UNIVERSIDADE DE SÃO PAULO
CAMPUS DE SÃO CARLOS
INSTITUTO DE CIÊNCIAS MATEMÁTICAS DE SÃO CARLOS
I.C.M.S.C.

Um ambiente de desenvolvimento baseado na abordagem operacional.

MASIERO, P.C., et al

nº 57

Notas do ICMSC - USP

ISSN 0103-2577

Um ambiente de desenvolvimento baseado na abordagem operacional.

MASIERO, P.C., et al

nº 57

São Carlos (SP)

1990

UM AMBIENTE DE DESENVOLVIMENTO BASEADO NA ABORDAGEM OPERACIONAL [1]

PAULO CESAR MASIERO
MARITZA RODRIGUEZ
DENISE BATISTA PEREIRA
INES APARECIDA GASPAROTTO

Universidade de São Paulo
Instituto de Ciências Matemáticas de São Carlos
Depto. de Ciências de Computação e Estatí stica
Endereço: ICMSC-USP
CX.P. 668
13560 - São Carlos, SP.

Sumário

É apresentado um ambiente de desenvolvimento em estágio atual de projeto e implementação de seus vários componentes, baseado na abordagem operacional. O ambiente fundamenta-se nos métodos JSD e no conceito de Tipos Abstratos de Dados, para permitir a especificação de um sistema a ser desenvolvido e a geração de um protótipo desse sistema.

PALAVRAS CHAVES:
Ambiente de Desenvolvimento,
Abordagem Operacional,
JSD (Jackson System Development)
Tipos Abstratos de Dados
Protótipos

[1] Este Trabalho contou com apoio financeiro do CNPq e CAPES

1 Introdução

Existe hoje uma vasta gama de métodos semi-formais que dirigem o Engenheiro de Software, através de passos bem definidos, na tarefa de especificar, projetar e implementar software ao nível de sistemas. Tais métodos levam, geralmente, à produção da especificação e do projeto como documentos em papel, que devem ser, posteriormente, mapeados para uma linguagem executável pelo computador. Esse processo é responsável por parte do tempo gasto no desenvolvimento de software e é propenso a gerar erros.

Ferramentas de apoio por computador ao desenvolvimento de software, designadas pela sigla CASE (Computer Aided Software Engineering), vêm sendo desenvolvidas de maneira bastante acentuada nos últimos anos. Dart e outros [DA87] classificam os ambientes de desenvolvimento de software em quatro tipos principais:

1. **Ambientes centrados em linguagens:** Oferecem um conjunto de ferramentas adequadas a uma certa linguagem. Exemplo: ADA e SMALLTALK.

2. **Ambientes orientados por estruturas:** Oferecem e incorporam técnicas independentes de linguagens, que permitem a manipulação direta de estruturas de programas, levando à noção de geradores de ambientes. Exemplos: Gandalf, Mentor, Cornell Program Synthesizer.

3. **Ambientes baseados em conjunto de ferramentas:** Oferecem conjuntos de ferramentas ("tool kits") independentes de linguagem e relativamente isoladas, para auxílio a tarefas tais como: gerencimento de configurações e controle de versões. Exemplos: Arcadia, PCTE, Unix/PWB.

4. **Ambientes baseados em métodos (ou metodologias):** Oferecem um conjunto de ferramentas , com maior ou menor grau de integração, reunidas em torno de um método de análise e projeto de sistemas. Exemplos: Ambientes em torno dos métodos de Análise e Projeto Estruturado: Excelerator, Nastec CASE 2000, TeamWork,etc.; ambientes em torno de métodos de modelagem de dados: ERD (Entity-Relationship Designer / Chen), IEW (Knowledgeware) .

Os ambientes do tipo 4 procuram facilitar o trabalho do Engenheiro de Software, oferecendo editores gráficos, editores de texto estruturados e outros tipos de ferramentas, dentro de um ambiente padronizado, para agilizar a criação

da especificação e do projeto do sistema, segundo as diretrizes estabelecidas por um certo método.

Embora a documentação contendo a especificação do sistema esteja armazenada dentro do computador, a sua transformação para um sistema executável pelo computador continua essencialmente manual, havendo, portanto, um hiato na passagem do projeto para a implementação. A maioria das ferramentas atualmente em disponibilidade comercial, como as citadas acima, são desse tipo. Algumas, como PROMOD [HR83] por exemplo, oferecem algum auxílio na passagem do projeto para a implementação, criando esqueletos de programas que o programador depois preencherá.

Este trabalho está organizado da seguinte forma: nesta seção 1 situa-se rapidamente o papel e o estado atual dos softwares do tipo CASE dentro da Engenharia de Software. A seção 2 define o que é a abordagem operacional e justifica a sua escolha para dar suporte ao desenvolvimento de um Ambiente de Desenvolvimento baseado na Abordagem Operacional (daqui por diante também denominado pela sigla de Projeto ADAO), descrito neste trabalho. Na seção 3 faz-se uma descrição sucinta dos quatro primeiros passos do método JSD, relevantes para o ambiente em desenvolvimento, que é descrito na seção 4, mostrando-se sua arquitetura e comentando-se brevemente cada um de seus componentes. A seção 5 oferece um exemplo hipotético do funcionamento do protótipo, para o exemplo mostrado na seção 3 e a seção 6 comenta o estado atual do ambiente ADAO e alguns dos problemas atualmente enfrentados.

2 A abordagem operacional

O ambiente de desenvolvimento descrito neste trabalho enquadra-se no tipo 4 descrito na seção anterior e seu objetivo é dar apoio às fases do ciclo de vida de um software, desde a especificação até a sua implementação e manutenção. Alguns aspectos dos ambientes dos tipos 1 e 2 serão eventualmente utilizados.

A abordagem operacional é definida por Zave [ZA84] como uma estratégia de desenvolvimento de software onde, "durante a fase de especificação , o especialista em computação formula um sistema para resolver um problema e especifica esse sistema em termos de estruturas independentes da implementação, que geram o comportamento especificado do sistema". A especificação operacional pode ser, em tese, executada por um (ou mais) processador(es) adequado(s). Balzer, que juntamente com outros colegas, desenvolveu a linguagem

Gist [BA82], a classifica como uma linguagem operacional, afirmando que "a especificação em Gist pode ser avaliada para produzir o comportamento (uma seqüência de estados), dado um estado inicial".

Nesse mesmo artigo, Zave classifica JSD, o seu próprio método - PAISley - e Gist, como abordagens operacionais e conclui que JSD era a única instância da abordagem operacional, até aquele momento, com procedimentos e diretrizes bem desenvolvidos de forma a permitir, na prática profissional, a geração de uma especificação operacional.

Como parte dos objetivos do Projeto ADAO é a geração automática, a partir da especificação, de um protótipo da aplicação a ser desenvolvida (também chamada daqui por diante de sistema alvo), foi escolhido como base teórica o método JSD [JA83], [CA86], adaptado de forma a se integrar com o conceito de Tipos Abstratos de Dados e funcionar como um método orientado a objetos [MA88a]. Nesta primeira etapa do projeto, dar-se-á ênfase ao domínio de aplicações em Sistemas de Informação, com a geração de protótipos interativos.

Motivos adicionais, visando a evolução futura do ambiente, foram considerados na escolha de JSD: JSD permite a especificação de sistemas de tempo-real e tem certa afinidade com a especificação de processos paralelos e/ou comunicantes, pois incorpora, ainda que de forma não explícita, conceitos da linguagem CSP [HO78]. Ambos os aspectos deverão ser explorados em etapas posteriores do projeto.

Bjorner, em [BJ89, p.6], ao comparar VDM e JSD, afirma que ambos se fundamentam numa base matemática razoavelmente firme, embora JSD "evite cuidadosamente aborrecer seus usuários, exigindo que eles tenham conhecimento desses fundamentos".

Vários esforços têm sido relatados na literatura de ferramentas para dar apoio por computador ao uso do método JSD, procurando aproveitar seu aspecto operacional. Cameron [CA86], por exemplo, comenta sobre o desenvolvimento de um editor de diagramas de estrutura com capacidade para gerar texto em linguagens de programação, como o COBOL e Ambrósio [AM88] descreve uma ferramenta para apoio automático à especificação JSD e outra ferramenta para simulação da especificação JSD.

3 Revisão sucinta do método JSD

Esta seção tem o objetivo de fazer uma apresentação sucinta de JSD para os que não estão familiarizados com o método, usando como exemplo um sistema de controle de fitas em um Video Clube. Entretando, apenas alguns pequenos trechos da especificação desse exemplo serão mostrados. A especificação completa do exemplo pode ser encontrada em [MA89] e a descrição completa do método JSD em [JA83]. O exemplo será utilizado nas seções seguintes para ilustrar alguns aspectos do projeto ADAO, em especial o protótipo.

O método é composto de seis passos, agrupados em três fases principais: A fase de modelagem, na qual as entidades relevantes são selecionadas e definidas, bem como suas ações; a fase de construção da rede de processos, na qual a especificação é completada com a interconexão dos processos que modelam o comportamento das entidades e dos processos funcionais que recuperam e formatam as informações requeridas pelos usuários do sistema; e, a fase de implementação, na qual os processos são adaptados aos processadores existentes e os dados organizados e ajustados à memória disponível. No projeto ADAO esta última fase será substituida pelo protótipo, razão pela qual não será considerada nesta seção.

Consideremos o exemplo do Video Clube. Entidades relevantes nesse contexto seriam FITA, MEMBRO, RESERVA, etc. Na figura 1 encontra-se uma descrição parcial da entidade FITA. Cada entidade é definida pelo conjunto de ações que executa no mundo real e as ações são identificadas por um conjunto de atributos.

A fase de modelagem prossegue criando-se um diagrama de estrutura para cada entidade. Os componentes no nível mais baixo da estrutura hierárquica são as ações das entidades. O diagrama de estrutura modela cronologicamente a ocorrência das ações. Por exemplo, uma instância particular de fita deve primeiro ser comprada para então poder ser utilizada no Video Clube. As utilizações ocorrem repetidamente, como seqüências de saídas e retornos da fita como conseqüência de aluguéis. Ao final a fita é descartada por algum motivo, como por exemplo: estragou-se, perdeu-se, etc. Um exemplo desse diagrama é mostrado na figura 1(a).

Entidade : FITA	Identificador	: ID-FITA
	ATRIBUTOS	TIPOS
Descrição: Esta entidade descreve a vida de uma cópia de certa fita. A fita deve ser comprada e então passa a poder ser retirada (alugada) pelos membros do video clube e devolvidas. Quando estragar ou se perder ela é descartada do acervo do video clube	TITULO	: Char (40)
	VOLUMES	: Integer
	DATA-COMPRA	: Data
	RESUMO-FILME	: Char (200)
	DATA-ALUGUEL	: Data
	DATA-DEVOL	: Data
	DATA-DESCAR.	: Data
	MOTIVO-DESC	: Char (1)
Ação : COMPRAR	ATRIBUTOS	TIPOS
O video clube adquire uma nova fita.	COD-FITA	: ID-FITA
	TITULO	: Char(40)
	VOLUMES	: Integer
	DATA-COMPRA	: Data
	RESUMO-FILME	: Char(200)
Ação: SAIR	COD-FITA	: ID-FITA
	MEMBRO	: ID-MEMBRO
Uma fita é retirada do acervo... etc...	DATA-ALUGUEL	: Data

Tabela 1: Descrição parcial da entidade FITA

A fase de construção da rede de processos consiste da criação de um diagrama, denominado Diagrama de Especificação do Sistema. Esse diagrama é composto das entidades relevantes ao problema (representadas por retângulos com nomes com sufixos = 0) que se conectam a processos de modelagem (com o mesmo nome das entidades, mas com sufixo maior que zero). Cada ação da entidade no mundo real é enviada como entrada para o processo de modelagem correspondente, que simula internamente o evento ocorrido externamente no mundo real.

O modelo em rede do sistema é acrescido dos processos funcionais, isto é, dos processos que retiram informação de dentro do sistema, passando-as para os usuários na forma de consultas, relatórios, etc. A figura 2 mostra um pequeno trecho do diagrama em rede do exemplo do Video Clube. Aparecem nele

as entidades FITA e MEMBRO, com seus respectivos processos de modelagem. Mostram-se também duas funções, uma consulta à situação da fita (Disponível ou Alugada) e outra gerando uma estatística mensal da quantidade de fitas alugadas por um membro.

Na figura 2 os processos se comunicam por seqüências de dados, como FI, FA e ME, e nesse caso a iniciativa de enviar a seqüência de dados está no processo onde ela se origina. JSD permite também que um processo verifique o conteúdo (ou estado) das variáveis de outro processo, sem alterá-lo, e essa consulta é denominada "consulta ao vetor de estado" do processo.

4 Arquitetura do Projeto ADAO

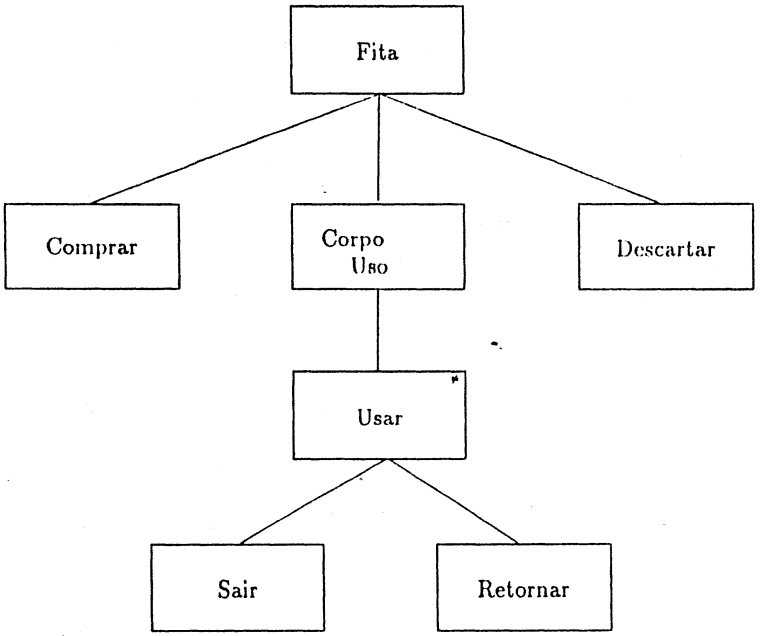
O ambiente, em sua versão atual, é constituido de quatro módulos principais, que podem ser vistos na figura 3. Cada um deles será descrito nas sub-seções seguintes.

4.1 Gerenciador da Base Meta

A idéia de se utilizar uma meta base de dados para armazenar informações a respeito da especificação de software teve um de seus precursores no projeto ISDOS, com a criação do sistema PSL/PSA e de seu sucessor, o sistema SEM [TE80]. Esses sistemas têm características específicas para o gerenciamento de base de dados de engenharia, no caso, de software. No Brasil, essas idéias foram utilizadas no sistema SACDS e depois ampliadas e reimplementadas como o núcleo da base de dados do projeto SIPS, desenvolvido no CTI, em Campinas [TR86].

O módulo gerador da base meta (GeBaM), será utilizado para armazenar o modelo do método JSD. Para sua criação, o método JSD foi modelado utilizando-se os conceitos de Entidade-Relacionamento. Para que o modelo possa ser compilado e armazenado na base meta, deve ser mapeado para uma linguagem descritiva correspondente ao Modelo de Representação de Objetos utilizado pelo GeBaM [TR89]. Uma descrição mais detalhada desse processo, modelando os métodos de Análise e Projeto Estruturados, encontra-se em [MA88b].

Basicamente, o primeiro passo consiste em definir os objetos relevantes em JSD, tais como: Entidades, Ações, Atributos, Processos de Modelagem,



(a)

```
FITA seq
   Comprar;
   Corpo-Uso itr
     Usar seq
       Sair;
       Retornar;
     Usar fim
   Corpo-Uso fim
   Descartar;
FITA fim
```

(b)

Figura 1: Diagrama de Estrutura da Entidade FITA

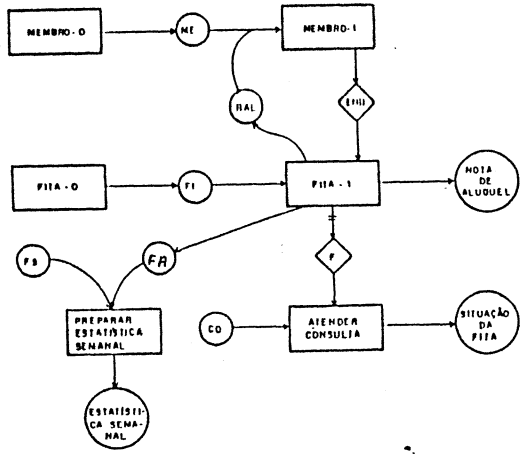


Figura 2: Diagrama de Especificação do Sistema

Seqüências de Dados, etc. Em seguida, são definidas as relações entre esses objetos. Por exemplo: Entidade executa ação, Ação tem atributo , etc.

O Modelo Entidade-Relacionamento completo para o método JSD é mostrado no apêndice A. O diagrama contém todos os objetos conceituais considerados em JSD, seus atributos e seus relacionamentos.

A base de dados meta é criada uma única vez pelos desenvolvedores (ou administradores) do ambiente e não deve mudar, a não ser em outras versões do ambiente. As ferramentas do ambiente, como o analisador da especifição e o executor, utilizam essa base e conhecem a sua estrutura interna.

4.2 Analisador da Especificação Operacional

A tarefa do Analisador da Especificação Operacional (AnEspO) é permitir a entrada da especificação do sistema alvo, isto é , a aplicação a ser desenvolvida, cobrindo os quatro primeiros passos de JSD. A especificação será analisada, erros informados ao projetista e, ao final, será criada uma base de dados (Base Especificação) contendo a especificação do sistema alvo.

O analisador da especificação operacional permite a entrada de todas as informações contidas nas figuras 1 e 2 e na tabela 1. A interface do AnEspO está sendo projetada de forma a evitar o máximo possível de erros por parte do projetista mas, para permitir a entrada de dados de forma incremental, algumas consistências não serão feitas imediatamente e serão fornecidas posteriormente na forma de relatórios de consistências solicitados pelo Engenheiro de Sistemas.

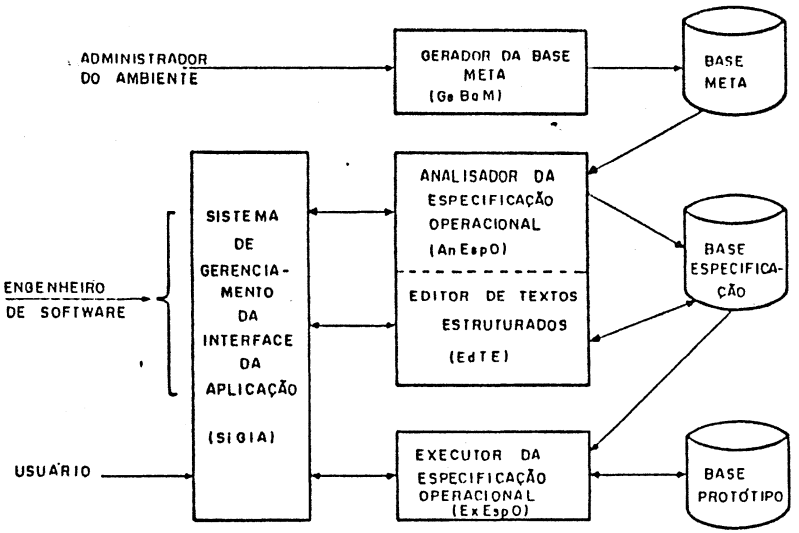


Figura 3: Arquitetura do Projeto ADAO

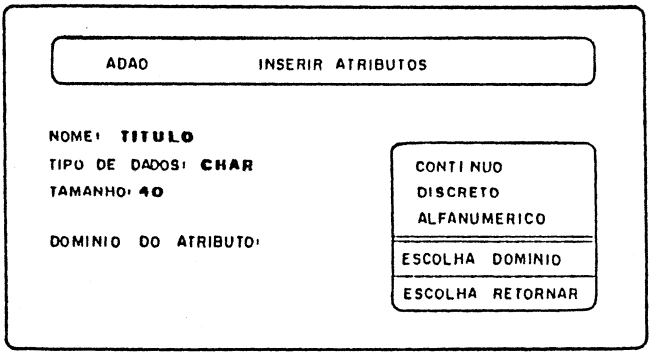


Figura 4: Tela de Entrada do AnEspO

A interface deste e dos demais módulos será baseada em janelas, com formulários de entrada e saída, combinada com diálogos comandados por "prompters" e menus "pull down". No caso do Diagrama de Estrutura das entidades, a forma de entrada será feita através de uma linguagem formal bastante simples.

A figura 4 mostra uma tela típica do AnEspO, onde está sendo inserido o atributo TÍTULO. O usuário digitou o nome do atributo, o tipo e o tamanho de sua estrutura de dados e o AnEspO mostra agora um menu superposto para que seja escolhida a opção sobre o domínio de valores do atributo. Após a escolha esse menu desaparece.

Um subcomponente bastante importante do AnEspO é o Editor de Textos Estruturados (EdTE). Este módulo obterá da Base da Especificação, para cada entidade, a estrutura correspondente ao diagrama da figura 1(a) e o converterá num texto estruturado, segundo a notação de JSD, conforme é mostrado na figura 1(b).

A partir desse macro texto estruturado, EdTE permitirá que o Engenheiro de Software refine o processo, segundo as regras de JSD. Cada ação será decomposta em operações elementares e os predicados das iterações e seleções serão definidos. Poderão ser acrescentadas construções de "backtracking", atribuições e expressões aritméticas. As leituras e gravações de seqüências de dados serão acrescentadas, assim como as consultas a vetores de estado. O AnEspO armazenará durante a entrada do modelo em rede, todas as trocas de mensagens entre os processos de forma a, ao mesmo tempo, validar e auxiliar o profissional durante a alocação dessas operações na estrutura de cada processo.

A criação e edição de processos funcionais ocorrerá de forma semelhante à descrita acima para os processos de modelagem, com a única diferença de

que o ponto de partida não será um esqueleto de processo, mas sim um processo novo.

O EdTE deve ser orientado pela sintaxe de JSD e também permitir a visão de níveis hierárquicos diferentes, de forma a poder permitir a visão do processo no seu nível mais alto (como na figura 1(b), assim como entrar nos detalhes das ações quando for necessário. No final, cada ação completamente detalhada será desmembrada da sua estrutura e armazenada como uma 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 detalhados de cada ação, formará um tipo abstrato de dados [MA88a]. O texto fonte de cada ação será armazenado como um atributo da ação na Base Especificação.

4.3 Executor da Especificação Operacional

Uma vez que a especificação completa do sistema alvo esteja armazenada na base de especificação, o Executor da Especificação Operacional (ExEspO) poderá simular a sua execução, como um protótipo funcional. Mas antes que isso possa acontecer, um programa preparador deve ser executado, para fazer a transição da especificação para o protótipo. Esse procedimento deve ser disparado pelo Engenheiro de Software após ter completado a especificação. Em caso de mudanças, a especificação deve ser alterada adequadamente e o procedimento de transição novamente ativado. Serão controladas as versões da especificação e do protótipo. Isto será feito de forma bastante simples: toda vez que a fase de transição é executada com sucesso, permitindo a execução do protótipo, será gerado um identificador interno igual para ambas as bases de dados: de especificação e do protótipo. Se a especificação for alterada o usuário será informado de que ela não corresponde ao protótipo em funcionamento e terá a opção de utilizá-lo assim mesmo ou gerá-lo novamente.

A transição é composta de duas partes principais: a preparação dos vetores de estado e a preparação dos processos.

Para a criação dos vetores de estado, o programa de transição deve reunir todos os atributos de uma entidade, eliminando os que aparecem duplicadamente em mais de uma ação e, através de seus tipos, formar a estrutura de um registro, que será utilizado para guardar as instâncias de cada entidade daquele tipo na base de dados do protótipo. Alguns campos de controle também serão acrescentados ao vetor de estado, como por exemplo o que controla a última ação executada pela entidade, para efeito de validação da seqüência de ações de uma entidade durante a execução do protótipo.

Para a preparação dos procedimentos relativos a cada processo de modelagem (composto por suas ações) e a cada processo funcional, uma primeira transformação deverá ser feita: as leituras e gravações de seqüências de dados de e para fora dos limites do sistema serão transformadas em chamadas de rotinas do Sistema de Gerenciamento de Interface da Aplicação. As gravações de seqüências de dados internas (como FA, na figura 2) serão transformadas em chamadas de procedimentos, sendo chamador o processo que envia a seqüência de dados e chamado o processo que recebe a seqüência de dados. A seqüência de dados será passada como parâmetro nessa chamada, adicionando-se outras informações se for necessário.

Consultas a vetores de estado serão transformadas em consultas ao sistema de gerenciamento da base de dados do protótipo. As operações de atribuição envolvendo atributos da entidade, dentro dos processos de modelagem, mudando o estado dessas variáveis, serão processadas da mesma forma, mas numa área de memória correspondente ao vetor de estado da entidade, que ao final do processamento será rearmazenado na base de dados do protótipo.

O Executor da Especificação Operacional propriamente dito, após a fase de transição, funcionará como um escalonador de processos, gerenciando chamadas à Base Especificação, à Base protótipo e ao Sistema de Gerenciamento de Interface da Aplicação. O exemplo ilustrado na seção 5 esclarecerá melhor o seu funcionamento.

4.4 Sistema de Gerenciamento da Interface da Aplicação

Tanto o Engenheiro de Software quanto o usuário final do protótipo devem interagir com o ambiente através de uma interface padronizada. No caso do primeiro, isso se dará através de rotinas de manipulação de janelas, menus e diálogos, controlados pelos módulos AnEspO e EdTE. Está sendo utilizado para isso um conjunto de rotinas (tool-kit) com funções variadas de controle de vídeo e de janelas, para a linguagem C.

O que está sendo chamado de Sistema de Gerenciamento da Interface da Aplicação (SiGIA) é uma camada de software superposta ao Sistema de Gerenciamento de Janelas, de forma a adequa-lo ao domínio da aplicação, no caso Sistemas de Informação, e mais ainda, adequá-lo aos conceitos utilizados pelos métodos de desenvolvimento de sistema, que no caso foram JSD e TADs.

Um Sistema de Gerenciamento de Janelas pode ser visto como um subconjunto do "tool kit" de rotinas de gerenciamento de vídeo (abrir e fechar

janelas, mover janelas, aumentar ou diminuir seu tamanho, etc.) acrescido de rotinas de controle das janelas na tela, como por exemplo: controlar a conjunto de janelas mostradas na tela e a pilha de sobreposição, controlar a janela ativa, mover uma janela para o topo da pilha etc.

O SiGIA, entretanto, gerenciará uma tela padrão onde as entidades modeladas em JSD e constantes da Base Especificação aparecerão como ícones na faixa superior da tela. Os processos funcionais, agrupados adequadamente, também aparecerão como ícones nessa mesma faixa da tela. A seleção de um desses ícones fará aparecer um menu "pull down" contendo, no caso das entidades, a lista das ações disponíveis e, no caso das funções, uma lista com os nomes das funções daquele tipo. A seleção de uma ação (ou função) no menu, faz com que o SiGIA devolva o controle ao executor da especificação operacional que então ativará a ação (ou função) adequada.

A ação, ao executar, também poderá interagir com o SiGIA solicitando a abertura de uma janela tipo formulário de entrada. Estão previstas opções para auxílio ao usuário: dois cliques sobre qualquer conceito JSD representado na tela, como por exemplo, entidades, ações, atributos, etc., fará aparecer uma janela contendo informações sobre o conceito - descrição da entidade ou da ação contendo as informações mostradas na figura 1, domínio de valores de atributos, descrição de funções, etc.. Prevê-se a possibilidade de transformar em ícones os formulários (de saída) na tela, levando-os para uma faixa à direita do vídeo (isso corresponde a um congelamento dessa consulta num certo instante, para uso posterior através de uma operação reversa de reabrir o ícone em uma janela); a passagem de dados de um formulário para outro (cortar e colar); rolamento vertical e horizontal de formulários tipo relatórios; etc. A especificação completa deste módulo pode ser encontrada no documento [PE89].

5 Um exemplo hipotético do Protótipo

Consideremos, por exemplo, o protótipo do sistema para o Video Clube, utilizado ao longo deste trabalho. Ao iniciar a execução do protótipo o ExEspO chamará uma rotina do SiGIA para estabelecer a interface padrão, conforme pode ser visto na figura 5(a).

Digamos que o usuário deseja executar uma ação da entidade FITA. Para isso deve ser selecionado o seu ícone (com o mouse ou teclas de setas), o que imediatamente fará aparecer o menu de ações, como na figura 5(b). O mesmo processo de seleção de uma ação fará com que o SiGIA devolva o controle ao

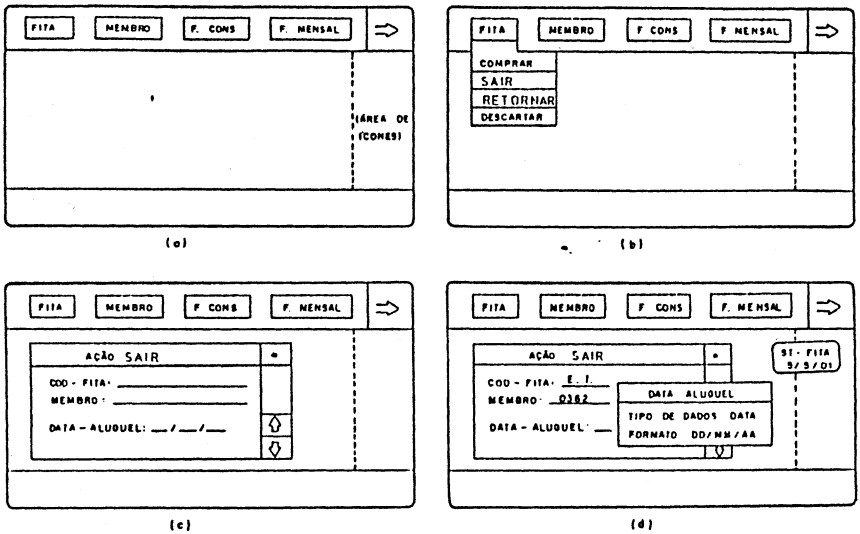


Figura 5: **Exemplos de telas do protótipo**

executor indicando a ação selecionada. Este, por sua vez, recuperará o texto estruturado correspondente à ação selecionada e o interpretará. Suponhamos que a ação selecionada tenha sido SAIR. Para obter os atributos da ação sair, uma nova rotina do SiGIA será chamada, fazendo aparecer a janela tipo formulário que pode ser vista na figura 5(c).

O procedimento será semelhante caso seja escolhido um ícone funcional. A consulta ou relatório resultante poderá ser, opcionalmente, remetida para a impressora ou para a tela. Neste último caso, depois de analisado, poderá ser transformado num ícone, indo para uma faixa vertical à direita da tela. A figura 5(d) mostra uma janela com uma consulta transformada em ícone e uma tela de auxílio, após dois cliques sobre o campo DATA ALUGUEL na janela da ação SAIR.

A breve visão dada acima permite que se tenha uma idéia de como

funcionará o protótipo e de como se integrarão a Base Especificação, a Base Protótipo, o ExEspO e o SiGIA.

6 Considerações finais

A situação atual do projeto ADAO é a seguinte: o modelo do método JSD está completo e a base meta gerada. O analisador da Especificação Operacional está completamente especificado [RO89] e sua implementação está em fase final. O editor de textos estruturados está em fase inicial de especificação e o executor da especificação operacional, incluindo-se a transição da especificação para o protótipo, está em fase adiantada de especificação. O sistema de Gerenciamento da Interface da Aplicação já se encontra especificado [PE89] e está em fase de programação.

A programação está sendo feita na linguagem C em ambiente de microcomputadores tipo PC. Estuda-se a idéia de migrar o sistema todo para um ambiente tipo Unix. O ambiente que nos parece ideal seria uma estação de trabalho tipo SUN, com terminais de maior resolução, onde poder-se-ia acrescentar a opção de entrada da especificação através de editores gráficos poderosos. Entretanto, equipamentos desse tipo ainda não estão disponíveis.

Prosseguindo-se com o ambiente computacional atual, planeja-se que as ferramentas descritas neste trabalho deverão estar todas implementadas até meados do próximo ano.

É importante acrescentar que o protótipo só rodará dentro do ambiente ADAO. Caso o usuário queira, será oferecida a possibilidade de geração de programas fontes em C, relativos a cada tipo abstrato de dados correspondente a uma entidade, contendo as declarações dos vetores de estado e os procedimentos correspondentes às ações. Nesse caso, o Engenheiro de Software deverá fazer algumas alterações, principalmente nas rotinas de entrada e saída e criar o seu próprio programa escalonador para implementar o sistema definitivo.

Referências

[AM88] AMBROSIO, A. & VELASCO, F.R.D *Um sistema de especificações JSD*. II Simpósio Brasileiro de Engenharia de Software, Canela, RS, Outubro de 1988.

[BA82] BALZER,R.M.; GOLDMAN,N.M. & WILE,D.S. *Operational Specification as the Basis for Rapid Prototyping*. ACM SIGSOFT Software Engineering Notes, Vol. 7, (5), Dec 1982.

[BJ89] BJORNER,D. *The VDM Specification and Implementation Methodology*. State the Art Seminar on Formal Description of Programming Concepts, Petrópolis, Brazil, 18-28 Abril de 1989.

[CA86] CAMERON,J.R. *An Overview of JSD*. IEEE Transactions on Software Engineering, vol.SE-12, february 1986.

[DA87] DART,S.A.; ELLISON,R.J.; FEILER,P.H. and HABERMANN, A.N. *Software Development Environments*. Computer, november 1987.

[HO78] HOARE C.A.R., *Communicating Sequencial Processes*. Communications of the ACM, vol 21 no. 8, august 1978.

[HR83] HRUSCHKA,P. *The Software Engineering Environment PROMOD*. Proceedings of ESA/ESTEC Software Engineering Seminar, Noorwijk, 1983.

[JA83] JACKSON,M. *System Development*, Prentice-Hall, 1983.

[MA88a] MASIERO,P.C. & GERMANO,P.C. *JSD as an Object Oriented Design Method*. Software Engineering Notes, Vol. 13, No 3, july 1988.

[MA88b] MASIERO,P.C. *O Processo de Instanciação 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.

[MA89] MASIERO,P.C. *Uma Visão Geral do Método JSD*, Notas Didáticas do ICMSC, No. 03, 1989.

[PE89] PEREIRA,D.B. *Um Sistema de Gerenciamento de Aplicações*. Monografia preparada para o exame de qualifacação ao mestrado, ICMSC-USP, 1989.

[RO89] RODRIGUEZ,M. *Um sistema analisador de especificações JSD*. Monografia preparada para o exame de qualificação ao mestrado, ICMSC-USP, 1989.

[TR86] TRAINA Jr.,C. et al. *SIPS - Estado Atual de Desenvolvimento*. XIX Congresso Nacional de Informática, agosto, 1986.

[TR89] TRAINA Jr.,C. e SLAETS, J.F.W *Um modelo de representação de objetos*, (Em preparação), 1989.

[TE80] TEICHROEW,D. et alli *Apllication of the Entity-Relationship Approach to Information System Analysis and Design*. In: CHEN,P.P. (Ed.). Entity Relationship Approach to Systems Analysis and Design, North Holland, 1980.

[ZA84] ZAVE,P. *The Operational versus the Conventional Approach to Software Development*. Communications of the ACM, Vol 27 (2), 1984.

A Modelo de dados do Método JSD