

Engenharia de Software

Capítulo 5

ANÁLISE DE REQUISITOS

1. INTRODUÇÃO

O completo entendimento dos requisitos de software é um ponto fundamental para o sucesso de um projeto de software. Independente da precisão com a qual um software venha a ser projetado e implementado, ele certamente trará problemas ao cliente/usuário se a sua análise de requisitos foi mal realizada.

A Análise de Requisitos é uma tarefa que envolve, antes de tudo um trabalho de descoberta, refinamento, modelagem e especificação das necessidades e desejos relativos ao software que deverá ser desenvolvido. Nesta tarefa, tanto o cliente como o desenvolvedor vão desempenhar um papel de grande importância, uma vez que caberá ao primeiro a formulação (de modo concreto) das necessidades em termos de funções e desempenho, enquanto o segundo atua como indagador, consultor e solucionador de problemas.

Esta etapa é de suma importância no processo de desenvolvimento de um software, principalmente porque ela estabelece o elo de ligação entre a alocação do software a nível de sistema (realizada na etapa de Engenharia de Sistema) e o projeto do software. Desta forma, ela permite que o engenheiro de sistemas especifique as necessidades do software em termos de funções e de desempenho, estabeleça as interfaces do software com os demais elementos do sistema e especifique as restrições de projeto. Ao engenheiro de software (ou analista), a análise de requisitos permite uma alocação mais precisa do software no sistema e a construção de modelos do processo, dos dados e dos aspectos comportamentais que serão tratados pelo software. Ao projetista, esta etapa proporciona a obtenção de uma representação da informação e das funções que poderá ser traduzida em projeto procedimental, arquitetônico e de dados. Além disso, é possível definir os critérios de avaliação da qualidade do software a serem verificados uma vez que o software esteja concluído.

2. AS ATIVIDADES DA ANÁLISE DE REQUISITOS

A etapa de Análise de Requisitos é caracterizada basicamente pela realização de um conjunto de tarefas, as quais serão discutidas nas seções que seguem.

2.1. A ANÁLISE DO PROBLEMA

Nesta tarefa inicial, o analista estuda os documentos de Especificação do Sistema e o Plano do Software, como forma de entender o posicionamento do software no sistema e revisar o escopo do software utilizado para definir as estimativas de projeto. Um elo de comunicação entre o analista e o pessoal da organização cliente deve ser estabelecido, sendo que o gerente de projetos pode atuar na coordenação dos contatos. A meta do

analista neste contexto é identificar os principais fatores do problema a ser resolvido, pela ótica do cliente.

2.2. A Avaliação e Síntese

Esta segunda tarefa envolve principalmente uma análise dos fluxos de informação e a elaboração de todas as funções de tratamento e os aspectos de comportamento do software. Ainda, é importante que uma definição de todas as questões relacionadas à interface com o sistema, além de uma especificação das restrições de projeto.

Terminada a análise, o analista pode iniciar a síntese de uma ou mais soluções para o problema. Na síntese das eventuais soluções, o analista deve levar em conta as estimativas e as restrições de projeto. Este processo de avaliação e síntese prossegue até que o analista e o cliente estejam de acordo sobre a adequação das especificações realizadas para a continuidade do processo.

2.3. A Modelagem

A partir da tarefa de avaliação e síntese, o analista pode estabelecer um modelo do sistema, o qual permitirá uma melhor compreensão dos fluxos de informação e de controle, assim como dos aspectos funcionais e de comportamento. Este modelo, ainda distante de um projeto detalhado, servirá de referência às atividades de projeto, assim como para a criação da especificação de requisitos.

Em muitas situações, como forma de reforçar o conhecimento sobre a viabilidade do software a ser desenvolvido, pode ser necessário o desenvolvimento de um protótipo de software como alternativa ou como trabalho paralelo à análise de requisitos. Este ponto será discutido mais adiante neste documento.

2.4. Especificação dos Requisitos e Revisão

A etapa de Análise de Requisitos culmina com a produção de um documento de **Especificação de Requisitos de Software**, o qual registra os resultados das tarefas realizadas. Eventualmente, pode ser produzido como documento adicional um Manual Preliminar do Usuário. Embora pareça estranho, a produção deste manual permite que o analista passe a olhar para o software da ótica do cliente/usuário, o que pode ser bastante interessante, principalmente em sistemas interativos. Além disso, a posse de um Manual de Usuário, mesmo em estágio preliminar permite ao cliente uma revisão dos requisitos (de interface, pelo menos) ainda num estágio bastante prematuro do desenvolvimento de software. Desta forma, algumas decepções resultante de uma má definição de alguns aspectos do software podem ser evitadas.

3. PROCESSOS DE COMUNICAÇÃO

O desenvolvimento de um software é, na maior parte dos casos, motivado pelas necessidades de um cliente, que deseje automatizar um sistema existente ou obter um novo sistema completamente automatizado. O software, porém, é desenvolvido por um desenvolvedor ou por uma equipe de desenvolvedores. Uma vez desenvolvido, o software será provavelmente utilizado por usuários finais, os quais não são necessariamente os clientes que originaram o sistema.

Isto significa que, de fato, existem três partes envolvidas no processo de desenvolvimento de um produto de software: o cliente, o desenvolvedor e os usuários. Para que o processo de desenvolvimento seja conduzido com sucesso, é necessário que os desejos do cliente e as expectativas dos eventuais usuários finais do sistema sejam precisamente transmitidos ao desenvolvedor.

Este processo de transmissão de requisitos, do cliente e dos usuários ao desenvolvedor, invariavelmente apresenta relativa dificuldade, considerando alguns aspectos:

- geralmente, os clientes não entendem de software ou do processo de desenvolvimento de um programa;
- o desenvolvedor, usualmente, não entende do sistema no qual o software vai executar.

Estes dois aspectos provam que existem, efetivamente, um problema de comunicação a ser resolvido para que o processo de desenvolvimento seja bem sucedido. A importância desta etapa está no fato de que é através dela que as idéias do cliente sobre o problema a ser resolvido pelo sistema a desenvolver são expressas na forma de um documento, se possível, que utilize ferramentas formais.

Uma Análise de Requisitos bem sucedida deve, normalmente, representar corretamente as necessidades do cliente e dos usuários, satisfazendo, porém às três partes envolvidas (incluindo o desenvolvedor). O que é verificado, em boa parte dos projetos de software é que o cliente nem sempre entende perfeitamente quais são as suas reais necessidades, assim como os usuários têm dificuldades para exprimir as suas expectativas com relação ao que está sendo desenvolvido. Um primeiro resultado desta etapa deve ser, sem dúvida, o esclarecimento a respeito do que são estas necessidades e expectativas.

A obtenção bem sucedida de informações é um fator preponderante para o sucesso da Análise de Requisitos, particularmente nas duas primeiras tarefas descritas anteriormente. A compilação das informações relevantes para o processo de desenvolvimento é uma tarefa bastante complexa, principalmente porque entram, muitas vezes, em jogo um conjunto de informações conflitantes. Ainda, quanto mais complexo é o sistema a ser desenvolvido, mais inconsistência haverá com relação às informações obtidas.

Neste contexto, o analista deve ter bom senso e experiência para extrair de todo o processo de comunicação as "boas" informações.

4. PRINCÍPIOS DE ANÁLISE

Independente do método utilizado para realizar a análise de requisitos, existem algumas preocupações que são comuns a todos eles, os quais discutiremos nos parágrafos que seguem.

4.1. O Domínio de Informação

Todo software é construído com a função básica de processar dados, não importa a área de atuação considerada. Como já discutimos no capítulo anterior, é possível representar um software na sua concepção mais clássica como um sistema que recebe um conjunto de informações (ou dados) de entrada, realiza o tratamento ou processamento desta informação e produz informações de saída.

Além dos dados, um software é capaz de processar eventos, onde cada evento está relacionado a um aspecto de controle do sistema, como por exemplo, um sensor que é capaz de detectar a ultrapassagem de um determinado limite de pressão ou temperatura pode gerar um evento (um sinal de alarme) para que o software de monitoração alerte o operador ou efetue alguma ação previamente definida.

Isto significa que os dados e os eventos fazem parte do domínio de informação do software, sendo que basicamente três pontos de vista podem ser considerados para abordar esta questão:

- o **fluxo da informação**, o qual representa a forma pela qual os dados e os eventos se modificam ao longo do sistema. o que pode ajudar a determinar quais

são as transformações essenciais às quais os itens de informação são submetidos;

- o **conteúdo da informação**, que permite representar os dados e os eventos que estejam relacionados a um determinado item de informação (semântica da informação);
- a **estrutura da informação**, a qual permite expressar a forma como itens de dados e/ou eventos estão organizados (sintaxe da informação); esta informação pode vir a ser importante para a etapa de projeto e implementação das estruturas de informação via software (linguagens de programação).

4.2. Modelagem

A modelagem é um outro aspecto de importância no processo de análise de requisitos de um software, uma vez que ela permite uma melhor compreensão das questões arquiteturais e comportamentais do problema a ser resolvido com o auxílio do software. Uma boa modelagem de software deve permitir a representação da informação a ser transformada pelo software, das funções (ou sub-funções) responsáveis das transformações e do comportamento do sistema durante a ocorrência destas transformações.

Um modelo realizado durante a etapa de Análise de Requisitos deve concentrar-se na representação do **que** o software deve realizar e não em **como** ele o realiza. Normalmente, é desejável que os modelos sejam expressos através de uma notação gráfica que permita descrever os diferentes aspectos citados no parágrafo anterior. Nada impede, porém, que um modelo seja dotado de descrições textuais complementares, sendo que normalmente, estas descrições devem ser realizadas ou por meio de linguagem natural ou de uma linguagem especializada que permita expressar, sem ambigüidades, os requisitos estabelecidos.

A obtenção de um modelo representativo dos requisitos do software pode ser útil às diferentes partes envolvidas no desenvolvimento do software:

- ao **analista**, para uma melhor compreensão da informação, as funções e o comportamento do sistema, o que pode tornar a tarefa de análise mais sistemática;
- ao **pessoal técnico** como um todo, uma vez que ele pode ser uma referência de revisão, permitindo a verificação de algumas propriedades, como a completude, a coerência e a precisão da especificação;
- ao **projetista**, servindo de base para o projeto através da representação dos aspectos essenciais do software.

4.3. Particionamento

Em boa parte das vezes, os problemas a serem resolvidos são excessivamente complexos, apresentando grande dificuldade para a sua compreensão (e conseqüente resolução) como um todo. Com a finalidade de dominar de forma completa os problemas sob análise, um princípio fundamental é a decomposição do mesmo em partes menores, o que denominaremos de particionamento. A partir do particionamento de um problema e a partir da análise de cada parte estabelecida, o entendimento fica mais facilitado. Desta forma, é possível estabelecer as interfaces de cada parte do problema de modo a que a função global do software seja realizada.

Segundo este princípio, as funções, os aspectos de comportamento e as informações a serem manipuladas pelo programa poderão ser alocadas às diferentes partes.

De um ponto de vista genérico, o procedimento básico de particionamento é o estabelecimento de uma estrutura hierarquizada de representação da função ou da

informação dividindo em partições o elemento superior, esta divisão podendo ser efetuada segundo uma abordagem vertical (deslocando-se verticalmente na hierarquia) ou horizontal. As figuras 5.1 e 5.2 representam a aplicação sobre um exemplo das abordagens horizontal e vertical, respectivamente.

No caso dos exemplos ilustrativos, o particionamento é realizado sobre o aspecto funcional do software, mas poderia ser aplicado, segundo uma ou outra abordagem, sobre os aspectos informacionais ou comportamentais.

4.4. Conceções essenciais e de implementação

Os requisitos de software podem ser especificados segundo dois critérios, dependendo do grau de conhecimento que se tem do sistema ou dependendo do estágio de análise no qual nos encontramos.

O primeiro critério é o da concepção essencial, onde o objetivo é contemplar os aspectos essenciais do problema sob análise, sem preocupação com detalhes de implementação. Considerando o exemplo de problema ilustrado nas figuras 5.1 e 5.2, a concepção essencial da função **ler status** ignora o formato dos dados de status ou o tipo de sensor que será utilizado. A vantagem de realizar a concepção essencial é de deixar em aberto as alternativas de implementação possíveis, o que é adequado para as atividades iniciais do desenvolvimento.

O segundo critério é o da concepção de implementação, o qual permite expressar os requisitos do software segundo as funções de processamento e as estruturas de informação do sistema real.

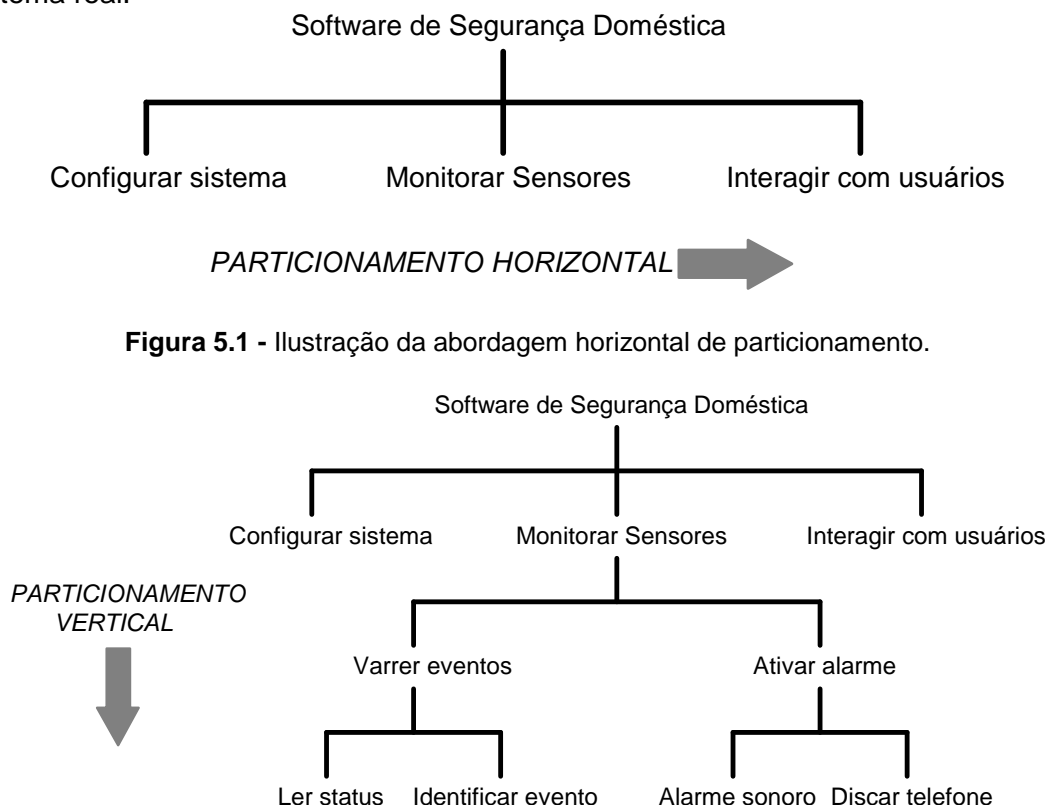


Figura 5.1 - Ilustração da abordagem horizontal de particionamento.

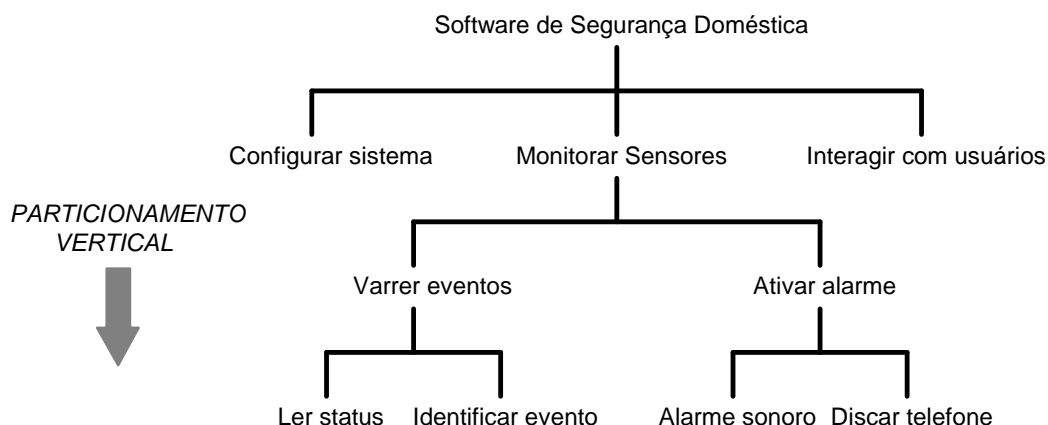


Figura 5.2 - Ilustração da abordagem vertical de particionamento.

Em alguns casos, uma representação física é o ponto de partida da etapa de projeto do software mas, na maioria dos casos, grande parte dos sistemas computacionais é especificada acomodando alguns detalhes de implementação. O conhecimento de alguns destes detalhes pode auxiliar o analista (e, em seguida, o projetista) na definição de restrições impostas ao software pelo sistema.

5. PROTOTIPAÇÃO DE SOFTWARE

5.1. Motivações e Etapas da Prototipação

A etapa de Análise de Requisitos engloba um conjunto de atividades de fundamental importância para o processo de desenvolvimento de um software e deve ser, portanto, realizada independentemente da abordagem e técnica utilizadas.

Em muitos casos, é possível aplicar-se técnicas de análise de modo a derivar uma representação em papel (ou em arquivo, no caso da utilização de uma ferramenta CASE) que sirva de referência para o projeto do software.

Em outras situações, a partir da coleta de requisitos um modelo do software — um **protótipo** — pode ser construído de modo a permitir uma melhor avaliação por parte do desenvolvedor e do cliente.

O paradigma da prototipação tem como ponto de partida uma **Solicitação de Proposta** encaminhada pelo cliente. A partir desta solicitação, os seguintes passos são realizados:

- análise da solicitação para identificar a necessidade da prototipação; normalmente, softwares interativos e/ou gráficos ou softwares que exijam a utilização de algoritmos combinatórios podem ser objeto de prototipação; no entanto, o fator complexidade deve permitir determinar a realização ou não do protótipo; outro ponto que deve ser pesado nesta decisão é se a equipe de desenvolvimento tem experiência e ferramentas adequadas à prototipação;
- especificação resumida dos requisitos, realizada pelo analista, de modo a representar o domínio da informação e os domínios funcionais e comportamentais do software, de modo que a atividade de particionamento do problema possa ser efetuada;
- revisada a especificação resumida dos requisitos, é realizado um projeto do protótipo, concentrando-se principalmente nos aspectos arquitetônicos e de dados e deixando de lado os aspectos procedimentais;
- desenvolvimento do protótipo, se possível a partir da utilização de blocos de construção de software preexistentes (o que na maioria dos casos são de difícil obtenção); por outro lado, a construção de um protótipo pode ser facilitada graças à existência de diversas ferramentas orientadas a esta atividade;
- apresentação do protótipo (testado) ao cliente, para que este possa efetuar sua avaliação; a partir desta avaliação, o cliente pode sugerir extensões ou reformular alguns requisitos de modo a que o software possa melhor corresponder às reais necessidades;
- repetição iterativa dos dois passos anteriores, até que todos os requisitos tenham sido formalizados ou até que o protótipo tenha evoluído na direção de um sistema de produção.

5.2. Métodos e Ferramentas de Prototipação

O desenvolvimento de um protótipo é uma atividade que deve ser realizada num tempo relativamente curto, mas de forma a representar fielmente os requisitos essenciais do software a ser construído. Para conduzir de modo eficiente esta atividade, já se dispõe de três classes de métodos e ferramentas, as quais são apresentadas a seguir:

- as **técnicas de quarta geração (4GT)**, as quais englobam conjuntos de linguagens para geração de relatórios e de consulta a bancos de dados e derivação de aplicações e programas; a área de atuação destas técnicas é

atualmente limitada aos sistemas de informação comerciais, embora estejam aparecendo algumas ferramentas orientadas a aplicações de engenharia;

- os **componentes de software reusáveis**, os quais permitem a "montagem" do protótipo a partir dos "blocos de construção" (*building blocks*), dos quais não se conhece necessariamente o seu funcionamento interno, mas as suas funções e interfaces são dominadas pelo analista; o uso desta técnica só é possível a partir da construção (ou existência) de uma biblioteca de componentes reusáveis, catalogados para posterior recuperação; em alguns casos, um software existente pode ser adotado como "protótipo" para a construção de um novo produto, mais competitivo e eficiente, o que define uma forma de reusabilidade de software;
- as **especificações formais** e os **ambientes de prototipação**, que surgiram em substituição às técnicas baseadas em linguagem natural; as vantagens da utilização destas técnicas são, basicamente, a representação dos requisitos de software numa linguagem padrão, a geração de código executável a partir da especificação e a possibilidade de validação (da parte do cliente) de modo a refinar os requisitos do software.

6. ESPECIFICAÇÃO DOS REQUISITOS DE SOFTWARE

Como foi exposto no início do capítulo (item 2.4), a etapa de Análise de Requisitos culmina com a produção de um documento de Especificação dos Requisitos de Software. Este resultado pode vir na forma de um documento em papel utilizando linguagem natural ou gráfica, pode ter sido produzida com o auxílio de uma ferramenta CASE, ou ainda pode ser complementada ou expressa a partir de um protótipo construído nesta etapa.

Este documento é basicamente o resultado de um trabalho de representação das necessidades para a solução do problema analisado, devendo expressar os aspectos funcionais, informacionais e comportamentais do software a ser construído.

De modo a obter uma especificação eficiente, alguns princípios podem ser considerados:

- a separação entre funcionalidade e implementação;
- utilização de uma linguagem de especificação orientada ao processo;
- a especificação deve levar em conta o sistema do qual o software faz parte e o ambiente no qual o sistema vai operar;
- a especificação deve representar a visão que os usuários terão do sistema;
- uma especificação deve ser operacional, no sentido de que ela deve permitir, mesmo num nível de abstração elevado, algum tipo de validação;
- uma especificação deve ser localizada e fracamente acoplada, o que são requisitos fundamentais para permitir a realização de modificações durante a análise de requisitos.

A figura 5.3 apresenta uma proposta de estrutura para o documento de Especificação dos Requisitos do Software. O documento inicia com uma Introdução, que apresenta as principais metas do software, descrevendo o seu papel no contexto do sistema global. As informações prestadas nesta seção podem ser, inclusive, inteiramente extraídas do documento de Plano de Software.

1.0 - Introdução
2.0 - Descrição da Informação
2.1 - Diagramas de Fluxos de Dados
2.2 - Representação da Estrutura dos Dados
2.3 - Dicionários de Dados
2.4 - Descrição das Interfaces do Sistema
2.5 - Interfaces Internas
3.0 - Descrição Funcional
3.1 - Funções
3.2 - Descrição do Processamento
3.3 - Restrições de Projeto
4.0 - Critérios de Validação
4.1 - Limites de Validação
4.2 - Classes de Testes
4.3 - Expectativas de Resposta do Software
4.4 - Considerações Especiais
5.0 - Bibliografia
6.0 - Apêndices

Figura 5.3 - Estrutura do documento de Especificação de Requisitos.

A seção de Descrição da Informação deve apresentar informações sobre o problema que o software deve resolver. Os fluxos e a estrutura das informações devem ser descritos nesta seção (utilizando um formalismo adequado, como, por exemplo, os DFDs e os Dicionários de Dados), assim como os elementos relacionados às interfaces internas e externas do software (incluindo hardware, elementos humanos, outros softwares, etc...).

A seção seguinte, Descrição Funcional, apresenta as informações relativas às funções a serem providas pelo software, incluindo uma descrição dos processamentos envolvidos no funcionamento do software. Ainda nesta seção, devem ser apresentadas as restrições de projeto, detectadas nesta etapa.

A seção de Critérios de Validação, que apesar de ser a mais importante é aquela sobre a qual menor atenção é dispensada, vai estabelecer os parâmetros que permitirão avaliar se a implementação do software vai corresponder à solução desejada para o problema. Definir os critérios de validação significa ter um entendimento completo dos requisitos funcionais e de processamento de informação do software. Uma definição importante a ser encaminhada nesta seção é o conjunto de testes que deverá ser aplicado à implementação para que se tenha uma garantia do funcionamento do software nos moldes estabelecidos pela especificação dos requisitos.

Finalmente, devem ser registrados neste documento a lista de documentos relacionados ao software a ser desenvolvido (Plano de Software, por exemplo) e uma seção de Apêndices, onde podem ser apresentadas informações mais detalhadas sobre a especificação do software (descrição detalhada de algoritmos, diagramas, gráficos, etc...).

Um outro aspecto interessante, não apresentado na figura, refere-se a quando um software terá características de interatividade com usuários. Neste caso, é interessante que um Manual de Usuário (em versão preliminar) seja elaborado, o qual vai conduzir a duas consequências positivas: forçar o analista a enxergar o software do ponto de vista do usuário; permitir ao cliente uma avaliação do que será o software nesta etapa de desenvolvimento.

7. ANÁLISE ESTRUTURADA

A Análise Estruturada ou SSA (para *Structured System Analysis*) foi desenvolvida em meados dos anos 70 por Gane, Sarson e De Marco. A técnica SSA é baseada na utilização de uma linguagem gráfica para construir modelos de um sistema, incorporando

também conceitos relacionados às estruturas de dados. Os elementos básicos da Análise Estruturada são: o Diagrama de Fluxo de Dados, o Dicionário de Dados, as Especificações de Processos e as Técnicas de Estruturação de Bases de Dados.

Nos parágrafos que seguem, apresentaremos os principais fatores e notações que regem a Análise Estruturada, ilustrando por meio de exemplos, quando necessário.

7.1. Os Diagramas de Fluxos de Dados

Em primeira instância, um sistema computacional pode ser representado segundo as informações que ele manipula e o fluxo destas informações ao longo do sistema. À medida que se deslocam ao longo de um software, as informações de entrada vão sofrendo transformações no sentido da obtenção das informações de saída.

Um Diagrama de Fluxo de Dados (DFD) é uma técnica gráfica de representação que permite explicitar os fluxos de informação e as transformações que são aplicadas à medida que os dados se deslocam da entrada em direção à saída. Um DFD assume o formato de um esquema como o ilustrado na figura 5.4.

Num DFD, os processos ou atividades de transformação são caracterizados por **círculos** ou **bolhas** identificadas por uma expressão que descreva precisamente o processo ou o tipo de transformação realizada sobre os dados. O fluxo de dados é expresso por **setas rotuladas** que interligam os processos e que permitem indicar o caminho seguido pelos dados. **Retângulos rotulados** vão definir as origens ou destinos dos dados envolvidos no sistema, representando os geradores ou consumidores das informações. Os geradores ou consumidores das informações não são considerados objeto de análise do problema. Finalmente, uma **linha dupla** preenchida com um rótulo permite expressar depósitos de dados. A figura 5.5 apresenta os símbolos básicos de um DFD.

Esta representação pode ser adotada seja para o sistema seja para o software como elemento de um sistema, sendo possível definir diversas partições como forma de apresentar diferentes níveis de detalhamento do software ou do sistema.

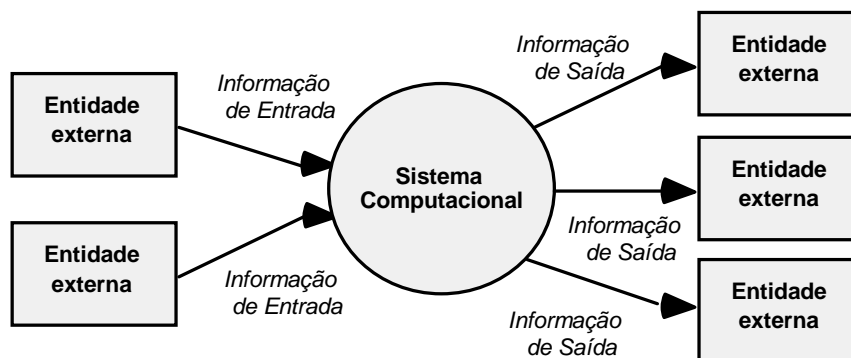


Figura 5.4 - Formato genérico de um DFD.

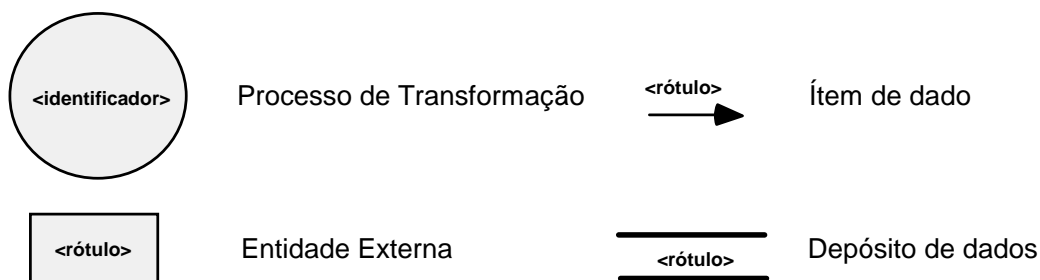


Figura 5.5 - Símbolos gráficos de um DFD.

O primeiro nível de representação, ou nível 0 do DFD é o **modelo fundamental do sistema** (ou **modelo de conteúdo**), utilizado para representar o elemento de software global como um único círculo e as informações de entrada e saída representadas pelas setas rotuladas entrando e saindo do círculo. Neste primeiro nível são representados por retângulos os elementos externos ao software que geram e consomem as informações. À medida que se vai evoluindo na análise, novos processos de transformação vão sendo acrescentados nos DFDs de níveis inferiores, correspondendo aos detalhes realizados em processos de um dado nível. A figura 5.6 ilustra esta situação.

Um exemplo de DFD, resultante da análise de um sistema de folha de pagamento é apresentado na figura 5.7. Quando múltiplos dados são necessários por um processo, um asterisco (*) é introduzido junto aos diferentes fluxos de dados (representando um relacionamento "E").

Um relacionamento do tipo "OU" pode também ser definido, se necessário, pela introdução de um sinal de adição (+) entre os fluxos de dados.

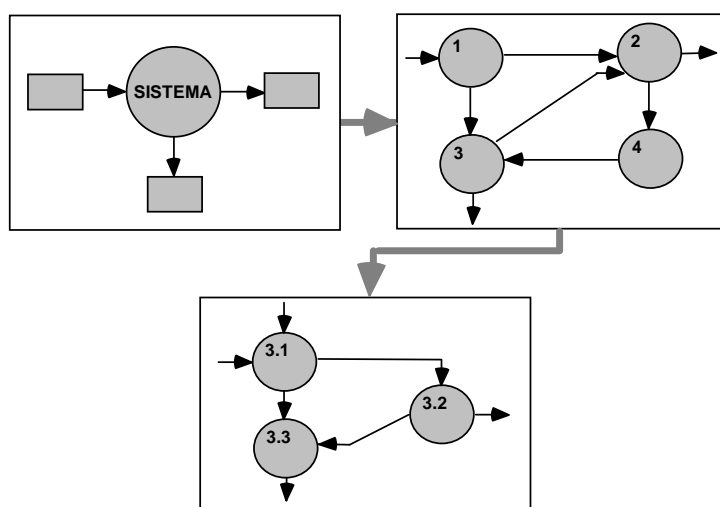


Figura 5.6 - Refinamento de um DFD.

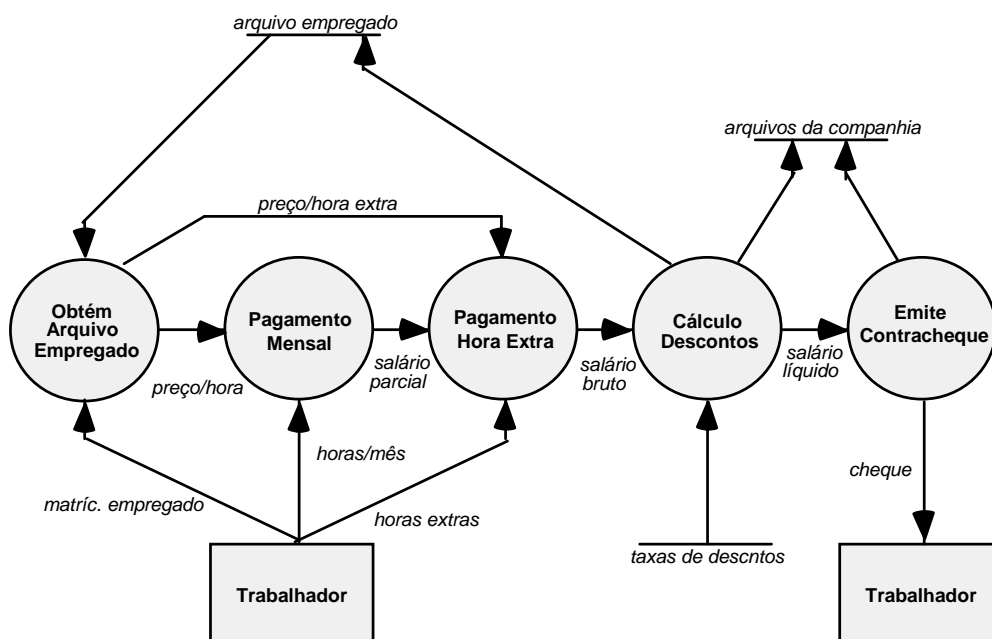


Figura 5.7 - DFD de um sistema de folha de pagamento.

É importante ressaltar que o DFD da figura 5.7 corresponde a um sistema de folha de pagamento, independente da forma como este é implementado (manual ou automático), o que sugere o carácter de abstracção deste método.

Nesta primeira visão do sistema exemplo, detalhes relativos ao seu funcionamento não são apresentados (por exemplo, o que fazer se um erro na ficha de horas/mês é detectado?). Isto permite ter uma visão mais simples do sistema global. Caso alguns detalhes sejam considerados importantes, o DFD poderá ser refinado numa etapa posterior.

É preciso destacar aqui que um DFD não é um fluxograma. Um DFD exprime apenas o fluxo de dados, enquanto o fluxograma exprime o fluxo de controle. Num DFD, questões relacionadas a procedimentos de execução devem ser omitidas sempre; aspectos como decisões, malhas de controle, etc... não fazem parte de um DFD.

O objetivo principal de um DFD é expressar **quais** as transformações são aplicadas aos dados, sem preocupação em representar **como** são processadas estas transformações.

O primeiro passo para a construção de um DFD para um dado problema é a identificação das principais entradas e saídas. Informações de entrada e saída de menor importância (como, por exemplo, mensagens de erro) podem ser ignoradas num primeiro tempo. Uma vez identificadas as entradas, uma abordagem pode ser partir das entradas e identificar as principais transformações pelas quais estas passam, até atingirem (ou transformarem-se em) as saídas. Um outro princípio seria caminhar no sentido inverso, ou seja, das saídas em direção às entradas.

Os passos a seguir correspondem a algumas sugestões para a construção de um DFD:

- iniciar a construção do DFD, partindo das entradas para as saídas ou vice-versa, procurando definir as principais (poucas, em princípio) transformações de dados; refinando, em seguida, estas transformações em conjuntos de transformações mais específicas;
- evitar sempre a expressão de fluxo de controle; eliminar da análise qualquer raciocínio que sugira decisão ou malhas;
- rotular as setas com os nomes dos dados apropriados; entradas e saídas de cada transformação devem ser cuidadosamente identificadas;
- utilizar os operadores * e + sempre que necessário;
- quando possível, traçar diversos DFDs de um mesmo problema, ao invés de adotar o primeiro construído.

7.2. O Dicionário de Dados

Num DFD, os dados são identificados por um rótulo único, que represente claramente o significado da informação. Por outro lado, a definição da estrutura de dados que vai representar a informação não é feita a nível do DFD. Para isto, é construído um **Dicionário de Dados**, o qual se constitui de um depósito de todos os fluxos de dados explicitados no DFD associado. Os componentes de um fluxo de dados do DFD podem ser então especificados no dicionário de dados, assim como a estrutura dos eventuais arquivos definidos no diagrama. As notações mais diversas podem ser utilizadas para definir as estruturas dos dados, sendo que uma proposta de notação possível inclui as seguintes informações:

- **nome**, o identificador principal do item de dados, do depósito de dados ou de uma entidade externa;
- **alias**, eventualmente outros nomes utilizados para o mesmo item;
- **utilização**, em que contexto (onde e como) o item de informação é utilizado;
- **descrição**, uma notação que permita explicitar o conteúdo do item;

- **informações complementares** a respeito do item de dados, como valores iniciais, restrições, etc...

Atualmente, a obtenção do dicionário de dados é resultante do uso de uma ferramenta CASE, sendo que esta pode inclusive checar a consistência da especificação com relação a este aspecto da definição. Por exemplo, caso o analista nomeie um item de dados recém derivado com um identificador que já faça parte do dicionário, a ferramenta devolve uma mensagem de erro indicando a duplicação do identificador. Outro aspecto positivo do uso de uma ferramenta CASE é a geração de algumas informações de forma automatizada. A informação **utilização** é um exemplo disto, uma vez que ela pode ser derivada a partir dos fluxos de dados.

Apresentamos abaixo o dicionário de dados definido para alguns dos itens de dados apresentados no DFD da figura 5.7.

```

nome      : matrícula_empregado
alias     : nenhum
utilização : obtém arquivo empregado (entrada)
descrição : matricula_empregado = dígito + dígito + dígito + dígito

nome      : preço/hora
alias     : nenhum
utilização : obtém arquivo empregado (entrada)
descrição : preço/hora = valor em dolar

```

Para formalizar a descrição dos itens de dados, utiliza-se um conjunto de operadores que permita compor representações que poderão ser mapeadas futuramente em estruturas de dados de uma dada linguagem de programação. Alguns dos operadores e construtores utilizados são:

$x = a + b$	x possui os elementos de dados a e b
$x = [a \mid b]$	x possui a ou b (escolha)
$x = (a)$	x possui um elemento de dados opcional a
$x = \{a\}$	x possui de zero a mais ocorrências de a
$x = y(a)$	x possui y ou mais ocorrências de a
$x = (a)z$	x possui z ou menos ocorrências de a
$x = y(a)z$	x possui entre y e z ocorrências de a

7.3. A Descrição de Processos

Um outro aspecto importante na análise de requisitos é a Especificação de Processos. O objetivo da especificação de processo é auxiliar o analista a descrever, de forma precisa, o comportamento de alguns componentes do sistema (ou processos) definidos nos DFDs de nível mais baixo. A especificação de processos pode ser realizada segundo diversas técnicas, mas uma das mais interessantes é o texto estruturado, o qual é baseado num grupo limitado de verbos e substantivos organizados de modo a representar um compromisso entre a legibilidade e a precisão da especificação. O texto estruturado é baseado, principalmente, nos seguintes elementos:

- um conjunto limitado de verbos de ação (por exemplo, calcular, encontrar, imprimir, etc...);
- estruturas de controle já conhecidas da programação estruturada (IF-THEN-ELSE, DO-WHILE, CASE, etc...);
- elementos de dados definidos no Dicionário de Dados.

Um exemplo de especificação de processo é apresentada abaixo, representando um dos processos definidos no DFD da figura 5.7.

PROCESSO Calcula_Desconto

```
IF Empregado é isento de descontos
  THEN Salário Líquido é igual a Salário Bruto
ELSE
  IF Salário Bruto é maior que X
    THEN Desconto é de 25%
  ELSE
    IF Salário Bruto é maior que Y
      THEN Desconto é de 20%
    ELSE Desconto é de 15%
```

FIM Calcula_Desconto

Outras técnicas podem ser utilizadas para a especificação de um processo, como os fluxogramas, as tabelas de decisão, etc...

7.4. Representação da Relação entre os Dados

Um quarto elemento relacionado à análise estruturada objetiva o estabelecimento de uma relação entre os diferentes dados definidos no DFD e no dicionário de dados. Uma forma de representar estas relações é através dos diagramas Entidade-Relação. É uma ferramenta gráfica que permite definir as relações entre as diferentes entidades definidas num dado sistema. Fugindo um pouco do exemplo da figura 5.7, é apresentado, na figura 5.8, um exemplo de diagrama Entidade-Relação para um sistema de tráfego aéreo.

Como se pode notar no exemplo, os retângulos representam as entidades do sistema e os losangos as possíveis relações entre estas.

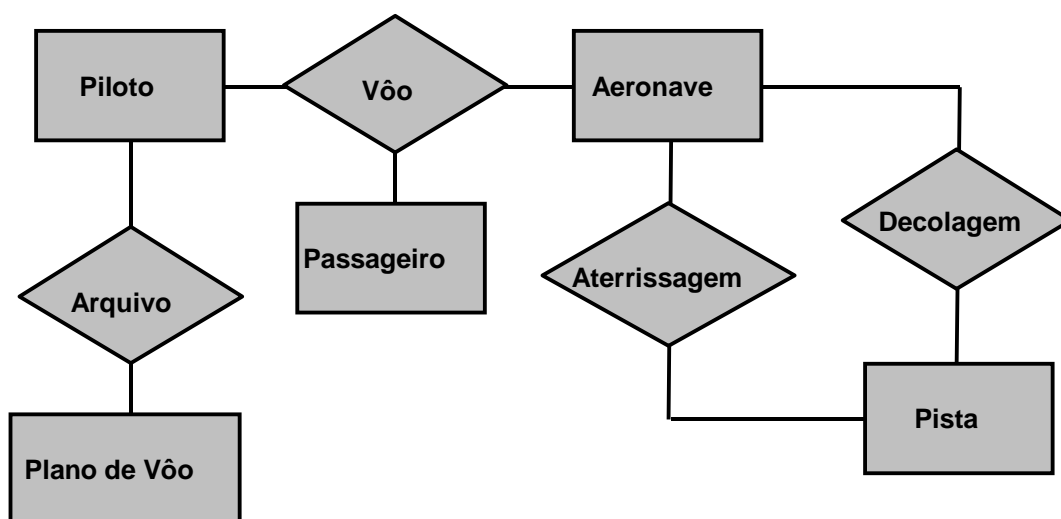


Figura 5.8 - Exemplo de Diagrama Entidade-Relação.

8. TÉCNICAS DE ANÁLISE

8.1. Técnicas Orientadas Estado

Apesar de apresentarem aspectos interessantes no que diz respeito ao desenvolvimento da tarefa de Análise de Requisitos, as ferramentas da técnica de Análise Estruturada de Sistemas não permitem cobrir todos os aspectos importantes de determinadas classes de sistemas.

Um exemplo disto são os sistemas reativos e de tempo-real, onde a ordenação de eventos e os aspectos de temporização podem assumir uma importância fundamental no entendimento do problema a ser resolvido.

Por esta razão, nestes casos, é, sem dúvida, mais interessante realizar a formalização do problema utilizando uma técnica baseada no conceito de estados. Os parágrafos a seguir apresentarão algumas das técnicas orientadas a estados que são úteis na etapa de Análise de Requisitos.

8.1.1. As Tabelas de Decisão

As tabelas de decisão são utilizadas para expressar decisões lógicas de alta complexidade. Estas são caracterizadas por quatro quadrantes: o quadrante de condições, o quadrante de ações, as entradas de condições e as entradas de ações.

O quadrante de condições contém todas as condições que serão examinadas na análise de um problema. As entradas de condições servem para combinar as condições com as regras de decisão estabelecidas na parte superior da tabela.

O quadrante de ações define todas as ações que deverão ser tomadas como resposta às regras de decisão. As entradas de ação relaciona as regras de decisão às ações.

A figura 5.9 ilustra o formato padrão de uma tabela de decisão.

A figura 5.10 apresenta um exemplo de uma tabela de decisões com entradas limitadas. As entradas possíveis são **S**, **N**, - e **X**, que denotam, respectivamente, SIM, NÃO, TANTO FAZ e REALIZE A AÇÃO. Como se pode verificar na figura, um determinado empréstimo é autorizado nas seguintes situações: se o limite de crédito não foi ultrapassado, se o limite foi ultrapassado mas a experiência de pagamentos é positiva, se uma liberação especial de crédito pode ser obtida. Em outras condições, a ordem é rejeitada.

As entradas **S**, **N** e - em cada coluna caracterizam uma *regra de decisão*. Caso duas ou mais regras tenham as entradas idênticas, a tabela é considerada *ambígua*.

	Regras de Decisão			
	<i>Regra 1</i>	<i>Regra 2</i>	<i>Regra 3</i>	<i>Regra 4</i>
<i>Quadr. de Condições</i>		<i>Entradas de Condições</i>		
<i>Quadrante de Ações</i>		<i>Entradas de Ações</i>		

Figura 5.9 - Formato geral de uma tabela de decisões.

	Regras de Decisão			
	Regra 1	Regra 2	Regra 3	Regra 4
Limite de crédito é satisfatório	S	N	N	N
Experiência de pagamentos é favorável	–	S	N	N
Liberação especial é obtida	–	–	S	N
Autorize empréstimo	X	X	X	
Rejeite empréstimo				X

Figura 5.10 - Uma tabela de decisões com entradas limitadas.

Caso duas ou mais regras idênticas conduzam às mesmas ações, elas são consideradas *redundantes*. Caso elas conduzam a ações diferentes, elas são consideradas *contraditórias*. Regras contraditórias não são necessariamente indesejáveis; Elas podem ser utilizadas para expressar o não-determinismo de um dado problema.

Uma tabela de decisões é considerada completa se todas as possíveis combinações de condições podem ser associadas a uma ação. Numa tabela com N entradas de condição, vão existir 2^N possíveis combinações. A não especificação de uma ou mais combinações vai resultar numa tabela de decisões incompleta.

8.1.2. As tabelas de transição

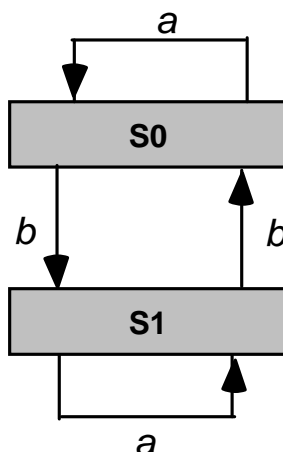
As tabelas de transição são utilizadas para especificar mudanças no estado de um sistema como resultado da ação de eventos característicos. O estado de um sistema caracteriza as condições de todas as entidades do sistema num dado instante. Para um estado S_i , a ocorrência de uma condição C_j vai provocar a passagem ou transição ao estado S_k .

Um exemplo de tabela de transição é apresentada na figura 5.11, que representa um sistema composta de dois estados (S_0, S_1) nos quais duas entradas (a e b) podem ocorrer. Como se pode notar na figura, as entradas rotuladas por a fazem com que o sistema permaneça no seu estado atual (S_0 ou S_1). As entradas rotuladas por b vão causar uma transição para S_1 , se o sistema estiver em S_0 ou para S_0 se o sistema estiver em S_1 .

8.1.3. Os diagramas de transição de estado

Uma técnica também interessante é o **diagrama de transição de estados**. Neste diagrama, *retângulos* ou *círculos* representam os **estados** de um processo ou sistema. A passagem ou **transição** de um estado para o outro é representada, neste diagrama, por *setas* que definem a direção da transição. Tanto os estados quanto as transições são rotuladas; os primeiros indicam nomes para os estados; os rótulos associados às setas definem os eventos que provocam as transições de estados, assim como as ações resultantes destas transições. A figura 5.12 ilustra um diagrama de transição de estado para o sistema representado pela tabela de transições da figura 5.6.

ESTADO CORRENTE	ENTRADA CORRENTE	
	<i>a</i>	<i>b</i>
S0	S0	S1
S1	S1	S0

Figura 5.11 - Uma tabela de transição com entradas limitadas.**Figura 5.12** - Diagrama de Transição de Estados.

8.2. SADT

Uma das ferramentas mais difundidas para a realização da análise de requisitos é, sem dúvida, o **SADT** ou **Structured Analysis and Design Technique**. A técnica SADT vem sendo utilizada há mais de 15 anos por um grande conjunto de organizações em todo o mundo.

O SADT foi concebido de modo a apresentar as principais características desejáveis num modelo orientado à Análise e Especificação de Requisitos. Dentre estas características, podemos relacionar:

- o uso de uma linguagem gráfica de comunicação;
- a possibilidade de representação de um problema obedecendo a uma política de decomposição estruturada, de cima para baixo (top-down);
- limitar a quantidade de informações podendo conter num dado nível de representação de um problema, segundo os limites que a mente humana pode absorver;
- permitir a representação de um problema segundo dois pontos de vista (o ponto de vista das atividades e ponto de vista dos dados) o que satisfaz a um dos princípios importantes da Análise de Requisitos que é o princípio da projeção.

Um modelo **SADT** consiste de um conjunto ordenado de diagramas, onde cada diagrama é traçado numa página única. Um diagrama é composto em média de três a seis blocos, interconectados por um conjunto de arcos dirigidos.

8.2.1. A Linguagem de Representação Gráfica

O princípio que rege o modelo SADT é o fato de que o conhecimento sobre o mundo e seus sistemas é fundamento sobre **coisas** e **acontecimentos**, ou melhor, de **objetos** e **eventos**, ou ainda, de **dados** e **atividades**.

A linguagem de representação proposta no SADT é baseada na construção de um conjunto de diagramas, onde os elementos básicos são os **blocos** e as **setas**. O tipo de informação representado pelas setas depende do tipo de diagrama construído no modelo.

Existem basicamente dois tipos de diagramas definidos no modelo SADT: os actigramas e os datagramas. Nos **actigramas**, os blocos designam as atividades relacionadas ao sistema sob análise, sendo que os arcos representam os fluxos de dados entre as atividades. Os actigramas são, de certo modo, um equivalente dos diagramas de fluxos de dados dentro do modelo SADT. Nos **datagramas**, os blocos especificam objetos de dados e os arcos as atividades ou ações definidas sobre estes dados. Os actigramas e os datagramas têm, estabelecida entre si, uma relação de dualidade.

Na prática, os actigramas são utilizados com muito mais frequência que os datagramas. Entretanto, o uso de datagramas assume um nível elevado de importância por duas razões principais:

- para especificar todas as atividades relacionadas com um dado objeto de dado;
- para verificar a coerência de um modelo SADT pela construção de datagramas a partir de actigramas.

A forma geral dos blocos de um actigrama e de um datagrama é apresentada na figura 5.13. Em 5.13.(a) é mostrado o bloco relacionado a um actigrama. Na sua forma geral, os possíveis fluxos de dados são os de entrada, de saída, de controle e o mecanismo. As **saídas** de um bloco de actigrama podem representar entradas ou controles de um outro bloco do mesmo actigrama e deverão, desta forma, ser conectadas a outros blocos do actigrama ou ao ambiente externo.

As **entradas** e **controles** de um bloco de actigrama deverão, similarmente, ser saídas de outros blocos ou dados do ambiente externo. Os **controles** de uma atividade são dados utilizados na atividade mas que não são modificados por ela. Os **mecanismos** são os elementos necessários à realização daquela atividade, como por exemplo, os processadores.

Um exemplo de representação por actigrama é apresentado na figura 5.14. No exemplo, o processo de escrita de um texto, a **atividade** é representada por um bloco rotulado pelo verbo **redigir**. O dado de entrada da atividade é a idéia e a saída é o texto.

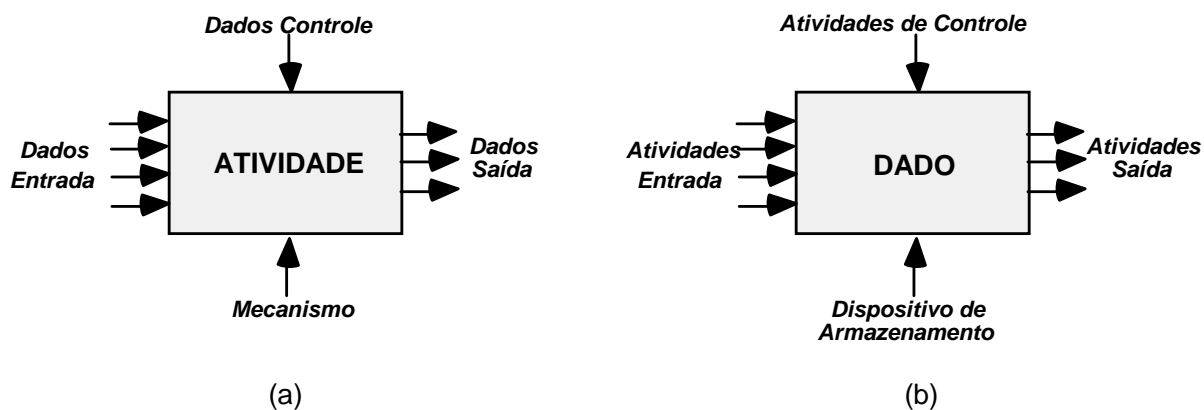


Figura 5.13 - Blocos de composição de um actigrama (a) e de um datagrama (b).

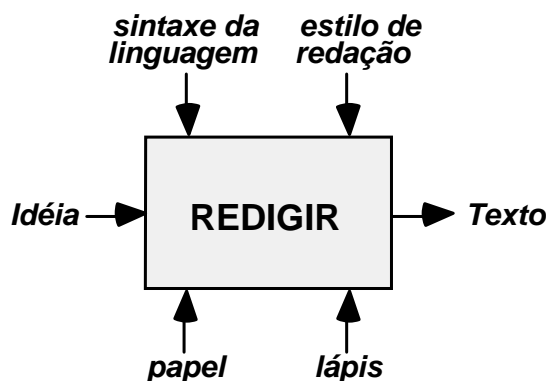


Figura 5.14 - Exemplo de representação em SADT: o processo de redação de um texto.

Como controle, tem-se a **sintaxe da linguagem** e o **estilo de redação**. Finalmente, os **mecanismos** considerados são o **lápiz** e o **papel**.

Num datagrama, as entradas de um bloco são as atividades que geraram o dado representado pelo bloco, as saídas são as atividades que vão utilizar o objeto de dado. Os controles são as condições nas quais o objeto de dado será ativado. Os mecanismos são os elementos necessários ao armazenamento do dado.

8.2.2. Refinamento de Diagramas

De forma similar ao modelo do Diagrama de Fluxo de Dados, o nível de complexidade de um sistema sob análise pode conduzir a construção de um conjunto de diagramas SADT, onde um diagrama construído num dado nível representa um refinamento de um bloco de um diagrama de nível superior.

Com exceção do primeiro diagrama criado, os demais diagramas correspondem a detalhamentos de um bloco de diagrama de nível superior – o **bloco pai** o qual pertence ao **diagrama pai**. Os diagramas que representam os detalhamentos de blocos de um dado diagrama são denominados **diagramas filhos**.

A figura 5.15 apresenta um exemplo de refinamento de um bloco de um actigrama. No exemplo, o bloco **A1**, que é conectado ao sistema pelos fluxos de dados **I1** (entrada), **O1** e **O2** (saídas), **C1** (controle) e **m** (mecanismo), é refinado em três blocos, no caso **A11**, **A12** e **A13**. Um aspecto interessante de ser observado é o aparecimento, além dos fluxos de dados relativos aos blocos representados no novo actigrama, dos fluxos de dados originais do bloco **A1**, o que permite verificar a coerência do refinamento efetuado.

Como no caso dos DFDs, o refinamento dos diagramas não é infinito, mas composto de tantos níveis quantos se julguem necessários para representar precisamente o sistema. O processo de refinamento termina quando considera-se que o processo pode ser descrito precisamente numa outra linguagem (por exemplo, texto estruturado, diagramas de estado, etc...) e que o espaço ocupado pela descrição não ultrapasse uma única página.

8.2.3. Referência aos Diagramas

A referência de cada diagrama é localizada no canto esquerdo inferior da folha de cada diagrama, sendo que cada referência é definida como sendo um **nó**. O nó é especificado pela letra **A**, seguida de um número que indique o bloco pai que está sendo detalhado. Cada bloco num diagrama apresenta um número de referência escrito no seu canto superior direito, o qual o identifica de forma única para aquele diagrama. A figura 5.16 ilustra a forma de referência aos diagramas e aos blocos num diagrama.

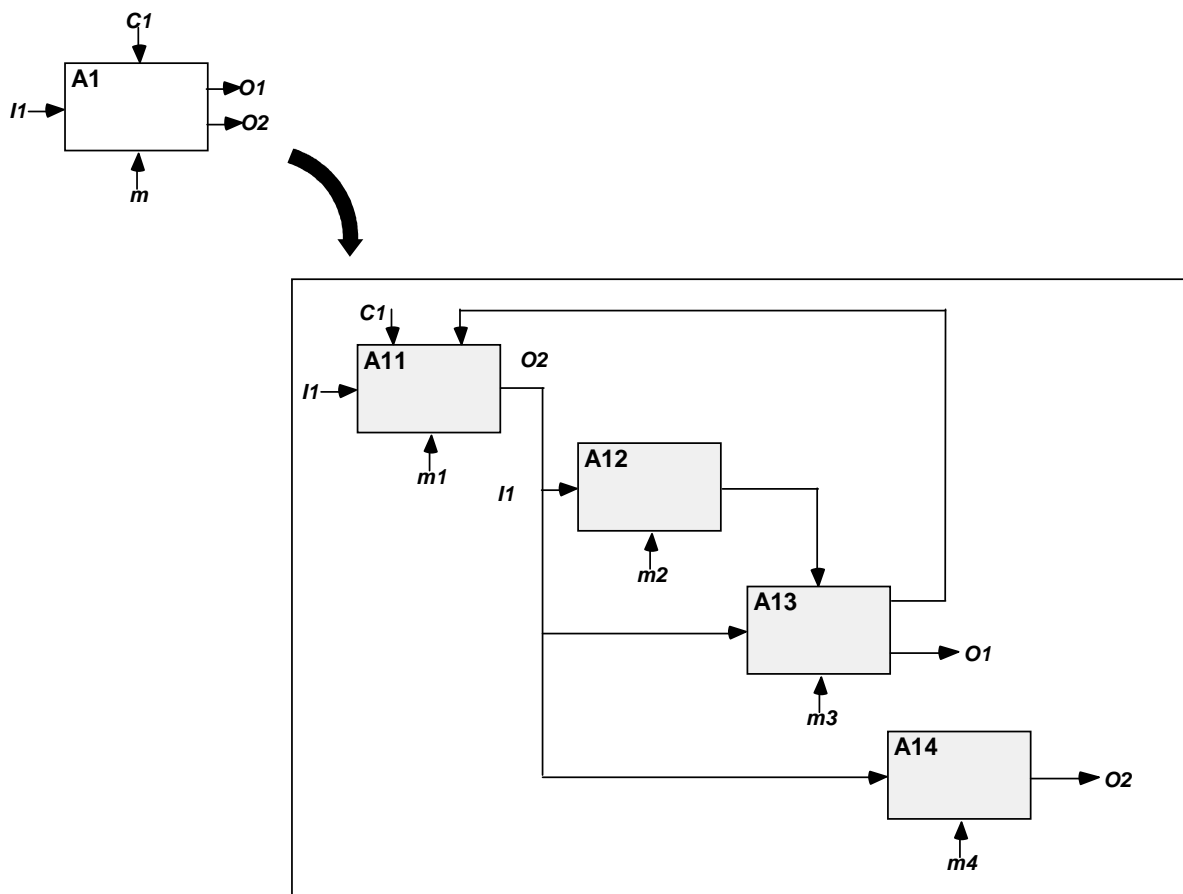


Figura 5.15 - Exemplo de refinamento de um bloco num actigrama.

A representação da estrutura hierárquica do modelo completo do sistema pode ser representada em duas formas distintas:

- um índice de nós, como mostra a figura 5.17;
- uma tabela de nós, como mostrado na figura 5.18.

8.2.4. Etiquetagem das Setas

Uma regra importante a ser seguida, a qual reflete fortemente na consistência do modelo em realização, é que, durante o refinamento de uma caixa, nenhuma adição ou supressão de informação deve ser feita, particularmente no que diz respeito às interfaces (entradas, saídas, controles e mecanismos) e ao nome do bloco pai.

As setas que apresentam as suas extremidades livres num dado diagrama correspondem às interfaces do bloco pai. Após a verificação de que todas as setas com extremidades livres correspondem à totalidade das interfaces do bloco pai, elas devem ser rotuladas com as letras **I**, **C**, **O** e **M** (designando uma **entrada**, um **controle**, uma **saída** ou um **mecanismo**, respectivamente), sendo que a letra deve ser seguida de um número que indique a sua posição geométrica relativa no bloco pai. A ordenação das setas é feita, num diagrama, de cima para baixo e da esquerda para a direita. Uma seta rotulada por **C3** corresponde ao terceiro controle do bloco pai.

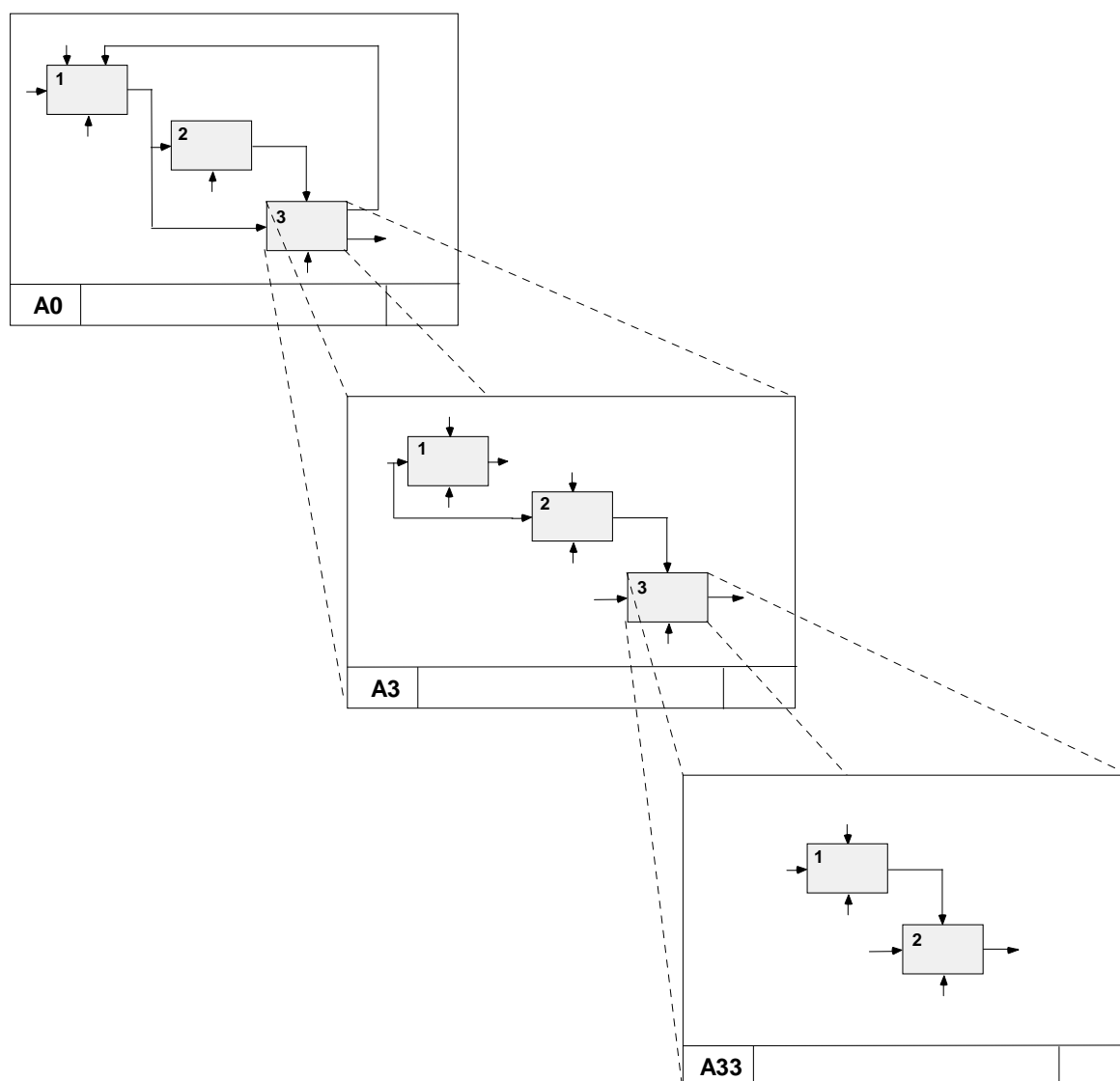


Figura 5.16 - Ilustração das referências aos diagramas e aos blocos.

Por outro lado, quando é realizado o refinamento de um dado bloco (bloco pai), as interfaces não serão necessariamente vistas da mesma forma que no bloco pai. Um exemplo disto é apresentado à figura 5.19, onde um controle (C2) no bloco pai corresponde a uma entrada de um dos blocos do diagrama filho.

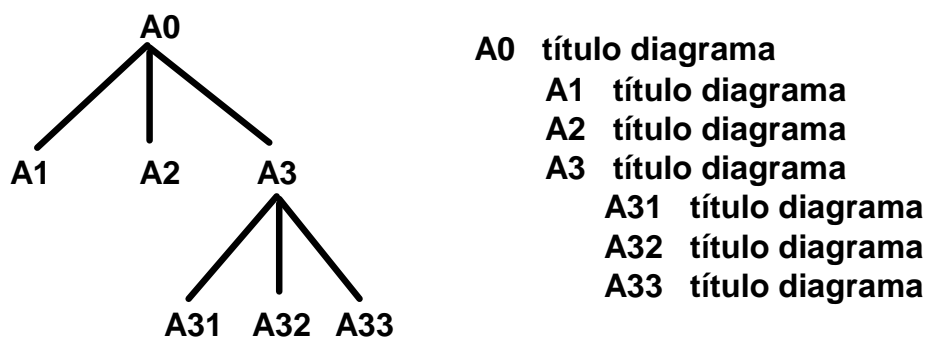


Figura 5.17 - Exemplo de um Índice de Nós.

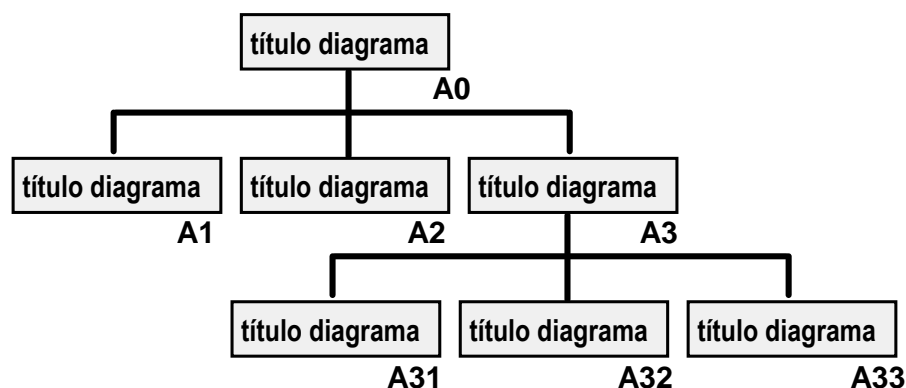


Figura 5.18 - Exemplo de uma Tabela de Nós.

8.2.5. A Análise de um Diagrama

A leitura de um diagrama deve ser efetuada segundo um **caminho principal** que parte da seta de entrada ou de controle não conectada mais importante e chega na seta de saída não conectada mais importante. As seguintes regras devem ser observada quando da análise de uma diagrama:

- ① examinar apenas os blocos do diagrama sob análise;
- ② retornar ao diagrama pai para observar como as setas I, C, O e M estão conectadas no bloco pai e qual a sua configuração no diagrama filho;
- ③ identificar a seta de entrada (ou de controle) e a seta de saída mais importantes do diagrama;
- ④ examinar o diagrama percorrendo o caminho principal;

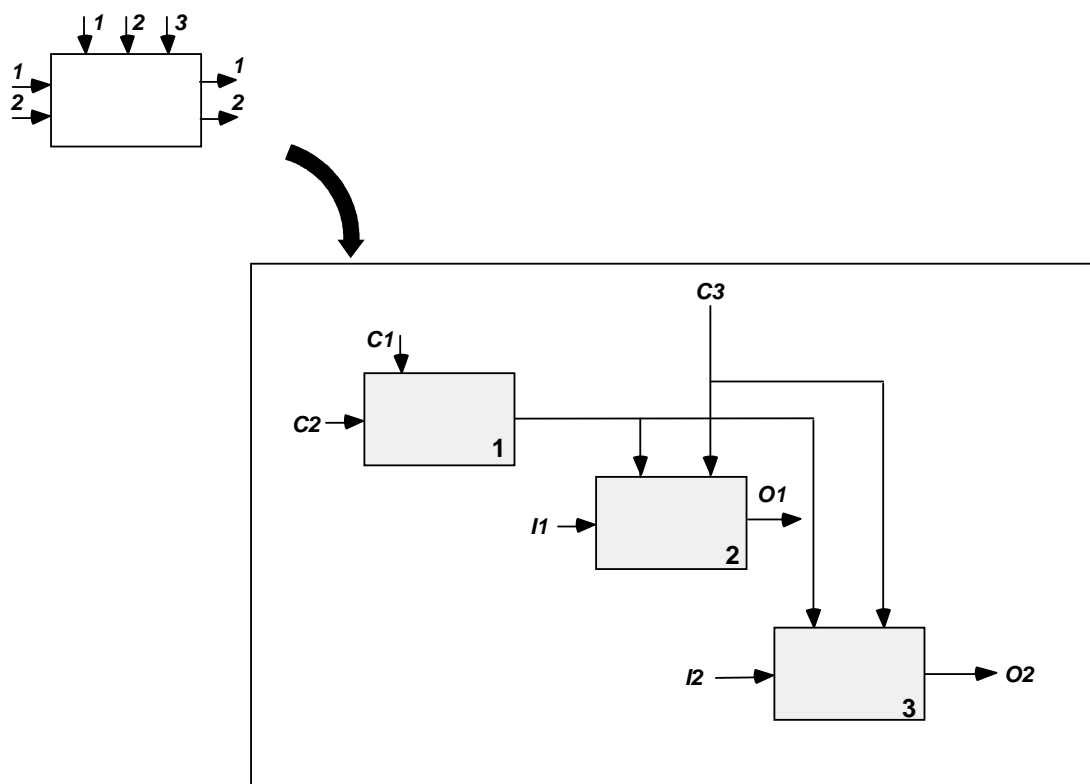


Figura 5.19 - Exemplo de etiquetagem das setas num diagrama.

- ⑤ percorrer o diagrama, observando como as diferentes setas intervêm em cada caixa e definindo os caminhos secundários;
- ⑥ ler as notas associadas ao diagrama para um melhor entendimento.

8.2.6. Normas de representação de um diagrama

Os diagramas deverão ser apresentados em folhas de formato normalizado, observando a configuração mostrada na figura 5.20.

Dentro de um diagrama (pai), um bloco (pai) é referenciado da seguinte forma (ver figura 5.21):

- o número do bloco é posicionado em baixo à direita de cada bloco;
- a referência do diagrama filho que representa o refinamento do bloco é posicionado abaixo do bloco (esta referência representa a página do documento de diagramas, onde o refinamento do bloco está apresentado; a ausência de referência implica na não existência de um refinamento daquele bloco).


 UFSC	Nome do Projeto:		ETAPAS	LEITOR	DATA
	Autor:		proposta		
	Data:	Versão:	aprovação		
			publicação		
Nó: Nº do nó	Título: TÍTULO DO BLOCO PAI				Referência: Ref. do Diag. filho

Figura 5.20 - Folha padrão para construção do diagrama.

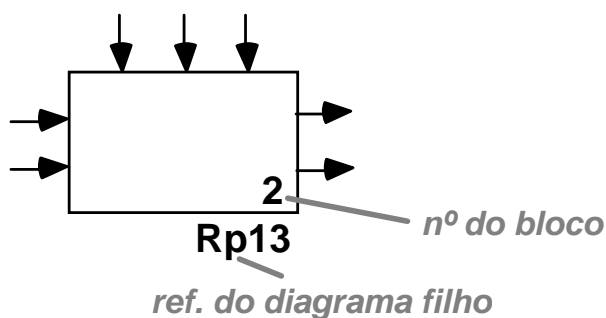


Figura 5.21 - Referências dos blocos num diagrama.