode causar problemas graves e difíceis de corrigir no produto final, dificultando de sobremaneira a manutenção dos produtos. Pouca ênfase é dada à documentação.

Atualmente quase todos os processos de desenvolvimento utilizam protótipos, mas não um ciclo de vida de prototipagem.

Usamos “protótipos descartáveis” quando o protótipo é usado apenas para levantar alguns ou todos os requisitos e depois abandonado, em troca de uma implementação mais organizada. Um “protótipo operacional” é um software feito rapidamente para atender uma demanda do usuário e que é usado, mais tarde, como modelo de especificação para uma nova implementação do sistema.

É possível diferenciar “protótipos” de “*mock-ups*”. Um protótipo apresentaria o comportamento correto, ou aproximadamente correto, e seria caracterizado principalmente pela falta de rigor na implementação. Um *mock-up* apresentaria apenas uma interface que serviria como prévia da interface final.



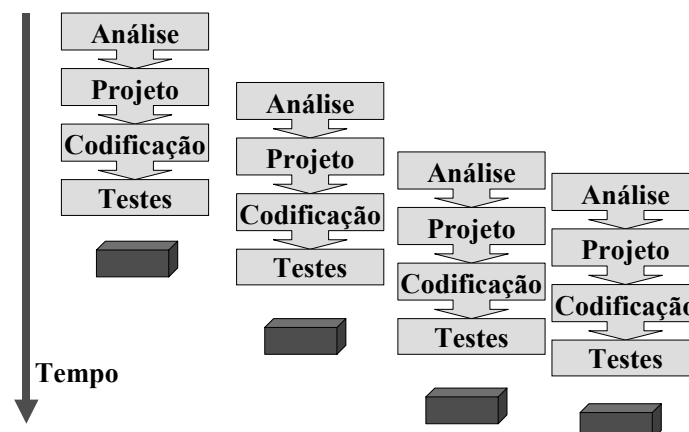
**Figura 6. O Processo de Prototipagem**

### III.2.2.3 Processos Evolucionários

Os modelos de processo evolucionários reconhecem que sistemas complexos se alteram com o tempo, usando a iteração do ciclo de desenvolvimento para acompanhar a evolução do sistema.

#### III.2.2.3.1 Processo Incremental

Pode ser visto como combinando o linear com a prototipagem. Tem o foco principal na entrega do produto. Para realizá-lo, repetimos a sequência linear em vários calendários defasados no tempo. Busca implementar funcionalidades essenciais o mais cedo possível.

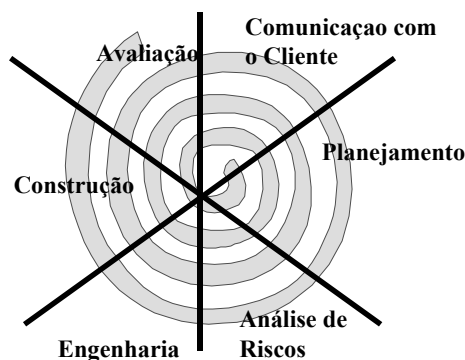


**Figura 7. O Ciclo de Vida Incremental**

### III.2.2.3.2 Processo Espiral

O Processo Espiral se caracteriza pelo desenvolvimento por uma série de produtos desenvolvidos em sequência, cada vez mais complexos e mais próximos do produto final desejado. Ele se diferencia do Processo incremental porque os produtos de cada ciclo não são “subsistemas” do sistema original, mas sim produtos específicos que atendem necessidades específicas do projeto, como por exemplo, “teste de viabilidade” e “definição da interface com o usuário”.

Em cada ciclo da espiral, algumas atividades são realizadas em ordem sequencial: comunicação com o cliente, planejamento, análise de riscos, engenharia, construção e, finalmente, avaliação dos resultados.



**Figura 8. O Processo em Espiral, visão abstrata**

### III.2.2.4 Processo Win-Win

O Processo Todos Ganham - Espiral (Win-Win Spiral) é a unificação de dois trabalhos distintos de Barry Boehm. No primeiro, o Processo espiral, ele propõe que um processo de software seja feito de acordo com um ciclo de especificações cada vez mais detalhadas que resultam em versões incrementais das capacidades operacionais desejadas. Cada ciclo envolve a elaboração dos objetivos, restrições e alternativas do produto, a avaliação das alternativas e riscos, a elaboração da definição do produto e do processo e o planejamento do próximo ciclo. No segundo, a Teoria-W, ele propõe que todas as decisões tomadas em um processo gerencial devem gerar uma situação de “todos ganham”<sup>4</sup>.

As fases para esse Processo são

- Identificar X
- Identificar condições de ganho para cada X
- Conciliar condições de Ganho
- Estabelecer objetivos, restrições e alternativas.
- Avaliar alternativas de produto e processo, resolver riscos.

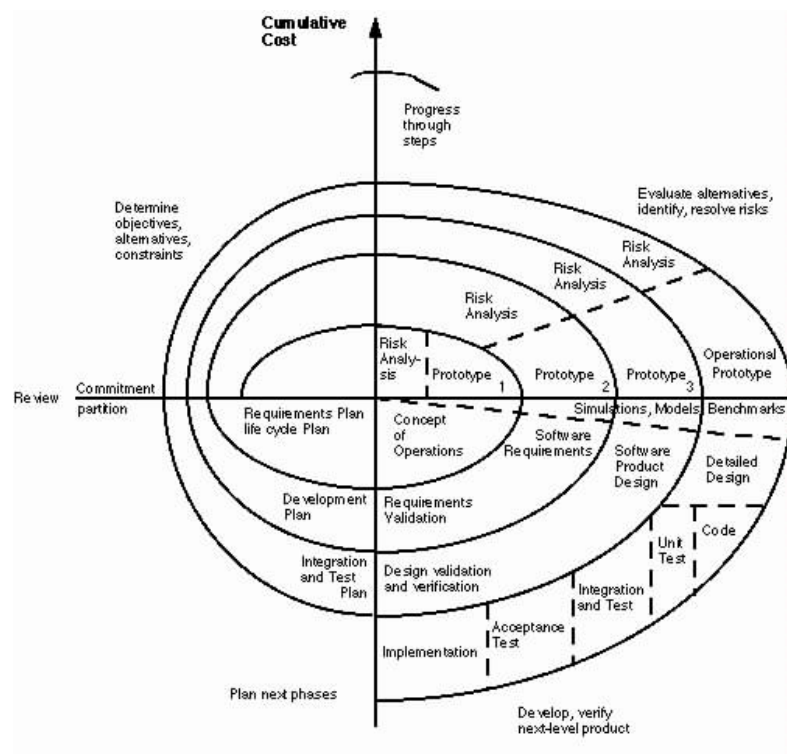
---

<sup>4</sup> Em contrapartida com o caso normal, onde um ganha e os outros perdem, ou, ainda pior, com os casos onde a decisão tomada é vista como uma situação onde todos perdem.

- Definir produto e processo, incluindo partições.
- Validar produto e processo, incluindo partições.
- Revisar e alcançar compromisso (commit)

### III.2.2.5 Desenvolvimento Acelerado

Devido a grande pressão pela produtividade que as empresas sofrem no processo de desenvolvimento de software, constantemente são propostas novas técnicas com a finalidade de acelerar o processo de desenvolvimento. Entre elas podemos citar a própria prototipagem, Rapid Application Development (RAD), Adaptive Programming, Extreme Programming e toda uma gama de processos ágeis.



**Figura 9. O processo espiral como definido originalmente por Boehm (1988)**

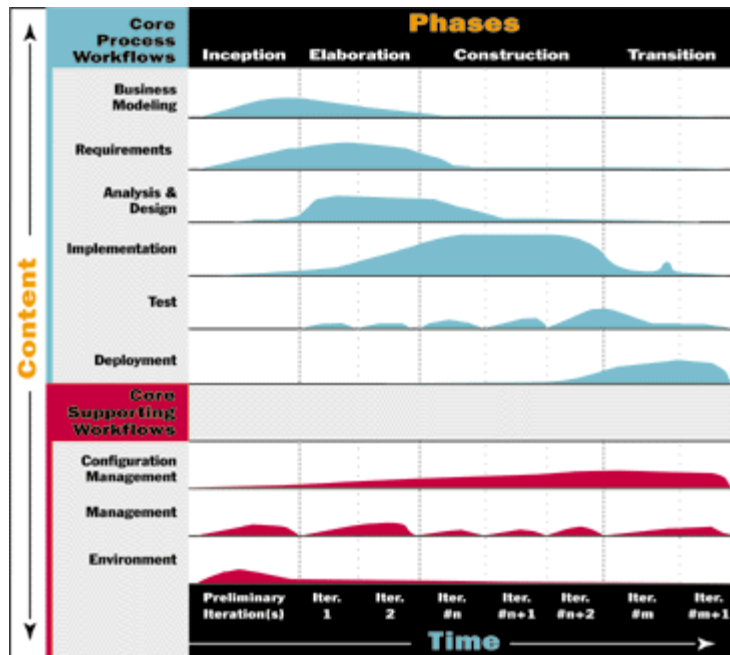


Figura 10. O RUP (Rational Unified Process) é um dos processos de desenvolvimento modernos que derivam do processo espiral

### III.3 A Equipe de Desenvolvimento

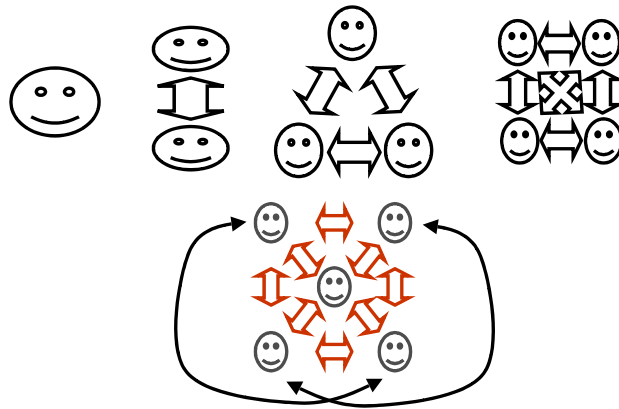
A equipe de desenvolvimento é o conjunto de pessoas responsáveis por construir o software. Dela fazem parte pessoas com diferentes habilidades. Em sistemas de informações tradicionais teremos gerentes de desenvolvimento, analistas, projetistas, programadores, administradores de banco de dados, etc. Em sistemas mais modernos, como sistemas multimídia e websites, podemos ainda ter profissões novas como artistas e professores.

É importante verificar que as pessoas em uma equipe de desenvolvimento se comunicam de alguma forma. Seguindo a regra de quanto maior o projeto, maior o número de pessoas, muito maior o número de formas em que essas pessoas podem se comunicar.

A figura a seguir tenta demonstrar essa idéia. Como uma pessoa, não há nenhuma comunicação. Com duas pessoas, só há uma maneira delas se comunicarem. Com 3 pessoas, escolhendo apenas a comunicação entre duas pessoas, já existem 3 formas. Com 4 pessoas, são 6 formas, com 5 pessoas são 10 formas. Basicamente, com N pessoas, existem  $(N \times (N-1)) / 2$  formas delas se comunicarem duas a duas.

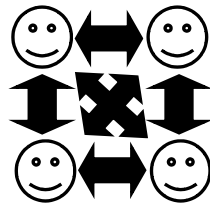
Em projetos enormes, se todos puderem falar com todos para fazer algo, a situação fica incontrolável.

Por isso, temos que organizar a equipe de desenvolvimento de alguma forma.



**Figura 11. As linhas de comunicação entre as pessoas crescem rapidamente, segundo uma explosão combinatória.**

Em uma equipe com organização democrática, todos podem se comunicar com todos. Esse tipo de equipe é razoável para projetos pequenos, com equipes de até 5 ou 6 pessoas, onde a comunicação incentivará a descoberta. Normalmente essas equipes são encontradas em universidades e no desenvolvimento de pequenos projetos de alta tecnologia (um web site, por exemplo).



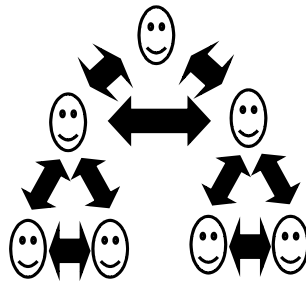
**Figura 12. Em uma equipe democrática todos falam com todos**

A forma mais tradicional de organizar uma equipe é a forma hierárquica, baseada nas teorias clássicas de administração (que por sua vez, são baseadas na forma de organização do Exército e da Igreja).

Na equipe hierárquica temos um ou mais níveis de gerência. Cabe a gerência controlar o funcionamento do projeto. Os níveis mais baixos são responsáveis pela execução do projeto.

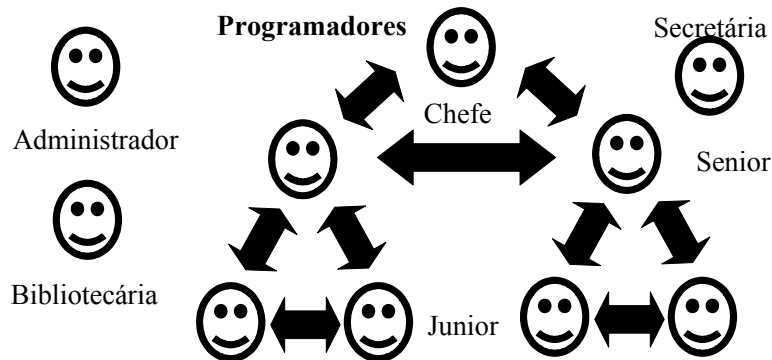
A equipe hierárquica é boa para manter regras e padrões, sendo uma escolha boa para projetos repetidos e que exigem grande estrutura. Muitas empresas utilizam esse tipo de organização, porém ele é considerado fraco para o desenvolvimento de software novo, pois a estrutura coíbe a criatividade.

Em relação ao uso dos profissionais, podemos indicar um defeito importante: o profissional de desenvolvimento experiente é transformado em gerente e “tira a mão da massa”. Muitas vezes, inclusive, ele é transformado em um mau gerente...



**Figura 13. Em uma equipe hierárquica, diminuem-se as linhas de comunicação.**

Brooks[6] propõe uma equipe conhecida como “Equipe do Programador Chefe”. Nesse tipo de equipe o desenvolvedor vai recebendo cada vez mais responsabilidade em relação ao desenvolvimento, mas não em relação à tarefa diária de administração, que é tratada a parte. Cada programador-chefe é responsável por parte do projeto e delega tarefas aos programadores seniores, que por sua vez podem delegar tarefas aos programadores juniores. Alguns desses programadores compõem uma equipe de teste.



**Figura 14. Equipes do tipo programador chefe se baseiam na capacidade do programador chefe**

Existem muitas outras formas de organizar equipes. Cada tipo de produto exige um tipo especial de equipe. Afinal, não desenvolveríamos um software para controle de voo de um avião com o mesmo tipo de equipe para o qual desenvolvemos um software para controle de uma biblioteca, não é?

## III.4 Exercícios

### III.4.1 Trabalho em Grupo (Projeto de Cadeira)

A turma deve se dividir em grupos de três a cinco pessoas, com quatro sendo o tamanho ideal. Cada grupo deve escolher um projeto de um sistema de informação. Essa escolha deve se basear nos seguintes princípios e conselhos:

1. O sistema deve ser simples o bastante para que todos o entendam
2. Algum participante do grupo deve ter experiência com o sistema ou com um sistema parecido

3. A melhor opção é escolher um aspecto do funcionamento de um negócio familiar que algum membro do grupo tenha acesso. Exemplos típicos: estoque de uma loja, atendimento de um consultório, notas de um curso, pagamento de mensalidades de uma academia, etc.
4. Outra opção é um sistema que algum membro do grupo tenha desenvolvido (ou participado do desenvolvimento).
5. Não será aceito um sistema de locadora de vídeo, por ser um exemplo muito utilizado na literatura, permitindo plágio facilmente.
6. Na prática, um trabalho de cadeira deve conter de 10 a 20 eventos e mais de cinco entidades para ter “alguma graça”. Porém nesse ponto o aluno não sabe ainda o que é um evento ou uma entidade. O professor deve orientar os alunos para que o sistema faça aproximadamente 15 “coisas”, incluindo cadastros, relatórios e processamentos.
7. Os alunos muitas vezes misturam em suas “definições” mais de um sistema entre os sistemas de informação típicos. O professor deve orientar e explicar a diferença. Confusões comuns incluem sistemas de vendas, estoque, compras e controle de produção.
8. Os alunos devem preparar uma descrição informal desse projeto.

Esse tema será utilizado no desenvolvimento de um projeto durante todo o curso



## Capítulo IV. Usuários e Requisitos

---

*A hundred objective measurements didn't sum the worth of a garden;  
only the delight of its users did that.  
Only the use made it mean something.*

*Lois McMaster Bujold, A Civil Campaign, 1999*

Stakeholders
Requisitos Funcionais
Requisitos Não Funcionais
Requisitos Verdadeiros
Requisitos Falsos
Elicitação de Requisitos
Entrevistas
JAD

A principal tarefa de um analista é descobrir o que o sistema deve fazer e como deve se comportar segundo as expectativas de seus usuários e outros interessados. Usualmente, chamamos isso de “**requisitos** do sistema”. O problema é que, no início do desenvolvimento, ninguém realmente sabe o que um sistema desejado deve fazer ao ficar pronto, inclusive o cliente. Descobrir os requisitos de um sistema é uma tarefa investigativa, criativa e contínua.

O objetivo da análise é capturar todos os requisitos para o software sendo desenvolvido e apenas aqueles requisitos verdadeiros.

Qualquer que seja o sistema, existem várias formas de entendê-lo. Similarmente, existem vários tipos de requisitos, que são aplicáveis ou não, dependendo da visão necessária naquele instante.

Um **requisito do usuário** é algum comportamento ou característica que o usuário deseja do software ou o sistema como um todo; o que o usuário quer. São escritos pelo próprio usuário ou levantados por um analista de sistemas que consulta o usuário.

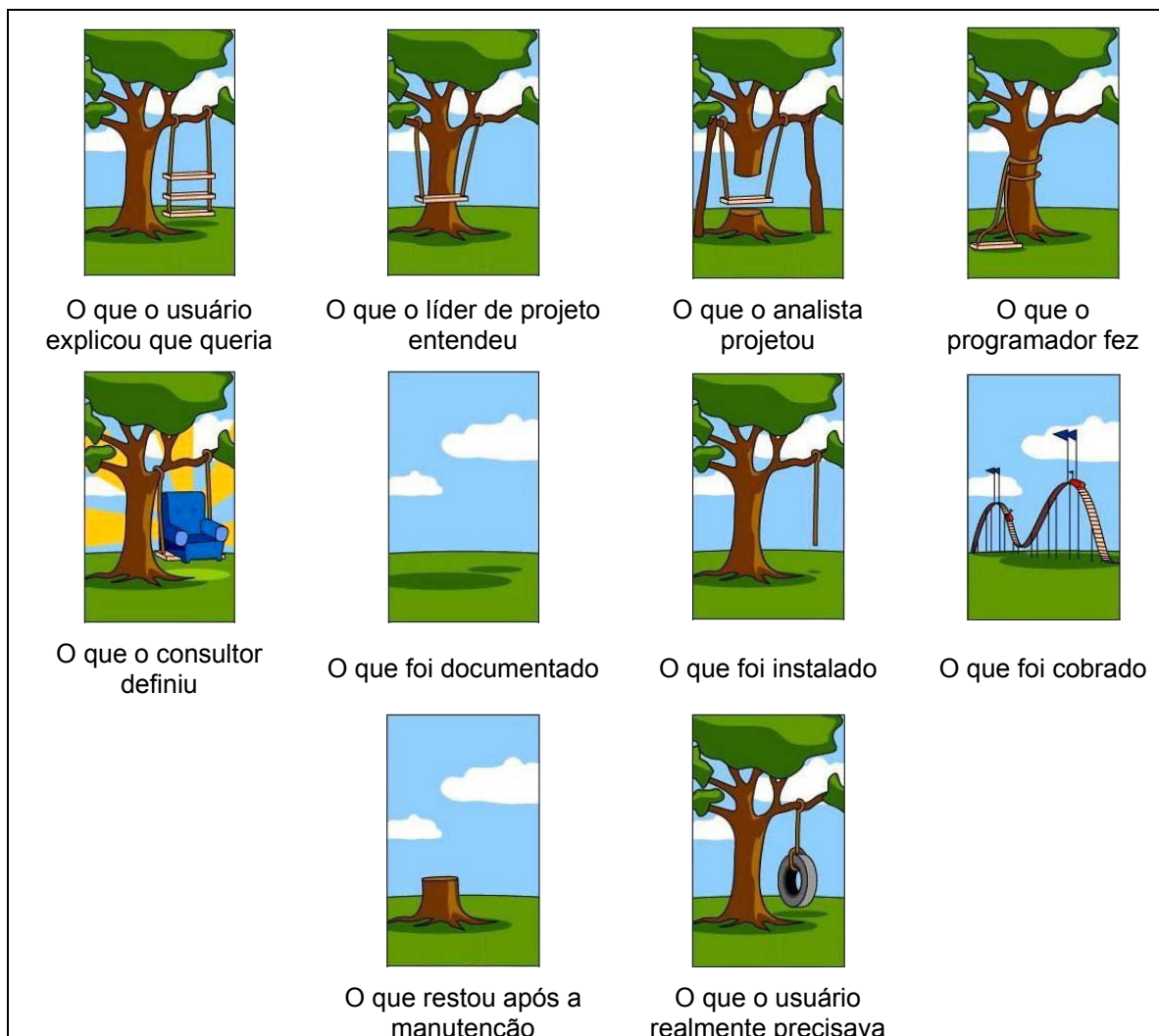
Um **requisito do sistema** é algum comportamento ou característica exigido do sistema como um todo, incluindo hardware e software. O comportamento desejado do sistema. São normalmente levantados por engenheiros ou analistas de sistemas, refinando os requisitos dos usuários e os transformando em termos de engenharia.

Um **requisito do software** é algum comportamento ou característica que é exigido do software. São normalmente levantados por analistas de sistemas.

### IV.1 Stakeholders e Usuários

*Stakeholders* são todos aqueles com algum interesse no sistema, afetando ou sendo afetados por seus resultados. Esse grupo é bem maior que o grupo de usuários, pois envolve não só estes, mas também desenvolvedores, financiadores, e outros.

Um tipo de *stakeholder* que é muitas vezes esquecido são aquelas pessoas que são afetadas pelo funcionamento do sistema, mesmo sem saber que ele existe ou está funcionando. Dizemos que essas pessoas estão “na sombra do sistema”.



**Figura 15. Uma das piadas clássicas sobre a dificuldade de determinar o que é necessário e implementá-lo, incluindo as visões de diferentes usuários.**

É interessante levantar todos os *stakeholders* de um sistema e mapear, de alguma forma, seus interesses e interações com o mesmo.

Abordagens simplificadas permitem identificar imediatamente três tipos de *stakeholders*: desenvolvedores, compradores e usuários. Uma investigação mais profunda pode achar muitos outros interessados, como: gerentes dos usuários finais, auditores, responsáveis pela operação, responsáveis pela manutenção, usuários de sistemas que enviam ou recebem dados para o sistema específico, etc.

Usuários são todos aqueles que usam o sistema com algum objetivo. O nome pode ser entendido de forma restrita, indicando apenas aqueles usuários finais, isto é, que realmente usam o sistema dentro do escopo do seu objetivo, ou de forma ampla indicando

todos aqueles que usam o sistema de alguma forma, o que inclui também os desenvolvedores.

É interessante definir uma tabela indicando que objetivos cada usuário e que interesse cada *stakeholder* tem no sistema. Isso pode ser feito de forma preliminar nessa fase.

Objetivos e Interesses de Stakeholders			
Agente ou Interessado	Objetivo	Interesse	Prioridade
Cliente	Fazer pedido		1
Cliente	Obter status do pedido		2
Gerente	Obter lista de pedidos diária		1

**Tabela 1. Lista de objetivos e interesses**

#### **IV.1.1 Perspectivas dos Usuários**

Quando estamos fazendo a análise de um sistema, por meio de entrevistas ou reuniões, interagimos com pessoas com visões e descrições diferentes do que é o sistema. Um cuidado importante que devemos ter é quanto à posição da descrição que está sendo feita em relação ao sistema.

Nossa metodologia está interessada em eventos que partem do ambiente, de fora, para dentro do sistema. Assim, devemos descrever cada evento com essa perspectiva. Nossos clientes, porém, não têm sua perspectiva limitada pelo nosso modelo, afinal eles conhecem seu negócio e não precisam de um método como o nosso, onde as limitações têm como objetivo clarificar a compreensão do sistema pelo analista.

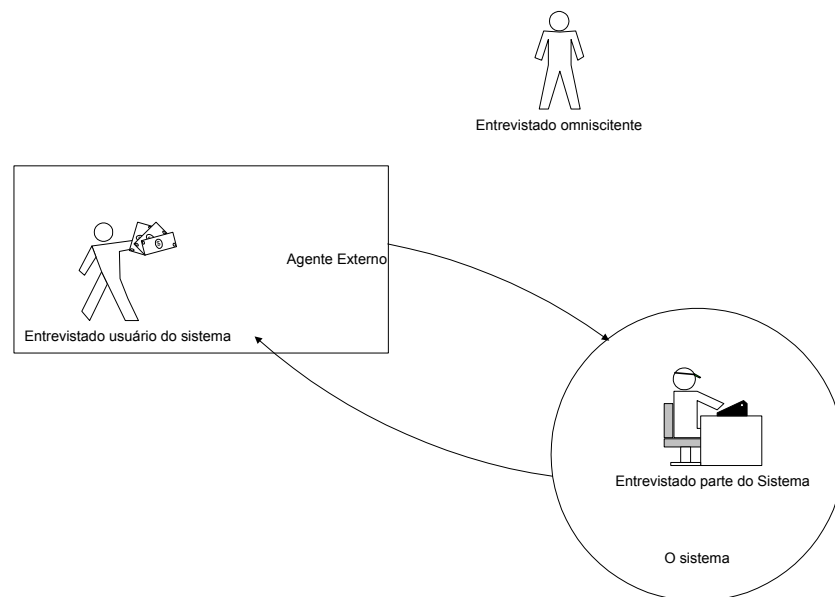
Assim, muitas vezes um entrevistado descreve o sistema como se fosse uma entidade mágica, outros descrevem o sistema como se fizessem parte dele, outros descrevem apenas as saídas do sistema, etc.

Devemos estar preparados para isso e garantir uma modelagem condizente com o modelo essencial que atenda todos os pedidos do usuário. Devemos também fazer, sutilmente, o usuário compreender a nossa forma, essencial, de descrever os eventos. Muitos usuários entendem facilmente o conceito de eventos quando traduzidos para português mais simples, como “entrada de dados” e “entrada de comandos”.

As perspectivas básicas que encontramos em entrevistas e reuniões são as seguintes:

- Entrevistado onisciente: descreve o sistema como “o sistema”, indicando coisas que ele “deve fazer”. Vê tanto o sistema como os seus usuários de uma perspectiva externa, aparentemente conhecendo os mecanismos tanto por dentro quanto por fora. Normalmente é a posição da alta gerência e de quem contratou o sistema. É comum que não conheça os procedimentos internos do sistema como acontecem realmente, mas apenas de forma geral ou como aconteciam no passado. Exige funcionalidade do sistema, principalmente para atender o nível gerencial.
- Entrevistado usuário: descreve o sistema como se o estivesse usando diretamente, muitas vezes já usando o sistema atual. Exige funções do sistema, principalmente para atender o seu nível de atuação (gerencial ou operacional). Pode apresentar alguma desconfiança, pois o novo sistema pode exigir novos conhecimentos. Conhece a entrada e a saída do sistema, mas não necessariamente os procedimentos internos.
- Entrevistado parte do sistema: descreve o sistema visto por dentro. Muitas vezes é quem vai ter o trabalho substituído, em todo ou em parte, pelo sistema, o que pode causar desconfiança e até mesmo franca hostilidade. Conhece os procedimentos na forma como são realizados e as exceções que podem acontecer.

Não podemos deixar de entender que alguns usuários fornecem uma perspectiva mista, principalmente quando envolvidos com diferentes partes do sistema.



**Figura 16. Os tipos de entrevistados em relação ao sistema**

## IV.2 Tipos de Requisitos

### IV.2.1 Requisitos Funcionais e de Informação

Uma maneira de dividir os requisitos do sistema é separá-los entre requisitos funcionais e não funcionais.

Um **requisito funcional** representa algo que o sistema deve fazer, ou seja, uma função esperada do sistema que agregue algum valor a seus usuários. Exemplos típicos incluem a emissão de relatórios e a realização e manutenção de cadastros.

Como veremos mais tarde, **os eventos essenciais definem todos os requisitos funcionais do sistema**, dado que a função dele é responder a todos os eventos. Apesar de não ser fácil levantar corretamente os requisitos funcionais, a metodologia essencial fornece um arcabouço eficiente para essa tarefa, que é perfeitamente adequado para sistemas de informação.

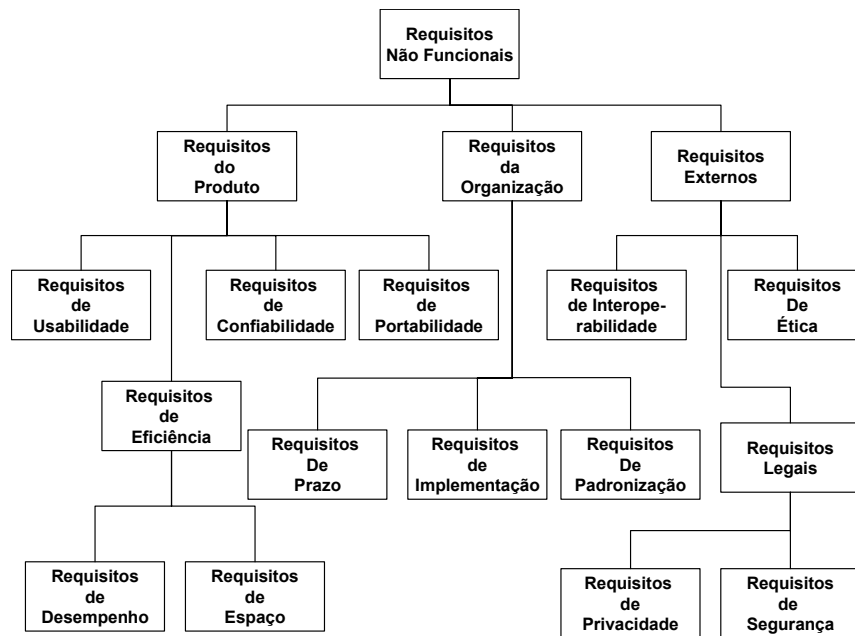
**Requisitos de Informação** representam a informação que o cliente deseja obter do sistema. Requisitos de informação também são atendidos por eventos. Muitas vezes o cliente expressa requisitos de informação de forma funcional. Outras vezes o cliente está preocupado em conseguir uma informação, mas não sabe como fazê-lo na forma de um requisito funcional. Em todo caso, sempre nos preocuparemos em levantar todos os requisitos de informação, pois eles representam as respostas fundamentais do sistema.

#### **IV.2.2 Requisitos não funcionais**

**Requisitos não funcionais** falam da forma como os requisitos funcionais devem ser alcançados. Eles definem propriedades e restrições do sistema. Muitos requisitos não funcionais são também requisitos de qualidade, como exigências de desempenho e robustez. Outros são restrições ou exigências de uso de uma ou outra tecnologia.

Porém, muitas vezes não é só difícil descobrir quais são os requisitos não-funcionais, mas também produzir uma especificação do sistema que possa ser cumprida em custo razoável e prazo hábil de forma a atender os usuários. Um exemplo típico seriam dois requisitos não funcionais que, genericamente, são opostos: velocidade e transportabilidade. Ora, para fazer um software muito veloz você precisa adaptá-lo especificamente para o ambiente onde ele está funcionando. Para fazer um software transportável, você precisa implementá-lo de forma a funcionar no maior número possível de ambientes. Normalmente esses dois requisitos se relacionam de forma inversa e para implementá-los simultaneamente é necessário um grande investimento de recursos.

Existem muitas formas de se dividir os requisitos não funcionais, a figura a seguir apresenta uma dessas formas, apenas para deixar clara a quantidade de fatores que podem envolver um requisito não funcional, mas sem considerá-la melhor ou pior que outras.



**Figura 17. Tipos de requisitos funcionais, adaptado de <http://dme.uma.pt/jcardoso/Teaching/EMR/Lectures/11> (8/ago/2003) e Ian Sommerville, Software Engineering.**

Esse texto tem como foco requisitos funcionais e requisitos de informação. Ainda não existe uma forma plenamente aceita de tratar requisitos não funcionais, mas o leitor interessado poderá encontrar diferentes métodos na literatura de Engenharia de Software.

### IV.2.3 Outros tipos de Requisitos

James e Susan Robertson [7] identificam mais três tipos de requisitos: restrições de projeto, temas de projeto e impulsionadores de projeto.

As **restrições de projeto** são os limites impostos ao sistema para que funcione no seu ambiente operacional. Estas restrições tanto podem ser técnicas, envolvendo necessidades ou disponibilidades de fatores como hardware, software, rede e interoperabilidade com outros sistemas, quanto podem ser ligadas ao negócio. Restrições podem ser vistas como requisitos não funcionais de certa forma especiais, pois são limitantes.

Os **impulsionadores de projeto** são as forças do negócio que fazem o projeto acontecer. Podem ser considerados requisitos na medida em que são os geradores originais dos requisitos funcionais e não funcionais. Tanto o objetivo inicial do sistema como todos os nele interessados (*stakeholders*) são impulsionadores.

**Assuntos de projeto** servem para completar o quadro geral de todos os fatores que influenciarão o sucesso ou o fracasso do projeto.

### IV.3 Requisitos Verdadeiros e Falsos

Um **requisito é verdadeiro** quando o sistema deve cumpri-lo qualquer que seja a tecnologia de implementação escolhida. Se um sistema pode cumprir sua finalidade sem que um requisito seja implementado, então esse **requisito é falso**.

Requisitos falsos aparecem de várias formas dentro do sistema: por cópia de sistemas antigos, por hábito do analista, por pensarmos na tecnologia antes do tempo. Dividem-se em dois grupos:

- **Requisitos falsos tecnológicos** são aqueles incluídos em um sistema por antecipação de alguma característica tecnológica futura, como a linguagem de programação a ser utilizada, ou por alguma característica tecnológica passada, como o tipo de arquivo em que estão guardados os dados atualmente. Podem, novamente, ser divididos em:
  - Requisitos tecnológicos incluídos pelo **passado**, quando incluimos na especificação algo que existe na implementação existente, mas que não é necessário ao funcionamento do sistema;
  - Requisitos tecnológicos incluídos por **antecipação**, quando incluimos algo na especificação em função de alguma tecnologia escolhida para a implementação;
- **Requisitos falsos arbitrários** são incluídos pelos analistas, ou por preciosismo<sup>5</sup> ou por características da ferramenta de modelagem sendo utilizadas. Também podem ser divididos em:
  - Requisitos arbitrários incluídos por **influência da ferramenta de modelagem**, quando incluimos na especificação algo desnecessário para fazer o sistema, mas necessário por alguma característica da ferramenta de modelagem, e
  - Requisitos arbitrários incluídos por preciosismo, quando incluimos uma função espúria no sistema, i.e., uma função que foi não solicitada pelo usuário.

O principal problema de introduzir requisitos falsos é que eles aumentam de várias formas o risco do projeto não se completar a contento. Um requisito falso, por si só, pode mascarar um requisito verdadeiro. Além disso, o acúmulo de requisitos falsos aumenta a complexidade do sistema, de tal modo que pode tornar sua implementação inviável.

Assim, a busca dos requisitos verdadeiros, e apenas esses, deve ser uma das principais formas de garantir o sucesso de um projeto.

Para que tenhamos um sistema apenas com requisitos verdadeiros, imaginamos que ele será implementado com uma **tecnologia perfeita**. Assim, evitamos os requisitos falsos causados pela tecnologia, seja ela passada ou antecipada. Em um sistema de tecnologia perfeita, como diz o nome, os processadores são infinitamente velozes, não

---

<sup>5</sup> Chamado em inglês de *gold-plating*

gastam energia e não cometem erros, as memórias são infinitamente grandes, os dados são transportados sem gastar tempo.

A Análise Essencial fornece conceitos importantes para que possamos nos guiar na descoberta de todos os requisitos verdadeiros e para que evitemos a inclusão de requisitos falsos em nossos sistemas.

#### **IV.3.1 O Usuário não Sabe Tudo**

Um problema comum é o usuário pedir algo como requisito porque ele pensa que esta é a forma de implementar um funcionalidade desejada. Esse erro, além de comum, é provavelmente prejudicial ao sistema se não for detectado. A verdade é que o usuário não tem a visão interna do sistema, ele cria essa visão a partir da percepção que tem da interface do sistema, real ou ainda em sua imaginação. Cabe ao desenvolvedor perceber o verdadeiro objetivo do usuário e determinar o método mais correto de implementá-lo.

### **IV.4 Descrevendo Requisitos**

Requisitos de projeto podem ser descritos com as seguintes informações [8]:

- Número identificador,
  - para facilitar a discussão, identificamos todos os requisitos unicamente.
- Tipo
  - Classificando-o como funcional, não funcional,...
- Evento que o atende<sup>6</sup>
- Descrição
- Justificativa
- Fonte do requisito
  - A pessoa ou o grupo que o originou
- Critério de aceitação
  - Uma medida que possa ser usada para garantir que o requisito foi alcançado.
- Satisfação do usuário
  - Um grau, de 1 (nenhum interesse) a 5 (extremamente satisfeito), por exemplo, indicando a satisfação do cliente se esse requisito for alcançado.
- Insatisfação do usuário

---

<sup>6</sup> Eventos Essenciais serão descritos no Capítulo VIII.



- Um grau, de 1 (nenhum interesse) a 5 (extremamente insatisfeito), por exemplo, indicando a satisfação do cliente se esse requisito **não** for alcançado.
- Dependências
  - Referências a outros requisitos que dependem de alguma forma desse requisito
- Conflitos
  - Referência aos requisitos que de alguma forma conflitam com esse
- Material de apoio
  - Listagem de material de apoio para atender esse requisito
- Histórico
  - Documentação da criação e das mudanças efetuadas

Esses autores ainda propõem que os requisitos sejam levantados em cartões padronizados, que facilitam sua discussão.

#### **IV.4.1 Dependência de Requisitos**

É importante notar que os requisitos não são independentes uns dos outros. Muitos requisitos só podem ser implementados se outros requisitos forem implementados antes. Por exemplo, é impossível fazer um relatório de vendas sem que se cadastrem as vendas previamente no sistema. Uma das atividades mais importantes da gerência de requisitos é manter esse relacionamento de dependência, que influenciará em todo desenvolvimento e Processo do sistema.

Para isso existem algumas soluções possíveis. Uma delas é manter uma tabela de dependência de requisitos, outra manter um banco de dados de requisitos que inclua relações de dependência. Existem alguns produtos no mercado especializados na gerência de requisitos.

#### **IV.4.2 Priorizando Requisitos**

Existe uma tendência grande de o sistema crescer muito durante a análise. Principalmente se entrevistamos um grande número de pessoas, existe uma facilidade natural das pessoas para propor novas funcionalidades para um sistema que ainda não existe, por imaginarem alguma utilidade nessas funções propostas.

Assim, muitas vezes nos vemos envolvidos com uma quantidade de requisitos tão grande que é óbvio que o sistema a ser feito não poderá ser entregue no prazo ou pelo custo combinado, ou que se pensava em combinar.

Nesse caso, algumas técnicas podem ser utilizadas para caracterizar o que deve ser realmente feito ou, pelo menos, em que ordem as coisas devem ser feitas.

A primeira técnica disponível é associar a cada requisito do sistema uma importância. Uma escala de três ou cinco valores é adequada para isso, como em: “Imprescindível para o sucesso do sistema”, “Funcionalidade Importante, mas podemos

esperar algum tempo”, “Ajudaria ter, mas é possível viver sem essa funcionalidade”, “Benefícios mínimos”, “Desnecessário”.

A segunda técnica disponível é planejar o sistema para ser entregue em várias versões, mesmo que nem todas as versões estejam incluídas nesse contrato, e pedir para o cliente determinar que funcionalidades devem aparecer em cada versão. Nesse caso pode ser interessante dar um peso ou custo para cada requisito, de modo que o cliente possa “controlar seus gastos”.

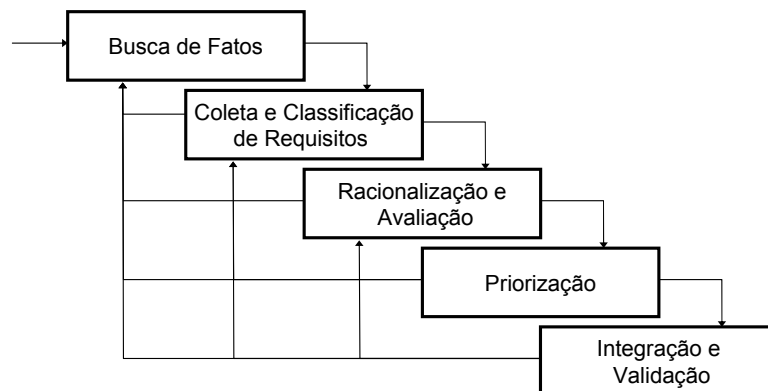
Uma terceira técnica disponível é dar uma moeda virtual para o cliente, por exemplo, 1000 dinheiros, e pedir para ele distribuir quanto pagaria por cada função, priorizando no desenvolvimento aquelas funções que o cliente decidir pagar mais.

Todas essas técnicas, porém, ficam dependentes de uma outra priorização importante dos requisitos: a priorização por dependência.

## IV.5 Métodos de Elicitação de Requisitos

A elicitação de requisitos é o levantamento, registro e validação [9] das expectativas dos diversos interessados no sistema, seguido da consolidação e validação dessas expectativas em requisitos formais. Contemplar essas diferentes visões implica projetar interesses divergentes e conciliá-los.

A elicitação de requisitos pode ser modelada em um processo como o proposto por Christel e Kang, apresentado nas figuras a seguir.



**Figura 18. O processo de elicitação de requisitos, adaptado de Christel e Lang [9].**

<b>Tarefas da Elicitação de Requisitos</b>		
<b>Atividade</b>	<b>Tarefas orientadas ao usuário</b>	<b>Tarefas orientadas ao desenvolvedor</b>
<b>Busca de Fatos</b>	<p>Identificar grupos relevantes através de múltiplos níveis da organização.</p> <p>Determinar os contextos operacional e do problema, incluindo a definição dos modos operacionais, objetivos e cenários de missão como apropriados.</p> <p>Identificar sistemas similares.</p> <p>Realizar análise de contexto.</p>	<p>Identificar especialistas do domínio da aplicação e de desenvolvimento.</p> <p>Identificar modelo de domínio e modelo de arquitetura.</p> <p>Conduzir pesquisas tecnológicas, para mais tarde fazer estudo de viabilidade e análise de risco.</p> <p>Identificar custos e restrições à implementação impostas pelo patrocinador.</p>
<b>Coleta e Classificação dos Requisitos</b>	<p>Levantar a lista de desejos de cada grupo.</p>	<p>Classificar a lista de acordo com funcionais, não funcionais, restrições de ambiente, restrições de projeto e ainda de acordo com as partições definidas pelo modelo de domínio e pelo paradigma de desenvolvimento.</p>
<b>Racionalização e Avaliação</b>	<p>Responder questões da forma “Por que você precisa de X?”, a partir de raciocínio abstrato. Isso auxilia a transformar o raciocínio das questões sobre “como?” para as questões sobre “o quê?”.</p>	<p>Realizar uma análise de riscos, investigando técnicas, custos, prazos e incluindo análise de custos e benefícios e viabilidade baseado na disponibilidade da tecnologia.</p>
<b>Priorização</b>	<p>Determinar funcionalidades críticas para a missão.</p>	<p>Priorizar requisitos baseados em custo e dependência. Estudar como o sistema pode ser implementado de forma incremental, investigando modelos arquiteturais apropriados.</p>
<b>Integração e Validação</b>	<p>Resolver a maior quantidade possível de pontos em aberto.</p> <p>Validar que os requisitos estão concordando com os objetivos iniciais.</p> <p>Obter autorização e verificação para passar ao próximo passo de desenvolvimento (e.g. a demonstração e a validação).</p>	<p>Resolver conflitos e verificar consistência.</p>

**Tabela 2. Tarefas do processo de elicitação de requisitos, adaptado de Christel e Kang [9].**

Existem muitas técnicas avançadas de elicitação de requisitos que não cabem no contexto desse livro. Aqui trataremos de duas técnicas básicas que coleta de informações: a entrevista e reuniões de JAD.

#### **IV.5.1 A entrevista**

Entre as técnicas mais importantes de elicitação de requisitos está a entrevista. Ela está constantemente presente dentro de outras as técnicas porque, quase sempre, a Elicitação de Requisitos – em algum ponto - exige comunicação direta com os usuários e outros interessados e a única forma de comunicação, seja de que forma esteja disfarçada, é a entrevista.

A entrevista procura explicitar o pensamento do entrevistado a respeito das suas relações com seu universo em determinada instância, tanto como indivíduo quanto como profissional, revelando o conhecimento do entrevistado sobre as pessoas, objetos, fatos e procedimentos com os quais interage.

Os objetivos são os mais diversos, por exemplo: estabelecer expectativas do consumidor, verificar níveis de satisfação e necessidades atuais e futuras; estudar tendências na satisfação do cliente ao longo do tempo ou avaliar programas em andamento.

O primeiro passo de uma entrevista é determinar, entre outros aspectos: seus propósitos ou objetivos; a informação necessária, e de quem obter, para alcançar esses objetivos e os recursos disponíveis para implementar e manter o processo de pesquisa. Outros fatores merecem destaque: precisão na determinação da amostra; abrangência e relevância do conteúdo da pesquisa e, essencial, a validação. Os resultados da entrevista são sumariados em um relatório interpretativo que inclui relato sobre os achados e as recomendações mais importantes. A análise pode ser qualitativa ou quantitativa. Normalmente, as entrevistas abertas estão no primeiro caso, enquanto que os questionários são analisados quantitativamente.

A responsabilidade do entrevistador também é grande. Normalmente, além das entrevistas, é ele quem integra as diferentes interpretações, objetivos, metas, estilos de comunicação, e o uso de terminologia. Há também outras tarefas complexas como decidir a oportunidade ou não de incluir certo tipo de informação no conjunto inicial de requisitos. Finalmente, entrevistar e integrar toma tempo. Como os requisitos são voláteis, quanto mais longo o processo mais facilmente os requisitos deixam de atender as necessidades e expectativas dos interessados. Todos esses encargos recomendam que o analista conheça tanto as técnicas de desenvolvimento quanto o domínio no qual se insere.

Existem vários tipos de entrevistas. Durante o processo de análise, elas normalmente seguem o seguinte processo:

- Introdução inicial
- Familiarização com o ambiente
- Busca de fatos
- Verificação de informações conseguidas de outras formas

- Confirmação de informações conseguidas com os candidatos
- Acompanhamento, amplificação ou clarificação de entrevistas anteriores.

Outra grande variável da entrevista é o seu objetivo, entre outros:

- Levantar informações sobre a organização
- Levantar informações sobre uma função de negócio
- Levantar informações sobre um processo ou atividade
- Descobrir problemas
- Verificar fatos previamente levantados
- Fornecer informação
- Obter dicas para entrevistas futuras

Há várias formas de entrevista, entre elas: entrevista por questionário; entrevista aberta; entrevista estruturada.

#### **IV.5.1.1 Entrevista por Questionário**

O questionário é muito usado como técnica de entrevista, principalmente em pesquisas de mercado e opinião. Exige preparação elaborada. Alguns aspectos particulares do processo merecem destaque: emprego de vocabulário adequado para o público entrevistado; inclusão de todos os conteúdos relevantes e de todas as possibilidades de respostas; cuidado com os itens redundantes ou ambíguos, contendo mais de uma idéia ou não relacionados com o propósito da pesquisa; redação clara; execução de testes de validade e confiabilidade da pesquisa.

Há uma tensão não resolvida entre o uso do questionário como um evento interativo ou como instrumento neutro de medida. Por um lado, como entrevista, é visto como uma interação. Por outro lado, no interesse de torná-lo um instrumento, muitos recursos da interação existentes na conversação não são permitidos, suprimindo recursos de medida de incertezas de relevância e interpretação.

Dificuldade importante é o fato das palavras possuírem significados diferentes para pessoas diferentes em diferentes contextos. Em interações normais essas questões de interpretação são negociadas entre os participantes, mas em entrevistas com questionários o treinamento e o método utilizados proíbem essa negociação. Além disso, há necessidade do uso de técnicas específicas – nem sempre do conhecimento dos projetistas - para a construção e aplicação de questionários. A menor ambigüidade é uma das principais vantagens da entrevista via questionário.

Para gerar bons itens de questionário, devemos:

- Evitar palavras ambíguas ou vagas que tenham significados diferentes para pessoas diferentes;
- Redigir itens específicos, claros e concisos e descartar palavras supérfluas;
- Incluir apenas uma idéia por item;

- Evitar itens com categorias de respostas desbalanceadas;
- Evitar itens com dupla negação;
- Evitar palavras especializadas, jargões, abreviaturas e anacronismos;
- Redigir itens relevantes para a sua pesquisa;
- Evitar itens demográficos que identifiquem os entrevistados

#### **IV.5.1.2 Entrevista Aberta**

Esse tipo de entrevista evita muitos dos problemas dos questionários, porém também cria outros. O entrevistador formula uma questão e permite que o entrevistado responda como quiser. O entrevistador pode pedir mais detalhes, mas não determina os termos da entrevista. Permanecem, entretanto, as questões: as perguntas podem ser respondidas? A resposta faz parte do repertório normal do discurso do entrevistado? Há muitas coisas que as pessoas sabem fazer, mas tem dificuldade de descrever, como há também o conhecimento tácito, que é de difícil elicitación.

Os benefícios das entrevistas abertas são: a ausência de restrições, a possibilidade de trabalhar uma visão ampla e geral de áreas específicas e a expressão livre do entrevistado.

Há desvantagens também. A tarefa de entrevistar é difícil e desgastante. O entrevistador e o entrevistado precisam reconhecer a necessidade de mútua colaboração ou o resultado não será o desejado. Há falta de procedimentos padronizados para estruturar as informações recebidas durante as entrevistas. A análise da informação obtida não é trivial. É difícil ouvir e registrar simultaneamente; principalmente, porque há fatos que só tomam importância depois de outros fatos serem conhecidos, e aí ele já não foi registrado. Daí a importância da gravação e da respectiva transcrição; fica mais fácil selecionar e registrar o que é relevante e validar com o entrevistado.

São exigências para o relacionamento entre os participantes de uma entrevista: respeito ao conhecimento e habilidade do especialista; percepção de expressões não verbais; sensibilidade às diferenças culturais; cordialidade e cooperação.

#### **IV.5.1.3 Entrevista Estruturada**

A entrevista estruturada extrai informações sobre perguntas específicas. Nesse tipo de entrevista, é importante entrevistar a pessoa certa. É uma boa técnica para ser usada após uma pesquisa com questionário, quando é possível selecionar, entre as respostas, as partes interessadas com maior potencial de geração de outras informações. Suas vantagens são: Respostas diretas, com menos ambigüidade, informação detalhada. Sua desvantagem básica é que as questões relevantes precisam ser identificadas com antecedência.

#### **IV.5.1.4 O processo da entrevista**

O processo de entrevista não se resume ao ato específico da entrevista. Na verdade ele começa muito antes e acaba muito depois. O processo normal da entrevista inclui:

- Determinação da necessidade da entrevista

- Especificação do objetivo da entrevista
- Seleção do entrevistado
- Marcação da entrevista
- Preparação das questões ou do roteiro
- A entrevista propriamente dita
- Documentação da entrevista, incluindo os fatos e a informação conseguida durante a entrevista.
- Revisão da transcrição da entrevista com o entrevistado
- Correção da transcrição
- Aceitação por parte do entrevistado
- Arquivamento

#### **IV.5.1.5 Preparando a entrevista**

A preparação é uma necessidade básica da entrevista. Não só precisamos preparar a entrevista propriamente dita, mas também preparar a nós mesmos, como entrevistadores, e ao entrevistado.

Uma entrevista deve ter um **objetivo**. As perguntas ou o roteiro devem ser coerentes. Para isso é importante a determinação desse objetivo,. O entrevistado deve ter noção clara da finalidade da entrevista e perceber sua utilidade. Isso se faz por meio de palestras, textos de divulgação e, principalmente, se explicando ao entrevistado, no início da entrevista, seu objetivo e importância.

Muitas vezes esse objetivo não é específico, principalmente na fase inicial do projeto. Mas deve ser claro, isso é, quando expressado deve permitir que entrevistador e entrevistado compreendam o motivo da entrevista. Assim, no início do projeto os objetivos podem ser: “Conhecer o ambiente de trabalho”, “Levantar expectativas iniciais dos usuários”. Já com o passar do tempo do projeto o objetivo se torna mais detalhado, como em: “Levantar os documentos utilizados no processo de compra” ou “Avaliar as telas relativas ao cadastro de bens”.

A escolha do entrevistado é o segundo aspecto importante. Devem ser escolhidas as pessoas que permitam obter no final das entrevistas uma visão clara e o mais completa possível do problema, das diversas formas de analisá-lo e solucioná-lo. Nunca se deve tratar um problema a partir de um único nível funcional, nem de uma única visão organizacional, pois estaríamos correndo o sério risco de obter uma visão distorcida. Devemos lembrar que o sistema afetará todos os níveis funcionais e departamentos da instituição.

Dependendo do tipo de entrevista, será necessário um roteiro ou um questionário. No início da análise os roteiros levam a execução de entrevistas abertas, no final geralmente temos entrevistas por questionários. Entrevistas estruturadas são preparadas principalmente para esclarecimento de processos e atividades.

Todos os roteiros e questionários devem seguir um modelo padrão, incluindo a apresentação e a conclusão da entrevista. Quanto maior o número de entrevistadores, maior a importância de seguir um padrão.

Outros aspectos fundamentais a serem preparados são:

- A linguagem
- A coerência das perguntas
- A programação dos horários

É importante estar preparado para a linguagem a ser usada na entrevista. Nisso influenciam vários fatores, como nível cultural do entrevistado, terminologia do trabalho, jargão da área, etc. Devemos evitar ao máximo usar os nossos termos técnicos e aproveitar ao máximo a oportunidade de aprender os termos técnicos do entrevistado. Se necessário, ler um pequeno texto esclarecedor sobre a área e, sempre, ler o glossário do projeto. O entrevistador deve sempre esclarecer com o entrevistado todas as dúvidas quanto ao vocabulário utilizado no ambiente onde o sistema será implantado.

Marque a entrevista com antecedência, com confirmação de data, hora, duração e local por todas as partes. As seguintes regras devem ser observadas quanto ao horário;

- As entrevistas devem ter 30, 60 ou 90 minutos e, no máximo, duas horas.
- As entrevistas iniciais podem ser mais longas, enquanto as entrevistas finais devem ser mais rápidas.
- Evite horários perto da hora do almoço ou no final de expediente, ou em uma tarde de sexta-feira ou véspera de feriado.
- Obtenha o telefone do entrevistado, para poder avisá-lo de sua ausência em caso de urgência.
- Chegue sempre 10 minutos adiantado e esteja preparado para esperar e para ter que encerrar a entrevista mais cedo, principalmente com a alta gerência.
- Se possível, caso a entrevista seja mais curta que o combinado, marque imediatamente a sua continuação.

Quanto ao material necessário para uma entrevista, além do roteiro:

- Prepare e teste o equipamento, principalmente um gravador. Atualmente existem bons gravadores digitais a preços razoáveis no mercado.
- Tenha pelo menos 2 horas de gravação e um jogo de pilhas extras.
- Tenha um caderno de anotações (é melhor que um bloco) reservado para o projeto. Canetas de várias cores, lápis, borrachas.

#### **IV.5.1.6 Realizando a entrevista**

O objetivo normal de uma entrevista é conseguir informações do entrevistado, para isso devemos fazer não só que o usuário fale, mas também que ele pense. É



importante para o entrevistador não assumir nada e não fazer pré-julgamentos, caso contrário correrá o risco de fazer uma entrevista “viciada”.

O entrevistador deve manter o controle o assunto da entrevista. Não deixe o entrevistador mudar de assunto ou tergiversar, mantendo suas perguntas direcionadas para o objetivo da entrevista.

As duas principais armas do entrevistador são a pergunta e o silêncio. Para perguntar devemos ter consciência do tipo de pergunta que escolhemos. Se quisermos que o usuário explique algo, então devemos utilizar uma pergunta aberta. Isso é muito comum em entrevistas abertas no início da análise.

O importante é fazer o usuário pensar, para isso, o entrevistador deve evitar perguntas que conttenham a própria resposta ou as que podem ser respondidas apenas com um sim ou não. As perguntas fechadas devem ser utilizadas para tirar dúvidas do entrevistador. Use questões começando com “quem”, “qual”, “quando”, “onde”, “porque” e “como” sempre que possível. Tente completar o ciclo (quem – qual – quando – onde – porque – como) para todos os assuntos.

Em dúvida, pergunta novamente de outra forma. O entrevistador deve pedir que processos complicados sejam explicados mais de uma vez, preferencialmente sob perspectivas diferentes.

É importante estabelecer exemplos concretos para o que está sendo descrito pelo usuário. Também, em caso de uma dúvida, é melhor descrever um exemplo concreto (o que aconteceria se ...) do que uma dúvida abstrata. O entrevistador deve estar consciente que é muito difícil encontrar um entrevistado capaz de raciocinar plenamente de forma abstrata sobre um problema. Mesmo nesse caso, normalmente a forma abstrata se resume ao “caso perfeito”, sendo que as exceções são melhores explicadas com exemplos.

Não tenha pressa, não responda pelo entrevistado. Não se preocupe com a demora para responder ou o silêncio. O silêncio, inclusive, é uma boa tática para fazer o entrevistado continuar falando. Deixe o entrevistado pensar, olhe para ele curiosamente.

Antes de mudar de assunto, verifique sua compreensão, explicando de forma resumida o que acabou de ouvir. Isso permite ao entrevistado pensar e dar uma clarificação se necessário

Esteja atento para a ausência de críticas por parte do candidato. Isso pode ser causado pela falta de confiança do entrevistado em você ou porque o problema é constrangedor demais para ser tratado. O analista deve constatar esse fato no processo de análise, mas não durante a entrevista.

Observe (e anote) as interrupções casadas por fatores externos (telefone, pessoas que entram e que saem, etc.).

Separe o que é fato do que é opinião.

Conclua a entrevista de forma positiva

#### **IV.5.1.7 O Comportamento do Entrevistador**

Esteja atento ao próprio comportamento. Lembre-se que não importa sua intenção ao fazer ou deixar de fazer algo, mas a interpretação que o entrevistado dará ou que fizer ou não fizer.

No passado era comum que consultores sempre se vestissem de terno, até mesmo apenas ternos escuros. A maioria das empresas hoje utiliza um código de vestimenta informal. A regra mais atual é que o entrevistador ou consultor tome cuidado para não provocar um grande desnível entre a sua roupa e a roupa do entrevistado ou cliente, se adaptando as normas de vestimenta do cliente (ou do mercado ao qual o cliente pertence).

Fisicamente, não faça movimentos desnecessários como bater o lápis na mesa, mexer as chaves no bolso, etc. Movimentos automáticos e cacoetes distraem o entrevistado e, além disso, podem ser interpretados como falta de atenção. Não fume, mas também não evite que seu entrevistado fume. Não constranja o entrevistado comentando sobre os males do fumo. Não peça café, mas pode aceitar o oferecido. Se necessário, pode pedir água.

Estabeleça um horário para a entrevista e o cumpra rigidamente. Devido aos constantes problemas de trânsito da cidade, e a necessidade de se identificar para seguranças e secretárias, o entrevistador deve sempre planejar chegar ao local com uma folga de tempo, algo em torno de 15 minutos.

Mantenha o interesse. Tome notas, mas não seja obsessivo, principalmente não interrompa o candidato para manter suas notas atualizadas. Grave a entrevista e a reveja mais tarde se necessário. Escute ativamente sem interromper. O entrevistado é que deve falar a maior parte do tempo.

Utilize um tom educado e cortês. Não seja engraçado, sarcástico ou depreciativo. Não faça comentários pejorativos ou preconceituosos. Não faça comentários sobre política e religião, ou outro tema controverso. Seja cordial, mas sem deixar de ser profissional. Pergunte e responda com cortesia e honestidade. Não de opiniões particulares, mesmo quando pedido. A entrevista é o momento de levantar informações, não de emití-las.

Não dê a um entrevistado informações passadas por outros entrevistados. Educadamente, responda que não cabe a você a decisão ou a opinião.

Evite, de toda a forma, confrontar o entrevistado. Não torne a entrevista um interrogatório. Evite discutir, mesmo que não concorde com o usuário. Em caso de discussão, defina claramente o motivo do desacordo, seja ele motivado por fato ou por opinião. Utilize perguntas para restabelecer a comunicação em caso de desacordo. Se necessário, peça desculpas.

Basicamente o entrevistador deve ser muito educado.

#### **IV.5.1.8 Roteiro Básico**

- Apresente-se ao entrevistado: “Olá, muito prazer, eu sou fulano-de-tal, responsável por parte do projeto XYZ”. Apresente seu cartão de visitas se for o primeiro encontro.

- Informe ao entrevistado o motivo da entrevista e porque foi selecionado: “Estou aqui para levantar o funcionamento da sua área, e seu nome foi escolhido por ser o funcionário mais experiente” ou “Estou aqui para levantar o funcionamento da atividade X, que é de sua responsabilidade”.
- Deixe clara a idéia que o conhecimento e as opiniões do entrevistado são importantes e serão úteis no processo de análise
- Diga o que vai acontecer com a informação levantada
- Garanta que o entrevistado lerá a transcrição da entrevista e terá a oportunidade de corrigi-la, garanta que nada será passado a outras pessoas sem a revisão e verificação do entrevistado.
- Determine os assuntos confidenciais ou restritos a serem tratados na entrevista
- Deixe claro que não haverá consequências negativas em função do resultado da entrevista
- Solicite permissão para gravar a entrevista
- Se autorizado, inicie a gravação com um texto de apresentação: “Entrevista realizada no dia X...”.
- Faça a entrevista até faltarem 5 ou 10 minutos para o tempo determinado
- Avise ao entrevistado que o tempo está acabando e pergunte se gostaria de adicionar alguma informação
- Solicite ao candidato que responda as perguntas de conclusão
- Se necessário, marque outra entrevista.
- Entregue ao candidato o formulário de avaliação de entrevista e o envelope correspondente. Ensine-o a enviar a avaliação preenchida.
- Despeça-se educadamente, agradecendo a atenção e o tempo dispensado.

Muitas vezes a entrevista é precedida por um bate-papo informal de apresentação. Tente manter essa conversação em um tempo mínimo razoável.

#### **IV.5.1.9 Documentando a Entrevista**

A entrevista deve ser documentada logo após sua realização. Ao documentá-la rapidamente, estará garantindo que recuperará mais informação.

A documentação da entrevista deve fornecer a seguinte informação.

- A data, hora e local da entrevista.
- Nome do entrevistador
- Cargo do entrevistador
- Nome do entrevistado
- Função do entrevistado e a descrição desse cargo

- Se necessário, informações de background do entrevistado, como experiência no cargo ou com computadores.
- Organograma do entrevistado (superior imediato, colegas do mesmo nível, subordinados).
- O objetivo da entrevista
- Nomes e títulos de todos os outros presentes na entrevista
- Uma descrição completa dos fatos descritos e opiniões do entrevistado
- Opcionalmente, uma transcrição da entrevista, possivelmente expurgada das falas que não tinham relação com o assunto da entrevista.
- Todas as conclusões tiradas dos fatos e opiniões como apresentados
- Todos os problemas de negócio levantados durante a entrevista
- Exemplos de todos os relatórios, diagramas, documentos, etc., discutidos durante a entrevista.
- Todos os desenhos e diagramas feitos a partir ou durante a entrevista
- Qualquer comentário relevante feito pelo entrevistado
- Todos os números relevantes (quantidades, volume de dados, etc.) coletados durante a entrevista.

É importante notar que o relatório da entrevista deve ser aceito pelo entrevistado. É normal o entrevistado remover alguma coisa ou colocar algo a mais. O analista deve ficar atento aos motivos do usuário em fazer modificações.

Se houver discussão quanto à interpretação de algo e o analista achar essencial manter sua versão no relatório, deve também permitir que o entrevistado coloque sua versão.

#### **IV.5.1.10 As perguntas de conclusão**

Ao final da entrevista, é importante realizar uma avaliação da percepção do entrevistado sobre a entrevista que acabou de ser realizada. Para isso é necessário que seja respondido um formulário, contendo perguntas como:

- Você acha que essa entrevista cobriu tudo que era necessário?
- Você acha que foram feitas as perguntas certas?
- Você acha que era a pessoa mais certa para responder essas perguntas?

#### **IV.5.2 JAD**

Outra técnica importante de elicitación de requisitos, que merece um tratamento separado, é o JAD.

Muitas vezes, quando um grupo de informática entrega um sistema de informação aos seus clientes escuta a frase: “não era isso que eu queria!” Isto acontece porque os

desenvolvedores não conseguiram levantar com os usuários suas verdadeiras necessidades.

Este problema de comunicação pode ter diversas causas: linguagem especializada de ambas as partes, desconhecimento da área de atuação pelos desenvolvedores, preocupações com a tecnologia empregada ao invés das necessidades dos usuários, etc. Na fase inicial do projeto, conhecida como análise, a dificuldade de comunicação entre clientes e equipe de desenvolvimento é a principal causa de defeitos que serão encontrados no produto final.

Para enfrentar os problemas de comunicação entre desenvolvedores e usuários, foram desenvolvidos, ao final da década de 1970, vários métodos onde o desenvolvimento de sistemas é baseado na dinâmica de grupo.

Na forma tradicional de desenvolver sistemas os analistas entrevistam os usuários, um após outro, tomando notas que são mais tarde consolidadas e então validadas com o usuário, finalmente se transformando em um documento de análise.

O JAD, Joint Application Design, ou Método de Projeto Interativo, substitui as entrevistas individuais por reuniões de grupo, onde participam representantes dos usuários e os representantes dos desenvolvedores.

Quando o método é aplicado de forma correta, os resultados alcançados ultrapassam os objetivos imediatos da obtenção de informação dos usuários e cria um ambiente de alta sinergia na equipe, onde todos se sentem comprometidos com as soluções encontradas. Esse comprometimento permite que todos se considerem “proprietários” e “colaboradores” no desenvolvimento do sistema.

#### **IV.5.2.1 O Objetivo do Método**

O objetivo do método é extrair informações de alta qualidade dos usuários, em curto espaço de tempo, através de reuniões estruturadas para alcançar decisões por consenso.

#### **IV.5.2.2 Os Componentes**

O **líder de sessão** tem como tarefa número um conduzir o grupo para soluções de consenso. Esse líder de sessão age como **facilitador**, um servidor neutro dentro do grupo que não avalia nem contribui com idéias. A responsabilidade do facilitador é sugerir métodos e procedimentos que ajudem o grupo a concentrar energia em tarefas específicas, garantindo a todos os membros do grupo condições de participar.

O **documentador** é um auxiliar imparcial do líder de sessão, responsável pelo registro das decisões e especificações produzidas. Apenas as informações relevantes são documentadas, segundo orientação do líder de sessão.

O **patrocinador** detém a autoridade formal sobre as áreas afetadas pelo sistema, estabelecendo diretrizes e objetivos do projeto.

A **equipe** é responsável pelo conteúdo da sessão, representando as áreas envolvidas no projeto.

#### **IV.5.2.3 A Dinâmica**

A base de trabalho é a equipe presente na reunião. Para que não aconteça como na figura abaixo devemos combinar algumas “regras de jogo”, de modo a alcançar o máximo de produtividade. É natural que em um grupo de 15 pessoas surjam discussões, conversas paralelas, interrupções, etc. Em respeito ao tempo precioso dos participantes vamos é necessário estabelecer um código de cooperação.

#### **IV.5.2.4 O Ambiente**

O Ambiente físico da reunião é fundamental para a produtividade dos trabalhos. Os seguintes aspectos devem ser considerados:

- Os participantes devem estar organizados de forma a poderem se olhar e olhar para o líder de sessão. A melhor arrumação é a em forma de ‘U’.
- Não devem acontecer interrupções aos participantes.
- Todos devem cumprir a agenda, principalmente o início e o fim da reunião.

#### **IV.5.2.5 O Consenso**

A forma mais produtiva de decisão do grupo é aquela obtida por consenso. Consenso não é a unanimidade de opiniões, mas, sim, a situação em que cada membro concorda que a solução encontrada é a melhor para o grupo e que é possível conviver com ela sem ferir convicções ou valores essenciais.

### **IV.6 Exercícios**

#### **IV.6.1 Projeto 1: Livraria ABC**

Baseado em todos os textos disponíveis sobre a Livraria ABC, faça:

Uma lista de requisitos funcionais preliminares

Uma lista de requisitos não-funcionais preliminares

Uma lista de requisitos de informação preliminares

#### **IV.6.2 Projeto de Curso**

1) Para o seu projeto de curso, faça uma lista com:

- i. Requisitos funcionais preliminares
- ii. Requisitos de informação preliminares
- iii. Requisitos não funcionais preliminares

2) Documente cada requisito dessa lista de acordo os descritores de requisitos mostrados nesse capítulo.

3) Para o seu projeto de curso, prepare:

- i. Um roteiro de uma entrevista inicial do projeto
- ii. Um questionário a ser feito com os usuários atuais do serviço com a finalidade de descobrir sua qualidade

## Capítulo V. Uma Proposta Inicial

---

*It isn't that they can't see the solution.  
It is that they can't see the problem.*

*Chesterton, G. K. (1874 - 1936) in  
The Point of a Pin in The Scandal of Father Brown.*

Objetivo
Problemas
Oportunidades
Metas
Métricas
Requisitos Iniciais
Descrição Sucinta do Sistema Atual
Restrições
Proposta Inicial

### V.1 O Primeiro Contato e Os Primeiros Resultados

Quando vamos iniciar o desenvolvimento de um sistema, somos em geral convidados por um cliente em potencial para conhecer seus problemas, seus desejos e propor uma solução.

Essa é uma fase muito difícil para o desenvolvedor, pois precisa tomar muitas decisões com poucas informações. Algumas vezes é obrigado a fornecer uma proposta de trabalho, incluindo previsão de custos ou preço<sup>7</sup>, a partir de uma entrevista curta. Essa situação está muito longe da ideal, onde só teríamos que fornecer uma proposta de trabalho após saber exatamente o que será preciso fazer. Outro problema para o desenvolvedor é que esta é uma fase de investimento. Entre várias propostas apresentadas, apenas algumas serão aceitas. O tempo gasto nas propostas não aceitas é um custo a mais, a ser dividido por todos os projetos aceitos.

Esta fase inicial de negociações deve ter um objetivo: desenvolver um documento chamado “**Proposta Inicial**”<sup>8</sup>. A proposta inicial é o primeiro passo para atingir a proposta de desenvolvimento e pode ser mantida como documento interno se o cliente não exigir uma proposta imediatamente.

---

<sup>7</sup> O custo de um sistema é quanto será gasto para o seu desenvolvimento. O preço do sistema é quando será cobrado ao cliente. O custo é função direta do tamanho do sistema, o preço é uma questão de mercado.

<sup>8</sup> Não chamaremos esse documento de proposta de desenvolvimento por considerar que são necessários mais alguns dados para desenvolver uma verdadeira proposta de desenvolvimento

## V.2 A solicitação do cliente

Normalmente o cliente, ao solicitar um software, tem idéia do que necessita que esse software faça, possuindo inclusive um documento descrevendo essa idéia. É nesse documento, ou a partir das entrevistas iniciais, que o analista deve procurar as principais motivações e necessidades do cliente.

É sempre importante focalizar nas necessidades de negócio do cliente. Muitas vezes o cliente acredita precisar de um sistema para resolver um problema em seu negócio, quando na verdade precisa de outro. Deixando claras as expectativas, teremos clientes mais contentes.

## V.3 Objetivo do Sistema

O **Objetivo do Sistema** é descrito por uma sentença de poucas linhas que permite identificar imediatamente sua finalidade. A sentença deve ser clara, de forma a permitir uma definição rápida do seu escopo, isto é, da fronteira que define o que faz parte e o que não faz parte do sistema. Deve também esclarecer a razão do projeto existir e ser necessário.

Alguns objetivos razoáveis são:

“O Sistema deverá controlar a recepção e envio de pacotes pelo setor de cargas”.

“O Sistema permitirá o gerenciamento de cardápios em uma rede de restaurantes populares, definindo pratos e receitas e verificando sua aceitação”.

“O Sistema controlará a entrada e saída de funcionários, realizando o serviço de ponto e fornecendo dados para a folha de pagamento”.

Em todos esses objetivos podemos, até certo ponto, que tarefas podem fazer parte e que tarefas não devem fazer parte do sistema<sup>9</sup>.

Não devemos dar vários objetivos a um sistema. Quando isso parece necessário, temos a forte indicação que estamos tratando de mais de um sistema.

O objetivo deve identificar junto ao cliente, de forma mais padronizada possível frente ao mercado, que tipo de sistema está sendo desenvolvido. Ao mesmo tempo, deve fornecer informações suficientes que caracterizem de que forma esse sistema é especial. Assim, um bom objetivo exemplo é o seguinte:

“Desenvolver um sistema de controle de vendas para uma loja de materiais de construção que permita encomenda de mercadorias”.

Nesse exemplo descrevemos o sistema de forma bastante geral e reconhecida no mercado (“sistema de controle de vendas”), logo após declaramos que esse sistema deve ser específico para um tipo de negócio (“loja de materiais de construção”) e mais ainda,

---

<sup>9</sup> Alguns exemplos de objetivos mal definidos:

“O Sistema otimizará o desempenho da empresa na sua área de atuação”

“O Sistema possibilitará o controle dos clientes”



deve suportar uma função que não é necessariamente comum nesse tipo de aplicação como um requisito primordial (“permita encomenda de mercadorias”).

Ao declarar um objetivo devemos evitar o uso das conjunções “e” e “também”, ou ainda outra construção que leve a representar mais de uma idéia. Quando um objetivo tem mais de uma idéia proposta, geralmente temos na verdade mais de um sistema sendo descrito.

## V.4 Problemas

Ao conversar com o cliente, principalmente nas entrevistas iniciais, temos a oportunidade de ouvir muitas reclamações sobre o sistema atual, seja ele informatizado ou não. Isso é esperado, pois se não existissem defeitos na forma como o sistema é executado no momento, não haveria necessidade de implementar um novo sistema. Assim, devemos entender que a verdadeira motivação que o cliente tem em chamar uma equipe de desenvolvimento de sistema é a de corrigir os seus problemas, principalmente os que mais afetam negativamente o seu negócio.

Um problema pode ser classificado como:

- De negócio
- Funcional
- Operacional

Um **problema operacional** ocorre quando uma funcionalidade do sistema funciona de forma errada. Por exemplo, na automatização do processo de gerência das contas (pedido e fechamento de conta) de um restaurante, problemas operacionais comuns são: a dificuldade de fechar a conta e os erros de cálculo que acontecem.

Um **problema funcional** ocorre quando o sistema não permite uma funcionalidade. No mesmo sistema de contas de restaurante, por exemplo, pode ser impossível ou muito difícil saber quanto cada garçom vendeu em um dia.

Finalmente, os **problemas de negócio** estão relacionados à manutenção do negócio propriamente dito e podem tanto ser isolados quanto causados por problemas operacionais ou funcionais. Um problema operacional como a demora ao calcular o valor final de uma conta pode causar um problema de negócio como filas na saída do restaurante.

Para identificar problemas, podemos adotar algumas estratégias [10], como: verificar os resultados obtidos atualmente nos processos e compará-los com objetivos da empresa ou padrões do mercado, observar o comportamento dos empregados e levantar a opinião de fornecedores, clientes e vendedores (representantes ou distribuidores) da empresa. Sinais específicos que podem ser verificados são:

- Quanto às tarefas realizadas, se contém erros, se são feitas vagarosamente, se não são mais feitas como definidas em documentos da companhia, se não são completadas;

- Quanto aos funcionários, se estão desestimulados, se não podem descrever suas responsabilidades e objetivos, se a taxa de demissões é alta.
- Quanto aos parceiros externos (clientes, fornecedores e vendedores): reclamações, sugestões de melhoria, queda nas vendas, vendas com perdas.

Para auxiliar o processo de levantamento de problemas, pode ser criada uma tabela com as colunas: Causa, Resultado, Valor, Processo Causador, Dados causadores, Sugestão de solução.

Problemas do Negócio						
#	Causa	Resultado	Valor	Processo Causador	Dados Causadores	Sugestão de Solução
1	Como não podemos analisar planos de negócio alternativos	Não é possível analisar cenários de mercado	Acima de 1 milhão	Planejamento	-	Modelagem de cenários
2	Como não sabemos o custo real de produção futura	Não é possível fazer contratos de longo prazo	30% do faturamento poderiam ser passados de curto prazo para longo	Compras	Preços	Sistema de acompanhamento de custos

**Tabela 3. Identificando problemas de negócio**

## V.5 Oportunidades

As oportunidades são ofertas que fazemos ao nosso cliente. Devido ao nosso conhecimento técnico e tecnológico, normalmente ficamos à vontade para oferecer **oportunidades tecnológicas**. Uma oportunidade tecnológica, como diz o nome, é a oferta de uma tecnologia específica de implementação que oferecerá alguma vantagem ao cliente. Por exemplo, ao implantar um novo sistema de estoque podemos oferecer a entrada e saída de produtos pelo uso de código de barras. A oportunidade tecnológica não altera a funcionalidade que o cliente necessita, mas sim sua forma de implementação.

Algumas vezes também podemos oferecer **oportunidades de negócio**. Uma oportunidade de negócio é alguma funcionalidade não prevista pelo cliente, mas que sabemos ser possível implementar. Devemos ter muito cuidado com oportunidades de negócio, pois elas podem ser, na verdade, falsos requisitos. Um exemplo típico é a proliferação de relatórios em sistemas que na prática usam apenas alguns.

Uma oportunidade de negócio deve ser exaustivamente discutida com o cliente de forma a ficar claro que ela realmente traz benefícios, que esses benefícios são consideráveis, que o risco do projeto não aumenta muito e que ela será realmente utilizada.

Um exemplo que podemos dar é, em um controle de estoque, a funcionalidade de prever que produtos estão próximos de sua data de vencimento. Essa não é uma função “normal” de sistemas de estoque. Exige um custo adicional não só de desenvolvimento, mas também de operação, como identificar a data de vencimento, controlar diferentes datas para um produto, etc. Em muitos contextos, essa função pode parecer interessante mais ser, na prática, inútil. Em uma oficina mecânica, por exemplo, ela pode ser totalmente inútil. Em um grande mercado, ela pode ser bastante útil. Porém em um pequeno mercado, onde o proprietário tem na verdade um controle mental do estoque e precisa do sistema de estoque apenas para fazer um controle legal ou financeiro, essa funcionalidade pode parecer útil a princípio, mas nunca ser usado no dia a dia.

As oportunidades devem produzir resultados úteis para a empresa, como acelerar processos, eliminar passos desnecessários, reduzir erros na entrada e saída de dados, aumentar a integração entre sistemas, aumentar a satisfação do usuário e facilitar a interação com os parceiros externos da organização (fornecedores, representantes e clientes). [10]

## V.6 Metas e suas métricas

Enquanto o objetivo define para que serve o projeto, as metas justificam a sua existência para o negócio.

Uma meta é um efeito positivo no negócio do cliente que é esperado com a implantação do novo sistema. Metas são benefícios trazidos pelo novo sistema ao negócio.

A análise das metas permite ao cliente verificar qual a relação custo/benefício da implantação de um novo sistema. Baseado nas metas o cliente é capaz de fazer uma avaliação econômico-financeira, comparando o preço do sistema, ou melhor, ainda, com o custo total de propriedade (TCO – Total Cost of Ownership) do sistema, com o valor equivalente dos benefícios trazidos pelo sistema.

Por exemplo, um sistema que transforme um trabalho de 1 hora por dia de uma pessoa que ganha R\$ 10,00 por hora para 15 minutos, poupa R\$ 7,50 por dia. Isso corresponde a aproximadamente R\$ 165,00 por mês, ou ainda R\$ 1980,00 por ano. Como se admite que um investimento tenha retorno em dois anos, o benefício total causado apenas pela aceleração do processo, em dois anos, seria de R\$ 3960,00. Caso esse fosse o único benefício do sistema, esse valor seria então um limite superior para o TCO.

O uso de metas para nortear o desenvolvimento de sistemas é uma proposta diferente das tradicionais, porque as metas não estão relacionadas a funcionalidades do sistema, mas sim aos resultados obtidos quando inserimos o sistema novo no ambiente.

Assim, devemos definir metas que possam ser verificadas quando o sistema for instalado. Cada meta virá acompanhada de uma **métrica**, uma medida objetiva que

permite comprovar que a meta foi alcançada. Metas que não possuem métricas objetivas, como "satisfação do cliente", devem ser evitadas ou devem ser associadas a um instrumento de medida que permita verificar conceitos subjetivos, nesse caso uma pesquisa de opinião.

Ao implantar um sistema de atendimento automático, poderíamos ter como metas:

- Acelerar o atendimento, com a métrica “tempo médio de atendimento”.
- Diminuir o tamanho das filas, com a métrica “tamanho médio da fila”.

Para cada par meta - métrica, podem ser necessários um procedimento de medida e um procedimento de levantamento de dados passados. Isso não é uma prática comum no desenvolvimento de sistemas, porém é algo desejável.

A principal vantagem de associar metas ao sistema que não fazem parte dos requisitos do sistema é demonstrar a utilidade do sistema, isto é, como o sistema faz o cliente ganhar mais dinheiro ou realizar melhor sua tarefa.

Devemos tomar cuidado, porém, com as armadilhas que existem na escolha das metas. Primeiro, é muito comum que um cliente deseje uma meta que não pode ser alcançada por um sistema com o objetivo definido. A questão, nesse caso, se resume a negociar a mudança da meta ou negociar a mudança do objetivo. Como exemplo, podemos citar o caso de um cliente que desejava um sistema de controle de pedidos para os fornecedores baseado em pedidos de clientes e desejava que o sistema diminuísse o prazo de entrega para os clientes. Analisado o funcionamento da empresa, comprovamos que era impossível acelerar o prazo meramente controlando os pedidos. Era possível, porém, fazer uma previsão mais acertada do prazo de entrega e estar preparado para atrasos, mediante o acompanhamento dos pedidos.

Outra armadilha é ter uma meta que é afetada por muitas coisas além do sistema. Em um caso simples, tínhamos a proposta de um sistema de reservas de hotel que se propunha a aumentar a ocupação dos quartos. Ora, o nível de ocupação dos quartos de um hotel depende de muitos fatores, inclusive da economia geral. Entendemos que o sistema poderia ter como meta, por exemplo, “diminuir o número de reservas não cumpridas”. Essa meta é mensurável e pode ser diretamente influenciada pelo sistema (por meio de verificações com o cliente, por exemplo). É interessante notar que a meta selecionada é um dos fatores que afeta a meta proposta inicialmente. Essa é outra armadilha comum, escolher uma meta (e uma métrica) que na verdade é derivada da meta (e da métrica) que o sistema tem condições de atingir.

Se não é possível nenhuma medida, não será possível também saber se a meta foi alcançada, o que a torna supérflua.

### **V.6.1 Metas subjetivas**

É possível que sejam definidas metas não mensuráveis de forma objetiva. Isso pode ser resolvido por meio de avaliações subjetivas. Por exemplo, uma meta comum é “melhorar o atendimento ao usuário”. Essa meta pode, em alguns casos, ser transformada em outra meta, como “diminuir o tempo de atendimento”, diretamente mensurável. Mas também pode ser medida por meio de entrevistas, com avaliações subjetivas, ou observação do serviço.

### **V.6.2 Levantando Objetivos e Metas**

A melhor maneira de levantar objetivos e metas é entender simultaneamente: o que o cliente deseja, o que está funcionando agora, quais os problemas atuais e quais oportunidades existem para um novo sistema.

As entrevistas iniciais para levantamento de necessidades de informação são geralmente feitas com executivos. Um bom conjunto inicial de questões que podem ser feitas é apresentado a seguir (sugeridas em Gillenson & Goldberg).

1. Quais seus objetivos?
2. Quais suas responsabilidades?
3. Que medidas você usa?
4. Que informações você precisa?
5. Quais são seus problemas de negócio?
6. Que mudanças você vê no futuro (que vão impactar a infraestrutura do seu negócio)?
7. Quais são os fatores críticos de sucesso?
8. Qual a informação mais útil que você recebe?
9. Como você classificaria as informações que recebe quanto à adequação, validade, duração, consistência, custo, volume, etc.?

Alguns pontos podem ser notados sobre essas perguntas. A pergunta sobre objetivos muitas vezes pode não ser muito bem respondida, assim a segunda pergunta, sobre responsabilidades, permite uma resposta mais prática e mais fácil de ser dada. A terceira pergunta visa buscar uma compreensão sobre os dados importantes para o entrevistado, abrindo uma sequência de perguntas sobre o tema. A quinta pergunta, sobre os problemas de negócio, é a mais importante do conjunto e deve ser dada a maior atenção e quantidade de tempo disponível. É interessante que o problema seja estruturado em um formato causa-efeito, por exemplo: "por causa da falta de dinheiro, o resultado é que não é possível comprar peças de reposição". Também é importante verificar se a causa primordial do problema é um processo ou um dado. Além disso, o entrevistador deve tentar obter alguma informação de valor (financeiro) sobre o impacto do problema, seja em valores absolutos ou relativos, objetivos ou subjetivos.

## **V.7 Os Requisitos Preliminares**

É importante registrar todos os requisitos que forem percebidos durante essa fase inicial do contato com o cliente. Esses requisitos aparecem informalmente, mas devem ser anotados diligentemente, pois talvez sejam até mesmo esquecidos por parte do cliente (o que não garante que sejam menos importantes, apesar de ser um indicativo).

### **V.7.1 Definindo o Escopo**

Uma boa estratégia complementar a especificação dos requisitos preliminares é a definição do escopo por meio de uma lista Dentro/Fora (in/out list) [11]. Essa lista é

simplesmente uma tabela contendo tópicos e a indicação se aquele tópico está dentro ou não do escopo do sistema.

Escopo do Sistema		
Tópico	Dentro	Fora
Receber pedido do cliente	✓	
Enviar nota fiscal		✓
Atender pedido parcialmente	✓	
Analisar crédito	✓	

**Tabela 4. Exemplo de uma tabela de definição de escopo.**

## **V.8 O sistema atual**

Uma das tarefas mais importantes é compreender, perfeitamente, como funciona o sistema atual. Desprezar o sistema atual, por mais antigo ou mal feito que ele seja, é um dos erros mais freqüentes dos desenvolvedores. Só podemos criar um sistema novo após conhecer perfeitamente o sistema atual, como funciona e porque funciona dessa forma.

Idealmente, deveríamos recuperar o “Modelo Ambiental” do sistema atual. Isso, porém, nem sempre é possível, devido às pressões de tempo e custo que sofremos na vida real. Uma descrição em português pode ser bastante satisfatória para sistemas pequenos. Para sistemas maiores pode ser necessário indicar outras fontes de obtenção de informação, como manuais existentes e os próprios programas.

Nesse ponto nossa abordagem é bem diferente da abordagem essencial tradicional [12], que defende a criação de um modelo da encarnação atual. Entendemos que o tempo e esforço necessários para levantar o sistema atual de forma detalhada, para depois transformá-lo em um novo sistema, com requisitos bastante diferentes, podem inviabilizar um projeto. Assim, normalmente por pressões do negócio, somos obrigados a tratar cada projeto segundo a metodologia essencial para derivar sistemas novos. Temos de nos perguntar se é válido documentar um sistema obsoleto para implementar um sistema novo. A resposta essencial é sim, pois não envolve o cálculo de custos. A resposta prática, baseada na experiência real de levantar sistemas novos para substituir sistemas já existentes é não. Não temos tempo, recursos ou pessoas para isso.

### **V.8.1 Problema do sistema atual**

Após o levantamento do sistema atual, devemos apontar, baseado nas reclamações dos clientes e em nossas observações, quais são os problemas do sistema atual.

Novamente é importante lembrar que os problemas de negócio são mais importantes que os problemas técnicos. Muitas lojas hoje em dia utilizam sistemas feitos em MS-DOS<sup>10</sup> porque eles atendem perfeitamente seus requisitos de negócio.

---

<sup>10</sup> Sistema operacional simples antecessor do Microsoft Windows.

Os problemas atuais, principalmente quando relacionados ao negócio, podem ser os principais fornecedores de metas para o sistema. Assim, se o problema for lentidão, o aumento do desempenho é uma meta importante. Da mesma forma, se o problema for a dificuldade de utilização, a meta pode ser facilitar a utilização do sistema e as métricas podem estar relacionadas ao tempo de aprendizado ou ao número de erros na entrada de dados.

## **V.9 Visão do novo sistema**

Devemos tentar responder como vai funcionar o novo sistema em uma descrição simples, em linguagem corrente. Chamamos essa descrição de **Visão** do novo sistema. Essa visão deve ser escrita com forte apoio do cliente, senão pelo próprio cliente.

Normalmente a Visão e a descrição do sistema atual têm o mesmo nível de abstração dentro de uma mesma proposta inicial.

A visão pode incluir o protótipo de algumas telas do novo sistema, com a finalidade de mostrar a diferença do sistema novo para o velho ou ainda mostrar como será o comportamento de uma nova finalidade.

A visão do sistema pode incluir não só o funcionamento do sistema, mas também expectativas de comportamento e de efeitos do sistema no negócio. Deve ficar claro que a visão do sistema é uma declaração do usuário, não necessariamente um comprometimento do desenvolvedor. Isso, porém, deve ficar claro na documentação.

A visão do sistema inclui também os requisitos já detectados, informalmente, pelo analista. Como descrito no capítulo anterior esses requisitos podem ser divididos em requisitos funcionais, requisitos de informação e requisitos não funcionais. Obviamente só estamos interessados em requisitos verdadeiros.

### **V.9.1 Oportunidades para o novo sistema**

Devemos incluir em nossa proposta oportunidades que detectamos a partir do nosso conhecimento do que é factível atualmente ou em futuro próximo.

As oportunidades que detectamos para um novo sistema estão normalmente ligadas a novas tecnologias ainda não utilizadas pelos clientes.

### **V.9.2 Pontos Críticos ou Pontos Chave**

Os pontos críticos (ou chave) de sucesso para um projeto são aquelas questões que, não estando diretamente relacionada ao desenvolvimento propriamente dito, são essenciais para o bom andamento do projeto.

Exemplos: compromisso de certos funcionários, fornecimento de certa informação, chegada de alguma máquina ou software, compromisso de entrega de dados ou equipamentos pelo cliente, etc.

Os pontos críticos devem ser levantados detalhadamente, pois os compromissos do desenvolvedor só poderão ser cumpridos caso sejam resolvidos satisfatoriamente.

### **V.9.3 Restrições**

Muitos sistemas têm restrições, que devemos considerar em nossas propostas. Um tipo importante de restrições são as exigências de implementação, como banco de dados, linguagem, sistema operacional, etc.

Quanto mais cedo forem detectadas as restrições, mais cedo o analista evitará que haja desperdício de recursos pela equipe de desenvolvimento.

## **V.10 Gerência de Riscos**

Todo projeto envolve um conjunto de riscos que deve ser tratado convenientemente para garantir o seu sucesso. A análise de riscos tem sido usada nos últimos anos como uma das melhores práticas de desenvolvimento de software.

Em projetos de software encontramos riscos ligados a vários fatores: tamanho, impacto do projeto no negócio, ao processo escolhido, ao comprador e clientes do projeto, à tecnologia utilizada, aos ambientes de desenvolvimento, de testes e operacionais, à equipe e outros.

As principais características de um risco são: sua incerteza, pois não sabemos se vai acontecer ou não, e a perda associada ao seu acontecimento. Ao analisar um risco, devemos nos preocupar com várias características, criando uma tabela de risco que descreva: o risco propriamente dito, uma categoria para cada risco, uma probabilidade e um impacto (incluindo a perda financeira associada à realização do risco). Normalmente o impacto é classificado em quatro níveis: desprezível, marginal, crítico e catastrófico (quando envolve a destruição de pessoas físicas ou jurídicas).

A partir dessa tabela devemos priorizar o risco. As melhores práticas indicam que devemos vigiar atentamente os 10 riscos principais. É comum classificar os riscos multiplicando suas probabilidades pelos seus impactos.

Ao longo do projeto os riscos serão tratados. Para isso, principalmente em grandes projetos, é criado um plano de gerenciamento de riscos, que inclui medidas para evitar, mitigar seus efeitos e monitorar os riscos.



Tabela de Riscos							
Risco	Class.	Prob.	Custo Financeiro	Impacto	Medidas para Evitar	Medidas de Mitigação	Fatores para Monitorar
Perda de um funcionário importante com conhecimento do negócio	Pessoal	15%	R\$20.000	Crítico	Ações de RH Perspectiva de treinamento	Treinamento de novos funcionários	Faltas não explicadas Descontentamento
Clientes não aceitarem o novo sistema	Negócio	10%	?	Catastrófico	Publicidade Interface amigável Manter o comportamento visível do sistema atual	Campanhas publicitárias	Volume de vendas Quantidade de reclamações

**Tabela 5. Uma tabela de riscos que pode servir para pequenos projetos**

A seguir apresentamos um plano de riscos para um risco específico, baseado em um plano utilizado em um projeto real, mostrando o risco principal. Notamos que, nesse caso, houve uma abordagem informal, sem a utilização de números para quantificar impacto e probabilidade. A “Complexidade Excessiva” é um risco comum e era o principal risco do projeto. Durante o projeto o risco se realizou, com forte impacto no prazo de entrega e causando dificuldades na adoção do sistema. Nesse caso os principais motivos foram a falta de experiência da equipe e a falta de interesse dos usuários em participar. Esse plano de riscos está acima da necessidade de grande parte das propostas feitas.

<b><i>Risco</i></b>	Complexidade Excessiva
<b><i>Motivo do Risco</i></b>	Nossa análise mostra que vários projetos de desenvolvimento de sistema já foram feitos na Prefeitura, todos atingindo um sucesso parcial, principalmente na questão de atendimento das necessidades dos usuários. A Secretaria de Fazenda engloba um grande número de atividades, todas interligadas. Devemos substituir de uma única vez todos os projetos.
<b><i>Impacto</i></b>	Crítico É um dos riscos mais graves, podendo ter impacto na entrega, tanto no prazo quanto na funcionalidade, e mesmo inviabilizar o desenvolvimento.
<b><i>Abordagem geral</i></b>	Em geral pelo desenvolvimento de uma arquitetura de software que permita o desenvolvimento de um grande número de funções sem aumentar a complexidade. Pela integração de funcionalidades por meio de “funcionalidades gerais”. Cuidado na análise. Envolvimento do usuário. Uso de equipe experiente.

<b><i>Abordagens específicas</i></b>	<p>Esse risco está sendo tratado da seguinte forma:</p> <ol style="list-style-type: none"> <li>1. Uso de reuniões de JAD para entender melhor o funcionamento comum do sistema.</li> <li>2. Desenvolvimento de funcionalidades básicas, como controle de processos de trabalho, acompanhamento de processos, protocolo, log e segurança, usados de forma conjunta em todos os projetos, cadastro de contribuintes, etc., diminuindo o custo de desenvolvimento de sistemas específicos.</li> <li>3. Desenvolvimento em paralelo, com múltiplas equipes coordenadas, acelerando a descoberta de requisitos e implementação de sistema.</li> <li>4. Uso da análise essencial, permitindo a reorganização das tarefas de forma a aumentar o conhecimento dos processos internos.</li> <li>5. Integração das informações por meio de um SGDB, diminuindo a complexidade atual, baseada em múltiplos sistemas que se comunicam por arquivos.</li> <li>6. Uso de protótipos junto aos usuários <ul style="list-style-type: none"> <li>• O risco aumentará de nível se até agosto não for desenvolvido um modelo claro das necessidades do sistema, incluindo uma especificação clara do escopo do sistema.</li> <li>• O risco aumentará de nível se não for possível organizar reuniões periódicas com todos os responsáveis pelos sistemas sendo substituídos.</li> </ul> </li> </ol>
<b><i>Responsáveis</i></b>	O Coordenador Geral e os Analistas devem acompanhar a complexidade do projeto como um todo
<b><i>Tempo</i></b>	Até final de agosto devemos ter o escopo do sistema (eventos a serem respondidos e relatórios a serem gerados) bem definido
<b><i>Esforço</i></b>	Estimamos que a equipe atual represente o esforço necessário para o levantamento dos dados necessários até o prazo especificado.
<b><i>Monitoramento</i></b>	<p>Atraso nos prazos iniciais de entrega da análise.</p> <p>Reprovação de documentos de análise.</p> <p>Dificuldade de marcar entrevistas e reuniões.</p> <p>Reprovação de protótipos.</p> <p>Solicitações de requisitos originalmente fora do escopo.</p>
<b><i>Probabilidade</i></b>	<p>Alta</p> <p>O sistema é bastante complexo e precisa ser tratado como tal.</p>

**Tabela 6. Exemplo de plano de riscos inicial**

## **V.11 Construindo um Glossário**

Uma das atividades mais importantes da análise é compreender o domínio da aplicação. Essa compreensão só pode acontecer se o analista souber o significado exato das palavras típicas do negócio do cliente. Assim, desde o início da análise o analista deve construir um glossário, ou seja, um dicionário especializado nos termos do negócio do cliente.

Não podemos enfatizar demais a importância de aprender a linguagem do negócio do cliente. Isso não só facilita a comunicação como dá ao cliente uma confiança maior no analista.

## **V.12 Entendendo o que será feito**

Existe uma informação mínima a ser levantada para o desenvolvimento de um sistema. Essa informação compõe a Proposta Inicial e se divide nas seguintes partes:

- Solicitação do cliente
- Objetivo
- Problemas
- Oportunidades
- Stakeholders
  - Objetivos de Negócio e Interesses
- Metas e Métricas
- Sistema atual
- Visão do novo sistema
  - Escopo
- Requisitos preliminares
  - Requisitos funcionais preliminares
  - Requisitos de informação preliminares
  - Requisitos não funcionais preliminares
- Pontos críticos (ou pontos chave)
- Restrições
- Riscos
- Glossário

## **V.13 Proposta Comercial**

Se necessário, a proposta inicial se encerra com os termos comerciais sendo propostos ao cliente, incluindo preço, tempo de desenvolvimento, formas de cobranças, etc.

É de praxe no mercado, apesar de não recomendado, o cliente aceitar uma proposta e dispensar a assinatura de um contrato ou discutir os termos legais do contrato enquanto se inicia o trabalho de análise.

### **V.13.1 Calculando Preço e Custo**

Antes de começar essa discussão, devemos deixar clara a diferença entre preço e custo: preço é o que cobramos para fazer um sistema, custo é quanto gastamos para desenvolvê-lo.

Não é razoável ensinar como calcular o **preço** de um sistema em um curso de análise, pois esse é um problema de mercado, não um problema de desenvolvimento de sistemas. O que um analista pode fazer é calcular o **custo** de desenvolvimento<sup>11</sup> de um sistema.

Apenas no final de um projeto temos certeza absoluta do custo que tivemos para desenvolvê-lo. Até lá, o que podemos fazer é levantar, de forma mais ou menos educada, uma previsão de custos para o projeto. Essa previsão deve ser atualizada constantemente enquanto o projeto caminha.

Devemos compreender que o grande fator de custo no desenvolvimento de um software é a mão de obra. Normalmente esse custo é levantado em homem-hora ou homem-mês. O uso dessa mão de obra é proporcional<sup>12</sup> a complexidade do projeto, fruto de seu tamanho e de suas exigências operacionais ou funcionais.

Existem várias técnicas de previsão do tamanho de um sistema. No capítulo “Qual o tamanho do software” estudaremos algumas dessas técnicas. Na prática, porém, devemos compreender que a maioria das empresas ainda as desconhece ou não as implementa. Nesse caso, elas acabam utilizando a experiência de seus funcionários para, informalmente, fazer previsões que normalmente se mostram pouco acuradas.

Quanto ao preço, podemos fazer algumas observações. A primeira é que não há necessariamente uma relação direta entre custo e preço. Apesar de muitas empresas de desenvolvimento fazerem uma previsão de custo e calcularem o preço em cima dessa previsão, usando margens de lucro e de risco previamente acordadas, o verdadeiro preço vem do valor de mercado do produto, ou ainda do ganho previsto. Isso pode inclusive ser uma boa oportunidade de negócios. Muitas empresas, ao perceberem que um produto específico vai dar um lucro direto ao seu cliente, propõem fazer esse produto de graça em troca de um percentual sobre o lucro. Em longo prazo isso pode ser altamente vantajoso para a empresa de desenvolvimento, enquanto que para o cliente pode ser a única forma de conseguir um sistema que exigiria um investimento inicial muito alto para seu caixa.

## V.14 O Resumo Executivo

O resumo executivo deve permitir que o leitor tenha uma visão completa do texto do documento em uma página. Resumos executivos têm esse nome por partir da hipótese que um executivo, ao tomar uma decisão, não tem tempo para ler longos documentos. Na prática eles leriam o resumo executivo e folheariam os documentos.

---

<sup>11</sup> E ainda implantação, manutenção e operação.

<sup>12</sup> De forma exponencial

**Se você nunca fez análise de sistemas, essa proposta pode parecer incompreensível. Isso acontece porque para fazer a proposta inicial devemos fazer, de maneira extremamente simplificada, uma análise de sistema. Assim, se não compreende o que deve ser feito, espere alguns capítulos.**

## **V.15 Estrutura da Proposta Inicial**

Uma proposta de trabalho pode se configurar de várias formas, em função de que pontos desejamos ressaltar ou em que ordem desejamos apresentá-la. A seguir apresentamos uma estrutura possível para uma proposta inicial que uma empresa de desenvolvimento de software poderia fazer para um cliente.

1. Resumo Executivo
2. Apresentação do Documento
  - 2.1. Identificação do Projeto
    - 2.1.1. Objetivo<sup>13</sup>
  - 2.2. Identificação do Cliente
  - 2.3. Identificação do Prestador de Serviço
  - 2.4. Histórico do Apresentador de Serviço
3. Proposta Técnica
  - 3.1. Pequena Descrição da Solicitação do Cliente
  - 3.2. Descrição Sucinta do Sistema Atual
    - 3.2.1. Descrição
    - 3.2.2. Identificação de Problemas
    - 3.2.3. Identificação de Oportunidades
  - 3.3. Descrição do Sistema Proposto
    - 3.3.1. Objetivo do Sistema
    - 3.3.2. Stakeholders
      - 3.3.2.1. Objetivos de Negócios e Interesses
    - 3.3.3. Metas
      - 3.3.3.1. Métricas para avaliação das Metas
    - 3.3.4. Visão
      - 3.3.4.1. Requisitos Funcionais Iniciais
      - 3.3.4.2. Requisitos de Informação Iniciais

---

<sup>13</sup> O Objetivo do projeto é implementar o sistema, que tem um objetivo para cumprir no negócio.

- 3.3.4.3. Requisitos Não Funcionais Iniciais
    - 3.3.4.4. Escopo
  - 3.3.5. Pontos Críticos
  - 3.3.6. Restrições
- 4. Glossário
- 5. Proposta Comercial
  - 5.1. Cronograma Proposto
  - 5.2. Investimento Proposto<sup>14</sup>
  - 5.3. Outras Cláusulas (opcionais)
    - 5.3.1. Exclusividade
    - 5.3.2. Forma de pagamento
    - 5.3.3. Reajustes
    - 5.3.4. Renegociação
    - 5.3.5. Confidencialidade

## **V.16 Exercício**

### **V.16.1 Projeto 1: Livraria ABC**

Faça uma proposta inicial para a Livraria ABC, de acordo com o item □.

### **V.16.2 Projeto de curso**

Os grupos devem fazer uma proposta inicial, de acordo com o item V.15 e apresentá-la ao professor para discussão e aprovação.

---

<sup>14</sup> Sutilmente, o preço do sistema é proposto como “investimento”, o que de fato é verdade na visão da empresa contratante.

## Capítulo VI. Modelo de Negócio

---

*The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work.*

*John Von Neumann*

Organograma
Funções de Negócio
Processos de Negócio
EPC
Diagrama de Atividades
Regra de Negócio

A Modelagem de Negócio não faz parte da modelagem essencial ou da modelagem estruturada, mas cada vez mais vem sendo usada como uma ferramenta principal ou de auxílio do processo de desenvolvimento de software, visando o levantamento completo dos requisitos do sistema.

Nesse capítulo trataremos de diferentes formas de modelagem de negócio:

- Organograma<sup>15</sup>
- Modelagem de Funções de Negócio
- Modelagem de Processos
- Modelagem de Regras de Negócio

Um organograma é uma descrição da organização de uma empresa, amplamente divulgada, descrevendo as áreas da empresa e as hierarquias entre elas. O Organograma é ferramenta essencial na compreensão de uma empresa e suas linhas de poder.

A modelagem de funções de negócio permite a compreensão do funcionamento da empresa sem sofrer a intervenção da forma de organização da empresa. De certa forma, pode ser desenvolvido como tanto como um modelo da encarnação do sistema atual como quanto uma ferramenta de substituição ou complementação da análise essencial.

A modelagem de processos demonstra como funciona a empresa, passo a passo, no seu dia a dia. A partir dela pode ser possível levantar os pontos a serem automatizados de um processo e como os processos realmente realizados diferem dos processos normatizados da empresa.

A modelagem de regras de negócio permite a compreensão da empresa de forma mais detalhada que os modelos anteriores. As regras de negócio podem ser utilizadas para ajudar a levantar o modelo essencial, o modelo conceitual de dados, ou para ajudar a

---

<sup>15</sup> O organograma é uma das formas mais simples e antigas de modelar uma empresa.

implementá-los. Em alguns métodos, pode até mesmo ser utilizada no lugar desses modelos.

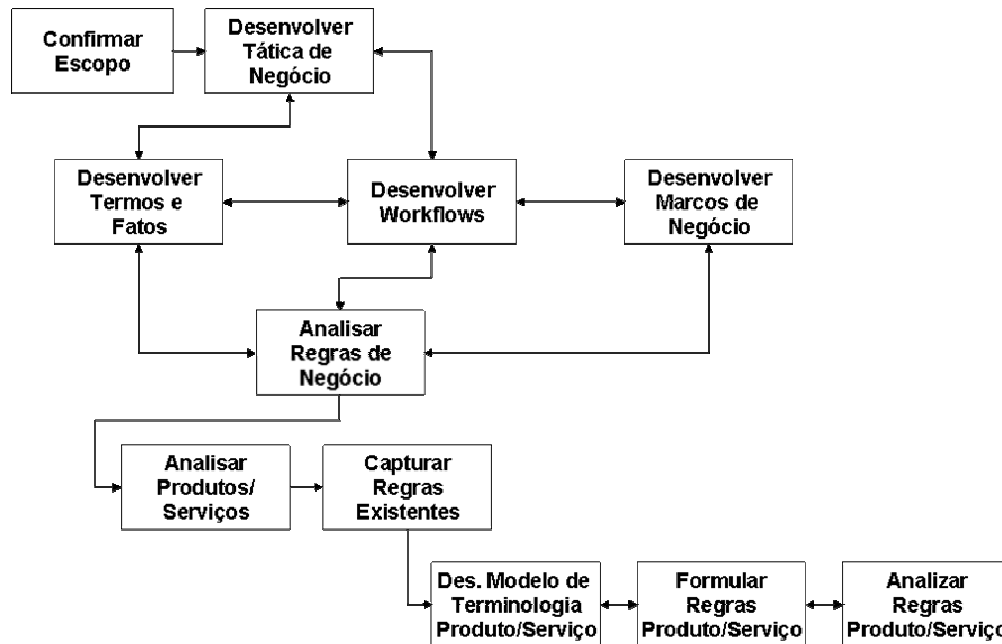


Figure 19. Passos da Modelagem de Negócios adaptado de Ross e Lam [13].

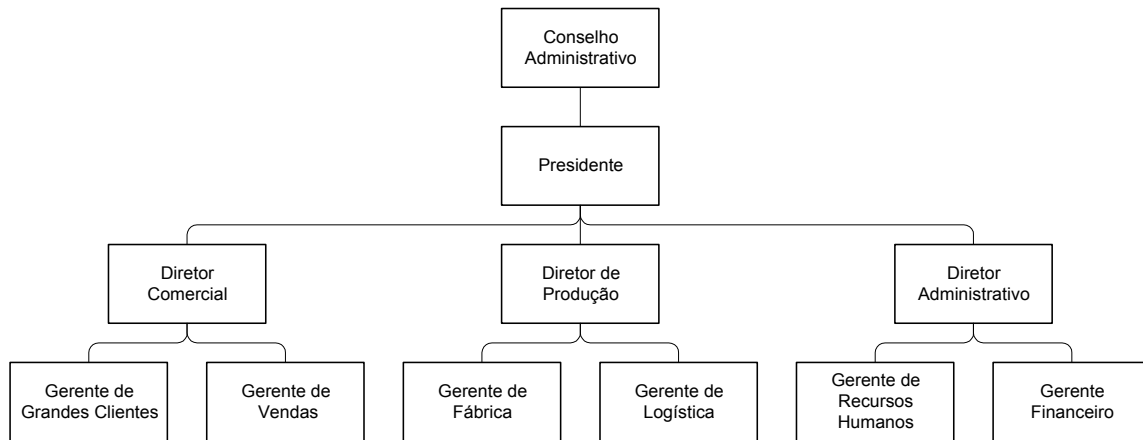
## VI.1 O Organograma

Organogramas descrevem cargos, por meio de retângulos, e linhas hierárquicas, por meio de linhas. Alguns autores utilizam uma notação levemente mais complicada com o objetivo de descrever diferenças sutis em uma organização.

Em geral o analista não precisa levantar o organograma, pois a empresa já o possui, mas é comum que haja algumas mudanças. É importante também levantar não só a hierarquia de cargos, mas também quem ocupa cada cargo, principalmente para apoio às tarefas de análise.

A importância de conhecer o organograma da empresa se reflete tanto na modelagem propriamente dita, pois ele fornece a descrição da empresa que será convertida objetos do modelo, como no processo de modelagem, pois a partir do organograma temos o conhecimento de cargos e responsabilidades, definindo pessoas a serem entrevistadas.





**Figura 20. Um exemplo de organograma simples, contendo apenas cargos.**

Um organograma pode ser utilizado para representar diferentes formas de subordinação, como a subordinação direta (onde o subordinado deve cumprir as ordens de seu chefe), a assessoria (onde o assessor fornece conselhos e pareceres) e a subordinação funcional (onde o superior pode determinar funções e métodos a outras áreas). Normalmente a subordinação direta é representada por uma linha cheia vertical, a assessoria por uma linha cheia horizontal e a subordinação funcional por uma linha pontilhada.

Ao levantar o organograma, pode ser interessante também levantar as descrições dos cargos, se elas existirem (o que não é comum, principalmente em pequenas e médias empresas).

## VI.2 Modelagem de Funções de Negócio (IDEF0)

As funções de uma instituição são os grupos de processos que gerenciam um recurso básico da organização [14] Elas descrevem o que a organização faz, devendo estar de acordo com os objetivos da empresa.

Funções de negócio mantêm a organização em operação, formando um conjunto de atividades (processos) relacionadas que tem como objetivo alcançar a missão ou objetivos da empresa.

Segundo Modell [3] uma função deve:

- Ser identificável
- Ser definível, por si só e em termos das atividades e responsabilidades associadas.
- Não necessariamente ser mensurável (o que é influenciado pelo seu grau de abstração);

Além disso, ainda segundo Modell, uma função pode:

- Ser uma área principal de controle ou atividade da organização

- Ser composta de uma ou mais subfunções
- Ser realizada em uma ou mais áreas<sup>16</sup>
- Ser realizada por um indivíduo, um grupo, grupos de grupos, áreas da organização ou até por toda a organização.
- Envolver um ou mais atividades distintas, sejam elas dependentes ou independentes.
- Ser identificada e definida mesmo que não seja executada.

Também é possível entender dois tipos de funções: as de negócio, ou seja, aquelas presentes na cadeia de valor da empresa e que têm relação com o aspecto operacional do negócio, e as de administração (ou suporte).

IDEF0, Integration Definition Language 0, ou “Integration Method for Function Modelling” é uma forma de representar sistemas e empresas por meio de uma rede de atividades interconectadas. Esses modelos representam funções do sistema, relacionamentos funcionais e dados que suportam a integração do sistema.

Segundo o padrão IFIPS 183, “um modelo IDEF0 é composto por uma série hierárquica de diagramas que apresentam, gradativamente, um nível maior de detalhe, descrevendo funções e suas interfaces no contexto de um sistema....”. A descrição a seguir é fortemente baseada e algumas vezes a tradução literal do padrão IFIPS 183.

Os componentes da sintaxe de IDEF0 são diagramas (formados de caixas, setas, linhas), textos e glossários.

As funções, definidas como atividades, processos ou transformações, são representadas caixas, que são conectadas uma às outras por meio de setas com significados distintos, representando dados ou objetos relacionados a cada função, como na Figura 21.<sup>17</sup> Todas as caixas e conectores são rotulados, sendo que um dicionário deve ser usado para definir detalhadamente cada rótulo. Todas as caixas devem também receber um número (no canto inferior direito).

Diagramas IDEF0 são construídos de forma hierárquica, a partir de um diagrama inicial, chamado A-0 (A menos zero)<sup>18</sup> que sempre contém uma única atividade, numerada 0, a partir do qual são feitos detalhamentos<sup>19</sup> sucessivos. Cada detalhamento é representado por outro diagrama, contendo atividades interligadas, permitindo uma compreensão *top-down* do processo sendo descrito. O método limita o número de subfunções para uma função entre 3 e 6.

IDEF0 não representa necessariamente a sequência de atividades, apenas as interações entre as atividades. Pode ser usado tanto para representar processos já

---

<sup>16</sup> Não há necessariamente um mapeamento direto entre funções e o organograma

<sup>17</sup> A metodologia IDEF0 é derivada de uma metodologia anterior conhecida como SADT.

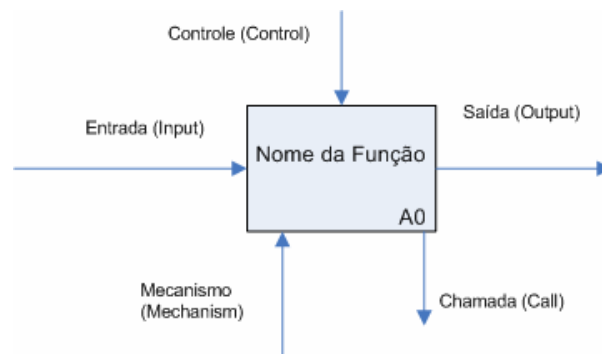
<sup>18</sup> O processo inicial é sempre o A-0, não existindo um processo B-0 em nenhum caso. A letra “A” vem de atividade.

<sup>19</sup> Também conhecidos informalmente como “explosões”.

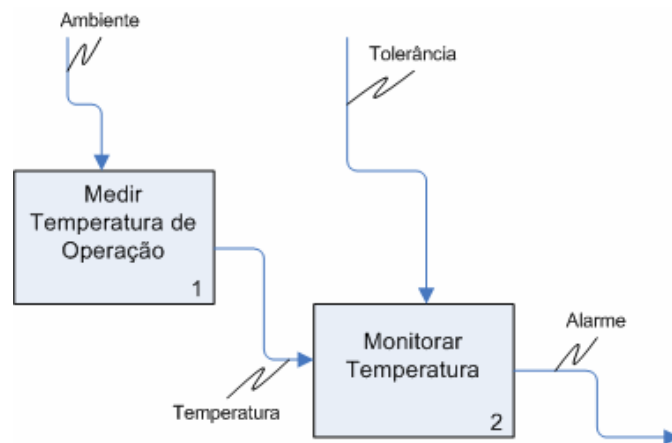
existentes quanto para representar processos novos. No primeiro caso se sugere uma estratégia de construção *bottom-up*, no segundo caso a estratégia *top-down* pode ser mais apropriada.

A função de cada uma das setas é dada pelo seu posicionamento na caixa da atividade, como descrito na figura a seguir.

- Entradas são dados ou objetos que são transformados ou consumidos na saída pelo processo.
- Controles são condições necessárias para produzir a saída correta, podendo ou não ser transformados na saída. Controles são restrições na operação do processo.
- A saída apresenta o resultado do processo.
- Os mecanismos ou recursos são os meios necessários para a realização da função, porém não são consumidos para produzir a saída.
- É possível que uma seta saia da parte de baixo do diagrama. Isso indica uma “chamada de função”, que na verdade representa que o processo chamador é explicado pelo processo chamado.



**Figura 21. Um processo em IDEF0 e suas entradas e saídas.**



**Figura 22. Exemplo, em um fragmento de IDEF0, dos usos de controles e entradas.**

Algumas regras sintáticas:

- Diagramas são identificados (Node) na forma An, onde n é um número.
- O diagrama A-0 (A menos zero) contém só uma caixa (a caixa zero), que é expandida no diagrama A0 (A zero).
- Pode existir, opcionalmente, um diagrama que coloque o diagrama A-0 dentro de um contexto maior, chamado A-1 (A menos 1).
- O número de um diagrama é formado pelas iniciais do autor e um número seqüencial.
- Um diagrama por ser FEO (ver abaixo) ou conter apenas texto ou glossário. Nesse caso, o nó recebe o seu identificador seguido respectivamente das letras F, T ou G.
- Os diagramas são desenhados em formulários padronizados.

IDEF KIT DIAGRAM FORM

Used at	Author:	Date:	WORKING	READY	DATE	Context:
	Project:	Rev:	DRAFT			
	Notes: 1 2 3 4 5 6 7 8 9 10		RECOMMENDED			
			PUBLICATION			

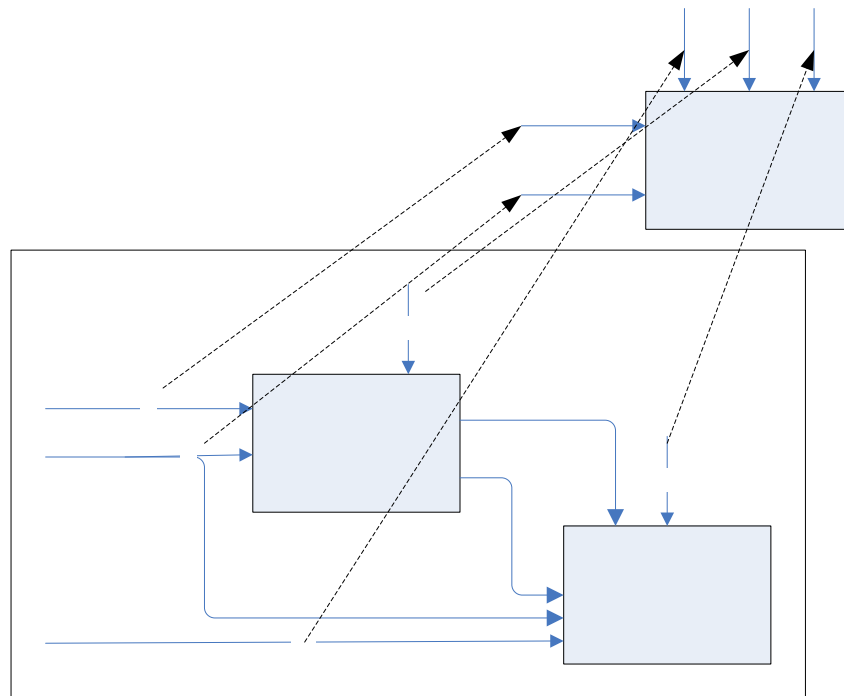
  

Node:	Title:	Number:	Page:
-------	--------	---------	-------

**Figura 23. Formulário padrão para um diagrama IDEF0**

- Caixas denotam atividades, por isso devem ser ou conter verbos em seu nome.
- Cada caixa é numerada adicionando mais um número inteiro entre 1 e 6 (número máximo de caixas em um diagrama) ao número da caixa pai.

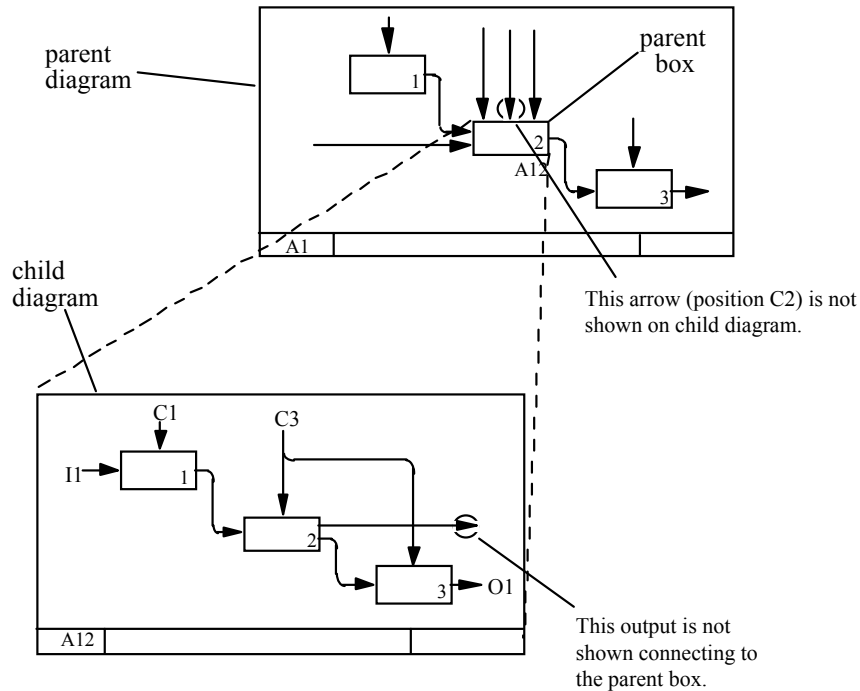
- As caixas dos diagramas são numeradas 1, 2, 3,...
- As caixas são indicadas pelo nome do diagrama adicionadas do número da caixa (a caixa 1 do diagrama A1 se chama A1.1)
- Quando uma caixa é detalhada em outro diagrama, é colocada uma referência a esse diagrama abaixo do canto inferior esquerdo. Essa referência é conhecida como DRE.
- Cada caixa (função) é representada por um rótulo centralizado formado por um verbo ou um verbo-objeto e um segundo rótulo, no canto inferior direito, representando a identificação (“número”) do rótulo.
- Caixas são retangulares com cantos arredondados.
- Cada diagrama deve conter todas as setas que entram e saem do seu diagrama superior, que podem ser indicadas pela seguinte notação (ICOM):
- Controle: C1, C2, C3..., contados da esquerda para a direita na caixa explodida.



**Figura 24. Exemplo de numeração ICOM e seu significado**

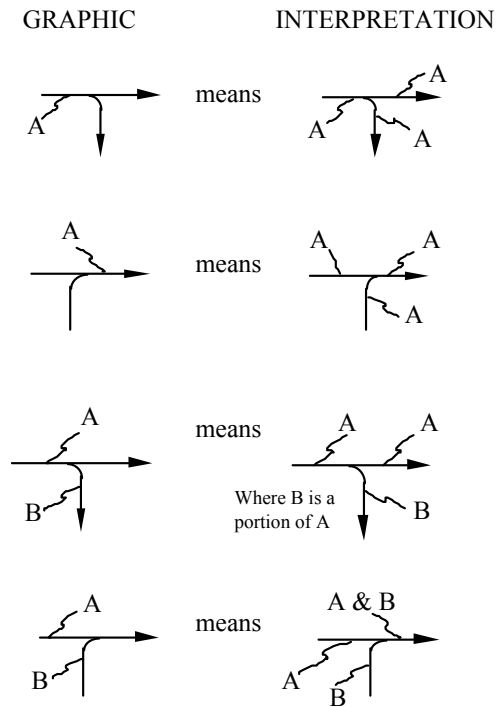
- Entradas: I1, I2, I3, contadas de cima para baixo na caixa explodida.
- Saídas: O1, O2, O3, contadas de cima para baixo na caixa explodida.
- Mecanismos: M1, M2,... contados da esquerda para a direita na caixa explodida.
- Setas denotam objetos ou dados (por isso devem ser substantivos)
- As setas só fazem curvas de 90°, não apresentam inclinações.

- Setas não representam fluxo, mas sim como os dados e objetos necessários para o funcionamento de uma função são obtidos.
- Setas podem ser “tuneladas”. Isso significa que não apareceram no diagrama filho de uma caixa, mas apenas em “diagramas netos”. Para “tunelar” uma seta coloque parênteses em torno da ponta ou da raiz da seta (formando um “túnel”)



**Figura 25. Exemplo de tunelamento e DRE (na caixa 2 do diagrama A1)**

- Uma seta pode ser dividida ou setas podem ser agregadas. Os segmentos resultantes devem ser nomeados adequadamente para representar as partes. Por exemplo, uma seta “identificação de usuário” e uma seta “solicitação de serviço” podem ser unificadas na seta “solicitação identificada”. O inverso também pode acontecer.
- Se uma seta contém dados e controles, ou se estamos incertos se contém controle ou dados, devemos mostrá-la como controle.




**Figura 26. Fork e join não rotulados têm interpretações padronizadas**

- Uma caixa possui
- No mínimo 1 seta de controle
- No mínimo 1 seta de saída
- No máximo 1 seta de chamada
- Zero ou mais setas de entrada e mecanismo
- Informação de suporte pode ser colocada em um texto associado ao diagrama.
- Abreviações, jargão, etc., devem ser colocados em um glossário (único para o modelo).





- Ao se referenciar a objetos do diagrama, a seguinte notação deve ser usada:

<b>Notação de referência para o IDEF0</b>	
<b>Notação de Referência</b>	<b>Significado</b>
2I1	Caixa 2 Entrada 1
O2	A seta cujo código ICOM é O2 (Saída 2)
2O2 para 3C1 ou 2o2 para 3c1	A seta de 2O2 para 3C1 (I, C, O ou M podem ser maiúsculas ou minúsculas).
I2 para 2I3 para 2O2 para (3C1 e 4C2)	Da seta com código ICOM I2 para a caixa 2, entrada 3, através da ativação da caixa 2 que fornece a saída 2, para a disponibilidade (por meio de um <i>fork</i> ) dessa saída como controle 1 na caixa 3 e controle 2 na caixa 4.
A21.3C2	No diagrama A21 nesse modelo, veja o controle 2 da caixa 3. O ponto significa “olhe especificamente para”.
A42. 	No diagrama A32, veja a nota de modelo 3.
A42. 3	Notação opcional para “No diagrama A32, veja a nota de modelo 3”, usando barras verticais em vez de uma caixa para identificar a nota.
A42.3	No diagrama A42 desse modelo, veja a caixa 3.
MFG/A42.1	NO diagrama A42 do modelo MFG veja a caixa 1

**Tabela 7. Notação de referência para o modelo IDEF0, segundo IFIPS 183.**

O padrão IDEF0 pede que um modelo seja publicado como no formato dado na figura a seguir.

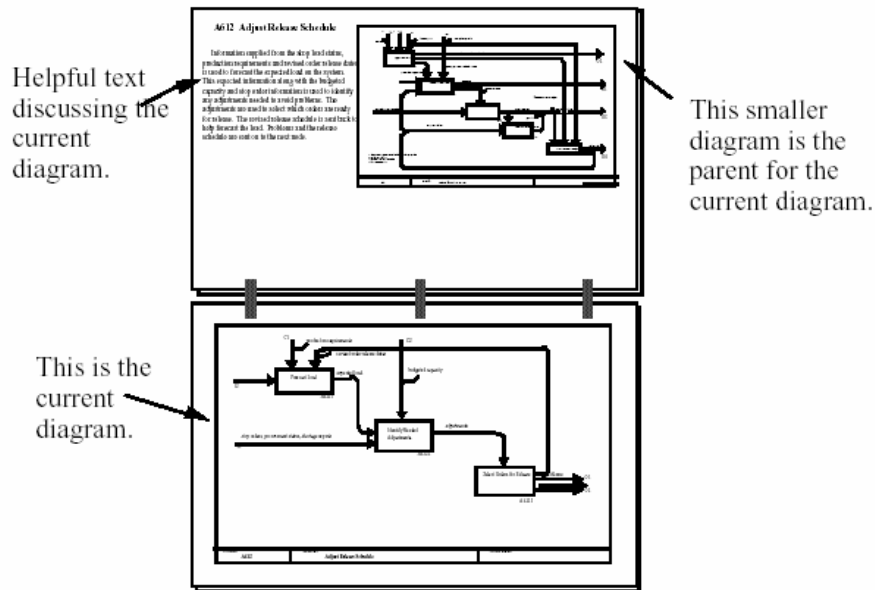


Figura 29. Formato de publicação pedido pelo padrão IDEF0

## VI.3 Processos de Negócio

Processos de negócio são grupos de decisões e atividades, logicamente relacionadas, requeridas para o gerenciamento de recursos da empresa<sup>20</sup>.

Podemos entender “processos de negócio como uma sequência de passos e decisões, iniciadas em resposta a um evento de negócio, que alcança um resultado específico e mensurável, tanto para o consumidor do processo como para outros interessados (*stakeholder*).”[15] Além disso, é necessário que identifiquemos instâncias específicas dos resultados.

Não é trivial identificar processos, pois eles acontecem dentro da organização de forma esparsa, provavelmente envolvendo diversas pessoas e departamentos. Também não é trivial representar processos, pois corremos vários riscos, como fazer uma representação muito complexa ou muito simples, ser impreciso ou utilizar o método de forma errada.

Normalmente, sistemas de informação são utilizados para automatizar processos de negócios. Pode ser necessário, antes de fazer o levantamento de requisitos de um sistema, levantar como funciona o processo onde ele está inserido ou que vai substituir.

Nesse tipo de modelagem estamos preocupados com a forma em que os processos são executados dentro da empresa. Existem várias formas de se tratar a descrição de processos atualmente, variando em diferentes níveis de complexidade.

<sup>20</sup> Esta é a definição do BSP

### VI.3.1 Fluxogramas

Fluxogramas são provavelmente a forma mais tradicional de modelar processos. Atualmente, fluxogramas são poucos utilizados, tendo sido substituídos por outras formas, semelhantes, que foram criadas com a finalidade de evitar problemas comuns encontrados na criação dos fluxogramas.

É importante frisar que fluxogramas podem ser utilizados em vários níveis do desenvolvimento de sistemas. Seu uso mais comum, no passado, era na especificação de algoritmos, mas está totalmente superado e praticamente abandonado em todas as suas formas. Seu uso na especificação de processos também está em franca decadência, mas nesse caso o que se vê é a sua substituição por diagramas mais modernos (incluindo o uso de fluxogramas em diagramas de raia<sup>21</sup>).

### VI.3.2 EPC

EPC é a sigla em inglês para *Event Driven Process Chain* (Cadeia de Processos Dirigida por Eventos). Esse método é parte simplificada do método ARIS usada para modelagem de processo e tem grande aceitação no mundo, estando muitas vezes associado à implantação de sistemas de ERP SAP/R3.

Nesse método, um processo é modelado segundo fluxo de eventos e funções.

As principais primitivas, descritas na figura abaixo, são:

- Funções, que representam atividades, tarefas ou passos do processo que precisam ser executadas. Funções são possivelmente iniciadas ou habilitadas por eventos. Funções possivelmente geram eventos. Funções consomem recursos, exigem gerenciamento, tempo, e atenção. Funções podem representar:
  - Atividades tangíveis
  - Decisões (mentais)
  - Processamento de Informações
- Eventos, que representam situações, ou estados do sistema, antes ou depois da execução de uma função. Um evento pode ser uma pré-condição ou uma pós-condição para uma função. Um evento não consome tempo nem recursos por si só.
- Conectores Lógicos, que permitem a unificação e separação de fluxos segundo os conceitos de E, OU ou OU-exclusivo.
- Caminho, que indica que um passo é descrito por meio de um diagrama completo EPC.

---

<sup>21</sup> swimlane diagrams

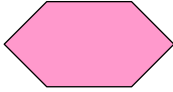






Tipo	Símbolo	Definição
Evento		Um Evento descreve uma ocorrência que causa um efeito (função)
Função		Uma função descreve uma transformação (uma mudança no estado do sistema)
Conectores	  	Um conector estabelece conexões lógicas entre eventos e funções
Fluxo		Um fluxo descreve uma relação lógica ou temporal entre funções e eventos
Caminho		Um caminho estabelece uma relação entre processos.

Figura 30. Principais componentes de um EPC

A figura a seguir demonstra um diagrama EPC simples demonstrando parte de um processo de recebimento de pedidos.

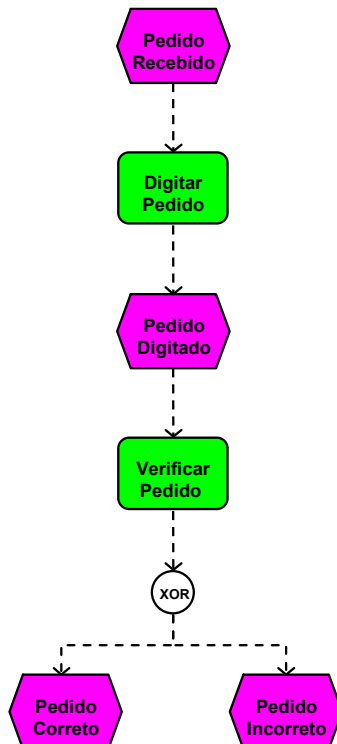


Figura 31. Exemplo de um EPC de um processo de recebimento de pedidos.

Segundo a versão original de EPCs, sempre deveria haver um evento entre dois processos. Atualmente é permitido que uma sequência de processos não tenha nenhum evento entre eles.

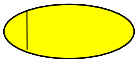


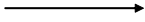



De acordo com as regras sintáticas para EPCs, é possível que um processo produza um ou mais eventos simultaneamente, pelo conector E, ou não, pelos conectores OU ou OU-EXCLUSIVO. Já um evento só pode habilitar um grupo de processos simultaneamente, pelo conector E, não sendo viável que de um evento se tenha uma opção de caminho, não sendo possível a partir de um evento alcançar diretamente um conector OU ou OU-EXCLUSIVO.

#### **VI.3.2.1 eEPC**

eEPC é a sigla em inglês para *Extended Event Driven Process Chain* (Cadeia de Processos Dirigida por Eventos). Nessa extensão é possível declarar mais algumas informações sobre o processo sendo descrito.

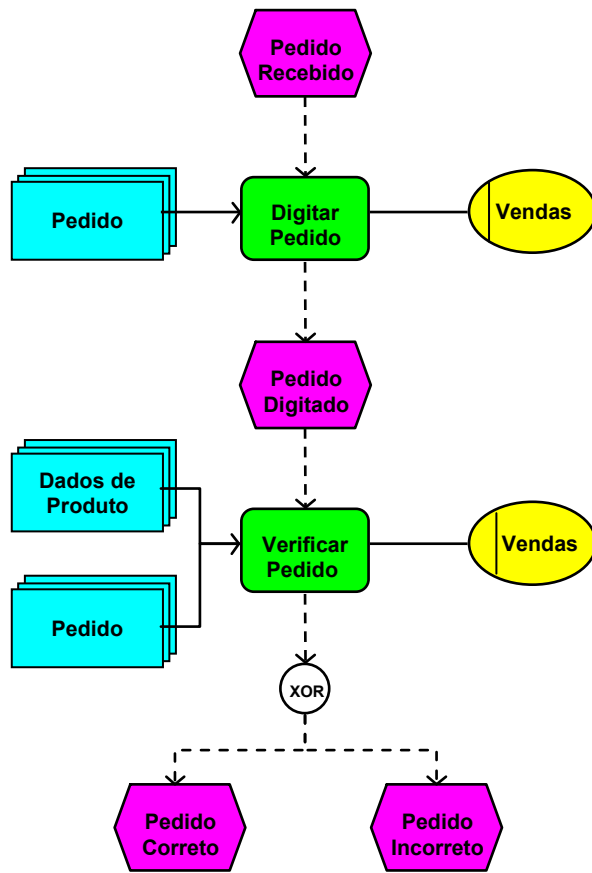
Os principais elementos adicionais em um eEPC são:

- Unidades Organizacionais, que representam departamentos envolvidos em um processo.
- Pessoas, que representam pessoas ou papéis envolvidos em um processo.
- Informação ou dados, que representam informação utilizada ou gerada em um processo.

Tipo	Símbolo
Unidade Organizacional	
Informação	
Pessoa ou Cargo	
Fluxo de Informação	
Relações Organizacionais	
Produto ou Serviço	
Objetivo	

**Figura 32. Elementos complementares dos diagramas eEPC.**

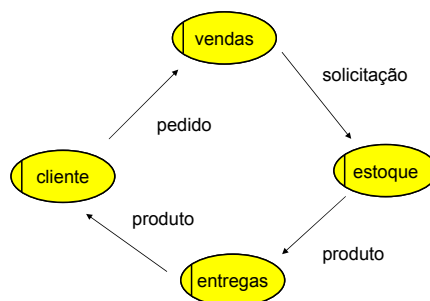
A seguir, apresentamos um exemplo de diagrama EPC para demonstrar os usos dos elementos adicionais.



**Figura 33. Exemplo de eEPC**

#### VI.3.2.2 Formas diferentes dos diagramas

É possível desenhar os EPCs de diferentes formas. Uma dessas formas permite que iniciemos apenas com as comunicações entre unidades, como é mostrado na figura a seguir. Isto permite um conhecimento inicial do processo que pode ser muito útil na sua discussão durante reuniões ou entrevistas.



**Figura 34. EPCe baseado apenas em unidades da organização, observe que nesse nível são permitidos *loops*.**

### **VI.3.2.3 EPC e 5W2H**

Um evento indica quando (*when*) algum processo, função ou tarefa deve ser iniciado.

Uma função ou tarefa indica o quê (*what*) deve ser feito.

Uma unidade organizacional indica quem (*who*) deve fazer.

### **VI.3.2.4 Passos para construir modelos EPC/EPCE[16]**

- Identifique os eventos que iniciam as funções, que servem como gatilhos para o processo se iniciar. Normalmente vem de “fora para dentro” do processo.
- Identifique as funções do processo, associando-as aos eventos que as iniciam e sua sequência.
- Decomponha as funções, verificando se são ações lógicas simples ou compostas, executadas por uma ou mais pessoas (ou ainda um sistema de computador). Verifique também se a função é uma transação isolada ou pode ser dividida em partes, se pode ser interrompida em um momento específico e se existe um evento que a interrompa ou que a faça funcionar novamente.
- Analise os eventos novamente, definindo-os e refinando-os se necessário. Garanta que são necessários e suficientes para iniciar a função. Analise se existem casos especiais nos quais as funções acontecem ou não. Use operadores lógicos para montar as relações entre os eventos.
- Identifique os eventos de finalização e as saídas (tanto de material quanto de informação). Procure identificar quem processos e pessoas no resto da organização que dependem do processo sendo analisado.

EPCs podem ser muito pequenos ou enormes, dependendo unicamente do tamanho do processo que está sendo mapeado.

### **VI.3.2.5 Regras de ouro de EPCs[16]**

- Não existem nós isolados
- Não existem loops
- Funções e eventos têm apenas uma entrada e uma saída
- Operadores lógicos contêm vários fluxos de entrada e um de saída, ou um único fluxo de entrada e vários de saída.
- Conexões entre operadores lógicos são acíclicas.
- Dois nós só podem possuir um único link entre eles
- Existe um evento inicial e um evento final
- Eventos não tomam decisões, logo só possuem uma saída.



### VI.3.3 Diagramas de Atividade

O Diagrama de Atividade é uma das formas que UML propõe para modelar os aspectos dinâmicos de um sistema, sendo basicamente um tipo de fluxograma mostrando como o controle flui entre atividades. Um diagrama de atividades também é um tipo de diagrama de transição de estados que permite a modelagem de concorrência.

Em um diagrama de atividades, cada atividade é modelada em um estado (*activity state*). Atividades podem, eventualmente, ser detalhadas em ações (*action states*), que são atômicas. Não existe uma diferença na notação entre atividades e ações, ambas são representadas pelo mesmo símbolo.

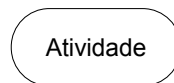


Figura 35. Símbolo para uma atividade (com texto).

Quando uma atividade ou ação é terminada, o controle passa imediatamente para o próximo estado, o que é indicado por meio de uma transação (seta). No diagrama, o fluxo sempre se inicia em um estado inicial e termina em um estado final.

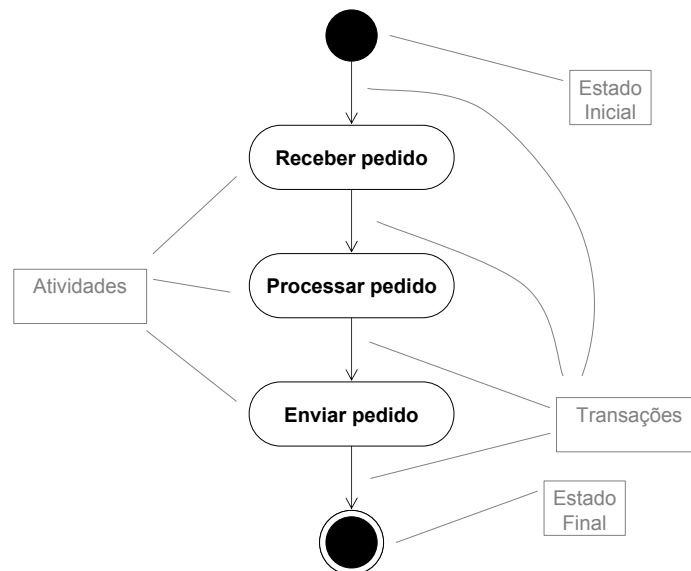
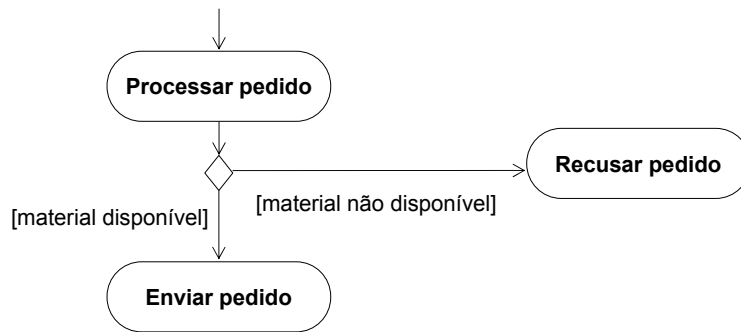


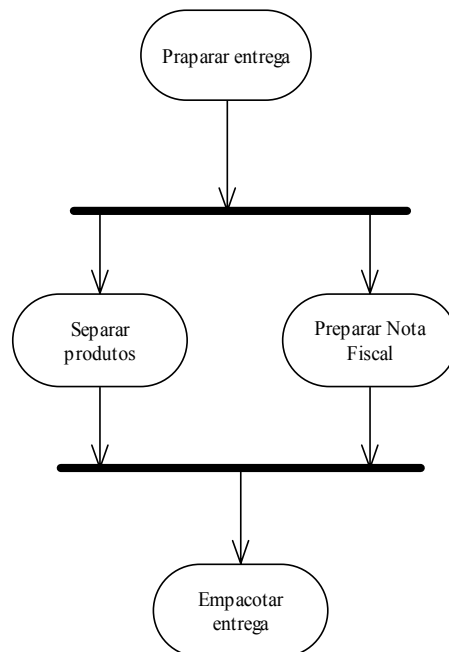
Figura 36. Um diagrama de atividades de UML simples (linear), indicando as suas figuras básicas.

Também é possível indicar a possibilidade de tomar caminhos diferentes em função de uma decisão. Isso é indicado por um losango, sendo que cada caminho possível deve receber como rótulo uma expressão que indique a decisão (*guard expression*).

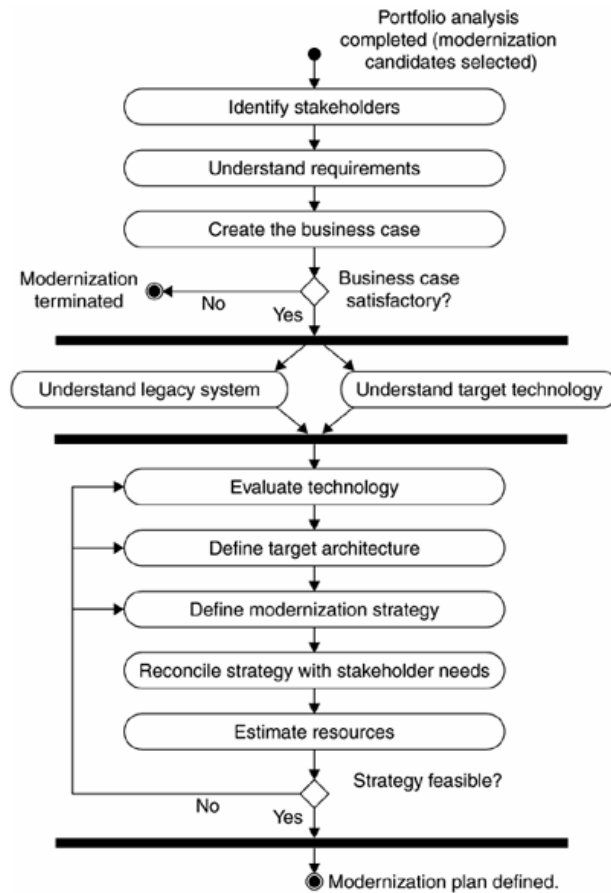


**Figura 37. Fragmento de diagrama de atividade mostrando uma decisão**

Outra característica interessante de diagramas de atividade é que permite a criação de caminhos paralelos, e sua possível sincronização (*fork* e *join*).



**Figura 38. Exemplo de caminhos em paralelo e sincronização. Forks e Joins devem ser balanceados, mas não necessariamente de forma imediata como na figura.**



**Figura 39. Exemplo de diagrama de atividades descrevendo, em alto nível, o processo de definição de um plano de modernização de software legado, segundo Seacord et al. [17] Note que há uma pequena modificação na notação de uma decisão.**

#### VI.3.4 Diagramas de Raias

Qualquer diagrama que passe a idéia de um fluxo de execução, como um fluxograma ou um diagrama de atividades, pode ser construído dentro de um espaço que modele alguma partição dessas atividades. Normalmente o que se faz é utilizar colunas (ou linhas) para modelar os agentes que realizam a atividade específica, dando a impressão de “raias de natação” ao diagrama (*swimlanes*).

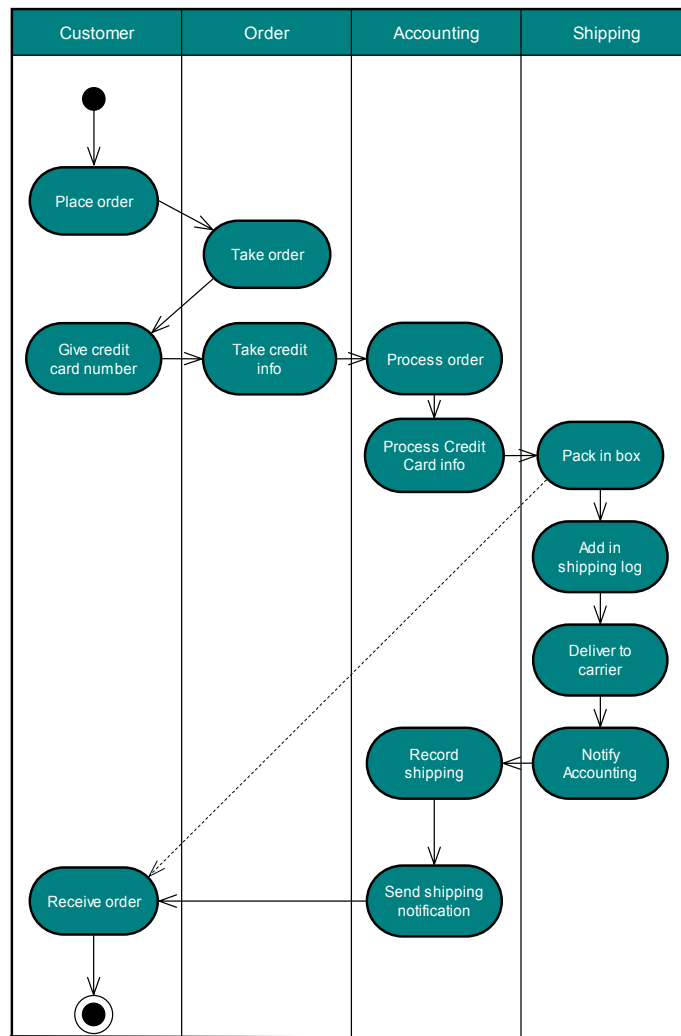


Figura 40. Exemplo de diagrama de atividades com raias

## VI.4 Regras de Negócio

“Uma regra de negócio é uma sentença que define algum aspecto do negócio. Tem o objetivo de afirmar a estrutura do negócio ou de controlar ou influenciar o comportamento do mesmo. Regras de negócio são atômicas, isto é, não podem ser quebradas”[18].

Regras de negócio são muito estudadas hoje em dia. A maioria dos importantes autores americanos se reuniu em um grupo conhecido com “Business Rule Group”<sup>22</sup>, que produziu alguns documentos [18;19] que forneceram a estrutura básica que estamos apresentando aqui. Todas as definições são ou versões dos originais em inglês.

Uma regra de negócio pode ser de três categorias.

<sup>22</sup> Originalmente “The Guide Business Rules Project”

1. Declarações Estruturais, um conceito ou a declaração de um fato que expressa algum aspecto da estrutura da organização. Podem ser termos simples ou fatos relacionando esses termos. Normalmente são descritas por meio de um diagrama de entidades e relacionamentos<sup>23</sup>.
2. Declarações de Ação, que descrevem aspectos dinâmicos do negócio, sendo uma expressão de uma restrição ou de uma condição que controla as ações de uma organização.
3. Derivações, a declaração de um conhecimento que é derivado a partir de outro.

#### VI.4.1 Declarações Estruturais

Inclui os termos de negócios e fatos relacionando termos.

##### VI.4.1.1 Definição de Termos de Negócios

“O elemento básico de uma regra de negócio é a linguagem utilizada para expressá-la. A definição das palavras utilizadas nessa linguagem descreve como as pessoas pensam e falam” [18].

Normalmente um projeto só começa realmente a progredir quando a equipe de desenvolvimento compreende o discurso dos clientes. Para isso, nas primeiras entrevistas ou reuniões, é feito um esforço para levantar um glossário de termos, e, mais tarde, muitos desses termos podem aparecer no Modelo Conceitual de Dados do sistema.

Um exemplo tirado de um sistema governamental:

- Contribuinte: é a pessoa física ou jurídica responsável pelo pagamento de um imposto.

A definição de um termo muitas vezes envolve uma relação com outros termos, como no exemplo acima.

Um termo pode ser um **tipo** (uma classe) ou um **literal** (uma instância). No caso de regras de negócio costumamos trabalhar principalmente com tipos. Um tipo especial de tipo é um **sensor**, que representa algo que detecta e reporta valores que mudam constantemente no ambiente (mundo externo). Existe um sensor especial, o **relógio**, que relata a passagem do tempo, cujo valor é sempre o instante corrente (data e hora corrente).

Os termos definidos são reunidos em um glossário e em um dicionário de dado.

##### VI.4.1.2 Declaração de Fatos

A partir dos termos, devemos construir sentenças que descrevam o negócio a partir das relações entre termos e da estrutura criada por essas relações. Normalmente esses fatos são caracterizados nas entrevistas e reuniões, a partir de declarações do cliente de como o negócio funciona.

---

<sup>23</sup> Não sendo necessariamente igual ao modelo conceitual de dados, mas podendo fornecer elementos para esse.

Fatos relacionando termos são bastante fáceis de serem encontrados. Muitas vezes encontramos primeiro um fato e depois analisamos o significado dos seus termos.

As declarações estruturais podem declarar atributos, generalizações ou participações. Uma participação, por sua vez, pode ser um papel, uma agregação ou uma associação simples.

Exemplos:

<b>Tipo de Fato</b>	<b>Exemplo</b>
<b>Atributo</b>	DRE é um atributo de aluno
<b>Generalização</b>	Aluno de graduação é um tipo de Aluno
<b>Papel</b>	Um aluno pode ser representante de classe
<b>Agregação</b>	Uma turma precisa ter alunos
<b>Associação simples</b>	Um aluno deve fazer provas

**Tabela 8. Tipos de Fatos e exemplos**

#### **VI.4.2 Declarações de Ações (ou Restrições)**

Representam as restrições ou condições que as instituições colocam em suas ações. Podem aparecer por força de lei, prática de mercado, de decisão da própria empresa ou ainda outros motivos.

Exemplos:

- Um aluno deve ter um DRE
- Um aluno não pode se registrar em dois cursos que acontecem no mesmo horário

Uma Declaração de Ação pode ser uma condição, uma restrição de integridade ou uma autorização. Uma condição diz que se alguma coisa for verdade, então outra regra deve ser aplicada (se - então). Uma restrição de integridade é uma declaração que deve sempre ser verdade. Uma autorização dá uma prerrogativa ou privilégio a um termo, normalmente permitindo que uma pessoa possa realizar uma ação.

Declarações de ações podem ser de uma variedade de outros tipos. Recomendamos a leitura de

Uma declaração de ação pode dizer que precisa acontecer (controle) ou o que pode acontecer (influência).

#### **VI.4.3 Derivações**

São regras que mostram como transformar conhecimento em uma forma para outro tipo de conhecimento, possivelmente em outra forma, incluindo leis da natureza[19]. Geralmente são regras ou procedimentos de cálculo ou manipulação de dados.

Exemplo:

- O valor a ser pago do imposto predial é 3% do valor venal do imóvel.
- A lista de devedores inclui todos os devedores a mais de dois anos.

#### VI.4.4 Descrevendo Regras de Negócio

Existem muitas formas de descrever regras de negócio, de acordo com o grau de formalismo e a necessidade de execução (ou compreensão por serem humanos) que desejamos. A tabela a seguir define quatro formas básicas de descrição.

Fragmento de conversação de negócios	Versão em linguagem natural	Versão em uma linguagem de especificação de regras	Versão em uma linguagem de implementação de regras
Pode não ser relevante	Relevante	Relevante	Executável
Pode não ser atômica	Atômica	Atômica	Pode ser procedural
Pode não ser declarativa	Declarativa	Declarativa	
Pode não ser precisa	Não totalmente precisa	Precisa	
Pode ser incompleta	Pode ser incompleta	Completa	
Pode não ser confiável	Confiável	Confiável	
Pode não ser autêntica	Autêntica	Autêntica	
Pode ser redundante	Pode ser redundante	Única	
Pode ser inconsistente	Pode ser inconsistente	Consistente	

**Tabela 9. Formas de descrever regras de negócio e suas características adaptado de [20].**

Halle[20] propõe uma classificação e um conjunto de *templates* que pode ser utilizado para descrever regras de negócio (aqui adaptado para português), apresentado na tabela a seguir.

Classificação	Descrição Detalhada	Template
<b>Termo</b>	Nome ou sentença com uma definição acordada que pode ser: <ul style="list-style-type: none"> <li>• Conceito, classe, entidade,</li> <li>• Propriedade, detalhe, atributo,</li> <li>• Valor,</li> <li>• Conjunto de valores</li> </ul>	<termo> é definido como <texto>
<b>Fato</b>	Sentença que relaciona termos em observações relevantes ao negócio <ul style="list-style-type: none"> <li>• Relacionamentos entre entidades</li> <li>• Relacionamentos entre entidades e atributos</li> <li>• Relacionamentos de herança</li> </ul>	<termo1> é um <termo2> <termo1> <verbo> <termo2> <termo1> é composto de <termo2> <termo1> é um papel de <termo2> <termo1> tem a propriedade <termo2>
<b>Computação</b>	Sentença fornecendo um algoritmo para calcular o valor de um termo	<termo> é calculado como <formula>
<b>Restrição obrigatória</b>	Sentença que indica restrições que devem ser verdade em informações fornecidas ao sistema (input)	<termo1> deve ter <no mínimo, no máximo, exatamente n> <termo2> <termo1> deve ser <comparação> <termo2> ou <valor> ou <lista de valores> <termo> deve ser um de <lista de valores> <termo> não pode star em <lista de valores> se <regra> então <restrição>
<b>Guideline</b>	Sentença que indica restrições que deveriam ser verdade em informações fornecidas ao sistema (input)	<termo1> deveria ter <no mínimo, no máximo, exatamente n> <termo2> <termo1> deveria ser <comparação> <termo2> ou <valor> ou <lista de valores> <termo> deveria ser um de <lista de valores> <termo> não poderia star em <lista de valores> se <regra> então <restrição>
<b>Conhecimento inferido</b>	Sentenças que expressam circunstâncias que levam a novas informações	se <termo1> <operador> <termo2, valor, ou lista de valores> ... então <termo3> <operador> <termo4> Onde operador pode ser: =, <, =<, >=, <,>, contém, não contém, tem no máximo, tem no mínimo, tem exatamente, etc.
<b>Iniciador de ação</b>	Sentença expressando condições que levam ao início de uma ação	se <termo1> <operador> <termo2, valor, ou lista de valores> ... então <ação>

**Tabela 10. Classificação para regras, definição e templates adaptada de (Halle, 2002).**

Obviamente, como fizemos aqui, a forma textual é bastante útil. Uma maneira bastante comum é utilizar Diagramas de Entidades e Relacionamentos[19].

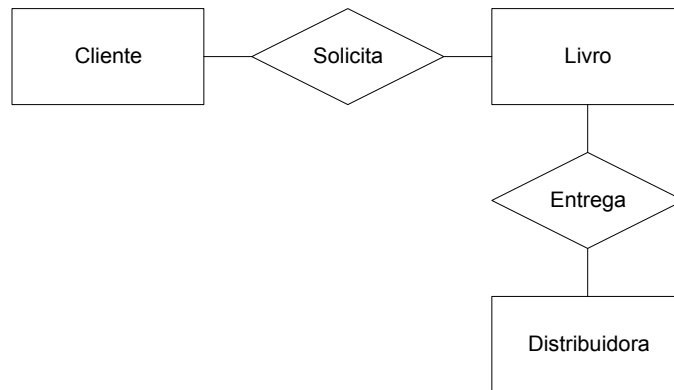
**Deve ficar claro, porém, que a utilização de DER para definir regras de negócio não é igual à utilização de DER para definir o modelo conceitual de dados de um sistema.**



DERs, porém, não representam todas as características das regras de negócio. Ross [21], propôs uma notação mais complexa, incluindo (muitos) novos símbolos em um DER. Essa notação, porém, foge do escopo desse texto.

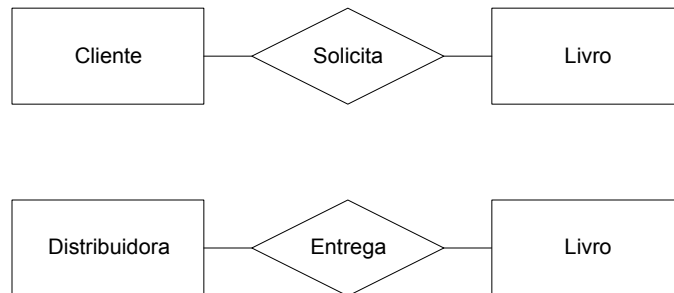
Vejamos a descrição de duas regras para uma livraria (Figura 41):

- Um cliente deve pedir livros.
- Livros são entregues por distribuidoras.



**Figura 41. Exemplo de duas regras de negócio descritas utilizando uma notação de DER simplificada.**

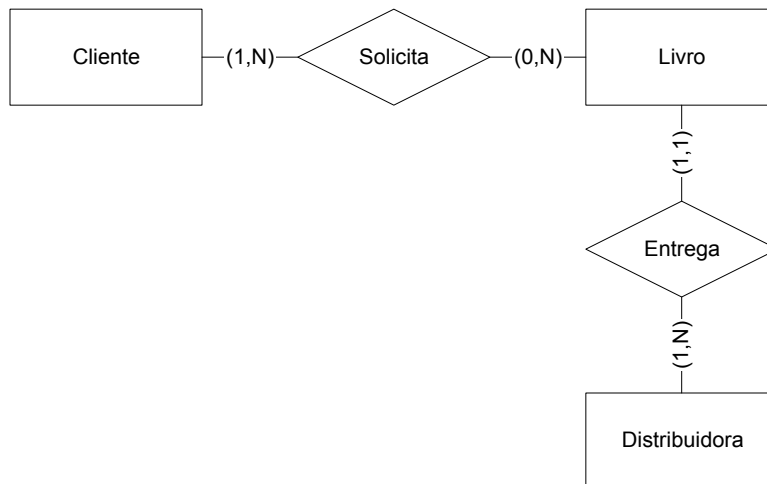
Alguns autores fazem uma descrição regra a regra, como na Figura 42.



**Figura 42. Descrição das regras uma a uma.**

As regras de negócio serão utilizadas nos próximos passos para definir modelos mais formais do sistema. Mas podemos, por exemplo, definir a cardinalidade dos termos em cada relacionamento descrito, como na Figura 43 e no texto a seguir:

- Um cliente pede um ou mais livros,
- Livros podem ser pedidos por nenhum, um ou mais clientes,
- Uma distribuidora entrega um ou mais livros,
- Um livro é entregue por apenas uma distribuidora.



**Figura 43. Regras com cardinalidades.**

É interessante notar que regras de negócio devem ser feitas de forma declarativa, **não** indicando como vão ser implementadas, onde vão funcionar, quem será responsável ou quando devem ser testadas. Desse jeito as regras serão flexíveis o suficiente para ser utilizadas na modelagem do negócio. Como devem ser declarativas, elas também não devem ser escritas como um procedimento.

## VI.5 Outras Ferramentas de Modelagem

O uso de tabelas permite relacionar informações levantadas separadamente. Elas são bastante importantes nas conferências dos modelos.

### VI.5.1 Responsáveis por decisão (Processo x Organização)

Essa tabela associa as pessoas da organização, que foram representadas no organograma, com as funções da empresa, que podem ter sido representadas anteriormente de várias formas, como o diagrama funcional.

O objetivo da tabela é determinar o envolvimento dessas pessoas com as decisões relativas às funções da empresa. Esse envolvimento é determinado em três níveis: principal responsável pela decisão, grande envolvimento com a decisão e, finalmente, algum envolvimento com a decisão.

	Principal responsável pela decisão
	Grande envolvimento com a decisão
	Algum envolvimento com a decisão

**Tabela 11. Tipos de envolvimento com a decisão**

	Processos	
	Vendas	Produção

		Gerência do Território	Venda	Administração	Pedidos de Serviço	Agendamento	Planejamento	Fornecimento	Operação
Organização	VP de Vendas								
	Gerente de pedidos								
	Gerente de vendas								
	VP Engenharia								
	VP Produção								
	Diretor da Fábrica								

**Tabela 12 Exemplo de uma tabela Processo x Organização**

### VI.5.2 Dados x Processos (CRUD)

Esta tabela é uma das mais interessantes e relaciona as entidades (inicialmente do modelo de conceitual de dados) com os processos que as utilizam, indicando pelas letras CRUD se o processo **C**ria, **R**ê (**Read**), **U**nter (**Update**) ou **D**eleter (**Delete**) a entidade.

Uma de suas características principais é que pode ser construída com vários pares linha x coluna, dependendo do tipo de abstração que está sendo usado. No nosso caso usaremos principalmente no mapeamento de ações (CRUD) entre eventos essenciais e entidades (os dois temas serão tratados mais tarde), mas podem ser usados em modelos de negócio (processo x dados utilizados) ou orientados a objetos (casos de uso x objetos). Na prática, é uma tabela de trabalho muito interessante, que permite a visualização rápida da relação entre funcionalidade e dados.

Como usaremos esta tabela para relacionar eventos e entidades, deixaremos esse tema para ser tratado mais tarde, na unificação dos modelos funcional e de dados.

	Dados					
	Dado 1	Dado 2	Dado 3	Dado 4	Dado 5	Dado 6
Processos						
Processo A	CRUD	R	R			
Processo B		CRUD	R		R	
Processo C		RUD	C			
Processo D				C	R	
Processo E				R	C	
Processo F		R			RU	C

**Figura 44. Exemplo simplificado de uma Matriz CRUD**

### VI.5.3 Corrente/Planejado

Esta tabela indica que processos são correntes e que processos estão apenas planejados. Pode ser feita tanto a nível de negócios quanto a nível de sistemas. No primeiro caso estamos interessados nos processos correntes ou implementados no dia a dia da empresa. No segundo caso estamos interessados em processos automatizados.

Corrente x Planejado	
Função	Situação
Cadastro de Cliente	Corrente
Cadastro de Fornecedor	Corrente
Cadastro de Pedidos	Planejado
Criação de Pedidos	Planejado

**Tabela 13. Exemplo de tabela Corrente x Planejado**

## VI.6 Resumo da Modelagem de Negócio

Não há uma forma correta única de se fazer a modelagem de negócio. Recomendamos que sempre seja levantado o organograma da empresa. A seguir, os modelos podem ser escolhidos e levantados de forma adequada a um processo ou projeto específico.

No capítulo vimos, em ordem decrescente de abstração, métodos que podem ser utilizados tanto de forma isolada como unificada. Deve ser levado em consideração que o método IDEF0 faz parte de uma família de métodos que devem ser utilizados juntos (no caso de modelagem de processo existe o método IDEF3, que não usaremos nesse texto por considerarmos o EPC um método superior).

## VI.7 Exercícios

### VI.7.1 Projeto 1: Livraria ABC

Faça todos os modelos de negócios descritos nesse capítulo para a Livraria ABC, da forma mais completa possível.

## Capítulo VII. Modelo Conceitual de Dados

*Models are to be used, not believed.*

*H. Theil 'Principles of Econometrics'*

*Data is not information, Information is not knowledge, Knowledge is not understanding, Understanding is not wisdom.*

*Cliff Stoll & Gary Schubert*

Modelo de Dados
Entidades
Relacionamentos
Atributos
Chave Candidata, Chave Primária
Formas Normais

O Modelo Conceitual de Dados, como o nome diz, é um modelo abstrato que descreve as informações contidas no sistema. O objetivo final do modelo de dados é a criação do banco de dados do sistema, seja ele por meio de simples arquivos ou sofisticados sistemas de gerenciamento de banco de dados<sup>24</sup>. Em geral, as informações contidas no modelo serão aquelas que o sistema precisa ter para executar uma função e que não são fornecidas pelo mundo exterior no momento que a função é solicitada, mas sim anteriormente.

A criação do modelo de dados é um processo de especificações sucessivas. Inicialmente descrevemos um modelo do ambiente observado, normalmente descrevendo o mundo como ele é visto pelo usuário. Esse modelo é conhecido como **Modelo Conceitual de Dados**. No final devemos possuir uma descrição na visão do desenvolvedor, descrevendo o mundo de uma forma específica, otimizada para os dispositivos sendo utilizados na implementação do sistema (linguagens de programação, SGDB, sistema operacional e hardware), o **Modelo Físico de Dados**. Entre o modelo conceitual e o modelo físico devemos passar por um passo intermediário, onde assumimos compromissos com uma tecnologia específica, mas não com os produtos sendo utilizados. Este passo é conhecido como **Modelo Lógico**. Muitas metodologias de modelagem de dados, como a IDEF1X, utilizam apenas dois passos, o modelo lógico e o físico, assumindo já no primeiro modelo algumas regras típicas do modelo relacional.

A principal forma de modelagem de dados, e a forma adotada por esse texto, é o Modelo de Entidades e Relacionamentos, ou MER, por meio do Diagrama de Entidades e Relacionamentos, ou DER. No mo Modelo de Entidades e Relacionamentos contamos com três abstrações para modelar o mundo: entidades, atributos e relacionamentos. De

---

<sup>24</sup> Na análise essencial discutiremos a necessidade de um sistema de ter uma memória que permite ao sistema “se lembrar” de fatos passados ao atender um evento. A modelagem de dados é a técnica que nos permitirá definir como é essa memória, isto é, que informações deve guardar.

maneira simples, podemos dizer que entidades representam as “coisas e conceitos” do mundo, atributos representam as características dessas “coisas e conceitos” e relacionamentos representam as relações existentes entre essas “coisas e conceitos”.

## VII.1 Modelos e Abstrações

*The human mind has first to construct forms, independently, before we can find them in things.*

*Einstein, Albert (1879-1955)*

Todo **modelo** é uma abstração de algo que existe ou se imagina que possa existir no mundo real. **Abstração** é o processo mental de separar um ou mais elementos de uma totalidade complexa de forma a facilitar a sua compreensão por meio de um modelo.

No nosso dia a dia utilizamos a abstração para poder trabalhar com toda a informação que o mundo nos fornece. Um mapa, por exemplo, é um modelo de uma cidade. Dependendo da informação que queremos, colocamos alguns símbolos e tiramos outros do mapa. Um mapa também não pode ser “perfeito”, tem que “abstrair” as informações que não são necessárias naquele instante, ou teria que ter o mesmo tamanho da cidade.

No desenvolvimento de sistemas, utilizamos alguns processos de abstração típicos:

- Classificação,
- Composição,
- Generalização e
- Identificação.

### VII.1.1 Classificação (é um membro de)

No processo de **classificação** eliminamos a parte da individualidade do objeto ou sistema analisado e o consideramos como um exemplar de uma “classe padrão”. Quando fazemos isso, aceitamos que esse objeto, agora uma **instância** da classe, divide com todas as outras instâncias da classe um conjunto de características. Na classificação o que estamos fazendo é imaginar uma idéia única que descreve, de forma abstrata, todos os objetos de uma classe. Ao eliminar a necessidade de tratar cada objeto de forma única, simplificamos o problema em questão.

Exemplos típicos de classificação:

Instâncias	Classe
Flamengo, Fluminense, São Paulo	Times de Futebol
Brasil, Estados Unidos	País
Pelé, Zidane, Romário	Jogador de Futebol

**Tabela 14. Exemplo de classificação**

O processo reverso da classificação é a **instanciação**. O conjunto de todas as instâncias de uma classe é a **extensão** dessa classe.

### VII.1.2 Composição ou Agregação (é feito de)

Na **composição** entendemos um objeto complexo formado de um conjunto de outros objetos como um só objeto. É como vemos uma bicicleta ou um carro. Ao eliminar a necessidade de descrever as partes, simplificamos a compreensão do objeto analisado.

Exemplos típicos de composição:

Partes	Objeto
Pneus, motor, etc.	carro
Capa, centenas de folhas, etc.	livro
Cabelo, pele, ossos, etc.	homem

**Tabela 15. Partes-Objeto na relação de composição**

O processo reverso da composição é a **decomposição**.

### VII.1.3 Generalização (é um, é como)

Com a **generalização** nós somos capazes de entender como uma classe pode ser descrita por outra classe, mais geral. É importante ver a diferença entre a classificação e a generalização: a primeira trata da relação entre objeto e classes, enquanto a segunda trata da relação entre classes.

Com a generalização podemos compreender uma relação muito comum entre classes, que é a que permite que qualquer objeto de uma classe possa ser visto, de uma forma mais geral, como um objeto de outra classe. Utilizando judiciosamente a generalização podemos simplificar a forma de tratar objetos de classes similares.

Exemplos típicos de generalização:

Classes	Classe mais geral
Funcionário, Aluno, Professor	Pessoa
Automóvel, Avião, Navio	Meio de Transporte
Computador, Rádio, Televisão	Aparelhos Eletrônicos

**Tabela 16. Exemplo de generalização**

O processo reverso da generalização é a **especialização**.

### VII.1.4 Identificação (é identificado por)

Com a **identificação** nós somos capazes de entender como caracterizar unicamente um objeto. Um nome identifica uma pessoa, por exemplo. Ao identificar unicamente um objeto podemos separá-lo de outro objeto semelhante e atribuir a entidades específicas atributos e características que só pertencem a ela, e não pertencem a outros elementos daquela classe.

Há uma diferença entre instanciar e identificar. Uma instância deve possuir uma identificação e uma identificação se aplica a uma instância. A identificação permite a que duas instâncias sejam reconhecidas como distintas ou como representações de um mesmo objeto (normalmente devendo ser reunidas em uma).

### VII.1.5 Trabalhando com as abstrações

Imagine que precisamos descrever comprar um carro. É óbvio que todo carro possui quatro pneus, um motor, etc. Isso é uma classe bastante geral. Porém, desejamos ainda falar sobre um modelo específico: uma Ferrari Testarossa, por exemplo. Logo, acabamos de especializar nosso modelo, mas ainda não chegamos ao nível de objeto. Quando vemos o carro específico, aí temos o objeto. Ele é identificável como instância daquela classe porque apesar de dividir várias características em comum com outros objetos da classe, também tem algumas características únicas, como o número de série do chassi. Finalmente, desejamos trocar a cor do assento do carro. Nesse instante, já estamos vendo uma parte do carro, decompondo-o em suas partes.

## VII.2 A Memória do Sistema

Na análise essencial é importante compreender o conceito de memória do sistema. Para cada necessidade do cliente, como relatórios e tomadas de decisão, o sistema precisa de certa quantidade de dados. Esses dados são sempre fornecidos pelo mundo exterior (pois o sistema não pode “inventar” dados), porém nem sempre no mesmo momento em que a função necessária é realizada. Normalmente, por sinal, um sistema de informações é composto por funções que coletam dados para que sejam, mais tarde, utilizados por funções que fornecem relatórios.

Dessa forma, esses dados necessários para realizar uma função precisam estar em algum lugar. Na análise essencial nós abstraímos a localização e forma física dos dados, supondo que o sistema possui uma **memória**. Com a Modelagem Conceitual de Dados damos a forma abstrata a essa memória, de maneira a entender o que deve estar guardado na memória sem decidir, prematuramente, sua localização e estrutura.

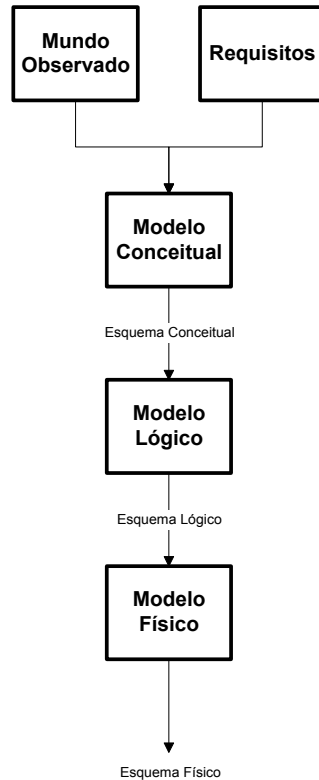
### VII.2.1 Modelo Conceitual (MC)

O objetivo da modelagem conceitual é fornecer aos desenvolvedores uma descrição abstrata de alto nível, independente de tecnologia, da informação contida no sistema. Essa descrição é conhecida como o **esquema conceitual** da base de dados.

A Modelagem Conceitual de Dados pode ser feita de muitas formas, algumas vezes com sutis diferenças. Alguns autores defendem a “modelagem do domínio”, onde tentamos descrever o domínio de aplicação sendo utilizados, outros tratam diretamente do sistema sendo desenvolvido. Neste texto trabalharemos com uma abordagem mais próxima do sistema sendo desenvolvido, pois estamos buscando uma ferramenta que se encaixe com a análise essencial.

O modelo conceitual é construído a partir da análise de requisitos, em geral simultaneamente ao desenvolvimento da análise essencial. Na prática, o primeiro modelo essencial usará como memórias os objetos descritos no DER.





**Figura 45. Etapas da Modelagem de Dados**

Um dos subsídios mais importantes para a criação do DER é o conjunto de regras de negócio levantadas. Muitas das regras de negócio são representadas diretamente no modelo conceitual. Veremos mais tarde que termos e fatos são candidatos naturais para serem objetos nos nossos modelos conceituais. Não devemos confundir, porém, regras de negócio com modelos de dados. Um relacionamento em uma regra de negócio pode representar uma função do sistema, enquanto um relacionamento no MER representa algo que deve pertencer à memória do sistema.

### **VII.2.2 Modelo Lógico**

O modelo lógico descreve a informação contida no sistema de acordo com uma tecnologia adotada, sem utilizar, porém, detalhes de implementação. Ele descreve a estrutura do banco de dados que será processado por um SGDB.

Atualmente, o **modelo mais utilizado é o modelo relacional**, porém existe uma tendência à utilização do modelo relacional-objeto ou de outros modelos relacionais estendidos. Além disso, alguns modelos distintos podem ser encontrados em aplicações especiais, como data-warehousing e sistemas de informação geográfica. O modelo de objetos, considerado por muitos o mais moderno, não tem no momento grande aceitação no mercado.

### **VII.2.3 Modelo Físico**

No modelo físico devemos levar em conta não só a tecnologia sendo utilizada, mas também os produtos específicos e a interação do sistema com o ambiente de desenvolvimento e operação.

É nessa etapa que nos preocupamos com as principais questões de desempenho, como escolha de índices, particionamento, etc.

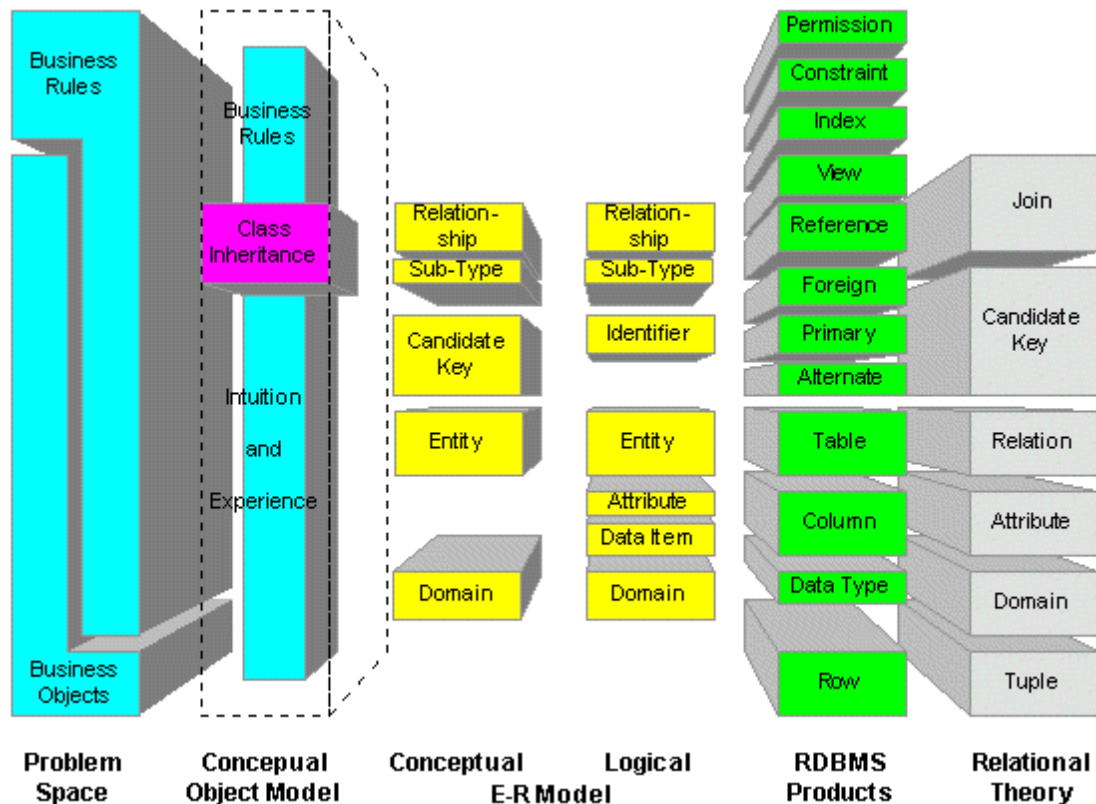


Figura 46. Um diagrama abstrato representando os conceitos utilizados em diferentes modelos criados para tratar a informação<sup>25</sup>.

### VII.3 Modelo de Entidades e Relacionamentos

O Modelo de Entidades Relacionamentos, segundo Paulo Cougo [22], descreve o mundo como: "...cheio de coisas que possuem características próprias e que se relacionam entre si".

Essas coisas podem ser pessoas, objetos, conceitos, eventos, etc. Elas são as entidades. *A priori*, só exigimos de uma entidade que ela possa ser identificada distintamente, isso é, tenha identidade própria. Cada coisa distintamente identificada é uma instância. Por exemplo, o funcionário José é uma instância da entidade funcionário, a aluna Maria é uma instância da entidade aluna.

As entidades, ou melhor, suas instâncias, são classificadas em tipos (ou classes). No nosso caso, funcionário e aluno são os tipos de entidade. Estamos usando nesse momento a abstração de classificação: resumir uma quantidade de características comuns por meio da criação de uma classe. Assim sabemos que o funcionário José e o

<sup>25</sup> [http://www.aisintl.com/case/user\\_groups/PowerDesigner/library/sd-business\\_rules.html](http://www.aisintl.com/case/user_groups/PowerDesigner/library/sd-business_rules.html)

funcionário Joaquim, por serem instâncias de um mesmo tipo, possuem características comuns (como trabalhar na empresa, ter um salário, etc.).

No diagrama de entidade e relacionamentos cada tipo de entidade é representado por um retângulo identificado pelo nome do tipo. Normalmente confundimos o termo entidade com o tipo da entidade, deixando o termo instância (e algumas vezes registro) para falar de uma entidade identificada.

Apenas algumas entidades do mundo real (ou imaginário) são de interesse para o sistema. Durante a modelagem conceitual nos preocupamos com as “coisas” que o sistema deve lembrar e colocamos essas “coisas” no modelo de entidade e relacionamentos.

Cada entidade tem dois tipos de características importantes: seus atributos e seus relacionamentos. Os atributos são características que toda a instância de um tipo possui, mas que podem variar entre as instâncias. Uma instância do tipo “aluno” tem os atributos “nome” e “ano de matrícula”, por exemplo. Atributos caracterizam a informação que deve ser guardada sobre uma entidade. Só devemos colocar como atributos aquelas informações que o sistema precisa lembrar em algum momento. Assim, uma instância de “aluno” não precisa ter o atributo “nome do animal de estimação” em um sistema acadêmico, pois apesar de ser algo importante para o “aluno” propriamente dito, não tem importância alguma para o sistema.

Cada característica deve possuir um **domínio**. O domínio indica o conjunto de valores válidos para a característica. No caso de “nome”, geralmente aceitamos qualquer seqüência de caracteres, enquanto no caso de “altura” podemos aceitar apenas valores reais positivos menores que 2,5.

Atributos eram originalmente descritos por círculos no modelo E-R. As notações mais modernas anotam os atributos dentro dos retângulos da entidade a que pertencem.

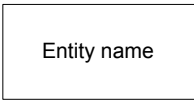
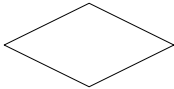

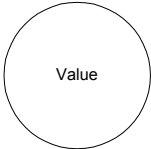
Finalmente, como indica o nome do modelo, entidades podem se relacionar entre si. Essa característica é a principal força do modelo de entidades e relacionamentos, pois permite que, de certa forma, “naveguemos” no modelo.

Podemos indicar relacionamentos apenas pelas entidades envolvidas, como “cliente-pedido”, ou usar um termo que descreva o relacionamento “cliente solicita pedido”.

Modelos de Entidades e Relacionamentos para serem completos exigem também um conjunto de restrições. Algumas dessas restrições, como a cardinalidade dos relacionamentos que veremos a seguir, podem ser descritas em algumas (ou todas) notações. Porém, a maioria das descrições é muito complexa para ser descrita em um diagrama. Nesse caso são necessárias anotações ao diagrama descrevendo as descrições. Isso pode ser feito em linguagem natural ou em alguma notação formal específica, dependendo de escolhas da equipe de projeto ou do método utilizado.

## VII.4 O Diagrama de Entidades e Relacionamentos

Diagramas de Entidades e Relacionamentos descrevem o mundo em geral ou um sistema em particular de acordo com os objetos que o compõe e os relacionamentos entre esses objetos.

Figura	Significado
	Entidade
	Relacionamento
	Ligação entre entidades, por meio de um relacionamento, com a descrição da cardinalidade.
	Atributo

**Tabela 17. Figuras básicas de um diagrama de Entidades e Relacionamentos segundo Peter Chen**

Existem muitas notações para Diagrama de Entidades e Relacionamentos. A notação original foi proposta por Chen e é composta de entidades (retângulos), relacionamentos (losangos), atributos (círculos) e linhas de conexão (linhas) que indicam a cardinalidade de uma entidade em um relacionamento. Chen ainda propõe símbolos para entidades fracas<sup>26</sup> e entidades associativas.

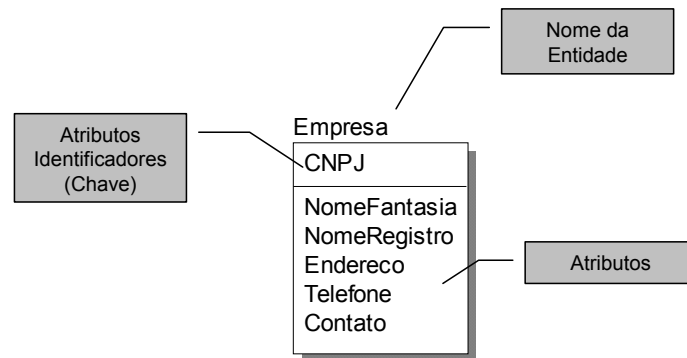
As notações modernas abandonaram o uso de símbolos especiais para atributos, incluindo a lista de atributo, de alguma forma, no símbolo da entidade. Consideramos as notações como as mais interessantes na atualidade:

- IDEF1X, utilizada pela ferramenta ERWIN, bastante difundida no mercado
- Engenharia de Informação, bastante difundida e também presente como notação alternativa no ERWIN.
- Notação de Setzer, difundida no Brasil por seu autor.
- Notação de Ceri, Bertini e Navathe [23], pouco difundida, mas com aspectos teóricos interessantes.
- Uso da UML para representar modelos de dados não-orientados a objetos.

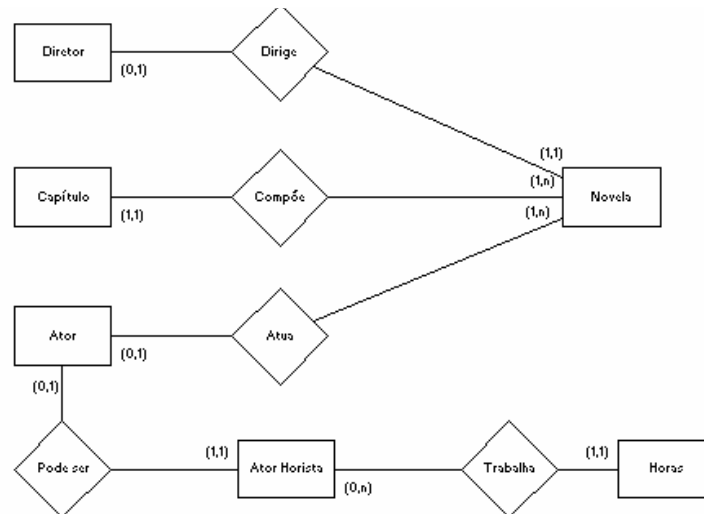
---

<sup>26</sup> Deploramos o termo entidade fraca, que leva os alunos a acreditar que não devem existir entidades fracas em um DER, associando o termo “fraco” ao conceito de errado,

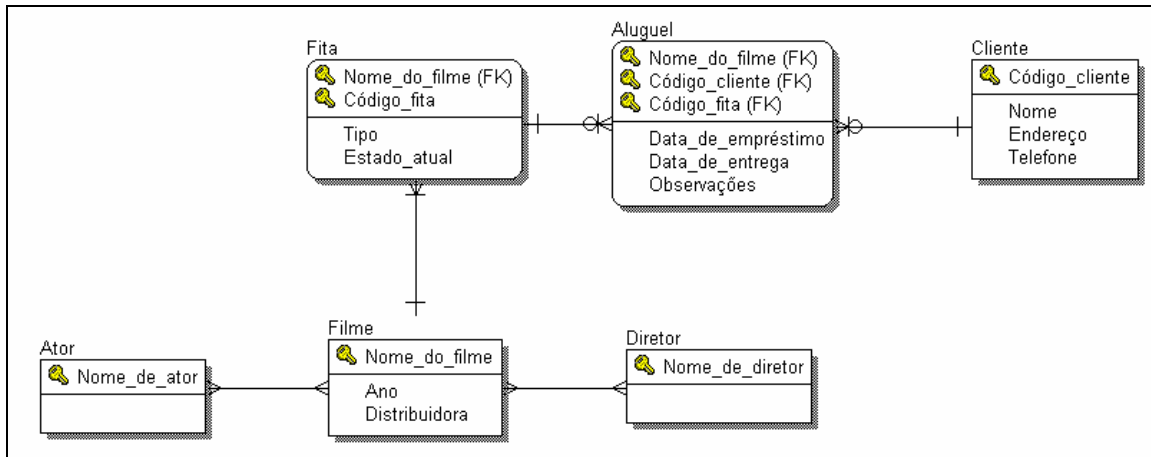
Toda a notação moderna tem como característica importante definir a cardinalidade mínima e máxima em uma relação, não utilizar um símbolo especial para relacionamentos, mas sim a linha, e descrever atributos dentro do símbolo de entidades.



**Figura 47.** Uma entidade pode ser representada também dessa forma, bem mais moderna e compacta que a proposta original. Esta é a notação utilizada no software Erwin (tanto no modelo IDEF1X e quanto na Engenharia da Informação). Os atributos identificadores ficam acima de uma linha divisória.



**Figura 48.** Um exemplo simples de DER, sem atributos.

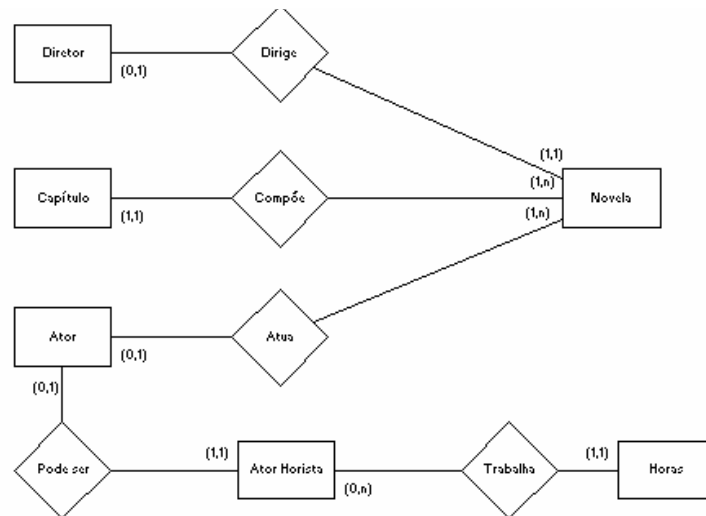


**Figura 49. Um exemplo de DER usando a Notação da Engenharia da Informação, feita com o software ERWIN. Os ícones ajudam a identificar as chaves (já identificadas pela sua posição sobre a linha divisória).**

#### VII.4.1 Exemplo de Modelo E-R

O modelo a seguir, utilizando a notação de Bertini et al. [23], pode ser lido da seguinte forma:

- Entidades do modelo: Diretor, Novela, Capítulo, Ator, Ator Horista e Hora
- Um diretor dirige no máximo uma novela, podendo não dirigir nenhuma, e uma novela é dirigida por um e apenas um diretor.
- Um capítulo compõe uma e apenas uma novela e uma novela tem no mínimo um capítulo, podendo ter um número ilimitado deles.
- Um ator atua em uma novela, podendo não atuar em nenhuma e uma novela tem ao menos um ator, podendo ter vários.
- Um ator pode ser um ator horista e um ator horista é obrigatoriamente um ator.
- Um ator horista trabalha de zero a várias quantidade de horas, mas uma quantidade de horas é trabalhada por apenas um ator.



**Figura 50. Exemplo de modelo conceitual**

No modelo anterior não apresentamos nenhum atributo. A notação original para atributos, o uso de círculos ligados aos retângulos que representam as entidades, complica muito o diagrama. As notações mais modernas anotam os atributos dentro dos retângulos.

## VII.5 Desenvolvendo o Modelo Conceitual

Desenvolver um modelo conceitual correto para um sistema, completo e consistente, não é uma tarefa fácil. Já desenvolver um modelo conceitual razoavelmente correto, que dê uma idéia do sistema e do negócio e que, de forma evolutiva, resulte em um modelo conceitual correto, é uma tarefa razoavelmente fácil para um analista experiente. Para o analista de sistemas iniciante, porém, parece uma tarefa extremamente difícil.

Isso acontece porque o modelo conceitual de dados exige duas coisas: um alto grau de abstração e a internalização, pelo analista, de conceitos bastante vagos, como “entidades” e “relacionamentos”. Assim, o analista iniciante precisa seguir algumas estratégias para entender melhor como desenvolver um modelo conceitual.

A primeira estratégia é a estratégia dos exercícios e exemplos. Nada é tão útil ao aprendizado como colocar a mão na massa. Os exemplos, por seu lado, servem não só como orientação geral, mas também como exemplos de pontos específicos da modelagem e de como especialistas resolvem problemas de modelagem, sejam eles simples ou complicados.

A segunda estratégia é desenvolver uma lista de dicas de trabalho. Essas dicas funcionam para o analista como as pistas funcionam para um detetive, mostrando que caminho seguir até encontrar a solução do problema.

## VII.6 Entidades

**Uma entidade é uma pessoa, objeto, local, animal, acontecimento, organização ou outra idéia abstrata sobre a qual o sistema deve se lembrar alguma coisa.**  
**Cada instância de uma determinada entidade tem características similares (mas não iguais), o mesmo comportamento e uma identidade própria.**

O primeiro passo na determinação das entidades é o levantamento dos candidatos à entidade. Durante as entrevistas e reuniões de análise de sistema, vários objetos e conceitos serão descritos como parte do sistema. Algumas vezes esses objetos são bastante concretos, como um “produto” dentro de um “estoque”, outras vezes são descritos como documentos que guardam alguma informação, como uma “nota fiscal”, outras vezes são abstratos, como um “curso”.

No discurso fluente durante uma entrevista, entidades são geralmente substantivos ocupando o papel de sujeito ou objeto, enquanto relacionamentos geralmente são encontrados na forma de verbo. Muitas vezes uma regra de negócio, como “alunos cursam turmas” ou “clientes fazem pedidos” nos indica entidades e relacionamentos<sup>27</sup>.

Outro sinal importante da necessidade de uma entidade é o fato de algo que precisa ser lembrado representar um conceito ou idéia completa. Em um sistema acadêmico precisamos nos lembrar do nome do aluno, da data de matrícula do aluno, do curso em que está o aluno, etc. O “aluno” é a nossa idéia completa que aparece várias vezes, algumas vezes caracterizado por seu nome, outras vezes por seu curso. É um bom candidato a entidade.

Alguns autores propõem uma determinação bottom-up das entidades, sugerindo que elas sejam construídas pela partição de todos os dados atômicos que o sistema deve lembrar (nome de aluno, data de matrícula do aluno, etc.). Assim, construiríamos uma lista de atributos para depois agrupá-los em entidades.

Preferimos, porém, uma abordagem de busca direta das entidades. Um sistema com poucas dezenas de entidades pode ter centenas de atributos, o que torna tudo bem mais confuso.

Segundo Shlaer e Mellor [24], uma entidade pode estar em cinco grandes categorias:

- Objetos tangíveis
- Papéis exercidos
- Eventos
- Interações

---

<sup>27</sup> Novamente, lembro que apesar de usarmos a notação E-R para descrever regras de negócio, estamos falando de duas atividades diferentes e que tem resultados diferentes (apesar de poderem, por coincidência terem o mesmo resultado).



- Especificações

Podemos facilmente ver porque objetos tangíveis são bons candidatos a entidades: normalmente, sistemas de informação falam em algum momento de objetos tangíveis, como produtos e equipamentos. Algumas vezes, porém, um objeto tangível, como uma pessoa, assume uma função ou papel específico, como aluno ou professor.

Eventos, ou interações, acontecem em algum momento do tempo e representam classes importantes de entidades. Um exemplo de evento é uma “reunião” em uma agenda. Normalmente eventos exigem atributos como data e duração.

Exemplos típicos de interações são: “contratação de serviço” ou “venda de produto”. Interações são semelhantes a relacionamentos ou a objetos tangíveis ou eventos, sendo muitas vezes representadas dessa forma.

Finalmente, especificação são tipos especiais de entidades que classificam outra entidade. Um bom exemplo é “fábrica”, que é uma especificação para “automóvel”. Geralmente, especificações também podem ser implementadas como um atributo na entidade especificada, sendo essa uma decisão de análise.

Já Coad, ao buscar uma forma mais objetiva de modelar objetos, sugere que busquemos inicialmente 4 tipos de objetos (que podemos entender como entidades):

- Momentos ou Intervalos: um momento ou um intervalo representa qualquer coisa que precisa ser acompanhada, por motivos de negócio ou legais, e que acontecem em um instante de tempo ou por um período de tempo. Muitas vezes pode ser mais fácil começar nossa análise por esse tipo de entidade, pois estamos tratando de atividades de negócio que devem exigir a participação das outras entidades. Exemplos são: aulas, consultas, contratação, etc.
- Papéis: representam papéis assumidos pelas pessoas que estão envolvidas com o sistema sendo analisado. Cuidado, pois não são apenas os usuários, nem representam os cargos que as pessoas ocupam nas empresas necessariamente. Exemplos são: aluno, professor.
- Pessoas, Locais ou Coisas: representam os objetos tangíveis e localidades. Exemplos são: sala, automóvel.
- Descrições: são basicamente as especificações propostas por Shlaer e Mellor. Modelos de um produto é um bom exemplo.

James e Suzanne Robertson [25] sugerem algumas regras para que verifiquemos se um conceito deve ser realmente escolhido como uma entidade:

- Toda entidade deve ter um papel único e definido no negócio, se você não pode explicá-la, provavelmente não precisa se lembrar dela.
- Entidades devem ter ao menos um atributo que as descrevam, e é preferível que tenham vários.
- Entidades devem ter mais de uma instância. Se a instância é única, então não deve ser uma entidade, mas uma informação constante, que é parte do negócio da empresa (uma regra de negócio?).

- Entidades devem possuir instâncias unicamente identificáveis.
- Entidades não possuem valores, apenas atributos possuem valores.
- Pessoas e organizações que interagem com o sistema são candidatos a entidade quando precisamos nos lembrar alguma coisa específica sobre elas, para gerar relatórios ou processar dados entrados. Isso não se aplica a “logons” ou “passwords” utilizados para a segurança do sistema, pois segurança é um problema tratado no projeto físico. Devemos aplicar essa regra em relação à necessidade de identificação e endereçamento, por exemplo.
- Relatórios raramente são entidades. Normalmente eles são apenas os resultados de um processo que acessa várias entidades.
- Linhas de relatório geralmente são entidades. Nomes de colunas indicam entidades ou seus atributos. Porém, nenhum valor calculado ou derivado é atributo ou entidade.
- Substantivos em regras de negócio são normalmente entidades
- Produtos, quando não são únicos, são normalmente entidades.
- Papéis, como funcionário, atendente, apostador, etc., são bons candidatos para entidades.
- Um grupo de dados que se repete em uma entrada ou saída de dados é normalmente uma entidade (ou mais).

#### **VII.6.1 Onde encontrá-las**

Além de encontrá-las em entrevistas e em regras de negócio, podemos utilizar alguns documentos para encontrar entidades:

- Relatórios
- Formulários de entrada de dados
- Arquivos, tanto de papel quanto no computador.
- Fichas, como fichas de cadastro, de empréstimo, etc.
- Pedidos, requisições e documentos do gênero.
- Documentos contábeis e fiscais, como nota fiscal.
- Planilhas de dados, em papel ou eletrônicas.
- Listagens, registros, agendas, protocolos e outros documentos de trabalho.
- Sistemas já existentes
- Bancos de dados já existentes

Outra forma de encontrar entidades é buscar sistemas semelhantes já resolvidos e padrões de projeto<sup>28</sup> ou padrões internacionais sobre o assunto sendo tratado.

### VII.6.2 Descrevendo Entidades

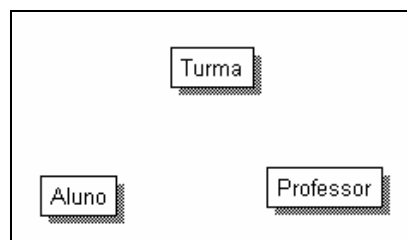
É extremamente importante a descrição precisa de cada entidade<sup>29</sup>, pois sua descrição não serve só de documentação, mas também de teste para verificar se entendemos realmente sua presença no sistema.

Uma boa descrição de entidade conter os seguintes itens:

- Nome, incluindo uma listagem de sinônimos e homônimos<sup>30</sup>
- Definição
- Exemplos
- Atributos (veremos a seguir)
- Relacionamentos (veremos a seguir)
- Eventos que a utilizam (veremos no próximo capítulo)
- Correlação, descrevendo outras partes da análise que se referem a ela.
- Regras e exceções relacionadas a essa entidade, incluindo regras de negócio.
- Outros comentários e observações
- Uma idéia da quantidade esperada de instâncias no sistema

Durante a definição devemos tentar responder várias perguntas, procurando deixar claro o porquê dessa entidade fazer parte do sistema. Assim devemos nos preocupar em dizer o que é essa entidade, o que faz e para que está no sistema, quando algo é ou não é uma dessas entidades, quando passa a ser ou deixa de ser, ou se é permanentemente.

Quando algum elemento passa de uma entidade para outra devemos tomar bastante cuidado para descrever as ações necessárias para tal fato.



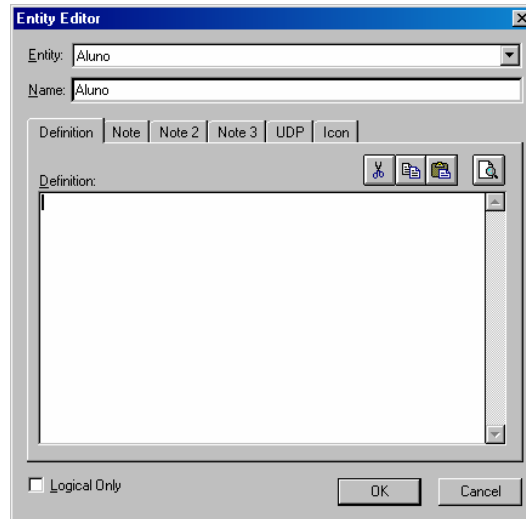
**Figura 51. No início da modelagem temos apenas entidades isoladas**

---

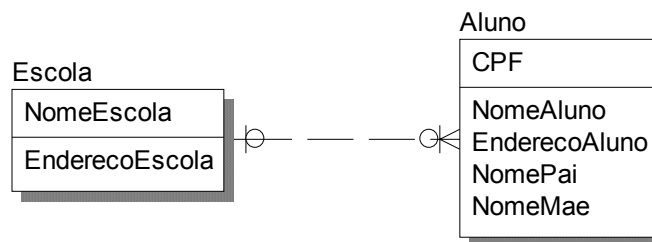
<sup>28</sup> Como no excelente livro Princípios de Modelagem de Dados de David Hay[26].

<sup>29</sup> Devemos confessar que não temos a mesma preocupação com atributos, por exemplo.

<sup>30</sup> Homônimos são objetos diferentes porém com o mesmo nome



**Figura 52.** Uma tela descrevendo uma entidade no software ERWIN 3.5. Atenção para o fato que apesar de não cumprir todos os nossos requisitos em campos distintos, apresenta vários campos de anotação.



**Figura 53.** Como veremos mais adiante, segundo a notação da Engenharia da Informação, apresentamos nessa figura duas entidades que se relacionam: Escola e Aluno.

## VII.7 Relacionamentos

A principal característica das entidades que compõe um sistema é se relacionarem umas com as outras. É impossível imaginar uma entidade isolada em um sistema de informação. Toda entidade deve possuir ao menos um relacionamento que a coloque em contato com as outras entidades do sistema.

Relacionamentos representam que existe alguma conexão entre entidades dentro do negócio sendo analisado [25]. Cada relacionamento deve ser também uma regra de negócio e é utilizado em pelo menos um processo que lida com as entidades envolvidas.

Relacionamentos indicam a possibilidade de buscar um grupo de entidades a partir de outra entidade. Assim, permite encontrar os “visitantes” que “emprestaram” um “livro” específico (navegando de livro para clientes), ou descobrir que “produtos” um “cliente” “pediu” (navegando de clientes para livros). Indicam também que precisamos nos lembrar de algo que envolve, simultaneamente, duas ou mais entidades do sistema, e que essa lembrança só faz sentido quando todas as instâncias envolvidas são recuperáveis simultaneamente ou sequencialmente.

Existem muitos relacionamentos comuns, encontrados em muitos sistemas, como “compõe” (peças compõe máquinas), “é um” (bicicleta “é um” meio de transporte), “faz” ou “gera” (cliente faz ou gera pedido), “atende” (visita atende solicitação de reparo), “usa” (cliente “usa” produto), etc.

O relacionamento “é um” é tão comum, e tão útil, que foi escolhido como um relacionamento especial em muitos métodos derivados da Modelagem Entidade e Relacionamento original. É a relação de herança, onde dizemos que uma entidade “herda” todas as características de outra entidade. A herança equivale à abstração de generalização/ especificação.

Existem duas formas básicas de herança. Na herança exclusiva dividimos uma classe em categorias. Essa forma de herança traz poucas dificuldades na modelagem e é conhecida como separação em categorias. Uma pessoa, por exemplo, pode ser dividida em duas categorias, a dos homens e a das mulheres. Quando a divisão não é exclusiva, ou seja, quando é possível que uma instância de uma entidade específica seja classificada em duas (ou mais) de suas subclasses, temos alguns problemas que devem ser resolvidos na modelagem lógica. Por exemplo, uma pessoa pode ser aluno e professor simultaneamente em uma faculdade.

Também é possível que existam instâncias que não fazem parte de nenhum dos tipos de entidade especializados, mas fazem parte do tipo geral. Isso também exige um tratamento especial durante a modelagem lógica.

Nós dizemos então que:

- A cobertura é total, se cada elemento da classe genérica é mapeado em ao menos um elemento das subclasses.
- A cobertura é parcial, se existe ao menos um elemento da classe genérica não mapeado nas estruturas das subclasses.
- A cobertura é exclusiva, se cada elemento da superclasse é mapeado em no máximo um elemento das subclasses.
- A cobertura é sobreposta, se existe um elemento da superclasse mapeado em mais de um elemento das subclasses.

Devemos tentar obter apenas heranças totais e exclusivas, pois são mais fáceis de serem tratadas.

Dado um grupo de entidades candidatas a construir um relacionamento de herança, devemos analisar se existe um atributo ou relacionamento que é aplicável a apenas um subconjunto dessas entidades, se simplificamos o modelo e se aumentamos sua compreensão. Ou seja, devemos usar a herança para aumentar a semântica do modelo sem causar excesso de informação.

Outro relacionamento tão comum que mereceu um tratamento especial em muitos métodos é o relacionamento “é parte de”. Esse relacionamento equivale à abstração de composição. É normal que utilizemos esse relacionamento apenas quando a parte só existe em função do todo, porém não é uma exigência muito forte.

Relacionamentos podem unir indiferentemente entidades do mesmo tipo ou entidades de tipos diferentes. Quando relaciona entidades do mesmo tipo dizemos que é um auto-relacionamento. Ao especificar um auto-relacionamento devemos ter mais cuidado em declarar os papéis das entidades no relacionamento, além de atentar para não produzir um *loop* infinito no relacionamento.

Normalmente trabalhamos apenas com relacionamentos entre duas entidades. O método original de Chen permitia relacionamentos múltiplos. Atualmente transformamos relacionamentos múltiplos em entidades.

O mesmo acontece com o uso de atributos no relacionamento. Apesar do método original permitir, atualmente criamos uma entidade para representar esse relacionamento.

Bertini et al. [23] mostram algumas operações que, aplicadas a um modelo e-r, criam diagramas diferentes que podem representar uma mesma realidade. Assim, algo que foi representado como uma entidade em um modelo pode ser representado como duas em outro, ou um relacionamento em um modelo pode ser transformado em uma entidade que se relaciona com as entidades originais (ou vice-versa) sem que haja uma representação falsa da realidade. Pode acontecer de uma ou outra representação ser mais interessante em um contexto.

Relacionamentos podem ser condicionais ou incondicionais, isto é, uma entidade pode ser obrigada a ter um relacionamento com outra ou não. Por exemplo: um automóvel é obrigatoriamente fabricado em uma fábrica, mas nem todos os livros em uma livraria já foram vendidos. Como veremos adiante, o fato de um relacionamento ser opcional é representado pela definição da cardinalidade mínima do relacionamento, que pode ser 0 ou 1.

Também é importante notar que existem também relações que ocorrem entre relacionamentos. Dois relacionamentos podem ocorrer sempre juntos (contingentes) ou nunca ocorrer juntos (mutuamente exclusivos). Existem métodos que permitem anotar diretamente no diagrama essas características, porém são pouco utilizados.

**Tudo que não puder ser anotado no diagrama deverá ser anotado em um documento associado.** O principal tipo de anotações são as regras de negócio que funcionam como restrições, como “um professor só pode dar aulas para alunos da escola em que trabalha”. Restrições são geralmente impossíveis de desenhar diretamente no diagrama<sup>31</sup>.

Normalmente associamos restrições a ciclos no diagrama. Por exemplo, se temos que fazer pedidos de livros para uma editora, então temos um relacionamento entre livro e pedido, um entre livro e editora e um entre editora e pedido, formando um ciclo. A restrição é que “um pedido só pode conter livros da editora indicada no pedido”. É possível desenhar o diagrama sem ciclos, eliminado, por exemplo, a ligação entre pedido e editora, porém aconteceriam duas coisas: primeiro teríamos que escrever uma restrição que é possivelmente mais complexa (“um pedido só pode conter livros da mesma editora”), segundo não teríamos nenhuma indicação no diagrama que o pedido é feito

---

<sup>31</sup> Ross, porém, propõe uma linguagem gráfica que permite a definição de restrições. A linguagem é muito complexa e não existe ainda nenhuma ferramenta CASE que a suporte.

para a editora, exigindo uma nova regra. Finalmente, a falta do ciclo funciona também como falta de indicação que existe uma restrição, pois todo ciclo é um aviso de restrição<sup>32</sup>.

### VII.7.1 Cardinalidade

Para bem representar um relacionamento, devemos indicar a cardinalidade desse relacionamento, isto é, quantas vezes uma instância da entidade pode se relacionar com instâncias de outras entidades.

Veja por exemplo o relacionamento “mãe-filha”. Uma filha só pode ter uma mãe, mas uma mãe pode ter várias filhas.

Existem três tipos básicos de relacionamentos: o 1:1, um para um, o 1:N, um para muitos, e o N:M, muitos para muitos. Nesse caso só estamos falando da cardinalidade máxima. A cardinalidade máxima indica quantas vezes uma entidade pode aparecer em um relacionamento.

No relacionamento 1:1 cada entidade só pode se relacionar com uma entidade do outro conjunto. Geralmente indica semelhança, igualdade, utilização conjunta, etc.

No relacionamento 1:N cada entidade de um conjunto pode se relacionar com várias entidades do outro conjunto, mas as entidades do segundo conjunto só podem se relacionar com uma entidade do primeiro conjunto. Geralmente indicam relações de posse, hierarquia ou de composição.

No relacionamento N:M qualquer número de relacionamentos é válido. Podem indicar várias coisas, como eventos, contratos, acordos, ligações temporárias como empréstimos e aluguéis, etc. É normal aparecerem também quando o relacionamento é do tipo 1:1 ou 1:N em certo momento ou período (como o aluguel de uma fita de vídeo), mas se deseja manter a história de todos os relacionamentos.

Quando falamos também da cardinalidade mínima usamos notação de par ordenado, (0,1):(1,N) por exemplo, onde o primeiro número do par indica a cardinalidade mínima e o segundo a máxima. A cardinalidade mínima indica uma exigência da participação de uma instância da entidade em relacionamentos. A cardinalidade mínima 0 em ambos os lados indica a existência própria de ambos os objetos. A cardinalidade mínima 1 pode indicar a necessidade de um objeto pertencer ou ser criado por outro.

É comum evitarmos relacionamentos onde ambos os lados exijam como cardinalidade mínima “1”. O motivo é que um par de entidades só pode ser colocado na memória do sistema em uma mesma transação, não permitindo que primeiro coloquemos a instância de uma entidade na memória e depois uma instância relacionada da outra entidade.

Temos então os seguintes tipos de relacionamentos:

- Relacionamentos um para um.
  - (0,1):(0,1)

---

<sup>32</sup> Mas a ausência de um ciclo não significa que não existe restrição.

- Esse relacionamento significa que uma instância do primeiro conjunto pode ou não se relacionar com uma instância do segundo conjunto, porém pode ter apenas um relacionamento. O mesmo vale do segundo conjunto para o primeiro.
- Esse relacionamento é encontrado quando é possível formar pares entre duas entidades, mas esses pares são opcionais.
- Um exemplo seria o caso da alocação cabines e reboques de caminhões em uma empresa de aluguel de viaturas. Cabines e reboques podem ser trocados arbitrariamente. Em certo momento, cada cabine só pode ter um reboque e vice-versa. Além disso, algumas vezes uma das partes fica guardada na garagem enquanto a outra é utilizada.
- Outro caso que podem mostrar são auto-relacionamentos desse tipo. Uma igreja ou um templo, por exemplo, pode ter um catálogo de freqüentadores e querer saber quem é casado com quem (e quem é solteiro, ou seja, não tem nenhum relacionamento).

○  $(1,1):(0,1)$

- Esse relacionamento significa que a primeira entidade obrigatoriamente tem um relacionamento, mas ele é opcional para a segunda entidade. Em ambos os casos apenas um relacionamento é permitido.
- Esse relacionamento é encontrado quando uma entidade possui ou controla de alguma forma outra. Em alguns casos as duas entidades podem ser unidas em uma só.
- Ele é menos comum que o relacionamento  $(1,1):(0,N)$ .
- Um exemplo seria uma distribuição de papéis de uma peça de teatro em uma companhia de atores. Cada ator só pode fazer um papel, alguns atores podem não ter papel, mas todos os papéis têm um ator, e apenas um ator.

○  $(0,1):(1,1)$ , similar ao anterior

○  $(1,1):(1,1)$

- Esse relacionamento é pouco comum, pois indica que uma entidade não pode existir sem estar relacionada com outra, e tudo isso apenas uma vez. Normalmente pode ser substituído pela unificação das duas entidades em uma só.
- Algumas vezes é utilizado para diferenciar aspectos diferentes da mesma entidade. Por exemplo, um avião é uma entidade que tem visões comerciais, de mecânica, de



operação, etc. Fica muito complicado, em um modelo ER, colocar todos os atributos, que chegam a centenas, em uma só entidade, assim podem ser criados relacionamentos (1,1):(1,1) para tratar essa modelagem.

- Esse relacionamento não é recomendado, pois exige que ambas as entidades sejam sempre criadas juntas.

- Relacionamentos 1 para N

- (0,1):(0,N)

- A primeira entidade pode ou não participar do relacionamento, mas apenas uma vez. A segunda entidade pode ou não participar do relacionamento, e ainda pode fazê-lo várias vezes.
    - Esse é um relacionamento muito comum. Normalmente significa que dois objetos que não possuem nenhum relacionamento de propriedade ou restrição de existência podem ser colocados em uma relação hierárquica.
    - Exemplo: esse tipo de relacionamento pode ser encontrado em locais onde temos um estoque de objetos que são alocados a departamentos da empresa, por exemplo, computadores. Um computador só pode estar alocado em um departamento ou pode estar no estoque (sem alocação). Um departamento pode ter 0, 1 ou vários computadores alocados para si.

- (0,N):(0,1), similar ao anterior

- (0,1):(1,N)

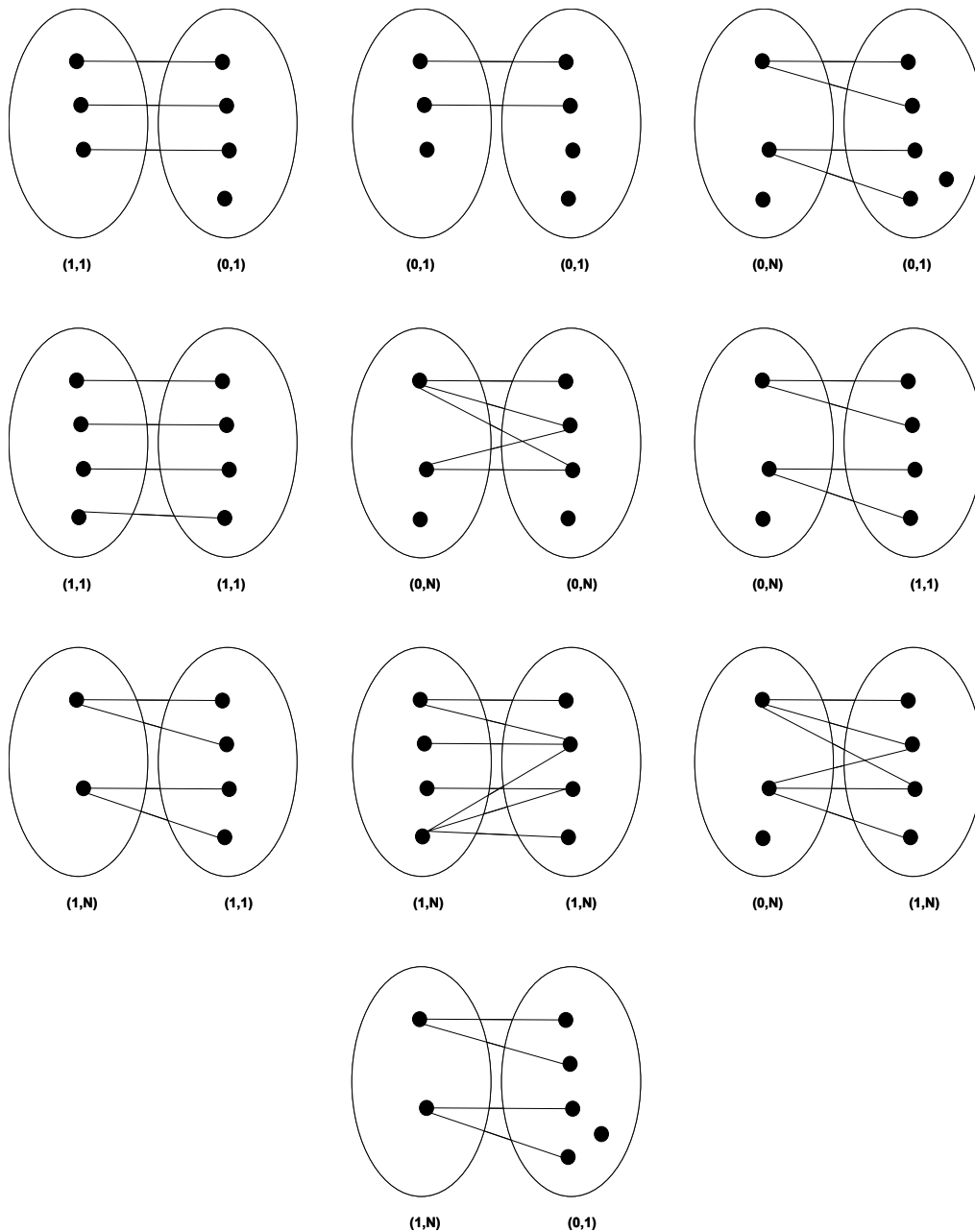
- Esse relacionamento normalmente também indica uma relação hierárquica.. O primeiro objeto pode opcionalmente pertencer a essa relação, enquanto o segundo objeto obrigatoriamente pertence a relação.
    - Não é muito comum, pois exige que uma instância tenha no mínimo uma “filha”, mas as filhas podem existir de forma independente.
    - Pode ser usado quando algo para existir deve ter ao menos uma parte, mas estas partes tem vida própria, mesmo só podendo ser usadas em um lugar.
    - Isso pode ser encontrado, por exemplo, no relacionamento entre uma venda e os itens (quando únicos) vendidos. Uma loja de carros novos, por exemplo, pode em uma mesma venda negociar vários carros, mas necessariamente a venda contém um carro. Já o carro pode ter sido vendido ou não.

- (1,N): (0,1), similar ao anterior

- (1,1):(0,N)
  - Indica uma “maternidade” da segunda entidade em relação à primeira, ou seja, cada instância da primeira entidade é obrigada a possuir uma “mãe”, e apenas uma, que seja instância da segunda entidade.
  - É um dos relacionamentos mais comuns.
  - Pode ser encontrado, por exemplo, na relação entre automóveis de uma empresa e multas recebidas. Cada multa é de apenas um automóvel, mas podem existir automóveis com 0, 1 ou mais multas.
- (0,N):(1,1), similar ao anterior
- (1,1):(1,N)
  - Indica uma “maternidade” da segunda entidade em relação à primeira, ou seja, cada instância da primeira entidade é obrigada a possuir uma “mãe”, e apenas uma, que seja instância da segunda entidade. Além disso, obrigatoriamente a “mãe” deve possuir uma filha.
  - Esse relacionamento apresenta o inconveniente de exigir criar uma entidade “filha” para criar a entidade “mãe”.
  - Pode ser encontrado, por exemplo, em um cadastro de pessoas jurídicas, que devem possuir um endereço, mas podem possuir mais de um.
- (1,N):(1,1), similar ao anterior
- Relacionamentos N para M
  - (0,N):(0,N)
    - Esse relacionamento é muito comum. Representa a forma mais geral de relacionamento, opcional e com todas as possibilidades para ambos os lados.
    - Pode ser encontrado, por exemplo, na relação entre alunos e cursos oferecidos em um semestre em uma universidade. Alguns cursos não recebem inscrição, alguns alunos não fazem inscrição em nenhum curso.
  - (0,N):(1,N)
    - Semelhante ao (0,N):(0,N). Também muito comum, porém agora exigimos que haja pelo menos um relacionamento na segunda entidade.
    - Pode ser encontrado, por exemplo, na relação entre músicas e CDs onde estão gravadas, para controle de uma discoteca. Uma mesma música pode estar em vários CDs, mas não é

possível registrar um CD sem músicas (deve existir pelo menos uma). Porém uma música pode nunca ter sido gravada.

- $(1,N):(0,N)$ , similar ao anterior
- $(1,N):(1,N)$ 
  - Aqui temos um relacionamento múltiplo que deve existir pelo menos uma vez.
  - Um exemplo é o relacionamento entre salas de uma empresa e móveis colocados nessa sala.
  - Essa representação muitas vezes é verdadeira, mas é evitada, sendo trocada pelo relacionamento  $(0,N):(1,N)$ , pois exige que ambas as entidades sejam sempre criadas juntas.



**Figura 54. Tipos de relacionamentos entre entidades, baseado no conceito que uma entidade é um conjunto de instâncias.**

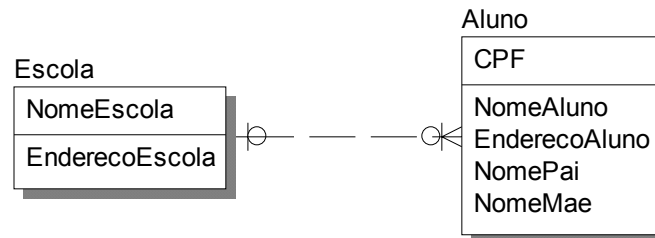
### VII.7.2 Descrevendo Relacionamentos

Relacionamentos podem ser descritos por linhas ligando duas entidades ou por um losango ligado por linhas às entidades. Em ambos os casos é possível anotar os relacionamentos com nomes e com a sua cardinalidade (ver exemplos mais a frente).

O nome escolhido para o relacionamento pode estar na voz ativa (mãe gera filho) ou na voz passiva (filho é gerado por mãe). Algumas notações permitem que se usem os dois nomes (um por cima e um por baixo da linha de relacionamento). Geralmente se usa

o nome que permite a leitura do relacionamento da esquerda para a direita na parte de cima da linha (ou se dá preferência a esse nome quando apenas um pode ser utilizado).

Relacionamentos também devem ser descritos e comentados, sendo importante responder qual sua função no sistema, o que eles representam, como e quando são estabelecidos ou destruídos.



**Figura 55. Um relacionamento entre escola e alunos. Como veremos mais adiante, segundo a notação da Engenharia da Informação, uma escola pode ter 0,1 ou mais alunos, enquanto um aluno está em uma escola ou em nenhuma escola.**

## VII.8 Atributos

Todo atributo descreve de alguma forma a instância da entidade. Alguns atributos são especiais e definem a entidade, mesmo que não de forma unívoca. Esses são os atributos nominativos. Outros atributos permitem definir outro objeto que não é o sendo tratado, são ou atributos referenciais. Um exemplo de atributo referencial é “fábrica” para “automóvel”, referenciando a fábrica onde foi construído. É uma opção do analista criar ou não entidades que permitem a substituição de um atributo referencial por um relacionamento<sup>33</sup>.

### VII.8.1 Descrevendo Atributos

Devemos definir as seguintes características:

- Nome
- Descrição
- Domínio (valores válidos, como inteiro, real, string ou uma lista de valores).
- Exemplos

Na descrição devemos nos preocupar em explicar qual a finalidade do atributo, como são atribuídos os valores, o que significa cada valor, quem define a escolha do valor, quando, por que e por quem o valor é atribuído ou alterado, etc.

Atributos são atualmente denotados no mesmo retângulo da entidade, como mostrado nos exemplos a seguir.

<sup>33</sup> Especificações ou descrições

## VII.9 Identificando Entidades

Como vimos no início do capítulo, uma abstração importante é a identificação. No caso de modelos ER, é essencial que cada instância de uma entidade possa ser identificada unicamente com um objeto ou conceito do mundo real. Para fazer essa identificação são utilizados atributos e relacionamentos identificadores.

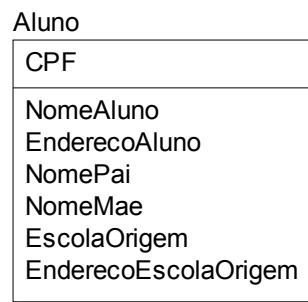
Algumas vezes mais de um atributo, ou relacionamento, serve como identificador único, porém de forma independente. No Brasil, esse é o caso das placas de carro e dos números de chassi. Definido um, o outro está automaticamente definido, mas ambos podem ser escolhidos de forma independente como identificador principal. Dizemos que ambos são chaves candidatas. Uma chave candidata não pode conter atributo que não auxiliem na identificação única da instância. O que for escolhido será a chave primária, ou outros são conhecidos como chaves alternativas.

### VII.9.1 Atributos Identificadores (Chaves Candidatas e Chaves Primárias)

Alguns atributos têm o poder de distinguir as ocorrências das entidades, isto é, servem para identificar univocamente uma instância de entidade à instância do mundo real<sup>34</sup>. Definido o valor desse atributo, os outros valores são dependentes e não podem ser escolhidos, mas sim devem possuir um valor exato seguindo a realidade.

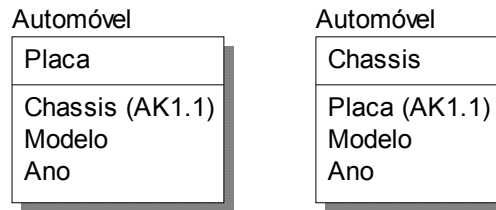
Um atributo identificador típico em sistemas financeiros é o CPF de uma pessoa física ou o CNPJ de uma pessoa jurídica. Definido o CNPJ, a empresa, e todos os seus dados, estão univocamente definidos (nome fantasia, endereço, etc.) no mundo real, e assim deve seguir o sistema que estamos construindo.

Muitas vezes precisamos de mais de um atributo identificador para realmente identificar uma instância. Dizemos então que a chave primária é composta. Se usarmos apenas um atributo como identificador, então dizemos que a chave primária é simples.



**Figura 56. Entidade Aluno, identificada pelo atributo CPF, na notação da Engenharia da Informação.**

<sup>34</sup> Ou seja, servem para modelar a abstração de identificação



**Figura 57. A entidade automóvel pode ser identificada unicamente tanto pela placa como pelo Chassi, levando a dois modelos diferentes como mostrado nesta figura. A notação fornecida pela ferramenta Erwin permite a identificação das chaves alternativas (AK – Alternate Key).**

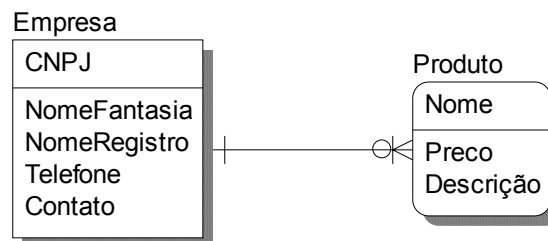
## VII.9.2 Relacionamentos Identificadores

Algumas instâncias são identificadas também, ou até mesmo unicamente, por seus relacionamentos. Uma forma de denotar isso é utilizar uma linha mais grossa no relacionamento ou algum símbolo específico.<sup>35</sup>

Alguns autores chamam as entidades que são identificadas por seu relacionamento com outras entidades de “entidades fracas” ou “entidades dependentes”. Atualmente esses nomes são considerados derogatórios para entidades que podem ser muito importantes em um modelo. Os alunos também, muitas vezes, tendem a achar que não devemos modelar “entidades fracas”, conclusão que está absolutamente errada.

### VII.9.2.1 Chaves Estrangeiras

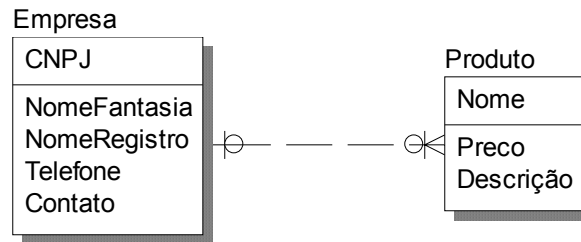
No modelo conceitual não existe o conceito de chave estrangeira, que é uma característica do modelo relacional. Uma chave estrangeira é uma chave de outra tabela que usamos em uma tabela para indicar o relacionamento. Porém, é comum que as ferramentas de modelagem copiem as chaves estrangeiras automaticamente<sup>36</sup>. Em benefício da prática atual, e em detrimento da pureza teórica, mostramos a seguir algumas possibilidades da notação de relacionamento.



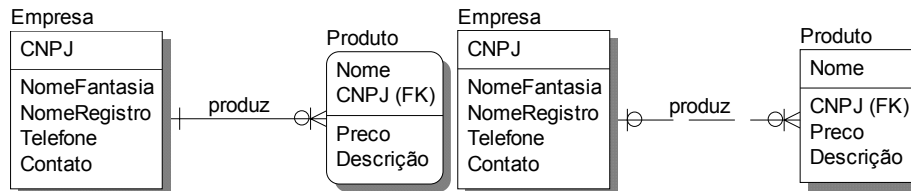
**Figura 58 Entidades identificadas por um relacionamento (Produto) podem ser denotadas de uma forma diferenciada, para declarar que sua existência é dependente da existência de outra entidade. Software Erwin.**

<sup>35</sup> A notação IDEF1X, por exemplo, usa um círculo negro.

<sup>36</sup> No Erwin essa cópia é chamada “migração” e é opcional no modelo lógico.



**Figura 59. O mesmo relacionamento, agora não identificador. Percebemos que a entidade Produto agora tem o seu símbolo normal. Software Erwin.**



**Figura 60. Uma visão do modelo lógico ainda do mesmo relacionamento, agora recebendo um nome. Perceba que, dependendo do relacionamento ser identificador ou não, a chave da entidade “mãe” é copiada para a chave ou para os atributos comuns da entidade “filha”. Software Erwin**

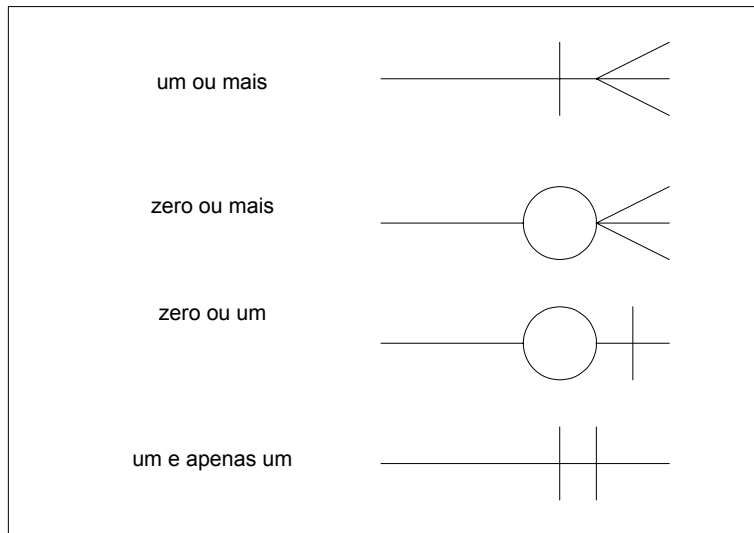
## VII.10 Descrição Gráfica do Modelo

Várias são as notações existentes para o modelo de entidade e relacionamento. Usaremos nesse texto a notação de Martin, também conhecida como Information Engineering, fornecida pelo software Erwin.

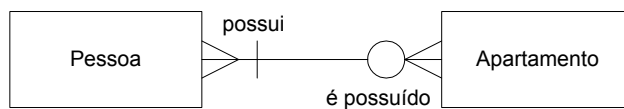
Nessa notação não temos um símbolo para relacionamentos, apenas um retângulo para entidades. Os relacionamentos são indicados por linhas. Uma linha cheia indica um relacionamento identificador. Uma linha tracejada indica um relacionamento não identificador. Por isso, não podemos usar relacionamentos com atributos, necessitando de uma nova entidade nesse caso. Também não podemos criar relacionamentos múltiplos, necessitando de criar entidades para representá-los.

Apesar de parecer que temos um modelo menos poderoso, temos na verdade apenas uma sintaxe mais simples, com o mesmo poder de modelagem. Algumas decisões também ficam tomadas automaticamente também. Por exemplo, não precisamos decidir se um “objeto com atributo” é um relacionamento ou uma entidade, pois relacionamentos não têm atributos em nosso modelo. Acreditamos que a modelagem segundo as regras do IDEF1X ou da Engenharia de Informação possibilita encontrar mais facilmente um modelo essencial do sistema que as regras tradicionais de Chen ou ainda extensões as mesmas.

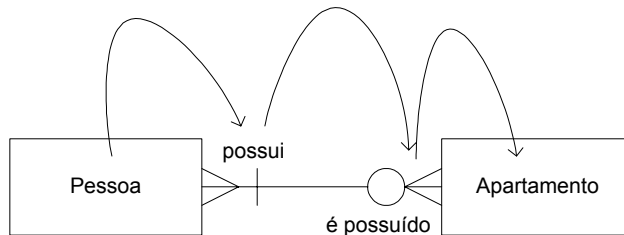




**Figura 61. Notação para a cardinalidade**



**Figura 62. Uma pessoa possui zero ou mais apartamentos e um apartamento é possuído por pelo menos uma pessoa ou mais**



**Figura 63. As setas indicam a forma de leitura do diagrama**

A cardinalidade é indicada por três símbolos usados na ponta da linha que indica o relacionamento: uma linha indica 1, um círculo indica 0 (zero), e um “pé-de-galinha” indica n. Dessa forma podemos anotar o mínimo e o máximo da cardinalidade usando dois símbolos em cada ponta.

O nome do relacionamento é colocado acima (à esquerda) da linha que o indica, sendo o nome do relacionamento inverso colocado abaixo (à direita). Normalmente se lê a notação partindo de uma entidade, lendo o nome do relacionamento, lendo a cardinalidade da ponta oposta e finalmente o nome da segunda entidade.

Nessa notação os atributos podem ser colocados dentro da caixa que representa as entidades, como apresentado na próxima seção.

### VII.10.1 Exemplos de notações

Vamos descrever a seguir o seguinte modelo em várias notações:

Apresentaremos o modelo de uma locadora de vídeo. A locadora trabalha com **fitas de vídeo**. Cada **fita de vídeo** contém um **filme**, porém cada fita deve ser identificada unicamente, pois elas podem ser **dubladas** ou **legendadas**. As fitas são **emprestadas** para **clientes** em um dia e hora específico. Um cliente pode ficar com várias fitas, ou nenhuma. Uma fita pode estar com apenas um cliente, ou estar na loja e não estar com cliente nenhum. É importante saber para quem cada fita específica foi emprestada, para auditar clientes que estragam fitas, por isso todas as fitas são numeradas com um código único. Os **filmes** são dirigidos por **diretores** e contém **atores**.

Observamos que o cliente é um papel assumido por uma pessoa, a fita de vídeo é um objeto físico, existente, o filme é uma obra de arte que está representada na fita (um conceito), diretor e ator são também papéis assumidos por pessoas dentro da idéia de filme e que um aluguel é um contrato entre o cliente e a locadora.

Atenção para outro detalhe: não existe a entidade locadora, pois este sistema é destinado a uma só locadora. Seria uma entidade única, que claramente é uma constante do sistema. A presença de entidades desse tipo é um erro comum nos modelos feitos por principiantes. Porém, se tivéssemos um software para uma rede de locadoras, seria interessante guardar em que locadora está cada fita, o que exigiria essa entidade.

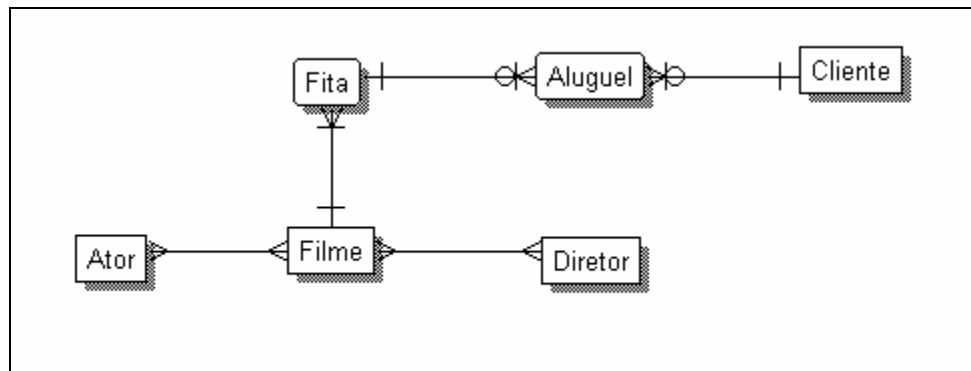
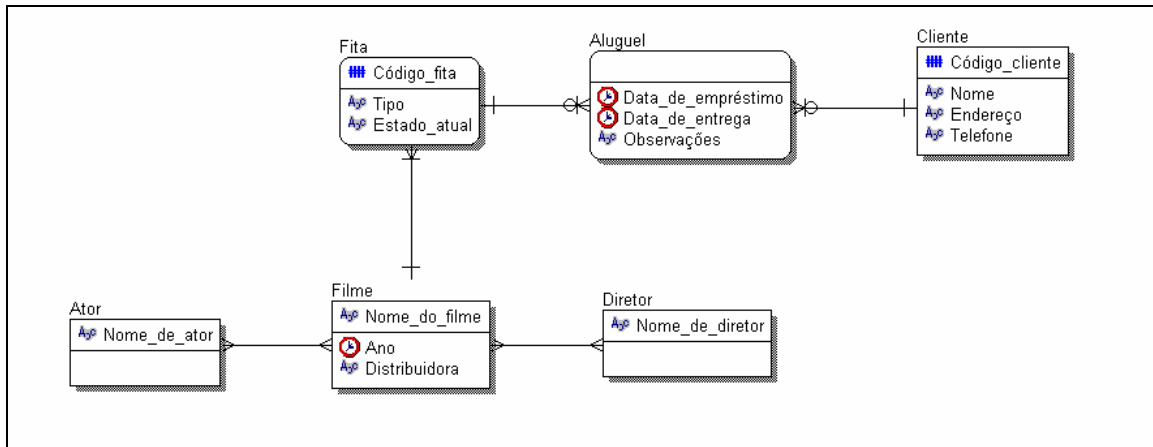
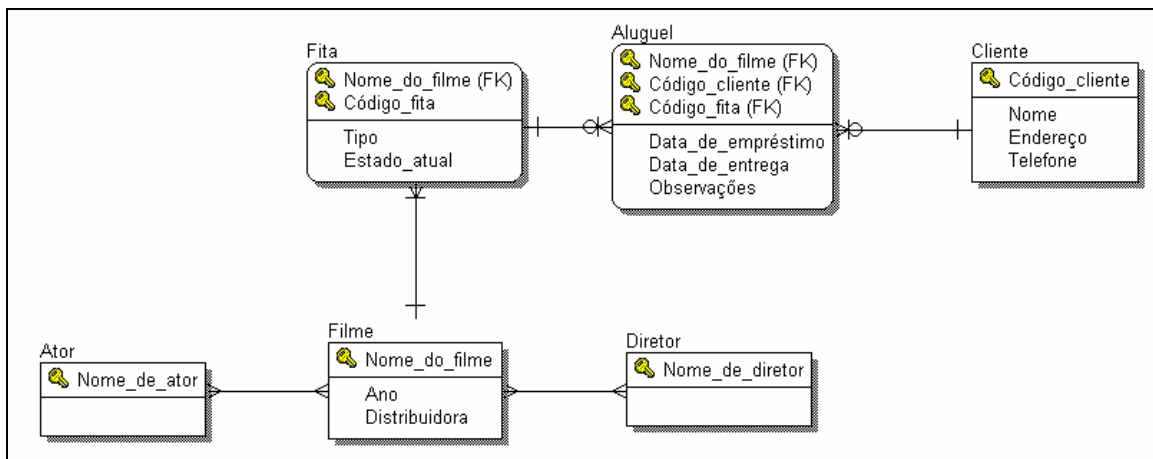


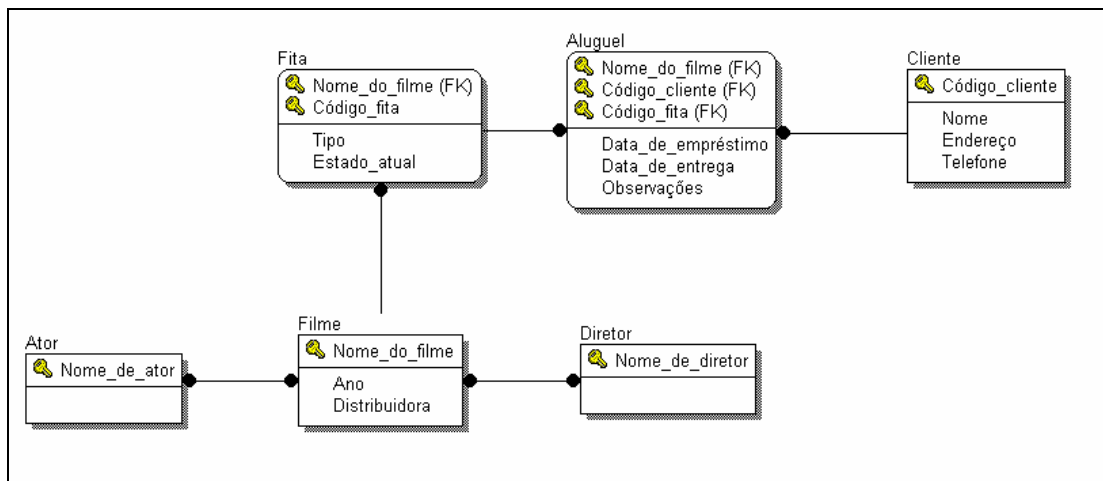
Figura 64. Modelo inicial da locadora, notação Information Engineering, software ERWIN 3.5.



**Figura 65. Modelo da locadora, com atributos e os tipos (com ícones) dos atributos, notação EI, software ERWIN 3.5.**



**Figura 66. Modelo da locadora, mostrando chaves primárias e chaves estrangeiras (FK) (que não deviam estar em um modelo conceitual!), notação EI, software ERWIN 3.5.**



**Figura 67. O mesmo modelo anterior, com a notação IDEF1X, software ERWIN 3.5.**

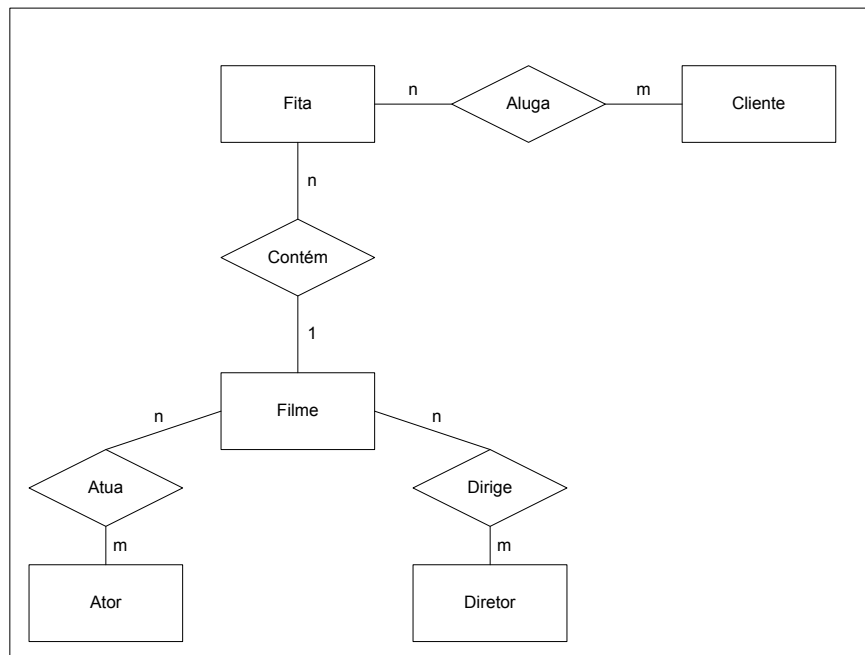
#### VII.10.1.1 A Cardinalidade nas Notações

Podemos identificar pelo menos três escolas quanto à maneira de denotar a cardinalidade das notações.

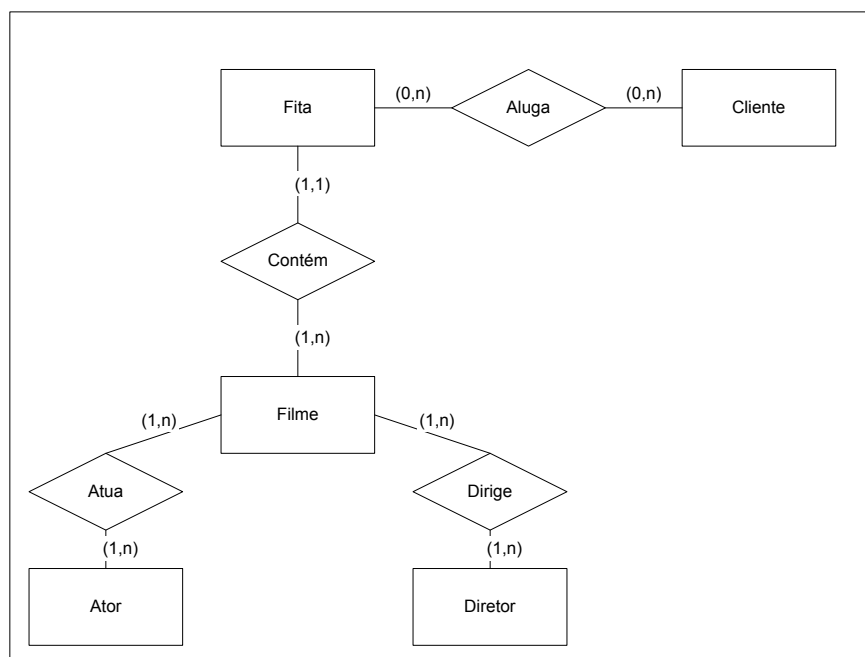
A primeira, e original, é chamada **associativa**, e indica junto à entidade quantas ocorrências da mesma podem estar associadas a uma determinada entidade. Veja na Figura 68 que um filme está associado a várias fitas.

É a que encontramos na notação da Engenharia da Informação. A segundo é a **participativa**, que indica quantas vezes uma entidade participa de um relacionamento. Na Figura 69 um filme participa até várias vezes do relacionamento. Essa interpretação está mais perto da idéia matemática que o relacionamento é um conjunto de pares ordenados.

Finalmente, modelos com IDEF1X usam uma notação própria, com significado dependente de uma combinação especial de símbolos.



**Figura 68. O mesmo modelo, segundo a notação original de Peter Chen, associativa. Note que escolhemos manter o aluguel como um relacionamento nesse modelo, que permite relacionamentos com atributos, software Visio 2000.**



**Figura 69. Mesma figura anterior, agora usando a notação participativa, com mínimos e máximos de cardinalidade proposta por Ceri. Atenção que as cardinalidades ficam na posição contrária à notação anterior, software Visio 2000.**

### VII.10.2 Notação adotada

Podemos adotar qualquer notação, contanto que seja utilizada de forma consistente em um projeto ou em uma empresa. No curso que ministramos, porém, sugerimos as seguintes notações:

No trabalho: se escolher a ferramenta Erwin, utilize a notação de information engineering. Se escolher o Visio 2002, utilize a notação que utiliza “crow-foot”. Apresente os tipos dos atributos e as chaves primárias, mas não represente as chaves estrangeiras.

Na prova: Muitas vezes é melhor representar os atributos como círculos do que dentro das caixas.

## VII.11 Técnicas de Desenvolvimento do Modelo

A seguir apresentamos quatro estratégias básicas para desenvolver o modelo ER de um sistema. Nenhuma estratégia é melhor que as outras em todos os casos, nem todas as estratégias vão levar a mesma solução.

### VII.11.1 Técnica Top-Down

Na técnica top-down, desenvolvemos o modelo ER partindo de entidades altamente abstratas e aplicando transformadas que permitem encontrar entidades menos abstratas e mais representativas do sistema sendo desenvolvido. O processo termina quando todos os requisitos foram representados.

Essa técnica necessita que o analista possa construir um modelo abstrato em sua mente, o que pode ser difícil em grandes sistemas.


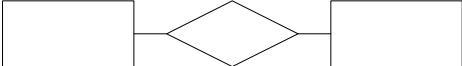
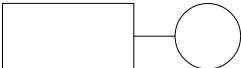
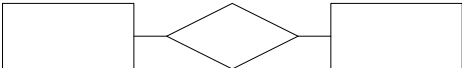

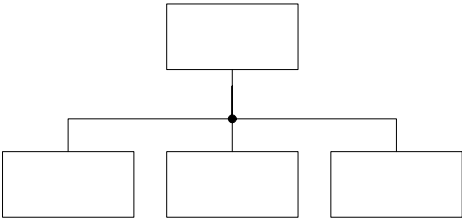


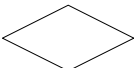
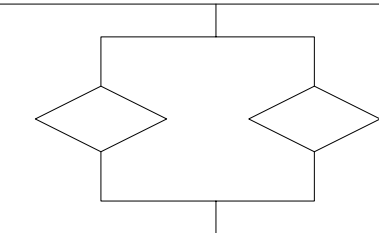
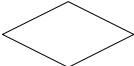


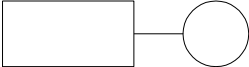
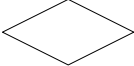
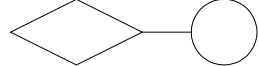
Heuser [27] sugere os seguintes passos para essa técnica:

1. Construção de um modelo superficial
  - a. Levantamento das entidades
  - b. Identificação dos relacionamentos, com cardinalidades máximas.
  - c. Identificação dos atributos
  - d. Determinação dos atributos identificadores
  - e. Verificação do aspecto temporal
2. Construção do modelo detalhado
  - a. Determinação dos domínios dos atributos
  - b. Determinação das cardinalidades mínimas dos relacionamentos
  - c. Levantamento das restrições de integridade não representáveis no modelo
3. Verificação do modelo, buscando construções redundantes ou deriváveis
4. Validação junto ao usuário

A seguir veremos essas transformações, tratando algumas vezes de uma questão que só é abordada um pouco mais tarde, as formas normais.

- Transformar uma entidade em duas entidades relacionadas: muitas vezes dentro de uma entidade estamos na realidade olhando duas entidades mais específicas. Duas formas são possíveis: a entidade original existe, mas contém dentro dela outra entidade, ou a entidade original é na verdade dividida em duas entidades. No primeiro caso estamos “extraíndo” a nova entidade da entidade original. No segundo caso estamos detalhando parte do nosso modelo. O primeiro caso pode se confundir naturalmente com o caso a seguir, onde um atributo é transformado em uma entidade, dependendo do momento da especificação. Muitas vezes uma entidade na verdade contém dados que compõe duas entidades.
- Transforma atributo, ou conjunto de atributos, em entidade e relacionamento: isso pode acontecer quando um atributo é múltiplo (quebrando a primeira forma normal), é uma especificação (por exemplo, uma marca), ou ainda em casos que quebram a segunda e terceira forma normal.
- Criação de entidades mais específicas: algumas vezes identificamos inicialmente uma entidade que é um tipo muito geral e mais tarde descobrimos tipos específicos que representam melhor o domínio da aplicação.

- Transformar uma entidade em várias entidades não relacionadas: isso não é muito normal, pois entidades não relacionadas normalmente não se encontram modeladas dentro de uma mesma entidade, mesmo que temporariamente. Mesmo assim, pode ser um primeiro passo para depois descobrirmos qual é o relacionamento que as une.
- Transformar um relacionamento em dois relacionamentos em paralelo: apesar de não acontecer muitas vezes em um mesmo sistema, é uma transformação comum. Acontece quando entendemos que duas entidades são relacionadas e apenas mais tarde compreendemos que elas são relacionadas de várias formas diferentes e não de uma só forma.
- Transformar um relacionamento em uma entidade: essa modificação é muito comum. Muitas vezes nossa compreensão inicial de um evento ou de uma relação qualquer entre duas entidades é muito simples. Mais tarde, compreendendo melhor esse relacionamento, entendemos que é mais interessante representá-lo como uma entidade.
- Criação de atributos: é a modificação mais simples e comum. Só estamos citando para completar o conjunto de operadores top-down.

Uma entidade pode ser transformada em duas entidades relacionadas		
Um atributo de uma entidade pode ser transformado em uma entidade relacionada		
Uma entidade pode ser transformada em uma entidade e um conjunto de especializações.		
Uma entidade pode ser transformada em um número de entidades não relacionadas		
Um relacionamento pode ser transformado em dois relacionamentos em paralelo		
Um relacionamento pode ser transformado em uma entidade relacionada com as duas entidades ligadas pelo relacionamento		
Um entidade pode receber um atributo		
Um relacionamento pode receber um relacionamento		

**Figura 70. Transformadas Top-Down**

#### **VII.11.1.1 Como escolher entre um atributo ou Entidade e Relacionamento**

Se o conjunto de valores for fixo durante toda a vida do sistema, pode ser modelado como atributo. Se for variável, então deve ser um relacionamento com outra entidade.

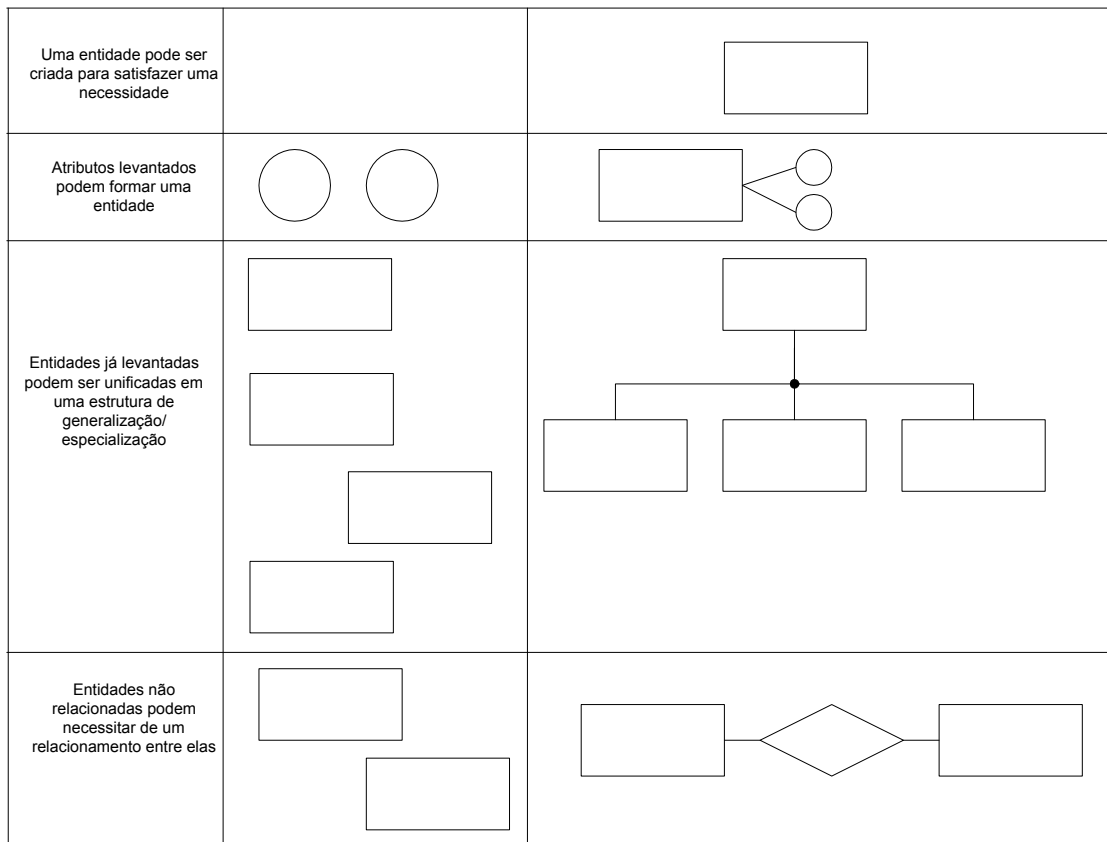
Se tiver relação com outro objeto deve ser um relacionamento. Caso contrário, pode ser um atributo.



### VII.11.2 Técnica Bottom-Up

Na técnica Bottom-Up, partimos das partes para construir o todo, partindo dos conceitos mais elementares para construir conceitos mais complexos. Os requisitos são decompostos, analisados de forma independente e agregados em um esquema global [23].

- Criação de entidade ou atributo: são as operações básicas da técnica bottom-up.
- Unificação de atributos em uma entidade:
- Organização de entidades em uma hierarquia de herança:
- Criação de relacionamento entre entidades



**Figura 71. Transformadas Bottom-Up**

### VII.11.3 Técnica Inside-Out

Inicia nos conceitos mais importantes e navega em direção aos menos importantes. É comum que modelos E-R se desenvolvem em torno de algumas entidades que representam os conceitos mais importantes de um domínio ou aplicação. A partir desses conceitos buscam-se entidades relacionadas, possivelmente usando tanto operações das técnicas top-down como bottom-up.

#### **VII.11.4 Técnica Mista**

Normalmente, modelos E-R não são desenvolvidos de forma Top-Down ou Bottom-up, mas sim de uma forma mista, principalmente quando a uma grande quantidade de entidades no esquema. Dessa forma, um esquema inicial de alto nível é dividido, de forma que cada partição possa ser considerada separadamente.

#### **VII.11.5 Que técnica usar**

Desaconselhamos o uso da técnica bottom-up. Acreditamos que a verdadeira modelagem E-R deve seguir uma técnica mista, a partir da compreensão do analista do que são as entidades e seus atributos, mais parecida com a técnica inside-out.

#### **VII.11.6 Equivalência de modelos**

Dois modelos são equivalentes quando representam uma mesma realidade.

Algumas equivalências são facilmente identificáveis:

- Relacionamentos  $n \times m$  podem ser substituídos por uma entidade<sup>37</sup>.
- Relacionamentos  $1 \times 1$  podem ser eliminados, unificando-se as entidades<sup>38</sup>.

### **VII.12 Representando o Aspecto Temporal**

Muitas vezes uma aplicação necessita que sejam representados aspectos temporais. Isso acontece, por exemplo, quando o preço de um contrato depende de sua data de contratação, de acordo com vários planos.

Dois efeitos são interessantes de serem discutidos: a necessidade de manter um histórico do valor de um atributo (por exemplo, para responder a perguntas como “quanto custava uma ligação telefônica no dia 14 de novembro de 2001”), como manter um histórico de relacionamentos (por exemplo, para saber quais fitas o cliente alugou no passado, permitindo que uma fita seja alugada várias vezes).

No caso de necessitarmos de um histórico do valor do atributo, é necessário criar uma nova entidade que represente o valor e a data de validade desse valor, sendo que essa entidade se relaciona com a entidade original.

No caso de necessitarmos que um relacionamento seja mantido no histórico, é necessário criar atributos que indiquem a validade desse relacionamento. Na prática, o relacionamento se torna uma entidade.

---

<sup>37</sup> Mas não precisam ser obrigatoriamente substituídos. Modelos conceituais admitem e até ficam mais claros com a presença de relacionamentos  $n \times m$ .

<sup>38</sup> Principalmente relacionamentos  $(1,1):(1,1)$ . Qual o motivo de ter um par de entidades que só podem existir juntas no sistema e considerar como entidades distintas?

## VII.13 Formas Normais

As formas normais foram criadas para o modelo relacional<sup>39</sup>, para serem aplicadas no modelo lógico, porém existem vantagens em utilizá-las no modelo conceitual, pois melhoram a qualidade do modelo em relação a alguns quesitos<sup>40</sup>. O ato de normalizar implica na criação de algumas tabelas que não são necessariamente criadas pelo analista preocupado apenas com o modelo conceitual, influenciando sua longevidade e simplicidade total, diminuindo a redundância e favorecendo a possibilidade de dois analistas chegarem ao mesmo modelo por vias independentes, ou concordarem em adotar um modelo único. Também permitem que algumas discussões, do tipo “X é entidade ou atributo” sejam decididas imediatamente. Essas características são objetivos claros da modelagem essencial e por isso nos parece bastante adequado utilizar modelos conceituais normalizados.

O tratamento dado às formas normais nesse texto é apenas introdutório devendo o leitor se referir a livros de bancos de dados ou de modelagem de dados para uma abordagem mais completa.

### VII.13.1 Primeira Forma Normal (1FN)<sup>41</sup>

Algumas definições equivalentes, próprias do modelo relacional:

- Diz-se que uma tabela está na primeira forma normal quando todos os seus atributos são atômicos.
- Diz-se que uma tabela está na primeira forma normal se cada coluna contém apenas um valor e se cada linha contém as mesmas colunas.
- Diz-se que uma tabela está na primeira forma normal quando não contém tabelas aninhadas.
- Diz-se que um modelo está na primeira forma normal se:
  - Está integrado por tabelas
  - As linhas da tabela são unívocas
  - A linha não contém itens repetitivos
  - Os atributos são atômicos
  - O atributo não contém valores nulos<sup>42</sup>

---

<sup>39</sup> Se você tem dúvidas sobre a diferença entre o modelo relacional e o modelo de entidades e relacionamentos: o modelo relacional fala sobre a representação de dados como tuplas (relações matemáticas) em tabelas, o modelo de entidades e relacionamentos fala da representação do mundo real em um modelo abstrato composto de tipos de entidades e relacionamentos entre essas entidades.

<sup>40</sup> Alguns autores não sugerem as formas normais em seus modelos conceituais e normalizam apenas seus modelos lógicos. Essa prática pode ser altamente prejudicial ao entendimento do problema, pois as formas normais nos auxiliam a desenvolver um modelo mais correto.

<sup>41</sup> Uma tabela que não está na primeira forma normal é denotada como NÑ (não normalizada) ou NFNF (Non-first normal form) NF<sup>2</sup>

<sup>42</sup> Essa regra foi abandonada na prática

A seguinte tabela não está na 1FN:

Gerente	Empregado
Jim	Susan, Rob, Beth
Mary	Alice, John, Asim
Renee	Mike
Joe	Alan, Tim

**Tabela 18. Tabela que não está na 1FN**

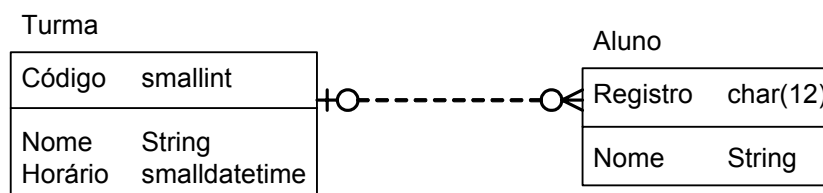
Gerente	Empregado
Jim	Susan
Jim	Rob
Jim	Beth
Mary	Alice
Mary	John
Mary	Asim
Renee	Mike
Joe	Alan
Joe	Tim

**Tabela 19. A mesma informação da tabela anterior normalizada em 1FN**

Turma

Código	smallint
Nome	String
Horário	smalldatetime
Alunos	Lista de Alunos

**Figura 72. Uma tabela não normalizada pode conter uma lista em um dos campos.**



**Figura 73. Normalizando a tabela turma, aparece a tabela aluno, que estava escondida em um atributo não atômico.**

Recomendamos enfaticamente o uso da primeira forma normal, ou seja, a não utilização de atributos multivalorados, no modelo conceitual. Dessa forma evitamos a ocultação de entidades e relacionamentos dentro de outras entidades, o que pode causar um grande desnivelamento entre o modelo conceitual e a real dificuldade de implementação.

**Um modelo de entidades e relacionamentos está na primeira forma normal se todos seus atributos são tipos atômicos.**

### VII.13.2 Algumas Anomalias Resolvidas pelas Formas Normais

Imaginemos a seguinte tabela, apenas na 1FN.

Título	Ano	Duração	Tipo	Estúdio	Estrela
Guerra nas Estrelas	1977	124	Cor	Fox	Carrie Fisher
Guerra nas Estrelas	1977	124	Cor	Fox	Carrie Fisher
Guerra nas Estrelas	1977	124	Cor	Fox	Carrie Fisher
Might Ducks	1991	104	Cor	Disney	Emilio Estevez
Wayne's World	1992	95	Cor	Paramount	Dana Carvey
Wayne's World	1992	95	Cor	Paramount	Mike Meyers

**Tabela 20. Uma tabela na 1FN, adaptada de [XXX Batini].**

Podemos verificar que essa tabela está na 1FN, porém ainda apresenta os seguintes problemas, que serão resolvidos pelas próximas duas formas normais:

- Redundância
  - Muita informação está repetida desnecessariamente em várias tuplas. Por exemplo, o fato que o Estúdio Fox fez Guerra nas Estrelas aparece 3 vezes.
- Atualização
  - Se precisarmos alterar um dado de Guerra nas Estrelas, sua duração, por exemplo, teremos que fazer isso várias vezes.
- Eliminação
  - Se precisarmos apagar um filme, por exemplo, “Might Ducks”, perdemos a informação que existe um estúdio chamado Disney.
- Inclusão

- Só podemos incluir um estúdio se tivermos também um filme para incluir.

### VII.13.3 Segunda Forma Normal (2FN)

**Um modelo de entidades e relacionamentos está na segunda forma normal quando, além de estar na primeira forma normal, não contém dependências parciais da chave, incluindo-se nessa chave atributos e relacionamentos identificadores.**

Uma dependência (funcional) parcial ocorre quando uma coluna depende apenas de parte de uma chave primária composta.<sup>43</sup>

Se A é dependente funcional de X, usamos a notação  $X \twoheadrightarrow A$ .

Uma tabela está 2FN se está na 1FN e cada uma das colunas não pertencentes à chave primária não for dependente parcialmente dessa chave.

Primeiro devemos notar que isso significa que uma tabela cuja chave seja formada por um único atributo está automaticamente na 2FN.

### VII.13.4 Terceira Forma Normal

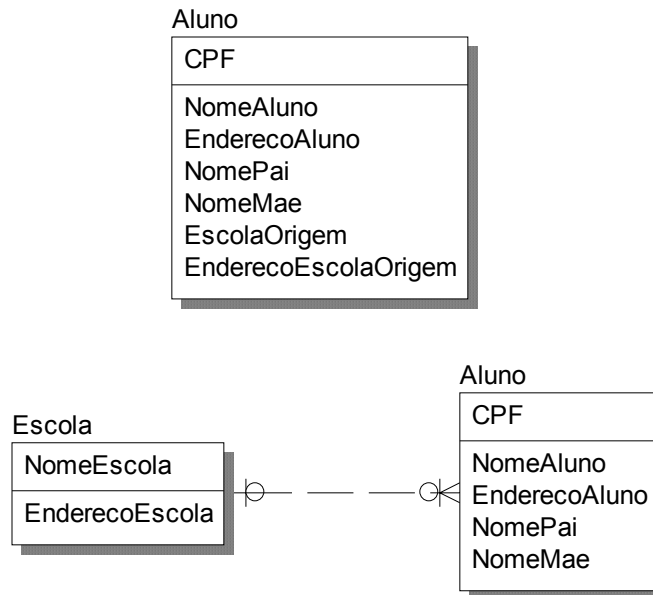
**Um modelo de entidades e relacionamentos está na terceira forma normal quando, além de estar na 2FN, não contém dependências transitivas.**

Uma dependência funcional transitiva ocorre quando um atributo, além de depender da chave primária da entidade, depende de outro atributo ou conjunto de outros atributos não identificadores da entidade.

Um exemplo de dependência transitiva pode ser encontrado em um sistema acadêmico universitário hipotético onde em uma entidade “aluno” fosse mantida a informação “escola de origem” e “endereço da escola de origem”. O endereço é dependente da escola, que depende do identificador do aluno. Assim, para normalizar, criamos a entidade escola, contendo nome e endereço (e outros campos necessário), eliminamos esses campos da entidade “aluno”, e finalmente criamos o relacionamento entre aluno e escola.

---

<sup>43</sup> Logo se a chave primária for simples, a tabela na 1FN está automaticamente na 2FN.



**Figura 74. Transformando a entidade aluno para a 3FN**

Uma entidade está na terceira forma normal se está na 2FN e se nenhum atributo não pertencente à chave fica determinado transitivamente por esta.

A segunda e terceira formas normais fazem com que cada atributo que não seja identificador forneça um fato sobre a entidade indicada pela chave e apenas pela chave [XXX Kent 83]. Normalizar corresponde a criar relacionamentos 1:N ou 1:1.

### VII.13.5 Outras formas normais

Existem ainda outras formas normais, que são praticamente abandonadas no dia a dia. A 4NF, a 5NF e a Boyce/Codd (BCNF). Todas elas lidam com fatos multivalorados, que devem corresponder a relacionamentos N:M ou N:1. De acordo com o modelo relacional temos as definições (e explicações) que se seguem.

**Uma tabela está na 4NF se, além de estar na 3NF, não contém dependências multivaloradas.**

Uma dependência funcional multivalorada ocorre quando um atributo de uma tabela implica na existência de uma lista de valores para outro atributo na mesma tabela (coluna dependente).

**Uma relação R está na BCNF se sempre que  $X \rightarrow A$  for verdade em R, e A não estiver contido em X, então X é uma chave candidata para R..**

Finalmente, a quinta forma normal trata da possibilidade de reconstruir uma informação a partir de um modelo composto de partes menores com chaves primárias diferentes. Se isso não for possível, e o modelo está na 4NF, então estará também na 5NF. Por exemplo, a tabela a seguir pode ser substituída por três outras que a seguem.

Vendedor	Empresa	Modelo
João	Ford	Caminhão
João	Ford	Automóvel
João	GM	Caminhão
João	GM	Automóvel
José	Ford	Automóvel

**Tabela 21. Relação que não está na 5NF**

Vendedor	Empresa	Empresa	Produto	Vendedor	Produto
João	Ford	Ford	Caminhão	João	Caminhão
João	GM	Ford	Automóvel	João	Automóvel
José	Ford	GM	Caminhão	José	Automóvel
		GM	Automóvel		

**Tabela 22. Três tabelas que substituem a tabela anterior**

### VII.13.6 Formas Normais e o Modelo E-R

Podemos ver que as formas normais podem ser facilmente aplicadas ao modelo de entidades e relacionamento simplesmente substituindo a palavra “tabela” pela palavra “entidade”

## VII.14 Outros termos

Entidade Associativa: transformação de um relacionamento em entidade para permitir relacioná-lo em outro relacionamento.

Entidade Fraca: entidade que não possui identificador por si mesma, dependendo de outra para sua existência. Não aprovamos essa classificação pelo peso do nome fraca. Muitas vezes as entidades fracas são as mais importantes em um sistema.

## VII.15 Verificando o Modelo

Alguns erros comuns:

- Estabelecimento de associações desnecessárias
- Usar uma entidade como atributo em outra entidade (e não fazer o relacionamento, que seria o correto).
- Modelar entidades únicas, principalmente uma que represente “o sistema”.
- Permitir redundâncias, como relacionamentos redundantes, principalmente os que podem ser resolvidos por uma navegação em uma sequência de relacionamentos. Também atributos redundantes, com a cópia de um atributo que já pode ser alcançado por meio de um relacionamento.
- Esquecimento do aspecto temporal, isto é, dos históricos das transações, como atributos que mudam de valor com o tempo ou relacionamentos que são alterados com o tempo. Nesse caso, pode ser necessário criar novas entidades.
- Entidades isoladas, em geral incorretas, apesar de não necessariamente incorretas. Muito mais raramente no modelo conceitual. Entidades isoladas



podem algumas vezes aparecer no modelo relacional para guardar constantes do sistema.

- Entidades únicas, entidades que possuem apenas uma instância estão geralmente erradas.
- Entidades sem atributos: geralmente erradas, a menos que estejam sendo usadas no lugar de um relacionamento  $n \times m$  (mesmo assim, não seriam necessárias na modelagem conceitual).

## **VII.16      Leitura Complementar**

Aconselhamos avidamente o livro de Paulo Cougo, Modelagem Conceitual de Dados [22] e o livro de Bertini, Ceri e Navathe, [23], Conceptual Modelling. O tratamento extensivo dado ao problema da modelagem conceitual de dados nesses dois livros pode ser de grande ajuda ao analista iniciante ou experiente.

Outro livro nacional importante é “Projeto de Banco de Dados”, de Carlos Alberto Heuser. Muito bom mesmo.

Para a obtenção de padrões de projeto, o livro de David Hay, que em português recebeu o nome de Princípios de Modelagem de Dados, mas em inglês se chama Data Patterns, ou seja, Padrões de Dados, é também excelente. Ele também leva o leitor a compreender seus padrões por meio de uma modelagem Top-down, o que pode ser utilizado como exemplos no aprendizado.

Para verificar como a modelagem de dados pode ser usada junto com a análise essencial, recomendamos o livro Complete System Analysis, de James e Suzanne Robertson [25], um dos melhores livros de análise no mercado, contando com um longo exemplo completo e exercícios. Dois outros autores brasileiros trataram desse assunto, Pompilho [28] e Barbieri [29].

Para uma abordagem mais prática, considerando desde o início alguns fatores tecnológicos, o livro de Ruble, Practical Analysis & Design for Cliente/Server and GUI Systems [30] é bastante útil.

## Capítulo VIII. Modelo Funcional Essencial

Modelagem Estruturada
Modelagem Essencial
Atividades Essenciais
Eventos Essenciais (Externo, Temporal, Não-Evento, Esperado, Não-esperado)
Memória Essencial

O Modelo Funcional tem como objetivo definir “o que”<sup>44</sup> o sistema deve fazer, ou seja, as funções que deve realizar para atender seus usuários.

Na Análise Essencial fazemos essa definição de acordo com um conjunto de princípios que nos permite escolher um modelo funcional específico entre vários possíveis, o Modelo Essencial.

### VIII.1 Perspectiva Histórica

A modelagem funcional de sistemas teve grande desenvolvimento a partir da criação das metodologias estruturadas de análise e projeto. Entre diferentes propostas, a Análise Estruturada foi a que teve maior repercussão, sendo que o Diagrama de Fluxo de Dados (DFD) se tornou uma ferramenta obrigatória em todos os cursos de análise de sistemas. Em um DFD, o sistema é descrito pela composição de quatro objetos básicos: agentes externos<sup>45</sup>, que interagem com o sistema, processos (ou funções), que caracterizam o sistema, memórias<sup>46</sup>, que contêm os dados necessários o sistema funcionar, e fluxos de dados entre esses objetos.

A Análise Estruturada, e outras técnicas equivalentes, se propunham a tratar das questões lógicas do desenvolvimento do sistema, em detrimento das questões físicas, que seriam tratadas na fase de projeto. O problema é que nenhuma técnica deixava claro quais eram essas questões, ou melhor, como diferenciar o “físico” do “lógico”. Além disso, ainda foram encontrados vários problemas na Análise Estruturada, entre eles: a dificuldade de manter o modelo atualizado e a possibilidade de várias pessoas fazerem modelos diferentes de um mesmo sistema.

O primeiro problema é atendido pelo uso de ferramentas CASE. É importante deixar claro que é impossível desenvolver uma verdadeira Análise Estruturada sem o uso

---

<sup>44</sup> Devemos entender que quando nos propomos a descobrir “o que” o sistema deve fazer, estamos simultaneamente evitando nos preocupar com “como” o sistema deve fazer, o que será feito nas fases mais avançadas do desenvolvimento.

<sup>45</sup> Originalmente chamadas de entidades externas. Esse nome não é utilizado neste texto para não confundir com o conceito de entidade (de dados). Atualmente, nos modelos orientados a objeto, é conhecido como ator.

<sup>46</sup> Originalmente chamadas de depósitos de dados.

de software para manter essa análise atualizada e correta. O segundo problema, porém, é muito mais grave, pois ele não trata da forma de uso, mas do método propriamente dito.

Em 1984, dezesseis anos antes de este texto ser escrito<sup>47</sup>, McMenamim e Palmer [12] conseguiram definir de forma clara um método de desenvolvimento que permite dividir, sem sombra de dúvidas, o que é essencial do que é encarnação. “Essencial” e “Encarnação” podem ser vistos como uma nova maneira de definir o que é “lógico” ou “físico”, mas vamos ficar com os nomes dados por Palmer e McMenamim para evitar toda carga cognitiva trazida pelos nomes “lógico” e “físico”. Eles também permitiram que descobríssemos, com perguntas simples, se um requisito do sistema é verdadeiro ou falso. Esse método é uma evolução da Análise Estruturada e é conhecido como Análise Essencial. Yourdon, mais tarde, vai denominar uma nova versão da Análise Estruturada, baseada na Análise Essencial, de Análise Estruturada Moderna. Existem pequenas diferenças na forma de Palmer e McMenamim e Yourdon. Esse texto é baseado em versões ainda mais modernas, principalmente nos textos de [25] e Ruble [30], mas ainda mantendo a “essência” da Análise Essencial.

## VIII.2 A Análise Essencial

O objetivo da Análise Essencial é descobrir, e documentar, todos os requisitos funcionais verdadeiros de um sistema e apenas esses requisitos. Para que isso seja possível, adotamos um conjunto de princípios e conceitos que nos permitem identificar esses requisitos dentro de toda a informação levantada durante um processo de análise.

O Método Essencial não é eficaz em qualquer tipo de projeto. Na verdade, estamos preocupados basicamente com sistemas de informação que sejam **sistemas interativos de respostas planejadas**. Esses sistemas funcionam sempre em resposta a algum evento fora do seu controle para o qual possamos definir uma resposta planejada. Deve ficar claro que não estamos interessados em eventos que exigem respostas *ad-hoc*, isto é, caso a caso. Usaremos o exemplo clássico do vendedor de passagens de avião: podemos fazer um sistema capaz de responder as perguntas típicas como “qual o preço da passagem para São Paulo” ou “Quando sai o próximo voo para Brasília”, porém não podemos considerar com esse método um sistema que responda a absolutamente todas as perguntas que um ser humano poderia responder, como “Qual foi o resultado do último jogo do América?”.

## VIII.3 Os princípios da Modelagem Essencial

Os princípios da modelagem essencial serão os nossos guias no processo de análise. O que acontece nesse processo é que várias vezes temos a opção de tomar dois ou mais caminhos. Na modelagem estruturada tradicional temos apenas vagas

---

<sup>47</sup> É impressionante que tanto tempo depois da criação e divulgação da Modelagem Essencial e da Análise Estruturada Moderna, tantas pessoas ainda trabalhem com as técnicas anteriores e, surpresos, declarem ser totalmente ineficazes. É o equivalente a esperar obter rendimento e velocidade de modelos atuais em automóveis de 30 anos atrás!

recomendações que nos auxiliam a escolher esse caminho, na modelagem essencial temos princípios específicos que nos orientam nessa escolha.

Os princípios da Análise Essencial são:

- O orçamento para a complexidade
- A neutralidade tecnológica
- A tecnologia interna perfeita
- O modelo essencial mínimo

A esses princípios somaremos um quinto, já apresentado, o princípio da ausência de surpresas, por acreditarmos que é perfeitamente condizente com os princípios essenciais.

### **VIII.3.1 O Orçamento para a Complexidade**

Esse princípio nos orienta a modelar um sistema que possamos compreender. Para isso devemos manipular a complexidade do modelo de forma a manter tanto o todo como cada uma de suas partes em um nível de complexidade compatível com a inteligência humana.

Para isso utilizamos técnicas de particionamento do sistema e o controle de características que aumentam a complexidade do modelo, entre elas:

- Controle do número de componentes de cada parte do modelo;
- Controle da complexidade interna de cada parte do modelo;
- Controle da complexidade da interface entre componentes;
- Manutenção da qualidade dos nomes utilizados no modelo, e
- Manutenção da qualidade da representação do modelo, por exemplo, quanto à clareza dos diagramas.

Um das técnicas mais citadas para controlar a complexidade é a de manter o número de componentes de cada modelo ou sub-modelo entre 5 e 9. Isso decorre de uma pesquisa [31] que determinou que o ser humano médio tem a capacidade de se concentrar em 7 elementos, com variação de  $\pm 2$ . O artigo é interessante, apesar de antigo.

### **VIII.3.2 A Neutralidade Tecnológica**

O princípio da neutralidade tecnológica exige que um modelo essencial não inclua em nenhuma de suas partes indícios da tecnologia de implementação [12].

Essa exigência, apesar de importante, é das mais quebradas pelos analistas. Isso acontece por que geralmente já sabemos qual a tecnologia em que vamos implementar o sistema. É importante manter a neutralidade não só para permitir uma análise mais objetiva do verdadeiro problema do usuário, mas também para aumentar a durabilidade dessa análise.

Alguns autores já criticam a neutralidade tecnológica, ou simplesmente “passam por cima” dessa questão, colocando preocupações de tecnologia já nessa fase. A questão tem relação com a necessidade de se assumir algumas premissas para obter soluções mais

eficientes. Em todo caso, na definição de sistemas de informação, a metáfora evento-atividade-memória fornecida pela análise essencial é na maioria dos casos suficiente para fornecer todo o arcabouço necessário para uma boa análise de requisitos. Devemos então não só seguir esse princípio, mas também usá-lo como ferramenta de conferência em cada um dos nossos passos, verificando se há ou não comprometimento com uma tecnologia específica, corrigindo cada ponto onde for encontrado esse comprometimento para uma especificação tecnologicamente neutra.

### **VIII.3.3 A Tecnologia Interna Perfeita**

O sistema deve ser modelado com a suposição que a tecnologia **interna** ao sistema é perfeita.

Por tecnologia interna perfeita queremos dizer que todos os recursos do sistema são ideais. A velocidade do sistema perfeito é infinita, significando que não há espera para conseguir um resultado. A memória de um sistema perfeito também não possui limitações, podendo guardar qualquer quantidade de informação por um período indeterminado, sem nenhum atraso no tempo de busca. O sistema perfeito nunca apresenta falhas ou necessidade de manutenção.

Devemos, porém, lembrar que não fazemos essa suposição sobre a tecnologia externa ao sistema, apenas sobre a tecnologia interna ao sistema. Além disso, essa suposição só é feita na fase de análise, sendo esquecida na fase de projeto, onde temas como velocidade, tamanho de memória e gerência de riscos passam a fazer parte de nossas preocupações, junto com outras características que, apesar de não fazer parte da suposição de um sistema perfeito, também são postergadas para a fase de projeto, como controle de acesso (segurança).

Na prática é interessante imaginar o sistema como um “gênio da lâmpada”, capaz de fazer qualquer coisa em tempo zero, se possuir a informação necessária.



**Figura 75. O sistema deve ser visto como um gênio “todo-poderoso”, capaz de realizar qualquer pedido imediatamente, se tiver a informação necessária.**

### **VIII.3.4 O Modelo Essencial Mínimo**

Os princípios anteriores vão definir claramente a nossa forma de trabalho, porém muitas vezes serão inúteis para ajudar a escolher qual o é o modelo essencial entre dois

modelos possíveis. Precisamos, porém, para garantir que nosso método tem uma resposta única, ter uma forma de escolher, entre dois modelos, qual o modelo essencial, mesmo que eles cumpram todos os requisitos anteriores.

O princípio do modelo essencial mínimo exige que, entre dois modelos possivelmente essenciais, a definição menos complicada é o modelo essencial. Assim possuímos uma forma clara de escolha.

### VIII.3.5 O Princípio da Ausência de Surpresas

Esse princípio não faz parte da análise essencial, mas foi proposto pelo GUIDE MVS group [19] para auxiliar a análise de negócios. Ele é essencialmente auto-explanatório. O princípio é conservado quando o produto:

- Faz o que o usuário espera
- Responde de forma previsível e consistente aos estímulos.
- Comporta-se de forma limitada a sua razão de existência.
- É regular e mínimo, apesar de completo.
- Quando falha, o faz de forma graciosa e recuperável.
- Quando a dificuldade de utilizá-lo ou modificá-lo é compatível com a dificuldade da área de aplicação.

Essa é uma importante filosofia e que, junto com os princípios da modelagem essencial apresentados mais tarde, servirá de guia para todo o nosso processo de desenvolvimento. Cada vez que tivermos que tomar uma decisão relacionada ao sistema, será nesses princípios que buscaremos a nossa resposta.

## VIII.4 A Essência

**A essência do sistema é tudo que precisaria ser incluído no sistema para que o mesmo funcionasse quando implementado em um ambiente de tecnologia perfeita.** Isso compreende velocidade infinita no processador, tamanho infinito de memória, custo nulo para todas as operações, infalibilidade, etc.

Esse conceito será de grande valia quando estivermos analisando se um requisito é verdadeiro ou falso. A primeira pergunta que faremos é “Esse requisito seria necessário se tivéssemos um computador perfeito?” Se a resposta for não, o requisito é falso.

Um sistema de tecnologia perfeita é como um gênio da lâmpada. Se quisermos que o gênio da lâmpada pague nossos funcionários, ainda teremos de alguma forma dizer quem são nossos funcionários, logo “cadastrar funcionários” é um requisito verdadeiro de um sistema de pagamentos de funcionários.

Sistemas essenciais possuem dois tipos de componentes: **atividades** e **memórias**. Cada tarefa que o sistema de tecnologia perfeita tem que realizar para cumprir a finalidade do sistema é uma **atividade essencial**. Essas atividades existem em duas formas: as **atividades fundamentais** e as **atividades custodiais**. As atividades essenciais, para poderem executar suas tarefas, precisam guardar informação. Essa

informação é guardada em memórias. Da forma que tratamos a análise, a memória do sistema, vista como um todo, é o modelo conceitual de dados, discutido no Capítulo VII. Cada atividade essencial, porém, pode ter apenas uma visão parcial dessa memória, de acordo com suas necessidades.

As **atividades fundamentais** são aquelas que justificam a existência do sistema [12]. Certamente, ao comprar um sistema, precisamos fundamentalmente das saídas que ele nos disponibilizará. Assim, atividades fundamentais precisam incluir alguma saída para agentes externos. Suponha que você deseja comprar um sistema de controle de ponto para sua empresa. Várias atividades são necessárias ou possíveis nesse sistema, porém poucas são fundamentais. A atividade fundamental é, certamente, informar a presença de cada funcionário, pois é para saber disso que você comprou o sistema. As atividades fundamentais necessitam de dados para funcionar, que podem ser fornecidos diretamente por um **agente externo**, um ser humano ou outro sistema que faz parte do ambiente, interagindo com o sistema, ou estar guardada em uma **memória**.

As **atividades custodiais** se destinam a criar e manter as memórias necessárias para o funcionamento das atividades fundamentais. Um sistema, mesmo de tecnologia perfeita, não pode criar dados sozinho. Em ambos os exemplos, da folha de pagamento ou da lista de presença, precisamos saber quem são nossos funcionários. Manter essa lista de funcionários é uma atividade custodial. Atividades custodiais, fique bem claro, são essenciais ao funcionamento do sistema. Acontece que, enquanto o usuário conhece a grande maioria das atividades fundamentais que precisa, muitas atividades custodiais ficam de fora de sua lista. Assim, ao analisar um sistema, devemos estar sempre alerta para atividades custodiais necessárias para manter nossas memórias em ordem.

Algumas atividades são custodiais e fundamentais simultaneamente, sendo então chamadas de compostas ou mistas.

É possível ter uma visão menos funcional e mais pragmática, que perde muito da qualidade filosófica, mas ganha em simplicidade: atividades fundamentais têm uma saída para o mundo externo, enquanto atividades custodiais alteram memórias. Isso nos chama atenção que não existem atividades que não sejam fundamentais ou custodiais, isso é, uma atividade deve pelo menos escrever em uma memória ou fazer um relatório.

	Abordagem	
	Filosófica	Pragmática
Fundamental	Requisito original do cliente	Emitir informação
Custodial	Necessário para o sistema funcionar	Recebe informação

**Tabela 23. Tipos de Atividades Essenciais e como classificá-las**

Uma atividade essencial é posta em funcionamento por meio de um estímulo, isto é, um fluxo de dados ou um comando que provém de um **agente externo**. A esse estímulo damos o nome de **evento essencial**.

## VIII.5 Agentes Externos

Representamos as pessoas, departamentos, empresas, máquinas ou sistemas que interagem com o sistema sendo analisado, enviando ou recebendo dados, por meio de **agente externos**, indicado no DFD por meio de um quadrado<sup>48</sup>. Agentes externos, também são chamados de **entidades externas**<sup>49</sup> ou **terminadores**.

Normalmente agentes externos representam pessoas, pois a maioria das funções de um sistema de informação se dá por meio da interação com uma ou mais pessoas. Eventualmente um sistema de informação interage com outro sistema, recebendo ou enviando dados, ou ainda com sensor ou com um atuador.

Os agentes externos controlam o funcionamento do sistema. São eles que detêm o poder de iniciar as atividades essenciais, ao enviar um estímulo ao sistema. São eles também que recebem todas as respostas emitidas pelo sistema.

O modelo essencial está interessado nos agentes externos que detêm realmente o controle dos estímulos ou que realmente recebem a informação. Usuários do sistema implementado, como digitadores, não são modelados nos DFDs da análise essencial. Usuários do sistema que apenas servem como interlocutores dos verdadeiros agentes externos são considerados **transportadores** de dados. Ao documentar um evento devemos documentar a existência de transportadores, como veremos no dicionário de eventos<sup>50</sup>.

Essa distinção entre agentes externos e transportadores é algumas vezes muito difícil. Devemos entender que a verdadeira essência de um sistema está relacionada com sua função no negócio em que ele está inserido. Os usuários do sistema não são necessariamente os agentes externos que interagem com o sistema. Devemos considerar como agentes externos aquelas pessoas ou artefatos tecnológicos que detêm o poder de iniciar o evento. Por exemplo, normalmente, em um sistema de vendas, o agente externo que inicia o evento é o comprador e não o vendedor que está usando o sistema. Por isso diferenciamos, na descrição completa do evento, o **iniciador**, que é o agente externo que inicia o sistema, do **transportador**, que é o usuário do sistema responsável por introduzir os dados fornecidos pelo iniciador.

Um bom teste para verificar se o agente externo é o verdadeiro iniciador do evento ou apenas um transportador, no contexto de um negócio, é perguntar se ele pode realizar, por sua vontade, aquele evento. Vemos que, no caso de um vendedor, ele não pode vender sem receber um pedido de um comprador (a não ser no caso específico onde ele estará pagando a compra, sendo um comprador ele mesmo). Logo, o vendedor **não** é o agente externo iniciador (podendo, porém, receber alguma saída do sistema para ele destinada). Na verdade, de uma forma mais essencial, porém pouco reconhecida, o

---

<sup>48</sup> Em alguns formatos de DFD, as entidades externas não possuem símbolo ou até mesmo não são indicadas.

<sup>49</sup> Não confundir com as entidades do modelo de entidades e relacionamento.

<sup>50</sup> Modernamente, na técnica de “Casos de Uso”, considera-se a possibilidade de dois modelos, o do negócio e o do sistema. No modelo do sistema são utilizados os usuários reais.



vendedor é parte do sistema, pois é apenas uma implementação da interface com o comprador<sup>51</sup>.

Ao fazer a análise não estamos preocupados com os motivos dos agentes externos, ou em modificar suas ações, ou com o que eles fazem com os dados que obtém do sistema. Por isso eles não são estudados, definindo a fronteira do sistema e os limites do trabalho de análise.

Normalmente agentes externos são descritos por nomes que representam classes ou tipos de usuário dentro de um negócio. Assim, um sistema para uma loja que apresenta os usuários “gerente”, “vendedor” e “cliente”, terá agentes externos com esses nomes. Nunca usamos o nome pessoal de um usuário, mas sim seu papel ao utilizar o sistema ou seu cargo na empresa.

Outro tipo comum de agente externo é aquele que representa uma instituição ou departamento externo ao ambiente de uso do sistema. Nesse caso o agente externo é nomeado com o nome desse departamento ou da instituição (ou ainda, do tipo da instituição).

Finalmente, existem os agentes externos que são máquinas, como sensores, ou sistemas, sendo nomeados diretamente com o nome dos mesmos.

É importante perceber que muitos agentes devem ser representados não só fora do sistema, mas também em sua memória. Isso acontece quando o sistema deve guardar dados sobre um agente externo, de forma a poder reconhecer ou referenciar um agente externo, por exemplo, enviando uma conta para o agente externo. Nesse caso, haverá pelo menos uma entidade no modelo de dados representando no total ou parcialmente o agente externo. Isso não se aplica, porém, no caso da segurança, pois essa não é tratada na análise essencial.

Como exemplo podemos ver o caso de um sistema acadêmico, onde os alunos são agentes externos, pois solicitam boletins, e são entidades, pois devemos guardar dados sobre os alunos. Já os funcionários da secretaria da escola são agentes externos, pois podem pedir alguns documentos específicos guardados no sistema, mas o sistema não precisa saber quem são<sup>52</sup>.

## **VIII.6           Eventos Essenciais**

Na modelagem essencial tudo ocorre em função de eventos. Isso acontece porque imaginamos que o sistema possui dois estados: em atividade ou esperando um evento. É o acontecimento do evento que faz com que o sistema entre em funcionamento e então realize todas as tarefas necessárias para atender aquele evento, ou seja, a atividade essencial correspondente ao evento.

---

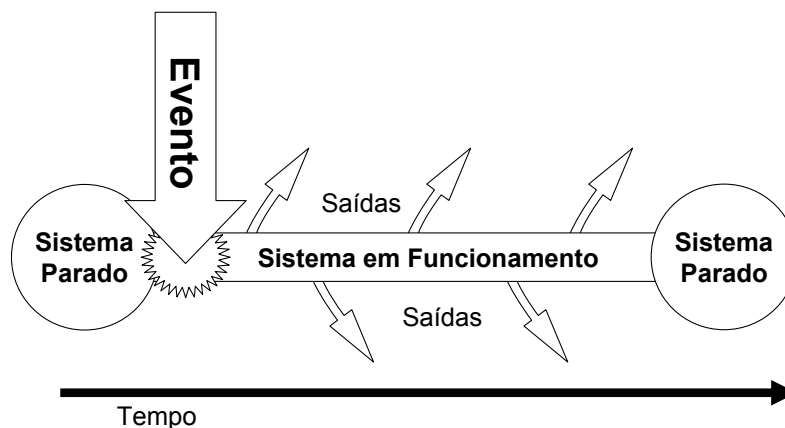
<sup>51</sup> Atualmente é fácil entender como o vendedor pode ser substituído por uma interface Web e o sistema manter sua essência.

<sup>52</sup> A modelagem essencial não está preocupada com questões como segurança, backup, etc.

É importante notar que o sistema só tem a oportunidade de funcionar quando acontece um evento. Esses eventos, por definição, **não são controláveis pelo sistema**, ou seja: o sistema é incapaz de gerar um evento<sup>53</sup>.

Cada atividade é iniciada com um único **evento**, que define um **estímulo**, e compreende **todo o conjunto de ações efetuado pelo sistema para executar a atividade**, ou seja, a **resposta planejada**. A atividade relativa a um evento compreende toda a cadeia de ações causada por esse evento, até que o sistema tenha que parar porque todos os fluxos de dados atingiram seus objetivos (agentes ou memórias). Como apenas um evento inicia a atividade, então **apenas um único agente externo pode enviar informações para uma atividade**<sup>54</sup>. Isso é uma regra importante da análise essencial, pois indica como o sistema será particionado.

O evento funciona como um gatilho, disparando uma reação em cadeia, que para apenas pela impossibilidade de realizar qualquer outra atividade. Nessa reação em cadeia não devemos nos preocupar no modo como as ações ocorrem no sistema existente, na encarnação atual, pois elas podem sofrer interrupções espúrias, que dividem um evento entre vários processadores.



**Figura 76. O esquema mostra como o sistema reage a um evento. Um Sistema parado só pode começar a funcionar ao receber um evento. A partir desse evento, realiza uma série de processamentos que podem envolver saídas para o mundo externo, até parar novamente. Não podem existir outros eventos ou entradas do mundo externo durante o funcionamento, porém é possível consultar as memórias, que são internas ao sistema.**

Muitas vezes o iniciante na análise essencial imagina que será travado um diálogo com o usuário durante a atividade. Esse diálogo interno a uma atividade não existe na análise essencial. O estímulo relacionado ao evento, isto é, o fluxo de dados que parte do agente externo em direção à atividade, possui toda a informação necessária para realizar a atividade, incluindo partes opcionais. Caso o diálogo seja realmente necessário para o

<sup>53</sup> Não existem, logo, eventos essenciais internos ao sistema.

<sup>54</sup> Yourdon não obedece essa regra, permitindo que algumas entidades externas forneçam informações sob demanda de uma atividade essencial.

funcionamento do sistema, então temos na verdade dois, ou mais, eventos e suas respectivas atividades. Isso acontece porque uma atividade, por definição, não pode ficar “esperando” por uma entrada de dados. A regra que usamos é: **se o sistema para, só pode voltar a funcionar com um evento**. O mesmo raciocínio se aplica quando falamos de vários agentes externos.

O evento funciona como um gatilho, disparando uma reação em cadeia, que para apenas pela impossibilidade de realizar qualquer outra atividade. Nessa reação em cadeia não devemos nos preocupar com o modo como as ações ocorrem no sistema existente, na encarnação atual, pois elas podem sofrer interrupções devidas à implementação, que dividem um evento entre vários processadores.

Segundo a análise essencial, **não existem eventos internos ao sistema**, isto é, não dizemos que um processo do sistema se inicia por causa de um evento causado por outro processo. A análise essencial parte do conceito que **eventos iniciam atividades** essenciais e que essas atividades são executadas até que todas as respostas necessárias sejam geradas. O sistema então para e fica esperando pelo acontecimento de outro evento de essencial para voltar a funcionar. Devemos ter bastante atenção à regra que uma atividade contém a resposta completa para um evento (e apenas para um único evento), pois é ela que vai definir o particionamento do sistema sendo modelado.

Outra característica da análise essencial é que os eventos são vistos por dentro do sistema. Assim eles são descritos tendo como sujeito o agente externo que os iniciam, como em “Aluno solicita matrícula”<sup>55</sup>.

Resumindo, os eventos acontecem fora do sistema, correspondem a um estímulo que cruza a fronteira do sistema de fora para dentro e são vistos e descritos na perspectiva de um ser imaginário que habita sistema.

Também é importante frisar que **atividades essenciais não se comunicam diretamente**, isto é, não se comunicam por meio de fluxos de dados. Toda comunicação entre atividades essenciais é feita por meio do uso da memória do sistema

### VIII.6.1 Tipos de Eventos

Cada **evento** pode ser **externo** ou **temporal**. Um evento é **externo** quando parte do ambiente para dentro do sistema. Um comando ou um pedido do usuário, por exemplo. Um evento é **temporal** quando é provocado por uma mudança no tempo, como um alarme de relógio ou uma data no calendário.

Originalmente só tratávamos esses dois tipos de eventos (externos e temporais). Com o tempo, porém, fomos aprendendo a trabalhar e a classificá-los de forma a melhorar nosso trabalho.

Eventos temporais podem ser **absolutos** ou **relativos**. Um evento é relativo quando é definido em função do decorrer de um prazo depois do acontecimento de outro evento. Eventos absolutos ocorrem em função unicamente do calendário e do relógio<sup>56</sup>.

---

<sup>55</sup> A frase pode ser lida como significando “Aluno solicita matrícula ao sistema”.

<sup>56</sup> eventos temporais podem ser implementados como eventos internos, mas não necessariamente. Isso costuma causar confusão, pois temos o hábito imediato de pensar na implementação e de associar a

Um evento temporal ocorre porque o sistema tem um contrato para entregar informação a um agente em um momento específico [25].

Eventos externos podem ser: **esperados** ou **não-esperados**. Um evento é **esperado** quando sabemos que ele vai acontecer em um instante específico, ou que tem um limite de prazo para acontecer. Ele pode também ser chamado de **evento agendado**. Um evento é **não-esperado** quando não podemos determinar momento ou limite para seu acontecimento. Ele pode também ser chamado de **evento não agendado**. Os nomes “esperado” e “não-esperado” podem causar alguma confusão. O sistema, na realidade, espera que ambos os tipos de eventos ocorram. Porém, alguns têm um prazo ou data para ocorrer. Por isso podemos usar, com mais precisão, o termo “agendado”.

#### VIII.6.1.1 Não-Eventos

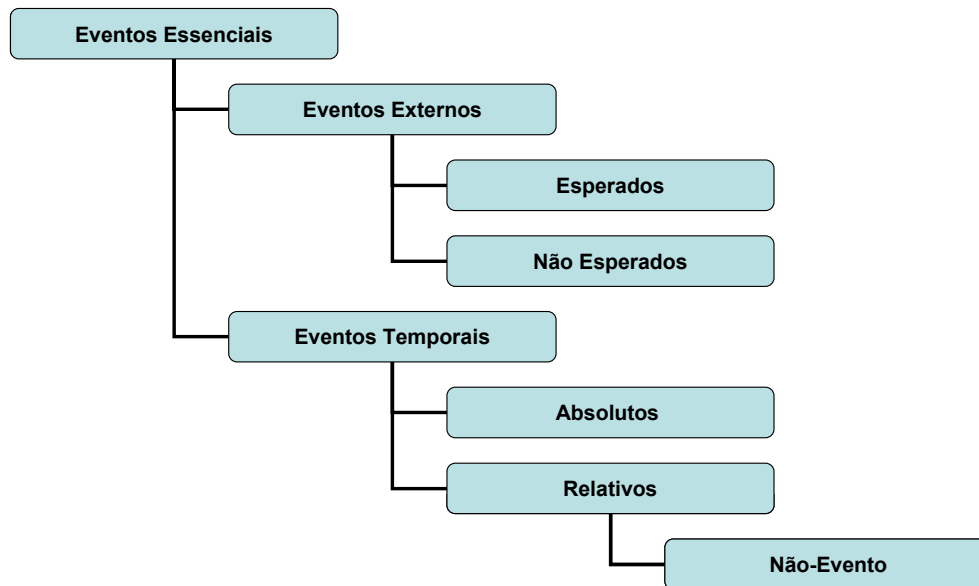
Eventos esperados formam uma categoria importante, pois sabemos que no mundo real, quando um evento esperado não acontece (o pagamento de uma conta, por exemplo), pode ser necessário tomar uma atitude específica. Dizemos então que eventos esperados podem necessitar que sejam definidos **não-eventos**. Esse é o nome que damos para eventos que acontecem em função de outro não ter acontecido, possivelmente a partir de um prazo, como na sentença: “uma semana após esgotar o prazo de pagamento, enviar lembrete”. Novamente, o nome “não-evento” pode causar confusão. Um não-evento é um evento, em especial, é um evento temporal relativo.

Um só evento esperado pode necessitar de vários não eventos, que ocorrem normalmente em prazos distintos. Assim, se temos um evento esperado “cliente paga conta, até o dia 30 do mês”, podemos ter vários não eventos para os prazos de 1 dia, 1 semana, 1 mês, 3 meses e 1 ano, por exemplo.

<p><b>Um não-evento é um evento temporal relativo que deve acontecer se um evento esperado não ocorre, possivelmente considerando um prazo.</b></p>
---

---

proibição de eventos internos na análise como a proibição de eventos internos na implementação. Exigimos apenas, na análise essencial, que não existam *eventos essenciais* internos. A implementação não está limitada por essa regra.



**Figura 77. Tipos de eventos essenciais e seus subtipos.**

### **VIII.6.2 Um Exemplo Esclarecedor**

Vamos tentar entender melhor a motivação para termos uma regra tão rígida sobre um evento só receber dado de um agente externo.

Imagine um sistema para uma loja de automóveis onde o mecânico, um agente externo, faz um pedido de peça. Esse pedido é anotado e repassado, pelo sistema, para outro sistema, o sistema de estoque, que é outro agente externo, que responderá se a peça está disponível ou não. Quando o sistema de estoque responder, então avisaremos o mecânico da disponibilidade da peça.

Segundo nossa regra, temos os seguintes eventos:

- Mecânico solicita peça
  - Nesse evento é enviado um pedido de peça para o Sistema de Estoque
- Sistema de Estoque avisa disponibilidade de peça
  - Nesse evento o mecânico recebe um aviso da disponibilidade

Já segundo alguns autores, fazendo uma interpretação simplificada, mas, porém não essencial, apenas o primeiro evento é necessário, pois nesse evento o Sistema de Estoque é consultado.

Agora vamos nos lembrar de nossos princípios da análise essencial. Não podemos considerar a tecnologia do sistema. Também, seguindo nossa definição de evento, não podemos controlar quando o mundo externo faz alguma coisa.

Dessa forma, não podemos controlar quando o Sistema de Estoque vai responder a pergunta que fizemos. Isso é, devemos esperar por um evento de resposta.

Quanto à tecnologia, podemos usar várias tecnologias como teste da “essencialidade” de uma solução. A solução essencial deve permitir que o pedido seja

enviado ao estoque com qualquer tecnologia, por exemplo, cartas, telefone, pombo-correio ou computadores ligados na Internet. Porém, fica claro ao analisar um pedido feito por cartas que o sistema ficará parado esperando a resposta, logo a resposta é outro evento, que faz com que o sistema volte a funcionar.

### VIII.6.3 Habilitando Eventos

Da mesma forma que um evento esperado pode precisar de um não-evento, é interessante também perceber que muitos eventos esperados só podem acontecer caso outro evento aconteça antes. Por exemplo, em um sistema de controle de prestações, “Cliente paga parcela de prestação” é normalmente um evento esperado, porém ele só pode acontecer depois que “Cliente solicita crédito” ocorre. Dizemos que um evento habilita o outro. Isso equivale a uma mudança no estado do sistema que deve ser anotada em alguma memória.

Em muitos sistemas podemos ver eventos sendo habilitados por outros eventos.

Novamente, devemos tomar cuidado para não confundir o fato de um evento poder habilitar outro evento com as respostas de um evento. Uma resposta é algo que essencialmente pode ser realizada em função do acontecimento do evento. Um evento habilitado necessita de um outro estímulo para acontecer, porém ele é impossível sem que algum evento anterior o “libere” para funcionar. Um evento desse tipo altera o estado do sistema.

Não há uma notação especial para eventos que habilitam outros eventos, apenas a consulta à especificação da atividade ou ao dicionário de eventos pode dar essa informação.

Algumas habilitações podem ser anotadas em uma memória. Porém, cuidado, pois pode ser que a memória não tenha sido desenvolvida no modelo de dados.

## VIII.7 Descrevendo Eventos Essenciais

Um evento externo sempre é caracterizado pela existência de agente externo, que é a pessoa ou um outro sistema que faz com que o evento aconteça, enviando para o sistema o estímulo correspondente. Assim, na nossa descrição de um evento externo **sempre devemos colocar o nome do agente externo que o causa**.

No caso de eventos temporais, devemos colocar o fato que faz o evento acontecer. Eventos temporais **não possuem um agente externo que forneça o estímulo**.

Alguns estímulos são bastante complicados, contendo dezenas ou centenas de dados, outros são bastante simples, contendo apenas um comando ou solicitação ao sistema. Eventos cujo estímulo é apenas um comando de execução podem ser chamados **eventos de controle**<sup>57</sup>.

A sintaxe para definir eventos externos é:

---

<sup>57</sup> E possuem, em algumas variações de DFD, uma notação específica, utilizando uma seta tracejada em vez de sólida.

*<agente externo - sujeito> <verbo no presente> <objeto direto>*

Como em: “Cliente solicita lista de produtos”.

Opcionalmente, no caso de um evento esperado, pode ser usado o seguinte padrão:

*<agente externo - sujeito> <verbo no presente> <objeto direto> , <prazo>*

Onde *prazo* pode ser absoluto ou relativo a outro evento ou resposta de evento. Como em: “Fornecedor envia produtos pedidos, até 30 dias depois do pedido”.

A sintaxe para definir eventos temporais é:

*<condição temporal>, é (hora|dia|etc.) de <verbo no infinitivo> <objeto>*

ou simplesmente

*<condição temporal>, <verbo no infinitivo> <objeto>*

Como em: “Todo dia 30, é dia enviar declaração de vendas” ou “dia 30, enviar declaração de vendas”.

A sintaxe para um não evento é:

*<condição de não acontecimento de um evento>, <prazo ou condição temporal>,  
<verbo no infinitivo>, <objeto>*

Como em: “Fornecedor não enviou produtos pedidos, depois de 30 dias do pedido, avisar comprador”.

O objeto da oração é normalmente um objeto direto que, de alguma forma, representa uma informação tratada pelo sistema, e possivelmente também um objeto indireto indicando para quem ou para onde a informação é enviada.

Vejamos alguns exemplos comuns, descritos de forma correta:

- Cliente envia pedido de compra.
- Fornecedor entrega mercadoria
- Fornecedor entrega mercadoria, até 10 dias depois do pedido.
- Vendedor solicita mercadoria.
- Filial envia vendas diárias.
- Cliente aluga fita.
- Ao final do mês, imprimir folha de pagamento.
- Ao fim do dia, imprimir resumo de vendas.
- Gerente solicita relatório de produção.
- Caso o cliente não pague a conta, 20 dias depois, invocar departamento jurídico.
- Caso o aluno não apresente o boletim assinado, 10 dias depois, enviar aviso aos pais.

Vejam agora alguns eventos descritos de **forma incorreta**:

- Enviar pedido (sem agente externo ou indicação de tempo)
- Gerente imprime relatório (quem imprime é o sistema, o gerente solicita, além disso, “relatório” é um termo muito vago).

## VIII.8 Levantando os Eventos Essenciais

A primeira e mais importante tarefa da modelagem funcional é o levantamento dos eventos essenciais. Isso feito, nós teremos uma idéia bastante clara do tamanho do sistema.

Geralmente nossa estratégia deve ser: levantar uma lista inicial de eventos e refinar essa lista enquanto analisa a resposta para cada evento. Duas são as formas básicas de levantamento de requisitos funcionais: entrevistas e reuniões JAD.

Várias são as formas de descrever uma lista de eventos. Uma maneira é partir dos eventos fundamentais e depois listar todos os eventos necessários para que os eventos fundamentais funcionem. Outra maneira é seguir uma abordagem seqüencial, partindo dos eventos que geram dados para chegar aos eventos que geram relatórios no final.

Alguns sistemas são muito seqüenciais, facilitando claramente a segunda abordagem. Nesse caso é importante numerar os eventos no tempo, para garantir que a seqüência ficou bem clara e nenhum passo foi perdido.

Uma lista de eventos tem a seguinte aparência:

- |  |
|--|
| <ol style="list-style-type: none"><li>1. Gerente cadastra distribuidora</li><li>2. Gerente cadastra livro</li><li>3. Cliente pede livros</li><li>4. Sexta-feira, é hora de fazer requisições de livros para as distribuidoras.</li><li>5. Distribuidora entrega livros</li><li>6. Gerente solicita relatório de vendas</li><li>7. Gerente solicita relatório de livros em atraso</li><li>8. Se a distribuidora não entregar os livros pedidos, depois de 15 dias, cancelar pedido.</li></ol> |
|--|

**Figura 78. Exemplos de uma lista de eventos**

Para um evento ser aceito como um evento essencial deve passar pelos seguintes testes:

- Deve ocorrer em um momento específico no tempo
- Deve ocorrer no ambiente e não dentro do sistema
- Sua ocorrência deve estar sob o controle do ambiente e não do sistema



### VIII.8.1 Classificando os Eventos

Apesar de termos descrito os eventos como podendo ser de vários tipos, não é extremamente necessário identificar todos os eventos. Porém, fazer isso traz a vantagem de podermos testar a forma como o evento está descrito. Aqui estão algumas perguntas de teste:

- Eventos externos:
  - São esperados ou agendados?
    - Se esperados, possuem um não-evento correspondente?
  - São não-esperados ou não agendados?
  - São uma solicitação ou possuem dados?
- Eventos temporais:
  - Só ocorrem dessa forma?
  - São realmente temporais ou podem ser calculados antes (sendo, nesse caso, uma saída do sistema)?
  - São Relativos?
    - São não-eventos?
    - Qual é o evento original?
  - O evento original existe sempre?

Opcionalmente, podemos construir uma tabela de classificação de eventos, como a apresentada a seguir. Todo evento deve ser facilmente classificado. A dificuldade de classificar um evento demonstra que ele não foi compreendido e indica que ele pode não estar correto, tanto na sua interpretação ou na sua descrição, ou até que seja um requisito falso.

Além disso, a tabela permite que verifiquemos se todos os eventos esperados possuem um ou mais não eventos correspondentes.

Número	Evento	Classificação				
		Externo		Temporal		
		Esperado	Não- Esperado	Relativo	Absoluto	Não- evento (p/ evento)
1	Gerente cadastra distribuidora		✓			
2	Gerente cadastra livro		✓			
3	Cliente pede livros		✓			
4	Sexta-feira, é hora de fazer requisições de livros para as distribuidoras				✓	
5	Distribuidora entrega livros	✓				
6	Gerente solicita relatório de vendas		✓			
7	Gerente solicita relatório de livros em atraso		✓			
8	A distribuidora não entregou...					✓ (5)

**Tabela 24. Exemplo de uma tabela de classificação de eventos**

### VIII.8.2 Encontrando Eventos

Os principais eventos são os pedidos que são feitos ao sistema. Eles normalmente podem ser encontrados em formulários, memorandos, documentos que chegam e observando o atendimento que os usuários do sistema prestam.

Relatórios devem ser gerados por algum evento. Eles, porém, são as respostas aos eventos e não os eventos propriamente ditos. Todo evento externo esperado deve precisar de um e pode precisar de mais não eventos.

No mundo real encontramos ainda outras características que indicam novos eventos:

- Pedidos normalmente podem ser cancelados.
- O que é enviado pode retornar, exigindo uma ação específica.
- Documentos podem ser perdidos e segundas vias podem ser necessárias
- Fiscais (ICMS, ISS, Trabalho,...) podem aparecer e solicitar relatórios (que podem ser obrigatórios em um sistema).
- Processos que ocorrem em uma ordem podem ter que ser “acelerados” para atender um cliente preferencial.
- Pagamentos podem ser feitos com o valor errado, para menos ou para mais, exigindo emissão de novas cobranças ou de créditos.

### VIII.8.3 Simplificando Eventos

Segundo a análise essencial original, as operações de incluir, eliminar e alterar uma memória exigiriam pelo menos três eventos distintos, como em:

- Proprietário cadastra produto
- Proprietário altera produto
- Proprietário apaga produto

Isso, porém, pode não representar a verdade e ser muito complicado em alguns sistemas. Na vida real é fácil termos um sistema com 30 a 40 entidades. Isso exigiria no mínimo 90 eventos para cumprir as necessidades de manter a memória. Não fazemos isso!

Em primeiro lugar, não criamos atividades custodiais que não são necessárias. Isso acontece quando a memória já é gerenciada em uma atividade fundamental. Em segundo lugar, quando uma memória necessita de uma funcionalidade que permita tratar esses três casos, podemos utilizar uma notação mais simples:

- Proprietário mantém produtos

## VIII.9 As Respostas aos Eventos

Para cada evento o sistema deve executar uma resposta. Essa resposta é representada pela **atividade essencial** correspondente ao evento e produz dois tipos de resultados: **alterações no estado do sistema** e **emissão de informação** para o ambiente (alterações do estado do ambiente).

Uma alteração no estado do sistema significa que uma ou mais memórias foram alteradas. Nisso incluímos a criação de registros, a mudança de valores dentro de registros e a eliminação de registros.

Como emissão de informação temos várias formas de emissão de relatórios, *feedback* para os agentes externos e comandos para atuadores externos.

Cada evento pode exigir uma ou mais repostas, obrigatórias ou opcionais, do sistema. O processamento de todas essas repostas, juntas é a atividade essencial.

Algumas respostas a eventos são óbvias a partir do nome do evento, como por exemplo, em “Gerente solicita relatório de vendas”. Uma resposta óbvia é “relatório de vendas”. Porém, poderíamos, em um caso real, ter outras respostas, como por exemplo, “aviso ao diretor”.

Logo após levantar a lista de eventos é importante levantar a lista de respostas para cada evento. Apesar de não ser importante classificar cada resposta, é recomendável que o analista saiba se a resposta é opcional ou obrigatória, e, caso seja opcional, estar preparado para fornecer, mais tarde, as regras que indicam sua necessidade.

### VIII.9.1 Confundindo eventos e respostas

É muito comum também que o iniciante confunda uma resposta a um evento com um evento. Para isso podemos usar uma tática de verificação: perguntar por que um

evento acontece. Se a resposta for “Esse evento acontece porque o usuário X enviou um dado” ou “porque se passaram X dias”, estamos em um bom caminho e provavelmente temos um evento. Porém, se respondermos com algo do tipo “Esse evento acontece porque o sistema...” ou “Esse evento acontece quando é verdade que...” então estamos em um mau caminho, pois não existem eventos gerados pelo sistema para o sistema.

Outra coisa importante é verificar se existe algum motivo para o sistema começar a funcionar sozinho (o evento). Se não existe, provavelmente escolhemos como evento algo que é resposta para outro evento.

Um exemplo muito comum acontece em “casos especiais”. Vamos supor que temos um sistema que deve produzir um relatório a cada 100 vendas informadas por cada vendedor. A resposta correta é ter um evento “Vendedor informa venda”, com uma saída (além das outras necessárias) “Relatório de Vendas”. Geralmente analistas iniciantes “inventam” um evento especial “Vendedor faz centésima venda”.

Vamos tentar aplicar nossos princípios. Temos um que diz “Não surpreenda o usuário”. Seria uma surpresa para o usuário ter que usar uma tela diferente e ele mesmo controlar sua centésima venda. Muito mais natural seria que o sistema controlasse isso. Outro princípio diz: tecnologia interna perfeita. Ora, se a tecnologia interna é perfeita não nos custa nada fazer toda a lógica necessária para esse controle, também um bom sinal. Finalmente, o princípio da solução mínima nos indica que é melhor ter apenas um evento do que dois.

Esse raciocínio pode ser aplicado em todos os casos em que temos uma **saída opcional**. É interessante notar que, em um DFD, as saídas e entradas em um processo não são obrigatórias, mas opcionais. É a lógica do processo que vai decidir se elas existem ou não. Assim, podemos incluir em um evento todos os casos especiais que são identificáveis pelo sistema. Obviamente, se o sistema não tiver um meio de descobrir que é um caso especial, então devemos ter outro evento.

## VIII.10 A Memória do Sistema

Como memória do modelo essencial deve ser utilizado o modelo de entidades e relacionamentos, descrito no Capítulo VII. .

Para cada evento e atividade essencial é importante definir que memórias serão utilizadas. Isso pode ser feito por meio de uma Matriz CRUD ou por meio de DFDs e mini-especificações.

### VIII.10.1 Eventos x Entidades (Matriz CRUD)

A Matriz CRUD é uma tabela que relaciona processos e dados, podendo ser utilizada em diferentes momentos do desenvolvimento de software. Nessa tabela representamos processos nas linhas e dados nas colunas, preenchendo as células resultantes com uma ou mais letras entre “C”, “R”, “U” e “D”, indicando que o processo Cria, Read ou Lê, Update ou Altera, ou Delete ou Apaga um registro,

No caso da modelagem essencial, esta tabela se torna muito interessantes, pois permite relacionar facilmente os eventos essenciais com as entidades do modelo ER.

Mais tarde, se necessário na fase de projeto, essa tabela pode ser reconstruída utilizando os processos sendo implementados e as tabelas do banco de dados.

Eventos ou Atividades	Entidades					
	Ator	Diretor	Novela	Ator Horista	Horas	Capítulo
Cadastrar Diretor		CRUD				
Cadastrar Novela		R	CRUD			
Cadastrar Ator	CRUD			CRUD		
Receber Capítulo	R	R	R	R		CR
Registrar Horas Trabalhadas	R			R	C	
Enviar Formulário	R	R	R	R		

**Tabela 25. A Matriz CRUD do sistema da Rede Bobo de Televisão.**

### VIII.10.2 Verificando a consistência

Um dos principais usos da Matriz CRUD é verificar a consistência do modelo. É obrigatório que cada entidade seja criada e lida por algum evento. Com a Matriz CRUD essa verificação é muito fácil, basta checar se todas as colunas possuem ao menos um “C” e um “R”. Além disso, é interessante, mas não obrigatório, que as entidades também possam ser alteradas e apagadas. Assim, para cada coluna onde não há nenhum “C” ou “R” devemos imediatamente questionar a necessidade da entidade ou buscar eventos que as justifiquem junto ao usuário. Do mesmo jeito, para cada coluna sem “U” ou “D” devemos verificar se essa entidade foi tratada de forma completa em relação aos desejos do usuário.

Existe uma exceção à obrigatoriedade de leitura de uma entidade, que é quando ela está sendo criada para guardar dados que só serão utilizados em uma versão futura do sistema.

### VIII.10.3 Manipulando a Matriz CRUD

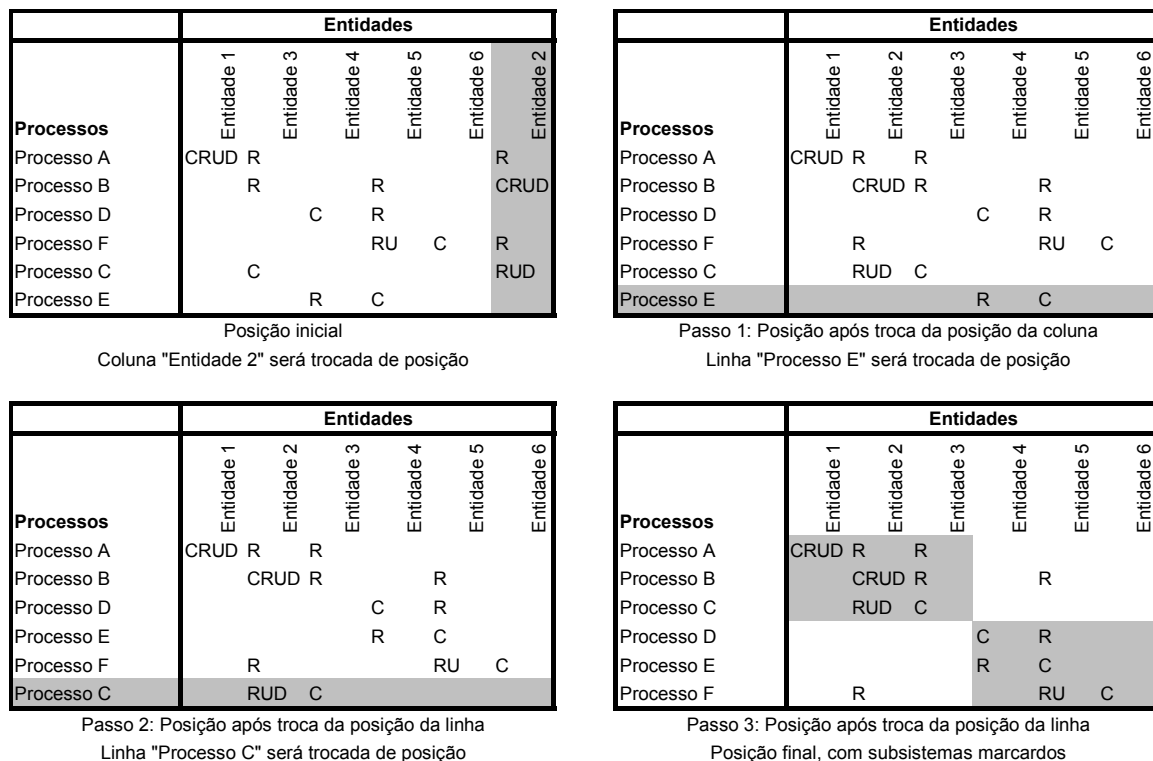
Manipula-se a Matriz CRUD para se obter subsistemas. Um subsistema é identificado pela formação de um *cluster*, isto é, um grupo de células próximas com as mesmas características, que no nosso caso é estarem sendo usadas.

A manipulação é feita alterando-se as posições das linhas e das colunas. Com isso é possível agrupar atividades e entidades que se relacionam mais fortemente em grupos, permitindo a identificação de subsistemas. Os subsistemas interagem normalmente por meio da leitura, por um processo, de uma entidade mantida em outro processo.

A Figura 79 tenta mostrar a dinâmica da manipulação da Matriz CRUD com objetivo de encontrar subsistemas. Essas operações são facilmente feitas em uma planilha eletrônica. As operações de transposição de linha ou coluna podem ser feitas em qualquer

ordem. O objetivo é obter uma matriz onde as células próximas a diagonal sejam bem preenchidas, formando os grupos que caracterizam os subsistemas, enquanto as outras células estão normalmente vazias, apresentando eventualmente operações que indicam a interação entre dois subsistemas.

De certa forma, manipular a Matriz CRUD dessa forma é uma ação equivalente a manipular o DFD Global para se obter o DFD Hierárquico. A diferença básica é que ao tratar a Matriz CRUD estamos em busca de achar configurações da mesma que definam subsistemas de maneira clara, enquanto quando estamos grupando processos em DFD Global buscamos apenas uma forma de organizar o DFD que facilite sua leitura e compreensão.



**Figura 79. Exemplo da manipulação de uma Matriz CRUD em busca de encontrar subsistemas.**

#### VIII.10.4 Extensões da Matriz CRUD

É possível estender a Matriz CRUD para incluir outros conceitos. Uma idéia é transformá-la em uma Matriz CRUDL, onde L significa Listar (List). É claro que para listar precisamos ler, mas isso faz uma diferenciação entre consultas que retornam muitos registros (e que possivelmente retornam apenas parte desses registros) e consultas que retornam um registro inteiro.

Sulaiman<sup>58</sup> et al. propõe a criação, originalmente em modelos onde se está fazendo a análise reversa de uma aplicação em operação, da criação de Cubos CRUD,

<sup>58</sup> [www.cos.ufrj.br/publicacoes/reltec/es61603.pdf](http://www.cos.ufrj.br/publicacoes/reltec/es61603.pdf)

onde a dimensão adicional representa o tempo, ou seja, as células passam a ser vetores, indexados por intervalos de tempo ligados as fases do negócio.

#### **Outros usos da Matriz CRUD**

A Matriz CRUD, na verdade, pode ser utilizada em várias fases do projeto. Antes de termos um modelo de dados e uma lista de eventos, por exemplo, podemos construir tabelas CRUD a partir dos requisitos originais do projeto.

### **VIII.11 Entendendo um Evento**

Para garantir que entendemos completamente um evento, devemos nos perguntar as sete perguntas do método 5W2H: Who, When, Where, What, Why, How, How Much.

- Who? Ou Quem?
  - Quem são os agentes externos?
  - Quem é o iniciador?
  - Quem é o transportador?
  - Existem outras pessoas ou sistemas envolvidos nesse evento?
  - Essa atividade precisa de mais agentes externos?
- When? Quando?
  - Quando ocorre essa atividade?
  - Alguma coisa precisa acontecer antes dessa atividade?
  - Alguma coisa deve acontecer depois dessa atividade?
  - Essa atividade está limitada no tempo por algum outro evento? Por exemplo, só podemos vender após a loja abrir e até a loja fechar.
  - Quando os dados (de entrada ou de saída) são necessários?
- Where? Onde?
  - Onde ocorre a atividade, em que setor ou departamento?
  - De onde vem o estímulo?
  - Para onde vai cada saída?
- What? O que?
  - O que deve ser feito pela atividade?
  - Que dados devem vir no evento externo?
  - Que saídas devem ser feitas?
  - Que dados são necessários?

- Why? Por que?
  - Porque o evento acontece?
  - Porque alguns dados são necessários?
- How? Como?
  - Como a atividade acontece detalhadamente?
  - Como são as saídas (relatórios) e entradas?
- How much? Qual o valor? Quanto custa?
  - Quanto custa implementar o evento?
  - Quanto custa o evento para a empresa cliente?
  - Quanto custa um erro na atividade que realiza o evento?
  - O Diagrama de Fluxo de Dados

## VIII.12 O Dicionário de Eventos

Com o tempo, a Lista de Eventos é entendida para um dicionário de eventos, que descreve detalhadamente as características de cada evento.

Cada entrada no Dicionário de Eventos é composta de:

- **Identificador do evento**, um número único identificar do evento. Esse número é obrigatório.
- **Número de seqüência do evento no tempo**, se existe. Novamente um número, porém indicando a ordem do evento no tempo, se existir. O número é opcional. Vários eventos podem possuir a mesma ordem (pois aconteceriam no mesmo intervalo de tempo).
- **Nome do evento**, uma sentença que identifica o evento, de acordo com as regras análise essencial.
- **Descrição do evento**, uma descrição mais longa do evento, possivelmente contendo informações não essenciais (como a motivação do agente externo), porém que aumentam a compreensão do evento. É um resumo do que é o evento.
- **Classificação do evento** (externo (E/NE), temporal (R/A), Não-evento).
- **Iniciador**, o agente externo que envia o estímulo.
- **Transportador**, i.e., quem inserirá os dados no sistema
- **Dados presentes no estímulo**, descritos segundo alguma linguagem de dicionário de dados, como a descrita nesse texto.



- **Atividade**, descrição sucinta da atividade, por meio de alguma linguagem. Possivelmente uma descrição algorítmica em português estruturado ou como uma sequência de passos. Uma solução interessante é descrever a atividade de acordo com suas pré-condições e pós-condições, possivelmente em uma linguagem formal como VDM ou Z.
- **Informação emitida na atividade, efeito da atividade no ambiente**, descrição de cada saída do sistema de acordo com uma linguagem de dicionário de dados ou equivalente.
- **Efeito da atividade no sistema**, descrição em linguagem natural ou em outra linguagem das modificações que ocorrem no estado global do sistema, ou com entidades específicas, com a execução da atividade. Efeitos colaterais das atividades são descritos aqui. Por exemplo: a atividade pode cadastrar um cliente na lista de clientes inadimplentes, um efeito seria “o cliente está proibido de realizar outros gastos na empresa”.
- **Tempo**, limites de tempo do evento, ligado aos eventos esperados, quando devem acontecer.
- **Lista de entidades utilizadas** (tiradas do modelo conceitual de dados), ou Matriz CRUD.

Esse texto é acompanhado de um software que permite a criação de um dicionário de eventos. Na figura a seguir apresentamos a tela para o dicionário.

**Figura 80. Tela de um dicionário de eventos**

Entidades utilizadas por Eventos

Identificador

Evento

Entidades

Nome de Entidade	Cria	Lê	Altera	Apaga
<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Record:  of 1

Record:  of 1

**Figura 81. Tela complementar, indicando as entidades (memórias) envolvidas em cada evento.**

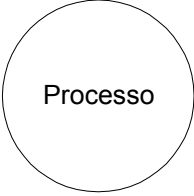
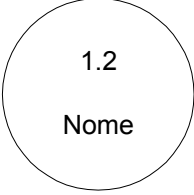
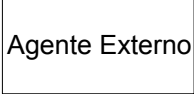
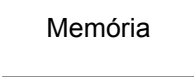


## VIII.13 O Diagrama de Fluxo de Dados

O objetivo de um Diagrama de Fluxo de Dados (DFD) é descrever, graficamente, o fluxo de informação e as transformações que são aplicados nos dados [32]. Ele pode ser utilizado para representar, em diferentes níveis de abstração, sistemas de informação, não necessariamente automatizados. DFDs permitem a modelagem funcional de um sistema em vários níveis de abstração.

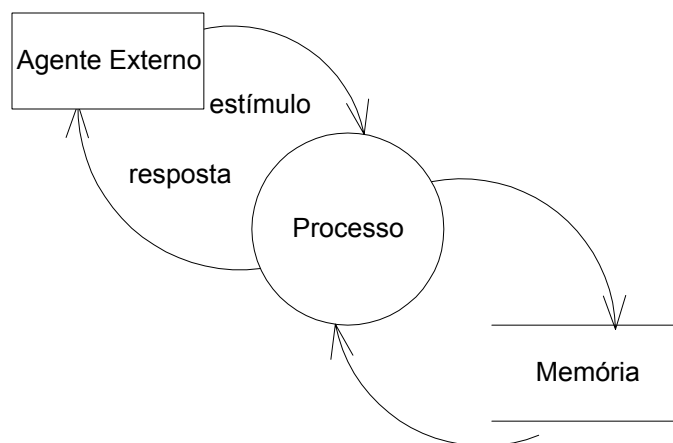
Cada diagrama de um DFD é composto de 4 tipos de objetos: processos, memórias, agentes externos e fluxos de dados. Cada um desses objetos tem um símbolo, um nome e possivelmente um rótulo associado.

No passado existiram duas correntes de símbolos para desenhar DFDs. Os que desenhavam processos como caixas com as bordas arredondadas, baseados na proposta de Gane e Sarson [33], e os que desenhavam processos com círculos, baseados na proposta de Coad e Yourdon [34]. Atualmente a forma de Yourdon é a mais difundida no mercado.

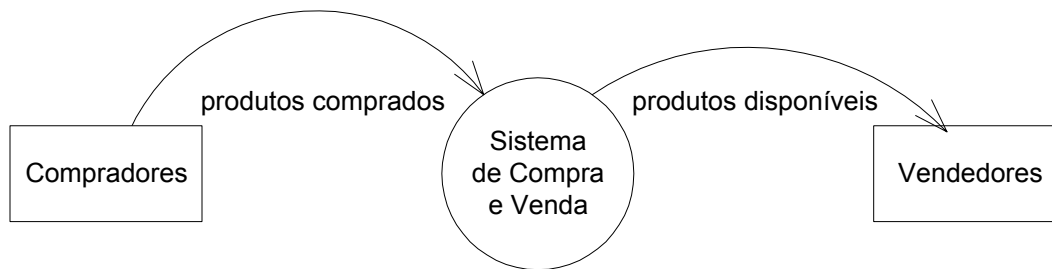
Cada processo em um DFD pode ser expandido em outro DFD, que o descreve. Dessa forma, DFDs são utilizados para descrever um sistema em níveis cada vez mais detalhados de informação. Usualmente também usamos o termo DFD para descrever um conjunto de diagramas construídos com essa notação.

Figura	Descrição
	Um processo identificado apenas pelo nome
	Um processo com número e nome
	Um Agente Externo identificado
	Uma memória identificada
	Um fluxo de dados sem identificação
	Um fluxo de dados identificado

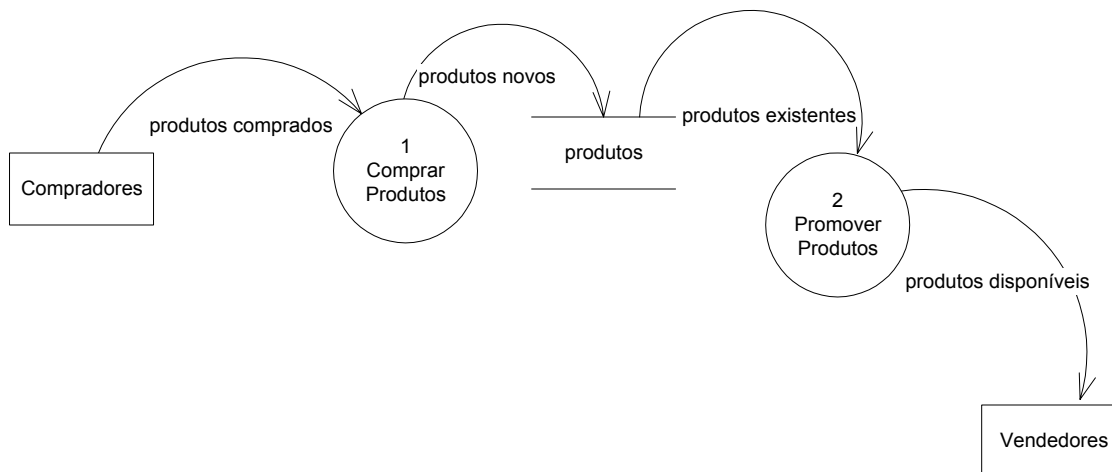
**Tabela 82. Símbolos e seus significados para a construção de um Diagrama de Fluxo de Dados**



**Figura 83. Um DFD simples**



**Figura 84. Um DFD de Contexto muito simples**



**Figura 85. A explosão do DFD de Contexto da Figura 84**

DFDs não descrevem uma seqüência de execução ou qualquer outra relação de controle<sup>59</sup>. Se uma seta indica que dados passam de um processo A para um processo B, é possível que A chame B, B chame A, ou ainda que outro processo superior chame os dois e faça a transferência de dados. DFDs também não dão nenhuma informação sobre a lógica de execução, como repetições ou condições. Qualquer fluxo de dados ou processo presente pode ou não executar, uma ou mais vezes. Na verdade, existem técnicas especiais para derivar relações de controle entre processos a partir da estrutura de um DFD. Essas técnicas, porém, não se aplicam ao caso em estudo nesse livro, pois não são muito eficazes para desenvolver sistemas Cliente/Servidor. Finalmente, o DFD também não informa se algo acontece (ou seja, se os fluxos de dados realmente comunicam os dados entre os processos), mas sim que algo pode acontecer (ou seja, que é possível que um processo envie seus fluxos de dados).

<sup>59</sup> De certa maneira, descrições feitas com DFDs são semelhantes a descrições feitas com diagramas IDEF0, pois estes são descendentes de uma outra notação, SADT, que era usada de forma alternativa a DFDs.

### VIII.13.1 Algumas regras sintáticas

Dados só podem passar de agentes externos para memórias por meio de um processo. Agentes externos ou memórias também não podem se comunicar diretamente. Isso significa que **fluxos de dados devem começar ou acabar em um processo**.<sup>60</sup>

Processos devem ler alguma informação, de um agente externo, outro processo ou memória, e gerar alguma informação, para um agente externo, outro processo ou memória. Não **existem processos que criam informação do nada** (fontes) **nem processos que consomem informação** (“buraco negro” ou *sink*).

**Um processo tem um nome e um número.** O nome deve ser formado por um verbo no infinitivo e um objeto (atender cliente, vender produto, etc.). O número deve identificar unicamente o processo. Se estivermos usando um DFD para descrever um processo que aparece em outro DFD, então usamos uma numeração hierárquica. Por exemplo, os processos que ficam em um DFD que explica o processo número “2” devem ser numerados “2.1”, “2.2”, “2.3” e assim sucessivamente. Se precisarmos de outro DFD para explicar o processo “2.2”, os processos nesse terceiro DFD serão numerados “2.2.1”, “2.2.2”, sucessivamente.

**Agentes externos e memórias** (depósitos de dados) possuíam um rótulo na notação original de Gane e Sarson [33], porém na notação de Yourdon, que utilizamos agora, **não possuem rótulos**.

### VIII.13.2 Entendendo os Fluxos de Dados

Em um DFD, na verdade, temos três tipos de fluxos de dados. O primeiro tipo de fluxo, que ocorre entre um processo e um agente externo, representa dados que estão cruzando a fronteira do sistema. Esse tipo de fluxo indica que cada dado descrito no fluxo estará, possivelmente de forma opcional, sendo apresentado por ou a um agente externo. Ele representa, realmente, um “fluxo” dos dados nele descritos.

Já entre um processo e uma memória, o fluxo tem um significado levemente diferente. Em primeiro lugar ele ocorre dentro do sistema, não cruza nenhuma fronteira. Mas, principalmente, um fluxo saindo da memória em direção ao processo indica uma consulta a uma memória, possivelmente usando operações equivalentes a seleções e projeções da lógica relacional, sem modificá-la. Por outro lado, um fluxo saindo de um processo e entrando em uma memória indica uma alteração dessa memória, que pode significar a criação, a alteração ou a eliminação de um ou mais registros.

Finalmente temos fluxos de dados entre processos. Esses fluxos querem dizer apenas que os dados utilizados em um processo são fornecidos por outro processo, porém não há nenhum compromisso, no DFD, que isso seja feito dinamicamente. Um fluxo desse tipo é, *a priori*, equivalente ao primeiro processo salvar os dados em uma memória e o segundo processo ler dessa memória. As implementações podem ser várias, incluindo o primeiro processo ativar o segundo, o segundo ativar o primeiro ou os dois serem

---

<sup>60</sup> Nos diagramas particionados isso é um ou-exclusivo, ou seja, um fluxo de dados pode partir ou chegar em uma atividade, mas não ambos. Nos outros diagramas um fluxo pode indicar a comunicação entre dois processos.

ativados de forma assíncrona. Devemos notar que não aparecem fluxos de dados entre processos em DFDs particionados por eventos.

### VIII.13.3 Entendendo as Memórias

Na análise de sistemas fazemos a modelagem de dados simultaneamente a modelagem funcional. Isso permite que as memórias do modelo essencial sejam modeladas diretamente nas entidades do modelo de entidades e relacionamentos, uma das técnicas recomendadas originalmente e a mais adequada para o desenvolvimento de sistemas na atualidade<sup>61</sup>.

Ao definir as memórias utilizamos uma regra simples: toda a entidade do modelo de dados utilizada pelo processo deve ser tocada por uma seta. Se a seta estiver indo do processo para a memória, dizemos que há uma alteração dessa memória, pela adição, exclusão ou alteração de um ou mais registros. Se a seta estiver vindo da memória para o processo então dizemos que está acontecendo uma leitura na memória, ou uma busca, porém fica claro que seu conteúdo não é modificado.

É razoável tanto utilizar como memórias as entidades ainda não normalizadas na terceira forma normal (modelo conceitual tradicional) quanto às normalizadas.

### VIII.13.4 Tipos de DFD

Em um **DFD de Contexto** todo o sistema é representado por apenas um processo. Não aparecem memórias internas ao sistema em um DFD de Contexto, apenas agentes externos e, segundo alguns autores, memórias que pertencem a outros sistemas e que são utilizadas pelo sistema sendo descrito. Geralmente o DFD de Contexto é o primeiro diagrama de um DFD nivelado. O processo que aparece nesse diagrama geralmente tem o nome do sistema sendo implementado.

O DFD de contexto pode ser criado imediatamente a partir do dicionário de eventos, ou da lista de eventos e respostas.

Um **DFD Nivelado** ou **Hierárquico** é na verdade um conjunto de vários DFD interligados de forma hierárquica de modo que exista um DFD inicial e que cada DFD além desse possa ser alcançado a partir da expansão sucessiva dos processos em DFDs. DFDs Nivelados são a forma mais tradicional de descrever um sistema de forma *top-down* em Engenharia de Software.

Alguns autores chamam de DFD nível zero o DFD de Contexto. Outros chamam de DFD nível zero o DFD gerado pela expansão do processo que aparece no DFD de Contexto. Nós ficaremos com a segunda opção, isto é: o primeiro DFD é o de Contexto, contendo um único processo. Esse processo é expandido para o DFD nível zero, que contem um número razoável de processos (entre 5 e 9, normalmente).

Um **DFD particionado** representa cada atividade essencial como um diagrama isolado, cujo significado será explicado mais tarde. DFDs particionados têm apenas um processo por diagrama, a atividade essencial, que recebe dados de no máximo um agente

---

<sup>61</sup> Isso porque a modelagem conceitual de dados se encaixa perfeitamente com o conceito de modelo essencial e ainda permite uma transição rápida para os SGBD relacionais (oops! Estamos falando de tecnologia?).

externo e de uma ou mais memórias. Nosso método de desenvolvimento é baseado no uso de DFDs particionados e nas expansões de seus processos.

Um **DFD global** é a unificação em um só diagrama de todos os DFDs particionados de um sistema, eliminando as repetições de agentes externos e memórias. Um DFD global pode ser usado como passo intermediário para transformar um DFD particionado em um DFD nivelado. DFDs globais são normalmente ferramentas de rascunho e não de análise. Também é possível substituir um DFD global por uma Matriz CRUD.

### **VIII.13.5 Criando o DFD Particionado**

A principal forma de comunicar a modelagem essencial é pela criação de um DFD Particionado. Nesse DFD apresentamos um diagrama para cada evento. Cada um desses diagramas contém apenas um evento e apenas uma atividade essencial.

Quando dizemos que cada diagrama contém apenas um evento, estamos também dizendo existe no máximo um fluxo de dados entrando o sistema vindo de um agente externo. Da atividade para os agentes externos podem existir vários fluxos, que representam a saída de dados do sistema. Entre atividades e memórias também podem existir quantos fluxos forem necessários para representar a busca, inserção, alteração e eliminação de dados da memória.

Os DFD particionados podem ser resumidos em um DFD de Contexto. Nesse DFD não aparecem as memórias internas ao sistema e o sistema é representado como apenas um processo. A principal função do DFD de contexto é representar, em um só diagrama, todas as interações do ambiente com o sistema (o contexto da aplicação!).

Vejamos a seguir uma descrição simples de um sistema e os DFDs particionados equivalentes.

### **VIII.13.6 Criando o DFD hierárquico**

O Diagrama de Fluxo de Dados Hierárquico (ou nivelado) já foi uma das principais ferramentas de análise e documentação de sistemas. Atualmente, devido à metáfora de programação gerada pelos ambientes GUI, que é baseada em eventos, ele é normalmente dispensado. Também devemos notar que a análise essencial, apesar de facilitar sua criação, dificulta a manutenção de um DFD hierárquico, pois esse deve ser reconstruído a cada modificação dos eventos particionados.

Em todo caso, o DFD nivelado permite uma visão abstrata do sistema composta de visões parciais cada vez mais detalhadas. Como essa característica é muito boa, algumas vezes ainda podemos desenvolver DFD hierárquicos.

Para isso, iniciamos com o DFD Global do sistema. O DFD global não é nada mais que a unificação de todos os DFD particionados do sistema em um único DFD, onde cada memória, atividade essencial e agente externo só aparece uma vez. Ao fazer o DFD global escolhemos uma folha de grande tamanho (A3 ou maior) e colocamos os DFDs particionados nessa folha um a um, sempre reaproveitando todos os símbolos já existentes. Essa forma de construir pode ser difícil para sistemas muito grandes, sendo outra estratégia colocar no centro da folha todas as memórias, depois colocando os processos ao redor das memórias e os agentes externos ao redor dos processos. Se o

sistema ficar realmente muito complicado, é aceitável duplicar agentes externos. A duplicação de memórias para facilitar a construção do diagrama deve ser evitada ao máximo, mas é aceitável em alguns casos, como memórias que são claramente acessadas por um número grande de processos. A duplicação de processos é proibida.

O diagrama global representa um nível intermediário do sistema, o nível onde cada processo representa um evento. A partir do diagrama global construiremos o DFD particionado pela seleção de atividades essenciais em processos.

As seguintes heurísticas podem ser utilizadas para isso:

- Agregar em um único processo todas as atividades essenciais que usam uma memória específica. Como resultado, essa memória também será representada dentro desse processo em um nível superior do DFD,
- Agregar todas as atividades de custódia que acessem uma memória específica,
- Agregar funcionalidades que são utilizadas por um agente externo específico e
- Agregar por meio de funções de negócio.
- Manter o nível de complexidade de cada processo criado entre números recomendáveis ( $7 \pm 2$  objetos em cada processo).

O resultado será um conjunto de processos onde cada processo contém várias atividades essenciais e talvez uma ou duas memórias, interligados por sua conexão com memórias do sistema.

Se o número resultante de processos for entre 5 e 9, alcançamos um nível abstrato razoável para o diagrama de nível zero. Caso contrário, repetimos esse processo até que isso aconteça.

Finalmente, alcançado o nível zero, temos que construir o DFD nivelado por meio da explosão de cada processo e criação de um DFD específico para cada explosão, respeitando-se todas as regras normais de criação de Diagrama de Fluxos de Dados.

## VIII.14 Especificando Processos

O nome de cada processo, incluindo as atividades essenciais, é formado de um verbo no infinitivo e de um objeto direto, que indicam como o sistema responde ao evento.

Exemplos:

Evento	Atividade
Gerente solicita relatório de Vendas	Emitir Relatório de Vendas
Aluno solicita boletim	Emitir Boletim
Fornecedor entrega mercadoria	Receber mercadoria

**Tabela 26. Nomes de atividades para alguns eventos**



#### VIII.14.1 Especificação do Tipo Caixa Preta

A primeira especificação que devemos fazer de um processo ou atividade essencial deve enxergar esse processo ou atividade como uma caixa preta. Dessa forma, a descrição deve discutir apenas seus efeitos nas memórias e as entradas e saídas dos agentes externos.

Esta especificação inicial deve ser feita utilizando a linguagem do cliente.

Por exemplo:

- Especificação do Processo “Cadastrar Cliente”:
  - Após conferir se o CGC é válido, deve registrar as informações passadas pelo cliente na memória “CLIENTE”.

#### VIII.14.2 Especificação do Tipo Caixa Branca

A especificação de processo, na modelagem essencial, é chamada de mini-especificação. Cada processo, e nisso se incluem as atividades essenciais, deve ser especificados por meio de uma mini-especificação ou refinado por meio de outro DFD.

Uma mini-especificação pode ser escrita em português estruturado, usando tabelas ou árvores de decisão. Qualquer que seja a linguagem escolhida, deve permitir o entendimento claro do que deve ser feito durante aquele processo.

Por exemplo:

- Especificação do Processo “Cadastrar Cliente”:
  - **Se** CGC é Válido **Então**
  - **Salvar** Cliente.

#### VIII.14.3 Mini-especificações

Uma especificação de um processo (mini-especificação) tem como objetivo especificar “o que” o processo deve fazer (e não como). Uma especificação em português estruturado deve ser clara. Em geral, cada empresa ou projeto deve determinar como escrever sua mini-especificação.

##### VIII.14.3.1 Português Estruturado

Algumas regras básicas:

- Toda a lógica deve ser expressa na forma de estruturas sequenciais, estruturas de decisão, estruturas de case, ou iterações.
- As palavras chaves devem ser destacadas (negrito ou letras maiúsculas).
- Identificar claramente os blocos de comando, mostrando sua hierarquia.
- Destaque as palavras ou frases definidas no dicionário de dados, para indicar que têm um significado específico (sublinhe ou itálico).
- Seja atento no uso das condições “ou”, “e”, “maior”, “maior ou igual”.
- Sintaticamente falando:
- Uma mini-especificação é composta de uma sequência de comandos.

- Um comando pode ser um comando estruturado ou um comando simples
- Um comando estruturado pode ser
  - Um bloco
  - Um comando se - então
  - Um comando se - então - senão
  - Um comando faça - caso
  - Um comando faça - enquanto
  - Um comando repita - até
  - Um comando para\_cada - faça
  - Outro comando acertado entre o grupo
- Um comando simples pode ser
  - Um comando de atribuição
  - Um comando de busca em memória
  - Um comando de escrita em memória
  - Um comando de leitura na interface
  - Um comando de escrita na interface
  - Outro comando acertado entre o grupo

O importante é manter a consistência. Veja o exemplo a seguir:

PROCESSO Preparar\_Segunda\_Via

INÍCIO

LER tipo\_pessoa

SE *tipo\_pessoa*="FISICA" ENTÃO

INÍCIO

LER cpf\_pessoa

BUSCAR quem EM Contribuinte COM *quem.cpf=cpf\_pessoa*

*codigo\_cont* := *quem.codigo*

FIM

SENÃO

INÍCIO

LER cgc\_empresa

BUSCAR empresa EM Contribuinte COM *empresa.cgc=cgc\_empresa*

*codigo\_cont* := *empresa.codigo*

FIM

FIM\_DO\_SE

LER ano

```
BUSCAR iptu EM impostos COM iptu.ano=ano E iptu.codigo=codigo_cont  
IMPRIMIR iptu  
FIM
```

#### VIII.14.4 O Diagrama de Transição de Estados

Um diagrama de transição de estados, ou simplesmente diagrama de estados, ou ainda DTE, é uma das abstrações mais gerais que temos para um sistema ou objeto. O objetivo de um DTE é descrever como um sistema ou objeto muda de estado em função de eventos que ocorrem no ambiente, e que respostas estão associadas a cada mudança de estado.

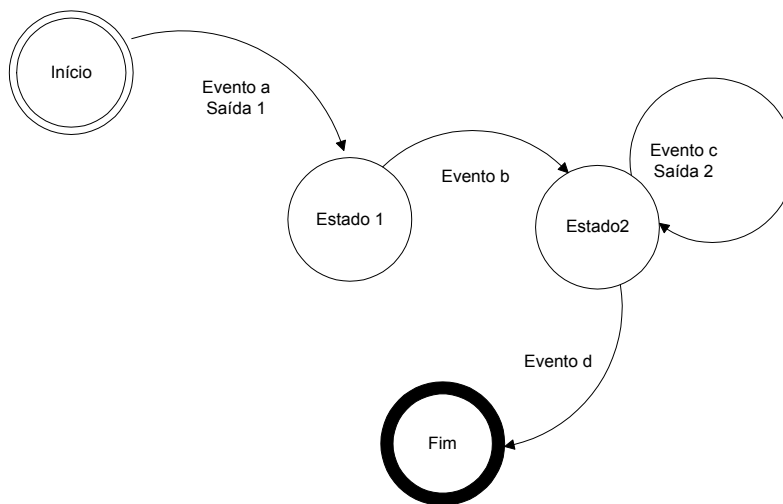
Diagramas de estados são compostos de dois símbolos básicos: estados (círculos ou caixas) e transições (setas). Os estados têm um nome ou um número, enquanto as transições são indicadas com um evento, que ativa a transição, e uma saída, provocada pela transição.

Apesar de sua simplicidade, DTEs são ferramentas muito poderosas para modelagem. Podem ser usados de diferentes formas na modelagem essencial: para descrever o funcionamento do sistema como um todo ou de parte dele, para descrever os estados possíveis de uma entidade. Mesmo uma mini-especificação pode, em alguns casos, ser mais bem representada por um diagrama de estados.

Yourdon [34] sugere que as seguintes perguntas devem ser feitas para verificar um DTE:

- Todos os estados foram definidos?
- Todos os estados podem ser atingidos?
- Todos os estados têm uma saída?
- Em cada estado o sistema reage adequadamente a todos os eventos?

Existem no mercado diferentes ferramentas, gratuitas ou comerciais, de desenho e verificação de DTEs. Existem também outras formas similares de modelagem, como Redes de Petri e StateCharts.



**Figura 86. Um diagrama de estados simples**

Não existe uma notação padrão para diagramas de estado, mas a notação utilizada na Figura 86, com símbolos especiais para os estados iniciais e finais, será a adotada nesse texto.

Diagramas de estados podem ser representados por tabelas, como a seguir:

Estado Atual	Evento	Resposta	Próximo Estado
Início	Evento a	Saída 1	Estado 1
Estado 1	Evento b	-	Estado 2
Estado 2	Evento c	Saída 2	Estado 2
Estado 2	Evento d	-	Fim

**Tabela 27. Tabela representando o diagrama da Figura 86**

#### VIII.14.5 Tabelas de Decisão

Uma tabela de decisão descreve que ações devem ser tomadas quando uma série de fatos acontece. Uma tabela de decisão tem duas partes. Na primeira parte cada linha representa um fato. Na segunda parte, cada linha representa uma ação. As colunas significam combinações de fatos, determinados pelos valores: “verdadeiro”, “falso” e “não aplicável”.

Eventos ou fatos	Livro Existe na Base	V	V	V	F	F	F
	Cliente Cadastrado	V	V	F	V	V	F

	Cliente deve dinheiro	V	F	NA	V	F	NA
Respostas ou Ações	Cadastra cliente			✓			✓
	Prepara pedido		✓	✓			
	Recusa pedido	✓			✓	✓	✓

**Tabela 28. Uma tabela de decisões**

#### VIII.14.6 Pré-condições e pós-condições

Baseadas em linguagens formais de especificação e implementadas em algumas linguagens, como Eiffel, pré-condições (e pós-condições) são declarações que devem ser verdade antes (e depois) do código ser executado.

Por exemplo, um cadastro poderia ser especificado (de forma bastante simplificada) como:

**pré**

input = (x,y,z) onde x ∈ Nome , y ∈ Endereço, z ∈ Telefone

**pós**

(x,y,z) ∈ Aluno

### VIII.15 Dicionário de Dados

O Dicionário de Dados é uma definição formal e estruturada dos fluxos de dados, elementos de dados, memórias, entidades e relacionamentos [25]. Funciona, para os dados um sistema sendo descrito, como um dicionário funciona para a língua que falamos.

Existem várias notações diferentes para construir dicionários de dados, mas todas são equivalentes. Nossa notação terá o seguinte formato básico:

*<termo definido> = <definição do termo>*

O DD deve conter a descrição de todos os fluxos e de todas as memórias (entidades) do sistema.

O termo definido é normalmente uma palavra, enquanto a definição do termo pode assumir várias formas:

- Combinação de outros termos (uso do +)

*<termo> + <termo> ...*

- Repetição de termos (uso das chaves, possivelmente com limites mínimos e máximos)

*{<termo>}*

*1:3{<termo>}*

- Termos opcionais (uso dos parênteses)

(*<termo>*)

- Uma escolha entre termos (uso dos colchetes)

[*<termo1> | <termo2> ...*]

- Valores possíveis (uso de aspas)

*“valor1”*

- Comentários (uso de asteriscos)

*\*um nome de pessoa\**

A seguinte definição é válida:

```
comprador = nome + [cgc | cnpj] + endereço + 0:2{telefone} +
(pessoa_de_contato)
```

Significando que um comprador deve ser representado por um nome, um CGC ou CNPJ, o endereço, entre 0 e 2 telefones e, opcionalmente, uma pessoa de contato.

Itens simples serão descritos por um comentário. Quando o significado é óbvio, usa-se o comentário *\* dado elementar \**.

Também é comum usar o símbolo “@” para marcar os termos que compõe a chave de outro termo.

Podemos guardar também regras de cálculo (e regras de negócio) no dicionário de dados.

#### **Modelos de Yourdon**

Yourdon definiu dois modelos como objetivos do processo de análise, o modelo ambiental e o modelo comportamental.

O **Modelo Ambiental**<sup>62</sup> é composto de: Objetivos, Lista de Eventos e Diagrama de Contexto.

O **Modelo Comportamental**<sup>63</sup> é composto de: Diagrama de Contexto, Lista de Eventos, Declaração de Objetivos, DFD hierárquico completo, Mini-especificações para todos os processos que não forem expandidos em um DFD, Diagrama de Entidades e Relacionamentos completo, Conjunto completo de diagramas de transições de estado e Dicionário de dados completo.

## **VIII.16 Exercício**

### **VIII.16.1 Rede Bobo de Televisão**

O Núcleo de Novelas da Rede Bobo de Televisão contratou você para fazer um sistema para controlar suas novelas. O sistema deve permitir que um operador cadastre,

<sup>62</sup> O modelo ambiental foi definido por Yourdon[34], sendo questão típica de concurso.

<sup>63</sup> O modelo comportamental foi definido por Yourdon [34], sendo questão típica de concurso.

em separado, novelas, atores e diretores. Os atores podem ser contratados ou horistas. Os atores e diretores só podem trabalhar em uma novela, mas uma novela pode ter vários atores e apenas um diretor.

Quando um ator passa a trabalhar em uma novela, isso deve ser registrado no sistema. Se ele for ator contratado, deve ser impresso um aviso para o departamento de pessoal, dizendo que ele está alocado na novela.

Quando um autor envia um capítulo (com resumo), o sistema deve cadastrar esse capítulo e produzir uma cópia para cada ator e diretor da novela.

Toda segunda feira, o sistema deve emitir cópias, para a imprensa, dos 6 resumos dos capítulos da semana.

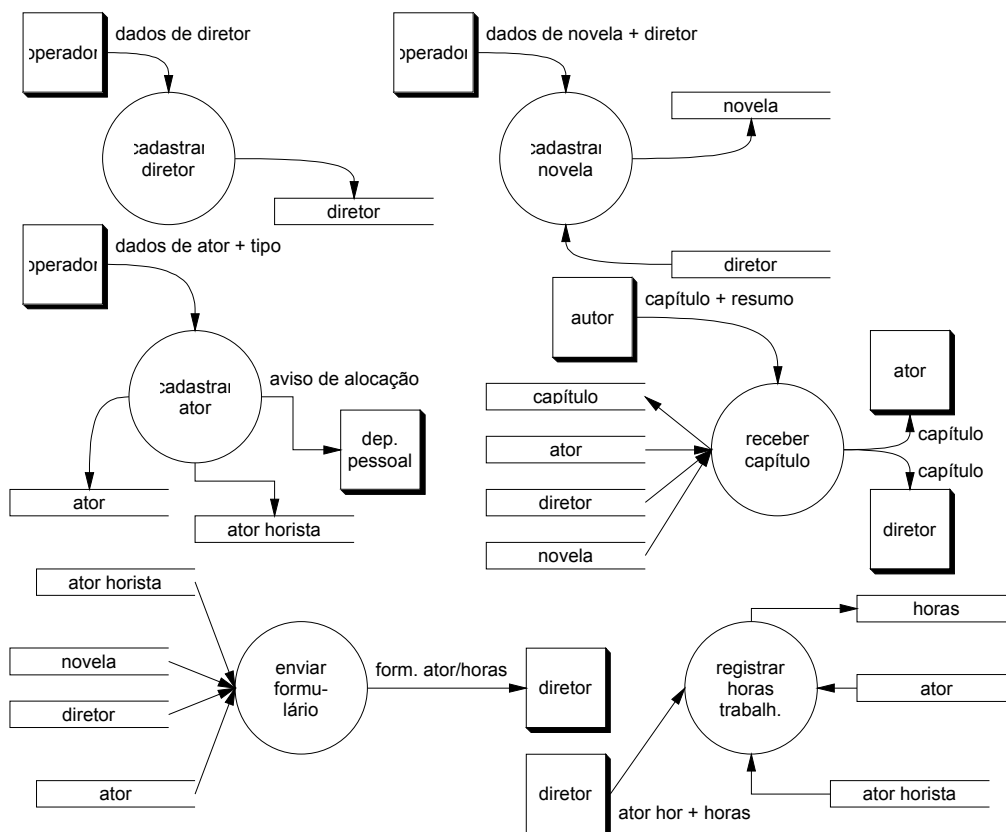
No final do mês (dia 25), o sistema deve gerar, para a pagadoria, uma listagem dos atores horistas que devem ser pagos. Para isso, dia 15 deve ser enviado um formulário aos diretores, com o nome de todos os atores horistas associados à novela com um espaço para marcar. Os diretores devem devolver esses relatórios preenchidos, com o registro de horas trabalhadas por ator horista, que são registrados no sistema. Se dia 23 existir um diretor que não devolveu o relatório, deve ser enviada uma carta de notificação ao diretor.

Lista de Eventos<sup>64</sup>:

- Operador Cadastra Diretor (custodial, externo, não esperado)
- Operador Cadastra Novela (custodial, externo, não esperado)
- Operador Cadastra Ator (custodial, externo, não esperado)
- Autor Envia Capítulo (composto, externo, não esperado, na prática deveria ser esperado, mas não é assim que foi descrito)
- É dia de enviar formulário (composto, temporal)
- Diretor envia formulário preenchido (composto, externo, esperado, tem um não evento)
- É dia de enviar carta de notificação ao diretor (não evento) (esse evento não está no diagrama por um defeito de fabricação do mesmo, mas aparecerá na próxima versão).

---

<sup>64</sup> Atenção para a forma como cada evento é descrito



**Figura 87. Todos os componentes do DFD particionado do sistema para Rede Bobo de Televisão<sup>65</sup> em uma só imagem**

## VIII.17 Modelos Adicionais

### VIII.17.1 Finalizando a Análise

Um dos trabalhos mais importantes e aborrecidos da fase de análise é manter todos os documentos consistentes. Isso fica mais difícil quando estamos usando métodos diferentes em ferramentas distintas.

Assim, se temos em uma mesma ferramenta CASE o diagrama de entidades e relacionamentos e todos os nossos eventos, é provável que seja possível nunca conseguir escrever uma especificação inconsistente ou pelo menos verificar se existe alguma inconsistência. Porém, se utilizamos mais de uma ferramenta CASE<sup>66</sup>, fica mais difícil manter essa consistência.

<sup>65</sup> Esse DFD particionado é uma figura inexistente durante o projeto. Em um documento real, cada diagrama apareceria em uma página, e não todos agrupados. Além disso, devemos notar a ausência de um não evento possível.

<sup>66</sup> Se não utilizamos ferramentas CASE estamos incorrendo em um erro grave.



## Capítulo IX. Modelo de Interface

---



*Dilbert © by Scott Adams*

Muitos autores consideram que a modelagem da interface com o usuário é uma ação característica da fase de projeto de um sistema. Porém, a prática de desenvolvimento de sistemas mostra que o usuário só tem uma verdadeira visão da funcionalidade do sistema quando pode ter alguma amostra do seu funcionamento. Sem essa visão, é muito difícil para o usuário validar uma análise. Em decorrência disso, é muito comum que um modelo de dados ou um modelo funcional desenvolvido e validado com o usuário contenha falhas. Isso acontece porque estes modelos são modelos abstratos criados em uma linguagem mais próxima e conhecida do desenvolvedor do que do cliente.

Este capítulo não fala profundamente sobre a questão de como deve ser uma interface com o usuário, apenas apresenta alguns métodos de modelagem para a mesma, com o intuito de fornecer aos interessados em um sistema uma visão do seu comportamento, funcionamento e aparência.

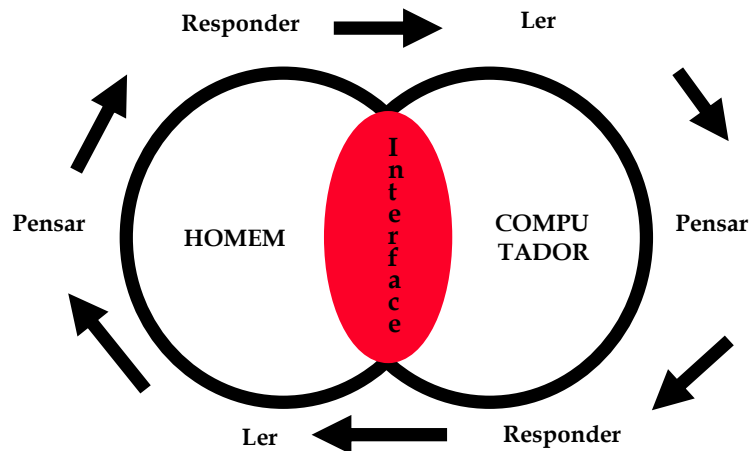
O leitor interessado nestas questões deve procurar textos próprios das seguintes áreas: Interação Homem-Computador (Human-Computer Interaction, HCI), Projeto centrado no usuário (user centered design), Interface com o Usuário (User-Interface, UI), Engenharia Cognitiva, Projeto Participativo (Participatory Design), Ergonomia, Avaliação de Interfaces com o Usuário e ainda outras. Uma boa dica é consultar o site <http://www.hcibib.org/>

### IX.1 A Interação Homem Computador (IHC)

Um sistema não é composto apenas pelo programa de computador, mas também por aqueles que se comunicam de programa, sejam eles humanos ou outros programas. Podemos mudar o projeto de um computador ou de um programa, mas não podemos mudar os seres humanos. Precisamos, então, compreendê-los melhor para poder desenhar interfaces melhores.

A interação homem computador corresponde a um ciclo, onde cada parte recebe informações, as processa e emite novas informações. Uma das partes é um ser humano, a outra é um computador. Cada uma possui características próprias para cada uma dessas atividades. No caso do processamento, por exemplo, computadores (bem programados) têm uma capacidade de cálculo inigualável pelo ser humano, enquanto seres humanos são capazes de proezas no reconhecimento de padrões.

Deve ficar claro que não estamos aqui nem humanizando o computador, nem vendo o homem como uma máquina. Apenas procuramos uma metáfora que nos permita entender melhor como se realiza a interface entre eles. Também não esquecemos que computadores foram programados por seres humanos. Uma das formas interessantes de ver a IHC é como um diálogo entre o desenvolvedor e o analista, porém não trataremos desse assunto aqui.



**Figura 88. A Interação Homem Computador**

## **IX.2 Projetando a Interface com o Usuário**

As escolhas feitas na modelagem de interface com o usuário são um problema ainda em aberto e que envolvem muitos fatores, sendo estudados por muitos autores nos campos de interação homem computador e projeto (design) de interface.

Entre os principais fatores envolvidos estão aqueles que envolvem aspectos humanos. A ciência cognitiva é uma das ferramentas utilizadas, pois estuda o conjunto de processos mentais relacionados ao pensamento, a percepção e a tarefas como reconhecimento e classificação. A ergonomia é outro fator importante.

Outros fatores importantes em projetos reais são as ferramentas disponíveis e o hábito e a cultura do usuário. Enquanto um projeto acadêmico, ou um web-site de propaganda, pode ousar muito na sua concepção de interface, projetos empresariais muitas vezes devem seguir padrões ou se adaptar a condições de uso que não são as ideais. Projetos internacionais ainda têm outros problemas, pois a variação de cultura de um local para o outro pode invalidar premissas do desenvolvedor. Como exemplo simples, enquanto as línguas como português e inglês são escritas da esquerda para direita, árabe e hebraico são escritas da direita para esquerda. Isso afeta toda uma forma de entender a interface com o usuário.

Dentro do fator humano e cognitivo, uma questão que sempre deve ser levada em conta é que o usuário faz um modelo mental do sistema, fazendo suposições sobre o que deve ocorrer em uma parte do sistema baseado no aprendizado que teve em outra parte do mesmo. Vamos dar um exemplo simples: se em uma parte do sistema ao tentar apagar um objeto o usuário tem a oportunidade de desistir ou voltar atrás, vai esperar que o mesmo

aconteça em todo o sistema. Quando por algum motivo essa regra é quebrada o usuário fica surpreso, pois há uma quebra do seu modelo mental “apagar permite desistir ou voltar atrás”. Isso faz com que nossos modelos de interação com usuário busquem não só a consistência, mas também facilitar a criação do modelo mental do usuário por meio do uso de metáforas ou outras formas de indicação.

### IX.2.1 Introdução ao Modelo Cognitivo usado em IHC

Como vimos anteriormente, podemos fazer um modelo do ser humano (e do computador) dividindo suas atividades ao usar o computador em três tipos: percepção (responsável pela “leitura” de informações), atividades cognitivas (memória e processamento) e sistema motor (responsável pela “saída” de informações).

Uma característica importante de seres humanos é que eles podem ser modelados efetivamente como possuindo duas memórias: uma memória de curto prazo (MCP) e uma memória de longo prazo (MLP). A primeira é rápida e limitada, com pequena duração. A segunda é infinita, faz associações muito complexas e não tem acesso confiável. Além disso, o sistema cognitivo humano tem um processamento lento, apesar de ser poderoso em algumas atividades, como o reconhecimento de padrões visuais.

Cabe ao projetista da interface lidar com todas as características do ser humano, e também das infinitas variedades entre os seres humanos.

## IX.3 Coletando Informações Sobre o Usuário

Uma das mais importantes atribuições do analista ou *designer* ao projetar uma interface é conhecer seu usuário. Usuários diferentes têm demandas diferentes para um mesmo sistema.

Um modelo simplificado de usuários admite três tipos de usuários: um usuário que não conhece a aplicação e nem o sistema (novato), um usuário que conhece a aplicação e o sistema (experiente) e o usuário que conhece bem a aplicação, mas pouco o sistema (eventual). Normalmente um sistema tem que atender simultaneamente a todos esses tipos de usuário e ainda ajudar a transformar um usuário novato em experiente.

É necessário, então, coletar informações sobre os usuários do sistema. Isso pode ser feito por meio de formulários. Esses formulários devem levantar a formação do usuário, seu tempo no cargo e na empresa, sua experiência com computadores, Internet, outros sistemas da empresa ou sistemas semelhantes. Além disso, é interessante conhecer que atividades o usuário faz em seu trabalho e com que frequência (diária, semanal, mensal, raramente). Outras informações podem ser levantadas, como impossibilidades de usar uma ferramenta específica ou incapacidades físicas. Existem vários modelos de formulários que podem ser utilizados como referência na literatura e na Internet, a seguir apresentamos um exemplo muito simples.

### IX.3.1 Questionário para identificação de usuários

Pergunta	Respostas
----------	-----------

Pergunta	Respostas				
Formação	Sem educação formal	1º grau	2º grau	3º grau	Pós-graduado
Tempo na empresa	Menos de 6 meses	Entre 6 meses e 1 ano	Entre 1 e 2 anos	Entre 2 e 5 anos	Mais de 5 anos
Tempo no cargo	Menos de 6 meses	Entre 6 meses e 1 ano	Entre 1 e 2 anos	Entre 2 e 5 anos	Mais de 5 anos
Tempo de uso de computador	Nunca usou	Menos de 6 meses	Entre 6 meses e 2 ano	Entre 2 e 5 anos	Mais de 5 anos
Tempo de uso da Internet	Nunca usou	Menos de 6 meses	Entre 6 meses e 2 ano	Entre 2 e 5 anos	Mais de 5 anos
<b>Liste suas tarefas no cargo</b>	<b>Indique a frequência de realização</b>				
	Diária	Semanal	Mensal	Rara	Outra _____
	Diária	Semanal	Mensal	Rara	Outra _____
	Diária	Semanal	Mensal	Rara	Outra _____
	Diária	Semanal	Mensal	Rara	Outra _____
	Diária	Semanal	Mensal	Rara	Outra _____
	Diária	Semanal	Mensal	Rara	Outra _____
	Diária	Semanal	Mensal	Rara	Outra _____
	Diária	Semanal	Mensal	Rara	Outra _____
<b>Qualidade do ambiente de trabalho</b>					
Iluminação	Muito ruim	Ruim	Razoável	Boa	Ótima
Ruído	Muito alto	Alto	Aceitável	Pouco	Quase nenhum
Postura ao trabalhar	Muito ruim	Ruim	Aceitável	Boa	Ótima
Risco ao trabalhar (venenos, explosões...)	Muito alto	Alto	Pouco	Baixo	Nenhum

## IX.4 Protótipos

Levando em conta a necessidade de mostrar algo menos abstrato do que modelos aos clientes, qualquer processo de desenvolvimento atual tem a tendência de acelerar a criação da interface com o usuário, que é representada pelas telas ou formulários com o qual o usuário interage e os relatórios e outras formas de aviso que recebe do sistema.

Um **protótipo** é uma implementação simplificada do sistema, podendo inclusive ser descartável, com diferentes finalidades, como validar um modelo de interface ou um modelo de funcionamento ou ainda um algoritmo. Normalmente protótipos são construídos utilizando a ferramenta em que o sistema está ou será desenvolvido (podendo servir inclusive para validar essa ferramenta) e apresentam algum comportamento, mesmo que simulado.

Um **mock-up** é uma representação da interface que não cumpre nenhuma finalidade a não ser demonstrar a uma proposta para a aparência final do sistema, sem a capacidade de simular seu comportamento (a não ser, possivelmente, a navegação entre telas). Um mock-up não precisa ser feito com uma ferramenta de programação, podendo ser feito com ferramentas de desenho, como os softwares Visio® ou SmartDraw®. Um **protótipo de baixa fidelidade** é um mock-up feito a mão da interface, basicamente um conjunto de desenhos, com a finalidade de demonstrar a aparência básica e simular, manualmente, o comportamento do sistema. Um **protótipo de alta fidelidade** é um mock-up feito de forma a se assemelhar ao software final, sendo normalmente construído na linguagem de desenvolvimento ou em uma ferramenta com resultados similares. Protótipos de alta fidelidade são executados pelo computador.

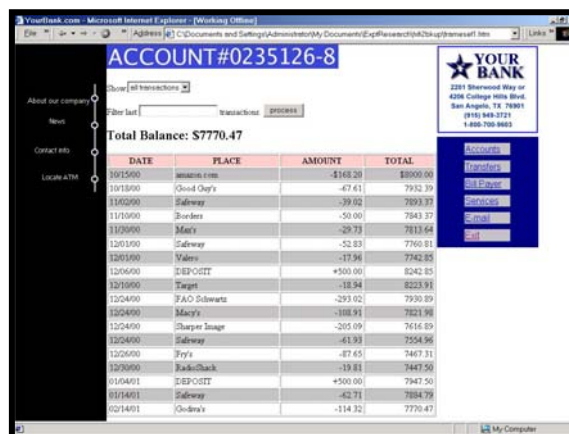
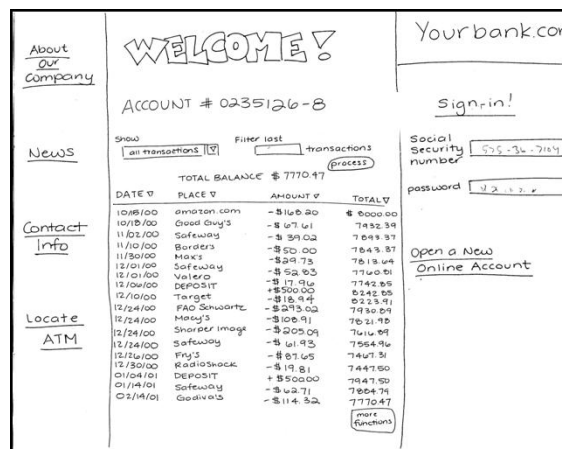


Figura 89. Um protótipo de baixa fidelidade (acima) e um protótipo de alta fidelidade correspondente (Landay).

O uso de protótipos desde o início do desenvolvimento permite ao desenvolvedor experimentar diferentes abordagens e discuti-las com o usuário, obtendo feedback mais cedo durante o ciclo de desenvolvimento e também antecipando a indicação, pelo usuário, de erros de análise e projeto.

Segundo McConnell<sup>67</sup>, a prototipagem de interface com o usuário tem um bom potencial de redução do tempo do projeto, diminui o risco de insucesso e é um excelente fator para o sucesso a curto (primeira-vez) e longo prazo de um sistema. Os maiores riscos associados a essa técnica, ainda segundo McConnell, seriam a possibilidade de perder tempo fazendo melhorias de baixa importância no protótipo, além de outros riscos de qualidade historicamente associados aos protótipos, como a utilização no código final de código que foi criado para ser jogado fora, e conseqüentemente sem seguir regras e padrões de desenvolvimento.

O uso de protótipos pode ser um motivador para a participação dos usuários no sistema, diminuindo o antagonismo e aumentando a cooperação entre desenvolvedores e futuros usuários. A visão de um algo já realizado melhora a moral do projeto como um todo, porém pode trazer como risco a confiança demasiada e um otimismo exagerado em relação a prazos, como uma decepção proporcional a seguir.

Com certeza, protótipos facilitam em muito a validação de sistemas, principalmente de novos sistemas, onde há certo grau de exploração da solução mais adequada. Além disso, podem facilitar a encontrar funções desnecessárias ou funções esquecidas, principalmente com usuários já acostumados com sistemas semelhantes.

Protótipos podem ser criados de forma parcial. Por exemplo, se um sistema exige a manutenção (incluir, excluir e alterar) de várias entidades, como aluno, professor e funcionário em um sistema acadêmico, a prototipagem de uma dessas interações pode servir de forma de validação para todas as outras.

#### **IX.4.1 Protótipos de Baixa Fidelidade & StoryBoarding**

A construção de protótipos de baixa fidelidade é algo que muitos fazemos sem nem mesmo nos darmos conta. Um protótipo de baixa fidelidade (PBF) é um desenho, provavelmente feito a mão, usado por uma pessoa para demonstrar o comportamento do sistema para outra pessoa. Esse tipo de protótipo é totalmente diferente do protótipo tradicional proposto normalmente e que inclui a construção de um software.

PBF podem ser desenhados em quadros negros, quadros brancos, sobre papel, sobre transparência, em “tablet PCs”, ou qualquer outra forma, incluindo a união das citadas. Podem ser feitos a mão livre ou com auxílio de réguas, gabaritos ou outros materiais de desenho. Pode ser preto e branco ou colorido. Podem usar materiais pré-desenhados, como *frames* de janelas, ou serem feitos a partir de colagens.

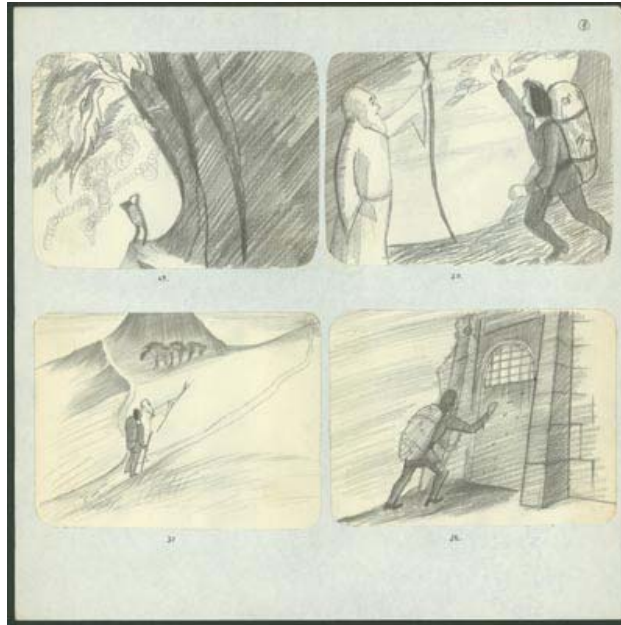
Poucos são as habilidades necessárias para desenhar um PBF. Seu custo também é baixo e o ciclo de interação com o usuário muito rápido. PBFs podem ser desenhados junto com o usuário, inclusive em uma reunião de JAD. Um dos efeitos psicológicos

---

<sup>67</sup> McConnell, Steve. Rapid Development: Taming Wild Software Schedules. Microsoft Press, Redmond, Washington, 1996

interessantes é que o usuário, não vendo uma implementação, fica mais disposto a propor mudanças, pois não vê nenhum custo ou esforço associado às mesmas, o que, na prática, é verdade.

A principal característica de PBF é que eles são explicados e “executados”, ou melhor, “encenados”, manualmente. Partes do protótipo podem ser desenhadas com detalhes, enquanto outras podem ser só indicadas. Alguns objetos podem ser cortados em um tamanho apropriado, como caixas mensagens e menus. A construção dessa encenação é semelhante à técnica de *storyboarding* usada no cinema.



**Figura 90. Um storyboard para um desenho animado indica como deve ser a seqüência de cenas. Original encontrado em (<http://www.surrart.ac.uk/arc/archive/digitation.html>)**

Como o comportamento de um PBF é executado por uma pessoa, em uma estratégia conhecida como “Mágico de Oz”<sup>68</sup>, sua documentação não é tão simples. A partir da encenação do uso da interface é feita uma avaliação, que pode levar a necessidade de aceitá-la, melhorá-la ou até mesmo iniciar tudo do início.

---

<sup>68</sup> Na história, uma pessoa manipula o “Mágico de Oz” atrás de uma cortina.

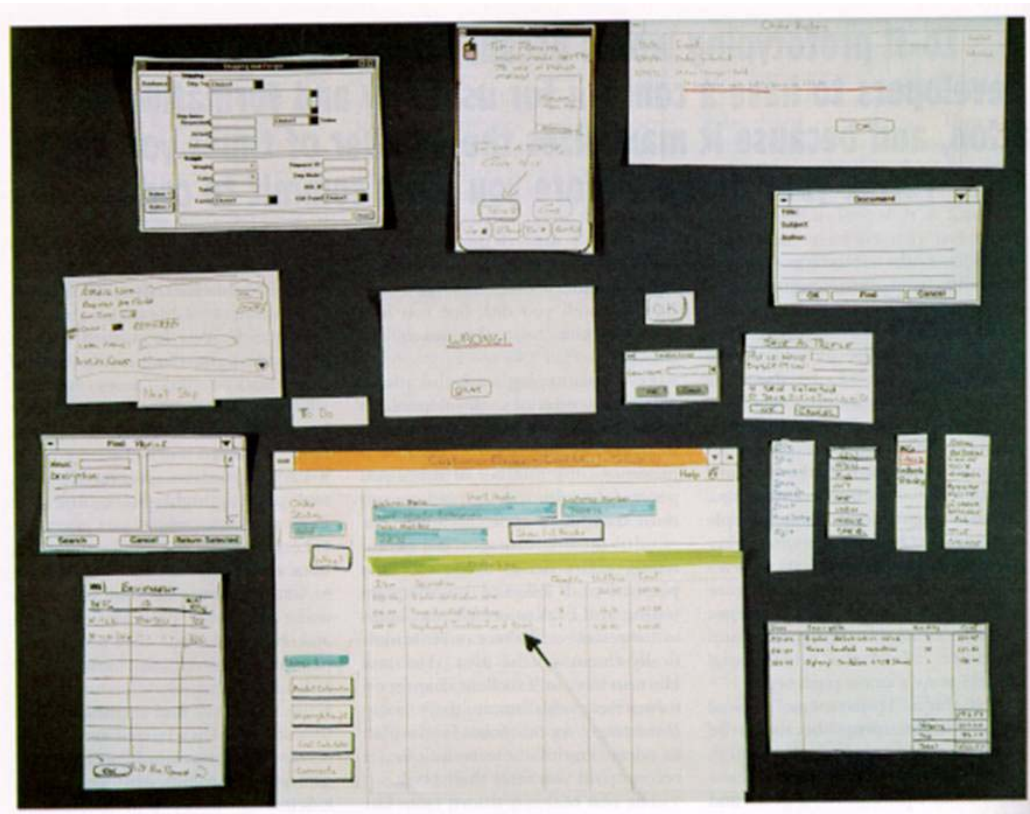


Figura 91. Exemplo de partes de um protótipo de baixa fidelidade. Algumas janelas foram desenhadas a mão, outras em um computador. O mouse é um ponteiro que pode ser mexido com a mão (Landay).

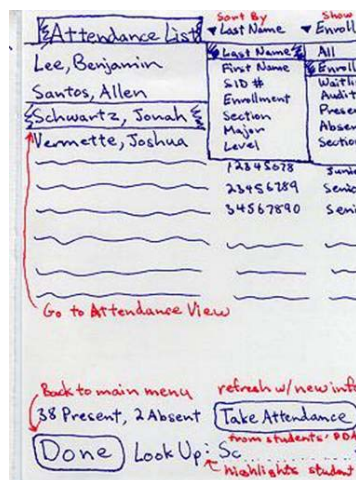


Figura 92. Um protótipo de baixa fidelidade de uma tela, indicando operações possíveis (em vermelho) (Landay).



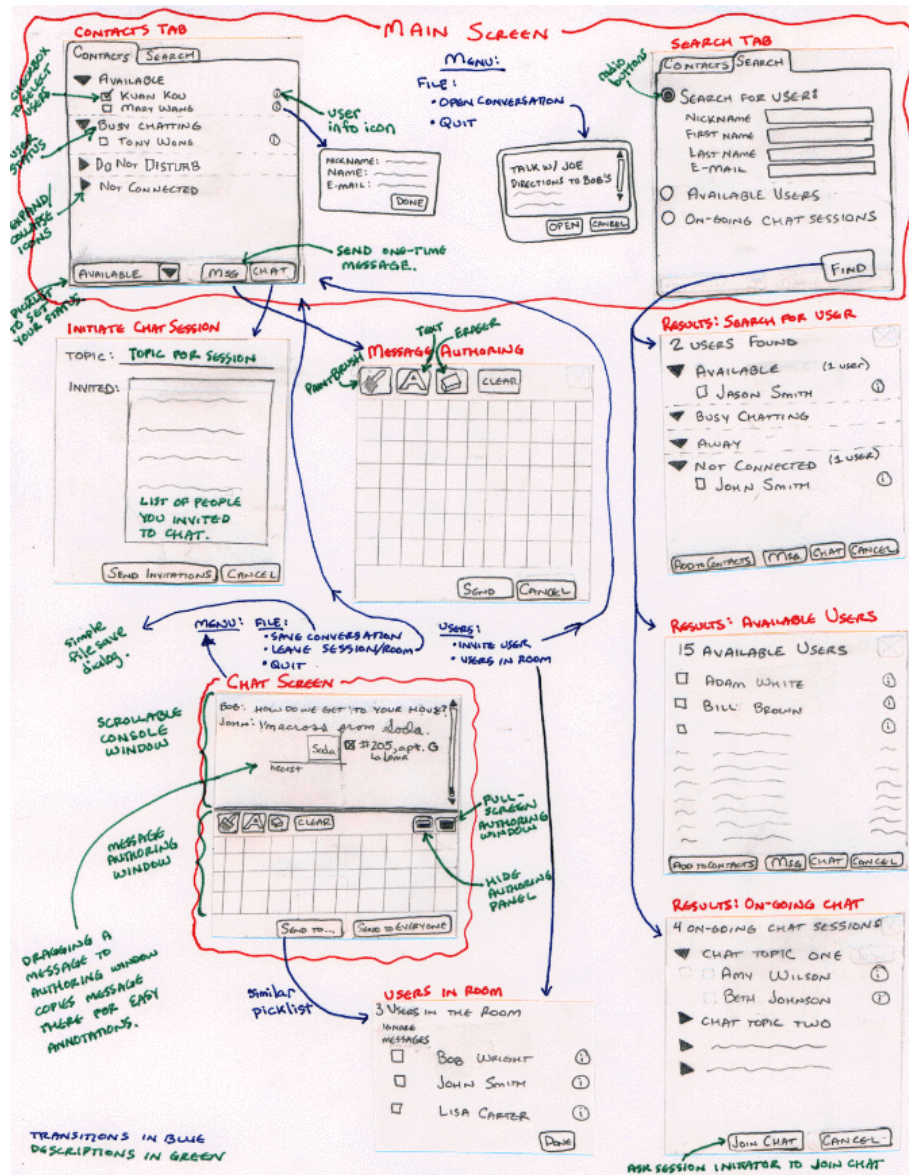


Figura 93. Vários protótipos de baixa fidelidade em uma folha compoendo um **storyboard** (Landay).

### Softwares para prototipagem de baixa fidelidade

Com computadores se tornando presente em todos os ambientes, é possível pensar em construir protótipos de baixa fidelidade diretamente no computador e não apenas com papel. O software DENIM (<http://guir.cs.berkeley.edu/projects/denim/>) foi projetado para isso. Ele permite que um grupo de pessoas desenhe a interface a mão e associe algum comportamento que é executado pelo automaticamente. Na verdade, o software é mais adequado a ambientes que usam canetas (como um *tablet PC*) do que mouse.

Uma das vantagens de usar um software desse tipo é que ele pode executar a interface automaticamente. DENIM fornece essa possibilidade, por meio do comando Run. No futuro, DENIM deve permitir a identificação automática dos *widjets* mais comuns em ambientes de desenvolvimento.

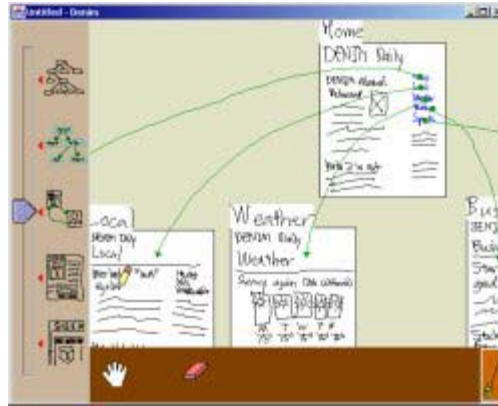


Figura 94. O software DENIM, para desenho de protótipos de baixa fidelidade e construção de storyboards (Landay).

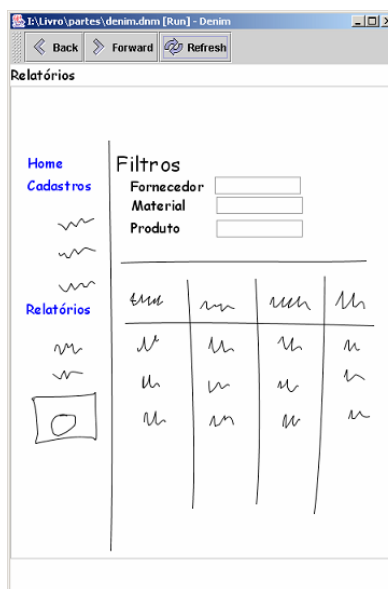


Figura 95. Uma página sendo executada em DENIM (Landay).

## IX.5 Modelos de Navegação

Modelos funcionais e de dados também não representam uma característica importante que é o comportamento esperado do sistema pelos usuários. Novamente, alguns autores defendem que essa modelagem deve ser feita na fase do projeto, e provavelmente ela realmente é deve ser detalhada nessa fase, porém a ausência dessa modelagem na fase de análise pode levar a graves erros de estimativa de custos, pois comportamentos distintos para alcançar a mesma funcionalidade podem ter custos de desenvolvimento muito diferenciados.

Por exemplo, imaginem um sistema cuja finalidade é registrar as multas de trânsito aplicadas por policiais e gerar alguns relatórios. A especificação funcional é simples, o principal evento é “policia! envia auto de infração”. Porém isso pode ser recebido de várias formas: por meio de um digitador que lê o auto de infração, por meio da digitalização e reconhecimento de caracteres ou recebendo arquivos gerados por

PDA's, como já acontece em algumas cidades. O custo de implementação de cada uma dessas formas é muito diferente, além do que um sistema pode exigir que todas essas formas sendo implementadas, o que aumenta ainda mais o custo.

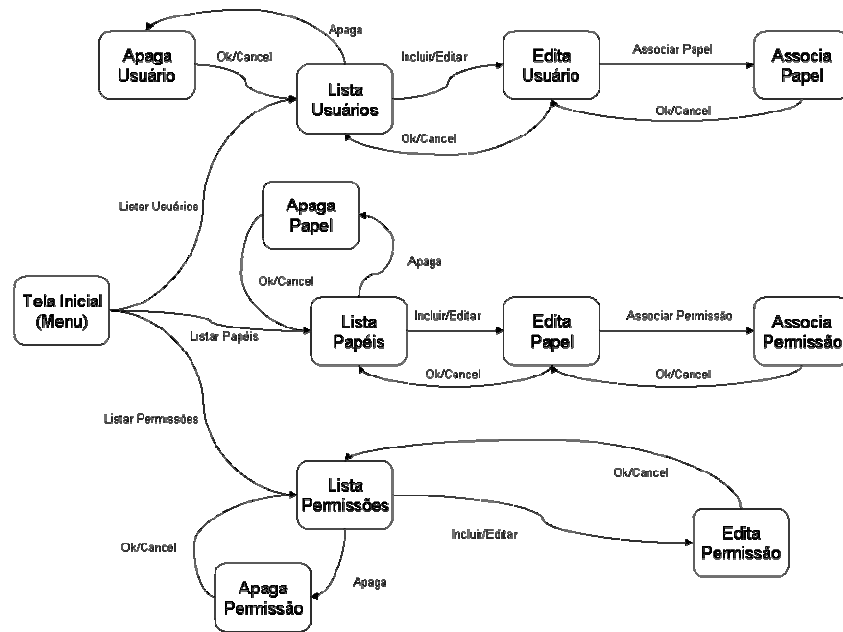
Por isso, é importante criar um modelo do comportamento do sistema ainda na análise, respondendo, de certa forma, como as funções serão executadas, mesmo que em um nível muito alto de abstração. Esse modelo pode ser muito simples, porém não existem ferramentas padronizadas para criá-lo, o que complica um pouco o seu desenho. A seguir veremos algumas propostas já utilizadas com diferentes graus de sucesso em sistemas distintos.

### **IX.5.1 O Diagrama de Navegação de Telas**

Esse é um diagrama muito simples, semelhante a um diagrama de transições estado, que mostra quais são as possíveis navegações entre as telas de um sistema. Vários autores sugerem notações similares.

O exemplo da figura a seguir utiliza algumas regras arbitrárias, que são comuns em modelos de navegação. Não se faz distinção entre a funcionalidade, apenas na navegação. Assim, por exemplo, de “Apaga Usuário” navegamos para “Lista Usuário” qualquer que seja a opção (“OK” ou “Cancel”). Nesse diagrama não indicamos a necessidade de selecionar elementos para apagar ou editar, o que pode ser feito em alguma outra notação. O importante aqui é ter uma idéia de quantas “telas abstratas” serão necessárias e ter uma noção do comportamento do sistema (isso porque o cadastro de um usuário, por exemplo, pode exigir várias telas, o que só seria visto no projeto). Outra coisa que podemos notar é que só está sendo considerado, nesse modelo, o comportamento normal. Outros modelos, ou um refinamento deste modelo, podem permitir a inclusão de telas de ajuda ou mensagens de erro, por exemplo.

Entre as vantagens de construir um diagrama de navegação estão a sua simplicidade e informalidade. Apesar de abstratos, podem ser usados em discussões com o usuário com certa facilidade. Além disso, servem também para dar aos desenvolvedores uma visão geral do comportamento do sistema.



**Figura 96.** Um diagrama de navegação de telas, feito com o software PowerPoint.

# Capítulo X. Qual o Tamanho do Software



*User Friendly by Illiad*

Como medir software?

Preço

Esforço

Tamanho

Medidas diretas

Medidas indiretas

Pontos de Função

COCOMO

## X.1 Qual o tamanho do sistema?

Uma das perguntas mais frequentes em desenvolvimento do software é “qual o tamanho do sistema?” Essa pergunta pode vir sob várias formas:

- Qual o preço do sistema?
- Qual o custo do sistema?
- Qual o esforço para desenvolver o sistema?
- Quantas pessoas serão necessárias para fazer esse software?
- Em quanto tempo o sistema ficará pronto?
- Quantas linhas de código tem o sistema?
- Quantos pontos de função tem o sistema?
- Qual o tamanho do sistema?
- Que recursos são necessários para o sistema?

Nesse capítulo veremos como responder essas questões.

## X.2 Preço e Custo

Quando queremos saber o preço ou o custo do sistema a preocupação é econômica. No nosso contexto vamos separar os conceitos de custo e preço: custo é quanto se gasta para desenvolver o sistema, preço é quanto se cobra pelo sistema. Apesar de existir uma relação entre preço e custo, cabe aos responsáveis pelo desenvolvimento prever, acompanhar e calcular o custo do sistema (incluindo muitas vezes não só o desenvolvimento, mas também o custo de implantação, operação e manutenção). O preço, porém, é determinado por uma relação comercial entre cliente e fornecedor<sup>69</sup>.

Fica claro então que a principal e primeira pergunta que o desenvolvedor deve fazer é “quanto o sistema custa para ser desenvolvido”. A partir desse valor, pode-se começar, se necessário, a pensar em um preço a ser cobrado.

Para sistemas de informação o **principal fator de custo é o gasto com pessoal**<sup>70</sup>. Assim, o custo do software é diretamente ligado à quantidade de pessoas envolvidas no seu desenvolvimento e ao tempo que elas dedicam a essa atividade.

## X.3 Esforço e Tempo de Desenvolvimento

Tendo em vista que o principal fator de custo no desenvolvimento de software é o gasto com pessoal, uma das principais preocupações da Engenharia de Software é determinar qual será a quantidade de pessoas e o tempo por elas dedicado a um projeto. Para isso usamos o conceito de **Esforço** que representa a quantidade de trabalho realizado, medido em pessoa-mês<sup>71</sup>, ou seja, o trabalho feito por uma pessoa em um mês.

Assim, podemos dizer que um sistema precisa de 4 pessoas-mês para ser realizado, ou seja, que uma pessoa trabalhando 4 meses ou 4 pessoas trabalhando um mês. Acontece que sistemas de informação são um pouco como bebês: não podemos ter a gestação de um bebê com nove mães em um mês. Na verdade, Boehm achou uma relação entre o esforço necessário e o tempo necessário para fazer um sistema, e conseqüentemente o tamanho médio da equipe.

Obviamente, o que fazemos no início do projeto é tentar prever o esforço necessário para realizá-lo com sucesso, e conseqüentemente o tempo necessário para completá-lo. Essas previsões são mais ou menos acertadas de acordo com vários fatores, incluindo a maturidade da empresa, a disponibilidade de dados históricos, a experiência dos responsáveis pela predição e a capacidade intrínseca do modelo utilizado na predição.

Um dos principais modelos de predição do esforço necessário é o COCOMO II. Esse modelo, baseado em equações matemáticas derivadas a partir da análise estatística

---

<sup>69</sup> Nessa relação o cliente tenta pagar o menor preço possível para o sistema e o fornecedor tenta cobrar o maior preço possível. Tanto fornecedores quanto clientes devem evitar fazer um acordo com risco de futuro “arrependimento”.

<sup>70</sup> Outros custos adicionais comuns são equipamento e software. Mesmo quando já possuímos o equipamento e o software temos que nos lembrar de amortizar o seu custo original de alguma forma entre os vários projetos.

<sup>71</sup> Em sistemas menores a medida é pessoa-hora.

de casos reais é bastante completo. No COCOMO II, que apresentaremos de forma reduzida a seguir, o esforço é calculado a partir de uma previsão do tamanho do software.

## **X.4 O Tamanho do Software**

Até agora descobrimos que para saber o preço de um software precisamos saber seu custo, e que para saber seu custo precisamos saber o esforço necessário para desenvolvê-lo. Finalmente chegamos à questão mais difícil: como prever o tamanho do software, de forma a determinar o esforço necessário?

A primeira questão a responder é como medir o tamanho do software. Vários autores já discutiram amplamente sobre essa questão. Atualmente duas formas são aceitas internacionalmente para essa medida: milhares de linhas de código fonte (KSLOCs, a partir do acrônimo em inglês – kilo source lines of code) e pontos de função (PFs ou FPs, do inglês). Ambos possuem defensores e detratores, vantagens e desvantagens mais óbvias ou mais sutis.

Uma linha de código, no contexto do COCOMO II, deve ser uma linha executável, ou uma declaração ou uma diretiva para o compilador, que não tenha sido gerada com um gerador automático e que tenha sido feita pela empresa. Linhas de código, obviamente, só podem ser contadas após a implementação do software.

Um ponto de função é uma medida abstrata que representa a funcionalidade entregue ao cliente, em função das interfaces do sistema com o usuário, com outros sistemas e ainda com a informação vista pelo cliente. Pontos de função são tratados detalhadamente em um capítulo posterior. Pontos de função podem ser contados a partir desde a especificação do sistema até sua implementação final.

Essas duas medidas podem ser convertidas uma na outra, por meio de estatísticas globais ou específicas da empresa.

A vantagem de KSLOC é que podemos simplesmente contá-las, a principal crítica é que medir produtividade ou custo por KSLOCs pode provocar distorções, pois bons programadores deveriam utilizar menos linhas de código para realizar uma função.

Pontos de função podem ser criticados por ser uma medida abstrata demais, porém permitem a comparação de sistemas de forma direta.

## **X.5 Uma visão reduzida do modelo COCOMO II**

COCOMO (Constructive Cost Model) é um método de previsão de custos de software. Atualmente é possível conseguir gratuitamente um software COCOMO baseado em pontos de função, o que facilita o cálculo de custos de projeto de sistemas de informação.

A relação entre o tempo de desenvolvimento e o esforço necessário, que apresentamos em sua forma completa a seguir, é parte importante do modelo COCOMO<sup>72</sup>:

$$TDEV = [C \times (PM_{NS})^{(D+0.2 \times (E-B))}] \times \frac{SCED\%}{100}$$

**Equação 1. Tempo de Desenvolvimento calculado pelo modelo COCOMO**

Nessa fórmula  $PM_{NS}$  é a quantidade nominal de pessoas/mês<sup>73</sup>, SCED é a compressão necessária no tempo de desenvolvimento, B,C e D são constantes calibráveis e E é um coeficiente calculado a partir de coeficientes de escala.

$$E = B + 0.01 \times \sum_{j=1}^N SF_j$$

**Equação 2. Cálculo de E, a partir dos coeficientes de escala.**

Na fase inicial do projeto, os coeficientes de escala são 5 (N=5), e representam a precedência do sistema, a flexibilidade do projeto, o risco do projeto e da arquitetura, a coesão da equipe e a maturidade do processo. Em situação normal para todos os casos, o valor de  $E$  é igual a 0.2807.

Ficaremos então com uma fórmula simplificada, capaz de atender plenamente os objetivos desse livro. Caso necessário, o leitor pode recorrer ao livro “Software Cost Estimation With COCOMO II” ou ao web site:

<http://sunset.usc.edu/research/COCOMOII/index.html>

$$TDEV = [3.67 \times (PM_{NS})^{0.32}]$$

**Equação 3. Equação simplificada que pode ser usada para ter uma previsão do tempo de desenvolvimento a partir do esforço necessário em pessoas-mês, baseada no modelo COCOMO II.**

Fica então a pergunta: como descobrir o esforço necessário. O modelo COCOMO II também nos fornece uma fórmula, desta vez baseada na quantidade de linhas de código previstas para o software. Desta vez forneceremos logo a fórmula simplificada:

$$PM = 2.94 \times MLDC^{0.28}$$

**Equação 4. Equação simplificada de cálculo do esforço, onde MLDC significa milhares de linhas de código.**

A	2.94
B	0.91

<sup>72</sup> Atualmente em sua segunda versão (COCOMO II)

<sup>73</sup>  $PM_{NS}$  é um valor de pessoas-mês sem considerar esforços de tradução automática e ainda sem considerar efeitos da necessidade de acelerar o desenvolvimento



C	3.67
D	0.28

**Tabela 29. Valores atuais de A, B, C e D.**

### **X.5.1 Distribuição do Esforço**

Uma questão importante na previsão do esforço a ser feito para o desenvolvimento de um software é como esse esforço será distribuído nas fases do processo de desenvolvimento. O COCOMO 81 tratava desse problema com certo detalhe que não foi continuado no COCOMO II em 2000. Os dados do COCOMO 81, porém, foram reaproveitados com as metodologias do COCOMO II.2000 para nos fornecer valores provisórios com que trabalhar.

Para um processo seguindo o modelo em Cascata, os seguintes resultados são esperados:

<b>Fase</b>	<b>Esforço %</b>	<b>Tempo %</b>
Planos e Requisitos	7 (2-15)	16-24 (2-30)
Projeto do Produto	17	24-28
Programação	64-52	56-40
Programação: Projeto Detalhado	27-23	
Programação: codificação e Teste de unidade	37-29	
Integração e Testes	19-31	20-32
Transição	12 (0-20)	12.5 (0-20)

**Tabela 30 Divisão do esforço nas fases de desenvolvimento do software segundo Boehm**

## **X.6 Análise de Pontos de Função**

Uma das mais importantes informações que podemos ter sobre um produto de software é o esforço necessário para desenvolvê-lo. Usualmente definimos este esforço pela quantidade de homens/hora ou homens/mês necessários para desenvolver o produto. A principal utilização desta informação tem como objetivo **prever e acompanhar** o esforço que será gasto no desenvolvimento de um produto de porte semelhante.

Porém, como saber qual o porte de um produto de software?

Para isso foram inventadas várias medidas, algumas delas arbitrárias, outras fáceis de medir. É comum que digamos o tamanho de um software pela quantidade de linhas de código, pelo número de horas gasto para desenvolvê-lo ou pelo custo final do mesmo.

Porém, até 1979 não tínhamos uma boa medida do tamanho do software em função da sua funcionalidade **como vista pelo usuário**. Apenas nessa data, Albrecht [35] apresentou uma medida conhecida como **Pontos de Função**, cujo objetivo era servir como avaliador e preditor do tamanho de um sistema.

Um Ponto de Função (PF) é uma medida abstrata<sup>74</sup> e relativa que conta “o número de funções de negócio entregues ao usuário”. Um relatório simples, por exemplo, pode medir “4 Pontos de Função”. Da mesma forma que um “metro” ou “um litro”, Pontos de Função só fazem sentido quando comparados com um padrão. Assim, um sistema com 1.000 PF entrega o dobro de funcionalidade de que um sistema com 500 PF. Com o tempo, aprendemos a ter uma idéia absoluta do tamanho de um sistema a partir da contagem de seus PFs.

A maneira padronizada de contar pontos de função é fornecida pelo International Function Points User Group (IFPUG), que fornece aos seus associados um manual contendo detalhes do que deve e do que não deve ser contado. Esse capítulo é apenas uma introdução ao método, descrevendo uma forma levemente simplificada da metodologia oficial, servindo para fazer cálculos aproximados do número de pontos de função de um sistema.

## **X.6.1 Procedimento de Contagem**

Para contar pontos de função, primeiramente identificamos as funções de negócio como percebidas pelo usuário, dividindo-as em 5 grupos, agrupados em 2 tipos:

### **X.6.1.1 Funções Transacionais**

- **Saídas externas (SE ou EO)** são informações de negócio que o usuário final pode receber, representando relatórios, telas e mensagens de erro como um todo e não em suas partes individuais;

- **Consultas externas (CE ou EQ)**, que são saídas simples e imediatas, sem alteração na base, usualmente caracterizáveis por chaves simples de consulta.

- **Entradas externas (EE ou EI)**, que são informações de negócio enviadas para o sistema pelo usuário final, cruzando a fronteira da aplicação e mantendo um ALI.

### **X.6.1.2 Funções de Dados**

- **Arquivos lógicos internos (ALI ou ILF)**, que contém os dados permanentes, relevantes para o usuário e mantidos e utilizados pelo sistema. O sistema cria, altera e apaga esses dados.

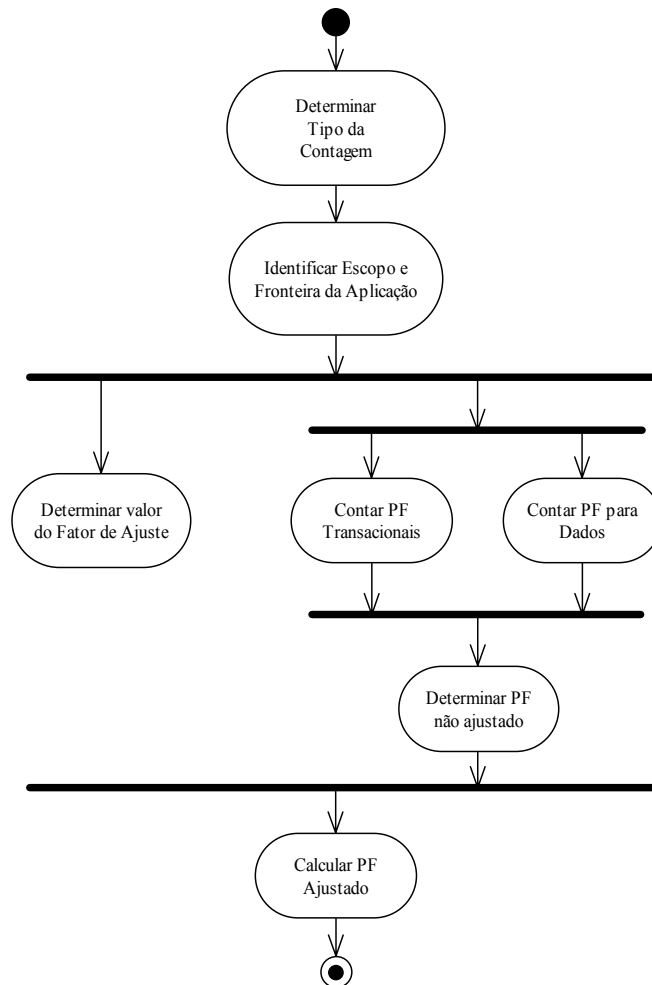
- **Arquivos de interface externos (AIE ou EIF)**, que contém dados permanentes e relevantes para os usuários, guardados em algum lugar por outra aplicação, mas referenciados pela aplicação em questão. Outro sistema mantém esses dados.

O segundo passo é determinar a **complexidade** de cada função de negócio. A complexidade fornece um peso para cada função de negócio encontrada. O somatório do número de funções multiplicadas pelo seu peso fornece o **número básico de pontos de função**. Esse número é um indicador preliminar do tamanho do sistema.

---

<sup>74</sup> O leitor deve perceber que o metro e o litro também são medidas abstratas, apenas temos muita experiência com eles.

Finalmente, no terceiro passo, o número básico de pontos de funções é corrigido em função de fatores que aumentam ou diminuem a complexidade do sistema.



**Figura 97. Procedimento de contagem dos pontos de função, adaptado de XXX.**

### X.6.2 Identificando Funções de Negócio

Para identificar as funções de negócio devemos partir de algum documento que aponte as funções **aprovas** e pelo usuário e **úteis** para o negócio. Não devem ser contadas funções necessárias por causa da tecnologia aplicada. Basicamente, só é cobrado o que o usuário pode ver e está disposto a pagar. Também é importante que as funções de negócio sejam cobradas **como o usuário as percebem**. Isto significa que não interessa se estamos usando um ou vinte arquivos para guardar uma informação, mas sim de quantas formas o usuário pode acessar essa informação.

Além disso, devemos identificar as funções seguindo certa ordem. A ordem é importante porque encontrar um tipo de função de negócio ajuda a encontrar as funções de outro tipo. Assim, em um sistema novo devemos usar a ordem: saídas, consultas, entradas, arquivos e interfaces. Por outro lado, em um sistema já existente devemos usar a ordem arquivos, interfaces, saídas, consultas e entradas.

Para serem contados as funções devem:

- Beneficiar claramente o usuário,
- Ser especificamente aprovado pelo usuário e
- Influenciar em algum grau mensurável o projeto, desenvolvimento, implementação e suporta à aplicação.

No manual da IFPUG podemos encontrar vários exemplos que nos permitem dirimir as dúvidas de como realizar a contagem. A técnica é simples, porém difícil de dominar.

#### **X.6.2.1 Identificando Saídas**

Para contar as saídas é necessário contar todas as informações que o sistema gera para o ambiente de forma procedimental. Tipicamente, saídas são relatórios impressos, telas, janelas e arquivos gerados para outras aplicações.

Deve ser contadas como saídas distintas cada formato utilizado. Basicamente, se for necessário fazer outro procedimento para produzir a informação, contamos como uma saída distinta. Também contamos cada tipo de lógica utilizada para fazer gerar a informação. Assim, se um relatório de vendas contém as vendas por vendedor e por loja, contaremos como duas saídas, pois são necessários procedimentos lógicos distintos para calcular cada um desses valores. Linhas de sumário e total, porém, não são contadas como novas saídas.

Uma saída externa pode ter uma “parte de entrada”, para, por exemplo, selecionar os registros necessários em um relatório, usando alguns campos como filtro. Essa “parte entrada” **não** é contada a parte, já está considerada nessa contagem.

#### **X.6.2.2 Identificando Consultas**

Consultas são, na prática, saídas simplificadas. Normalmente utilizadas para achar informações para modificá-las ou apagá-las. São sempre no monitor, não existe uma consulta em relatório de papel.

Uma consulta **não** pode calcular nenhum valor. Em caso de cálculo de qualquer valor, temos uma saída.

#### **X.6.2.3 Identificando Entradas**

Entradas permitem adicionar, modificar e apagar informações. Se uma tela permite estas 3 funções, são contadas 3 entradas. Normalmente as funções de modificar e apagar ainda exigem consultas correspondentes para achar a informação que será alterada.

Um comando específico para o sistema executar algo é uma entrada.

Mensagens de erro que fazem parte de um processo não são contadas isoladamente, mas sim como um DET. Por exemplo, se você esquecer de colocar um campo obrigatório e receber uma mensagem de erro, não deve contar uma saída a mais, e sim essa mensagem como um DET da entrada ou consulta.

Mensagens de notificação, por outro lado, são processos elementares e devem ser contados como uma saída a parte. Por exemplo, ao tentar comprar um produto que não existe e receber uma mensagem com essa notificação foi feito um processamento, que é contado como uma saída externa adicional.

Forma de Processamento Lógico	Tipo de Função Transacional		
	EI	EO	EQ
Realiza validação dos dados	P	P	P
Realiza cálculos ou fórmulas matemáticas	P	O*	N
Converte valores equivalentes	P	P	P
Filtra dados e seleciona usando critérios específicos para comparar múltiplos conjuntos de dados	P	P	P
Analisa condições para determinar qual é aplicável	P	P	P
Altera ou inclui ao menos um ILF	O*	O*	N
Referencia ao menos um ILF ou EIF	P	P	O
Recupera dados ou informação de controle	P	P	O
Cria dados derivados	P	O*	N
Altera o comportamento do sistema	O*	O*	N
Prepara e apresenta informação fora das fronteiras do sistema	P	O	O
É capaz de aceitar dados ou informação de controle que entra pela fronteira da aplicação	O	P	P
Reordena ou reorganiza um conjunto de dados	P	P	P

**Tabela 31. Tipo de Função Transacional e lógica de processamento correspondente (P=possível, O=obrigatório, O\*=obrigatório ao menos um, N=não é possível).**

#### **X.6.2.4 Identificando Arquivos Internos**

Arquivos representam um grupamento lógico requerido pelo usuário. Podem incluir uma ou mais tabelas ou entidades.

Esse é uma das partes mais difíceis da contagem de pontos de função, pois devemos separar o que o usuário pensa do modelo que criamos. Nosso modelo muitas vezes usa vários grupos de dados, ou tabelas, ou entidades, para modelar algo que o usuário vê como um conceito único. Mesmo na modelagem conceitual, a tendência do analista é incluir entidades que o usuário não “vê” naturalmente.

Um Tipo de Elemento de Registro (RET, Record Element Type) é um subgrupo de elementos de dados dentro de um arquivo ou interface. Na prática, em um modelo de dados, um arquivo do usuário (ILF) é composto de um ou mais objetos do modelo.

Outra característica difícil de contar é que cada forma de acesso a um arquivo lógico conta novamente. Assim, por exemplo, se o usuário exige acessar um automóvel tanto por sua placa quanto por seu número do chassi, temos 2 arquivos lógicos para contar.

Exemplos: Uma nota fiscal é um arquivo lógico, com dois RETs: dados da nota, item de nota.

#### **X.6.2.5 Identificando Arquivos Externos**

Arquivos Externos representam um agrupamento lógico requerido pelo usuário. Podem incluir uma ou mais tabelas ou entidades.

Arquivos Externos são mantidos por outras aplicações. Arquivos importados contam também como Entrada Externa, arquivos exportados contam também como Consulta Externa ou Saída Externa.

#### **X.6.2.6 Identificando Itens de Dados (DETs)**

Itens de dados ou elementos de dados (DETs) campos únicos, reconhecidos pelos usuários, desconsiderando-se recursão e repetição. DETs também podem invocar ações. Exemplos de DETs são:

- Em Entradas externas
  - campos de entrada de dados
  - mensagens de erro
  - valores calculados que são guardados
  - botões
  - mensagens de confirmação
  - campos repetidos contam apenas como um DET
- Em Saídas Externas
  - Campos em relatório
  - Valores calculados
  - Mensagens de erro
  - Cabeçalhos de coluna que são gerados dinamicamente em um relatório
- Em Consultas Externas (além dos anteriores que se enquadrem)
  - Campos usados em filtros de procura
  - O clique do mouse

Em GUIs, botões onde só se pode fazer uma seleção entre muitas (normalmente *radio buttons*) devem ser contados como um DET apenas. Já *check boxes* são normalmente contadas uma a uma. Botões de comando devem ser contados como um elemento de dados levando em conta o fato de executarem uma função.

### X.6.3 Determinando a Complexidade

Para todos os tipos de função de negócios, é importante determinar o número de itens de dados referenciados.

Para saídas, consultas e entradas, devemos contar o número de arquivos acessados.

Para arquivos e interfaces, devemos contar o número de RETs e o número de campos de dados do arquivo.

Saídas	Itens de dados referenciados		
Arquivos Referenciados	1-5	6-19	20+
0 –1	Simples (4)	Simples (4)	Média (5)
2-3	Simples(4)	Média(5)	Complexa(7)
4+	Média (5)	Complexa (7)	Complexa (7)

**Tabela 32. Cálculo da complexidade de saídas externas**

Entradas	Itens de dados referenciados		
Arquivos Referenciados	1-4	5-15	16+
0 –1	Simples (3)	Simples (3)	Média (4)
2	Simples (3)	Média (4)	Complexa (6)
3+	Média (4)	Complexa (6)	Complexa (6)

**Tabela 33. Cálculo da complexidade de entradas externas**

Consultas	Escolher o maior valor obtido entre a parte entrada e a parte saída		
Saída	Itens de dados referenciados		
Arquivos Referenciados	1-5	6-19	20+
0 –1	Simples (4)	Simples (4)	Média (5)
2-3	Simples (4)	Média (5)	Complexa (7)
4+	Média (5)	Complexa (7)	Complexa (7)
Entradas	Itens de dados referenciados		
Arquivos Referenciados	1-4	5-15	16+
0 –1	Simples (3)	Simples (3)	Média (4)
2	Simples (3)	Média (4)	Complexa (6)
3+	Média (4)	Complexa (6)	Complexa (6)

**Tabela 34. Cálculo da complexidade de consultas externas**

Arquivos Internos	Itens de dados referenciados		
RETs	1-19	20-50	51+
1	Simples (7)	Simples (7)	Média (10)
2-5	Simples (7)	Média (10)	Complexa (15)
6	Média (10)	Complexa (15)	Complexa (15)

**Tabela 35. Cálculo da complexidade de arquivos lógicos internos**

Arquivos Externos	Itens de dados referenciados		
RETs	1-19	20-50	51+
1	Simples (5)	Simples (5)	Média (7)
2-5	Simples (5)	Média (7)	Complexa (10)
6	Média (7)	Complexa (10)	Complexa (10)

**Tabela 36. Cálculo da complexidade de interfaces lógicas externas**

#### **X.6.4 As Perguntas**

São 14 as perguntas que devem ser feitas e ajudaram a determinar a quantidade de PF relativa a um sistema. Cada uma deve ser respondida com um número, de 0 a 5, indicando a importância da característica que se pergunta sobre o sistema, da seguinte forma:

- 0 - Não tem influência
- 1 - Influência incidental



- 2 - Influência moderada
- 3 - Influência média
- 4 - Influência significativa
- 5 - Influência essencial em todo o sistema

Para cada pergunta, o padrão IFPUG determina tipos de respostas padronizadas que nos permitem dar a resposta (entre 0 e 5) mais facilmente, como é exemplificado no item 1 (Comunicação de Dados). Foge ao objetivo desse texto fornecer um detalhamento completo do padrão de contagem, que pode ser obtido junto ao IFPUG.

1. Quantas facilidades de comunicação existem para facilitar a transferência ou troca de informação com a aplicação ou sistema?
  - 1.1. Aplicação em batch ou computador isolado: 0
  - 1.2. Aplicação em batch com entrada ou (exclusivo) impressão remota: 1
  - 1.3. Aplicação em batch com entrada e impressão remota: 2
  - 1.4. A aplicação é um *front-end* que necessita de executar coleta de dados ou teleprocessamento para um sistema de fazer o processamento em batch ou de consultas: 3
  - 1.5. A aplicação é mais que um *front-end*, porém só executa um tipo de protocolo de teleprocessamento: 4
  - 1.6. A aplicação é mais que um *front-end* e executa vários protocolos de teleprocessamento: 5
2. Como será tratada a distribuição de dados e processamento?
3. O usuário exige tempos de resposta ou throughput, ou seja, o desempenho é crítico?
4. Quão fortemente é utilizada a plataforma (hardware) onde a aplicação vai ser executada?
5. Qual a frequência das transações (diárias, semanais, altas o suficiente para exigir um estudo de desempenho)?
6. Que percentagem das informações é inserida on-line?
  - 6.1. Se mais de 30% das transações forem entradas de dados interativas, a nota é 5.
7. A aplicação é projetada para eficiência para o usuário final?
8. Quantas ILFs são alteradas por transações on-line?
9. A aplicação tem processamento lógico ou matemático extensivo?
10. A aplicação é desenvolvida para atender um ou muitos tipos de usuários diferentes?
11. Qual a dificuldade de conversão e instalação?
12. Qual a eficiência e grau de automação de inicialização, backup e recuperação?
13. A aplicação foi especialmente projetada, desenvolvida e suportada para funcionar em locais diferentes em diferentes organizações?

14. A aplicação foi especialmente projetada, desenvolvida e suportada para facilitar mudanças?

### X.6.5 Cálculo dos Pfs Finais

Os PF são calculados em etapas:

1. contam-se os números de entradas, saídas, consultas, arquivos e interfaces do sistema;
2. Para cada entrada se determina um grau de complexidade, de acordo com as tabelas complexidade da função (Tabela 32 até Tabela 36), e somando os resultados (Tabela 37) se obtém a **contagem básica de pontos de função**;
3. responde-se a uma série de perguntas, as quais fornecem, cada uma, um valor de 0 a 5 ( $p_i$ ,  $0 \leq i \leq 5$ ),
4. calcula-se o número de pontos de função com a equação:

$$PF = \text{total-de-2} \times (0,65 + 0,01 \times \sum(p_i)).$$

**Devemos notar que se o cálculo de PF for usado para fazer previsões seguindo o método COCOMO, apenas a contagem básica é necessária (só devem ser feitos os passos 1 e 2).**

		Complexidade (passo 2)						
Função	Contagem Total (passo 1)	Simple	×	Média	×	Complexa	×	Total
Saídas Externas			4		5		7	=
Consultas Externas			4		5		7	=
			4		4		6	=
Entradas Externas			3		4		6	=
Arquivos Lógicos Internos			7		10		15	=
Arquivos Lógicos Externos			5		7		10	=
		Total						

**Tabela 37. Guia de cálculo para os pontos de função. Primeiro deve ser feita a contagem total, por tipo de função. Para cada item contado deve então ser determinada a complexidade, o que permitirá encontrar o total (que é o total-de-2).**

### X.6.6 Contando PFs na Análise

São três as principais informações que temos para contar pontos de função a partir da análise que fizemos:



Podemos a partir da Figura 98 criar uma nova tabela de contagem de PFs. A surpresa é que a contagem é diferente, pois entre o modelo essencial e o modelo de implementação aparecem novas funções para o usuário.

Tela	Função	Funções Transacionais		
		Entradas	Saídas	Consultas
...	Lista Itens	-	-	1
Lista Itens	Inserir Item	-	-	-
Lista Itens	Editar Item	-	-	1
Lista Itens	Apagar Item	-	-	1
Apagar Item	OK	1	-	-
Apagar Item	Cancel	-	-	-
Editar Item	OK	1	-	-
Editar Item	Cancel	-	-	-
Inserir Item	OK	1	-	-
Inserir Item	Cancel	-	-	-
<b>TOTAL</b>	<b>-</b>	<b>3</b>	<b>0</b>	<b>3</b>

**Tabela 39. Contagem dos PFs transacionais a partir do modelo de interface.**

#### **X.6.7 Inflação de PFs ao decorrer do projeto**

É normal que o número de PFs se modifique ao decorrer o desenvolvimento do projeto. Isso acontece por dois motivos: alteração dos requisitos e necessidade de implementar mais pontos de função para atender ao comportamento desejado pelo usuário do que estritamente exigido pelo modelo essencial.

Isso pode ser verificado facilmente nas contagens feitas na seção anterior. Enquanto o modelo essencial da Tabela 39 exigia a contagem de 3 entradas e 1 consulta, a interface modelada na Figura 98 para implementar (encarnar) a essência indica 3 entradas e 3 consultas.

É normal que os pontos de função previstos em um início de análise aumentem em até 25% até o final do projeto.

#### **X.6.8 Utilizando Pfs para previsões**

Para utilizar os Pfs para fazer previsões é necessário primeiro conhecer a produtividade em PF/(homem/mês) da equipe que realizará o software. É importante levar em conta o ambiente onde esta equipe trabalha, suas qualidades e defeitos, as ferramentas disponíveis e o tipo de trabalho que realizam afetam o cálculo dessa produtividade. Por exemplo, é comum que equipes de manutenção obtenham produtividade (aparente) muito maior que equipes de desenvolvimento quando é utilizada a medida de pontos de função.

Sabida a produtividade da equipe, basta calcular o número de PFs para o sistema proposto, por exemplo, analisando o resultado da análise essencial, e dividir esse número pela produtividade, obtendo o esforço necessário para implementar o produto.

## **X.7 Ligando COCOMO II e Pontos de Função**

Boehm et al. [XXX] propõe que o método COCOMO II pode ser utilizado para prever o tempo e o esforço necessário para o desenvolvimento de um software a partir da previsão de pontos de função necessários. Para isso, usam uma tabela de conversão entre pontos de função e linhas de código lógicas, apresentada originalmente por Caper Jones [XXX Jones 96].

A previsão se torna simples, basta calcular o número de pontos de função não corrigido (pois o COCOMO II já apresenta correções similares em seu cálculo), converter para linhas de códigos com essa tabela e aplicar as fórmulas.

Caper Jones, porém, alerta que os dados apresentados não são confiáveis, sendo médias que não levam em conta diversas particularidades do desenvolvimento de software por um grupo específico, usando uma arquitetura e ferramentas específicas. Isso pode ser confirmado usando a Tabela 41, fornecida pela empresa QSM, e que mostra não só a média, mas como valores mais altos e mais baixos, o que demonstra variação de até 28 vezes (como em Cobol) entre a menor e maior quantidade de linhas de código por ponto de função encontradas em projetos distintos.

Linguagem	LOC/PF
Assembly (Basic)	320
Assembly (Macro)	213
C	128
C++	53
COBOL (Ansi 85)	91
Crystal Reports	20
Delphi	29
Excel 5	6
Fortran 77	107
Java	53
Linguagem Natural	3200
Linguagens Associativas	64
Linguagens de 1ª Geração	320
Linguagens de 2ª Geração	107
Linguagens de 3ª geração	80
Linguagens de 4ª Geração	20
Linguagens de 5ª Geração	5
Linguagens Estatísticas	32
Linguagens Fortemente Tipadas	91
Linguagens Não-Procedurais	36
Linguagens Orientadas a Objeto	29
Linguagens Orientadas a Problema	71
Linguagens Procedurais	107
Oracle Developer/2000	23
Planilhas Eletrônicas	6
PowerBuilder	16
SQL	13
SQL-Windows	12
Visual C++	34

**Tabela 40.** Tabela de conversão, apresentando número médio de linhas de código equivalentes a 1 ponto de função, adaptado de Caper Jones, previamente encontrada em <http://www.spr.com> e baseada em [XXX Caper 96]<sup>75</sup>.

Linguagem	SLOC/FP			
	Média	Mediana	Menor	Maior
Access	35	38	15	47
Ada	154	-	104	205
ASP	69	62	32	127
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
Clipper	38	39	27	70

<sup>75</sup> Até poucos anos atrás, a tabela utilizada criada e mantida por Capers Jones na SPR por Boehm era divulgada gratuitamente na Internet. Atualmente ela é vendida por meio de uma assinatura no valor de US\$75.00. Como referência apresentamos alguns dados como eram divulgados anteriormente por Caper Jones e novos dados divulgados

Linguagem	SLOC/FP			
	Média	Mediana	Menor	Maior
COBOL	77	77	14	400
Excel	47	46	31	63
FoxPro	32	35	25	35
HTML	47	53	15	60
Informix	42	31	24	57
Java	62	63	53	77
JavaScript	58	63	42	75
JSP	59	-	-	-
Lotus Notes	21	22	15	25
Natural	60	52	22	141
Oracle	30	35	4	217
Oracle Dev 2K/FORMS	41/42	30	21/23	100
Perl	60	-	-	-
Powerbuilder	32	31	11	105
Smalltalk	26	19	10	55
SQL	40	37	7	110
VBScript	36	34	27	50
Visual Basic	47	42	16	158
Web Scripts	44	15	9	114

**Tabela 41. Tabela de conversão de pontos de função para linhas de código apresentada pela empresa QSM em [XXX]**  
<http://www.qsm.com/FPGearing.html>, em 1/12/2003

## X.8 Estimando o Tamanho

Precisamos então de metodologias que nos permitam prever o tamanho de um produto de software. A partir de uma estimativa podemos então utilizar as fórmulas.

No caso de pontos de função a metodologia principal é fazer uma análise inicial e contar os pontos de função presentes no resultado da análise. Dependendo da qualidade do processo da empresa e da profundidade dessa análise inicial o erro nessa contagem será bem pequeno.

Já no caso de linhas de código precisamos de um método de predição, como por exemplo, solicitar a previsão a um especialista com experiência em projeto semelhante. Um dos métodos sugeridos no passado foi a Técnica de Delfos<sup>76</sup>. Atualmente a técnica não aparece muito nos livros didáticos, provavelmente porque não atende as necessidades de velocidade do mundo atual.

### X.8.1 A Técnica de Delfos

A Técnica de Delfos é uma forma de fazer com que especialistas entrem em consenso sobre uma predição ou estimativa sem que haja uma discussão frente a frente.

<sup>76</sup> Ou Delphi. Baseado no Oráculo de Delfos, única relação direta com a linguagem Delphi.

Segunda a técnica a primeira ação a ser feita é a definição do produto sobre o qual a estimativa será feita. No nosso caso a pergunta em que estamos interessados é “quantas linhas de código” serão necessárias para implementar o software. Normalmente, a definição faz uma divisão do software em módulos, para facilitar a estimativa.

São então escolhidos especialistas, para os quais são distribuídos os questionários com a descrição dos módulos. Para cada módulo os especialistas devem fazer uma predição de tamanho. Este é o primeiro ciclo.

A partir do primeiro ciclo são feitos ciclos sucessivos onde são distribuídos os mesmos questionários, porém com a informação, não identificada, de cada resposta dada. Normalmente esta informação é dada em um gráfico indicando ainda a estimativa mínima, a máxima, a mediana e a média. A partir do segundo ciclo os especialistas devem justificar suas respostas, de maneira a facilitar a convergência de opiniões. Esse tipo de ciclo é repetido até que o consenso seja atingido.

Apesar da Técnica de Delfos não ser muito rápida, compreender seu funcionamento pode ajudar a determinar como deve ser feito o trabalho de predição de especialistas.

### **X.8.2 Cenários**

Outra técnica alternativa é determinar cenários de pior caso, caso mais provável e de melhor caso. O valor final da estimativa é dado por:

$$V_e = \frac{V_{pc} + 4 \cdot V_{cmp} + V_{mc}}{6}$$

**Equação 5. Cálculo do valor estimado ( $V_e$ ) em função do valor de pior caso ( $V_{pc}$ ), valor do caso mais provável ( $V_{cmp}$ ) e do valor do melhor caso ( $V_{mc}$ ).**

## **X.9 Verificando a Sanidade da Estimativa**

É importante que qualquer estimativa seja verificada quanto a sua qualidade. Uma das melhores formas é tentar formas alternativas de estimar e verificar se os resultados são compatíveis.

Por exemplo, é interessante verificar a estimativa dada por um método (por exemplo, COCOMO II) com estimativas dadas por um segundo método, de preferência feitas por pessoas diferentes.



## Capítulo XI. Projeto 1: Livraria ABC

---

O exemplo a seguir, a Livraria ABC, é uma adaptação de um problema que é apresentado em muitos cursos universitários no Rio de Janeiro. A seguir, descrevemos o negócio da Livraria ABC.

### XI.1 Entrevista 1

A Livraria ABC atua no mercado de venda de livros de arte sob encomenda.

Sua estratégia de atuação **não** prevê a manutenção de livros em estoque. Todos os livros solicitados por seus clientes são, semanalmente, encomendados às editoras. As editoras e os livros oferecidos são selecionados pela Direção da Livraria.

Para usar os serviços da livraria, os clientes devem se cadastrar previamente. A diretoria aprova o pedido de cadastro.

Os clientes enviam seus pedidos pelo correio, telefone ou fax. O pedido é aceito se o cliente e o livro estiverem previamente cadastrados. Caso contrário, o pedido é rejeitado com um aviso ao cliente.

Nas sextas-feiras, a livraria emite requisições de livros para as editoras, com base nos pedidos recebidos.

Quando os livros são entregues pelo fornecedor, a livraria confere a nota de entrega da editora com a requisição, devolvendo as que contiverem erros.

Os pedidos dos clientes são atendidos imediatamente quando completos, isto é, quando todos os livros pedidos foram enviados pelos fornecedores (ou forem cancelados). O atendimento consiste na emissão de uma nota fiscal, de um boleto de pagamento, que são enviados junto com o livro. Cópias da nota fiscal e do boleto são enviadas a tesouraria.

Se depois de 30 dias o fornecedor não enviou um livro requisitado, a livraria cancela o pedido junto ao fornecedor e elimina o livro do pedido do cliente. É enviado um aviso ao cliente desse fato, junto com o restante do pedido, se existir, ou isoladamente pelo correio.

### XI.2 Entrevista 2

A seguir descrevemos fragmentos de uma conversa com o Sr. José Letrado, proprietário da Livraria ABC, que deseja um SI para sua empresa.

A livraria funciona da mesma forma há muitos anos, com tudo feito de forma manual. Porém, com o aumento do interesse em livros de arte, nosso trabalho aumentou muito. Primeiro pensei em contratar mais pessoas, porém isso não ia diminuir a confusão de papéis com que estamos lidando e a falta de informações, então resolvi que seria necessário informatizar a empresa para, se necessário, crescer.

Hoje nosso processo, como é todo manual, tem muitos defeitos. Temos muita informação repetida, pois temos que controlar os pedidos dos clientes e as requisições aos fornecedores, que se cruzam. Muitas vezes recebemos um livro e por um problema de anotação não sabemos para qual cliente se destina. Então gastamos um bom tempo procurando, cliente por cliente, quem pediu aquele livro.

Mantemos também vários arquivos de clientes, o que facilita em certos casos, mas dificulta em outros. Temos os clientes com pedido em andamento, os clientes com pedidos atendidos, mas não pagos, os clientes freqüentes e os outros clientes, que não se enquadram em nenhuma dessas categorias.

Como estamos sempre manipulando fichas, algumas vezes esquecemos de requisitar a um fornecedor um ou mais livros pedidos pelo cliente. Isso atrasa o atendimento e resulta em reclamações que não são boas para a livraria.

Claramente, a primeira coisa que o sistema deve fazer é suportar o nosso funcionamento diário. Estou falando da operação básica de atendimento aos clientes, não da cobrança ou outras coisas do gênero, pois para essas vou comprar sistemas prontos<sup>77</sup>.

A partir dessa operação, também são necessários alguns dados para ajudar a gerenciar melhor a empresa. Dois relatórios são muito importantes para mim: um relatório de vendas em um período e um relatório de gastos por fornecedor. Outro relatório que me ajudaria muito é o de pedidos não atendidos.

Os desenvolvedores devem se lembrar que essa empresa é antiga e tradicional. Tanto os funcionários quanto a muitos dos clientes tem pouco hábito de usar computadores. O sistema deve ser muito simples de ser usado. Além disso, como pretendemos ter mais de um computador, o sistema deve funcionar em rede.

Outra coisa importante é que já compramos um sistema gerenciador de banco de dados, por causa do sistema de ERP que instalamos. O sistema tem que usar esse SGDB.

### **XI.3 Entrevista 3**

A seguir, algumas informações levantadas em uma reunião com Lúcia Pinho, gerente e funcionária de confiança da Livraria ABC.

Fazemos as coisas do mesmo jeito aqui há muitos anos, mas com a carga de trabalho aumentando, concordo que um sistema de vendas pode ajudar.

O importante é que se leve em conta que estamos comprando outros sistemas de informação não específicos para a Livraria ABC no mercado. Por exemplo, teremos um sistema de contas a pagar e a receber que deverá receber as informações do sistema que vocês vão fazer.

O que eu mais preciso é melhorar o nosso relacionamento com o cliente. Para isso, a informação que vem do sistema de compras é essencial. Uma coisa importante, por exemplo, é saber que clientes freqüentes não compram mais na freqüência que

---

<sup>77</sup> O funcionamento diário da empresa Livraria ABC é descrito no Projeto 1.

compravam. Outro é classificar os clientes de acordo com o tipo de livro que gostam, para podermos fazer ofertas de livros novos.

Tenho pensado muito em como um sistema pode nos ajudar. Até mesmo pensei se não seria interessante vender livros pela Internet, o que vocês acham disso? Seria uma grande novidade para nós e nossos clientes.

## Capítulo XII. Projeto 2: Empresa de *Clipping* ClipTudo

---

A empresa de *clipping* ClipTudo trabalha coletando matérias de jornal que são de interesse de seus clientes, preparando um *portfolio* periódico contendo cópias de todas as matérias, uma pequena avaliação de cada matéria segundo alguns critérios específicos e ainda um relatório mensal global.

A empresa funciona da seguinte maneira.

Com um entrevistador, o cliente define tópicos de interesse. Cada tópico é uma palavra ou uma sentença que pode ser identificada facilmente, como o nome da empresa do cliente ou um termo como “reforma da previdência” ou “venda de bebidas”. A partir desses termos iniciais o entrevistador cria uma lista expandida, com sinônimos e conceitos semelhantes. Essa lista é verificada pelo cliente, gerando uma lista final de tópicos de interesse.

Novamente com o entrevistador, o cliente define critérios de análise. Os critérios são quase sempre os mesmos, como viés da notícia em relação à necessidade do cliente (boa, ruim, neutra, propaganda paga), área da notícia, número estimado de leitores, etc.

Também junto com o entrevistador, o cliente define que meios de comunicação escrita serão monitorados, em uma lista mantida pela empresa de clippings.

A partir da quantidade de tópicos, da complexidade da análise, do período de geração de *portfolios* e da quantidade e frequência dos meios de comunicação, é feito um preço básico do serviço para o cliente. Além do preço básico o cliente ainda paga pela quantidade de cópias que deseja do *portfolio* e do relatório mensal.

No trabalho diário da empresa são empregados dois tipos de funcionários: leitores, classificadores e analistas. Os leitores lêem todas as publicações e copiam os artigos desejados, classificando segundo os tópicos de um ou mais clientes. Os classificadores pegam todas as notícias, classificando-as por cliente e criando um *portfolio* básico. Os analistas fazem as análises diárias e guardam os dados para as análises mensais, completando os *portfolios*.

Todo dia, até as 12h00min, são enviados os *portfolios* referentes a toda publicação obtida até as 19h00min do dia anterior, de acordo com o período pedido pelo cliente (diário, semanal, etc.).

Mensalmente os analistas pegam os dados que guardaram diariamente e fazem os relatórios mensais de resumo.

Cada vez que é emitido um *portfolio*, uma cópia de *portfolio* ou um relatório mensal é enviado um aviso ao sistema de cobranças.

## Capítulo XIII. Bibliografia

---

- [1] Gene Bellinger, Durval Castro, and Anthony Mills. Data, Information, Knowledge, and Wisdom. <http://www.systems-thinking.org/dikw/dikw.htm> . 6-8-2003.
- [2] Laudon, K. and Laudon, J., *Essentials of Management Information Systems*, 4th ed. Prentice Hall, 2001.
- [3] Modell, M. E., *A Professional's Guide to Systems Analysis*, 2nd ed. New York: Mc-Graw Hill, 1996.
- [4] Pressman, R. S., *Software Engineering - A Practitioner's Approach*, 4th ed. New York: McGraw-Hill, Inc., 1997, pp. -852.
- [5] Machado, C. A. F., "Normas e Modelos de Maturidade," in da Rocha, A. R. C., Maldonado, J. C., and Weber, K. C. (eds.) *Qualidade de Software: Teoria e Prática* São Paulo: Prentice-Hall, 2001.
- [6] Brooks Jr., F. P., *The Mythical Man-Month* Reading, Massachusetts: Addison-Wesley Publishing Company, 1982, pp. -195.
- [7] Robertson, J. and Robertson, S. Volere Requirements Specification Template. <http://www.volere.co.uk/template.doc> in 06/11/2003 [9]. 2003. Atlantic System Guild.
- [8] Robertson, J. and Robertson, S. Volere Requirements Specification Template. <http://www.volere.co.uk/template.doc> [9]. 2003. Atlantic System Guild. 6-11-2003.
- [9] Christel, M. and Kang, K. Issues in Requirements Elicitation. <http://www.sei.cmu.edu/pub/documents/92.reports/pdf/tr12.92.pdf> CMU/SEI-92-TR-012. 8-8-2003. Software Engineering Institute / CMU. 8-8-2003.
- [10] Kendall, J. and Kendall, K. **System Analysis and Design**. 4. 6-8-2001. Pearson, Prentice Hall.
- [11] Cockburn, A., *Writing Effective Use Cases* Addison Wesley, 2001.
- [12] Palmer, J. F. and McMenamim, S. M., *Análise Essencial de Sistemas* São Paulo: McGraw-Hill : Makron Books, 1991, pp. -567.
- [13] Ross, R. G. and Lam, G. S. Developing the Business Model: The Steps of Business Rules Methodology. 2003. Business Rule Solutions.
- [14] Furlan, J. D., *Modelagem de Negócio* São Paulo: Makron Books, 1997.

- [15] Sharp, A. The 7 Deadly Sins of Process Modeling. [www.drma-seattle.org/200202-pdf.pdf](http://www.drma-seattle.org/200202-pdf.pdf) . 7-8-2003. 7-8-2003.
- [16] Alonso, E. Untitled. [http://www.soi.city.ac.uk/~pauline/S930\(3\)fin1.ppt](http://www.soi.city.ac.uk/~pauline/S930(3)fin1.ppt) . 2003.
- [17] Seacord, R. C., Plakosh, D., and Lewis, G. A., *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices* Addison-Wesley, 2003.
- [18] Hay, D. and et alli, *Defining Business Rules: What they are really?*, Version 1.3 ed. The Business Rule Group, 2000.
- [19] Hay, D., Anderson, J. C., and and others. **GUIDE Business Rule Project: Final Report**. 1997. GUIDE International.
- [20] Von Halle, B., *Business Rules Applied: Building Better Systems Using the Business Rule Approach* New York: John Wiley & Sons, 2002.
- [21] Ross, R. G., *The Business Rule Book: Classifying, Defining and Modeling Rules*, Second ed. Boston, Massachusetts: Database Research Group, 1999.
- [22] Cougo, P., *Modelagem Conceitual e Projeto de Banco de Dados* Rio de Janeiro: Campus, 1999.
- [23] Bertini, C., Ceri, S., and Navathe, S. B., *Conceptual Database Design* redwood City, California: The Benjamin/Cummings Publishing Company, 1992.
- [24] Shlaer, S. and Mellor, S. J., *Object-Oriented Systems Analysis, Modelling the World in Data* 1999.
- [25] Robertson, S. and Robertson, J., *Complete System Analysis* New York: Dorser House, 1998.
- [26] Hay, D., *Princípios de Modelagem de Dados* São Paulo: Makron Books, 1999.
- [27] Heuser, C. A., *Projeto de Banco de Dados*, 4 ed. Porto Alegre: Editora Sagra Luzzatto, 2001.
- [28] Pompilho, S., *Análise Essencial: Guia Prático de Análise de Sistemas* Rio de Janeiro: IBPI Press, 1995.
- [29] Barbieri, C., *Modelagem de Dados* Rio de Janeiro: IBPI Press, 1994.
- [30] Ruble, D. A., *Practical Analysis & Design for Client/Server and GUI Systems* Upper Saddle River: Yourdon Press, 1997.
- [31] Miller, G. A., "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *The Psychological Review*, vol. 63 1956.

- [32] Pressman, R. S., *Software Engineering - A Practitioner's Approach* New York: McGraw-Hill, Inc., 1992, pp. -792.
- [33] Gane, C. and Sarson, T., *Structured System Analysis: Tools and Techniques* Englewood Cliffs, N.J.: Prentice-Hall, 1979.
- [34] Yourdon, E., *Análise Estruturada Moderna* Rio de Janeiro: Editora Campus, 1990, pp. -836.
- [35] Albrecht, A. J., "Measuring Application Development Productivity," *Proc. IBM Aplic. Dev. Symposium* Monterey, CA: 1979, pp. 89-92.

## Capítulo XIV. Índice

---

- Abstração, 94
  - Classificação, 35, 88, 94, 154, 160
  - Composição, 94, 95
  - Generalização, 86, 94, 95
- Agente Externo, 163
- Análise, 12, 14, 32
- Análise Essencial, 1, i, 1, 32, 138, 139, 140, 213, 214
  - princípio da neutralidade tecnológica, 140
- análise estruturada, 138
- Análise Estruturada, 138, 139, 215
- Ator, 102, 157, 175
- Boehm, Barry, 19, 20, 190, 193, 205, 206
- Cenários, 208
- Chen, Peter, 100, 110, 120, 124
- Comportamento, 42
- Cor, 133
- DENIM, 185, 186
- Diagrama de Entidades e Relacionamentos, 88, 89, 93, 96, 97, 100, 101, 102, 174
- Diagrama de Fluxo de dados hierárquico, 166
- Diagrama de Fluxo de Dados, 138, 144, 150, 156, 158, 160, 162, 163, 164, 165, 166, 167, 168, 169, 174, 176, 203
  - de contexto, 164, **166**, 167
  - nível zero, 166
  - nivelado, **166**
  - particionado, 167
- Engenharia de Informação, 100, 120
- Entrevista, 37, 38, 43, 209, 210
  - Aberta, 38
  - Estruturada, 38
  - por Questionário, 37
- EPC, 2, 63, 75, 76, 78, 80, 92
- estímulo, 143, 144, 146, 147, 150, 159, 160
- Evento, 32, 138, 147, 154, 159, 168, 172, 203
  - não-esperado, 148, 153
  - não-evento, 154, 160
  - temporal, 130, 138, 154
- Fatos, 35, 85, 86
- Funções de Negócio, 63, 65, 195
- Furlan, José Davi, 213
- IDEF0, 2, 65, 66, 67, 68, 72, 73, 74, 92, 164
- IDEF1X, 93, 100, 101, 119, 120, 123, 124
- Interface com o Usuário, 177, 178
- JAD, 2, 25, 36, 44, 45, 58, 152, 182
- Martin, James, 120
- McMenamim, 139, 213
- Memória, 96, 138, 156
- Microsoft, 54, 182
- Modelagem Conceitual de Dados, 96, 137
- Modelo de Processo, 15, 16, 17, 18, 19, 33, 50, 90, 91, 169
- Modelo Funcional, 14, 138, 203
- Palmer, 139, 213
- Pontos de Função, 189, 193, 194, 205



arquivos, <b>194</b>	Requisitos falsos, 31
consultas, <b>194</b>	SAD, 7
entradas, <b>194</b>	SADT, 66, 164
interfaces, <b>194</b>	SIG, 7
saídas, <b>194</b> , 196	Sistemas de informação, 4, 6
Pressman, 13, 213, 215	Tecnologia Interna Perfeita, 141
Regras de Negócio, 63, 84, 87	Tela, 161, 162, 204
Relatórios, 106, 154	Termos, 85, 174
Requerimentos	Testes, 17, 193
verdadeiros, 31, 32	Win-Win, 19
Requisitos, 16, 25, 28, 29, 30, 31, 32, 33, 34, 35, 36, 46, 47, 53, 59, 61, 62, 193	Yourdon, Edward, 139, 146, 162, 165, 171, 174, 214, 215