UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FERRAMENTA DE APOIO À ESPECIFICAÇÃO UTILIZANDO O MODELO TF-ORM

por
NINA EDELWEISS
ALFREDO KENGI KOJIMA

RP-259 Março/96
Relatório de Pesquisa

UFRGS - II - CPGCC
Caixa Postal 15064 - CEP 917401-970
Porto Alegre - RS - BRASIL
Telefone: (051) 336-8399
Fax: (051) 336-5576
E-mail: PGCC@INF.UFRGS.BR

SUMÁRIO

RESUMO

TF-ORM (*Temporal Functionality in Objects with Roles Model*) é um modelo de dados temporal orientado a objetos que visa ser utilizado na especificação formal de sistemas de informação.

A especificação de aplicações em TF-ORM pode ser bastante complexa, levando a possibilidade de erros. Para auxiliar o processo de especificação de aplicações no modelo TF-ORM, enfatizando o aspecto orientado a objetos do modelo e buscando a redução da possibilidade de erros, foi desenvolvida um ambiente de apoio a especificações. Ambiente este, composto por duas ferramentas distintas: uma *Ferramenta de Apoio ao Projetista de Aplicações* e uma *Ferramenta de Apoio ao Engenheiro de Aplicações*.

Este documento visa explicar o uso e funcionamento da *Ferramenta de Apoio ao Projetista de Aplicações* — ferramenta usada para a construção de aplicações no modelo TF-ORM.

PALAVRAS-CHAVE: MODELAGEM TEMPORAL, MODELO DE DADOS ORIENTADO A OBJETOS, AMBIENTE DE ESPECIFICAÇÃO, REUTILIZAÇÃO.

ABSTRACT

TF-ORM (*Temporal Functionality in Objects with Roles Model*) is a temporal object-oriented data model meant to be used in the information system's formal specification.

The specification of TF-ORM aplications can be very complex, leading to the possibility of errors. A specification support environment was developed to help the specification process of TF-ORM aplications, reducing errors and taking benefit of the object-oriented aspect of the model. This environment is composed of two distinct tools: an *Application Designer Support Tool* and an *Application Engineer Support Tool*.

This document is meant to explain the *Application Designer Support Tool* and it's usage.

KEYWORDS: TEMPORAL MODELING, OBJECT-ORIENTED DATA MODEL, SPECIFICATION ENVIRONMENT, REUSE.

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Este documento traz a descrição da implementação de um ambiente de apoio a especificações, que enfatiza a reutilização de classes armazenadas em uma biblioteca. Como método de especificação é utilizado o modelo de dados temporal orientado a objetos TF-ORM (*Temporal Functionality in Objects with Roles Model*) [EDE 93b, 94]. O ambiente está sendo implementado em PROLOG [STE 86]. Duas versões estão disponíveis — uma para ser utilizada em computadores PC-compatíveis e outra, para estações de trabalho SUN.

Dois diferentes *usuários* são identificados no ambiente: o engenheiro de aplicações e o projetista de aplicações. O *engenheiro de aplicações* é um especialista encarregado da construção da biblioteca de classes. É também encarregado de posteriormente, a intervalos periódicos, fazer a manutenção da biblioteca com base nas informações referentes às especificações realizadas. Estas informações são armazenadas pelo ambiente em uma biblioteca auxiliar durante os processos de especificação. O *projetista de aplicações* é o usuário que utiliza o ambiente como apoio para a construção da especificação de uma aplicação. O próprio ambiente, através da ferramenta aqui descrita, guia esta especificação enfatizando a utilização de classes da biblioteca.

O ambiente conta com duas ferramentas: (i) *apoio à especificação* — uma ferramenta que promove a especificação apoiando o projetista da aplicação; durante a especificação é incentivada a reutilização de especificações anteriormente realizadas, através da utilização de classes de objetos armazenadas em uma biblioteca de classes; e (ii) *construção e manutenção da biblioteca* — uma segunda ferramenta que se encarrega de apoiar o engenheiro de aplicações na montagem e na manutenção da biblioteca de classes, com base em informações contidas em uma biblioteca auxiliar, a biblioteca de manutenção, onde são armazenados dados referentes a especificações realizadas. Neste Relatório de Pesquisa será feita a descrição da primeira ferramenta — apoio à especificação.

No capítulo 2 deste trabalho será feita uma descrição sucinta do modelo de dados TF-ORM. No capítulo 3 será feita a descrição do ambiente todo, detalhando os bancos de dados envolvidos na especificação. A ferramenta de apoio à especificação é descrita no capítulo 4. No capítulo 5 serão explicados os procedimentos necessários para se começar a usar a ferramenta e no capítulo 6 será explicado o uso da ferramenta. No capítulo 7 são descritos os modelos de dados internos de cada um dos bancos de dados usados na implementação do ambiente.

2. MODELO DE DADOS TF-ORM

TF-ORM é um modelo de dados temporal orientado a objetos. Seu objetivo principal é o de ser utilizado na especificação formal de requisitos de sistemas de informação. Neste capítulo será feita uma descrição sucinta do modelo e de sua linguagem de definição — uma descrição mais detalhada e um exemplo completo de modelagem podem ser encontrados no Relatório de Pesquisa nº 258.

2.1 Conceito de Papéis

O comportamento dos objetos é descrito em TF-ORM através do conceito de *papéis*. Nos modelos orientados a objetos tradicionais, um elemento especificado pode ser instância de uma única classe de objetos. O conceito de papéis amplia esta dependência, permitindo que diferentes comportamentos de um mesmo elemento sejam representados através de instâncias de papéis diferentes. Como exemplo, considere um objeto representando um *pessoa*. Este objeto pode desempenhar simultaneamente o papel de *cliente* e de *funcionário* de uma loja. Nos modelos orientados a objetos tradicionais *cliente* e *funcionário* seriam representados como subclasses da classe *pessoa*, e não seria possível o mesmo objeto estar simultaneamente nas duas subclasses. Através do conceito de papéis, *cliente* e *funcionário* são representados como papéis diferentes dentro da mesma classe *pessoa*, sendo então possível apresentar uma instância em cada um dos dois papéis ao mesmo tempo. Além disso, no modelo TF-ORM um objeto pode possuir mais de uma instância de um determinado papel — no exemplo apresentado, uma pessoa pode possuir dois empregos de tempo parcial na mesma loja, sendo cada um deles representado por uma instância diferente do papel funcionário.

A utilização do conceito de papéis não anula a possiblidade de definir subclasses — na definição de uma subclasse, os papéis contidos na superclasse sofrem os mesmos mecanismos de herança definidos para as propriedades — uma subclasse pode ou não herdar os papéis da superclasse, e os papéis herdados podem ter sua definição modificada ou estendida. No modelo TF-ORM é permitida a herança múltipla. A abstração de agregação também pode ser utilizada.

2.2 Tipos de Classes

Três tipos diferentes de *classes* são definidos no modelo TF-ORM: classes de agentes, de recursos e classes de processos. Uma *classe de agente* representa uma pessoa que interage com a aplicação que está sendo especificada, podendo executar tarefas e tomar decisões. Os papéis representam os diferentes comportamentos apresentados por esta pessoa no desenvolvimento da aplicação. Uma *classe de recurso* define a estrutura de um recurso envolvido na aplicação (dado, documento), em termos dos papéis que o recurso pode apresentar durante seu ciclo de vida. *Classes de processos* integram as classes de recursos e de agentes, permitindo a descrição do trabalho realizado quanto à sua organização e cooperação entre agentes.

2.3 Definição de Classes e de Papéis

Uma *classe* é definida através do seu nome e de um conjunto de papéis. Um *papel* é definido por um nome, um conjunto de propriedades (descrições abstratas dos tipos de dados implementados como variáveis de instâncias), um conjunto de estados que este papel pode apresentar, um conjunto de mensagens que este papel pode enviar ou receber e um conjunto

de regras (regras de transição de estados e regras de integridade). Nas classes de agentes, os papéis apresentam ainda as decisões que estes agentes podem tomar, desencadeando regras de transição. Condições escritas em lógica temporal, associadas a regras de transição de estados, restringem a execução destas transições podendo também representar egras de integridade.

As *propriedades* podem ser estáticas (não mudam de valor ao longo da existência da instância) ou dinâmicas (quando podem assumir diversos valores com a passagem do tempo). Os valores das propriedades dinâmicas são associados ao tempo de transação (instante temporal em que o valor da propriedade foi introduzido no banco de dados que modela a aplicação) e ao seu tempo de validade (instante de tempo em que a informação inicia a modelar a realidade).

A cada propriedade definida (estática ou dinâmica) deve ser associado um *domínio*. Um domínio é definido através do nome de um objeto, de um conjunto de objetos ou de uma lista de objetos. O modelo TF-ORM apresenta um conjunto de objetos pré-definidos para serem utilizados na definição destes domínios, denominados tipos de dados (por exemplo *integer*, *real*, *date*).

Os *estados* que um papel pode assumir são definidos através de seus nomes. Papéis diferentes apresentar nomes de estados iguais — os nomes dos estados somente serão utilizados nas regras de transição de estados do papel onde são definidos.

Na definição das *mensagens* de um papel deve ser identificado, para cada mensagem, se é enviada ou recebida, e qual o papel que recebe/envia a mensagem. Mensagens podem relacionar papéis de um mesmo objeto (mesma classe), assim como papéis de objetos de classes diferentes. Mensagens podem apresentar parâmetros, através dos quais são passados os valores entre os papéis/classes. Ao definir uma mensagem deve também ser definido o seu conjunto de parâmetros, sendo necessário associar a cada um deles um domínio.

As *decisões* podem ser tomadas somente por classes de agentes. Podem apresentar também um conjunto de parâmetros, através dos quais são tranferidos valores relativos à decisão tomada pelo agente.

As *regras de transição de estados* apresentam a seguinte forma geral:

r_n: state(s_1), msg(mi) $\Rightarrow$ msg(mo), state(s_2); (<condição de transição>)

Seu significado é o seguinte: estando a instância no estado inicial s_1, recebendo a mensagem mi e sendo a condição de transição satisfeita, é enviada a mensagem mo e é realizada a transição para o estado s_2. Todos os elementos desta regra são opcionais, inclusive a condição de transição. Várias mensagens podem ser recebidas (a regra é ativada quando todas as mensagens tiverem chegado, em qualquer ordem e tempo) e várias mensagens podem ser enviadas, para diferentes papéis de diferentes classes.

A *regra de integridade* apresernta a seguinte forma:

constraint (<pré-condição> $\Rightarrow$ <pós-condição>)

Sempre que a pré-condição for satisfeita, a pós-condição também deve ser satisfeita.

O modelo TF-ORM possui uma linguagem de lógica temporal de primeira ordem para ser utilizada na condição de transição das regras de transição de estados e nas condições das regras de integridade. A descrição desta linguagem pode ser encontrada no Relatório de Pesquisa n° 258.

2.4 Papel Básico, Mensagens e Propriedades Pré-definidas

Toda classe apresenta um *papel básico*, no qual são descritas propriedades herdadas pelos demais papéis e que descreve as características iniciais das instâncias dos demais papéis. No papel básico são definidas somente propriedades (globais para todos os outros papéis), regras de transição de estados que manipulam os demais papéis, nas quais podem ser utilizadas somente mensagens pré-definidas no modelo, e regras de integridade também válidas para todos os demais papéis. Os estados do papel básico são pré-definidos (active e suspended). As mensagens pré-definidas, que podem ser utilizadas tanto nas regras do papel básico como nas dos demais papéis, estão descritas na tabela 2.1.

Tabela 2.1 - Mensagens Pré-definidas

create_object (C, Oid)	cria uma instância da classe C, cujo identificador é devolvido pelo sistema através do argumento OId
suspend_object (Oid)	suspende temporariamente a atividade da instância de classe cujo identificador é fornecido. Esta instância não pode enviar nem receber mensagens, com excessão das mensagens que indicam a retomada de suas atividades ou a sua destruição
resume_object (Oid)	permite à instância de classe identificada por Oid a retomada de suas atividades
kill (Oid)	termina a existência de uma instância de classe
allow_role (Oid, R)	permite a criação de instâncias de um determinado papel da classe identificada por Oid. Não é necessária a identificação da classe quando a mensagem for utilizada na mesma classe em que está sendo criada a instância do papel
add_role (Oid, R, Rid)	cria uma instância do papel R da instância de classe identificada por Oid, sendo o identificador desta instância de papel devolvido através do argumento Rid
suspend_role (Rid)	suspende temporariamente as atividades da instância de papel identificada por Rid
resume_role (Rid)	retoma as atividades da instância de papel Rid
terminate_role (Rid)	termina a existência da instância de papel Rid
forbid_op (Rid, Direção, Mensagem)	impede que uma determinada mensagem seja temporariamente enviada ou recebida pela instância de papel Rid
allow_op (Rid, Direção, Mensagem)	permite novamente o envio ou recebimento de uma mensagem que tinha sido desabilitada

O modelo apresenta uma classe pré-definida, *OBJECT*, da qual todas as outras classes definidas são subclasses. As características desta superclasse são herdadas por todas as demais classes, não podendo ser redefinidas. No papel-base de *OBJECT* são definidos: (i) uma propriedade estática *oId* que armazena o identificador da instância da classe, um identificador

único, cujo valor é gerado pelo sistema; (ii) uma propriedade dinâmica *object_instance* que guarda os instantes de tempo em que a instância foi criada, suspensa e reativada; (iii) uma propriedade dinâmica *end_object* que, quando seu conteúdo for *nonull*, tem associado o instante de tempo em que a instância foi destruída (o tempo de transação desta destruição e seu tempo de validade); (iv) o conjunto de estados do papel-base, o qual pode estar somente ativo ou desativado; e (v) o conjunto inicial de mensagens pré-definidas. Todos os papéis-base das classes definida através do TF-ORM herdam este papel-base, podendo ser definidas, além das regras correspondentes a cada papel-base, propriedades adicionais.

A classe *OBJECT* apresenta um papel *ROLE*, cujas características são herdadas por todos os demais papéis das classes derivadas. Neste papel são definidos: (i) uma propriedade estática *rId* que armazena o identificador da instância do papel, um identificador único gerado pelo sistema; (ii) uma propriedade dinâmica *role_instance* onde são armazenados os instantes de tempo em que a instância é criada, suspensa e reativada; e (iii) uma propriedade dinâmica *end_role* que guarda o instante de tempo em que a instância foi destruída.

Os valores das propriedades *oId* e *rId* podem ser referenciados nas condições que expressam as restrições de integridade e na linguagem de descrição dos métodos que implementam as mensagens. São utilizados para identificar respectivamente uma instância da classe e uma instância de um papel. Não podem, entretanto, ser modificados pelos programas de aplicação.

3. DESCRIÇÃO DO AMBIENTE

3.1 Bancos de Dados

Três diferentes bancos de dados são utilizados pelo sistema que implementa este ambiente: o banco de dados especificação, a biblioteca de classes e a biblioteca de manutenção.

No *banco de dados especificação* é armazenada uma especificação enquanto está sendo construída. Somente uma aplicação pode ser especificada de cada vez. Ao final do processo a especificação construída é armazenada em um arquivo externo e o banco de dados especificação é liberado. Cada especificação realizada é associada a um domínio de aplicação. As classes armazenadas na biblioteca de classes também estão associadas a domínios de aplicação específicos. A pesquisa na biblioteca a partir de domínios de aplicação mais restritos diminui o tempo de busca para reutilização de classes.

A *biblioteca de classes* armazena as classes a serem reutilizadas durante as especificações. A biblioteca é inicialmente povoada com um conjunto de classes abstraídas do domínio de aplicação considerado, um subdomínio da área de sistemas de informação de escritórios. São escolhidas aquelas classes que apresentam possibilidade de serem utilizadas em mais de uma aplicação deste domínio. Posteriormente esta biblioteca sofre processos de manutenção periódicos, com base nos processos de reutilização realizados durante as especificações. Nestas manutenções classes novas podem ser incluídas na biblioteca, classes que não mostraram utilidade para reutilização podem ser retiradas e classes já existentes podem ser modificadas.

Na *biblioteca de manutenção* são armazenadas as informações que podem ser úteis nos processo de manutenção. São armazenadas todas as informações referentes às reutilizações de classes efetuadas—nomes das classes reutilizadas, sinônimos e descrições de classes e papéis novos, quando estas forem fornecidas pelo usuário. Além disso, uma cópia das novas classes definidas também é armazenada.

3.2 Ferramentas de Apoio

O ambiente apresenta duas ferramentas independentes, uma para cada tipo de usuário previsto: (i) ferramenta de apoio ao engenheiro de classes e (ii) ferramenta de apoio ao projetista de aplicações. Somente uma das ferramentas pode ser executada de cada vez. O objetivo das ferramentas é apoiar as tarefas dos usuários, apresentando uma interface interativa coloquial.

A ferramenta de apoio ao engenheiro de classes apresenta duas funções independentes: (i) contrução inicial de uma biblioteca de classes, e (ii) manutenção de uma biblioteca já existente, com base nas informações contidas na biblioteca de manutenção. A construção de uma biblioteca é efetuada pela definição de um conjunto de classes e de suas metainformações. A manutenção desta biblioteca deve ser feita periodicamente, com base nas informações armazenadas na biblioteca de mantenção.

A ferramenta de apoio ao projetista de aplicações tem por objetivo ser utilizada durante a especificação de uma aplicação. O projetista de aplicações é guiado na definição de classes de objetos segundo o modelo TF-ORM. A definição de classes é atemporal, isto é, o esquema é estático, sendo entretanto representados os aspectos temporais da aplicação. A definição de

classes é feita apoiada na reutilização de classes anteriormente definidas, contidas na biblioteca de classes.

Na figura 3.1 estão representados os bancos de dados utilizados pelo ambiente, as ferramentas disponíveis para manipular as informações referentes a estes bancos de dados e os fluxos de informações efetuados pelas ferramentas entre os bancos de dados.

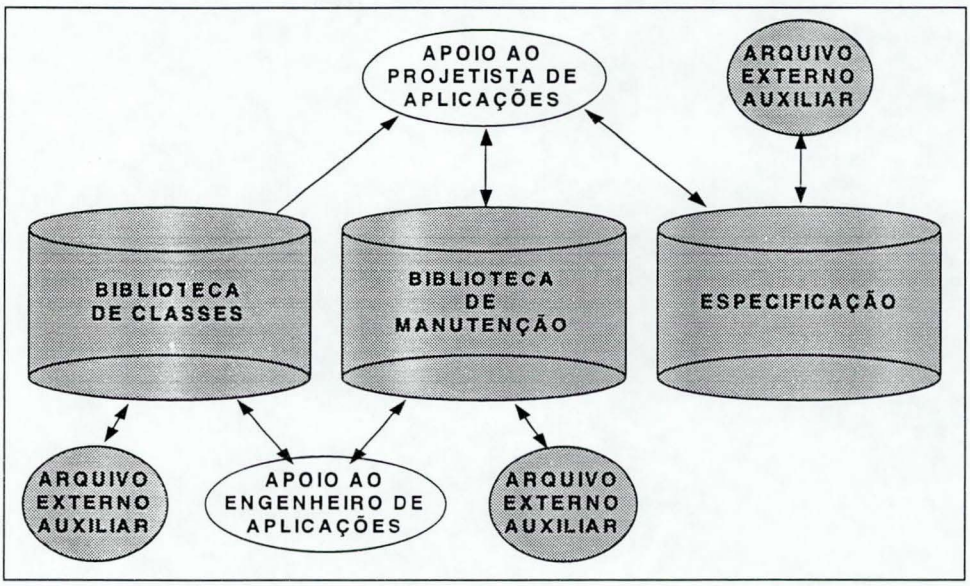

Figura 3.1: Bancos de Dados e Ferramentas do Ambiente

4. FERRAMENTA DE APOIO À ESPECIFICAÇÃO

A ferramenta de apoio à especificação implementada é textual, interativa. Através de perguntas e respostas o usuário (projetista de aplicações) é guiado através da especificação.

Inicialmente é identificado o domínio de aplicação do sistema a ser especificado. Poderá ser utilizado um dos domínos já existentes na biblioteca de classes ou definido um novo domínio.

Após a identificação do domínio, as classes da biblioteca associadas a este domínio podem ser reutilizadas. A ferramenta permite reutilizar classes também de outros domínios, quando assim desejado. Várias opções de pesquisa das classes disponíveis na biblioteca são oferecidas — todas as classes da biblioteca, classes de um determinado domínio de aplicação, classes de agentes, classes de recursos, classes de processos, sinônimos de um nome de classe, descrição textual de uma determinada classe, etc.

Uma classe pode ser definida como:

- nova classe, independente das classes da biblioteca;
- cópia de uma classe encontrada na biblioteca de classes;
- modificação de uma classe da biblioteca.

Quando não for encontrada nenhuma classe na biblioteca, adequada à reutilização na aplicação em desenvolvimento, o projetista de aplicações opta por definir uma nova classe. A ferramenta de apoio guia esta definição, de acordo com o modelo utilizado, sendo a classe armazenada na base de dados da especificação e uma cópia feita para a biblioteca de manutenção.

O nome de uma classe deve ser único em uma especificação. Nada impede que uma nova classe seja criada com nome igual a outra contida na biblioteca de classes, sem se basear nela. Cabe ao engenheiro de aplicações, posteriormente, analisar a possibilidade de introduzir esta nova classe na biblioteca, devendo então seu nome ser alterado.

Quando uma classe for identificada na biblioteca de classes, duas formas de reutilização podem ser empregadas: com ou sem modificações. No caso de ser utilizada uma classe sem modificações, as informações correspondentes são copiadas para a base de dados. As informações referentes a estatísticas e às metainformações não são copiadas para a base de dados, embora o projetista de aplicações tenha acesso às mesmas para informação. Em seguida é incluída na biblioteca de manutenção uma indicação da reutilização.

Quando uma classe é definida com base em uma classe encontrada na biblioteca, necessitando de alterações, a ferramenta faz uma cópia, com controle do especificador, das informações contidas na biblioteca para a base de dados especificação. Uma vez copiada a classe, a mesma pode ser modificada conforme as necessidades. Podem ser acrescentados novos atributos, novos papéis, modificados ou suprimidas características da classe.

Uma especificação em TF-ORM pode ser realizada em vários níveis de detalhamento. O nível de detalhamento considerado para uma determinada informação é armazenado junto a esta informação na base de dados especificação. O nível inicial de detalhamento de uma classe é sempre considerado como sendo de número 1, recebendo os detalhamentos posteriores desta classe números incrementados de uma unidade cada um. O maior nível de detalhamento associado à informações de uma classe representa a especificação final desta classe. São estas as informações referentes à classe copiadas para a biblioteca de manutenção.

Diversos **testes de consistência** são realizados por esta ferramenta, em três momentos diferentes: (1) durante a especificação de uma classe, (2) logo ao final da especificação de uma classe, e (3) ao final da especificação da aplicação. Entre os testes de consistência realizados durante a especificação de uma classe podem ser citados:

- nomes únicos para classes de objetos, para nomes de papéis dentro de uma determinada classe de objetos, e para nomes de estados e de mensagens dentro de um determinado papel de uma classe de objetos;
- referência a propriedades e mensagens dentro de expressões lógicas, pertencentes às propriedades e às mensagens definidas para o papel considerado, para o papel básico da classe considerada ou para o papel básico de uma superclasse da classe considerada;
- consistência nas mensagens constantes das regras de transição compatível com sua localização na regra;
- consistência de número de argumentos em mensagens de expressões lógicas.

Logo após o término da especificação de uma classe são realizados os seguintes testes:

- se em cada conjunto de regras de transição dos diferentes papéis aparece o estado inicial, alcançado pelo recebimento da mensagem de criação de uma instância do papel considerado;
- se cada estado definido aparece pelo menos em alguma regra de transição de estados;
- se algum dos estados é inatingível a partir da regra inicial;
- se todas as mensagens aparecem nas regras de transição de estados.

Ao final de uma especificação a ferramenta realiza testes de consistência complementares, verificando a especificação como um todo. Neste momento são verificados:

- se todas as classes definidas como superclasses e como classes componentes de novas classes estão definidas;
- se todas as classes de objetos utilizadas nos domínios de propriedades estão definidas;
- se todos os papéis de classes utilizados na definição de mensagens estão definidos;
- se as mensagens enviadas por um determinado papel para outro aparece listada no conjunto de mensagens recebidas por este segundo papel e se apresenta os mesmos argumentos.

Quando o projetista dá por encerrada uma especificação, após a realização dos testes de consistência finais, a ferramenta faz uma cópia da versão final das classes para a biblioteca de manutenção para posterior análise do engenheiro de aplicações. São copiadas as novas classes definidas e as classes reutilizadas porém modificadas. Indicações sobre reutilização de classes são também armazenadas — nomes das classes reutilizadas sem modificações, nomes anteriores de classes reutilizadas caso o nome tenha sido alterado e nomes das classes reutilizadas com modificações. Neste último caso, são também armazenadas algumas informações fornecidas pelo projetista de aplicações, referentes às modificações efetuadas, para posterior análise por parte do engenheiro de aplicações e resultando eventualmente na modificação da classe constante da biblioteca de classes. Algumas metainformações relativas às classes são também incluídas na biblioteca de manutenção, tais como sinônimos dos nomes das classes e dos papéis, explicações textuais sobre classes e papéis, e explicações textuais das condições lógicas.

5. INSTALAÇÃO DA FERRAMENTA

A Ferramenta foi desenvolvida usando-se o interpretador Arity/Prolog versão 6.1.46 da Arity Corporation, porém possivelmente funcionará na versão 5 ou posterior. A Ferramenta não pôde ser executada com êxito em versões anteriores à 4 (inclusive) do mesmo interpretador. Ela não foi testada em outros interpretadores, porém, possivelmente poderá ser usada com poucas ou nenhuma modificação em interpretadores semelhantes.

No caso de se usar um interpretador com um ambiente integrado de desenvolvimento recomenda-se que opções como bordas e cores sejam desativadas[1]

A instalação da Ferramenta é simples, bastando copiar todos os arquivos da mesma, inclusive as Bibliotecas de Classes em um diretório único. Os arquivos necessários para uso da ferramenta são:

`SAIDAS1.ARI`	- Rotinas de E/S
`SAIDAS2.ARI`	- Rotinas de E/S
`SAIDAS4.ARI`	- Rotinas de E/S
`SAIDAS3.ARI`	- Rotinas de E/S
`INI.ARI`	- Carregador e configuração da ferramenta
`GENER.ARI`	- Rotinas diversas
`TFORM.ARI`	- Programa principal
`INTEGRID.ARI`	- Testes de integridade
`ALTERA.ARI`	- Alteração de informações
`PAPEIS.ARI`	- Especificação de papéis
`DUMP.ARI`	- Salvamento de informações para depuração(opcional)

Em alguns casos será necessária a modificação do arquivo de inicialização, descrita na próxima seção.

5.1 Configuração

Informações sobre configuração da Ferramenta estão localizadas no arquivo `INI.ARI`. Normalmente não é necessária qualquer modificação sobre a mesma.

Os parâmetros que podem ser configurados são:

`defClassLib(NomeArquivo).`— define o nome do arquivo da Biblioteca de Classes. O valor padrão é `CLASSES.BIB` e está definido como:

```
defClassLib('CLASSES.BIB').
```

O nome do arquivo deverá estar entre aspas como no exemplo acima.

`defMaxLines(Linhas de texto - 2).`define o número de linhas de texto disponíveis no terminal usado menos 2. Este deve ser um valor numérico, como no seguinte exemplo:

---

[1]No Arity/Prolog 6 deve-se mudar a altura da janela de forma que a última linha da tela seja apagada; a cor de texto para branco com fundo preto sem brilho e fundo e frente das bordas para preto sem brilho.

```
defMaxLines(23).
```

Esta informação é usada pela ferramenta ao gerar listagens com telas "roláveis".

`setupLog.`– define se deve ou não ser ativada o *logging* de todos os dados digitados pelo usuário para o arquivo `LOGFILE`. Esta opção é útil quando se deseja verificar todos os passos executados até o surgimento de um erro, para posterior análise e depuração do programa. Para habilitar esta função deve-se inserir esta linha ou tirar o símbolo de comentário (%) da mesma no arquivo `ini.ari` junto com as outras informações de configuração. Para desabilitar, insira o símbolo de comentário no início da linha.

Para habilitar a função:

```
setupLog.
```

6. UTILIZAÇÃO DA FERRAMENTA

6.1 Considerações Iniciais

Como a Ferramenta está implementada na linguagem Prolog, existem algumas considerações que devem ser tomadas para a sua utilização:

1. ao entrar um dado qualquer, sempre deve ser digitado um ponto '.' antes de pressionar `RETURN`;
2. no início de palavras (respostas para o programa, nomes de classes, estados etc.) não devem ser utilizadas letras maiúsculas;
3. sinais de pontuação, espaços e caracteres acentuados não devem ser usados, exceto em frases, como na descrição de classes (no caso deve se colocar a frase entre aspas simples) e o ponto no final de um dado (item 1);
4. a utilização direta de algum predicado explicado neste documento deve ser feita no *prompt* do interpretador;
5. não use `CTRL-BREAK` para terminar o programa. Apesar de se poder reiniciar o programa sem apagar os dados definidos até o momento, isso pode provocar estados de inconsistência interna na Ferramenta. No caso de ser necessário abortar o programa com `CTRL-BREAK`, salve as bibliotecas com o predicado `salva_bib/1`, saia do interpretador, retorne ao mesmo e execute novamente a ferramenta.

6.2 Inicialização

Antes de se usar a Ferramenta, deve-se executar o interpretador Prolog a partir do diretório principal da Ferramenta. Para carregar a ferramenta e iniciar, digite `['ini.ari']` logo após o prompt do interpretador Prolog (`?-`). Este predicado carrega os arquivos pertinentes à Ferramenta no interpretador[2].

Se a ferramenta for carregada com sucesso, o *prompt* `?-` deverá aparecer no interpretador. Caso contrário é possível que algum erro tenha ocorrido durante o carregamento da ferramenta. Verifique se todos os arquivos estejam no diretório da ferramenta (que deve ser o diretório corrente) e que nenhum arquivo tenha sido modificado acidentalmente. Caso tenha ocorrido o último, copie novamente o(s) arquivo(s) original(is) da ferramenta.

Em caso de sucesso na carga da ferramenta, o comando menu. deve ser utilizado para inicia-la.

A tela de inicialização da ferramenta com mensagens da ferramenta, do interpretador e predicados digitados, é mostrada na Fig. 6.1.

6.3 Especificação

6.3.1 Definição do Domínio de Aplicação

Ao se executar `menu.` a tela de apresentação será exibida e será perguntado se deseja-se carregar uma especificação previamente salva em disco. Ao se carregar uma aplicação do

---

[2]Não carregue a Ferramenta por mais de uma vez, antes de apagá-la da memória. Isso pode trazer efeitos colaterais imprevisíveis. Para carregar a Ferramenta sem antes sair do interpretador, use o comando `recon`.

```
?- ['ini.ari'].

Ferramenta de Apoio a Especificacao de Aplicacoes...

...

Carregando Ferramenta.........Ferramenta Carregada

                                    ...

Digite menu para iniciar.

yes.

?- menu.

...
```

Figura 6.1: Tela inicial da ferramenta

disco é pedido o caminho do diretório de onde se quer uma listagem das especificações presentes.

Digite o caminho entre aspas simples, seguido de um ponto. Por exemplo:

```
Qual o diretório (ponha o caminho entre apostrofes)?
:? '\TFORM\APLICACAO'.
```

Se for desejado o diretório corrente digite:

```
Qual o diretório (ponha o caminho entre apostrofes)?
:? '.'.
```

O nome da especificação a ser carregada não deve ser seguido da extensão do arquivo (que é `.ESP`).

Se não for desejado a carga de uma especificação gravada anteriormente deve-se fornecer o nome de um dos domínios disponíveis na biblioteca de classes ou um um novo nome de domínio a ser definido. No caso de se escolher um domínio de aplicação da biblioteca de classes, a listagem dos nomes de domínios disponíveis será mostrada. Todas as consultas sobre classes para reuso usarão o domínio escolhido como escopo padrão para pesquisa.

6.3.2 Menu Inicial

Após a definição do Domínio de Aplicação ou carregamento de uma especificação é apresentado o menu inicial (Fig. 6.2)

```
Operacoes que podem ser efetuadas sobre a Aplicacao exemplo

        1 - Especificacao de Informacoes

        2 - Alteracao de Informacoes

        3 - Listagem de Especificacoes ja armazenadas

        4 - Apagar Especificacao atual

        fim - para terminar

   Qual a sua opcao? (1/2/3/4/fim)
```

Figura 6.2 - Menu inicial

1 - Especificação de Informações é usado para criar novas classes e papéis e incluir propriedades, mensagens, estados, regras e decisões aos papéis.

2 - Alteração de Informações leva ao menu de alteração de nomes de classes, papéis etc., de remoção das mesmas e outros.

3 - Listagem de Especificações já armazenadas oferece listagens das diferentes informações armazenadas na ferramenta.

4 - Apagar Especificação atual remove todas as informações referentes a especificação carregada na memória no momento.

fim - Terminar processo de especificação. A ferramenta questiona se a especificação deve ser salva em disco, salvando-a se necessário e termina.

6.4 Especificação de Informações

Esta opção é usada para criar classes e adicionar informações a classes já existentes. A figura 6.3 mostra o menu de especificação de informações.

1) **Conjunto de Classes-** define um conjunto de classes e seus respectivos papéis além de todas informações relativas aos papéis;
2) **Uma nova Classe-** define uma única classe;
3) **Um novo Papel-** define um papel dentro de uma classe já criada. Outras informações relativas a um papel, como propriedades e mensagens, também são definidas;
4) **Conjunto de Propriedades-** define um conjunto de propriedades (estáticas e dinâmicas) em um papel já definido;
5) **Uma Propriedade-** define uma propriedade em um papel;
6) **Conjunto de Mensagens-** define um conjunto de mensagens a serem recebidas e enviadas por um papel;
7) **Uma Mensagem**- define uma mensagem;
8) **Conjunto de Estados**- define um conjunto de estados que um papel pode assumir;
9) **Um Estado-** define um estado;

```
          Processo de Especificacao de Informacoes

    Definicoes que podem ser efetuadas:
       1 - Conjunto de Classes
       2 - Uma nova Classe
       3 - Um novo Papel
       4 - Conjunto de Propriedades
       5 - Uma Propriedade
       6 - Conjunto de Mensagens
       7 - Uma Mensagem
       8 - Conjunto de Estados
       9 - Um Estado
      10 - Conjunto de Regras
      11 - Uma Regra
      12 - Conjunto de Decisoes
      13 - Uma Decisao
     fim - para terminar
    Qual a sua opcao (Codigo de 1 a 13):
```

Figura 6.3 - Menu de especificação de informações

10) **Conjunto de Regras-** define um conjunto de regras de transição de estados e de integridade de um papel;

11) **Uma Regra-** define uma regra de transição de estados e de integridade;

12) **Conjunto de Decisões-** define um conjunto de decisões que um papel de uma classe agente pode tomar;

13) **Uma Decisão-** define uma decisão.

A ordem normal de definição de uma classe é a seguinte:

1. **Classe-** define-se o tipo da classe, nome e metainformações para a biblioteca de manutenção ou nome da classe da biblioteca de classes (BC) a ser reutilizada.
2. **Papéis-** inclusão de papéis à nova classe, fornecendo o nome do papel e informações para a biblioteca de manutenção.
   a) **Propriedades-** propriedades estáticas e dinâmicas de cada um dos papéis;
   b) **Estados-** estados que o papel pode assumir;
   c) **Decisões-** definição de decisões que este papel pode tomar. Esta informação só é disponível na definição de uma classe agente;
   d) **Mensagens-** mensagens que o papel pode enviar ou receber de outros papéis/classes;

e) **Regras de Transição de Estados-** definição de mudanças de estados do papel de acordo com o estado inicial, mensagem recebida, mensagem enviada, o estado após a transição e condição associada à transição.

f) **Regras de Integridade**

3. **Testes de Integridade** Esta etapa é efetuada automaticamente pela ferramenta. Se erros ou inconsistências forem detectados, eles são mostrados e a especificação é retomada.

Apesar da ordem apresentada acima ser a mais natural, cada informação pode ser definida individualmente, sem seguir esta ordem, iniciando os processos de especificação desejados no menu "Especificação de Informações". Neste caso, o processo de especificação também é semelhante ao da especificação de uma classe na ordem normal.

Ao se definir uma nova classe, a ferramenta automaticamente gera o papel base (`baserole`) e algumas informações relativas a ela, como estados e mensagens pré-definidas.

6.4.1 Definição de Classes

A primeira coisa a se fazer na especificação de uma classe é escolher o tipo de classe dentre as três disponíveis:

- agente;
- recurso;
- processo.

A letra correspondente ao tipo desejado (`a`,`r` ou `p`) deve ser digitada. Definido o tipo da classe, pode se importar uma classe da biblioteca de classes (BC) para reuso ou contruir uma nova classe desde o início.

6.4.1.1 Definição da Classe com Reuso

Se for escolhida a importação de uma classe, a ferramenta perguntará pelo nome do domínio de aplicação da BC de onde quer se retirar a classe (somente se no início da especificação não foi usada uma aplicação da BC). Serão listadas as classes existentes da aplicação escolhida, além de seus respectivos sinônimos e descrições, quando disponíveis. Caso exista alguma classe que aparentemente atenda às suas necessidades, indique o nome ou sinônimo[3] desta classe. A listagem de cada uma dos papéis da classe será mostrada para exame. Os papéis sem utilidade para a especificação podem ser excluídos da reutilização, porém neste caso, o papel base (`baserole`) deverá ser mudado manualmente, retirando todas as referências (mensagens e outros) aos papéis não incluídos na aplicação. O nome da classe reutilizada poderá ser mudada antes da importação da BC.

Cada papel da classe é mostrado e será perguntado se ele deverá ser importado (reutilizado) com ou sem alterações. Se for desejado a modificação do papel, o papel será marcado internamente e novas informações poderão ser incorporadas posteriormente ao papel. A remoção ou alteração delas só poderá ser feita posteriormente através de outras funções da ferramenta.

---

[3]Mesmo que a classe seja escolhida pelo seu sinônimo, posteriormente ela será referida pelo seu nome primário/original.

6.4.1.2 Definição de uma nova classe

Se não for desejada a reutilização de uma classe, deve-se informar o nome da classe e opcionalmente, sinônimos e uma descrição da classe a ser criada.

Podem ser definidos mais de um sinônimo para uma classe. Para terminar a inclusão de sinônimos entre `fim`.

6.4.2 Papéis

Todas as classes de uma aplicação possuem um papel-base. Na definição de uma classe, o papel-base (`baserole`) é automaticamente gerado com algumas informações pré-definidas. Estas informações são:

1. Estados
   a) active
   b) suspended
2. Mensagens (envio e recebimento)
   a) create_object
   b) suspend_object
   c) resume_object
   d) kill
   e) allow_role
   f) forbid_role
   g) add_role
   h) resume_role
   i) suspend_role
   j) terminate_role
   k) forbid_op
   l) allow_op
3. Propriedades
   a) oId (object ID): integer
   b) end_object: instant
   c) object_instance: instant

As informações pré-definidas não são incluídas na especificação salva em disco. Os estados e mensagens do papel-base não são mostrados nas listagens da especificação.

No papel-base não podem ser incluídas novos estados ou mensagens. Somente propriedades e regras podem ser definidas e as informações pré-definidas do papel-base não podem ser alteradas.

A definição de um papel (com exceção do papel-base) começa com a especificação de seu nome, sinônimos (opcional) e descrição (opcional).

A seguir passa-se à definição das demais informações.

6.4.3 Propriedades

Primeiramente deve-se definir o nome da propriedade a se incluir no papel. Este nome não deve coincidir com o de outras propriedades deste papel. Se for desejada a modificação de alguma informação relativa a uma propriedade já existente, deve se apagar esta propriedade primeiro e criar uma nova com o mesmo nome.

A seguir deve-se definir seu tipo: estático ou dinâmico. Propriedades estáticas são propriedades de caráter invariável com relação ao tempo, ou seja, possuem valor constante. Exemplos de propriedades estáticas seriam a data de nascimento ou o número do CPF de uma pessoa. Já as propriedades dinâmicas podem ter seu valor alterado ao longo da vida do papel, como o horário de uma consulta ou o endereço de uma pessoa.

O domínio (tipo de dado) da propriedade pode ser simples ou complexo. Os tipos de dados simples disponíveis são:

boolean	booleano	**integer**	inteiro
real	real	**string**	texto
instant	instante	**date**	data
year	ano	**month**	mês
day	dia	**time**	hora (hh:mm:ss)
hour	hora	**minute**	minuto
week	semana	**semester**	semestre
interval	intervalo	**span**	duração
class	classe	**string list**	lista de strings (enumeração)

Listas (*list of*) e conjuntos (*set of*) são tipos complexos. As listas ou conjuntos podem ser de qualquer um dos tipos simples citados anteriormente.

Para se escolher o tipo de domínio da propriedade basta digitar o número correspondente ao tipo desejado[4].

Para classes deve ser dado ainda o nome da classe desejada. Para intervalos o tipo dos limites (**integer**, **instant**, **date**, **year**, **month**, **day**, **time**, **hour** e **minute**) e o tipo do intervalo (**closed**, **open**, **upopen**, **downopen**, **upfloat** e **downfloat**).

Para duração deverá ser escolhida uma das unidades de medida temporal disponíveis: **year**, **month**, **day**, **hour**, **minute**, **week** e **semester**.

Para os tipos complexos deve se ainda escolher o tipo de lista ou conjunto. Para isto deve se seguir os mesmos passos da definição normal de tipos simples.

A propriedade será apresentada ao final da especificação de todas as informações da propriedade para confirmação de sua criação. Caso não se queira criar esta propriedade, todas as informações relativas a ela serão apagadas.

---

[4]Os tipos marcados com um asterisco * no menu não estão implementados e não podem ser escolhidos.

Para terminar a definição de propriedades, digite `fim`.

6.4.4 Estados

A definição de estados é bastante simples. Digite cada um dos estados definidos seguidos de um ponto e `Return`. Para terminar de acrescentar estados, entre `fim`.

Obs.: Não podem ser acrescentados estados ao papel-base.

6.4.5 Mensagens

Mensagens não podem ser definidas no papel-base de uma classe. Somente as mensagens pré-definidas estão disponíveis.

Várias mensagens poderão possuir o mesmo nome, porém recomenda-se que se evite isto por dificultar a compreensão da especificação posteriormente.

Após fornecer o nome da mensagem, deve se indicar se a mensagem sendo definida é uma mensagem a ser enviada (*to*) ou a ser recebida de outro papel[5] (*from*). Digite o número correspondente à opção. A seguir forneça o nome da classe que irá receber ou mensagem ou mandar a mensagem. Não é necessário que a classe já exista neste momento, ela poderá ser criada mais tarde. O mesmo se aplica ao nome do papel a receber (ou enviar) a mensagem.

Agora deverão ser indicadas os argumentos que esta mensagem irá enviar ou receber. Se a mensagem não possui argumentos, indique 0 (zero) como número de argumentos. O restante da definição dos argumentos é semelhante à definição de propriedades.

Decisões

Decisões são um tipo especial de mensagem que só existem en classes agente. A definição delas é semelhante à das mensagens normais, porém ela somente pode ser enviada, portanto a informação sobre isto é excluída da definição de decisões.

6.4.6 Regras de Transição de Estados

A especificação de uma regra de transição de estados inicia com a definição do nome do estado inicial do papel, antes da transição. Esta informação é opcional, se ela não for necessária forneça o valor `null`.

A seguir indica-se o nome da mensagem que o papel deve receber para a transição ser feita. Indique `null` para deixar este campo em branco. No papel-base, somente podem se utilizar as mensagens pré-definidas e nos demais papéis as mensagens já devem ter sido definidas previamente. Se o estado inicial do papel-base for *suspended* as únicas mensagens que ele poderá receber são *resume_role* e *kill.*

O mesmo se aplica para a mensagem a ser enviada na transição de estados.

O estado final do papel na transição também é opcional. Se esta informação não for necessária, indique `null`.

Por fim, opcionalmente pode-se definir uma condição para a transição. Esta condição deverá estar entre apóstrofes; se não existir condição para a transição, digite dois apóstrofes seguido de um ponto. A ferramenta não fará nenhuma verificação sobre esta condição.

---

[5] Ao se definir uma mensagem a ser enviada, ela será automaticamente criada no papel a receber essa mensagem, se este papel já existir e a mensagem ainda não estiver especificada.

```
     Processo de Alteracao de Especificacoes Armazenadas
     Operacoes que podem ser efetuadas:
Alteracao do nome de --> 1 - Uma Classe
                         2 - Um Papel
                         3 - Uma Propriedade
                         4 - Uma Mensagem
                         5 - Um Estado
                         6 - Uma Decisao
       Eliminacao de --> 7 - Uma Classe
                         8 - Um Papel
                         9 - Uma Propriedade
                        10 - Uma Mensagem
                        11 - Uma Decisao
                        12 - Um Estado
                        13 - Uma Regra de Transicao
                        14 - Uma Regra de Integridade

                        15 - Mudanca na numeracao de regras
                        fim - para terminar
          Qual a sua opcao? (codigo de 1 a 13):
```

Figura 6.4 - Menu de alteração de especificações

6.4.7 Regras de Integridade

As regras de integridade também não são submetidos a testes de consistência. Elas são definidas através de uma pré-condição e uma pós-condição. Na aplicação, a pós-condição é verificada somente se a pré-condição for satisfeita.

6.5 Alteração de Especificações

As alterações que podem ser feitas sobre a especificação são a alteração de nomes de informações, remoção de informações ou mudança no ordenamento de regras de transição. A figura 6.4 mostra o menu de alterações.

6.5.1 Alterações de Nome

Podem ser mudados os nomes de classes, papéis, propriedades, mensagens, estados, regras de integridade e de transição de estados.

Obs.: As informações pré-definidas não podem ter seu nome mudado.

O processo de alteração de nomes é semelhante para todas as informações da especificação. É necessário ir escolhendo os nomes de classes, papéis etc. até alcançar a informação que se deseja modificar.

Por exemplo: se o nome de uma propriedade deve ser mudado, os seguintes passos devem ser seguidos:

1. escolher o item 3 no menu de alteração de especificações;
2. fornecer o nome da classe a que pertence a propriedade;
3. fornecer o nome do papel;
4. fornecer o nome da propriedade cujo nome se quer mudar;

5. fornecer o novo nome da propriedade.

6.5.2 Remoção de Informações

A remoção de especificações feitas também pode ser feita para qualquer tipo de informação, inclusive a especificação toda.

Para se apagar a especificação toda, escolha o quarto item do menu principal do programa. Após a confirmação da operação será perguntado se a especificação deve ser salva em disco. Recomenda-se que se faça isso, pois se a especificação for apagada sem ser gravada antes em disco, não haverá como recuperar estas informações. Depois de a especificação ser apagada o processo de especificação será reinicializado.

Para apagar outras informações os passos são semelhantes aos da alteração de nomes de informações. **Informações apagadas não poderão ser recuperadas posteriormente, portanto cuidado ao apagar especificações de informações de níveis de abstração altos (classes e papéis).**

As regras de transição têm uma ordem fixa dentro da especificação. A primeira regra definida é a primeira na ordem do papel. Regras novas que forem sendo definidas são acrescentados com numeração crescente a partir do número da última regra definida, sem levar em conta as regras de número inferior que eventualmente tenham sido removidas. Portanto ao remover uma regra tenha certeza de reordenar as regras.

6.5.3 Reordenamento de Regras de Transição

Para reordenar regras de transição de estados (após ter encontrado ela entre as classes e papéis já especificadas) forneça o número correspondente à regra cuja ordem se quer mudar, na listagem que será mostrada. Esta listagem mostrará o número de ordem correspondente a cada regra. A seguir forneça um o número da nova ordem da regra. Este número pode ser o de outra regra, neste caso as duas regras terão suas ordens trocadas entre si.

6.6 Listagem de Especificações

Os tipos de listagens de especificações disponíveis são (figura 6.5):

- Nomes de Classes;
- nomes dos papéis de uma classe;
- propriedades de um papel;
- estados de um papel;
- mensagens de um papel;
- regras de um papel;
- todas as informações referentes a uma classe e
- todas as informações referentes a um papel.

A obtenção de listagens das especificações é auto-explicativa. Basta escolher o tipo de listagem e fornecer as informações pedidas para encontrar as informações desejadas.

1) **Nome das classes especificadas-** mostra uma listagem simples com o nome de cada uma das classes especificadas;

2) **Nome dos papéis de uma classe-** mostra o nome de todos os papéis de uma classe;

3) **Propriedades de um papel-** mostra as propriedades estáticas e dinâmicas de um dado papel;
4) **Estados de um papel-** mostra a lista de estados definidos para um papel;
5) **Mensagens de um papel-** mostra as mensagens a serem enviadas e recebidas por um papel;
6) **Regras de um papel-** mostra as regras de integridade e de transição de estados definidos em um papel;
7) **Todas as informações de uma classe-** mostra cada um dos papéis definidos em uma classe, com seus conjuntos de informações associados;
8) **Todas as informações de um papel-** mostra todas as informações referentes a um papel.

```
Processo de Listagem de Especificacoes Armazenadas

 Listagens que podem ser efetuadas:

   1 - Nome das classes especificadas
   2 - Nome dos papeis de uma classe
   3 - Propriedades de um papel
   4 - Estados de um papel
   5 - Mensagens de um papel
   6 - Regras de um papel
   7 - Todas as informacoes de uma classe
   8 - Todas as informacoes de um papel
 fim - para terminar
  Qual a sua opcao (codigo de 1 a 8)?:
```

Figura 6.5 - Menu de listagem de especificações

7. MODELOS DE DADOS INTERNOS

Durante um processo de especificação os três bancos de dados são utilizados simultaneamente. Como o ambiente é implementado em PROLOG, um só banco de dados físico é utilizado. Os modelos internos para representar as informações referentes a cada um dos bancos de dados é por isso diferente, de modo a permitir a individualização das informações referentes a cada um deles [EDE 93a]. A seguir apresentamos os modelos internos correspondentes a cada um dos bancos de dados utilizados.

7.1 Modelo Interno do Banco de Dados Especificação

A seguir apresentaremos os predicados utilizados internamente para armazenar a especificação; utilizaremos como convenção de representação as palavras reservadas iniciando por letras minúsculas e as variáveis, cujos valores são fornecidos pelo projetista da aplicação, iniciando por letras maiúsculas.

A definição do domínio de aplicação no banco de dados especificação é feito através do seguinte predicado:

aplication (Nome_Aplicação).

Uma especificação pode ser construída em diversos níveis de detalhamento, este processo no modelo TF-ORM é *top-down*, detalhando a definição das classes em refinamentos sucessivos. Todos os diversos níveis de detalhamento de especificação de uma classe ficam armazenados no banco de dados especificação durante o processo, podendo ser consultados. Para isto foi criada uma indicação do nível que está sendo considerado. É adotado o maior número de ordem para o nível mais detalhado da especificação, o qual constitui a especificação final. O nível inicial recebe o número 1 pelo sistema. A definição do nível em que uma classe se encontra é associada à definição de cada uma das informações relativas a uma classe. A reutilização de classes da biblioteca em uma especificação se dá somente no nível inicial de definição de uma classe.

Conforme o tipo de classe definido (classe de agente, de recurso ou de processo), o seu nome é definido através de um dos seguintes predicados:

agclass(Nome_Classe).

resclass(Nome_Classe).

proclass(Nome_Classe).

Os diversos papéis definidos para uma classe são associados ao nome da classe através de predicados da seguinte forma:

role(Classe, Nível, Nome_Papel).

O nome de cada papel deve ser único para uma mesma classe de objetos. O papel básico recebe internamente o nome de *baserole*.

Os tipos de dados permitidos para domínios de propriedades são pré-definidos no sistema: **integer, real, boolean, string, text, instant, date, year, month, day, time, hour, minute, week, semester, weekday, interval, span.**

Os valores válidos para tipos de limites de intervalos são **integer, instant, date, year, month, day, time, hour** e **minute**. Valores para o tipo de intervalo são **closed, upopen downopen, open, upfloat** e **downfloat.**

Unidades de medida temporal que podem ser usadas para durações (*span*) podem ser **year**, **month**, **day**, **hour**, **minute**, **week** e **semester**.

As propriedades estáticas e dinâmicas são representadas internamente por:

statprop(Classe, Nível, Papel, Nome_Propriedade, Tipo_Domínio).

dynprop(Classe, Nível, Papel, Nome_Propriedade. Tipo_Domínio).

Quando o domínio de uma propriedade é uma classe, o nome desta é associado à propriedade através do seguinte predicado:

classdom(Classe, Nível, Papel, Nome_Propriedade, NomeClasse).

Para domínios de propriedades que são durações ou intervalos, as informações sobre suas características são representados respectivamente por:

interval(Classe, Nível, Papel, NomeProp, Tipo_intervalo, Tipo_limites).

span(Classe, Nível, Papel, NomeProp, Tipo_duração).

Caso o domínio seja uma lista de *strings* (enumeração), seus elementos são representados através do predicado:

enum(Classe, Nível, Papel, NomeProp, Elemento).

Cada um dos elementos é representado separadamente pelo predicado.

Se o domínio da propriedade for um domínio complexo, o tipo do domínio simples, associado ao domínio complexo é indicado pelos seguintes predicados:

set(Classe, Nível, Papel, NomeProp, Nome_domínio_simples).

list(Classe, Nível, Papel, NomeProp, Nome_domínio_simples).

É importante ressaltar que a associação dos tempos de transação e de validade às propriedades dinâmicas se dá somente no momento de execução de uma aplicação, não durante a sua especificação.

Os estados são armazenados da seguinte forma:

state(Classe, Nível, Papel, Nome_Estado)

As mensagens enviadas e recebidas e os argumentos são representadas respectivamente pelos predicados:

mess(Classe, Nível, Papel, Nome_Mensagem, **to**, Nome_Classe, Nome_Papel, Nro_Argumentos).

mess(Classe, Nível, Papel, Nome_Mensagem, **from**, Nome_Classe, Nome_Papel, Nro_Argumentos).

argum(Classe, Nível, Papel, Mensagem, Nro_Ordem, Nome, Tipo).

Tipos de argumentos que necessitam de informações adicionais, além do seu nome, são representadas de forma similar aos domínios de propriedades:

classdom2(Classe, Nível, Papel, Nome_Mensagem, Nro_argum, NomeClasse).

interval2(Classe, Nível, Papel, Nome_Mensagem, Nro_argum, Tipo_intervalo, Tipo_limites).

span2(Classe, Nível, Papel, Nome_Mensagem, Nro_argum, Tipo_duração).

enum2(Classe, Nível, Papel, Nome_Mensagem, Nro_argum, Elemento).

list2(Classe, Nível, Papel, Nome_Mensagem, Nro_argum, Nome_domínio_simples).

set2(Classe, Nível, Papel, Nome_Mensagem, Nro_argum,Nome_domínio_simples).

Uma regra de integridade é representada internamente através do seguinte predicado:

statintegr (Classe, Nível, Papel, Condicao1, Condicao2).

Atualmente nenhuma análise das condições fornecidas está sendo realizada, ficando sua sintaxe a cargo do projetista da aplicação. Os argumentos das mensagens e condição das regras de transição de estados, quando existentes, são representadas através dos seguintes predicados:

statetrans(Classe, Nível, Papel, Nro_Regra, Estado1, Estado2, Mens1, Mens2).

argumento(Classe, Nível, Papel, Nro_Regra,Direção,Nro_Ordem,Nome).

dynintegr(Classe, Nível, Papel, Nro_Regra, Condição).

A numeração das regras de transição de estados de um papel é feita pelo sistema. A associação entre uma regra e sua condição é definida pelo número correspondente à regra. Também para esta condição não está sendo atualmente feita nenhuma análise sintática, ficando isto sob responsabiblidade do projetista.

As abstrações de especialização e agregação são representadas através dos seguintes predicados:

subclass(Classe, Nível, Superclasse).

agregation(Nova_Classe, Classe_Componente, Nível).

Em caso de herança múltipla, para cada superclasse é definido um destes predicados. Também para cada classe componente de uma agregação é definido um predicado associando-a à nova classe gerada.

7.2 Modelo Interno da Biblioteca de Manutenção

Os nomes dos predicados utilizados na biblioteca de manutenção são diferenciados daqueles utilizados na especificação através de um prefixo, uma vez que durante a execução da especificação a base de informações é única. Os predicados referentes à biblioteca de manutenção são os seguintes:

m_classnamesyn(Nome_Classe, Sinônimo).

m_rolenamesyn(Classe, Nome_Papel, Sinônimo).

m_classdescr(Classe, Descricão).

m_roledescr(Classe, Papel, Descrição).

m_conddescr(Classe, Papel, Regra, Descrição).

m_agclass(Aplicação, Nome_Classe).

m_resclass(Aplicação, Nome_Classe).

m_proclass(Aplicação, Nome_Classe).

m_role(Classe, Nome_Papel).

m_statprop(Classe, Papel, Nome_Propriedade, Tipo_Domínio).

m_dynprop(Classe, Papel, Nome_Propriedade, Tipo_Domínio).

m_classdom(Classe, Papel, Nome_Propriedade, Classe).

m_interval(Classe, Papel, Nome_Propriedade, Tipo_intervalo, Tipo_limites).

m_span(Classe, Papel, Nome_Propriedade, Tipo_duração).

m_enum(Classe, Papel, Nome_Propriedade, Elemento).

m_list(Classe, Papel, Nome_Propriedade, Nome_domínio_simples).

m_set(Classe, Papel, Nome_Propriedade, Nome_domínio_simples).

m_state(Classe, Papel, Nome_Estado).

m_mess(Classe, Papel, Mensagem, **to**, Nome_Classe, Nome_Papel, Nro_Parâmetros).

m_mess(Classe, Papel, Mensagem, **from**, Nome_Classe, Nome_Papel, Nro_Argumentos).

m_argum(Classe, Papel, Mensagem, Nro_Ordem, Nome, Tipo).

m_interval2(Classe, Papel, Nome_Mensagem, Nro_argum, Tipo_intervalo, Tipo_limites).

m_span2(Classe, Papel, Nome_Mensagem, Nro_argum, Tipo_duração).

m_enum2(Classe, Papel, Nome_Mensagem, Nro_argum, Elemento).

m_list2(Classe, Papel, Nome_Mensagem, Nro_argum, Nome_domínio_simples).

m_set2(Classe, Papel, Nome_Mensagem, Nro_argum,Nome_domínio_simples).

m_statetrans(Classe, Papel, Nro_Regra, Estado1, Estado2, Mens1, Mens2).

m_argumento(Classe, Papel, Nro_Regra,Direção,Nro_Ordem,Nome).

m_statintegr (Classe, Papel, Condicao1, Condicao2).

m_dynintegr(Classe, Papel, Nro_Regra, Condicao).

m_subclass(Classe, Superclasse).

m_agregation(Nova_Classe, Classe_Componente).

Informações a respeito da forma de reutilização de classes são também armazenadas na biblioteca de manutenção pela ferramenta de apoio à especificação. É feita a estatística de quantas vezes a classe foi reutilizada sem nenhuma modificação e de quantas vezes foi reutilizada e posteriormente modificada. Estas informações são armazenadas através dos seguintes predicados:

m_equal_reuse(Classe, NrReusos).

m_modif_reuse(Classe, NrReusos).

Quando uma classe da biblioteca de classes for modificada para sua reutilização em uma especificação, as seguintes informações são armazenadas na biblioteca de manutenção: (i) uma cópia completa da classe final (nível mais detalhado) constante da reutilização; (ii) uma indicação de qual a classe de origem, pois o nome da classe pode ter sido modificado durante sua adaptação à especificação; e (iii) sugestões do projetista de aplicações para o engenheiro de classes. As informações relativas aos itens (ii) e (iii) são armazenadas através dos predicados:

m_reused_class(Classe, Classe_Biblioteca).

m_hint(Classe_Biblioteca, Sugestão).

O software gerenciador da especificação impede a existência, na biblioteca de manutenção, de duas classes com nomes idênticos, o que geraria problemas para a identificação de seus componentes. Caso encontre uma classe com o nome daquela que está sendo definida, pede ao projetista de aplicações um sinônimo para evitar a duplicação.

A ferramenta de apoio ao projetista de aplicações cria um arquivo separado contendo a biblioteca de manutenção de cada especificação realizada. O nome deste arquivo é o mesmo nome de arquivo fornecido para o salvamento do banco de dados especificação, porém com sua extensão alterada para `.MAN`.

7.3 Modelo Interno da Biblioteca de Classes

As informações armazenadas nesta biblioteca apresentam forma semelhante àquelas dos dois bancos de dados anteriormente apresentados. Um prefixo específico faz a diferenciação entre os predicados que têm a mesma forma.

A representação interna das informações da biblioteca de classes é a seguinte, sendo o significado de cada um análogo à representação interna das demais bibliotecas:

b_aplication(Nome_Aplicação).

b_classnamesyn(Nome_Classe, Sinônimo).

b_rolenamesyn(Classe, Nome_Papel, Sinônimo).

b_classdescr(Classe, Descricão).

b_roledescr(Classe, Papel, Descrição).

b_agclass(Aplicação, Nome_Classe).

b_resclass(Aplicação, Nome_Classe).

b_proclass(Aplicação, Nome_Classe).

b_role(Classe, Nome_Papel).

b_statprop(Classe, Papel, Nome_Propriedade, Tipo_Domínio).

b_dynprop(Classe, Papel, Nome_Propriedade, Tipo_Domínio).

b_classdom(Classe, Papel, Propriedade, Classe).

b_interval(Classe, Papel, Nome_Propriedade, Tipo_intervalo, Tipo_limites).

b_span(Classe, Papel, Nome_Propriedade, Tipo_duração).

b_enum(Classe, Papel, Nome_Propriedade, Elemento).

b_list(Classe, Papel, Nome_Propriedade, Nome_domínio_simples).

b_set(Classe, Papel, Nome_Propriedade, Nome_domínio_simples).

b_state(Classe, Papel, Nome_Estado).

b_mess(Classe, Papel, Nome_Mensagem, **to**, Nome_Classe, Nome_Papel, Nro_Argumentos).

b_mess(Classe, Papel, Nome_Mensagem, **from**, Nome_Classe, Nome_Papel, Nro_Argumentos).

b_argum(Classe, Papel, Mensagem, Nro_Ordem, Nome, Tipo).

b_interval2(Classe, Papel, Nome_Mensagem, Nro_argum, Tipo_intervalo, Tipo_limites).

b_span2(Classe, Papel, Nome_Mensagem, Nro_argum, Tipo_duração).

b_enum2(Classe, Papel, Nome_Mensagem, Nro_argum, Elemento).

b_list2(Classe, Papel, Nome_Mensagem, Nro_argum, Nome_domínio_simples).

b_set2(Classe, Papel, Nome_Mensagem, Nro_argum,Nome_domínio_simples).

b_statetrans(Classe, Papel, Nro_Regra, Estado1, Estado2, Mens1, Mens2).

b_argumento(Classe, Papel, Nro_Regra, Direção, Nro_Ordem, Nome).

b_statintegr (Classe, Papel, Condição1, Condição2).

b_dynintegr(Classe, Papel, Nro_Regra, Condição).

b_subclass(Classe, Superclasse).

b_agregation(Classe, Classe_Componente).

São ainda definidos alguns predicados para fins de estatísticas de reutilização de classes:

b_equal_reuse(Classe, Nro_Reutilizações).

b_modif_reuse(Classe, Nro_Reutilizações).

8. CONCLUSÃO

Por ser a especificação de aplicações uma tarefa cansativa, complexa e sujeita a erros, uma ferramenta que auxiliar este processo é quase que indispensável. Erros de diversos tipos, como erros de sintaxe ou a especificação incompleta de informações, podem ser facilmente introduzidos à medida que o tamanho da aplicação cresce e a detecção manual destes é em muitos casos um trabalho penoso e extenso.

A ferramenta apresentada neste trabalho auxilia o usuário encarregado da construção de aplicações guiando-o através de uma seqüência lógica, reduzindo a quantidade de informações manipulada pelo usuário e tentando detectar o maior número possível de erros ou mesmo impedir a introdução de erros na especificação.

O conceito de reutilização de especificações anteriormente contruídas é explorado pela ferramenta através da Biblioteca de Classes, de onde classes podem ser incorporadas à especificação tanto com modificações quanto sem. À medida que novas especificações forem sendo feitas, as classes criadas nestas podem ser incorporadas na Biblioteca de Classes por meio de outra ferramenta, ampliando o número de classes para reuso.

Apesar da ferramenta ser de fácil uso em grande parte do processo de especifcação de aplicações, sua interface pode ser de difícil uso em certas situações. Em alguns casos, como na modificação de informações já armazenadas, um grande número de ações deve ser tomado para a realização de uma ação simples. A correção desta falha não é muito simples e sua implementação pode ser bastante onerosa, exigindo tempos de planejamento e implementação elevados.

Formas alternativas de reutilização de informações e especificação são objetos de estudos mais aprofundados.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[EDE 93a] EDELWEISS, Nina; OLIVEIRA, José Palazzo M. de. Representação Interna de um Banco de Dados para Suportar o Modelo Temporal F-ORM. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 8., 12-14 maio, 1993, Campina Grande. *Anais*. Campina Grande: SBC, 1993. p.297-311.

[EDE 93b] EDELWEISS, N.; OLIVEIRA, J. P. M.; PERNICI, B. An Object-oriented temporal model. *Proceedings* of the 5th International Conference on Advanced Information Systems Engineering - CAISE'93, Paris, France, June 8-11, 1993. pp.397-415. (Lecture Notes in Computer Science n. 685).

[EDE 94] EDELWEISS, Nina. *Sistemas de Informação de Escritórios: um Modelo para Especificações Formais*. Porto Alegre: CPGCC/UFRGS, junho 1994. 187p. (Tese de doutorado).

[STE 86] STERLING, L.; SHAPIRO, E. *The Art of Prolog - Advanced Programming Techniques*. Cambridge: The MIT Press, 1986.