Um Analisador de Especificações Operacionais

Maritza de las Mercedes Rodríguez Rodríguez

Orientação:

Prof. Dr. Paulo César Masiero

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos – USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP - São Carlos
Dezembro de 1989

*Aos meus pais René e Mercedes que
sempre me apoiaram, mesmo de longe.
Ao Mario pelo incentivo e amor que sempre me
ofereceu no transcurso de nossa vida juntos.
Aos meus filhos Maria Soledad e Mario Alejandro
pela felicidade de sua existência.*

Agradecimentos

Ao Professor Dr. Paulo César Masiero pela orientação, apoio e atenção que sempre me dispensou durante a elaboração deste trabalho.

Ao Professor Dr. Caetano Traina Junior pela colaboração no aprendizado das rotinas de manipulação da base de dados e interface utilizadas neste trabalho.

Às amigas Solange Rezende Rodrigues e Maria Inés Castiñeira pelos estudos realizados juntos, proporcionando uma sólida amizade que continuará, apesar da distância que nos separará.

A todos os professores, funcionarios e alunos do ICMSC pela receptividade com que sempre fui acolhida.

Resumo

Neste trabalho é apresentada uma análise do método JSD (Jackson System Development), como uma abordagem operacional para o desenvolvimento de sistemas. Faz-se também um breve resumo das características principais das Ferramentas de Software para Apoio por Computador ao Desenvolvimento de Software (CASE), visando dar uma idéia da evolução dessa área.

Baseando-se nesses conceitos é apresentada uma ferramenta do tipo CASE, para apoio à especificação de sistemas chamada Analisador de Especificações Operacionais (AnEspO). O AnEspO é parte do Ambiente de Desenvolvimento baseado na Abordagem Operacional - ADAO, que apoiará parte do ciclo de vida de desenvolvimento, de maneira a obter um protótipo executável do sistema alvo.

O AnEspO é uma ferramenta dedicada exclusivamente a apoiar a fase de especificação, de sistemas que são desenvolvidos usando as diretrizes dos quatro primeiros passos do método JSD. A especificação criada é armazenada em formato interno em uma base de dados, estando disponível para obtenção de relatórios auxiliares de consistência, validação e referências cruzadas, bem como para ser utilizada na geração de um protótipo do Sistema de Informação.

Abstract

The Jackson System Development Method (JSD) is analysed in this dissertation as an Operational Specification method for software development. It is also presented a summary of the Computer Aided Software Engineering field, specially tools to support the JSD Method.

A tool (called AnEspO) to support analysis and design of software based on the JSD method is discussed and a prototype developed to validate these ideas is shown. This prototype is part of a more comprehensive environment being developed which is known by the acronym ADAO (Software Development Environment based on Operational Approach).

The AnEspO Tool allows one to enter the especification of a target system, storing it in a data base where it can be later retrieved and displayed or printed. It can also be used by other tools being developed as for example a prototype generator.

Conteúdo

1	Introdução	1
1.1	Considerações Iniciais	1
1.2	Motivação do Trabalho	1
1.3	Contexto do Trabalho	5
1.4	Organização da Dissertação	8
2	Revisão Bibliográfica	10
2.1	Considerações Iniciais	10
2.2	Bibliografia Básica sobre o Método JSD	10
2.3	Ferramentas de Desenvolvimento Baseadas em JSD e outros Mé- todos Correlatos	16
2.3.1	Ferramentas de Apoio por Computador ao Desenvolvimen- to de Software	16
2.3.2	Ferramentas de Apoio a JSD	19
2.4	Reflexões sobre a Bibliografia Revisada	21
2.4.1	Sobre o Método JSD	21
2.4.2	Sobre as Ferramentas de CASE	22
2.5	Considerações Finais	25

3	Sobre o Projeto do AnEspO	26
3.1	Considerações Iniciais	26
3.2	O Modelo de Dados do Método JSD	27
3.2.1	Introdução ao Sistema SIPS	27
3.2.2	O Processo de Criação de uma Definição Meta para JSD .	30
3.3	Arquitetura do Sistema AnEspO	37
3.3.1	Tipos de Dados	38
3.3.2	Entidades - Ações - Atributos	40
3.3.3	Estruturação Cronológica	41
3.3.4	Modelo Inicial	43
3.3.5	Rede de Processos	45
3.3.6	Relatórios	48
3.4	Aspectos Principais da Especificação do AnEspO	48
3.5	Considerações Finais	51
4	Sobre o Protótipo Desenvolvido	53
4.1	Considerações Iniciais	53
4.2	Arquitetura do Protótipo	53
4.3	Uso do Protótipo	55
4.4	Um Caso	58
4.5	Considerações Finais	71
5	Conclusões Finais	72
5.1	Avaliação do AnEspO	72
5.2	Sugestões para a Evolução do AnEspO e outras Pesquisas	74

Bibliografia	75
A O Modelo de Dados JSD	79
A.1 Dicionário de Dados da Modelagem do Método JSD	80

Capítulo 1

Introdução

1.1 Considerações Iniciais

Este documento faz uma análise do método de Jackson para o desenvolvimento de sistemas (Jackson System Development, ou JSD, como será referenciado de agora em diante), visando ao desenvolvimento de uma ferramenta que permita realizar a especificação de um Sistema de Informação, nesse método, com o apoio de computador.

A ferramenta será chamada de Analisador de Especificações Operacionais (AnEspO) e se insere dentro de um projeto maior chamado Ambiente de Desenvolvimento baseado na Abordagem Operacional (Projeto ADAO), cujo objetivo principal é permitir, a partir da especificação operacional, criar automaticamente um protótipo do sistema alvo.

O objetivo deste documento é descrever os estudos e trabalhos publicados até o momento, sobre especificações operacionais, e no caso, sobre o método JSD, para em seguida apresentar o protótipo de um Analisador de Especificações Operacionais.

1.2 Motivação do Trabalho

A importância dos Sistemas de Informação vem crescendo bastante ao longo das últimas décadas. Os Sistemas de Informação estão sendo aplicados em mais e mais áreas dentro de nossas organizações e sociedade, exigindo técnicas

e ferramentas mais aperfeiçoadas para fazer face ao aumento de complexidade, qualidade e produtividade exigidos.

Quando de sua introdução, no início dos anos 50, como sistemas para processamento de dados científicos, estatísticos e militares, e depois como sistemas de processamento de dados comerciais, os sistemas de software começaram sendo produzidos de modo artesanal, isto é, com pouca disciplina formal ou sistematizada no processo de desenvolvimento.

Mais tarde, o alto custo da manutenção desses sistemas chegou a níveis alarmantes, tornando-se um dos fatores a desencadear a chamada "**Crise de Software**". Nasceu então a Engenharia de Software, com o objetivo de estabelecer e usar os princípios de engenharia para obter maior segurança e confiabilidade no desenvolvimento de sistemas, com um menor custo, como foi dito por Bauer na primeira reunião realizada para a discussão desse problema em 1969 [Pr87].

A Engenharia de Software introduziu inicialmente o paradigma do ciclo de vida como uma abordagem sistemática sequencial para o desenvolvimento de software. Este ciclo de vida, conhecido também como "**Modelo Cascata**", Figura 1.1, é formado basicamente por cinco etapas [McC88]:

- **PLANEJAMENTO DO SISTEMA:** reúne informações sobre os problemas do usuário, coloca os objetivos e critérios da empresa e gera soluções alternativas.
- **ANÁLISE:** determina as necessidades do usuário e as restrições do sistema, testa soluções alternativas para esses requisitos e restrições, e gera uma especificação e um modelo lógico para a melhor solução.
- **PROJETO:** elabora um projeto arquitetural onde são identificados os componentes do software, bem como a relação entre eles, sendo imprescindível uma documentação de todas as decisões do projeto.
- **IMPLEMENTAÇÃO:** o projeto é traduzido para uma linguagem executável, e posteriormente o software é testado e instalado.
- **MANUTENÇÃO:** esboça e implementa planos para, continuamente, ajustar, corrigir e melhorar o sistema em funcionamento.

À medida que as aplicações cresceram em complexidade e mais pessoas se envolveram no desenvolvimento e manutenção de Sistemas de In-

Figura 1.1: O CICLO DE VIDA CONVENCIONAL

formação, começaram a aparecer as denominadas “**Metodologias para Desenvolvimento de Sistemas**”, para fornecer uma abordagem organizada ao desenvolvimento. Em meados dos anos 70, surgiram as chamadas “**Metodologias Estruturadas**”. Alguns de seus principais criadores são: DeMarco [DeM78], Yourdon e Constantine [Yo78], Gane e Sarson [Ga76], Jackson [Ja75, Ja83], Martin [Ma86], Mellor e Ward [Me86]. Elas são corpos de regras, métodos e postulados usados para organizar todas as etapas do ciclo de vida de desenvolvimento de software, de tal maneira que guiam o analista de sistemas na tarefa de especificar, projetar e implementar Sistemas de Informação e software em geral.

As metodologias estruturadas usam técnicas de diagramação manuais e geralmente criam a especificação como um documento em papel, que deverá ser posteriormente transformado em uma linguagem executável pelo computador. Esse processo de transformação, embora muito importante até hoje, tende a gerar erros e é um dos responsáveis diretos pelo tempo demasiado gasto atualmente na implementação de Sistemas de Informação. Este problema no ciclo de vida convencional tem sofrido muitas críticas e foi considerado por McCracken e Jackson como um problema crônico desse modelo [Mc81]. Várias idéias novas para desenvolvimento começaram a surgir, mas essas idéias, tais como especificações executáveis e transformação de programas, são difíceis de serem desenvolvidas na abordagem convencional. Elas podem ser organizadas dentro de uma abordagem alternativa, que Zave chamou de “**Abordagem Operacional**” [Za84]. Essa abordagem vem crescendo em importância, dado o grande número de pesquisas atualmente se desenvolvendo com base nela, principalmente as relacionadas ao método JSD, a linguagem PAYSLEY, etc.

Para facilitar o uso e fortalecer essas abordagens, surgiram as ferramentas de apoio por computador ao desenvolvimento de software, como uma área designada pela sigla **CASE (Computer Aided Software Engineering)**. Elas fornecem facilidades gráficas automatizadas para a produção de diagramas, dicionários de dados, ferramentas de análise, geradores de documentação e geradores de código, entre outros.

O desenvolvimento de ferramentas de CASE tem aumentado consideravelmente nos últimos 20 anos [Da87]. Vários softwares desse tipo já estão disponíveis em escala comercial, dando apoio a metodologias mais conhecidas e difundidas, como por exemplo, a Análise Estruturada [Ga76, Tr86, Ma86, Ms88a, Ve86].

JSD é uma das metodologias para desenvolvimento de software que tem tido bastante atenção na literatura técnica especializada por suas características únicas [Ja83, Ja75, Ca86]. Em [Za84], por exemplo, Zave destaca

JSD como o método operacional em estágio mais avançado de formalização e utilização prática, pois JSD tem procedimentos e diretrizes claras para derivar a especificação operacional. JSD fundamenta-se na criação de um modelo da realidade, baseada em eventos do mundo real, isto é, o modelo simula a realidade, dentro do computador. A especificação criada por JSD, pode ser, em tese, diretamente executada pelo processador [Ca86].

Todos os aspectos mencionados acima motivaram a proposta deste trabalho, que com o desenvolvimento de uma ferramenta automatizada para a **Análise de Especificações Operacionais (AnEspO)** ajudará o engenheiro de software na tarefa de especificar um Sistema de Informação.

O AnEspO é baseado na abordagem operacional, em especial no método JSD, e cobre os quatro primeiros passos do método. A especificação criada é armazenada em formato interno em uma base de dados, estando disponível para obtenção de relatórios auxiliares de consistência, validação e referências cruzadas, bem como para ser utilizada na geração de um protótipo do Sistema de Informação.

Inicialmente se dará ênfase ao domínio de aplicações em Sistemas de Informação, mas, no futuro, o domínio de aplicações de Tempo Real também será incluído neste projeto. O AnEspO é independente do domínio de aplicações, mas as fases seguintes do ciclo de vida, como a geração do protótipo, são afetadas pelo domínio da aplicação.

1.3 Contexto do Trabalho

O Ambiente de Desenvolvimento baseado na Abordagem Operacional (ADAO), é um projeto que tem como objetivo dar apoio às fases do ciclo de vida de desenvolvimento de um software, desde a especificação até a sua implementação e manutenção. O projeto ADAO gerará automaticamente, a partir da especificação, um protótipo interativo do sistema alvo, e utilizará como base teórica, a abordagem operacional do método JSD e conceitos de programação com tipos abstratos de dados [Ms89a].

O ambiente será constituído de cinco módulos principais, como mostra a figura 1.2, os quais serão detalhados a seguir:

- **Gerador da Base Meta (GeBaM)**

Este módulo permite a geração da Base Meta do sistema onde

Figura 1.2: PROJETO ADAO

será guardada a definição da metodologia JSD em termos dos elementos que a compõem, como por exemplo: entidades, ações, atributos, vetores de estado, sequência de dados, etc. O modelo de dados correspondente a essa metodologia foi construído utilizando o modelo Entidade-Relacionamento estendido [Ch76] e posteriormente foi mapeado para uma linguagem descritiva analisável pelo Interpretador do Gerador de Base de Dados Meta descrito em [Tr86]. A geração da base de dados meta é uma tarefa que estará a cargo do administrador do ambiente e será gerada uma única vez.

- Analisador da Especificação Operacional (AnEspO)

O Analisador de Especificações Operacionais (AnEspO) permite ao engenheiro de software a entrada da especificação do Sistema de Informação. Este módulo utiliza a definição gerada por GeBaM e guarda a especificação do sistema na base de dados de especificações. Com respeito à abordagem operacional utilizada (JSD), este módulo cobre os quatro primeiros passos do método.

A especificação é analisada e erros são informados ao engenheiro de software para ao final obter uma especificação consistente do sistema alvo, isto é, do sistema em desenvolvimento.

- Editor de Textos Estruturados (EdTE)

Dentro do projeto, o módulo Editor de Textos Estruturados (EdTE) também é uma ferramenta de apoio à especificação do sistema. Trata-se de um editor orientado por sintaxe que permitirá ao engenheiro de software refinar os processos segundo as regras de JSD. Cada ação será decomposta em operações elementares e os predicados das iterações e seleções serão definidos. Serão acrescentadas as operações de leitura e gravações de sequências de dados, assim como as consultas a vetores de estados.

O EdTE permitirá também a visão de níveis hierárquicos diferentes, isto é, visões de processos no seu nível mais alto, assim como entrar nos detalhes das ações quando for necessário. No final, cada ação completamente detalhada será desmembrada da sua estrutura e armazenada como um procedimento associado à entidade. Cada entidade, com a sua estrutura de dados reunida a partir dos atributos associados a cada uma de suas ações, mais os procedimentos associados a cada ação, formará um tipo abstrato de dados [Ms89a].

- Executor da Especificação Operacional (ExEspO)

O módulo Executor da Especificação Operacional (ExEspO) tem como objetivo simular a execução do Sistema de Informação como um protótipo

funcional. Está composto por uma primeira parte, que é a Transição da Especificação para o Protótipo. Nesta parte, o ExEspO prepara o sistema para ser executado. Ele utiliza a especificação resultante do uso do AnEspO e do EdTE, que se encontra armazenada na base de especificações e gera a estrutura da “Base de Dados Protótipo”, onde serão armazenadas as instâncias da especificação.

A segunda parte do ExEspO funcionará como um alocador de processos, gerenciando chamadas à Base de Especificação, à Base Protótipo e ao sistema de Gerenciamento de Interface da Aplicação.

- Sistema de Gerenciamento da Interface da Aplicação (SiGIA)

Este módulo será responsável pela comunicação dos módulos AnEspO, EdTE e ExEspO com os usuários do ambiente, através de uma interface padronizada. Os módulos AnEspO e EdTE utilizarão um conjunto de rotinas de manipulação de janelas, menus e diálogos (um “TOOLKIT”), para controlar a interface com o engenheiro de software. O ExEspO usará o Sistema de Gerenciamento da Interface da Aplicação (SiGIA) para interagir com o usuário final do protótipo. O SiGIA é uma camada de software superposta ao sistema de gerenciamento de janelas, de forma a adequá-lo ao domínio de Sistemas de Informação e também aos conceitos do método JSD.

Em geral, a interface com o usuário será baseada nos conceitos de janelas, menus superpostos, formulários eletrônicos e ícones.

Portanto, esta dissertação trata apenas do componente AnEspO, do projeto ADAO. Os demais componentes estão sendo desenvolvidos por outros membros do grupo, como trabalhos de mestrado ou iniciação científica.

1.4 Organização da Dissertação

No capítulo 1 foram apresentadas as bases que motivaram este trabalho, dentro do interesse atual de pesquisas, no Brasil e em outros países, no desenvolvimento de Sistemas de Informação.

Na seção 1.2 apresentou-se uma descrição rápida das abordagens convencional e operacional para desenvolvimento de software e a maneira como as metodologias estruturadas apareceram para guiar o engenheiro de software nesse desenvolvimento. Dessas metodologias estruturadas, foi mencionada JSD como sendo a mais evoluída dentro da abordagem operacional, a qual dará suporte

teórico a este trabalho. Na seção 1.3 apresentou-se um resumo do projeto ADAO com o objetivo de dar uma visão global do ambiente no qual o AnEspO se encontra inserido.

O capítulo 2 apresenta os pontos relevantes do estudo bibliográfico realizado para o desenvolvimento deste trabalho. Descreve-se de uma maneira cronológica as publicações básicas que tratam do método JSD, assim como também se oferece uma visão geral sobre os esforços de desenvolvimento de ferramentas de apoio por computador (CASE) baseadas em JSD e em outros métodos correlatos.

O capítulo 3 está dedicado à apresentação do projeto do AnEspO. Nesse capítulo será apresentado o modelo de dados do método JSD, que será o suporte do ambiente ADAO. Em seguida será feita uma descrição geral dos módulos que compõem o AnEspO e os aspectos principais da sua especificação e implementação.

O uso do protótipo desenvolvido, suas restrições e um caso prático será apresentado no capítulo 4.

Finalmente, no capítulo 5 faz-se uma avaliação geral da ferramenta e da sugestões para possíveis linhas de pesquisas futuras.

Capítulo 2

Revisão Bibliográfica

2.1 Considerações Iniciais

A revisão bibliográfica apresentada neste capítulo tem o propósito de dar uma visão geral das publicações de interesse utilizadas neste trabalho para formar a base teórica necessária ao entendimento do método JSD e à formulação da proposta do AnEspO.

Não será feito uma análise detalhada de cada publicação, mas apenas um breve resumo de seu conteúdo, enfatizando-se sua importância para o desenvolvimento da ferramenta que será descrita no capítulo seguinte.

O capítulo foi dividido em duas seções: primeiramente na seção 2.2, revisa-se cronologicamente a bibliografia mais relevante relativa ao método JSD, composto de várias publicações na forma de artigos e livros. Posteriormente, na seção 2.3 apresenta-se a bibliografia sobre os esforços de desenvolvimento de ferramentas, com o mesmo objetivo do AnEspO, de dar apoio por computador ao desenvolvimento de Sistemas de Informação, e, também, as ferramentas atualmente disponíveis, que apoiam o método JSD.

2.2 Bibliografia Básica sobre o Método JSD

En 1978, Jackson, em [Ja78], propôs que um sistema fosse criado como um modelo da realidade à qual pertence e que as funções fossem superpostas a esse modelo. A forma do modelo proposto é uma rede de processos

que se comunicam por seqüências de dados. Isso permite uma representação clara das modificações ou atividades do sistema ao longo do tempo, separando as restrições orientadas para o problema das orientadas para a solução. Diz também que o modelo não pode ser eficientemente executado num único processador sem transformações nas especificações. Nesse trabalho apresenta vários exemplos para ilustrar o método e conclui que seu objetivo principal é separar o **modelo da função** e o **projeto da implementação**; é aqui que Jackson apresenta pela primeira vez o conceito de **entidade** no momento da especificação do sistema, assim como o conceito de **transformação** quando se fala em implementação. Segundo [Pe84], este é o trabalho de transição entre o método JSP [Ja75] e JSD [Ja83].

Em 1981, Jackson, em [Ja81], editado no livro [Co81], apresenta JSD como sendo um método adequado para o desenvolvimento de sistemas por computador. Afirmar que JSD é uma extensão do método JSP (estudo dos princípios de programação estruturada, descrito no seu livro original publicado em 1975 [Ja75]), dentro das áreas de análise de sistemas, especificação, projeto e implementação. Descreve os quatro princípios em que o método se baseia:

1. Um método deve decompor a atividade de desenvolvimento em tarefas distintas e ordenadas, oferecendo para cada tarefa uma ferramenta adequada à sua execução e critérios para determinar se foi completada corretamente;
2. O desenvolvedor do sistema deve começar considerando a realidade que o sistema vai modelar, ao invés das funções que o sistema vai realizar;
3. Uma realidade ativa só pode ser modelada por um modelo ativo, assim, um modelo inerte pode modelar apenas uma realidade inerte;
4. O projeto deve preceder e não ser confundido com a implementação.

Nesse mesmo trabalho Jackson classifica os passos para o desenvolvimento de um sistema pelo método JSD em três grupos: o primeiro grupo refere-se à especificação do modelo do mundo real; o segundo grupo à especificação das funções do sistema e o terceiro à implementação do sistema. Comenta que naquela época a experiência com JSD não havia atingido o ponto em que os passos individuais dentro dos grupos pudessem ser definidos, mas que a separação de um grupo do outro estava bem estabelecida.

No mesmo ano, Cameron, em [Ca81], reimpresso em [Be81], diz que JSD é um método direcionado para aqueles sistemas que tratam de objetos e entidades cujo comportamento e estado variam ao longo do tempo. Ilustra

os principais pontos do método, apresentando vários exemplos, nos quais, ele mesmo diz, omite muitos detalhes da especificação. Afirmar que os primeiros passos da abordagem tratam da construção de uma descrição formal da parte correspondente ao mundo real que interessa ao sistema.

Essa descrição é chamada de **modelo** e consiste de um número sequencial de processos que permanecerão ativos durante todo o período de tempo requerido pelo modelo; assim, se uma hipoteca, por exemplo, é feita para 25 anos, o modelo descreverá todos os 25 anos. Os passos seguintes do método correspondem à transformação da especificação formal num sistema executável. Os últimos passos são os da implementação, eles manipulam os textos dos processos especificados e adicionam novos programas para articular esses processos e o acesso a seus vetores de estado.

Já em 1983, Jackson, publicou o livro [Ja83], onde descreve o método JSD de uma maneira mais conceitual e com detalhes. Ele faz uma primeira distinção entre a especificação e implementação. Divide o processo de desenvolvimento em JSD em 6 etapas: escolha das entidades e ações, estruturação cronológica das ações, construção do modelo inicial, adição das funções, articulação temporal dos componentes do sistema e, projeto e implementação do sistema. As primeiras quatro etapas tem a ver com a criação de uma especificação do sistema requerido e as duas últimas com a implementação dessa especificação.

No livro, ele dedica um capítulo para cada passo do método e os conceitos são mostrados através de três exemplos: um sistema de apoio à realização de concursos, um sistema para controle de tráfego de um elevador e um sistema de atendimento de pedidos. Da forma em que os passos vão sendo apresentados, são aplicados ao exemplo que melhor pode ilustrá-los. Salienta que JSD não se preocupa com toda a atividade associada ao desenvolvimento de sistemas, já que não se preocupa com atividades tais como: seleção do projeto, planejamento e gerenciamento do projeto e análise de custo/benefício; exclui também procedimentos para aceitação do sistema e instalação e não se preocupa com mecanismos de diálogo.

Em 1984, Penteado em [Pe84], estuda JSD como uma abordagem operacional para o desenvolvimento de Sistemas de Informação, usa diagramas de estrutura para documentar cada um dos passos desse método, com seus correspondentes textos estruturados, que são combinados com recursos existentes no sistema PSL/PSA (Problem Statement Language/ Problem Statement Analyzer) [Te77].

Para que os profissionais da área pudessem absorver melhor os princípios e técnicas recomendadas por Jackson, Penteado apresenta o sistema de atendimento de pedidos extraído de [Ja83], desde a especificação até a implementação em COBOL, demonstrando dessa maneira o uso do método JSD como uma solução dos problemas reais.

Em 1986, Cameron, em [Ca86] diz que as especificações em JSD consistem principalmente de uma rede distribuída de processos sequenciais que se comunicam por troca de mensagens de leitura e escrita de seqüências de dados e por consulta aos seus vetores de estado; por essa razão afirma que uma especificação JSD pode ser diretamente executável, pelo menos em princípio, e algumas vezes também na prática.

Comenta, além disso, que as especificações são desenvolvidas a partir de um conjunto inicial de processos modelados que definem um conjunto de eventos, os quais limitam o escopo do sistema, definem sua semântica e formam a base da definição de dados e saídas. Salienta que a implementação do sistema envolve frequentemente uma transformação da rede de processos para executar em um pequeno número de processadores reais ou virtuais.

Diz também que um dos aspectos concernentes à fase de implementação é a escolha das estruturas de armazenamento (projeto físico da base de dados nos Sistemas de Informação, por exemplo) onde serão guardados os dados que manipulam os processos.

Refere-se ao método JSD como formado de 3 fases principais: a fase de MODELAGEM, na qual são selecionados e definidos os processos modelados, a fase de REDE na qual é desenvolvido o resto da especificação e a fase de IMPLEMENTAÇÃO na qual os processos e seus dados são encaixados nos processadores disponíveis e na memória. Ilustra as fases do método JSD com um exemplo simplificado de uma biblioteca.

Cameron analisa a fase de modelagem progressivamente. Inicialmente mostra o escopo do sistema da biblioteca como sendo constituído só pelas ações de um processo LIVRO. Com esse modelo introduz os conceitos básicos da fase de modelagem, onde inclui os diagramas de estruturação cronológica, como mostrado na figura 2.1 reproduzida desse documento, e a construção dos respectivos textos estruturados a partir desses diagramas.

Posteriormente estuda modelos mais complexos do mesmo exemplo, incluindo restrições do mundo real às ações já modeladas. Dessa maneira introduz o conceito da composição paralela existente entre processos que tem ações comuns e a compara com o paralelismo envolvido em CSP [Ho78].

Figura 2.1: DIAGRAMA DE ESTRUTURA DA ENTIDADE LIVRO
(Exemplo da biblioteca de [Ca86])

Finalmente faz uma terceira análise da fase de modelagem com respeito aos modelos de dados, em resposta à abordagem usada para desenvolvimento de sistemas comerciais, na qual se dá maior ênfase ao modelo de dados devido à argumentação de que estes são mais estáveis comparados com as funções da empresa, que geralmente tendem a mudar com maior facilidade. Diz que JSD pode ser visto como uma generalização dessa abordagem, já que inclui adicionalmente a dimensão de tempo no modelo de dados; ele argumenta que é tão importante captar a dinâmica da empresa na descrição que forma a base do sistema, quanto captar suas propriedades estáticas, e que para isto JSD descreve primeiro a dinâmica (o que acontece?, em que ordem?) e em seguida define os estados da empresa em termos dessa dinâmica (isto é, o que é armazenado acerca do que ocorreu?).

Na fase de rede, Cameron continua com a especificação, adicionando novos processos à rede e elaborando os que já estão nela; explica detalhadamente

a maneira de proceder para adicionar esses novos processos e posteriormente apresenta alguns exemplos dos tipos de funções de saída na aplicação da biblioteca para ilustrar na prática os conceitos que descreveu anteriormente. Para finalizar a etapa de rede faz uma análise das primitivas de comunicação, como a sincronização das leituras e escritas das seqüências de dados e as consultas aos vetores de estados.

Na fase de implementação ele coloca dois pontos relevantes de análise: como executar os processos da especificação e como armazenar os dados envolvidos nessa especificação. O primeiro ponto refere-se particularmente às seqüências de dados e o segundo às consultas aos vetores de estado da especificação. Salienta que o mecanismo utilizado é o de *suspender-e-retornar* (*suspend-and-resume*) convertendo um processo em uma subrotina onde suas entradas serão passadas como parâmetros da chamada. Diz que cada vez que a subrotina é chamada, executa parte de um programa sequencial de longa duração e devolve o controle quando está pronto para ler a próxima entrada.

Finalmente, Cameron dedica uma seção inteira do seu trabalho para analisar vários tópicos: faz uma comparação da modelagem JSD com respeito a métodos de análise com abordagem funcional; discute a composição e decomposição de processos como estratégias de desenvolvimento; analisa a variedade de caminhos com que as fases de JSD podem ser mapeadas em estruturas gerenciais de um plano de projeto; e, finalmente, faz uma breve descrição dos projetos que estão usando ou que usaram JSD para seu desenvolvimento.

Em [Ca88], Cameron explora a relação da fase de modelagem de JSD com técnicas convencionais de modelagem de dados. Afirma novamente que a fase de modelagem em JSD pode ser considerada uma extensão da modelagem de dados. Faz uma correspondência entre as duas abordagens: as entidades correspondem, em uma provável implementação, aos registros da base de dados; os atributos das entidades aos campos dos registros; as ações correspondem às transações que atualizam as entidades; e as regras de ordenação correspondem às restrições de integridade na base de dados.

Diz que os modelos JSD contém explicitamente entidades e atributos e implicitamente relações; para isto explica que cada chave estrangeira pode ser interpretada como uma relação, reconhecendo como chave estrangeira o atributo de uma entidade cujo tipo é o identificador de outra; por isto conclui que o modelo JSD contém uma forma simples e reconhecível de modelo de dados. Analisa algumas dificuldades relacionadas com a classificação de eventos individuais, ações e entidades de JSD e as extensões que se introduziram no método para manipular esses casos. Também apresenta exemplos para ilustrar seus estudos.

Em [Ms88], Masiero e Germano demonstraram que JSD pode ser utilizado como um método orientado a objetos. Para isso tomaram o exemplo da biblioteca de Cameron [Ca86], e mapearam o Diagrama de Implementação do Sistema para um projeto orientado a objetos e o implementaram em Modula-2. Este estudo foi realizado em resposta à classificação feita por Wiener e Sincovec [Wi84], na qual o JSD era incluído entre os métodos que não se enquadram como orientados a objetos, possivelmente por confundí-lo com o método anterior a ele, o JSP.

Também em 1988, Sutcliffe publicou um novo livro sobre JSD [Su88]. Seu objetivo é descrever o método, que foi proposto desde seu lançamento em 1983, como uma solução potencial aos problemas de confiabilidade e manutibilidade do desenvolvimento tradicional de sistemas, para ajudar estudantes e analistas a usá-lo no desenvolvimento de sistemas. O livro está baseado nas publicações anteriores de Jackson e em experiências próprias do autor. Descreve as etapas do método em capítulos separados e apresenta dois estudos de casos: um sistema de depósitos em bancos e um sistema de piloto automático de avião.

2.3 Ferramentas de Desenvolvimento Baseadas em JSD e outros Métodos Correlatos

2.3.1 Ferramentas de Apoio por Computador ao Desenvolvimento de Software

O desenvolvimento de ferramentas de apoio ao desenvolvimento de software (CASE) é considerado “uma arte e uma ciência” pelos profissionais da área [McC88]. A tecnologia de CASE está baseada na automatização de metodologias para desenvolvimento de sistemas, desde a etapa de planejamento até a etapa de manutenção. Essa tecnologia está voltada para automatizar a parte tediosa do desenvolvimento de sistemas, de tal maneira que libera o projetista para resolver outros problemas.

Com o aparecimento das metodologias estruturadas para desenvolvimento de sistemas, que ajudam a sistematizar e padronizar o desenvolvimento e a manutenção de sistemas, as ferramentas de CASE se tornaram muito importantes nesta área, já que são a esperança de solução para as técnicas de diagramação manuais, lentas e tediosas, embora muito importantes, que essas abordagens sugerem.

Cada ferramenta de CASE dá suporte a uma ou várias metodologias estruturadas. McClure, [McC88] agrupa as ferramentas de CASE nas seguintes categorias:

- **Ferramentas de diagramação** para representação de especificações de sistemas.
- **Formatadores / Editores de telas e relatórios** para criação de especificações de sistemas e para prototipagem simples.
- **Dicionários de dados** e sistemas de gerenciamento de informação.
- **Ferramentas de análise de especificação**, para detectar especificações incompletas de sistemas, sintaticamente incorretas e inconsistentes.
- **Geradores de código**, para gerar código executável a partir de especificações de sistemas simples.
- **Geradores de documentação**, para produzir documentação técnica e do usuário, requerida pelas metodologias estruturadas.

Inicialmente, muitas das ferramentas de CASE eram desenvolvidas para dar apoio a uma etapa do ciclo de vida de software, sem se preocupar com a integração de outras ferramentas do mesmo tipo que apoiavam outras etapas do desenvolvimento. Atualmente, a integração de ferramentas de CASE é um dos principais aspectos considerados pelos pesquisadores e desenvolvedores de CASE. A integração de ferramentas de CASE que automatizam as atividades do ciclo de vida de desenvolvimento de software cria um conjunto de ferramentas usualmente denominadas de “Ambientes de Desenvolvimento de Software”, [Da87].

Na literatura técnica especializada existem várias alternativas para classificar a integração de ferramentas de CASE. McClure por exemplo, divide a integração de ferramentas em quatro tipos principais:

I. Conjunto de Ferramentas (TOOLKITS)

Oferecem suporte à integração de ferramentas que são orientadas à automatização de uma só etapa do ciclo de vida de desenvolvimento de software. Atualmente encontram-se disponíveis no mercado várias ferramentas desse tipo. Por exemplo:

Para apoio à fase de análise e projeto:

- “The Developer”, desenvolvido pela empresa ASYST Technologies Inc.

- “Excelerator” e “Excelerator/RTS”, desenvolvidos pela Index Tech.
- “DesignAid”, desenvolvido pela Nastec Corporation. Para apoio à fase de programação
- “DEcase”, gerador de código COBOL, desenvolvido pela Digital Equipment Co.
- “Interlisp”, ambiente para linguagem Lisp
- “Smalltalk”, ambiente para linguagem Smalltalk

Para apoio à fase de manutenção:

- “INSPECTOR” e “VIA Insight”, ferramentas para manutenção de programas COBOL.

II. Bancadas (WORKBENCHES)

Permitem a integração de ferramentas para automatizar todo o ciclo de vida de desenvolvimento de software. As saídas de uma fase do ciclo de vida são direta e automaticamente passadas para a próxima fase e o produto final é um sistema de software executável, junto com a sua documentação. Exemplos de BANCADAS existentes no mercado são:

- “COVISION”, apoia a metodologia de Engenharia da Informação e também é um gerador de código, foi desenvolvido pela Cortex Corporation.
- “Information Engineering Workbench”, é uma bancada para apoio às metodologias de Engenharia da Informação e Análise Estruturada, é também um gerador de código COBOL; foi desenvolvida pela Knowledgeware Inc.
- “Information Engineering Facility (IEF)”, foi desenvolvida pela Texas Instruments; apoia a Engenharia da Informação e gera código COBOL.

III. Integradores (FRAMEWORKS)

Os integradores fornecem uma infraestrutura para integrar e gerenciar ferramentas de software individuais como ferramentas de CASE, ferramentas de implementação de programas tradicionais e ferramentas de gerenciamento de base de dados. Geralmente, os integradores têm uma interface do usuário comum a todas as ferramentas, compartilham interfaces de dados e rodam no mesmo ambiente de hardware. São exemplos de integradores:

- “MULTI/CAM” desenvolvida por AGS Management Systems, usada para gerenciamento de projetos.
- “Life-Cycle Productivity System” é uma ferramenta para gerenciamento de projetos e de planejamento; foi desenvolvida pela empresa American Management Systems Inc.
- “VM/ Software Engineering” que é uma ferramenta para gerenciamento de configurações desenvolvido pela IBM.

IV . Ambientes Baseados em Metodologias (METHODOLOGY COMPANION)

Oferecem um conjunto de ferramentas integradas que dão suporte a uma metodologia estruturada e guiam automaticamente o desenvolvedor através das etapas de desenvolvimento sugeridas por essa metodologia. Alguns exemplos destas ferramentas são:

Ambientes em torno dos métodos de Análise e Projeto Estruturado:

- Nastec CASE 2000, desenvolvido pela Digital Consulting Inc.
- Teamwork.

Ambientes em torno de métodos de Modelagem de Dados:

- IEW.

2.3.2 Ferramentas de Apoio a JSD

Cameron, em [Ca86], refere-se a três ferramentas, atualmente disponíveis, que dão suporte a JSD. Essas ferramentas foram desenvolvidas por Michael Jackson Systems Ltd. e são:

- **PDF**, é um editor gráfico para os diagramas em árvore usados em JSD, e para as listas de operações; possui também um gerador de código a partir desses diagramas para uma variedade de linguagens frequentemente usadas. A idéia do pacote é que os diagramas sejam a origem do programa. PDF roda em microcomputadores IBM, e VAX sob o sistema operacional UNIX.

- **SPEEDBUILDER** é planejado para ser uma série de produtos que apoiam JSD. Cameron diz que até a época (1986) só estava em funcionamento a **UNIT ONE**. A **UNIT ONE** é uma base de dados para armazenar uma especificação em JSD; possui uma interface amigável para o usuário e um sistema de documentação, o qual permite que subconjuntos da base de dados sejam impressos em um formato definido pelo usuário. **SPEEDBUILDER** executa em equipamentos do tipo IBM PC.
- **JSP-COBOL** é um processador de COBOL que automatiza o mecanismo de suspender-e-retornar (suspend-and-resume) descrito na seção 2.2, e fornece uma variedade de ajudas para testes. **JSP-COBOL** roda em uma variedade de mini computadores e computadores de grande porte.

Em 1986, Velasco, [Ve86], desenvolveu o **JSD-TOOL** como uma ferramenta para apoio ao desenvolvimento de sistemas, baseado nas regras do método JSD. A ferramenta é formada por uma linguagem que pode ser usada para desenvolver especificações JSD para as cinco primeiras etapas do método. A verificação da especificação criada com **JSD-TOOL** é feita através de checagens automáticas de consistência e completude realizadas em cada passo do método e mostradas ao usuário através de relatórios. Oferece também a opção de emitir vários outros tipos de relatórios que poderão ser úteis ao analista na documentação do sistema. No final cria um arquivo contendo a especificação desenvolvida.

O **EXECUTOR**, [Am88], é uma ferramenta de apoio a JSD que executa diretamente uma especificação desenvolvida nesse método. O **EXECUTOR** se dedica especificamente à etapa de implementação do método JSD. Utiliza uma linguagem de especificação similar à usada no **JSD-TOOL** para descrever o Diagrama de Especificação do Sistema (DES) e a estrutura textual dos processos. Por essa razão o Executor poderia ser visto como um complemento do **JSD-TOOL**, mas não está integrada com ele.

Para executar a especificação JSD, o Executor, faz uma simulação em memória utilizando um mecanismo capaz de criar máquinas virtuais para a execução concorrente dos vários processos especificados.

Em 1989, Engmam e Van Hoeve, [En89], apresentam o **Gerador de Programas de Processamento de Arquivos** dentro do ambiente TUBA (Tools for User-friendly Business Applications). Neste gerador, as estruturas dos programas são derivadas de estruturas de dados especificadas segundo o método JSP. Ele foi especialmente projetado para a geração de programas que processam um ou mais arquivos. Lê sequencialmente um dos arquivos de entrada e dispara

operações de acesso ou escrita para outros arquivos. A especificação dos dados de saída é feita através de uma interface interativa e a especificação do programa é armazenada em um dicionário. A partir dessas especificações podem ser gerados programas executáveis. O sistema foi implementado num computador de grande porte do tipo DECsystem-20.

Outra contribuição ao conjunto de ferramentas de apoio a JSD é o **JSD Prototyping Environment** [Fe89], desenvolvido pela Michael Jackson System Ltd. (MJSL). Trata-se de um ambiente no qual o analista de sistemas usa as ferramentas JSP-Tool e SPEEDBUILDER para desenhar os diagramas de estruturas e armazenar as especificações JSD. O analista define operações usando uma combinação de comandos COBOL e macros. Quando a especificação do sistema está completa, o analista simplesmente "digita um comando" e o Prototyping Environment gera um sistema executável.

A geração de processos é feita da seguinte maneira: primeiro o Speedbuilder gera relatórios que fornecerão informação das entidades, ações e atributos. Esta informação estará guardada em arquivos em disco. Os arquivos são processados e convertidos em esqueletos de programas fonte COBOL e em um conjunto de MACROS. O código fonte e as macros são processados pelo JSP-Macro (também desenvolvido na MJSL), que expande os esqueletos de programas para programas completos em COBOL. Cada processo no sistema é representado por um programa. Posteriormente esses programas são compilados usando o compilador Microfocus COBOL e ligados para criar os módulos executáveis (arquivos com extensão .EXE). Neste ponto o usuário tem um sistema protótipo completamente executável.

Ferg diz que esta ferramenta prova que JSD não é uma teoria meramente interessante, já que JSD pode ser colocada numa aplicação prática e usada com ferramentas de CASE que a tornam muito efetiva. Conclui também que este ambiente ajuda a demonstrar o embasamento teórico de JSD.

2.4 Reflexões sobre a Bibliografia Revisada

2.4.1 Sobre o Método JSD

Jackson fundamentou-se em conceitos matemáticos e teóricos de projeto de programas para desenvolver um método prático que começou em 1975 com o JSP (Jackson Structured Programming) e mais tarde evoluiu para o JSD

[Su88].

O próprio Jackson considera o método JSD como uma extensão do método JSP, dentro das áreas de análise de sistemas, especificação, projeto e implementação, pois ele incorpora o conceito central de JSP da comunicação de processos sequenciais, que alimentam e são alimentados por fluxos de dados sequenciais [Ja81]. O método é formado por seis passos distintos, sendo que os quatro primeiros referem-se à especificação do sistema e os dois últimos à implementação do sistema.

Por ser um método mais formal que outros, seu uso em problemas práticos apresenta algumas dificuldades. Em [Ja83], alerta-se para o perigo de, no passo de escolha das entidades, se chegar a conclusões sem um estudo adequado. Por isso, Jackson recomenda que a lista inicial de entidades seja extensa, podendo, entretanto, a lista de entidades resultantes após sucessivos refinamentos ser pequena e simples.

Em JSD, não há preocupação com os aspectos tradicionais de análise de sistemas. Por exemplo: ele não trata do problema de como obter do usuário os requisitos do sistema. A comunicação com o usuário, utilizando os recursos que o método oferece, não é fácil, porque ele precisa ter conhecimento do método de forma a poder fornecer subsídios ao analista para que este possa modelar um sistema que satisfaça suas necessidades. Já o programador encontrará facilidade para a programação, com a utilização dos textos estruturados que serão desenvolvidos pelo analista, na etapa de estruturação cronológica, e completados na construção do modelo inicial e adição das funções.

A partir da publicação do livro de Jackson, vários outros artigos relacionados ao método têm aparecido regularmente em revistas da área, o que mostra o interesse despertado pelo método e a sua validação na prática profissional. Em [Ca88] dá-se ênfase especial à fase de modelagem, relacionando-a com técnicas convencionais de modelagem de dados, como foi apresentado na seção 2.2.

2.4.2 Sobre as Ferramentas de CASE

Na última década, a disponibilidade de técnicas mais rigorosas, como análise, projeto e programação estruturada, para o desenvolvimento de software, têm ajudado a melhorar a qualidade dos sistemas. Embora essas técnicas sejam usadas na prática, elas são complexas e consomem muito tempo. Na ausência de ferramentas automatizadas, a implementação dessas técnicas não tem melho-

rado o nível de produtividade desejado pelos gerentes de sistemas. Daí a grande necessidade das organizações em contar com um suporte automatizado ao desenvolvimento do ciclo de vida de software e à manutenção. As organizações não só requerem metodologias, mas também um conjunto de ferramentas automatizadas de apoio ao desenvolvimento de software.

As ferramentas de CASE atualmente existentes no mercado podem ter uma ou mais das seguintes características:

- Gráficos para diagramas de estruturas, fluxo de dados e diagramas de modelagem de dados.
- Documentação automática dos requisitos do sistema e do projeto do sistema.
- Sistema de gerenciamento de base de dados.
- Dicionário de dados e/ou enciclopédia/depósito para armazenamento de informação.
- Verificadores de consistência e completude.
- Geradores de código e reestruturadores.
- Editores de telas e relatórios.
- Gerenciadores de projeto e de protomodelagem.
- Assistência automática para desenvolvimento de metodologias, tais como Constantine-DeMarco, Gane-Sarson, Yourdon, Jackson ou Engenharia da Informação.

A tabela 2.1 mostra alguns recursos recopilados da revisão dos CASE estudados na seção 2.3, e os compara com o projeto ADAO. Os dois primeiros recursos da tabela : **Editor de diagramas de fluxo de dados, e editor de Diagramas Hierárquicos**, não podem ser incluídos no ADAO, pois ele apoia principalmente o método JSD, e esses diagramas não fazem parte do método. O **editor de diagramas Entidade-Relacionamento e diagramas de estrutura** são recursos que atualmente estão em desenvolvimento, em fase de especificação. A **geração automática de protótipos** se encontra também em desenvolvimento, em fase de implementação. O **Gerador de código e Interface com SGBDs** são ferramentas possíveis de serem integradas ao ambiente ADAO, mas que para a primeira versão do ambiente não foram projetadas. O **Apoio**

a sistemas de tempo real é uma característica implícita do método JSD, é por isso que o ambiente pode suportar especificações de sistemas de tempo real; como foi dito no capítulo 1. O projeto ADAO conta com um dicionário de dados que se encontra armazenado na base de especificações do ambiente. Finalmente o ambiente é capaz de emitir relatórios de consistência e também relatórios gerais.

Recursos Principais	EXCELERATOR	IEW	PDF	SPEED BUILDER	ADAO
Editor de DFD	X	X			⓪
Editor de DHM	X	X			⓪
Editor de diagramas E-R	X	X			*
Editor de diagramas de estrutura			X		*
Dicionário de dados	X	X			X
Gerador de código	X	X	X	X	+
Editor de Textos estruturados					X
Interface com SGBDs	X	X			+
Geração automática de protótipos					*
Apoia sistemas de tempo real /RT					X
Relatórios de consistência	X	X			X
Relatórios gerais	X	X			X

⓪ Não faz parte do método JSD

* Em desenvolvimento

+ Possível de ser feito. Previsto para outras versões

TABELA 2.1 Comparação das Características de Alguns dos CASE Disponíveis no Mercado e o Projeto ADAO

2.5 Considerações Finais

Da leitura dos textos revisados neste capítulo, verifica-se que a preocupação de dar apoio por computador ao desenvolvimento de Sistemas de Informação, vem crescendo rapidamente nos últimos anos. Em particular, é abundante o número de ferramentas que se fundamentam em métodos conhecidos de análise e projeto de sistemas de informação, dentre eles o JSD. Entretanto, para este método a quantidade de ferramentas disponíveis não é tão grande e são poucas as opções comerciais.

É possível concluir que o método JSD tem procedimentos e diretrizes bem desenvolvidas, de forma a permitir, na prática profissional, a geração de uma especificação operacional. Essa é uma das justificativas para a escolha desse método como base do desenvolvimento da ferramenta Analisador de Especificações Operacionais cuja descrição será apresentada no capítulo seguinte.

Capítulo 3

Sobre o Projeto do AnEspO

3.1 Considerações Iniciais

Neste capítulo apresenta-se uma descrição modular do Analisador de Especificações Operacionais (AnEspO), para que o leitor tenha uma idéia mais clara do propósito da ferramenta desenvolvida.

Inicialmente na seção 3.2, é apresentado o Modelo de Dados do Método JSD, fazendo uma breve introdução ao Sistema Integrado para Produção de Software (SIPS), de tal maneira a oferecer uma visão geral do sistema e mostrar a base de dados que o projeto ADAO utilizará, do qual o AnEspO é parte.

A seção 3.3 se dedicará à apresentação modular do sistema AnEspO, fazendo uma comparação de cada módulo com os quatro primeiros passos do método JSD, ao mesmo tempo que são introduzidos vários conceitos relevantes do método, necessários para a descrição de cada módulo.

Finalmente, na seção 3.4, são mostrados os aspectos principais da especificação do AnEspO, ressaltando a interface amigável sob a qual foi desenvolvida.

3.2 O Modelo de Dados do Método JSD

3.2.1 Introdução ao Sistema SIPS

O Sistema Integrado para Produção de Software (SIPS) é um sistema expansível para apoio à produção de software, que foi desenvolvido pelo Centro Tecnológico para Informática. Seu objetivo é dar suporte ao desenvolvimento integrado de software, abrangendo todas as fases do ciclo de vida, através de ferramentas automatizadas que contribuam para uma maior agilidade do processo de desenvolvimento de sistemas, aumento de produtividade e uma melhor qualidade do software produzido.

O SIPS segue uma linha de desenvolvimento que se caracteriza por ser um sistema projetado para auxiliar na automatização de metodologias já existentes. Ele pode ser instanciado para dar apoio automatizado a muitas metodologias, mesmo simultaneamente, servindo tanto para apoiar metodologias de forma automatizada, como para atuar na forma de um elo de integração entre várias metodologias usadas em fases distintas do ciclo de desenvolvimento de software [Tr86].

O SIPS está contituído por dois subsistemas: um que permite a um especialista em metodologias a Integração de Metodologias, criando uma **Base-Meta**; e outro chamado **SIPS-Sistema**, que permite aos projetistas o uso do SIPS para apoiar o desenvolvimento de sistemas empregando metodologias já integradas. A figura 3.1 mostra os componentes do SIPS, assim como os vários tipos de usuários que com ele interagem; a linha tracejada vertical indica a separação entre os dois subsistemas do SIPS.

A Incorporação de uma Metodologia ao SIPS é feita por um Especialista da Metodologia, que tenha dela um profundo conhecimento e que conheça como essa tarefa é realizada com o SIPS.

As principais etapas para a incorporação de uma metodologia ao SIPS são:

1. Preparação do modelo da metodologia usando o Modelo Entidade - Relacionamento estendido [Ch76]. Nessa modelagem, os elementos da metodologia são representados através de um conjunto de Tipos de Entidades, de um conjunto de Tipos de Relacionamento, e de um conjunto de Tipos de atributos. Cabe mencionar que o termo "ENTIDADES" no SIPS tem como sinônimo o termo "OBJETO", reservando-se aquela palavra para designar

Figura 3.1: COMPONENTES DO SIPS E INTERAÇÃO COM O USUÁRIO

o conceito técnico que ela tem em JSD.

2. O segundo passo consiste na transformação do modelo obtido para uma linguagem descritiva, analisável pelo computador. O SIPS possui um interpretador para essa linguagem, capaz de representar todos os aspectos necessários dessa modelagem. Nessa linguagem existem comandos com os quais serão descritos os conjuntos de Tipos de Entidades, Tipos de Relacionamentos e Tipos de Atributos, bem como a sintaxe entre eles.
3. O terceiro passo consiste em usar o Interpretador Meta, (ver figura 3.1), para inserir essa descrição como uma Meta-descrição em uma base de dados, que é chamada Base Meta por conter uma descrição da linguagem, que será usada a partir de então na descrição dos sistemas do usuário.

Após a definição de uma metodologia, o SIPS pode ser usado por projetistas de sistemas, para obter apoio automatizado no uso dessa metodologia.

Para isso o SIPS conta com um conjunto de ferramentas textuais e gráficas, que podem ser usadas genericamente com qualquer metodologia. No entanto, faz parte das diretrizes do projeto SIPS sua extensibilidade, no sentido de serem acopladas novas ferramentas, concebidas para um apoio específico a alguma determinada metodologia. Essas ferramentas irão sendo acrescentadas sucessivamente, ampliando a capacidade de descrição e análise do SIPS, à medida que seja aumentada a quantidade de metodologias que ele atende.

Várias metodologias podem coexistir simultaneamente no SIPS, desde que os modelos de cada metodologia descrita possam ser interligados. Assim, se um grupo de desenvolvimento de software usa um conjunto de metodologias, mesmo que de forma manual, mas conseguindo algum grau de integração entre elas, o SIPS poderá suportá-las de forma automatizada, com o mesmo grau de integração entre elas. Dessa forma, a integração de uma nova metodologia com outras já existentes, será feita com maior coesão quanto maior for o grau de compatibilidade existente entre elas.

O que garante a possibilidade de integração entre as metodologias concebidas de forma independente, é a utilização de um modelo de representação de informações interno igual para todas as metodologias, que tem como embrião o Modelo Entidade - Relacionamento.

Atualmente, existem integradas ao SIPS a metodologia Análise Estruturada, a metodologia de Representação do Grafo de Controle do Sistema SARA e a Técnica de desenvolvimento de sistemas em tempo real apoiada no Diagrama de Fluxo de Controle de Dados [Tr86]. Para essas metodologias existem

algumas ferramentas específicas como: Dicionário de Dados da Análise Estruturada, integrada ao Editor Gráfico de Diagramas de Fluxo de Dados; Dicionário de Dados de Rede de Petri, integrado ao Editor Gráfico e ao Simulador de Redes de Petri; e Dicionário de Dados de Gerenciamento de Projeto [Ak89].

Embora tenha sido descrito todo o ambiente SIPS nesta seção, para o projeto desenvolvido neste trabalho, apenas o núcleo da base de dados desse ambiente foi utilizada, isto é, nenhuma das ferramentas de uso geral nem as ferramentas de gerenciamento de projeto que se encontram atualmente integradas ao SIPS, foram utilizadas. Na figura 3.1 esse núcleo corresponde ao Interpretador Meta, a Base de Dados Meta e a Base de dados Sistema, conjuntamente com o conjunto de rotinas que manipulam a informação dessas bases [Si88]. É por isso que de agora em diante esse núcleo da base de dados será referenciado como “Base de Dados MRO”, devido à origem do modelo (Modelo de Representação de Objetos [Tr89]), utilizado pelo SIPS na construção do seu núcleo.

O presente trabalho incorporou a definição de uma nova metodologia à Base de Dados MRO, correspondente ao Modelo do Método JSD, sob o qual foi projetado o ambiente ADAO. Dentro do contexto desta base de dados, o ambiente utiliza uma Definição Meta do método JSD que se encontra armazenada na “Base de Dados Meta”, que servirá para cobrir parte do ciclo de vida de desenvolvimento de software. A ferramenta Analisador de Especificações Operacionais (AnEspO), que faz parte desse ambiente e que também forma parte deste trabalho, permitirá ao engenheiro de software a entrada da especificação de um sistema de informação que será armazenada na “Base de Dados Sistema”.

3.2.2 O Processo de Criação de uma Definição Meta para JSD

Seguindo os passos indicados na seção anterior, para a integração de uma metodologia na Base de Dados MRO, foi preparado o modelo de dados do método JSD na forma de um diagrama Entidade-Relacionamento. O diagrama completo, e seu correspondente dicionário de dados pode ser encontrado no apêndice A deste documento. No restante desta seção detalha-se a preparação deste modelo.

A primeira etapa dessa atividade é a determinação do conjunto de tipos de entidades relevantes para o método, isto é, as entidades as quais o método permite a representação e manipulação.

O método JSD necessitou de quatorze tipos de entidades para a re-

apresentação das quatro primeiras etapas que o compõem. Cabe alguma discussão quanto às decisões tomadas nesse particular. O princípio geral aplicado foi o de que se a entidade tem uma representação própria, ela será então considerada como uma entidade independente. Isso deve-se à restrição imposta pela versão da Base de Dados MRO utilizada, que não permite o conceito de generalização.

Dessa forma, as seqüências de dados, que são de quatro tipos: seqüência de dados de entrada externa (marcador de intervalo de tempo - MIT), seqüência de dados de entrada externa do mundo real, seqüência de dados de entrada interna e seqüência de dados de saída, foram consideradas como quatro entidades diferentes; SEQ-ENTRADA-EXTERNA-MIT, SEQ-ENTRADA-EXTERNA-MUNDO-REA, SEQ-ENTRADA-INTERNA, SEQ-SAIDA. A outra solução seria ter apenas uma entidade chamada SEQUENCIA-DE-DADOS e um atributo de tipo de seqüência com quatro possíveis valores alternativos: entrada externa MIT, entrada externa do mundo real, entrada interna e saída.

A mesma justificativa do parágrafo anterior se aplica as entidades PROCESSO-FUNCIONAL E PROCESSO-MODELAGEM. As seguintes entidades complementam a descrição do método JSD: AÇÃO, ATRIBUTO, DIAGRAMA-ESTRUTURA, ENTIDADE E VETOR-ESTADO.

Existem outros tipos de entidades que foram escolhidas apesar de não fazerem parte explícita da descrição de JSD, mas são necessárias para guardar informação relevante para o ambiente em desenvolvimento. A entidade SISTEMA foi incluída para armazenar informação correspondente à identificação do sistema alvo. A entidade TIPO-DE-DADO foi incluída porque representa o fundamento para que a especificação armazenada possa ser executada automaticamente. Finalmente, a entidade VALOR guardará informação do domínio discreto dos atributos. Ao todo são quatorze entidades (ou objetos, no jargão da Base de dados MRO).

A figura 3.2 mostra as 14 entidades escolhidas para representar o método JSD dispostas em dois grupos: entidades explícitas e implícitas ao método.

Cada entidade ou objeto pode ter atributos que permitem sua melhor caracterização. Na Base de dados MRO, estes atributos podem ser de vários tipos, mas para o nosso caso foram utilizados somente dois: PROPRIEDADE e COMENTARIO. O primeiro refere-se a tipos de atributos comuns que não tem nenhuma característica especial que os distinga dos demais (exemplo: valores numéricos, nomes, datas, etc.) e o segundo tem a ver com "campos longos" ou "textos", isto é, informação para o usuário que não será analisada pelo sistema

Figura 3.2: TIPOS DE OBJETOS DE JSD

Figura 3.3: PROPRIEDADES DE JSD

Figura 3.4: COMENTÁRIOS DE JSD

(exemplo: descrições, comentários, etc.). As figuras 3.3 e 3.4 mostram os atributos das entidades de JSD separadas pelo tipo: propriedade e comentário.

Para finalizar o modelo do método JSD só falta agora a especificação dos relacionamentos existentes entre as entidades escolhidas. Para isto foi selecionado um grupo de dez tipos de relacionamentos que poderão ser utilizados no relacionamento das entidades especificadas, de maneira a englobar todos os aspectos que envolvem os quatro primeiros passos do método. Para aumentar o conteúdo semântico do modelo foi utilizado o “relacionamento oposto”, que geralmente é formado pelo verbo que identifica o relacionamento, em voz passiva. Abaixo faz-se uma listagem dos relacionamentos selecionados e seus opostos:

Relacionamento	Relacionamento Oposto
modela	modelado-por
comunica	comunicado-por
consulta	consultado-por
executa	executado-por
identifica	identificado-por
compoe	composto-por
gera	e-gerado-por
intercala	intercalado-por
recebe	recebido-por
tem	fazem-parte-de

Esses relacionamentos podem ser explicados dentro das quatro primeiras etapas do método JSD. Essas etapas poderiam ser subdivididas em duas grandes fases: Fase de Modelagem e Fase de Construção da Rede de Processos. Formam parte da fase de modelagem: a escolha das entidades e ações e a estruturação cronológica das ações. A fase de construção da rede de processos está formada pela construção do modelo inicial e a adição das funções. Em seguida mostram-se alguns exemplos de aspectos selecionados nessas fases, com o nome de cada relacionamento e uma explicação do seu significado, para que os relacionamentos escolhidos sejam melhor entendidos. É importante salientar que, para dar um melhor sentido ao significado dos relacionamentos, nestes exemplos, foram utilizados alguns “relacionamentos opostos”. Entretanto, qualquer um dos dois tipos de relacionamento poderia ser utilizado na lista abaixo porque, na base de dados, inserindo um deles, tem-se o outro automaticamente.

Fase de Modelagem

ASPECTO: ESCOLHA DAS ENTIDADES E AÇÕES

RELACIONAMENTO: "Executa" - Indica que cada entidade do sistema alvo executa várias ações do mundo real.

RELACIONAMENTO: "Identificado-por" - Indica que cada entidade terá um identificador correspondente a um tipo de dados.

RELACIONAMENTO: "Tem" - Indica que toda ação está caracterizada por seus respectivos atributos.

ASPECTO: ESTRUTURAÇÃO CRONOLÓGICA DAS AÇÕES

RELACIONAMENTO: "Fazem-parte-de" - Indica que a ordenação no tempo das ações de uma entidade faz parte de um diagrama de estrutura.

Fase de Construção da Rede de Processos

ASPECTO: CONSTRUÇÃO DO MODELO INICIAL

RELACIONAMENTO: "Recebe" - Indica que um processo de modelagem pode receber uma seqüência de entrada externa do mundo real.

RELACIONAMENTO: "Consulta" - Indica que um processo de modelagem para a construção do modelo inicial pode consultar um vetor de estados.

ASPECTO: ADIÇÃO DAS FUNÇÕES

RELACIONAMENTO: "Gera" - Indica que um processo funcional pode gerar seqüências de entrada interna (função interativa), ou seqüências de saída.

RELACIONAMENTO: "Recebe" - Indica que um processo funcional pode receber como entrada uma seqüência de dados de entrada interna ou uma seqüência de dados de entrada externa do tipo marcador de intervalo de tempo.

RELACIONAMENTO: "Consulta" - Indica que um processo funcional pode consultar um vetor de estado de outro processo da rede.

RELACIONAMENTO: "Intercala" - Indica que um processo funcional pode intercalar seqüências de entrada interna e marcadores de intervalo de tempo.

O segundo passo na incorporação da metodologia é a transformação do modelo para a linguagem de descrição do modelo de dados da Base de Dados MRO. Para facilitar essa transformação, mapeou-se o Diagrama Entidade Relacionamento para um Diagrama de Representação de Objetos (DRO) [Tr89], não porque este mapeamento seja verdadeiramente necessário, mas devido a alguns conceitos que a versão da Base de dados MRO não considera, como a generalização, que foi utilizado no diagrama original da modelagem, e que tiveram que ser transformados em objetos individuais no momento da codificação para a linguagem de descrição de dados da Base de Dados MRO.

Posteriormente foi integrada efetivamente a metodologia JSD à Base de dados MRO, utilizando o Interpretador Meta, que gera a Base Meta e os relatórios de análise da geração da base.

O DRO, e os relatórios da informação meta se encontram também no apêndice A.

3.3 Arquitetura do Sistema AnEspO

No capítulo 1 foi descrito o Analisador de Especificações Operacionais como sendo parte do projeto ADAO, que permitirá ao engenheiro de software entrar com a especificação de um Sistema de Informação na base de dados de especificações. Para realizar esta tarefa, o AnEspO seguiu as diretrizes dos quatro primeiros passos do método JSD.

O sistema AnEspO está dividido em cinco módulos principais: Tipos de dados, Entidades-Ações-Atributos, Estruturação Cronológica, Modelo Inicial e Rede de Processos. Esses módulos formam parte do menu principal da ferramenta. Adicionalmente está sendo desenvolvido um sexto módulo que envolve relatórios gerais da informação armazenada na base.

Cada módulo do AnEspO fará checagens de consistência do método, para no final obter uma especificação do sistema consistente, tanto quanto possível, com a base de especificação. Para o caso de erros que não poderão ser checados automaticamente, estará disponível a opção de emitir relatórios que ajudem ao engenheiro de software a encontrar esses possíveis erros de consistência antes de passar para a fase de Transição, quando o protótipo será gerado.

Uma visão geral do AnEspO dentro do ambiente ADAO é mostrada na figura 3.5. Como mostra essa figura, existe uma interação entre os módulos "Entidades-Ações-Atributos" e "Rede de Processos" com o módulo "Tipos de Dados" em razão da exigência adotada de forma padronizada pelo sistema, de que o tipo de dados seja criado antes de ser utilizado. Na mesma figura o ambiente ADAO está também mostrado de uma forma modular, diferente da apresentada no capítulo 1.

Nas seções posteriores descreve-se cada um desses módulos, relacionando-os com os conceitos envolvidos nos diferentes passos do método JSD.

3.3.1 Tipos de Dados

Embora os tipos de dados não apareçam explicitamente em nenhuma fase do método JSD, eles foram incluídos como parte da especificação e implementação do sistema porque representam o fundamento para que a especificação armazenada possa ser executada automaticamente, nas próximas etapas do ambiente ADAO.

Os tipos de dados estão relacionados com os atributos, porque todo atributo tem um tipo de dado; com os vetores de estado, porque um vetor de estado é um tipo de registro composto por atributos; e com as seqüências de dados, porque elas também possuem seus próprios tipos de dados. Todas essas relações podem ser consideradas como relações implícitas no modelo [Ca88], mas é necessário que o engenheiro de software as defina para dar consistência à especificação que está sendo criada.

O AnEspO fornece alguns tipos de dados primitivos, como por exemplo: Inteiros, Booleanos, Valores Monetários, Caracteres, Datas e Endereços, mas o usuário também tem opção de criar novos tipos de dados. Os tipos de dados definidos pelo usuário podem ser divididos em duas categorias: Tipos de Dados Simples, definidos em função de um tipo de dado já existente; e, Tipos de Dados Compostos, definidos em função de vários tipos de dados (simples ou compostos) já existentes.

Como foi mencionado no item anterior, um princípio básico do AnEspO é que em algumas partes do sistema se exige que o tipo de dados seja criado antes de poder ser utilizado. Esta exigência foi adotada para diminuir a complexidade da interface do sistema e tentar evitar que o engenheiro de software cometa erros com muita facilidade, como esquecer de declarar um tipo de dados, declarar tipos inconsistentes, etc.

Figura 3.5: O ANALISADOR DE ESPECIFICAÇÕES OPERACIONAIS DENTRO DO AMBIENTE ADAO

Entretanto, em certas situações iniciais, para não truncar o fluxo normal de entrada da especificação, o sistema aceita que uma entidade tenha como identificador um tipo de dados ainda não definido, o que poderá ser feito mais tarde.

3.3.2 Entidades - Ações - Atributos

Em JSD, o Sistema de Informação a ser desenvolvido deve ser estudado visando a percepção de quais são as entidades de interesse no ambiente real do problema. As entidades são melhor percebidas pelas ações que elas executam ou sofrem no mundo real. As ações são eventos que afetam o ambiente e mudam a situação de cada entidade. O sistema deve manter um registro da situação de cada entidade e de como ela se altera a cada ação executada. Esse registro é mantido por diversos atributos relacionados à entidade [Ms89a].

Dentro do AnEspO, o módulo Entidades-Ações-Atributos relaciona-se com a primeira etapa do método JSD, brevemente descrita acima. Este módulo permite ao engenheiro de software a entrada das entidades, ações e atributos do sistema. Esses elementos são guardados na base de dados de especificação como entidades separadas, que estão relacionadas entre si.

Pela forma hierárquica como o módulo Entidades-Ações-Atributos foi especificado, se garante a consistência da informação que ele manipula no que se refere à dependência entre entidades, ações e atributos que o método sugere. Por exemplo, para inserir uma ação, o usuário deve escolher primeiro a entidade à qual ela pertence. Se essa ação já existe na base de especificações, ou seja, trata-se de uma ação compartilhada por outra(s) entidade(s), o sistema alerta o engenheiro de software, mostrando-lhe essa informação e faz com que ele confirme se a inserção a ser feita está correta. Esse tipo de controle implícito, como o descrito acima, garante que não possam ser criadas ações ou atributos que não estejam ligadas diretamente a uma entidade.

A relação deste módulo com o módulo Tipo de Dados, também pode ser mostrada. No caso, os atributos estão sempre relacionados com um tipo específico de dados, como pode ser visto no diagrama Entidade-Relacionamento do modelo de dados do método JSD, no apêndice A (figura A.1). Neste módulo, então, para todo atributo inserido, deve ser indicado o tipo de dados ao qual ele pertence. Essa informação é obrigatória para manter a consistência da base da especificação. Se um atributo pertence a um tipo de dados ainda não inserido, o usuário será forçado a voltar ao módulo Tipos de Dados para criá-lo; daí a iteração entre esses módulos, que foi mencionada na seção 3.3.

3.3.3 Estruturação Cronológica

Em JSD a estruturação cronológica mostra a ordenação no tempo das ações de uma entidade; basicamente, é um diagrama que mostra a seqüência de ações, iteração de ações, e ações alternativas. Para cada entidade haverá um Diagrama de Estruturação cronológica (DE).

No AnEspO, este módulo permite a entrada do diagrama de estruturação cronológica, utilizando uma gramática simples. Essa gramática está baseada nos conceitos de seqüência, seleção, e iteração de Jackson, que são:

- **Seqüência:** é um componente de processo que consiste de duas ou mais partes, cada uma das quais ocorre uma vez a cada ocorrência da seqüência. Normalmente lê-se a seqüência da esquerda para a direita, em sua forma diagramática (um retângulo com o nome do componente).
- **Seleção:** é um componente de processo que consiste, da mesma forma que a seqüência, de duas ou mais partes, das quais ocorre apenas uma a cada ocorrência da seleção. Graficamente é denotada por retângulos com um círculo no canto superior direito.
- **Iteração:** é um componente de processo formado por uma parte que ocorre zero ou mais vezes a cada ocorrência da iteração. No diagrama gráfico é representada por retângulos com um asterisco no canto superior direito.

A forma gráfica destes três componentes está mostrada na figura 3.6.

A gramática utilizada é a mesma que [Ve86] usa no JSD- TOOL. A descrição dessa gramática será detalhada a seguir:

<COMANDO-SEQ> <— SEQ <nome-seqüência> : <nome-item>, <nome-item>;

<nome-seqüência> <— nome-entidade | nome-estrutura

<nome-item> <— nome-ação | nome-estrutura

Neste comando, <nome-seqüência>, é definido como uma seqüência de <nome-item>s. Se <nome-seqüência> é um “nome-estrutura” (isto é, um <nome-seqüência>, um <nome-seleção> ou um <nome- iteração>) que já se encontra em uso, então a nova definição a substitui.

Figura 3.6: NOTAÇÃO DE JACKSON PARA OS COMPONENTES DE UM DIAGRAMA DE ESTRUTURA

$\langle \text{COMANDO-SEL} \rangle \leftarrow \text{SEL } \langle \text{nome-seleção} \rangle : \langle \text{nome-item} \rangle$

$\{ \text{ALT } \langle \text{nome-item} \rangle \} [\text{ALT}];$

$\langle \text{nome-seleção} \rangle \leftarrow \text{nome-entidade} \mid \text{nome-estrutura}$

Neste comando $\langle \text{nome-seleção} \rangle$ é definido como uma seleção de $\langle \text{nome-item} \rangle$ s, com uma cláusula vazia no final do comando. Igual que no comando anterior, se $\langle \text{nome-seleção} \rangle$ é um "nome- estrutura" que já se encontra em uso, então a nova definição a substitui.

$\langle \text{COMANDO-ITER} \rangle \leftarrow \text{ITER } \langle \text{nome-iteração} \rangle : \langle \text{nome-item} \rangle;$

$\langle \text{nome-iteração} \rangle \leftarrow \text{nome-entidade} \mid \text{nome-estrutura}$

Neste comando $\langle \text{nome-iteração} \rangle$ é definido como um $\langle \text{nome-item} \rangle$.

Um exemplo de como essa gramática deverá ser utilizada para inserir um diagrama de estrutura no AnEspO é mostrado na figura 3.7 (b) correspondente ao DE da figura 3.7 (a). A estrutura inserida será checada sintaticamente, os nomes de entidades e ações serão conferidos na base de dados de especificação para, no final, armazenar um texto livre de erros. Adicionalmente este módulo têm uma opção de "ajuda" (HELP) contendo a descrição da gramática mencionada acima.

Figura 3.7: EXEMPLO DO USO DA GRAMÁTICA DO AnEspO

Os DEs são então inseridos como um texto utilizando essa gramática para, posteriormente, no EdTE, serem transformados em textos estruturados mais detalhados.

Na modelagem da metodologia JSD (figura A.1 do apêndice) foi incluído um atributo chamado "GRÁFICO", na entidade "DIAGRAMA DE ESTRUTURA", visando a permitir o armazenamento de uma estrutura gráfica dos DEs em versões posteriores do ambiente ADAO.

3.3.4 Modelo Inicial

No AnEspO este módulo está relacionado com a fase de construção do modelo inicial no método JSD, onde o modelo iniciado nas fases anteriores deve ser conectado com o mundo real. Cada ação ocorrida na realidade é recebida pelo sistema e uma operação é alocada para processar essa ação, fazendo com que o

sistema ajuste-se internamente ao que ocorreu no seu exterior.

No método JSD existem duas formas de conexão de processos que são usadas tanto para conectar os processos de modelagem ao mundo real como para conectá-los a processos internos ao sistema. Essas conexões são:

- **Conexão por seqüência de dados**, onde um processo escreve uma seqüência de dados, consistindo de um conjunto ordenado de mensagens ou registros e o outro processo da rede de processos lê essa seqüência.
- **Conexão por vetor de estados**, onde um processo lê um conjunto de variáveis locais pertencentes, internamente, ao outro processo da rede de processos.

Na fase do modelo inicial, inicia-se a construção do Diagrama de Especificação do Sistema (DES), contendo os processos de modelagem e estabelecendo suas conexões com os processos do mundo real. estabelecidas.

Graficamente, no DES, os processos são representados por retângulos; as conexões por seqüência de dados através de círculos; e as conexões por vetor de estados através de losangos. As flechas indicam o sentido da conexão.

No modelo inicial, o DES é composto por processos de dois níveis: nível zero e nível maior que zero. Os processos de nível zero recebem o nome da entidade seguido do sufixo “-0” e representam o objeto do mundo real que fornece as entradas ao sistema. Os processos de nível maior que zero são os correspondentes aos processos de modelagem que simulam o comportamento da entidade dentro do computador, eles são nomeados com o próprio nome da entidade, seguido do sufixo “-[número >0]”.

O objetivo do módulo “Modelo Inicial”, dentro do subsistema AnEspO, é permitir ao engenheiro de software inserir o modelo inicial descrito acima na base de especificação. Para realizar esse processo, o sistema pede ao usuário para escolher a entidade correspondente ao modelo inicial o qual deseja inserir. No final da transação o AnEspO gera os nomes dos processos do mundo real (sufixo 0) e do sistema (sufixo 1) e em seguida pede o tipo de conexão desses processos.

Como pode ser visto no diagrama de modelagem da metodologia (figura A.1 do apêndice A), as conexões entre processos correspondem a relacionamentos do processo de modelagem e do processo funcional com seqüências de dados e vetores de estado. É por isso que durante a inserção do modelo inicial,

o AnEspO pede o tipo de dados ao qual essas seqüências de dados ou vetores de estados pertencem.

3.3.5 Rede de Processos

Continuando com o Diagrama de Especificação do Sistema (DES), o módulo REDE DE PROCESSOS do AnEspO permite a inserção de outros processos de modelagem do sistema, assim como também de todos os processos que geram as saídas a serem produzidas pelo sistema, chamados em JSD de processos funcionais.

Para poder fazer a descrição deste módulo, alguns conceitos do método JSD devem ser introduzidos [Ja83]:

Com respeito às entradas do sistema e/ou processos:

- **MÚLTIPLAS ENTRADAS:** Um processo pode ter múltiplas entradas e múltiplas saídas, as quais podem ter instantes diferentes de chegada ou partida, respectivamente. As mensagens das seqüências de dados devem ser observadas de acordo com regras determinadas na especificação, mantendo uma ordem de leitura que melhor satisfaça ao sistema em desenvolvimento. O tratamento de múltiplas entradas por um único processo é chamado de **INTERCALAÇÃO FIXA (FIXED MERGE)** quando as leituras das seqüências de dados ocorre numa ordem predeterminada, ou então de **INTERCALAÇÃO POR DATA (DATA MERGE)**, quando duas seqüências são lidas de acordo com uma certa "data".
- **INTERCALAÇÃO NATURAL (ROUGH MERGE):** Este tipo de intercalação ocorre quando um processo recebe múltiplas entradas e as lê na ordem de chegada das mensagens, independentemente da ordem na qual elas foram escritas ou da velocidade do canal de comunicação. O processo que absorve as seqüências recebe as mensagens como se elas chegassem por uma única entrada. Graficamente, no DES, a intercalação forçada é representada por duas ou mais seqüências de dados cujas setas se unem no lado do processo que as absorve. A figura 3.8 mostra essa representação.
- **MARCADOR DE INTERVALO DE TEMPO:** para modelar uma ação que seja executada a partir de um instante de tempo ou mesmo por um período de tempo, JSD tem o conceito de **MARCADORES DE INTERVALO DE TEMPO**, que indicam a chegada, no mundo real, desse instante no tempo.

Figura 3.8: NOTAÇÃO USADA NO DES PARA INDICAR INTERCALAÇÃO NATURAL

Com respeito às saídas do sistema:

As saídas ou funções do sistema são adicionadas ao modelo como PROCESSOS FUNCIONAIS. Os processos funcionais recebem entradas dos processos de modelagem e produzem saídas em impressoras, terminais ou discos. Jackson introduz vários tipos de funções que podem ser modelados no DES:

- **FUNÇÃO EMBUTIDA:** quando a combinação dos eventos é simples e a computação da função é direta, as operações da função podem ser diretamente embutidas na estrutura do processo de modelagem.
- **FUNÇÃO IMPOSTA:** É uma função realizada por um processo funcional que obtém dados do modelo, inspecionando diretamente o vetor de estado do processo de modelagem.
- **FUNÇÃO INTERATIVA:** Ocorre quando o processo funcional consulta o vetor de estados de um processo de modelagem, e posteriormente a saída gerada será entrada desse ou de outro processo de modelagem.

A representação gráfica desses tipos de funções é mostrada na figura 3.9 (a), (b) e (c), onde M representa um processo de modelagem; F, um proceso

Figura 3.9: TIPOS DE FUNÇÕES EM JSD

funcional; MV a consulta ao vetor de estado do processo de modelagem; R um marcador de intervalo de tempo; S uma saída; MI a sequência de dados do processo de modelagem; e, MF, a sequência de dados do processo funcional.

O módulo REDE DE PROCESSOS engloba todos os conceitos descritos acima para manter consistência com as diretrizes de JSD.

Para cada processo de modelagem e funcional, no AnEspO, indicam-se todos os fluxos de seqüências de dados e/ou vetores de estados que entram e saem dele. Os fluxos de entrada são identificados como RECEBE SEQÜÊNCIA DE DADOS. Os fluxos de saída são chamados de GERA SEQÜÊNCIA DE DADOS. As inspeções aos vetores de estados são denominadas CONSULTA A VETOR DE ESTADOS. Dependendo do tipo de processo que está sendo inserido (Modelagem ou Funcional), o AnEspO pede a informação correspondente a cada caso. Por exemplo, se se trata de um processo funcional, e a opção escolhida foi RECEBE SEQÜÊNCIA DADOS, o AnEspO tem que saber se aquela seqüência de dados recebida é um marcador de intervalo de tempo ou não.

Neste módulo foram adotados alguns princípios básicos com respei-

to aos marcadores de intervalo de tempo e às seqüências de dados de saída. Assim, os marcadores de intervalo de tempo (MIT) poderão ser do tipo “CONSULTA”, “FIM DE SEMANA”, “ESTATISTICA”, etc., isto é, eles são limitados aos tipos de MIT fornecidos pelo sistema.

As seqüências de dados de saída (figura A.1 do apêndice A), que correspondem aos relatórios do sistema alvo, terão opção de escolher o tipo de formato com o qual serão mostrados na tela, no momento da execução. Esses formatos são VERTICAL E HORIZONTAL. Essa informação servirá para a prototipação do sistema. O formato vertical será utilizado principalmente para responder a consultas no terminal de vídeo, enquanto que o formato vertical será mais utilizado para emitir relatórios via impressora. No primeiro, os campos aparecerão dispostos horizontalmente, isto é, cada campo seguido de seu conteúdo, numa linha. No segundo, haverá um cabeçalho, contendo o título dos campos e em seguida aparecerão as várias linhas de detalhe.

3.3.6 Relatórios

Como foi mencionado anteriormente, o sistema AnEspO tem a disposição do engenheiro de software alguns relatórios, que podem servir como documentação do sistema, assim como para verificação de consistência da base. Inicialmente foram projetados quatro tipos de relatórios: Entidades e Ações, Ações e seus Atributos, Resumo da Rede de processos e Tipos de dados não definidos (correspondentes a tipos de dados que são identificadores de entidades). Posteriormente podem ser adicionados outros tipos de relatórios.

No módulo Executor da Especificação Operacional (ExEspO), outros tipos de inconsistências serão checadas e caso haja erros, outros relatórios serão emitidos.

3.4 Aspectos Principais da Especificação do AnEspO

Uma das maiores preocupações do Analisador de Especificações Operacionais é facilitar o trabalho do engenheiro de software na fase de especificação de um Sistema de Informação. Várias características próprias do AnEspO ajudarão a atingir esse objetivo.

Por ser uma ferramenta totalmente interativa (on-line), uma dessas características é a interface amigável projetada para a interface homem-máquina. O AnEspO conta com uma interface baseada em janelas, menus superpostos e formulários eletrônicos, que foi desenvolvida com a ajuda de um conjunto de rotinas para gerenciamento de interface com o usuário (TOOLKIT) desenvolvida por [Tr89a]. Dessa maneira, o AnEspO guia o engenheiro de software dentro das fases do método JSD.

Na medida que a informação é inserida na Base de Especificação (Base de dados Sistema), ela pode vir a aparecer em seguida num menu de escolha, evitando ao máximo que o engenheiro de software tenha que digitar informação que já se encontra armazenada e que pode causar erros de inconsistência. Por exemplo, para inserir um atributo será necessário saber a que entidade e ação ele pertence; e essa informação, por já estar inserida na base de especificações, será solicitada ao usuário mediante menus de escolha.

A manipulação da informação no AnEspO é padronizada. Para todos os módulos descritos na seção anterior existe a opção de: Inserir, Consultar, Modificar e Eliminar informação.

Para desenvolver a especificação dos requisitos do AnEspO foi utilizada a estratégia de prototipagem. Segundo [Bo84], essa estratégia ajuda na definição dos requisitos de um sistema, já que as necessidades do usuário são extraídas, apresentadas e desenvolvidas conforme a construção de um modelo de trabalho rápido e em contexto, para obter o sistema final. O protótipo desenvolvido foi um "PROTÓTIPO EM PAPEL" que mostrava passo a passo como funcionaria o AnEspO, mediante a sequência de telas do sistema.

Para que o protótipo fosse consistente se utilizou adicionalmente a ajuda de Diagramas de Transição de Estado, da maneira como foi sugerido por [Wa82]. Eles indicam a ordem em que as telas devem seguir e as ações que o sistema deve realizar entre a exibição de uma tela e outra. Um exemplo desses DTE é mostrado na figura 3.10 que define a sequência de telas e ações a serem executadas, do módulo Estruturação Cronológica. A especificação completa do sistema pode ser encontrada no documento de trabalho [Ro89].

A implementação do AnEspO foi feita utilizando a linguagem de programação C, (compilador C Aztec versão 4.1), em ambiente de microcomputadores tipo PC e sob o sistema operacional MS-DOS 3.1. Foi utilizado o núcleo da base de dados do Sistema Integrado para Produção de Software (Base de Dados MRO), como suporte de armazenamento da informação.

Figura 3.10: DIAGRAMA DE TRANSIÇÃO DE ESTADOS DO MÓDULO ESTRUTURAÇÃO CRONOLÓGICA DO AnEspO

3.5 Considerações Finais

Foi apresentada neste capítulo, uma pequena introdução ao Sistema Integrado para Produção de Software (SIPS), de maneira a ressaltar a utilização de modelos Meta para a especificação de ferramentas que apoiam o desenvolvimento de software.

O modelo de dados para o processo de criação de uma definição meta para a metodologia JSD, na qual se encontra baseado esse ambiente, foi descrito na seção 3.2.2.

Posteriormente foi apresentada a arquitetura do AnEspO como sendo constituída de cinco módulos principais, cada um dos quais foi devidamente relacionado com as etapas de JSD. Um resumo desses módulos é dado abaixo:

- **TIPOS DE DADOS:** este módulo permite ao engenheiro de software relacionar os tipos de dados com os atributos, vetores de estado e seqüências de dados do sistema alvo.
- **ENTIDADES-AÇÕES-ATRIBUTOS:** este módulo permite a entrada das entidades, ações e atributos do sistema em desenvolvimento e relaciona-se com a primeira etapa de JSD.
- **ESTRUTURAÇÃO CRONOLÓGICA:** em JSD a estruturação cronológica é representada por um diagrama que mostra a seqüência de ações, iteração de ações e ações alternativas. No AnEspO este módulo permite a entrada dessa estruturação cronológica no tempo, utilizando uma forma textual.
- **MODELO INICIAL:** Este módulo está relacionado com a terceira fase do método JSD. Ele permite que o modelo iniciado nas fases anteriores seja conectado ao mundo real usando conexões de seqüência de dados ou vetor de estados.
- **REDE DE PROCESSOS:** permite inserir outros processos de modelagem e completar as entradas do sistema alvo, assim como também as saídas que podem ser produzidas pelo sistema, chamadas em JSD de processos funcionais.

O módulo adicional RELATÓRIOS também foi descrito, de maneira a oferecer ao engenheiro de software uma ajuda na depuração da especificação do sistema alvo criada.

No final foram descritos alguns aspectos relevantes da especificação do AnEspO e os recursos computacionais utilizados para sua implementação.

Capítulo 4

Sobre o Protótipo Desenvolvido

4.1 Considerações Iniciais

Este capítulo tem o objetivo de demonstrar o protótipo do AnEspO desenvolvido para validar parte das idéias contidas no projeto ADAO, isto é, desenvolver um ambiente de software baseado em especificações operacionais, gerando inclusive, um protótipo do sistema alvo.

No capítulo anterior foram destacados alguns aspectos principais da especificação do AnEspO. Neste capítulo são destacados os pontos principais da implementação do protótipo desenvolvido, assim como decisões de projeto que levam a restrições do seu uso.

No restante do capítulo mostram-se quais devem ser os procedimentos iniciais para o uso do protótipo e apresenta-se um caso, onde a especificação de um sistema exemplo de pequeno porte é desenvolvida passo a passo, ilustrando a interface do protótipo.

4.2 Arquitetura do Protótipo

Como foi dito anteriormente, o AnEspO foi implementado utilizando-se a linguagem de programação C. A escolha desta linguagem deveu-se, basicamente, a que as rotinas de manipulação da Base de Dados MRO estavam implementadas em C. Além disso, C é uma linguagem de propósitos gerais com características de expressão, controle de fluxo e estruturas de dados econômicas.

Sua generalidade torna-a mais conveniente e eficiente para muitas tarefas do que outras linguagens supostamente mais poderosas [Ke78].

Para uma boa estruturação do sistema, ele foi dividido em vários módulos menores, de maneira a facilitar a implementação e a manutenção, melhorando assim a produtividade durante o desenvolvimento do protótipo por permitir a construção de componentes reusáveis. O programa principal do projeto ADAO chama todas as ferramentas que o compõem, dentre elas o AnEspO. O AnEspO, por sua vez, está encarregado de chamar todos seus componentes: Tipos de dados, Entidades, Ações, Atributos, Estruturação Cronológica, Modelo Inicial, Rede de Processos e Relatórios. Cada um desses componentes constitui um módulo na implementação do protótipo.

Como já foi citado anteriormente, a interface dos módulos do AnEspO está padronizada em quatro operações: inserir, consultar, modificar, e eliminar, que também constituem submódulos dos módulos mencionados no parágrafo anterior.

Para uma melhor visualização da arquitetura de implementação do AnEspO foi contruído um diagrama hierárquico envolvendo os principais processos do sistema. Esse diagrama é mostrado na figura 4.1. Adicionalmente, a figura 4.2 mostra o diagrama hierárquico do módulo Estruturação Cronológica, correspondente ao diagrama de transição de estados mostrado na figura 3.10 do capítulo anterior. Todos os outros módulos do AnEspO têm uma hierarquia de módulos similar à mostrada nessa figura.

Uma das limitações da implementação realizada é o fato de ter sido imposta a condição de que cada sistema alvo especificado tenha sua própria Base de dados de Especificação. Isso constitui uma restrição, pelo fato de que a Base de Dados MRO não restringe esse aspecto, isto é, seria possível ter várias instâncias de sistemas com a mesma definição da Base de Dados, mas isso envolveria um tipo de gerenciamento mais minucioso que não foi realizado neste primeiro protótipo. A implementação não impede que existam vários sistemas, mas simplesmente ela não controla que a informação de entidades, ações, etc, correspondam a diferentes sistemas.

O número de entidades, ações, atributos, processos de modelagem, processos funcionais, etc, que o sistema aceita, foi limitado a trinta elementos nesta implementação do protótipo, isto é, trinta entidades para o sistema, trinta ações para cada entidade, trinta atributos para cada ação, etc. Esse número poderia ser aumentado facilmente, tendo só como limite a capacidade do equipamento utilizado. Cabe mencionar que esta também não é uma limitação da Base

Figura 4.1: DIAGRAMA HIERÁRQUICO DOS PROCESSOS PRINCIPAIS DO AnEspO

de Dados MRO, mas sim do protótipo desenvolvido.

A limitação mencionada acima se deve ao fato de que os objetos devem estar colocados numa tabela; armazenados na memória principal antes de serem mostrados nas janelas (menus) da interface. Aumentando-se o tamanho dessas tabelas definidas estaticamente aumenta-se o número de objetos que podem ser manipulados pela interface.

No que se refere ao volume de programas, o protótipo foi desenvolvido com um total de aproximadamente 37 programas específicos e 35 rotinas de propósito geral. O programa executável tem um tamanho aproximado de 200 Kbytes.

4.3 Uso do Protótipo

Para que o projeto ADAO entre em execução é preciso ter definida a descrição da base de dados da especificação, onde será armazenada a especificação do sistema alvo.

Esta tarefa é realizada com a ajuda de uma ferramenta da Base de

Figura 4.2: DIAGRAMA HIERÁRQUICO DO MÓDULO ESTRUTURAÇÃO CRONOLÓGICA

Dados MRO chamada SIPS_G. Essa ferramenta cria uma base de dados e estabelece as pessoas que poderão ter acesso a ela, assim como a forma desse acesso. A interação dessa ferramenta com o gerente do projeto é feita a través de um diálogo, onde o gerente do projeto deve responder a perguntas que especificarão para a Base de Dados MRO algumas informações tais como:

- Nome da descrição a ser criada;
- Nome e código do gerente;
- A Base-Meta que será utilizada. Neste caso, a Base Meta contém a definição do método JSD que será usado para a especificação do sistema alvo e será chamada "JSD";
- Nome e código de outros participantes do grupo de projetistas, além do gerente;
- Grau de proteção quanto ao acesso a ferramentas do ambiente que cada participante possui.

A partir das respostas fornecidas a essas perguntas, a Base-Sistema ou Base da Especificação é inicializada, gerando um conjunto de arquivos que guardarão a informação de objetos, atributos, comentários, relacionamentos, etc.

Figura 4.3: MENU PRINCIPAL DO PROJETO ADAO

Esse conjunto de arquivos gerados pelo SIPS-G para guardar a informação do sistema alvo, deverá estar na mesma área de trabalho onde o AnEspO será executado.

A base de dados criada terá sempre que ser aberta e fechada, antes e depois de sua utilização. As ferramentas do projeto ADAO serão as encarregadas de fechar a base, sempre que saiam de execução. Enquanto que para a abertura haverá duas possibilidades de fazê-lo, na primeira opção o projetista entra com a informação de abertura: Nome da base, Nome do gerente e Código do gerente; na segunda opção o próprio ADAO, abre internamente a base, criando essas informações através do programa principal. Para começar a execução do AnEspO, basta digitar o nome do ambiente no qual se encontra inserido: "ADAO". Por ser parte de um ambiente completo para desenvolvimento de software, ele será ativado para execução quando a opção ESPECIFICAÇÃO do menu principal desse ambiente for escolhida, como é mostrado na figura 4.3.

Imediatamente após essa escolha, pede-se o nome do sistema que está sendo desenvolvido e informações gerais tais como: Descrição do Sistema e Responsável. Posteriormente aparece o menu principal do AnEspO, o qual ativa os diferentes módulos da ferramenta, que foram descritos no capítulo anterior. As opções do menu principal do AnEspO são: Tipo de dados, Entidades, Ações, Atributos, Estruturação Cronológica, Modelo Inicial, Rede de Processos e Relatórios. A figura 4.4 mostra como esse menu aparece na tela.

A escolha de qualquer uma dessas opções, (com exceção da opção Relatórios, que tem outro tipo de tratamento), ativará o menu de operações do AnEspO: Inserir, Consultar, Modificar e Eliminar. A partir daí a interação com o engenheiro de software será feita dependendo das necessidades de cada módulo,

Figura 4.4: MENU PRINCIPAL DO AnEspO

utilizando sempre mensagens que o alertarão sobre a informação a ser manipulada. A interface amigável, sob a qual foi desenvolvido este protótipo permite a que ele seja bastante fácil de ser utilizado, sem necessidade de maior treinamento.

4.4 Um Caso

O exemplo escolhido para mostrar o processo de especificação de sistemas usando a ferramenta AnEspO desenvolvida é de um “Videoclube de médio porte”, extraído de [Ms89a], e que será descrito sucintamente a seguir:

“Os membros do videoclube se inscrevem mediante o pagamento inicial de uma módica quantia e o preenchimento de uma ficha cadastral. Após isso estão aptos a retirarem quantas fitas quiserem, desde que uma de cada vez, pagando um preço fixo diário por fita retirada. O objetivo do Sistema de Informação a ser projetado é controlar o pagamento dos clientes e a situação de cada fita”.

Com base nas características do videoblube, será mostrada a solução adotada por Masiero. À medida que a especificação vai sendo desenvolvida

seguindo as diferentes etapas do método JSD, serão apresentadas algumas telas do AnEspO, correspondentes à mesma fase da especificação, mas utilizando o protótipo desenvolvido.

PASSO DAS ENTIDADES, AÇÕES E ATRIBUTOS

As entidades consideradas como relevantes no sistema do videoclube são: MEMBRO e FITA. A tabela 4.1 mostra a lista dessas entidades, de suas ações e dos atributos associados às ações.

No AnEspO, o passo das entidades, ações e atributos é realizado em três etapas, pois será necessário entrar primeiro com as entidades do sistema, para posteriormente poder entrar com suas ações respectivas. Da mesma maneira, será necessário ter armazenadas na base as ações, para poder entrar com seus respectivos atributos. Isto ocorre devido a relação existente entre entidades, ações e atributos, e a interface foi projetada dessa forma para garantir a consistência da especificação que está sendo inserida, isto é, não se pode inserir uma ação sem que ela esteja associada a uma entidade, um atributo deve estar associado a uma ação, etc.

ENTIDADE	AÇÃO	ATRIBUTO
MEMBRO	INSCREVER	- identificação do membro - nome do membro - endereço, etc.
	ALUGAR	- identificação do membro - identificação da fita - total devido data-aluguel
	PAGAR	- identificação do membro - data do pagamento - valor do pagamento, etc.
FITA	COMPRAR	- identificação da fita - título - volumes - data da compra - resumo do filme, etc.
	SAIR	- identificação do membro - identificação da fita - data da saída
	RETORNAR	- identificação da fita - data do retorno
	DESCARTAR	- identificação da fita - data do descarte - motivo

TABELA 4.1 ENTIDADES, AÇÕES E ATRIBUTOS DO SISTEMA DO VIDEOCLUBE

No AnEspO será requerida informação adicional à da tabela 4.1. No caso das entidades será pedida a descrição da entidade e seu identificador. Um exemplo de como aparecerá essa informação na operação de consulta da entidade FITA é mostrada na figura 4.5.

As ações guardam como, informação adicional, sua descrição. Da mesma forma, os atributos armazenam o seu tipo de dados e seu domínio (contínuo, discreto, ou alfanumérico). A figura 4.6 (a), (b), (c) e (d) mostra a sequência de telas necessárias para se realizar a inserção de um atributo da entidade FITA

* ADAO *	CONSULTAR ENTIDADES
SISTEMA = videoclube	
Nome : Fita	
Descricao: fitas disponiveis no videoclube para serem alugadas	
Identificador: id-fita	
Continuar	Retornar

Figura 4.5: INFORMAÇÃO DA ENTIDADE FITA

de maneira a que o leitor possa perceber como estão relacionadas as informações das entidades, ações e atributos, dentro do AnEspO.

PASSO DA ESTRUTURAÇÃO CRONOLÓGICA

O Diagrama de estruturação Cronológica das ações ou Diagrama de Estrutura (DEs), mostra a ordenação no tempo das ações das entidades MEMBRO e FITA. A figura 4.7 mostra o DEs para essas entidades.

No AnEspO o passo da estruturação cronológica é realizado usando a forma equivalente para a representação dos DEs, que é o texto estruturado. No capítulo três foi apresentada a gramática utilizada para a inserção deste texto. A figura 4.8 mostra como aparece esse texto na operação de consulta à Estruturação Cronológica.

PASSO DE CONSTRUÇÃO DO MODELO INICIAL

Neste passo são criados os processos de modelagem que vão simular, no computador, o comportamento das entidades especificadas. Além dos processo são definidas as conexões entre eles e o mundo real.

* ADAO *
INSERIR ATRIBUTOS

SISTEMA = videoclub

Membro
(1)(2)
* Retornar *

(a)

* ADAO *
INSERIR ATRIBUTOS
Entidade=Fita

SISTEMA = videoclub

COMPRAR
Sair
Retornar
Descartar

(b)

Figura 4.6: SEQÜÊNCIA DE TELAS PARA A INSERÇÃO DE UM ATRIBUTO DA AÇÃO COMPRAR DA ENTIDADE FITA

* ADAO * INSERIR ATRIBUTOS Entidade=Fita	
SISTEMA = videoclube	
Nome : preco de compra	<div> ano cep dia mes rua char data value </div>

(c)

* ADAO * INSERIR ATRIBUTOS Entidade=Fita	
SISTEMA = videoclube	
Nome : preco de compra	<div> Contínuo Discreto Alfanumerico * Retornar * </div>
Tipo : value	
ESCOLHA O DOMINIO	

(d)

Figura 4.6: CONTINUAÇÃO ...

Figura 4.7: DIAGRAMA DE ESTRUTURAÇÃO CRONOLÓGICA DAS AÇÕES MEMBRO E FITA

* ADÃO *	CONSULTAR ESTRUTURA	Entidade=FitaA
SISTEMA = videoclub		
Nome da Estrutura : Estrutura fita SEQ Fita: Comprar, Corpo uso, Descartar; ITR: Corpo uso: Aluguel; SEQ: Aluguel: Sair, Retornar;		
Continuar		Retornar

Figura 4.8: TEXTO ESTRUTURADO PARA A ENTIDADE FITA

Figura 4.9: DIAGRAMA DE ESPECIFICAÇÃO DO SISTEMA DO VIDEOCLUBE, NO PASSO DO MODELO INICIAL.

No sistema de videoclube são criados os processos FITA-1 e MEMBRO-1. A entidade FITA, (ou o processo correspondente FITA-0), comunica-se com o processo FITA-1 através de uma conexão por seqüência de dados, cada vez que ela sofre ou executa uma ação. Da mesma forma, a entidade MEMBRO, comunica-se com o processo de modelagem através de uma conexão por seqüência de dados.

O Diagrama de Especificação do Sistema (DES) definido para o sistema de videoclube, no passo do modelo inicial, é mostrado na figura 4.9.

Da mesma maneira, esse modelo inicial será inserido no AnEspO. O nome do processo de modelagem, também chamado de processo do sistema no protótipo, será gerado automaticamente, adicionando o sufixo “ -1 ” ao nome da entidade correspondente. Posteriormente será pedido o tipo de comunicação (seqüência de dados, ou vetor de estados) e as informações necessárias para essa comunicação, isto é, nome da comunicação, descrição e tipo de dados. A figura 4.10 mostra o modelo inicial da entidade FITA da maneira como essa informação estará disposta na tela.

PASSO DE CONSTRUÇÃO DA REDE DE PROCESSOS

Figura 4.10: MODELO INICIAL DA ENTIDADE FITA

Neste passo deve-se completar o modelo inicial do sistema com os processos funcionais. Continuando com o exemplo do videoclube, pode-se imaginar que os usuários solicitaram os três relatórios seguintes:

- Produzir automaticamente as notas de aluguel
- Informar a situação de cada fita, isto é, se está fora ou se está disponível para alugar.
- Emitir um relatório semanal com a estatística das fitas alugadas durante a semana.

O Diagrama de Especificação do Sistema, mostrado na figura 4.11, reúne as três funções especificadas e adicionadas ao modelo inicial.

A seguir são comentados as decisões tomadas para a construção desse diagrama.

Produção automática de notas de aluguel

Quando uma fita sai do videoclube, assinalando que um aluguel está em curso, o processo FITA-1 deve enviar uma mensagem referente à ação ALUGAR, para o processo MEMBRO-1. Este, ao receber a mensagem, deve registrar os dados referentes a essa ação, tais como: valor a pagar, data do aluguel, etc. Dessa forma, a operação ALUGAR deixa de passar pelo canal de comunicações

Figura 4.11: DES DO SISTEMA DO VIDEOCLUBE

que liga o mundo real ao processo FITA-1 (MEM na figura 4.9) e passa a ser gerada automaticamente por FITA-1, após cada ação SAIR.

O processo FITA-1 deve consultar o registro dos dados, (chamado em JSD de Vetor de Estados) de MEMBRO-1, para saber o nome e endereço que deverá constar da nota de aluguel.

Essa função, por ser bastante simples, foi embutida dentro do processo de modelagem FITA-1. Na figura 4.11 pode ser observada a inclusão da consulta ao vetor de estado de MEMBRO-1, do envio da mensagem RAL (registro de aluguel) ao processo MEMBRO-1, e do processo funcional “Emitir Nota de Aluguel”, embutido dentro de FITA-1.

Consulta a situação da fita

Quando o usuário deseja realizar uma consulta, deve informar ao sistema através de algum sinal. No caso de um sistema manual, a própria existência da ficha externa da fita no painel, indica que ela se encontra disponível para aluguel. Num sistema automatizado, o sinal seria um fluxo de dados vindo externamente ao sistema e indicando o desejo de ativar o processo que realiza a consulta. Este consulta ao vetor de estado do processo de modelagem FITA-1 e retorna a mensagem apropriada.

O sinal que vem externamente para o sistema e a seqüência de dados CO da figura 4.11 que indicará a requisição de consulta, e que por ser um sinal externo corresponde a um marcador de intervalo de tempo.

Geração de estatísticas semanais

Esta função difere das outras porque ela deve se comunicar com o processo de modelagem FITA-1 através de uma seqüência de dados (Figura 4.11). Isso ocorre porque é FITA-1 quem toma a iniciativa de comunicar que uma fita foi retirada ou devolvida. Só ele possui essa informação. Ao receber um fluxode dados ALU, o processo funcional Preparar-Estatística-Semanal acumula os totalizadores apropriados. Quando pelo mesmo canal, aparece a informação de que é um Fim de semana (FS), a estatística é enviada para o mundo real através de uma mensagem de saída.

A intercalação de FS e ALU indica que o processo preparar estatística semanal recebe os dois fluxos por um mesmo canal de entrada, na ordem em que aparecerem. FS indica que os totais acumulados devem ser emitidos e reinicializados para os próximos ALUs. A diferença entre a comunicação por seqüência de dados, nesta função e por vetor de estado na outra, deve-se ao fato de que nesta a decisão de enviar o fluxo está em FITA-1 ao passo que na anterior a decisão de consultar está no processo funcional.

No AnEspO este passo pode ser realizado da mesma maneira como foi descrito acima, isto é, acrescentando a rede de processos função por função. Difere da forma gráfica apresentada na figura 4.11, porque o engenheiro de software terá que indicar o fluxo das seqüências de dados e/ou vetores de estado que entram e saem de cada processo de modelagem ou funcional, em forma textual, isto é, usando as seguintes opções do AnEspO: Recebe Seqüência de Dados, Gera Seqüência de Dados e Consulta Vetor de Estados, como foi descrito no capítulo anterior. A figura 4.12 mostra o menu de escolha correspondente a esses tipos de fluxos no módulo rede de processos do AnEspO. As figuras 4.13 e 4.14 mostram a operação de consulta correspondente ao processo FITA-1 e ao processo funcional Preparar estatística semanal, respectivamente, para indicar como essa informação está guardada na base.

Adicionalmente aos módulos correspondentes aos quatro passos do método JSD, o AnEspO fornece ao engenheiro de software o módulo de RELATÓRIOS, como, foi descrito no capítulo anterior. Eles facilitam a documentação do sistema e o controle de consistência da base com a especificação do sistema alvo. A figura 4.15 mostra uma parte do relatório de Entidades e Ações do sistema do videogame, emitido pelo AnEspO.

* ADAO * INSERIR REDE DE PROCESSOS Processo = Fita-1
SISTEMA = videoclub
<div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: 80%;"> <p style="text-align: right; margin: 0;">Processo de Modelagem</p> <div style="border: 1px solid black; padding: 5px; margin: 5px auto; width: 80%;"> <p>Recebe Seq. de Dados</p> <p>Gera Seq. de Dados</p> <p>Consulta Vetor de Estados</p> <p>Intercalacao de Seq Dados</p> <p>* Retornar *</p> </div> </div>

Figura 4.12: MENU DE ESCOLHA DOS TIPOS DE FLUXOS DE DADOS PARA CONSTRUÇÃO DA REDE DE PROCESSOS

* ADAO * CONSULTAR REDE DE PROCESSOS		
SISTEMA = videoclub		
<div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: 90%;"> <p style="text-align: center; margin: 0;">INFORMACAO GERAL</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 60%; vertical-align: top;"> <p>Processo: Fita-1</p> <p>Consulta Vetor de Estados: END</p> <p>Gera Seq. Dados: RAL</p> <p>Gera Seq. Dados: Nota aluguel</p> <p>Gera Seq. Dados: ALI</p> <p>Recebe Seq. Dados: FIL</p> </td> <td style="width: 40%; vertical-align: top; text-align: right;"> <p>* Processo de Modelagem *</p> </td> </tr> </table> </div> <div style="text-align: center; margin-top: 10px;"> <div style="border: 1px solid black; display: inline-block; padding: 2px 10px;">Continuar</div> <div style="border: 1px solid black; display: inline-block; padding: 2px 10px; margin-left: 20px;">Retornar</div> </div>	<p>Processo: Fita-1</p> <p>Consulta Vetor de Estados: END</p> <p>Gera Seq. Dados: RAL</p> <p>Gera Seq. Dados: Nota aluguel</p> <p>Gera Seq. Dados: ALI</p> <p>Recebe Seq. Dados: FIL</p>	<p>* Processo de Modelagem *</p>
<p>Processo: Fita-1</p> <p>Consulta Vetor de Estados: END</p> <p>Gera Seq. Dados: RAL</p> <p>Gera Seq. Dados: Nota aluguel</p> <p>Gera Seq. Dados: ALI</p> <p>Recebe Seq. Dados: FIL</p>	<p>* Processo de Modelagem *</p>	

Figura 4.13: REDE DE PROCESSOS DE FITA-1

* ADÃO *		CONSULTAR REDE DE PROCESSOS	
SISTEMA = videoclube			
INFORMAÇÃO GERAL			
Processo: Estatística		* Processo Funcional *	
Gera Seq. Dados: Est. Semanal			
Intercala: FS			
Intercala: ALU			
Recebe Seq. Dados: FS			
Recebe Seq. Dados: ALU			
Continuar		Retornar	

Figura 4.14: REDE DE PROCESSOS DE PREPARAR ESTATÍSTICA SEMANAL

SISTEMA: videoclube

ENTIDADE: Fita

IDENTIFICADOR: id-fita

DESCRIÇÃO: fitas disponíveis no videoclube para serem alugadas

A C A O : Comprar

DESCRIÇÃO: O dono do videoclube compra uma nova fita

A C A O : Sair

DESCRIÇÃO: Uma fita sai do videoclube para ser alugada

A C A O : Retornar

DESCRIÇÃO: A fita retorna ao acervo do videoclube

A C A O : Descartar

DESCRIÇÃO: A fita não está mais disponível para aluguel

Figura 4.15: RELATÓRIO PARCIAL DA INFORMAÇÃO DE ENTIDADES E AÇÕES DO SISTEMA DO VIDEOCLUBE

Toda a informação da especificação armazenada na base, pode também ser obtida usando a operação de CONSULTA nas diferentes etapas do sistema.

Para o caso da operação de ELIMINAÇÃO, o engenheiro de software deve estar ciente do impacto da remoção da informação da base, devido a associação entre os objetos da base, tais como entidades, ações, atributos, tipos de dados, etc. Por exemplo, a remoção de uma entidade implica na remoção de todos os relacionamentos dessa entidade.

4.5 Considerações Finais

Neste capítulo foi apresentada a arquitetura do protótipo do AnEspO, mostrando como ele foi implementado em termos de módulos de programas. São também apresentadas algumas limitações do protótipo, salientando os motivos pelos quais essas limitações ocorrem.

Posteriormente, foram descritos os passos que devem ser seguidos para iniciar a sessão de trabalho com o Analisador de Especificações Operacionais, assim como a utilização da ferramenta SIPS.G que gerará a descrição da base sistema onde será armazenada a especificação do sistema alvo.

Finalmente, foi descrito o exemplo de um Videoclube, mostrando a especificação do sistema seguindo os passos do método JSD e ao mesmo tempo ilustrando os passos correspondentes aos módulos do AnEspO.

Capítulo 5

Conclusões Finais

5.1 Avaliação do AnEspO

Este trabalho teve como objetivo principal mostrar a viabilidade de se desenvolver um Analisador de Especificações Operacionais para dar apoio ao engenheiro de software na fase de especificação do desenvolvimento de sistemas.

Para atingir esse objetivo, foi estudado o método de Jackson (Jackson System Development - JSD), considerado pela literatura técnica especializada como um método com passos bem definidos na condução da especificação à implementação, usando uma abordagem operacional.

Foi também feito um levantamento dos esforços de desenvolvimento de ferramentas de apoio por computador ao desenvolvimento de software (CASE), baseadas no método JSD e outros métodos correlatos, de maneira a perceber a importância delas no âmbito do desenvolvimento de sistemas.

O AnEspO é uma ferramenta do tipo CASE que se encontra inserida num ambiente maior chamado Ambiente de Desenvolvimento para Abordagem Operacional (ADAO). O projeto ADAO é um ambiente que integra várias ferramentas para apoiar parte do ciclo de vida de desenvolvimento de sistemas: desde a fase de especificação até a implementação de sistemas de informação.

Para o desenvolvimento do protótipo do AnEspO, foi inicialmente preparado o modelo de dados do método JSD, de maneira a possibilitar a criação da base de Dados Meta onde se encontra armazenada a definição da metodologia, que será utilizada por todas as ferramentas do projeto ADAO.

Segundo [Ms89b], a idéia de se utilizar uma base de dados META para armazenar informações a respeito da especificação de software não é nova. O projeto ISDOS foi um de seus precursores com o desenvolvimento dos sistemas PSL/PSA [Te77] e SEM [Te80]. No Brasil, essa idéia foi utilizada no sistema SACDS [Tr82] e depois no núcleo de base de dados do Sistema de Integração para Produção de Software (SIPS), desenvolvido no CTI [Tr86], e utilizado para a elaboração deste trabalho.

Sob essa Base de Dados Meta foi desenvolvido o AnEspO, como uma ferramenta para armazenar especificações operacionais e que posteriormente será utilizada para a geração do protótipo executável do sistema alvo. O engenheiro de software também poderá ter a sua disposição relatórios de consistência e completude da especificação, de maneira a conseguir uma especificação favorável a suas necessidades.

A interface amigável com a qual foi desenvolvido este protótipo facilita sua utilização, mas é importante salientar que uma interface gráfica para os diagramas de estruturação cronológica e da rede de processos seria muito mais amigável. Esforços nesse sentido estão sendo iniciados, para o desenvolvimento de um editor gráfico.

A implementação realizada foi testada com testes de casos individuais para cada programa ou conjunto de programas e também com um caso completo de um sistema de informação para um videoclube, como foi apresentado no capítulo 4.

Acreditamos que o trabalho atingiu seu objetivo na medida de demonstrar a viabilidade de desenvolver uma ferramenta baseada na abordagem operacional de JSD, e que contribuiu para facilitar o trabalho do engenheiro de software na etapa de especificação, que geralmente é feita com técnicas manuais. O desenvolvimento de sistemas de informação usando a técnica de protomodelagem [Bo84], faz com que a barreira entre o usuário final e o engenheiro de software seja quebrada, devido ao tempo de resposta menor com que os resultados são obtidos. Dispor, então, de uma especificação automatizada, para gerar esse protótipo do sistema final é muito importante. Logicamente, esta ferramenta será mais interessante quando as outras ferramentas do projeto ADAO estiverem integradas, já que com isso poderá ser completado o objetivo do projeto.

Os testes realizados com o sistema de videoclube mostraram que o tempo de resposta do sistema é bastante satisfatório, mas outros testes com a base carregada devem ser realizados.

5.2 Sugestões para a Evolução do AnEspO e outras Pesquisas

Como foi dito na seção anterior, um ponto merece consideração especial para futuras pesquisas: o desenvolvimento de uma interface gráfica para os diagramas de estruturação cronológica e especificação do sistema utilizados no método JSD. O diagrama de implementação não está sendo considerado porque o objetivo do projeto ADAO é gerar um protótipo do sistema alvo, sem passar pela fase de implementação de JSD. Essas idéias estão descritas em [Ms89b].

A implementação de mais relatórios, é também interessante, para fortalecer a documentação da especificação do sistema alvo e oferecer mais opções ao engenheiro de software.

Logicamente, a integração com as outras ferramentas do projeto ADAO, como o Editor de Textos Estruturados, o Executor da Especificação Operacional e o Sistema gerenciador da Interface da Aplicação, é de muita importância para conseguir complementar o objetivo do projeto, isto é, ser um ambiente de apoio a parte do ciclo de vida de desenvolvimento de software.

Com relação especificamente ao AnEspO, outros casos de teste devem ser desenvolvidos com o seu apoio para fazer uma validação mais abrangente do ferramental desenvolvido e, se for necessário, realizar modificações ou melhorias. Isso deverá ser feito com o uso da ferramenta em cursos de graduação no ICMSC, onde serão desenvolvidos projetos com o auxílio do AnEspO.

Bibliografia

- [Ak89] AKHRAS, N.F. - Meta-Modelos para Ferramentas Genéricas Integráveis ao Ambiente SIPS .*III Simpósio Brasileiro de Engenharia de Software*, Recife, PE, 1989, pp 113-122.
- [Am88] AMBROSIO, A.M. - Um sistema para execução de especificações JSD, *Dissertação de Mestrado*, Instituto de Pesquisas Espaciais, 1988.
- [Be81] BERGLAND, G.D. - Software Design Strategies (Tutorial) , *IEEE Computer Society*, Catalog 389, 1981
- [Bo84] BOAR, B. - *Application Prototyping* , Wiley Interscience, 1984.
- [Ca81] CAMERON, J. R. - Two Pairs of Examples in the Jackson Approach to System Development . *In: Bergland, G.D. e Gordon, R.D. (EDS). Software Design Strategies (Tutorial)*, IEEE Computer Society, Catalog 389, 1981, pp 155-175.
- [Ca86] CAMERON, J.R. - An Overview of JSD , *IEEE Transaction on Software Engineering*, vol. SE-12, february 1986, pp 222-240.
- [Ca88] CAMERON, J. R. - The modelling phase of JSD , *Information and Software Technology*, vol. 30, no. 6, july/august, 1988, pp 373-383.
- [Co81] COTTERMAN, W.W.; et all - *Systems Analysis and Design: A Foundation for the 1980's* . New York, North-Holland, 1981.
- [Ch76] CHEN, P. - The Entity-Relationship Model: Towards a Unified View of data , *ACM Trans. on Database Systems*, January 1976, pp 9-36.
- [Da87] DART,S.A.; ELLISON,R.J.; FEILER,P.H. and HABERMANN, A.N. - Software Development Environments , *Computer*, november 1987, pp 18-28.
- [DeM78] DEMARCO, T. - *Structured Analysis and System Specification* , Yourdon Inc., New York, USA, 1978.

- [En89] ENGMANN, R. & VAN HOEVE, F. - Generation of File Processing Programs based on JSP . *Software Practice and Experience*, Vol 19 (4), April 1989, pp 393 - 409.
- [Fe89] FERG, S. - *A New JSD CASE Tool*, Jackson Methods Newsletter, no 7, April 1989, pp 1-2.
- [Ga76] GANE,C. & SARSON,T. - *Structured Systems Analysis: Tools and Techniques* . Prentice-Hall, Inc, Englewoods Cliffs, NJ, (1979, 1a. ed. 1976).
- [Ho78] HOARE, C.A.R., - Communicating Sequential Processes , *Communications of the ACM*, vol 21 no. 8, august 1978, pp 666-677.
- [Ja75] JACKSON, M. - *Principles of Program Design* . Academic Press, 1975.
- [Ja78] JACKSON, M. - Information Systems: Modelling, Sequencing and Transformations . *In: Proceedings of the Third International Conference on Software Engineering. IEEE*, 1978. Reprinted in: Bergland G.D. e Gordon, R.D. (EDS) *Software Design Strategies (Tutorial)*, IEEE Computer Society, Catalog 389, 1981, pp 139-153.
- [Ja81] JACKSON, M. - Some Principles Underlying a System Development Method . *In: Systems Analysis and Design: A Foundation for 1980's*. W.W. Cotterman et all (EDS). North-Holland, pp 185-198.
- [Ja83] JACKSON,M. - *System Development* , Prentice-Hall, 1983.
- [Ke78] KERNIGHAN,B.W. & RITCHIE,D.M. - *The C Programming Language*. Prentice-Hall,1978.
- [Ma86] MACDONALD, I.G., Information Engineering, An Improved, Automatable Methodology for the Design of Data Sharing Systems. *In: OLLE,T.W et alli (EDS), Information Systems Design Methodologies: Improving the Practice*, North-Holland, 1986, pp 173-224.
- [Mc81] McCRACKEN, D.D. & JACKSON, M.A. - A Minority Dissenting Position . *In: Systems Analysis and Design: A Foundation for 1980's*. W.W. Cotterman et all (EDS). North-Holland, pp 551-553.
- [McC88] McCLURE, C. - The CASE for Strutured Development , *PC Tech. Journal*. Vol 6 (8), August 1988, pp 51-67.
- [Me86] MELLOR, J.S. & WARD, T.W. - *Structured Development for Real-Time Systems* . Yourdon Inc. New York, N.Y., 1986.

- [Ms88] MASIERO, P.C. & GERMANO, F.S.R. - JSD as an Object Oriented Design Method . *ACM Sigsoft Software Engineering Notes*. Vol 12(3), July 1988, pp 22-23.
- [Ms88a] MASIERO, P.C. - O Processo de Instânciação do Sistema SEM para Criação de um Ambiente de Apoio a um Método de Desenvolvimento de Sistemas . *II Simpósio Brasileiro de Engenharia de Software*, outubro de 1988, Canela, RS, pp 135-147.
- [Ms89a] MASIERO, P.C. - Uma visão geral do método JSD, *Notas Didáticas do ICMSC*, USP, no. 3, 1989.
- [Ms89b] MASIERO, P.C & RODRIGUEZ, M. - *Um Ambiente de Desenvolvimento baseado na Abordagem Operacional* , (Em preparação), 1989.
- [Pe84] PENTEADO, R.A. - Sobre uma Abordagem Operacional para Desenvolvimento de Sistemas de Informação . *Dissertação de mestrado*, ICMSC, Universidade de São Paulo, 1984.
- [Pr87] PRESSMAN, R.S. - *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 1987.
- [Ro89] RODRIGUEZ, M. - *Especificação de um Analisador de Especificações Operacionais* , documento de trabalho no 1, 1989.
- [Si88] - *SIPS Manual de Referência* . III Escola Brasileira Argentina de Informática. Curitiba, 1988.
- [Su88] SUTCLIFFE, A. *Jackson System Development*. Prentice Hall International (UK) Ltd., 1988.
- [Te77] TEICHROEW, D. & HERSHEY, E. - PSL/PSA: A Computer-Aided Technique for Structures Systems , *IEEE Transactions on Software Engineering* - Vol. SE-3 (1), 1977, pp 41-48.
- [Te80] TEICHROEW, D., MACASOVIC, P.; HERSHEY, E.A.; YANAMOTO, Y. - Application of the Entity-Relationship Approach to Information Processing System Modeling. In CHEN, P.P. (ED). *Entity Relationship Approach to Information System Analysis and Design*, North Holland, 1980, pp 15-38.
- [Tr82] TRAINA Jr., C. - Sistema de Apoio por Computador à Documentação de Sistemas . *Dissertação de mestrado*, ICMSC, Universidade de São Paulo, 1982.

- [Tr86] TRAINA Jr., C. et al. - SIPS - Estado Atual de Desenvolvimento . *XIX Congresso Nacional de Informática*, agosto, 1986, pp 297-306.
- [Tr89] TRAINA Jr., C. e SLAETS, J.F.W - *Um modelo de representação de objetos* , (Em preparação), 1989.
- [Tr89a] TRAINA Jr., C. - *Rotinas de Acesso a Console do Microtec XT2002* , (Em preparação), 1989.
- [Ve86] VELASCO, F.R.D. - JSD - Tool Reference Manual . *Tech. Report 86-08*, Wang Institute of Graduate Studies, Tyngsboro, MA., 1986.
- [Wa82] WASSERMAN , A.I. - The User Software Engineering Methodology: An Overview . *Proc. IFIP Working Group 8.1 Conf. Comparative Reviews of Information Systems Design Methodologies*, IFIP, Genova, 1982, pp 591-628.
- [Wi84] WIENER, R. & SINCOVEC, R. - *Software Engineering with Modula-2 and ADA* . John Wiley & Sons, 1984.
- [Yo78] YOURDON, E. & CONSTANTINE, L. - *Structured Design* , Yourdon Press, 1978.
- [Za84] ZAVE, P. - The Operational Versus The Conventional Approach to Software Development. *Communications of the ACM*, 27(2), february, 1984, pp 104-118.

Apêndice A

O Modelo de Dados JSD

Neste apêndice apresenta-se o Diagrama Entidade Relacionamento completo, correspondente ao método JSD que foi descrito no capítulo três, conjuntamente com o dicionário de dados correspondente.

Em seguida, apresenta-se o Diagrama de Representação de Objetos (DRO), para o qual foi mapeado o diagrama anterior, como ajuda na codificação da linguagem do Interpretador Meta.

Finalmente, mostram-se os relatórios emitidos pelo Interpretador Meta depois da criação da definição da metodologia JSD na Base de Dados Meta.

A.1 Dicionário de Dados da Modelagem do Método JSD

AÇÕES :

É algum evento que ocorre no mundo real o qual o sistema deve ser informado, causando uma modificação atômica na base de dados.

- **NOME:** identificação da ação
- **DESCRIÇÃO:** descreve a função da ação com respeito à entidade correspondente.

ATRIBUTOS:

São as características de cada ação.

- **NOME:** identificação do atributo
- **TAMANHO:** corresponde ao tamanho do atributo para o caso de tratar-se de um atributo relacionado com um tipo de dados "Character".
- **DOMÍNIO:** guarda o tipo de domínio (discreto ou contínuo), do atributo.
- **VALOR MÁXIMO:** guarda o valor máximo do atributo para o caso de um domínio contínuo.
- **VALOR MÍNIMO:** guarda o valor mínimo do atributo para o caso de um domínio contínuo.

DIAGRAMA DE ESTRUTURA:

É um diagrama que representa o diagrama de estrutura de uma sequência de dados ou um texto de processo.

- **ID:** nome que identifica o diagrama de estrutura dentro do AnEsp. Esse nome será de uso interno do sistema já que na realidade cada diagrama corresponde a uma entidade.
- **TEXTO:** Texto do diagrama de estrutura que será fornecido pelo Engenheiro de Software em uma linguagem pré-definida.

- **GRÁFICO:** Campo previsto para guardar o diagrama de estrutura em uma forma gráfica, em versões posteriores do ambiente ADAO.

ENTIDADES:

São objetos do mundo real que executam ou sofrem um conjunto de ações ordenadas no tempo.

- **NOME:** identificação da entidade
- **DESCRIÇÃO:** descreve a função que desempenha a entidade dentro do sistema alvo.

PROCESSO DE MODELAGEM:

É um processo sequencial que modela os acontecimentos do mundo real.

- **NOME:** identificação do processo de modelagem
- **DESCRIÇÃO:** descreve, na forma de um texto, o processo do mundo real.
- **CONJUNTO DE INSTRUÇÕES:** são todas as instruções de entrada e saída (Read e Write) que estão envolvidas no processo de modelagem.

PROCESSO FUNCIONAL:

Constitui um processo que é somado ao sistema com o propósito de produzir saídas.

- **NOME:** é o nome que identifica o processo funcional
- **ESPECIFICAÇÃO:** É o texto estruturado do processo funcional, descrevendo seu comportamento.
- **CONJUNTO DE INSTRUÇÕES:** são todas as instruções de entrada e saída (Read e Write) que estão envolvidas no processo funcional.
- **DESCRIÇÃO:** é uma descrição do processo funcional fornecida pelo usuário.

SEQÜÊNCIA DE DADOS:

É um conjunto ordenado de registros ou mensagens através dos quais dois processos se comunicam, os registros são lidos por um processo na mesma ordem em que foram escritos por outro processo.

- **NOME:** identificação da seqüência de dados
- **DESCRIÇÃO:** descreve o conteúdo da seqüência de dados

Esses dois atributos serão válidos para todos os tipos de seqüências de dados. Como mostra o diagrama do modelo de dados as seqüências são de vários tipos:

SEQÜÊNCIA DE DADOS DE ENTRADA: Englobam todas as seqüências de dados que têm um fluxo de chegada para um processo de modelagem ou funcional. Na sua vez elas podem ser:

- **SEQÜÊNCIA DA DADOS INTERNA:** representa uma seqüência de entrada a um processo que foi produzida internamente no sistema por uma função interativa.
- **SEQÜÊNCIA DE DADOS EXTERNA:** É também uma seqüência de entrada que chega ao processo vinda do mundo externo ao sistema. Esse tipo de seqüência de dados pode ser classificada em dois subtipos:

DO MUNDO REAL: quando trata-se de seqüências de entrada que chegam do mundo real e são absorvidas por um processo de modelagem.

MARCADOR DE INTERVALO DE TEMPO: quando trata-se de uma seqüência de entrada que é absorvida por processos funcionais na forma de marcadores de tempo como seu nome mesmo diz. Esses tipos de seqüência tem um atributo adicional, "TIPO", que indica a freqüência de ativação do processo. Exemplo: Consulta, Fim de Semana, Fim de Mes, etc.

SEQÜÊNCIA DE DADOS DE SAÍDA: Constitui um processo que é somado ao sistema com o propósito de produzir saídas.

- **FORMATO:** opção da forma na qual a saída será formatada na tela. Pode ser formato vertical ou horizontal.

SISTEMA:

É um componente do diagrama da modelagem JSD que não faz parte do método, mas foi criada para guardar a identificação do sistema alvo.

- **NOME:** é o nome que identifica ao sistema alvo na base de especificações.
- **DESCRIÇÃO:** descreve o objetivo do sistema.
- **RESPONSÁVEL:** guarda o nome do Engenheiro de Software responsável pelo sistema.
- **VERSÃO:** é um número ou identificação da versão do sistema.

TIPO DE DADOS:

Componente da modelagem da metodologia JSD que mesmo, sem formar parte dos passos de JSD, será o fundamento para que a especificação armazenada possa ser executada automaticamente nas etapas posteriores do ambiente.

- **NOME:** campo que identifica o tipo de dados.
- **TIPO:** guardará o “tipo” de dados propriamente dito que pode ser fornecido pelo sistema (Inteiro, Booleano, Caracter, etc), ou pelo usuário (tipos de dados simples ou compostos).
- **TAMANHO:** consiste do tamanho do tipo de dados no caso de tratar-se de um tipo de dados “Caracter” ou um tipo de dados “Composto”.

VALOR:

É um componente que foi criado para guardar os valores dos domínios discretos de um atributo.

- **CÓDIGO:** é o código ou identificador do domínio discreto.
- **DESCRIÇÃO:** é a descrição do código do domínio discreto.

VETOR DE ESTADOS:

São as variáveis locais de um processo.

- **ID. VE:** é o nome que identifica ao vetor de estados.
- **PROCESSO CONSULTADO:** cada vetor de estados, pertence a um processo. Este campo guardará o nome do processo ao qual pertence.

Figura A.2: DIAGRAMA DE REPRESENTAÇÃO DE OBJETOS - DRO

Figura A.3: DRO - Continuação

SIPS - SISTEMA INTEGRADO PARA PRODUCAO DE SOFTWARE

RELATORIO DAS INFORMACOES META

estrutura geral

0 10 95 1 0 0 0 0 0 0 0 0 0 0 0

mestre de comandos

440 -15360

estrutura da linguagem

3 5 19 58 59 60 68 70 72 91 94 0 0 0 0

n. classes = 2

n. entidades = 14

n. palesp = 40

n. ent. destino = 36

n. sinonimos = 5

n. atrib de relac. = 0

n. considere = 1

tabela de classes

0 - GLOBAL

1 - SISTEMA

tabela de entidades

1 - ACAA

2 - ATRIBUTO

3 - DIAGRAMA-ESTRUTURA

- 4 - ENTIDADE
- 5 - PROCESSO-FUNCIONAL
- 6 - PROCESSO-MODELAGEM
- 7 - SEQ-ENTRADA-EXTERNA-MIT
- 8 - SEQ-ENTRADA-EXTERNA-MUNDO-REA
- 9 - SEQ-ENTRADA-INTERNA
- 10 - SEQ-SAIDA
- 11 - SISTEMA
- 12 - TIPO-DE-DADO
- 13 - VALOR
- 14 - VETOR-ESTADO

tabela de palavras especiais

- 1 - CJ-INSTRUcoes
- 2 - COMPOE
- 3 - COMPOSTO-POR
- 4 - COMUNICA
- 5 - COMUNICADO-POR
- 6 - CONSIDERE-GLOBAL
- 7 - CONSULTA
- 8 - CONSULTADO-POR
- 9 - DA
- 10 - DESCRICAO
- 11 - DOMINIO
- 12 - E-GERADO-POR
- 13 - ESPECIFICACAO
- 14 - ESPECIFIQUE
- 15 - ESTRUTURA
- 16 - EXECUTA
- 17 - EXECUTADO-POR
- 18 - FAZEM-PARTE-DE
- 19 - FORMATO
- 20 - GERA
- 21 - GRAFICO
- 22 - HELP
- 23 - IDENTIFICA
- 24 - IDENTIFICADO-POR
- 25 - INTERCALA

palesp	tipo	relop	sit	revtr	carac	nulo
1	4	0	0	0	0	0
2	2	3	0	0	0	0
3	2	2	0	0	0	0
4	2	5	0	0	0	0
5	2	4	0	0	0	0
6	6	0	0	0	0	0
7	2	8	0	0	0	0
8	2	7	0	0	0	0
9	8	0	0	0	0	0
10	4	0	0	0	0	0
11	3	0	0	0	0	0
12	2	20	0	0	0	0
13	4	0	0	0	0	0
14	7	0	0	0	0	0
15	4	0	0	0	0	0
16	2	17	0	0	0	0
17	2	16	0	0	0	0
18	2	35	0	0	0	0
19	3	0	0	0	0	0
20	2	12	0	0	0	0
21	4	0	0	0	0	0
22	4	0	0	0	0	0
23	2	24	0	0	0	0
24	2	23	0	0	0	0
25	2	26	0	0	0	0
26	2	25	0	0	0	0
27	2	28	0	0	0	0
28	2	27	0	0	0	0
29	9	0	0	0	0	0
30	3	0	0	0	0	0
31	2	32	0	0	0	0
32	2	31	0	0	0	0
33	3	0	0	0	0	0
34	3	0	0	0	0	0
35	2	18	0	0	0	0
36	4	0	0	0	0	0
37	3	0	0	0	0	0
38	3	0	0	0	0	0
39	3	0	0	0	0	0
40	3	0	0	0	0	0

estrutura e abertura de classes

6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I

matriz de sintaxe de palavras especiais

tipo entidade: de 1 a 14 ; palavra especial: de 1 a 20

0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4	3	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	8128	0	0	0	0	0	1	0	0	0	0	0	0	1	0	3	0	0
1	0	0	0	0	0	0	14	0	0	1	0	0	1	0	0	0	0	0	0	148
1	0	0	4	0	0	0	14	0	0	1	0	0	1	0	0	0	0	0	0	134
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0160	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0156	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	12	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	141	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	13	0	0
0	0	0	4	0	0	0	152	0	0	0	5	0	0	0	0	0	0	2	0	0

tipo entidade: de 1 a 14 ; palavra especial: de 21 a 40

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
0	0	14	0	0	0	0	0	0	0	0	0	0	0	1136	0	0	1	1	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0139	1	0	0	0	0	0
0	0	0	12	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0

```

0 0 0 0146 0 0 0 0 0150 0 0 0 0 0 0 0 0
0 0 0 0132 0 0 0 0 0130 0 0 0 0 0 0 0 0
0 0 0 0 0 5 0 0 0 0 5 0 0 12 0 1 0 0
0 0 0 0 0 6 0 0 0 0 6 0 0 12 0 0 0 0
0 0 0 0 0158 0 0 0 0 0162 0 0 12 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 4 0 0 0 0 0 1 0 0 0 0 1
0 0 4 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2 0 0 0 0 0 1 0 0 0 0154 0 0 0 0

```

lista de entidades destino

X+128	iProx	codEntD	Proximo
128	0	6	1
129	1	14	0
130	2	9	3
131	3	8	0
132	4	9	5
133	5	8	0
134	6	9	7
135	7	10	0
136	8	13	9
137	9	12	10
138	10	14	0
139	11	4	12
140	12	1	0
141	13	2	14
142	14	14	15
143	15	9	16
144	16	8	17
145	17	7	0
146	18	9	19
147	19	7	0
148	20	9	21

149	21	10	0
150	22	7	23
151	23	9	0
152	24	6	25
153	25	5	0
154	26	12	27
155	27	12	0
156	28	5	29
157	29	6	0
158	30	6	31
159	31	5	0
160	32	6	33
161	33	5	0
162	34	6	35
163	35	5	0

sintaxe de atributos de relacionamentos

relac. atrib.

tabela de sinonimos

sinonimo	CodTPC	CodPCP
CODIGO	1	13
ID-DE	1	3
ID-VE	1	14
MIT	1	7
MUNDO-RE	1	8