I.C.M.S.C.

INSTITUTO DE CIÊNCIAS MATEMÁTICAS DE SÃO CARLOS

SISTEMA DE APOIO POR COMPUTADOR À DOCUMENTAÇÃO DE SISTEMAS

CAETANO TRAINA JUNIOR

Orientador: FERNÃO S.DE R.GERMANO

UNIVERSIDADE DE SÃO PAULO

SÃO CARLOS - SÃO PAULO
BRASIL

UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS DE SÃO CARLOS

SISTEMA DE APOIO POR COMPUTADOR À DOCUMENTAÇÃO DE SISTEMAS

CAETANO TRAINA JUNIOR

Orientador: FERNÃO S.DE R.GERMANO

Departamento de Ciências de Computação e Estatística

São Carlos

1982

Aos meus pais,
pela permanente fonte de vida
na qual sempre me renovo,
dedico este trabalho.

AGRADECIMENTOS

Ao Prof. Dr. Fernão Stella de Rodrigues Germano, meu particular amigo, e orientador não só deste trabalho, como também de todos os empreendimentos em que me lancei, dentro e fora da esfera profissional.

Aos Professores Doutores Odelar Leite Linhares e Maximilian Emil Hehl, pelo incentivo e exemplo, e por toda a dedicação em criar os meios de que me servi para realizar este trabalho.

Ao Prof. João Sahão Junior, meu grande amigo e companheiro de todas as horas, até mesmo da elaboração deste trabalho, através das constantes sugestões e discussões de suas idéias.

Ao Prof. Dr. Jan Franz Willen Slaets, por ter me mostrado o que é o espírito de trabalho em uma universidade.

Ao amigo Eduardo Cleto Pires, que me iniciou na vida universitária e me acompanhou nos primeiros passos dados na ciência da computação.

Ao Roberto Durrer Leite, pelo auxílio na implementação dos programas descritos neste trabalho.

Às secretárias, Maria Eunice Dória, Conceição Ap. M. Martínez e Sueli Ap. H. Ferreira, pela dedicação na datilografia a este trabalho.

Ao Claudio Rondon, pela boa vontade em realizar todas as inúmeras tarefas que lhe solicitei, e a Carmem Lucia Pagadigoria pela digitação de vários programas e textos.

Ao CNPq e FINEP pelo apoio aos programas de pós graduação deste Instituto, e em especial à FAPESP pela colaboração através das bolsas de Iniciação Científica a mim outorgadas.

Agradeço com especial carinho meu irmão Antonio Fernando Traina, não somente a revisão do texto, mas principalmente o incentivo que sempre me deu e a Agma Juci Machado, que muito me auxiliou na apresentação final da dissertação, mas principalmente por sua dedicação a mim, e pelo estímulo que me deu para levá-lo a termo.

Agradeço também a todos que direta ou indiretamente colaboraram para a realização deste trabalho.'

R E S U M O

As características de um sistema para permitir a um computador dar apoio à documentação de sistemas são apresentadas. Verificam-se quais são as características principais que um tal sistema deve ter; em termos de quais as operações que ele deve tornar disponíveis. Para que isso seja possível, três subsistemas são considerados: Um para definir a linguagem que deve ser usada na descrição; outro para manipular a descrição propriamente dita; e o terceiro para tarefas auxiliares às operações com a armazenagem da descrição. Qualquer linguagem relacional pode ser definida especificando-se os tipos de objetos e os tipos de relação que caracterizam o tipo de sistema em consideração. Após a definição de uma linguagem, qualquer sistema, do tipo considerado pode ser documentado. A descrição pode ser feita em etapas sendo que novas informações vão sendo incorporadas a uma base de dados. Em qualquer estágio da descrição, podem ser solicitados testes de consistência e completeza, e relatórios sobre qualquer aspecto do sistema já descrito. Apresenta-se uma implementação de um sistema de apoio por computador à documentação de sistemas com essas características, composta de cerca de 200 rotinas, escritas em FORTRAN IV para um computador PDP 11/45 operando sob o Sistema Operacional RSX-

-11M. Essa implementação teve como um dos objetivos principais a facilidade de ser levada para outros computadores, com o mínimo de adaptações. Além disso ela não limita a dimensão que uma descrição pode ter, sendo essa limitação fixada pelo computador onde estiver operando. No computador PDP 11/45 até um máximo de 32766 objetos e o mesmo número de relações podem ser especificados em cada descrição. São apresentados 2 exemplos de aplicação do sistema implementado: um utilizando uma linguagem definida para descrição de orquestras, e outro utilizando uma linguagem definida para descrição de Sistemas de Informação, baseada na metodologia HIPO da IBM. Inclui-se 71 referências bibliográficas, sendo que 13 são trabalhos anteriores do próprio autor, que vem trabalhando nessa área desde 1974.

A B S T R A C T

Computer-aided Systems Documentation System

The requirements for a system to be used for documenting any type of system are presented. To avoid confusion, the system to be documented is herein called target system. In the requirements, three subsystems are considered: one for the definition of the language to be used in the description; another for manipulating the description itself and the third to be used as an utilitarian subsystem. Any relational description language can be defined. For such definition, the types of objects and types of relations that form the language have to be specified. After the definition of a language, any target system describable in it can be documented. The description of a system can be fed incrementally in a computer data-base. Once described, a variety of reports can be obtained. Completeness and consistency in the description can be carried by the computer. An implementation of such a system is presented. The corresponding software is composed of about 200 rotines, written in FORTRAN IV for a PDP 11/45 under RSX-11M Operating System. The System implementations has been conceived aiming portability so that the effort to install it in other computers should be minimun. Limitations on the PDP implementation are a maximun of 32766 objects and the same number of relations, for each target systems description. Two

application of the implemented system are presented. One regarding a language for describing an orchestra, considered as a system composed of types os objects like musician, instrument, part, etc. and types of relations like plays, conducts, etc. Another regarding a language for describing Information Systems based on IBM HIPO. Seventy-one bibliographical references are included on basic and related works. Of these, 13 are of the author previous work, which has been going since 1974.

I N D I C E

CAPÍTULO I

INTRODUÇÃO

1.1 - Considerações Gerais

Nosso interesse pela Documentação de Sistemas data de 1974. Nessa época, como Técnico Operador de Máquinas Periféricas, junto ao Laboratório de Computação do ICMSC, tivemos a oportunidade de desenvolver para a calculadora programável Hewlett Packard 9820 um conjunto de programas de apoio ao estudo de Teoria dos Grafos, com base em [Hr 69] , [Ma 72] e [Bg 73], dando especial ênfase aos grafos dirigidos.

Foi com base nesses estudos que fizemos a implementação na calculadora programável Hewlett Packard 9830 A de uma versão simplificada do Sistema PSL/PSA (Problem Statement Language/Problem Statement Analyser) documentada em [Tr 76a] e [Tr 76b]. Essa versão simplificada foi escolhida para atender às conveniências da pesquisa relatada em [Ge 76], que mostra que o modelo matemático dessa versão corresponde a um dígrafo categorizado com rótulos. Desde então, dentro de um plano de bolsa de iniciação científica da FAPESP, passamos a pesquisar a melhor maneira de representar Sistemas em Computador [Tr 77]. Conforme relatado em [Tr 78] , chegamos à conclusão que tal maneira baseia-se na representação de dígrafos por listas.

A implementação da versão simplificada do sistema

PSL/PSA foi bastante utilizada como instrumento de pesquisa, a pesar de seu porte reduzido. Extensões ao sistema PSL/PSA foram por nós a ela incorporadas com base em desenvolvimentos teóricos feitos no contexto de trabalhos de doutorado e mestrado. Um exemplo disso foram os recursos para aplicação de testes de consistência em descrições de Sistemas de Informação em PSL documentados em [Tr 76c], [Tr 76d] e [Tr 76e]. Fizemos a incorporação desses recursos à implementação da versão simplificada do Sistema PSL/PSA, com base no desenvolvimento teórico relatado em [Ge 76]. Outro exemplo foram os recursos para descrição e análise da Dinâmica de Sistemas de Informação documentados em [Tr 76f] e [Tr 76g]. A incorporação desses recursos à implementação da versão simplificada do sistema PSL/PSA com base no desenvolvimento teórico relatado em [Sh 76], que por sua vez serviu de base nos trabalhos [Ge 78] e [Sh 78].

A principal simplificação da versão do sistema PSL/PSA implementada, dizia respeito às relações entre objetos que constituem um Sistema de Informação: na versão plena essas relações podem ser binárias ou ternárias, isto é, podem envolver dois ou três categorias de objetos, enquanto que na versão simplificada somente podem ser binárias. Passamos então a pesquisar a melhor maneira de remover essa restrição na implementação efetuada na calculadora programável HP 9830 A. O modelo matemático da versão simplificada foi estendido à versão plena do Sistema PSL/PSA com base em [Ge 76] , conforme relatado em [Ni 77b]. Foi mostrado em [Ni 77a] que, apesar do modelo matemático da versão plena do sistema PSL/PSA corresponder a um hiperdígrafo dirigido, sua representação em termos de dígrafos

categorizados com rótulos era não só possível como conveniente. Assim, a pesquisa relatada em [Tr 78] permitiu o desenvolvimento na própria calculadora programável HP 9830A de uma versão do sistema PSL/PSA, já sem a restrição relativa às relações ternárias. Foi com base nesses estudos, que projetamos a Base de Dados, as principais estruturas de dados e os algoritmos de manipulação correspondentes, de que trata esta dissertação.

Para apoiar o trabalho descrito em [Sh 76] , onde são propostas extensões à Linguagem PSL para melhorar sua capacidade de descrição dos aspectos relativos à Dinâmica de Sistemas de Informação, foi feita uma extensão na implementação da versão simplificada do Sistema PSL/PSA [Tr 76f], onde a modificação principal correspondia às modificações na estrutura de dados que definia a Linguagem PSL. Desses esforços surgiu a idéia de se desenvolver um sistema que permitisse a definição de extensões na linguagem de descrição. Com o início da operação do computador PDP 11/45 no Laboratório de Computação do SCE no início de 1978, passou-se a implementação do sistema PSL/PSA na sua forma plena nesse computador, aproveitando todos os desenvolvimentos teóricos conseguidos [Ge 76], [Ni 77a] [Sh 76][Tr 77], e a experiência obtida com as implementações na calculadora HP 9830 A.

Nos estudos preliminares para essa nova implementação, ao se considerar a forma pela qual o analisador deveria aceitar as extensões da Linguagem, pode-se observar que a linguagem poderia ser totalmente redefinida, permitindo ao analisador operar com uma grande quantidade de linguagens diferentes e independentes. Observou-se que a quantidade de tipos de objetos que ocorrem em sistemas, bem como de tipos de relações

que podem existir entre eles é limitada. Notou-se ainda que poder-se-ia estabelecer regras quanto à possibilidade de ocorrência de cada tipo de relação entre cada par de tipos de objetos. Após o estudo e análise comparativa de várias metodologias para projeto, desenvolvimento, descrição e análise de Sistemas de Informação relatado em [Ge 79] e depois em [Ge 82],foi estabelecido como deveria ser um sistema que fosse flexível o suficiente para servir de apoio à aplicação de qualquer metodologia, e por extensão, para servir de apoio àquelas atividades com qualquer tipo de sistema. Disso resultou a especificação e o início da implementação de um sistema no qual a linguagem a ser utilizada é definida pelo usuário. Dessa forma, pesquisadores de áreas distintas da ciência podem dele se utilizar, definindo suas próprias linguagens, e fazendo portanto o sistema se comportar de modo adequado às suas necessidades. Esse sistema, a que foi designado o nome de Sistema de Apoio por Computador à Documentação de Sistemas (SACDS) é o objeto desta dissertação.

1.2 - Apresentação

No capítulo 2 desta dissertação é apresentado um panorama de literatura atual nas áreas de Sistemas de Informação e de Engenharia de Software, que são as áreas onde este trabalho se situa. Nesse capítulo mostra-se, de uma forma geral o sentido que as pesquisas tomam atualmente, e a seguir serão discutidos com maior detalhe, os trabalhos publicados sobre assuntos diretamente envolvidos com sistemas semelhantes ao apre

sentado nesta dissertação.

No capítulo 3 são apresentadas as características que um sistema para apoio à documentação e análise de sistemas deve ter, e é, feita a proposta de um sistema com o objetivo de permitir a implementação de um sistema de Software capaz de fornecer o apoio apresentado. Essa proposta é feita de forma genérica, sendo apenas comentadas características desejáveis, e como elas poderiam ser postas à disposição do usuário por intermédio desse sistema, sendo portanto especificado como o Sistema de Apoio por Computador à Documentação de Sistema (SACDS) deveria ser.

No capítulo 4 são descritos alguns módulos do SACDS já implementados em um computador. Esses módulos são apresentados, e se mostra que as características de tais módulos correspondem àquelas que no capítulo 3 se demonstrou serem desejáveis. Dessa forma, a implementação do SACDS está iniciada, já sendo possível obter desta alguns resultados, e mostrando que a proposta feita no capítulo 3 pode ser implementada.

No capítulo 5 são descritos 2 exemplos de uso do SACDS. O primeiro exemplo ilustra o uso do SACDS para a documentação de um sistema que não seja de informação, sendo apresentada uma linguagem de descrição de uma orquestra. O segundo exemplo ilustra o uso do SACDS para a documentação de Sistemas de Informação, sendo apresentada uma linguagem de descrição de sistemas adaptada para ser usada com a metodologia HIPO [IBM 74].

O capítulo 6 apresenta vários tópicos de pesquisas de aplicações e de desenvolvimentos do SACDS que podem ser usa

dos como sugestões de pesquisa, e podem dar continuidade a este trabalho. São apresentadas 3 linhas gerais de pesquisa correspondentes à: implementação do SACDS, desenvolvimento de novos recursos para o SACDS e técnicas para aplicação do SACDS a tipos de sistemas específicos.

CAPÍTULO 2

SITUAÇÃO DO PROBLEMA NO PANORAMA DA LITERATURA

2.1 - Considerações Gerais

Neste capítulo procura-se estabelecer a situação atual em que se encontram as técnicas de definição e análise de linguagens de Especificação de Sistemas de Software, em cujo contexto se enquadra este trabalho. São ainda analisadas as publicações recentes nessa área envolvendo o estudo de métodos, técnicas e instrumentos que estão sendo objeto de estudos, e se situa o SACDS, objeto deste trabalho, em relação a eles. Com isso procura-se estabelecer um quadro geral da situação atual, e como este trabalho se posiciona neste contexto.

2.2 - Panorama da Literatura

Davis [Dv 82], define adequadamente estratégias, metodologias e métodos com o sentido usado nesta dissertação, sendo que aqui se fala também de técnicas e instrumentos. Uma estratégia é uma abordagem geral para atingir um objetivo. Um método é um procedimento ordenado ou sistemático. Uma metodologia é um conjunto de métodos e técnicas. Estratégias são abordagens gerais, métodos e metodologias são meios detalhados de colocá-las em ação. As técnicas dizem respeito a como executar

as ações preconizadas nos métodos, e os instrumentos destinam-se a facilitar a execução dessas ações.

O SACDS é um instrumento de apoio à documentação de Sistemas, desenvolvido dentro da técnica de apoio por computador à análise e projeto de Sistemas de Informação. Outros instrumentos notáveis desenvolvidos dentro dessa técnica são o PSL/PSA [Te 77] , o SREM [Te 79] , etc... O uso racional desses instrumentos é feito dentro de um certo método que por sua vez integra uma metodologia, tudo isso para seguir uma certa estratégia. Nessas condições, esta revisão bibliográfica cuida de citar referências relativas não só a outros instrumentos de modo a situar o SACDS em relação a eles, como também relativas à técnica de apoio por computador à análise e projeto de Sistemas de Informação. Cuida ainda de metodologias dentro das quais esses instrumentos são utilizáveis e de estratégias nas quais essas metodologias são usadas.

A definição e análise de Linguagens de Especificação de Sistemas de Software vem sendo objeto de atenção na literatura técnico-científica, sendo que alguns esforços corporativos merecem especial destaque. O IEEE, por exemplo, vem dedicando a ele espaço em suas publicações ordinárias, e em certos casos criando eventos e publicações nos quais é dada ênfase a esse assunto. Exemplos disso são as "International Conferences on Software Engineering", números especiais de publicações como "Computer" e "Proceedings of the IEEE", dedicados a esse tema.

O número de setembro de 1980 do "Proceedings of the IEEE" foi dedicado especificamente a Software Engineering. Note-se que essa publicação é geral para todos os assuntos de in

teresse do IEEE, assim esse número especial tem alcance muito maior, pois divulga esses tópicos em toda a comunidade de seus membros, com o que revela a importância dada ao assunto. Merecem destaque nesse número especial trabalhos [Di 80], [Ha 80] e [Le 80] que situam os métodos de análise e projeto de Sistemas de Informação, bem como as técnicas e o instrumental a eles relacionados no panorama da Engenharia de Software. Menção específica é feita ao instrumental que incorpora apoio por computador do qual o SACDS é um exemplo.

Distaso [Di 80] salienta a importância de metodologias para a determinação dos requisitos ou exigências a que um software deve satisfazer, e cita o importante papel das técnicas de documentação apoiadas por computador. Menciona a necessidade de acompanhar o cumprimento dessas exigências durante o desenvolvimento do sistema. Destaca a importância da estruturação no projeto e na programação, e no uso da documentação como instrumento de desenvolvimento do sistema, vital para que a sua manutenção seja feita de modo eficiente.

Hatvany [Ha 80] mostra que essas técnicas e métodos podem contribuir para obter melhor apoio por computador ao projeto e controle de fabricação, caso especial de projeto apoiado por computador - CAD (Computer Aided Design) sendo de notar que esse próprio instrumental é um caso especial de CAD. o próprio SACDS pode ser analisado como um instrumento de CAD.

Lehman [Le 80] salienta a importância da manutenção no ciclo de vida de Sistemas de Software, e coloca em destaque as consequências na fase de manutenção das atividades desenvolvidas na fase de definição do sistema. Salienta o papel na fa-

se de definição do sistema, dos métodos e técnicas aqui focalizados, realçando o instrumental de apoio por computador.

O número de maio de 1982 do "Computer" foi dedicado a especificações orientadas a aplicações, procurando desenvolver a ligação entre o usuário e o código em sistemas de Engenharia de Software. Apresenta ainda um instrumento de apoio por computador à especificação dos requisitos de sistemas de Controle de processos [Lu 82], discute linguagem para a especificação dos requisitos de Sistemas de Informação [Da 82] e [Le 82] , descreve um instrumento semelhante ao SACDS [De 82], e discute instrumentos e métodos de desenvolvimento de Sistemas de Informação [La 82].

Ludewig [Lw 82] utiliza uma ilustração muito interessante para figurar a conexão entre o usuário e o código em Engenharia de software; por parte do usuário a contribuição para a conexão assume a forma de estalactite em que as gotas de informação fluem delicadamente enquantro que a estalagmite construída em solo firme com todo o apoio de compiladores, sistemas operacionais, sistemas de gestão de bases de dados, e outros instrumentos de apoio por computador ao desenvolvimento de Sistemas de Informação, procura subir ao máxino para encontrar a estalactite, fazendo uso das gotas de informação que chegam até ela. O instrumento de apoio por computador, à especificação dos requisitos de Sistemas de Controle de Processos apresentado por Ludewig é o Sistema Espreso constituído por uma linguagem (Espreso-S) e um software (Espreso-W) para verificar, armazenar e avaliar as especificações nela expressas. Ludewig chama porém a atenção para um importante aspecto relativo aos Instrumentos análogos ao por ele apresentado, um instrumento deve ser usado

no contexto de um método. Assim, tendo adotado um método cabe procurar instrumentos para aplicá-lo, mas é errado primeiro munir-se de um instrumento e só depois adicionar-lhe um método.

Lauber [La 82] classifica os instrumentos de desenvolvimento de Sistemas em Cognitivos, Ampliativos e Notacionais. Chama de instrumentos cognitivos os destinados a melhorar a capacidade intelectual de seu usuário, de modo a produzir melhores sistemas; por exemplo dando roteiros para o desenvolvimento, critérios para a modularização, etc. Chama de Instrumentos Ampliativos aqueles que aumentam a capacidade de produção de seu usuário; por exemplo gerando documentação para atender a multiplas finalidades a partir de uma descrição básica, etc. Chama de instrumentos notacionais aos destinados a facilitar ao seu usuário a expressão da informação, reunida durante o processo de desenvolvimento de Sistemas. Apresenta um instrumento, EPOS, que classifica como ampliativo e notacional, e descreve sua concepção das tendências de evolução no futuro dos instrumentos de apoio por computador ao desenvolvimento de Sistemas. Nota-se que o SACDS, segundo essa classificação, pode ser considerado um instrumento ampliativo com extrema versatilidade para apoiar a criação e aplicação de instrumentos notacionais.

Demetrovics [De 82], como nós, baseou-se no "meta - system" desenvolvido pelo projeto ISDOS para formular o SDLA (System Descriptor and Logical Analyser), cuja arquitetura é bastante semelhante à do SACDS. O SDLA também permite a especificação de Linguagens e o uso dessas linguagens para descrever sistemas. A diferença entre a arquitetura do SACDS e a do SDLA é principalmente que enquanto a arquitetura do SACDS permite a especificação de relatórios para atender às necessidades de seus

usuários, a arquitetura do SDLA considera relatórios fixos, apenas com parâmetros possíveis de serem definidos pelo usuário . Demetrovics refere-se também ao SEMS (the System Encyclopedia Management System) desenvolvido como parte do projeto ISDOS , ressaltando que o seu propósito é o mesmo do SDLA, e que a diferença entre ambos é apenas em estruturas de dados e arquitetura. O objetivo do autor é obter a possibilidade do desenvolvimento de software automaticamente, usando os sistemas descritos. Como o SACDS é semelhante a esses dois instrumentos, presta-se também ao mesmo objetivo, com a diferença que se pode obter com este, relatórios mais próximos a esse fim, se definidos.

Levene [Lv 82] caracteriza os objetivos que uma Linguagem de Especificação de Necessidades deve ter, e faz um levantamento dos instrumentos já existentes para apoiar essas linguagens. Dessa forma descreve rapidamente os instrumentos SREM, PSL/PSA, SDS e SEM. Faz então uma breve comparação entre esses três e o SDS, que é a seguir descrito em detalhe.

Trata-se de um instrumento semelhante ao SACDS, que permite a definição das linguagens que serão usadas. Como resultado, o SDS pode então selecionar informações da descrição com 2 propósitos: verificar a coerência entre as informações descritas e efetuar testes na descrição. Os relatórios que podem ser gerados são fixos, (não podem ser definidos) e as análises que se podem realizar são feitas manualmente. Essas 2 últimas características são as principais diferenças que apresentam em relação ao SACDS.

O número 1 do volume 21 do "IBM Systems Journal" de

de 1982 é dedicado à análise de empresas, e dá destaque a estratégias para a determinação dos requisitos de Sistemas de Informação [DV 82] , compara metodologias para planejamento de Sistemas de Informação a nível de empresa [Za 82], e apresenta sugestões para a integração do instrumental disponível para desenvolvimento de Sistemas de Informação [Ne 82]. Além disso, demonstra a utilidade de um instrumento desenvolvido com outras finalidades, para apoiar a aplicação de um método para planejamento de Sistemas de Informação [Sa 82].

Davis [Dv 82] discute 4 estratégias para a determinação dos requisitos de um Sistema de Informação: a) Perguntar, b) Derivar (com base em um Sistema de Informação existente) , c) Sintetizar (com base nas características do sistema usuário) e d) Descobrir (com base em experiência com um Sistema de Informação evolutivo). Fornece critérios para seleção de uma dessas estratégias com base em características do Sistema Usuário, Sistema de Informação, usuários e analistas. Chama Sistema Usuário, o sistema em cujo funcionamento é apoiado pelo Sistema de Informação. Detalha essas estratégias citando métodos utilizáveis dentro de cada uma. Dentre as metodologias mencionadas, para determinação dos requisitos de um Sistema de Informação dentro da estratégia de sintetizar com base nas características do Sistema Usuário, menciona a de análise de entradas - processamento - saídas. Integrando essa metodologia, refere-se entre outros métodos ao ISAC [Lu 81], ADS[Co 74] e aos diagramas de fluxo de dados [Gr 77] , dentro dos quais a técnica de apoio por computador tem sido usada. Quanto ao ADS, um instrumento análogo ao SACDS, só que de âmbito específico, foi desenvolvido por Nunamaker [Th 74] . Quanto aos diagramas de fluxo

de dados, um instrumento de apoio por computador para produzí-los foi desenvolvido por [D1 82] e quanto ao ISAC, Lundeberg [Lu 81] desenvolveu um instrumento de apoio por computador à sua aplicação.

Zachman [Za 82] compara dois métodos para a análise de empresas: o BSP [IBM 81] e o BICS [Ke 80] que está sendo desenvolvido com base no BIAIT [Bu 79], [Ca 79] pela IBM, enquanto que Sakamoto [Sa 82] mostra como utilizar o "DB/DC Data Dictionary" da IBM como instrumento de apoio à metodologia BSP. Na verdade, Sakamoto utiliza características de expansibilidade do dicionário de dados da IBM para criar tipos de categorias e de relações adequados ao modelo de dados do BSP. Situa ainda o instrumento por ele adaptado com relação a outros mais específicos como é o caso do sistema PSL/PSA [Te 77].

O uso do Dicionário de Dados como proposto, poderia ser substituído com vantagens pelo SACDS, que já possui a estrutura para gerar os relatórios que Sakamoto sugere implantar. Além disso, o SACDS tem por objetivo exatamente o tipo de definição de categorias e de relações propostos, ou seja, para que o SACDS faça o papel do instrumento descrito por Sakamoto, basta criar um Arquivo de Definição de Linguagem conveniente.

Newman [Ne 82] relaciona as metodologias e os instrumentos disponíveis para o desenvolvimento de Sistemas de Informação, e apresenta sugestões para melhor integrá-los, dando ênfase aos aspectos de documentação, assunto para o qual o SACDS está apontado.

Wasserman [Wa 82] analisa o instrumental disponível para desenvolvimento de Sistemas de Informação e relaciona características desejáveis em métodos de desenvolvimento de Sis-

temas, entre as quais figuram: 1) Abranger todo o ciclo de desenvolvimento, 2) Facilitar a transição entre as fases que compoem o ciclo, 3) Possibilitar verificação da coerência entre as fases, 4) Abranger toda uma classe de problemas, 5) Utilizar apoio de computador, 6) Apoiar a evolução do Sistema.

O SACDS proporciona a possibilidade de apoio por computador a métodos com essas características, pois permite a definição de linguagens adequadas a qualquer das fases do ciclo de desenvolvimento, linguagens essas todas de mesma natureza, e por isso pode proporcionar as verificações desejáveis ao longo de todo o ciclo.

Wasserman menciona ainda características desejáveis em Instrumentos de Desenvolvimento de Sistemas, entre as quais figuram: a) Propósito bem definido, b) Facilidade de uso, c) Auto-documentalibilidade, d) Compatibilidade entre as funções desempenhadas, e) Adaptabilidade às necessidades do usuário , etc. Como se pode observar nos capítulos subsequentes, o SACDS possui essas características.

Delisle [D1 82] apresenta um instrumento de apoio à metodologia de "Análise Estruturada a la Constantine" [Co 74] com grande potencial gráfico que também possui essas características. Outros instrumentos de grande utilidade que poderiam ser compatíveis com o SACDS são apresentados por Lundeberg [Lu 81], Tichy [Ti 82], e White [Wh 82].

Blum [B1 82] descreve um instrumento para desenvolver Sistemas de Informação centrados em base de dados. Sua abordagem é diferente da do SACDS, mas esse último poderia ter sua aplicação estendida para desempenhar função análoga.

Como resultado da verificação de que os problemas da engenharia de software, tais como aumento do custo de desenvolvimento e manutenção do software, eficiência questionável, e alta incidência de erros, são comuns tanto em instituições comerciais quanto em agências do governo americano, o Departamento de Defesa dos Estados Unidos apoiou a preparação de um trabalho publicado em 1979, intitulado "Research Directions in Software Technology". Seu objetivo foi o de "... apresentar um panorama do estado da arte e direção da evolução de todas as áreas - todos os enfoques conhecidos que poderiam contribuir para uma eventual evolução do software". Esse trabalho foi preparado por um conjunto de especialistas, de todas as áreas relacionadas à ciência de computação e apresentada na forma de um livro em que cada capítulo é um artigo cuja responsabilidade cabe a um especialista da área em questão. De especial interesse são os artigos de Boehm [Bo 79b] e McGowan [Mg 79].

Boehm analisa a situação atual da indústria de desenvolvimento de software, cobrindo em sua análise as técnicas e tendências atuais empregadas em todo ciclo de vida de um sistema. Na fase de especificação do projeto, analisa especialmente os sistemas de descrição de especificações PSL/PSA do Projeto ISDOS e SREP, ressaltando que o sistema PSL/PSA é o pioneiro em dispor de apoio por computador para análise de especificações de necessidades de software. O sistema SREP (Software Requirements Engineering Program) foi em parte desenvolvido tendo por base o projeto ISDOS, e é considerado nesse artigo como o maior sistema em desenvolvimento. Está sendo desenvolvido para o "U.S.Army Ballistic Missile Defence Advanced Technology

Center", mantendo a mesma organização do PSL/PSA, sendo constituído de uma Linguagem de definição das especeficações (RSL) e um analisador (REVS). O SACDS foi também desenvolvido a partir do projeto ISDOS, sendo diretamente compatível com o PSL/PSA, e portanto o PSL/PSA pode ter o SACDS como apoio, bem como o SREP.

Nas demais fases, o artigo apresenta técnicas atuais utilizadas em cada área. Principalmente nas fases de projeto de software e teste de sistemas, as técnicas discutidas podem ser auxiliadas com o uso do SACDS. Mc Gowan [Mg 79] considera que, dada a crescente complexidade dos projetos de software em desenvolvimento, a maior fonte de benefícios e barateamento desses projetos são os avanços nas atividades de gerência, e nas formas de comunicação técnica entre as pessoas envolvidas nesses projetos. Apresenta também uma descrição dos documentos mais comuns, tradicionalmente usados e produzidos durante o ciclo de vida de um sistema, e analisa esses documentos sob o enfoque de que eles são produzidos dessa forma, principalmente em decorrência de necessidades administrativas de produção dos sistemas, tais como custos e prazos de produção. A seguir o trabalho considera que o conceito de ciclo de vida de um sistema tem auxiliado a evolução desses sistemas, porém não se pode aceitar que um determinado sistema está em uma fase específica, mas sim em várias delas simultaneamente, pois frequentemente várias partes do sistema são refeitas, ou redefinidas, enquanto outras não. Daí conclui que os documentos tradicionalmente usados devem ser reestruturados, para evitar as frequentes incompatibilidades que existem entre eles, e entre os sistemas produzidos e sua documentação. Baseando nisso, Mc Gowan aponta a

necessidade de melhores métodos de projeto de software e de especificação de necessidades. Quanto a especificação de necessidades, elas são divididas em metodologias gráficas e metodologias apoiadas por computador. Entre as metodologias gráficas, dá especial atenção ao HIPO e ao SADT. A metodologia HIPO é tratada no capítulo 5 desta dissertação, onde se mostra uma linguagem, que foi criada com o objetivo de possibilitar ao SACDS apoiar essa metodologia. Entre as metodologias apoiadas por computador cita-se o PSL/PSA e SREM.

Stevens [St 82] apresenta uma técnica de modularizar programas de forma a que cada módulo seja independente de qualquer sinal de controle proveniente de outros módulos, ou seja, segundo o modelo de Projeto Estruturado [St 74] seriam módulos que apresentam o nível de acoplamento por Dados. Nesse artigo, é apresentada uma sugestão de como a descrição da interligação de tais módulos pode ser feita. O SACDS pode assumir a função de interpretar essa descrição, e produzir relatórios que mostram como o controle de tais módulos deve ser feito, indicando as inconsistências que as ligações apresentam e a sequência que as interligações devem ser feitas.

Mekly [Mk 80] sugere uma representação útil para engenharia de software que seja compatível, com as mesmas características e vantagens, das tradicionalmente usadas em outras modalidades de engenharia, tais como esquemas e plantas. Usa-se a teoria das Redes de Petri para introduzir o conceito de processos abstratos (AP-net). A representação é em essência um dígrafo, que pode ser descrito através de palavras como Sequência, Seleção e Iteração. Dessa forma pode-se ter o SACDS como

uma ferramenta que lhe permite dispor de computador, para que este gere os gráficos que a representação propõe, com a vantagem de, por ter um modelo matemático bem estabelecido, permitir que um computador verifique todas as condições de completeza que propõe.

Estes 2 últimos artigos apresentam conceitos de novos modelos para desenvolvimento de software, a nível de técnica de programação [St 82] e a nível de representação [Mk 80] que podem ser apoiados pelo SACDS.

2.3 - Evolução dos Instrumentos

O SACDS, conforme explicado no início deste capítulo, é um instrumento de apoio à documentação de sistemas. Em toda a literatura revista, existe a menção a apenas 5 instrumentos com o mesmo objetivo e que são relevantes para este trabalho. São eles o PSL/PSA [Te 77], [Te 75], o SREM [Al 77], [Be 77], o SEM [Te 79], o SDLA [De 82] e o SDS [Lv 82], sendo que os três primeiros são sistemas desenvolvidos pelo projeto ISDOS, ou com forte apoio deste (caso do SREM).

O sistema PSL/PSA é pioneiro no apoio por computador à análise de especificação de sistemas de informação. É constituído de uma linguagem relacional (PSL) e um analisador dessa linguagem (PSA). A linguagem permite a definição de objetos classificados como sendo de determinado tipo dentre um conjunto de tipos de objetos pré-determinado, e de relações entre esses objetos, escolhidas também dentre um conjunto de tipos de relações predeterminado. As descrição feitas nessa linguagem

podem ser submetidas a um computador através do analisador (PSA), que verifica sua validade e a armazena então em uma base de dados. Com essas informações, a pedido do usuário gera relatórios envolvendo toda a descrição ou aspectos específicos indicados pelo usuário. Existem mais de 40 tipos de relatórios à disposição do usuário, que pode usá-los para documentar, analisar e corrigir seu sistema. Dispõe ainda de uma linguagem de consulta, que permite ao usuário selecionar os dados que ele deseja incluir em determinado relatório.

O sistema PSL/PSA é um instrumento que poderíamos considerar um "instrumento fixo", pois destina-se a ser utilizado em conjunto com uma metodologia bem estabelecida, com uma linguagem que por esse motivo é pré-definida, "fixa", e portanto inalterável. O analisador é feito para essa linguagem explicitamente, dispondo da capacidade de gerar relatórios sobre as informações fornecidas, nessa linguagem. Como não se podem definir os relatórios, também os relatórios gerados são "fixos".

O sistema SREM, foi desenvolvido em grande parte como uma evolução do Sistema PSL/PSA, tendo como objetivo principal as necessidades de especificação de projetos de desenvolvimento de software em tempo real. Basicamente mantém a mesma estrutura do sistema PSL/PSA, contituído de uma Linguagem (RSL) e um analisador e avaliador dessa linguagem (REVS). A lingua - gem RSL é estruturalmente semelhante ao PSL, sendo basicamente substituídos os conjuntos de tipos de objetos e tipos de relações. O analisador destina-se a essa linguagem e portanto pode-se considerar também o SREM um instrumento fixo. O REVS, como o PSA é capaz de produzir grande número de relatórios e também

dispõe de linguagem de consulta. Seu principal avanço no entanto, é sua capacidade de gerar simuladores funcionais a partir das especificações, com ampla capacidade de verificação de consistência e corretividade.

A possibilidade de se mudar uma linguagem através da substituição do conjunto de tipos de objetos e tipos de relações, com a utilização de grande parte do software de apoio já desenvolvido levou à proliferação de várias linguagens com seus respectivos analisadores, como por exemplo, o PCSL [Lw 82].

Dada a possibilidade de se alterar uma linguagem, e como com isso sempre havia a necessidade de se reescrever parte do software de apoio, foi criado o "System Encyclopedia Management System" (SEM). Trata-se de um Sistema de Software completamente independente de uma linguagem específica, desde que determinada linguagem corresponda em estrutura ao modelo utilizado (modelo Entity-Relationship). Dispõem de um sistema para a definição de linguagem (ISLDS), que estabelece a linguagem, as análises e os resultados que são manipulados pelo SEM. Pode gerar também um grande número de relatórios, sendo que os relatórios específicos de uma dada linguagem devem ser construídos através de software que não é incluído com o SEM, sendo que amplo apoio a isso é dado através de rotinas de acesso à base de dados gerada pelo SEM, e de acesso às informações descritas para o ISLDS. Uma característicca que deve ser notada é que o SEM é plenamente compatível com o PSL/PSA, pois uma descrição feita por este pode ser analisada pelo SEM operando com a linguagem PSL.

O sistema SEM é um instrumento que poderíamos chamar de um "meta-sistema", pois possui capacidade de aceitar u-

ma definição da linguagem que irá utilizar para interpretar a descrição dos sistemas com que irá ser usado.

O sistema SDLA, desenvolvido em cooperação com o projeto ISDOS é também um "meta-sistema" com características semelhantes ao SEM, porém com capacidade de manipular linguagens de estrutura diferente deste. O modelo da linguagem também é caracterizado por objetos e relações entre eles, porém utilizando o conceito de atributos, que é caracterizado pelo "tipo" associado a um objeto (nesse sistema chamado de conceito). Como os demais sistemas analisados, o SDLA dispõe de uma componente para geração de relatórios independente da linguagem utilizada que é definida através de um "meta-interpretador" que gera uma "meta-base de dados", para ser usada pelo analisador e gerador de relatórios.

O sistema SDS é um outro "meta-sistema", com a possibilidade de aceitar a definição de linguagens com estrutura semelhante àquela usada pelo SEM, e que apresenta a análise da descrição efetuada numa dada linguagem como a sua característica mais destacada, acessível através de uma linguagem de consulta, com capacidade de aceitar a definição e executar macros.

2.4 - Conclusões

Nesta revisão, pode-se observar que o assunto tratado neste trabalho tem sido objeto de muito estudo, e que várias abordagens diferentes tem sido propostas. Pela grande incidência de publicações recentes dedicadas a esse assunto, e pelo conteúdo delas, pode se verificar que é um assunto atual

e que o SACDS representa um instrumento que corresponde às necessidades mencionadas, bem como apresenta algumas características não abrangidas pelos demais, entre elas a possibilidade de ser usado com sistemas que não são necessariamente sistemas de informação.

Em relação aos demais instrumentos analisados pode-se dizer que o SACDS é um "meta-sistema", com a possibilidade de aceitar definição de linguagens com estrutura que tem como modelo o modelo Entity-Relationship, semelhante às aceitas pelo SEM, podendo como este aceitar a descrição de um sistema em PSL, desde que disponha da definição dessa linguagem. Comparado com os sistemas apresentados, o SACDS apresenta-se com características que abrangem a maior parte do que os demais sistemas apresentam, além de possuir características próprias, como a maior abrangência de tipos de sistemas com que pode ser usado, e a possibilidade de se definir relatórios e condições de consistência e completeza como parte da definição da linguagem com que irá operar.

CAPÍTULO 3

ESPECIFICAÇÃO DO SISTEMA

3.1 - Considerações Gerais

Neste capítulo é feita uma descrição das principais características que um sistema de apoio por computador para a documentação e análise de sistemas deve ter, para que seu uso seja vantajoso no apoio à maior parte dos tipos de sistemas.

Partindo da experiência com o desenvolvimento de outros sistemas por nós realizados, entre os quais situam-se aqueles mencionados no capítulo 1, procurou-se neste capítulo reunir as características desejáveis que um sistema com esse objetivo deve possuir. Sistemas similares, foram desenvolvidos especificamente para satisfazer alguma necessidade existente , ao passo que no desenvolvimento do SACDS, procurou-se inicialmente estabelecer o que o sistema deveria ter, sem se preocupar em como isso poderia ser feito. Neste capítulo, toda a especificação teórica de um tal sistema é apresentada, ficando para o capítulo 4 a descrição da forma como parte de tal sistema foi implementado.

Um sistema construído com esse objetivo pode ser subdivido para fins de estudo, em 2 subsistemas: Um que se encarregaria da definição das características gerais próprias dos sistemas de um determinado tipo e o outro, que, apoiado nessas

definições, se encarregaria da descrição e análise de cada sistema particular. Inicialmente é feita uma descrição das atividades do sistema como um todo, e como atividades podem ser desenvolvidas como componentes específicas. A seguir as características de um sistema genérico que devem ser definidas são analisadas e é proposta uma forma capaz de receber de um analista a definição de tais características, em termos de linguagem para a descrição de um sistema, os parâmetros de interesse desse sistema, as condições de Consistência e Completeza, e uma descrição de como são desejáveis os relatórios referentes a esse sistema.

A seguir são descritas as componentes que comporiam o segundo subsistema, analisando os mesmos aspectos anteriores, agora sob o enfoque da descrição de um sistema particular.

Parte do assunto aqui tratado encontra-se em [Tr 82d].

3.2 - Componentes Funcionais

O sistema de Apoio por Computador à Documentação de Sistemas (SACDS), proposto neste capítulo, se destina a permitir que um sistema de qualquer tipo possa ser descrito de uma forma analisável por computador, no qual passa-se então a armazenar todas as informações fornecidas, e com base nelas, se possa emitir relatórios de interesse para as pessoas envolvidas no sistema.

Para que isso seja possível, é necessário dispor de uma linguagem através da qual a descrição possa ser feita, que seja voltada para o tipo de sistema que está sendo descrito. As

sim, existirá uma linguagem específica para cada tipo de sistema, porém todas as linguagens devem ter a mesma estrutura, para que possam ser manipuladas pelo SACDS.

Como um sistema pode ter seu tipo caracterizado pelos tipos de objetos que o compõem, e pelos relacionamentos possíveis entre eles, a linguagem para a descrição de um sistema pode ser feita com esses tipos de objetos e relações como "palavras chaves". O sistema passa a ser descrito especificando-se cada objeto, e os relacionamentos que o envolvem. A linguagem pode ser a seguinte:

- Identifica-se que serão fornecidas informações a respeito de um determinado objeto, indicando-se o tipo desse objeto precedendo o nome do objeto. Como o tipo de um objeto é uma palavra chave da linguagem, ele passa a ser o identificador que indica para o analisador da linguagem, o início da descrição de um novo objeto, que passa a ser o objeto identificado.

- As relações que envolvem esse objeto são então indicadas através da palavra chave que indica o tipo da relação seguida de um ou mais nomes de objetos que se relacionam com o objeto identificado. Com todos os objetos que se relacionam através daquele tipo de relação com o objeto identificado já especificados, passa-se a outro relacionamento, indicando-se o tipo de uma outra relação e os nomes dos objetos que se relacionam dessa forma com o objeto identificado. Com todos os relacionamentos de um objeto já especificado, indica-se o tipo e o nome de um outro objeto, que se torna então o objeto identifido, e especificam-se seus relacionamentos. Dessa forma, to-

dos os objetos do sistema são descritos, bem como a forma pela qual eles se relacionam, e portanto o sistema é descrito. A linguagem é específica para cada tipo de sistema, uma vez que usa como palavras chaves os tipos dos objetos e das relações específicas de cada tipo de sistema, ao mesmo tempo que mantém a mesma estrutura para qualquer tipo de sistema.

Tal linguagem é do tipo relacional, pois a descrição é feita através da especificação de vários objetos, e de todas as relações que os envolvem.

O SACDS deve ser capaz de manipular linguagens com a estrutura descrita acima, isto é, para aceitar a descrição de um determinado sistema, deve dispor da especificação dos tipos de relações e dos tipos de objetos que nele existem.

Tal especificação poderia ser fornecida ao SACDS, através de uma componente que deveria interpretar tais especificações, e codificá-las de maneira que pudessem ser facilmente utilizadas pelas demais componentes. Além disso, tal componente poderia aceitar também a especificação de como deveriam ser os relatórios sobre o sistema descrito, em um formato adaptado às necessidades daquele tipo de sistema.

Dispondo das especificações dos tipos de objetos e relações, o SACDS deveria ser então capaz de aceitar as especificações de cada sistema daquele tipo em particular, armazenando tais informações em uma base de dados. Isso poderia ser feito através de uma componente, que deveria verificar a coerência da descrição com as especificações do sistema, enquanto os dados fossem sendo fornecidos.

Uma vez dispondo da descrição do sistema o SACDS

estaria em condições de gerar relatórios sobre esse sistema. Alguns tipos de relatórios, que em geral são úteis a qualquer tipo de sistema, deveriam estar disponíveis a qualquer usuário do SACDS. Outros relatórios adaptados a um particular tipo de sistema, cujas especificações deveriam ter sido fornecidas à componente de especificações, deveriam estar disponíveis aos usuários de tal tipo de sistema.

Na figura 3.1 está representada a sequência que tais atividades deveriam seguir.

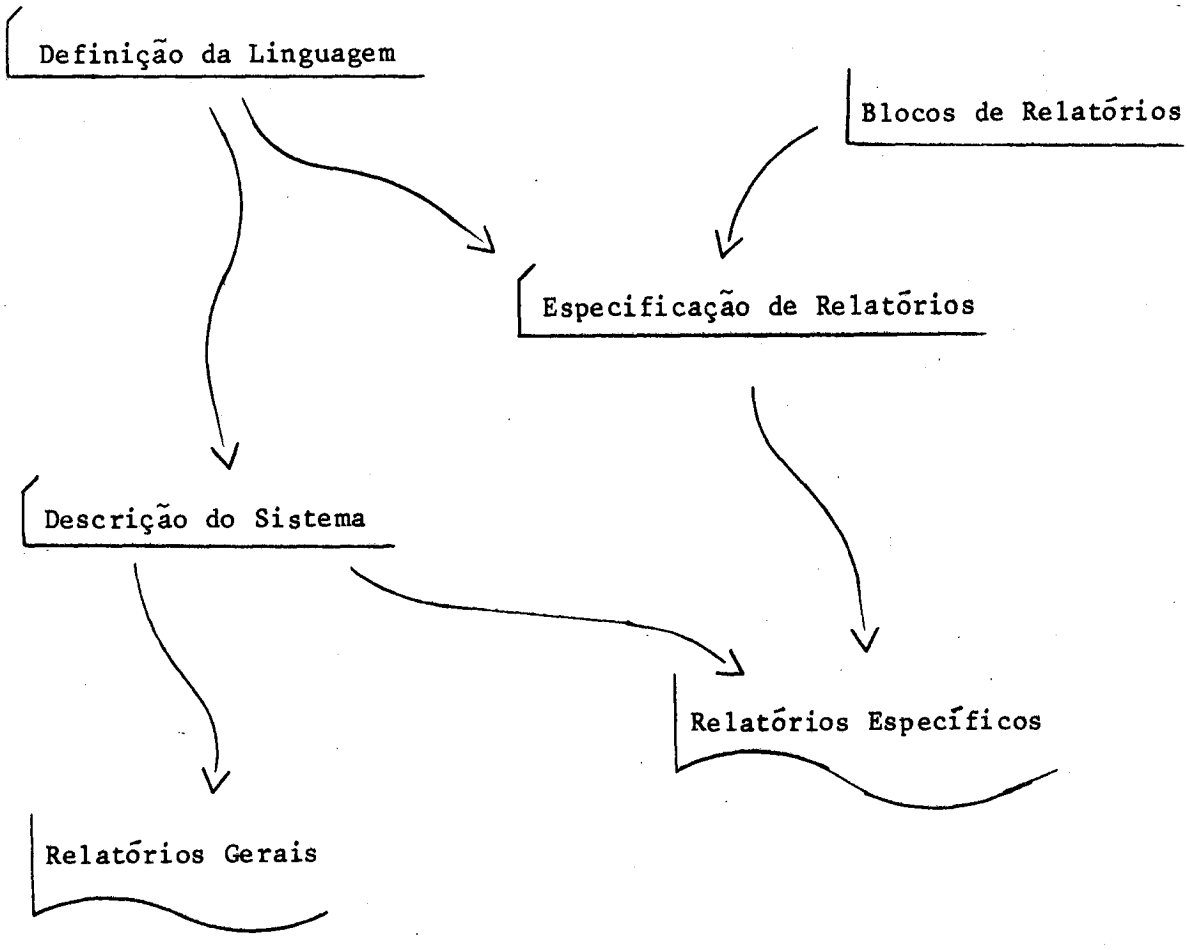


Figura 3.1

Para a geração de relatórios, em geral não é necessário envolver todos os objetos descritos, mas apenas parte deles. Para a seleção dos objetos de interesse, uma componente deveria ser capaz de receber do usuário especificações de quais objetos selecionar, com base em critérios de seleção.

O SACDS deveria ser capaz também de analisar a descrição de um sistema, para verificar se a descrição está logicamente completa e consistente. Como cada tipo tem particularidades a respeito desses aspectos, a especificação de o que significa estar completo e o que significa estar consistente em um determinado tipo de sistema, deveria ser também fornecida juntamente com as demais características daquele tipo de sistema.

3.3 - A Definição de um Tipo de Sistema

A característica do SACDS, de ser aplicável a qualquer tipo de sistema, obriga que sejam especificadas as características de um particular tipo de sistema, antes que seja possível usá-lo com sistemas desse tipo. Torna-se então necessária a existência de uma componente, que permita a definição de todas as características específicas de um particular tipo de sistema, e que as armazene em uma estrutura, de forma a permitir que sejam usadas pelas outras componentes do SACDS com facilidade.

Assim associado a uma definição de um determinado tipo de sistema, o SACDS pode se tornar uma ferramenta específica para a documentação e análise de sistemas desse tipo. A definição deve estar sempre disponível às demais componentes do

SACDS, e portanto deve ser preparada antes de qualquer outra operação. Não é necessário que se faça uma definição para cada sistema que se deseja usar com o SACDS, pois uma vez feita a definição do tipo de sistema, pode-se usá-la para quantos sistemas daquele tipo se desejar.

3.3.1 - Definição da Linguagem de Descrição

Uma das mais importantes definições necessárias é a da linguagem de descrição do sistema, onde se definem os tipos de objetos e de relações da linguagem, pois baseado nela é possível a definição de outras características do sistema. Para ser o mais geral possível, qualquer tipo de particularização neste ponto deve ser evitada, e a especificação deve ser feita utilizando-se a menor quantidade possível de símbolos e regras.

Tal definição deve poder ser feita listando-se todos os tipos de objetos, e todos os tipos de relações que um sistema do tipo sendo definido puder ter. Dispondo desses tipos, as demais especificações podem ser feitas já utilizando tais palavras como símbolos.

Para estabelecer a sintaxe da linguagem, devem poder ser especificadas a seguir, características das relações e a forma como elas se aplicam aos objetos.

As relações devem poder ser classificadas nas categorias: simples, triplas, sinônimos e comentários.

1 - Relações Simples:

São relações que envolvem apenas 2 objetos. Toda relação simples tem um objeto origem e um objeto destino. Para cada relação simples existe uma outra relação simples, chamada

relação oposta, que tem os objetos origem e destino trocados em relação à sua oposta.

2 - Relações Triplas:

São relações que envolvem mais de 2 objetos. Os objetos envolvidos em um relacionamento triplo, podem ser de 3 classes diferentes, distribuídos de tal forma que exista um objeto da classe 1, um objeto da classe 3, e um ou mais objetos da classe 2. Um relacionamento triplo pode ser visto como vários relacionamentos simples interligados, sendo sempre de 2 tipos (mais suas opostas). O objeto da classe 1 se relaciona com os objetos da classe 2 através de relações do 1º tipo, e se relaciona com o objeto da classe 3 através de 1 relação do 2º tipo. Como existe uma interligação entre essas relações, existe uma interligação entre as relações do 1º tipo com a do 2º tipo, e uma interligação da relação do 2º tipo com as relações do 1º tipo. A Figura 3.2 ilustra tal disposição:

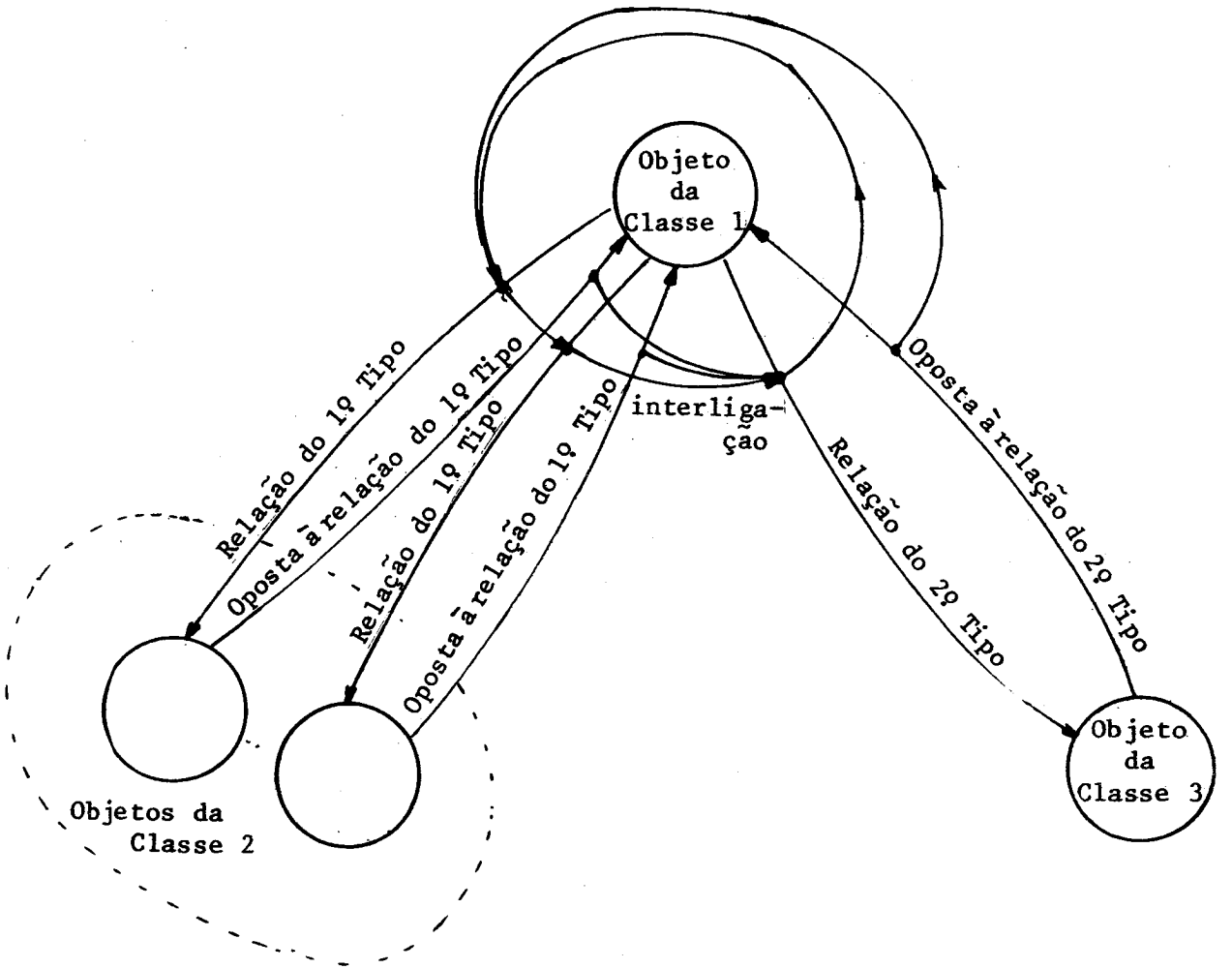

Objeto da Classe 1
Relação do 1º Tipo
Oposta à relação do 1º Tipo
Relação do 1º Tipo
Oposta à relação do 1º Tipo
interligação
Relação do 2º Tipo
Oposta à relação do 2º Tipo
Objetos da Classe 2
Objeto da Classe 3

Figura 3.2

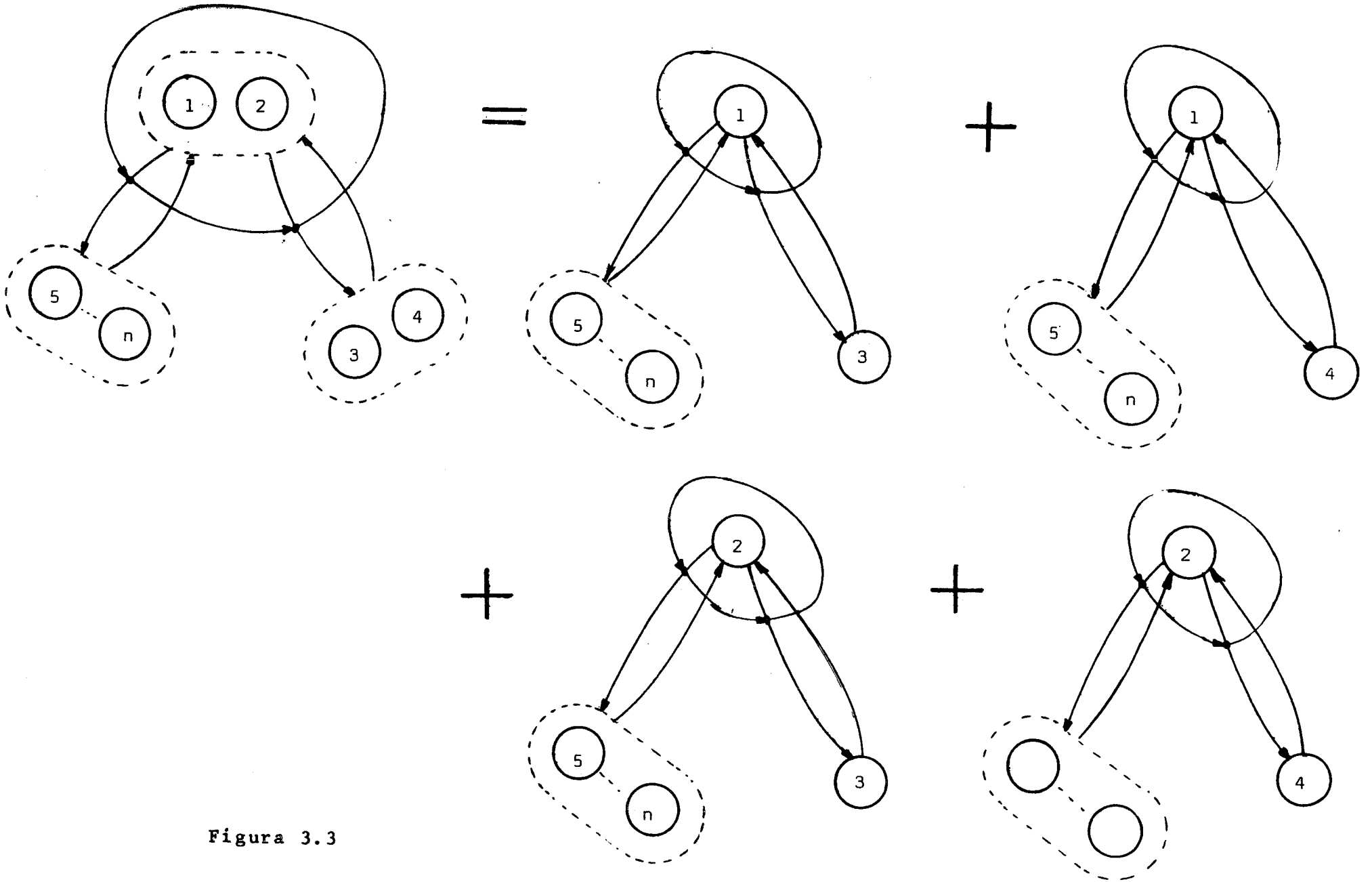


Figura 3.3

Dessa forma pode-se ter acesso a todos os objetos envolvidos em um relacionamento triplo, a partir de qualquer um deles através de uma relação (simples) e uma interligação. Tal forma de acesso, através de uma interligação e não através dos relacionamentos, é o que caracteriza a relação tripla.

Poder-se-ia pensar em uma maior generalização do relacionamento triplo, aceitando-se a existência de mais de um objeto nas classes 1 e 3, porém um relacionamento desse tipo, nada mais é do que relacionamentos iguais ao proposto repetido várias vezes, uma para cada objeto que ocupar a classe 1 ou 3 (Figura 3.3).

3 - Relações Sinônimo:

São relações de um objeto com ele próprio, que associam a esse objeto outros nomes. Todo objeto tem um tipo e um nome. Com uma relação sinônimo, seria possível associar mais de um nome ao mesmo objeto. O tipo de sinônimo é caracterizado pelo tipo de relação com que o nome está envolvido com o objeto.

4 - Relações Comentário:

São relações entre um objeto e um texto. Algumas características do tipo de Sistema que se pretende descrever, podem não ser importantes para serem descritas de maneira formal nesta linguagem, ou podem não ser adequadamente descritíveis desta forma (por exemplo, uma tabela). Nesses casos, seria conveniente permitir que fossem armazenados textos incorporados à descrição formal, que não fossem analisados pelo SACDS, mas apenas listados quando solicitados. Tais textos estariam

associados a objetos específicos podendo existir até vários tipos de textos. O tipo do texto seria caracterizado pelo tipo de relação que o associa ao objeto. Como um texto não é analisado, ele se comporta como um comentário dentro da linguagem .

Além dessas categorias, deve ser possível definir quais os tipos de relações que podem envolver cada tipo de objeto. Tal definição pode ser feita através de todas as ternas (<tipo de objeto origem>, <tipo de relação>, <tipo de objeto destino>) possíveis na linguagem.

3.3.2 - Definição de Parâmetros

O SACDS deve ser capaz de realizar uma grande variedade de operações, como por exemplo, aceitar e modificar uma descrição, selecionar dados específicos em uma descrição, formatar os dados selecionados de vários modos diferentes, etc.

Como os tipos de sistemas em que tais operações se aplicam são bastante variados, deve ser possível especificar diversos parâmetros que efetuam cada uma dessas operações para cada tipo de sistema.

Muitos desses parâmetros dizem respeito à forma pela qual o usuário poderia solicitar ao SACDS a execução de uma dada operação da forma como lhe interessa. Para isso seria conveniente a existência de uma pequena linguagem para cada uma das operações que o SACDS dispõe. Tais linguagens, deveriam ter todas uma mesma estrutura, e as operações que as instruções de uma linguagem desencadeariam, seriam as mais gerais possíveis. Na definição de uma linguagem, seriam escolhidas quais

operações seriam relevantes, e seriam fixadas algumas características, constantes para sistemas de tal tipo.

Para algumas formas de relatórios, que não seriam específicos a nenhum tipo de sistema, mas comuns a todos eles, a especificação dos parâmetros correspondentes seria feita no sentido de se especificar, para aquele relatório, como obter da descrição os aspectos que devem ser mostrados.

Outra característica que deveria ser possível especificar seria tipos de mensagem que as componentes do SACDS devem emitir, como por exemplo mensagens de erro, ou o resultado de análises ou pesquisas na descrição solicitada pelo usuário.

3.3.3 - Definição de Condições de Consistência e Completeza da Descrição

Quando é especificada a sintaxe da linguagem com a qual um sistema é descrito, especificam-se regras que definem todos os tipos de relações que podem envolver um determinado tipo de objeto. Tais regras devem ser verificadas para que uma determinada descrição seja aceita. Porém, para cada tipo de sistema, existem situações em que uma descrição, apesar de sintaticamente correta, não está semântica e logicamente correta. Isso pode ocorrer por exemplo quando um objeto de determinado tipo pode estar envolvido por relações de um certo tipo x, ou por relações de um certo tipo y, mas não pelas 2 relações ao mesmo tempo. A sintaxe da linguagem apenas indica que ambas as relações são admissíveis, nada indicando sobre a invalidade da existência de ambas simultaneamente. A existência na descrição de uma situação assim torna a descrição inconsistente.

Além disso, a sintaxe da linguagem não indica quando uma relação deve ocorrer, apenas indica que ela pode ocorrer. Se por exemplo objetos de determinado tipo precisarem sempre estar envolvidos por determinada relação, e na descrição existir um objeto daquele tipo que não tem tal relação, teremos uma descrição incompleta.

A verificação da coerência e completeza de um sistema pode ser uma tarefa complexa, porque não só as relações que envolvem diretamente um objeto necessitam ser verificadas. Pode existir regras de consistência e completeza que envolvem vários objetos interrelacionados de forma específica, e a verificação de tais situações significa percorrer uma descrição através de vários níveis de relacionamento.

O SACDS deve ter condições de executar testes em uma descrição, de acordo com regras de consistência e completeza especificadas para cada tipo de sistema. A maior ou menor complexidade de um teste, depende das regras que forem especificadas.

Para que um tal sistema seja flexível o suficiente para permitir a verificação da maior quantidade possível de situações, cada regra deve ser decomposta em regras as mais simples possíveis, envolvendo apenas um objeto origem, uma relação e um objeto destino. A partir dessas regras simples, serão então construídas regras que podem indiciar como percorrer uma descrição da forma que se desejar.

O fato de um objeto estar ou não de acordo com uma regra, pode ser visto como uma variável lógica, assumindo os valores verdade e falso, podendo assim ser tratada através da álgebra

booleana. Dessa forma, as regras podem ser baseadas em características que um objeto tenha ou não. Tais características podem ser combinadas em expressões booleanas, para criar qualquer regra que se queira. A possibilidade de haver características de um objeto que dependam do fato dele estar relacionado ou não com objetos de certa característica, permite que se estabeleçam regras que verificam profundamente a forma como um objeto está relacionado, em uma descrição.

Um conjunto de características que pode ser usado como base é o seguinte:

1 - Ser de determinado tipo - *Tipo (T)*

2 - Estar relacionado através de determinada relação - *Relação (R)*

3 - Ser da característica C_n e C_m - $E(C_n, C_m)$

4 - Ser da característica C_n ou C_m - ou (C_n, C_m)

5 - Não ser da característica C_n - *NÃO* (C_n)

6 - Ter mais do que determinado número de relações de determinado tipo - *NÚMERO(R, n)*

7 - Estar relacionado através de determinada relação com objetos de determinada característica - *RELACIONADO(R,* C_n*)*

8 - Ser um objeto de uma determinada seleção - *SELEÇAO (S)*

9 - Ter um objeto como destino de 2 de suas relações - *MESMO (R1, R2)*.

As mesmas regras podem ser usadas para se verificar tanto a consistência quanto a completeza e a validade de uma descrição. Para isso basta indicar a forma como a verificação deverá ser feita. Poder-se-ia dispor de 3 especificadores:

Validade: INV (C_n) - é inválido um objeto com a característica C_n .

Completeza: COMP (C_n) - para estar completa sua descrição, um objeto deve ter a característica C_n.

Consistência: CONS (C_n , C_m) - se um objeto tem a característica C_n , então deve ter a característica C_m.

As características que um objeto pode ter, são quaisquer das características básicas, ou quaisquer combinações delas. No quadro 3.1 estão exemplos simples de regras que podem ser especificadas dessa forma.

Para a implementação de recursos que permitam aos SACDS a verificação de condições de consistência e completeza, deve ser prevista a especificação de 2 outros parâmetros para cada regra: quais objetos são englobados pela regra, e quando o teste deve ser feito.

A regra deve ser verificada para cada objeto que satisfaça determinada condição, condição esta que pode ser especificada na própria regra. Porém, se antes de se fazer a verificação completa, os objetos passíveis de serem verificados forem selecionados, a verificação pode ficar muito mais eficí-

ente. Por exemplo, pode-se indicar que determinada regra deve ser testada com todos os objetos de determinado tipo.

A verificação das regras é uma atividade que demora algum tempo para ser executada. Idealmente, uma descrição deve ser testada quanto a todos os aspectos de coerência e completeza, sempre que uma alteração na descrição for efetuada. Porém, devido à complexidade de algumas regras, e à quantidade de regras que podem existir, e ao tempo que tal verificação levaria, ela deve ser feita apenas quando solicitada pelo usuário, ou quando algum evento especial e não frequente ocorrer . Além disso, seria de valia haver possibilidade de agrupar regras, de maneira que se pudesse solicitar a verificação independente de algumas regras, ou especificar que regras devem ser executadas quando determinado evento ocorrer. Por exemplo, poder-se-ia selecionar algumas regras simples de rápida verificação, e bastante abrangentes, para serem verificadas mesmo sem solicitação, sempre que terminasse uma operação de modificação de descrição.

3.3.4 - Definição de Relatórios

A possibilidade de se conseguir usar um computador para apoiar o desenvolvimento e a documentação, da maior gama possível de tipos de sistemas com o SACDS, depende em grande escala de sua capacidade de gerar relatórios expressivos das informações contidas na descrição que lhe é fornecida. Deve existir um módulo bastante flexível para a geração de relatórios que baseado em especificações a respeito de sua forma, gere o relatório desejado.

Validade:

- Limite da quantidade de relações tipo *R* que objetos do tipo *T* podem ter é *n:*

  INV(E(Tipo(T), Número(R,n))).

- Um objeto não pode estar relacionado com outro através das relações *R1* e *R2* simultaneamente:

  INV(MESMO(R1,R2)).

- Um objeto não pode ao mesmo tempo ter relações dos tipos *R1* e *R2* :

  INV(E(RELAÇÃO(R1), RELAÇÃO(R2))).

Completeza:

- Um objeto tipo *T* deve ter no mínimo 1 relação tipo *R* :

  COMP(E(Tipo(T), Número(R,1))).

- Todo objeto do tipo *T* que não tenha uma relação tipo *R1* deve ter uma relação tipo *R2* :

  COMP(E(Tipo(T), ou (RELAÇÃO(R1), RELAÇÃO(R2)))).

Consistência:

- Se um objeto tem uma relação tipo *R1*, ele deve ter uma relação tipo *R2* :

  CONS(RELAÇÃO(R1), RELAÇÃO(R2))

- Se o objeto destino de uma relação tipo *R* de um determinado objeto é do tipo *T* , então todos os objetos destino das relações tipo *R* desse objeto devem ser do tipo *T*:

  CONS(RELACIONADO(R,TIPO(T)), NÃO(RELACIONADO(R,NÃO(TIPO(T))))).

Quadro 3.1

Em geral, um relatório é constituído de 3 partes: um cabeçalho, contendo uma identificação de qual o assunto e a-brangência do relatório; um corpo, que em essência constitui o relatório propriamente dito; e um resumo, finalizando-o com alguma informação que globaliza as informações contidas no relatório.

A informação contida em cada uma das partes é sempre obtida diretamente ou através de alguma manipulação das informações contidas na descrição do sistema. O formato pelo qual tais informações são colocadas no relatório é que deve então ser específico de cada relatório. Os formatos que em geral são utilizados podem ser padronizados, e constituem módulos que podem ser combinados para se conseguir o relatório da forma desejada.

Dependendo do formato que cada módulo produz, uma série de parâmetros deve ser especificada. Alguns desses parâmetros devem poder ser obtidos de resultados de módulos utilizados em partes anteriores dos relatórios, para que seja possível a geração dos relatórios de acordo com as informações obtidas quando da geração do cabeçalho, bem como para possibilitar a produção do resumo final do relatório.

Os módulos que são utilizados em qualquer uma das 3 partes do relatório devem ser sempre os mesmos, para uma maior facilidade na especificação dos módulos, e também uma maior flexibilidade, pois permite que cada parte seja composta por qualquer combinação dos formatos disponíveis.

Os formatos disponíveis devem ir desde simples nomes de objetos colocados em determinada posição, até tabelas ,

matrizes de dados, etc., sendo desejável até a existência de formatos gráficos.

Para ser possível a geração de relatórios que contenham várias partes distintas, devem existir também módulos que executam funções de controle dos outros módulos, ao invés de produzir um determinado formato. Devem então existir módulos de seleção de objetos, de decisão para a escolha de alternativas para a continuação do relatório, etc.

A combinação de vários módulos, para se especificar como o relatório deve ser gerado, deve ser feita através dos parâmetros que especificam como um módulo deve ser, em cada situação. Cada módulo deve ter um conjunto de parâmetros que defina completamente o relatório, sendo portanto diferentes para cada módulo. Porém os parâmetros devem ser padronizados em vários tipos, para permitir combinações as mais variadas possíveis.

3.4 - O Preparo da Descrição

Associado à descrição de um tipo de sistema, o SACDS torna-se uma ferramenta capaz de auxiliar a: analisar, desenvolver e a documentar qualquer sistema daquele tipo. Para isso, é necessário que seja feita uma descrição do sistema, sempre atualizada, sobre a qual todas as demais operações se apoiarão. A descrição deve conter todos os dados do sistema em todos os aspectos em que o SACDS deve auxiliar, bem como, dados a respeito de como o desenvolvimento deve ser feito, recursos disponíveis, pessoas envolvidas, tempos associados a qual-

quer desses dados, etc. Enfim, qualquer informação que possa ser útil à análise, desenvolvimento e documentação do sistema nos aspectos em que se deseja usar o SACDS.

De uma maneira geral, a coleta de dados para a elaboração dessa descrição, pode ser feita tanto pelas pessoas envolvidas quanto pelo próprio computador. Dados a respeito do sistema fornecidos por pessoas, podem estar na forma de descrições feitas em uma linguagem preestabelecida, textos, tabelas, ou até gráficos. Dados que o próprio computador pode coletar seriam conseguidos de uma monitoração feita em um sistema em que pelo menos parte dele já estivesse em atividade.

Este trabalho se restringirá a informações fornecidas ao SACDS através de descrições feitas em uma linguagem preestabelecida, porque assim é possível fornecer todas as informações que se poderia fornecer com os demais métodos, e porque, além de constituir a maneira pela qual a grande maioria das informações são normalmente fornecidas, os demais métodos, de qualquer forma, necessitam de uma linguagem que descreva ao menos como os próprios dados estão dispostos ou como monitorar tais dados.

A componente do SACDS com a função de preparar a descrição, deve então ser capaz de analisar a descrição feita na linguagem pre-definida, verificar sua sintaxe e armazenar os dados fornecidos de forma a torná-los disponíveis às demais componentes. Além disso, deve dispor de recursos que permitam ao usuário, efetuar qualquer alteração na descrição.

3.4.1 - A Elaboração da Descrição

Entre outras características que definem um tipo de sistema, figura a linguagem de descrição de sistemas, composta de símbolos que identificam os tipos de objetos e de relações, de caractarísticas das relações, e da forma como as relações se aplicam aos vários tipos de objetos. Tal linguagem, acrescida de comandos que permitam solicitar ao SACDS várias operações que auxiliam a tarefa de se preparar uma descrição, deve constituir a linguagem de elaboração da descrição.

A elaboração da descrição normalmente é feita em etapas sucessivas, em que novos dados são acrescentados, e alterações nos dados já fornecidos são feitas. O módulo que deve executar tais operações pode ser visto como um editor, que manipula a descrição, e seja capaz de executar operações como inclusão, retirada e substituição de dados na descrição.

Essas operações são, em resumo as operações que sempre se realizam para se elaborar uma descrição, qualquer que seja a forma que ela assuma. No entanto, no caso da componente do SACDS que realize tais operações, algumas variantes auxiliariam em muito.

Devem existir comandos que permitam incluir, retirar e substituir dados em pequena escala, propiciando condições de se efetuar correções localizadas em qualquer ponto da descrição, assim como comandos que permitam essas mesmas operações em grandes volumes de dados, quando se necessitam correções envolvendo todo um determinado aspecto da descrição, ou mesmo toda a descrição. Essas operações devem poder ser realizadas através de comandos diretos do usuário, ou através de

instruções que realizem essas operações automaticamente, por exemplo a correção de dados semelhantes em vários pontos da descrição.

Sempre que o usuário estiver elaborando a descrição de um sistema, os dados em que ele está interessado, estão interrelacionados, e seria de grande conveniência que o usuário dispusesse continuamente de informações detalhadas dos aspectos da descrição, tal como disponível ao computador naquele instante, nos pontos de seu interesse. Isso pode ser feito, se o usuário comunicar-se com o computador através de um terminal tipo vídeo. Nesse caso, seria possível a escolha de quais as informações seriam desejáveis, e o terminal teria sempre aquele tipo de informações, mostradas em relação aos objetos da descrição onde o usuário estaria alterando a descrição.

Muitas vezes, as operações executadas durante a elaboração, ou parte da elaboração da descrição, envolvem objetos nos quais poderá haver interesse posterior. Deve então ser possível indicar que determinados objetos devem ser selecionados, e essa seleção deve ser armazenada para referência futura.

Todas essas operações tornam o programa apto a operar iterativamente com o usuário, proporcionando-lhe condições de manipular a descrição desde pequenas alterações localizadas, até operações envolvendo grande parte da descrição, ao mesmo tempo em que o informa da situação da descrição, nos aspectos de seu interesse.

3.4.2 - Testes de Sintaxe

Existem 3 tipos de erros de sintaxe que o usuário pode cometer com a linguagem usada para a elaboração da descri

ção de seu sistema. Conforme foi dito na seção anterior, essa linguagem é composta da linguagem de descrição de sistema, e dos comandos que efetuam as operações.

O primeiro tipo de erro, envolve a sintaxe dos comandos das operações, e como tal não apresentam nenhum caso especial. Apenas deve ser previsto, principalmente para os comandos que causam alterações de grande parte da descrição, mecanismos especiais que impeçam que, inadvertidamente o usuário ative tais operações. Em casos extremos, como por exemplo, quando o comando indica a operação de retirar da descrição muitos dados, seria conveniente a necessidade de uma confirmação por parte do usuário.

O segundo tipo de erro envolve a sintaxe da linguagem de descrição de sistemas. Durante a elaboração da descrição, os procedimentos normais de teste de sintaxe dessa linguagem pelo SACDS são normalmente seguidos. Porém, como existem comandos que afetam vários objetos, os testes de sintaxe devem ser mais profundos, envolvendo todos os objetos afetados. Dependendo dos tipos de objetos envolvidos, tais considerações podem restringir o uso da linguagem aos aspectos comuns aos tipos de objetos envolvidos.

A existência de comandos que permitem a modificação automática de parte da descrição, leva à necessidade de se usar os recursos de teste de sintaxe para controlar essas operações. Uma forma de se realizar isso é dividindo os procedimentos de teste em uma parte de verificação de sintaxe, que apenas indica a validade ou invalidade de uma frase, e uma parte de execução que toma diferentes atitudes dependendo do resulta

do da verificação. No caso de um teste de sintaxe, a parte de execução aceitaria ou rejeitaria a instrução ao passo que no caso de um comando automático, a parte de execução auxiliaria a construção das estruturas usadas para a execução do comando.

O terceiro tipo de erro, envolve a verificação da compatibilidade das operações que devem ser executadas com as informações já contidas na descrição. Esse erro pode ocorrer quando se executa uma operação de inclusão ou alteração na descrição. Um exemplo de quando esse tipo de erro pode ocorrer, é quando se estabelece o tipo de um objeto antes indefinido, ou quando se altera o tipo desse objeto. As relações que o objeto já tem, devem ser verificadas quanto à sua validade para com o novo tipo.

A análise de erros desse tipo, envolve a definição da sintaxe da linguagem de descrição e a definição das condições de consistência da descrição. Tal análise é feita através da verificação dos dados já presentes na descrição. Dependendo do volume de dados associados aos objetos sendo verificados, o tempo gasto pode ser bastante grande, pois é necessária uma busca de todos os dados da descrição referente a cada um dos objetos envolvidos na alteração da base requisitada. Levando-se em conta que muitas das verificações seriam repetidas em decorrência de comandos sucessivos, pode-se pensar em deixar os testes mais demorados para serem feitos após o usuário terminar suas operações de edição da descrição.

Uma alternativa satisfatória seria verificar a sintaxe da linguagem para cada comando, mantendo sempre a descrição sintaticamente correta, e verificar as condições de consistência apenas uma vez, no final da edição.

Conforme foi dito na seção 3.3.3, as condições de consistência testadas seriam apenas aquelas escolhidas pelo usuário para tal, mantendo-se ainda a possibilidade de que condições específicas fossem testadas quando requisitado.

Para tornar possível a execução dos testes dessa forma, é necessário que os objetos afetados em qualquer operação de alteração da descrição sejam marcados. Tal tarefa pode ser realizada através da criação de uma seleção de objetos feita pelo programa de elaboração da descrição automaticamente, contendo os objetos afetados. Tornando essa seleção posteriormente disponível ao usuário, tem-se a vantagem de poder gerar qualquer relatório sobre os objetos alterados.

Outra alternativa seria ter mais um conjunto de condições de consistência (e possivelmente de completeza) que seria sempre verificado para cada comando, além do teste posterior realizado ao final da edição.

Essa alternativa aumenta o tempo gasto em relação à alternativa anterior, para o sistema executar cada comando. Porém as condições indicadas nesse conjunto são escolhidas pelo usuário, e portanto, caso o aumento de tempo torne a edição desconfortável, é de sua escolha transferir ou não qualquer teste para o conjunto de testes que deve ser verificado no final da edição, ou mesmo para um conjunto que deve ser verificado apenas quando requisitado.

A vantagem dessa alternativa é que ela possibilita o fornecimento de diagnósticos à medida que os comandos são emitidos, permitindo ao usuário tomar medidas corretivas imediatamente, ou mesmo alterando o sentido de sua edição. Tal alter

nativa seria de especial valia para usuários com um conhecimento restrito do sistema ou do SACDS. Nesse caso, os testes e as mensagens associadas poderiam ser escolhidas de maneira a orientar o usuário para os procedimentos padronizados pela equipe.

3.4.3 - A Armazenagem da Descrição

A descrição de um sistema deve ser armazenada de forma que a ela se possa ter acesso eficiente, para a realização das várias operações que o sistema SACDS efetua. A estrutura que armazena a descrição constitui uma base de dados, que deve prover facilidades para armazenagem adequada de todos os tipos de dados descritos em um sistema.

Pode-se considerar que existem 4 tipos de dados que formam uma descrição: objetos, com suas várias propriedades, como por exemplo tipo, nome, sinônimos, etc.; relações que descrevem como os objetos interagem, com seus tipos, número de objetos envolvidos, etc.; comentários associados aos objetos, descrevendo informações que não têm necessidade de serem analisadas pelo SACDS, e por esse motivo descritas sem necessidade de qualquer formato ou linguagem específica; seleções com as mesmas propriedades, propriedades essas, específicas de cada sistema descrito.

As operações fundamentais efetuadas na base de dados são: busca de um objeto; listagem de qualquer um dos 4 tipos de dados; inclusão de objetos, relações e comentários; retirada de objetos, relações e comentários; alteração em qualquer propriedade dos objetos e em relações; criação de seleções; cancelamento de seleções; e navegação das relações.

A primeira característica que se deve levar em conta em uma base de dados é o que diz respeito à sua dimensão. Dependendo da forma como uma descrição é feita, a proporção da quantidade de dados de vários tipos pode variar livremente. A estrutura usada na base de dados deve ser tal, que os únicos limites à expansão da descrição sejam os impostos pelo equipamento onde o SACDS opera.

A armazenagem dos objetos é feita através de uma sequência contendo em cada elemento da sequência todos os parâmetros de um objeto. Para tornar eficiente a busca de um objeto (o que se faz através de seu nome ou através de um sinônimo seu), deve existir uma estrutura de dados envolvendo todos os objetos. Estruturas adequadas, pela facilidade com que permitem operações de inclusão e retirada de dados, bem como pela rapidez de busca, são estruturas de árvore e de partição (hashing). Uma estrutura em árvore alia ainda a vantagem de facilitar uma listagem ordenada dos objetos. Uma estrutura em partição não contribui para as operações de listagem, mas pode ser mais eficiente na busca que uma estrutura em árvore.

As propriedade dos objetos não são em geral necessárias, antes que o nome do objeto tenha sido localizado na descrição, e portanto podem ser armazenadas simplesmente justapostas a este. Os sinônimos do objeto constituem a única propriedade que necessita uma atenção maior. Estes podem ser armazenados como se fossem nomes normais de um objeto, porém deverão conter uma indicação de seu estado de sinônimo, juntamente com um ponteiro especificando qual o objeto do qual ele é sinônimo e uma indicação de qual a forma pela qual ele é um sinônimo (abrevia

tura, tradução, "sinônimo", etc.), formas estas que dependem da linguagem de descrição definida pelo usuário. O objeto principal por sua vez deve ter uma lista de todos os seus sinônimos. Feito dessa forma, a propriedade "sinônimos de um objeto" pode passar a ser vista como um tipo de relação entre um objeto e ele próprio, usando um outro nome.

A armazenagem das relações deve levar em conta as suas várias características. A armazenagem de relações simples e triplas pode ser feita usando-se várias formas de armazenagem em lista. Como as relações triplas podem ser decompostas em várias simples entrelaçadas (conforme foi mostrado na seção 3.3.1) , a mesma estrutura de dados pode ser usada em ambas as formas, apenas com a adição de um campo capaz de prover a interligação das relações triplas. Uma discussão detalhada quanto à melhor forma de representação de relações simples em uma base de dados foi descrita em [Tr 78] .

A armazenagem de relações tipo comentário também pode ser feita através de uma lista, porém por relacionar apenas um texto a um objeto, essa lista pode ser feita como uma lista simples unidirecional, interligando cada linha (registro) dos comentários referentes a um objeto. A identificação de cada tipo de comentário pode ser feita em cada linha, e mantendo ordenados os registros por esse tipo, teremos os textos correspondentes a cada tipo de comentário diretamente acessíveis através de registros consecutivos da lista.

Dispondo dos objetos, relações e comentários já armazenados, pode-se considerar a descrição de um sistema completa, a menos da especificação de quais dos objetos descritos go

zam de determinadas características, próprias de cada sistema. Um sistema poderá dispor de tantos conjuntos de objetos desses, chamados arquivos de seleção, quantos necessários, sem nenhuma restrição quanto a quais objetos pertencem a cada conjunto. A especificação dos arquivos de seleção pode ser feita através de um programa específico, que permite ao usuário indicar quais são os objetos pertencentes a cada arquivo, explicitamente ou indicando propriedades dos dados já descritos, ou através do resultado de operações executadas pelos vários comandos do SACDS.

A armazenagem dos arquivos de seleção pode ser feita simplesmente indicando-se sequencialmente, quais os objetos que pertencem a cada arquivo. A quantidade limite de arquivos de seleção dependerá apenas da capacidade que o usuário dispõe para armazená-los em seu equipamento.

3.5 - Testes de Consistência e Completeza

Para cada tipo de sistema, existem situações nas quais, apesar de sintaticamente correta, uma descrição contém incorreções que podem passar despercebidas pelo usuário. A verificação de todas as possibilidades de erro, ou pelo menos, daquelas que possam ser importantes para o usuário, deve ser possível sempre que solicitada por ele, ou sempre que ocorra algum evento significativo, durante a operação do SACDS.

Os testes efetuados para cada tipo de sistema devem ser definidos quando é feita a definição das características de sistemas daquele tipo. Os testes devem ser agrupados, e

quando é necessária uma verificação, todos os testes de um grupo devem ser verificados.

Para cada teste, existem 2 fases de operações que devem ser executadas antes que se possa concluir algo a respeito da descrição, quanto àquele teste: iniciamente deve-se proceder à escolha de quais são os objetos da descrição que devem ser verificados, e a seguir é efetuada a verificação propriamente dita.

Em geral cada grupo de testes deve estar voltado para a verificação de determinados aspectos da descrição, e portanto é possível que mais de um teste do grupo deva agir sobre um mesmo conjunto de objetos. Nesse caso, o procedimento de escolha dos objetos que devem ser verificados é comum a esses testes. Para evitar procedimentos repetitivos, os grupos de testes devem estar organizados de forma que os testes de cada grupo que verificam os mesmos objetos sejam executados um após o outro, criando-se então sequências de testes dentro de cada grupo. A organização de cada grupo de testes deverá então ser uma ou mais sequências de testes, cada sequência envolvendo um ou mais testes que agem sobre o mesmo conjunto de objetos. O formato de definição dos grupos de testes deve refletir essa organização.

A escolha dos objetos que serão afetados por um (ou vários) testes, pode ser feita de forma semelhante àquela que é empregada para a criação de arquivos de seleção. A diferença principal é que no caso da escolha de objetos para testes, a especificação da escolha é feita baseada apenas em propriedades já descritas para o objeto, e é feita antes que a descri -

ção exista, durante a fase de definição das características de sistemas desse tipo. As seleções criadas para a verificação de um grupo de testes não devem ser colocadas à disposição do usuário, para evitar que, mesmo após alterações na descrição, a mesma seleção seja usada caso os mesmos testes voltem a ser verificados. Essas seleções devem ser criadas sempre que os testes forem efetuados. Porém, uma seleção dos objetos que mostraram incorreções diante de ao menos um teste do grupo pode ser de valia para o usuário, e seria desejável a possibilidade do usuário poder solicitá-la.

Quando é feita a especificação de um teste, deve ser estabelecida uma mensagem que deve ser impressa sempre que um objeto se mostra incorreto para aquele teste. Se o teste for de verificação simples, a mensagem pode simplesmente indicar que tal teste falhou em determinado objeto. Porém testes mais complexos devem ter condições de fornecer mensagens que especifiquem melhor onde o teste falhou. Devem existir então meios de se especificar como uma mensagem deve ser, montando frases que constam de textos fixos entremeados com informação retirada da descrição pelo procedimento de testes. Padronizado os tipos de informação que podem ser obtidos da base (que podem ser nomes de objetos, tipos de objetos ou de relações ou outras propriedades dos objetos), a especificação de uma mensagem , na maioria dos casos pode ser feita com facilidade. Porém a obtenção dos dados necessários para se completar uma mensagem, a partir dos dados manipulados para se verificar o teste, poderia levar a mensagens imprecisas ou sem significados para o usuário. Isso porque a verificação de um teste pode ir buscando

informações na base recursivamente, obtendo um volume explosivo de informações. A escolha correta de todas as informações que efetivamente levam à constatação do erro deve ser empreendida, e transcrita então para o usuário. Isso é feito voltando-se, a partir do objeto onde o erro foi constatado (ou a partir da ausência de uma relação necessária, etc.) cada uma das buscas à descrição até o objeto inicial onde o teste começou descartando-se todas as demais informações. Técnicas detalhadas para essa operação podem ser encontradas em [He 76], bem como técnicas de representação dos dados obtidos, de forma a facilitar seu uso com esse objetivo.

3.6 - Geração de Relatórios

O sistema de Apoio por Computador à Documentação de Sistemas deve ser composto de várias componentes, cada uma voltada à execução de uma determinada operação, e que deve ser na medida do possível, a única que executa tal operação. De forma geral, as várias operações podem efetivamente ser modularizadas, embora em alguns casos resultem módulos altamente complexos. Uma operação que dificilmente pode ser bem executada por um único módulo é a geração de relatórios, dada a grande diversidade de relatórios que podem ser necessários.

De forma geral, pode-se dividir os relatórios que um determinado tipo de sistema deve gerar em 2 categorias: na primeira estariam relatórios comuns a qualquer tipo de sistema, relatórios que em geral são necessários a sistemas de qualquer tipo; na segunda categoria estariam relatórios específicos de um particular tipo de sistema.

Tais categorias devem ser observadas pelo SACDS, que deve gerar os relatórios comuns sem que o usuário se preocupe em definí-los extensivamente. Quanto a relatórios específicos, o SACDS deve permitir a descrição do relatório em detalhes, e a partir dessa descrição (acessível apenas a sistemas do tipo para o qual ela foi preparada), gerar tal relatório.

Entre os relatórios específicos de um tipo de sistema, alguns relatórios necessitam uma pesquisa na descrição do sistema que envolve uma navegação da base de dados, enquanto que a maioria dos relatórios específicos necessitam de uma pesquisa de muito menor envergadura. Como a especificção de pesquisas que envolvem navegações são em geral bastante diferentes das demais, o SACDS deveria poder gerar essas 2 formas de relatórios de forma independente.

Em vista disso, poder-se-ia separar a parte de geração de relatórios em três componentes: geração de relatórios padronizados, geração de relatórios específicos; geração de relatórios de navegação. Essa separação permite que cada componente se especialize no seu tipo de relatório, permitindo que se produzam relatórios mais sofisticados, principalmente nas componentes que geram relatórios específicos e de navegação.

Sob alguns aspectos, tal separação pode levar à duplicação de algumas funções, porém como a natureza das operações necessárias à geração de cada tipo de relatório muda muito de uma componente para outra, a maior parte das funções não serão duplicadas. Entre as funções que efetivamente existem em mais de uma componente, a maior parte delas não aparecem de forma semelhante nas componentes, pois as necessidades das várias componentes são muito diferentes. Nesses casos a separação bene

ficia a execução dessas funções, que podem ser feitas em cada componente da forma mais adequada àquela componente. Por exemplo, as funções necessárias à geração de relatórios padronizados são muito mais simples e diretas do que quando necessárias para a geração de relatórios específicos. Dessa forma, as desvantagens de se ter duplicadas as funções resultam atenuadas, frente à maior especificidade conseguida no uso dessas funções.

Uma função necessária a qualquer tipo de relatório, e semelhante em todos eles é a seleção dos objetos que devem ser usados como objetos centrais para a geração dos relatórios. Ao mesmo tempo, essa é uma função que, para qualquer tipo de relatório, pode ser conveniente ter disponível a possibilidade de se definir os critérios de seleção quando da geração efetiva do relatório (ou pelo menos, da possibilidade de refinar tais critérios). Assim, seria conveniente que o SACDS dispusesse, além das 3 componentes para gerar 3 tipos de relatórios uma componente específica para aceitar do usuário, a qualquer instante, comandos de seleção de objetos para a criação de seleções a serem submetidas às componentes de geração dos relatórios.

3.6.1 A Seleção de Objetos

Sempre que existe uma descrição de um sistema, existem vários aspectos dessa descrição que sucitam interesse de diferentes maneiras, em geral no sentido de se desejar informações de várias formas sobre os objetos que constituem esses aspectos. Cada um desses aspectos deve ser reconhecido pelo SACDS, armazenando os objetos na forma de um conjunto de seleção de

objetos.

A criação de um conjunto de seleção pode ser realizada de muitas formas:

- decorrente de operações efetuadas pelo usuário;
- decorrente da atividades de alguma componente;
- por propriedades dos objetos especificados na descrição;
- por combinações de conjuntos já existentes.

Os conjuntos obtidos a partir de operações efetuadas pelo usuário, por exemplo os que indicam os objetos acrescentados na descrição em uma dada seção de atualização, ou os objetos que se relacionavam com objetos retirados da descrição. São criados pelas componentes automaticamente, porém o usuário deve poder solicitar ou inibir a criação. Por serem criadas de forma automática, somente será possível ao usuário obter seleções quanto a aspectos que tenham sido previstos quando da confecção da componente correspondente. Por exemplo, não será possível ao usuário obter o conjunto dos objetos dos quais ele alterou o nome em uma dada seção de atualização se tal previsão não tiver sido feita.

A variedade das operações efetuadas em uma componente é em geral extensa, e a existência da possibilidade de selecionar os objetos, de forma independente, em cada uma das operações, leva a um aumento muito grande na complexidade das componentes, e da própria estrutura de dados e comandos postos à disposição dos usuários.

Esses dois fatos opostos (possibilidade de o usuário solicitar uma seleção dos objetos afetados por qualquer operação que tenha comandado, e complexidade que isso causa), po-

dem ser compatibilizados se for levado em conta que vários conjuntos podem ser combinados para se obter outros. Assim pode-se construir componentes com capacidade para criar automaticamente conjuntos de seleção resultantes de algumas operações apenas. O que se deve levar em conta é a possibilidade de, com os conjuntos criados, se obter posteriormente os conjuntos dos objetos afetados por qualquer operação. Dessa forma, a escolha de quais conjuntos podem ser criados automaticamente deve ser criteriosa, para se conseguir esse objetivo.

Os conjuntos conseguidos a partir da atividade de alguma componente são os resultantes de atividades como por exemplo testes da base de dados, geração de relatórios ou operações realizadas na base por programas específicos. Frequentemente a sua criação é o objetivo da própria operação, como por exemplo quando se deseja conhecer quais objetos tem sua descrição logicamente incompleta, em que a criação de conjuntos de seleção faz parte da própria componente. Em outros casos, a criação dos conjuntos pode ser apenas um subproduto da atividade principal, por exemplo quando se deseja uma seleção com os objetos envolvidos de determinada forma em um determinado relatório. Nesses casos, a criação deve ser opcional, somente sendo feita quando solicitada pelo usuário, e deve estar prevista na componente que gera os relatórios padronizados, para permitir a seleção dos objetos nos casos mais significativos. Porém a própria amplidão de situações que podem ocorrer impede que a solicitação de tais seleções seja completa.

Quanto às componentes que geram os relatórios específicos e de navegação, tais seleções poderiam ser muito mais

completamente especificadas, pois possibilita ao próprio usuário definir quais seleções devem ser criadas por um dado relatório, quando esse relatório é definido. A efetiva criação de uma seleção prevista quando da definição do relatório deve continuar sendo controlável no instante da geração do relatório.

Para se conseguir criar conjuntos de seleção a partir das propriedades dos objetos, ou a partir de combinações de conjuntos já existentes deve existir no SACDS uma componente especial para esse propósito. Tal componente deve ser capaz de aceitar comandos do usuário para localizar qualquer seleção já existente, para combiná-las de forma como ele especificar, para criar conjuntos novos baseados em propriedades dos objetos descritos, bem como para criar conjuntos novos baseados em relacionamentos dos objetos de seleções já existentes.

Os conjuntos já criados anteriormente, não importa de que forma, devam ser igualmente acessíveis a essa componente para que ela possa combiná-los, criando novos conjuntos. A forma de combiná-los deve ser especificada pelo usuário em uma linguagem simples, com a maior parte dos elementos da linguagem tomados das definições feitas pelo próprio usuário, pois quanto menor for a quantidade de símbolos impostos pelo sistema, mais próxima da linguagem que o usuário está acostumado a usar no tratamento de seus sistemas será a linguagem de seleção, principalmente porque, não importa como seja feita a especificação, as operações serão sempre as usuais da teoria dos conjuntos.

Para se obter novos conjuntos de seleção baseados nas propriedades dos objetos, essa componente deve ser capaz de receber do usuário indicação das propriedades desejadas,

tais como tipo do objeto, existência ou não de sinônimos, datas de criação e/ou atualização, etc., e efetuar a seleção correspondente.

Deve por fim ser capaz de obter seleções de objetos que se relacionam de forma específica com os objetos de seleções já existentes. Os níveis de relacionamento aceitos devem poder ser razoavelmente extensos, porém não é necessário que sejam recursivos, pois tal capacidade é provida pela componente de geração de relatórios de navegação.

A forma proposta de armazenagem dos objetos que constituem determinado aspecto de interesse da descrição em conjuntos de seleção, que podem existir em qualquer quantidade, deixa os conjuntos com um grau de independência muito grande em relação à base de dados que constitue a descrição. Tal independência pode levar a problemas quando a descrição é alterada depois que um conjunto de seleção é criado. Esses problemas são os seguintes:

- uma seleção pode se tornar incompleta quando são incluidos na base novos objetos com as mesmas propriedades que foram usadas para se gerar aquela seleção.

- uma seleção pode ter objetos que fogem das características que se especificou para aquela seleção, se objetos incluídos nessa seleção tiverem suas propriedades alteradas.

- uma seleção pode conter a indicação de objetos que não mais existem por terem sido eliminados da descrição.

Os dois primeiros problemas podem ser solucionados pelo próprio usuário se lhe interessar, criando novamente uma seleção da mesma forma como tinha criado a anterior. O terceiro problema pode aparentemente ser resolvido acrescentando-se à

componente de seleção de objetos um recurso que lhe permita recuperar seleções antigas e a partir delas criar novas seleções contendo apenas os objetos que ainda estão presentes na descrição. Tal recurso no entanto pode levar a resultados incorretos pois no caso de ter havido eliminações e a seguir inclusões de objetos na descrição, pode não ser possível reconhecer se um objeto é original ou se apenas está ocupando o lugar de outro. Torna-se pois aconselhável que qualquer problema devido à independência dos conjuntos de seleção, seja resolvido sempre pela recriação do conjunto diretamente pelo usuário.

3.6.2 - A Geração de Relatórios Padronizados

A geração de um relatório de determinado formato baseado na descrição de um sistema de um tipo genérico, na maior parte das vezes é quase sem significado. Porém, alguns formatos são mais universais e são significativos para a grande maioria dos tipos de sistemas. Mas apesar dessa universalidade, não necessariamente esses relatórios são de fácil definição. Além disso é desejável a possibilidade de se obter relatórios imediatamente após a definição do tipo de sistema e uma descrição, mesmo sumária de um sistema daquele tipo, deixando a definição de relatórios específicos para quando a definição do sistema já estiver completa.

Por essas razões, seria de grande utilidade a existência de uma componente no SACDS que pudesse gerar relatórios em formatos predeterminados necessitando para isso de um mínimo de especificações.

Um relatório frequentemente necessário é uma listagem da descrição, ou de objetos selecionados da descrição, mos

trando todos os dados presentes na descrição relativos a cada objeto, em um formato semelhante ao usado para a criação da descrição.

Outro relatório útil poderia ser constituido pela listagem de objetos pré-selecionados com relações específicas colocadas em um formato predefinido. Relatórios assim necessitam que seja feita uma especificação de quais propriedades devem ser usadas para se selecionar um objeto, quais tipos de relação devem ser considerados para a criação do relatório e em que posição ela deve ser colocada no relatório. Essas especificações deveriam ser feitas na fase de definição do tipo de relatório, e tornaria possível definir mais de um relatório com o mesmo formato, apenas alterando-se essas definições.

Relatórios desse tipo podem ser postos à disposição do usuário em uma grande variedade, servindo para fornecer informações gerais a respeito da descrição.

3.6.3 - Geração de Relatórios Específicos

Quando uma descrição evolui no sentido de conter informações mais detalhadas do sistema, o volume de informação tende a crescer, e um relatório que se limite a fornecer listagens genéricas dessas informações, mesmo que restritas a determinado aspecto da descrição, conterá uma quantidade muito grande de dados, obrigando o usuário a penosas análises. Grande parte dessas análises poderiam ser automatizadas e ao invés de obter um relatório genérico, o usuário deveria receber o resultado da análise, com muito menos dados e muito mais significado.

O problema associado à geração de tais relatórios é

que a análise necessária é sempre especifica àquele tipo de sistema. Os relatórios não podem então ser de tipo padrão, mas deve ser possível ao usuário definir quantos relatórios forem necessários a seu tipo de sistema. A definição de cada formato de relatório que poderá ser solicitada, deve ser feita juntamente com a definição de seu tipo de sistema.

Baseado na definição do relatório previamente feita, e na descrição, uma componente deveria ser capaz de gerar o relatório desejado. Os formatos que os relatórios podem assumir são muito variados, porém a grande maioria pode ser vista como sendo constituída de alguns poucos formatos comuns interrelacionados.

Os relatórios podem então ser especificados como combinações desses vários formatos, a que chamaremos módulos de formato, pois cada formato ocupa uma área do relatório correspondente a um módulo.

Os módulos podem ter apenas a função de criar no relatório um formato específico, ou a função de controlar outros módulos, ou ambas as funções. Os módulos se combinam através de informações que transferem entre si. Essas informações se referem a números indicando posições no relatório, objetos da descrição, código de controle e estado de operação daquele módulo. Dessa forma, a execução dos módulos pode ser sequencial ou pode ser concorrente, com módulos passando controle a outros, ou recebendo informações sobre o estado de outros, de modo a alterar sua própria execução, para se ajustar à operação de gerar o relatório como um todo.

Para efeito de discussão podemos tomar os seguintes módulos como exemplos de módulos de formato não gráfico.

- *nome* - para escrever o nome de um objeto, junto com alguma propriedade do objeto
- *coluna* - para criar colunas de informações relativas a um objeto
- *relação* - para obter uma sequência de relacionamentos de um determinado objeto
- *estruturas* - para criar estruturas de objetos
- *matriz* - para obter dados cruzados de objetos através de relações comuns
- *totalizador* - para obter sequências de objetos.

Exemplos de módulos de formato gráficos.

- *nome* - para escrever o nome de um objeto, juntamente com alguma propriedade do objeto na forma de uma figura.
- *X Y* - para escrever alguma informação em uma posição especificada.
- *estrela* - para obter as posições de uma série de objetos em torno do outro.
- *árvore* - para obter as posições de uma série de objetos de forma hierárquica.

Exemplos de módulos de controle.

- *cálculo* - para obter o resultado de uma expressão
- *mínimo* - para obter o mínimo entre vários números
- *condição* - para verificar se alguma condição é válida
- *repetição* - para obter a execução de um (ou vários) módulo um número predeterminado de vezes.

No quadro 3.2 está a relação dos blocos de parâmetros que transferem informações para os módulos *nome*, *coluna* e

relação. Os blocos que importam ou exportam dados são usados para transferir informações diretamente entre módulos. Os blocos que se referem a definições, dependem de especificações feitas pelo usuário quando da definição do relatório, ou são resultados de operações de módulos em partes anteriores do relatório (cabeçalho, corpo ou resumo).

parâmetros:

NOME

i - posição inicial na linha e objeto

e - posição final na linha

d - o que deve ser impresso junto

COLUNA

i - posição inicial na linha e estado

i - objeto

e - estado de cada coluna

e - objeto de cada coluna

e - posição inicial mínima de cada coluna

d - relações a serem seguidas em cada coluna

RELAÇÃO

i - posição inicial na linha e estado

e - objeto

e - estado da relação

d - relações a serem seguidas

i - importa; e - exporta; d - bloco de definição

QUADRO 3.2

3.6.4 - A Geração de Relatórios de Navegação

A geração de um relatório qualquer a respeito de uma descrição é baseada nas propriedades dos objetos, e nos relacionamentos entre objetos. Quando um relatório específico é solicitado, ele é gerado a partir de definição de como os relacionamentos envolvendo objetos predeterminados estão descritos. Através desses relacionamentos, outros objetos são envolvidos, que por sua vez permitem a verificação de outros relacionamentos, e assim por diante, uma grande quantidade de níveis relação-objeto pode ser seguida (navegação da descrição).

Se os tipos de objetos e de relações são sempre os mesmos, ou seguem um padrão imutável, a componente para a geração de relatórios específico já discutida é suficiente para gerar relatórios dessa forma. Porém quando vários tipos de objetos e/ou relações estão envolvidos, em uma forma não previsível, essa componente não será adequada para a geração de tais relatórios, pois a forma de se fazer a pesquisa dos dados na descrição será diferente.

Para essa pesquisa seria conveniente a existência de uma componente dedicada à geração de relatórios que envolvam vários níveis de objetos e relações de tipos variados, chamados relatórios de navegação.

A utilização de um relatório contendo informações, obtidas através de navegação da descrição é bastante ampla, por exemplo, a verificação de uma sequência específica que pode ocorrer na descrição, a simulação de alguma ocorrência específica, a localização de erros ou condições críticas no sistema descrito, etc. Os formatos como tais informações devem ser apresen

tadas ao usuário podem ser os mesmos produzidos pela componente de geração de módulos específicos, e por isso, a atividade principal que esta componente deve executar, é a busca das informações na descrição. Os formatos produzidos por ela devem se restringir àqueles que são específicos a tal tipo de relatório.

Para evitar a duplicação de funções entre esta componente e a que gera relatórios específicos, esta componente poderia armazenar as informações obtidas da base em uma estrutura consistindo basicamente de uma sequência de blocos, contendo os mesmos dados que os existentes nos blocos de exportação de dados, usados pelos módulos de formato da componente de geração de relatórios específicos. Dessa forma a atividade principal da componente de geração de relatórios de navegação seria buscar na descrição do sistema as informações especificadas na descrição do relatório, e depois de selecioná-las e ordená-las na forma descrita, colocá-las disponível para a componente de geração de relatórios específicos. Esses dados seriam analisados por aquela componente como se tivessem sido preparados em uma parte anterior do próprio relatório e usando os formatos próprios daquela componente.

Seria conveniente porém, que alguns formatos próprios de um relatório de navegação, por exemplo relatórios tipo fluxograma, estivessem disponíveis nesta componente, e para isso deveriam existir nela módulos de formato próprios, semelhantes aos existentes na componente de geração de relatórios específicos.

Dessa forma, se a componente pudesse gerar relatórios em formatos próprios, e preparar dados para serem formatados pela componente de geração de relatórios específicos, o SACDS te

ria flexibilidade para gerar virtualmente qualquer tipo de relatório de navegação, dependendo apenas da definição desses relatórios quando da definição do tipo de sistema.

3.7 - Conclusões

A descrição feita neste capítulo se restringiu às funções essenciais que um sistema como o proposto deve apresentar, e mesmo assim apenas nos aspectos mais significativos de cada função. Sobre cada uma das componentes descritas, várias outras características existem, várias outras funções seriam desejáveis existir. Porém, para o objetivo que esta descrição foi feita, um nível maior de detalhe tornaria a descrição mais confusa e provavelmente menos significativa, pois dificilmente uma implementação poderia ser feita seguindo até o final tais especificações. Da forma como foi feita, atendo-se apenas às características fundamentais do sistema, qualquer implementação que se realize, para ser completa, deverá incluir todas essas características.

Isso não significa que todas as funções necessárias à análise e documentação de sistemas tenham sido cobertas. Componentes para auxiliar à monitorar o sistema descrito uma vez implementado (capacidade de aceitar dados em formatos mais gerais), componentes para efetuar conversões na linguagem de descrição, para aceitar que usuários com conhecimentos diferentes possam interagir com o sistema, características de proteção de partes da descrição contra alteração ou verificação por diversos usuários, são apenas mais alguns aspectos que podem

se tornar necessários. No entanto, essas funções não são necessárias em todos os tipos de sistemas, e por isso não foram tratadas.

CAPITULO 4

A IMPLEMENTAÇÃO REALIZADA

4.1 - Considerações Gerais

Neste capítulo é feita a descrição de uma implementação que está sendo feita, de um sistema com as características preconizadas no capítulo 3. Com a concepção adotada para a implementação, todas as características mencionadas no capítulo 3 foram adotadas e são realizáveis, de modo que o sistema, tal como especificado pode ser completamente implementado.

As componentes já implementadas constituem módulos correspondentes nos dois subsistemas especificados no capítulo 3, sendo que para cada função existente no Subsistema de Manipulação da Descrição, existe o correspondente módulo de definição de linguagens e parâmetros necessários no Subsistema de Definição de Linguagens. A implementação descrita possui já prontas as componentes de Elaboração da Descrição do Sistemas (EDS), Seleção de Objetos (SO) e uma pequena capacidade de geração de relatórios genéricos. A descrição de como a implementação foi efetuada pode ser encontrada em [Tr 82a], e particularmente, a implementação do módulo SO do subsistema SMD é descrita em [Em 81]

Na figura 4.1 é apresentado um esquema geral da implementação realizada. O subsistema de utilitários incluído nessa implementação tem por objetivo realizar algumas funções com

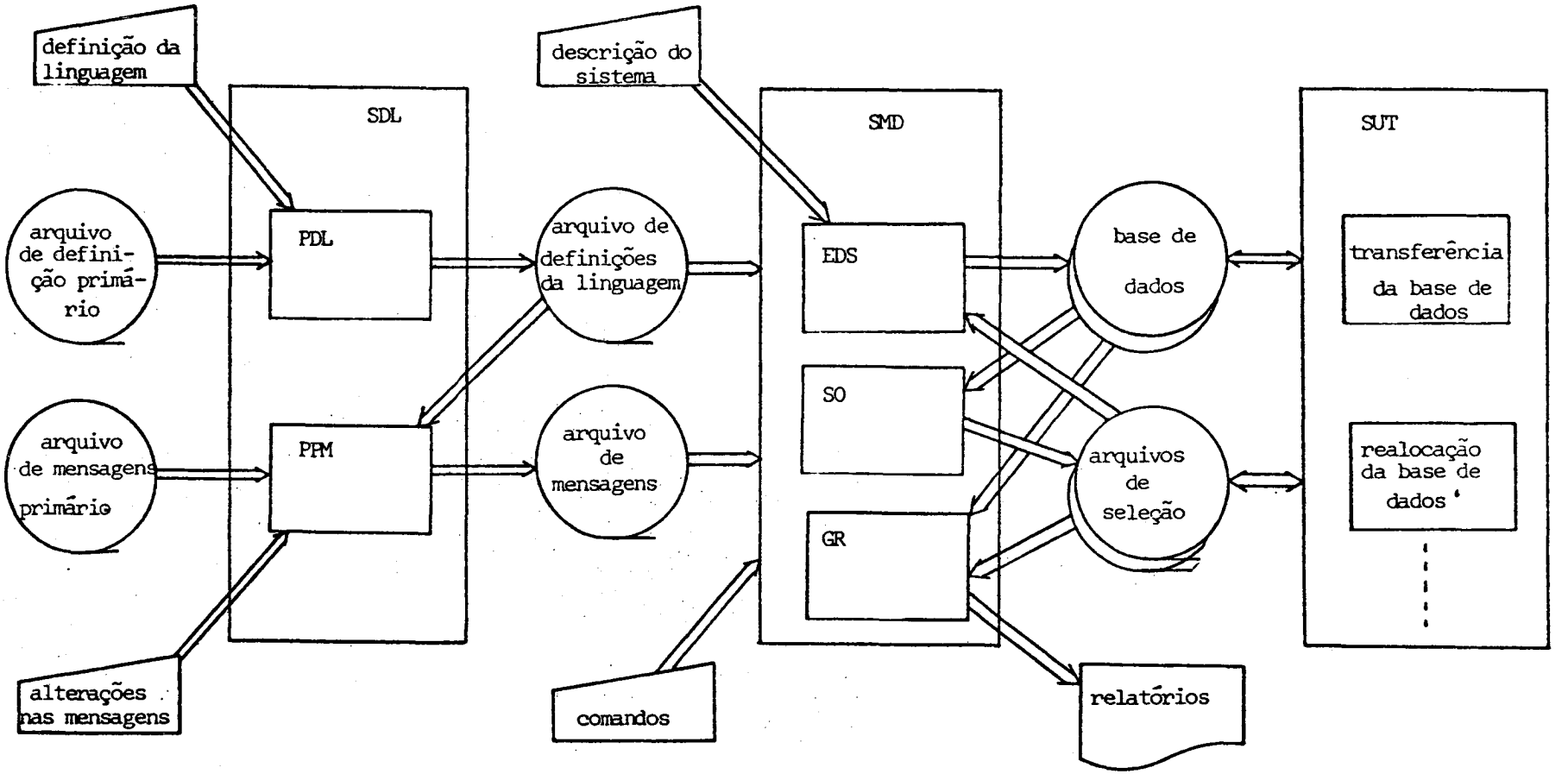

definição da linguagem
SDL
descrição do sistema
SMD
SUT
arquivo de definição primário
PDL
arquivo de definições da linguagem
EDS
base de dados
transferência da base de dados
SO
arquivo de mensagens primário
PPM
arquivo de mensagens
arquivos de seleção
realocação da base de dados
GR
alterações nas mensagens
comandos
relatórios

Figura 4.1

a Base de Dados como um todo, como por exemplo: copiar ou redimensionar uma Base de Dados. No Anexo desse trabalho, estão indicados os significados das abreviaturas usadas.

Para ilustrar a forma como a implementação aqui descrita é utilizada, será usado um exemplo que abrange todos os aspectos mais importantes da implementação.

Uma característica que foi levada em conta para a escolha desse exemplo é que èle é simples o suficiénte, para que não haja dificuldade em se compreender sua implementação, como decorrência do difícil entendimento do exemplo utilizado, ou de ser o sistema descrito estranho.

O tipo do sistema escolhido foi uma orquestra, pois a descrição de seus músicos, repertório, etc., é facilmente compreendido, ao mesmo tempo em que apresenta situações que permitem ilustrar completamente a implementação realizada. Tal exemplo foi sugerido em [Kr 80] onde é proposta uma linguagem para o auxílio à execução de um sintetizador digital, usando para isso uma analogia com uma orquestra. O Objetivo dessa linguagem é permitir ao músico o controle em tempo real do sintetizador para produzir música, de uma forma semelhante àquela que é usada por um maestro regendo uma orquestra. Nesse contexto, as músicas que serão tocadas, bem como uma preespecificação de qual instrumento toca qual parte de cada partitura, já está definida para o sintetizador (orquestra), e ao músico (maestro) são dadas condições de controlar algumas características da execução, como andamento, dinâmica, alterar a execução de alguns instrumentos, etc.

A descrição de uma orquestra usando o SACDS, pode ser dividida em 2 aspectos principais: descrição estática, on-

de são descritos os instrumentos que ela possue, os músicos que a compõem, as partituras que cada músico é capaz de tocar, etc., descrição dinâmica, onde são descritos como determinados fatos ocorrem ou podem ocorrer em decorrência de outros.

Através da descrição estática, podem ser analisadas situações como por exemplo, que músicas deixam de ser executáveis se faltar algum músico, ou como reorganizar quem executa qual parte da partitura se um novo instrumento é adquirido. Através da descrição dinâmica, podem ser analisadas situações que implicam em alterações na execução em decorrência de ações tomadas pelo maestro (alterações na música executada pelo sintetizador em decorrência de alterações em seus controles), como por exemplo que instrumentos devem ter seu volume diminuido para que um solo de clarineta possa ser feita com uma flauta, sem que se perca a nitidez do movimento.

Neste capítulo, será usada apenas a parte estática da descrição deste exemplo, por ser mais fácil de ser compreendida, e suficiente para ilustrar tudo o que é necessário. A descrição completa, feita com os objetivos do referido artigo, é incluída no capítulo seguinte.

4.2 - O Subsistema de Definição de Linguagem

Quando o SACDS deve ser usado para um determinado tipo de sistema, as características de sistemas desse tipo devem estar disponíveis para que uma descrição possa ser feita e analisada. O preparo dessas características em um formato adequado é uma operação distinta das demais, pois enquanto a confecção de

uma descrição, sua análise, e geração de relatórios nela baseados, etc., são operações repetitivas e interativas, o preparo das características de sistemas daquele tipo é feita uma única vez, e usada depois por todas as demais operações. Além disso, enquanto a descrição de um sistema, bem como as operações que com ela se realizam são essencialmente dinâmicas, a definição das características do tipo de sistemas são imutáveis.

Por causa dessas diferenças entre o preparo das características de um tipo de sistema e as demais operações, o SACDS dispõem de um subsistema específico para o preparo dessas características, denominado Subsistema de Definição de Linguagem - SDL. Sua função é aceitar do usuário a definição de características particulares dos sistemas do tipo que ele irá usar, e criar um arquivo com todas essas características estruturadas em formato adequado à consulta para o restante do SACDS. Esse arquivo é denominado Arquivo de Definição de Linguagem - ADL.

Os dados que definem as características de um particular tipo de sistema podem ser distribuídos em 2 grupos: os que definem as características do sistema sem levar em conta qualquer característica do SACDS, e os que acoplam características do sistema à operações específicas do SACDS. O Subsistema de Definição de Linguagem foi feito levando-se em conta esses 2 grupos de dados, bem como a forma com que eles podem ser fornecidos pelo usuário e usados pelo SACDS.

4.2.1 - A Linguagem

A linguagem que o usuário irá utilizar para descrever seu sistema constitui o principal dado que deve ser fornecido ao SDL, e é o dado que não leva em conta qualquer caracte

rística do SACDS. Deve ser uma linguagem relacional, onde se define o conjunto de tipos de objetos e o conjunto de tipos de relações que constituem os sistemas do tipo descrito pelo usuário.

O subsistema SDL foi construido de forma a que a Linguagem de Descrição de Sistemas (LDS) seja obrigatoriamente o primeiro dado a ser fornecido pelo usuário para a montagem do arquivo ADL. Dessa forma, a criação do arquivo é feita através da definição da linguagem, e esta, uma vez definida é imutável.

A definição da linguagem é feita em 5 fases:

- definição do conjunto de tipos de objetos
- definição do conjunto de tipos de relações
- definição das palavras ruído
- especificação de características das relações
- definição da sintaxe da linguagem

As 3 primeiras fases são usadas para fornecer ao SDL a definição das palavras-chave que constituirão a linguagem que está sendo definida. Para que a linguagem seja o mais flexível possível, o usuário deve poder fazer a sua descrição usando o mínimo possível de comandos e palavras-chave do SDL. As três primeiras fases necessitam que o usuário use apenas as palavras reservadas OBJETOS, RELAÇÕES, RUÍDOS e EOF, para classificar as palavras-chave de sua linguagem como tipos de objetos, tipos de relações e palavras sem significado para a linguagem que poderão ser usadas no texto para melhorar a inteligibilidade da descrição. A partir da 4ª fase, essas palavras são usadas pelo SMD juntamente com seus próprios comandos, para descrever o restante da linguagem.

A 4ª fase classifica as relações e lhes fornece algumas características. As relações são classificadas como simples,

triplas, sinônimos e comentários. Para cada relação simples (que em geral são a maioria) é especificada qual é a sua relação oposta. Para cada relação tripla, deve ser especificada qual é a relação simples que corresponde a ela.

Para essas especificações são usadas as palavras reservadas do SDL: OPOSTA, TRIPLA, COMENTÁRIO e SINÔNIMO.

A 5ª fase estabelece os relacionamentos possíveis para cada tipo de objeto. Para cada um são indicados os tipos de relações para as quais objetos desse tipo podem ser objetos origem, e para cada tipo de relação possível, quais os tipos dos objetos que podem ser objetos destino da relação. Dessa forma a sintaxe da linguagem é definida, e todos os relacionamentos possíveis entre cada tipo de objeto estarão estabelecidos.

Nessa fase, a única palavra reservada é OBJETO, usada para indicar que se iniciará a descrição das relações possíveis para objetos do tipo indicado.

Quando a definição da linguagem houver atingido a 5ª fase, o SDL já terá as informações sobre as relações opostas a cada relação. Dessa forma sempre que é especificada uma relação possível do tipo (objeto-origem relação objeto-destino), a relação oposta é assumida automaticamente.

Durante a especificação da Linguagem de Descrição, o SDL assume a existência do objeto tipo INDEFINIDO. Esse tipo de objeto é incluído na linguagem definida, e será usado pelos demais módulos do SACDS sempre que um objeto não tiver tipo definido.

Uma vez fornecidas ao SDL as informações correspondentes às cinco fases descritas, a Linguagem de Descrição de Sistemas estará completa. O SDL prepara então todas essas informa-

ções para uso pelo restante do SACDS, gravando-as no arquivo ADL que define essa linguagem. Tais informações são imutáveis, e se alguma alteração for necessária, um outro arquivo ADL terá que ser criado.

Com essas informações, os comandos do SMD que dependem apenas da Linguagem de Descrição já podem ser usados, porém a maioria dos comandos dependem de mais informações, que acoplam características específicas dos sistemas a serem descritos a operações do SACDS. O SDL deve então ser usado novamente para fornecer essas informações.

4.2.2 - Características e Comandos

O Subsistema para a Manipulação da Descrição (SMD) do SACDS é constituído de vários módulos, cada um com uma tarefa específica. Muitas dessas tarefas necessitam de informações que dependem do tipo de sistema que está sendo descrito. Tais informações são especificadas ao SDL, que as acrescenta às demais informações no arquivo ADL.

Tais informações não são necessárias para a operação do SACDS com sistemas de cada tipo, e portanto não é obrigatória a sua definição. O usuário pode preparar arquivos ADL, apenas com as informações que são necessárias aos módulos do SMD em que ele está interessado ativar. No quadro 4.1 está apresentada uma lista das características cuja especificação é necessária à execução de cada módulo do SMD. Nas seções seguintes estão explicados os significados de cada característica.

A Linguagem de Descrição de Sistemas, já explicada, é necessária a todos os módulos do SMD, e a sua especificação causa a criação de um novo arquivo ADL. As demais especificações

podem ser feitas em qualquer ordem.

4.2.2.1 - Linguagens de Comando Específicas

Alguns módulos do SMD podem desempenhar uma grande

	EDS	Incluir-dados	LDS	Seleção
Linguagem de Descrição	X	X	X	X
Linguagem Específica	X^{LEDS}			X^{LSELC}
Abreviações	X			
Aspecto	X			
Seleção	0		0	X

X - necessita

0 - usa de disponível

Quadro 4.1 - Especificação necessárias aos Módulos do SMD

variedade de serviços ao usuário, e para que este possa controlar o que deseja, cada módulo pode dispor de uma linguagem adequada. Os módulos correspondentes às operações de Elaboração da Documentação de Sistemas (EDS) e Seleção de Objetos da Descrição (SO), são os módulos já implementados que necessitam de uma linguagem de comando específica.

A linguagem de comando é uma forma do usuário especificar a ativação de um determinado serviço que o módulo está preparado para executar. Dessa forma, especificar uma linguagem de comando significa apenas definir de que forma um determinado serviço será executado.

Para manter uma padronização em todo o SACDS, a es-

trutura de todas as linguagens é a mesma, sendo uma estrutura hierárquica, com tantos níveis quanto necessário, ou sejam uma linguagem é constituída de vários comandos, cada comando pode ter vários sub-comandos e assim sucessivamente. A correspondência entre um comando (ou subcomando) e os serviços é feita através de uma sequência de números que é fixada nos módulos. Dessa forma, para a especificação de uma linguagem de comando, o usuário especifica para cada palavra chave de sua linguagem o número de código correspondente ao serviço que ele pretende com aquela palavra.

Em geral, a especificação de uma linguagem não deve diferir daquela linguagem padrão que é descrita no Manual de Uso do SMD [Tr 82b], principalmente porque não é possível modificar a forma com que um serviço é feito. Porém, a possibilidade do usuário definir as palavras de sua linguagem, lhe traz a vantagem de poder solicitar algum serviço usando termos mais adequados ao seu trabalho, ou evitar que algum comando possa ser confundido com algum termo de sua Linguagem de Descrição de Sistemas. Além disso, como mais de um comando podem ter o mesmo número de código de serviços, é possível especificar sinônimos, ou abreviações de um comando. É também possível bloquear o uso de algum serviço, deixando de especificar comandos que o ativem. Esse procedimento pode ser útil por exemplo quando se quer evitar que uma descrição seja alterada por determinadas pessoas. Nesse caso, podem ser criados 2 arquivos ADL, um com uma Linguagem de Elaboração da Documentação de Sistemas (LEDS) completa, e o outro com a mesma linguagem sem os comandos que causam alterações na descrição, permitindo-se o uso do 2º arquivo para as pessoas que devem apenas consultar a descrição, e o uso do 1º arqui-

vo para as pessoas que podem modificar a descrição.

Cada item de um comando pode ser um entre os seguintes tipos:

1 - palavra chave da linguagem de comando

2 - nome de um tipo de objeto da linguagem de descrição

3 - nome de um tipo de relação da linguagem de descrição

4 - nome de um objeto da descrição

5 - caracter de separação e/ou repetição

6 - texto livre

7 - número

O tratamento dado a comandos de um mesmo tipo é o mesmo, até a sua utilização pelo módulo para a execução do serviço. Dessa forma, todas as linguagens de comando apresentam-se padronizadas, ao mesmo tempo em que permitem toda a flexibilidade para a especificação de comandos em qualquer módulo.

4.2.2.2 - Abreviações

As palavras chaves que especificam tipos de objetos e tipos de relação na Linguagem de Descrição de Sistemas, são frequentemente muito extensas, e poderiam ser abreviadas. Em algumas situações, principalmente na geração de relatórios ou na listagem de alguma informação da descrição, é conveniente que o nome do tipo de objeto ou de relação tenha um número máximo de caracteres bastante restrito. Para isso, os nomes dos tipos devem ser abreviados, e o arquivo ADL pode conter uma abreviação de até 6 caracteres para cada tipo de objeto ou de relação. Essas a

breviações, mesmo quando especificadas, não são usadas para reconhecer um comando do usuário, mas tão somente para os relatórios gerados pelos SACDS.

4.2.2.3 - Aspectos

Um sistema pode ser visto como constituido de vários aspectos interdependentes, e todas as componentes do SACDS interagem com a maior parte desses aspectos. No SACDS, os tipos de relação são classificados de acordo com determinados aspectos escolhidos pelo usuário para o tipo de sistemas por ele definido.

O SDL aceita a definição de até 10 aspectos, e todas as relações devem então ser classificadas como pertencente a um dos aspectos definidos.

O objetivo dessa classificação, é fornecer um meio simples de se escolher as relações de interesse em uma descrição para a geração de um relatório.

4.2.3 - Um Exemplo

Será apresentada aqui a definição de uma linguagem para a descrição de uma orquestra. Na figura 4.2 está apresentada a listagem do arquivo usado para a definição da Linguagem de Descrição de Sistemas (LDS).

Cada uma das 5 fases da definição é encerrada pela palavra EOF. As 3 fases iniciais (definição do conjunto de tipos de objetos, de tipos de relação e de palavras ruído) podem aparecer em qualquer ordem, e portanto é necessário que cada uma seja precedida das palavras-chave que as identificam (OBJETOS,

```
 1 BASE_1____=>_OBJETOS
 2             -> MUSICO
 3             -> INSTRUMENTO
 4             -> T-INSTRUMENTO
 5             -> REPERTORIO
 6             -> PARTITURA
 7             -> MOVIMENTO
 8             -> MAESTRO
 9             -> ORQUESTRA
10             -> EOE
11 BASE_2____=>_RELACOES
12             -> REGE
13             -> REGIDA POR
14             -> ESCOLHE, ESCOLHIDO
15             -> CONDUZ, CONDUZIDA
16             -> ELEMENTOS, COMPONENTE DE
17             -> INSTRUMENTAL, PERTENCE
18             -> APRESENTA, APRESENTADA POR
19             -> CONSTITUIDO DE, FAZ-PARTE DE
20             -> EXECUTA, EXECUTADA POR
21             -> PARTICIPA DE, ENVOLVE
22             -> NA-MUSICA
23             -> EXECUTANDO
24             -> TOCA, TOCANDO , TOCADO POR
25             -> USA, USADO POR, USANDO
26             -> PARTE DE, PARTES-ENVOLVIDAS, NA-PARTITURA
27             -> PODE SER, E-DO-TIPO
28             -> CONHECIDA COMO
29             -> BIOGRAFIA, HISTORICO
30             -> EOE
31 BASE_3____=>_BUIDOS
32             -> DE, E', SAO, NA, POR, SER
33             -> COMO
34             -> EOE
35 BASE_4____=>__IDENTIFICACAO_DE_CARACTERISTICAS_DAS_RELACOES
36             -> REGE OPOSTA E' REGIDA
37             -> ESCOLHE OPOSTA E' ESCOLHIDO
38             -> CONDUZ OPOSTA E' CONDUZIDA
39             -> ELEMENTOS OPOSTA E' COMPONENTE DE
40             -> INSTRUMENTAL OPOSTA E' PERTENCE
41             -> APRESENTA OPOSTA E' APRESENTADA
42             -> CONSTITUIDO OPOSTA E' FAZ-PARTE DE
43             -> EXECUTA OPOSTA E' EXECUTADA
44             -> PARTICIPA OPOSTA E' ENVOLVE
45             -> NA-MUSICA TRIPLA DE PARTICIPA ;3
46             -> EXECUTANDO TRIPLA DE EXECUTA
47             -> TOCA OPOSTA E' TOCADO
48             -> TOCANDO TRIPLA E' TOCA
49             -> USA OPOSTA E' USADO
50             -> USANDO TRIPLA DE USA
51             -> PARTE OPOSTA E' PARTES-ENVOLVIDAS
52             -> NA-PARTITURA TRIPLA DE PARTE ;3
53             -> PODE SER OPOSTA E' E-DO-TIPO
```

FIGURA 4.2 (CONTINUA)

```
54              -> CONHECIDA E' SINONIMO
55              -> BIOGRAFIA E' COMENTARIO
56              -> HISTORICO E' COMENTARIO
57              -> EOE
58 FASE 5       -> OBJETO MAESTRO
59              -> REGE PARTITURA
60              -> ESCOLHE REPERTORIO
61              -> CONDUZ ORQUESTRA
62              -> OBJETO ORQUESTRA
63              -> ELEMENTOS MUSICO
64              -> INSTRUMENTAL INSTRUMENTO
65              -> APRESENTA PARTITURA
66              -> OBJETO REPERTORIO
67              -> CONSTITUIDO DE PARTITURA
68              -> OBJETO PARTITURA
69              -> PARTES-ENVOLVIDAS MOVIMENTO
70              -> CONHECIDA COMO PARTITURA
71              -> ENVOLVE MUSICO
72              -> OBJETO MOVIMENTO
73              -> USA INSTRUMENTO, T-INSTRUMENTO
74              -> ENVOLVE MUSICO
75              -> OBJETO MUSICO
76              -> TOCA T-INSTRUMENTO, INSTRUMENTO
77              -> EXECUTA MOVIMENTO
78              -> OBJETO INSTRUMENTO
79              -> E-DO-TIPO T-INSTRUMENTO
80              -> OBJETO T-INSTRUMENTO
81              -> PODE SER INSTRUMENTO
82              -> EOE
83              ->
```

FIGURA 4.2 (FINAL)

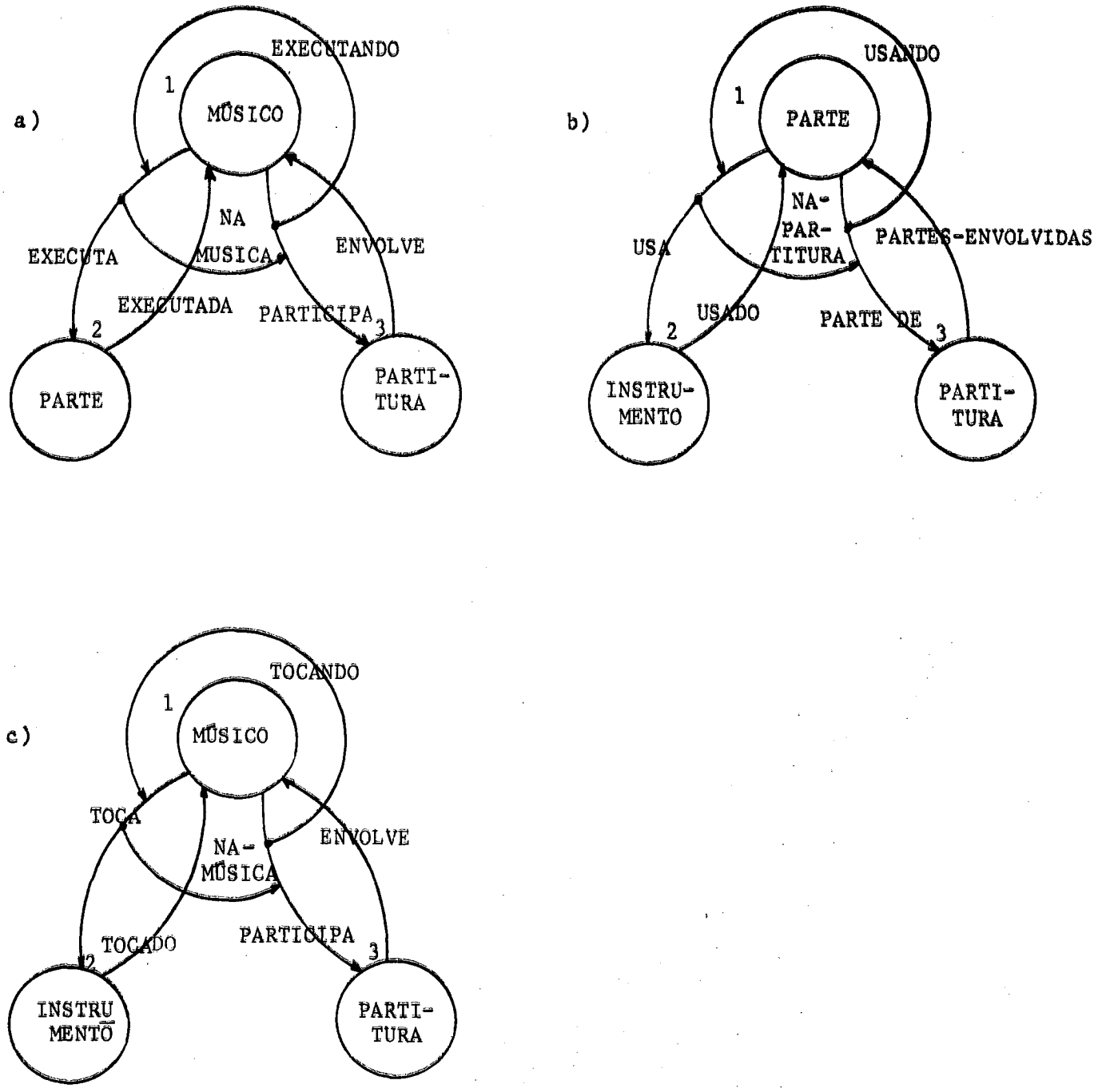

a)
EXECUTANDO
1
MÚSICO
NA
MUSICA
EXECUTA
EXECUTADA
ENVOLVE
PARTICIPA
2
3
PARTE
PARTI-
TURA
b)
USANDO
1
PARTE
NA-
PAR-
TITURA
USA
USADO
PARTES-ENVOLVIDAS
PARTE DE
2
3
INSTRU-
MENTO
PARTI-
TURA
c)
TOCANDO
1
MÚSICO
TOCA
NA-
MÚSICA
ENVOLVE
TOCADO
PARTICIPA
2
3
INSTRU
MENTO
PARTI-
TURA

Figura 4.3

RELAÇÕES e RUÍDOS). As 2 fases restantes devem vir sempre nessa ordem, (especificação das características das relações e definição da sintaxe da Linguagem), e portanto não necessitam de identificação.

Os tipos de objetos que comstituirão essa linguagem são definidos nas linhas 1 a 10 da figura 4.2, e tem o seguinte significado:

Maestro - uma pessoa que conduz a orquestra, e tendo se em vista o uso da descrição para a análise de um sintetizador, quem recebe os estímulos produzidos e que atua nos comandos do instrumento.

Orquestra - o objeto central da descrição (pode ser o próprio sintetizador).

Músico - um componente da orquestra, e que executa um instrumento.

Instrumento - um instrumento musical, capaz de produzir sons.

Tipo-de-Instrumento - um conjunto de características que definem como um som deve ser criado.

Repertório - uma coleção de partituras

Partitura - a especificação para cada instrumento, de como uma música deve ser.

Movimento - a especificação de um trecho de uma música para um instrumento (ou grupo de instrumentos iguais)

Na 4ª fase as relações são especificadas como sendo simples, triplas, sinônimos ou comentários. Se uma relação é simples, basta definir a sua oposta, como por exemplo ocorre na li

```
OBJETO TIPO INDEFINIDO

OBJETO TIPO INSTRUMENTO
          COM A RELACAO E-DO-TIPO                          OBJETO T-INSTRUMENTO
          COM A RELACAO PERTENCE                           OBJETO ORQUESTRA
          COM A RELACAO TOCADO POR                         OBJETO MUSICO
          COM A RELACAO USADO POR                          OBJETO MOVIMENTO

OBJETO TIPO MAESTRO
          COM A RELACAO CONDUZ                             OBJETO ORQUESTRA
          COM A RELACAO ESCOLHE                            OBJETO REPERTORIO
          COM A RELACAO REGE                               OBJETO PARTITURA

OBJETO TIPO MOVIMENTO
          COM A RELACAO ENVOLVE                            OBJETO MUSICO
          COM A RELACAO EXECUTADA POR                      OBJETO MUSICO
          COM A RELACAO PARTE DE                           OBJETO PARTITURA
         -COM A RELACAO USA                                :
                        OBJETO INSTRUMENTO
                        OBJETO T-INSTRUMENTO

OBJETO TIPO MUSICO
          COM A RELACAO COMPONENTE DE                      OBJETO ORQUESTRA
          COM A RELACAO EXECUTA                            OBJETO MOVIMENTO
         -COM A RELACAO PARTICIPA DE                       :
                        OBJETO MOVIMENTO
                        OBJETO PARTITURA
         -COM A RELACAO TOCA                               :
                        OBJETO INSTRUMENTO
                        OBJETO T-INSTRUMENTO

OBJETO TIPO ORQUESTRA
          COM A RELACAO APRESENTA                          OBJETO PARTITURA
          COM A RELACAO CONDUZIDA                          OBJETO MAESTRO
          COM A RELACAO ELEMENTOS                          OBJETO MUSICO
          COM A RELACAO INSTRUMENTAL                       OBJETO INSTRUMENTO

OBJETO TIPO PARTITURA
          COM A RELACAO APRESENTADA POR                    OBJETO ORQUESTRA
          COM A RELACAO CONHECIDA COMO                     OBJETO PARTITURA
          COM A RELACAO ENVOLVE                            OBJETO MUSICO
          COM A RELACAO FAZ-PARTE DE                       OBJETO REPERTORIO
          COM A RELACAO PARTES-ENVOLVIDAS                  OBJETO MOVIMENTO
          COM A RELACAO REGIDA POR                         OBJETO MAESTRO

OBJETO TIPO REPERTORIO
          COM A RELACAO CONSTITUIDO DE                     OBJETO PARTITURA
          COM A RELACAO ESCOLHIDO                          OBJETO MAESTRO

OBJETO TIPO T-INSTRUMENTO
          COM A RELACAO PODE SER                           OBJETO INSTRUMENTO
          COM A RELACAO TOCADO POR                         OBJETO MUSICO
          COM A RELACAO USADO POR                          OBJETO MOVIMENTO
```

FIGURA 4.4

nha 36 da figura 4.2.

Uma relação tripla é definida através das 2 relações simples que a constituem, e das correspondências feitas entre elas. Por exemplo, o relacionamento triplo mostrado na figura 4.3 a) é feito através dos comandos das linhas 43, 44, 45 e 46 da figura 4.2.

Para definir uma relação como tipo sinônimo ou comentário, basta fazer essa indicação, como por exemplo nas linhas 54 e 55 da figura 4.2.

Na 5ª fase é feita a definição da sintaxe da linguagem. São definidos para cada objeto os relacionamentos simples em que ele está envolvido, e as relações sinônimo que cada objeto pode ter. Tal definição é feita nas linhas 58 a 81 da figura 4.2.

Na figura 4.4 está apresentada uma listagem da sintaxe assumida pelo SDL com os comandos da figura 4.2. Nessa listagem não aparecem os relacionamentos triplos nem tipo comentários.

Uma vez definida a Linguagem de Descrição de Sistemas, é feita a especificação de características da Linguagem em relação a operações específicas do SACDS. Cada característica é definida em um arquivo próprio, e as suas inclusões no arquivo ADL são solicitadas através de comandos dados diretamente ao SDL. Na figura 4.5 é apresentada a sequência de comandos usada para completar o arquivo ADL usado como exemplo no restante deste capítulo. Na figura 4.6 é apresentada a listagem dos arquivos de abreviações e de aspectos tal como gerado pelo SDL.

O comando PADRÃO especifica ao SDL que devem ser incluídas no arquivo ADL as linguagens de comando específicas dos

```
LISTAR
ABREVIACOES ARQUIVO=ORQSTR.ABR
ASPECTOS ARQUIVO=ORQSTR.ASP
PADRAO
LISTAR
```

Figura 4.5

módulos de Elaboração da Documentação de Sistemas (LEDS),e de Seleção de Objetos (LSELEC) sem nenhuma alteração em relação ao padrão descrito no Manual de uso do SMD [Tr 82b].

```
 1 ->
 2 -> ------
 3 -> LISTAR
 4 -> ------
 5 ->
 6 ->
 7 ->
 8 ->  ## REGISTROS DE CONTROLE ##
 9 ->       1 = 286,   4,  19, 169, 178, 212, 219, 220, 221,   0,   0,   0,
10 ->       2 =  20,  29,  37,  38,  82, 119, 120, 127, 135, 136, 136, 136,
11 ->       3 = 157, 166, 168, 169, 169, 169,   0,   0,   0,   0,   0,   0,
12 ->       4 =   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
13 ->     220 =   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
14 ->     221 =   1,   1,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,
15 ->
16 ->  ## RELATORIOS AUTORIZADOS ##
17 -> INCLUIR-DADOS
18 -> LISTAGEM-DA-DESCRICAO
19 -> VER-BASE
20 ->
21 -> ------------------------------
22 -> ABREVIACOES ARQUIVO=ORQSTR.ABR
23 -> ------------------------------
24 ->
25 -> MUSICO=MUSICO
26 -> INSTRUMENTO =INSTR
27 -> T-INSTRUMENTO= T-INST
28 -> REPERTORIO=REP
29 -> PARTITURA=PARTT
30 -> MOVIMENTO=MOVM
31 -> MAESTRO=MAESTR
32 -> ORQUESTRA=ORQSTR
33 -> INDEFINIDO=INDEF
34 -> REGE =REGE
35 -> REGIDA =REGIDA
36 -> ESCOLHE=ESCLH
37 -> ESCOLHIDO=ESCLHD
38 -> CONDUZ=CONDZ
39 -> CONDUZIDA=CONDZD
40 -> ELEMENTOS=ELEMTS
41 -> COMPONENTE = COMPNT
42 -> INSTRUMENTAL =INSTR
43 -> PERTENCE=PERTNC
44 -> APRESENTA=  APREST
45 -> APRESENTADA=APRESD
46 -> CONSTITUIDO =CONSTD
47 -> FAZ-PARTE =FAZ-PT
48 -> EXECUTA=EXCT
49 -> EXECUTADA =EXCTD
50 -> PARTICIPA =PARTCP
51 -> ENVOLVE=ENVOLV
52 -> NA-MUSICA= NA-MUS
53 -> EXECUTANDO=EXCTND
54 -> TOCA=TOCA
55 -> TOCANDO=TOCNDO
56 -> TOCADO =TOCADO
```

FIGURA 4.6 (CONTINUA)

```
 57 -> USA=USA
 58 -> USADO =USADO
 59 -> USANDO=USANDO
 60 -> PARTE =PARTE
 61 -> PARTES-ENVOLVIDAS=PT-ENV
 62 -> NA-PARTITURA=NA=PRTT
 63 -> PODE =PODE
 64 -> E-DO-TIPO=DO-TIP
 65 -> CONHECIDA =CONH
 66 -> BIOGRAFIA=BIOGR
 67 -> HISTORICO=HIST
 68 ->
116 -> ---------------------------
117 -> ASPECTOS ARQUIVO=ORQSTR.ASP
118 -> ---------------------------
119 ->
120 -> RECURSOS, PESSOAL, CONHECIMENTO, COMANDO
121 -> EOF
122 -> REGE = CONHECIMENTO
123 -> REGIDA = CONHECIMENTO
124 -> ESCOLHE = COMANDO
125 -> ESCOLHIDO = COMANDO
126 -> CONDUZ = COMANDO
127 -> CONDUZIDA = COMANDO
128 -> ELEMENTOS = PESSOAL
129 -> COMPONENTE = PESSOAL
130 -> INSTRUMENTAL = RECURSOS
131 -> PERTENCE = RECURSOS
132 -> APRESENTA = CONHECIMENTO
133 -> APRESENTADA = RECURSOS
134 -> CONSTITUIDO = CONHECIMENTO
135 -> FAZ-PARTE = CONHECIMENTO
136 -> EXECUTA = CONHECIMENTO
137 -> EXECUTADA = CONHECIMENTO
138 -> PARTICIPA = PESSOAL
139 -> ENVOLVE = PESSOAL
140 -> NA-MUSICA = PESSOAL
141 -> EXECUTANDO = CONHECIMENTO
142 -> TOCA = CONHECIMENTO
143 -> TOCADO = CONHECIMENTO
144 -> TOCANDO = CONHECIMENTO
145 -> USA = RECURSOS
146 -> USADO = RECURSOS
147 -> USANDO = RECURSOS
148 -> PARTE = RECURSOS
149 -> PARTES-ENVOLVIDAS = RECURSOS
150 -> NA-PARTITURA = RECURSOS
151 -> PODE = RECURSOS
152 -> E-DO-TIPO = RECURSOS
153 -> CONHECIDA = CONHECIMENTO
154 -> BIOGRAFIA = PESSOAL
155 -> HISTORICO = RECURSOS
156 -> EOF
```

FIGURA 4.6 (CONTINUA)

```
157 -> APRESENTA                          (APREST) = CONHECIMENTO
158 -> APRESENTADA POR                    (APRESD) = RECURSOS
159 -> BIOGRAFIA                          (BIOGR ) = PESSOAL
160 -> COMPONENTE DE                      (COMPNT) = PESSOAL
161 -> CONDUZ                             (CONDZ ) = COMANDO
162 -> CONDUZIDA                          (CONDZD) = COMANDO
163 -> CONHECIDA COMO                     (CONH  ) = CONHECIMENTO
164 -> CONSTITUIDO DE                     (CONSTD) = CONHECIMENTO
165 -> E-DO-TIPO                          (DO-TIP) = RECURSOS
166 -> ELEMENTOS                          (ELEMTS) = PESSOAL
167 -> ENVOLVE                            (ENVOLV) = PESSOAL
168 -> ESCOLHE                            (ESCLH ) = COMANDO
169 -> ESCOLHIDO                          (ESCLHD) = COMANDO
170 -> EXECUTA                            (EXCT  ) = CONHECIMENTO
171 -> EXECUTADA POR                      (EXCTD ) = CONHECIMENTO
172 -> EXECUTANDO                         (EXCTND) = CONHECIMENTO
173 -> FAZ-PARTE DE                       (FAZ-PT) = CONHECIMENTO
174 -> HISTORICO                          (HIST  ) = RECURSOS
175 -> INSTRUMENTAL                       (INSTR ) = RECURSOS
176 -> NA-MUSICA                          (NA-MUS) = PESSOAL
177 -> NA-PARTITURA                       (NA-PRT) = RECURSOS
178 -> PARTE DE                           (PARTE ) = RECURSOS
179 -> PARTES-ENVOLVIDAS                  (PT-ENV) = RECURSOS
180 -> PARTICIPA DE                       (PARTCP) = PESSOAL
181 -> PERTENCE                           (PERTNC) = RECURSOS
182 -> PODE SER                           (PODE  ) = RECURSOS
183 -> REGE                               (REGE  ) = CONHECIMENTO
184 -> REGIDA POR                         (REGIDA) = CONHECIMENTO
185 -> TOCA                               (TOCA  ) = CONHECIMENTO
186 -> TOCADO POR                         (TOCADO) = CONHECIMENTO
187 -> TOCANDO                            (TOCNDO) = CONHECIMENTO
188 -> USA                                (USA   ) = RECURSOS
189 -> USADO POR                          (USADO ) = RECURSOS
190 -> USANDO                             (USANDO) = RECURSOS
292 ->
293 -> ------
294 -> LISTAR
295 -> ------
296 ->
297 ->
298 ->
299 ->  ## REGISTROS DE CONTROLE ##
300 ->      1 = 418,   4,  19, 169, 178, 212, 219, 220, 221,   0,   0,   0,
301 ->      2 =  20,  29,  37,  38,  82, 119, 120, 127, 135, 136, 136, 136,
302 ->      3 = 157, 166, 168, 169, 169, 169,   0,   0,   0,   0,   0,   0,
303 ->      4 =   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
304 ->    220 = 287, 295, 329, 334, 392, 393, 407, 330, 333, 408, 418,   0,
305 ->    221 =   1,   1,   1,   0,   0,   1,   1,   0,   0,   0,   0,   0,
306 ->
307 ->  ## RELATORIOS AUTORIZADOS ##
308 -> INCLUIR-DADOS
309 -> LISTAGEM-DA-DESCRICAO
310 -> VER-BASE
311 -> SELECIONAR-OBJETOS
312 -> ELABORAR-DESCRICAO
```

FIGURA 4.6 (FINAL)

4.3 - A Elaboração de uma Descrição

Associado a um Arquivo de Definição de Linguagem (ADL), o SACDS torna-se uma ferramenta útil para documentação de sistemas do tipo descrito no arquivo ADL. Dessa forma, uma associação SMD - arquivo ADL pode ser usada para auxiliar a documentação e análise de qualquer sistema daquele tipo. Isto é feito fornecendo-se ao SMD uma descrição do sistema, que cria uma base de dados para aquela descrição. Essa base pode então ser usada pelos demais módulos do SMD para análise e geração de relatórios.

A componente do SMD que recebe do usuário a descrição de seu sistema é chamada Módulo para a Elaboração da Descrição do Sistema - EDS. É um módulo preparado para receber informações de forma interativa, operando a partir de um terminal de vídeo. O usuário fornece as informações referentes ao seu sistema utilizando-se da linguagem LDS definida no arquivo ADL, além de dispor da linguagem de comando do EDS, com a qual ele pode especificar as operações que deseja executar, tais como acrescentar, retirar ou substituir dados, bem como controlar os dados da descrição que são mostradas no vídeo.

4.3.1 - A Entrada de dados através do EDS

A descrição de sistema através do SACDS é feita especificando-se os objetos e relações que constituem o sistema. A especificação de objetos e suas relações é feita usando-se a linguagem LDS, e o módulo EDS, que converte tais especificações em alterações na base de dados que armazena a descrição daquele sistema. As alterações que o sistema pode realizar se resumem em in

cluir, retirar e substituir objetos e relações da descrição.

Dessas operações, a inclusão de objetos e relações, e a retirada de relações da descrição é especificada pelo usuário diretamente através da linguagem LDS, tal como especificada no arquivo ADL. Nessas operações, a especificação de um comando nessa linguagem causa a imediata alteração daquele item na descrição.

A operação de retirada de objetos da descrição é feita apenas através de um comando explícito para se retirar aquele objeto. Quando um objeto é retirado, todos os dados da descrição que envolvem aquele são retirados.

A operação de substituir objetos e/ou relações assumem diversos aspectos, pois existem várias formas de substituições possíveis, variando desde a alteração do nome de um objeto, até a troca de objetos destino (ou origem) de relações. De qualquer forma, a operação de substituição é sempre solicitada através de comandos explícitos para o EDS, e não apenas utilizando a linguagem LDS.

É através de comandos dados ao EDS, especificando as operações que são desejadas, e da definição de dados do sistema, através da linguagem LDS, que o usuário prepara a descrição de seu sistema. As alterações feitas na descrição se refletem imediatamente nas informações que são mostradas em vídeo, para permitir uma rápida conferência do estado real de descrição por parte do usuário.

4.3.2 - A Estrutura do Módulo EDS

As várias operações que o EDS executa são realizadas em módulos que cuidam especificamente de cada operação. A estru

tura pela qual esses módulos se entrelaçam se reflete na estrutura da linguagem de comando do próprio EDS. Por isso, existem seis modos de operação na linguagem.

Além disso, existem 2 formas de se indicar os objetos que participarão de determinada alteração da descrição. A primeira forma, que será chamada de "escolha de objetos", é usada para se alterar um ou poucos objetos escolhidos. Quando existe apenas um objeto escolhido (que constitui o caso mais frequente), todas as informações solicitadas a respeito daquele objeto são mostradas no vídeo simultaneamente com qualquer alteração que seja feita.

A segunda forma de se indicar objetos, que será chamada de "seleção de objetos" é usada para se efetuar alterações semelhantes em um número grande de objetos. Nesse caso, no vídeo são mostradas apenas as alterações que são comuns a todos os objetos selecionados.

Para se proceder à escolha de objetos, o usuário deve indicar cada objeto individualmente. Sempre que é feita a escolha de um ou mais objetos, o conjunto de objetos escolhidos anteriormente deixa de ter qualquer significado. Quanto à seleção de objetos, o programa EDS oferece poucos recursos de seleção, limitando-se a selecionar todos os objetos de determinado tipo que existem na descrição, ou obter uma seleção previamente preparada, com o programa de seleção de objetos - SO, que esse sim, oferece amplos recursos de seleção.

Os seis modos de operação de EDS atuam diretamente sobre os objetos escolhidos ou selecionados, e são os seguintes:

1 - Modo de comando - é o modo onde são analisados todos os comandos da linguagem de comando do EDS emitidos pelo u-

suário, e onde todos os comandos que executam operações que não envolvem modos especiais são realizadas. É também o modo que comanda a execução de todos os demais modos, e o único que permite ao usuário selecionar objetos.

2 - Modo de inclusão - nesse modo, a linguagem utilizada é apenas a LDS. Todas as informações fornecidas são analisadas quanto à sintaxe, e se corretas incluídas na descrição para todos os objetos presentemente escolhidos. Os objetos escolhidos em qualquer outro modo podem continuar sendo usados neste modo, bem como podem ser feitas novas escolhas neste modo, dinamicamente.

3 - Modo acrescentar - este modo é semelhante ao de inclusão, com a diferença que as informações fornecidas são colocadas na descrição para todos os objetos da seleção presente.

4 - Modo retirar - também neste modo a linguagem utilizada é a LDS. Todas as informações fornecidas são buscadas em cada um dos objetos escolhidos, e sempre que encontradas são retiradas. Em termos de escolha de objetos, atua da mesma forma que o modo de inclusão. Este modo pode retirar da descrição qualquer informação pertinente aos objetos escolhidos, mas não retira nunca um objeto.

5 - Modo eliminar - este modo é semelhante ao modo retirar, com a diferença que as informações fornecidas são retiradas de cada um dos objetos selecionados.

6 - Modo substituir - neste modo é possível substituir os nomes de cada um dos objetos da seleção. Não é necessária nenhuma linguagem especial para este modo, bastando a especificação dos novos nomes de objetos que terão seus nomes substituídos.

Todas as operações que envolvam colocar ou retirar dados da descrição são executadas pelos modos: Incluir, Acrescentar, Retirar e Eliminar. As operações de substituição de dados são executadas pelos modos Substituir e de comando.

O modo de comando, executa todas as operações de controle dos demais modos, bem como algumas operações de mudança da descrição que não são executadas pelos outros modos,como por exemplo as operações de retirar objetos da descrição, substituição de relações de determinados objetos, ou de nomes de objetos escolhidos, e a edição de comentários.

4.3.3 - Operações de Controle do EDS

Além dos comandos de incluir, retirar e substituir dados da descrição aceitos pelo EDS, existem comandos que permitem controlar essas operações, bem como controlar o tipo de informação que é mostrada interativamente a medida que a descrição vai sendo elaborada.

Para se controlar o que deve ser mostrado no vídeo, existem comandos que indicam se há relações, comentários e/ou sinônimos de cada objeto devem ser mostrados, e para as relações e comentários, sobre que aspectos devem ser mostrados. Uma vez especificado o que deve ser mostrado, tais especificações permanecem em efeito para todos os modos de operação, até que sejam explicitamente alteradas.

Além disso, existem comandos que solicitam explicitamente informações da descrição, sem importar com as especificações estabelecidas, e sem alterá-las. Entre estes existem comandos que indicam inclusive os comandos da própria linguagem que podem ser solicitados.

OBJETO ou tipo-de-objeto	- E' feita uma nova escolha de objetos, que passam a ser os objetos indicados seguindo ao comando;
SELECIONE	- E' feita uma nova seleção de objetos;
ATIVE	- Indica que uma relação deve ser automaticamente incluída em todos os novos objetos criados;
DESATIVE	- Indica que uma relação não deve mais ser automaticamente incluída;
RETIRE	- Retira objetos da descrição;
ELIMINE	- Retira da descrição todos os objetos selecionados;
SUBSTITUA	- Substitui o nome de um objeto por outro;
REFORMATE	- Reformata o vídeo;
EDITE	- Permite a edição de comentários como texto;
AGLUTINE	- Junta dois objetos da descrição em um apenas;
MODO	- Especifica um novo modo em que se irá operar;
LISTE	- Lista no vídeo os dados requisitados;
MOSTRE	- Indica os dados que devem ser mostrados sobre cada objeto escolhido.

Figura 4.7 - Linguagem de Elaboração de Descrição de Sistemas

```
OBJETO  objeto [,objeto ]...

Tipo-de-objeto objeto [,objeto ]...

SELECIONE critério

ATIVE relação objeto

DESATIVE relação objeto

RETIRE objeto [,objeto ]...

ELIMINE  | TUDO   |
         | ALGUNS |

SUBSTITUA objeto [ = objeto ]

REFORMATE

EDITE [objeto] relação <Linguagem-de-edição>

AGLUTINE objeto [ = objeto ]

MODO  | INCLUIR     |
      | ACRESCENTAR |
      | RETIRAR     |
      | ELIMINAR    |
      | SUBSTITUIR  |

LISTE  | PARAMETROS      |
       | SELEÇÃO         |
       | OBJETO = objeto |

[NÃO] MOSTRE  | COMENTARIOS      |
              | SINONIMOS        |
              | RELACOES         |
              | ASPECTO aspécto  |
```

Figura 4.8 - Estrutura da LEDS

Para se controlar a inclusão de objetos na descrição, existe a possibilidade de solicitar ao EDS a colocação de relações automaticamente sempre que um novo objeto é incluído. Caso tal operação seja solicitada, até um máximo de 3 relações podem ser colocadas nesse objeto, no caso da sintaxe permitir tais relações. Esse recurso é útil para, por exemplo, em uma descrição feita por várias pessoas, indicar quem é a pessoa responsável por cada objeto criado. Ou, em uma descrição envolvendo várias fases, ou aspectos do problema, indicar-se a que fase ou aspecto do problema os objetos se referem.

Para a preparação dos comentários de cada objeto, existe um editor de comentários, que trata cada comentário de cada objeto como um texto independente, e fornece vários recursos para formatação interativa desse texto, como sempre apoiado nas operações básicas de inclusão, retirada e substituição, no caso de caracteres e/ou linhas.

Existem ainda comandos capazes de realizar a escolha e/ou seleção de objetos da descrição que serão afetados pelos comandos seguintes.

Na figura 4.7 está uma listagem dos comandos que normalmente constituem a linguagem de comando do EDS. Uma descrição detalhada de cada comando encontra-se em [Tr 82b]. Na figura 4.8 os mesmos comandos estão dispostos em forma hierárquica, mostrando a forma como são analisados, onde é possível notar a estrutura de linguagem descrita na seção 4.2.2.1 (deste trabalho).

Por ser um módulo de caráter exclusivamente interativo, o EDS não permite o uso de arquivos de comando. Dessa forma, todos os comandos somente podem ser fornecidos através de

terminal, e além disso o terminal deve ser de vídeo, com recursos de acesso a qualquer posição da tela através de comando direto de endereçamento da posição do caracter.

Para possibilitar a inclusão na descrição de uma grande quantidade de informações previamente preparada, os módulos de programa que constituem o modo de inclusão de dados do EDS foram alterados e criou-se uma nova operação para o SMD, cuja função é aceitar um arquivo sequencial contendo descrições de sistemas na linguagem LDS e colocá-las na descrição. Essa operação do SMD é normalmente chamada de INCLUIR-DADOS e tem por objetivo a adição de informações na descrição de forma não interativa.

4.4 - A Seleção de objetos em uma Descrição

A descrição de um sistema armazenada pelo SACDS, conterá uma grande quantidade de informações, e se vista como um todo, conterá informações sobre variados aspectos e estados do sistema, informações essas que estarão todas interrelacionadas, e portanto difíceis de serem visualizadas e aproveitadas.

Torna-se pois necessário, antes que o SMD forneça relatórios sobre qualquer aspecto do sistema, que o usuário selecione as partes da descrição que lhe interessa. Esse recurso é fornecido pela componente para Selecionar Objetos, SO, que permite ao usuário especificar quais informações da descrição lhe interessa selecionar.

A seleção é feita procurando-se na descrição os objetos que correspondem às especificações solicitadas pelo usuário, e indicando-se quais são esses objetos em um arquivo, chamado arquivo de Seleção. Podem ser criados tantos arquivos de sele-

ção quanto se quiserem.

Para o preparo de arquivos de seleção, a componente SO opera com conjuntos de objetos, que inicialmente possuem características comuns facilmente identificáveis, e através de sucessivas operações lógicas com esses conjuntos, vão sendo produzidos novos conjuntos até se chegar àquele que contém exatamente os objetos nos quais o usuário está interessado. Os conjuntos de objetos são estruturas de dados internas da componente SO, e para que um conjunto de seleção possa ser usado em outras componentes do SACDS, é necessário que um conjunto seja copiado em um arquivo de seleção.

O número de conjuntos de seleção existentes simultaneamente pode atingir um máximo de 20. Esses conjuntos podem ser controlados pelo próprio usuário, designando-lhes nomes, ou pelo próprio módulo SO, que cuida de sua criação e eliminação quando não mais necessários.

A linguagem de comando do módulo SO pode ser vista como constituída de comandos que especificam operações a serem feitas com conjuntos e arquivos de seleção, e critérios de seleção (que operam com conjuntos de seleção, e indicam buscas a serem efetuadas na descrição).

Na figura 4.9 está uma listagem da linguagem normalmente aceita pela componente SO. Uma descrição detalhada de cada comando, e de cada critério de Seleção encontra-se em [Tr 82 b], e uma descrição da estrutura interna da componente SO está em [Tr 82a]. Na figura 4.10 está uma listagem dos comandos da LSO com uma explicação de cada comando.

```
<comando> ::= MOSTRE
            | LISTE <conjunto de seleção>
            | GRAVE <conjunto de seleção> NO <arquivo de seleção>
            | OBTENHA <conjunto de seleção> DO <arquivo de seleção>
            | <conjunto de seleção> = <expressao para seleção>

<expressao para seleção> ::= <critério>
                           | <critério> <operador> <critério>
                           | <expressao para seleção> <operador> <critério>
                           | <tipo-de-relação> ( <expressao para seleção> )
                           | <expressao para seleção> '

<critério> ::= <tipo-de-objeto>
             | COMENTARIOS
             | SINONIMOS
             | ISOLADO
             | TODOS

<operador> ::= + | * | -
```

Figura 4.9 - Linguagem de Seleção de Objetos

MOSTRE - Lista os nomes dos conjuntos de seleção;

LISTE - Lista os nomes dos objetos selecionados no conjunto indicado;

GRAVE - Grava o conjunto indicado no arquivo de seleção indicado;

OBTENHA - Copia o arquivo de seleção indicado no conjunto indicado;

expressao - Seleciona na descrição os objetos que condizem com o critério especificado;

critérios de seleção = expressoes logicas com operadores sobre conjuntos. Os conjuntos podem ser obtidos a partir de outros critérios, ou a partir de caracteristicas dos objetos.

Figura 4.10 - Estrutura de uma LSO

4.5 - Geração de Relatórios

No estágio atual de implementação do SACDS, a capacidade de geração de relatórios do SMD se restringe ao relatório denominado Listagem de Descrição do Sistema - LDS. Esse relatório pode ser considerado como o mais geral possível de ser gerado pelo SMD, pois para cada objeto que aparece no relatório são impressas todas as informações referentes a ele existentes na descrição. Os objetos que aparecem no relatório são os indicados em um arquivo de seleção especificado pelo usuário. Caso um arquivo não seja especificado, são colocados no relatório, em ordem alfabética, todos os objetos existentes na descrição.

Com o objetivo de facilitar a geração de relatórios, foram criadas várias rotinas de acesso à descrição, e de formatação de textos que podem ser usadas para se obter relatórios. Usando-as é possível a confecção de programas que forneçam, os relatórios desejados com relativa facilidade, porém essa é uma solução transiente, pois é necessária a criação de uma linguagem específica e um programa analisador, que possa controlar a execução dessas rotinas a partir de especificações do usuário dadas nessa linguagem específica, adequada ao seu tipo de sistema.

4.6 - Conclusões

A implementação aqui descrita foi realizada no Computador PDP 11/45 do Laboratório de Computação do Departamento de Ciências de Computação e Estatística do Instituto de Ciências Matemáticas de São Carlos da Universidade de São Paulo, operando com o Sistema Operacional RSX-11M. Foi totalmente elaborado na

Linguagem Fortran IV, tendo como um dos objetivos da implementação o de ser facilmente transportável para outros computadores no futuro. As duas únicas partes da implementação que dependem da instalação são as que dizem respeito às operações de manipulação de arquivos que dependem do sistema operacional (identificação, abertura e fechamento de arquivos), e as que usam um terminal de vídeo de cursor endereçável.

As operações de manipulação de arquivos dependentes do sistema operacional foram agrupadas em poucas rotinas, facilmente identificáveis, e com funções bem definidas, de forma que possam ser substituídas com facilidade se o SACDS tiver que ser levado para outro equipamento.

As operações que usam um terminal de vídeo com cursor endereçável foram feitas especificamente para o terminal VT52, usadas apenas no módulo EDS. Tais operações também estão agrupadas em algumas poucas rotinas, e podem ser modificadas com facilidade no que diz respeito à forma como o controle do vídeo é efetuado. Porém, algumas características desse terminal estão profundamente arraigadas no módulo EDS, que são o número de linhas e colunas de vídeo, e o fato dele ter o cursor endereçável, caracter por caracter.

Cada subsistema do SACDS foi implementado como um programa independente, usando rotinas comuns para o acesso aos vários arquivos usados pelo sistema. Foram criadas duas bibliotecas de rotinas para apoio à funções específicas necessárias.

- biblioteca de rotinas de manipulação de cadeias de caracteres: contém 32 rotinas.
- biblioteca de rotinas de acesso à base de dados: é constituída das rotinas de acesso à base de dados

que armazena a descrição de um sistema. Contém 11 rotinas.

Todos os subsistemas foram feitos usando técnicas de programação modular, sendo constituídos de rotinas com em média uma página de listagem fonte. O número de rotinas em cada subsistema é o seguinte:

- Subsistema de Definição de Linguagem: 34 rotinas
- Subsistema de Manipulação da Descrição: 165 rotinas
- Subsistema de Utilitários: 15 rotinas.

A quantidade de memória em disco, gasta pelos subsistemas pelas tarefas executáveis (imagem de memória) é aproximadamente o seguinte:

- SDL 105 Kbytes
- SMD 260 Kbytes
- SUT 60 Kbytes

A descrição de um sistema é armazenado em uma base de dados, composta de 4 arquivos, cujo total de memória em disco necessária, pode ser calculado como:

$$\text{Número total} = 2 + \frac{NO}{60} + \frac{NR}{256} + \frac{NC}{24}$$

onde

NO = número de nomes de objetos mais números de sinônimos da descrição

NR = número de relações simples da descrição

NC = número de linhas de comentários da descrição.

Um arquivo de Definição de Linguagem usa em média 15 Kbytes, dependendo da linguagem definida (LDS). Além disso, os

arquivos de seleção gerados para armazenar as seleções de objetos usam em geral de 0,5 a 2,5 Kbytes, dependendo do número de objetos selecionados.

Pode-se perceber então, que apesar do volume bastante grande dos programas que constituem o SACDS, ocupando aproximadamente 450 Kbytes de disco, o consumo de espaço proveniente de sua atividade é bem pequemo, sendo que descrições, mesmo contendo grande volume de informações, raramente ultrapassam 50 Kbytes.

O tempo de resposta aos comandos é bom, sendo que nas operações do módulo EDS, que são interativas, em descrições envolvendo até 300 objetos não se notou atraso apreciável na resposta aos comandos. Os módulos SO e de geração de relatórios, que manipulam a descrição inteira, o tempo de resposta aos comandos aumenta com o crescimento da descrição, porém como são comandos que produzem listagens extensas, o atraso é tolerável. Exemplos típicos para o módulo SO, em uma descrição de 300 objetos, são tempos de 15 a 30 segundos para se efetuar uma seleção (dependendo da complexidade dos critérios de seleção). O tempo de geração de um relatório é em geral equivalente ao da impressão desse relatório em uma impressora de 300 Lpm.

CAPITULO 5

EXEMPLOS DE APLICAÇÃO DO SACDS

5.1 - Considerações Gerais

No capítulo 3 deste trabalho foram apresentadas as características que um sistema de apoio por computador à documentação de sistemas deveria ter, para ser genérico o suficiente para que pudesse ser usado com qualque tipo de sistema. No capítulo 4 foi apresentada uma implementação realizada, que procurou tornar disponível na prática um sistema com aquelas características. Neste capítulo serão apresentados dois exemplos de tipos de sistemas que podem usar o SACDS para seu apoio.

Inicialmente é apresentado o uso do SACDS com um sistema que não é um Sistema de Informação, e para isso o exemplo escolhido é uma orquestra. Esse mesmo exemplo é o que foi usado no capítulo 4 para ilustrar as considerações feitas sobre o uso de um sistema particular, onde o exemplo era apenas usado para ilustrar determinadas situações. Neste capítulo o exemplo é detalhado, sendo o mesmo sistema básico usado para ilustrar seu uso na descrição tanto de orquestras tradicionais, quanto por algumas modificações na linguagem de descrição de sintetizadores eletrônicos musicais.

A seguir, é apresentado o uso do SACDS com um Sistema de Informação, sendo para isso usada a metodologia HIPO. A

linguagem usada neste exemplo é a mesma usada em [Tr 82c] para descrever o algoritmo descrito naquele trabalho, onde a linguagem é apenas usada para descrever o algoritmo, sendo que neste capítulo a linguagem é explicada, indicando-se em que situações cada tipo de objeto ou tipo de relação deve ser aplicado.

Os dois exemplos são apresentados com o objetivo de mostrar o uso do SACDS em Sistemas de qualquer tipo, com o mesmo enfoque usado no capítulo 3. Apesar da implementação apresentada no capítulo 4 ter sido usada para a preparação destes exemplos, e a notação e comandos dessa implementação terem sido aproveitados, os exemplos são apresentados de forma a mostrar o uso do SACDS sem se deter em detalhes da implantação efetuada.

5.2 - Exemplo de Descrição de um Sistema Qualquer

Aqui serão feitos alguns comentários sobre o exemplo que tem sido usado neste trabalho, para ilustrar o uso do SACDS com sistemas de um tipo genérico. O tipo de sistema escolhido é uma orquestra, pois é um sistema relativamente simples de ser compreendido, mesmo por quem não está familiarizado com o dia a dia de uma orquestra.

A idéia inicial para se criar a linguagem de descrição de uma orquestra partiu do artigo de Krasner [Kr 80], cuja motivação foi a seguinte: vem sendo desenvolvidos instrumentos eletrônicos musicais, buscando principalmente 2 objetivos: de um lado a simulação de instrumentos acústicos cada vez com maior perfeição; e por outro, a busca de novos recursos musicais que seriam impossíveis em instrumentos acústicos. Para que os frutos desses desenvolvimentos sejam aproveitados, é neces-

sário que se coloque à disposição dos músicos, controles que os permitam criar música. Normalmente o que se tem feito é associar ao instrumento; sensores de controle similares aos já existentes para instrumentos acústicos, como por exemplo um teclado semelhante ao de um piano. Com isso, pode-se imaginar um músico capaz de tocar qualquer instrumento "simulado" ou "inventado" através de um teclado. Dessa forma, pode-se pensar em uma orquestra, que poderia por exemplo simular uma orquestra convencional, porém com cada músico operando um teclado exatamente igual a todos os outros, com a diferença que cada qual simula determinado instrumento. Essa orquestra, uma vez criada, não necessitaria executar então apenas as partituras tradicionais, pois poder-se-ia nela incluir alguns teclados que tocam instrumentos "inventados", abrindo-se então todo um novo mundo de pesquisa musical, para a orquestra como um todo, ou em outras palavras, para o compositor ou para o maestro.

Visto pelo lado do músico, seu instrumento pode ser "programado" para simular um instrumento qualquer, seja ele convencional ou não. Porém, sempre que ele está tocando sua parte da música, essa programação é estática, e ele pode se concentrar em controlar a sequência de notas e tempos que constitui a parte da música que ele produz. Nesse instante, não importa que seja possível definir no instrumento como as notas devem soar.

Visto pelo lado do maestro, sua orquestra pode ser programada para executar uma partitura qualquer, bastando enviar para cada músico a parte que ele deve executar em seu instumento. Dessa forma, quando uma partitura está sendo executada, a definição da sequência de notas que devem ser produzidas é es

tática, pois o maestro não pode alterar a sequência de notas de uma música quando ela já está sendo tocada. Dessa forma, o maestro pode se concentrar no realce de cada instrumento, andamento, encadeamento, que constitui o trabalho final produzido pela orquestra. Nesse instante não importa que é possível definir para cada músico a sequência de notas e tempos que ele deve seguir.

Colocado dessa forma, é possível construir um "instrumento" eletrônico musical que seja uma orquestra, que aceita de seu executor a definição de cada instrumento e para cada instrumento a sua parte. Dessa forma, a "orquestra eletrônica" tem vários "instrumentos eletrônicos" que não são tocados por músicos, mas por controladores que fazem as vezes de músicos. O executor dessa orquestra deve então exercer sobre esse dispositivo o mesmo tipo de controle que um maestro exerce sobre uma orquestra constituída de músicos.

A pesquisa de como o executor se comunica com esse dispositivo é motivo para uma pesquisa ergonômica, e escapa a nossos objetivos, porém quais controles serão esses, e qual deve ser o fluxo de controle entre as suas várias partes, nos interessa profundamente.

Esse dispositivo vem sendo desenvolvido e produzido com o objetivo de simular orquestras, sejam elas convencionais ou não. Como tal, suas componentes são bastante semelhantes em função com as compostas de uma orquestra convencional. Mesmo os nomes das componentes podem ser os mesmos, e assim podemos nos referir por exemplo a instrumentos, movimentos, etc. A própria descrição rigorosa de uma orquestra convencional ajudaria o desenvolvimento desse instrumento. Portanto, será feita aqui a apresentação de uma linguagem de descrição de orquestras conven-

cionais. A seguir será feita uma identificação de cada componente da "orquestra eletrônica" e a apresentação de uma linguagem estendida, onde se incluem dados que são necessários em uma descrição de uma "orquestra eletrônica".

Recursos	*Pessoal*
Instrumental;pertence Apresentada Usa; Usado; Usando Parte; Partes Envolvidas Pode ser; é do tipo Histórico	Elementos;Componente Participa; Envolve; na Música Biografia
Conhecimento	*Comando*
Rege;Regida Apresenta Constituído;faz parte Executa;Executada Executando Toca tocado;tocando Conhecida	Escolhe;Escolhido Conduz;Conduzida

Figura 5.1

5.2.1 - Uma Linguagem para a Descrição de Orquestras

O objetivo da linguagem para a descrição de orquestras é a obtenção de uma descrição dos recursos que uma orquestra possui, e assim conseguir determinar, por exemplo:

- que partituras uma orquestra pode executar;
- que instrumentos faltam para que uma partitura possa ser executada;
- no caso de alguns músicos saberem tocar mais de um instrumento, como deve ser a distribuição de instrumentos para executar uma partitura;
- qual a configuração mínima da orquestra para determinado repertório; etc.

Na figura 4.3 está mostrada a sintaxe dessa linguagem, bem como os tipos de objetos que a constituem. Os tipos de relações são classificados segundo 4 aspectos, que estão mostrados na figura 5.1. Seu significado é o seguinte:

Recursos - neste aspecto está incluida a descrição dos recursos que a orquestra possui, e com os quais pode contar quando é necessário apresentar uma partitura. Está também incluída a descrição de que recursos são necessários à apresentação de cada partitura.

Pessoal - neste aspecto está incluída a descrição das pessoas que compõem a orquestra, ou que participam de cada execução.

Conhecimento - neste aspecto está incluida a parte da descrição que envolve o que as pessoas sabem, ou devem saber para que a orquestra apresente uma partitura.

Comando - Neste aspecto se inclui a descrição das ordens e escolhas que o maestro deve impor sobre a orquestra.

Para que a descrição de uma orquestra possa ser útil, é necessário que tal descrição não envolva exclusivamente a orquestra que é o foco de interesse, mas todas as informações que a possam afetar. Por exemplo, a descrição de quais

E N T R A D A D E D A D O S

```
ARQUIVO=RICKM,DSC,REFERENCIA-A-DESCRICAO,ATUALIZA,LISTAR-FONTE,SEM-INDICE-CRUZAD

 1 =>  ORQUESTRA RICK=*AKEMAN
 2 =>     ELEMENTOS SAO RICK=*AKEMAN
 3 =>     INSTRUMENTAL PIANO, PIANO-ACUSTICO, PROPHET, PROPHET-5,
 4 =>       PROPHET-10, ORGAO, OBX-1, RMI, MINI-MOOG, ORGAO-CS80,
 5 =>       SYNCLAVIER, STRING-MACHINE
 6 =>  ORQUESTRA RICK'S-BACKING-MUSICIANS
 7 =>     ELEMENTOS SAO STEVE-BARNACLE, TIM-STONE,
 8 =>      'TONY-FERNANDEZ, GARY-BARNACLE,FRANK-RICOTTI
 9 =>  MUSICO STEVE-BARNACLE
10 =>     TOCA GUITARRA-BAIXO
11 =>  MUSICO TIM-STONE, TONY-FERNANDEZ
12 =>     TOCA GUITARRA
13 =>  MUSICO GARY-BARNACLE
14 =>     TOCA SAXOFONE
15 =>  MUSICO FRANK-RICOTTI
16 =>     TOCA BATERIA
17 =>  ORQUESTRA WITHOUT-NAME
18 =>     ELEMENTOS ROTHSTEIN-J., KATZ-D., BRADLES-D., GOOD-T.,
19 =>       CLAY-L., MCGEE-A., DUKOV-B., COURSON-M., ROBERTSON-G.
20 =>       NEWLANDS-D., ANDRADE-L.
21 =>     ELEMENTOS DAZIEL-A., WILLISON-P., TRUMAN-B., ROBINSON-M.,
22 =>       RUNSWICK-D., MCGEE-R., GREGORY-J., SANDEMAN-D., WHITTING-T.,
23 =>       THEODORE-D., PUDDY-K., WEINBERG-T., SCHEEN-G., HAMMOND-H.
24 =>     ELEMENTOS THOMSON-M., EASTHOPE-P., JENKINS-J., MILLER-J.,
25 =>       WALLIS-J., HARDIE, WILSON, VICKIE-BROWN, SONIA-JONES,
26 =>       STEVIE-LANGE
27 =>  MUSICO ROTHSTEIN-J., KATZ-D., BRADLES-D., GOOD-T., CLAY-L.,
28 =>       MCGEE-A.,, DUKOV-B.
29 =>     TOCA VIOLINO
30 =>  MUSICO COURSON-M., ROBERTSON-G., NEWLANDS-D., ANDRADE-L.
31 =>     TOCA VIOLA
32 =>  MUSICO DAZIEL-A., WILLISON-P., TRUMAN-B., ROBINSON-M.
33 =>     TOCA VIOLONCELLO
34 =>  MUSICO RUNSWICK-D., MCGEE-R.
35 =>     TOCA BAIXO
36 =>  MUSICO GREGORY-J., SANDEMAN-D.
37 =>     TOCA FLAUTA
38 =>  MUSICO WHITTING-T., THEODORE-D.
39 =>     TOCA OBOE'
40 =>  MUSICO PUDDY-K., WEINBERG-T.
41 =>     TOCA CLARINETA
42 =>  MUSICO SCHEEN-G., HAMMOND-H.
43 =>     TOCA FAGOTE
44 =>  MUSICO THOMSON-M, EASTHOPE-P.
45 =>     TOCA CORNETA
46 =>  MUSICO JENKINS-J.
47 =>     TOCA TUBA
48 =>  MUSICO MILLER-J.,WALLIS-J.
49 =>     TOCA TROMPETE
50 =>  MUSICO HARDIE, WILSON
51 =>     TOCA TROMBONE
52 =>  MUSICO VICKIE-BROWN, SONIA-JONES, STEVIE-LANGE
53 =>     TOCA VOCAL
54 =>  PARTITURA 1984-OVERTURE
55 =>     PARTES-ENVOLVIDAS 1984-OVERTURE-PART-ONE, 1984-OVERTURE-PART-TWO,
56 =>       1984-OVERTURE-WAR-GAMES
57 =>     APRESENTADA POR WITHOUT-NAME, RICK'S-BACKING-MUSICIANS,
58 =>       RICK=*AKEMAN
59 =>     CONHECIDA COMO OVERTURE
60 =>  MOVIMENTO 1984-OVERTURE-PART-ONE
61 =>     USA VIOLINO, VIOLA, VIOLONCELLO, BAIXO, CLARINETA, FLAUTA, OBOE',
62 =>       FAGOTE, TROMBONE, TROMPETE, CORNETA, HARPA, FLAUTIN, BATERIA,
63 =>       MARIMBA
64 =>  MOVIMENTO 1984-OVERTURE-PART-TWO
65 =>     USA BATERIA, GUITARRA-BAIXO, GUITARRA, VIOLINO, VIOLA, CLARINETA,
66 =>       VIOLONCELLO, BAIXO, FLAUTA, OBOE', FAGOTE, CORNETA
67 =>  MOVIMENTO 1984-OVERTURE-WAR-GAMES
68 =>     ENVOLVE CHAKA-KHAN
69 =>      TOCANDO SOLO-DE-VOZ
70 =>     USA BATERIA, GUITARRA-BAIXO, GUITARRA, VOCAL
71 =>  MUSICO RICK=*AKEMAN
72 =>     TOCA PIANO-ACUSTICO, PROPHET
73 =>      NA-MUSICA 1984-OVERTURE-PART-ONE
74 =>     TOCA PIANO, PROPHET, RMI, ORGAO
75 =>      NA-MUSICA 1984-OVERTURE-PART-TWO
76 =>     TOCA ORGAO, PROPHET, RMI, PIANO
77 =>      NA-MUSICA 1984-OVERTURE-WAR-GAMES
78 =>  T-INSTRUMENTO BAIXO, BATERIA, CLARINETA, CORNETA FAGOTE FLAUTA
79 =>       VIOLA, VIOLINO, VIOLONCELLO, VOCAL
80 =>  T-INSTRUMENTO FLAUTIN, GUITARRA, GUITARRA-BAIXO, HARPA, MARIMBA,
81 =>       OBOE', SAXOFONE, SOLO-DE-VOZ, TROMBONE, TROMPETE, TUBA
82 =>  EOF
```

Figura 5.2

são as pessoas que compõem a orquestra, se refere à orquestra propriamente dita. Porém, um músico pode ser elemento de outras orquestras, ou grupos musicais, e isso deve ser incluído na descrição da orquestra.

Um exemplo de descrição de orquestra, e que inclusive envolve vários grupos musicais é o que aparece na figura 5.2.

5.2.2 - Uma Linguagem para a Descrição de Sistemas Musicais

Para que um sistema musical eletrônico possa criar música da mesma forma que uma orquestra, devem existir nesse sistema módulos que correspondam aos vários objetos que constituem uma orquestra convencional. Assim na linguagem de descrição desses sistemas musicais eletrônicos existirão pelo menos todos os tipos de objetos que existem na linguagem de descrição de orquestras. O significado de cada tipo de objeto passa a ser o seguinte:

Maestro - se refere ao músico que irá controlar o sistema. Descreve como e quais controles são postos a sua disposição, e portanto se comporta como elemento de ligação entre o sistema e o mundo exterior.

Orquestra - se refere aos recursos que podem ser usados para produzir música. Descreve quais instrumentos podem ser simulados e quanto da capacidade do sistema é gasto para a simulação de cada instrumento.

Tipo de Instrumento - se refere ao conjunto de dados que deve estar disponível à máquina para que um instrumento seja simulado. Descreve quando e como esses dados são usados, e como podem ser selecionados para controlar os módulos que produzem sons.

Instrumento - se refere a cada módulo do sistema que produz som. Um sistema musical terá uma certa quantidade de módulos capazes de sintetizar sons que simulam instrumentos, convencionais ou não. Com a definição de cada instrumento, disponível através dos objetos TIPO-DE-INSTRUMENTO, os módulos podem ser configurados para os conjuntos de instrumentos desejados.

Músico - se refere aos módulos do sistema capazes de controlar os geradores de som. Em uma orquestra convencional, um músico é uma pessoa que traduz uma partitura nos movimentos adequados que fazem seu instrumento produzir o som necessário. No sistema musical eletrônico, a descrição de um músico corresponderá à de um módulo que recebe uma definição de um movimento de uma partitura e controla um gerador de som associado a ele para produzir o som necessário no momento desejado.

Parte - corresponde ao tipo de objeto MOVIMENTO na linguagem para descrição de orquestras. Se refere a sequência de notas e tempos que constitui uma sequência indivisível e imutável associada a um único instrumento.

Partitura - se refere à definição do encadeamento de várias partes. Cada parte é associada a um instrumento, e sempre que uma parte está em execução ela não pode ser interrompida. Porém, o encadeamento, repetição, andamento, etc. entre as várias partes é controlada pelo maestro, e o resultado da execução das várias partes, é a execução da partitura. A descrição indica como o encadeamento deve se feito em linhas gerais, e quais devem ser os instrumentos associados a cada parte.

Repertório - se refere ao conjunto de partituras que está, ou pode ser definido no sistema.

As relações da linguagem de descrição de sistemas musicais eletrônicos envolvendo esses objetos são as mesmas da linguagem de descrição de orquestras, e os aspectos a que essas relações se referem são também os mesmos, porém com significado um pouco diferente, que é explicado a seguir:

Recursos - se refere a descrição dos recursos de simulação que o instrumento possui, bem como a descrição dos recursos necessários à execução de uma partitura. Por exemplo, inclui a descrição da quantidade de módulos de simulação e controle que existem, e que instrumentos cada módulo pode simular. Inclui também a descrição de quais são os instrumentos que devem ser simulados para a execução de uma dada partitura, e dessa forma, qual módulo usará qual definição de tipo de instrumento.

Pessoal - se refere à descrição de como os módulos de controle podem ser dispostos, para controlar os simuladores para cada instrumento em cada execução.

Conhecimento - se refere à descrição da capacidade do sistema armazenar a definição de partes e partituras, e às informações básicas sobre o encadeamento entre as várias partes.

Comando - se refere à descrição das ordens e escolhas que o maestro pode executar sobre o sistema e à definição nas partituras sobre em que pontos da execução as ordens podem ser atendidas.

Com o que foi apresentado da linguagem até aqui, é possível uma descrição estática do sistema, envolvendo ele próprio e as músicas que ele pode produzir. Para que uma descrição fique completa, é necessário acrescentar à descrição infor

mações sobre como as ordens emitidas pelo maestro passam de módulo para módulo, indicando a forma como elas atuam na atividade de cada um. Esse aspecto, que será chamado de dinâmica, inclui 2 novos tipos de objetos na linguagem:

Mensagem - indica a existência, ou possibilidade de existir uma ordem ou um conjunto de informações que deve ser enviado a algum objeto. Associada a uma mensagem existe a indicação de quando ela deve ser enviada, qual o objeto que a recebe e quais as informações que ela deve conter.

Situação - Um objeto tipo mensagem envia a informação que lhe é associada quando ocorre uma situação. O objeto tipo situação estabelece então quando uma mensagem deve ser enviada.

Os dois tipos de objetos constituem, na realidade, uma forma de se descrever uma situação lógica através da representação de uma rede de Petri, em que a mensagem é um sinal associado a uma informação, e a situação corresponde a condição de disparo.

Na figura 5.3 está apresentada a sintaxe da linguagem depois da inclusão dos 2 novos tipos de objetos e as relações que os envolvem.

```
OBJETO TIPO INDEFINIDO

OBJETO TIPO INSTRUMENTO
        COM A RELACAO E-DO-TIPO                          OBJETO T-INSTRUMENTO
        COM A RELACAO PERTENCE                           OBJETO ORQUESTRA
        COM A RELACAO RECEBE                             OBJETO MENSAGEM
        COM A RELACAO TOCADO POR                         OBJETO MUSICO
        COM A RELACAO USADO POR                          OBJETO MOVIMENTO

OBJETO TIPO MAESTRO
        COM A RELACAO CONDUZ                             OBJETO ORQUESTRA
        COM A RELACAO ESCOLHE                            OBJETO REPERTORIO
        COM A RELACAO PRODUZ                             OBJETO MENSAGEM
        COM A RELACAO RECEBE                             OBJETO MENSAGEM
        COM A RELACAO REGE                               OBJETO PARTITURA

OBJETO TIPO MENSAGEM
       -COM A RELACAO ASSOCIADA A                        :
                      OBJETO MAESTRO
                      OBJETO MUSICO
                      OBJETO ORQUESTRA
                      OBJETO PARTITURA
                      OBJETO REPERTORIO
                      OBJETO T-INSTRUMENTO
        COM A RELACAO ENVIADA POR                        OBJETO MENSAGEM
        COM A RELACAO ENVIE                              OBJETO MENSAGEM
       -COM A RELACAO RECEBIDA POR                       :
                      OBJETO INSTRUMENTO
                      OBJETO MAESTRO
                      OBJETO MOVIMENTO
                      OBJETO MUSICO
                      OBJETO ORQUESTRA
                      OBJETO PARTITURA
                      OBJETO REPERTORIO
                      OBJETO T-INSTRUMENTO
       -COM A RELACAO VERIFICA                           :
                      OBJETO MENSAGEM
                      OBJETO SITUACAO

OBJETO TIPO MOVIMENTO
        COM A RELACAO CONTRIBUI PARA                     OBJETO SITUACAO
        COM A RELACAO ENVOLVE                            OBJETO MUSICO
        COM A RELACAO EXECUTADA POR                      OBJETO MUSICO
        COM A RELACAO NECESSARIO PARA                    OBJETO SITUACAO
        COM A RELACAO PARTE DE                           OBJETO PARTITURA
        COM A RELACAO RECEBE                             OBJETO MENSAGEM
       -COM A RELACAO USA                                :
                      OBJETO INSTRUMENTO
                      OBJETO T-INSTRUMENTO

OBJETO TIPO MUSICO
        COM A RELACAO COMPONENTE DE                      OBJETO ORQUESTRA
        COM A RELACAO EXECUTA                            OBJETO MOVIMENTO
       -COM A RELACAO PARTICIPA DE                       :
                      OBJETO MOVIMENTO
                      OBJETO PARTITURA
        COM A RELACAO PRODUZ                             OBJETO MENSAGEM
        COM A RELACAO RECEBE                             OBJETO MENSAGEM
       -COM A RELACAO TOCA                               :
                      OBJETO INSTRUMENTO
                      OBJETO T-INSTRUMENTO
```

FIGURA 5.3 (CONTINUA)

```
BJETO TIPO ORQUESTRA
        COM A RELACAO APRESENTA                    OBJETO PARTITURA
        COM A RELACAO CONDUZIDA                    OBJETO MAESTRO
        COM A RELACAO ELEMENTOS                    OBJETO MUSICO
        COM A RELACAO INSTRUMENTAL                 OBJETO INSTRUMENTO
        COM A RELACAO PRODUZ                       OBJETO MENSAGEM
        COM A RELACAO RECEBE                       OBJETO MENSAGEM

BJETO TIPO PARTITURA
        COM A RELACAO APRESENTADA POR              OBJETO ORQUESTRA
        COM A RELACAO CONHECIDA COMO               OBJETO PARTITURA
        COM A RELACAO CONTRIBUI PARA               OBJETO SITUACAO
        COM A RELACAO ENVOLVE                      OBJETO MUSICO
        COM A RELACAO FAZ-PARTE DE                 OBJETO REPERTORIO
        COM A RELACAO NECESSARIO PARA              OBJETO SITUACAO
        COM A RELACAO PARTES-ENVOLVIDAS            OBJETO MOVIMENTO
        COM A RELACAO PRODUZ                       OBJETO MENSAGEM
        COM A RELACAO RECEBE                       OBJETO MENSAGEM
        COM A RELACAO REGIDA POR                   OBJETO MAESTRO

BJETO TIPO REPERTORIO
        COM A RELACAO CONSTITUIDO DE               OBJETO PARTITURA
        COM A RELACAO CONTRIBUI PARA               OBJETO SITUACAO
        COM A RELACAO ESCOLHIDO                    OBJETO MAESTRO
        COM A RELACAO NECESSARIO PARA              OBJETO SITUACAO
        COM A RELACAO PRODUZ                       OBJETO MENSAGEM
        COM A RELACAO RECEBE                       OBJETO MENSAGEM

BJETO TIPO SITUACAO
        COM A RELACAO CONTRIBUI PARA               OBJETO SITUACAO
       -COM A RELACAO DEVE-HAVER                  :
                    OBJETO MOVIMENTO
                    OBJETO PARTITURA
                    OBJETO REPERTORIO
                    OBJETO SITUACAO
        COM A RELACAO NECESSARIO PARA              OBJETO SITUACAO
       -COM A RELACAO PODE-HAVER                  :
                    OBJETO MOVIMENTO
                    OBJETO PARTITURA
                    OBJETO REPERTORIO
                    OBJETO SITUACAO

BJETO TIPO T-INSTRUMENTO
        COM A RELACAO PODE SER                     OBJETO INSTRUMENTO
        COM A RELACAO PRODUZ                       OBJETO MENSAGEM
        COM A RELACAO RECEBE                       OBJETO MENSAGEM
        COM A RELACAO TOCADO POR                   OBJETO MUSICO
        COM A RELACAO USADO POR                    OBJETO MOVIMENTO
```

FIGURA 5.3 (FINAL)

5.3 - Exemplo de Descrição de um Sistema de Informação

Muito se tem pesquisado e realizado sobre o projeto e análise de Sistemas de Informação, e associado a essas atividades, à descrição e documentação desses sistemas. Dessas pesquisas, tem surgido várias metodologias destinadas a facilitar ou sistematizar os trabalhos em Sistemas de Informação. Como o espectro dessa tarefa é muito amplo, e as condições e parâmetros em que ela se realiza muito variável, todas essas metodologias tem seu campo de aplicação, e se bem que não mutuamente exclusivos, cada uma tem seus próprios atrativos e vantagens, muitas até apresentando idéias e soluções que não são cobertas pelas demais.

Em comum a todas as metodologias está o fato de que elas manipulam um volume de dados bastante grande, durante a fase de projeto esses dados se alteram bastante e rapidamente. Cada metodologia tem seu próprio conjunto de regras para estruturar esses dados em formatos que lhe são convenientes. Dessa forma, muito trabalho de documentação de um sistema em determinada metodologia, consiste em se formatar os dados referentes ao sistema segundo tais regras. Quando os dados a respeito do sistema se alteram, pode ser necessário refazer parte dessa documentação, como por exemplo tabelas e gráficos. Além disso, como uma mesma informação pode aparecer em vários lugares, tais alterações incorrem no risco de se documentar situações erradas ou contraditórias.

A melhor forma de diminuir esses problemas é fazer toda a documentação com o apoio de um computador, que armazena as informações sem redundância, e gera os documentos sempre que solicitado, baseados sempre nas informações atualizadas.

Muitas das metodologias existentes dispoem de software destinado a esse apoio, enquanto muitas não. Muito do software necessário ao apoio à determinada metodologia é semelhante ao de várias outras variando apenas em decorrência do tipo de análise que deve ser feita sobre as especificações da metodologia, e dos tipos de relatórios que devem ser gerados.

Dessa forma, desenvolver novos sistemas de software para apoiar as demais metodologias levaria à construção de muitas partes redundantes.

O enfoque que o SACDS teve ao ser desenvolvido é tal que permite a definição de uma linguagem de descrição tendo como base aquela usada pela metodologia que deve ser apoiada, e os relatórios gerados definidos também de acordo com os especificados pela metodologia. Esse sistema pode então prestar apoio a metodologias para projeto e análise de Sistemas de Informação, principalmente nos aspectos comuns às várias metodologias. Os aspectos específicos de determinada metodologia dizem respeito aos formatos de relatórios associados a ela. O SACDS pode não dispor de capacidade para gerar determinado tipo de relatório, porém é possível ser expandido para dispor de tal capacidade, com a vantagem que uma vez implantado, tal formato estaria disponível também para outras metodologias que por ventura o estiver usando.

Na seção seguinte serão feitos alguns comentários sobre uma linguagem exemplo para o SACDS, destinada a prestar apoio a uma metodologia de documentação e projeto de Sistemas de Informação.

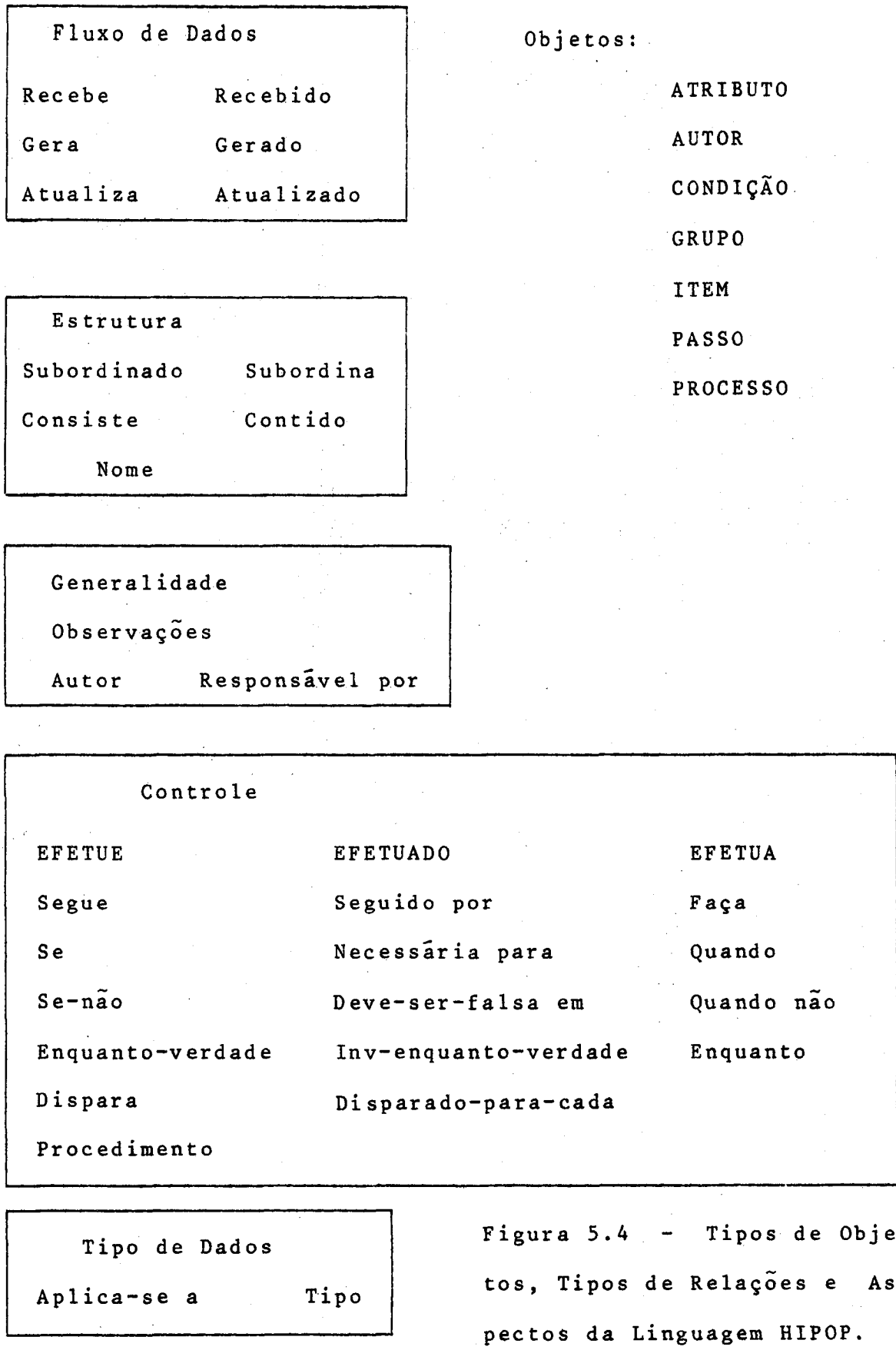


Figura 5.4 - Tipos de Objetos, Tipos de Relações e Aspectos da Linguagem HIPOP.

5.3.1 - Uma Linguagem para a Metodologia HIPO

Desenvolvida pela IBM, a metodologia HIPO (Hierarchy plus Input-Process-Output) tem como objetivo estabelecer normas de auxílio ao projeto, e documentação de programas ou sistemas de programação. Essas normas orientam o projetista a desenvolver um sistema estruturado, e a criar um conjunto de gráficos que documentam os programas e as estruturas de dados projetados em um formato padronizado.

Essa metodologia não usa por si nenhum recurso de computação, limitando-se a estabelecer normas de documentação. Existe um utilitário da própria IBM desenvolvido para permitir a descrição de relatórios tipo HIPO em computador, que pode então gerá-los em uma impressora. Esse utilitário, chamado HIPO DRAW, aceita do usuário a descrição do formato que o relatório deve ter, limitando-se a fazer uma descrição das posições onde as linhas, traços e textos serão colocados na folha, não realizando nenhuma análise sobre o significado desses textos ou traços.

Uma descrição detalhada do HIPO e do HIPO DRAW pode ser encontrado em [IBM 74].

O SACDS pode prestar auxílio a quem usa a metodologia HIPO, se dispuser de uma linguagem adaptada a ela, e de capacidade para gerar os relatórios correspondentes. Nesta seção é apresentada uma linguagem com esse objetivo. Quanto à geração de relatórios, é necessário software adequado, até a presente data não desenvolvido.

A linguagem aqui apresentada, que foi denominada Linguagem HIPOP, contém 5 aspectos de descrição, feita através de

objetos de 7 tipos diferentes e 34 tipos de relacionamentos. Na figura 5.4 estão listados esses elementos da linguagem.

Um conjunto de gráficos que constitui a documentação de um sistema em HIPO contêm diagramas de 2 formatos distintos: um que descreve a estrutura do sistema (hierarchy) e um que descreve os algoritmos e fluxo de dados no sistema (input-process-output). Na linguagem HIPOP os tipos de objetos correspondem aos blocos de construção dos diagramas, os aspectos correspondem à parte dos diagramas onde cada informação é descrita e as relações estabelecem as situações particulares onde cada bloco ocorre nos diagramas. A seguir será feita descrição de cada tipo de objeto separadamente.

Processo - é o tipo de objeto que caracteriza os processos dos diagramas HIPO. Os processos dos diagramas são representados nesta linguagem como objetos do tipo PROCESSO ou PASSO. Pode-se fazer a distinção entre os dois estabelecendo-se que um objeto que representa um procedimento muito simples seja um PASSO, e um objeto que representa um procedimento mais elaborado seja um PROCESSO. A hierarquia de um sistema é descrita descrevendo-se para cada PASSO ou PROCESSO, quais os outros PASSOS ou PROCESSOS são a ele subordinados. Os PASSOS e PROCESSOS são os objetos ativos do sistema, e dessa forma são os objetos que RECEBEM, GERAM e ATUALIZAM os dados, estabelecem o fluxo de dados de controle, etc.

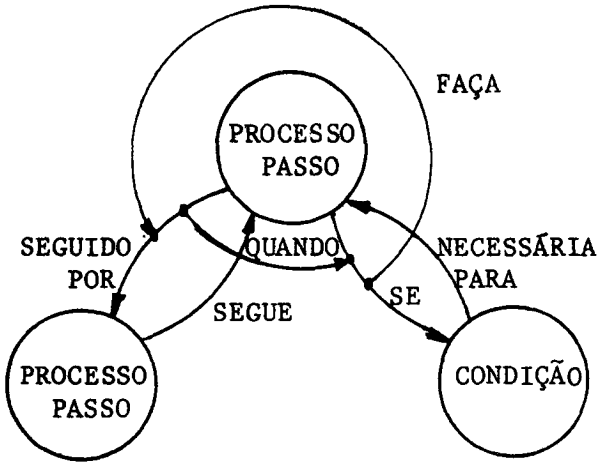
Passo - é o tipo de objeto que caracteriza os passos dos procedimentos do sistema. Tem as mesmas características dos objetos tipo PROCESSO, a menos das relações descritivas, por exemplo AUTOR, que para os objetos tipo PASSO não se aplicam.

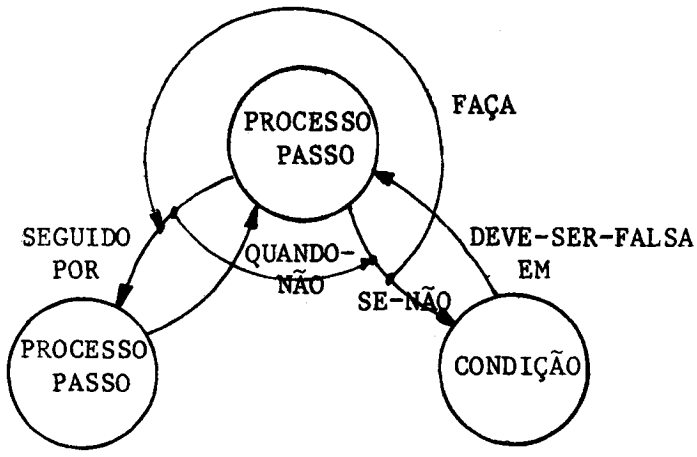
```
OBJETO TIPO ATRIBUTO
        -COM A RELACAO APLICA-SE A                          :
                        OBJETO GRUPO
                        OBJETO ITEM
                        OBJETO PROCESSO

OBJETO TIPO AUTOR
         COM A RELACAO RESPONSAVEL POR                       OBJETO PROCESSO

OBJETO TIPO CONDICAO
        -COM A RELACAO DEVE-SER-FALSA EM                    :
                        OBJETO PASSO
                        OBJETO PROCESSO
        -COM A RELACAO ENQUANTO-VERDADE                     :
                        OBJETO PASSO
                        OBJETO PROCESSO
        -COM A RELACAO NECESSARIA PARA                      :
                        OBJETO PASSO
                        OBJETO PROCESSO

OBJETO TIPO GRUPO
        -COM A RELACAO ATUALIZADO POR                       :
                        OBJETO PASSO
                        OBJETO PROCESSO
        -COM A RELACAO CONSISTE DE                          :
                        OBJETO GRUPO
                        OBJETO ITEM
         COM A RELACAO CONTIDO EM                           OBJETO GRUPO
        -COM A RELACAO DISPARA                              :
                        OBJETO PASSO
                        OBJETO PROCESSO
        -COM A RELACAO GERADO POR                           :
                        OBJETO PASSO
                        OBJETO PROCESSO
        -COM A RELACAO RECEBIDO POR                         :
                        OBJETO PASSO
                        OBJETO PROCESSO
         COM A RELACAO TIPO                                 OBJETO ATRIBUTO

OBJETO TIPO INDEFINIDO

OBJETO TIPO ITEM
        -COM A RELACAO ATUALIZADO POR                       :
                        OBJETO PASSO
                        OBJETO PROCESSO
         COM A RELACAO CONTIDO EM                           OBJETO GRUPO
        -COM A RELACAO DISPARA                              :
                        OBJETO PASSO
                        OBJETO PROCESSO
        -COM A RELACAO GERADO POR                           :
                        OBJETO PASSO
                        OBJETO PROCESSO
        -COM A RELACAO RECEBIDO POR                         :
                        OBJETO PASSO
                        OBJETO PROCESSO
         COM A RELACAO TIPO                                 OBJETO ATRIBUTO

OBJETO TIPO PASSO
        -COM A RELACAO ATUALIZA                             :
                        OBJETO GRUPO
                        OBJETO ITEM
        -COM A RELACAO DISPARADO-PARA-CADA                  :
                        OBJETO GRUPO
                        OBJETO ITEM
        -COM A RELACAO EFETUADO POR                         :
                        OBJETO PASSO
                        OBJETO PROCESSO
```

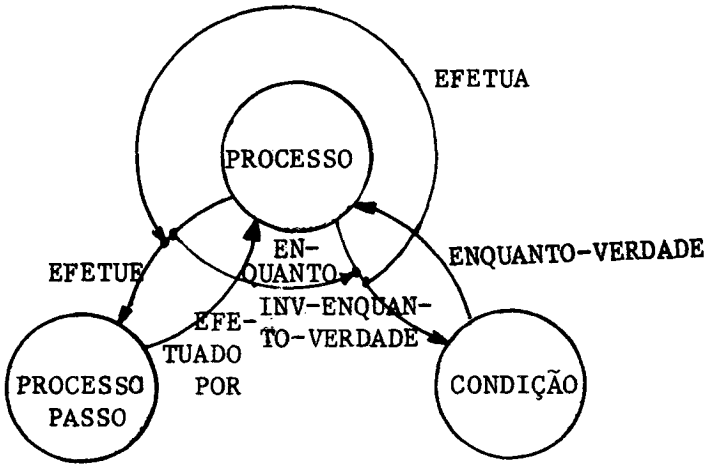
FIGURA 5.5 (CONTINUA)

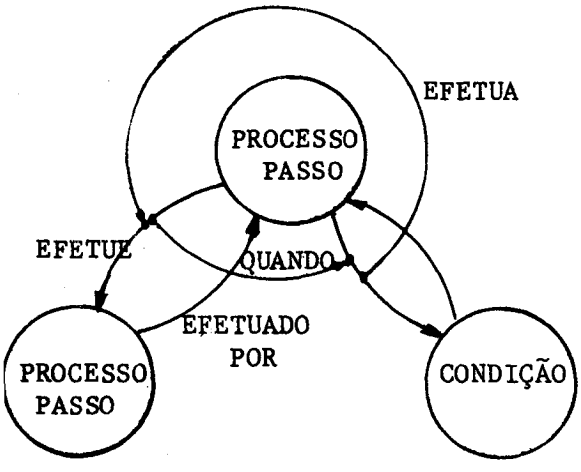
```
        -COM A RELACAO EFETUE                                    :
                          OBJETO PASSO
                          OBJETO PROCESSO
        -COM A RELACAO GERA                                      :
                          OBJETO GRUPO
                          OBJETO ITEM
         COM A RELACAO INV-ENQUANTO-VERDADE                       OBJETO CONDICAO
        -COM A RELACAO RECEBE                                    :
                          OBJETO GRUPO
                          OBJETO ITEM
         COM A RELACAO SE                                         OBJETO CONDICAO
         COM A RELACAO SE-NAO                                     OBJETO CONDICAO
        -COM A RELACAO SEGUE                                     :
                          OBJETO PASSO
                          OBJETO PROCESSO
        -COM A RELACAO SEGUIDO POR                               :
                          OBJETO PASSO
                          OBJETO PROCESSO
         COM A RELACAO SUBORDINA                                  OBJETO PASSO
        -COM A RELACAO SUBORDINADO A                             :
                          OBJETO PASSO
                          OBJETO PROCESSO

OBJETO TIPO PROCESSO
        -COM A RELACAO ATUALIZA                                  :
                          OBJETO GRUPO
                          OBJETO ITEM
         COM A RELACAO AUTOR E'                                   OBJETO AUTOR
        -COM A RELACAO DISPARADO-PARA-CADA                       :
                          OBJETO GRUPO
                          OBJETO ITEM
        -COM A RELACAO EFETUADO POR                              :
                          OBJETO PASSO
                          OBJETO PROCESSO
        -COM A RELACAO EFETUE                                    :
                          OBJETO PASSO
                          OBJETO PROCESSO
        -COM A RELACAO GERA                                      :
                          OBJETO GRUPO
                          OBJETO ITEM
         COM A RELACAO INV-ENQUANTO-VERDADE                       OBJETO CONDICAO
         COM A RELACAO NOME                                       OBJETO PROCESSO
        -COM A RELACAO RECEBE                                    :
                          OBJETO GRUPO
                          OBJETO ITEM
         COM A RELACAO SE                                         OBJETO CONDICAO
         COM A RELACAO SE-NAO                                     OBJETO CONDICAO
        -COM A RELACAO SEGUE                                     :
                          OBJETO PASSO
                          OBJETO PROCESSO
        -COM A RELACAO SEGUIDO POR                               :
                          OBJETO PASSO
                          OBJETO PROCESSO
        -COM A RELACAO SUBORDINA                                 :
                          OBJETO PASSO
                          OBJETO PROCESSO
         COM A RELACAO SUBORDINADO A                              OBJETO PROCESSO
         COM A RELACAO TIPO                                       OBJETO ATRIBUTO
```

FIGURA 5.5 (FINAL)


FAÇA
PROCESSO PASSO
SEGUIDO POR
QUANDO
NECESSÁRIA PARA
SE
SEGUE
PROCESSO PASSO
CONDIÇÃO


FAÇA
PROCESSO PASSO
SEGUIDO POR
QUANDO-NÃO
DEVE-SER-FALSA EM
SE-NÃO
PROCESSO PASSO
CONDIÇÃO


EFETUA
PROCESSO
EFETUE
EN-QUANTO
ENQUANTO-VERDADE
EFE-TUADO POR
INV-ENQUAN-TO-VERDADE
PROCESSO PASSO
CONDIÇÃO


EFETUA
PROCESSO PASSO
EFETUE
QUANDO
EFETUADO POR
PROCESSO PASSO
CONDIÇÃO

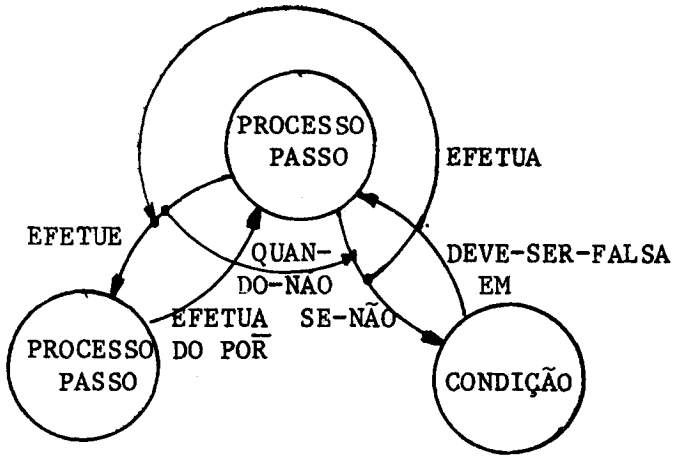

PROCESSO PASSO
EFETUA
EFETUE
QUAN-DO-NÃO
DEVE-SER-FALSA EM
EFETUADO POR
SE-NÃO
PROCESSO PASSO
CONDIÇÃO

Figura 5.6

Grupo - é o tipo de objeto que caracteriza os agrupamentos de dados do sistema. Um objeto tipo GRUPO consiste de outros objetos tipo GRUPO ou ITEM. Todos os dados do sistema descrito são caracterizados nesta linguagem através de objetos tipo GRUPO ou tipo ITEM. O tipo do dado no sistema (por exemplo, fita, fichário, etc) é descrito em HIPOP através de um ATRIBUTO.

Item - é o tipo de objeto que caracteriza os itens elementares de dados do sistema. De maneira geral segue as mesmas normas dos objetos tipo GRUPO.

Condição - é o tipo de objeto que descreve quando um PROCESSO do sistema é ativado. É bastante frequente que ao final de um PROCESSO (ou PASSO) o sistema continue a atividade executando um outro PROCESSO (ou PASSO), e assim por diante,os processos vão sendo executados sequencialmente. Quando existe uma situação em que o prosseguimento da execução depende de outros fatores, eles são descritos através de um objeto tipo CONDIÇÃO.

Atributo - é o tipo de objeto que se usa para caracterizar situações especiais ou parâmetros de objetos tipo GRUPO, ITEM ou PROCESSO.

Autor - é o tipo de objeto que descreve a pessoa responsável pela documentação de um PROCESSO.

Na figura 5.5 estão mostrados esses tipos de objetos, juntamente com todos os relacionamentos simples que podem ser definidos usando essa linguagem. Na figura 5.6 estão mostrados os relacionamentos triplos que podem ser definidos.

Os relacionamentos estão classificados em 5 aspectos, mostrados na figura 5.7. Seu significado é o seguinte:

```
160 -> APLICA-SE A                    (APLIC ) = TIPO-DE-DADOS
161 -> ATUALIZA                       (ATUALZ) = FLUXO-DE-DADOS
162 -> ATUALIZADO POR                 (ATLZDO) = FLUXO-DE-DADOS
163 -> AUTOR E'                       (AUTOR ) = GENERALIDADE
164 -> CONSISTE DE                    (CONSIS) = ESTRUTURA
165 -> CONTIDO EM                     (CONTDO) = ESTRUTURA
166 -> DEVE-SER-FALSA EM              (DEVEF ) = CONTROLE
167 -> DISPARA                        (DISPR ) = CONTROLE
168 -> DISPARADO-PARA-CADA            (DISPCD) = CONTROLE
169 -> EFETUA                         (EFETUA) = CONTROLE
170 -> EFETUADO POR                   (EFETDO) = CONTROLE
171 -> EFETUE                         (EFETUE) = CONTROLE
172 -> ENQUANTO                       (ENQNTO) = CONTROLE
173 -> ENQUANTO-VERDADE               (ENQV  ) = CONTROLE
174 -> FACA                           (FACA  ) = CONTROLE
175 -> GERA                           (GERA  ) = FLUXO-DE-DADOS
176 -> GERADO POR                     (GERADO) = FLUXO-DE-DADOS
177 -> INV-ENQUANTO-VERDADE           (*ENQV*) = CONTROLE
178 -> NECESSARIA PARA                (NECSS ) = CONTROLE
179 -> NOME                           (NOME  ) = ESTRUTURA
180 -> OBSERVACOES                    (OBS   ) = GENERALIDADE
181 -> PROCEDIMENTO                   (PROC  ) = CONTROLE
182 -> QUANDO                         (QUANDO) = CONTROLE
183 -> QUANDO-NAO                     (QDONAO) = CONTROLE
184 -> RECEBE                         (RECEBE) = FLUXO-DE-DADOS
185 -> RECEBIDO POR                   (RECEBD) = FLUXO-DE-DADOS
186 -> RESPONSAVEL POR                (RESP  ) = GENERALIDADE
187 -> SE                             (SE    ) = CONTROLE
188 -> SE-NAO                         (SENAO ) = CONTROLE
189 -> SEGUE                          (SEGUE ) = CONTROLE
190 -> SEGUIDO POR                    (SEGDO ) = CONTROLE
191 -> SUBORDINA                      (SUBORD) = ESTRUTURA
192 -> SUBORDINADO A                  (SUBRDD) = ESTRUTURA
193 -> TIPO                           (TIPO  ) = TIPO-DE-DADOS
194 ->
```

FIGURA 5.7

Fluxo de dados - neste aspecto são descritas as operações com os dados que os processos executam. Consta das relações que descrevem como os dados são gerados, recebidos e atualizados pelos processos, e caracterizam as informações que nos diagramas "input-process-output" relacionam as seções de "input" e "output" com a seção "process".

Controle - neste aspecto é descrita a forma como os vários processos e passos são executados, estabelecendo a sequência e as condições em que os processos são ativados.Consta das relações que indicam quando um processo ou passo segue outro, incondicionalmente ou sob que condição o faz, sob que condições um processo ou passo é repetido, ou quando um processo ou passo é efetuado como uma chamada de subrotina. Além disso, pode-se indicar quando a ocorrência de determinados dados para o sistema determinam o disparo de algum PROCESSO ou PASSO. As relações desse aspecto caracterizam as informações que nos diagramas HIPO "input-process-output" detalham a seção PROCESS.

Estrutura - neste aspecto são descritas as estruturas dos dados e dos processos do sistema. Consta das relações que descrevem como os processos e passos se subordinam um ao outro e de que dados os grupos e itens consistem. As relações desse aspecto constituem as informações dos diagramas tipo "hierarchy" e as tabelas de dados do HIPO.

Tipo-de-dados - neste aspecto são descritos os dados, através de seus tipos. As informações deste aspecto correspondem às seções "input" e "output" nos diagramas HIPO.

Generalidades - neste aspecto são descritas informações gerais sobre o sistema sendo documentado, como por exemplo, nomes alternativos dos objetos e responsabilidade da do-

cumentação de cada processo. As informações deste aspecto correspondem aos cabeçalhos dos diagramas HIPO.

5.4 - Conclusões

Os dois exemplos apresentados mostram o uso do SACDS com sistemas de tipos bastante diferentes. Pode-se observar que as características preconizadas no capítulo 3 deste trabalho podem ser postas em prática, e que são úteis nas atividades com esses sistemas, sendo que o maior esforço para se conseguir o apoio do SACDS às atividades de documentação e análise com algum tipo de sistema, consiste em se definir a Linguagem de Descrição para esse tipo de Sistema. Em [Tr 82c] se encontra um algoritmo que orienta o usuário em como ele deve proceder para fazer a definição de uma linguagem dessas.

Os dois exemplos se limitam a mostrar como a descrição de Sistemas do tipo mostrado pode ser feita para o SACDS, sem apresentar contudo uma descrição de um sistema particular. O motivo de não ter sido incluídas neste trabalho tais descrições; é o volume de informações e textos que tais descrições apresentam. Um exemplo de uma descrição completa de um sistema, usando a Linguagem HIPOP, pode ser encontrado em [Tr 82c].

CAPITULO 6

LINHAS DE FUTURAS PESQUISAS

6.1 - Introdução

Neste capítulo são apresentadas idéias e sugestões que podem dar prosseguimento ao trabalho, ora apresentado. O SACDS é em realidade constituído de 3 subsistemas, cuja implementação está apenas iniciada, e portanto grande parte do trabalho com ele deve incluir a continuação dessa implementação. Por outro lado, por ser um sistema que se destina a ser usado como apoio a uma grande variedade de tipos de sistema, através do preparo de uma linguagem de definição do sistema, muito trabalho deve também ser dispendido nas técnicas de preparo dessas linguagens. Além disso, o SACDS é bastante maleável para aceitar a incorporação de recursos e técnicas que se julguem desejáveis, e por isso abre campo também para a pesquisa de características que o expandam.

Essas 3 áreas de trabalho são discutidas a seguir, porém apesar dessa divisão, deve-se lembrar que na realidade qualquer futura pesquisa feita para o SACDS irá necessariamente envolver essas três áreas. A divisão é apenas para ressaltar o enfoque principal de determinada linha de pesquisa, ou onde ela trará mais benefícios.

6.2 - Implementação do SACDS

O SACDS é composto essencialmente de 3 programas, cada um chamado um subsistema [Tr 82]. Existem dessa forma o Subsistema de Definição de Linguagem (SDL), o Subsistema de Manipulação da Descrição (SMD) e o Subsistema de Utilitários (SUT) Desses 3 subsistemas, o SDL é usado apenas para se preparar um Arquivo de Definição de Linguagem (ADL), que torna o SACDS apto a apoiar a tarefa de documentação e análise de um determinado tipo de sistema. Esse arquivo é usado pelo SMD e pelo SUT sempre que alguma tarefa com sistemas do tipo definido pelo arquivo tiver que ser realizada. Como todas as tarefas realizadas por esses 2 subsistemas dependem diretamente de especificações contidas no ADL, em termos de implementação do SACDS, qualquer alteração no SMD ou no SUT deve ser precedida de uma alteração no SDL, para tornar as alterações coerentes com o que se descreve no ADL. Assim, o SDL sempre deve ser atualizado para que os arquivos ADL possam conter as especificações, para os novos recursos que os demais subsistemas vierem a ter.

O "coração" do SACDS é em realidade o SMD, pois uma associação SMD-ADL é a ferramenta que permite ao usuário o tratamento de seus sistemas do tipo especificado naquele ADL. Por isso, estas sugestões de futuras expansões da implementação do SACDS se concentram principalmente no SMD, com as correspondentes expansões do SDL.

6.2.1 - Estruturas de Dados Usadas pelo SACDS

O SACDS usa principalmente 2 estruturas de dados de grande porte [Tr 82a], que são respectivamente a Base de Dados que armazena a descrição e o arquivo ADL. Além dessas 2 estru-

turas existem várias outras de menor porte, que se destinam a aspectos não gerais da descrição, mas que devem manter uma compatibilidade e uma estrutura homogênea em todo o SACDS. Exemplos de estruturas assim são: a armazenagem de pequenas porções de objetos e relações da descrição, usada por exemplo para armazenar os objetos escolhidos no módulo de Elaboração da Descrição do Sistema (EDS); e a armazenagem de Linguagens específicas para um módulo ou atividade do SACDS usada, por exemplo, para armazenar a linguagem de comando do EDS (LEDS) e a linguagem de comando do módulo SO (LSO).

Todas as estruturas de dados já existentes, bem como qualquer nova estrutura que venha a ser necessária devem ser feitas de forma a constituir estruturas adequadas ao fim a que se destinam. Como a aplicação a que o SACDS se destina nem sempre é clara, o preparo de uma estrutura que será (ou está) incorporada deve ser feita com especial cuidado. Estruturas já incorporadas que poderiam ser analisadas, com o objetivo de verificar sua adequação ou de possíveis melhoramentos são a estrutura do ADL e a estrutura que armazena linguagens específicas (comum a todos os módulos do SACDS que disponham de linguagens de comando). Um estudo de armazenagem dos relacionamentos simples de uma descrição em uma base de dados foi feito em [Tr 78].

6.2.2 - Armazenagem de Seleções de Objetos

A Base de Dados que armazena 1 descrição de sistema no SACDS é composta de 4 arquivos que armazenam a descrição propriamente dita e mais um número variável de arquivos para arma

zenar as seleções de objetos feitas. Com os recursos do que o SACDS dispõe atualmente, todos os módulos que manipulam seleções de objetos o fazem explicitamente, através de Arquivos de Seleção (SEL).

A estrutura do SACDS é tal que pode existir qualquer número de objetos, relações e comentários em uma descrição, limitado apenas pelas características do computador onde estiver implantado. Isso significa que seleções de objetos podem conter qualquer quantidade de objetos. Os módulos já implantados se utilizam de arquivos de seleção, porém essa utilização não será eficiente quando o módulo usar várias seleções simultaneamente. Novos recursos que deverão ser implementados no SACDS (por exemplo, geração de relatório) irão apresentar essas características. Portanto será necessária uma estrutura de dados capaz de armazenar com eficiência, simultaneamente um número variável de seleções, cada seleção contendo qualquer quantidade de objetos, e além disso, sem que haja a obrigatoriedade de que um número predeterminado de seleções já estejam completas (pode-se estar criando tantas seleções simultaneamente quanto se queira). O módulo para a seleção de objetos (SO) já implementado dispõe de uma estrutura com esse objetivo, bastante rudimentar [Hl 81], pois limita o número de seleções existentes simultaneamente, bem como tem suporte para que apenas uma seleção possa estar sendo criada de cada vez. Porém, pode ser usada como base para o desenvolvimento de uma estrutura com as características desejáveis. Técnicas de paginação de memória, memória virtual e estruturas de listas poderão ser usadas para esse objetivo.

6.2.3 - Atividades Automáticas do SACDS

O objetivo de um instrumento como o SACDS é obter informações dos seus usuários, interrelacioná-las, e a seguir mostrar em formato desejável essas mesmas informações, apenas reorganizadas para mostrar tudo o que se refere a determinado assunto. As tarefas de obter informações e de mostrá-las nos relatórios solicitados são executadas sempre com controle direto do usuário. Porém, é bastante desejável que algumas tarefas sejam executadas automaticamente, mesmo que não sejam solicitadas pelo usuário. Exemplos dessas tarefas são a verificação de condições de consistência e completeza quando algumas situações ocorrem (por exemplo no fim de uma operação de edição da descrição) e a busca de relacionamentos implícitos em uma descrição. Tais tarefas devem ser definidas de forma diferente para cada tipo de sistema com que se deva tratar, e por isso devem ser definidas através do SDL, porém é o SMD que as executa. Este deve portanto, em pontos chaves verificar se existe a definição (ou solicitação declarada no ADL) de alguma atividade que deve ser executada naquele ponto.

Para isso, é necessário um estudo em diversos tipos de sistemas para conhecer quais atividades automáticas trariam benefício ao tratamento desses sistemas. No capítulo 3 foi citada a execução de conjuntos de testes de consistência e completeza, mas pode não ser essa a única atividade automática importante. Além disso, é necessário que se verifiquem todas as situações onde tais atividades deveriam ser automaticamente acionadas.

Cada atividade automática será executada por um mó-

dulo específico, provavelmente acessível também através de solicitação explícita do usuário. Para que tais atividades automáticas sejam possíveis, deve-se manter sempre a estrutura do SACDS como uma coleção de módulos que se comunicam somente através das estruturas de dados principais, e não exista subordinação de um módulo a outro, ou sinais de controle entre os vários módulos.

6.2.4 - Atividades Paralelas

As atividades que o SACDS executa podem, na maioria dos casos, ser divididas em várias tarefas interdependentes que, juntas, constituem aquela atividade, ou podem conter várias tarefas que complementem a atividade principal. Por exemplo, uma atividade de atualização da descrição é constituída da tarefa de:

a) analisar o comando de atualização emitido pelo usuário,
b) procurar na base de dados as informações relacionadas no comando,
c) verificar a validade da nova alteração em confronto com os dados já definidos,
d) alterar a base de dados,
e) indicar em arquivo de seleção correspondente quais objetos foram criados, atualizados, etc.

Quando executada uma atividade qualquer, o SACDS executa as tarefas que a compõem em sequência, pois esse módulo faz parte de um único programa. Porém, pelo exemplo citado é possível perceber que várias tarefas poderiam ser executadas

em paralelo. Neste exemplo, as atividades dos itens d) e e) poderiam ser executadas em paralelo, e em paralelo com a atividade do item a) do comando seguinte. Em termos de implementação do SACDS, poder-se-ia estudar a implementação de atividades paralelas, que trariam em tempo de execução, manutenção da independência entre seus módulos e auxílio às atividades automáticas. Para este último item, as atividades paralelas seriam de especial importância por ajudar a não prejudicar em demasia o tempo de respostas aos comandos explícitos do usuário.

6.2.5 - Utilitários

O subsistema de utilitários (SUT) tem por objetivo manipular descrições de sistemas como um todo, através da base de dados que as armazena. Atividades já implementadas são:

a) redimensionar uma base,

b) copiar descrições,

c) compactar descrições.

Várias outras atividades poderiam ser úteis se implantadas, e especialmente duas delas já se fizeram sentir:

a) recuperar uma descrição,

b) substituir uma linguagem.

Quanto a recuperar uma descrição, é uma atividade necessária para permitir a restauração de uma descrição que estava em uso, e antes que pudesse ser gravada em forma integral nos arquivos que constituem a base de dados, o sistema tenha tido sua execução interrompida. O utilitário deve então analisar a base de dados, e recuperar todas as informações possíveis, recriando uma base de dados íntegra, com o máximo de in-

formações que for possível recuperar, e dando indicações, sempre que possível , sobre os dados perdidos.

Quanto a atividade de substituir a linguagem usada em uma descrição, ela se deve ao fato de que quando uma descrição é iniciada com uma determinada linguagem, tal linguagem não pode mais ser alterada. Caso se faça uma alteração nela, incluindo novos tipos de objeto ou tipos de relações, a descrição feita antes da inclusão continua logicamente compatível com a nova linguagem, porém a estrutura do arquivo ADL se modifica de tal forma que incompatibiliza a nova linguagem com a descrição antiga. Poder-se-ia incluir no SUT uma opção que permitisse a especificação de uma descrição, uma linguagem antiga, e uma linguagem nova, e que gerasse uma nova base de dados com as mesmas informações da descrição fornecida compatível com a nova linguagem. Tal utilitário deveria verificar se a nova linguagem é totalmente compatível (uma extensão, sem nenhuma omissão) com a linguagem antiga, antes de iniciar a conversão.

6.3 - Técnicas para aplicação do SACDS

O SACDS é um instrumento de apoio à documentação e análise de Sistemas, mas para que possa ser usado no auxílio dessas atividades em um tipo particular de sistema, deve antes ser preparado um arquivo que descreve as características desse tipo de sistema (arquivo ADL), contendo a linguagem de descrição, especificações de relatórios e outras informações. O preparo de um arquivo ADL é realizado através do SDL, e constitui a tarefa que requer o maior grau de envolvimento com o SACDS, por parte do usuário. Para desempenhar essa tarefa o usuário de

ve ter um bom conhecimento do tipo de sistemas que irá descrever, do próprio SACDS, e principalmente de técnicas que o guie em como obter e estruturar as informações necessárias, bem de como reconhecer quais características do SACDS devem ser exploradas para se conseguir implantar neste (através do ADL) os recursos desejados, e assim poder aplicá-lo no apoio à documentação e análise de sistemas daquele tipo.

6.3.1 - O Preparo de Linguagens de Descrição de Sistemas

A primeira informação que deve ser incluída em um arquivo ADL é a Linguagem de Descrição de Sistemas (LDS). Como descrito nos capítulos 4 e 5 desta dissertação, essa linguagem é constituída dos conjuntos de tipos de relação e tipos de objeto, da especificação de características de cada tipo de relação e da sintaxe da linguagem. O preparo da LDS é o início da tarefa de preparo do arquivo ADL pois os 2 conjuntos de tipos definidos constituem-se em palavras-chave que serão usadas em todas as demais especificações. O preparo da LDS consiste em definir-se as informações citadas. Em [Tr 82c] encontra-se um algoritmo genérico que permite a uma pessoa, com bom conhecimento de uso do SACDS e de sistemas do tipo para o qual se deve preparar o arquivo ADL, obter e estruturar as informações citadas. No capítulo 5 desta dissertação são apresentadas as LDS de 2 tipos de sistemas, porém os exemplos citados são bastante simples, pela sua própria característica de exemplos. Outras LDS devem ser criadas, notadamente para as metodologias de análise e especificação de Sistemas de Informação mais divulgadas, para permitir ao SACDS apoiá-las.

O preparo de uma LDS deve ser feito com bastante cuidado, e deve ser testado, e uma boa prática pode ser descrever com o SACDS, usando o ADL criado, o problema exemplo da própria metodologia, avaliando-lhe os recursos, vantagens e desvantagens.

6.3.2 - Identificação dos Recursos a serem utilizados

Por ser um instrumento que tem por objetivo ser genérico para poder ser aplicado a uma grande gama de tipos de sistema, o SACDS apresenta variados recursos que frequentemente podem deixar de ser compreendidos ou sua aplicação em casos particulares deixa de ser visível. Um exemplo pode ser o seguinte:

Frequentemente ocorrem na literatura referente a assuntos semelhantes aos tratados nesta dissertação, referências ao fato de que uma linguagem com a estrutura daquelas que se pode definir com o SDL deveriam ser compostas de objetos, relações entre eles e atributos associados aos objetos e/ou às relações [Wi 79]. As linguagens LDS que se pode criar com o SDL não contêm referência explícita a nada que seja um atributo associável, nem a objeto, nem a relação. Porém em realidade um atributo aparece na linguagem da mesma forma que apareceria um objeto. Portanto um atributo pode ser apenas um objeto de tipo "ATRIBUTO". A diferença real que existe entre um objeto, e um atributo é que em sistemas de um determinado tipo o objeto diz respeito apenas ao sistema particular do qual faz parte integrante, enquanto que o atributo é um objeto que se poderia dizer ser comum a todos os sistemas daquele tipo. Um exemplo po-

de ser obtido da linguagem HIPOP descrita no capítulo 5 deste trabalho, e que pode ser usada com Sistemas de Informação. Nessa linguagem existe o tipo de objeto ATRIBUTO que se aplica a objetos do tipo GRUPO ou ITEM. Objetos do tipo GRUPO ou ITEM são objetos que fazem parte exclusivamente de cada um dos sistemas de informação que se descreve. Assim um objeto tipo GRUPO chamado INFORMAÇÃO-DE-EMPREGADO pode fazer parte de um sistema de folha de pagamento, mas não de qualquer sistema de informação documentável com o HIPO. Porém um atributo que a metodologia HIPO pode aplicar ao "grupo de dados" "informação-de-empregado" é que ele seja um "arquivo em cartão", para o qual existe uma notação específica.

Dessa forma "arquivo em cartão" é um atributo, comum a qualquer sistema de informação no qual se aplica a metodologia HIPO, e é um atributo associado, neste caso específico ao "grupo de dados" "informação dos empregados". Na linguagem HIPOP seria descrito que ARQUIVO-EM-CARTÃO é um objeto de tipo ATRIBUTO, que APLICA-SE ao objeto tipo GRUPO INFORMAÇÃO-DE-EMPREGADO. Percebe-se assim que nas Linguagens LDS que se pode criar, um atributo não é usado sintaticamente de forma diferente da forma com que se usa um objeto. A diferença é apenas no significado que é dado a cada caso particular e não na sintaxe. Por isso, a distinção entre o que é objeto e o que é atributo em um arquivo ADL deve ser feita quando se especifica como os objetos devem ser usados nas descrições, ou seja, na definição de testes de consistência da descrição. Um teste que se deve poder definir é que, todos os objetos usados como atributos estejam entre os atributos válidos para sistemas daquele tipo.

Pode-se então notar por exemplo, que muitos recur-

sos podem ser postos em um arquivo ADL que exploram os recursos do SACDS mesmo que a interligação não seja aparente à primeira análise. Um estudo mostrando as características de sistemas que em geral podem ser associadas a recursos específicos do SACDS, compilando as interligações mais frequentes deve ser levado à efeito, bem como uma descrição de métodos que permitam a um usuário visualizar a identificação de interligações em sistemas do tipo que está interessado em particular.

6.3.3 - Identificação de Relatórios e Condições de Consistência e Completeza.

O subsistema SMD é constituído genericamente de 4 módulos que são:

1 - Módulo de Elaboração da Descrição de Sistemas (EDS)

2 - Módulo de Seleção de Objetos (SO)

3 - Módulo de Verificação de Consistência e Completeza

4 - Módulo de Geração de Relatórios

Para que cada um dos Módulos possa ser usado, é necessário que exista no arquivo ADL as informações que cada módulo deve dispor sobre os sistemas do tipo com o qual estiver sendo usado. Uma informação que é usada por todos os módulos é a linguagem LDS, tratada na seção 6.3.1. Os módulos EDS e SO necessitam além disso da linguagem específica de comando de cada módulo (LEDS e LSO), que são bastante independentes das variações que existem entre vários tipos de sistemas. Os módulos de geração de relatórios e verificação de testes de consis

tência e completeza também necessitam de linguagens de comando próprias e independentes, porém devem usar informações específicas de cada tipo de sistema, que são a definição de como os relatórios devem ser gerados e a definição de quais testes e como cada teste é efetuado. Da mesma forma que para o preparo de uma LDS, seria conveniente a preparação de um documento, além dos manuais de uso dos próprios módulos e de como as informações são fornecidas ao SDL, que contivesse um algoritmo capaz de orientar o usuário na coleta de dados e na estruturação desses dados, de forma a conseguir criar as especificações corretas que levam à definição dos relatórios e testes desejados.

6.3.4 - Apoio a Outros Sistemas que não de Informação

O SACDS originou-se de trabalhos e idéias oriundas da convivência com Sistemas de Informação, e por isso, grande parte de sua estrutura é inspirada em trabalhos desenvolvidos com a descrição e implementação de Sistemas de Informação. O objetivo da implementação do SACDS é no entanto, que ele possa dar apoio à documentação (descrição) de outros tipos de Sistemas, além dos de informação. Algumas considerações devem, no entanto, ser levantadas quanto a esse enfoque, em decorrência do fato de que a Especificação de Sistemas de Informação ocupam uma posição muito especial entre os tipos de sistemas.

Considerando Sistema um conjunto de Elementos que interagem pode-se dizer que parte dos elementos que formam qualquer sistema são elementos de informação. Pode ser de interesse analisar algumas delas com o objetivo de, por algum motivo, interfirir no sistema. Para isso pode-se definir um Sistema de

Informação associado ao sistema em apreço, que pode ser especificado e a seguir implementado. Para o objetivo desta discussão a implementação de um Sistema de Informação não difere de Sistemas de outros tipos, aos quais se designa genericamente como sistemas reais. A Especificação de Sistemas de Informação, ou seja, a descrição de como um Sistema de Informação deve ser sem levar em conta como ele pode ser implementado constitui o núcleo de interesse do SACDS. A respeito de um determinado sistema, podemos descrever o próprio sistema ou descrever em sistema de informação associado a ele. Por exemplo, se for tomada a tarefa "gerar uma folha de pagamento", tal tarefa caracteriza um tipo de sistema real: Sistema de geração de folha de pagamento. Associado a esse tipo de sistema, pode ser definido um sistema de informação do sistema real de geração de folha de pagamento. Ao se descrever o sistema de informação, pode-se dizer, por exemplo, que o objeto tipo ENTRADA CARTÃO-DE-PONTO faz parte do objeto tipo ARQUIVO DADOS-DE-EMPREGADO, ao passo que ao se descrever um sistema de geração de folha de pagamento real ter-se-ia que descrever informações como: CARTÃO-DE-PONTO-Nº-1234 faz parte de DADOS-DO-EMPREGADO-FULANO-DE-TAL.

Partindo dessas considerações, pode-se verificar que o SACDS tem como características próprias aquelas que são destinadas à descrição de Sistemas de Informação. A descrição de um Sistema qualquer é normalmente realizada com um Sistema de Base de Dados, e se o objetivo do SACDS fosse apenas o de ser útil à descrição de sistemas de qualquer tipo, ele seria apenas um sistema de gestão de Base de Dados. Ressaltando o fato de que seu objetivo é o de aceitar a descrição de Sistemas de Informação, o SACDS passa a ser um instrumento que incorpora,

entre outros recursos, um Sistema de Gestão de Bases de Dados.

Para que desenvolvimentos ulteriores do SACDS possam manter os mesmos objetivos iniciais, deve-se levar em conta que se trata de um sistema para permitir a descrição de sistemas de qualquer tipo, e especialmente sistemas de informação. Sendo um sistema que permite a descrição de Sistemas de Informação com base em qualquer metodologia, e é essa a idéia básica que norteou o trabalho aqui apresentado.

6.3.5 - Apoio a Sistemas de Engenharia

Os sistemas de base de dados existentes são desenvolvidos de forma a manipular um volume de dados muito grande com estruturas relativamente pouco complexas. Os dados que existem nas atividades de engenharia, ao contrário, tem como características principais a existência de muitas estruturas de dados com comparativamente poucos dados em cada estrutura. Dessa forma a descrição de um sistema de engenharia assemelha-se bastante à descrição de Sistemas de Informação. Portanto o uso do SACDS para apoio à atividades com sistemas de engenharia é semelhante ao uso do SACDS para apoio à atividades com sistemas de informação, para a qual o SACDS foi desenvolvido. Por isso o SACDS é um instrumento útil para descrição e análise de sistemas dos tipos usados em engenharia, desde que seja preparado um arquivo ADL adequado.

Os exemplos e considerações feitas neste trabalho sempre mostraram a aplicação do SACDS a Sistemas de Informação. Aplicá-lo a Sistemas de Engenharia é possível através de procedimentos análogos, porém por se destinar a um tipo de usuário diferente, seria útil desenvolver-se procedimentos e documenta

ção destinados a esses usuários.

Um exemplo de sistemas de engenharia que poderia beneficiar-se do SACDS, é a área de projeto de Sistemas de Hardware. O uso do SACDS poderia fornecer ao projetista recursos de descrição de seu projeto, e possibilidade de obter informações quanto a validade de situações específicas, compatibilidade entre módulos descritos, e relatórios mostrando resultados gerais do sistema.

6.4 - Desenvolvimento de Novos Recursos

No estágio em que se encontra a implementação do SACDS, estão em operação os módulos de Elaboração da Descrição do Sistema (EDS) e de seleção de objetos (SO) do subsistema SMD, os módulos de definição das Linguagens LDS, LEDS e LSO do subsistema SDL e alguns utilitários do subsistema SUT, tal como é descrito no capítulo 4 deste trabalho. O SACDS completo, tal como descrito no capítulo 3, deverá incluir novos módulos de maneira a possibilitar a incorporação de outros recursos, em especial módulos que permitam a geração de relatórios, a verificação de testes de consistência e completeza. O que tais recursos devem possibilitar ao SACDS, bem como o objetivo de sua disponibilidade são discutidos no capítulo 3, porém resta ainda necessidade de se efetuar a implementação deles, e para isso estudos que definam as estruturas de dados, algoritmos e técnicas que a viabilize. Será feita a seguir uma sugestão de quais passos podem ser seguidos para que tais recursos possam vir a ser incorporados efetivamente ao SACDS.

6.4.1 - A Verificação de Testes de Consistência e Completeza

O módulo de verificação de testes de consistência e completeza (VCC) poderá ser ativado tanto por solicitação explícita do usuário quanto através de especificação definida no arquivo ADL para quando ocorrer uma situação específica. Por esse motivo, o módulo poderá ser ativado juntamente com outros módulos, e portanto as estruturas de dados que são necessárias para este módulo devem estar sempre disponíveis, bem como devem permitir que as estruturas de qualquer outro módulo coexistam com elas. Isso significa que as rotinas básicas de manipulação dessas estruturas estarão também sempre disponíveis. Manter as estruturas de dados e as rotinas sempre disponíveis significa ter uma parte da memória gasta pelo SACDS permanentemente usada para essa atividade, o que obriga a que esse consumo de memória seja o menor possível, e portanto as estruturas de dados devem ser compactas e de uso geral, para que as rotinas de tratamento correspondentes não sejam volumosas. A generalidade resultante é conveniente também para manter a generalidade das aplicações do SACDS.

A seguir, deve-se levar em conta que, por ser as vezes ativada sem que haja solicitação explícita por parte do usuário, o módulo VCC deve ser rápido, para não aumentar demais o tempo gasto.

O módulo VCC frequentemente usa parte considerável da descrição para efetuar um teste. Isso significa que as estruturas de dados envolvidas não podem ser dimensionadas para um número limitado de dados, mas deve ser capaz de manipular dados correspondentes à base inteira. Isso é possível mantendo

tais estruturas em arquivos de disco. No entanto, o uso de disco aumenta o tempo necessário a cada operação, sendo portanto, necessário obter um compromisso entre a velocidade de acesso e as características necessárias da estrutura de dados. A obtenção de algoritmos adequados é um dos mais importantes aspectos a ser desenvolvido para que esse recurso seja incorporado ao SACDS.

Para que um teste seja verificado pelo SMD, é necessário que o teste tenha sido definido no arquivo ADL, através de especificações fornecidas ao SDL pelo usuário, e isso somente é possível incorporando ao SDL um módulo de definição de testes de consistência e completeza (DCC). Tal módulo deve aceitar a definição dos testes através de uma linguagem de definição de testes, que deve ser ainda estudada e especificada. Porém deve ser feita procurando manter-se a mesma estrutura das linguagens de comando já existentes no SACDS (tais como a LSO e a LEDS).

6.4.2 - Geração de Relatórios

Da mesma forma que a verificação dos testes de consistência e completeza, os módulos de geração de relatórios também podem necessitar de pesquisa extensiva na base de dados, e portanto as mesmas considerações feitas na seção anterior a esse respeito continuam válidas, principalmente para os módulos de geração de relatórios específicos e geração de relatórios de navegação. O mesmo se pode dizer também a respeito da linguagem de descrição de relatórios. Porém, a respeito das estruturas de dados envolvidas, deve-se levar em especial consideração a forma como os vários módulos de formato contribuem com dados pa

ra a formação de uma página de impressão. A natureza desses módulos de formato, e a forma como eles se interagem leva a necessidade de estruturas de dados que possam ser usadas como se os módulos pudessem operar através de processamento paralelo.

No capítulo 3 deste trabalho é sugerida uma forma de interligação entre os módulos que especificam a forma como os relatórios podem ser definidos, e portanto indica como a linguagem de definição de relatórios pode ser, e como ela pode ser armazenada no arquivo ADL. Uma estrutura semelhante deve existir nos próprios módulos de geração de relatórios, porém capaz de armazenar as informações específicas que circulam entre os próprios módulos de formato. Tal estrutura deve ter como suporte algoritmos de tratamento e armazenagem que devem ser desenvolvidos com o intuito de manter a flexibilidade descrita no capítulo citado, e além disso mantendo sempre a capacidade de permitir que novos módulos de formato sejam progressivamente incluídos.

6.4.3 - Simulação de Sistemas Descritos

Quando o tipo de sistema descrito não é de Informação, é bastante comum que se queira obter do computador uma simulação de tal sistema. Conforme é descrito na seção 6.3.4, o objetivo do SACDS deve ser o de permitir a documentação e análise de qualquer tipo de sistema com ênfase nos sistemas de informação. Um sistema real pode ser simulado, mas não a sua especificação. Portanto a simulação da especificação de um Sistema de Informação encontrará restrições para ser feita plenamente, embora alguns aspectos típicos de simulação de grande uti-

lidade possam ser obtidos. A capacidade de simulação plena, se incorporada ao SACDS, teria por objetivo ser usada com Sistemas que não são de Informação. Como esse não é o objetivo principal do SACDS, aspectos de simulação possíveis de serem obtidos deveriam ser estudados enquanto fossem úteis para o apoio à Sistemas de Informação. Uma capacidade (incipiente) está incluida como resultado das análises efetuadas pelo módulo de geração de relatórios de navegação. Uma extensão a esse módulo com vistas à simulação pode ser estudada, porém sempre mantendo a filosofia básica do SACDS de que a especificação (neste caso, de como a simulação deve ser feita) deve ser feita através do SDL, e que apenas a solicitação da simulação e a especificação de parâmetros particulares de cada simulação devem ser feitas pelo usuário, para o módulo de geração de relatórios de navegação.

A P Ê N D I C E

ABREVIATURAS USADAS NESTE TRABALHO

ADS = Accurately Defined Systems

BIAIT = Bussiness Information Analyses and Integration Technique

BICS = Bussiness Information Control Study methodology

BSP = Bussiness Systems Planning

EPOS = Engeneer and Process - Oriented development Support System

ESPRESO-S = System zur Erstellung der Spezifikation von Prozessrech ner-software-sprache (language)

ESPRESO-W = System zur Erstellung der Spezifikation von Prozessrech ner-Software-werkzeug (tool)

HIPO = Hierarchy plus Input-Process-Output

ISAC = Information Systems for Administrative Control

ISDOS = Information System Design and Optimization System

ISLDS = Information System Language Definition System

PCSL = Process Control Software Specification Language

PSL = Problem Statement Language

PSL/PSA = Problem Statement Language / Problem Statement Analyser

REVS = Requirements Evaluator and Validation System

RSL = Requirements Specification Language

SADT = Structured Analysis and Design Technique

SDLA = System Descriptor and Software Analyser

SDS = Software Development System

SEM = System Encyclopedia Manager

SEMS = System Encyclopedia Management System

SREM = Software Requirements Engeneering Methodology

SREP = Software Requirements Engeneering Program

ABREVIATURAS USADAS NO SACDS

ADL = Arquivo de Definição de Linguagem
EDS = Módulo de Elaboração da Descrição de Sistemas
GR = Módulo de Geração de Relatórios
LDS = Linguagem de Descrição de Sistemas
LEDS = Linguagem de Elaboração da Descrição de Sistemas
LSELEC = Linguagem de Seleção de Objetos
LSO = Linguagem de Seleção de Objetos
MSG = Arquivo de Mensagens
PDL = Programa para a Definição da Linguagem
PPM = Programa de Preparo das Mensagens
SACDS = Sistema de Apoio por Computador à Documentação de Sistemas
SDL = Subsistema de Definição de Linguagem
SEL = Arquivo de Seleção de Objetos
SMD = Subsistema de Manipulação da Descrição
SO = Módulo de Seleção de Objetos
SUT = Subsistema de Utilitários

B I B L I O G R A F I A

[Al 77] - Alford, M. - "A Requirements Engineering Methodology for Real-Time Processing Requirements" - IEEE Transaction on Software Engineering, Vol 3 nº 1, 1977.

[Be 77] - Bell, T.E.; Bixler, D.C.; Dyer, M.E. - "An Extendable Approach to Computer-Aided Software Requirements Engineering" - IEEE Transaction on Software Engineering Vol 3 nº 1, 1977.

[Bg 73] - Berge, C. - "Graph and Hipergrahs" - North-Holland Publising Company, 1973.

[Bo 79a] - Boehm, B.W. - "Software Engineering as it is" in Proceedings of 4th International Conference on Software Engineering IEEE Catalog 79CH1479-5C,1979.

[Bo 79b] - Boehm, B.W. - "Software Engineering: R&D Trends and Needs" in Research Direction in Software Technology - Peter Wegner (Ed) The Mit Press, 1979.

[Bl 82] - Blum, B. - "A Tool for Developing Information Systems" - in Automated Tools for Information Systems Design - Schneider H.J. & Wasserman, A.I. (Eds) - North-Holland, 1982.

[Bu 79] - Burnstine, D.C. - "The Theory Benhind BIATI-Bussiness Information Analysis and Integration Technique" - BIAIT International, Inc.; Fox Hollow, Petersburg, NY, 1979.

[Ca 79] - Carlson, W.M. - "Business Information Analysis and Integration Technique (BIAIT) - The New Horizon" - DATA BASE 10 nº 4, Spring 79.

[CO 74] - Couger, J.D. - "System Analysis Techniques"-New York, John Wiley & Sons, 1974.

[Da 82] - Davis, A.M. - "The Design of a Family of Application Oriented Requirements Languages" - IEEE Computer, vol. 15 nº 5, may 1982.

[De 82] - Demetrovics, J.; Knuth, E.; Radó, P.-"Specification Meta Systems" - IEEE Computer, Vol. 15 nº 5, may 1982.

[Di 80] - Distaso, J.R. - "Software Management - A Survey of the Practice in 1980" - Proceedings of the IEEE, Vol. 68 nº 9, September 1980.

[Dl 82] - Delisle, N.M.; Menicosy, D.E.; Kert, N.L.-"Tools for Supporting Structured Analysis" - in Automated Tools for Information System Design - H.J. Schneider and A.I. Wasserman (Eds.) - North-Holland Publishing Co. - IFIP, 1982.

[Dv 82] - Davis, G.B. - "Strategies for Information Requirements Determination" - IBM Systems Journal, Vol. 21 nº 1, 1982.

[Ge 76] - Germano, F.S.R. - "Modelos Matemáticos da Teoria dos Sistemas de Informação" - Tese de Doutorado apresentada ao ICMSC-USP - São Carlos, SP, 1976.

[Ge 78] - Germano, F.S.R.; Sahão Jr., J.; Traina Jr.,C. - "Evolução dos Recursos para a Descrição da Dinâmica de Sistemas de Informação em PSL" - Comunicação apresentada na 30ª Reunião Anual da SBPC, São Paulo. Julho de 1978.

[Gn 77] - Gane,C.; Carson, T. - "Structured Systems Analysis: tools and Techniques"-Improved Systems Technologies Inc., 1977.

[Ha 80] - Hatvany, J.; Jãnos, J. - "Software Products for Manufacturing. Design and Control" - Proceedings of the IEEE, Vol. 68 nº 9, September, 1980.

[He 76] - Henschen, L.J. - "Tutorial on Resolution" - IEEE - Transactions on Computer - Vol. C-25 nº 8 - August 1976.

[Hl 81] - Hehl, M.E.; Sahão Jr.,J.; Lauand, M. - "Seleção de Objetos na Descrição de Sistemas de Informação" - In Anales Panel'81 Expodata/12 JAIIO, Buenos Aires, 1981.

[Hr 69] - Harary, F. - "Graph Theory" - Addison-Wesley, 1969.

[IBM 74] - HIPO - A Design Aid and Documentation Technique" - GC 20-1851-0, IBM Corporation, Technical Publications/Systems Dept. October, 1974.

[IBM 81] - "Bussiness Systems Planning-Information System Planning Guide, Aplication Manual", GE 20-0527, IBM Corporation, July, 1981.

[Ke 80] - Kerner, D.V. - "Introduction to Business Information Control Study Methodology (BICS)" - Symposium on the Economics of Information Processing, December 1980 - IBM Systems Research Institute.

[Kr 80] - Krasner, G. - "Machine Tongues VIII: The Design of a Smaltalk Music System" - Computer Music Journal, Vol. 4 nº4, Winter 1980.

[La 82] - Lauber, R.J. - "Development Support Systems" - IEEE - Computer, Vol. 15 nº 5, may 1982.

[Lb 79] - Lauber, R.; Biewald, J.; Gohner, P; and Schelling, H. "EPOS - A Specification and Design Technique for Computer Controlled Real-time Automation System" -

IEEE Catalog 79CH1479-5C, 1979.

[Le 80] - Lehman, M.M. - "Programs, Life Cycles, and Laws of Software Evolution" - Proceedings of the IEEE,Vol. 68 nº 9, September, 1980.

[Lu 81] - Lundeberg, M.; Goldkuhl, G.; Nilsson,A.-"Information Systems Development: a Systematic Approach" - Englewood Cliffs, Prentice Hall, 1981.

[Lu 82] - Lundberg, B. - "IMT - An Information Modelling Tool" - in Automated Tools for Information System Design - Schneider, H.J. e Wasserman, A.I. (Eds) -North-Holland, 1982.

[Lv 82] - Levene, A.A.; Mullery, G.P. - "An Investigation of Requirement Specification Languages: Theory and Practice" - IEEE Computer Vol. 15 nº 5, may 1982.

[Lw 82] - Ludewig, J. - "Computer-Aided Specification of Process Control Systems" - IEEE Computer, Vol. 15 nº 5, may, 1982.

[Ma 72] - Mayeda, W. - "Graph Theory" - Wiley - Interscience, John Wiley & Sons, Inc., 1972.

[Mf 79] - McGuffin, R.W.; Elliston, A.E.; Tranter, B.R.; & Westmacott, P.N. - "CADES - Software Engineering in Practice" - IEEE Catalog 79CH1479 - 5C, 1979.

[Mg 79] - McGowan, C.L. - "Software Management in Research Directions Software Technology" - Peter Wegener (Ed.) - The MIT Press - 1979.

[Mk 80] - Mekly, L.J.; Yau, S.S. - "Software Design Representation Using Abstract Process Networks" - IEEE Transactions on Software Engineering,Vol. 6 nº 5, Sept 1980.

[Mo 79] - Morrissey, J.; Wu, L.S.Y. - "Software Engineering-An Economic Perspective" - IEEE Catalog 79CH1479-5C, 1979.

[Mu 79] - Mullery, G.P. - "A Method for Controlled Requirement Expression" - IEEE Catalog 79CH1479-5C, 1979.

[Ne 82] - Newman, P.S. - "Towards and Integrated Development Environment" - IBM Systems Journal, vol. 21 nº 1, 1982.

[Ni 77a] - Nicoletti, M. - "Sobre Modelos Matemáticos do Projeto Lógico de Sistemas de Informação" - Dissertação de Mestrado apresentada ao ICMSC-USP, São Carlos, 1977.

[Ni 77b] - Nicoletti, M.; Linhares, O.; Germano, F.S.R. - "Representação do Modelo Matemático da Linguagem PSL do Sistema PSL/PSA em termos de Dígrados Categorizados com Rótulos" - Comunicação apresentada à 29ª Reunião Anual da SBPC, S.Paulo-SP, 1977.

[Sa 82] - Sakamoto, J.G.: Ball, F.W. - "Supporting Business Systems Planning Studies with the DB/DC Data Dictionary"- IBM Systems Journal, Vol. 21 nº 1, 1982.

[Sh 76] - Sahão Jr., J. - "Estudo de um Modelo para Descrição, Projeto e Análise da Dinâmica de Sistemas de Informação" - Dissertação de Mestrado apresentada ao ICMSC-USP, São Carlos, SP, 1976.

[Sh 78] - Sahão Jr., J. - Linhares, O.L.; Germano, F.S.R.;Traina Jr., C. - "Um Modelo para a Descrição, Projeto e Análise da Dinâmica de Sistemas de Informação"- Comunicação apresentada na 30ª Reunião Anual da SBPC, São Paulo, SP, julho 1978.

[St 74] - Stevens, W.; Meyers, G.J.; Constantine, L.L. - "Structured Design" - IBM System Journal, Vol. 13 nº 1, 1974.

[St 82] - Stevens, W.P. - "How Data Flow can Improve Application Development Productivity" - IBM Systems Journal Vol. 21 nº 2, 1982.

[Te 75] - Teichroew, D.; Germano, F.S.R.; Moraes, P.S. - "Computer-Aided, Structured Documentation of Information Processing Systems Requirements" - In Anais do VIII Congresso Nacional de Processamento de Dados, São Paulo, 1975.

[Te 77] - Teichroew, D.; Hershey III, E.A. - "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems" - IEEE Transaction on Software Engineering, Vol. 3 nº 1, 1977.

[Te 79] - Teichroew, D.; Macasovic, P.; Hershey III, E.A.; Yamamoto, Y. - "Application of the Entity - Relationship Approach to Information Processing Systems Modeling" - Proc. Int./Conf. Entity-Relationship Approach to Systems Analysis and Design, 1979.

[Th 74] - Thall, R.M. - "A Manual for PSA/ADS: A Machine-aided Approach to Analysis of ADS" - ISDOS working Paper 107, June, 1974.

[Ti 82] - Tichy, W.F. - "A Data Model for Programming Support Enviroments and its Applications" in Automated Tools for Information Systems Design - Schneider, H.J. & Wasserman, A.I. (Eds) - North-Holland, 1982.

[Tr 76a] - Traina Jr., C. - "Manual de Uso do Sistema PSL/PSA

9830A" ISDOS Documento de Trabalho, São Carlos, 1976.

[Tr 76b] - Traina Jr., C. - "Manual de Referência do Sistema PSL/PSA 9830A" ISDOS Documento de Trabalho, São Carlos, 1976.

[Tr 76c] - Traina Jr., C. - "Manual de Uso do Teste de Consistência" ISDOS Documento de Trabalho, São Carlos, 1976.

[Tr 76d] - Traina Jr., C. - "Aplicação dos Testes de Consistência para Correção da Descrição de um Sistema de Pagamento de Pessoal" ISDOS Documento de Trabalho, São Carlos, Abril de 1976.

[Tr 76e] - Traina Jr., C. - "Verificação de Eficiência dos Testes de Consistência" ISDOS Documento de Trabalho, São Carlos, SP, Abril de 1976.

[Tr 76f] - Traina Jr., C. - "Manual de Programa para obtenção do Relatório de Fluxo Dinâmico" ISDOS Documento de Trabalho, São Carlos, SP, 1976.

[Tr 76g] - Traina Jr., C.; Sahão Jr., J. - "Um Exemplo de Aplicação de Programa Gerador de Fluxos Parciais" ISDOS Documento de Trabalho, São Carlos, SP, 1976.

[Tr 77] - Traina Jr., C. - "Estudo Comparativo entre possíveis representações para descrições feitas na Linguagem PSL" ISDOS Documento de Trabalho, São Carlos, Maio de 1977.

[Tr 78] - Traina Jr., C.; Germano, F.S.R.; Sahão Jr., J. - "Uso de Dígrafos para Esquematização de uma Base de Dados" - In Anais do 2º Congresso Brasileiro de Informática, Florianópolis-SC, Setembro de 1978.

[Tr 82a] - Traina Jr., C. - "SACDS - SMD - Manual de Referência" - SCE documento de trabalho, outubro de 1982.

[Tr 82b] - Traina Jr., C. - "SACDS - SMD - Manual de Uso" - SCE documento de trabalho, outubro de 1982.

[Tr 82c] - Traina Jr., C. - "A Preparação de uma linguagem de Descrição de Sistemas" - SCE Documento de Trabalho, novembro de 1982.

[Tr 82d] - Traina Jr., C.; Germano, F.S.R. - "Sistema de apoio por computador à Documentação de Sistemas" - In Anales del Novena Conferencia Latinoamericana de Informática, PANEL'82 - Lima - Peru, Agosto,1982.

[Wa 82] - Wasserman, A.I. - "Automated Tools in the Information System Development Enviroment" - in Automated Tools for Information Systems Design - Schneider, H.J. & Wasserman, A.I. (Eds) - North-Holland,1982.

[We 79] - Wegner, P. (ed) - "Research Directions in Software Technology" - MIT Press, Cambridge, Ma, 1979.

[Wh 82] - White, J.R. - "A Decision Tool for Assisting with the Comprehension of Large Software Systems" - in Automated Tools for Information System Design - Schneider, H.J. & Wasserman, A.I. (Eds) - North-Holland, 1982.

[Wi 79] - Willis, R.R.; Jensen, E.P. - "Computer Aided Design of Software Systems" - IEEE Catalog 79CH1479-5C , 1979.

[Wl 79] - Wilson, M.L. - "A Semantic-based Requirements and Design Method" - IBM Tr 03.072 - IBM Corporation, General Products Division, September 1979.

[Wl 80] - Wilson, M.L. - "The Measurement of Usability" - IBM Santa Teresa Laboratory, San Jose California, 1980.

[Za 82] - Zachman, J.A. - "Business Systems Planning and Business Information Control Study: A Comparison" - IBM Systems Journal, Vol. 21 nº 1, 1982.