ANA-RE

um método para ANÁlise e especificação de REquisitos

Este exemplar corresponde a redação final da dissertação devidamente corrigida e defendida pelo Sr. CARLOS MIGUEL TOBAR TOLEDO e aprovada pela Comissão Julgadora.

Campinas, 1 de junho de 1989

Profa. Dra.

CLAUDIA MARIA BAUZER MEDEIROS

Dissertação apresentada ao INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E CIÊNCIA DA COMPUTAÇÃO, UNICAMP, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação

AGRADECIMENTOS

À Profa. Dra. Claudia Maria Bauzer Medeiros, pela orientação, paciência e esforço para que esta dissertação chegasse a ser o que é, meus mais sinceros agradecimentos.

À Direção do CTI que permitiu e incentivou a elaboração da dissertação, meu reconhecimento.

RESUMO

Esta dissertação introduz o ANA-RE, um novo método para elaboração de modelos analíticos de problemas, bem como apresenta as características do ambiente automatizado SAES (Sistema de Apoio à Especificação de Sistemas), que o suporta. Para mostrar a viabilidade desta automação, a dissertação descreve o desenvolvimento de um protótipo de um sistema de banco de dados que suporta o ANA-RE implementado em SMALLTALK [*].

O método ANA-RE é o resultado de estudos para a automação do SADT [+] e visa permitir o desenvolvimento de um ambiente automatizado que suporte e complemente a técnica e os instrumentos descritos na literatura sobre SADT. A motivação é oferecer um apoio para a elaboração de modelos que permitam a análise, definição e comunicação de requisitos de sistemas de software. ANA-RE, além de permitir a sua automação, aumenta o espectro de problemas passíveis de especificação formal comparativamente ao SADT.

Os objetivos principais da dissertação são:

1) A definição e formalização do método ANA-RE e sua comparação com o SADT, em termos de capacidade de especificação de requisitos de sistemas e facilidades para suporte automatizado.

2) A especificação parcial do ambiente automatizado SAES (suporte do ANA-RE), com a descrição de seus componentes funcionais e dos componentes de sua base de dados.

3) A implementação de um banco de dados que suporte a automação do ANA-RE.

[*] SMALLTALK é marca registrada da Xerox Corporation

[+] SADT System Analysis and Design Technique é marca registrada da SofTech Inc

ÍNDICE

1. DEFINIÇÕES E COMENTÁRIOS

Este capítulo tem por finalidade definir e comentar os conceitos relativos às fases e tarefas envolvidas no desenvolvimento de software, necessários à especificação do método ANA-RE e do ambiente SARES.

Os aspectos principais a serem considerados quando do início de um trabalho de desenvolvimento e o seu tratamento serão utilizados como referencial nos capítulos subseqüentes.

1.1. Ciclo de Vida

Um ciclo de vida para software, pelo convencional, organiza as atividades executadas durante o seu desenvolvimento e seu suporte [McDo85]. Existem ciclos de vida onde não são estabelecidas fases, sendo por isso, denominados ciclos de vida não convencionais. Estes se baseiam no fato de que um software e sua vida são um único processo de transformações, sem estágios. Os ciclos de vida não convencionais não são considerados neste texto.

Existem diversas divisões propostas para o ciclo de vida convencional de um software. A mais simples, em que apenas é considerada a fase de desenvolvimento, consiste em três atividades básicas: a identificação de uma necessidade, o desenvolvimento de uma solução e a implementação desta solução. A definição dada por Freeman [Free83] é mais complexa: nela, o processo de desenvolvimento se divide em oito estágios que formam o ciclo análise, especificação funcional, projeto da arquitetura, projeto detalhado, programação, teste, integração e evolução. Há ainda a proposição de McDonald e outros [McDo86] na qual, desconsiderando como é feito o processo de criação e evolução, os estágios são: concepção, definição, especificação, liberação, instalação, congelamento e aposentadoria.

Para efeito de nossa discussão, que se restringe à identificação e entendimento de um problema, é suficiente adotar o ciclo proposto em [SESu80]: análise e especificação de requisitos, projeto, implementação, validação e manutenção, que nada mais é do que o ciclo de manufatura.

1.2. Requisitos e Restrições

A fase inicial do desenvolvimento de um sistema, na qual ele é concebido, é o momento propício para a definição de suas propriedades. A essas características chamaremos **requisitos**. Após a identificação dos requisitos, estes devem ser especificados, originando um documento que se chama de **especificação de requisitos** ou simplesmente **especificações**. A **especificação de projeto** é o próximo passo, cuja análise está além do escopo do método e ambiente propostos na dissertação.

Roman, em sua excelente taxonomia [Roma85], define a especificação de requisitos e a especificação de projeto como as descrições da percepção do engenheiro frente a uma necessidade e seu entendimento da solução. A especificação de requisitos estabelece as características funcionais e de desempenho desejáveis a algum componente isolado de alguma realização, enquanto que a especificação de projeto descreve a estrutura interna real e o comportamento do componente. Enquanto a especificação de requisitos facilita o entendimento, a especificação de projeto apresenta as estruturas lógicas e físicas que implementam os requisitos.

Antes da especificação de requisitos existe a necessidade de um passo inicial relacionado à busca, coleta e entendimento dos requisitos a que chamaremos **análise de requisitos**, utilizando a definição de Leite em [Leit87].

Teichroew [Teic74] classifica requisitos em: **funcionais**, que descrevem o que o sistema faz; de **desempenho**, que descrevem quão bem (quantitativamente) são executadas as funções; e **outros**, que só podem ser descritos de forma qualitativa.

Segundo Yeh [YehR82], existem apenas dois tipos de requisitos: os **funcionais** e os **não-funcionais** (também conhecidos como **restrições**). Um requisito funcional apresenta a natureza da interação entre um componente e seu ambiente. Uma restrição limita os tipos de soluções que podem ser consideradas. Entre as restrições podem existir aquelas originadas por necessidades de interface, desempenho, operação, ciclo de vida e eventual metodologia adotada, econômicas e, finalmente, políticas.

1.3. Terminologia Básica e Notação

Neste texto são utilizados termos como *método*, metodologia etc, cujos significados são utilizados na bibliografia de maneiras diversas, conforme o autor, e muitas vezes de forma ambígua. Os termos que se seguem utilizam as definições de Boria [Bori87], do Projeto Ethos [ETH088] e de McDonald [McDo86]:

Ambiente: Ambiente automatizado, de agora em diante simplesmente denominado ambiente, é um conjunto integrado de ferramentas que auxiliam no processo de criação e evolução de software, segundo um determinado método ou metodologia.

Método: Método é um conjunto de diretivas e regras com vistas à seleção e aplicação de técnicas e instrumentos no desenvolvimento ou manutenção parcial ou total de software.

Técnica: Técnica é um conjunto de princípios para a execução de uma tarefa específica do processo de desenvolvimento ou de manutenção de software. Em geral as técnicas são suportadas por conjuntos de ferramentas.

Instrumento: Instrumento é uma convenção notacional aliada às respectivas operações, normalmente automatizáveis, que suportam atividades específicas do processo de desenvolvimento ou de manutenção de software (por exemplo, linguagens, diagramas, tabelas).

Ferramenta: Ferramenta é a implementação computacional de um instrumento em um determinado ambiente.

Metodologia: Metodologia é um conjunto de métodos coerente, consistente e completo, juntamente com princípios e orientações de como, onde, quando e porque devem ser empregados no desenvolvimento e manutenção de um sistema de software. Uma metodologia deve ser uma filosofia geral aplicável a todo o ciclo de vida do software.

Além dos termos definidos anteriormente, este trabalho utiliza a seguinte terminologia:

número seqüencial: designa um número inteiro sequencial a partir de 1.

contexto: indica o assunto delimitado em uma caixa de um diagrama SADT c. ANA-RE, passível de detalhamento.

modelo: representação que especifica (alguns dos) atributos de uma entidade ou problema.

atributo: uma característica ou propriedade da entidade ou problema.

modelagem: atividade relacionada à representação, formalização e organização das informações de um problema.

multi-modelo: conjunto de modelos especificados via SADT ou ANA-RE. Normalmente refere-se ao conjunto de modelos que compõem a análise de um determinado assunto, cada qual examinado segundo uma orientação.

1.4. O Papel da Especificação de Requisitos durante o Desenvolvimento

Uma especificação de requistos deve exercer várias funções durante o desenvolvimento de software, entre as quais destacamos:

a. descrever o que o sistema faz, de acordo com os requisitos e necessidades do usuário, permitindo-lhe fazer revisões e acompanhar o que está sendo feito;

b. descrever o que a equipe deve produzir para permitir a gerência do trabalho de desenvolvimento [Boeh77];

c. ser uma referência completa a respeito do comportamento externo do sistema, evitando que os programadores tomem decisões a respeito do que é melhor para o usuário [Parn82] e reduzindo problemas de coordenação entre programadores durante a implementação [Rama77];

d. ser o elemento contra o qual são feitas as validações da especificação de projeto [Wass83];

e. ser a base necessária (mas não suficiente) para a geração de boas estimativas a respeito da quantidade de trabalho e de recursos necessários para o desenvolvimento [Parn82];

f. ser um seguro contra a rotatividade de pessoal envolvido no desenvolvimento [Parn82];

g. ser o elemento básico (referencial) para o desenvolvimento do plano de testes e aceitação [Parn82].

Segundo Ramamoorthy [Rama77], três tipos de problemas respondem por 85% do total de erros encontrados durante o desenvolvimento de software de qualquer porte: especificações incorretas, especificações inconsistentes e incompatíveis, e especificações dúbias.

1.5. O Produto da Fase de Análise e Especificação de Requisitos

Segundo Parnas [Parn82], o documento resultante da fase de Análise e Especificação de Requisitos deve conter tudo o que é necessário saber para desenvolver um software que seja aceitável pelo cliente, e nada além disto.

Ao documento gerado pela fase de Análise e Especificação de Requisitos chamaremos de Especificação de Requisitos do Sistema (**ERS**). A forma final dos requisitos no ERS deve levar em consideração o papel dos requisitos durante o ciclo de vida do sistema. Segundo o Padrão ANSI/IEEE [IEEE84], um bom ERS caracteriza-se por ser: não ambíguo, completo, verificável, consistente, modificável, navegável ('traceable') e útil à operação e manutenção. Roman [Roma85] acrescenta que o ERS deve ser conceitualmente limpo, isto é, simples, claro e compreensível, além de ser testável e executável. Para uma definição clara de cada característica recomendamos a leitura destas referências.

Parnas [Parn82] ressalta que o ERS deve ser organizado como um documento de referência ao invés de uma narrativa sobre o sistema onde os assuntos incompletos devem estar indicados explicitamente.

O sucesso de um método de especificação de requisitos depende de vários aspectos como por exemplo: formalismo, capacidade de poder ser suportado automaticamente, capacidade de poder ser tolerante à incompletude temporária e de ser adaptável a modificações.

1.6. Organização da Dissertação

Este capítulo apresentou a terminologia utilizada na dissertação. O segundo capítulo apresenta uma revisão bibliográfica. O terceiro capítulo é uma descrição do método SADT. No capítulo quatro analisa-se o trabalho de automatizar o método SADT. O capítulo cinco é uma apresentação informal do método ANA-RE e do ambiente SAES. No capítulo seis discute-se e apresenta-se as modificações propostas ao SADT e que oricinam o ANA-RE. O capítulo sete descreve a modelagem e implementação de um Banco de Dados para suporte do ANA-RE. Por último são apresentadas as conclusões e as perspectivas futuras para o assunto enfocado.

2. REVISÃO BIBLIOGRÁFICA

Revisar a bibliografia relacionada a este trabalho é uma tarefa trabalhosa, uma vez que o assunto tem despertado grande interesse na comunidade de informática e esta, por sua vez, tem adotado diferentes abordagens para seu estudo e aplicação. Este capítulo discute trabalhos relacionados à fase de análise e especificação de requisitos, do ponto de vista de metodologias e ambientes.

Segundo Ceri [Ceri86], as conferências internacionais IFIP de 1982 e 1983 estimularam o interesse por metodologias na área de RCA ('Requirements Collection and Analysis') a tal ponto que uma das revisões apresentadas mencionava 50 'metodologias'. Estas abordagens vão desde a aplicação a ciclos de vida convencionais até ciclos de vida não convencionais, passando por banco de dados, inteligência artificial e automação do processo de desenvolvimento a partir da formalização das especificações. Para dar uma idéia da proliferação mencionada, citamos as siglas de linguagens, técnicas, métodos ou metodologias citadas em [Ceri86]: ACM/PCM, ARDI, BISAD, BSP, CIAM, DATAID, D2S2, DADES, EDM, ETHICS, HIPO, IML, ISAC, ISDOS, ISSM, MERISE, NIAM, OAM, OSIRIS, REMORA, SADT, SDLA, SDP, SYSDOC e USE.

Gehani e McGettrick [Geha86] enfocam algumas técnicas para a especificação formal de requisitos. Seu livro é uma coletânea de 21 trabalhos publicados sobre especificação e dentre eles destaca-se o suporte a linguagens de especificação.

O livro de Blank e Krijger [Blan83] apresenta uma comparação de técnicas e métodos com relação à cobertura de um ciclo de vida convencional. Esta comparação se faz através de uma matriz de avaliação onde são cruzadas as etapas do ciclo de vida com quatro aspectos que, segundo os autores, devem ser cobertos por qualquer técnica ou método: filosofia, procedimento de trabalho, método de documentação e método de gerenciamento. Os métodos comparados são: BSP, SADT, ISAC, MOS, SASO, NIAM, BC, Warnier-Orr, Jackson e PSL/PSA.

Rocha [Roch87] cita os seguintes métodos e metodologias relacionados com análise e projeto de sistemas: SAMM, PSL/PSA, SSA, HDM, SREM, SDS, USE e 'Software Factory'.

Os trabalhos sobre análise e especificação de requisitos são classificados de várias maneiras. Por exemplo, a classificação dada por Freeman e Wasserman [Free84] especifica:

- técnicas de análise, onde se incluem SADT, SSA e ISAC;
- métodos de especificação, onde se incluem HIPO, JSD, PSL/PSA e SREM;

- técnicas para projeto de arquitetura com ênfase em controle, onde se incluem SD (Stevens), SARA, CSP (Hoare) e DREAM;

- técnicas para projeto de arquitetura com ênfase em dados, onde se incluem abstração de dados, orientação por objetos e BASIS;

- técnicas para detalhamento de projeto, onde se incluem PDL, ADA Design Methodology e diagramas Nassi-Shneiderman; e

- metodologias para desenvolvimento de software, onde se incluem a metodologia ADA, SDS e USE.

A classificação de Leite [Leit87] agrupa as referências segundo trabalhos em:

- engenharia de software, onde se inclui os trabalhos em que a idéia de ciclo de vida convencional é predominante e onde há uma divisão entre especificação e projeto (53 referências);

- novos paradigmas em engenharia de software, onde se inclui os trabalhos em que o ciclo de vida convencional é criticado juntamente com a separação entre especificação e projeto; três idéias básicas influenciam os novos paradigmas: protótipos, técnicas de inteligência artificial e o inter-relacionamento entre especificação e projeto (30 referências);

- engenharia de software e gerenciamento da informação, onde se inclui os trabalhos em que são apresentadas idéias de engenharia de software orientadas ao gerenciamento da informação (10 referências);

- banco de dados, onde se inclui os trabalhos em que a análise de requisitos está aplicada, através da utilização de modelagem, ao desenvolvimento de aplicativos na área de banco de dados (9 referências);

- inteligência artificial, onde se inclui os trabalhos relacionados com representação e aquisição de conhecimento (12 referências); e

- gerência da informação, onde se inclui trabalhos em que a análise de sistemas tradicional é o assunto predominante. O enfoque se diferencia da análise de requisitos nos aspectos de interação com o usuário e busca, coleta e entendimento de requisitos (9 referências).

Outro enfoque encontrado ([Rama77], [Press82] e [Shoo83]) para a classificação é o de adotar apenas dois agrupamentos: aquele em que as técnicas e princípios são orientados pelo fluxo de controle e aquele onde são orientados pelo fluxo de dados. Em [Rama77], por exemplo, existe referência a 68 trabalhos.

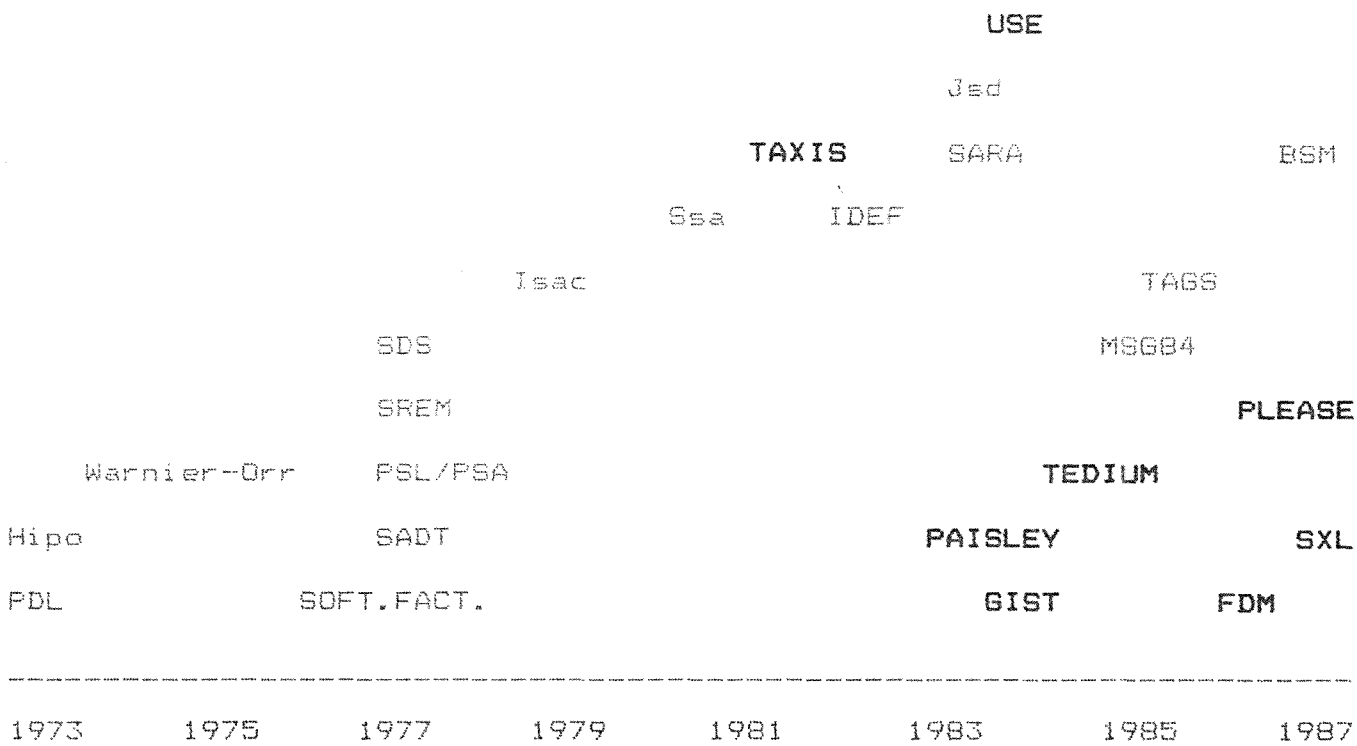
A classificação adotada nesta revisão bibliográfica baseia-se na classificação de Leite [Leit87], sem incluir os agrupamentos relativos a banco de dados e inteligência artificial. Assim sendo, a organização da bibliografia relativa a trabalhos na área de especificação de requisitos está dividida nos seguintes agrupamentos, de acordo com sua ênfase:

. suporte a ciclos de vida convencionais (seção 2.1);

. análise e gerenciamento da informação (seção 2.2); e

. novos paradigmas (seção 2.3).

O diagrama a seguir relaciona as siglas das linguagens, técnicas, métodos ou metodologias revisadas em relação ao ano de sua criação ou o ano em que o trabalho correspondente foi publicado, apresentado no eixo horizontal.



2.1. Ciclo de Vida Convencional

Neste agrupamento são discutidos: PDL (1973), Software Factory (1976), PSL/PSA (1977), SREM (1977), SDS (1977), IDEF (1981), SARA (1981), MSG84 (1984) e TAGS (1985). O SADT (1977) que também se encaixa nesta classificação, é desenvolvido no capítulo 3, por ser a base do ANA-RE.

PDL - Program Design Language

PDL é uma linguagem para a especificação da estrutura do projeto (projeto detalhado) na forma 'top-down'. PDL foi inicialmente utilizada em 1973 na firma Caine, Faber & Gordon Inc. com ótimos resultados de aumento de produtividade [Cain75].

SOFTWARE FACTORY

Bratman e Court [Brat75] [Brat77] definem SOFTARE FACTORY como um conjunto integrado de ferramentas que suportam os conceitos de programação estruturada, desenvolvimento 'top-down', produção de bibliotecas e modularização. A este conjunto de ferramentas soma-se o Manual de Desenvolvimento de software, onde estão descritos os procedimentos, princípios de organização e normas para o desenvolvimento de software. Trata-se de uma metodologia cujas ferramentas foram desenvolvidas para suportar as fases de projeto e implementação. É um projeto desenvolvido na SDC ('System Development Corporation') que foi utilizado como modelo base, principalmente pelos japoneses, para pesquisa e aumento de produtividade na área da engenharia de software.

As ferramentas que compõem a SOFTWARE FACTORY são:

- FACE ('Factory Access and Control Executive') para controle e 'interface' entre a base de dados e os outros processos, incluindo a linguagem de comandos;
- IMPACT ('Integrated Management, Project Analysis and Control Technique') para o gerenciamento de pontos de controle, tarefas, recursos, componentes do sistema e seus relacionamentos;
- AUTODOC ('Automatic Documentation') ferramenta para produção de documentação de programas e sistemas;
- PATH ('Program Analysis and Test Host') ferramenta para análise do fluxo dos programas-fonte;
- TCG ('Test Case Generator') ferramenta para geração de testes de programas;
- TOPS ('Top-down System Developer') ferramenta para a verificação e desenvolvimento de subsistemas e programas.

PSL/PSA - Problem Statement Language / Problem Statement Analyzer

Teichroew [Teic76] [Teic77] apresenta o PSL/PSA como uma técnica estruturada para análise e documentação de requisitos e para a elaboração de especificações funcionais. Trata-se de um método cuja técnica base é suportada por uma ferramenta para auxílio a análise, especificação e documentação de requisitos. Foi desenvolvido na Universidade de Michigan como parte do projeto ISDOS ('Information System Design and Optimization System').

A linguagem PSL é utilizada para a especificação do sistema a ser desenvolvido. Um sistema é descrito em termos de relacionamentos entre os vários objetos que o compõem: fluxos de entrada/saída, estrutura do sistema e dos dados, derivação dos dados, tamanho, volume, dinâmica do sistema e gerência do projeto. Sua representação é informal e complexa.

As informações do sistema são verificadas sintaticamente através do PSA, que gera referências cruzadas das mesmas. Após a análise sintática, as informações são mantidas em um banco de dados através de relações binárias, usadas para geração de relatórios sobre modificações, referências, sumários e análises.

SREM - Software Requirements Engineering Methodology

Bell [Bell77] e Alford [Alfo77] descrevem SREM como sendo um sistema de apoio à manutenção e análise de requisitos. Trata-se de um método para auxílio à análise, especificação e documentação de requisitos, cuja base é uma linguagem de especificação e um conjunto de ferramentas para análise das especificações. Foi desenvolvido pela TRW Defence and Space Systems Group como parte do programa Software Requirements Engineering Program para sistemas de defesa BDM ('Ballistic Missile Defense') e é voltado a problemas de controle de processos de tempo real.

SREM é composto por uma linguagem para especificação de requisitos, a RSL ('Requirements Statement Language'). As restrições são representadas por R-nets ('Requirements Networks') onde são identificados eventos, interfaces externas, pontos de validação (utilizados na simulação das especificações), unidades funcionais, sub-redes, bancos de dados, operações lógicas e pontos de término.

As R-nets, juntamente com o relacionamento hierárquico dos dados, informações para 'navegação' e informações gerenciais, constituem uma especificação completa que é armazenada em um banco de dados.

O conjunto de ferramentas para análise das especificações, REVS ('Requirements Engineering and Validation System'), é composto por três tipos de ferramentas, além do tradutor de R-nets em código PASCAL: aquelas destinadas à manipulação de R-nets através de interfaces gráficas, aquelas destinadas à análise estática da estrutura das R-nets, e aquelas destinadas à simulação das especificações.

SDS - Software Development System

Davis [Davi77] define SDS como sendo uma metodologia para o desenvolvimento de software específico para os sistemas de defesa de mísseis, que são grandes sistemas de tempo real cujo requisito básico é a confiabilidade. Trata-se de uma metodologia orientada à especificação de requisitos que também cobre as fases de projeto, implementação e teste. O trabalho para o desenvolvimento do SDS foi feito por empresas privadas junto ao BMDATC ('Ballistic Missile Defense Advanced Technology Center').

Nas fases do ciclo de vida suportado pela metodologia são utilizados: especificação de requisitos através de redes de Petri [Ager79] [Pete77] e SREM [Bell79], e projeto de processos através da linguagem PLDL.

IDEF - Icam DEFinition Method

IDEF0 é utilizado para a modelagem funcional de sistemas e ambientes nas atividades de análise e comunicação dentro do processo de manufatura no programa ICAM ('Integrated Computer Aided Manufacturing') da Força Aérea Americana [SofT81]. IDEF0 é a identificação dada a um subconjunto do método SADT e é o veículo de comunicação através do qual as companhias aeroespaciais concessionárias da USAF devem desenvolver subsistemas genéricos que possam ser utilizados em um grande número de companhias de forma a dar suporte as funções comuns de manufatura.

IDEF0 é parte do método IDEF que compreende mais dois métodos de modelamento: o IDEF1, utilizado para a modelagem de informações, representando as estruturas de informação necessárias para o suporte do modelo funcional, e o IDEF2, utilizado para a modelagem dinâmica, que compreende a representação do comportamento variável (em relação ao tempo) de funções, informações e recursos. O IDEF0 pode ser usado para modelar uma ampla gama de sistemas que podem incluir hardware, software e 'peopleware'.

Para sistemas novos, o IDEF0 pode ser usado para a especificação de requisitos e funções, além do projeto de uma implementação. Para sistemas existentes, o IDEF0 pode ser usado para análise das funções executadas pelo sistema e para o registro dos mecanismos utilizados nessa execução.

SARA - Systems ARchitect's Apprentice

SARA é um ambiente para suporte ao projeto de sistemas concorrentes no que tange à modelagem, análise e simulação [Estr86]. SARA era um sistema de programação baseado em uma metodologia baseada em requisitos, que oferecia várias ferramentas e um método para a modelagem de sistemas concorrentes, tanto de hardware como de software [Cano77]. Trata-se de um método, centrado na fase de projeto, a partir do qual foi desenvolvido um ambiente que auxilia a verificação de projetos de sistemas concorrentes e permite a simulação dos mesmos. SARA foi desenvolvido na UCLA e está baseado em GMB (Graph Model of Behavior) [Rama77].

A construção de modelos em SARA é dividida em três domínios: controle, dado e interpretação. SARA apresenta também uma interface gráfica para manipulação de detalhamentos.

MSG84

MSG84 é uma linguagem formal para a especificação de modelos nas fases de análise de requisitos e projeto. A semântica da notação empregada é definida em termos do formalismo de 'actors', baseado no paradigma de objetos (passagem de mensagens) [Berz85].

MSG84 apresenta tantos comandos e detalhes que pode ser considerada uma linguagem complexa. É apoiada por uma ferramenta que apresenta as especificações em forma gráfica para possibilitar a visualização [Leit87].

TAGS

TAGS é uma metodologia para a construção de software que permite a decomposição e detalhamento a partir da especificação até a geração do código. Além disto, afirma que a linguagem IORL claramente é uma combinação de SADT e fluxo de dados, apesar de não haver referência ao SADT. Não se trata de uma metodologia mas sim de um conjunto de métodos.

TAGS compõe-se de três partes: IORL ('Input/Output Requirements Language'), uma ferramenta para verificação de sintaxe e simulação da linguagem ADA e os métodos.

2.2. Análise e Gerenciamento da Informação

Neste agrupamento estão colocados HIPO (1973), Warnier-Orr (1974), ISAC (1978), ESA (1980), JSD (1982) e BSM (1986).

HIPO - Hierarhy plus Input-Process-Output

HIPO [Stay76] [HIPO73] é um método de documentação desenvolvido na IBM que provê uma linguagem gráfica para a especificação hierárquica de sistemas comerciais. Baseia-se na explicitação do relacionamento entre processos, subprocessos, fluxos de entrada e saída.

Warnier-Orr

É um método baseado na análise das saídas desejadas no sistema, pressupondo a identificação prévia dessas saídas. A análise é feita de maneira a identificar a estrutura dos dados, os elementos de dados independentes e, por conseqüência, as funções e entradas necessárias para sua geração. Estas funções necessitam ser especificadas juntamente com eventuais funções responsáveis pelas transações sobre uma base de dados lógica. São utilizadas técnicas para a documentação da análise através de diagramas de estrutura [Blan83].

ISAC - Information Systems work and Analysis of Changes

Lunderberg [Lund80] define ISAC como sendo uma metodologia de trabalho cujo objetivo é prover os usuários com modelos para a análise de informação, que posteriormente são utilizados como especificações para a etapa de projeto do sistema. Trata-se de um método com diversas técnicas para o desenvolvimento do documento de requisitos. ISAC foi desenvolvido e pesquisado no Instituto Real de Tecnologia e na Universidade de Estocolmo. Na bibliografia estudada é apontada a falta de ferramentas para suporte automatizado do método.

ISAC utiliza diferentes técnicas de notação, a saber: A-graphs ('activity') que são complementados com texto e tabelas de propriedades das atividades; I-graphs ('information'); C-graphs ('componente'); D-graphs ('data'); P-structure ('program') e E-graphs ('equipment').

Os A-graphs têm descrição gráfica simples e apresentam num mesmo gráfico o fluxo real (material, pessoal, etc.) e o fluxo de informações.

O método ISAC é dividido em três partes: a análise de mudanças, os estudos das atividades e a análise de informação. Provê uma organização com relação à documentação e à transição entre os diferentes passos (por exemplo, entre as especificações em A-graphs e I-graphs). Por ser orientado para o usuário, nele o analista de sistemas atua apenas como um catalisador de idéias e resultados.

SSA - Structured Systems Analysis

Gane [Gane79] [Gane79b] considera o SSA um processo prático que obteve sucesso na definição de requisitos para sistemas de processamento comercial, principalmente no que se refere à definição de requisitos para os dados. O processo orienta a definição dos subsistemas a serem projetados, a partir das especificações dos requisitos.

O conceito fundamental é a construção de um modelo através de uma linguagem gráfica, DFD ('Data Flow Diagram'), e da orientação de análise 'top-down' de detalhamento dos subsistemas até um nível adequado.

Além da linguagem gráfica, existe um dicionário de dados, onde devem estar detalhados todos os dados especificados nos diagramas DFD, e diagramas para descrição da estrutura lógica dos dados, tendo em vista a forma como serão recuperados. O SSA fornece orientação sobre como os diagramas DFD podem conduzir à definição dos subsistemas a serem implementados.

JSD - Jackson System Development

Cameron [Came82] [Came86] define JSD como um método para a geração de especificações formais de sistemas através de um conjunto de passos distintos. Nos passos iniciais as características externas relevantes são modeladas, para nos passos finais ser obtida a especificação funcional do sistema, cuja implementação pode ser feita via transformações automatizáveis. A especificação do sistema consiste basicamente em uma rede distribuída de processos seqüenciais.

São três os principais passos do método: modelagem, quando as estruturas dos dados (entidades externas) são selecionadas e definidas, definição da rede, quando são definidas as funções e processos através de uma rede onde se estabelecem os relacionamentos entre as entidades externas, e implementação.

BSM - Box Structure Methodology

Mills, Linger e Hevner [Mill86] [Mann87] estabeleceram uma metodologia baseada em quatro princípios: Transparência Referencial, Completude do Sistema, Descentralização das Informações e Reutilização Modular.

O processo de especificação está baseado em três estruturas: caixa preta, máquina de estados e caixa aberta. Todos os detalhes do sistema podem ser representados graficamente através dessas estruturas.

A caixa preta é a abstração de um sistema de informação que a partir de estímulos produz respostas. As caixas pretas podem ser expandidas em máquinas de estado que representam a primeira visão da estrutura interna do sistema de informação. A caixa aberta provê uma visão interna de uma caixa preta dentro da estrutura da máquina de estados, utilizando uma das quatro estruturas de controle: sequência, alternância, iteração e concorrência. A descrição dessas estruturas pode ser feita através da linguagem BDL ('Box Description Language').

2.3. Novos Paradigmas

Neste agrupamento se enquadram TAXIS (1980), USE (1982), GIST (1982), PAISLEY (1982), TEIUM (1983), FDM (1985), SXL (1986) e PLEASE (1986).

TAXIS

Mylopoulos [Mylo80] define TAXIS como uma linguagem para o projeto de sistemas de informação interativos (por exemplo, reserva de passagens). TAXIS se baseia em facilidades para a especificação de bases de dados relacionais, restrições de integridade e mecanismos para o tratamento de exceções. A linguagem para a especificação da base de dados utiliza os conceitos de classe, propriedade (característica) e generalização existentes nas linguagens orientadas para objetos.

USE - User Software Engineering

USE desenvolvido por Wasserman [Wass82] [Wass84] [Wass86] é uma metodologia para o desenvolvimento de sistemas interativos, que tem como destaque o envolvimento efetivo dos usuários nas primeiras etapas do desenvolvimento. Tal metodologia baseia-se no desenvolvimento de protótipos rápidos e na utilização de um conjunto de ferramentas gráficas. A análise de requisitos utiliza modelos não formais especificados através de SSA (vide 2.2), além de modelos semânticos para as informações. Os primeiros passos para o desenvolvimento da metodologia foram dados em 1975 na Universidade da Califórnia, São Francisco.

GIST

Balzer [Balz82] [Balz85] define GIST como uma linguagem formal para a especificação de requisitos que podem ser transformados em código executável ou ser simbolicamente avaliada. GIST se baseia no modelo STMB ('State Transition Model of Behavior') além de suportar hereditariedade (paradigma de objetos), definição de restrições globais, referências a estados que ocorreram previamente e chamadas a banco de dados. Foi desenvolvida na University of Southern California.

Uma especificação em GIST define o seguinte conjunto:

- declarações estruturais que definem o espaço para os estados potenciais do sistema,
- regras de estímulo/resposta que definem as situações em que são iniciadas atividades, e
- restrições que limitam o espaço de possíveis estados definidos pelas duas categorias anteriores.

PAISLEY

Zave [Zave86] define PAISLEY como uma linguagem executável para especificação de sistemas utilizando idéias de linguagens funcionais e processos assíncronos. PAISLEY tem as seguintes características: especificação de paralelismo síncrono e assíncrono, encapsulamento de todas as operações, possibilidade de executar as especificações mesmo quando incompletas. É possível a simulação de restrições de tempo, além da detecção de várias formas de inconsistências.

Especificações em PAISLEY são ditas operacionais, significando serem independentes de implementação, ao invés de não construtivas (ou seja, especificações de quais propriedades uma solução deve ter).

TEDIUM

Blum [Blum87] define TEDIUM como um ambiente para especificação de sistemas de informação interativos baseado na geração de código MUMPS diretamente de especificações formais. É um gerador de aplicativos, baseado no paradigma da automação total do ciclo de vida.

TEDIUM é muito mais que uma linguagem executável para especificação, é uma meta-linguagem de alto nível para MUMPS [Leit87].

FDM - Formal Develop Method

Berry [Berr87] define FDM como a base metodológica da linguagem formal para especificação INA JO. FDM é um método para o projeto de sistemas baseado em uma seqüência de refinamentos que vão desde os requisitos informais, transformados em requisitos formais, transformados nas especificações de uma implementação.

INA JO trata um sistema e seus dados como uma máquina de estados, que se compõe de um conjunto de variáveis capazes de armazenar um valor. Uma máquina pode ser descrita por uma especificação onde os estados são uma associação das variáveis com valores correntes [Wing89].

SXL

Sluizer e Lee [Slui87] definem SXL como uma linguagem executável não baseada em procedimentos mas sim em lógica. Uma especificação em SXL é um modelo, em alto nível, da função ou comportamento de um sistema.

Um modelo contém instâncias de entidades com seus atributos e relacionamentos. Assertivas podem ser formuladas através de predicados. As transições são definidas através de pré-condições e pós-condições.

PLEASE

Terwilliger e Campbell [Terw86] definem PLEASE como uma linguagem executável para especificação de sistemas que suporta refinamentos incrementais. É parte de um ambiente que suporta todo o processo de desenvolvimento, pesquisado através do projeto SAGA na University of Illinois at Urbana-Champaign.

Inicialmente são especificados componentes de software utilizando linguagens de programação convencionais e lógica de predicados. Estes componentes abstratos são refinados através de PLEASE, de forma a sempre satisfazer as especificações iniciais. As especificações em PLEASE podem ser transformadas em protótipos baseados em Prolog que executam pré-condições e pós-condições através das quais são definidas as funções.

2.4. Então porque SADT?

A dissertação propõe um método para análise e especificação de requisitos baseado em modificações ao SADT. Esta seção apresenta argumentos para justificar esta escolha.

Os primeiros estágios de um desenvolvimento têm como propósito o entendimento do problema e a comunicação desse entendimento [Free80].

Para Booch [Booc84], o processo de solução de problemas sempre é iniciado pela definição do problema, que é uma tarefa a ser desenvolvida da mesma forma que na análise de sistemas convencional. Aponta o SADT e o DFD ('Data Flow Diagrams') como apropriados para esta tarefa, pois permitem que a estrutura do problema seja entendida.

Para Ceri [Ceri86], o uso massivo de métodos, técnicas e linguagens formais está abaixo do que se poderia esperar, devido à resistência dos possíveis usuários. Considera que as técnicas para especificação de requisitos mais utilizadas são SSA, SADT e ISAC, ressaltando a importância da modelagem associada a uma representação gráfica.

SADT é o método mais citado na pesquisa de [Leit87], que envolve mais de 120 referências. As características do SADT consideradas relevantes por Leite são a redundância existente na especificação com enfoque em dados e em atividades, a organização hierárquica, a utilização de pontos de vista como base para diferentes modelos de um mesmo problema e o ciclo autor-comentarista para verificação dos modelos desenvolvidos.

Em uma comparação de 'técnicas' de análise de requisitos Yadav e outros [Yada89] comparam DFD, SADT e IDF, concluindo que o aprendizado do SADT é menos penoso que o dos outros dois, em contradição com Blank e Krijger [Blan83], que, por sua vez criticam, a necessidade de um esforço considerável para a aquisição do conhecimento SADT, tanto por parte da equipe de desenvolvimento como por parte dos usuários.

Booch considera o SADT como um método para definir um problema e comunicar essa definição. Em RML ('Requirement Modelling Language') [Gree84] [Borg85] utiliza-se o SADT para o desenvolvimento de um primeiro modelo estrutural. Em [Blan83] cogita-se a complementação do PSL/PSA através do SADT e vice-versa.

Nota-se, na maioria das referências revisadas, a necessidade de um passo inicial antes da especificação dos requisitos, o que em [Leit87] é chamado Análise de Requisitos. Este passo se relaciona intimamente com busca, coleta e entendimento de requisitos e envolve o estabelecimento de fatos inicialmente incompletos e inconsistentes. Uma forma para, de forma ordenada e sistemática, se estabelecer este primeiro modelo do problema é através da utilização do SADT.

O SADT, no entanto, apesar de sua popularidade, apresenta pontos fracos. Falta um suporte para experimentação (elaboração de protótipos), que permita a validação das especificações antes do avanço para outras etapas do desenvolvimento, e um suporte para a coleta de informações, que poderia ser levado a cabo, por exemplo, através de entrevistas, questionários, 'checklists' ou reutilização [Leit87].

Colter [Colt84] critica o SADT por este não cobrir de maneira satisfatória os seguintes tópicos: estrutura de controle, detalhes de procedimentos, detalhes de entrada e saída e de comunicação com o usuário.

Nos diagramas SADT é difícil a inserção de nuances na representação dos dados; o desenho dos diagramas SADT requer bastante esforço; a manutenção atualizada da documentação SADT demanda tempo e esforço de uma equipe bem treinada; os procedimentos de comunicação escrita do SADT requerem tempo e diminuem a flexibilidade da comunicação informal; e, na prática, apenas parte dos 40 componentes, ditos gráficos, da linguagem do SADT são utilizados.

Fica então a questão de, em se utilizando o SADT como método para as primeiras modelagens de problemas visando seu entendimento e a comunicação desse entendimento, qual a maneira de contornar os problemas por ele apresentados?

O método ANA-RE é uma tentativa de solução para alguns destes problemas.

3. SADT

O SADT, segundo Ross [Ross85], um de seus idealizadores, é um método [*] completo para tratar complexidade através de uma disciplina organizada de ação e pensamento, destinada a equipes, e que é acompanhado por uma documentação gráfica e textual concisa, completa e legível.

A idéia por trás do SADT é a da utilização de esboços ('blueprints') em uma operação de manufatura, onde um produto é criado e os passos para a sua montagem e produção são executados mentalmente por projetistas e engenheiros muito antes que as especificações e esboços sejam enviados ao departamento de produção [Ross77].

3.1. O que é o SADT

O SADT consiste de duas partes: uma linguagem de diagramação chamada 'Structured Analysis' (SA) e uma técnica de projeto ('Design Technique' ou DT).

Entende-se que o SADT é composto por três partes: uma linguagem gráfica para diagramação, uma técnica para análise estruturada (a SA) e uma técnica para organização e orientação da equipe envolvida no desenvolvimento de modelos (a DT).

A linguagem de diagramação serve para representar componentes, as suas inter-relações e como eles se encaixam em uma estrutura hierárquica com diversos níveis de detalhamento. A técnica de análise (SA) orienta de forma sistemática o desenvolvimento de modelos através de tres princípios: decomposição, limitação da informação e orientação. A técnica de projeto (DT) orienta e controla o processo de desenvolvimento e faz uso da linguagem gráfica e da SA para a geração de documentação e entendimento do assunto analisado.

Nem a linguagem gráfica, nem a SA, nem a DT pretendem resolver problemas. São mecanismos que permitem exprimir, entender, manipular e verificar opções para solucionar problemas [Ross85].

[*] O termo 'metodologia' utilizado por Ross é substituído neste texto por 'método', tendo em vista as definições estabelecidas no capítulo 1.

3.2. A Técnica de Análise Estruturada

[Ross85] agrupa os seguintes princípios para compor a técnica SA:

- decomposição 'top down', resultando em uma análise estruturada e hierárquica em vários níveis de detalhamento;

- limitação da informação, restringindo a decomposição de qualquer assunto a um máximo de seis (6) componentes e um mínimo de três (3). O limite de 6 assegura que, no contexto geral, o significado de cada componente não seja muito difícil de entender. Em [SofT76], cita-se que experimentos psicológicos mostram que é difícil ao ser humano incorporar mais do que 5 a 7 conceitos distintos de uma só vez e que o limite inferior de 3 é utilizado para assegurar a existência de detalhes suficientes, de maneira a fazer que a decomposição seja interessante;

- modelagem segundo uma orientação, que significa que toda solução de um problema é um modelo baseado em um contexto, um propósito e um ponto de vista. O contexto estabelece o assunto do modelo como parte de um todo maior, cria um limite com o ambiente através da descrição de interfaces externas. O propósito estabelece o objetivo do modelo. O ponto de vista determina o que pode ser visto no contexto e qual a perspectiva para isso.

Segundo a técnica SA, uma análise deve ser feita através da elaboração de modelos com orientações diversas, de maneira tal que os vários modelos, em conjunto, representem todos os detalhes relevantes do problema com vistas ao seu completo entendimento.

3.3. A Técnica de Projeto

A base da técnica DT é o ciclo autor-comentarista utilizado durante o desenvolvimento ou manutenção de software.

Diagramas ou partes de modelos são criados por 'autores' e distribuídos para revisão. Um autor produz 'kits' que podem ser compostos por diagramas, textos, glossário (definidos no capítulo 6) e qualquer outra informação considerada relevante.

A revisão é feita por 'comentaristas' e 'leitores'. Existe uma diferença entre um comentarista e um leitor: espera-se comentários do primeiro, não do segundo, ou seja, um leitor é o elemento da organização que não tem função no desenvolvimento de um kit, mas mesmo assim quer dar sua contribuição. A revisão é elaborada por escrito, em uma cópia do 'kit', sendo formada por comentários que são enviados ao autor. O autor responde a cada um dos revisores utilizando a mesma cópia do 'kit'. Caso o comentarista concorde com a resposta do autor, arquiva a cópia do 'kit'. Caso contrário, é marcada uma reunião para solucionar as diferenças.

Se a reunião não alcançar resultados satisfatórios, o problema é enviado para decisão das chefias, que podem optar pela re-análise do problema por parte do autor.

A DT envolve os seguintes tipos de pessoas na equipe necessária à realização de uma análise [Ross77], além de autores, comentaristas e leitores:

- especialistas - pessoas a partir de quem os autores obtêm informações a respeito do sistema e seus requisitos;
- comitê técnico - grupo de técnicos experientes responsáveis pela revisão final de cada processo. Resolve assuntos técnicos e recomenda decisões ao gerente do projeto;
- bibliotecário - responsável pela manutenção centralizada de um arquivo com os documentos do projeto, pela cópia e respectiva distribuição dos documentos, pela manutenção de versões, além do controle de distribuição de kits e revisões;
- gerente do projeto - responsável técnico pela análise do sistema;
- monitor - especialista em SADT que dá suporte à equipe quanto ao uso correto da técnica e da linguagem.

Um processo na DT documenta todas as decisões e as respectivas razões, para facilitar a futura manutenção do sistema.

Os criadores do SADT aconselham que seja feita uma análise por um autor através de dois modelos complementares: um modelo funcional e outro sobre a informação. Em outras palavras, a orientação do primeiro modelo é a de dar relevância às atividades ou funções e a do segundo às informações ou estruturas de dados.

Um modelo completo SADT deve conter:

- actigrams (diagramas do modelo funcional)
- datagrams (diagramas do modelo de informação)
- FEOs (diagramas utilizados para comentários - 'For Exposition Only')
- Texto (acompanha cada diagrama com explicações)
- Glossário (sobre a identificação e estrutura dos componentes do modelo)
- Índice de Nós (cada diagrama é um nó)

3.4. A Linguagem Gráfica

A linguagem gráfica se baseia na idéia de que sistemas podem ser compreendidos relacionando dados e atividades. Isto, segundo Ross [Ross77b], nada mais é do que a forma como entendemos situações reais e problemas, sendo a maneira pela qual as linguagens naturais podem representar objetos, utilizando substantivos (dados ou coisas) e verbos (atividades).

A figura a seguir, retirada de [SofT76], exemplifica uma decomposição SADT.

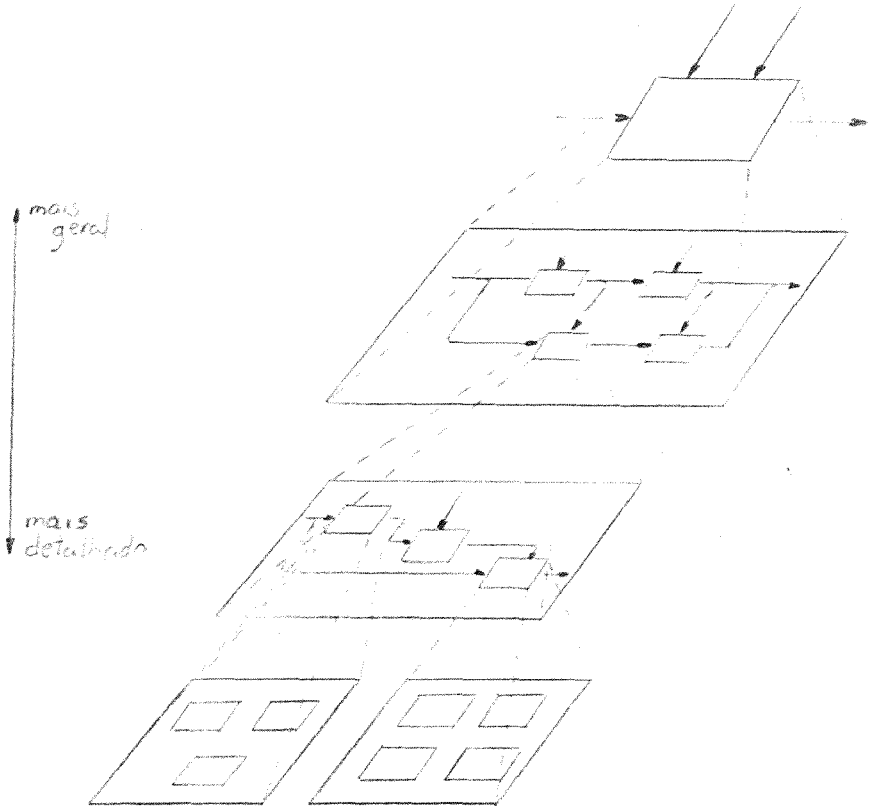


Fig. 1 Estrutura de decomposição

No processo de decomposição utilizado na técnica SA, um assunto pode ser sucessivamente decomposto até um nível de detalhamento considerado adequado (fig. 1). Dependendo da orientação, o objeto da decomposição pode ser uma atividade ou uma informação, que pode ser decomposta respectivamente em atividades ou informações. Cada componente de uma decomposição é representado por uma caixa (retângulo), cujos quatro lados têm respectivamente o seguinte significado: esquerda - **entrada**; direita - **saída**; superior - **controle**; e inferior - **mecanismo**. A cada lado podem ser associadas setas que representam fluxos de dados (caso a caixa represente uma atividade) ou fluxos de atividades (caso a caixa represente um dado).

Toda caixa, quando decomposta, é chamada caixa-pai, dando origem a um diagrama-filho com caixas-filhas. O diagrama da caixa-pai também é conhecido como diagrama-pai. Os filhos, por sua vez, podem tornar-se pais de novas decomposições.

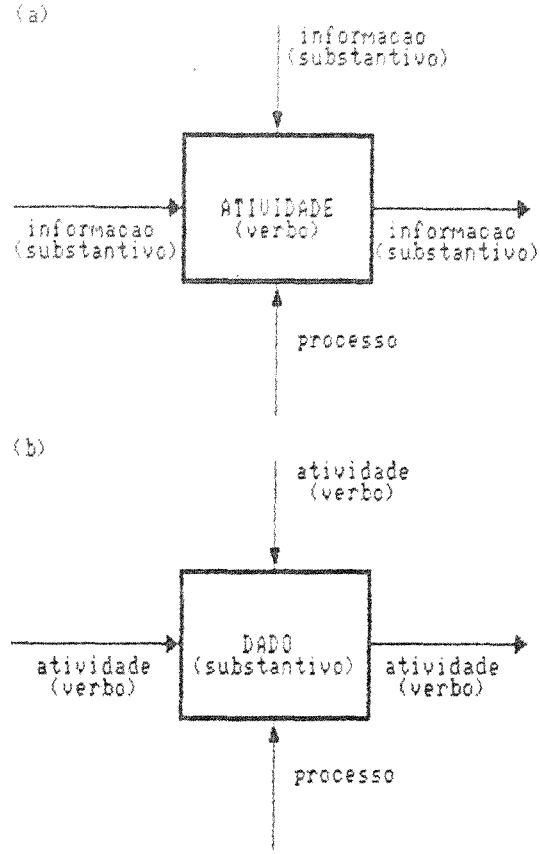


Fig. 2 Representação de assuntos segundo a orientação funcional (a) ou de dados (b)

Considerando o modelo funcional, uma caixa ('actigram') representa uma função que transforma entradas em saídas, segundo uma possível orientação que são os controles. Um controle indica a forma sob a qual a entrada é transformada na saída. Neste contexto, um mecanismo representa um processo ou elemento que efetivamente pode levar a cabo a função (fig. 2-a).

Já no modelo de dados, uma caixa ('datagram') representa um tipo de dado ou o estado de um dado que é resultado de uma transformação feita pela atividade-entrada e que pode ser transformado ou usado pela atividade-saída. Um controle indica a atividade que controla a criação ou uso da informação. Neste contexto, um mecanismo representa o meio ou dispositivo que armazena a informação (fig. 2-b).

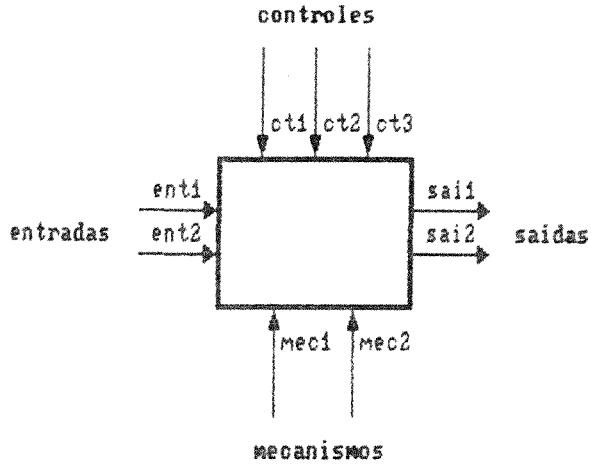


Fig. 3 Significado das setas e seus códigos

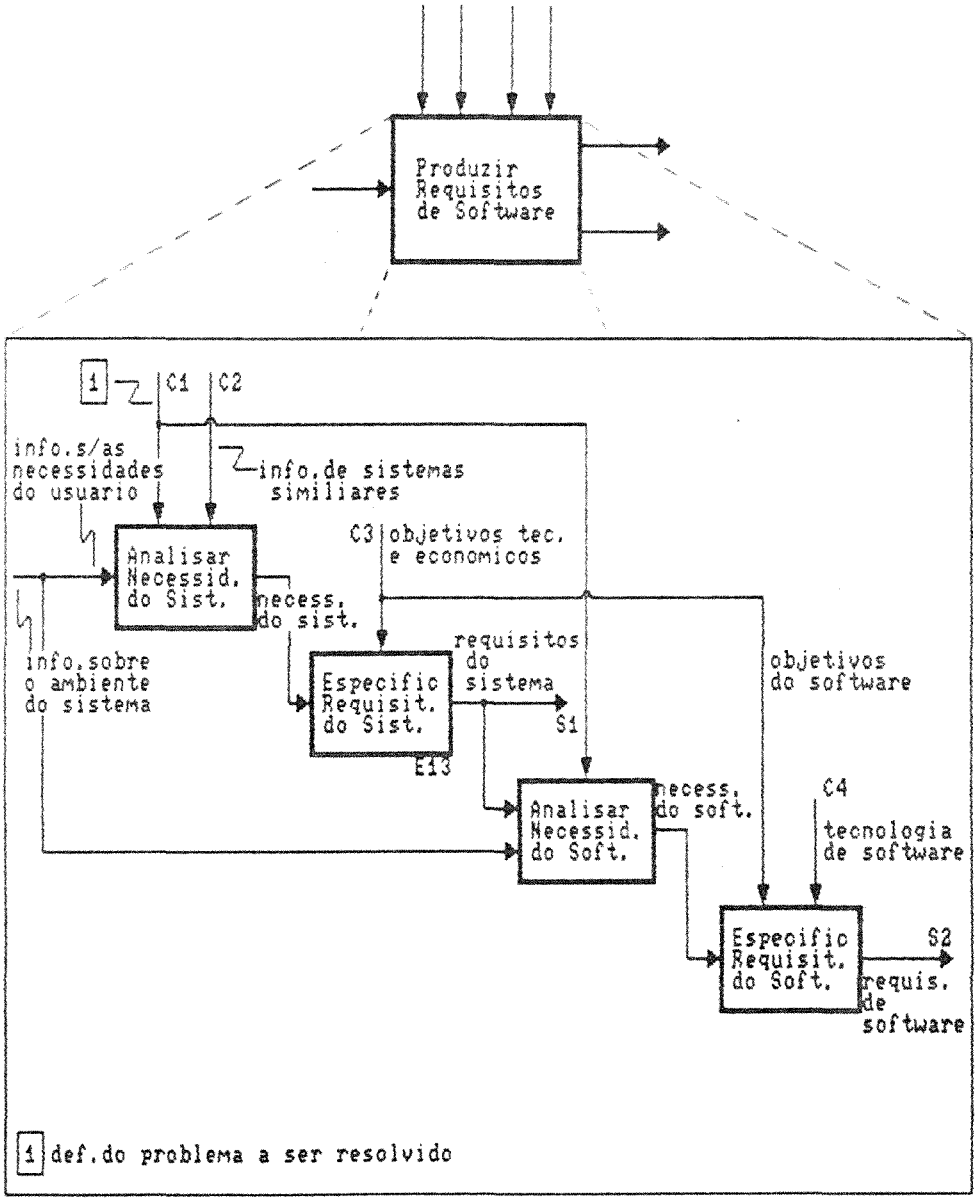


Fig. 4 Diagrama SADT mostrando o processo de produção de requisitos de software, segundo Freeman [Free83b]

Entradas, saídas e controles são interfaces entre os componentes e são totalmente diferentes dos mecanismos. Os componentes são conectados de maneira tal que, considerando o modelo funcional, a saída de uma transformação pode ser entrada de outra caixa ou pode controlar a transformação de entradas em outra caixa.

Caixas e setas são rotuladas de forma que atividades sejam verbos e dados sejam substantivos. As pontas das setas recebem um código ('ICOM code') conforme o lado da caixa a que estejam associadas. O código é formado por uma letra: I - entrada ('input'), C - controle ('control'), O - saída ('output') e M - mecanismo ('mechanism'), seguida por um número seqüencial, indicando a ordem de cima para baixo ou da esquerda para a direita (Fig. 3). Em um diagrama-filho devem aparecer códigos ICOM nas extremidades de setas associadas à caixa-pai (Fig. 4).

Na figura 4 podem ser observados outros aspectos (definidos no capítulo 6) da linguagem gráfica:

- a informação E1 (que é um 'detail reference expression - DRE') abaixo da caixa 2 do diagrama aponta a existência do diagrama E1, que é o detalhamento dessa caixa;
- a seta com código ICOM I1 tem o seu fluxo de dados 'info sobre o ambiente do sistema' dividido em 'info s/as necessidades do usuário', que é entrada da caixa 1, e 'info sobre o ambiente do software', que é entrada da terceira caixa;
- a seta com código ICOM C1 tem o seu fluxo de dados 'definição do problema a ser resolvido' duplicado para ser controle tanto da primeira como da terceira caixa;
- todas as setas têm um rótulo associado;
- devido à falta de espaço para rotulação, a seta com código ICOM C1 tem uma nota de rodapé associada;
- o 'layout' em escada pretende demonstrar o domínio de uma caixa sobre as outras, com o propósito de auxiliar a sua interpretação correta;

Todo modelo tem um diagrama inicial denominado A-0 (A menos zero), que é composto por uma caixa e setas (sem código ICOM) que representam os detalhes mais importantes do problema. Esta caixa é detalhada no diagrama A0, que deve seguir a sintaxe e a semântica da linguagem gráfica. As caixas subsequentes são numerados A1, A2 e assim por diante.

Todo diagrama tem duas identificações. A primeira individualiza o diagrama dentro da árvore hierárquica resultante dos detalhamentos. Por exemplo: A0, A1, A333; A333 representa o diagrama relativo ao detalhamento da terceira caixa do diagrama A33. A segunda identifica o diagrama segundo a ordem de desenvolvimento. Por exemplo, E13 representa o décimo terceiro diagrama desenvolvido pelo autor cujo nome tem a inicial E.

Em [Ross77b] Ross apresenta 40 componentes que representam o núcleo da linguagem gráfica, alguns deles constantes da figura 4. A estes somam-se outros que também fazem parte da linguagem e que são apresentados em outros artigos sobre SADT, como acontece em [Dick81]. No capítulo 6 são analisados estes componentes, um a um, de maneira a permitir a introdução dos componentes da linguagem do ANA-RE. O anexo 1 tem um quadro que resume estes componentes.

3.5. Aplicação do SADT

Ross afirma que o SADT é completamente geral e aplicável a qualquer situação [Ross85], podendo, em princípio, ser utilizado para qualquer propósito, além de prover a base para um método de solução de problemas. Na prática, porém, é conhecido como método para a definição de requisitos. O produto final de uma análise através do SADT é um modelo de entendimento bem estruturado que pode ser benéfico mesmo a nível individual. Assuntos filosóficos, políticos, sociais, legais, científicos ou artísticos estão sujeitos a análise via SADT e os modelos resultantes podem efetivamente comunicar as idéias a outras pessoas. O mesmo método e disciplina podem ser utilizados para modelar o ambiente, requisitos e soluções de um problema, além da organização, operação, cronograma, orçamento e plano de ação de um projeto [Ross77b].

Os criadores do SADT afirmam que o método pode ser aplicado durante todo o ciclo de vida de um projeto de software modelando o ambiente e operações correntes, os requisitos, as especificações de projeto, o plano de testes, o plano de conversão, as operações novas, o ambiente novo, além do desenvolvimento propriamente dito. Há, além disto, a possibilidade de especificar modelos segundo diferentes visões, contextos e propósitos.

4. A AUTOMAÇÃO DO SADT

Um ambiente automatizado é composto por um conjunto de ferramentas que auxiliem o processo de criação e evolução, parcial ou total, de sistemas de software, segundo um determinado método ou metodologia.

Em 09/76 Ross e Schoman [RossTI] afirmaram que, até a época, as ferramentas automatizadas existentes, que total ou parcialmente se aplicavam a requisitos, partiam do princípio de que os requisitos necessitavam ser especificados formalmente (ao nível de poderem ser compreendidos e tratados por um computador). Este formalismo, segundo os autores, gerava dois problemas básicos: o primeiro era que os requisitos especificados em forma textual não podiam ser traduzidos diretamente a uma interface com uma linguagem automatizada para descrição de problemas; o segundo era que uma ferramenta automatizada nunca poderia executar a tarefa de definição de requisitos, porque a definição e verificação de requisitos é uma tarefa a ser executada por analistas e usuários.

Em um de seus últimos artigos a respeito do SADT (04/85), Ross [Ross85] enfatiza mais uma vez o problema da ausência de formalização da linguagem gráfica. A razão apontada para esse problema é a alta complexidade dessa formalização. Desta maneira, considera que as tentativas então existentes de se automatizar SADT eram, na realidade, particularizações de interpretações de modelos SADT através de semânticas formais.

Os ambientes automatizados que suportam o método SADT ou métodos dele derivados, como é o caso do IDEF0, são todos de origem recente. Propagandas destes ambientes são encontradas em [Soft88]. Uma experiência de desenvolvimento de um ambiente de engenharia de software, desenvolvido na UFRGS [NuneSTI] [Lean87], visa, entre outras técnicas e métodos, a automação parcial do SADT como ferramenta para a fase de definição de requisitos.

4.1. O Que Automatizar no SADT

Considerando possíveis automatizações do SADT, Ross [Ross85] afirma que as técnicas manuais do SADT suportadas por um bibliotecário bem treinado e por um bom desenhista são difíceis de serem substituídas por similares automatizadas.

A afirmação de Ross não é de todo descabida, uma vez que parte da flexibilidade encontrada na linguagem gráfica do SADT é difícil, senão impossível, de se automatizar (por exemplo, a omissão de setas óbvias com a finalidade de simplificar um diagrama).

Conforme já mencionado, o método SADT compõe-se da linguagem gráfica, da técnica para análise estruturada SA e da técnica de projeto DT. Automatizar o SADT significa automatizar cada um destes componentes no que tange à busca e localização de informações (especificações de requisitos), navegação através da árvore hierárquica de diagramas, emissão de documentação padronizada, gerência de configuração e verificação de consistência entre diagramas de um mesmo modelo.

A automação dos componentes do SADT é entendida da seguinte forma:

- formalização da linguagem gráfica de maneira a representar de forma única cada um de seus componentes sintáticos;
- verificação da obediência aos princípios da SA; e
- diminuição de esforço e aumento na qualidade dos resultados dos trabalhos dos usuários (bibliotecários, desenhistas, chefias, autores, comentaristas e leitores) devido à técnica de projeto DT.

4.1.1. Formalização da Linguagem Gráfica

Todos os componentes automatizáveis dependem da formalização da linguagem gráfica, ou, como prefere Ross, de uma particular formalização da linguagem.

Formalizar a linguagem gráfica significa a representação de forma única de, pelo menos, cada um dos 40 componentes de seu núcleo (vide anexo 1). Isto pode implicar, por exemplo, em limitações no tamanho dos identificadores, no número de níveis hierárquicos em um modelo e no número de componentes de um diagrama, devido ao hardware e à resolução dos terminais de vídeo. Além disto, pode ser necessária a substituição de símbolos ou construções da linguagem.

É possível lançar mão de uma linguagem declarativa (não gráfica) através da qual se possa descrever todas as informações passíveis de existir em um diagrama SADT. A transformação de uma representação declarativa em representação gráfica é possível, mas requer recursos tais como terminais gráficos, cujo custo é dependente da qualidade final desejada.

A automação permite a verificação da sintaxe e parte da semântica da linguagem gráfica, pois necessariamente deve existir um reconhecedor da linguagem. Soma-se a isso a possibilidade (limitada) de se fazer a consistência de um diagrama com relação aos outros diagramas que compõem o modelo. Esta consistência pode envolver a detecção de, por exemplo, caixas ou setas isoladas ou duplicação de identificação.

4.1.2. Automação da Técnica de Análise Estruturada

São três os princípios básicos que compõem a técnica de análise estruturada [4]: decomposição, limitação da informação e orientação.

A automação da técnica pode fazer a verificação do princípio de limitação da informação através da consistência dos diagramas para detectar a existência de um máximo de 6 caixas e um mínimo de 3. Além disto, o princípio da decomposição pode ser suportado através de mecanismos que permitam a navegação e o controle dos diagramas nos diversos níveis de um modelo. Quanto ao princípio de orientação para a modelagem, a gerência de configuração contribui para o isolamento e integridade dos modelos.

4.1.3. Automação da Técnica de Projeto

A automação da técnica de projeto é possível desde que os diagramas especificados através de uma linguagem automatizada possam ser relacionados entre si, de maneira a permitir o estabelecimento das relações pai-filho e da pertinência a uma única versão de um mesmo modelo. Isto se torna factível através do uso de ponteiros e de informações comuns, gerenciável a partir de uma base de dados de diagramas.

A existência de hardware que permita a emissão gráfica de diagramas e informações afins com qualidade aceitável, acrescido de um processo-padrão automático de rotulação e identificação dos componentes dos diagramas, permite dispensar os desenhistas da equipe.

O bibliotecário transforma-se de um super-guarda-livros em um administrador de base de dados, responsável pela integridade e disponibilidade da casa.

O trabalho de análise centrado na base de dados de diagramas agiliza o trabalho da equipe de especificação, seja por permitir o acesso concomitante de vários autores, leitores e comentaristas aos diagramas, seja por permitir a emissão rápida de documentação atualizada e coesa, ou mesmo por manter disponíveis, de forma rápida, informações sobre o estágio de um projeto.

4.2. Extensões à Automação do SADT

Conforme discutido na seção 4.1., a automação é conseguida através da formalização da linguagem gráfica, suportada por uma base de dados de diagramas.

A existência de uma particular automação do SADT, como a considerada anteriormente, pode incorporar extensões de maneira a resultar em um ambiente para especificação e verificação de requisitos. Estas extensões são discutidas a seguir.

4.2.1. Complementação da Verificação de Consistência

A verificação de consistência de um diagrama em relação aos demais diagramas de um modelo pode ser fortalecida se for possível comprovar que as identificações das setas da caixa-pai se mantêm no diagrama-filho. Isto é contrário à orientação encontrada em [Ross77b], segundo a qual, para que a representação gráfica seja flexível, as setas limítrofes (sem origem ou sem destino) em um diagrama não necessitam ter o mesmo 'layout' geométrico, relacionamento ou rótulos das correspondentes setas da caixa-pai, as quais estão desenhadas em um diagrama totalmente diferente (vide 6.1.2-f).

As setas de uma caixa-pai estão consistentes com as correspondentes setas do diagrama-filho, segundo as orientações do SADT, desde que no diagrama-filho existam setas com códigos ICOM que correspondam às setas da caixa-pai. Isto não impede que uma mesma seta receba mais de um código ICOM ou que várias setas recebam o mesmo código ICOM, não sendo necessária a preservação dos rótulos associados à caixa-pai.

Qualquer erro de um autor na inserção de códigos ou na modificação do 'layout' ou ainda na rotulação de setas em um diagrama-filho pode gerar falsos relacionamentos entre identificações diferentes ou entre resultantes do desvio de uma seta e ela mesma.

A solução para este problema é a inclusão de mecanismos formais que permitam a modificação de relacionamento e de rótulos de setas dentro de um mesmo diagrama, desde que que as setas limítrofes preservem as funções e os rótulos, e sejam em número igual às da caixa-pai, não havendo objecões quanto ao 'layout'.

4.2.2. Simulação de Modelos

A execução das especificações funcionais descritas através de fluxo de dados é factivel e exemplos podem ser encontrados em trabalhos publicados [Tate85] [Dahl87]. A base para isto é a existência de um simulador que reconhece e executa as especificações das folhas da árvore hierárquica que compõe as especificações funcionais. Estas especificações devem estar representadas formalmente através de uma linguagem adequada, preferencialmente de alto nível e com características de fluxo de dados (linguagens funcionais por exemplo).

É necessário que as especificações dos diagramas-folha SADT ou de suas caixas se tornem mais rigorosas e completas. Este rigor se espelha no formalismo utilizado para as especificações e pode ser tão complexo e completo quanto uma liguagem de alto nível utilizada para programação.

Através da simulação de especificações obtém-se um sistema de protótipos rápidos que pode ser utilizado para a validação das especificações funcionais.

4.2.3. Outras Facilidades

Além das extensões já apontadas, um ambiente automatizado deve prover facilidades de correio eletrônico (para comunicação centro de uma equipe), editores gráficos, ferramentas para suporte à gerência dos trabalhos e mecanismos que garantam segurança em termos de permissão de acesso a diagramas. É desejável também a inclusão de comandos de ensino ('teach' ou 'help') para usuários esporádicos e do uso opcional de menus e janelas.

5. O METODO ANA-RE E O AMBIENTE SAES

O método ANA-RE, proposto por esta dissertação, é destinado a orientar a elaboração e o controle de modelos analíticos de problemas. ANA-RE se baseia no método SADT e, como este, é utilizável nas tarefas de ANÁlise de problemas e especificação de REquisitos.

O ANA-RE supera o SADT no que se refere à possibilidade de suporte automático de suas regras e difere do mesmo em sua linguagem de especificação, o que resultou em regras de aplicação e em linguagem diferentes. As diretivas relacionadas à constituição da equipe e à forma de trabalho são basicamente as mesmas do SADT.

Roman [Roma85], baseando-se no trabalho de Ramamoorthy e So [Rama78], estabelece os seguintes critérios para classificar métodos e técnicas de especificação de requisitos: base formal, escopo, nível de formalismo, grau de especialização, fase de especialização e técnica de desenvolvimento. Segundo esses critérios o ANA-RE tem a apresentação que segue.

5.1. Base Formal

A representação gráfica do método ANA-RE utiliza o paradigma de fluxo de dados, que basicamente consiste em elementos funcionais ou atividades de processamento interligados por fluxos.

Este paradigma é bastante atraente para a especificação de requisitos, uma vez que se adapta bem para modelar a estrutura e o comportamento de quase todos os tipos de organizações.

5.2. Escopo

O ANA-RE destina-se a tratar especificações de requisitos funcionais, suportando ainda a especificação conjunta de interrupções, o que permite a análise de modelos com requisitos de tempo real. Outros requisitos não funcionais podem ser especificados de forma textual, sendo tratados como texto descritivo ou comentários.

5.3. Nível de Formalismo

O ANA-RE apresenta um nível de formalismo que permite ao usuário manter a completeza, a não ambigüidade e a consistência dos requisitos funcionais. O ambiente que o suporta, o SAES, não permite a avaliação automática dos requisitos funcionais, porém provê facilidades de verificação via mecanismos de navegação através dos vários níveis de detalhamento. O SAES suporta fácil modificação de requisitos e por conseguinte a respectiva manutenção.

5.4. Grau de Especialização

Roman [Roma85] prevê três tipos de grau de especialização para um método: aqueles específicos a um domínio, aqueles sensíveis a um domínio e aqueles independentes de domínio. O ANA-RE representa, seguindo o SADT, um método independente de domínio, atendendo por isso, de forma genérica, a análise de uma grande gama de problemas.

5.5. Fase de Especialização

Considerando um ciclo de vida convencional para o desenvolvimento de software, o ANA-RE pretende atuar na fase de Análise e Especificação de Requisitos com ênfase no entendimento do problema, ao invés de se preocupar com os aspectos da fase de Projeto de Sistema ou na transição para esta, a partir da Análise.

5.6. Técnica de Desenvolvimento

O ANA-RE pretende que o desenvolvimento da fase de Análise e Especificação de Requisitos se faça na forma convencional, ou seja, que a especificação esteja completa antes da passagem para as fases seguintes do desenvolvimento. Não existem, porém, indicações contrárias a que seja utilizado em conjunto com a técnica de protótipos rápidos (o que não é previsível no SADT). O ambiente SAES pretende suportar facilidades para permitir a futura implementação da técnica de protótipos rápidos através de simulação das especificações de requisitos funcionais.

O SAES deve suportar uma interface gráfica e outra declarativa para permitir a especificação de requisitos funcionais, além de facilidades de menus e auxílio para a utilização do ambiente. Somam-se a isto a característica da especificação baseada na decomposição hierárquica dos problemas e as facilidades para a documentação e gerência das especificações de requisitos.

5.7. O Ambiente SAES e sua Automação

O SAES é uma proposta de ambiente que se destina a auxiliar apenas o processo de análise, definição e especificação de requisitos, automatizando o método ANA-RE, que se baseia em modificações e extensões sobre o SADT. O ambiente visa oferecer facilidades para permitir o entendimento, definição, especificação e verificação de requisitos de sistemas de software.

Um ambiente que suporte o SADT difere do SAES basicamente no método suportado (SADT e ANA-RE respectivamente). Desta forma, as considerações feitas no capítulo 4 valem também para o SAES: sua implementação requer a formalização de sua linguagem gráfica e a automação da técnica de análise estruturada e da técnica de trabalho com o suporte de uma base de dados de diagramas.

Todas as ferramentas automatizadas que se prestam a suportar as tarefas de especificação de requisitos utilizam uma base de dados que contém a representação das especificações, permitindo sua manipulação e também os inter-relacionamentos de informações.

Ross, em [Ross85], se apresenta cético com relação a estas bases de dados. Afirma que apesar de várias tentativas de se automatizar a parte gráfica do SADT, é certo que os modelos desenvolvidos com SADT contêm uma quantidade considerável de informações em um espaço compacto e que, mesmo sendo modestos os requisitos gráficos, a integração dos mesmos com técnicas convencionais de bancos de dados para busca, verificação e análise normalmente sobrecarregam demais um sistema.

O ambiente SAES deve ser desenvolvido a partir de um ambiente de software que disponha de um SGBD, de editores de texto e de facilidades para processamento gráfico em conjunto com o hardware necessário. Como se verá no capítulo 7, pressupõe-se o uso de um SGBD relacional, embora esta restrição não seja essencial ao SAES. Para a implementação de parte das extensões propostas, há ainda a necessidade de um serviço de correio eletrônico, um gerenciador de janelas e facilidades para criação de menus e para controle de acesso à informação.

Desconsiderando as extensões propostas no item 4.2 e considerando a possibilidade de se ter à mão um ambiente ideal para a automação do SAES, a complexidade maior desta automação recai no desenvolvimento de um projeto de definição para a base de dados, de uma interface gráfica para diagramas e de um processo de verificação de consistência. Esta dissertação se concentra na definição e implementação da base de dados, descrita no capítulo 7, validando assim parte da proposta.

Como se verá a seguir, o ANA-RE utilica basicamente a mesma notação gráfica do SADT e qualquer sistema especificado em SADT pode também ser especificado em ANA-RE.

O próximo capítulo dá a descrição do ANA-RE sob a forma de modificações propostas ao SADT.

6. ESPECIFICAÇÃO DO MÉTODO ANA-RE

O ANA-RE é o resultado de modificações propostas ao SADT visando a correção de algumas de suas falhas e o aumento do universo de problemas passíveis de especificação formal. Os princípios básicos da técnica SA foram adotados sem alteração (vide 3.2). Desta forma, este capítulo apenas apresenta a linguagem gráfica e a técnica DT para o ANA-RE. Salienta-se que a redação deste capítulo assemelha-se à de um manual de referência. Este estilo de apresentação é proposital, visando permitir a aplicação do ANA-RE por pessoas já familiarizadas com o SADT.

6.1. A Linguagem Gráfica do ANA-RE: Modificações no SADT

Esta seção apresenta a linguagem gráfica do ANA-RE. Inicialmente, são discutidos os componentes derivados do SADT, mostrando assim que o ANA-RE provê as mesmas facilidades gráficas que este último. Em seguida são apresentados componentes que não existem no SADT, mostrando que a linguagem do ANA-RE é mais poderosa que a do SADT.

Ao leitor que desejar ter uma idéia geral das diferenças entre as duas linguagens recomenda-se reportar-se ao fim desta seção, onde existe uma tabela comparativa. Vale observar que o SADT requer a especificação de dois modelos complementares: um constituído por actigrams, ou diagramas de atividades, e outro constituído por datagrams, ou diagramas de informação. O ANA-RE, ao contrário, requer apenas actigrams, ficando o datagram como forma complementar de informação a respeito apenas do correspondente actigram.

Nem todos os componentes constantes do núcleo gráfico do SADT são gráficos, no sentido de poderem ser parte de uma linguagem gráfica para especificação de diagramas. Como resultado da análise efetuada, a dissertação propõe uma classificação dos aspectos, para seu melhor entendimento. Pode-se, inicialmente, agrupá-los em gráficos e não gráficos. Dentre os componentes gráficos encontramos componentes puramente sintáticos (**seta** por exemplo), componentes puramente semânticos (**seta limítrofe** por exemplo) e componentes sintático-semânticos (**seta bidirecional** por exemplo). Dentre os componentes não gráficos encontramos componentes que complementam as informações dos diagramas (**texto** por exemplo), componentes para orientação do processo de elaboração de diagramas (**layout em escada** por exemplo) e componentes para permitir a referência aos componentes de um diagrama em uma frase explicativa (**código de seta** por exemplo).

A seguir cada um dos componentes do SADT é apresentado e analisado de maneira a introduzir o respectivo componente do ANA-RE. Esta apresentação é feita considerando cada um dos grupos discutidos anteriormente.

Para facilitar a leitura, só será feita menção ao método (ANA-RE ou SADT) quando houver diferenças introduzidas.

Sempre que conveniente são apresentados exemplos para melhor entendimento dos conceitos e propósitos envolvidos.

6.1.1. Componentes Sintáticos

Um componente sintático é um componente básico da linguagem gráfica que tem mais de um significado semântico associado, ou seja, pode originar componentes semânticos distintos em um diagrama.

São considerados componentes sintáticos **caixa**, **seta**, **nome** e **rótulo**. Todos eles são mantidos na linguagem gráfica do ANA-RE com o mesmo propósito, conceito e notação de SADT, exceto **nome** e **rótulo** cuja notação permite a utilização de abreviações.

a. Caixa

O componente **caixa** tem como propósito limitar um contexto onde se representa uma função (actigram) ou um dado (datagram). Este limite estabelece o que faz parte do contexto e o que não faz.

Em um diagrama as caixas são numeradas com um número sequencial colocado no canto direito inferior. Esta numeração (de 1 a 6) obedece a ordem das caixas da esquerda para a direita e de cima para baixo.

Notação:

b. Seta

O componente **seta** tem como propósito relacionar e conectar contextos (caixas), representando dessa forma dados (actigram) ou atividades (datagram). Uma seta estabelece uma origem e um destino.

Notação:

c. Nome e Rótulo

Os componentes **nome** e **rótulo** têm como propósito identificar, respectivamente, uma **caixa** e uma **seta**. Um **nome** se estabelece através de um verbo (actigram), que identifica o acontecimento levado a cabo, ou através de um substantivo (datagram), que identifica a informação definida pelo contexto. Um **rótulo** se estabelece através de um substantivo (actigram), que identifica a informação representada, ou através de um verbo (datagram), que identifica o acontecimento representado.

Apesar de se definir um **nome** ou **rótulo** como constando de um verbo (actigram), geralmente se utiliza um verbo e seu objeto, resultando em uma oração sem sujeito, muitas vezes longa (por exemplo Analisar Necessidades do Sistema na figura 4 em 3.4). Fato similar ocorre com o substantivo (datagram) que geralmente se transforma em substantivo e predicativos.

Notação:

Para um actigram, xx--xx é um verbo e yy--yy é um substantivo; para um datagram, xx--xx é um substantivo e yy--yy é um verbo. Além disso, no ANA-RE xx--xx deve ser um título em letras maiúsculas e yy--yy um título em letras minúsculas. Caso se deseje utilizar uma abreviação ou acrograma, o primeiro caráter do título deve ser um '_' (underscore). A expansão ou definição da abreviação deve ser feita no componente **texto** (vide 6.1.4-a).

6.1.2. Componentes Semânticos

Um componente semântico é um componente cuja forma gráfica ou sintática se compõe por uma combinação de componentes sintáticos. Esta forma sintática, no entanto, tem mais de um significado associado, que varia de acordo com a situação onde é colocada no diagrama, ou de acordo com a constituição da combinação.

São considerados componentes semânticos **interface**, **mecanismo**, **desvio**, **junção**, **união**, **separação**, **desvio-ou**, **junção-ou**, **número-c**, **seta limítrofe** e **código ICOM**.

O componente **chamada** está incorporado ao componente **mecanismo** e os componentes **para-todos** e **de-todos** estão incorporados respectivamente aos componentes **desvio** e **junção**.

a. Interface

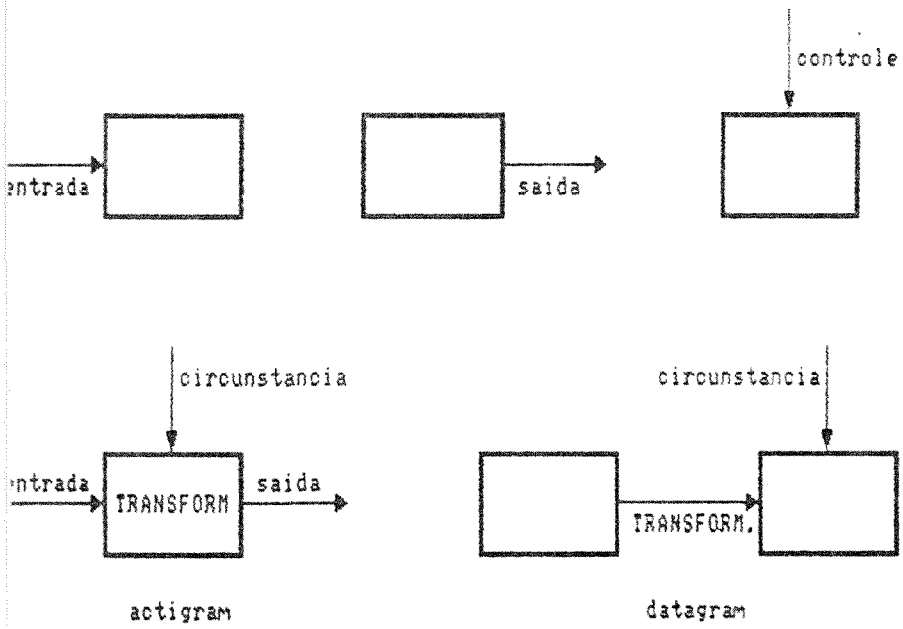
O componente **interface** tem dois propósitos distintos: o primeiro é representar uma transformação e o segundo representar uma circunstância.

Uma transformação envolve uma função e dois tipos de dados: insumo e produto da transformação. Uma transformação se estabelece através de entradas e saídas.

Uma circunstância envolve uma função e uma informação. A informação indica sob que forma deve ser levada a cabo a função (actigram), ou qual a função que controla a criação ou uso da informação (datagram). Uma circunstância se estabelece através de controles.

Entradas, saídas e controles têm como forma gráfica a **seta** e respectivo **rótulo** e se diferenciam de acordo com o lado a que se associam a uma **caixa:** esquerdo é entrada, direito é saída e superior é controle. Uma **seta** pode, no entanto, estar representando ao mesmo tempo duas funções (por exemplo, uma entrada e uma saída quando sai de uma caixa e entra em outra).

Notação:

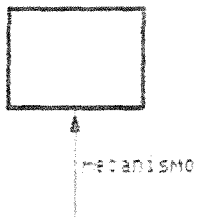


b. Mecanismo

O componente **mecanismo** tem como propósito representar um processo ou elemento que pode levar a cabo a função (actigram) ou o meio ou dispositivo que armazena a informação (datagram). Um mecanismo indica um processo (actigram) ou meio (datagram) existente que suporta a função (actigram) ou informação (datagram). Tal processo ou meio pode encontrar-se completamente especificado através de outro modelo SADT ou pode ser obviamente identificado por tratar-se de um suporte simples, concreto e elementar. Por exemplo, Compilador PASCAL pode ser o mecanismo da atividade Compilar Programa, ou a rotina adição pode ser o mecanismo da atividade Somar Parcelas.

Um mecanismo não é uma interface, como o próprio Ross explica, apesar de ter notação de seta. Para complicar ainda mais, tal seta não tem origem nem destino: é o meio de se conectar um outro modelo onde se encontra detalhado o contexto especificado na caixa. A seta que representa um **mecanismo** é associada ao lado inferior de uma **caixa**.

Notação SADT:



Proposta para o ANA-RE:

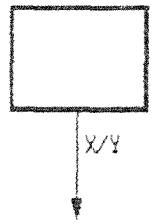
Utilizar outra forma gráfica, mantendo o propósito e o conceito, uma vez que a notação SADT é ambígua. Ao invés do uso da seta, utiliza-se a indicação de existência de **observação** (vide 6.2.5) ou a indicação da existência de detalhamento em outro modelo. Esta decisão se deve à necessidade de se representar de forma única cada um dos componentes para permitir a formalização da linguagem.

Notação ANA-RE:

* - o caráter '*' no canto superior direito de uma caixa indica a existência de uma **observação** onde se pode especificar como a função deve ser implementada

xxx - a identificação xxx no canto inferior direito deve indicar a **identificação do modelo** (vide 6.1.6-a) e/ou **número-c** (vide 6.1.3-a) onde se encontra detalhado o contexto da caixa.

Existe outro componente no SADT, **chamada**, cujo propósito é mostrar que há em outro modelo um detalhamento da caixa. Isto nada mais é que o propósito do componente mecanismo. Por este motivo o componente **chamada** não é discutido e detalhado em relação ao ANA-RE. A notação SADT utilizada em uma chamada é a seguinte:



Notar que é especificada uma seta no lado inferior de caixa apontando para fora, à qual são agregados a **identificação do modelo** (X) e o **número-c** do diagrama (Y).

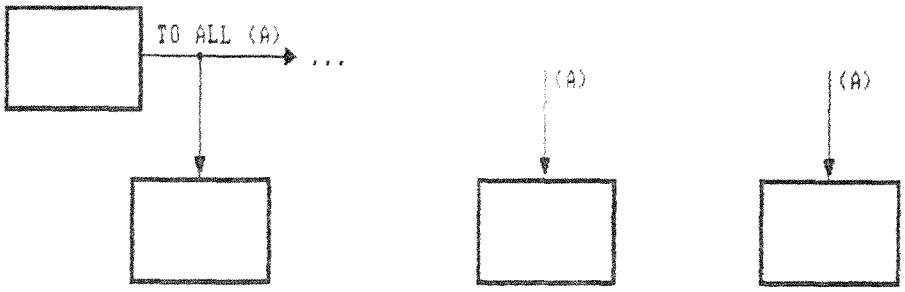
c. Desvio e Junção

Os componentes **desvio** e **junção** têm como propósito ser explícitos e evitar confusão em um diagrama. Um componente **desvio** ou **junção** se estabelece através da representação de duplicação de setas. A forma gráfica para desvio e junção é:

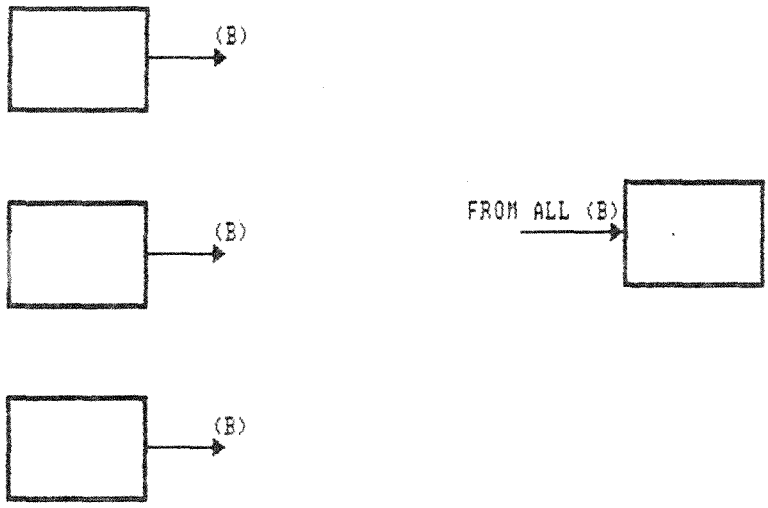
Notação:

Esta forma gráfica é idêntica à dos componentes **união** e **separação**, exceto pela constituição dos **rótulos**, que é o fator que em conjunto com a orientação das setas identifica de forma unívoca cada um dos quatro componentes.

Existem dois outros componentes no SADT, **para-todos** e **de-todos**, cujo propósito em muito se assemelha ao dos componentes desvio e junção. O conceito envolvido é idêntico, sendo operadores que multiplicam ou unem mais de duas setas, evitando o seu cruzamento através de rótulos especiais. A notação SADT desses componentes segue:



para-todos



de-todos

Apesar de evitar o cruzamento de setas, a utilização destes componentes não é recomendada por [Ross77]. Desta forma, esta dissertação considera os dois incorporados respectivamente aos componentes **desvio** e **junção**.

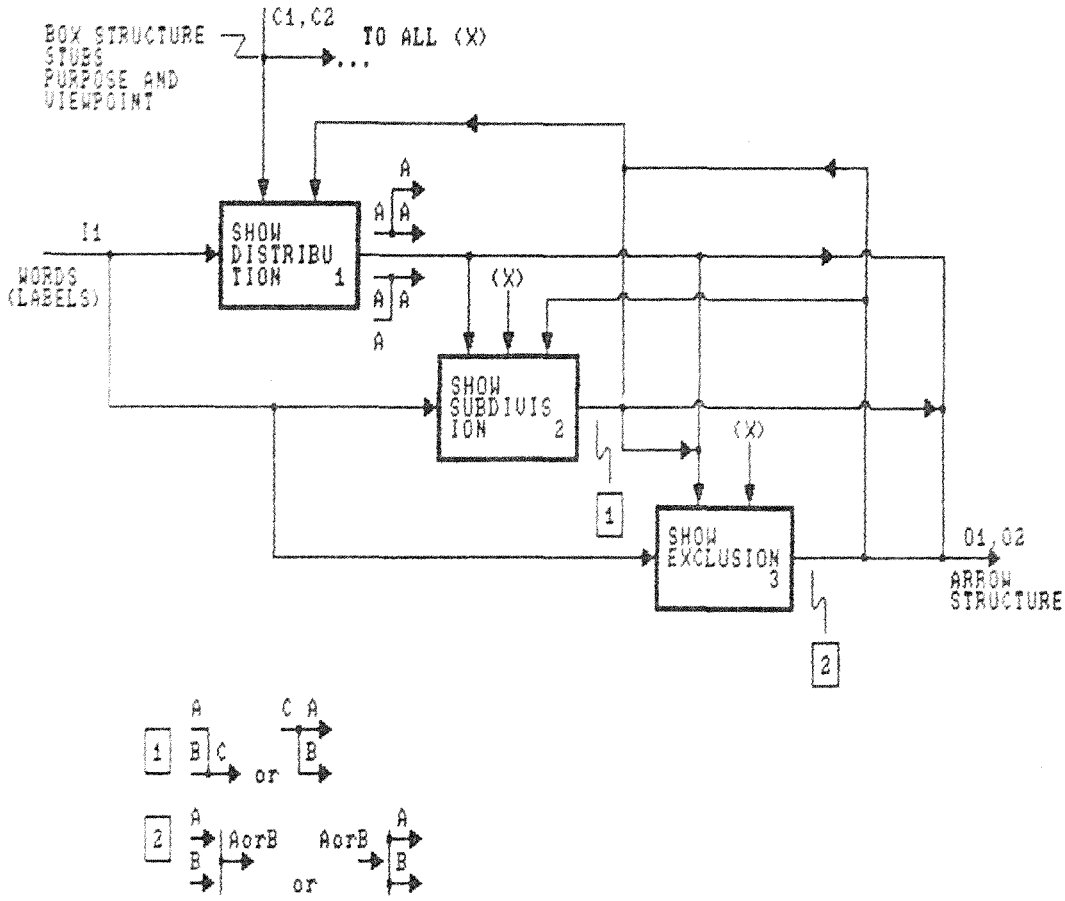


Fig 5 - Diagrama parcial SADT em 2º nível hierárquico de um modelo para estudo e análise do uso da linguagem gráfica, desenvolvido por [Ross77b]

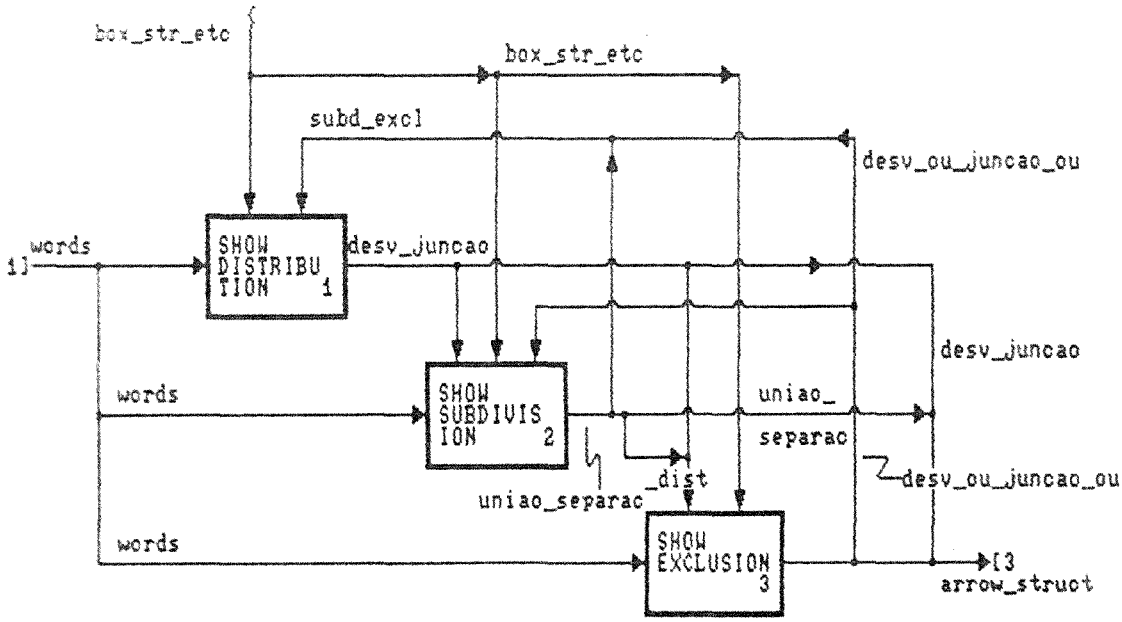


Fig. 6 - Diagrama ANA-RE que representa os mesmos componentes que a figura 5

As figuras 5 e 6 mostram **desvios** e **para-todos** em um diagrama parcial SADT e no diagrama ANA-RE correspondente. Foram excluídos alguns componentes do diagrama SADT. No diagrama ANA-RE, além disto, foi excluída a numeração das caixas. Note-se que a rotulação de setas com figuras não é permitida no ANA-RE.

d. União e Separação

Os componentes **união** e **separação** têm como propósito a concisão e clareza nos diagramas. Estes componentes se estabelecem através da representação de ramificações a partir de setas. A forma gráfica para **união** e **separação** é a mesma que para desvio e junção, exceto pela rotulação.

Notação:

Os operadores união e separação para o ANA-RE atuam estruturalmente sobre informações apresentando sua sub-estrutura (**separação**) ou formando uma super-estrutura englobando duas informações (**união**).

Como a diferenciação entre **união**, **separação**, **desvio** e **junção** se baseia na constituição dos rótulos, é obrigatória a especificação de pelo menos dois rótulos, para que em conjunto com possíveis rótulos subentendidos, possam indicar univocamente do que se trata.

Em um nível genérico, em especial no nível A-0, é praticamente impossível especificar todas as interfaces que aparecem nos níveis inferiores de detalhamento, seja devido à própria característica do detalhamento, seja devido à impossibilidade de se desenhar todas as setas. Isto fortalece o uso obrigatório dos componentes união e separação para detalhar a especificação estrutural das informações.

e. Desvio-ou e Junção-ou

Os componentes **desvio-ou** e **junção-ou** são operadores que atuam logicamente sobre informações resultando na separação exclusiva de informações que 'chegam' por uma seta (**desvio-ou**) ou por união de duas setas (**união-ou**).

[Ross77b] comenta que, na maioria das boas diagramações com SADT, estes componentes são raramente utilizados e só o devem ser caso introduzam melhor entendimento do diagrama.

A forma gráfica para **desvio-ou** e **separação-ou** é:

Notação SADT:

Proposta para o ANA-RE:

Manter o propósito e o conceito tal qual no SADT, simplificando a notação. Esta simplificação corresponde ao uso da mesma notação de desvio, junção, união e separação, modificando o rótulo para indicar a seta que não exige exclusividade.

Outra experiência, desenvolvida na UFRGS [Nune87] [Lean87], também propõe modificações na notação para **desvio-ou**, **junção-ou**, **união**, **separação**, **desvio** e **junção** visando a automação do método SADT.

Notação ANA-RE:

f. Seta Limítrofe e Código ICOM

Os componentes **seta limítrofe** e **código ICOM** existentes no SADT têm como propósito representar no diagrama-filho as interfaces e mecanismos associados à caixa-pai, estabelecendo uma conexão explícita entre os mesmos. As setas limítrofes podem ser vistas como setas que "penetram" na caixa-pai aparecendo no seu respectivo detalhamento que é o diagrama-filho. Os códigos ICOM identificam uma seta limítrofe de acordo com a interface ou mecanismo da caixa-pai.

Uma seta limítrofe é toda aquela que não conecta duas caixas, (apenas uma das extremidades está associada a uma das caixas do diagrama), recebendo um código ICOM na sua extremidade livre.

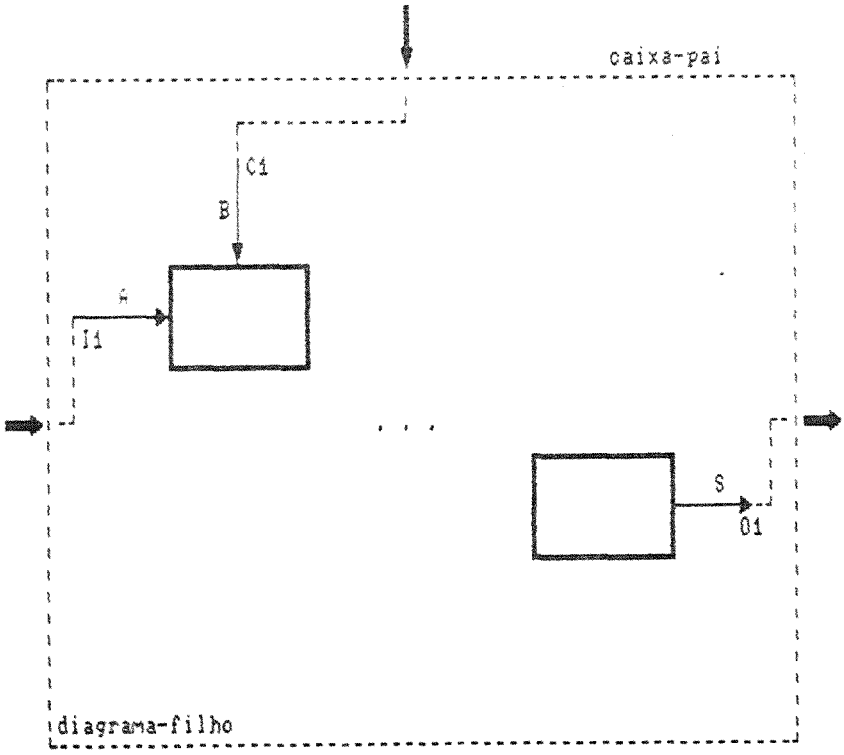
O código ICOM é composto por uma letra: I (entrada), C (controle), O (saída) ou M (mecanismo), seguida por um número inteiro seqüencial, indicando a ordem de especificação das setas na caixa-pai, de cima para baixo e da esquerda para a direita.

O SADT permite tamanha liberdade na especificação de rótulos para as setas limítrofes, que não é necessário a estas ter o mesmo 'layout' ou estar rotuladas da mesma forma que as interfaces e mecanismos correspondentes da caixa-pai. Desde que existam no diagrama-filho setas limítrofes com tantos códigos ICOM quantas forem as interfaces e mecanismos associados à caixa-pai, mantendo os mesmos papeis (entrada, saída, controle e mecanismo), a conexão entre caixa-pai e detalhamento existe e está bem definida. Esta liberdade apresenta o problema de uma mesma seta limítrofe poder receber mais de um código ICOM ou de um mesmo código ICOM poder aparecer em várias setas limítrofes.

O código ICOM pode, além de identificar setas limítrofes, constituir um **código de interface** (6.1.6-d) ou um **código de seta** (6.1.6-e).

Notação SADT:

A forma gráfica de uma seta limítrofe é de uma seta, com rótulo e código ICOM.



Proposta para o ANA-RE:

Devido à liberdade na rotulação de setas limítrofes no SADT, erros na inserção ou modificação de códigos, de 'layout' ou de rótulos podem gerar relacionamentos corretos do ponto de vista do SADT, porém logicamente inconsistentes. Isto causa grandes dificuldades para manter a consistência de rótulos em uma representação formal de diagramas. Desta forma, esta dissertação propõe uma modificação no componente código ICOM, mantendo conceito, propósito e notação para as setas limítrofes, exceto pela sua rotulação, que deve ser a mesma das interfaces da caixa-pai. Na experiência de automação desenvolvida por Leandro e Nunes [Lean87] [Nune87], todo diagrama-filho herda automaticamente todas as interfaces da atividade-pai com os respectivos rótulos e decorrentes entradas no **glossário** (6.1.4-c).

A conexão explícita entre diagrama-filho e caixa-pai deve ser feita através dos rótulos das setas limítrofes, mantidos no detalhamento, e cujas origens ou destinos são especificados de maneira especial no diagrama-filho em função dos componentes do diagrama-pai.

No caso de o destino ou origem da seta associada à caixa-pai ser uma caixa, a notação do destino da seta limítrofe compõe-se de um colchete esquerdo seguido do número da caixa destino e a notação da origem compõe-se do número da caixa origem seguido de um colchete direito.

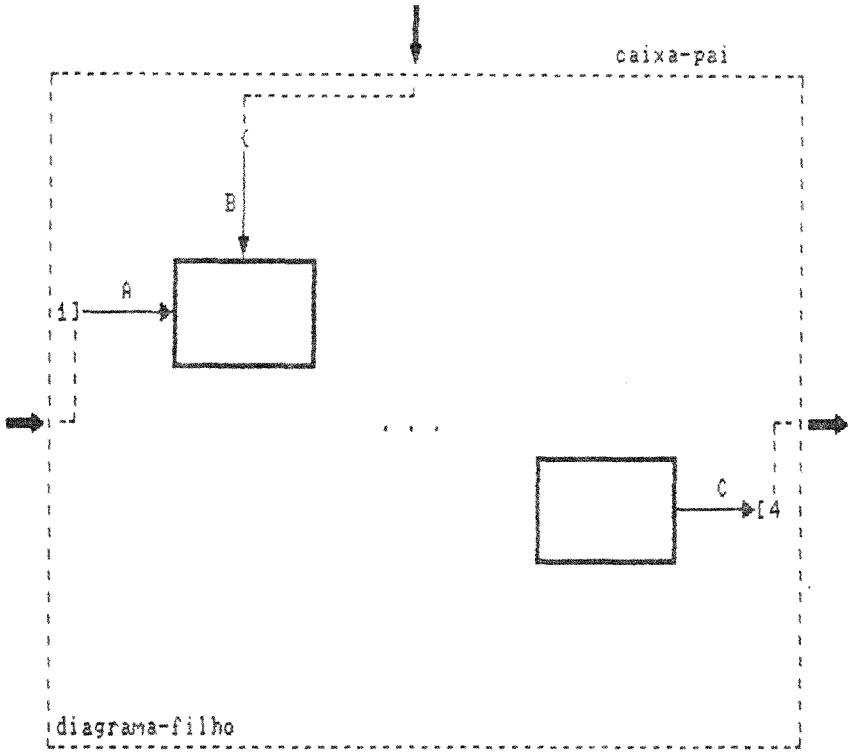
No caso de o destino ou origem da seta associada à caixa-pai ser um desvio, separação ou desvio-ou, a notação para a extremidade livre da seta limítrofe compõe-se de uma chave direita '{'. Para os casos de junção, união ou junção-ou a notação é uma chave esquerda '}'.

Estas modificações permitem a especificação clara no diagrama-filho das origens e destinos das setas limítrofes, além de permitir a verificação de consistência entre diagrama-pai e diagramas-filhos. O código ICOM torna-se, desta maneira, desnecessário para estabelecer a conexão entre caixa-pai e diagrama-filho. Assim sendo, são preservados apenas os demais propósitos e conceitos relacionados a este componente (**código de seta** e **código de interface**) e utilizados para referência. Além disto, devido às modificações propostas, no componente mecanismo, a constituição do código passa a ser apenas I, C ou O.

No ANA-RE o componente código ICOM deixa de existir, sendo substituído por um novo componente: o **código SECI.** Este é formado por uma letra: S (saida), E (entrada), C (controle) e I (interrupção), seguida por um número seqüencial, indicando a ordem de especificação das setas na caixa-pai, de cima para baixo e da esquerda para a direita.

O ANA-RE propõe outros três tipos de destino ou origem: **ente externo**, **memória** e **arquivo** cujas notações são apresentadas posteriormente. Propõe, além disto, o componente semântico **interrupção** (6.2).

Notação ANA-RE:



No diagrama-pai a seta correspondente à seta limítrofe A tem origem na caixa 1; a seta correspondente a B tem origem em um desvio (poderia ser separação ou desvio-ou); e a correspondente à seta C tem destino na caixa 4.

Os códigos SECI associados a A, B e C são, respectivamente, E1, C1 e S1.

6.1.3. Componentes Sintático-semânticos

Um componente sintático-semântico é um componente cuja forma gráfica também se compõe por uma combinação de componentes sintáticos (como nos componentes semânticos), porém tem apenas um único significado associado.

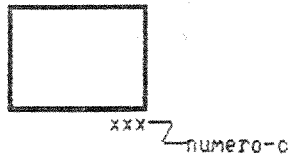
São considerados componentes sintático-semânticos **número-c**, **seta bi-direcional**, **seta mista**, **tunneling**, **nota**, **nota de rodapé**, **meta-nota** e **conector**.

a. Número-c

O componente **número-c** tem como propósito mostrar a existência de um diagrama-filho correspondente a uma determinada caixa, além de permitir que sua respectiva documentação possa ter uma referência única. O **número-c** é um código composto pelas iniciais do autor seguidas por um número seqüencial, que indica a ordem cronológica de criação dos diagramas. Pode ser substituído por um identificador que indique o número da página correspondente à documentação do detalhamento.

O componente número-c também é conhecido como DRE ('detail reference expression') e sua ausência significa a inexistência de detalhamento da caixa.

Notação SADT:



O número-c deve ser posicionado sob o canto inferior da caixa.

No ANA-RE, a constituição adotada é a do identificador único, composto pela letra minúscula 'c' seguida de um número seqüencial único que indica a ordem cronológica de criação.

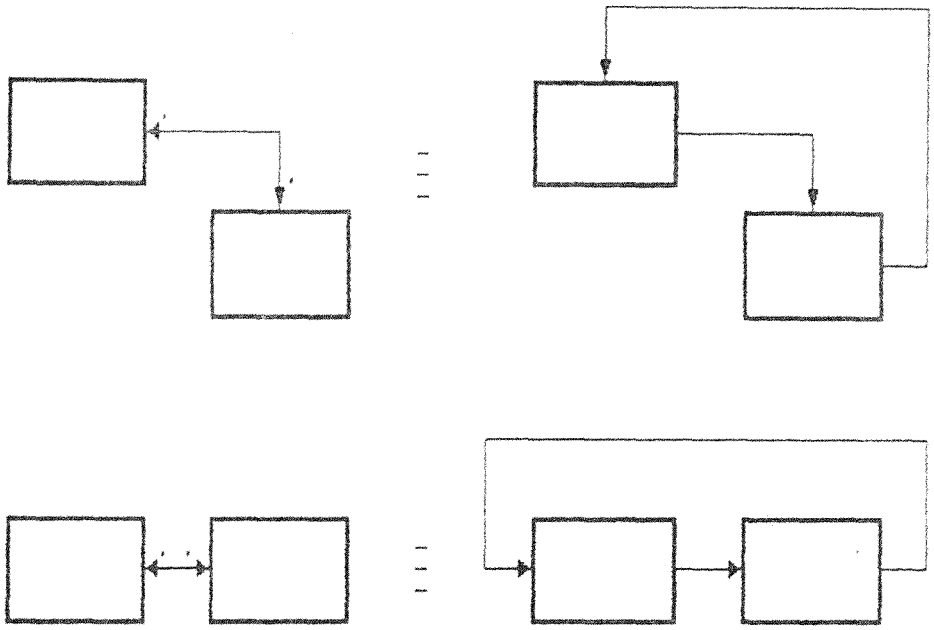
b. Seta bidirecional

O componente **seta bidirecional** tem como propósito mostrar cooperação entre duas caixas, estabelecendo um intercâmbio de responsabilidade que é representado através de uma seta com duas pontas, cada qual acompanhada por um caráter '.'. A caixa em cujo lado direito está a seta bidirecional é a caixa dominante.

As setas bidirecionais não são apenas uma maneira de simplificação da notação. Devem ser utilizadas quando as semânticas dos contextos das caixas são similares ou quando interagem como se cada uma partilhasse uma parte do assunto em questão.

Caso sejam necessários dois rótulos para uma seta bidirecional, os mesmos devem estar separados por um caráter '/', sendo que o primeiro corresponde à seta-saída da caixa dominante.

Notação SADT:



Proposta para o ANA-RE:

O componente seta bidirecional foi eliminado no ANA-RE, uma vez que pode ser substituído pelos componentes seta e observação. Além disto, tal seta pode causar dificuldades de representação quando do detalhamento das caixas envolvidas, pois à seta correspondem 4 códigos ICOM que podem não ser facilmente reconhecidos. Este componente deve ser eliminado pois formalmente não podem existir dois fluxos de dados distintos com mesma identificação, o que ocorre quando a seta bidirecional recebe um único rótulo.

c. Seta mista

O componente **seta mista** tem como propósito suprimir detalhes de intercâmbio quando uma seta unidirecional na caixa-pai necessita ser especificada no diagrama-filho como uma seta bidirecional. Uma **seta mista** é estabelecida por duas **setas** (sendo uma bidirecional) que se unem em suas pontas, estando a seta bidirecional indicada por um ponto.

As setas mistas são apropriadas para entendimento da comunicação a nível do diagrama-pai, onde ocorre relacionamento unidirecional entre caixas, enquanto que nos respectivos diagramas-filhos se verifica que uma cooperação bidirecional é necessária. Um exemplo citado em [Ross77b] é a relação patrão-empregado: o patrão (no diagrama-pai) ordena unidirecionalmente, enquanto que, nos diagramas-filhos patrão e empregado interagem nos dois sentidos.

Notação SADT:

Proposta para o ANA-RE:

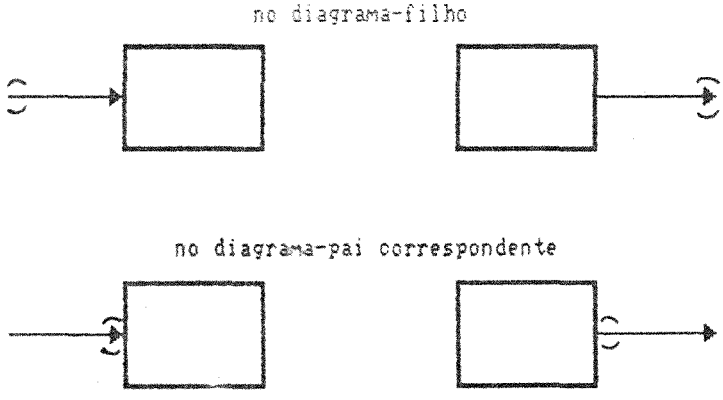
O componente seta mista foi eliminado no ANA-RE devido à não existência de setas bidirecionais e devido ao sentimento de que duas caixas em diagramas distintos não conseguem preencher os requisitos necessários para serem conectadas através de setas bidirecionais.

d. Tunneling

O componente **tunneling** tem como propósito evitar desordem devido à passagem de setas de um diagrama-pai, através dos diagramas-filhos seguintes, quando estas setas são relevantes apenas em um nível distante de detalhamento. O tunneling é estabelecido permitindo-se que setas saiam ou entrem em um diagrama sem necessitarem aparecer no diagrama-pai ou no diagrama-filho imediato. As setas resultantes de tunneling têm a origem ou destino especificados de forma especial.

Apesar de se esperar que a cada seta com indicação de tunneling corresponda uma outra em um nível distante no modelo, nos exemplos do próprio Ross o tunneling só é utilizado para as setas que entram ou saem nos diagramas com nível de detalhamento mais baixo, dando uma conotação de indicar setas que são detalhes irrelevantes ao nível do diagrama-ancestral.

Notação SADT:



Proposta para o ANA-RE:

No ANA-RE o componente relativo ao tunneling desaparece. Isto evita a necessidade de verificar a consistência entre níveis não imediatos de um modelo. Desta forma, diagramas não contêm setas 'passadas' via tunneling por um seu ancestral distante.

Só é possível o aparecimento de seta externa ao diagrama, que não tem correspondente no diagrama-pai, através do uso dos novos componentes **arquivo**, **memória** ou **ente externo** (vide 6.2).

e. Nota

O componente **nota** tem como propósito mostrar comentários necessários, estabelecendo a possibilidade do uso de palavras dentro de um diagrama.

Notação SADT:

NOTE: seguido do comentário a respeito de parte
ou todo o assunto detalhado no diagrama

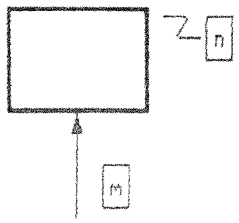
Proposta para o ANA-RE:

O componente nota inexiste no ANA-RE devido à existência do componente **texto** (vide 6.1.4-a), que cobre completamente o propósito e o conceito envolvidos, além de permitir a diminuição de componentes no diagrama, tornando-o mais claro.

f. Nota de Rodapé

O componente **nota de rodapé** tem como propósito evitar excesso de texto no diagrama, quando da rotulação de uma seta, ou ainda servir como indicação da existência de informações adicionais a respeito de setas ou caixas. Uma nota de rodapé é estabelecida através de um número inteiro dentro de um quadrado, que é colocado no local apropriado e indica a existência de informações em outro espaço livre no diagrama, onde esta indicação é repetida (ver exemplo na figura 4 em 3.4).

Notação SADT:



n comentario sobre a caixa

m rotulo da seta que, por exemplo, nao cabia no espaco

Proposta para o ANA-RE:

O componente nota de rodapé inexiste no ANA-RE pois pode ser substituído pela utilização de abreviações nos rótulos de setas ou do componente **observação** (vide 6.2.5).

g. Meta-nota

O componente **meta-nota** mostra a existência de comentários a respeito do diagrama, sendo estabelecida através de um número inteiro dentro de um círculo.

Uma meta-nota presta-se a dar informações a respeito do diagrama e não do assunto abordado no diagrama. Por exemplo, são meta-notas os comentários sobre o trabalho de um autor, sugerindo mudanças de 'layout'.

Notação SADT:

(n) comentário sobre o diagrama e não sobre o assunto do diagrama

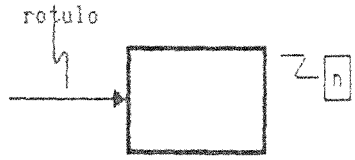
Proposta para o ANA-RE:

O componente meta-nota inexiste no ANA-RE devido ao fato de ser resultado do trabalho de leitores e comentaristas, não devendo fazer parte da linguagem utilizada pelo autor. Desta maneira, o propósito e o conceito deste componente são substituídos pelo novo componente **comentário** (vide 6.2.6).

h. Conector

O componente **conector** tem como propósito assegurar a associação apropriada de palavras no diagrama. Um conector é estabelecido através da associação de rótulos e notas de rodapé ou meta-notas ao assunto em foco via um rabisco.

Notação SADT:



Na linguagem ANA-RE um conector deve ser utilizado apenas quando a associação se fizer necessária e se aplica apenas a setas e rótulos, já que os outros tipos de componentes passíveis de associação via um conector inexistem no ANA-RE.

6.1.4. Componentes Complementares

Um componente complementar visa oferecer maior detalhamento das informações dos diagramas e não faz parte do conjunto de componentes que podem ser especificados em um diagrama, não podendo, portanto, ser considerado um componente gráfico.

São considerados componentes complementares **texto**, **FEO**, **glossário** e **índice** (constantes do núcleo gráfico do SADT) além de **formulário**, **documento** e **regra de ativação**.

a. Texto

O componente **texto** tem como propósito auxiliar o entendimento do contexto detalhado no diagrama, através de um conjunto de parágrafos que, segundo as regras de leitura [Ross77b], deve ser lido somente após o entendimento do diagrama.

O **texto** não tem padronização quanto ao conteúdo, estrutura ou tamanho (ver exemplo 4 em e).

Proposta para o ANA-RE:

Manter o propósito do componente tal qual no SADT. Um diagrama gráfico associado ao texto representa toda a documentação sobre o detalhamento de uma função.

O ANA-RE adota, como regra de padronização, que um **texto** deve ser composto por quatro partes, cada qual responsável pela complementação de informações de um componente do diagrama:

- o conjunto de **observações** (vide 6.2.5) indicadas nos nomes e rótulos cujas setas e caixas têm associado um '*',

- o conjunto de expansões de nomes e rótulos abreviados no diagrama,

- as identificações dos **arquivos** (vide 6.2.3) e **entes externos** (vide 6.2.1) especificados no diagrama como detalhes (primeira especificação no modelo) associadas aos respectivos códigos e a observações, e

- o conjunto de **descrições** (vide 6.2.7) das caixas no diagrama.

Cada uma destas partes tem uma padronização para conteúdo e estrutura (ver exemplo 5 em e).

As partes de um texto devem ser elaboradas pelo autor. E em caso de não serem especificadas, o ambiente SAES deve solicitar as definições.

b. FEO

O componente **FEO** ('for exposition only') tem como propósito ressaltar características que um diagrama deve ter. É utilizado para exposição de comentários e normalmente é gerado por leitores e/ou comentaristas. Um FEO é estabelecido através de um diagrama cuja identificação é a letra F seguida pelo número que compõe o código da caixa-pai que dá origem ao diagrama que se quer comentar.

Proposta para o ANA-RE:

O componente FEO inexiste no ANA-RE. A razão é que o ANA-RE tem o novo componente **comentário** (vide 6.2.6), que incorpora tanto um diagrama quanto um texto livre. Esta estrutura se presta a transmitir o resultado do trabalho de leitores e comentaristas e separa o vocabulário gráfico utilizável pelos autores do utilizável por comentaristas, uma vez, que estes últimos não necessitam seguir nenhuma das regras sintáticas ou semânticas do ANA-RE.

c. Glossário

O componente **glossário** é parte integrante de um modelo. Seu propósito é o de reunir os termos e respectivas definições utilizados nos diagramas e textos de um modelo.

O glossário deve ser a parte final da documentação de um modelo, deve estar ordenado alfabeticamente e suas páginas devem ser numeradas com G1, G2, etc. [SofT76].

Proposta para o ANA-RE:

O componente glossário deve existir em dois níveis distintos: nível de diagrama e nível de modelo. No primeiro caso é chamado glossário parcial, estando associado a determinado diagrama e fazendo parte inclusive de sua documentação, e compõe-se dos termos definidos pelo usuário. No segundo caso é chamado glossário global e é composto por todos os glossários parciais do modelo.

O glossário global impede a existência de entradas iguais originárias de glossários parciais distintos.

d. índice

O componente **índice** é parte da documentação de um modelo, sendo uma tabela de conteúdo da documentação do modelo. É estabelecido através da união de todos os **códigos de nodos** (vide 6.1.6-b) relativos aos diagramas do modelo, respectivos títulos (nomes) e indicação da página onde se encontram.

Proposta para o ANA-RE:

A geração do índice deve ser feita pelo SAES de forma automática, somente quando da solicitação de toda a documentação de um modelo.

e. Formulário

O componente **formulário** não é considerado parte do nucleo gráfico do SADT. É, porém, o padrão utilizado para documentar diagramas SADT. Tem como propósito uniformizar a documentação relativa a diagramas e apresentar informações pertinentes ao diagrama documentado.

Exemplo SADT:

A figura 7 apresenta um **formulário** SADT em branco. Nota-se que existem outras informações que complementam um diagrama, quais sejam:

- identificação da empresa ou setor da empresa ('USED AT'),
- identificação do autor ('AUTHOR'),
- identificação do projeto ('PROJECT'),
- data ('DATE'),
- revisão ('REV'): identifica a revisão devida aos possíveis resultados intermediários do ciclo autor-comentarista,
- número de notas ('NOTES'): indica o total de notas, notas de rodapé e meta-notas existentes no diagrama,

- estado do diagrama: indica em que etapa do processo de análise encontra-se o diagrama: em desenvolvimento ('WORKING'), proposta ('DRAFT'), aprovado ('RECOMMENDED'), ou pronto para publicação ('PUBLICATION'),

- informações sobre leitores e/ou comentaristas ('READER DATE') indicando os envolvidos em revisões e as datas em que ocorreram,

- contexto ('CONTEXT'): indica a posição da caixa-pai dentro do diagrama-pai (assumindo um 'layout' em escada, a caixa-pai é apresentada como um pequeno retangulo e as demais caixas como pequenas formas ovaladas), além do número-c e **código de nodo** (vide 6.1.6-b) do diagrama-pai,

- **código de nodo** ('NODE'): identifica o diagrama a partir do código de nodo do diagrama-pai e do número da caixa da caixa-pai,

- título ('TITLE'): nome da caixa-pai,

- **número-c** ('NUMBER'): número-c sob a caixa-pai.

Estas informações são utilizadas para permitir a navegação orientada através dos diagramas da árvore de detalhamento de um modelo.

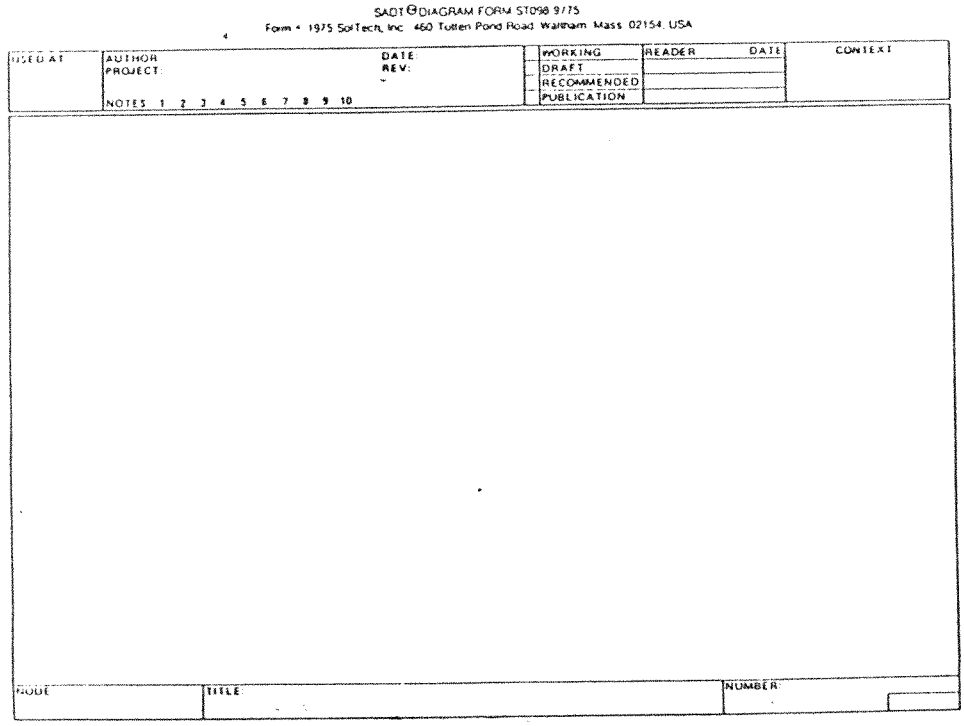


Fig. 7 - Formulário SADT

Proposta para o ANA-RE:

O padrão ANA-RE difere do padrão SADT pela ausência das informações 'data', 'número de notas', 'estado do diagrama', 'informações sobre leitores e/ou comentaristas' e 'contexto'. 'Número de notas' é desnecessário uma vez que notas, notas de rodapé e meta-notas inexistem no ANA-RE. 'Data', 'estado do diagrama' e 'informações sobre leitores e/ou comentaristas' são substituídas respectivamente por 'data de início', 'status' e 'obs'. 'Contexto' inexiste devido à sua inutilidade, uma vez que as informações dele provenientes podem ser conseguidas através do código de nodo e de facilidades do ambiente. O ANA-RE requer as informações 'data p/ revisão' (data limite para entrega do diagrama para os comentaristas), 'data p/ comentários' (data limite dada aos comentaristas para envio de comentários) e data de aprovação.

Caso existam comentários sobre o diagrama, a informação 'COMENTARIOS' aparece no campo 'obs'. Em 'status' aparece uma das seguintes indicações: 'EM DESENVOLVIMENTO', 'PROPOSTA' ou 'APROVADO'; a opção 'PRONTO P/ PUBLICAÇÃO' é desnecessária, uma vez que o desenho do diagrama é automático e não necessita de revisão.

As informações da parte superior do formulário (vide figura a seguir) são solicitadas aos usuários durante o processo de elaboração e ficam disponíveis para consulta junto com o diagrama. No instante da solicitação de emissão de um formulário, o usuário não necessita preencher qualquer campo de informações.

Exemplo ANA-RE:

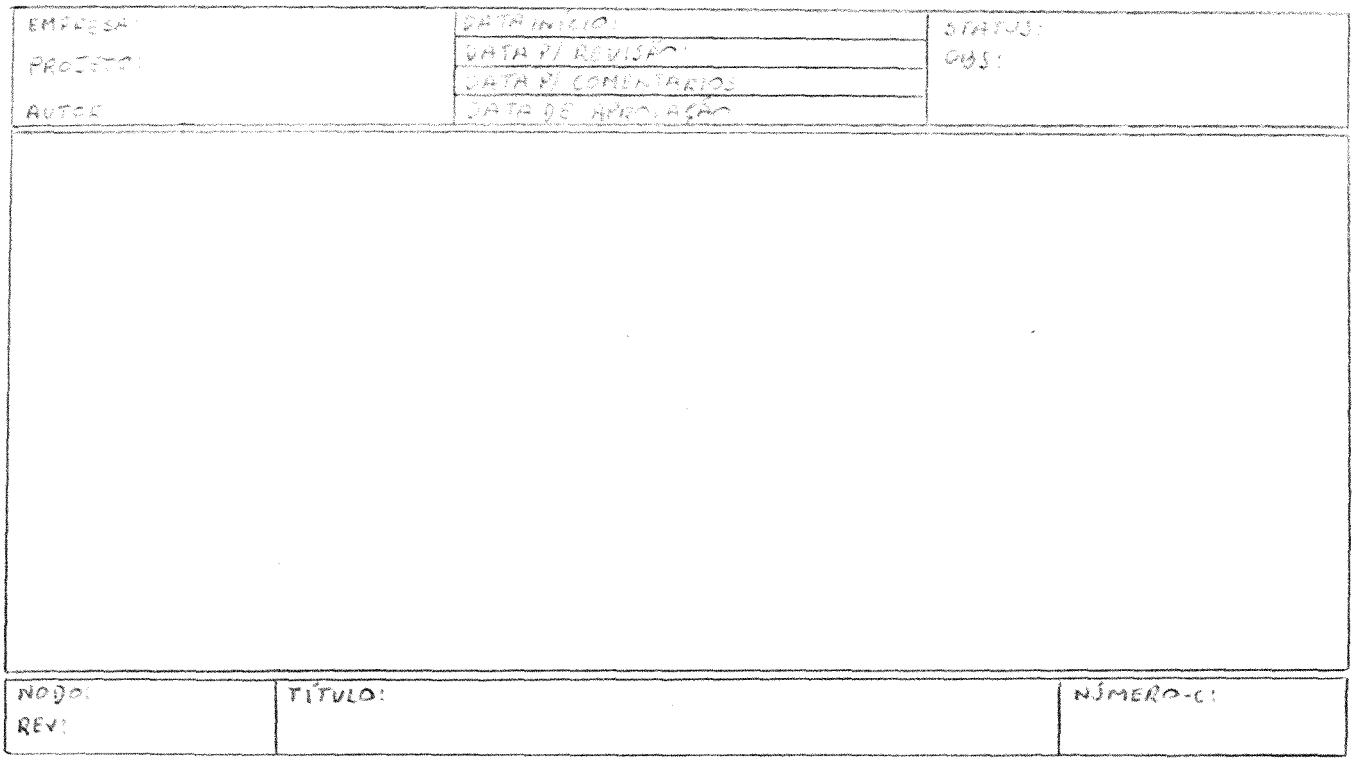


Fig. 8 - Formulário SAES

f. Documento

O componente **documento** não é considerado parte do núcleo gráfico do SADT, porém é o padrão utilizado para associar o componente formulário com o componente texto com vistas à publicação de toda a documentação de um diagrama.

A página com o texto e o diagrama-pai em formato reduzido deve ser colocada como contra-página daquela onde se encontra o formulário SADT com o diagrama (vide figura a seguir, retirada de [SofT76]).

Exemplo SADT:

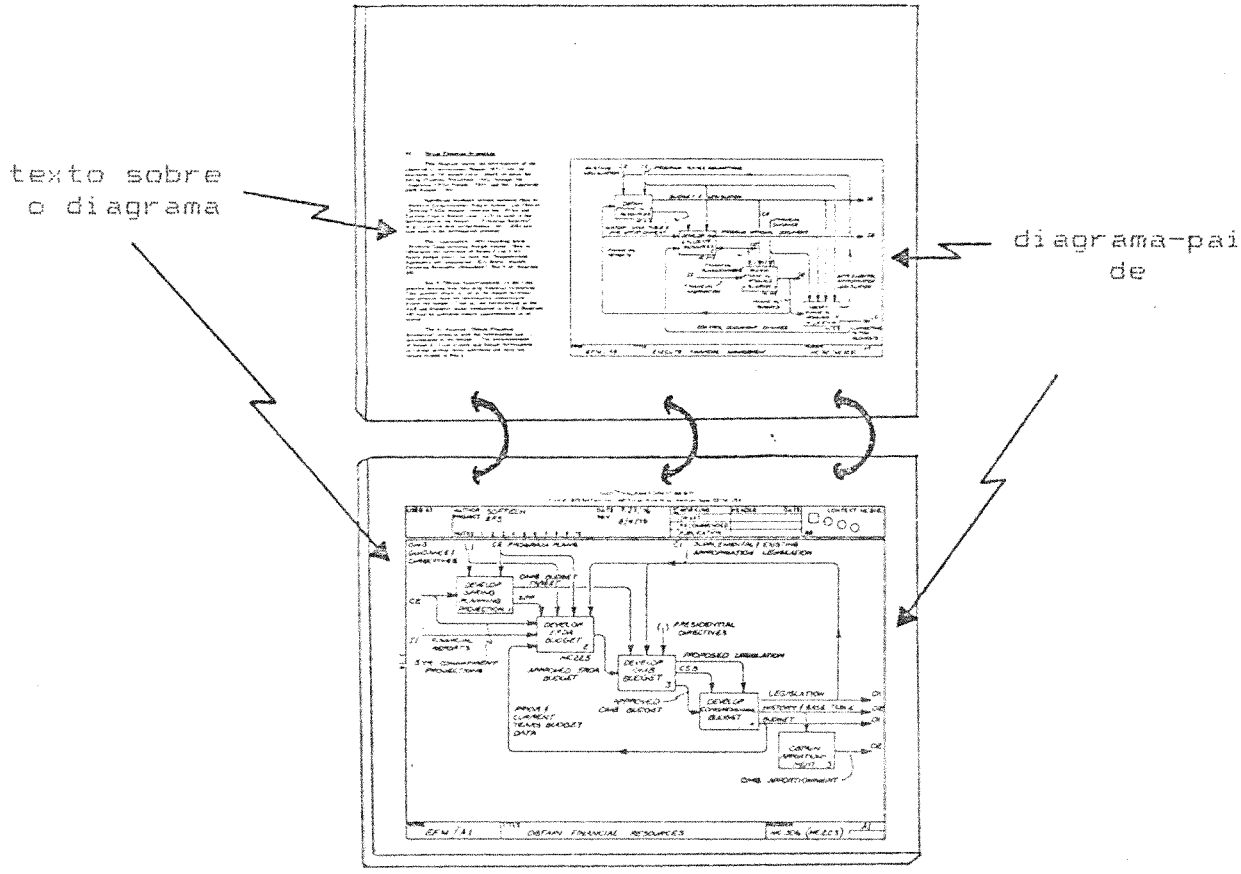


Fig. 9 - Documento SADT

Proposta para o ANA-RE:

Para o ANA-RE não existe uma forma de documento única, uma vez que o usuário pode solicitar a edição em tela ou impressora de qualquer conjunto de informações disponíveis a respeito de um determinado modelo. Por exemplo, podem ser solicitados apenas um diagrama, ou um diagrama e respectivos diagramas ancestrais, ou ainda apenas o texto relativo a uma das caixas de um diagrama. O SAES deve possibilitar ter na tela, em diferentes janelas, as informações que compõem o documento.

g. Regra de Ativação

O componente **regra de ativação** não é considerado parte do núcleo gráfico do SADT. Tem como propósito informar quais combinações de entradas e controles ativam a função representada pela caixa e por conseguinte produzem qual combinação de saídas. Em conjunto representam a potencialidade de paralelismo ou seqüencialidade entre as atividades de um modelo.

O componente **regra de ativação**, aparentemente, se aplica apenas a diagramas de atividades quando o SADT é utilizado como método de especificação na fase de Projeto. [Dick81] apresentam com detalhes este componente.

Notação SADT:

<código da caixa>/<nº da regra>:<pré-condições> -> <pós-condições>

onde <nº da regra> é um número seqüencial

<pré-condições> é uma combinação de rótulos de entradas e controles

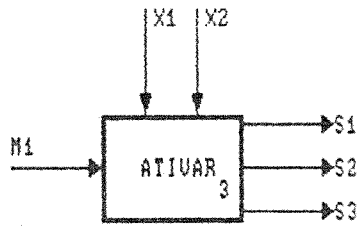
<pós-condições> é uma combinação de rótulos de saídas

As regras de ativação utilizam lógica de primeira ordem e notação de máquinas com estados finitos e são agregadas aos diagramas próximas às caixas a que correspondem [Dick81].

As pré-condições de uma regra de ativação podem especificar a ausência de entradas ou controles através do uso de uma barra horizontal colocada sobre os rótulos apropriados.

Já se as entradas ou controles são especificados com uma barra horizontal colocada sobre cada um dos respectivos rótulos nas pós-condições, então aquelas entradas ou controles são 'consumidos' pela ativação da caixa. Em outras palavras, ficam indisponíveis para a próxima ativação, necessitando ser gerados novamente.

Exemplos:



A ausência de uma regra de ativação é equivalente a ter todas as entradas e controles descritos nas pré-condições e todas as saídas nas pós-condições, ou seja, equivalente à seguinte regra de ativação:

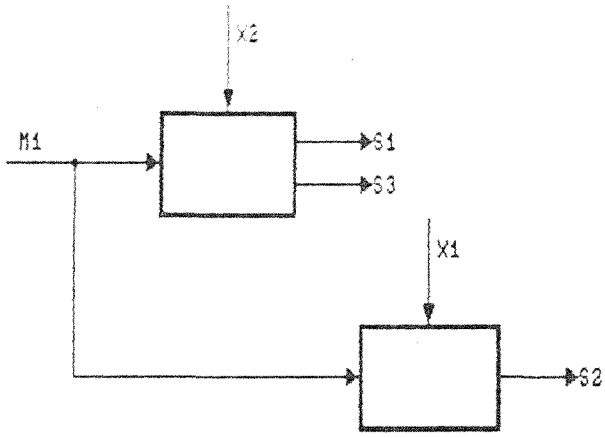
A3/1: M1 X1 X2 -> S1 S2 S3

Se as seguintes regras de ativação fossem agregadas à função ATIVAR:

A3/1: M1 X1 -> S2

A3/2: M1 X2 -> S1 S3

significariam o mesmo que o seguinte diagrama-filho parcial, com ausência de regras de ativação:



Já se a regra de ativação A3/2 for especificada como:

A3/2: M1 $\overline{X1}$ X2 -> S1 S3 $\overline{M1}$

isto significa que a entrada M1 e o controle X2 necessitam estar disponíveis juntamente com a indisponibilidade do controle X1 para ativação da caixa e produção das saídas S1 e S3. Além disto, há a implicação de consumo ou indisponibilidade da entrada M1.

Na literatura disponível não existem considerações com relação à interpretação que deve ser dada a:

- inexistência de uma entrada ou controle nas pré-condições de uma regra de ativação (tanto a interpretação de indisponibilidade como de 'indiferença' podem ser válidas);

- estado das interfaces não especificadas como consumidas nas pós-condições de uma regra de ativação (tanto a interpretação de disponibilidade para uma próxima ativação como de consumo implícito podem ser válidas);

. possibilidade de validação de duas pré-condições de regras de ativação diferentes de uma mesma caixa (são válidas duas interpretações: primeira, a especificação das interfaces deve ser complementar de uma regra em relação às demais, não permitindo a validação de regras de ativação diferentes num mesmo instante; e, segunda, a ordem de especificação das regras deve ser utilizada para a escolha de uma única regra);

Também não existem informações de como é possível verificar a consistência entre regras de ativação da caixa-pai e do respectivo diagrama-filho.

Proposta para o ANA-RE:

Apesar da regra de ativação ser utilizada para a elaboração de modelos na fase de Projeto, a sua utilização na fase de Análise e Especificação de Requisitos auxilia a esclarecer o que é uma entrada, controle ou saída para uma determinada caixa. Em outras palavras, existem situações onde uma interface é especificada para uma caixa, mas na realidade a interface é detalhe dessa caixa e deve ser especificada ao nível do diagrama-filho. É através da análise de quais interfaces contribuem para a ativação da caixa ou resultam dessa ativação que se conclui o nível em que devem ser especificadas corretamente essas interfaces.

Ceri afirma em seu trabalho [Ceri86] de comparação de métodos aplicáveis ao que chama 'Requirements Collection and Analysis':

> 'diagramas de fluxo de dados são fáceis de desenhar e interpretar, no entanto não permitem um entendimento preciso sobre o paralelismo implícito existente em uma simulação da execução das atividades. Isto se deve, em grande parte, ao fato de que a geração de saídas por uma atividade não implica necessariamente no término dessa atividade.'

Uma maneira de corrigir o problema apontado por Ceri é através da utilização de regras de ativação.

A notação proposta é essencialmente a notação SADT, com as seguintes diferenças:

. utilização de códigos SECI ao invés de rótulos nas pré e pós-condições,

. utilização de operadores AND e OR para indicar a lógica associada às interfaces,

. utilização de operadores NOT ao invés da barra horizontal para indicar ausência de entradas ou controles,

. utilização de parênteses para indicação de precedência, e

- utilização de colchetes para indicação de consumo nas pós-condições ao invés da barra horizontal (dentro de um par de colchetes especifica-se o conjunto de códigos SECI consumidos).

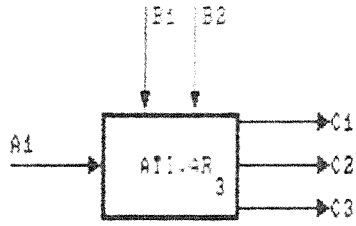
A inexistência de uma entrada ou controle nas pré-condições de uma regra implica que a existência ou não da interface é irrelevante para a avaliação da caixa a partir dessa regra.

As interfaces que não tenham sido especificadas como consumidas continuam com seu estado inalterado.

Não é permitida a especificação de pré-condições em diferentes regras de ativação que possam causar a ativação da caixa num mesmo instante. Por exemplo, as pré-condições **E1 AND C1** e **E1 OR C1** no instante em que **E1** e **C1** ficam disponíveis podem causar a ativação da caixa, podendo, no entanto, ter associadas diferentes pós-condições.

É possível verificar se as regras de ativação em um diagrama correspondem às regras de ativação da caixa-pai [Dick81].

Regras de ativação devem ser especificadas como parte do novo componente **descrição** de uma caixa, ao invés de junto à caixa.



Ao contrário do SADT, a presença de regras de ativação é obrigatória, ou seja, sua ausência não implica na situação 'default' de todas as entradas e controles estarem subentendidos nas pré-condições e de todas as saídas nas pós-condições. Para que essa situação 'default' ocorra, é obrigatória a especificação da seguinte regra de ativação, cada a figura acima:

A3/1: E1 AND C1 AND C2 -> S1 AND S2 AND S3

Note que, ao invés de usar rótulos nas regras, como quer o SADT, o ANA-RE utiliza códigos SECI como forma de evitar que as regras fiquem extensas devido ao tamanho dos rótulos.

A seguinte regra de ativação, utilizada apenas para exemplificar, significa que a ativação da caixa 3 ocorre com a disponibilidade da primeira entrada (A1) em conjunto com a ausência do primeiro controle (B1), resultando na geração da primeira e segunda saídas (C1 e C2), além do consumo da primeira entrada; ou ocorre com a disponibilidade do segundo controle resultando na geração das mesmas saídas e do consumo desse controle:

A3/1: (I1 AND NOT C1) OR C2 -> S1 AND S2 [I1 C2]

6.1.5. Componentes de Orientação

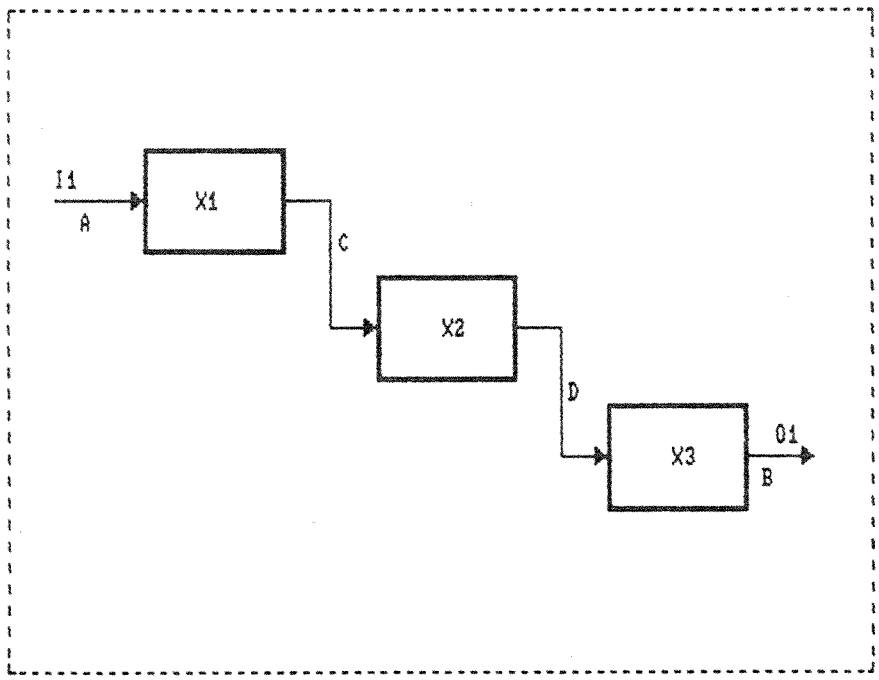
Um componente de orientação visa orientar o processo de elaboração de diagramas. É um princípio que deve ser seguido e, portanto, não pode ser considerado um componente gráfico mas sim deveria fazer parte da técnica DT.

São considerados componentes de orientação no SADT **passagem**, **restrição**, **mostrar relevância**, **seta omitida** e **layout em escada**. Todos estes, exceto **seta omitida**, são preservados no ANA-RE como princípios de orientação.

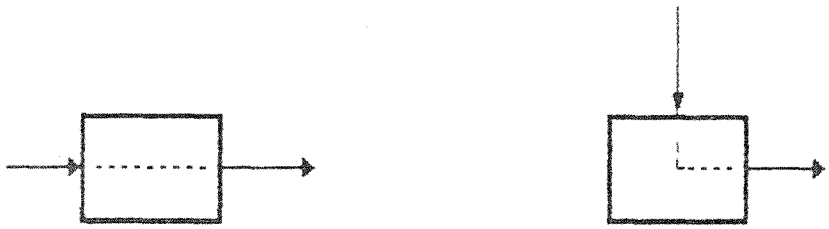
a. Passagem

O componente **passagem** é uma orientação que tem como propósito 'mostrar necessidade'. Seja uma caixa cujas interfaces representem uma transformação. No seu detalhamento, a preservação da transformação é 'necessária' e ocorre apenas se, no diagrama-filho, for estabelecido um encadeamento de entradas e saídas, cuja origem é a entrada da caixa-pai e cujo destino é a respectiva saída.

Exemplo:



Pode-se imaginar uma meta-notação (vide figura a seguir), indicando que no diagrama-filho o fluxo dos dados da entrada para a saída (actigram) ou das transformações do controle para a saída (datagram) deve ser preservado. No exemplo anterior a passagem existe pois a entrada A da atividade X é a origem de um encadeamento de entradas e saídas C e D, no diagrama-filho, cujo destino é a saída B. A figura 4 (vide 3.4) também exemplifica a passagem para diagramas funcionais.



b. Restrição

O componente **restrição** é uma orientação cujo propósito difere para diagramas de atividade e para diagramas de informação. Para os primeiros, o propósito é limitar as transformações de forma a controlar a geração de determinados resultados intermediários, o que é estabelecido pelos controles. Para os segundos, o propósito é impor qual versão de uma informação deve ser gerada.

Segundo [Ross77b], toda caixa funcional em um detalhamento requer ao menos uma restrição (controle) para estar bem definida.

c. Mostrar Relevância e Seta Omitida

O componente **mostrar relevância** é uma orientação que determina que as interfaces consideradas relevantes para o entendimento de um diagrama nele sejam desenhadas. Esta orientação se complementa com o componente seta omitida.

O componente **seta omitida** é uma orientação que tem como propósito omitir o óbvio, no sentido de que entradas óbvias, que um leitor sabe existir, devem ser omitidas dos diagramas de atividade, bem como os controles óbvios dos diagramas de informação. Segundo [Ross77b], esta omissão ajuda o entendimento.

A regra para uso do SADT, com relação ao componente seta omitida, é a de que controles nunca devem ser omitidos em diagramas de atividade, o mesmo ocorrendo com entradas nos diagramas de informação.

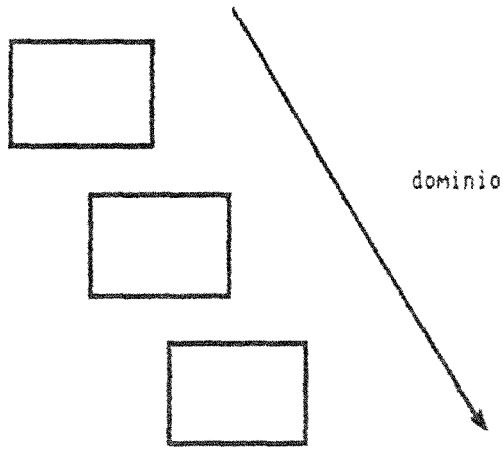
Proposta para o ANA-RE:

Qualquer interface, seja óbvia ou não, deve ser desenhada, pois a possível obviedade é subjetiva, podendo inclusive ser óbvia em um nível de detalhamento de um modelo e não o ser em outro. Acrecente-se o fato de que a automação do ANA-RE prevê os mecanismos de verificação de consistência e de simulação de modelos, que devem portanto ser completos.

d. Layout em Escada

O componente **layout em escada** é uma orientação preservada no ANA-RE que tem como propósito indicar que o contexto de uma caixa tem domínio sobre os contextos das demais. Um layout em escada se estabelece distribuindo as caixas no diagrama de maneira a formarem uma escada cujo 'degrau' mais alto é a caixa com maior domínio.

Exemplo:



6.1.6. Componentes para Referência

Um componente para referência visa permitir a referência a um determinado componente que se encontra especificado em um diagrama, não importando se este faz parte de um modelo externo. Não pode ser considerado um componente gráfico pois é utilizado normalmente nas partes descritivas de um modelo (texto, comentário ou observação) ou para complementar a informação de localização de componentes.

São considerados componentes para referência **identificação de modelo**, **código de nodo**, **ponto**, **código de interface** e **código de seta.**

Para poder referenciar um componente de outro modelo, compõe-se um código a partir da junção da **identificação de modelo** ao **código** do componente separados pelo caráter '!'.

a. Identificação de Modelo

O componente **identificação de modelo** tem como propósito permitir a referência, de forma única, a componentes em multi-modelos. A identificação de modelo não tem regra de formação, mas supõe-se que é composta por um identificador.

Proposta para o ANA-RE:

A identificação de modelo deve ser estabelecida através de uma sigla e de um código de 'release'. A sigla se presta a identificar univocamente o projeto e respectivo modelo. O código de 'release' identifica uma versão completa e coesa do modelo, uma vez que é possível haver variações de um mesmo modelo. Estas variações são decorrentes de atualização visando necessidades da aplicação, variações na configuração de 'hardware' ou desejo de incorporar facilidades na aplicação [Parn76]. Por exemplo, a identificação de modelo **SAES1.0** identifica o release **1.0** de um modelo, provavelmente único, do projeto Sistema de Apoio à Especificação de Sistemas.

b. Código de Nodo

O componente **código de nodo** tem como propósito permitir a referência unívoca de um diagrama em um modelo. O código de nodo é estabelecido através de uma árvore com os números das caixas, representando os níveis de detalhamento.

Enquanto o código de nodo aponta para uma posição única na árvore de detalhamento, o número-c identifica de forma única o diagrama, ou seja, vários diagramas podem ter o mesmo código de nodo, representando diferentes versões do mesmo diagrama. Cada versão, no entanto, tem um número-c único.

Em um modelo de diagramas de atividades, o código de nodo do diagrama A-0 (1º nível) é A0; os códigos de nodos nos diagramas de detalhamento de A0 são A1, A2, A3 etc. Para os demais níveis de detalhamento, este código é formado pelo código de nodo do diagrama-pai e pelo número da caixa-pai. Por exemplo, A313 é o diagrama-atividade relativo à caixa 3 do diagrama A31.

Para um modelo de diagramas de informação o código é o mesmo, havendo apenas a substituição da letra A pela letra D.

c. Ponto

O componente **ponto** mantido pelo ANA-RE tem como propósito permitir que interfaces ou caixas sejam referenciadas com relação a um determinado contexto. O ponto é estabelecido especificando-se o caráter '.' imediatamente após o código de nodo cujo contexto é referenciado. Por exemplo, A122.4I1 significa a primeira entrada da quarta caixa no diagrama A122, enquanto que A1224.I1 significa a seta limítrofe I1 no diagrama A1224; A1224 indica um diagrama e A122.4 indica a quarta caixa no diagrama A122.

d. Código de Interface

O componente **código de interface** tem como propósito permitir referência unívoca às interfaces em um modelo. O código de interface é estabelecido através da junção do código de nodo ao código ICOM correspondente à interface. Por exemplo, A3.4I1 referencia a interface correspondente à primeira entrada da quarta caixa atividade do diagrama A3.

Proposta para o ANA-RE:

Manter o propósito do componente tal qual no SADT. O conceito e a notação sofrem modificações decorrentes das substituições do código ICOM pelo código SECI.

e. Código de Seta

O componente **código de seta** tem como propósito permitir a referência unívoca às setas em um modelo. O código de seta é estabelecido através da junção de dois códigos de interface. Por exemplo, A3.4O1-A3.5C1 referencia a seta correspondente à primeira saída da quarta caixa que se liga ao primeiro controle da quinta caixa no diagrama A3.

Proposta para o ANA-RE:

O componente código de seta inexiste no ANA-RE devido ao fato de sua composição ser um pouco desordenada, criando dificuldades para sua interpretação, e de poder ser substituído pelo componente rótulo, que aceita abreviações.

6.2. A Linguagem Gráfica do ANA-RE: Extensões ao SADT

Além das modificações e eliminação de componentes do SADT descritas na seção 6.1, a dissertação propõe extensões aos componentes do núcleo gráfico do SADT, visando facilitar a automação, aumentar o poder de expressão da linguagem para permitir a especificação de um maior universo de aplicações. Outro objetivo é a solução de alguns dos problemas apontados por diversos autores que avaliaram e estudaram o SADT [Leit87] [Colt84] [Blan83] [Free83b].

Os componentes **ente externo**, **memória** e **arquivo** visam cobrir os detalhes de entrada, saída e de comunicação com o usuário, baseando-se em componentes similares existentes em diagramas DFD [Gane79] [Gane79b].

Os componentes **observação** e **descrição** visam apresentar detalhes padronizados que complementem as informações do diagrama com relação às atividades nele especificadas, sem que isto implique na necessidade de inclusão de frases no diagrama.

O componente **comentário** visa diferenciar os documentos gerados por leitores e comentaristas daqueles gerados por autores. Em termos funcionais, substitui o componente **FEO**.

O componente **interrupção** visa cobrir detalhes de estrutura de controle e permitir o tratamento de problemas de tempo-real.

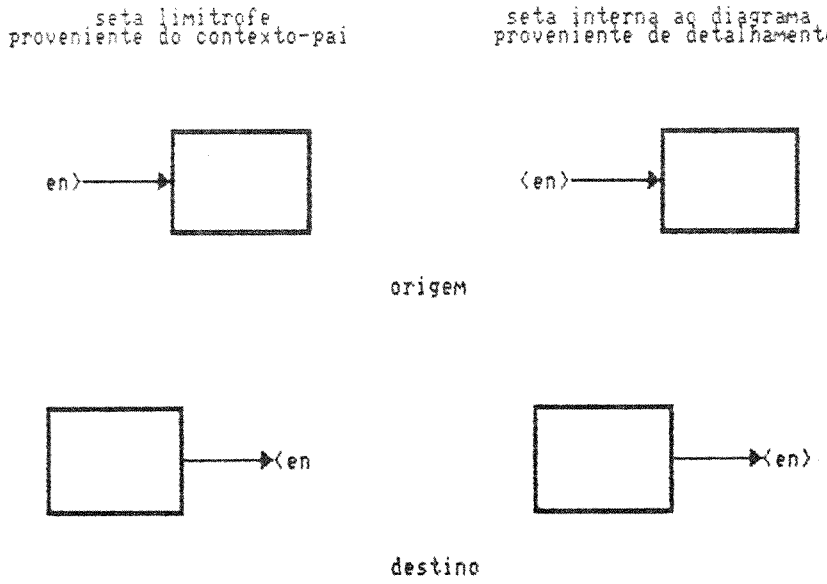
Considerando os agrupamentos de componentes utilizados em 6.1, as extensões seriam incorporadas da seguinte maneira: **ente externo**, **memória** e **arquivo** no grupo dos componentes sintático-semânticos; **interrupção** no grupo dos componentes semânticos; e **observação**, **comentário** e **descrição** no grupo dos componentes complementares.

6.2.1. Ente Externo

O componente **ente externo** tem como propósito identificar a origem ou destino de dados provenientes ou destinados a entes externos ao modelo funcional, como é o caso de interfaces para usuários ou equipamentos de controle de processos.

Todo ente externo tem uma identificação e um código associados. O código é a especificação do ente externo que aparece no diagrama e é constituído pela letra minuscula 'e' seguida por um número sequencial único para todo o modelo. Caso o ente externo não tenha sido especificado no diagrama como origem ou destino de uma seta limítrofe, a identificação deve constar do componente texto e estar associada ao respectivo código, acompanhada de observações contendo, por exemplo, pseudônimos e relacionamentos.

Notação proposta:

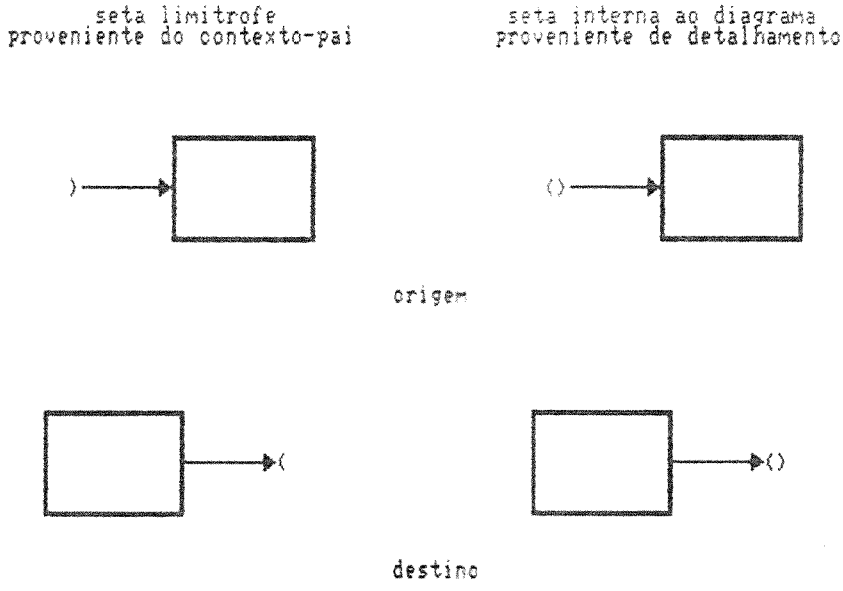


6.2.2. Memória

O componente **memória** tem como propósito permitir o armazenamento temporário de informações pertinentes às atividades de um modelo. Sua utilização evita a proliferação de setas entre atividades de diversos níveis. Pode ser comparada a um depósito para dados de tipos e/ou estruturas diversas, que, uma vez gerados por atividades do modelo, permanecem disponíveis para servirem como entrada ou controle para as demais atividades desse modelo. Este depósito de dados é volátil com relação à simulação ou avaliação do modelo.

Qualquer dado cuja origem seja memória e que seja proveniente do processo de detalhamento necessita ter sido gerado anteriormente, ou seja, em termos de ordem de ativação, a atividade geradora do dado tem sua ativação ocorrendo anteriormente à da atividade consumidora. Este fato só é verificável através de simulação ou avaliação do modelo.

Notação proposta:

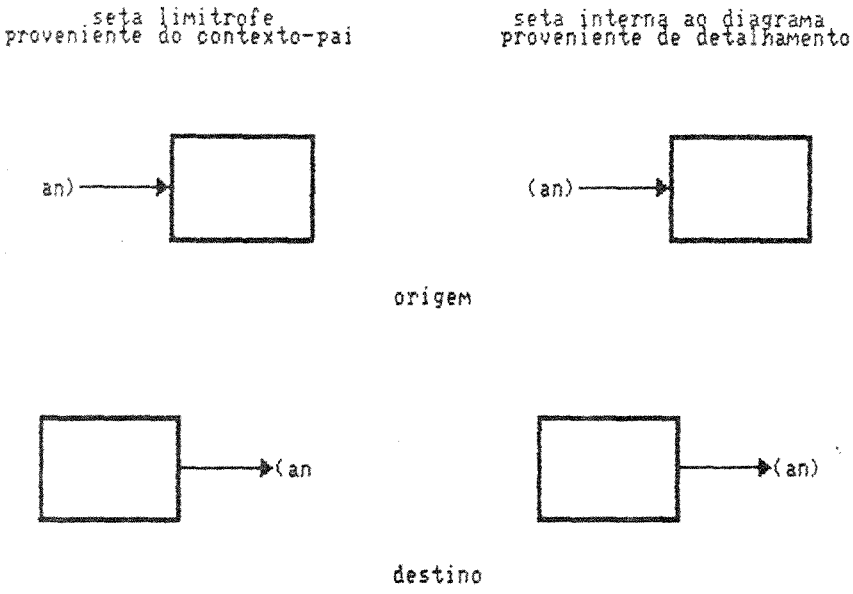


6.2.3. Arquivo

O componente **arquivo** tem como propósito permitir o armazenamento de informações pertinentes às atividades de um modelo. Pode ser comparado a um depósito para dados relacionados ou de mesmo tipo. Tal depósito de dados não é volátil e pode servir como interface de comunicação entre modelos distintos ou entre simulações ou avaliações de um mesmo modelo.

Todo arquivo tem uma identificação e um código associados. O código é a especificação do arquivo que aparece no diagrama e é constituído pela letra minuscula 'a' seguida por um número sequencial único para todo o modelo. Caso o arquivo não tenha sido especificado no diagrama como origem ou destino de uma seta limítrofe, sua identificação deve constar do componente texto e estar associada ao respectivo código. Dentre as informações que podem ser especificadas nas observações sobre arquivos destacam-se finalidade, conteúdo e características.

Notação proposta:



6.2.4. Interrupção

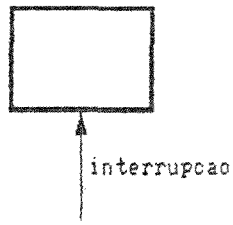
O componente **interrupção** tem como propósito representar a existência de uma interrupção que, supondo a simulação ou avaliação do modelo, pode suspender a avaliação de uma atividade. Permite a modelagem de problemas com características de tempo real.

A origem de uma interrupção é a saída de alguma caixa ou de algum ente externo, ou seja, a interrupção e sua origem constituem uma seta.

Uma atividade previamente suspensa através de uma interrupção pode ser reativada através de um controle, que deve constar de uma das pré-condições das respectivas regras de ativação (vide 6.1.4-g).

A origem da interrupção deve, após a suspensão da avaliação da atividade, reativar a atividade diretamente ou através de outra atividade, que por exemplo pode ter dado tratamento à interrupção.

Notação proposta:



6.2.5. Observação

O componente **observação** tem como propósito permitir a complementação das informações sobre caixas e setas. O conceito do componente mecanismo, quando indica 'como' uma função deve ser implementada, deve ser especificado através de uma observação. A observação também pode substituir o componente nota de rodapé.

Notação proposta:

> o caráter '*' no canto superior direito de uma caixa ou imediatamente após o rótulo de uma seta, indica a existência de uma observação no componente texto do diagrama

No texto, na parte relativa às observações, deve existir uma observação identificada pelo nome ou pelo rótulo respectivamente da caixa ou seta com indicação de existência de observação.

6.2.6. Comentário

O componente **comentário** tem como propósito mostrar a existência de comentários a respeito do diagrama. Um comentário é o documento elaborado por leitores ou comentaristas quando da crítica de um diagrama. Substitui os componentes meta-nota e FED. Um comentário, na realidade, é um formulário com duas partes: um espaço para um diagrama e um espaço para texto livre.

O diagrama e o texto são livres no sentido de não necessitarem ser padronizados, estruturados ou ter alguma sintaxe ou semântica pré-estabelecida, se prestando apenas para exposição de idéias.

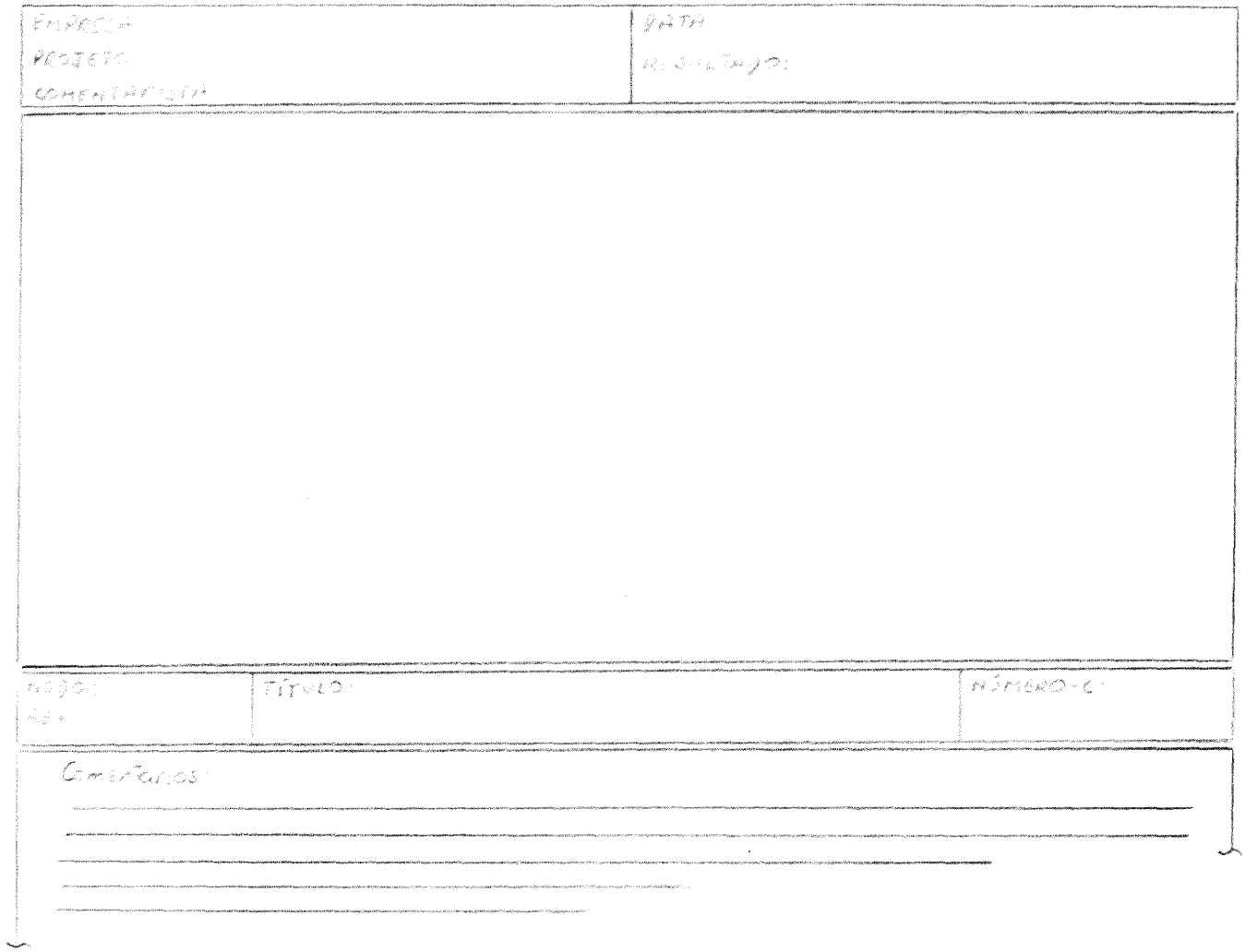
Exemplo SAES:

As informações que leitores ou comentaristas necessitam preencher em um comentário são:

- data de envio do comentário ao autor (DATA),
- resultado da revisão (RESULTADO), indicando APROVADO (sem comentários ou com pequenas modificações), APROVADO SOB RESTRIÇÕES ou DEVOLVIDO P/ REAVALIAÇÃO,
- diagrama ou desenho livre (espaço em branco),
- texto.

As demais informações são geradas automaticamente pelo SAES a pedido de um bibliotecário com o intuito de montar uma cópia do 'kit' para comentários:

- identificação da empresa ou setor da empresa (EMPRESA),
- identificação do projeto (PROJETO),
- identificação do comentarista (COMENTARISTA),
- código de nodo (NODO),
- revisão (REV),
- título (TÍTULO), e
- número-O (NÚMERO-O).



6.2.7. Descrição

O componente **descrição** tem como propósito prover detalhes de cada uma das funções de um diagrama e das respectivas interfaces associadas. Uma descrição é parte de uma caixa no que diz respeito ao detalhamento de informações relativas à atividade associada.

Uma descrição é dividida nas seguintes partes:

- introdução: utilizada para especificar o objetivo da atividade além de observações adicionais,
- detalhes sobre saídas, entradas, controles e interrupções: utilizado para descrever o fluxo de dados, com seu objetivo e estruturas de dados que o compõem,
- restrições: utilizada para especificar as restrições que se aplicam à atividade,
- regras de ativação, e
- exceções: utilizada para especificar as exceções que podem ocorrer durante uma avaliação da caixa.

Uma descrição deve obrigatoriamente ser constituída pela introdução e pelas regras de ativação, bem como detalhes das saídas, entradas, controles e interrupções que não constituam setas limítrofes. As demais partes são opcionais.

Se alguma informação existente no diagrama for mencionada nas partes de introdução, restrições ou exceções, pode-se utilizar o seu respectivo código de interface ou o seu rótulo. Este deve ser colocado entre dois carateres especiais que indicam o tipo SECI da interface: '/' para saída, '%' para entrada, '*' para controle e '#' para interrupção. Esta forma de distinção contribuí para o rápido e consistente entendimento da descrição [Hest81] [Heni80].

As regras de ativação devem envolver todas as interfaces associadas a entradas, saídas e controles.

As exceções devem ser parte importante de uma descrição [Hest80] e são relativas aos eventos inesperados que podem ser causados por interrupções, dados inválidos (entradas) ou mau funcionamento da atividade.

Segue-se o exemplo de uma descrição:

Descrição (INICIAR-SESSÃO)

.Introdução: INICIAR_SESSÃO tem por função validar o acesso para a utilização do SAES (código de usuário e possível senha).

.Saídas:

S1: restrições de acesso relativas ao usuário

S2: sinalização de requisição de entrada ao usuário

.Entradas:

E1: código de usuário e opcionalmente senha

E2: arquivo de usuários do SAES, senhas e suas características

.Restrições:

1. o único tipo de mensagem tratável por A1 é o código/senha

.Regras de Ativação:

A1/1: E1 AND E2 -> S1 AND S2

.Exceções:

1. inexistência do arquivo de usuários

6.3. Comparação de Componentes ANA-RE e SADT

O ANA-RE possui 31 componentes em comparação com os 40 do SADT. Foram modificados ou eliminados 24 componentes do SADT e criados 7 novos componentes.

São mantidos com mesmo propósito, notação e conceito os seguintes componentes **caixa**, **seta**, **interface**, **desvio**, **junção**, **união**, **separação**, **número-c**, **conector**, **índice**, **passagem**, **restrição**, **mostrar relevância**, **layout em escada**, **código de nodo** e **ponto**.

A substituição dos componentes **mecanismo** e **chamada** decorre da necessidade de evitar a ambiguidade de notação existente com interfaces, de forma a viabilizar a automação. Estes componentes são substituíveis ou pelo novo componente observação, se indicam 'como' uma atividade deve ser levada a cabo, ou pelo componente número-c, se indicam a existência de detalhamento em outro ou no mesmo modelo.

Os componentes **nome** e **rótulo** tiveram sua notação modificada para permitir abreviações.

Os componentes **desvio-ou**, **junção-ou**, **desvio**, **junção**, **união** e **separação** tiveram sua notação unificada para simplificar a linguagem e receberam orientações sintáticas para permitir a verificação de consistência.

Os componentes **seta limítrofe**, **tunneling**, **identificação do modelo**, **texto**, **glossário** e **regra de ativação** foram modificados visando permitir maior entendimento do assunto especificado através da complementação de informações, padronização de conteúdo e estrutura.

A necessidade de ranter componentes e conceitos coerentes com o restante da linguagem e eliminar inconsistências existentes resultou na modificação dos componentes código ICOM, código de interface, formulário e documento. O código ICOM, substituido pelo código SECI, é utilizado no SADT para estabelecer as conexões entre diagramas de níveis imediatos (o que é feito no ANA-RE através dos rótulos das setas limítrofes) e para referências de setas e interfaces. O código SECI incorpora as saídas, entradas e controles do código ICOM, porém substitui mecanismos por interrupções; é utilizado nas especificações das regras de ativação (o que é feito através de rótulos no SADT) e na referência de setas e interfaces.

A redundância, falta de utilidade prática ou conflito entre componentes são os fatores considerados para a eliminação dos componentes **chamada**, **seta omitida**, **para-todos** e **de-todos**, **seta bidirecional**, **seta mista**, **nota**, **nota de rodapé**, **meta nota**, **código de seta** e **FEO**.

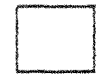
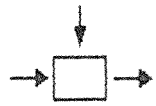
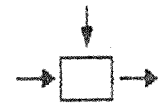
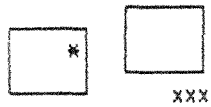
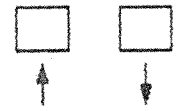
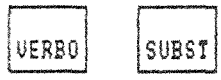
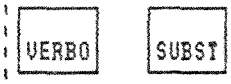
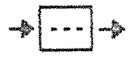
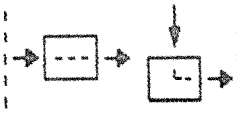
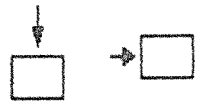
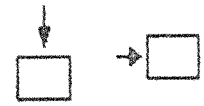
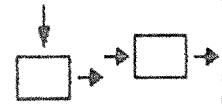
Ente externo, **memória** e **arquivo** são novos componentes que visam permitir a representação de detalhes a respeito de entradas, saídas e comunicação com o usuário, baseando-se em componentes similares existentes em diagramas DFD [Gane79] [Gane79b].

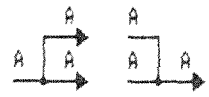
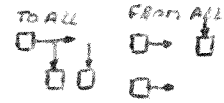
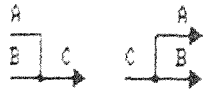
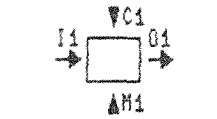
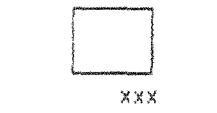
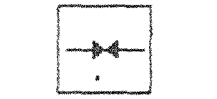
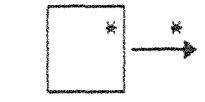
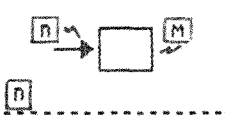
Os componentes **observação** e **descrição** visam apresentar detalhes padronizados que complementem as informações de um diagrama com relação às atividades nele especificadas, sem que isto implique na necessidade de inclusão de frases no diagrama.

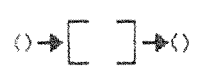
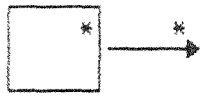
O componente **comentário** visa diferenciar os documentos gerados por leitores e comentaristas daqueles gerados por autores e evita que cada um destes atores tenha uma linguagem própria para poder especificar o mesmo assunto como ocorre na duplicação de componentes do SADT.

O componente **interrupção** visa cobrir detalhes de estrutura de controle e permitir o tratamento de problemas de tempo-real.

A tabulação seguinte apresenta uma comparação entre os componentes das linguagens utilizadas no ANA-RE e no SADT:

	Aspecto ANA-RE	Notacao ANA-RE	Aspecto SADT	Notacao SADT
1	caixa		caixa	
2	seta		seta	
3	interface		interface	
4	substituido por observacao ou numero-o		mecanismo e chamada	
5	nome		nome	
6	rotulo		rotulo	
7	passagem		passagem	
8	restricao		restricao	
9	nao tem similar		seta omitida	

	Aspecto ANA-RE	Notacao ANA-RE	Aspecto SADT	Notacao SADT
10	desvio e juncao		desvio e juncao	
11	desvio e juncao		para-todos e de-todos	
12	uniao e separacao		uniao e separacao	
13	desvio-ou e juncao-ou		desvio-ou e juncao-ou	
14	seta limitrofe e codigo SECI		seta limitrofe e codigo ICOM	
15	numero-c		numero-c	
16	nao tem similar		seta bi-direcional	
17	nao tem similar		seta mista	
18	nao tem similar		tunneling	
19	substituido por texto		nota	NOTE:
20	substituido por observacao		nota de rodape	
21	substituido por comentario		meta-nota	
22	conector		conector	
23	codigo de nodo	A13 D13	codigo de nodo	A13 D13
24	identificacao de modelo	SIGLA'RELEASE'	identificacao de modelo	NOME
25	codigo de interface	A1.3codigoSECI	codigo de interface	A1.3codigoICOM

	Aspecto ANA-RE	Notacao ANA-RE	Aspecto SADT	Notacao SADT
26	nao tem similar		codigo de seta	A1.301-A1.413
27	ponto	A122.4E1	ponto	A122.4I1
28	layout em escada		layout em escada	
29	texto		texto	
30	substituido por comentario		FEO	
31	glossario		glossario	
32	indice		indice	
33	formulario		formulario	
34	documento		documento	
35	regra de ativacao	A3/2: E1 -> S1	regra de ativacao	A3/2:ENT -> SAI
36	ente externo		nao tem similar	
37	memoria		nao tem similar	
38	observacao		substituido por mecanismo chamada e nota de rodape	
39	comentario		substituido por meta-nota e FEO	
40	arquivo		nao tem similar	
41	interrupcao		nao tem similar	
42	descricao		nao tem similar	

6.4. A Técnica de Desenvolvimento e o Modelo ANA-RE

A técnica DT do ANA-RE é baseada no SADT no que diz respeito a existência de autores, comentaristas, leitores, bibliotecários e gerente do projeto. Além disto, utiliza os mesmos preceitos na geração do 'kit' pelo autor, na sua distribuição pelo bibliotecário e na revisão de comentaristas e leitores. O ciclo e a disponibilidade dos documentos gerados sofrem modificações conforme o que se segue.

Dentro do prazo pré-estabelecido pelas chefias, os documentos-crítica (que não são constituídos pelas cópias do 'kit' original mas sim por comentários) são enviados ao autor, que deve esclarecer e responder às críticas apontadas, gerando um único documento com as respostas a todos os comentaristas. A chefia imediata deve acompanhar o processo e certificar-se de que as críticas recebam tratamento adequado. O ciclo de elaboração e revisão pode precisar ser repetido várias vezes.

O término ou reinício do ciclo é de responsabilidade das chefias. Eventualmente estas podem marcar reuniões entre comentaristas e autores devido à grande quantidade de críticas recebidas. Ao invés de o autor gerar um documento com as respostas às críticas, é elaborada uma ata da reunião com a agenda e resumo dos resultados alcançados.

Através do SAES todas as informações estão disponíveis às pessoas que têm acesso ao sistema. Desta maneira, especificações e revisões de todos os ciclos podem ser lidas como um todo, com a finalidade de troca de experiências, gerência e acompanhamento de projetos.

Ao conjunto de orientações para a elaboração de diagramas em um modelo foram anexados os componentes considerados de orientação (vide 6.1.5) que no SADT são erroneamente apresentados como componentes gráficos.

Uma análise a respeito da elaboração dos modelos funcional e de dados, complementares, revela que os diagramas funcionais não têm obrigatoriamente diagramas de informação correspondentes a partir do nível A0 e vice-versa. Em outras palavras, somente o diagrama A-0 tem no diagrama D-0 um seu dual, no sentido de serem expressão do mesmo assunto com enfoques distintos.

Apesar de em [SofT76] serem encontradas observações de que não há correspondência um-para-um entre 'actigrams' e 'datagrams', no restante da literatura sobre SADT prevalece a idéia de complementação a nível de diagrama, o que não é possível de se obter a partir do detalhamento de dois contextos independentes, um com enfoque em atividades e o outro com enfoque em dados (é o que ocorre se considerarmos os fluxos de dados em um 'actigram', que podem ser mais que 6, e as caixas-informação em um 'datagram', que podem ser no máximo 6).

Adicione-se a isto o fato de, praticamente, inexistirem exemplos de 'datagrams' na literatura, a qual se baseia exclusivamente nos 'actigrams' para apresentar o SADT. O método IDEF [SofT81], por exemplo, utiliza um outro método (o IDEF1), que não é baseado no SADT, para modelar informações.

Como resultado da experiência adquirida no uso do SADT para modelar sistemas de software, propõe-se que não seja feito um modelo inteiro com enfoque na informação. Ao invés disto, sugere-se que a cada diagrama de atividade corresponda um de informação que formalmente não necessita se relacionar com os demais 'datagrams' do modelo.

Desta maneira, um 'kit' elaborado por autores deve se constituir de: 'actigram', correspondente 'datagram', texto e glossário parcial.

O 'datagram' do método ANA-RE tem como característica complementar a análise do problema a nível de diagrama (ao invés de a nível de modelo). Sua especificação é opcional e a ele não se aplicam restrições sintáticas ou semânticas.

As informações existentes nos textos (observações, expansões de nomes e rótulos, identificações e observações de arquivos e entes externos, e descrições) em conjunto com os glossários parciais podem ser vistas pelo usuário do SAES como compondo o que Gane e Sarson [Gane79] denominam o dicionário de dados, o qual, segundo esses autores, deve ser composto fisicamente por informações sobre elementos de dados, estruturas de dados, fluxos de dados, arquivos, processos e entidades externas.

Algumas informações que no dicionário de dados devem ser inseridas e mantidas pelo usuário, no SAES são mantidas automaticamente (por exemplo, a origem e destino de fluxos de dados ou a informação de quais diagramas contêm determinado fluxo de dados). Estas informações e aquelas inseridas pelo usuário podem ser consultadas em conjunto ou isoladamente.

7. BANCO DE DADOS PARA SUPORTE DO SAES

Como visto na revisão bibliográfica do capítulo 2, a automação de métodos para especificação e análise de requisitos (ou ambientes que suportem estas atividades) se baseia no uso de banco de dados.

Esta dissertação mostra a viabilidade do suporte automatizado ao ANA-RE através do projeto de um banco de dados para o SAES, com a implementação de seu protótipo.

A primeira parte deste capítulo apresenta o projeto deste banco de dados, para suporte à manipulação de modelos ANA-RE, cuja implementação resulta no núcleo do ambiente SAES.

A segunda parte descreve decisões adotadas e características resultantes da implementação do protótipo. O anexo 2 contém exemplos de uma sessão para definição de um modelo SADT utilizando o protótipo.

A terceira parte discute aspectos sobre a integridade dos elementos no banco de dados. O anexo 3 enumera as exceções consideradas pela implementação e que representam, juntamente com as decisões de projeto, o grau de integridade atualmente oferecido pelo protótipo.

7.1. A Modelagem

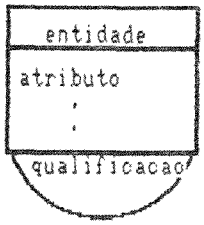
A modelagem do banco de dados do SAES é apresentada através da notação introduzida pela técnica OMT ('Object Modeling Technique') [Blah88] e por uma adaptação da técnica EMC ('Entity Model Clustering') [Feld86].

A técnica OMT é uma variação da modelagem E-R [Chen76] no que diz respeito a possibilidade de se estender a modelagem para suportar outros tipos de abstrações de dados dentro do padrão orientado a objetos. Permite a especificação de entidades e relacionamentos (chamados associações na OMT), além de relacionamentos de agregação e de generalização.

A técnica EMC permite que a modelagem E-R se faça de forma estruturada em níveis hierárquicos de detalhamento, de maneira a evitar que em um único diagrama sejam concentrados muitos detalhes.

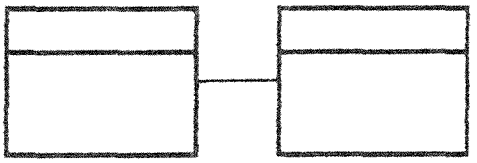
Nos diagramas deste capítulo, muitos detalhes são auto-explicativos, sendo, portanto, omitidos. Após cada diagrama, são feitas observações sobre domínios de alguns atributos, restrições sobre os objetos e suas funções. As regras de formação de valores de alguns atributos (por exemplo, rótulos de setas) já foram definidas anteriormente e não são reapresentadas.

As figuras a seguir introduzem a notação gráfica utilizada para a descrição do banco de dados (constituída dos diagramas das seções 7.1.1 a 7.1.9). Os diagramas são descritos textualmente para permitir seu entendimento por pessoas não familiarizadas com modelagem gráfica de bancos de dados.

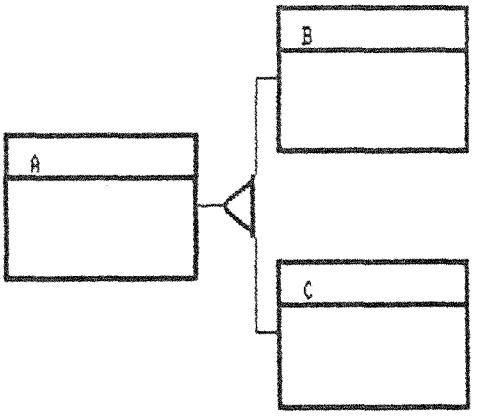


entidade (qualificacao que e a chave de identificacao opcional)

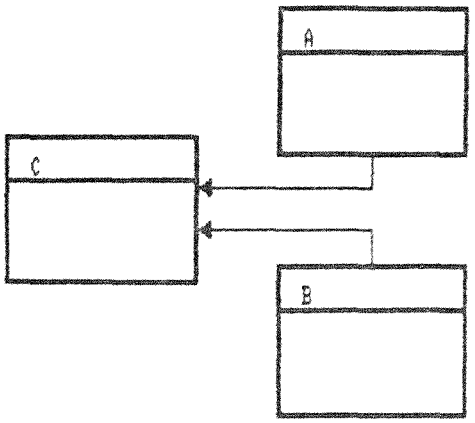
entidade cujo detalhamento encontra-se em outro diagrama



associacao (uma entidade relaciona-se com a outra)



generalizacao (B e C sao subclasses de A)

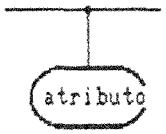


agregacao (A e B sao partes de C)

Considerando a multiplicidade na extremidade esquerda de um relacionamento qualquer (associacao, generalizacao ou agregacao):

muitos (zero ou mais)

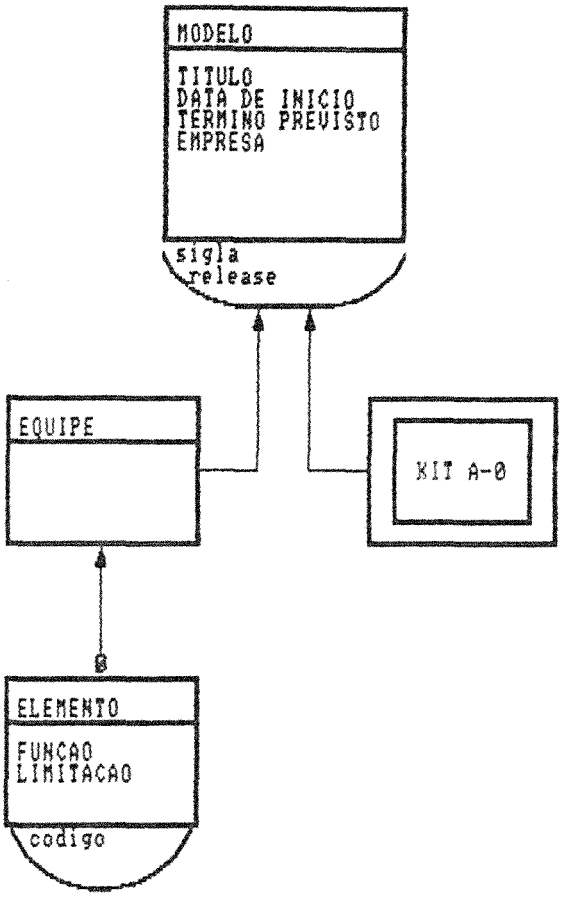
zero ou um



atributo de relacionamento

Obs.: adotou-se como simplificacao que apenas os relacionamentos do tipo associacao sao rotulados.

7.1.1. Modelo



O diagrama indica que **MODELO** é uma entidade resultado da agregação das entidades **EQUIPE** e **KIT A-0** (cujo detalhamento está em outro diagrama, no caso 7.1.3). A uma **EQUIPE** estão associados zero ou mais ocorrências da entidade **ELEMENTO.**

Foi definida a chave de identificação **Release** na entidade **MODELO** que serve para diferenciar variantes de um modelo decorrentes de atualizações. **Release** e versão têm conceitos semelhantes, porém aplicam-se a coisas distintas: um modelo tem diferentes releases e um diagrama tem diferentes versões. Isto implica que os diagramas que compõem um release de um modelo não necessitam ter a mesma versão.

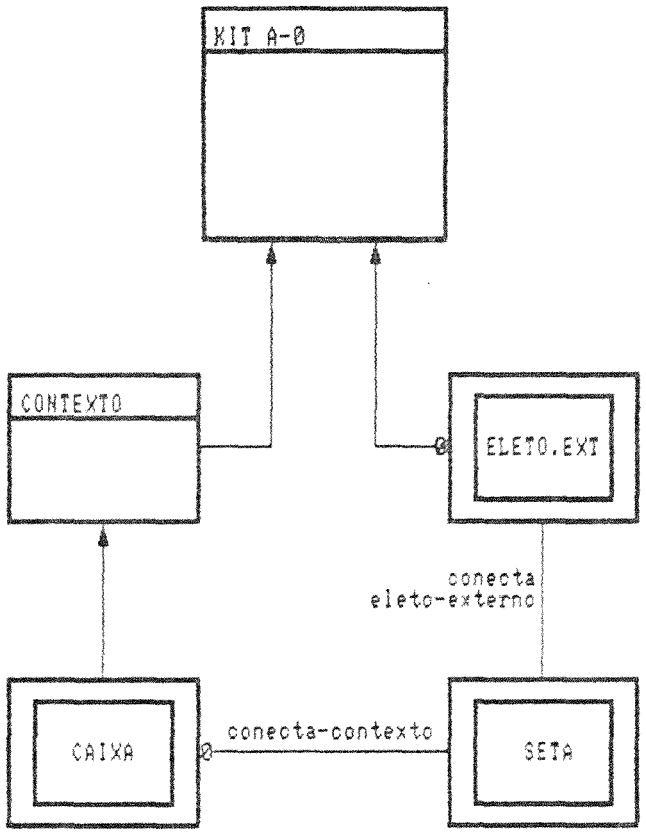
A chave de identificação **código** da entidade **ELEMENTO** permite que um usuário participe de várias equipes, sendo que em cada uma das equipes pode participar tendo mais de uma **FUNÇÃO.**

O atributo **FUNÇÃO** da entidade **ELEMENTO** tem como domínio [GERENTE, COORDENADOR, AUTOR, COMENTARISTA, LEITOR, BIBLIOTECÁRIO e combinações]. Em cada equipe devem existir apenas um gerente e um coordenador.

O atributo **LIMITAÇÃO** da entidade **ELEMENTO** tem como domínio [CÓPIA A TODO O MODELO, CÓPIA LIMITADA A UM CONJUNTO DE NÍVEIS (inicial-final), SÓ CONSULTA AO MODELO, CONSULTA UM CONJUNTO DE NÍVEIS (inicial-final)]. Ao gerente e ao coordenador não se aplicam limitações, como também aos usuários privilegiados. As limitações se aplicam aos 'kits' do modelo que foram aprovados e incorporados ao mesmo; um autor pode criar um diagrama em qualquer nível, desde que a respectiva caixa-pai tenha número-c associado, exceção feita ao nível 0 (a incorporação ao modelo, no entanto, é privilégio do coordenador).

Os diagramas que seguem completam a especificação do banco de dados sendo acompanhados por comentários.

7.1.2. Kit A-0

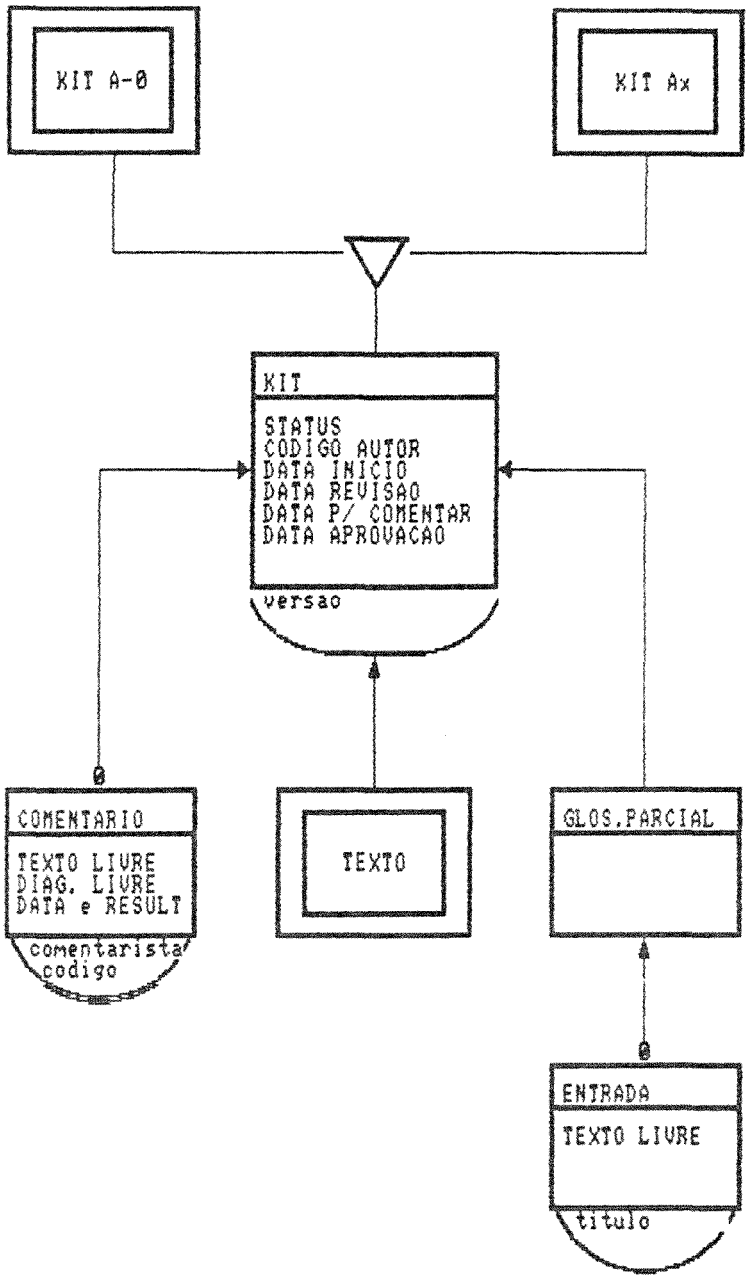


O diagrama indica que **KIT A-0** é uma entidade resultado da agregação das entidades **CONTEXTO** e **ELEMENTO EXTERNO** (cujo detalhamento está em outro diagrama, no caso 7.1.9). A um **KIT A-0** podem estar associados zero ou mais ocorrências da entidade **ELEMENTO EXTERNO** e uma única da entidade **CONTEXTO.**

O **CONTEXTO** de um **KIT A-0**, por sua vez, é composto por uma única ocorrência da entidade **CAIXA** (item 7.1.5) que se associa a ocorrências da entidade **SETA** (item 7.1.6) e cada uma destes a uma ocorrência da entidade **ELEMENTO EXTERNO.** Isto significa que um **KIT A-0** tem uma caixa que pode ter interfaces conectado-a a elementos externos.

Através da instância de **CAIXA**, um **KIT A-0** se relaciona com os kits que compõem seu detalhamento.

7.1.3. Kit



O diagrama indica que **KIT** é uma entidade resultado da agregação das entidades **COMENTARIO**, **TEXTO** (item 7.1.7) e **GLOSSARIO PARCIAL** . A um **KIT** podem estar agregadas zero ou mais ocorrências da entidade **COMENTARIO**, uma única da entidade **TEXTO** e uma única da entidade **GLOSSARIO PARCIAL.**

KIT é uma generalização das entidades **KIT A-0** (item 7.1.2) e **KIT Ax** (item 7.1.4), ou seja, sua abstração é herdada pelas duas outras entidades.

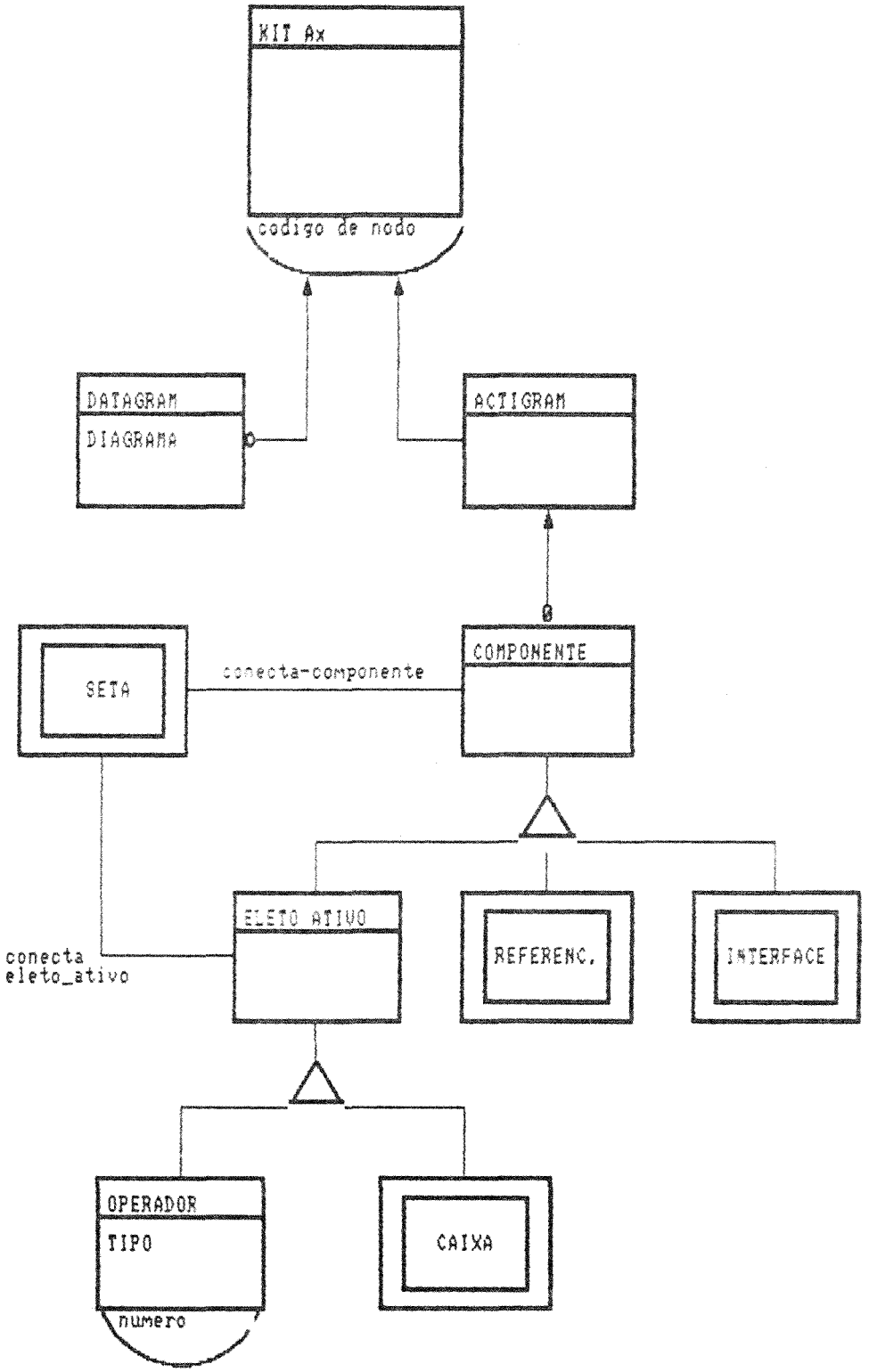
A chave de identificação **versão** da entidade **KIT** distingue instâncias diferentes devidas ao ciclo autor-comentarista e pertencentes ao mesmo modelo. Constituí o campo REV do componente **formulário** (item 6.1.4-e).

O atributo **CÓDIGO DO AUTOR** da entidade **KIT** contém o valor de uma chave de identificação para uma das ocorrências da entidade **ELEMENTO** (item 7.1.1). Os demais atributos de **KIT** seguem a definição dada para os campos do componente **formulário.**

A chave de identificação **comentarista código** para as ocorrências da entidade **COMENTÁRIO** também são valores de chaves de identificação para ocorrências da entidade **ELEMENTO.** Note-se que pode existir apenas um comentário de um determinado comentarista para cada **versão** de um **KIT.** Os atributos de **COMENTÁRIO** seguem a definição dada para os campos do componente **comentário** (item 6.2.6).

A um **GLOSSÁRIO PARCIAL** estão associados zero ou mais ocorrências da entidade **ENTRADA**, cuja chave de identificação é o seu **título** (item 6.1.4-c).

7.1.4. Kit Ax


KIT Ax
codigo de nodo
DATAGRAM
DIAGRAMA
ACTIGRAM
SETA
conecta-componente
COMPONENTE
ELETO ATIVO
REFERENC.
INTERFACE
conecta
eleto_ativo
OPERADOR
TIPO
numero
CAIXA

O diagrama indica que **KIT Ax** é uma entidade resultado da agregação das entidades **DATAGRAM** e **ACTIGRAM**. A um **KIT** pode estar agregada zero ou uma ocorrência da entidade **DATAGRAM** e uma única da entidade **ACTIGRAM**.

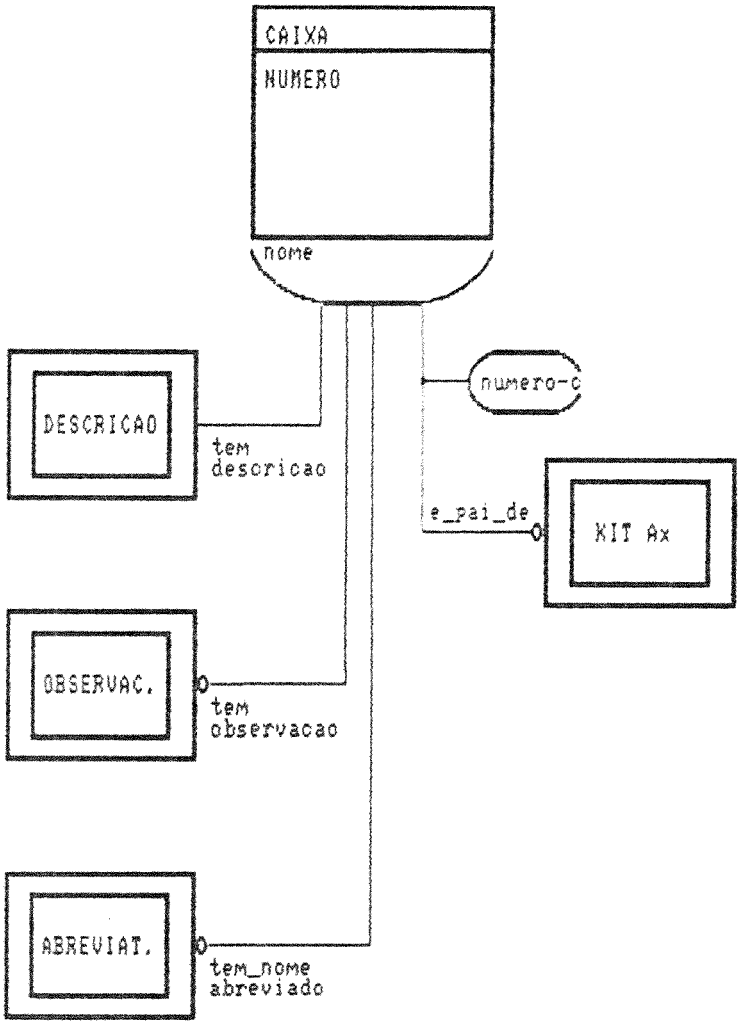
A entidade **ACTIGRAM**, por sua vez, é constituída por zero ou mais ocorrências da entidade **COMPONENTE**, que é uma generalização para as entidades **ELEMENTO ATIVO**, **REFERÊNCIA** (item 7.1.8) e **INTERFACE** (item 7.1.9).

A entidade **ELEMENTO ATIVO** é uma generalização para as entidades **OPERADOR** e **CAIXA** (item 7.1.5).

Uma ocorrência da entidade **SETA** (item 7.1.6) está associada a uma ocorência da entidade **COMPONENTE** (operador, caixa, referência ou interface) e a uma ocorrência da entidade **ELEMENTO ATIVO** (operador ou caixa). Isto significa que uma seta necessita estar conectando ou um operador a outro componente, ou uma caixa a outro componente (não pode conectar uma referência a uma interface, por exemplo).

Um **OPERADOR** é a representação das ramificações que podem ocorrer com setas em um diagrama. Seu atributo **TIPO** indica UNIÃO, SEPARAÇÃO, DESVIO, JUNÇÃO, DESVIO-OU ou JUNÇÃO-OU. A sua chave de identificação é um número único no diagrama a ser gerado e controlado pelo sistema.

7.1.5. Caixa



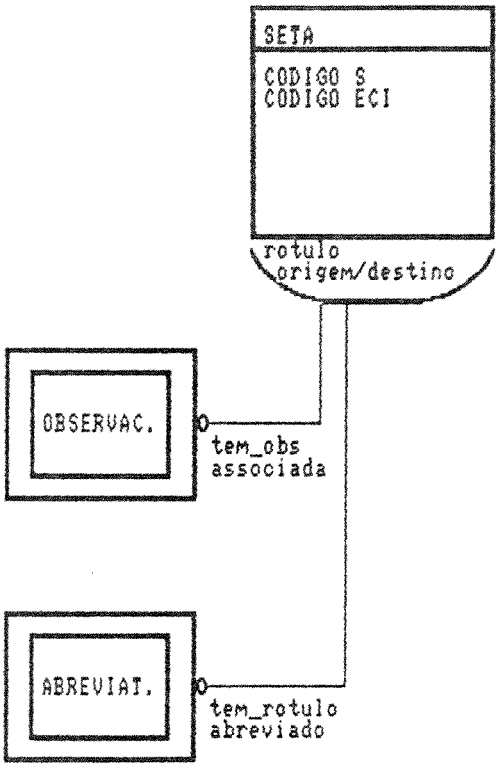
O diagrama indica que **CAIXA** é uma entidade resultado da agregação das entidades **DESCRIÇÃO**, **OBSERVAÇÃO** e **ABREVIATURA**. A uma **CAIXA** pode estar agregada zero ou uma ocorrência da entidade **ABREVIATURA**, zero ou uma ocorrência da entidade **OBSERVAÇÃO** e uma única da entidade **DESCRIÇÃO**.

Uma **CAIXA** deve ter sua **DESCRIÇÃO** e pode, opcionalmente, ter uma **OBSERVAÇÃO** e uma **ABREVIATURA** associada.

A chave de identificação **nome** para a entidade **CAIXA** indica que em um modelo não pode existir duas caixas com mesma identificação, permitindo com isso a utilização do aspecto chamada através do componente **número-c** (item 6.1.3-a).

O detalhamento de uma caixa é representado através do relacionamento tipo associação **é-pai-de**, que tem o atributo **número-c.** Note-se que este relacionamento pode não existir.

7.1.6. Seta



O diagrama indica que **SETA** é uma entidade resultado da agregação das entidades **OBSERVAÇÃO** e **ABREVIATURA.** A uma **SETA** pode estar agregada zero ou uma ocorrência da entidade **ABREVIATURA** e zero ou uma ocorrência da entidade **OBSERVAÇÃO.**

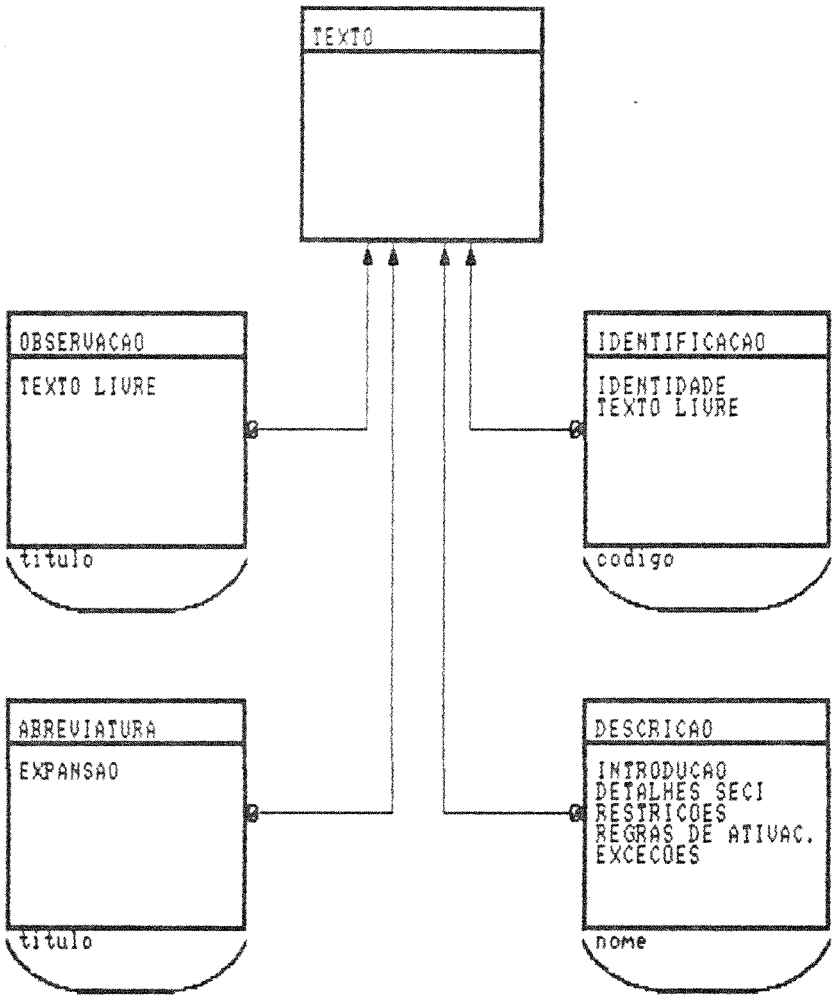
Uma conexão poderia ser representada apenas como um relacionamento de associação entre componentes de um diagrama, no entanto, devido a existência de seus atributos e relacionamentos com outras entidades (**OBSERVAÇÃO** e **ABREVIATURA**), optou-se por implementá-la como uma entidade, simplificando, desta maneira, a modelagem e a implementação.

O atributo **CóDIGO S** da entidade **SETA** recebe o código SECI (saída) correspondente, caso a origem seja uma caixa. O atributo **CóDIGO ECI** recebe o código SECI (entrada, controle ou interrupção), caso o destino seja uma caixa.

Origens ou destinos de uma **SETA**, caso sejam caixas, podem ter no máximo 4 setas associadas com o mesmo código SECI. Toda caixa em um detalhamento deve ter ao menos duas **SETAS** associadas, uma com código SECI igual a controle ou entrada e outra indicando saída.

As chaves de identificação **origem** e **destino** da entidade **SETA** contêm um código de arquivo (**an**), um código de ente externo (**en**), indicação de memoria (m), de caixa referência (**rn**), de operador (**on**) ou de caixa (**n**). As duas complementam **rótulo** devido à possibilidade de existir diversas setas com mesmo rótulo, resultado de ramificações.

7.1.7. Texto/Observação/Abreviatura/Identificação/Descrição



O diagrama indica que **TEXTO** é uma entidade resultado da agregação das entidades **OBSERVAÇÃO**, **ABREVIATURA**, **IDENTIFICAÇÃO** e **DESCRIÇÃO**. A um **TEXTO** podem estar agregadas zero ou mais ocorrências da entidade **OBSERVAÇÃO**, zero ou mais ocorrências da entidade **ABREVIATURA**, zero ou mais ocorrências da entidade **IDENTIFICAÇÃO** e zero ou mais ocorrências da entidade **DESCRIÇÃO**.

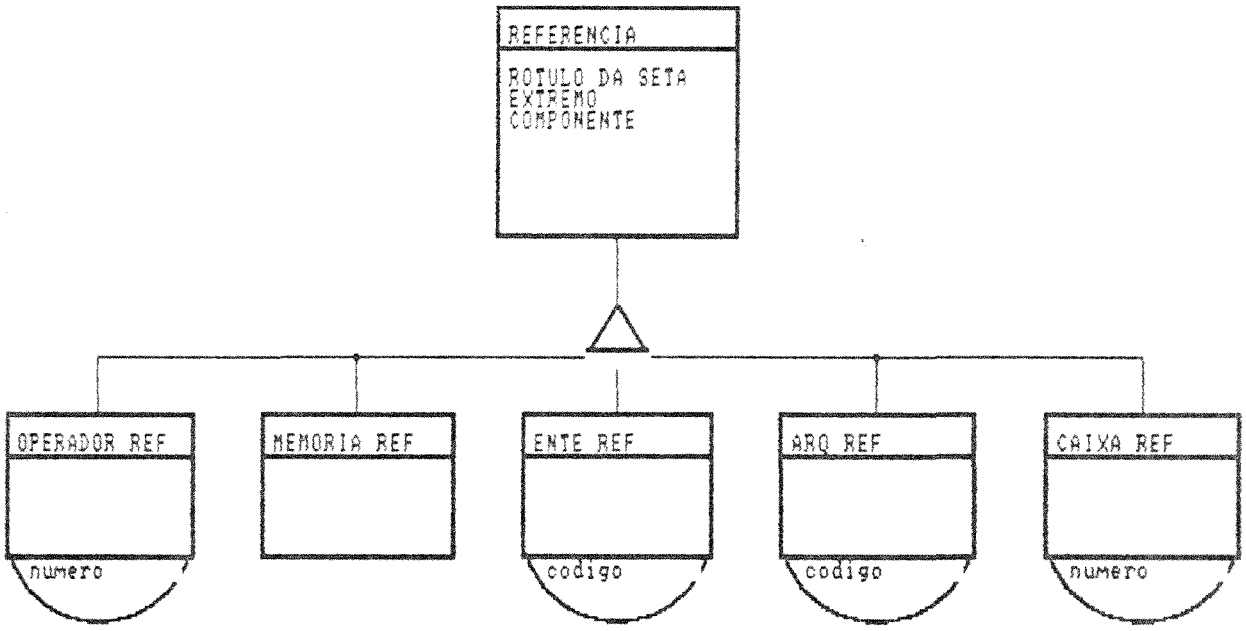
Um **TEXTO** é, na realidade, um conjunto de apontadores para ocorrências de outras entidades. Não necessita ter estrutura própria, uma vez que pode ser criada automaticamente no momento que se fizer necessária, não ocupando espaço desnecessário e não duplicando informações.

A chave de identificação **título** das entidades **OBSERVAÇÃO** e **ABREVIATURA** é constituída pelo nome ou rótulo do componente a que se aplica.

A chave de identificação **código** da entidade **IDENTIFICAÇÃO** é constituído pelo código do arquivo ou ente externo correspondente.

A chave de identificação **nome** da entidade **DESCRIÇÃO** é constituída pelo nome da caixa do actigram correspondente.

7.1.8. Referência

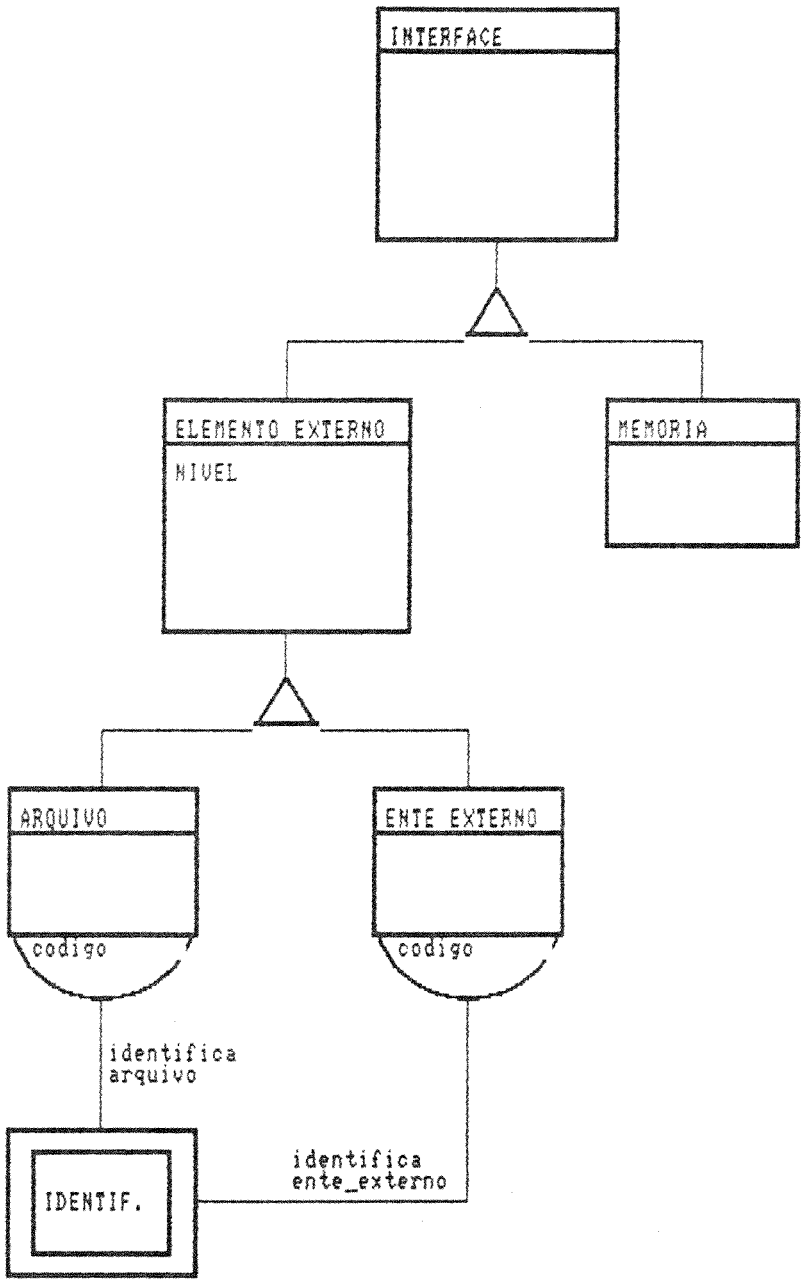


A entidade **REFERÊNCIA** é uma generalização das entidades **OPERADOR REFERÊNCIA**, **MEMÓRIA REFERÊNCIA**, **ENTE REFERÊNCIA**, **ARQUIVO REFERÊNCIA** e **CAIXA REFERÊNCIA**, ou seja, sua abstração é herdada pelas outras entidades.

A entidade **REFERÊNCIA** foi definida para isolar a manipulação das **setas limítrofes** (item 6.1.2-f) do usuário e, portanto, simplifica a implementação dos mecanismos de consistência de um diagrama.

As ocorrências da entidade **REFERÊNCIA** têm no atributo **rótulo da seta** o valor do rótulo correspondente à seta associada à caixa-pai. Somente ao ser conectada a outro componente é que uma instância do tipo **SETA** é criada no diagrama-filho. Este tipo de componente só pode ser removido quando o diagrama-filho inteiro é removido.

7.1.9. Interface/Elemento Externo



A entidade **INTERFACE** é uma generalização das entidades **ELEMENTO EXTERNO** e **MEMÓRIA**, ou seja, sua abstração é herdada pelas duas outras entidades.

ELEMENTO EXTERNO, por sua vez, é uma generalização das entidades **ARQUIVO** e **ENTE EXTERNO**.

O atributo **NíVEL** da entidade **ELEMENTO EXTERNO** foi definido para permitir que remoção ocorra apenas no diagrama em que ocorreu a respectiva criação.

Como as ocorrências das entidades **ARQUIVO** e **ENTE EXTERNO** são representados por códigos no diagrama, existe o relacionamento tipo associação para com suas identificações, que constituem ocorrências da entidade **IDENTIFICAÇÃO** (item 7.1.7)

7.2. Decisões de Projeto e Implementação

As decisões que originaram o projeto da seção 7.1 e sua implementação são apresentadas a seguir.

7.2.1. Decisões Gerais

Devido à extensão do ambiente SAES e conseqüente necessidade de esforço para desenvolvimento e diversidade de tópicos envolvidos (por exemplo, banco de dados, segurança, gerência de configuração, correio eletrônico, linguagem gráfica e gerência de recursos) optou-se por restringir a modelagem do sistema a ser implementado ao estritamente necessário para se suportar a definição de actigrams de modelos. Com isto, foi possível demonstrar a possibilidade da automação da linguagem ANA-RE e, conseqüentemente, do SADT.

Foi escolhida a linguagem SMALLTALK, suportada pelo ambiente Smalltalk/V da Digitalk [Digi86] e acessível para micros PC compatíveis (XT/AT). A razão da escolha foi o fato de ser baseada no paradigma de objetos, que permite, de forma natural, a implementação direta do projeto do banco de dados. Devido a esta decisão, não foi possível suportar a característica de ambiente multi-usuário proposta para o SAES.

A implementação resultou em, aproximadamente, 2000 linhas de código SMALLTALK, nas quais constata-se uma forte reutilização de software no que diz respeito a estruturas de dados, interface com o usuário e primitivas (conversão, seleção, iteração, etc).

Os seguintes aspectos definidos na modelagem do banco de dados não foram implementados como medida de simplificação, tendo em vista não contribuírem para a validação do ANA-RE.

Elementos não implementados:

- entidade equipe no diagrama Modelo;
- entidades comentário, texto e glossário parcial no diagrama Kit;
- atributo versão da entidade Kit no diagrama Kit;
- entidade datagrama no diagrama Kit Ax;

. entidades descrição e abreviatura nos diagramas Caixa e Texto; e

. entidade abreviatura nos diagramas Seta e Texto.

Funcionalidades omitidas:

. backup de modelos (salvamento e recuperação em dispositivos externos);

. consistência dos dados fornecidos pelo usuário (como datas, sigla, rótulos, nomes);

. geração de formulário e índice; e

. abreviação de nome e rótulo e conseqüente expansão.

7.2.2. Características da Implementação

O SMALLTALK provê abstração de dados, o que permitiu reutilização de software e intrinsicamente ofereceu o conceito de relacionamento do tipo generalização, através das classes e subclasses.

A interface implementada utiliza menus que permitem a definição de um modelo, tendo como base a identificação única dos diversos componentes em um actigram que podem ser interligados através de setas. O uso de menus tem a vantagem de permitir que o protótipo seja utilizado por usuários não especializados, em oposição a uma linguagem de especificação de diagramas.

A visualização da estrutura dos diagramas é possível através da exibição dos identificadores dos componentes e de símbolos que indicam os respectivos contextos. O resultado pode ser considerado como uma imagem do actigram descrito em uma linguagem declarativa, que, como mencionado anteriormente, prejudica muito o entendimento se comparada a uma interface gráfica. Como exemplo da simbologia adotada temos:

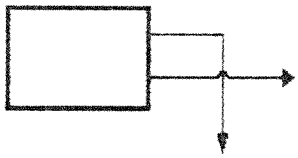
. () envolvem operadores que são seguidos pelo tipo;

. <> envolvem entes-externos que são seguidos pela identificação;

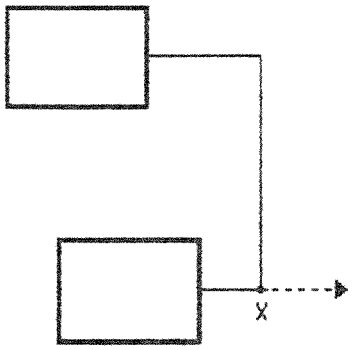
. setas são indicadas pelo código do componente/seta limítrofe origem seguido por '->', seguido pelo código destino e pelo rótulo;

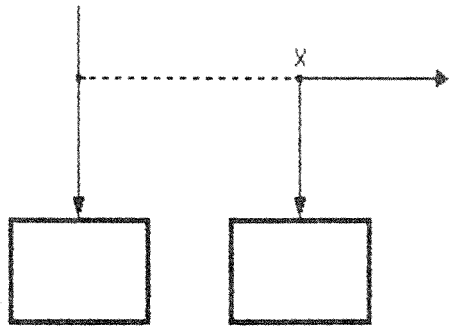
. existência de observação é indicada por um '*'.

Algumas das decisões de projeto podem causar problemas quando da implementação de uma versão mais completa do protótipo. Por exemplo, a modificação de componentes do tipo memória, arquivo ou ente-externo é realizada através de sua eliminação seguida da chamada da rotina para criação de componentes. Isto pode causar inconsistências através da criação de outro tipo de componente, já que não é verificado se a inserção é compatível com a eliminação. Da mesma forma, a remoção (ou modificação) de um arquivo, ente-externo ou ainda um Kit (número-c) não implica na reatribuição de números aos elementos que permanecem, ou seja, a sequência de numeração entre os elementos que permanecem pode ficar descontínua. Além disso, os identificadores mnemônicos para caixas e outros elementos são gerados automaticamente e não podem ser modificados; desta maneira os códigos SECI, que são estabelecidos segundo a ordem de criação das setas, podem resultar em cruzamento de setas conforme a seguinte figura:



Os operadores são inseridos e removidos automaticamente no actigram, impedindo desta maneira que sejam criadas inconsistências pelo usuário como eliminar setas que tenham como origem um operador do tipo união, junção ou junção-ou, representada nas seguintes figuras:





Outros detalhes de implementação se encontram descritos nos anexos.

7.3. Aspectos sobre a Integridade da Base

Existem dois níveis de consistência em relação a um kit: o nível interno e o nível externo. O nível interno diz respeito aos componentes do kit. O nível externo diz respeito a todos os kits do modelo. Na implementação do protótipo foi considerado um subconjunto das restrições de integridade internas e externas

7.3.1. Consistência Interna de um Kit

O nível de consistência interna do kit se refere aos inter-relacionamentos de seus componentes. Neste nível devem ser verificados os aspectos de orientação do método ANA-RE.

Uma vez verificada a consistência interna de um kit, ele pode ser integrado ao resto do modelo através da verificação de integridade externa.

Como conseqüência das decisões de simplificação, alguns aspectos não foram implementados. Em particular, o protótipo não verifica a consistência dos dados fornecidos pelo usuário, nem componentes de orientação ou validade da definição das regras de ativação. Além disso, não há cruzamento de referências de informações entre kits ou verificação da completude de um kit (mínimo de 3 caixas, descrição das caixas, etc). Os demais aspectos são validados (conforme pode ser visto pelos erros apresentados no anexo 3).

7.3.2. Consistência Externa de um Kit

O nível de consistência externa de um kit se refere aos inter-relacionamentos das funções, fluxos, entradas de glossário e demais componentes dos diversos kits de um mesmo modelo. Não faz sentido validar a consistência externa de um kit, se o mesmo não obedece às regras de consistência interna.

Como exemplo do tipo de consistência implementado, temos a preservação do contexto da caixa-pai no diagrama-filho, através de instâncias da entidade referência, que são criadas quando é solicitado o detalhamento da caixa e que não podem ser removidas ou modificadas. Outro exemplo é o da preservação do contexto importado pelo diagrama-filho, que se dá através da proibição da criação ou remoção de setas na caixa-pai.

Após realizada a validação de consistência de um kit (interna e externa), o seu status é modificado para PROPOSTA, o que significa que o kit está apto a sofrer revisão. Se revisto e aprovado pelos comentaristas, o kit é incorporado ao modelo e tem seu status modificado para APROVADO.

Qualquer modificação em um kit, independente de seu status, implica que o mesmo está EM DESENVOLVIMENTO. No caso do status anterior indicar APROVADO, o kit tem sua versão acrescida, permanecendo uma cópia da versão anterior disponível como APROVADA.

A verificação externa considera os kits APROVADOS do modelo, exceto, possivelmente, o próprio kit em sua versão anterior e seus filhos. Esta verificação não leva em consideração a versão dos demais kits. Uma vez validado o kit, este substitui a versão anterior. Seus kits filhos, que não tenham sido objeto das modificações do diagrama-pai, têm o status modificado para EM DESENVOLVIMENTO, devendo, por sua vez, ter sua consistência verificada.

8. CONCLUSÕES E EXTENSÕES

Esta dissertação apresentou o método ANA-RE, resultado de estudos para a automação do SADT, bem como o ambiente SAES e seu primeiro protótipo.

O ambiente SAES representa uma ferramenta de apoio para a elaboração de modelos que permitem a análise, definição e comunicação de requisitos de sistemas de software. É o suporte do método ANA-RE, que aumenta o espectro de problemas passíveis de especificação formal comparativamente ao SADT.

Os capítulos 4 e 5 constituem a especificação dos componentes funcionais do ambiente SAES, levando em consideração o objetivo de suportar o método ANA-RE.

O capítulo 6 define e formaliza o método ANA-RE através de sua comparação com o SADT.

Finalmente, o capítulo 7 valida a proposta do SAES através da descrição de um protótipo de banco de dados que suporta o ANA-RE, implementado em SMALLTALK.

8.1. Contribuições

As contribuições da dissertação enquadram-se em duas categorias: teóricas e práticas.

Do ponto de vista teórico, foi analisada exaustivamente a "linguagem gráfica" do SADT e proposta uma nova linguagem que incorpora os componentes sintáticos e semânticos existentes no SADT, que foram separados dos componentes de orientação, complementares e para referência, e aos quais acrescentaram-se extensões que tornam o ANA_RE mais expressivo que o SADT. Propõe-se um ambiente para suporte da nova linguagem, que, no entanto, também suporta o SADT.

Do ponto de vista de implementação, mostrou-se a viabilidade de se suportar a linguagem e os kits por ela definidos, através de um sistema de banco de dados desenvolvido como um primeiro protótipo.

Apesar de todas as simplificações impostas à implementação, o protótipo obtido pode ser utilizado para a edição e manipulação de qualquer tipo de diagrama ANA-RE ou SADT, para este último com pequenas exceções.

A implementação do banco de dados, utilizando-se o SMALLTALK como ambiente e linguagem de desenvolvimento, forçou a análise e definição dos limites de um gerenciador de banco de dados do tipo necessário para o ambiente SAES, caso se deseje um protótipo implementado a partir do paradigma de objetos.

Por um lado, considerou-se que a implementação do banco de dados deveria oferecer visibilidade de seus componentes, mesmo durante uma sessão para criação ou alteração de diagramas, e que esta não deveria ser gráfica. Por outro lado, a consistência parcial das informações nele residentes é uma característica passível de incorporação ao gerenciador. O protótipo oferece a interface de definição e exibição de componentes, além da gerência de consistência, relacionamento e recuperação de componentes.

A opção pela linguagem SMALLTALK permitiu que cada entidade da modelagem fosse implementada como uma classe. Além disso, o relacionamento de generalização foi facilmente adaptado à hierarquia das classes. Os relacionamentos de associação e agregação (e respectivos atributos) resultaram em estruturas de dados internas definidas nas entidades.

Além de validar a automação do método, o protótipo implementado fez com que algumas hipóteses iniciais fossem descartadas e outras ampliadas.

Algumas entidades artificiais foram criadas na modelagem para permitir maior flexibilidade de uso do sistema. Um exemplo é a incorporação da entidade referência, criada para permitir a incorporação e consistência de setas limítrofes.

Podemos citar como reflexos da implementação:

a. A comprovação de que a unicidade de identificação e, por conseguinte, representação dos componentes em um actigram, é o fator básico para a automação do método ANA-RE.

b. O fato de que o paradigma de objetos deve ser explorado, pois facilitou a definição e implementação das entidades e relacionamentos da modelagem.

c. A solução dada ao mecanismo de navegação na estrutura hierárquica do modelo, no qual um modelo é acessado através de um caminho iniciado sempre pelo kit A-0, sendo possível descer por seus ramos até o kit desejado e deste subir um nível de cada vez ou diretamente até o topo.

d. A solução de se criar a entidade referência para isolar o contexto-pai herdado pelo diagrama das facilidades de edição disponíveis ao usuário.

8.2. Extensões

A dissertação comporta extensões teóricas e de implementação. Faltaria, por exemplo, a seguinte relação de operações à implementação, de maneira a torná-la mais flexível e poderosa:

a. Exibição da árvore hierárquica de kits (estrutura das funções) e das estruturas de dados (composição dos fluxos).

b. Comando UNDO quando da edição de diagramas.

c. Possibilidade de modificação de posição dos componentes em um diagrama e ajuste automático de códigos.

d. Adoção do layout em escada como default para a definição de actigrams.

e. Possibilidade de salvamento e importação de partes de um diagrama (o que permite reutilização e evita a perda das definições quando é necessária uma mudança na caixa-pai).

f. Permissão de elaboração de desenhos com figuras distintas das existentes quando se tratar de comentários (algo semelhante ao utilitário Flow).

Além disto, é necessário verificar quais elementos devem ser ncorporados à linguagem ANA-RE para torná-la mais flexível. Este ipo de decisão só poderá ser feito após o uso extensivo do rotótipo.

Do ponto de vista teórico, é necessário o estudo, formalização incorporação de conceitos como regras de exceção (similares às egras de ativação), regras que relacionem os fluxos do actigram às egras de ativação (e às regras de exceção). Valeria, também, definir ma notação para indicar a existência de iteração, condição ou ecursividade em uma caixa (similar ao que ocorre em ISAC e PSL/PSA).

.3. Perspectivas Futuras

O ambiente SAES necessita ser implementado em um hospedeiro ulti-usuário, o que permitirá a sua utilização por equipes de suários.

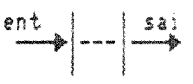
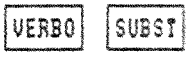
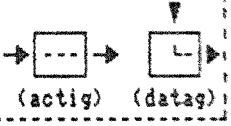
O SMALLTALK parece ser um bom ambiente para a implementação do AES e oferece facilidades para processamento gráfico, para rocessamento de textos e para processamento de janelas e menus. esta maneira, os próximos passos para o desenvolvimento do SAES evem atacar a interface gráfica e o tratamento de textos.

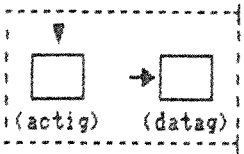
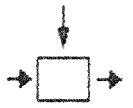
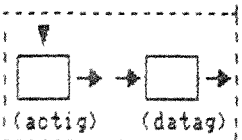
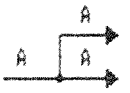
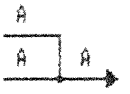
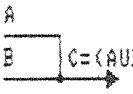
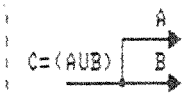
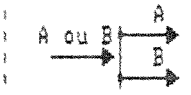
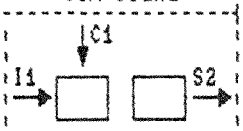
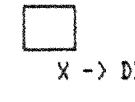
O ambiente SAES deve ser implementado pouco a pouco através da ncorporação gradativa de cada um dos módulos funcionais que o ompõem.

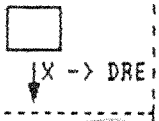
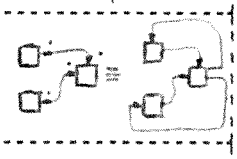
Os trabalhos de [Tate85] e [Dahl87] descrevem a simulação de iagramas de fluxos de dados sendo realizada sobre as folhas da rvore hierárquica, onde são especificadas as funções de maneira ormal. No SAES é interessante que a simulação seja possível onsiderando quaisquer níveis de detalhamento; com este objetivo, é ecessária a realização de estudos para que seja verificado se a inguagem ANA-RE, além das extensões a ela propostas, permite a nterpretação dos fluxos de dados envolvidos em qualquer actigram do odelo.

ANEXO 1 - NÚCLEO GRÁFICO DO SADT

A seguinte tabulação apresenta os 40 aspectos que compõem o núcleo da linguagem gráfica do SADT e nada mais é que uma tradução da tabela apresentada por Ross [Ross77b]:

	Proposito	Conceito	Mecanismo	Notacao
1	contexto limitado	dentro/fora	caixa SA	
2	relaciona / conecta	para/de	seta SA	
3	mostrar transformacao	entrada/saida	interface SA	
4	mostrar circunstancia	controle	interface SA	
5	mostrar significancia	suporte	mecanismo SA	
6	identificacao de caixas	acontecimentos (actigram) objetos (datagram)	nome SA	
7	identificacao de setas	objetos (actigram) acontecimentos (datagram)	rotulo SA	
8	mostrar necessidade	I-O (actigram) C-O (datagram)	passagem	

	Proposito	Conceito	Mecanismo	Notacao
9	mostrar dominio	C (actigram) I (datagram)	**restricao**	
10	mostrar relevancia	ICO	todas as **interfaces**	
11	omitir o obvio	C-O (actigram) I-O (datagram)	**seta omitida**	
12	ser explicito sem introduzir desordem	pipelines conduites e fios	**desvio**	
13			**juncao**	
14	ser conciso e claro	cabos e fios multiplos	**uniao**	
15			**separacao**	
16	mostrar exclusividade	alternativas explicitas	**desvio-ou**	
17			**juncao-ou**	
18	mostrar interfaces c/ diagrama-pai	as setas associadas a ciaxa-pai penetram no diagrama-filho	**seta limitrofe SA**	
19	mostrar explicitamente a conexao com a caixa-pai	codigos ICOM nas setas limitrofes	**codigo ICOM**	
20	mostrar decomposicao unica	DRE (detail reference expression)	**numero-c SA**	

	Proposito	Conceito	Mecanismo	Notacao
21	mostrar decomposicao compartilhada com outro modelo	DRE com a identificacao do modelo	**chamada** ou suporte SA	
22	mostrar cooperacao	intercambio de responsabilidade	**seta bidirecional**	
23	suprimir detalhes de intercambio	permitir pipeline bidirecional dentro de pipeline unidirecional	**seta mista**	
24	suprimir a passagem desordenada	permitir setas irem para fora do diagrama	**tunneling** SA com referencias	
25	suprimir desordem de setas necessarias	permitir pulos rotulados no diagrama	**para-todos** ou **de-todos**	
26	mostrar comentarios necessarios	permitir palavras no diagrama	**nota** SA	NOTE:
27	evitar espacos super utilizados	permitir a colocacao de palavras em locais distantes no diagrama	**nota de rodape** SA	 [n] - no.inteiro
28	mostrar comentarios s/ o diagrama	permitir palavras no (nao dentro) do diagrama	**meta-nota**	(n) - no. inteiro
29	assegurar a associacao apropriada de palavras	conectar palavras ao assunto em foco	**conector** SA	rabisco 
30	referencia unica as folhas	criacao cronologica	**numero-c** SA	iniciais do autor + no.
31	referencia unica aos diagramas	arvore dos numeros das caixas	**codigo de nodo** SA	A (actigram) ou D (datagram) + codigo do diagrama + no. da caixa
32	referencia unica aos diagramas em multi-modelos	utilizar o nome do modelo antes do codigo de nodo	**identificacao de modelo** SA	nome do modelo + codigo de nodo

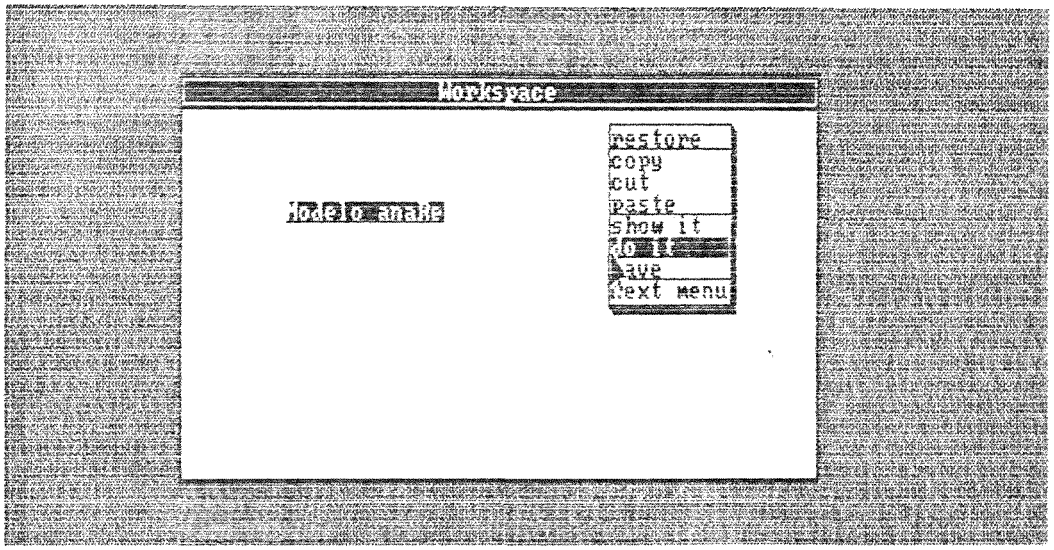
	Proposito	Conceito	Mecanismo	Notacao
33	referencia unica as interfaces	codigos ICOM com codigos de nodo	**codigo de interface** SA	codigo de nodo + codigo ICOM
34	referencia unica as setas	de - para	**codigo de seta**	node ICOM-1 + node ICOM-2
35	mostrar referencia ao contexto	especificar um ponto de referencia	**ponto** SA	A122.4I1
36	auxiliar a correta interpretacao	mostrar dominio geometrico	**layout em escada**	
37	auxiliar o entendimento	sumario de mensagens	**texto** SA	
38	revelar aspectos	efeitos especiais para exposicao somente	**FEQ** SA for exposition only	
39	definir termos	glossario com palavras e figuras	**glossario** SA	
40	organizar paginas	prover tabela de conteudo	**indice** SA	

ANEXO 2 - EXEMPLO DE UMA SESSÃO SAES

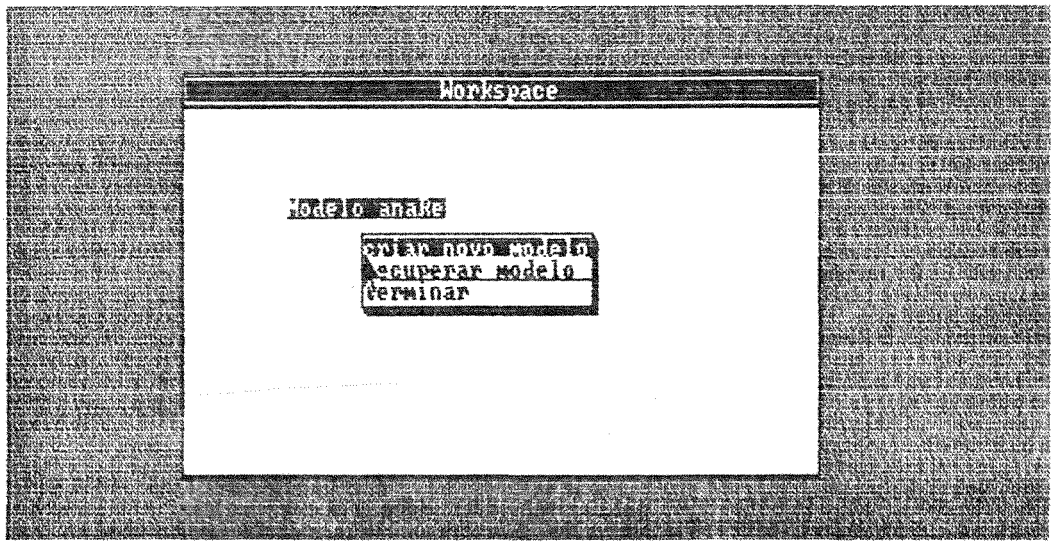
O exemplo de modelo SADT apresentado neste anexo, elaborado por Neighbors e citado em [Free87], descreve em detalhes a utilização pretendida do contexto DRACO.

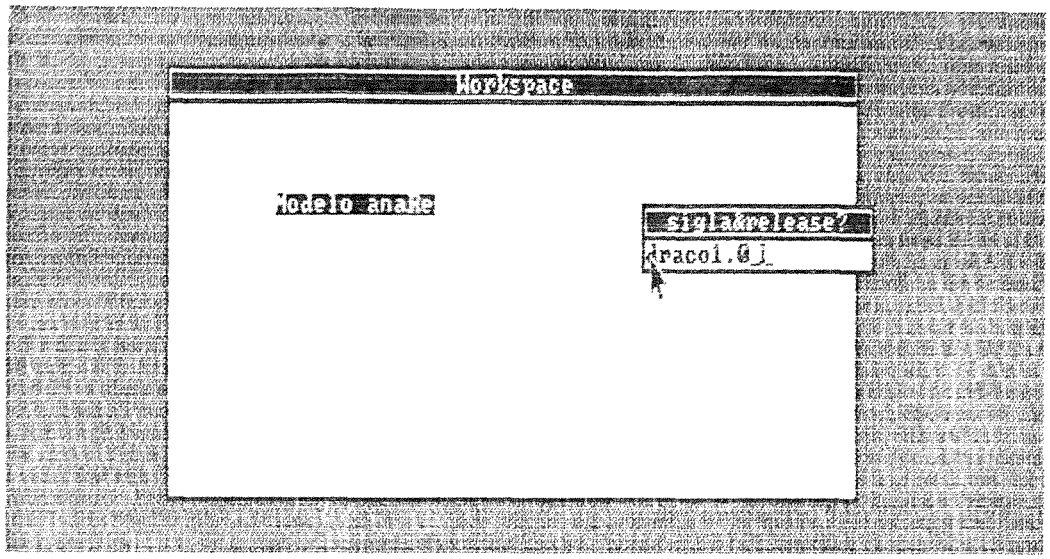
Para exemplificar a iteração de uma sessão de definição com o SAES, são apresentadas cópias das telas exibidas, juntamente com cópia dos originais SADT de Neighbors.

A primeira tela representa o instante da chamada do protótipo, que ocorre através do envio da mensagem **anaRe** à classe **Modelo**, onde está definida uma estrutura de dados que contém todos os modelos definidos através do SAES.



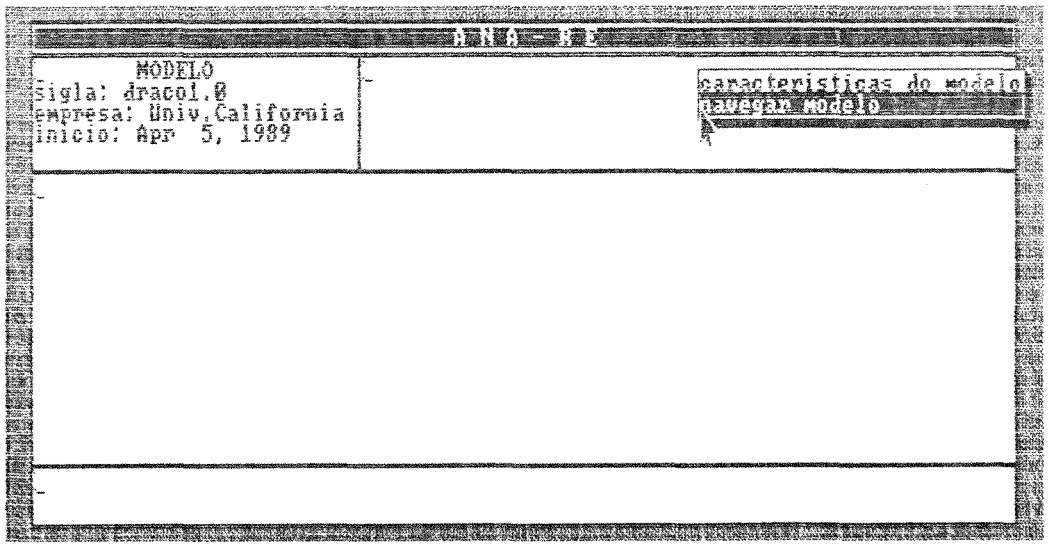
Na segunda tela pode ser observado o primeiro menu do protótipo, a partir do qual pode-se recuperar um modelo já definido ou iniciar a definição de um novo modelo (que é a opção solicitada).



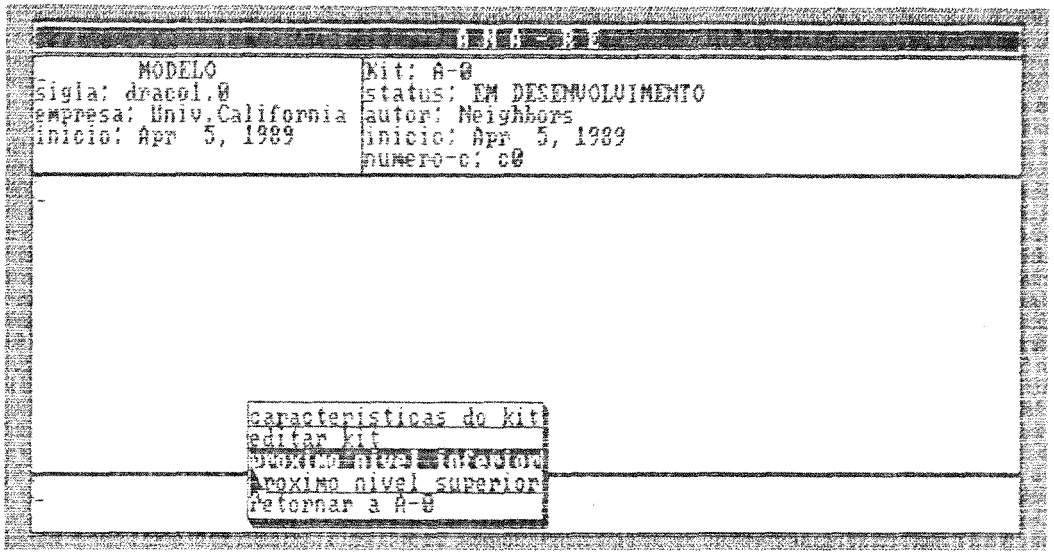


Entre a segunda e terceira telas ocorre um "diálogo" através do qual o SAES solicita a especificação das características do novo modelo. Estas características são exibidas na janela esquerda superior do conjunto de quatro janelas que compõem a interface de comunicação do protótipo com o usuário.

Na quarta tela pode ser observado o menu principal do modelo, a partir do qual pode-se modificar as suas características ou então iniciar a navegação através dos kits que o constituem.

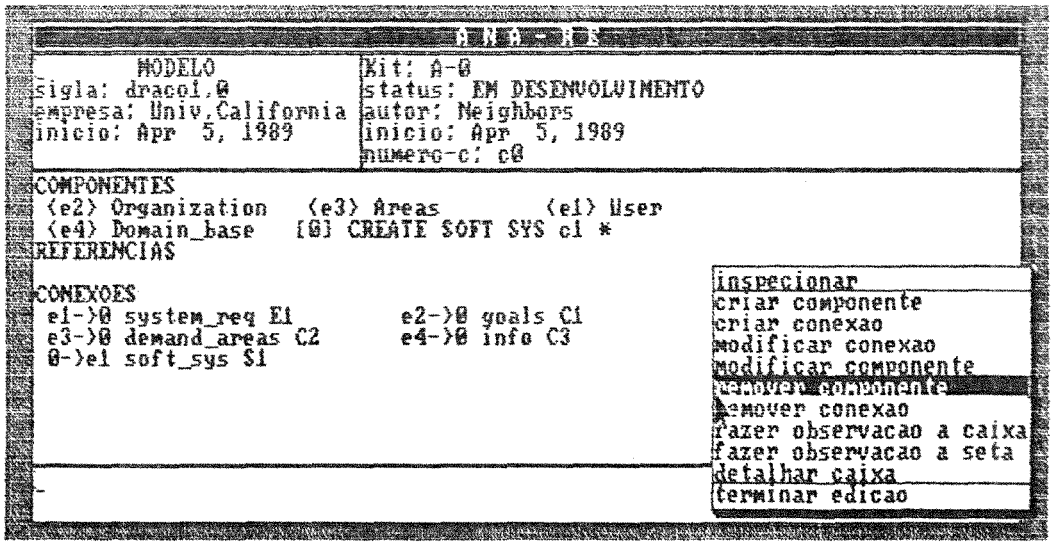


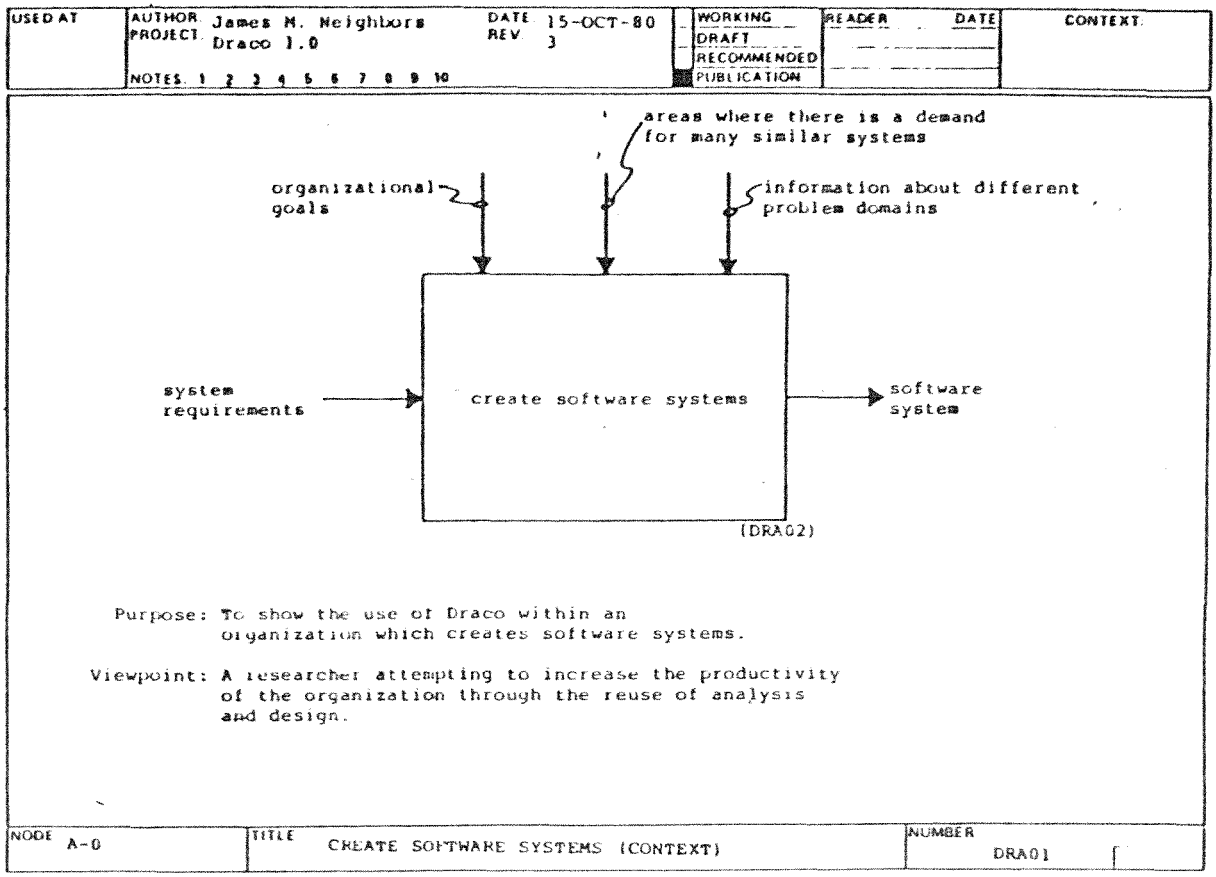
No nosso caso não existem kits definidos e como resposta solicitação de **navegar modelo** e pergunta-se ao usuário se é se desejo definir o kit A-0. Em caso positivo, o sistema se posiciona r diálogo de definição das características do kit e posteriormente apresenta o menu exibido na tela abaixo. Nessa tela é possíve verificar que as informações do kit até onde se navegou sã apresentadas na janela direita superior.



O menu do kit permite navegar para cima ou para baixo n estrutura de kits, editar o actigram do kit ou modificar as sua informações.

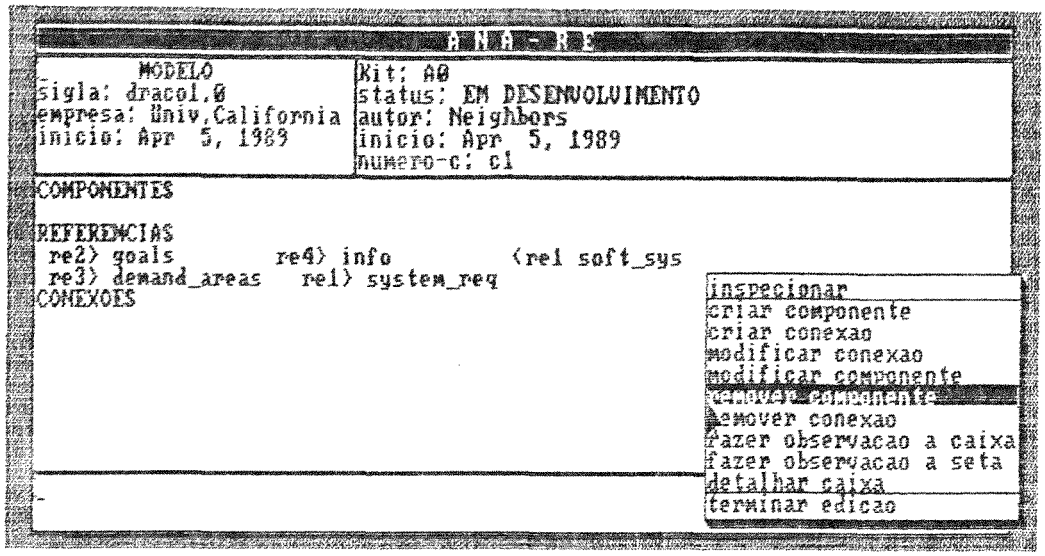
A próxima tela apresenta o menu de edição e a imagem do ki A-0, inteiramente definido. O respectivo diagrama SADT é apresentad em seguida.



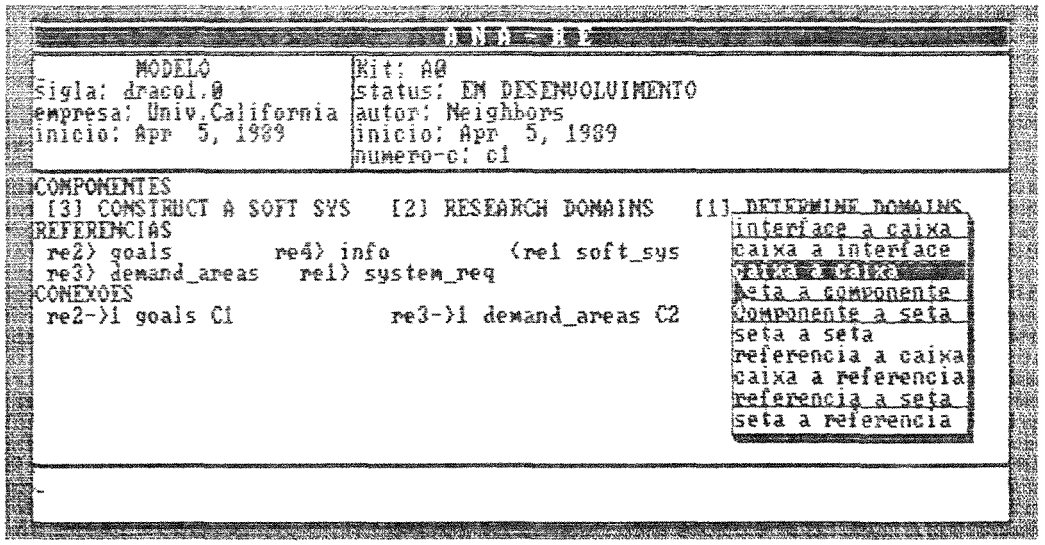


Observa-se que a caixa 0 é representada por **[0]** seguida por seu nome, pelo número-c relativo ao seu detalhamento e por um * que indica existir observação associada à caixa. Esta observação é visível e modificável através da opção **fazer observação a caixa.**

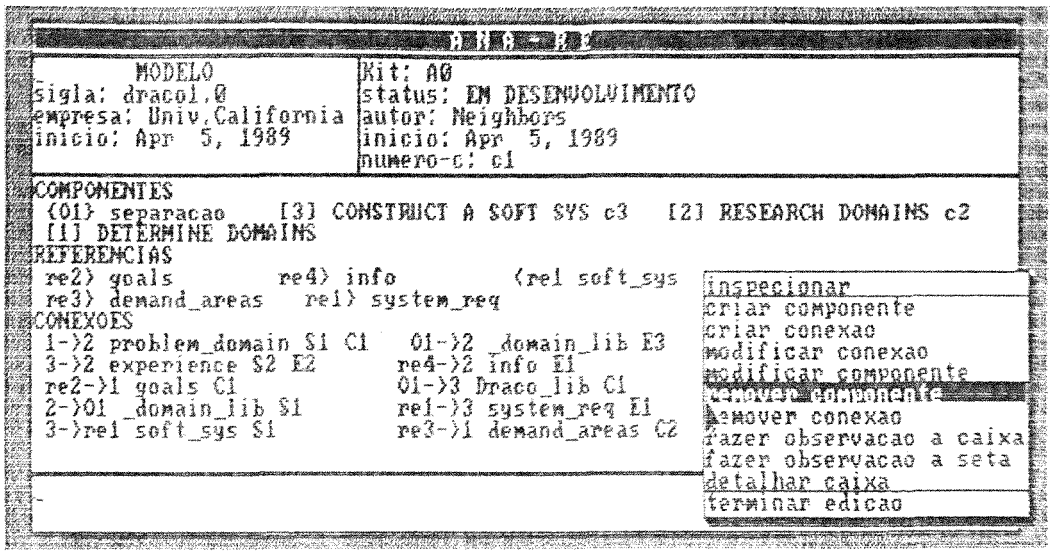
O número-c **c1** é definido automaticamente em conjunto com as setas limítrofes do diagrama-filho e são resultado da solicitação de **detalhar caixa.** A imagem do diagrama A0, logo após o detalhamento de sua caixa-pai, é apresentada na tela 6, obtida após o término de edição de A-0, solicitação de navegação para próximo nível inferior e solicitação de **editar kit.** Nela as referências representam as setas limítrofes.

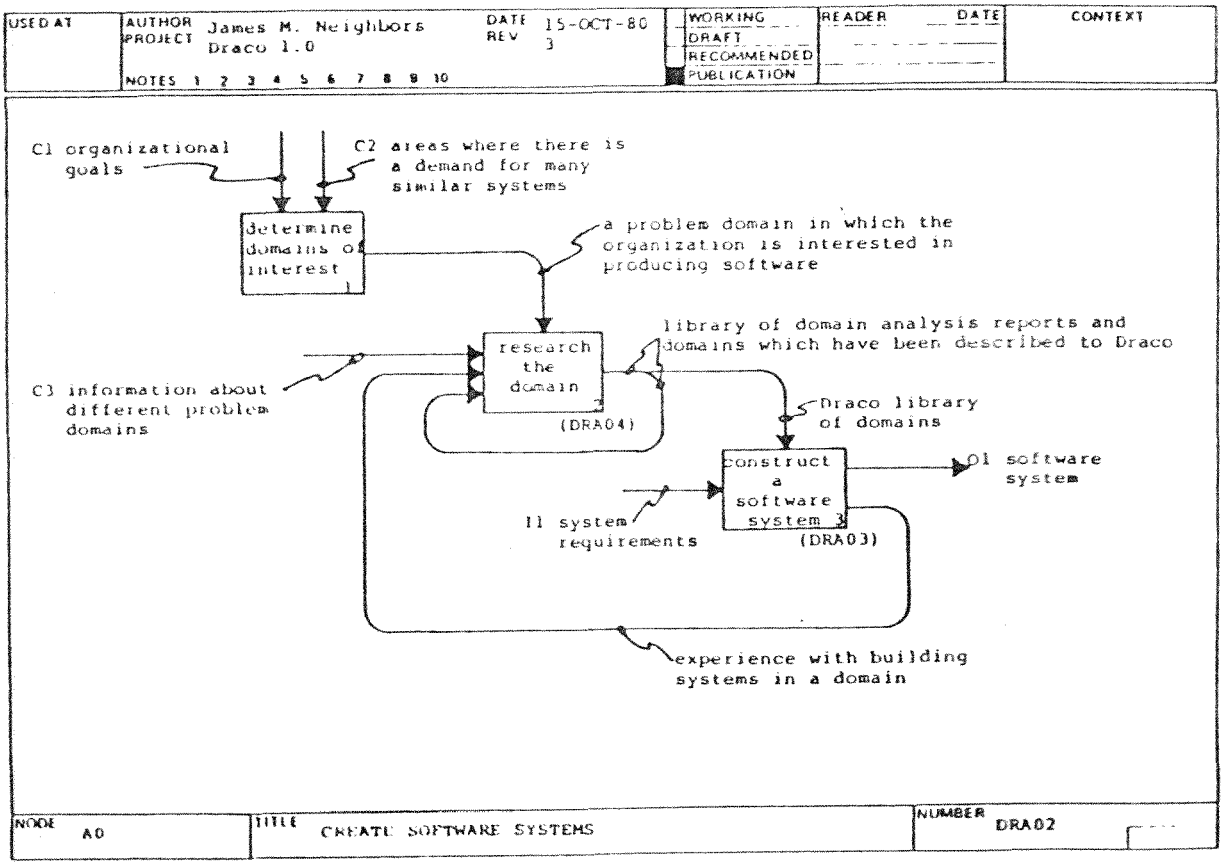


A tela 8 apresenta o instante em que se vai definir a conexão da caixa 1 à caixa 2, após a criação das 3 caixas e das setas relativas a C1 e C2 na caixa 1.

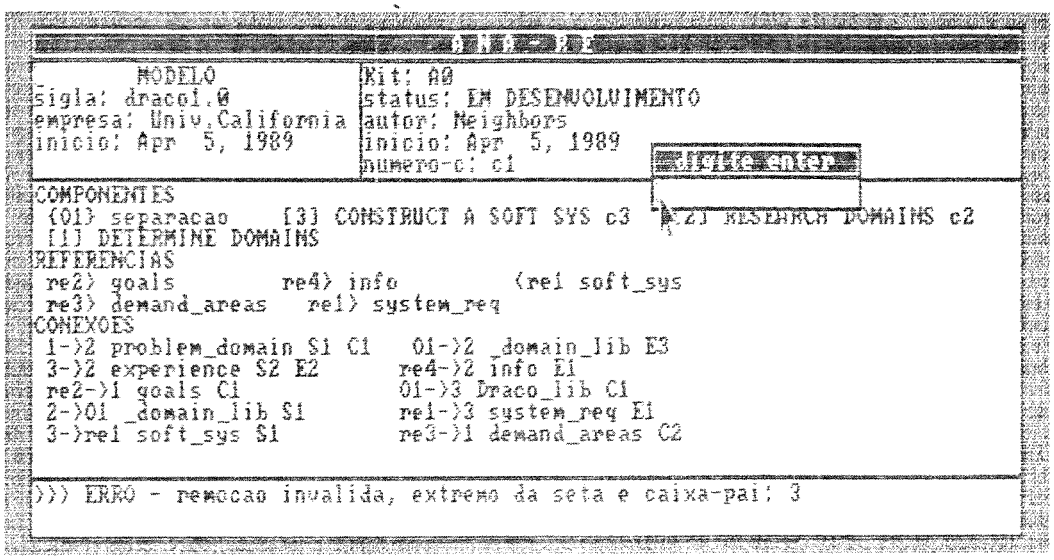


A tela 9 representa o diagrama SADT A0 (apresentado em seguida) totalmente definido, inclusive pela indicação de detalhamento das caixas 2 e 3.


MODELO
sigla: dracol,0
empresa: Univ.California
inicio: Apr 5, 1989
Kit: A0
status: EM DESENVOLVIMENTO
autor: Neighbors
inicio: Apr 5, 1989
numero-c: c1
COMPONENTES
(01) separacao [3] CONSTRUCT A SOFT SYS c3 [2] RESEARCH DOMAINS c2
[1] DETERMINE DOMAINS
REFERENCIAS
re2) goals re4) info (rel soft_sys
re3) demand_areas rel) system_req
CONEXOES
1->2 problem_domain S1 C1 01->2 _domain_lib E3
3->2 experience S2 E2 re4->2 info E1
re2->1 goals C1 01->3 Draco_lib C1
2->01 _domain_lib S1 rel->3 system_req E1
3->rel soft_sys S1 re3->1 demand_areas C2
inspecionar
criar componente
criar conexao
modificar conexao
modificar componente
remover componente
remover conexao
fazer observacao a caixa
fazer observacao a seta
detalhar caixa
terminar edicao


USED AT
AUTHOR James M. Neighbors
PROJECT Draco 1.0
DATE 15-OCT-80
REV 3
NOTES 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT
C1 organizational goals
C2 areas where there is a demand for many similar systems
determine domains of interest
a problem domain in which the organization is interested in producing software
library of domain analysis reports and domains which have been described to Draco
research the domain
(DRA04)
C3 information about different problem domains
Draco library of domains
construct a software system 3
(DRA03)
O1 software system
I1 system requirements
experience with building systems in a domain
NODE A0
TITLE CREATE SOFTWARE SYSTEMS
NUMBER DRA02

Na tela final pode-se observar que a janela inferior é utilizada para exibição de mensagens de erro, como a decorrente de se tentar remover a conexão entre a seta limítrofe I1 e a caixa 3, já detalhada.



NEXO 3 - EXCECÕES CONSIDERADAS

A seguir são apresentadas as condições de execção consideradas implementação do protótipo, cada qual seguida pela mensagem de rro exibida no caso de sua detecção:

- recuperação de um modelo inexistente - **modelo especificado não existe**;
- alteração da sigla de um modelo para um valor já existente - **nova sigla já existente**;
- remoção de uma seta associada a uma caixa-pai - **eliminação inválida, extremo da seta é caixa-pai**;
- conexão de uma seta a uma caixa-pai - **conexão inválida, seta tem como extremo caixa-pai**;
- inexistência de uma seta - **seta inexistente**;
- conexão de dois componentes em A-0, sem que um dos dois seja uma caixa - **conexão inválida**;
- conexão de caixa 0 a ela mesma em A-0 - **conexão inválida**;
- conexão da caixa 0 a um operador - **destino deve ser uma interface** ou **origem deve ser uma interface**;
- caixa com mais de 4 saídas - **número máximo de saídas por caixa é 4**;
- caixa com mais de 4 entradas - **número máximo de entradas por caixa é 4**;
- caixa com mais de 4 controles - **número máximo de controles por caixa é 4**;
- caixa com mais de 4 interrupções - **número máximo de interrupções por caixa é 4**;
- criação de duas setas com mesma caixa origem e mesmo rótulo, mesma caixa origem e mesmo destino, mesma caixa destino e mesmo rótulo, mesma caixa destino e mesma origem ou mesma origem e mesmo destino - **seta duplicada**;
- remoção da caixa 0 de A-0 - **caixa 0 não pode ser removida do diagrama A-0**;
- descida de um nível sem haver detalhamento - **caixa i ainda não detalhada**;
- solicitação de código de caixa e informação de um código diferente de caixa - **código não corresponde a uma caixa**;

- detalhamento de uma caixa que já tem detalhamento - **caixa já tem detalhamento**;
- inexistência de um componente - **componente inexistente**;
- remoção de um componente que é extremo de uma seta - **componente é extremo de uma seta**;
- remoção de um operador - **tipo de componente não removível**;
- inexistência de uma referência - **referência inexistente**;
- criação de mais do que 6 caixas em um actigram - **número máximo de caixas por diagrama é 6**;
- remoção de uma seta cujo destino/origem é um operador dependendo do tipo de operador - **deleção/modificação inválida devido ao operador**;
- conexão de uma caixa a ela mesma - **conexão inválida**;
- destino de uma seta é igual a origem de outra seta que se quer criar ou vice-versa - **conexão inválida de setas**;
- destino de uma seta do tipo seta a componente não é um componente - **destino deve ser um componente**;
- origem de uma seta do tipo componente a seta não é um componente - **origem deve ser um componente**;
- detalhamento de uma caixa sem setas associadas - **caixa-pai inválida, não tem setas associadas.**

REFERÊNCIAS

[Ager79] Agerwala, T.
"Putting Petri Nets to Work"
IEEE Computer, 12/79, pp. 85-94

[Alfo77] Alford, M.W.
"A Requirements Engineering Methodology for Real-Time Processing Requirements"
IEEE Transac. on Soft. Eng'g, SE3(1), 01/77, pp. 60-69

[Balz82] Balzer, R.
"Operational Specification as the Basis for the Rapid Prototyping"
ACM Sigsoft Soft. Eng'g Notes, 7(5), 12/82, pp. 4-16

[Balz85] Balzer, R.
"A 15 Year Perspective on Automatic Programming"
IEEE Transac. on Soft. Eng'g, SE11(11), 11/85,
pp. 1257-1268

[Bell77] Bell, T.E., Bixler, D.C. & Dyer, M.E.
"An Extendable Approach to Computer-Aided Software Requirements Engineering"
IEEE Transac. on Soft. Eng'g, SE3(1), 01/77, pp. 6-15

[Berr87] Berry, D.M.
"Towards a Formal Basis for the Formal Development Method and the Ina Jo Specification Language"
IEEE Transac. on Soft. Eng'g, SE13(2), 02/87,
pp. 184-201

[Berz85] Berzins, V. & Gray, M.
"Analysis and Design in MSG-84: Formalizing Functional Specifications"
IEEE Transac. on Soft. Eng'g, SE11(8), 08/85,
pp. 657-670

[Blah88] Blaha, M.R., Premerlani, W.J. & Rumbaugh, J.E.
"Relational Database Design using an Object-Oriented Methodology"
CACM, 31(4), 04/88, pp. 414-427

[Blan83] Blank, J. & Krijger, M.J.
"Software Engineering: Methods and Techniques"
Wiley-Interscience Pub. 1983

[Blum87] Blum, B.I.
"A Paradigm for Developing Information Systems"
IEEE Transac. on Soft. Eng'g, SE13(4), 04/87,
pp. 432-439

[Boeh76] Boehm, B.W.
"Software Engineering"
IEEE Transac. on Computers, C25, 12/76, pp. 1226-1241

[Boeh77] Boehm, B.W.
"Seven Basic Principles of Software Engineering"
Infotech State of the Art Report, 1977, pp. 77-113

[Booc84] Booch, G.
"Object Oriented Design"
Tutorial on Soft. Design Techniques, [Free84],
pp. 420-436

[Borg85] Borgida, A., Greenspan, S. & Mylopoulos, J.
"Knowledge Representation as the Basis for Requirements Specifications"
IEEE Computer, 18(4), 04/85, pp. 82-91

[Bori87] Boria, J.
"Ingenieria de Software" ed. preliminar
II EBAI ed. Kapelusz 1987

[Brat75] Bratman, H. & Court, T.
"The Software Factory"
IEEE Computer, 05/75, pp. 28-37

[Brat77] Bratman, H. & Court, T.
"Elements of the Software Factory: Standards, Procedures and Tools"
Infotech State of the Art Report, 1977, pp. 115-143

[Cain75] Caine, S.H. & Gordon, E.K.
"PDL - A Tool for Software Design"
Proc. of the National Computer Conference, 1975

[Camp77] Campos, I.M. & Estrin, G.
"Concurrent Software System Design Supported by SARA at the Age of One"
Proc. of the 3rd. International Conference on Soft. Eng'g, 1977, pp. 230-242

[Ceri86] Ceri, S.
"Requirements Collection and Analysis in Information Systems Design"
Elsevier Science Pub., North-Holland, 1986, pp. 205-214

[Chen76] Chen, P.P.
"The Entity-Relationship Model: Toward a Unified View of Data"
ACM TODS, 1(1), 03/76, pp. 9-36

[Colt84] Colter, M.A.
"A Comparative Examination of Systems Analysis Techniques"
MIS Quarterly, 03/84, pp. 51-66
referência em [Leit87]

[Dahl87] Dahler, J. et alii
"A Graphical Tool for the Design and Prototyping of Distributed Systems"
ACM Sigsoft Soft. Eng'g Notes, 12(3), 07/87, pp. 24-36

[Davi77] Davis, C.G. & Vick, C.R.
"The Software Development System"
IEEE Transac. on Soft. Eng'g, SE3(1), 01/77, pp. 69-84

[Dick81] Dickover, M.E., McGowan, C.L. & Ross, D.T.
"Software Design Using SADT"
Tutorial: Software Design Strategies, 2nd ed, IEEE 1981, pp. 397-409

[Digi86] Digitalk Inc.
"Smalltalk/V Tutorial and Programming Handbook"
11/86

[Estr86] Estrin, G., Fenchel, R.S., Razouk, R.R. & Vernon, M.K.
"SARA: Modeling, Analysis and Simulation Support for the Design of Concurrent Systems"
IEEE Transac. on Soft. Eng'g, SE12(2), 02/86, pp. 293-311

[ETHO88] Projeto ETHOS
Info. Final da Fase Preliminar, 01/88

[Feld86] Feldman, P. & Miller, D.
"Entity Model Clustering: Structuring a Data Model by Abstraction"
The Computer Journal, 29(4), 1986, pp. 348-360

[Free80] Freeman, P.
"A Perspective on Requirements Analysis and Specification"
IEEE Tutorial on Soft. Design Techniques, 4th ed, 08/83, pp. 86-96

[Free83] Freeman, P.
"Fundamentals of Design"
IEEE Tutorial on Soft. Design Techniques, 4th ed, 08/83, pp. 2-22

[Free83b] Freeman, P.
"Requirements Analysis and Specification: The First Step"
IEEE Tutorial on Soft. Design Techniques, 4th ed, 08/83, pp. 79-85

[Free84] Freeman, P. & Wasserman, A.I.
"Tutorial on Software Design Techniques"
IEEE Catalog no. EHD05-5, 4th ed., 1984

[Free87] Freeman, P.
"A Conceptual Analysis of the Draco Approach to Constructing Software Systems"
IEEE Transac. on Soft. Eng'g, SE13 (7), 07/87, pp. 830-844

[Gane79] Gane, C.P. & Sarson, T.
"Structured Systems Analysis: Tools and Techniques"
Prentice-Hall Inc., Software Series, 1979

[Gane79b] Gane, C.P.
"Data Design in Structured Systems Analysis"
Tutorial on Soft. Design Techniques, [Free84], pp. 115-132

[Geha86] Gehani, N. & McGettrick, A.D.
"Software Specification Techniques"
Addison-Wesley Pub., 1986

[Gree84] Greenspan, S.
"Requirements Modelling: A Knowledge Representation Approach to Software Requirement Definition"
Ph. D. Thesis, University of Toronto, 1984

[Heni80] Heninger, K.L.
"Specifying Software Requirements for Complex Systems: New Techniques and their Application"
IEEE Transac. on Soft Eng'g, SE6(1), 01/80, pp. 2-13

[Hest81] Hester, S.D., Parnas, D.L. & Utter, D.F.
"Using Documentation as a Software Design Medium"
The Bell System Journal, 60(8), 10/81, pp. 1941-1977

[HIPO73] IBM
"HIPO: Design and Documentation Tool"
IBM Manual SR20-9413-0, 1973

[IEEE84] IEEE
"Guide to Software Requirements Specifications"
IEEE, 1984

[Lean87] Leandro, M.A.C.
"Ambiente de Desenvolvimento de Software Utilizando a Metodologia SADT"
Anais do 1º Simposio Brasileiro de Engenharia de Software, 10/87

[Leit87] Leite, J.C.S.P.
"A Survey on Requirements Analysis"
Tech. Report, University of California at Irvine, 06/87

[Lund80] Lunderberg, M.
"An Approach for Involving the Users in the Specification of Information Systems"
Tutorial on Software Design Techniques, [Free84], pp. 133-155

[Mann87] Mannino, P.V.
"A Presentation and Comparison of Four Information Systems Development Methodologies"
ACM Sigsoft Soft. Eng'g Notes, 12(2), 04/87, pp. 26-29

[McDo86] McDonald, C.W., Riddle, W. & Youngblut, C.
"Stars Methodology Area Summary" vol. II
ACM Soft. Eng'g Notes, 11(2), 04/86, pp. 58-85

[Mill80] Mills, H.D.
"The Management of Software Engineering" Part I
IBM Systems Journal, 19(4), 1980

[Mill86] Mills, H.D., Linger, R.C. & Hevner, A.R.
"Principles of Information Systems Analysis and Design"
Harcourt Brace Jovanovich Pub., 1986

[Mylo80] Mylopoulos, J., Bernstein, P.A. & Wong, H.K.T.
"A Language Facility for Designing Database-Intensive Applications"
ACM Transac. on Database Syst., 5(2), 06/80, pp. 185-207
referenciado em [Leit87]

[Nune87] Nunes, D.J.
"Requisitos de Ambientes de Engenharia de Software"
Anais do 1º Simposio Brasileiro de Eng. de Soft., 10/87

[Parn76] Parnas, D.L.
"On the Design and Development of Program Families"
IEEE Transac. on Soft. Eng'g, SE2, 03/76, pp. 137-145

[Parn82] Parnas, D.L. & Clements, P.C.
"A Rational Design Process: How and Why to Fake It"
IEEE Transac. on Soft. Eng'g, SE12(2), 02/82, pp. 251-257

[Pete77] Peterson, J.L.
"Petri Nets"
Computing Surveys, 9(3), 09/77, pp. 223-252

[Pres82] Pressman, R.S.
"Software Engineering: A Practioner's Approach"
McGraw-Hill ed., 1982

[Rama77] Ramamoorthy, C.V. & So, H.H.
"Survey of Principles and Techniques of Software Requirements and Specifications"
Infotech State of the Art Report, 1977, pp. 265-318

[Rama78] Ramamoorthy, C.V. & So, H.H.
"Software Requirements and Specifications: Status and Perspectives"
Tutorial: Software Methodology, IEEE 1978, pp. 43-164

[Roch87] Rocha, A.R.C.
"Análise e Projeto Estruturado de Sistemas"
Editora Campos, 1987

[Roma85] Roman, G.
"A Taxonomy of Current Issues in Requirements Engineering"
IEEE Computer, 18(4), 04/85, pp. 14-23

[Ross77] Ross, D.T. & Schoman, K.E.
"Structure Analysis for Requirements Definition"
IEEE Trans. on Soft. Eng'g, SE3(1), 01/77, pp. 6-15

[Ross77b] Ross, D.T.
"Structure Analysis (SA): A Language for Communicating Ideas"
IEEE Trans. on Soft. Eng'g, SE3(1), 01/77, pp. 16-34

[Ross85] Ross, D.T.
"Applications and Extensions of SADT"
IEEE Computer, 18(4), 04/85, pp. 25-35

[SESu80] The Software Engineering Subcommitee
"Draft Report on Graduate Program in Software Engineering"
IEEE Comp. Society, 05/80

[Shoo83] Shooman, M.L.
"Software Engineering"
McGraw-Hill ed., 1983

[Slui87] Sluizer, S. & Lee, S.
"Using Executable Specifications to Model User Requirements"
Proc. of the 20th Intern. Conf. on Syst. Scienc., 1987

[SofT76] SofTech Inc.
"An Introduction to SADT"
Manual, 11/76

[SofT81] SofTech Inc.
"Integrated Computer-Aided Manufacturing (ICAM) Architecture"
Vol. 1 - 6, 06/81

[Soft88] IEEE Software
"CASE Technologies, Software Quality Framework, Relational Data Model"
Special Issue, 5(2), 03/88

[Stay76] Stay, J.F.
"HIPO and Integrated Program Design"
IBM Sys. Journal, 15(2), 1976, pp. 143-154

[Tate85] Tate, G. & Docker, T.W.G.
"A Rapid Prototyping System Based on Data Flow Principles"
ACM Sigsoft Soft. Eng'g Notes, 10(2), 04/85, pp. 28-34

[Teic74] Teichroew, D.
"Improvements in the System Life Cycle"
Proc. of the IFIP congress 1974

[Teic76] Teichroew, D. & Hersley, E.A.
"Computer Aided Structured Documentation and Analysis of Information Processing Systems Requirements"
Tech. Report, ISDOS Project, University of Michigan, 08/76

[Teic77] Teichhroew, D. & Hersley, E.A.
"PSL/PSA A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems"
IEEE Transac. on Soft. Eng'g, SE3(1), 01/77, pp. 41-48

[Terw86] Terwilliger, R.B. & Campbell, R.H.
"PLEASE: Executable Specifications for Incremental Software Development"
Tech. Report, Dept. of Comp. Scienc., University of Illinois at Urbana-Champaign, 1986

[Wass80] Wasserman, A.I.
"Information System Design Methodology"
Journ. American Soc. for Info. Science, 31(1), 01/80, pp. 5-24

[Wass82] Wasserman, A.I.
"The User Software Engineering Methodology: An Overview"
Tutorial on Software Design Techniques, [Free84], pp. 669-713

[Wass83] Wasserman, A.I.
"Software Engineering Environments"
Advances in Computers, 22, pp. 109-161

[Wass84] Wasserman, A.I.
"Characteristics of the User Software Engineering Methodology"
IEEE Proc. of the Soft. Process Workshop, 1984, pp. 125-130

[Wass86] Wasserman, A.I., Pircher, P.A., Shewmake, D.T. & Kersten, M.L.
"Developing Interactive Information Systems with the User Software"
IEEE Transac. on Soft. Eng'g, DE12(2), 1986, pp. 325-345

[Wing89] Wing, J.M. & Nixon, M.R.
"Extending InaJo with Temporal Logic"
IEEE Transac. on Soft. Eng'g, V.15(2), 02/89, pp. 181-197

[Yada89] Yadav, S.B. et alii.
"Comparison of Analysis Techniques for Information Requirements Determination"
CACM, 31(9), 09/88, pp. 1090-1097

[YehR82] Yeh, R.T.
"Requirements Analysis - A Management Perspective"
Proc. of the COMPSAC, 11/82, pp. 410-416

[Zave86] Zave, P. & Schell, W.
"Salient Features of an Executable Specification Language and its Environment"
IEEE Transac. on Soft. Eng'g, SE12(2), 02/86, pp. 312-328