

Capítulo I. 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 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:

- Organograma¹
- 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 implementá-los. Em alguns métodos, pode até mesmo ser utilizada no lugar desses modelos.

¹ O organograma é uma das formas mais simples e antigas de modelar uma empresa.

I.1 Níveis de abstração tratados nesse capítulo

Nesse capítulo analisaremos a empresa do nível mais abstrato para o mais detalhado.

O nível mais abstrato descreve as funções de negócio da organização, sem se preocupar como essas funções são executadas ou em que ordem. Nesse nível, estamos preocupados em definir as responsabilidades da organização, e as entradas e saídas necessárias para cumprir essas responsabilidades.

Em um nível mais detalhado, analisamos os processos da empresa. Nesse nível estamos preocupados em como a organização executa suas funções, com que passos, e por meio de que pessoas.

Finalmente, no nível mais detalhado de todos, descreveremos as regras de negócio que regem esses processos.

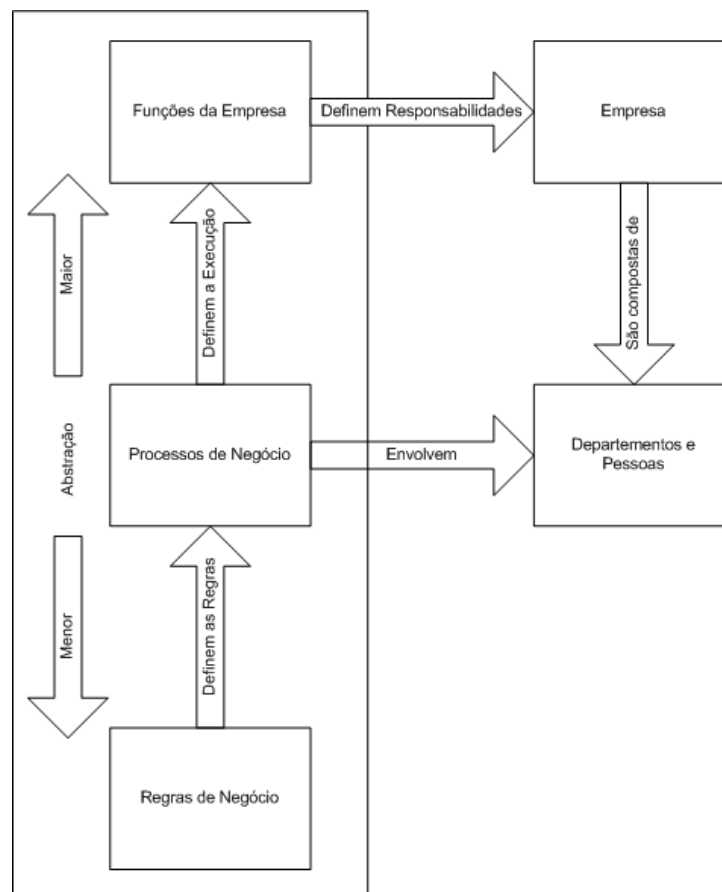


Figura 1. Níveis de abstração analisados nesse capítulo e suas relação com a empresa.

I.2 O Organograma

A forma mais simples de representar uma empresa é provavelmente o organograma.

Organogramas são diagramas que descrevem a estrutura formal de uma organização incluindo suas relações hierárquicas, normalmente por meio de linhas e retângulos. Normalmente são simples de ler, porém alguns organogramas são mais complicados, de forma a representar informações adicionais..

Normalmente um organograma tem o formato hierárquico, onde no alto está uma caixa com a posição mais importante da organização, e nos níveis abaixo aparecem caixas com seus subordinados. Cada caixa pode representar um cargo, uma função ou mesmo um departamento. A figura a seguir mostra um organograma simples.

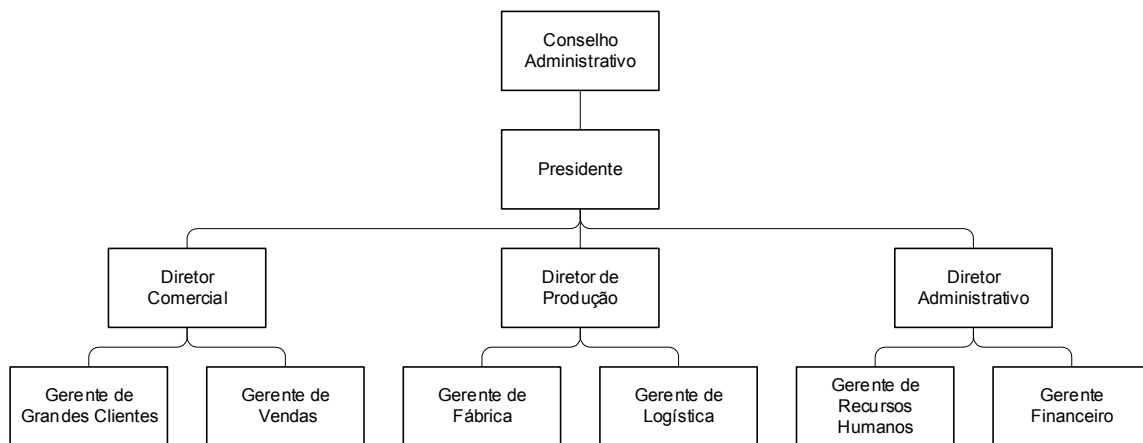


Figura 2. Um exemplo de organograma simples, contendo apenas cargos.

Em geral o analista não precisa levantar o organograma, pois a empresa já o possui, mas é comum que haja algumas mudanças. Na verdade, não é trabalho do analista de sistemas construir o organograma da empresa, porém ele precisa conhecê-lo para melhor desenvolver o seu trabalho. Para isso é importante obter esse documento junto ao cliente, e verificar não só a hierarquia de cargos, mas também quem ocupa cada cargo, e como entrar em contato com cada um dos membros da organização que possa ter interesse no sistema.

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 para 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.

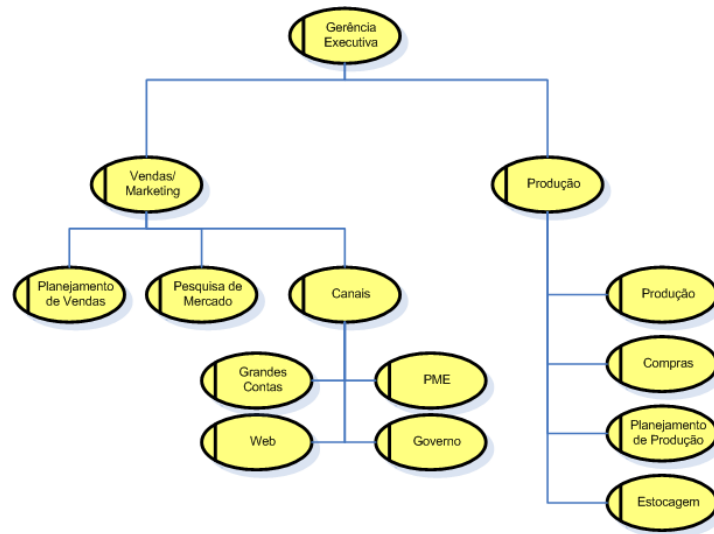


Figura 3. Exemplo de organograma simples, contendo departamentos, feito com símbolos da metodologia ARIS².

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).

Este texto não tratará do processo de levantamento do organograma, pois isso é mais afeito à administração. Fica, porém, o lembrete de sua importância como documento de referência ao analista.

I.2.1 Relações em um organograma

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.

² Outra técnica componente do ARIS, o EPC será vista ainda neste capítulo

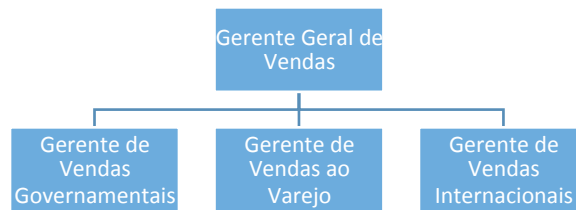


Figura 4. A relação de subordinação, normalmente mantida na horizontal, entre o gerente geral e seus gerentes subordinados.

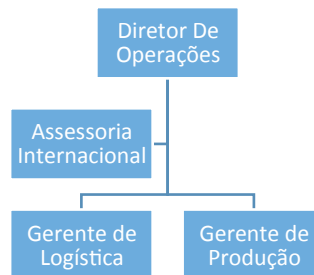


Figura 5. A relação de assessoria é normalmente feita com uma barra horizontal, como a Assessorial Internacional aparece no diagrama acima

I.2.2 Outros formatos de organograma

Na grande maioria das vezes serão encontrados nas organizações organogramas clássicos ou pequenas variações dos organogramas clássicos. Existem, porém, algumas formas alternativas.

Uma forma interessante, também com muitas variações, é a radial ou solar. A figura a seguir mostra um organograma radial com

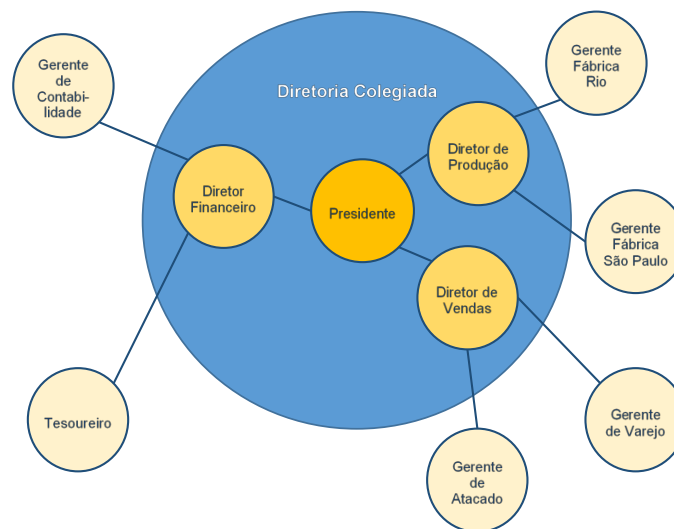


Figura 6. Um exemplo de organograma radial

Outra forma possível é inverter o organograma. O organograma a seguir, criado pelo site Guia de Direitos, mostra a estrutura do Poder Judiciário, não tratando de cargos, mas sim de partes desse poder.



Figura 7. Organograma do Poder Judiciário, em forma inversa, criado pelo site Guia de Direitos.

Devido à grande variedade de formas usadas, organogramas podem ser desenhados tanto em sistemas específicos como em softwares genéricos de desenho.

I.3 Funções de Negócio

As funções de uma instituição são os grupos de processos que gerenciam um recurso básico da organização [B20] 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 [B4] 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 áreas³
- 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).

I.4 Modelagem de Funções com IDEF0⁴

IDEF0, Integration Definition Language 0, ou “Integration Method for Function Modelling” [B21] é uma forma de representar sistemas, desde máquinas específicas até grandes empresas, por meio de uma rede de funções interconectadas e interfaces entre essas atividades. Esses modelos representam funções do sistema, relacionamentos funcionais e dados que suportam a integração do sistema.

O IDEF0 pode ser feito em vários níveis de abstração. Podemos descrever as funções de uma empresa, como “receber pedidos” ou “produzir encomendas”; ou podemos descrever as funções de uma pequena máquina, como “medir temperatura”.

A característica mais importante do IDEF0 é que ele não descreve como as coisas são feitas e também não dá nenhuma ordem. Ele apenas define as responsabilidades, ou, de certa forma, os requisitos funcionais de um sistema, isto é, que funções devem ser feitas e qual a interface de cada função. Assim, quando precisarmos descrever passos sequenciais, algoritmos ou outros detalhes, devemos usar outro modelo.

I.4.1 Introdução ao IDEF0

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.

O principal conceito representado no IDEF0 é a função. Uma função é uma atividade, um processo, uma transformação, ou ainda uma agregação de vários desses conceitos que acontecem dentro de um sistema. As funções são descritas por caixas que contém um nome e uma identificação.

³ Não há necessariamente um mapeamento direto entre funções e o organograma

⁴ Algumas imagens dessa seção **não** são originais, porém fazem parte de um documento (IFIP 183) em domínio público (obra do governo americano).



Figura 8. No IDEF0, uma caixa representa uma função. Para identificar a função é necessário um nome e um número.

O nome da função deve descrever de forma breve o que ela faz. Como de praxe em técnicas de Engenharia de Software, utilizamos a forma

<verbo no infinitivo> <objeto direto>

para dar o nome da função. Nomes razoáveis para funções, possíveis em diferentes sistemas, seriam: avaliar aluno, corrigir defeitos, preparar pedido, embrulhar pacotes, etc.

O número da função é dado por diagrama. Em cada diagrama, cada caixa recebe um número, que varia de 1 a 6 segundo o padrão.

Como o IDEF0 busca fazer uma representação hierárquica das funções do sistema sendo descrito, uma caixa pode descrever tanto uma função detalhada, como uma função muito abstrata que só podemos definir por meio da composição de outras funções.

Vamos adotar como exemplo um modelo IDEF0 que descreve o processo de seleção de candidatos em uma empresa. Nessa parte do texto, vamos discutir funções de vários níveis e tentaremos, ao longo do processo, organizar nossos diagramas.

Vamos então escolher o nome do sistema que estamos modelando: “Selecionar Profissionais”. Esse sistema é uma função comum em várias empresas, sendo uma função bastante abstrata, composta de várias outras funções. No nosso caso, existem três funções que compõe “Selecionar Profissionais”: “Preparar Perfil”, “Divulgar Anúncio” e “Selecionar Candidatos”.

Nossas caixas seriam como as das duas figuras a seguir:



Figura 9. Uma caixa que descreve o sistema Selecionar Profissionais.



Figura 10. Possíveis caixas definindo funções para o modelo IDEF0 do sistema “Selecionar Profissionais”

Nesse ponto vamos incluir um novo objeto do IDEF0, o diagrama. Cada diagrama do IDEF0 também representa uma função. Porém, enquanto as caixas representam uma função como “caixa preta”, um diagrama descreve essa função como uma “caixa branca”. Um diagrama IDEF0 é uma página com o formato da figura a seguir.

USED AT:	AUTHOR: Geraldo Xuxo	DATE: 1/9/2006	<input checked="" type="checkbox"/> WORKING	READER	DATE	CONTEXT:
	PROJECT: Nome do Modelo	REV: 1/9/2006	<input type="checkbox"/> DRAFT			TOP
			<input type="checkbox"/> RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10		<input type="checkbox"/> PUBLICATION			
NODE:	TITLE:			NUMBER:		
A-0						

Figura 11. Um quadro de um diagrama IDEF0, segundo o software BPWin (ou All Fusion Process Modeller), que segue basicamente as normas do padrão.

Esse diagrama tem algumas regiões, que identificamos a seguir.

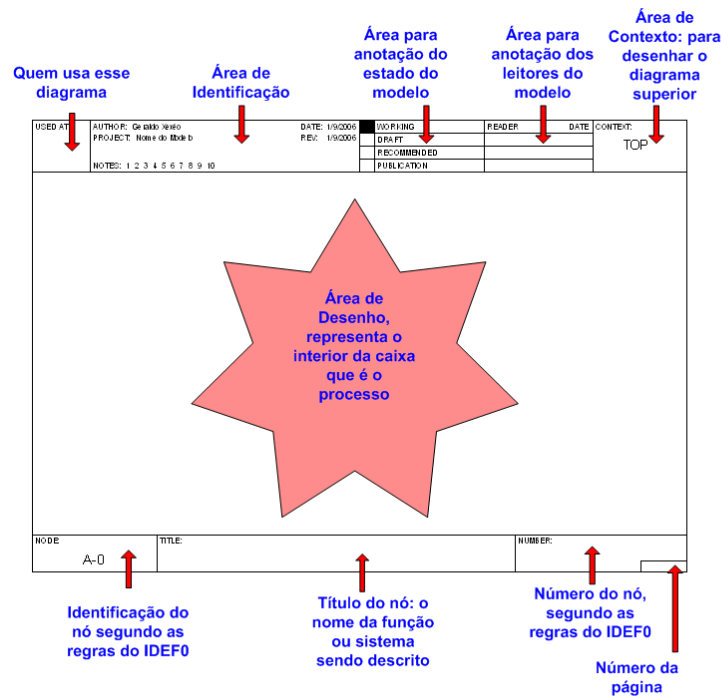


Figura 12. Áreas do diagrama

De cima para baixo podemos identificar as seguintes áreas do diagrama:

- USED AT, que indica em que diagramas esse diagrama é mencionado. Normalmente indicamos todos os diagramas filhos.
- Identificação, composta de vários campos, Autor, Nome do Projeto, Data e Data de Revisão.
- Notas: que são números que servem para ser cortados em função da criação de anotações no texto.
- Estado do trabalho, que pode ser “em trabalho”, “rascunho”, “recomendado” e “publicado”
- Anotações para os leitores marcarem que leram o projeto.
- Área de desenho do contexto do diagrama, que mostra um “resumo” do diagrama de nível superior, com a caixa sendo explicada em negro.
- Área de desenho.
- Nome do nó, que segue como regra colocar um “A” na frente do número que representa o caminho até chegar a caixa sendo descrita. Por exemplo, o diagrama que explica a caixa 1, que está no diagrama que explica a caixa 2, que explica a caixa 3 do diagrama A0 será chamado A321.
- Título do diagrama, que é o título da função sendo explicado ou outro título como “Glossário”.

- Número do diagrama, que se começa com as iniciais do criador e tem depois um número seqüencial, pela ordem de criação.
- Número da página do relatório.

Devemos notar que apesar de serem utilizados basicamente para servir na descrição detalhada de uma função, os diagramas do IDEF0 também servem para algumas outras funções, como criar índices ou glossários para a especificação completa.

Nosso modelo então deve começar com um diagrama. O primeiro diagrama é sempre o **Diagrama de Contexto** e contém também uma única caixa, que representa todo o sistema. Além disso, esse diagrama tem o mesmo nome da caixa.

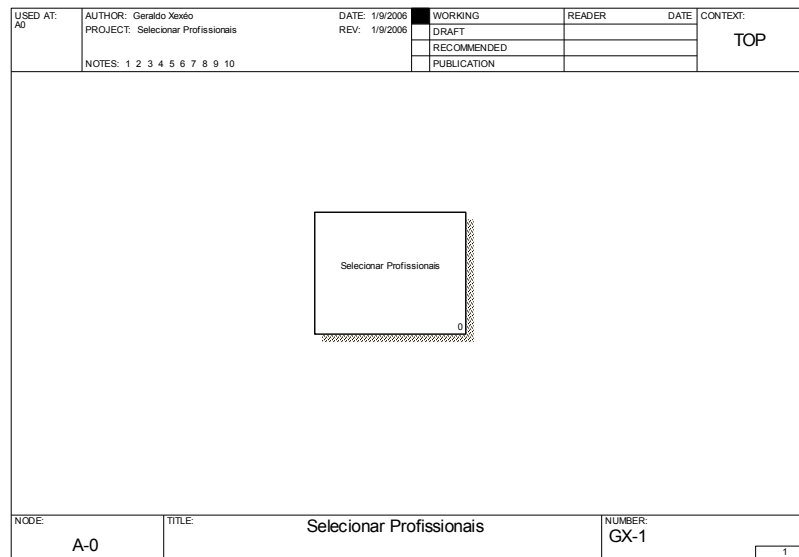


Figura 13. Diagrama de Contexto do IDEF0, contendo apenas a função que representa o sistema completo. Faltam ainda as setas.

O nome do primeiro diagrama é “A-0”, lido “A menos 0”. Esse nome, apesar de estranho, é padrão. Mais tarde discutiremos de forma mais detalhada a numeração. No diagrama “A-0” só existe a função “0”.

A função “Selecionar Candidatos”, do diagrama A-0, está descrita de uma forma muito geral. Para explica-la nós devemos “explodi-la”, ou seja, devemos criar um novo diagrama que a explique. A partir do diagrama “A-0” e da função “0” nós podemos criar o diagrama “A0”. Nesse diagrama colocaremos as funções que explicam como “Selecionar Profissionais” funciona.

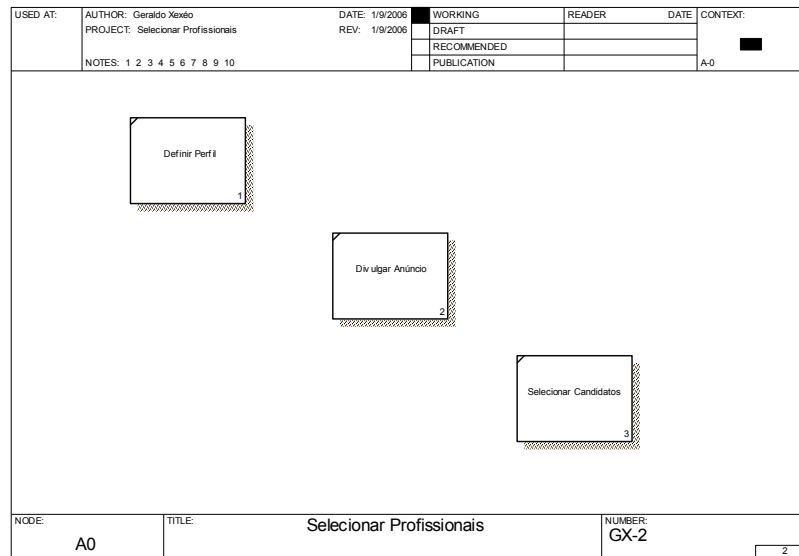


Figura 14. A explosão do processo “0”, do diagrama “A-0”, é o diagrama A0. Ainda faltam as setas.

Podemos agora continuar com nossa modelagem. Vendo os dois diagramas acima podemos compreender rapidamente que funções são feitas e a forma como uma função é composta pelas outras três. Porém, ainda pouca informação sobre essas funções: o que elas fazem realmente? Como elas se comunicam?

Para aumentar a informação que temos sobre as funções, nós descreveremos os fluxos de informações e objetos que trafegam entre elas. A função de cada uma das setas é dada pelo seu posicionamento ao redor da caixa da atividade, como descrito na figura a seguir.

- Entradas (setas entrando pela direita) são dados ou objetos que são transformados ou consumidos na saída pelo processo.
- Controles (setas entrando por cima) são condições necessárias para produzir a saída correta, podendo ou não ser transformados na saída. Controles são restrições na operação do processo.
- Uma saída (setas saindo pela esquerda) apresenta um **resultado** do processo, um artefato ou informação criada ou transformada por ele.
- Os mecanismos ou recursos (setas entrando por baixo) são os **meios, equipamentos**, ou mesmo **pessoas**, 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.

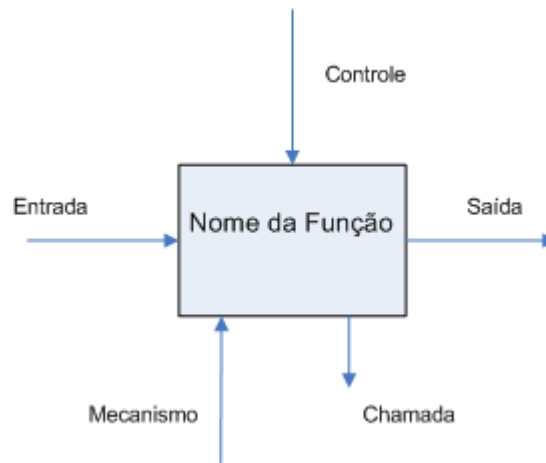


Figura 15. Setas que indicam fluxos de objetos ou informações e em que posições e direções elas são válidas.

Três tipos de setas entram na caixa e dois tipos de setas saem da caixa. Vamos diferencia-las um pouco mais:

- É praticamente impossível confundir uma saída com alguma outra coisa.
- Toda caixa deve obrigatoriamente ter pelo menos uma saída.
- Uma chamada de função é também fácil de identificar.
- Um mecanismo não pode ser confundido com Entradas e Controles. Mecanismos são usados para fazer algo, são equipamentos como telefones, fax. Eles não são transformados (como uma entrada) e também não podem representar uma informação (como o controle).

O padrão é levemente confuso ao explicar a diferença entre “Entradas” e “Controles”. Vejamos algumas diferenças:

- Entradas são opcionais, controles são obrigatórios.
- Entradas podem ser informações ou objetos.
- Controles são sempre informações ou comandos.
 - Um objeto não pode ser controle.
- Entradas são obrigatoriamente consumidas ou transformadas.
 - Entradas alteram alguma coisa no sistema.
- Controles dirigem (controlam) a função.
- Se uma informação é apenas consultada, como um catálogo, é um controle.
- Controles podem ser consumidos ou não.
- Se não for identificado nenhum controle, uma entrada deve ser “promovida” para controle.

- Em caso de ambigüidade, mantenha a seta como Controle até que seja resolvida.

Um catálogo de preços que é consultado em uma função de venda, por exemplo, é um controle. Uma atividade onde uma decisão é tomada sem que nenhum de seus dados sejam alterados possui apenas Controles.

Além de todas as diferenças e regras de utilização, o padrão também diz que em caso de dúvida entre controle e entrada o analista deve escolher representar a seta como controle. Vamos, então, definir as entradas e saídas de nosso sistema e ver como fica o diagrama na figura a seguir.

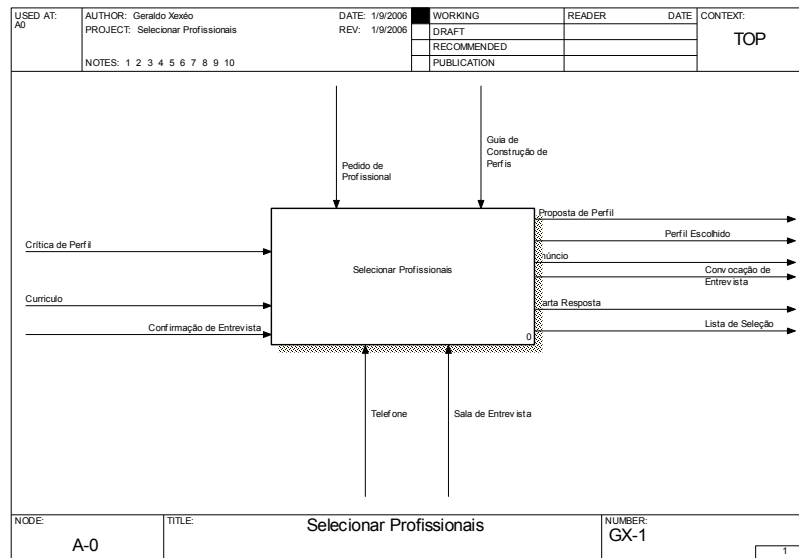


Figura 16. Diagrama de Contexto do Projeto “Selecionar Profissionais”, agora com entradas, controles, saídas e mecanismos.

Analisando o diagrama acima podemos ver algumas características do nosso exemplo e também do diagrama IDEF0:

Para selecionar profissionais, é necessário ter as seguintes informações;

- “Pedido de Profissional”,
- “Guia de Construção de Perfil”,
- “Currículo”
- “Crítica de Perfil”
- “Confirmação de Entrevista”

Porém, o diagrama não indica algumas coisas, como o fato que a “Crítica de Perfil” só é entregue ao sistema depois do sistema produzir uma “Proposta de Perfil”, e que se a “Crítica de Perfil” for uma aprovação, o sistema produz o “Perfil Escolhido”.

Aqui chegamos a uma característica das funções descritas pelos diagramas IDEF0: sem conhecer detalhadamente como a função realmente é executada, não podemos dizer a ordem dos fluxos ou se eles são opcionais ou obrigatórios. Isso, porém, não é um defeito, mas sim uma propriedade ligada a necessidade de ter alguma abstração. Vamos ver agora a explosão da caixa “0”, o diagrama “A0”.

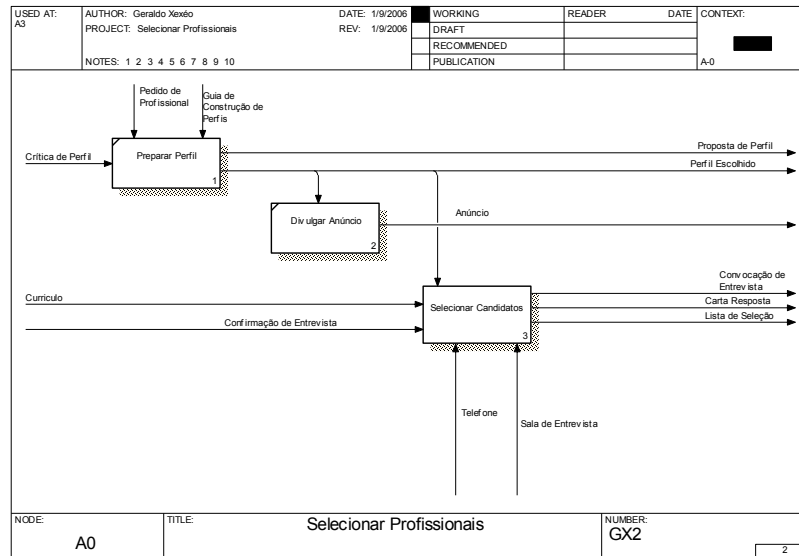


Figura 17. O nó A0 é a expansão da caixa “0” do nó “A-0”. Podemos notar que as setas se conservam entre os diagramas.

No diagrama acima, podemos ver algumas características importantes. É importante prestar atenção em algumas regras sintáticas do diagrama. As setas sempre entram ou saem do lado apropriado das caixas, e sempre partem ou vão em direção da margem ou de outra caixa.

As caixas se comunicam por meio dos fluxos. Os fluxos só dizem que função produz o fluxo e que função consome o fluxo, seja ele de objetos, comandos ou informações. Não há nenhuma suposição de uma função é capaz de invocar a outra, mandar mensagens ou exercer qualquer outra forma de comando. Entradas, saídas, controles e mecanismos são descritas como **responsabilidades** da função, sem nenhuma ordem, obrigatoriedade de uso ou prioridade.

O método IDEF0 não representa a sequência de atividades no tempo, apenas as interações entre as atividades.

Podemos ver também que os fluxos da caixa “0” do diagrama “A-0” são divididos entre as funções que compõe a função “Selecionar Profissional”. Nesse caso, nenhum fluxo é usado em mais de uma caixa, mas isso é plenamente aceitável.

Também podemos verificar que é possível criar fluxos internos, que demonstram a comunicação entre as funções. O fluxo “Perfil Escolhido” possui sub-fluxos internos.

Podemos também aprender a como referenciar um objeto qualquer dos diagramas. Para isso usamos uma notação simples, que será detalhada mais adiante, composta de:

- Número do diagrama: A-0, A0, A1...
- Número da caixa: 1,2,3...
 - Uma caixa “1” em um diagrama “A0” é denotada na forma “A0.1”
- Uma letra para o tipo da seta, seguida de um número, no formato X#, como em C1, O2
 - A letra deve ser uma de:
 - I para entrada (*input*)
 - C para controle (*control*)
 - O para saída (*output*)
 - M para mecanismos (*mechanism*)
 - O número é a ordem seqüencial da esquerda para direita ou de cima para baixo
- Estando claro o contexto, as partes mais gerais do código podem ser abandonadas

Assim, o código A0.2 representa a caixa “Divulgar Anúncio” e o código A0.3O2 representa o fluxo “Carta Resposta”.

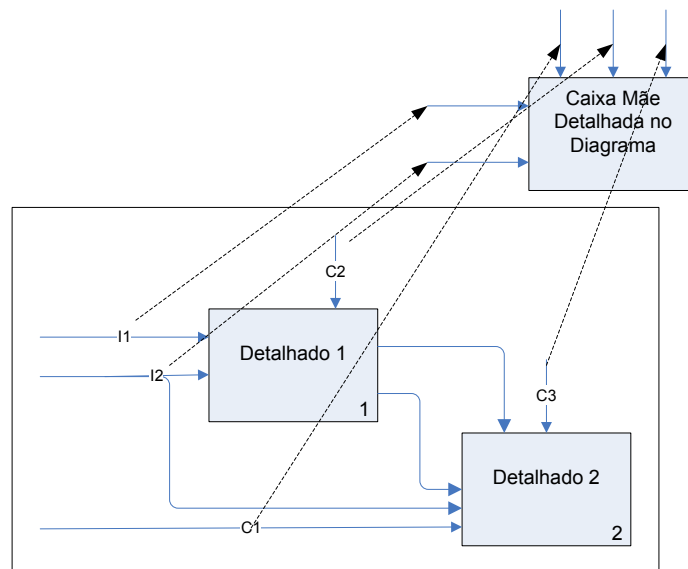


Figura 18. Exemplo de numeração ICOM e seu significado

Mais um detalhe que podemos perceber é que a caixa A03.3 não apresenta uma marca no canto superior esquerdo, enquanto que as caixas 1 e 2 no mesmo diagrama apresentam uma linha nesse canto. A linha significa que essas caixas não estão expandidas, e a ausência de linhas significa o inverso, isto é, a caixa possui um diagrama que a explica.

Vejamos então a expansão da caixa A0.3, que é o diagrama A3.

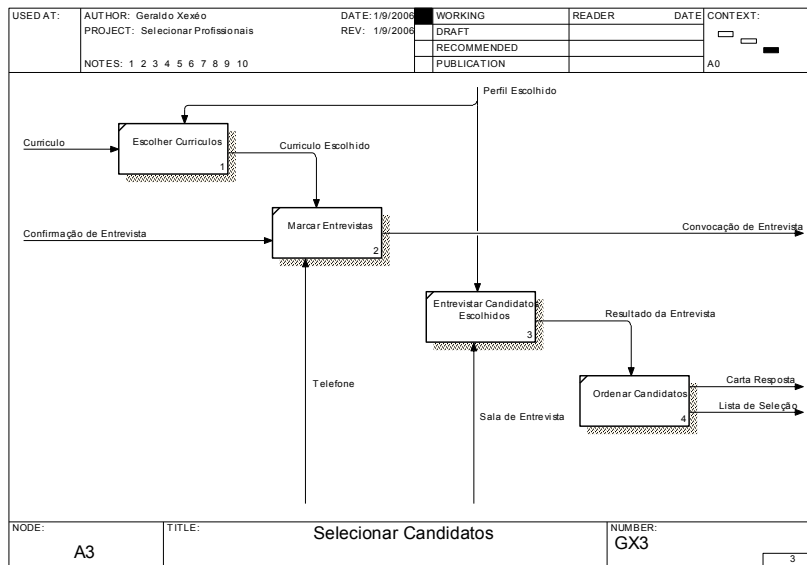


Figura 19. Diagrama A3 do modelo exemplo.

Ainda é possível criar um diagrama resumo, que apresentamos a seguir:

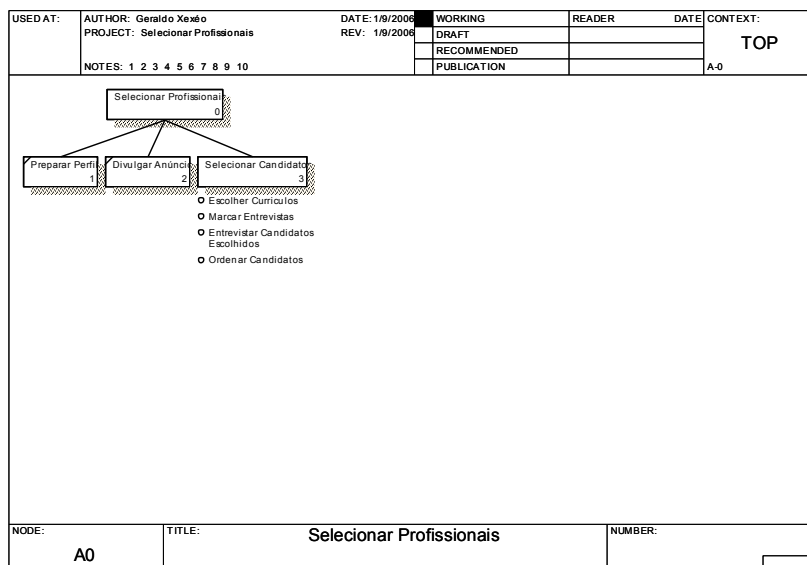


Figura 20. Diagrama resumindo nosso modelo IDEF0 (Árvore de nós).

1.4.2 Sintaxe do IDEF0

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 por caixas, que são conectadas uma às outras por meio de setas com

significados distintos, representando dados ou objetos relacionados a cada função⁵. 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)⁶ que sempre contém uma única atividade, numerada 0, a partir do qual são feitos detalhamentos⁷ 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.

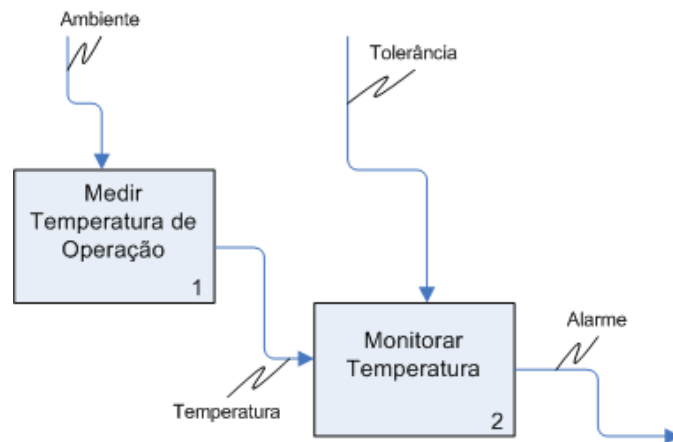


Figura 21. Exemplo, em um fragmento de IDEF0, dos usos de controles e entradas.

Algumas regras sintáticas:

- Os diagramas são desenhados em formulários padronizados.
- Diagramas são identificados (Node) na forma An, onde n é um número.
 - Um diagrama pode ser apenas para explicação (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, como em A43F ou A34T.
- 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 sequencial.

⁵ A metodologia IDEF0 é derivada de uma metodologia anterior conhecida como SADT.

⁶ O processo inicial é sempre o A-0, não existindo um processo B-0 em nenhum caso. A letra “A” vem de atividade.

⁷ Também conhecidos informalmente como “explosões”.

- Caixas denotam atividades, por isso devem ser nomeadas na forma <verbo> <objeto>.
- 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 são numeradas de 1 a 6, normalmente do canto superior esquerdo em direção ao canto inferior direito (sentido da leitura).
 - Em outras documentações, as caixas são referenciadas pelo nome do diagrama adicionadas do número da caixa (a caixa 1 do diagrama A1 se chama A1.1)

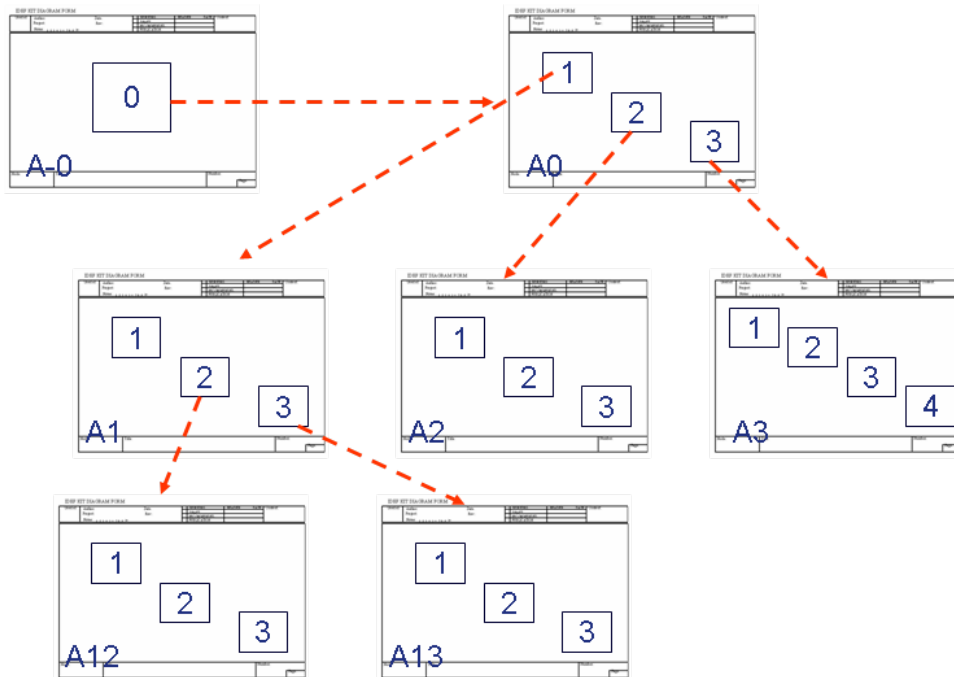


Figura 22. Exemplo de numeração de diagramas e caixas de acordo com explosões (setas vermelhas). As setas ICOM foram eliminadas para facilitar a leitura.

- 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.
- Cada diagrama deve conter todas as setas que entram e saem do seu diagrama superior, que podem ser indicadas pela seguinte notação (conhecida como ICOM):
 - Controle: C1, C2, C3..., contados da esquerda para a direita na caixa explodida.
 - A cada revisão deve ser marcado um número de revisão (no diagrama, ver NOTES 1 2 3...).

- 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”)

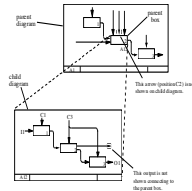


Figura 23. Exemplo de tunelamento e DRE (na caixa 2 do diagrama A1)⁸

- 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.

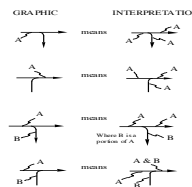


Figura 24. Fork e join não rotulados têm interpretações padronizadas⁹

- Uma caixa possui
 - No mínimo 1 seta de controle
 - No mínimo 1 seta de saída
 - No máximo 1 seta de chamada

⁸ Imagem pertencente a um padrão do governo americano e em domínio público.

⁹ Imagem pertencente a um padrão do governo americano e em domínio público.

- [illegible]

¹¹ Imagem pertencente a um padrão do governo americano e em domínio público.

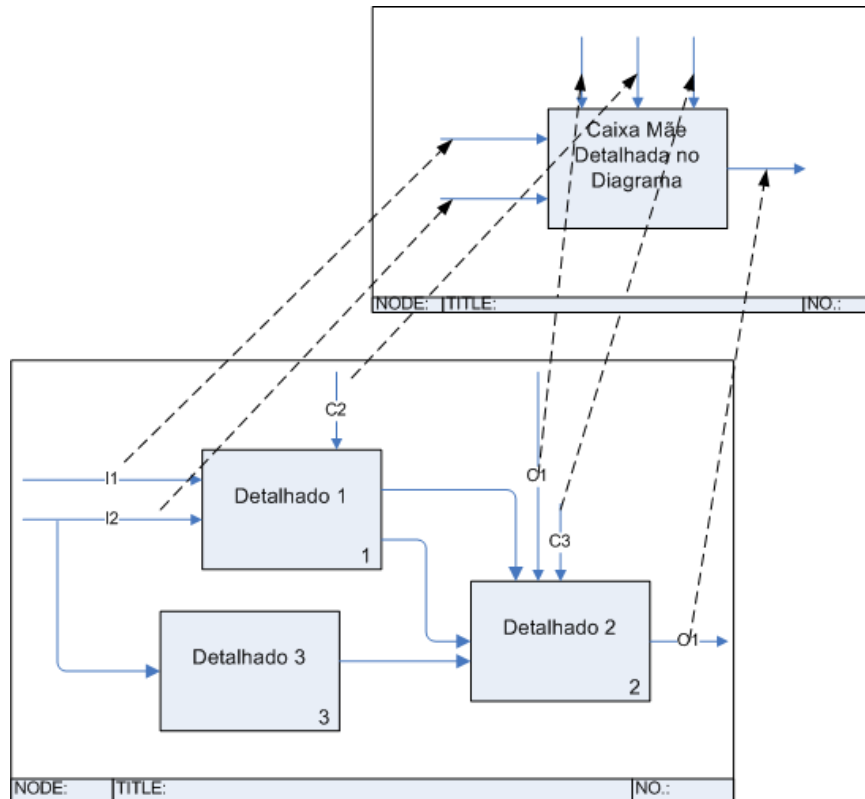


Figura 27. Numeração das setas de acordo com o diagrama superior.

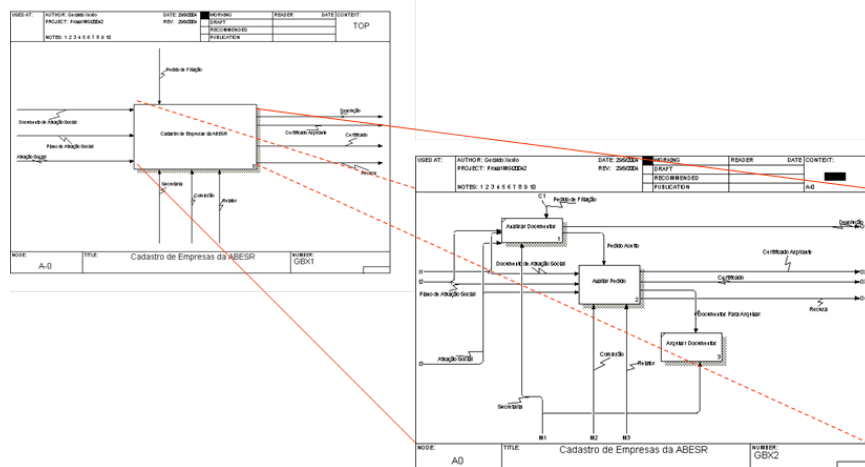


Figura 28. A explosão de uma função deve manter todos as setas.

- Diagramas apenas para exposição (FEO – "for exposition only") podem ser usados onde um nível adicional de conhecimento é necessário para entender adequadamente uma área específica do modelo.
- Diagramas FEO não precisam seguir as regras de sintaxe do IDEF0
- Diagramas FEO são numerados com um F no final de seu código, ou seja, do código que teriam se fossem um diagrama normal.

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

Notação de referência para o IDEF0	
Notação de Referência	Significado
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”.
A42.3	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 1. 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.

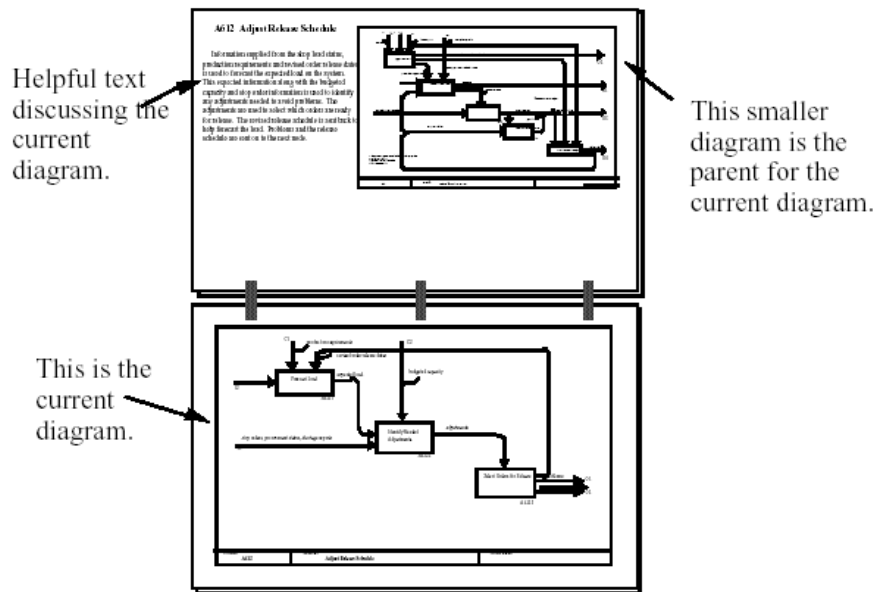


Figura 29. Formato de publicação pedido pelo padrão IDEF0¹²

1.4.2.1 O tunelamento

Existem duas posições para colocar um tunelamento (na forma de parênteses): na extremidade próxima a função ou na extremidade próxima à borda do diagrama:

Quando colocado em torno da extremidade conectada a função, isso significa que todas as sub-funções daquela função presentes no diagrama inferior possuem, de forma implícita, essa seta.

A seta pode ser retomada, se necessário, nos diagramas de níveis mais baixos, sem nenhum problema (ela “pula” um ou mais níveis do diagrama, do mais abstrato para o menos abstrato).

Quanto colocado em torno da extremidade próxima a borda do diagrama significa que essa seta existe implicitamente em todas as funções mais abstratas (diagramas superiores) que a função sendo descrita.

1.4.3 Construção de um modelo IDEF0

Um modelo IDEF0 deve ser construído normalmente da forma *top-down*, partindo do mais abstrato para o mais detalhado. O método *top-down* permite que nos preocupemos primeiro com questões gerais do sistema, como a sua justificativa, que funções deve realizar e, mais tarde, com sua realização. Outra característica importante é a necessidade de delimitar o escopo de análise e descrição do sistema, o que também é apoiado pela técnica *top-down* do IDEF0, já que ela exige que uma descrição abstrata do sistema tenha sua fronteira bem definida, na forma de entradas, saídas, controles e mecanismos {Pierre Nancy, 2004 1998 /id}.

¹² Imagem pertencente a um padrão do governo americano e em domínio público.

Ele pode ser usado tanto para representar processos já 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.

Um conjunto de diagramas IDEF0, conhecido como um kit IDEF0, tem que responder a duas perguntas: para que serve o sistema e como ele funciona.

1.4.3.1 Objetivo

Para começar a modelagem IDEF0 o analista deve primeiro determinar e descrever de forma clara qual o objetivo do modelo, em que ponto de vista as atividades serão descritas e em que contexto isso é feito. Isso funciona como uma especificação de requisitos do modelo que está sendo feito. Quando o objetivo do modelo é atingido, o modelo está completo.

Um objetivo possível é, por exemplo, “identificar oportunidades para consolidar funções já existentes de forma a melhorar o desempenho da organização”. Claro que esse objetivo sofre de um excesso de “linguagem de negócios” que pode ofuscar sua verdadeira utilidade. Normalmente devemos preferir termos mais diretos como “identificar as funções da organização em busca de estudá-las e propor um plano de melhoria de desempenho com possível reestruturação das mesmas”.

1.4.3.2 Ponto de Vista

O ponto de vista descreve a perspectiva tomada na construção, revisão e leitura do modelo, definindo, na prática, os limites do modelo e como as atividades do sistemas sendo descrito serão abstraídas ou idealizadas. A partir de um ponto de vista controlamos o escopo e o nível de detalhe de um modelo IDEF0. O ponto de vista é sempre único, apesar das sessões de modelagem incluírem normalmente diferentes participantes com múltiplos pontos de vista.

Uma forma de imaginar um ponto de vista e melhor descreve-lo é entender o IDEF0 como parte de um manual destinado a descrever o funcionamento do sistema para alguma pessoa ou algum grupo no contexto do negócio.

1.4.3.3 Contexto

O escopo do modelo é dividido em duas partes: a profundidade e a extensão. A profundidade define o nível de detalhe esperado do modelo. A extensão define as fronteiras do sistema sendo analisado.

É importante a definição inicial do contexto, mesmo tendo consciência que ele pode sofrer alterações (intencionais) durante o curso do processo de modelagem.

O contexto é representado fortemente no diagrama de contexto (A-0), principalmente pela definição de fronteira do sistema indicada pelas entradas (incluindo controles) e saídas.

1.4.3.4 Regras gerais de construção

O método sempre se inicia pela definição da caixa inicial (A-0) e sua expansão em um diagrama de primeiro nível (A0). A partir desse ponto, sempre que for necessário expandir uma função será criado outro diagrama filho, mantendo as seguintes regras {Pierre Nancy, 2004 1998 /id}:

- Cada função deve trazer algum valor agregado a suas entradas e controles.
- Cada função recebe um nome na forma verbo no infinitivo + objeto direto
- Os sub-sistemas de uma função devem suportar diretamente a função.
- Cada seta que entra ou sai de uma função deve ser encontrada em seu diagrama de expansão .
- As setas podem entrar ou sair de uma ou mais funções
- As setas pode ser divididas de forma a transportar parte de informação para uma função e parte para outra.
- Em casos especiais as setas podem não aparecer em um diagrama superior, em um processo conhecido como tunelamento, destinado a abstrair informações.
- Cada seta deve receber um nome ou um código que a identifique unicamente.
- Os mecanismos podem ser suprimidos se isso não afetar a compreensão do modelo
- Só devem ser mencionados os elementos necessários para o objetivo da construção do modelo
- As saídas indicam um valor agregado as entradas e controles, ou ainda resultados colaterais, sub-produtos, ou “dejetos” dos processos.
- A borda de um diagrama representa a borda da atividade expandida. Todas as atividades são realizadas nas “folhas”, isto é, na última atividade modelada (mais detalhada). As atividades superiores são apenas abstração que não desempenham nenhum procedimento real.

Algumas heurísticas interessantes para se usar durante a modelagem{Richard J.Mayer, 1992 1999 /id}:

- Questione as fronteiras.
- Essa atividade cai dentro do escopo da atividade superior?
- Esta atividade está conforme o escopo e o ponto de vista estabelecido no projeto?
- Observe os limites numéricos do modelo (3 a 6 sub-funções por diagrama)
- Não procure sempre 6 atividade, descreve as atividades como elas aparecem no mundo real.
- Cuidado com excesso de ligação entre as atividades (teia de aranha), que indica a falta de organização dos nível de abstração das atividades

1.4.3.5 Passos da construção do modelo

A seguir, os passos a serem seguidos quando da construção de um modelo IDEF0:

1. Defina objetivo e motivação
2. Responda as seguintes perguntas
3. Por que o processo está sendo modelado?
4. O que esse modelo vai mostrar?
5. O que os leitores desse modelo poderão fazer com ele?
 - Exemplo: “Identificar as tarefas de cada funcionário da loja, entendendo como elas se relacionam em detalhe suficiente para desenvolver um manual de treinamento”
6. Desenvolva as questões que o modelo deve responder
 - Exemplos: “Quais as tarefas do atendente?”, “Quais as tarefas do arrumador?”, “Como os produtos circulam na loja?”
7. Desenvolva o ponto de vista
8. Defina o escopo do sistema
9. Dê um nome ao sistema
10. Use um nome condizente com o escopo definido
 - Normalmente o nome de um sistema utiliza termos bastante genéricos
11. Defina os ICOMs principais
12. Defina as saídas, incluindo as saídas que acontecem quando o processo não acontece de forma satisfatória
 - Todas as saídas possíveis do processo devem estar presente no modelo
13. Defina as entradas
 - As entradas devem ser processadas para gerar as saídas
 - Normalmente o nome de uma entrada não permanece o mesmo na saída
 - Algumas vezes entradas recebem adjetivos como “simples” ou saídas recebem adjetivos como “verificada” para demonstrar que apesar de não haver uma modificação houve um processamento da entrada para a saída.
14. Defina os mecanismos
15. Defina os controles
16. Lembre que todas as atividades possuem ao menos um controle
 - Controle existem na forma de regras, políticas, procedimentos, padrões, etc.

- No caso de indecisão entre entrada e controle, modele como controle.
17. Numere as atividades e diagramas
 18. Se necessário, decomponha as atividades
 19. Repita o processo de modelagem, mantendo a consistência
 20. Se necessário, construa modelos FEO (apenas para informação)
 - Por exemplo, para indicar outro ponto de vista
 - Para ilustrar detalhes que não são suportados pela notação IDEF0
 21. Controle o tamanho do diagrama a partir do escopo (principalmente controlando a extensão do diagrama)
 22. Controle a profundidade do diagrama a partir do detalhe necessário para o objetivo do modelo.

I.4.4 Exemplo de IDEF0

O texto a seguir representa a descrição de um processo executado em uma empresa de software. O nome desse processo é “Atender Relatório de Bug”. O processo envolve dois departamentos da empresa: o departamento de testes (DT) e o departamento de desenvolvimento (DD). As informações que devem ser guardadas estão sublinhadas. Os eventos iniciais ou finais estão em negrito.

O processo acontece da seguinte forma:

Tudo começa quando um **bug é relatado** ao departamento de testes (DT), que recebe um relatório de bug. O DT deve, então, avaliar a existência do bug. Se o bug não for confirmado, então ele não existe e deve ser feito um esclarecimento do problema para quem relatou o bug. Nesse caso, o processo acaba com um **bug negado**.

No caso do bug ser confirmado, então o DT deve determinar a prioridade do bug. Novamente, duas opções são possíveis: o bug tem prioridade imediata ou prioridade apenas para a próxima versão.

No caso de prioridade para próxima versão, o departamento de desenvolvimento (DD) prepara uma solicitação de melhoria e o processo é encerrado com o **bug aceito** (mas não corrigido).

No caso de prioridade imediata, enquanto o DT prepara os Casos de Teste para o bug, o DD corrige o problema (isso é feito em paralelo). Quando os casos de teste e a correção estão prontos, o DT deve testar a correção. Caso a correção determine que o bug foi resolvido, o DD deve lançar uma nova versão do software e o processo se encerra com o **bug corrigido**. Caso a correção determine que o bug não foi resolvido ou foi parcialmente resolvido, então ele deve voltar para a avaliação de prioridade.

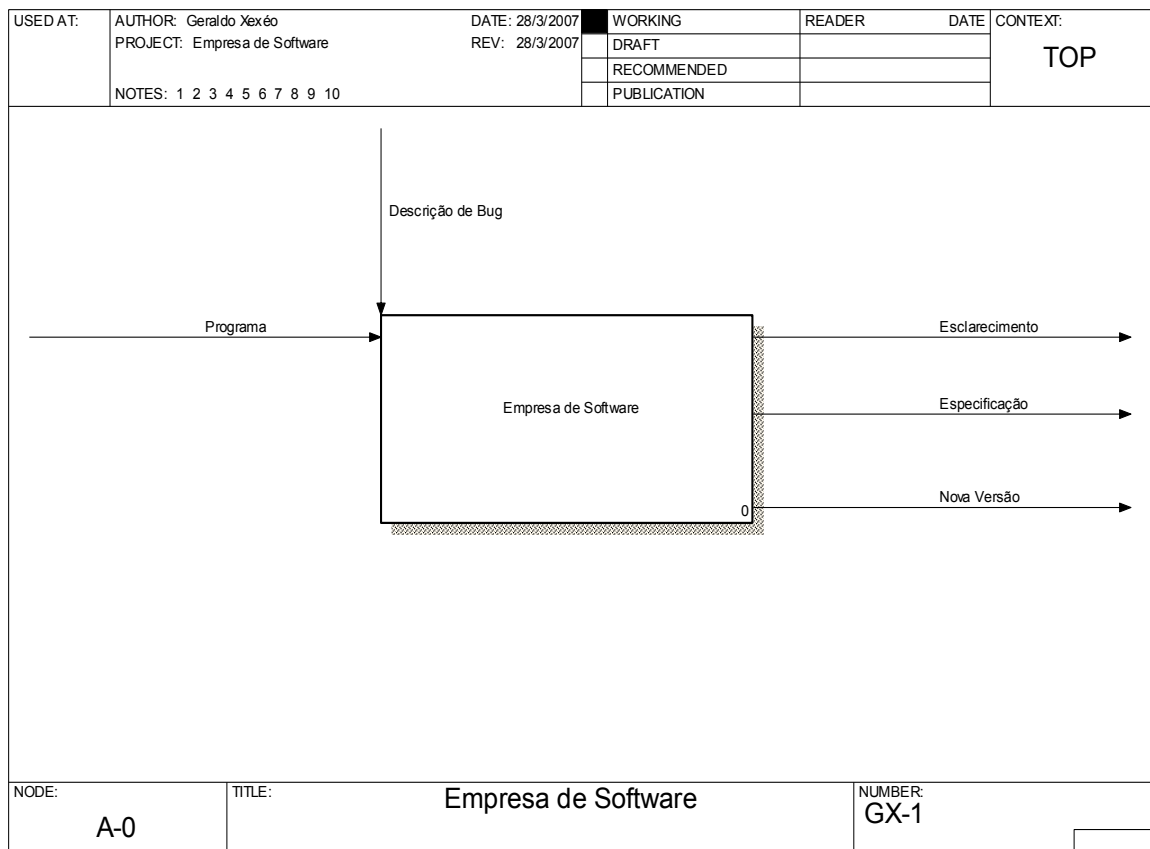


Figura 30. Diagrama de Contexto para o texto anterior.

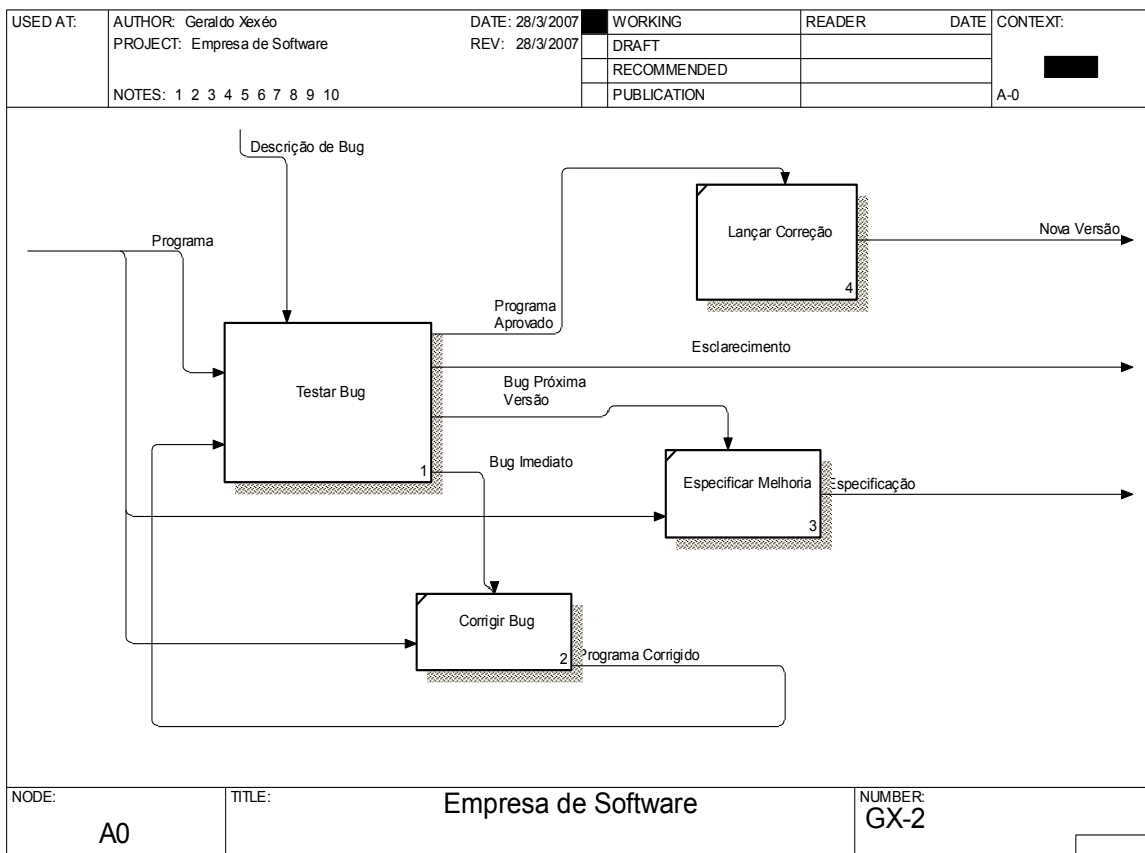


Figura 31. Diagrama de nível 0 para o texto anterior

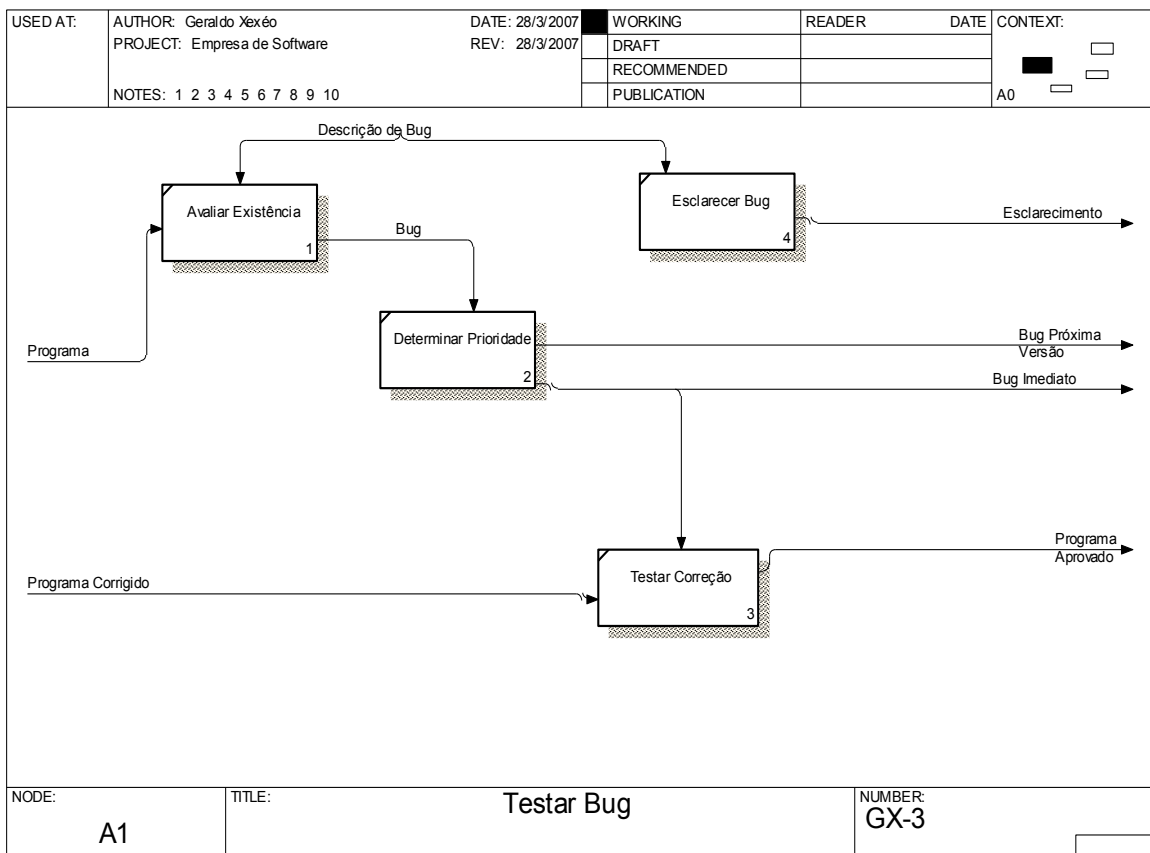


Figura 32. Diagrama A1 para o texto anterior.

I.4.5 IDEF0 Foram Abandonados?

Apesar de bastante informativos, IDEF0 não são mais uma ferramenta fortemente usada. Porém eles representam uma visão da empresa raramente vistas: o que ela faz, sem especificar como. Essa especificação também pode ser feita de forma simplificada com Diagramas de Fluxo de Dados.

I.5 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 [B41]. Ele pode ser utilizado para representar, em diferentes níveis de abstração, sistemas de informação, não necessariamente automatizados ou a funcionalidade de qualquer organização, como um IDEF0. DFDs também 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 [B42], e os que desenhavam processos com círculos, baseados na proposta de Coad e Yourdon [B43].

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.

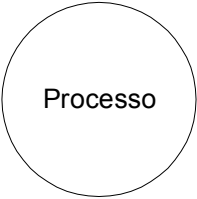
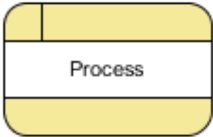
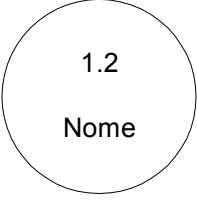
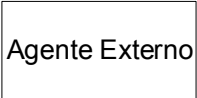
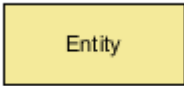
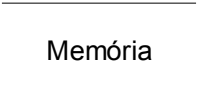
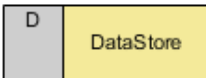


Figura de Yourdon	Figura de Gane/Sarson	Descrição
		Um processo identificado apenas pelo nome
		Um processo com número e nome
		Um Agente Externo identificado
		Uma memória identificada
		Um fluxo de dados sem identificação
		Um fluxo de dados identificado

Tabela 33. Símbolos e seus significados para a construção de um Diagrama de Fluxo de Dados

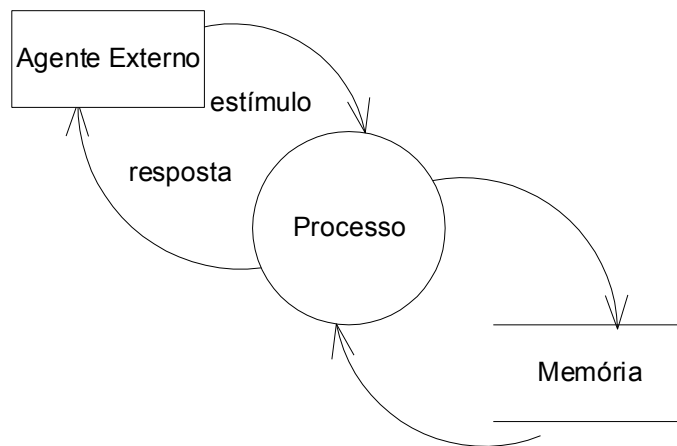


Figura 34. Um DFD genérico simples

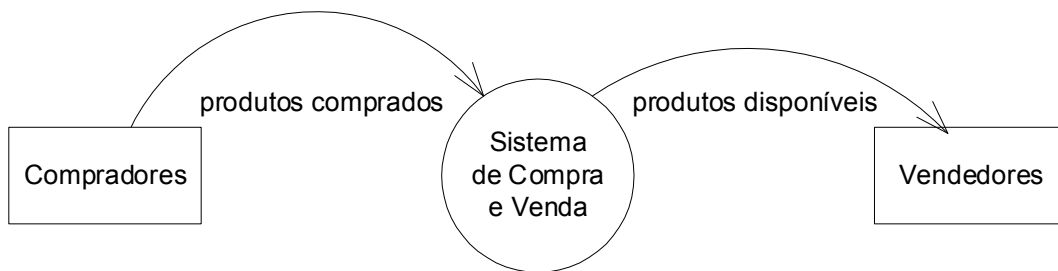


Figura 35. Um DFD de Contexto muito simples

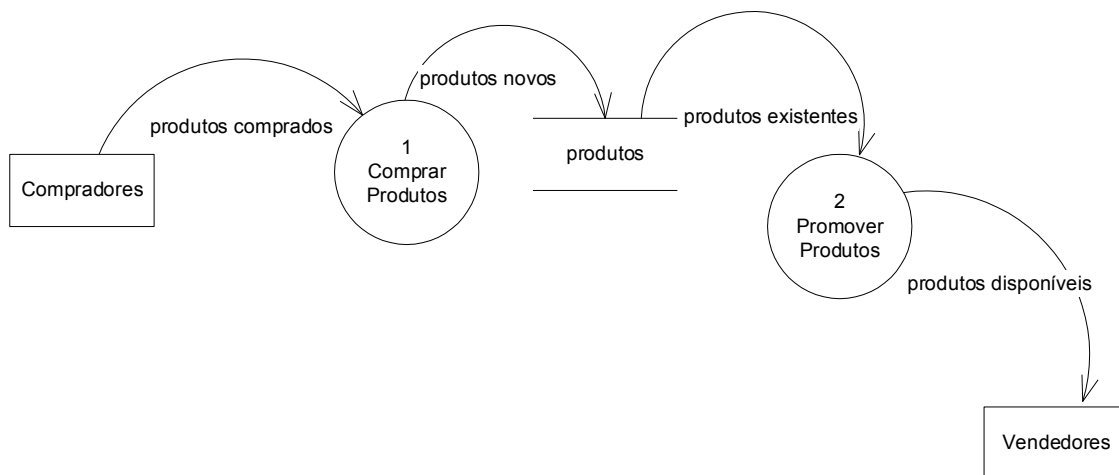


Figura 36. A explosão do DFD de Contexto da Figura 35

DFDs não descrevem uma seqüência de execução ou qualquer outra relação de controle¹³. 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).

1.5.1 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**.¹⁴

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 [B42], porém na notação de Yourdon, que utilizamos agora, **não possuem rótulos**.

1.5.2 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.

¹³ 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.

¹⁴ 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.

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 ativados de forma assíncrona. Devemos notar que não aparecem fluxos de dados entre processos em DFDs particionados por eventos.

I.5.3 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 atualidade¹⁵.

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.

I.5.4 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.

¹⁵ 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?).

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 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.

I.5.5 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.

I.5.6 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 ± 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.

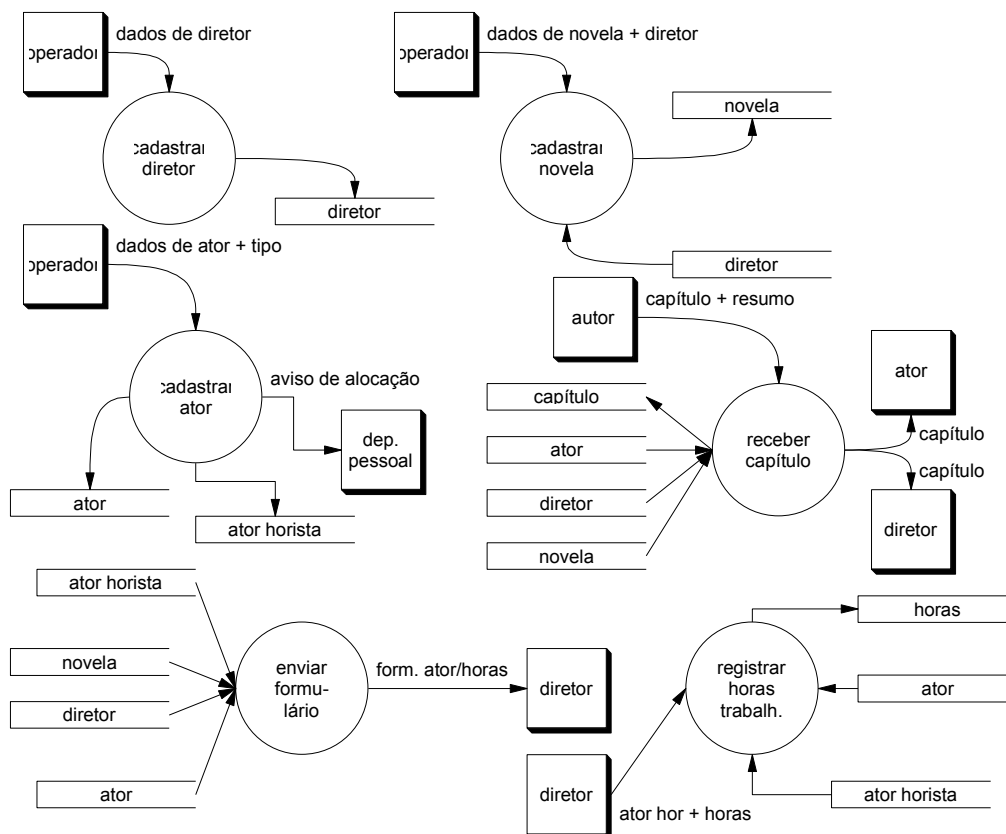


Figura 37. Todos os componentes do DFD particionado do sistema para Rede Bobo de Televisão¹⁶ em uma só imagem

I.6 Processos de Negócio

Processos de negócio são grupos de decisões e atividades, logicamente relacionadas, requeridas para o gerenciamento de recursos da empresa¹⁷.

Podemos entender “processos de negócio como uma seqüência 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*).”[B22] 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

¹⁶ 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.

¹⁷ Esta é a definição do BSP

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.

I.7 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.

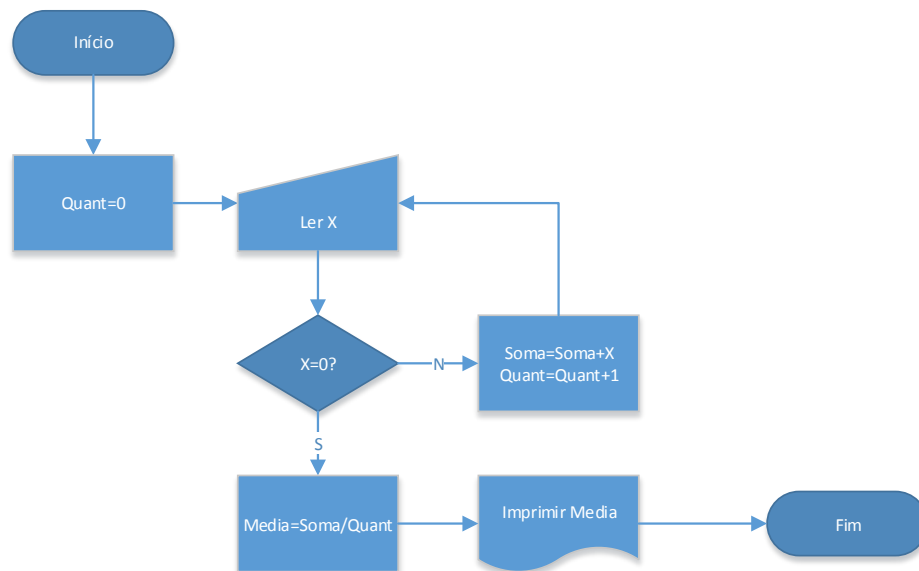


Figura 38. Um fluxograma que mostra o funcionamento de um programa de computador que calcula a média.

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 esta prática está totalmente superada, sendo normalmente substituído por programas em linguagens de alto nível. Seu uso na especificação de processos também está em franca decadência, nesse caso o que se vê é a sua substituição por diagramas mais modernos (incluindo o uso de fluxogramas em diagramas de raia¹⁸).

¹⁸ swimlane diagrams

I.8 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 atividades.

As principais primitivas, descritas na figura abaixo, são:

- **Atividades**, que representam funções, decisões, tarefas, processamento de informações ou outros passos do processo que precisam ser executadas e suportam um ou mais objetivos de negócio.
 - Possivelmente
 - São iniciadas ou iniciadas ou habilitadas por eventos,
 - Geram eventos
 - Consomem recursos,
 - Exigem gerenciamento, tempo, e atenção.
 - Podem representar:
 - Atividades tangíveis
 - Decisões (mentais)
 - Processamento de Informações
 - Um processo que é descrito por outro diagrama EPC.
 - Nesse caso, as ferramentas CASE fornecem alguma marca indicativa de que a função pode ser expandida.
- **Eventos**, que representam situações, ou estados do sistema, antes ou depois da execução de uma função.
 - Eventos podem ser
 - Ativadores de uma atividade
 - Ativados por uma atividade
 - “Estados comercialmente relevantes, que controlam ou influenciam a progressão subsequente de um ou mais processos de negócio”¹⁹
 - Um evento não consome tempo nem recursos por si só.
- **Regras**, que permitem a unificação e separação de fluxos segundo os conceitos de E, OU ou OU-exclusivo.

¹⁹ <http://www.ariscommunity.com/help/aris-express/35908/>

- **Interface de Processo**, que indica que os próximos passos são descritos por meio de um outro diagrama .




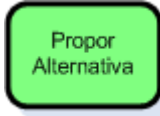


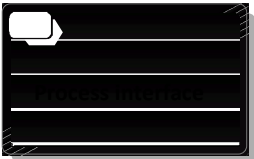
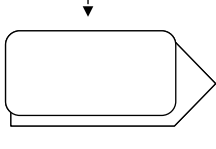
Tipo	Símbolo ARIS Express	Símbolo Tradicional
Evento		
Atividade		
Regras		
Interface de Processo		

Figura 39. Principais componentes de um EPC

EPCs (e eEPCs) são modelados semanticamente como Redes de Petri. O leitor interessado pode encontrar na Internet vários documentos tratando do assunto e demonstrando problemas semânticos no uso de algumas construções.

I.8.1 Uso Básico

No diagrama abaixo, podemos ver um exemplo simples de EPC seguindo as regras básicas de sintaxe.

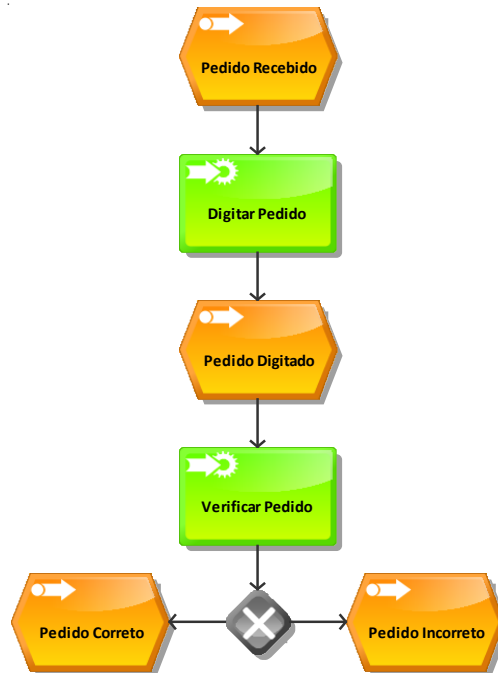


Figura 40. Exemplo de um EPC de um processo de recebimento de pedido

A seguir, descreveremos as formas básicas de uso dos componentes de um diagrama EPC.

1.8.1.1 Atividades (ou Funções)

A seguir, apresentaremos algumas regras básicas.

O nome de uma atividade é sempre da forma:

<verbo no infinitivo> <objeto direto>

Como, por exemplo, na figura a seguir:

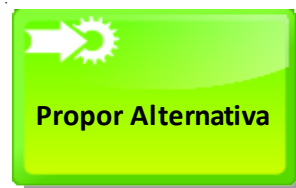


Figura 41. Exemplo de um processo.

As atividades podem ser vistas como transformações. Elementos que descrevem uma atividade são:

- Informação que entra
- Objetivos
- Recursos

- Consumíveis, como matéria prima
- Não consumíveis, como máquinas
- Procedimentos realizados
- Capacidades necessárias
- Informação que sai
- Produto ou serviço entregue

1.8.1.2 Eventos

Os Eventos representam a mudança do estado do mundo enquanto o processo é executado²⁰. Três são os tipos de eventos candidatos a serem registrados em um processo²¹:

1. A mudança de estado do mundo que faz com que o processo comece;
2. Mudanças internas de estado enquanto o processo é executado, e
3. O resultado final do processo que tem um efeito externo.

Claramente, eventos definem pré-condições para que uma atividade ocorra, e também pós-condições ao fim da atividade.

Um evento segue a sintaxe:

<Sujeito> <Verbo na voz passiva>

Ou, mais simplesmente:

<Substantivo> <Adjetivo>

Como os exemplos da figura a seguir:

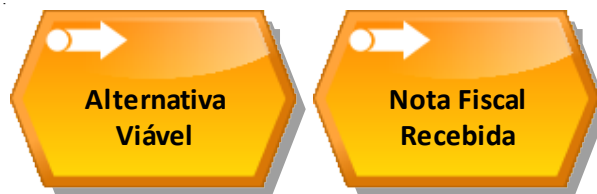


Figura 42. Exemplos de eventos

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. Logo, os diagramas das figuras a seguir podem ser usados como equivalentes.

²⁰ Aris Design Platform

²¹ Idem

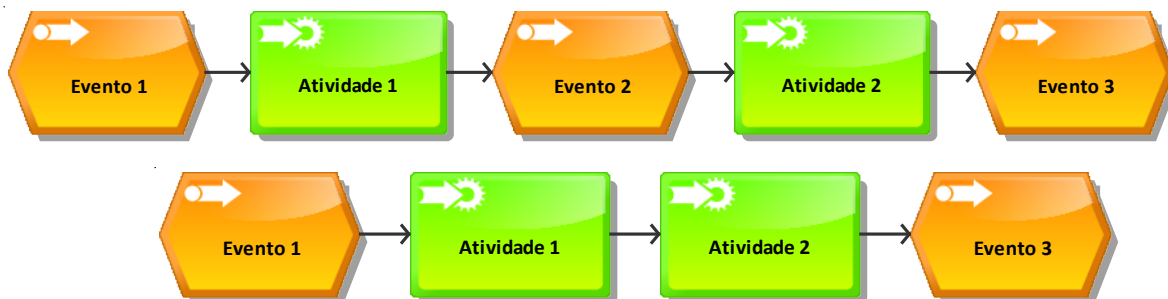


Figura 43. Duas cadeias que podem ser usadas de forma equivalente na notação atual de EPC. O evento intermediário, como não serve para indicar caminho, pode ser dispensado.

1.8.1.3 Conectores

Os conectores E, OU e OU-EXCLUSIVO funcionam de duas formas: como divisores de caminho (*split*) e como junção de caminhos (*join*). Não é possível utilizar um conector simultaneamente nas duas formas.

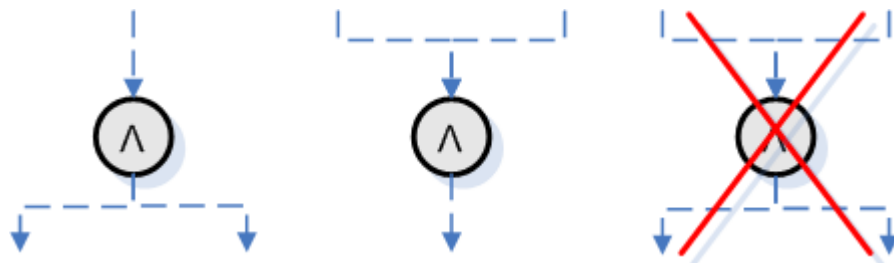


Figura 44. Um conector só deve ser utilizado na forma de divisor ou junção, e nunca das duas formas ao mesmo tempo.

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.

As figuras a seguir apresentam as configurações permitidas e proibidas para o uso de conectores OU-EXCLUSIVO. As regras para conectores OU são equivalentes. Para conectores E não há configurações proibidas. Sempre que apresentamos 2 eventos ou 2 processos é possível utilizar mais desses componentes.

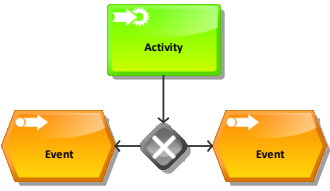
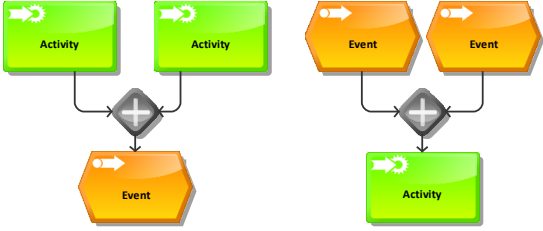
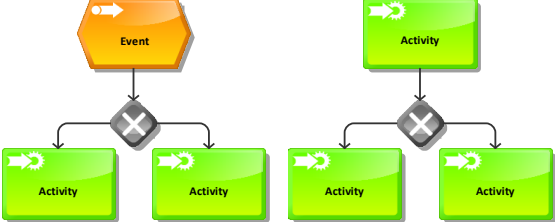
	<p>Configuração permitida para o conector OU-EXCLUSIVO (XOR), funcionando como marca da necessidade de escolher entre dois ou mais caminhos (como um if-then-else ou case). Os eventos indicam que caminho deve ser seguido. Os eventos não podem acontecer ao mesmo tempo.</p>
	<p>Configuração permitida para o conector OU-EXCLUSIVO (XOR), funcionando como ponto de convergência (join) entre dois caminhos que não serão atravessados simultaneamente.</p>
	<p>Configurações não permitidas para o conector OU-EXCLUSIVO (XOR), pois não é possível escolher que caminho seguir.</p>

Tabela 2. Configurações para o conector OU-EXCLUSIVO (XOR).

Conectores em branco podem ser utilizados para representar em um diagrama mais simples um conjunto de conexões muito complexas. Veja o exemplo nas figuras a seguir.

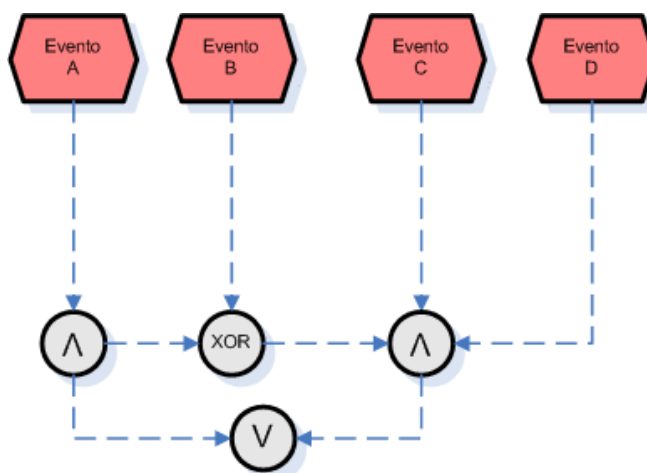


Figura 45. Exemplos de decisão complexa.

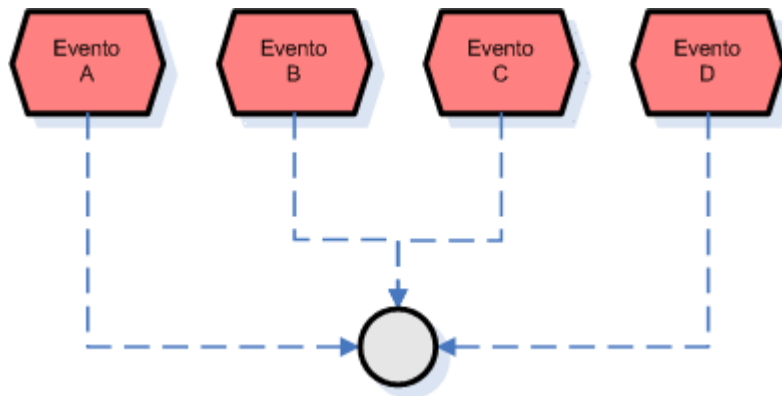


Figura 46. Exemplo de uso de conector em branco para esconder uma decisão complexa.

1.8.1.4 Interfaces de Processo

As interfaces de processos são usadas para indicar que um processo acontece quando outro processo acaba. Caminhos são sempre conectados a eventos, devendo aparecer antes ou depois deles, e recebem o nome do processo origem ou destino. Na figura a seguir é possível ver um exemplo de sua utilização.

Elas só devem ser usadas para ligar processos de um mesmo nível.

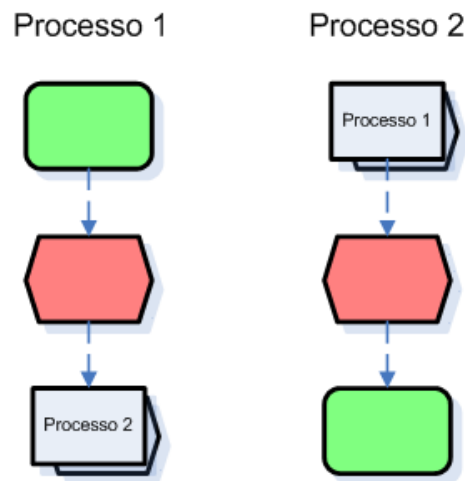


Figura 47. Como indicar que um processo é continuação lógica de outro processo. No “Processo 1” indicamos um caminho para o “Processo 2”, e nesse processo indicamos que o caminho é proveniente do “Processo 1”.

1.8.2 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.

Esses elementos adicionais funcionam basicamente como comentários ao processo que está sendo documentado. Assim, depois de descrito o processo pelo método

não estendido, colocamos sobre eles novos elementos documentando informações como quem realiza o processo, que informação utiliza, que produtos gera ou consome, etc...

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.
- Produtos ou serviços, que são gerados ou consumidos pelo processo.
- Objetivos, que representam o motivo da realização de um processo ou tarefa








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

Figura 48. Elementos complementares dos diagramas eEPC.



Figura 49. Nova lista e imagens dos elementos complementares dos diagramas EPC.

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

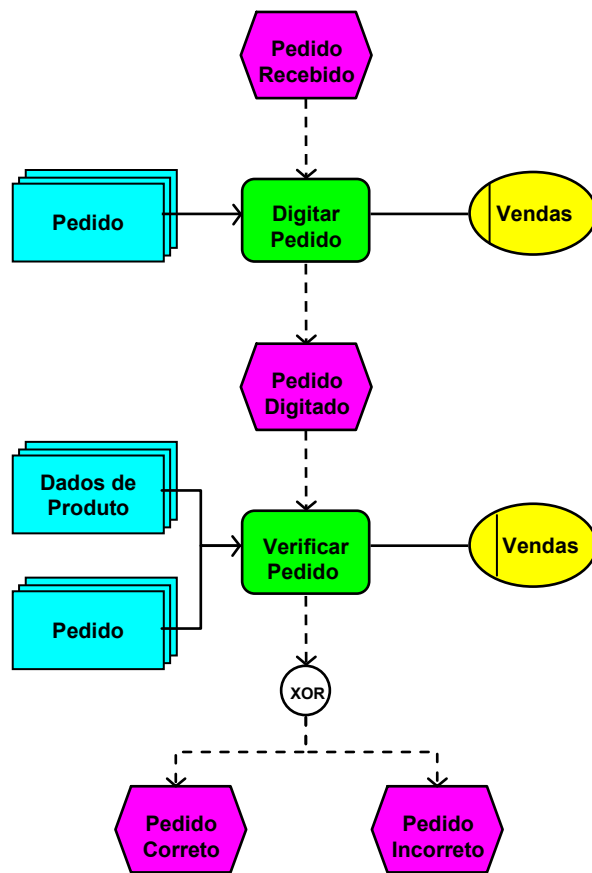


Figura 50. Exemplo de eEPC, a partir do EPC anterior. Os símbolos suplementares podem variar de acordo com a ferramenta case sendo usada.

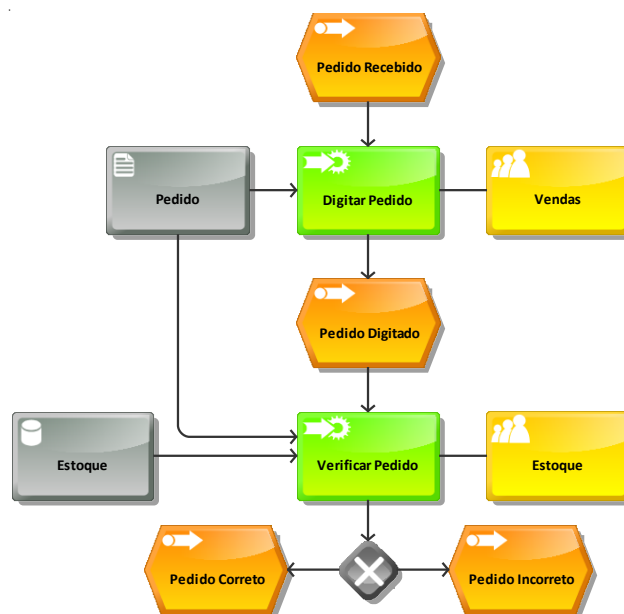


Figura 51. Desenho novo do modelo EPC

Algumas formas dos diagramas EPC estendidos apresentam várias versões. Por exemplo, o símbolo de informações apresenta pelo menos 3 versões diferentes, como na figura a seguir.



Figura 52. Diferentes versões para informação

A figura a seguir apresenta um exemplo usando alguns símbolos diferenciados.

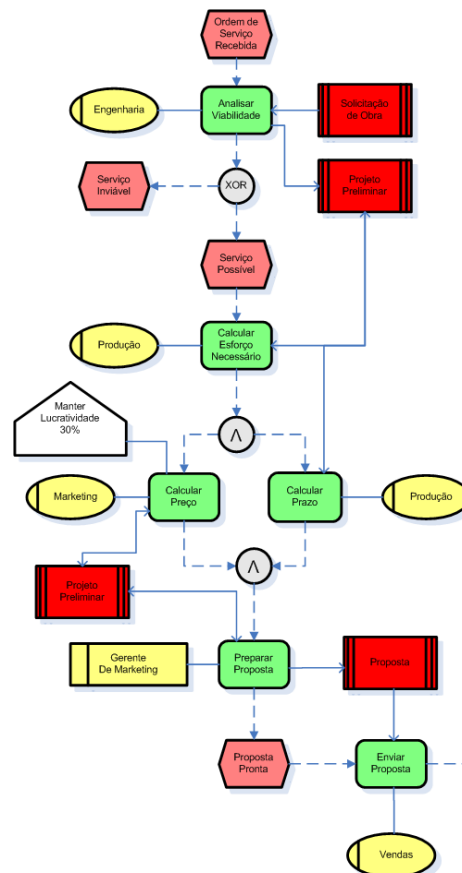


Figura 53. Um exemplo de EPC com símbolos de informação diferenciados, e ainda um objetivo indicado.

1.8.3 Diagramas de Resumo

Também é possível desenhar diagramas representando o resumo de um processo, como o exemplo a seguir.

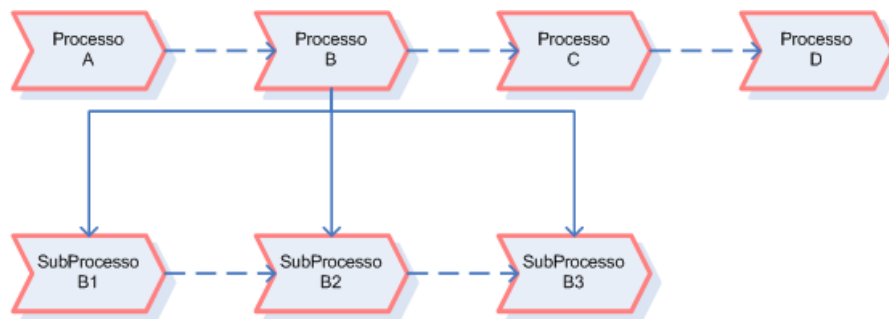


Figura 54. Um resumo de processo. A linha tracejada indica fluxo de controle, a linha cheia indica divisão hierárquica.

I.8.4 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.

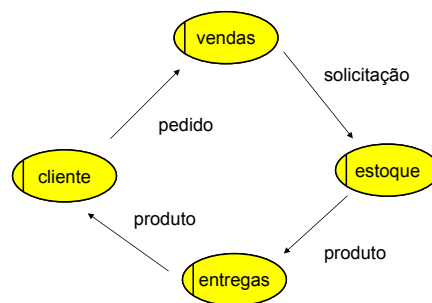


Figura 55. EPCe baseado apenas em unidades da organização.

Os mesmos símbolos também podem ser usados para construir organogramas.

I.8.5 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.

I.8.5.1 Passos para construir modelos EPC/EPCe[B23]

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.

I.8.6 Regras de ouro de EPCs[B23]

- Não existem nós isolados
- 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.
- Evite o uso de OU como junção, pois sua semântica não é clara.
- Cuidado com modelos complexos, que podem levar a *deadlocks*, principalmente com o uso do operador E.
- Mantenha o diagrama estruturado. Sempre que dividir um caminho com XOR ou E, feche o caminho com o mesmo símbolo.

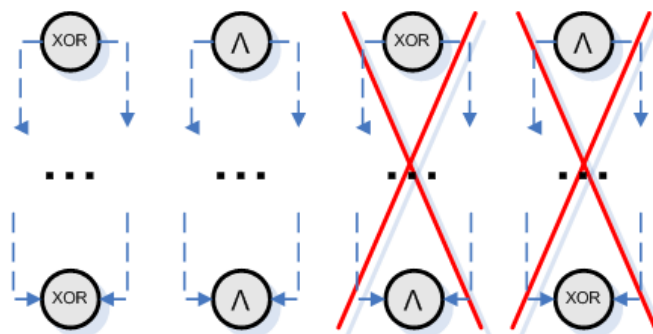


Figura 56. Manter os diagramas estruturados significa fechar cada bloco de caminhos com um conector do mesmo tipo que abriu esse bloco. Isso evitará deadlocks.

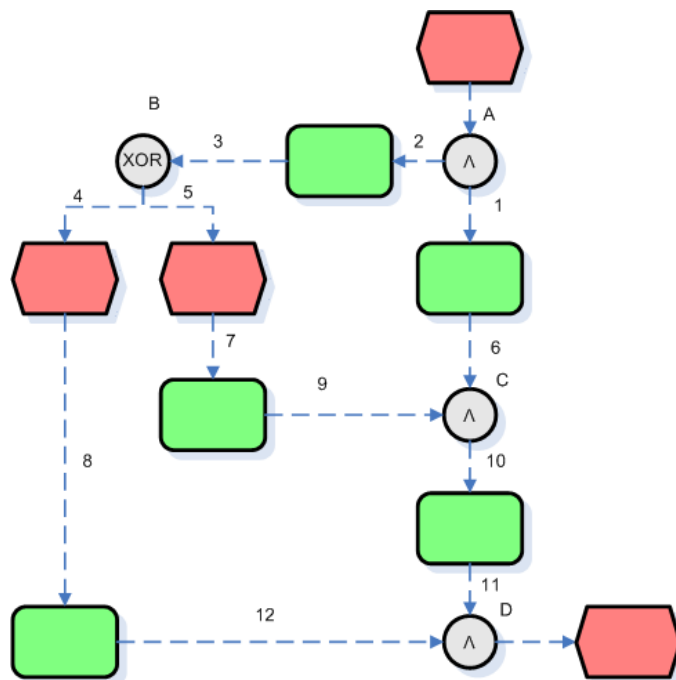


Figura 57. O modelo acima, demonstra um *deadlock*. Como comentário, os números identificam as setas e as letras identificam os conectores²². No conector XOR (B), um dos caminhos 4 ou 5 será escolhido. Porém, se for escolhido o caminho 5, o sistema para de funcionar no conector E (D), pois esse E nunca será satisfeito (já que 4-8-12 não será percorrido). Caso seja escolhido o caminho 4, o conector E (C) nunca será satisfeito (pois 5-7-9 não será percorrido), e o sistema vai parar em (C), esperando 9, e (D), esperando 11.

I.8.7 EPC e Loops/Laços

Por diferentes motivos, laços arbitrários não funcionam bem com EPCs. Alguns textos explicitamente proíbem laços, outros não apresentam exemplos, desconsiderando-os implicitamente. Porém, é possível descrever laços em EPML, a linguagem XML para descrever EPCs.

Fazemos as seguintes recomendações quanto ao uso de laços nos diagramas EPC:

- Evite os laços
 - Use apenas os laços estritamente necessários
- Use apenas laços simples, baseados em XOR
 - Busque usar laços apenas em processos de correção e similares.

²² Isso não faz parte da notação, está sendo usado apenas para explicar o *deadlock*.

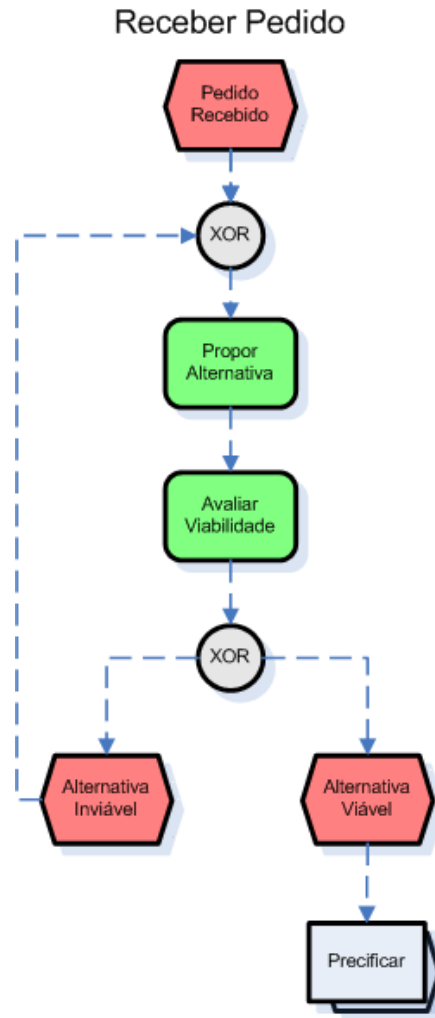


Figura 58. Um EPC válido, com um loop simples baseado em XOR. Também é dado um exemplo de como conectar esse EPC a um outro processo.

I.8.8 Algumas sentenças em EPC

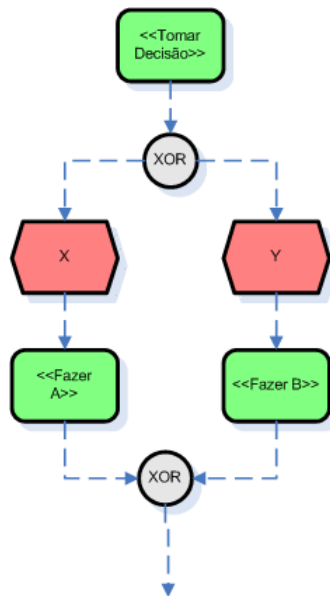


Figura 59. Descrição EPC para “Se X então Fazer A, Se Y, então Fazer B”

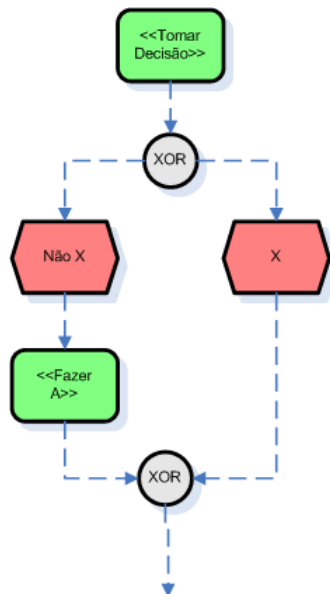


Figura 60. Descrição EPC para “Se Não X então Fazer A”

I.8.9 Exemplo

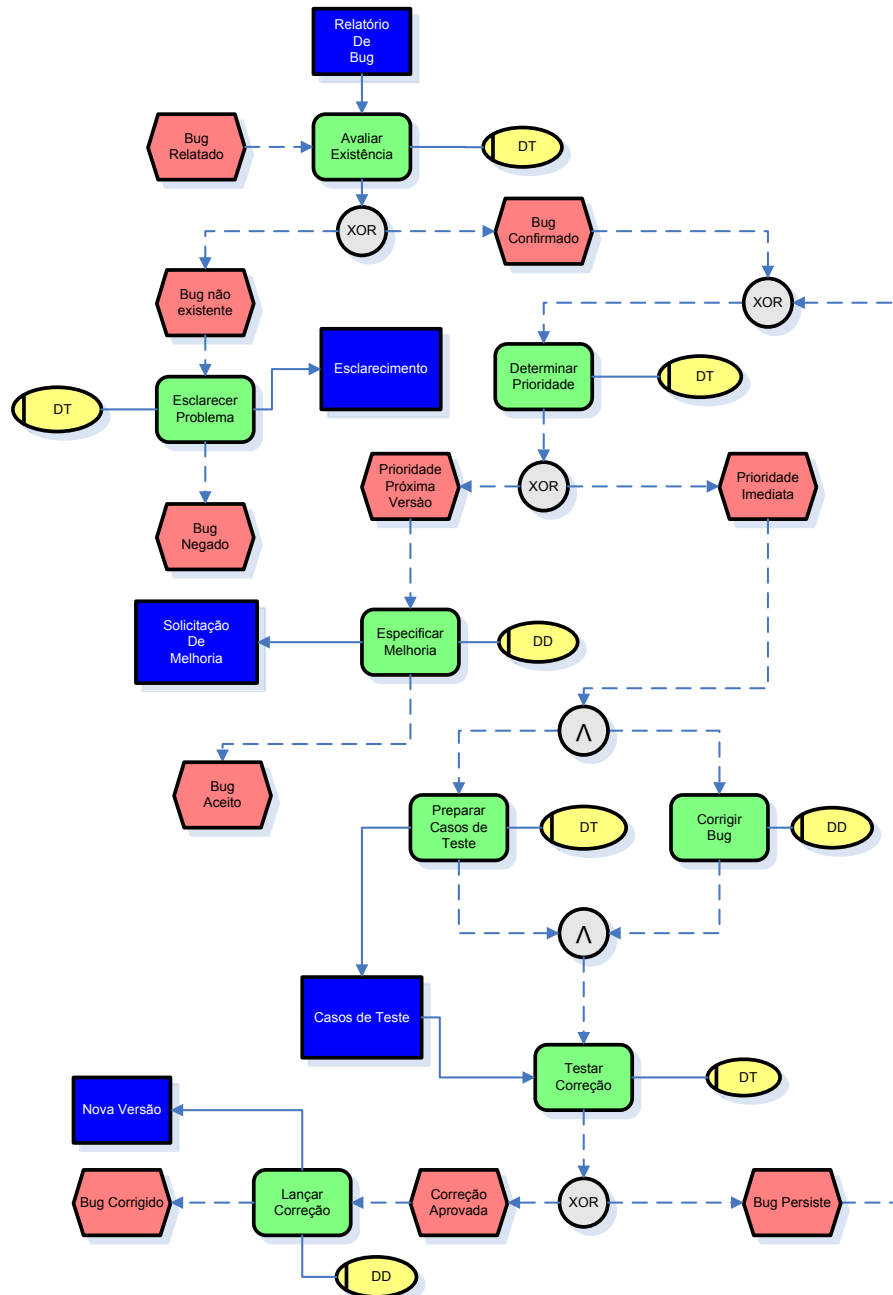


Figura 61 EPC para o texto apresentado em I.4.4

I.9 Diagramas de Atividade (UML 2.0)

O Diagrama de Atividade (DA) é uma das formas que UML [B24] propõe para modelar os aspectos dinâmicos de um sistema, sendo basicamente um tipo de avançado de fluxograma mostrando como o controle e dados fluem entre atividades. A partir da versão 2.0 um DA tem a semântica de uma Rede de Petri.

Segundo o padrão da UML 2.0, modelar uma atividade tem como ênfase modelar coordenação de comportamentos de mais baixo nível, as ações. Isso significa modelar a sequência e a condição em que esses comportamentos ocorrem.

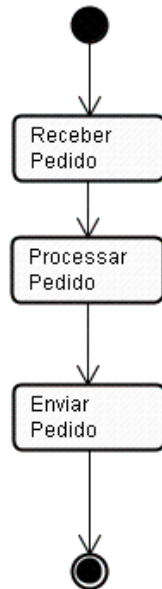


Figura 62. Exemplo de um Diagrama de Atividades simples, com uma sequência de três ações.

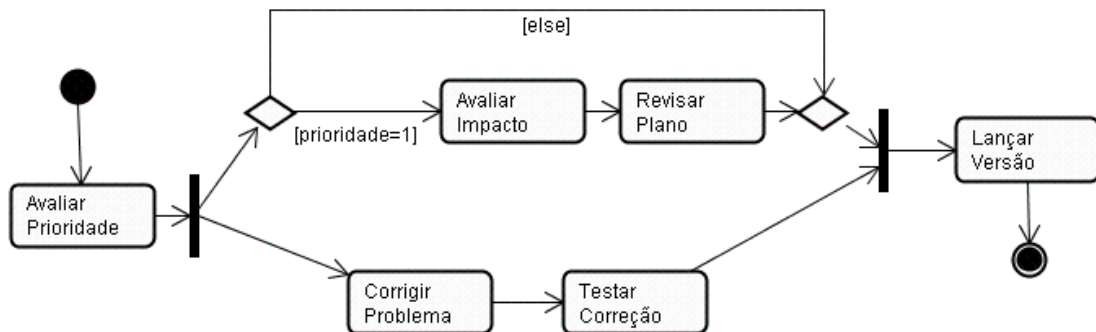


Figura 63. Exemplo de um diagrama de atividades mais complexo, contendo decisões e paralelismo.

I.9.1 Nós, vértices

Um DA é um grafo composto de nós e arestas direcionadas (isto é, que partem de um nó e chegam em outro nó, como uma seta). Os nós representam os passos em uma atividade e podem ser nós executáveis (ou ações), nós de controle ou objetos. As arestas representam fluxos, de dados ou de controles. Na Figura 62 podemos ver três tipos de nós (nó de início de atividade, nó de fim de atividade e três ações) e um tipo de vértice (de fluxo de controle).

A partir da especificação UML 2.0, Diagramas de Atividades são semelhantes a Redes de Petri. O significado disso pode ser descrito da seguinte forma: em qualquer momento, um ou mais nós de um DA podem estar marcados com “tokens”, isto é,

“marcas”. Uma marca pode ser entendida, de forma geral, como dizendo que aquele nó “está ativo” ou “é o passo corrente”.

A semântica de uma atividade é baseada no fluxo de tokens. Um token contém um objeto, um dado ou uma indicação de controle e está presente em um nó específico da atividade. Quando um nó começa a executar, ele aceita tokens de todos ou apenas alguns das arestas que apontam para ele. Quando termina a execução de um nó o token é removido desse nó e oferecido para todos ou apenas alguns das arestas que partem dele.

Por exemplo, na Figura 64 podemos ver um token caminhando entre os vários nós do diagrama, de acordo com

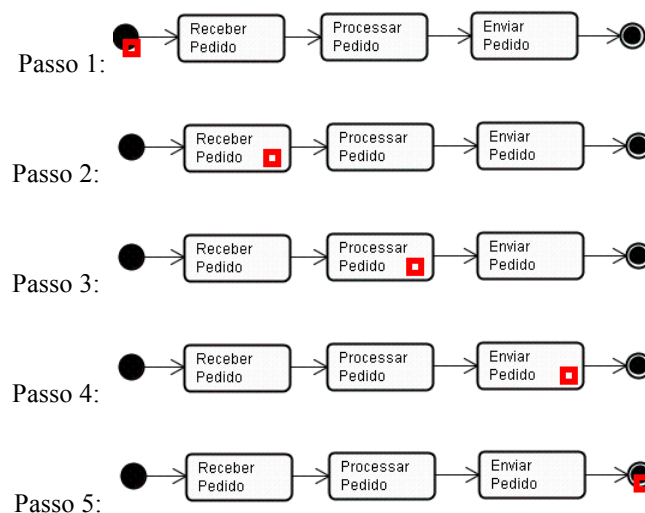


Figura 64. Um “token” vai caminhando no Diagrama de Atividades, indicando o nó ativo a cada instante.

I.9.2 Ações, Objetos e Fluxos

O componente básico do Diagrama de Atividades é a **ação**²³. Uma ação representa um passo simples que não é mais decomposto dentro do modelo da atividade sendo descrita. É importante notar que uma ação é simples apenas do ponto de vista da atividade sendo modelada naquele instante, mas pode implicar em muitos outros passos detalhados (e muitos outros níveis de atividades) em outros níveis de abstração mais detalhados. Em especial, uma ação pode representar uma chamada para outra atividade. As atividades, dessa forma, são reutilizáveis, mas as ações só existem no contexto da atividade em que aparecem.

Uma ação é descrita em um DA na forma de um retângulo com os cantos arredondados e um rótulo, com seu nome, no meio.

²³ As ações são um conceito novo em UML 2.0, tomando o lugar dos conceitos *ActionState*, *CallState* e *SubactivityState* existentes na UML 1.5.

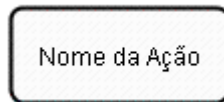


Figura 65. Símbolo para uma ação (com texto).

Pré e pós condições podem ser indicados usando notas (comentários) ligados a ação, com as palavras chaves `<<localPrecondition>>` e `<<localPoscondition>>`, como mostra o exemplo a seguir. Uma pré-condição sempre deve ser verdade antes da ação iniciar, enquanto uma pós-condição sempre será verdade após a ação ser executada.

Em relação a semântica do DA, uma ação só se inicia quando todas as arestas que chegam nela contém tokens. Quando a execução da ação está completa, ela oferece tokens para todas as arestas saindo dela. É interessante notar que se vários *tokens* estão disponíveis em uma mesma aresta, todos são consumidos.

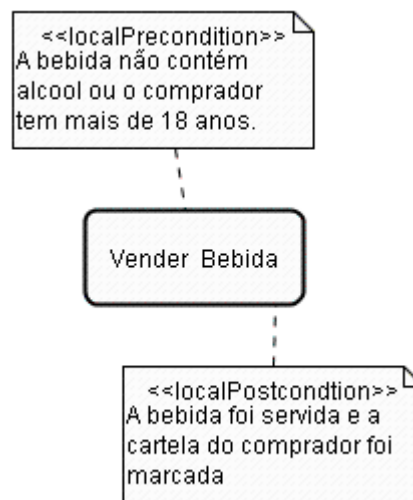


Figura 66. Exemplo de uso de comentários para indicar pré e pós condições.

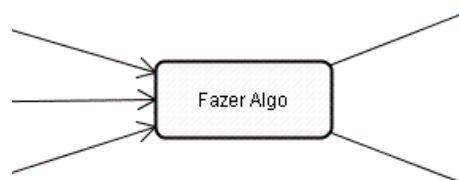


Figura 67. A ação “Fazer Algo” só se inicia quando todas as setas que chegam nela estão habilitadas. Quando ela termina, habilita todas as setas que partem.

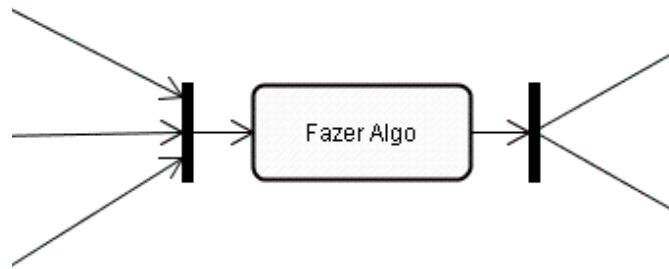


Figura 68. A descrição da ação “Fazer Algo”, feita na figura anterior, equivale a usar um *join* e um *fork* na sua entrada e saída, respectivamente.

Uma atividade é então um conjunto (ou seqüência) de ações, coordenadas de alguma forma. As arestas, ou fluxos, servem para indicar como as ações são seqüenciadas. Existem dois tipos de fluxos: fluxos de controle e fluxos de dados/objetos.

Os fluxos são indicados por meio de setas. Os fluxos de controle são indicados por setas simples e transportam apenas “tokens” de controle, como na Figura 69.

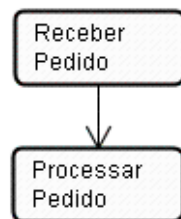
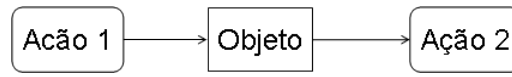
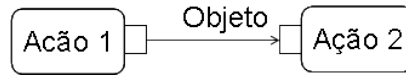


Figura 69 Notação para um fluxo de controle, usando apenas uma seta simples.

Um fluxo de objetos é representado por meio de três notações possíveis, apresentadas na Figura 70. Na primeira notação o objeto pertence ao fluxo. Na segunda notação, aparecem os “pins”, que podem ser de entrada ou de saída, alternativamente. Por um fluxo de objetos só pode passar um “token” de objeto. Na última notação, o objeto é “elidido”.



(notação com um objeto explícito)



(notação usando “pins”)



(notação com o objeto indicado)

Figura 70. Notações alternativas para fluxos de dados ou objetos.

Na primeira o objeto aparece explicitamente. Na segunda, são usados “pins”, as pequenas caixas junto as ações, e a seta recebe um rótulo com o nome do objeto ou dado. Na terceira o objeto é só indicado, mas não identificado.

Um fluxo de objeto entrando em uma ação representa um dado ou objeto necessário para a ação executar.

Na notação anterior apresentamos mais um tipo de nó, o objeto. Um objeto pode ser usado de muitas formas, sendo a primeira delas representar um objeto que passa por um fluxo de dados. Nesse caso, o objeto guarda apenas um token. Mais adiante no texto veremos outros uso para nós do tipo objeto.

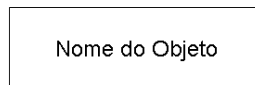


Figura 71. Representação de um objeto

Uma ação, para iniciar, deve ter disponíveis tokens em todos os seus fluxos de entrada, sejam de dados ou de controle²⁴. Ao terminar, deixa tokens disponíveis em todos os seus fluxos de saída²⁵.

Cada fluxos pode conter uma **expressão de guarda**, ou simplesmente uma **guarda**. Uma guarda é uma expressão lógica que indica se a seta pode ou não ser usada. A partir da versão 2.0, setas do DA não podem mais indicar eventos ou ações. As expressões de guarda são particularmente usadas junto aos nós de decisão.

Os fluxos apresentam algumas outras características que apresentaremos mais adiante no texto.

²⁴ Funcionando na prática como uma sincronização (join) implícita.

²⁵ Funcionando na prática como uma paralelização (fork) implícita.

1.9.3 Controle

Para caracterizar o comportamento de um diagrama de atividades é necessário controlar a sequência de execução das ações de alguma forma. O DA fornece dois mecanismos básicos para isso: a escolha de um entre vários caminhos possíveis e o seguimento de vários caminhos possíveis simultaneamente (paralelismo). Esses comportamentos, e outros necessários a uma melhor descrição de uma atividade, são descritos por nós de controle.

Os nós de controle podem ser de sete tipos, conforme a tabela a seguir:






Tipo do Nó	Gráfico correspondente
Início	
Fim de Fluxo	
Fim de Atividade	
Fork, ou início de paralelismo	
Join, ou sincronismo	
Decisão	
Merge, unificação	

Tabela 3. Tipos de nós de controle e seus gráficos

Um DA se inicia no “nó de **início**”. Podem existir vários desses nós, e todos são ativados simultaneamente. Um nó de início é indicado por um círculo preto (Tabela 3). De acordo com o padrão 2.1, um token é colocado em cada nó de início quando a atividade inicia.

Os nós de “**fim de fluxo**” e “**fim de atividade**” servem, respectivamente, para indicar o fim de um fluxo de controle ou o fim de toda a atividade. Quando um token chega a um nó de fim de fluxo, o token é destruído e o caminho seguido por ele acaba. Quando um token chega a um nó de fim de atividade todos os tokens são destruídos e atividade é encerrada.

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 de guarda que indique a decisão. A partir da versão 2.0 o nó de decisão passou a ser obrigatório, não sendo possível representar uma decisão diretamente a partir de uma ação. Pela semântica do DA, o nó de decisão recebe um token e o envia pela aresta que contém uma expressão de guarda que seja verdadeira. Segundo o padrão, apenas uma expressão de guarda será escolhida. Caso várias expressões de guarda sejam verdadeiras, não existe uma semântica definida.

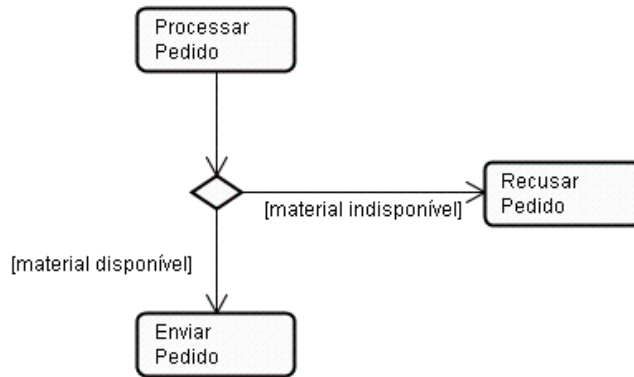


Figura 72. Fragmento de diagrama de atividade mostrando uma decisão.

Os nós “fork” e “join” permitem a execução paralela de ações, e sua possível sincronização (*fork* e *join*) futura. É importante lembrar que uma sincronização (*join*) sempre espera por todas as arestas terem tokens disponíveis, e as consome simultaneamente (o que evita *deadlocks*).

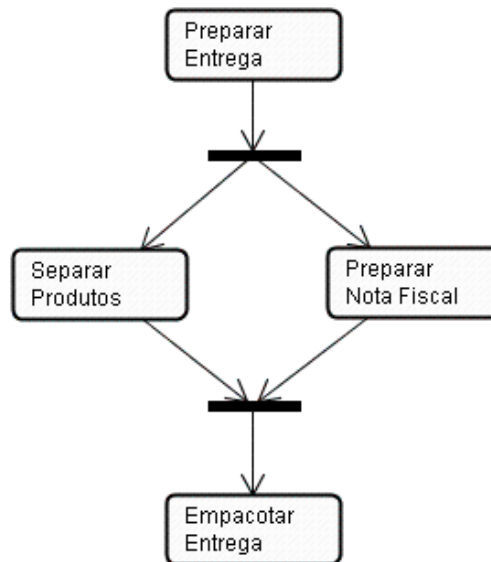


Figura 73. Exemplo de caminhos em paralelo e sincronização. “Empacotar entrega” só poderá executar após “Separar produtos” e “Preparar Nota Fiscal” completarem.

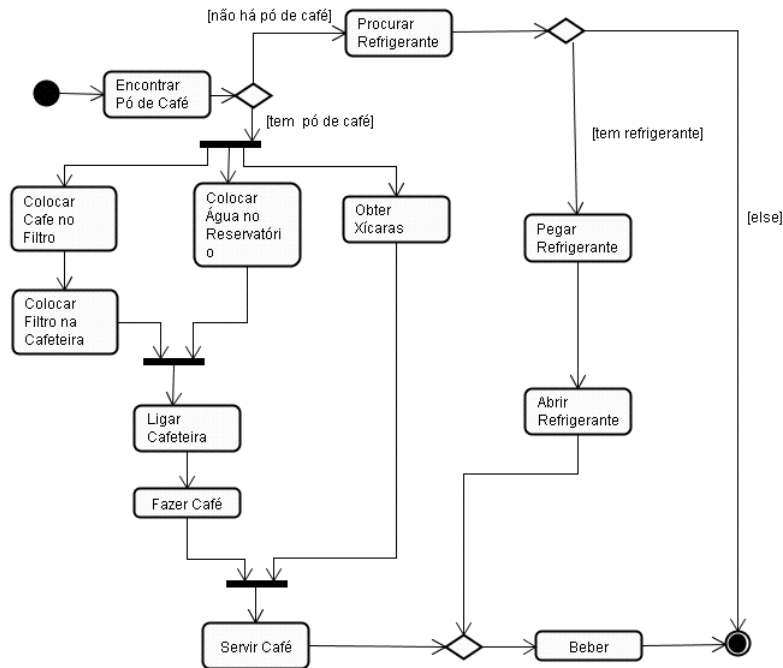


Figura 74. Exemplo de diagrama de atividades descrevendo o processo de tomar um café ou um refresco em um intervalo de trabalho, segundo a versão 2.1 do padrão UML. Note a necessidade de um nó de unificação antes da ação “Beber”. Caso os fluxos que entram nesse nó entrassem direto na ação “Beber”, essa nunca seria executada (*deadlock*), pois seria necessário que os dois fluxos contivessem tokens, o que nunca poderá acontecer, já que apenas um dos caminhos será escolhido.

I.9.4 Interrupções, Exceções e Sinais

Em alguns sistemas é importante modelar a capacidade de receber e emitir sinais. Por exemplo, durante o processamento de um pedido, a empresa pode receber um sinal que indica o seu cancelamento.

Outra capacidade importante é a modelagem de exceções. Na figura a seguir mostramos uma exceção que interrompe um processo ao recebimento de um sinal de cancelamento.

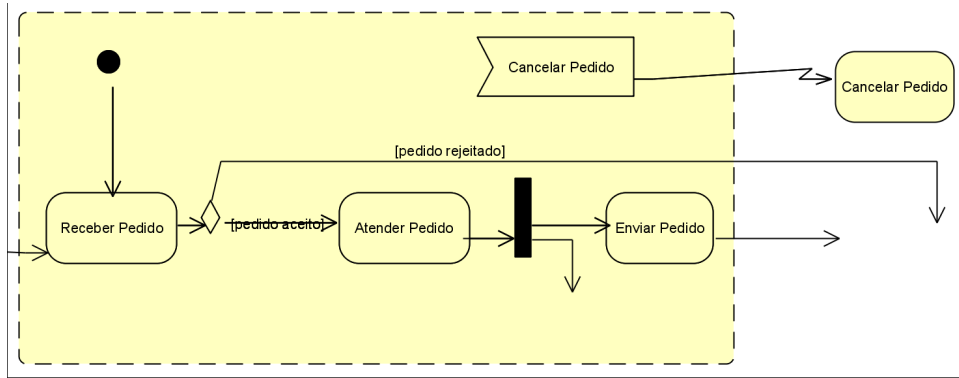


Figura 75. O detalhe de diagrama (UML 2.0) acima apresenta uma área onde pode acontecer uma interrupção, indicada pela ação de recepção de eventos “Cancelar Pedido”. Caso isso aconteça, será lançada uma exceção que invocará a ação “Cancelar Pedido”. A exceção é indicada pela seta com formato de raio. Uma exceção interrompe todas as ações dentro da área de interrupção.



Figura 76. Símbolos para envio de sinais e recepção de eventos (sinais) em UML 1.5

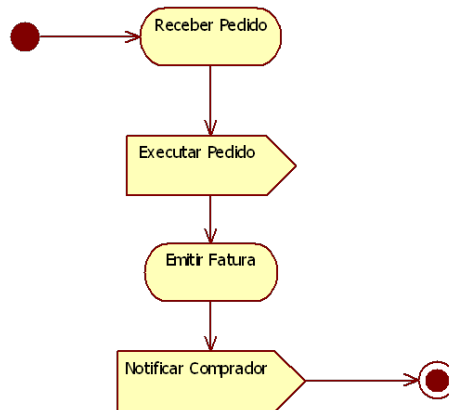


Figura 77. Exemplo de envio de sinais (UML 2.0).

I.9.5 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*).

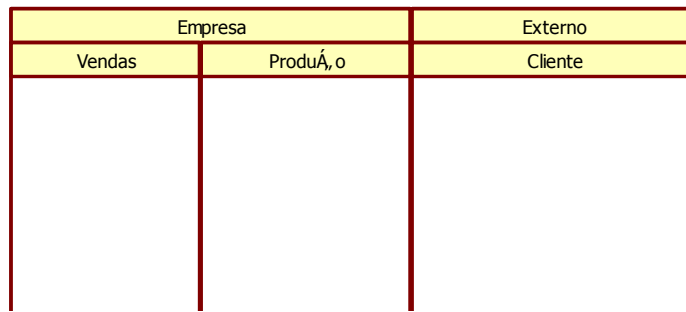


Figura 78. Exemplo de raia com partição hierárquica.



Figura 79. Exemplo de raia com partições múltiplas.

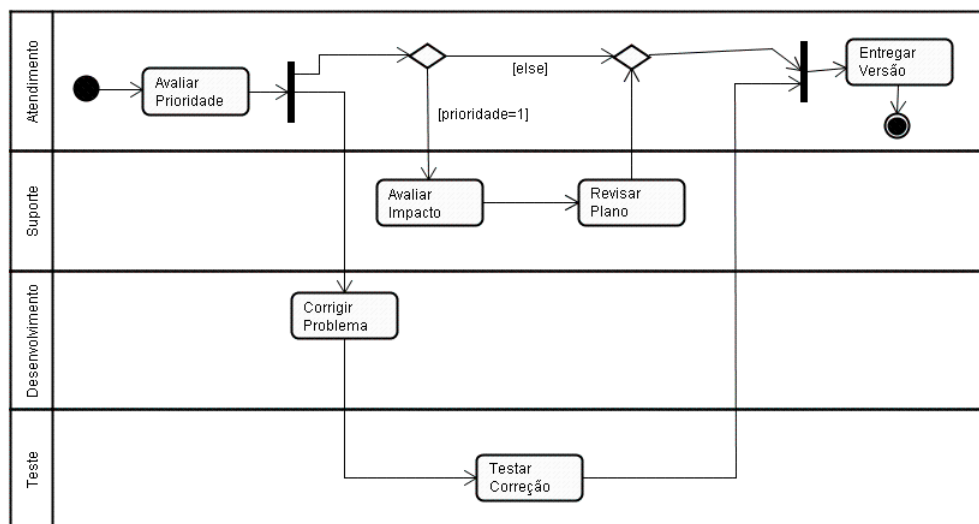


Figura 80. Exemplo de diagrama de atividades com raia

A seguir apresentamos dois diagramas interessantes se comparados. Ambos descrevem um mesmo comportamento final, porém com implementações diferentes. Deixamos ao leitor o desafio de ver como as seqüências de ações realizadas serão sempre as mesmas.

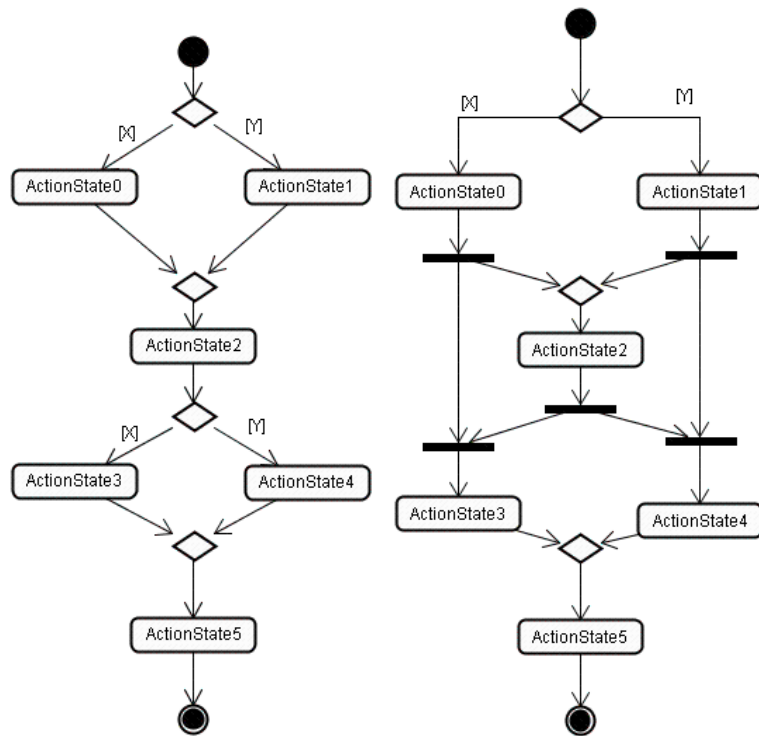


Figura 81 Os dois Diagramas de Atividade acima apresentarão a mesma seqüência de ações para uma mesma entrada, porém utilizando implementações diferentes.

I.9.6 Exemplo

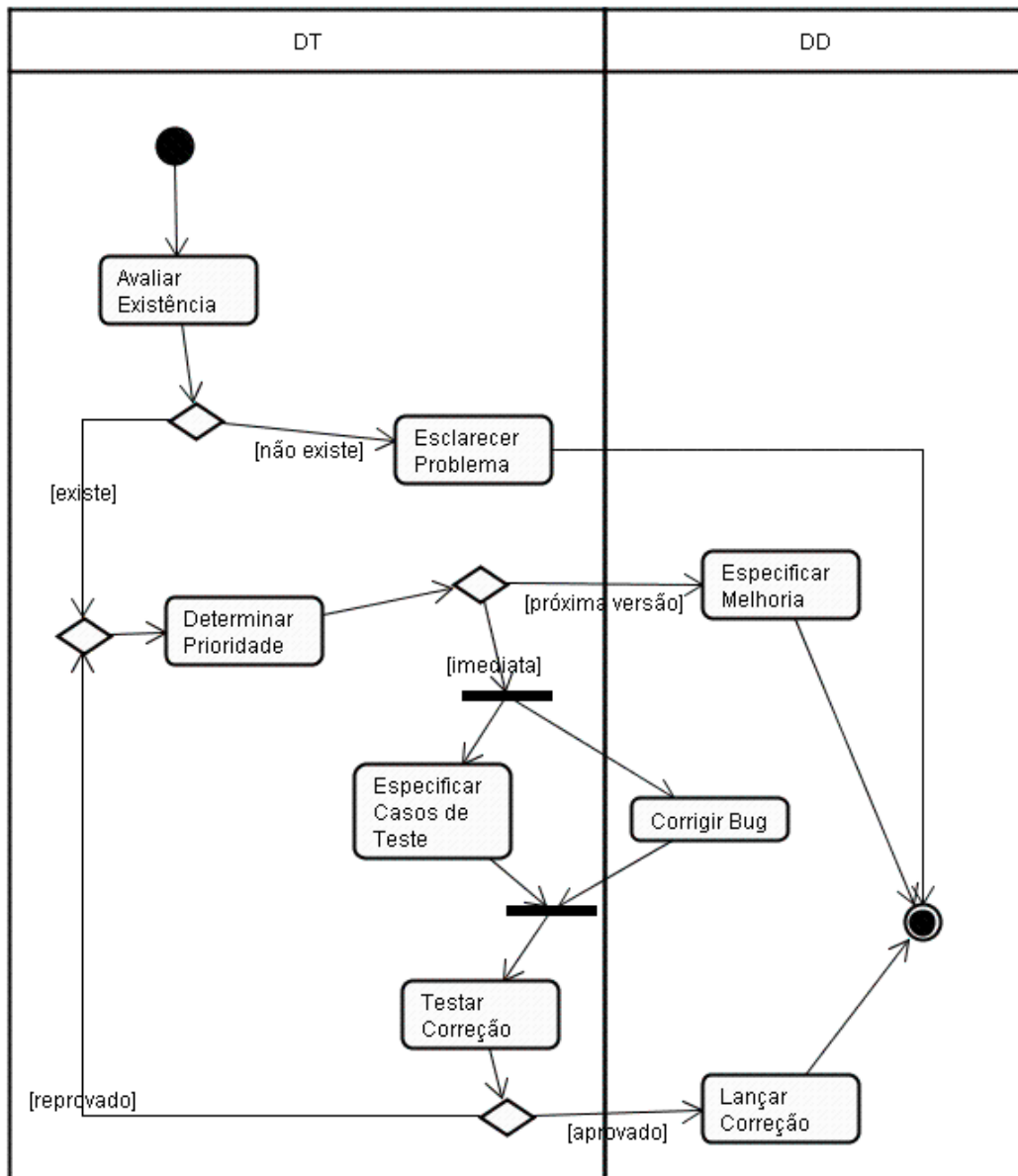


Figura 82. Diagrama de Atividades para o texto da Figura I.4.4

I.10 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”[B26].

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”²⁶, que produziu alguns documentos [B26;B27] 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.

- **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 relacionamentos²⁷.
- **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.
- **Derivações**, a declaração de um conhecimento que é derivado a partir de outro.

I.10.1 Declarações Estruturais

Inclui os termos de negócios e fatos relacionando termos.

I.10.1.1 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” [B26].

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.

²⁶ Originalmente “The Guide Business Rules Project”

²⁷ Não sendo necessariamente igual ao modelo conceitual de dados, mas podendo fornecer elementos para esse.

I.10.1.2 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.

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:

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

Tabela 4. Tipos de Fatos e exemplos

I.10.1.3 Declarações estruturais e o modelo de dados

Termos e fatos estabelecem a estrutura de um negócio. Esta estrutura é normalmente descrita em um modelo de dados. Assim, a maior parte do trabalho inicial com as regras de negócio estruturais pode ser descrita por meio de um modelo de dados conceitual, tema tratado no próximo capítulo.

I.10.1.4 Exemplos

A seguir damos alguns exemplos de declarações estruturais, para um sistema acadêmico imaginário:

- Alunos são pessoas matriculadas em um curso
- Um Aluno de Graduação é um tipo de Aluno
- Um Aluno de Pós-Graduação é um tipo de Aluno
- Um Curso é construído por um conjunto de Cadeiras
- Uma Cadeira é ministrada por um Professor
- Um Professor é um tipo de Funcionário
- Durante um Período, Alunos podem se matricular em Cadeiras

I.10.2 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).

I.10.3 Derivações

São regras que mostram como transformar conhecimento em uma forma para outro tipo de conhecimento, possivelmente em outra forma, incluindo leis da natureza[B27]. 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.

I.10.4 Regras e Eventos

Cada regra de negócio tem a possibilidade de ser violada em um ou mais eventos do sistema.

A questão do que é um evento será respondida mais a frente nesse texto, porém, no momento, basta entendermos que um evento é algo que exige que alteremos os fatos conhecidos pelo sistema (ou seja, um evento exige a alteração de algum dado na base de dados) ou que consultemos esses fatos a fim de informá-los, de alguma forma processada, ao usuário.

Analisando uma regra podemos ver que tipos de eventos são prováveis de necessitarem de alguma atenção do sistema. Por exemplo, suponha que um sistema de vendas para uma empresa possua as regras[B28]:

- Um aluno cliente solicita um produto
- Um vendedor é designado para atender um cliente permanentemente

A descrição acima faz com que nos perguntemos:

- O que acontece quando um cliente faz seu primeiro pedido(e não tem ainda um vendedor associado)?
- O que acontece quando um vendedor se desliga da empresa.

Em todo evento, um conjunto de regras deve ser ativado e cada erro deve ser reportado ao usuário.

I.10.5 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 5. Formas de descrever regras de negócio e suas características adaptado de [B29].

Halle[B29] 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
Termo	Nome ou sentença com uma definição acordada que pode ser: <ul style="list-style-type: none"> • Conceito, classe, entidade, • Propriedade, detalhe, atributo, • Valor, • Conjunto de valores 	<termo> é definido como <texto>
Fato	Sentença que relaciona termos em observações relevantes ao negócio <ul style="list-style-type: none"> • Relacionamentos entre entidades • Relacionamentos entre entidades e atributos • Relacionamentos de herança 	<termo1> é um <termo2> <termo1> <verbo> <termo2> <termo1> é composto de <termo2> <termo1> é um papel de <termo2> <termo1> tem a propriedade <termo2>
Computação	Sentença fornecendo um algoritmo para calcular o valor de um termo	<termo> é calculado como <formula>
Restrição obrigatória	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>
Guideline	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>
Conhecimento inferido	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.
Iniciador de ação	Sentença expressando condições que levam ao início de uma ação	se <termo1> <operador> <termo2, valor, ou lista de valores> ... então <ação>

Tabela 6. 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[B27].

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 [B30], 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 83**):

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

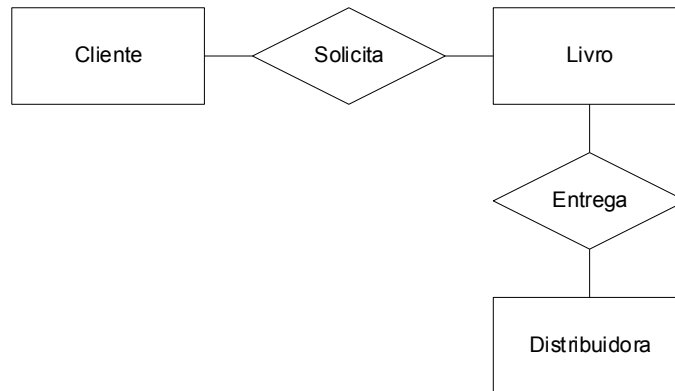


Figura 83. Exemplo de duas regras de negócio descritas utilizando uma notação de DER simplificada.

Alguns autores fazem uma descrição regra a regra, como na **Figura 84**.

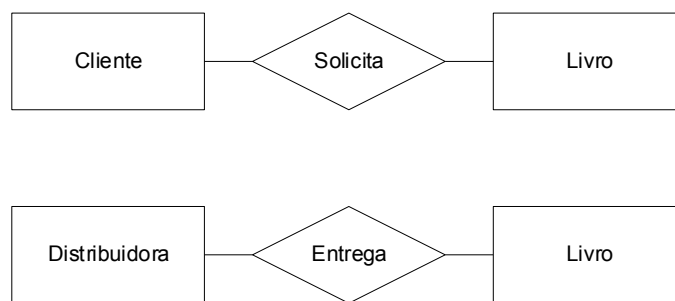


Figura 84. Descrição das regras uma a uma.

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 85** 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.

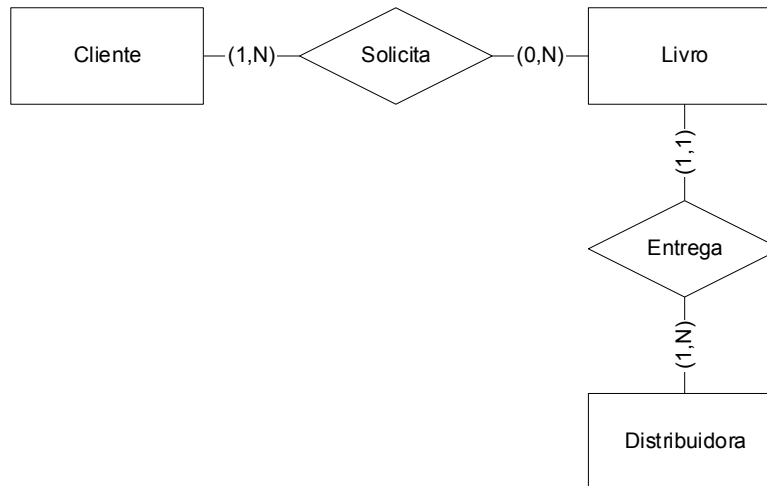


Figura 85. Regras com cardinalidades.

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.

I.11 Outras Ferramentas de Modelagem

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

I.11.1 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 7. 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 8 Exemplo de uma tabela Processo x Organização

I.11.2 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 Cria, lê (Read), altera (Update) ou apaga (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 tardes), 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 86. Exemplo simplificado de uma Matriz CRUD

I.11.3 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 9. Exemplo de tabela Corrente x Planejado

I.12 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 podem 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).

I.13 Exercícios

I.13.1 Projeto 1: Livraria ABC

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