Universidade Federal do Rio Grande do Norte

Departamento de Informática e Matemática Aplicada

Prof. Ricardo Kléber

Análise e Projeto Estruturados de Sistemas

Sumário

Introdução

O desenvolvimento de sistemas de computador é um processo que envolve diferentes etapas no intuito de se conceber, analisar, projetar, implementar e implantar um software. Desta forma, é possível partir da idéia (concepção) e evoluir até um produto concreto o qual deve oferecer aos usuários as funções necessárias para que ele possa realizar suas tarefas.

Porém, partir da idéia e chegar em um produto de software não é uma atividade trivial e para tanto existem propostas na literatura que buscam definir o processo de desenvolvimento de um sistema. Umas dessas propostas é a **Análise e Projeto Estruturados** que descreve uma série de técnicas que podem ser aplicadas no processo de desenvolvimento de software.

1.1 O Software

Uma definição de software apresentada por Lucena [LUC87] descreve **software** como um produto lógico, portanto abstrato, fruto da criação e interpretação do conhecimento humano.

Software é um conceito amplo que abrange:

- Programas de computador
- Entidade abstrata.
- Ferramentas (mecanismos) pelas quais:

  exploramos os recursos do hardware.

  executamos determinadas tarefas

  resolvemos problemas.

  interagimos com a máquina.

  tornamos o computador operacional.
- Instruções que executam uma função desejada.
- Estrutura de dados para manipular informação
- Documentos para desenvolver, operar e manter os programas.

O software pode ser caracterizado pelos seguintes aspectos:

- software não é um elemento físico; é um elemento lógico;
- software é uma abstração da realidade;
- custos estão concentrados no desenvolvimento e não na manufatura;
- o processo de desenvolvimento de software envolve uma equipe de desenvolvimento;
- não há peças de reserva. => manutenção, correção, aperfeiçoamento;
- o software exige exaustivos testes (os erro de projeto são mais difíceis de testar);
- existem diferentes abordagens para se chegar no produto final.

1.2. Ciclo de Vida do Software

Qualificar um produto é muito bom para que tenhamos certeza de que há seriedade e preocupação com a satisfação em tê-lo, mas, qualificar o processo de produção é mais importante para obter um produto melhor. Ambas as qualificações (da produção e do produto) são largamente utilizados na produção de muitos produtos inclusive no desenvolvimento de softwares. Hoje, temos normas da ISO 9003 que certificam o processo de produção de software bem como o software pronto. Tais normas exigem cada vez mais qualidade no gerenciamento do projeto e tais exigências são convertidas em benefícios para os usuários e desenvolvedores. Todo desenvolvimento de um software é caracterizado por fases que quando colocados seqüência obtem-se um Ciclo de Vida do Sistema e é este ciclo de vida que deve ter qualidade.

Conforme observações feitas durante os últimos 20 anos, em centenas de organizações que variam de tamanho de algumas até milhares de pessoas, com uma faixa de 1 até 1000 projetos simultaneamente a caminho. Como pode ser esperado, as menores firmas costumam ser relativamente informais: os projetos são iniciados como resultado de uma discussão verbal entre o usuário e o gerente do projeto, e o projeto prossegue da análise de sistemas até o projeto e implementação sem muita algazarra. Em grandes organizações, no entanto, as coisas são feitas em uma base muito mais formal. As várias comunicações entre os usuários gerência e equipe do projeto costumam ser documentados por escrito, e todos entendem que o projeto passará por diversas fases antes que seja terminado. Ainda assim existe grandes diferenças entre o modo com que dois gerentes de projetos na mesma organização conduzem seus respectivos projetos.

Normalmente fica a cargo do gerente de projeto determinar de que fases e atividades o seu projeto consistirá e como essa fases serão conduzidas.

Recentemente, no entanto, o método assumido para o desenvolvimento de sistemas começou a mudar. Mais e mais grandes e pequenas organizações estão adotando um único e uniforme ciclo de vida do projeto, ou simplesmente "o modo com que fazemos as coisas por aqui". Normalmente contido em um livro tão exuberante quanto o manual de padrões que se encontra (fechado) a mesa de cada programador, o ciclo de vida documentado do projeto fornece uma forma comum para que todos na organização passem a entender como pode ser desenvolvido um sistema de computador.

O método pode ser caseiro, ou alternativamente, a organização pode decidir comprar um pacote de gerenciamento de projeto e depois moldá-lo às necessidades da companhia. Deve ser aparente que, além de fornecer emprego para as pessoas que criam manuais de ciclo de vida de projetos, a metodologia de projeto é desejável. Qual, então, é a finalidade de se ter um ciclo de vida de projeto? Há três objetivos principais:

1. Definir as atividades a ser executadas em um projeto.

2. Introduzir a coerência entre muitos projetos na mesma organização

3. Fornece pontos de checagem para controle de gerência e pontos de checagem para a decisão "ir / não ir".

O primeiro objetivo é particularmente importante numa grande organização em que novas pessoas estão constantemente entretanto nas fileiras da gerência de projeto. O gerente de projeto inexperiente pode não examinar ou subestimar a significância de importantes fases do projeto se seguir apenas sua própria intuição. Na verdade, pode acontecer que os programadores e analistas de sistemas júnior não entendam onde e como os seus esforços se encaixam no projeto de um modo geral, a menos que tenham recebido uma descrição apropriada de todas as fases do projeto.

O segundo objetivo é importante em uma grande organização. Para os níveis mais altos de gerência, pode ser extremamente desconcertante tentar supervisionar 100 projetos diferentes, cada um deles sendo executado de uma forma diferente.

O terceiro objetivo de um ciclo de vida padrão do projeto refere-se à necessidade de gerência em controlar um projeto. Em projetos triviais, o único ponto de checagem é provavelmente o fim do projeto: ele foi terminado dentro do prazo e

com o orçamento previsto? Mas, para grande projetos, a gerência deve ter uma série de pontos de checagem intermediários, gerando oportunidade para determinar se o projeto está atrasado e se precisam ser obtidos recursos adicionais. Além disso, um usuário inteligente também pedirá pontos de checagem em diversos estágios no projeto, de modo que possa determinar se quer continuar prorrogando o prazo!

Apesar de tudo isto o ciclo de vida não irá fazer o projeto, mais ele ajudará a organizar as atividades, tornando mais provável que você atinja os problemas exatos na hora exata.

Não existe uma regra ou um ciclo de vida padrão para desenvolvimento de sistema. Você pode usar um ciclo pré-definido por um determinado autor, pode usar o ciclo de um autor e moldá-lo conforme seu trabalho ou ainda criar seu próprio ciclo. Porém, a escolha deve ser feita analisando dois fatores:

- Características do fornecedor (equipe de desenvolvimento)
- Habilidade - Analisar os conhecimentos técnicos da equipe.
- Visão - A equipe deve ter conhecimento do domínio do problema a ser informatizado.
- Recursos - Equipamentos e material humano
- Tempo - Tempo disponível para o desenvolvimento
- Características do cliente (usuário)
- Visão - Conhecimento de suas necessidades
- Tempo - Aceitável para a implantação
- Recurso financeiro - Ter disponibilidade financeira para o investimento em software e hardware
- Insegurança - Confiabilidade nos resultados a serem gerados pelo sistema

Mesmo adotando um modelo de desenvolvimento, se os itens acima não forem bem avaliados, a qualidade do produto final poderá não ser satisfatório. Tanto modelagem estrutural quanto na modelagem orientada existem muitos ciclos sendo os mais utilizados:

- Modelo em Cascata (*Waterfall*) (Ciclo de Vida Clássico)
- Modelo Incremental
- Modelo Interativo ou Espiral

1.2.1. Modelo em Cascara (Waterfall)

Este abordagem baseia-se no modelo cascata ou método linear de desenvolvimento (figura 2). Podem ser utilizados conceitos de Engenharia de Software, a qual prevê atividade de verificação (estamos fazendo o produto de forma correta?), validação (estamos fazendo o produto certo?) e de controle de qualidade.

O ciclo é representado pelas seguintes fases :

1. Viabilidade: definição preliminar do escopo do sistema, restrições e conceitos alternativos;
2. Análise: especificação funcional do sistema (Projeto Lógico);
3. Projeto: especificação completa da arquitetura de hardware e software, estruturas de controle, estruturas de dados do sistema, interfaces;
4. Implementação: codificação e teste individual dos programas;
5. Teste : teste dos componentes integrados do sistema;
6. Implantação : implantação de maneira gradativa, a fim de evitar insatisfação e possibilitando a correção do sistema. Implantação piloto / paralela e definitiva;
7. Operação e Manutenção : utilização do sistema e modificações decorrentes de erros,mudança de necessidades, etc.

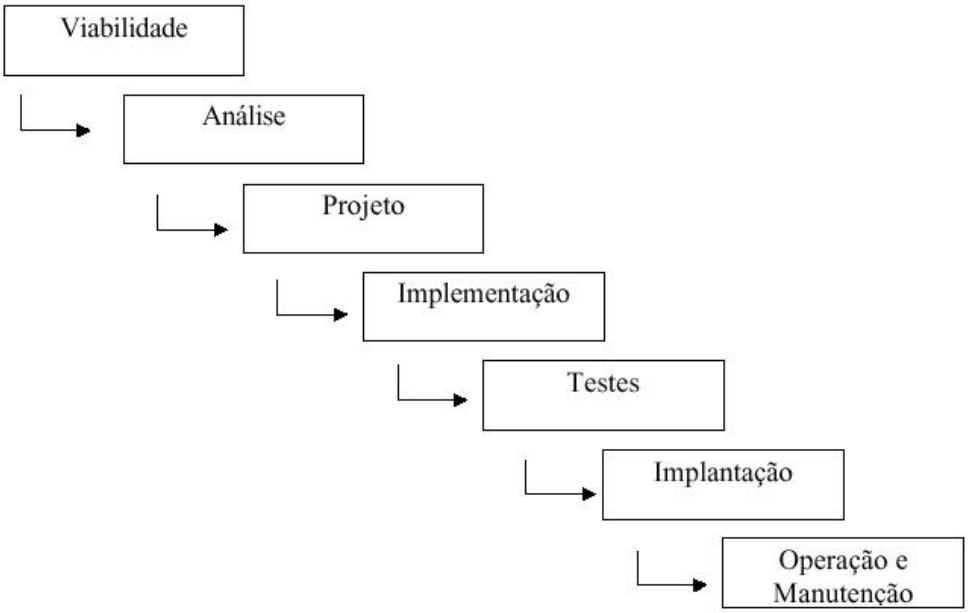


Figura 1 – Modelo em Cascata.

O modelo cascata é apropriado para sistemas transacionais onde as rotinas e procedimentos a serem automatizados são altamente estruturados.

A principal desvantagem desta abordagem é o alto custo de correção das especificações quando nas fases de Teste e Implantação.

1.2.2 Modelo Incremental

Modelo que divide o desenvolvimento do sistema em partes (módulos), cada uma das quais é desenvolvida seguindo as fases do modelo waterfall.

Tem como características liberar porções de código mais cedo, porém requer cuidadoso planejamento.

1.2.3 Modelo baseado em Prototipação

O modelo de prototipação se baseia na utilização de um protótipo do sistema real, para auxiliar na determinação de requisito. Um protótipo deve ser de baixo custo e de rápida obtenção, para que possa ser avaliado. Para isto, uma determinada parte do sistema é desenvolvida com o mínimo de investimento mas sem perder as características básicas, para ser analisada juntamente com o usuário.

A prototipação é um processo que habilita o desenvolvedor a criar um modelo do software que deve ser construído. O modelo pode tomar uma das 3 formas:

a) no papel ou num modelo baseado em PC em que a interação homem-máquina é desenhada para permitir ao usuário entender como tal interação irá ocorrer;

b) um protótipo trabalhando, que implementa algum subconjunto da função requerida no software desejado ou

c) um programa existente que executa parte ou toda a função desejada mas tem outras características que serão melhoradas no desenvolvimento do novo.

O protótipo pode servir como um "primeiro sistema". Alguns problemas surgem:
a) o cliente vê o que parece ser uma versão trabalhando do software, sem perceber que na pressa de oferecer algo trabalhando, não foram considerados aspectos de qualidade e de manutenção. Quando é informado de que o produto deve ser reconstruído, o cliente "implora" para não mude.

b) o desenvolvedor muitas vezes assume um compromisso de implementação para obter um protótipo trabalhando rapidamente. Um sistema operacional ou linguagem de programação inadequados podem ser usados simplesmente porque estão disponíveis e são conhecidos.

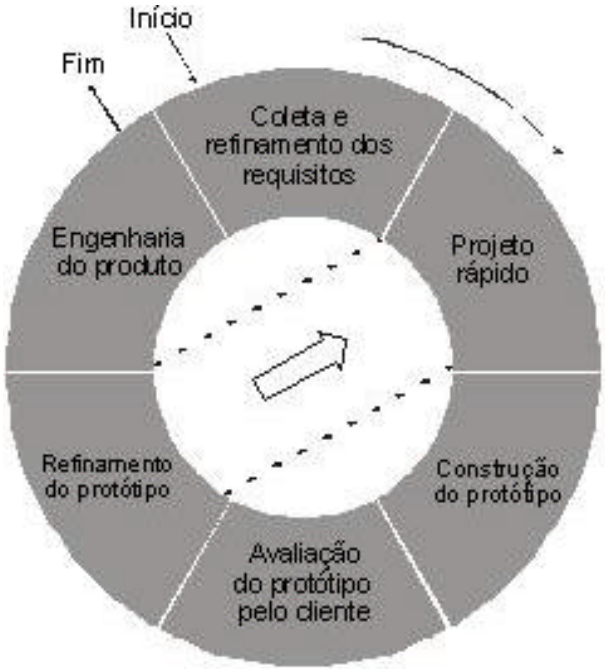


Figura 2 – Modelo baseado em Prototipação.

1.2.3 Modelo Interativo ou Espiral

O modelo espiral foi desenvolvido para abranger as melhores características tanto do modelo clássico como da prototipação, acrescentando ao mesmo tempo, um novo elemento - a análise dos riscos - que falta nos modelos anteriores. O modelo define quatro importantes atividades representadas pelos quatro quadrantes da figura 03:

Planejamento: determinação dos objetivos do sistema que será desenvolvido, restrições impostas à aplicação, tais como: desempenho, funcionalidade, capacidade de acomodar mudanças, meios alternativos de implementação;

Análise de risco: análise das alternativas e identificação / resolução dos riscos. Uma vez avaliados os riscos, pode-se construir protótipos para verificar se estes são realmente robustos para servir de base para a evolução futura do sistema;

Engenharia: desenvolvimento do produto do "próximo nível";

Avaliação do cliente: avaliação dos resultados de engenharia, pode-se efetuar a verificação e validação.

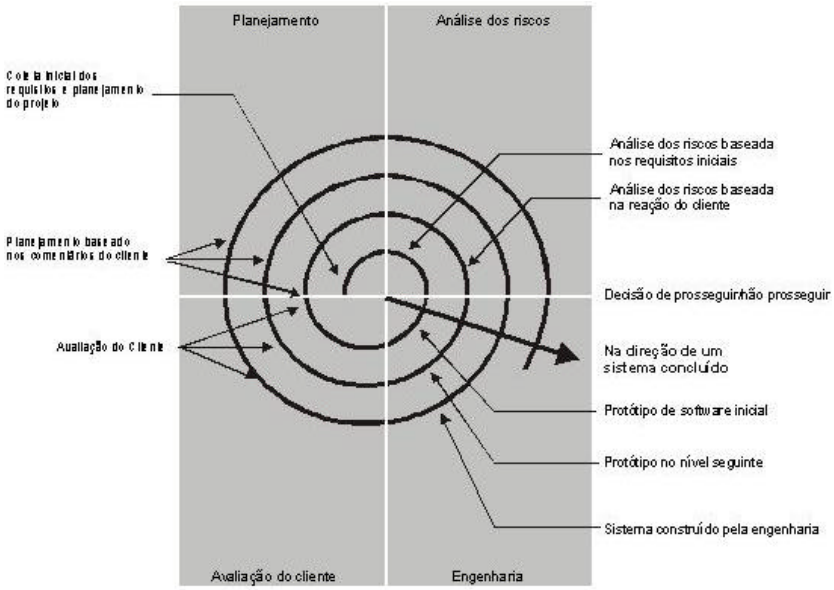


Figura 3 – Modelo Espiral.

Com cada iteração ao redor da espiral (começando no centro e trabalhando para fora), versões progressivamente mais completas do software são construídas. Durante o primeiro circuito ao redor da espiral, os objetivos, alternativas e restrições são definidas e riscos são identificados e analisados. Se a análise de risco indica que existem dúvidas nos requisitos, a prototipação pode ser usada no quadrante de engenharia para assistir tanto o desenvolvedor como o cliente. Simulações e outros modelos podem ser usados depois para definir o problema e refinar os requisitos.

O cliente avalia o trabalho de engenharia (quadrante de avaliação do cliente) e faz sugestões para modificações. Com base nas entradas do cliente, ocorre a próxima fase do planejamento e análise de risco. A cada loop ao redor da espiral, o ponto culminante da análise de risco resulta na decisão "continuar ou não". Se os riscos forem muito altos, o projeto pode ser encerrado.

No entanto, em muitos casos, o fluxo ao redor da espiral continua, com cada caminho movendo os desenvolvedores para fora em direção a um modelo mais completo do sistema e, ultimamente, o modelo operacional. Todo circuito ao redor da espiral requer engenharia (o quadrante inferior) que pode ser executado usando a abordagem de prototipação ou o modelo clássico de ciclo de vida.

O modelo espiral para engenharia de software pode ser considerado como o modelo mais realista para o desenvolvimento de sistemas grandes. Ele usa uma abordagem "evolucionária" para engenharia de software habilitando o desenvolvedor e o cliente a entender e a reagir aos riscos na evolução de cada nível. Usa o mecanismo de prototipação como um mecanismo de redução, mas mais importante, habilita o desenvolvedor a aplicar a prototipação em qualquer estágio na evolução do produto. Mantém também a abordagem por etapas do modelo

clássico, mas incorpora uma estrutura interativa que reflete de forma mais realista o mundo real. O modelo espiral demanda uma consideração direta dos riscos técnicos em todos os estágios do projeto, e se propriamente aplicado, reduziria os riscos antes que eles se tornem problemáticos.

1.3 Metodologias para Desenvolvimento de Sistemas

Hoje, nas empresas, as tecnologias de informação e as aplicações por elas geradas diferenciam produtos, sistemas e serviços, e proporcionam vantagens competitivas no mercado. Além disso, auxiliam na geração da ferramenta mais importante existente atualmente para as organizações - a informação.

Apesar da complexidade dos sistemas de informação, o seu desenvolvimento e manutenção dentro das organizações é uma tarefa realizada, na maioria das vezes, sem padrões, métodos ou técnicas bem definidas e sem práticas gerenciais de controle de qualidade e do acompanhamento dos projetos, gerando muitas vezes sistemas de informação que falham no atendimento aos requisitos dos usuários e consomem mais recursos (financeiros, humanos e computacionais) do que o esperado.

Para viabilizar o atendimento a estas necessidades em relação ao desenvolvimento e manutenção de sistemas de informação, surgem as metodologias de desenvolvimento de sistemas.

1.3.1. Definição de MDS

Uma metodologia de desenvolvimento de sistemas (MDS) é a integração de métodos, técnicas e ferramentas, informáticas ou organizacionais, que tem por objetivo planejar, orientar e acompanhar todo o processo de desenvolvimento e manuten ção de sistemas de informação dentro da organização.

Pode-se dizer que os objetivos básicos de uma metodologia são o de aumentar a produtividade e melhorar a qualidade dos projetos de sistemas de informação desenvolvidos na organização.

1.3.2. Vantagens de uma MDS

As principais vantagens de se utilizar uma MDS são:

– Padronização de técnicas, ferramentas e métodos para que toda a organização;

– Envolvimento de todos os participantes.

– Estabelecimento de padrões de qualidade e de planejamento e controle de projetos.
– Documentação incremental de todo o ciclo de vida do projeto.

1.3.3 Dificuldades do Processo de Implantação de uma MDS

A adoção efetiva de uma metodologia de desenvolvimento de sistemas abrange toda a organização, envolvendo uma grande diversidade de pessoas e culturas. Assim, é natural a existência dos seguintes obstáculos no decorrer do processo:

– Visão gerencial restrita: (associada à falta de planejamento)
– MDS como uma camisa de força
– MDS como simplesmente uma ferramenta documentacional:
– Resistência à mudança:

Leitura Adicional sobre MDS:

MASSA, Mônica de Souza. O USO DAS METODOLOGIAS DE DESENVOLVIMENTO DE SISTEMAS NA ORGANIZAÇÃO. Revista Estudos Acadêmicos. Novembro 1999.

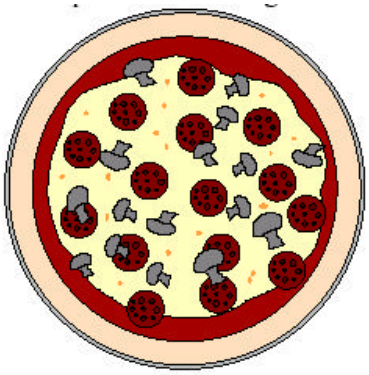
2. Análise Estruturada

A análise estruturada é uma atividade de construção de modelos que se preocupa com o que o sistema tem que fazer, definida no livro pioneiro sobre o assunto de Tom DeMarco.

2.1. Definição

A Análise Estruturada tem por objetivo a modelagem funcional dos sistemas através da especificação dos processos de transformação de dados.

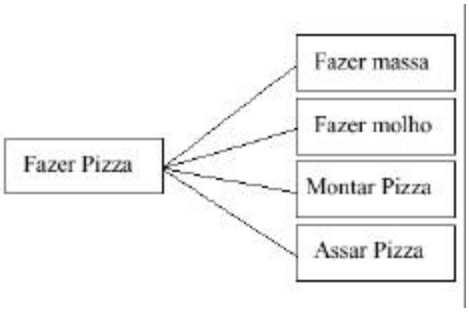
Exemplo: Pizza de Cogumelos



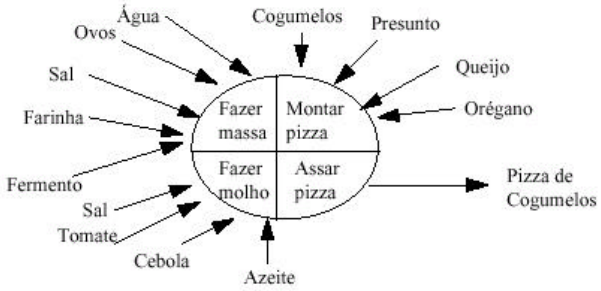
Toda receita tem um título referente à iguaria e deve-se conhecer todos os ingredientes antes de começar o processo de preparo. Assim, isola-se o processo de cozinhar do resto do ambiente.

Desta forma, temos um único processo, destacado do ambiente:

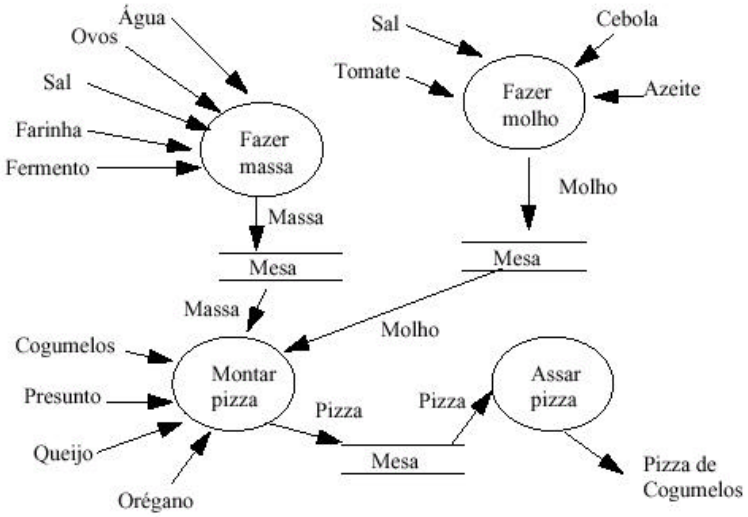
Quando a receita é grande, ela é dividida em “sub-receitas”. Antes era a “grande” caixa preta, agora são “caixas brancas”.



Todos os processos englobados em um único. Porém, deve-se mostrar a interação entre os processos.



Na visão da Análise Estrutura, estes processos ficariam definidos assim:



Então, com o uso da Análise Estruturada:

?? dota-se o desenvolvimento de software de uma linguagem comum;
?? utiliza-se ferramentas gráficas para visualização dos fluxos de informações e suas transformações.

A notação utilizada pela Análise Estruturada consiste em:

1. Diagrama de Fluxo de Dados (DFD), composto de:
   - Processos;
   - Fluxos de Dados;
   - Depósitos de Dados;
   - Entidades Externas.
2. Dicionário de Dados;
3. Ferramentas para descrever a lógica dos processos.

A Análise Estruturada realiza a decomposição funcional, através de uma abordagem *top-dow*n, por refinamentos sucessivos.

Na prática:

– para conhecer-se um processo é necessário isolá-lo do ambiente (caixa-preta);
– para dominar as regras de transformação de uma caixa preta complexa, deve-se decompô-la em caixas pretas mais simples (antes de abrí-la);
?? sempre que se executar a decomposição de um sistema, deve-se definir claramente as interfaces entre as partes.

2.2. Processos da Análise Estruturada

2.2.1. Levantamento de Informações

Quem é o "usuário"?

Entre as diversas atividades do analista, certamente estará incluída a necessidade de manter um diálogo constante com o usuário. É importante falar a língua do usuário e para tanto é imprescindível conhecê-lo.

Para conhecer o usuário é necessário saber exatamente com que tipo de usuário ou usuários, você está lidando. Este conhecimento refletirá no seu comportamento perante o usuário e permitirá uma preparação mais eficiente

dos levantamentos iniciais de informações sobre o sistema a ser desenvolvido ou modificado.

Os usuários podem ser classificados de três maneiras:
- pelas funções na organização;
- por nível de experiência no uso do computador e
- por características de personalidade.

Os usuários classificados pelas funções na organização, podem ser agrupados em:
- Operativos: normalmente têm visão local, são os executares das funções no sistema e tem visão física do sistema.
- Supervisores: pode ou não ter visão local, normalmente conhecem a operação e muitas vezes agem como intermediário entre os usuários e os níveis mais elevados da direção.
- Executivos: têm visão global, não tem muita experiência operativa e tem preocupações estratégicas.

Os usuários classificados por nível de experiência no uso do computador podem ser agrupados em:
- Amadores: extremamente leigos em computadores.
- Novatos: participaram (ou participam) de maneira mínima em algum trabalho ou até mesmo projeto maior que envolva o computador e consideram-se entendidos no assunto.
- Conhecedores: sabem dimensionar necessidades através da utilização de hardware e software.

Os usuários classificados por características de personalidade, podem ser divididos em:
- Conservadores: costumam não expor as idéias ou informações sobre a empresa de maneira clara, ou por desinteresse ou até mesmo por não concordar com as inovações pretendidas.
- Indiferentes: não aprovam ou reprovam a automação de atividades, mas muitas vezes prejudicam o levantamento de dados por falta de iniciativa ou motivação.
- Extrovertido: normalmente muito comunicativo, mas muitas vezes interfere no trabalho do analista, ou ainda dá uma atenção aos assuntos menos relevantes.

Uma vez caracterizado e conhecido que é o usuário é hora de iniciar a conversação com ele. Para tanto, existem várias técnicas que o analista pode dispor para efetuar um bom diálogo com os usuários, tais como:
- Entrevistas
- Observação Pessoal
- Questionários

Entrevistas

As entrevistas com usuários são uma técnica baseada no diálogo diretamente com os usuários reais do sistema a ser analisado. Esta técnica é ideal para auxiliar na primeira etapa do levantamento dos dados quando os usuários têm a liberdade de apresentarem toda a sua causa, todos os seus problemas, todas as suas necessidades, enfim, têm a liberdade de falarem o que bem entenderem, sobre as questões relativas ao seu trabalho e sobre como acreditam que um sistema de informação possa, eventualmente, auxiliá-los.

As entrevistas podem ser classificadas em duas fases ou momentos: um primeiro exploratório e um segundo analítico.

No primeiro momento, o entrevistador deverá ficar atento para as expressões utilizadas pelos usuários que:
- Possam transparecer que ele sente necessidade de informações:
" ... eu gostaria de gerar relatórios por fornecedores, mas não temos nenhum registro dos nossos fornecedores..."
- Caracterizam problemas que ele vem sentindo:
" ... para fechar o fluxo de caixa eu preciso calcular sempre o valor atual do ICMS..."
- Caracterizam objetivos de evolução da sua organização.
" ... hoje em dia só trabalhamos com regime de diárias mas pretendemos incorporar o regime de mensalistas em seis meses..."

Esses aspectos precisam ser anotados. O que caracteriza a possibilidade de um sistema de informações vir a ser útil é a presença de certas palavras, nas sentenças formuladas pelo usuário, tais como:

- verbo "saber", em expressões do tipo "eu nunca sei ..." ou "eu gostaria de saber ...".
  - sentenças como: "eles me perguntam ..." ou "o chefe me questiona sobre...". Essas
- sentenças são pistas de que alguma informação é necessária;
  - os verbos "planejar" e "controlar" em todas a suas variações possíveis, por exemplo,
- "nossa meta é ...", "os estoques estão fora de controle...":
  - sentenças como calcular, montar relatório, montar gráfico, registrar, consultar.

A fase seguinte (analítica) trata do levantamento dos dados, isto é da análise propriamente dita. Nessa fase o analista interpreta os resultados obtidos e estimula os entrevistados (os clientes) a fornecer detalhes sobre as respostas que eles esperam obter do sistema, bem como dos problemas para os quais esperam obter uma solução e suas causas. Um aspecto importante é que o entrevistador não deve solicitar detalhes sobre qualquer informação que estiver sendo transmitida nessa

fase do levantamento dos dados. Em hipótese nenhuma o entrevistado deverá ser estimulado, nessa fase, com recomendações de possíveis relatórios ou consultas que o analista presume que sejam importantes.

Outro fator importante é planejar a entrevista. Uma entrevista não pode ser apenas um diálogo (ou monólogo) conduzido pelo usuário mas deve ser um diálogo gerenciado pelo analista ainda que de forma transparente ao usuário.

Para você planejar uma entrevista observe as seguintes diretrizes:

1. · Planeje bem a entrevista para fazer uso eficiente do tempo;
2. · Use um estilo adequado de entrevistador;
3. · Procure conhecer bem a personalidade do entrevistado (Pessoa dinâmica, retraída, conservadora, etc). Facilitará em muito a sua postura durante as entrevistas;
4. · Evite emitir opiniões próprias;
5. · Evite entrevistar pessoas erradas em momentos errados;
6. · Evite fazer perguntas erradas e que ocasionam em respostas erradas;
7. · Procure distinguir fatos de opiniões e necessidades de desejos;
8. · Procure obter informações relevantes para as especificações do sistema;
9. · Confirme as informações obtidas.

Vale ressaltar que a técnica de entrevistas é utilizada em grupos relativamente pequenos de usuários geralmente ligados a gerência do processo (Usuários Executivos), isto é, os usuários que estão relacionados diretamente com as negociações do sistema.

Questionários

A técnica de questionários determina a elaboração de questionamentos que consistem em pequenas séries de perguntas para explorar a opinião do usuário sobre determinado assunto.

Podemos classificar estas perguntas como do tipo "*open-ende*d" ou "*closed*". As perguntas do tipo *open-ended* são aquelas que requisitam a participação do usuário, através do preenchimento de uma nota, por exemplo. Já questões onde o usuário tem um conjunto de possíveis respostas preestabelecidas para escolher (questões de múltipla escolha) são do tipo *close*d. Neste caso, o usuário pode ser especulado de maneira a expressar sua opinião concordando ou discordando de afirmativas propostas.

A figura 4-a traz um exemplo de questões tipo open-ended onde o usuário deve preencher o campo em branco registrando sua opinião. Este tipo de questão é muito utilizado para questionários onde o objetivo é explorar a opinião pessoal do usuário, como por exemplo:

- saber o que ele espera do novo sistema;
- saber quais as dificuldades do sistema (computacional ou não) antigo

- saber quais informações são necessárias

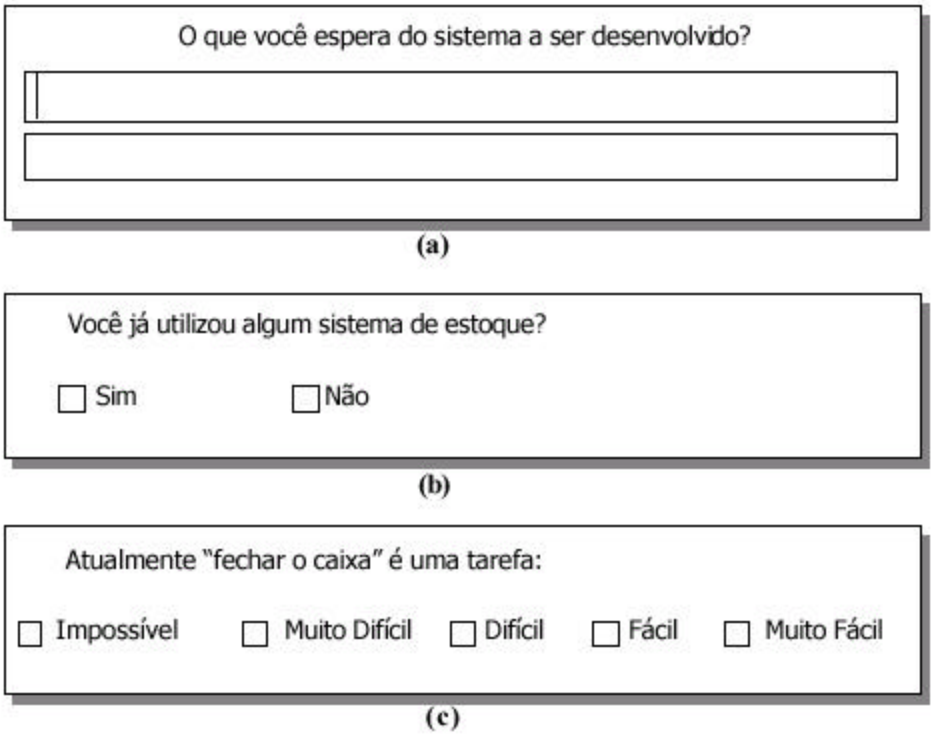


Figura 4 - Exemplos de Perguntas. (a) Tipo Open-ended. (b) Tipo Closed. (c) Tipo Closed.

Na Figura 4-b e Figura 4-c as questões ilustradas são do tipo closed. Este tipo de questão facilita a análise dos resultados, pois é facilmente mapeada para números, embora restrinja as possibilidades de resposta dos usuários. Como uma alternativa para enriquecer as opções de resposta, utilizam-se as escalas proporcionais.

As escalas proporcionais de avaliação são instrumentos que facilitam a conversão das respostas dos usuários para números, direcionando a opinião do usuário em intervalos escalares. Este tipo de recurso torna os resultados mais fiéis a realidade dos usuários, pois respostas mais elaboradas do que um simples "Sim ou Não".

Lea (1988) cita alguns tipos de escalas, entre os quais destacamos:

Escala Passo a Passo. Nesta escala o usuário escolhe uma opção dentre uma série delas, as opções são geralmente expressas por números ou adjetivos. (Figura 5)

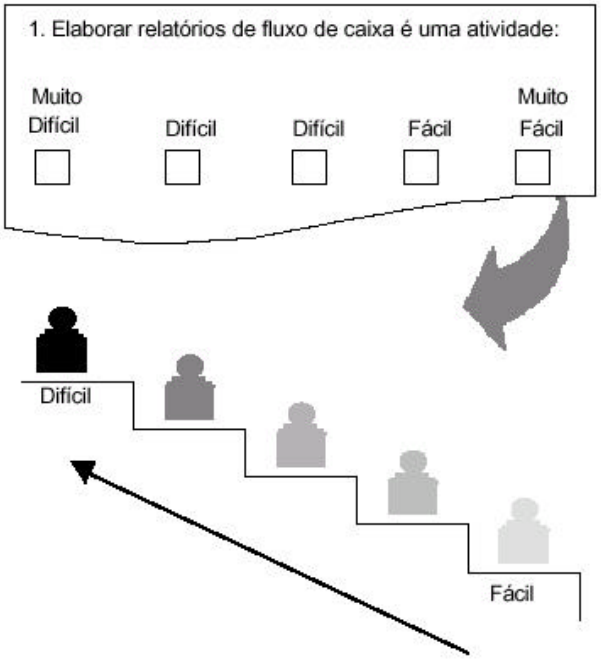


Figura 5- Escala Passo-a-Passo

Escala Comparativa. As escalas comparativas induzem o usuário a comparar elementos de uma solução de interface proposta com outros da mesma categoria, como pode ser observado na Figura 6.

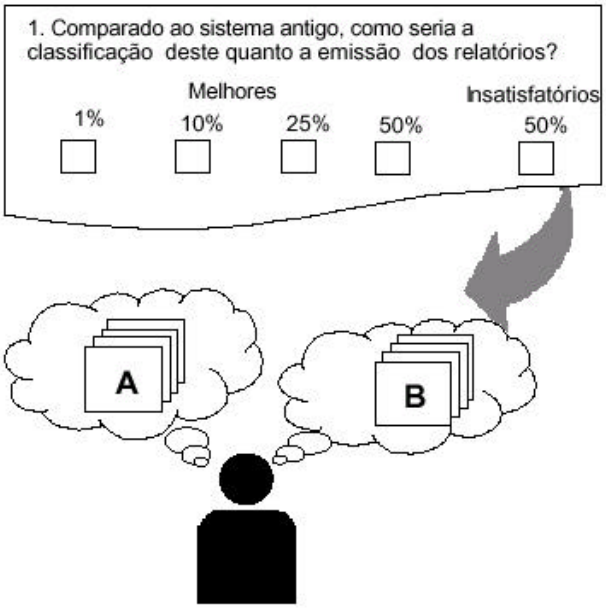


Figura 6- Escala Comparativa.

Escala de Likert. As escalas de Likert indicam o grau de sua concordância ou discordância dos usuários com as afirmações dadas, conforme o exemplo na Figura 7.

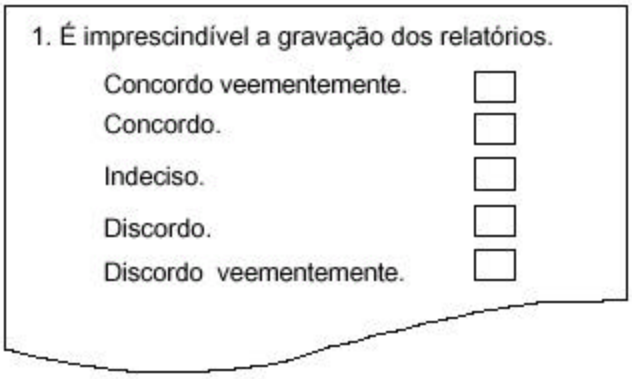


Figura 7 – Escala de Likert

Escala diferencial semântica. Este tipo de escala permite ao usuário assinalar sua opinião num subconjunto de alternativas entre pontos extremos de adjetivos opostos. A ilustração seguinte exemplifica esta escala.

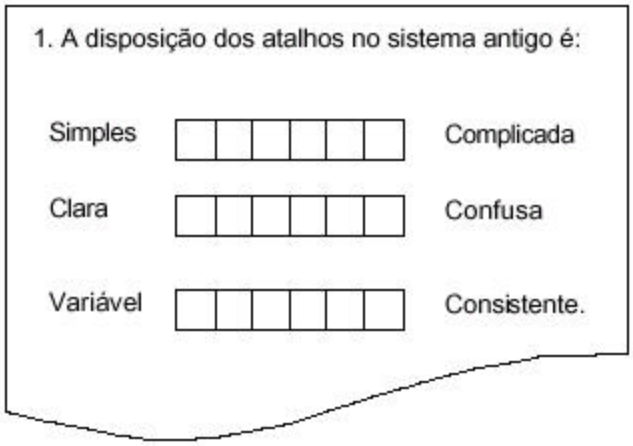


Figura 8 – Escala de diferencial semântica.

Então, a partir do uso de determinado tipo de escala ou de uma combinação de tipos pode-se através de simples questionamentos explorar informações importantes para o levantamento de dados.

A composição dos questionamentos é um ponto importante a ser considerado. Primeiramente, deve-se ter cuidado no planejamento das perguntas. As perguntas devem refletir os objetivos do questionário, isto é, estarem relacionadas com os elementos que estão em foco para avaliação. O segundo ponto é a forma adequada de abordar os usuários através do questionário, de modo que as perguntas formuladas não apresentem ambigüidades. Pois a precisão das perguntas é diretamente proporcional aos resultados.

Os resultados podem ser conseguidos através de uma análise estatística das respostas individuais, que serão unidas, com o propósito de compor uma resposta capaz de espelhar a opinião de todo um grupo de usuários, ou pode-se efetuar análises mais simples, as quais envolvam o cálculo de percentagens baseadas nos resultados obtidos.

Uma das grandes vantagens dos questionários é a amplitude de sua aplicabilidade. Essa técnica pode ser utilizada em grupos pequenos, médios e grandes de usuários que reflitam todas as camadas de usuários reais do sistema. Os questionários podem ser implementados como formulários em folha de papel, via e-mail ou formulário eletrônico.

Observação Pessoal

A idéia central da técnica de observação é a observação dos usuários em interação com sistema que está sendo analisado, isto é, executando ações a fim de cumprir com determinada tarefa do sistema, sendo esta computacional ou não.

Para observar o usuário devemos examinar cautelosamente os usuários em seu ambiente de trabalho e pesquisar os documentos e papéis utilizados na organização (relatórios, livros, arquivos, manuais, formulários, etc). Para observar o usuário é necessário analisar todas as suas manifestações, isto é suas ações, reações, necessidades e se possível for até seus pensamentos.

A Figura 7 ilustra a técnica de observação direta de usuários segundo Gomoll (1994) que afirma que a observação direta não é um experimento, portanto o objetivo não é traduzir ações do usuário em dados estatísticos, e sim em coletar dados úteis ao desenvolvimento do sistema.

A metodologia que define a observação direta na visão de Gomoll (1994) pode ser descrita através dos passos básicos apresentados a seguir:

1. Preparação. A observação de usuários requer uma etapa de preparação onde primeiramente será estabelecido o conjunto de tarefas dos usuários que deverão ser observadas. Essas tarefas devem ser as metas cruciais (ou dificultosas) do sistema. Por exemplo, fechamento do caixa, geração de relatórios ou protocolos internos da empresa. O passo seguinte é definir o grupo de usuários que será envolvido. Vale lembrar que muito do sucesso deste método está relacionado com este ponto. Portanto, deve-se ter cautela na escolha do grupo de usuários para que as necessidades dos usuários alvo do produto avaliado estejam representadas.
2. Apresentação. Os usuários devem conhecer o papel do analista e serem introduzidos ao processo do qual eles estão participando. O conhecimento do papel do analista é fundamental para eliminar a idéia de que o analista (ou o sistema a ser desenvolvido) vai substitui-lo na empresa. A ênfase aqui é não intimidar o usuário deixando claro que o alvo da observação é o entendimento do sistema (e da mecânica do processo) e não o desempenho do usuário na

suas funções diárias. Muitas vezes é difícil convencer o usuário que o foco do analista é apenas o de um observador do fluxo de trabalho e não do usuário.
3. Ruptura. Outro ponto importante é deixar o usuário a vontade para encerrar sua participação quando for conveniente.
4. Demonstração. Se o analista for utilizar algum equipamento extra para p registro da observação, como câmeras fotográficas, gravadores ou filmadoras é bom esclarecer ao usuário o uso da máquina fazendo uma pequena demonstração para não deixa-lo intimidado durante a observação.
5. Aplicar técnica de "Think Aloud". A técnica de Think Aloud (i.e. pensar em voz alta) deve ser aplicada durante a observação. Desta forma, os usuários devem ser instruídos a falar em voz alta aquilo que pensam durante a execução das tarefas.
6. Perguntas. Depois de todas as cartas na mesa é hora de dar início a observação. É prudente certificar-se se os usuários ainda apresentam alguma dúvida sobre o processo de observação ou sobre o seu papel, e, então iniciar o processo.
7. Conclusão. Após a observação deve-se agradecer a participação dos usuários e questioná-los se eventualmente ficou alguma dúvida. Aqui, as dúvidas são do observador, isto é, alguma manifestação do usuário que não tenha sido entendida pelo observador durante a execução de alguma tarefa, por exemplo.
8. Análise. A primeira tendência da análise dos dados coletados numa observação é atribuir as dificuldades encontradas aos próprios usuários, mas na maioria das vezes as falhas são da solução de design adotada. Portanto, a análise deve ser detalhada e centrar-se na identificação dos problemas e, então, verificar as causas e possíveis soluções dos mesmo.
Um importante fator é a documentação dos resultados da observação. A documentação pode acontecer de várias maneiras. A forma textual consiste em sintetizar os testes através de uma descrição detalhada do observador, que deve salientar os pontos detectados durante a observação. Os formulários, papéis, memorandos e manuais utilizados devem ser copiados e registrados pelo analista. A gravação em fita de vídeo ou cassete também constitui ferramenta de documentação. Deve-se sempre salientar os aspectos mais cruciais do processo. Etapas onde o usuário apresentou dificuldade na execução da tarefa ou que exigiu o cumprimento de vários passos devem ser destacadas.

2.2.2. Estudo de Viabilidade: Custos X Benefícios

O estudo de viabilidade de um sistema tem o objetivo de explicitar os pontos sensíveis (críticos) do projeto do sistema a ser desenvolvido. A documentação proveniente dessa fase deve ajudar tanto o gerente do projeto quanto o cliente responsável a decidir o futuro do projeto. Esta revisão deve resultar numa decisão de "prosseguir/não prosseguir".

O estudo de viabilidade de um sistema e a análise de riscos podem ser relacionadas de diversas maneiras. Para tanto, devemos avaliar quatro áreas de interesse fundamentais:

· viabilidade econômica;
· viabilidade técnica;
· viabilidade legal;
· alternativas de desenvolvimento do sistema.

Análise Econômica

A análise de custo-benefício delineia os custos para o desenvolvimento do projeto e compara-os com os benefícios tangíveis, isto é, diretamente mensuráveis em dinheiro, e os benefícios intangíveis de um sistema.
A maioria dos benefícios é intangível. Por exemplo: melhor qualidade das telas de entrada, satisfação do cliente por meio de controle programável, entre outros.

Possíveis benefícios do sistema de informação:

· Benefícios advindos de tarefas de calcular e imprimir.
· Benefícios das contribuições para as tarefas de manutenção de registros.
· Benefícios das contribuições para as tarefas de busca de registros.
· Benefícios das contribuições da capacidade de reestruturar o sistema.
· Benefícios das contribuições da capacidade de análise e simulação.
· Benefícios das contribuições para o controle de recursos e do processo.

Levando em conta os benefícios intangíveis, a decisão a ser tomada é se os resultados econômicos justificam o sistema. Possíveis custos de sistema de informação:

· Custos de aquisição: SO, plataforma, software especial.
· Custos de iniciação (start-up)
· Custos relacionados ao projeto.
· Custos permanentes.

Outro da análise custo-benefício leva em consideração o custo incremental aos benefícios adicionados, (melhor função e desempenho). Para um sistema computador, a relação custo-benfício incremental resume-se que:

– Em certas situações, o custo eleva-se proporcionalmente aos benefícios até ponto. Depois o benefício adicional é excessivamente caro.
– Em outras situações, o custo eleva-se proporcionalmente até certo ponto, e depois se aos benefícios adicionados, antes de se elevar drasticamente, para os subsequentes.

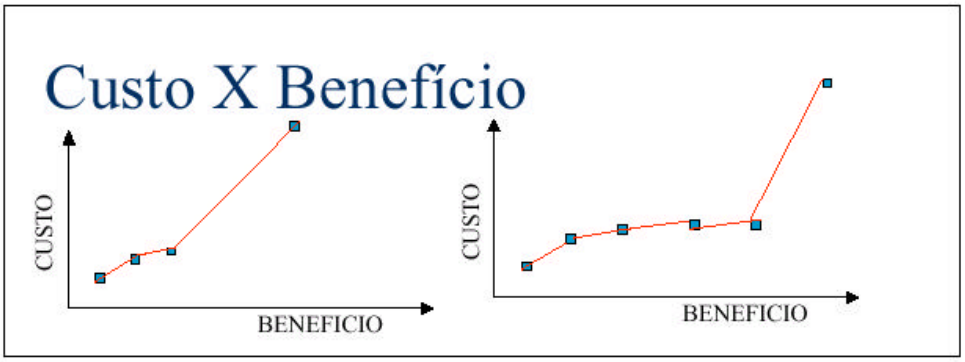


Figura 9 – Custo X Beneficio

Análise Técnica

Durante a análise técnica, o analista avalia os méritos técnicos da concepção do sistema, e, ao mesmo tempo, coleta informações adicionais sobre o desempenho, confiabilidade, manutenibilidade e capacidade de reutilização. Um modelo é criado, tendo como base observações do mundo real ou uma aproximação baseadas nas metas do sistemas.

O modelo deve incluir todos os fatores relevantes e ser confiável em termos de capacidade de repetir resultados. O projeto do modelo deve ser flexível a modificações e expansão.

Uma classe de CASE Tools, denominadas *Ferramentas de simulação e construção de protótipo*s, pode ser de grande ajuda na análise técnica.

Os resultados obtidos da análise técnica formam a base para outra decisão de "prosseguir / não prosseguir", com o sistema.

A viabilidade técnica é a área mais difícil de ser avaliada, uma vez que os objetivos, as funções e o desempenho são um tanto quanto vagos. Algumas considerações são: riscos de desenvolvimento, disponibilidade de recursos e tecnologia. A cautela deve prevalecer, na etapa de viabilidade técnica.

2.2.3. Notação Básica

As ferramentas utilizadas para modelagem e documentação segundo a abordagem estrutura são:

- Diagramas de Fluxo de Dados (DFD)
- Dicionário de Dados
- Diagramas E-R
- Diagramas de Transição de Estados

2.2.3.1. Diagrama de Fluxo de Dados

Um Diagrama de fluxo de dados é uma das mais usadas ferramentas (e há mais tempo) de modelagem funcional de sistemas. Um DFD pode ser entendido como uma rede que ilustra como circulam os dados no interior de um sistema.

Outros Sinônimos:
- DFD
- Diagramas de bolhas (ou bolas);
- Modelo de processos (ou de funções)
- Diagrama de fluxo de trabalho

Situação Exemplo:

Um cliente efetua uma reserva num hotel através de uma agência. Depois de verificada a disponibilidade do hotel escolhido é atribuído um quarto, calcula-se a conta, emitido um *voucher* ao cliente e informado o hotel da reserva.

Componentes de um DFD
Os DFD baseiam-se no princípio de que a funcionalidade de um sistema pode ser representada como uma rede que combina 4 componentes fundamentais:

Processos (ou bolhas ou bolas) - Transforma os fluxos de entrada em fluxos de saída. Cada processo tem um nome e um número único.

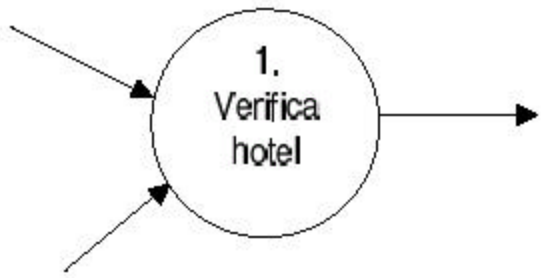


Figura 10 – Processo.

Depósitos de Dados -Repositórios ou depósitos para os dados existentes no sistema (e.g., arquivos, pastas de documentos, bases de dados).Cada arquivo tem um nome único.

Figura 11- Depósito de dados.

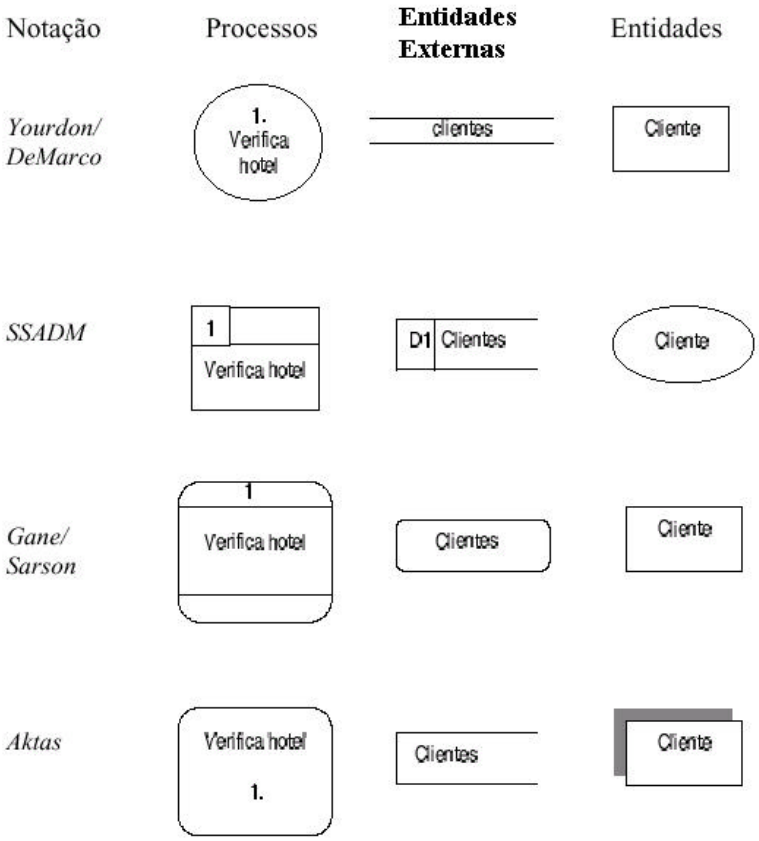
Entidades externas - Fornecem entradas ao sistema(fontes) ou recebem dados do sistema (terminadores). Existem fora da fronteira do sistema**.**

Figura 11- Entidades externas.

Fluxo de dados- Modelam a passagem de dados. A seta indica o sentido do fluxo e tem o nome do fluxo associado.

Figura 12- Fluxo de Dados.

Outras Notações

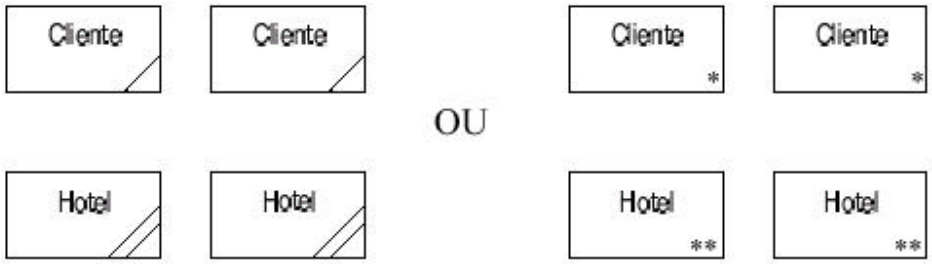


Convenções adicionais

O cruzamento de linhas de fluxos deve ser minimizado. Quando o cruzamento for inevitável, fazer o seguinte:

Figura 12 – Cruzamento de Linhas.

O mesmo arquivo poderá ser desenhado mais do que uma vez. A mesma entidade poderá ser desenhada mais do que uma vez. Exemplos:



Operadores lógicos:

* - conjunção (a e b);

? - ou exclusivo (a ou alternativamente b);

O - ou inclusivo (a ou b ou ambos)

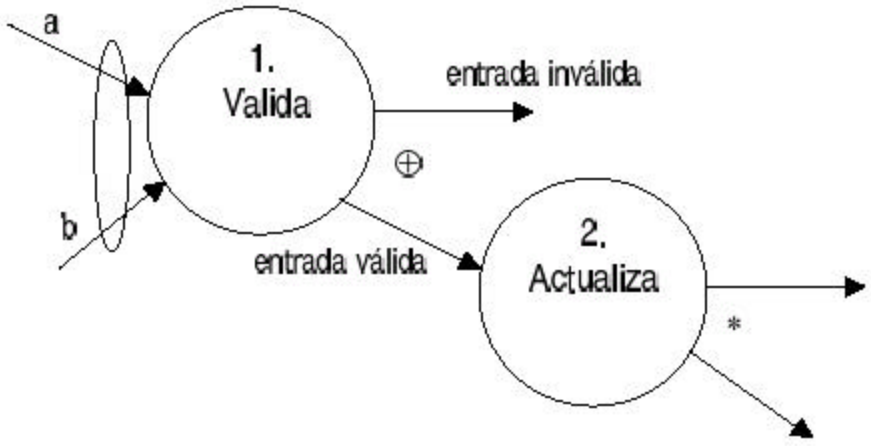


Figura 13 – exemplo de uso de operadores.

Decomposição ("Levelling")

O DFD de sistemas pequenos são normalmente fáceis de construir. Para sistemas não triviais, o número excessivo de processos leva a problemas de interpretação e desvaloriza os méritos do DFD.

Em sistemas de alguma complexidade, os DFD são organizados em níveis. Um DFD de um dado nível da hierarquia contém uma explicação funcional mais detalhada de um processo na hierarquia superior. A decomposição consiste na subdivisão de um DFD de alto nível numa hieraquia de DFD.

Convenções de decomposição

1. Cada processo num dado nível pode ser expandido (explodido) para se tornar num novo DFD;
2. Cada processo de um nível inferior está relacionado com o nível superior e é identificado por um número composto (e.g., 2.1.3);
3. Todos os fluxos de dados que entram e saem do nível superior devem aparecer no nível inferior (validação vertical);
4. Os depósitos de dados aparecem no nível em que são necessários e em todos os níveis subsequentes;
5. Em cada DFD, o limite superior recomendado para o número de processos recomendável é entre 2 e 7.
6. Processos que, pela sua simplicidade, não são expandidos são denominados processos primitivos ou primitivas funcionais;
7. Os processos primitivos são descritos usando um método de especificação de processos.

Regras e heurísticas de projeto

1. Estabelecer o contexto do DFD identificando todas as entidades externas do sistema. Sugestão: Escrever as fontes à esquerda e os terminadores à direita.
2. Identificar todas as saídas e entradas do sistema. Desenhar o diagrama de contexto.
3. Selecionar um ponto de partida para o projecto. Começar a desenhar os fluxos que são necessários para ir desde uma entrada até uma saída, partindo:
- da entrada para a saída;
- da saída para a entrada;
- ou partindo do centro.
4. Dar nomes significativos a todos os fluxos de dados e arquivos;
5. Rotular todos os processos com verbos operacionais que relacionem as entradas com os fluxos de saída;
6. Omitir os detalhes de programação como verificações de erros, inicializações e finalizações;
7. Não incluir controlo ou fluxo de controle;

8. Tentar planear o número de níveis. Não incluir demasiada informação num DFD;
9. Evitar cruzar fluxos de dados. Usar entidades e arquivos duplicados, se necessário;
10. Rever a consistência do DFD. Refazer, se necessário.

Validação vertical: os fluxos que entrem e saem de um processo, devem entrar / sair no DFD que resulta da explosão desse processo.

Validação horizontal: todo o que entra num processo é utilizado e tudo o que sai desse processo foi produzido.

11. Verificar, preferencialmente com o utilizador, se o DFD representa o sistema. Refazer, se necessário.

12. Depois de estabelecido um DFD explodir cada processo, se necessário. Repetir a decomposição até obter o detalhe suficiente.

Despiste de incorreções em DFD

- Processos com nomes ambíguos revelam falta de conhecimento sobre o sistema (ex. manipulação de entrada, gera saída);
- Fluxos de dados com nomes como "items de entrada", ou "vários dados" revelam um conhecimento pobre do sistema;
- Cruzamento de fluxos indicam que é potencialmente necessário decompor o DFD;
- Primitivas funcionais com grande número de entradas e saídas indicam a necessidade de decompor o DFD;
- DFD complexos significam fracasso.

2.2.3.2. Dicionário de Dados

O dicionário de dados pode ser visto como um depósito central que descreve e define o significado de toda a informação usada na construção de um sistema. Permite fazer a verificação de consistência entre os vários modelos.

Características dos Dicionários de Dados

Relativamente ao DFD, o contéudo de um dicionário de dados é compostos pela:
- especificação dos fluxos de dados;
- especificação de arquivos;
- especificação de processos;

O seu conteúdo deve ser preciso, conciso e não redundante;
Cada ocorrência deve contemplar, pelo menos, os seguintes aspectos:
1. **Nome**, ex., detalhes_de_quarto;
2. **Tipo**, ex., fluxo;
3. **Descrição**, conjunto de dados que caracterizam um quarto;
4. **Pseudônimos** (outros nomes possíveis)
5. **Especificação**, ex., detalhes_de_quarto = no + tipo + estado + extensão_telefonica + ([no_de_camas | lotação]);
6. **Comentários significativos**, que poderiam incluir tamanho, frequência, política de compartilhamento, segurança dos dados.

Especificação de dados - notação

Símbolo	Descrição	Alternativa
=	composição	
+	concatenação	
{ }	iteração: 0 ou mais ocorrências	
[ ]	selecção de 1 entre os presentes	
( )	opção	
" "	valor discreto	
@	chave de identificação	sublinhado
|	separador de alternativas	,
* *	comentário	

Exemplo:

Nome = tratamento + (primeiro_nome) + ultimo+nome
Tratamento = [Sr, Sra, Srta, Prof, Dr, Arq, Eng
Ultimo_nome=primeiro_nome={caracter}
Caracter=[A_Z | a-z]

Despiste de Inconsistências

Alterações nos modelos sem a manutenção adequada no Dicionário podem dar origem a:

Pseudônimos - Nomes diferentes para a mesma especificação de dados.

Impostores - Dados com o mesmo nome mas com especificações diferentes.

Órfãos - Dados que não pertencem a nenhum componente do sistema; desconhece-se a sua origem e o seu destino.

2.2.3.3. Modelo ER

O Modelo Conceitual de Dados é representado por um gráfico denominado DIAGRAMA ENTIDADES X RELACIONAMENTOS, proposto por P.P. Chen. Trata-se de um diagrama que detalha as associações existentes entre as entidades de Dados e utiliza componentes semânticos próprios.

A elaboração do Modelo Entidades x Relacionamentos (Modelo E x R) segue aos seguintes princípios:

1. cada Entidade é representada por um retângulo na posição horizontal;
2. cada tipo de relacionamento é representado por um losango;
3. pode haver um tipo de relacionamento entre mais do que duas Entidades;
4. os relacionamentos podem conter atributos;
5. pode haver mais de um tipo de relacionamento entre duas Entidades;

Para a compreensão do Modelo E x R é necessário considerar cada Entidade sob o enfoque de dados, não levando em conta os processos (procedimentos ou rotinas). Os relacionamentos se dão entre os dados de uma Entidade em relação aos dados das demais Entidades que formam o Modelo.

O princípio básico é o da Teoria de Conjuntos. Cada Entidade Conceitual deve ser vista como sendo um conjunto que pode ou não ter relacionamento (interseção) com outro conjunto.

Resumidamente, é a seguinte a terminologia básica do Modelo Entidade x Relacionamento:

ENTIDADE: são os componentes físicos e abstratos utilizados pela empresa, sobre os quais são armazenados dados;

ATRIBUTO: corresponde à representação de propriedades de uma Entidade. Um atributo não tem vida própria ou independente. Existe apenas para caracterizar uma Entidade;

OCORRÊNCIA: conjunto de atributos de uma determinada Entidade;

RELACIONAMENTO: é uma correspondência entre duas entidades. Uma associação entre dois conjuntos de dados (entidades);

IDENTIFICADOR ou ATRIBUTO DETERMINANTE: um atributo ou uma coleção de atributos que determina de modo único uma Ocorrência de Entidade;

GRAU DE RELACIONAMENTO: o número de entidades que participam da associação;

CLASSE DE RELACIONAMENTO ou CARDINALIDADE: quantas ocorrências de cada entidade estão envolvidas no relacionamento. Pode ser:

- 1:1 (um para um)
- 1:n (um para muitos)
- m:n (muitos para muitos)

A descoberta dos relacionamentos existentes entre as ENTIDADES é realizada pelo analista e usuário, obedecendo-se as seguintes etapas:

1. Individualizar uma ocorrência de uma entidade que integra o modelo conceitual;
2. Questionar se há algum relacionamento entre a ocorrência da entidade de origem com outra entidade do modelo (entidade de destino);
3. No caso de existir o relacionamento, batizá-lo com um nome representativo da associação;
4. Assinalar a classe de relacionamento entre a entidade origem e a entidade destino;
5. Questionar qual a classe de relacionamento entre a entidade destino e sua entidade de origem;
6. Assinalar a classe de relacionamento entre a entidade destino e a sua entidade de origem;
7. Realizar as etapas de 1 a 6 até que todos os relacionamentos do modelo sejam analisados, classificados os seus graus e assinalados no modelo.

2.2.3.4. Diagrama de Transição de Estados

Considerando que um sistema on-line interage diretamente com pessoas, é importante que o analista de sistemas ou o engenheiro de software projete com cuidado a interface com o usuário. Para isso, é necessário modelar todas as mensagens possíveis que o usuário poderá teclar no terminal e todas as respostas que o sistema pode fornecer - e todas as respostas do ser humano às consultas do computador, e assim por diante. Esta atividade é geralmente executada

identificando-se todos os estados em que o computador e o usuário podem se encontrar, bem como todas as modificações de estado. Um exemplo de um estado para o computador de um sistema bancário de "caixa eletrônico" seria: "O usuário colocou o cartão de crédito e se identificou, mas ainda não me informou sua senha particular". Uma possível mudança de estado poderia ser: "Ele me revelou sua senha e agora posso tentar saber se ele quer retirar dinheiro ou ver o saldo da conta". Outra mudança de estado seria: "Ele tentou digitar a senha três vezes sem sucesso e agora vou soar o alarme". Esses estados e alterações de estado são normalmente modelados nos **diagramas de transições de estado**.

Os principais componentes do diagrama são os **estados** e as **alterações de estados**. Os estados são representados como retângulos com cantos arredondados e as transições de estado são simbolizadas por setas.

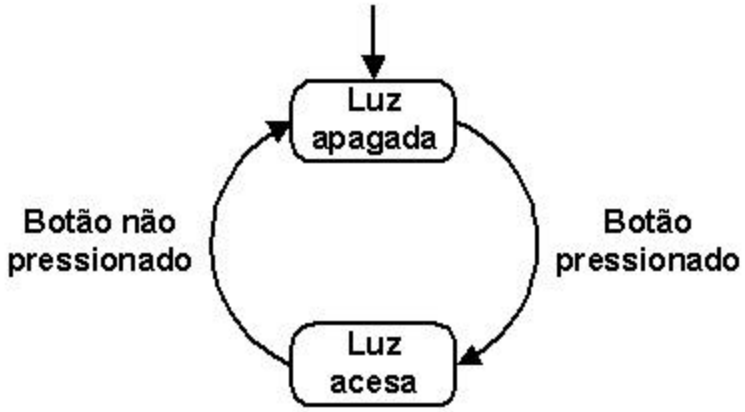


Figura 14 - DTE para controle de luz em relógio de pulso.

A Figura 15 ilustra um DTE que mostra um comportamento típico encontrado em secretárias eletrônicas. Neste DTE, as transições não estão rotuladas. Toda transição é causada por um evento, que geralmente está presente no rótulo da transição. Na Figura 14, existem dois eventos: "Botão pressionado" e "Botão não pressionado". Uma transição também pode ser associada a uma ação, descrita no rótulo da transição após o caracter "/", gerando um evento interno ao sistema. Este evento só poderá ocorrer como conseqüência de algum outro evento. As ações são instantâneas, não consumindo tempo.

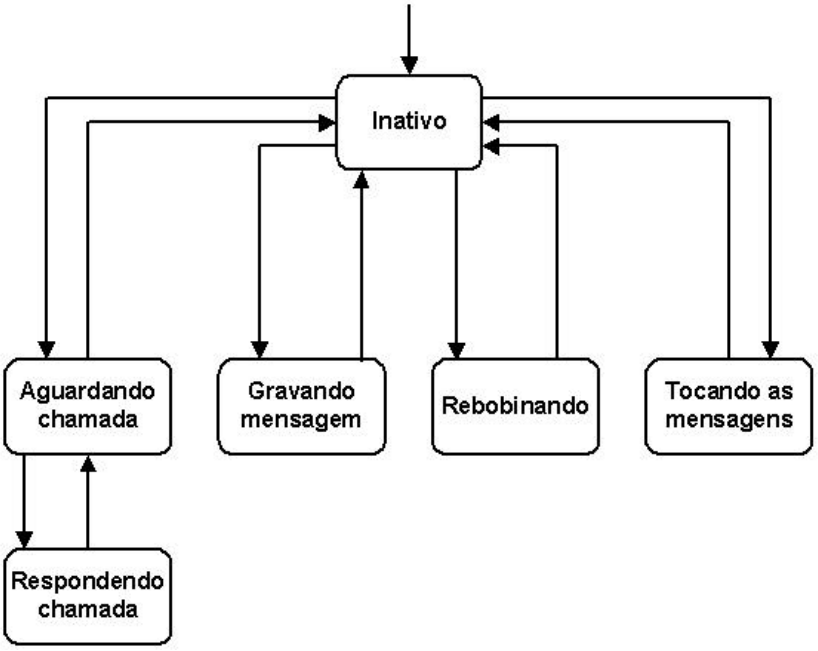


Figura 15 - DTE para secretária eletrônica.

O dicionário Webster define **estado** como um "conjunto de circunstâncias ou atributos que caracterizam uma pessoa ou objeto em determinado momento; modo ou forma de ser; condição". Assim, os estados típicos de um sistema podem ser:

- esperando que o usuário digite uma senha,
- aquecendo uma mistura química,
- aguardando o próximo comando,
- acelerando o motor,
- combinando ingredientes,
- aguardando dados para instrumentos,
- enchendo o tanque de combustível de um carro, ou
- ocioso.

Muitos destes exemplos demonstram que o sistema está aguardando pela ocorrência de alguma coisa e não indicam que o computador esteja realizando algo, porque um DTE é usado para desenvolver um modelo essencial do sistema – um modelo de como o sistema agiria se uma tecnologia perfeita estivesse disponível e com custo reduzido. Uma característica da tecnologia perfeita seria o computador funcionar com rapidez infinita, de forma que qualquer processamento ou cálculo que o sistema precisasse executar ou qualquer ação que ele precisasse realizar ocorresse em tempo zero. Assim, qualquer estado observável em que o sistema esteja pode corresponder apenas a períodos de tempo em que ele esteja:

- esperando algum fato acontecer no ambiente externo, ou

– esperando que a atividade atual do ambiente (misturando, lavando, enchendo, acelerando, etc.) mude para alguma outra atividade.

Isto não significa que os sistemas não conseguem executar uma ação ou que não se pretende exibir tais ações, mas sim que essas ações (instantâneas, neste modelo de tecnologia perfeita) não são equivalentes a estados, os quais simbolizam as situações observáveis nas quais o sistema pode estar. Assim, um estado representa um comportamento observável do sistema que ocorre durante um período finito de tempo.

Um sistema que existisse unicamente em um estado não seria tão interessante para ser estudado: ele seria estático e não faria sentido ser modelado por um diagrama de transições de estado. Na verdade, os sistemas geralmente modelados podem conter muitos diferentes estados. Porém, como um sistema passa de um estado a outro?

Se o sistema contar com normas regulamentares conduzindo seu comportamento, então apenas determinados tipos de transições de estado serão válidas e significativas.

As modificações válidas de estado no DTE são exibidas interligando-se por uma seta os pares de estados relevantes. Deste modo, a Figura 16, que ilustra parcialmente o ciclo de vida de uma mercadoria, mostra que uma mercadoria pode mudar do estado "Danificada" para o estado "Em estoque"; além disso, quando a mercadoria estiver em estoque pode passar ao estado "Vendida" ou voltar a se danificar. Portanto, o estado "Em estoque" apresenta dois estados sucessores, o que é comum nos DTEs porque qualquer estado pode levar a uma quantidade arbitrária de estados sucessores. Contudo, segundo este DTE, a mercadoria não pode passar diretamente do estado "Danificada" ao estado "Vendida". Por outro lado, o diagrama ainda revela que a mercadoria pode voltar diretamente do estado "Vendida" para o estado "Em estoque".

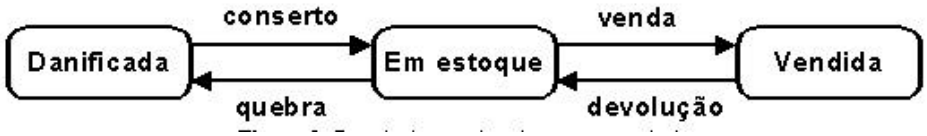


Figura 16 - Possíveis estados de uma mercadoria.

Embora a Figura 16 transmita algumas informações interessantes sobre o comportamento dependente do tempo de um sistema, ela não descreve algo que pode vir a ser muito importante: os estados **inicial** e **final** do sistema. Na verdade, a Figura 1 é um modelo estático de um sistema não computacional que tem permanecido ativo desde sempre e que prosseguirá ativo para sempre. A maioria dos sistemas apresenta um reconhecível estado inicial e outro final,

como ilustrado na Figura 15, que representa outro ciclo de vida possível para uma mercadoria.

Figura 17: Estados inicial e final .

O estado inicial normalmente é desenhado na parte superior do diagrama, embora não obrigatoriamente; o que identifica de fato o estado "Em estoque" na Figura 2 como estado inicial é a seta que não tem um estado origem. Da mesma forma, freqüentemente o estado final é incluído na parte inferior do DTE, mas isto também não é necessário. É a ausência de setas com origem nele que realmente identifica o estado "Quebrada" como estado final.

Conforme o senso comum, um sistema pode possuir apenas um estado inicial; contudo, ele pode ter vários estados finais, os quais são mutuamente exclusivos; portanto, apenas um destes estados finais pode acontecer na execução do sistema. A Figura 18 mostra um exemplo em que os possíveis estados finais são os estados 4 e 6.

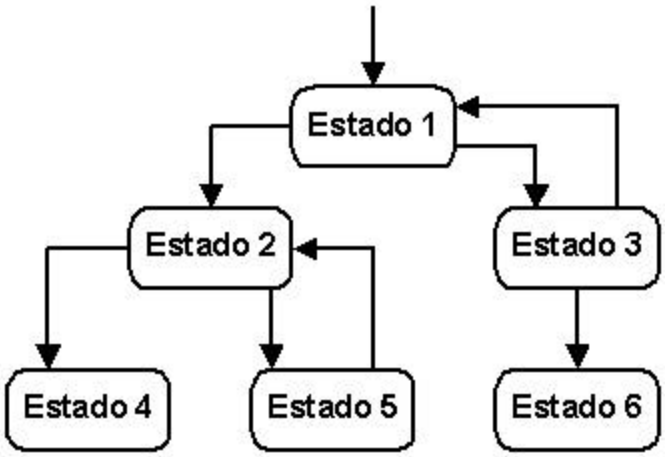


Figura 18: Dois estados finais para um sistema

Como os DTEs estão sendo utilizados para construir um modelo essencial, consideraremos que o sistema não necessita de tempo observável para mudar de um estado a outro. Quando os projetistas e programadores começarem a elaborar o **modelo de implementação**, isto será um problema real, porque normalmente um computador gasta alguns microssegundos para passar de uma atividade de processamento a outra e estes profissionais devem garantir que isto aconteça com rapidez suficiente para que o ambiente não fique fora de controle.

Para que o DTE fique completo, é necessário acrescentar os **eventos** que provocam uma mudança de estado e as **ações** que o sistema executa quando muda de estado. A Figura 4, que representa dois estados possíveis para um aparelho telefônico ("Ocioso" e "Ocupado"), também mostra dois eventos ("no gancho" e "fora do gancho") e uma ação ("reproduza tom de discagem").

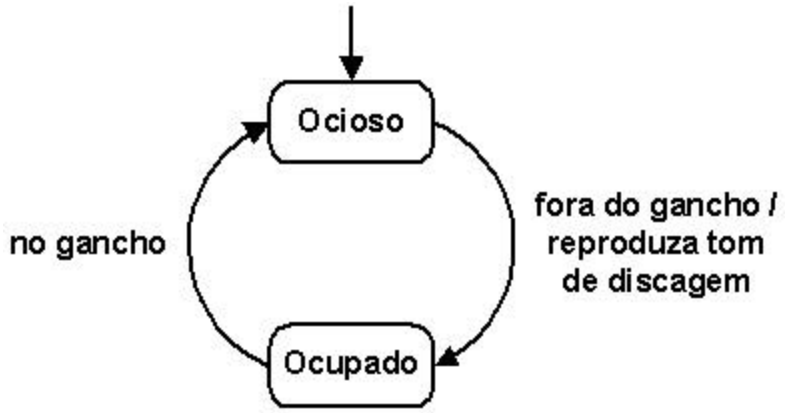


Figura 19: Indicação dos eventos e ações

Um evento é um acontecimento no ambiente externo que o sistema consegue detectar, podendo ser um sinal, uma interrupção ou a chegada de dados. Isto em geral fará o sistema passar do estado "aguardando X" para o estado "aguardando Y" ou do estado "executando a atividade X" para "executando a atividade Y".

São possíveis os seguintes tipos de eventos

Evento externo
: também conhecido como um evento do sistema, é provocado por algo ou alguém externo aos limites do sistema. Eventos externos observáveis precipitam a invocação de operações do sistema para responder a eles. Um evento externo ocorre, por exemplo, quando um caixa aperta o botão "registre item" em um ponto eletrônico de vendas.

Evento interno
: causado por algo localizado nas fronteiras do sistema. Em termos de *software*, um evento interno surge quando uma operação é ativada por meio de uma mensagem, ou sinal, enviada por outro objeto interno como, por exemplo, quando um objeto do tipo "Venda" recebe uma mensagem "produzaLinha DoPedido".

Evento temporal
: gerado pela ocorrência de uma data e hora específicas, ou pela passagem do tempo. Em termos de *software*, um evento temporal é dirigido por um relógio de tempo-real ou simulado. Um caso de evento temporal pode ser verbalmente especificado da seguinte forma: se, após finalizar uma compra em um ponto eletrônico de vendas, o pagamento

não ocorrer em cinco minutos, a compra atual é automaticamente descartada.

Contudo, como parte da mudança de estado, o sistema geralmente executará uma ou mais ações, produzirá uma saída, exibirá uma mensagem na tela, realizará um cálculo etc. Portanto, as ações apresentadas no DTE podem ser:

- respostas dirigidas ao ambiente externo,
- um outro evento, ou
- cálculos cujos resultados são armazenados no sistema para reagir a algum evento futuro.

3. Projeto Estruturado

Pode-se definir a palavra projeto como:

- Projeto (*design*) compreende atividades de concepção, desenvolvimento e especificação de um produto.
- Projeto (*project*) é um conjunto de tarefas.
- Projetos (*project*) são alternativas de investimento.

Projeto Estruturado é a abordagem ao processo de projeto, uma abordagem que resulta em módulos de caixa-preta, pequenos e independentes, dispostos em uma hierarquia, que é um modelo conceitual da área de trabalho em si, organizado em uma forma top-down (de cima para baixo), com os detalhes isolados na parte de baixo.

O projeto é integrante fundamental do processo de desenvolvimento de software. Numa abordagem estruturada o projeto inicia-se após a Engenharia de Requisitos utilizando uma Especificação de requisitos como parâmetro de entrada e fornecendo a etapa seguinte, codificação, uma especificação de projeto. Enquanto a especificação de requisitos é voltados a visão do cliente a especificação de projeto é voltada a interpretação do projetista dos componentes que irão conduzir a implementação do sistema.

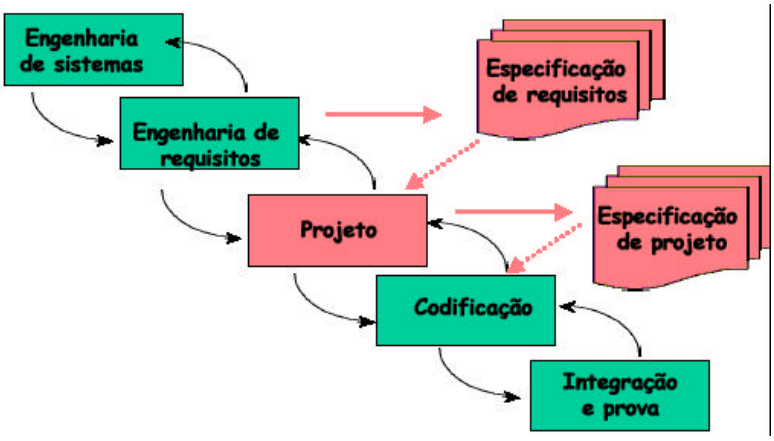


Figura 20: Projeto no processo de desenvolvimento de software.

O projeto de software transforma uma especificação do problema em uma solução computacionalmente viável. A Figura 21 ilustra a definição em um problema (p1, p2, p3, p4 e p5) em soluções computacionais (s1, s2, s3, s4 e s5). A adoção de uma solução computacional modularizada fornece uma visão da arquitetura de software do sistema.

A modularização é o atributo de software que permite abordar a complexidade do software. Através da modularização é mais fácil resolver um problema complexo, pois é possível dividi-lo em partes mais simples.

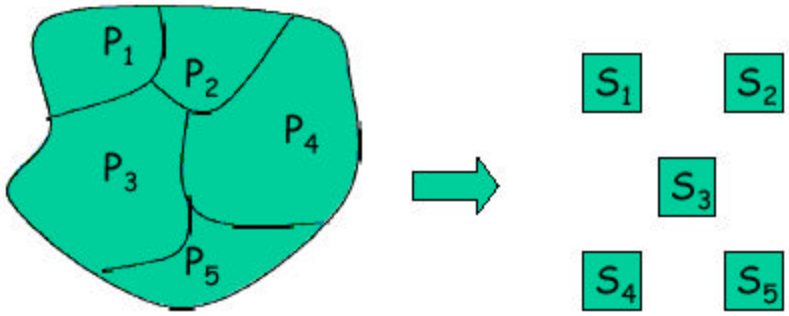


Figura 20: Projeto de Software.

3.1. Objetivos

O "Projeto" significa estabelecer a forma ou o método de uma solução. É o processo que caracteriza as principais características do sistema final, que estabelece os limites superiores em performance e qualidade que a melhor implementação pode alcançar e que pode até determinar qual será o custo final.

O projeto é caracterizado por um grande numero de decisões técnicas individuais.Se vamos progredir em relação ao desenvolvimento de programação para transforma-la em uma disciplina de engenharia, então precisamos sistematizar progressivamente estas decisões, torna-las mais explicitas e técnicas.

A engenharia é caracterizada pelo não contentamento do engenheiro pela obtenção de apenas uma solução. A engenharia busca a melhor solução em termos estabelecidos, dentro de limitações reconhecidas, respeitando ao mesmo tempo os compromissos requeridos pelo trabalho em um mundo real.

Para transformar o projeto de sistemas em uma disciplina de engenharia devemos primeiro definir objetivos técnicos claros para programas de computador como "sistemas". Os objetivos de sistemas globais determinam em ultima instância o critério pelo qual as decisões individuais de projeto serão feitas. Uma compreensão de exigências principais limitando as soluções admissíveis também é essencial. Para tomarmos decisões deliberadas e conscientes, precisamos saber também que existem pontos de decisão, a fim de que possamos reconhece-los quando os encontramos. Finalmente, precisamos de alguma metodologia ou técnica que nos auxiliem na tomada de decisões que estendam os objetivos e satisfaçam as exigências do projeto. Com isto - objetivos e exigências, decisões reconhecidas e metodologia efetiva de projeto- Teremos esperança de soluções lógicas para programas, em vez de obtê-las no ar.

3.2. Fases

O projeto de software pode ser visto nas seguintes fases:

• Projeto global

– Produto: Arquitetura do software

• Projeto detalhado

– Produtos:

Detalhes de processamento de cada módulo

Projeto de dados

A Arquitetura do Software estabelece a estrutura dos módulos do software e seus relacionamentos enfatizando coordenação (interação) sob computação (funcionalidade), como pode ser observado na Figura 21.

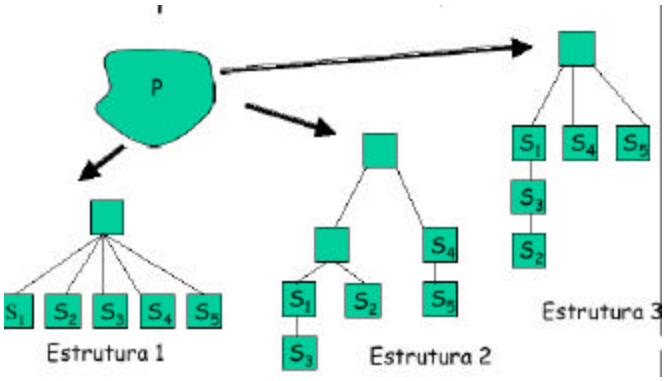


Figura 21: Arquitetura de Software.

3.3. Fundamentos

Pode se dizer que o projeto estrutura possui três métricas fundamentais que devem guiar o processo de projeto:

- Acoplamento
- Coesão
- Empacotamento

O tópico de acoplamento é central para a estratégia do Projeto Estruturado. É através da avaliação do acoplamento, que o projetista pode julgar a qualidade do seu projeto e ver que áreas precisam ser aprimoradas. Quando ele faz a mudança, o acoplamento ajuda-o e determina se esta mudança foi para melhor ou pior.

O **Acoplamento** é:

- Uma medida de interdependência de módulos, quanto mais alto o acoplamento, maior a probabilidade de que mudanças no interior de um módulo afetem o próprio funcionamento de outro módulo.
- Exerce também um forte efeito sobre a legibilidade do projeto e do código de cada módulo. Se o acoplamento é minimizado, você pode ler através de cada módulo, sem ter que olhar no interior de qualquer outro módulo.

A **coesão** é medida da força de associação dos elementos dentro de um módulo. Um módulo altamente coesivo é uma coleção de declarações e itens de dados, que deveriam ser tratados como um todo, porque estão relacionados muito intimamente. Qualquer tentativa em dividi-los só iria resultar no aumento do acoplamento e na diminuição da legibilidade.

O **empacotamento** refere-se à ação de moldar o projeto para acomodar o ambiente físico: a máquina, sistema operacional, linguagem de programação, limitações de memória e disco, restrições de tempo. O ponto de vista diferente que o Projeto Estruturado nos dá sobre empacotamento, não é visto sobre como trabalho com ele, mas quando trabalho com ele.

3.4. Ferramentas

Na literatura é possível encontrar muitas ferramentas que objetivam facilitar o projeto estruturado. Alguns exemplos são:

- Diagramas de Estrutura
- Modelo CPM/PERT
- Diagrama de Ação
- Diagrama de Warrier-Or
- Diagrama Hipo
- SADT (Structured Analysis Design Technique)

3.4.1. Diagramas de Estrutura

É a principal ferramenta utilizada em projeto estruturado para descrever a estrutura de um sistema, baseada em três componentes básicos: módulos, conexões e comunicações.

A notação adotada é baseada em retângulos e setas, onde os módulos representam subrotinas e as setas o fluxo. O fluxo pode ser de controle, de dados ou ambos.

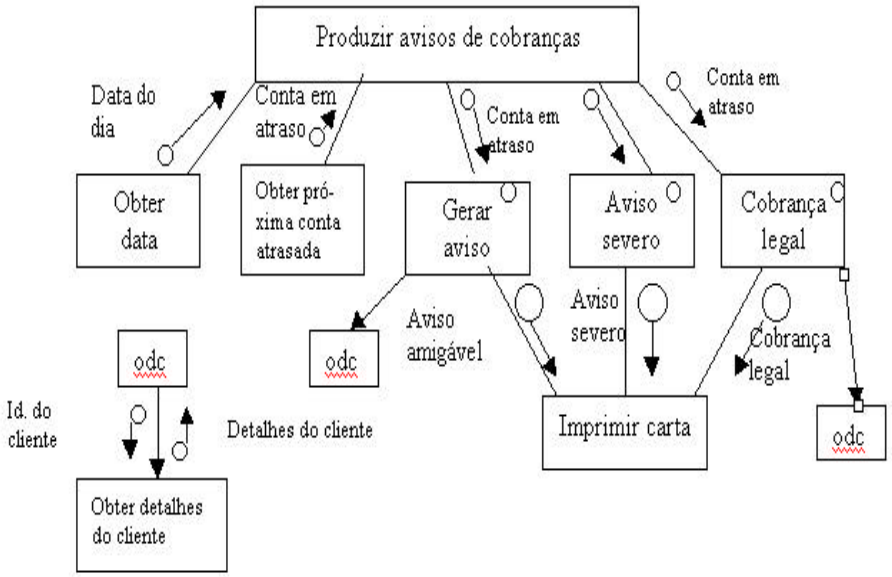


Figura 22: Exemplo de DE.

3.4.2. CPM-PERT

O modelo CPM-PERT é baseado em na junção de dois modelos:

- CPM (Método do Caminho Crítico)
- Pert (Técnica de Elaboração e Controle de Programas)

O CPM surgiu em 1957 e objetiva dividir um problema em partes menores. Utiliza um diagrama de flechas para possibilitar a visualização do projeto e das relações de dependência entre as atividades descritas.

O Pert teve origem em 1958 e foi concebido pela Equipe de projetos da Marinha dos Estados Unidos e as empresas: Lockheed e Cia. De Asses. Ind. Booz, Allen & Hamilton International Inc. O objetivo desse esforço foi promover a conclusão de projetos em um prazo prefixado. A maior conquista do Pert foi alertar para que problemas possam ser descobertos com antecedência.

Em termos de planejamento CPM e Pert apresentam as mesmas características, ou seja, dividir o problema em subpartes. No que diz respeito a programação o Pert diferencia-se do CPM pois não é possível se determinar o tempo de duração de uma atividade. O Pert trabalha com estimativas e não com intervalos precisos.

A Figura 23 apresenta uma especificação utilizando o modelo CPM/PERT onde pode-se observar os nós e os custos a eles associados.

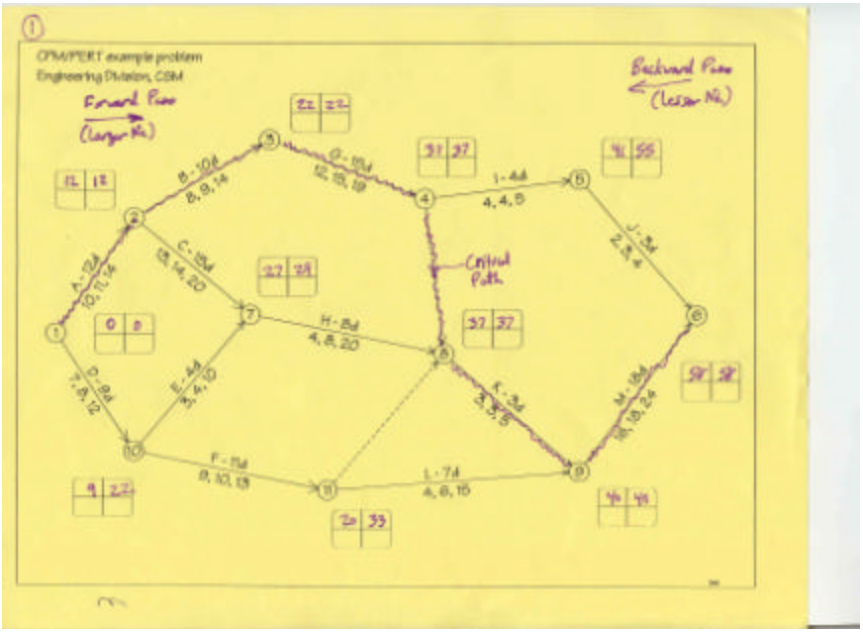


Figura 23: Exemplo de especificação em CPM/PERT.

3.4.3. Diagrama de Ação

Os diagramas de ação fornecem um meio natural de apresentar visões globais dos programas, de maneira análoga aos diagramas de estruturas, diagramas HIPO ou diagramas Warnier-Orr, e a lógica detalhada de modo similar aos fluxogramas ou diagramas de Nassi-Shneiderman.

As principais características dos diagramas de ação são:

- São desenhados e modificados com rapidez;
- São apropriados para o esboço manual e para a edição computadorizada;
- Possuem uma técnica única que expande a visão global de mais alto nível até os detalhes em nível de codificação (decomposição final);
- Representam as estruturas básicas da programação estruturada tradicional e são mais gráficos do que pseudocódigos;
- São projetados para serem relacionados com um modelo de dados;
- Funcionam bem com linguagens de Quarta geração (como FOCUS, RAMIS, IDEAL, MANTIS E NOMAD) e podem ser feitos sob medida para um dialeto específico;

– Permitem forte envolvimento do usuário final no projeto, por ser de fácil aprendizado.

– São projetados para a checagem cruzada, computadorizada, do uso de dados em especificações complexas.

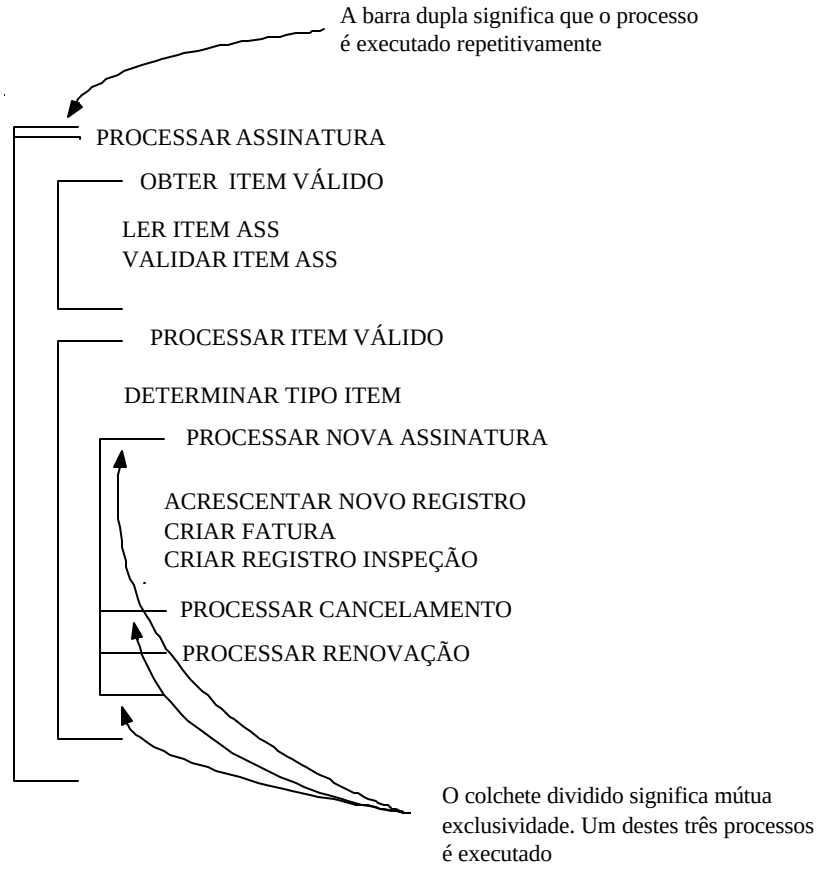


Figura 24: Exemplo de Diagrama de Ação.

3.4.4. Diagrama de Warrier-Orr

Representa graficamente a estrutura hierárquica de um programa, sistema ou estrutura de dados. Suas principais características são:

– é lido da esquerda para a direita e de cima para baixo;

– representado horizontalmente, transversalmente à página com chaves;

– cada chave do diagrama representa uma decomposição funcional do item especificado pela chave;

– quatro construções básicas: Hierarquia, Seqüência, Repetição, Seleção;

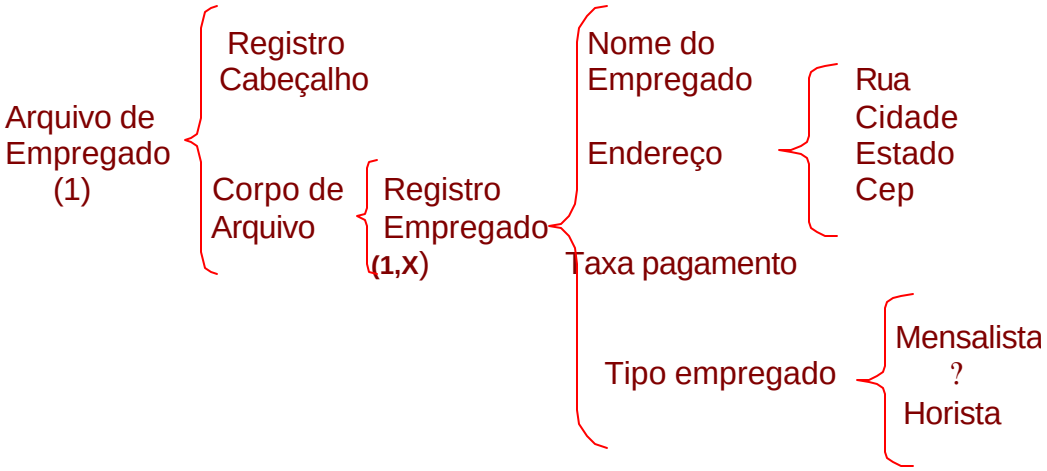


Figura 25: Exemplo de Warnier-Orr

3.4.5. Diagrama Hipo

Os diagramas HIPO foram desenvolvidos pela IBM nos anos 70. Atualmente, existem duas correntes de pensamento, baseadas na:

– Construção de diagramas usando apenas a essência da técnica HIPO;

– Utilização de procedimentos mais elaborados, complexos e próximo das recomendações da IBM.

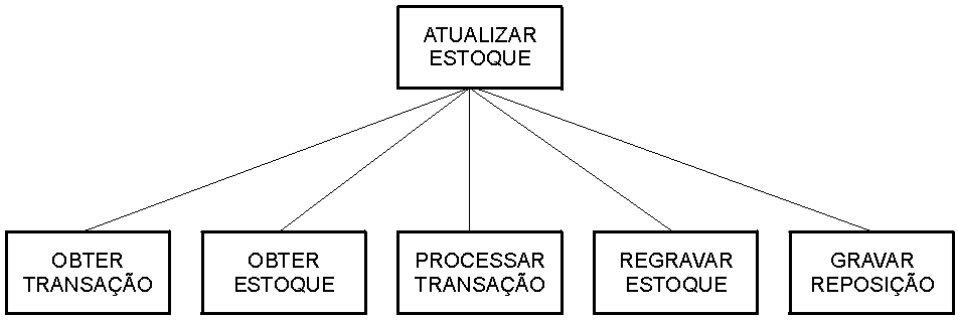
Sucintamente pode-ser dizer que as principais características dos diagramas HIPO são:

– Mostrar o que o sistema faz, e não como o sistema faz.

– Poder apresentar uma visão geral ou detalhada de um sistema.

– São inadequados sob vários aspectos e não constituem uma base para as ferramentas CASE

(a)



(b)

Quadro IPO

Sistema: Estoque
Módulo: Obter Estoque

Preparado por:João Silva
Data: 10/11/98

CHAMADO OU CONVOCADO POR:

Atualizar Estoque

CHAMA OU CONVOCA:

ENTRADAS:

NÚMERO-DA-PEÇA
SINAL-ERRO-BD

SAÍDAS:

ESTOQUE
SINAL-ERRO-BD

PROCESSO:

OBTER registro-BD do estoque
SE leitura do arquivo-BD for bem-sucedida
ENTÃO DESLIGAR sinal-erro-BD
CASO CONTRÁRIO (leitura do arquivo-BD não bem-sucedida)
ENTÃO LIGAR sinal-erro-BD

ELEMENTOS DE DADOS LOCAIS:

NOTAS: O sistema é on-line. Deve existir uma fila para transações pendentes.

Figura 26: Exemplo de Diagrama HIPO. (a) Diagrama HIPO (b) Quadro IPO.

3.4.6. SADT (Structured Analysis Design Technique)

É um modelo baseado na composição de Análise Estruturada (AS) na forma de linguagens de diagramação e técnicas de projeto (DT).

O método SADT é freqüentemente associado a outros métodos e ferramentas, principalmente na forma de diagramas de atividades e de diagramas de dados – na área de banco de dados.

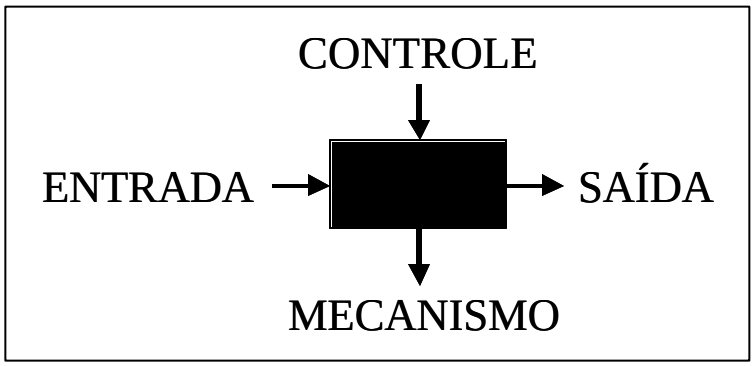


Figura 27: Primitivas Básicas do SADT:

A notação básica é baseada em caixas e setas. As caixas representam partes do todo de forma precisa e atividades ou dados. As setas definem entrada, saída, mecanismo e controle como pode ser observado na Figura 27.

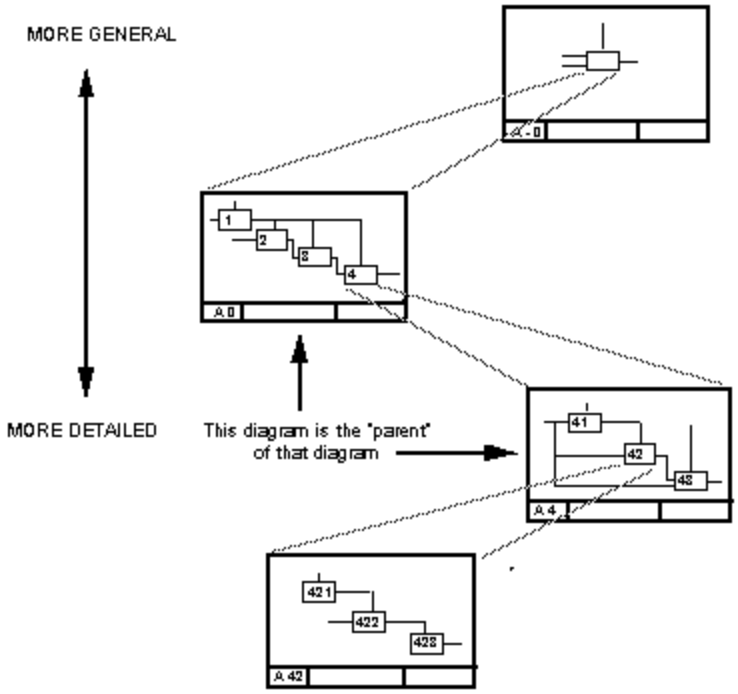


Figura 28: Modelo Estrutural SADT: