Especificação Formal de Instâncias Excepcionais

Ana Paula Ambrósio
Décio Fonseca

Departamento de Informática
Universidade Federal de Pernambuco
CP 7851, 50739 Recife - PE - Brasil

Resumo

Ao modelar-se um sistema que representa o mundo real geralmente não é possível prever todos os casos que podem surgir em determinada situação. Querer incluir casos excepcionais na modelagem do sistema pode levar à generalizações desnecessárias visto que eles podem eventualmente nem ocorrer. Além disto, esta prática fere um dos princípios básicos da engenharia de software — a abstração. Neste trabalho apresentamos uma forma de tratarmos ocorrências atípicas que podem surgir quando se utiliza um sistema de banco de dados e faremos a especificação formal de um tratamento de exceções para instâncias atípicas dentro do Modelo Estático/Dinâmico.

1 Introdução

Os Sistemas de Informações (SIs) têm como objetivo modelar parte do mundo real, armazenando as informações relevantes dessa porção da realidade. Ao fazer isto, eles facilitam o acesso a essas informações e permitem a investigação de situações hipotéticas, sem a necessidade de consultas ou modificações no mundo real.

Realidades como registros pessoais, crédito, reservas, contabilidade, etc., têm sido modeladas frequentemente. A maioria delas envolvem a manipulação de grandes quantidades de informações, sendo que a maior vantagem destes SIs é o armazenamento e recuperação eficiente destas informações.

Os SIs atuais não têm flexibilidade, e portanto não são capazes de lidar com informações que não se encaixam perfeitamente na estrutura do sistema. Sua flexibilidade só existe a custa de programas muito longos e complexos, onde cada caso tem que estar embutido no código, desviando o programador dos casos normais. Isto fere um princípio básico da Engenharia de Software, que prega a abstração nos primeiros estágios do desenvolvimento de software, para que detalhes sejam incluídos gradualmente através de refinamentos sucessivos.

Este princípio é importante tanto no momento do desenvolvimento, como na manutenção, pois torna os programas mais inteligíveis ao apresentar primeiramente os casos normais, para depois tratar os casos especiais. Estes princípios de programação também valem para o desenvolvimento de um banco de dados. Além do mais, incluir todos os casos atípicos no código dos bancos de dados pode não ser possível, pelo simples fato de que não se pode prever todos eles. Assim, quando estes casos atípicos surgem, ou não são tratados, ou uma modificação no código torna-se necessária.

Com o intuito de dar maior flexibilidade aos SIs, deseja-se dotá-los de mecanismos capazes de tratarem exceções de forma genérica, isto é, os casos excepcionais poderão ser tratados durante a execução dos programas através de mecanismos para tratamento de exceções semelhantes aos existentes em linguagens de programação. Nestes casos, uma estrutura de controle é inserida nas linguagens de programação para permitir que a continuação normal de uma operação possa ser substituída por uma continuação excepcional toda vez que uma exceção seja detectada.

Pelo fato do mundo não ser homogêneo, toda regra é passível de exceção, logo não existe uma forma de antecipar todas as exceções e generalizar regras para evitar contradições. Isto torna impraticável o uso de restrições muito severas, além de tornar inaceitável que informações contraditórias sejam excluídas dos bancos de dados, tornando-os incompletos, incorretos e desatualizados.

Para tratar este dilema, surgem os mecanismos de tratamento de exceções, cuja principal idéia é que estas restrições sejam vistas como condições de *normalidade* —que podem eventualmente conflitar com o estado atual do mundo [Bor85]. Para isto, é preciso permitir que violações às restrições persistam. Haverá então a coexistência de restrições e dados que conflitem com elas. Neste caso verifica-se que:

1. Pode ser necessária a suspensão de restrições dinâmicas de integridade para a atualização do banco de dados, assim como de restrições no relacionamento entre valores dos dados.
2. Pode tornar-se necessário atribuir valores fora do intervalo definido para atributos com valores escalares (inteiros ou strings). Por exemplo, se tivermos um atributo **valor** definido como um inteiro entre zero e cem e surgir uma exceção onde é preciso atribuir o valor mil para este atributo.
3. Pode tornar-se necessário atribuir valores de tipo completamente distinto do definido para um dado atributo. Isto é, um atributo foi definido como sendo um inteiro porém necessita receber um valor do tipo caracter.
4. Alguns objetos podem requerer que certos atributos definidos com uma restrição de unicidade contenham valores múltiplos. Podem requerer também atributos adicionais, não previstos no momento da modelagem.
5. Pode tornar-se necessário a inclusão de uma restrição de integridade que torne algumas instâncias normais em excepcionais por um determinado tempo, sendo que em um estado futuro elas podem voltar a normalidade. Um exemplo desta exceção seria o caso onde definiu-se uma restrição que determina que nenhum salário deve ser inferior a um valor **x**, mais tarde esta restrição é alterada para que o salário não possa ser inferior a **2x**. Neste momento várias instâncias passarão a ser excepcionais, já que não satisfazem a restrição. Esta situação porém deve ser temporária. A medida que os salários forem sendo reajustados eles deixarão de ser excepcionais.

A seguir apresentaremos uma noção sobre exceções em linguagens de modelagem conceitual, principalmente no que se refere a instâncias excepcionais. Na seção 3 apresentamos o Modelo E/D e a extensão proposta para permitir o tratamento de exceções. Finalmente especificaremos o mecanismo para tratamento de exceções e as conclusões obtidas no trabalho.

2 Linguagens de Modelagem Conceitual

Desejamos tratar exceções a partir da fase de modelagem do SI. Para isto, torna-se necesário incluir mecanismos de tratamento de exceções nas linguagens de modelagem, a exemplo do que já é feito em algumas linguagens de programação [Iss89][Cea88].

Por ser uma forma bastante natural e próxima da visão humana do mundo, o uso de um modelo baseado no paradigma de orientação a objetos facilita o desenvolvimento e acesso a SIs. Isto se deve principalmente ao fato de que modelar porções do mundo real usando conceitos de entidade, atributos e relacionamentos, onde todos são objetos, está bem próxima da visão conceitual que o homem tem da realidade.

Ao modelar-se um SI, pressupõe-se que exista um conjunto de definições consistentes para os termos visando descrever o domínio da aplicação correspondente. Porém como já vimos, no mundo real raramente existem definições rígidas, sendo elas geralmente prototípicas.

Logo podem surgir ocasiões onde se torna necessário tratar ocorrências atípicas. Estas podem se restringir a casos isolados ou podem formar um grupo de instâncias, justificando a definição de uma classe para eles dentro da modelagem. Assim existem duas classes de contradições que podem surgir em um SI:

- Instâncias atípicas - quando algum fato ocasional precisa se desviar das restrições definidas para o objeto.
- Subclasses atípicas - quando uma subclasse precisa contradizer partes da definição de sua superclasse.

Tratar estes tipos de exceções sem modificar toda a modelagem feita anteriormente tem inúmeras vantagens, sendo uma delas o fato de não ser preciso adaptar a informação para que ela satisfaça a especificação. Além disto, estas instâncias estão marcadas como exceções, permitindo que se programe de uma forma defensiva, isto é, pode-se isolar todos os casos que violam de alguma forma a especificação. No caso de instâncias atípicas, generalizar uma definição ou tornar mais solta uma restrição apenas por causa de uma ocorrência atípica pode gerar vários erros. Ao tratarmos estes casos separadamente, estes erros são evitados.

Neste artigo restringiremo-nos ao tratamento de instâncias atípicas. As subclasses atípicas serão tratadas em trabalhos futuros.

2.1 Instâncias Excepcionais

Para ilustrar melhor este tipo de exceção, tomemos o exemplo de um ALUNO de um curso de mestrado que possui os atributos de *nome, endereço, telefone, sexo*. Estes atributos têm sua estrutura especificada dentro do modelo. Pode surgir o caso isolado de um aluno ter que fazer parte de sua tese no exterior. A estrutura do endereço pode não mais se adaptar a definição de endereço feita na modelagem. Surge então a necessidade de criar-se uma instância excepcional para este caso.

Torna-se necessário o desenvolvimento de um mecanismo de tratamento de exceções que possa cuidar satisfatoriamente estes problemas. Propõe-se o uso de algumas características presentes em linguagens de programação. O mecanismo de tratamento de exceções permite que violações a restrições de integridade, incluindo restrições de tipo, que serão detectadas quando da inclusão ou alteração de dados, persistam.

Após a detecção, a transação pode ser desfeita ou confirmada, conforme decisão do usuário. Se for o caso de dados digitados de forma errada, a transação é desfeita. Se for o caso de uma exceção desejável, confirma-se a transação e as alterações são feitas mesmo que violem as restrições definidas no esquema. Com isso tornam-se necessários mecanismos que detectem e reajam a dados excepcionais que já se encontram no banco e que foram assim marcados quando houve a confirmação da transação.

A seguir veremos o Modelo Estático/Dinâmico (E/D), desenvolvido no Departamento de Informática da Universidade Federal de Pernambuco, como parte integrante do projeto APOIO [Fon89]. Apresentaremos sucintamente sua proposta e detalharemos aqueles aspectos relevantes à extensão proposta. Em seguida especificaremos um método para o tratamento de instâncias excepcionais.

3 O Modelo E/D Estendido

Seguindo o paradigma de orientado a objetos, o Modelo Estático/ Dinâmico [Ban89] propõe-se a modelar a visão conceitual do usuário "através do tratamento dos componentes estáticos e

dinâmicos de um SI". O **componente estático** trata dos aspectos relacionados aos dados propriamente ditos, enquanto o **componente dinâmico** cuida das mudanças de estados e evolução dos objetos.

O modelo gera um **Esquema Integrado**, composto dos objetos da aplicação, suas respectivas propriedades, os relacionamentos entre eles e uma representação gráfica.

A **representação estrutural** do modelo centra-se nos conceitos de Entidade, Atributo, Domínio e Relacionamento, que podem ser identificados através de uma Linguagem de Definição de Objetos (LDO) [Fon87].

Entidade

É o objeto básico do modelo e pretende modelar objetos do mundo real. Cada objeto do mundo real só pode ser identificado por uma única classe entidade.

Uma entidade só pode ter uma identificação, não podendo haver mais de uma classe entidade com a mesma identificação.

O modelo E/D não adota o conceito de chave para identificação de entidades por não acreditar que seja sempre possível determiná-las. A dependência funcional entre atributos de uma classe entidade identifica os atributos determinantes, que possuem implicitamente as restrições de **unicidade** e **não-nulidade**.

Deve ser definido um atributo ou um conjunto de atributos para uma classe entidade. Todas as verdades sobre uma classe entidade devem ser expressas sob a forma de restrições. A dependência funcional deve ser feita sobre o conjunto completo de atributos definidos na classe, incluindo os atributos herdados em associações hierarquicas (generalização/especialização).

Todas as ações (fatos) que venham a atuar sobre uma classe entidade modificando seu estado devem ser especificadas, assim como todas as condições que devam ser respeitadas para que as ações aconteçam devem ser definidas através de regras de comportamento.

Atributo

Atributos representam as propriedades das classes de entidades, assim como dos relacionamentos. Eles podem ser multivalorados ou não. Além disto podem ser compostos, formados por um ou mais atributos que podem ser simples ou multivalorados.

A todo objeto atributo deve ser atribuído um nome, não podendo haver mais de um atributo com a mesma identificação.

Todas as asserções ou qualificações de um atributo devem ser expressas sob a forma de restrições.

Todo atributo que determina um ou mais atributos de uma entidade ou relacionamento é um atributo determinante. Um atributo determinante possui implicitamente as restrições Não-Nulidade e Unicidade.

Domínio

Representa um contexto ou uma estrutura sob o qual um atributo está definido. Pode ser simples (inteiro, caracter, real, booleano, cadeia) ou complexo (voz, imagem, lista, registro, etc.).

A presença de um objeto domínio no modelo visa manter uma coerrência estrutural. No entanto este objeto em particular não foi tratado em profundidade pelo modelo, sendo aceito apenas o domínio simples. Este aspecto está sendo desenvolvido dentro do projeto do ambiente.

Para tratarmos exceções tornou-se necessária a inclusão do valor indefinido, que indica um valor nulo. A inclusão deste valor dentro de todos os domínio torna-se necessária porque um atributo só pode receber valores dentro de seu domínio. O valor indefinido será utilizado nos casos onde uma subclasse ou instância não tenha necessidade de usar o atributo.

PDF Compression, OCR, web optimization

Relacionamento

Representa as associações entre instâncias de classes de objetos entidades, podendo inclusive ser entre instâncias da mesma classe (auto-relacionamento). Sua existência depende da existência dos objetos relacionados.

O relacionamento é um objeto constituído de atributos, ações, regras de comportamento e restrições de integridade, que asseguram um conjunto de estados válidos. Por ser um objeto e não uma classe, não herda propriedades nem de outros relacionamentos.

Todo relacionamento pode ser classificado em total ou parcial. Além disto, todo relacionamento possui uma cardinalidade.

Todas as asserções ou qualificações de uma associação devem ser expressas através de restrições. A dependência funcional deve ser feita sobre o conjunto completo de atributos definidos para a associação, incluindo os atributos especificados em cada classe entidade envolvida.

Todas as ações (fatos) que venham atuar sobre um relacionamento, modificando seu estado, devem ser especificadas. Todas as condições que devam ser respeitadas para que ações aconteçam devem ser definidas através de regras de comportamento.

Restrições de Integridade

As restrições de integridade estão classificadas em pré-definidas, verificação e cardinalidade, subdivididas da seguinte forma:

Pré-definidas

São as restrições estruturais clássicas: dependência funcional, unicidade, não-nulidade e dependência de inclusão excludente.

Dependência funcional permite expressar a dependência entre atributos de uma classe entidade ou relacionamento. Através dela pode-se identificar os atributos determinantes, aos quais estão associadas as restrições de unicidade e não-nulidade. Um ou mais atributos podem determinar um mesmo conjunto de atributos.

Unicidade é definida sobre atributos de uma entidade ou de um relacionamento. Determina que cada instância do atributo especificado só pode ter um valor. Deve-se informar sobre qual entidade ou relacionamento está sendo definida a restrição unicidade, além dos atributos aos quais esta restrição é imposta.

A restrição de Não-Nulidade é semelhante a de unicidade. Determina que o atributo não pode ter valores nulos.

Dependência de Classes Exclusivas permite definir, numa hierarquia de classes, subclasses mutuamente excludentes. Impõe que uma instância de uma subclasse exclusiva não participe do conjunto de instâncias de outras subclasses. É uma restrição sobre classes associadas por um processo de hierarquia (especialização/generalização). A super classe associada às subclasses exclusivas deve ser informada, assim como as subclasses mutuamente excludentes.

Verificação

Especifica restrições sobre instâncias dos objetos entidade e relacionamento. Explicita também em que tipos de atualização (Inserção, Modificação, Remoção, Todas, Combinadas) estas restrições devem ser verificadas. Deve-se definir a expressão com os critérios a serem verificados.

Cardinalidade

É definida sobre associações entre relacionamentos. Indica como instâncias dos objetos envolvidos estão relacionadas. Para relacionamentos binários temos as seguintes classes: 1:1, 1:N, N:N, C:N, C1:C2, onde C, C1 e C2 são constantes.

Esta classificação para relacionamentos binários pode ser estendida para relacionamentos ternários.

Ação

Permite descrever ações que modificam estados básicos dos objetos entidade e relacionamento através de um conjunto de operações básicas. São elas inserção, modificação, remoção, recuperação e atribuição.

A definição de ação é recursiva, isto é, uma ação pode ser composta de uma ou várias ações.

Condição

Exprime os diversos tipos de condições e/ou critérios que devem ser satisfeitos em uma regra de comportamento. Está classificada em: condição operatória (se inserção, modificação, remoção, todas ou qualquer cominação delas) ou condição temporal (referente ao tempo: se DataSys, DiaSys, MesSys, AnoSys). Além disso, condições podem ser expressas através de critérios.

Regra de Comportamento

Representa as propriedades comportamentais dos objetos entidade e relacionamento. Permite exprimir uma ação condicional através de uma regra formada de SE condição/ ENTÃO ação, sendo condição e ação instâncias dos objetos com estes nomes já especificados anteriormente.

Transação

Permite a definição de um procedimento atômico, isto é, permite que se estabeleça uma sequência de ações e regras de comportamento que devem ser todas executadas e na ordem definida.

Meta-Regra

Representa o mecanismo de controle na aplicação das regras de comportamento, visando garantir a coerência do conjunto especificado em uma transação. O controle é feito a nível conceitual, sendo definido pelo usuário e incorporado ao Esquema Integrado da Aplicação. A validação das transações é feita baseada nestes controles.

O objeto permite definir prioridades de execução das regras de comportamento e controlar regras mutuamente excludentes, além de controlar a dependência de uma regra em relação a outra. Estes conceitos podem ser combinados dentro de uma mesma meta-regra.

4 Especificação do Mecanismo para Tratamento de Exceções

Usaremos o modelo E/D para modelarmos instâncias excepcionais. O conjunto de objetos especificados formam um módulo de controle para o tratamento de exceções. O módulo é manipulado exclusivamente pelo sistema.

A princípio todo objeto do modelo poderá ser marcado como excepcional. Isto quer dizer que uma instância dentro de sua classe será criada com um valor exceção. A razão para isto é que um objeto só pode ter atribuído aos elementos de sua estrutura um valor válido.

Com isto queremos dizer que se por exemplo, uma entidade ALUNO tem um atributo nome, este atributo só pode receber um valor válido dentro da estrutura especificada para ele. Assim, criamos uma instância dentro da classe nome que é uma exceção. Analogamente pode-se fazer o mesmo para todos os objetos.

Quando o usuário deseja incluir uma instância no sistema, ele entra com os dados e o sistema verifica se a identificação dada é única. Se este requisito for satisfeito, verificam-se os demais componentes da estrutura. Ocorrendo o caso de um dos componentes estruturais não estar recebendo um valor dentro do especificado, pergunta-se ao usuário se houve algum erro de digitação ou se é uma exceção. Sendo realmente uma exceção, ativa-se o mecanismo de tratamento de exceções.

Para tratar instâncias excepcionais alguns objetos foram criados. O objeto VIOLAÇÃO permite que se mantenha um registro das ocorrências excepcionais de um sistema. Sua estrutura consiste de três componentes: a entidade ou relacionamento onde ocorreu a exceção (TIPO), o objeto da estrutura que possui o valor excepcional (OBJETO) e uma referência à instância da entidade ou relacionamento onde ocorre a exceção (INSTÂNCIA).

Existe também o objeto EXCEÇÃO que faz a ligação entre a instância excepcional (INSTÂNCIA) e o valor que vai substituí-la (OBJETO). Vale ressaltar que este procedimento é dinâmico e permite a inclusão de novos objetos em tempo de execução. Assim se o objeto que vai substituir a exceção não tem sua estrutura especificada esta pode ser incluída na hora. Este procedimento pressupõe que o usuário tem autorização para fazer isto e um registro do responsável pela exceção será mantido.

Quando uma exceção vai ser incluída, cria-se uma instância excepcional dentro do objeto e esta instância referencía o objeto que vai substituí-la através da inclusão de uma instância em EXCEÇÃO. O registro desta exceção é mantido na estrutura VIOLAÇÃO.

A seguir especificaremos estas estruturas mais detalhadamente e usaremos as estruturas do Modelo E/D para meta-modelarmos o método de criação de uma instância.

4.1 Especificação das Estruturas

Para tratarmos instâncias excepcionais iremos especificar um conjunto de objetos. Eles serão definidos segundo a estrutura do modelo E/D. Incluimos dois objetos no modelo E/D. As violações e as exceções que serão especificados a seguir.

Violações

Este objeto serve para indicar que houve a violação de alguma restrição. Por ser uma classe ele pode ter diversas subclasses detalhando o tipo de violação ocorrida. Desta forma pode-se agrupar todas as violações de um certo tipo, como por exemplo VIOLAÇÃO-DE-DOMÍNIO.

No caso de uma exceção sobre um atributo, o objeto VIOLAÇÃO identifica o atributo excepcional, a instância que possui este atributo excepcional e a classe ou relacionamento à qual esta intância pertence.

Analogamente podemos tratar violações sobre qualquer objeto definido pelo modelo.

```
Objeto      :  VIOLAÇÃO
Descrição   :  Identifica as violações ocorridas no sistema
Tipo        :  <Identificação-Tipo>
Objeto      :  <Identificação-Objeto>
Instância   :  <Identificação-Instância>
Fim-Objeto
```

A identificação do tipo deve ser a identificação de uma entidade ou relacionamento especificado no modelagem. O Objeto deve ser um elemento da estrutura (atributo,restrição, etc.) do tipo especificado no elemento anterior e a Instância deve ser uma instância dentro do tipo.

Pode ser desejável em alguns casos que uma entidade ou relacionamento necessite de atributos adicionais, distintos daqueles especificados. Para tratar estes casos marcamos a entidade ou o relacionamento como excepcionais e incluimos uma instância dentro de VIOLAÇÃO com a estrutura Objeto recebendo o valor ATRIBUTO-ADICIONAL. Esta será uma classe definida, que tem como objetivo conter todos os casos onde uma entidade ou relacionamento incluiram algum atributo na sua estrutura, além dos especificados na modelagem.

```
Objeto      :  ATRIBUTO-ADICIONAL
Descrição   :  Identifica as violações à inclusão de um novo atributo na estrutura do objeto
Fim-Objeto
```

Exceções

Este objeto tem como objetivo identificar a instância do objeto marcado com excepcional e fazer uma referência para o objeto que vai substituí-lo.

```
Objeto      :  EXCEÇÃO
Descrição   :  Serve para indicar qual objeto será usado no lugar do excepcional.
Instância   :  <Identificação-Instância>
Objeto      :  <Identificação-Objeto>
Fim-Objeto
```

A Instância deve ser uma instância excepcional marcada dentro de uma classe definida. o Objeto deve ser um já especificado anteriormente ou deve-se especificá-lo na hora da inclusão da exceção.

Vale ressaltar que um atributo marcado como excepcional está se referindo exclusivamente a uma instância do atributo.

Assim, se tivermos um atributo ENDEREÇO, que pertence ao conjunto de atributos da classe PESSOA, podemos especificá-los da seguinte forma.

```
Objeto-Entidade   :  PESSOA
Descrição         :  Dados pessoais referentes a pessoas físicas
Atributos         :  Nome, Endereço, Telefone, Sexo, Estado-Civil, .....
Fim-Objeto-Entidade

Objeto-Atributo   :  ENDEREÇO
Descrição         :  Dados sobre endereço. Inclui rua, número, cep, cidade,estado, ...
Tipo              :  Composto-De (Rua, No., Bairro, Cep, Cidade, Estado)
Multivalorado     :  Não
Fim-Objeto-Atributo
```

Vamos supor que exista uma pessoa chamada JANE que mora em Brasília, cujo padrão de endereço não se encaixa no padrão especificado. Teremos então na instância de PESSOA referente a JANE, uma referência à instância de ENDEREÇO que deveria conter seu endereço. Como este é excepcional, está instância estará marcada como uma exceção.

Ao marcarmos um atributo como excepcional, diversas medidas devem ser tomadas para tratarmos esta instância excepcional. Primeiramente deve ser criada uma instância dentro da classe de VIOLAÇÕES, identificando o atributo excepcional, a classe a qual ele pertence e a instância na qual esta exceção ocorre.

Segundo, temos que especificar qual atributo tomará o lugar daquele marcado como excepcional. Isto é feito através da criação de uma instância de EXCEÇÕES, que indica a instância do atributo excepcional, e uma referência ao atributo que tomará o lugar dele neste caso específico.

Para exemplificar melhor está situação, tomemos o caso de um relacionamento É-ORIENTADOR, que especifica que um aluno tem que ter um orientador que é professor do departamento.

```
Objeto-Relacionamento   :  É-Orientador
Descrição               :  Todo aluno deve ter um orientador que é professor do departament
Participantes           :  Aluno(Matricula)
                        :  Professor(CPF,Departamento)
Restrições              :  Objeto-Cardinalidade(N-para-1)
Fim-Objeto-Relacionamento
```

Se houver uma situação onde o orientador não for um professor do departamento mas sim um funcionário, marcamos PROFESSOR como excepcional, criamos uma instância de VIOLAÇÕES com os valores de (É-Orientador, Professor, Instância-excepcional), e outra de EXCEÇÕES com os valores de (Instância-excepcional, Funcionário).

4.2 Especificação de Métodos sobre os Objetos

Na filosofia orientada a objetos cada objeto possui um conjunto de procedimentos associados (métodos) a ele, onde através destes é possível se ter acesso ao objeto. Para todo objeto, são definidos os seguintes métodos : Criar, Modificar, Remover, Consultar, Selecionar e Imprimir.

Para tratarmos exceções, estes métodos devem ser ligeiramente alterados para permitirem a detecção e tratamento de casos excepcionais.

No Modelo E/D não é feita uma especificação dos métodos sobre os objetos do modelo. Isto é, os métodos de inserção, remoção, modificação, etc. não têm uma especificação formal pelo modelo. A seguir utilizaremos as estruturas do modelo para fazermos uma meta-modelagem dos aspectos dinâmicos do modelo de tratamento de exceções.

Como exemplo modelaremos a criação de um objeto.

Primeiramente, no momento de criação do objeto o usuário deve entrar com os dados da instância deste objeto. Esta ação serve para todos os métodos.

```
Objeto-Ação   :  Dados-Objeto
Descrição     :  Entrada dos dados do objeto.
Sobre         :  Objetos
Ação          :  Todas
Fim-Objeto-Ação
```

A seguir deve-se verificar se a identificação do objeto é única para o tipo dado. Se não for única deve ser indicado um erro. Esta condição está definida como Identificação-Znica. O procedimento Trata-Erro não foi especificado por ser irrelevante ao assunto.

```
Objeto-Regra-Comportamento      :  Identificação
Descrição                       :  Se identificação não for única devolve erro.
Se                              :  Não (Identificação-Única)
Então                           :  Trata-Erro
Fim-Objeto-Regra-Comportamento
```

```
Objeto-Condição     :  Identificação-Única
Descrição           :  Identificação única para tipo dado
Se                  :  Verifica-Identificação
                       (se for verificado que a identificação é única
                       para o tipo dado, esta condição devolve o valor VERDADE)
Fim-Objeto-Condição
```

Se a identificação estiver correta passamos a verificar se a estrutura do objeto está sendo satisfeita. Se a estrutura estiver OK, o procedimento será normal. Caso haja algum problema, pergunta-se ao usuário se é uma exceção. Estes dois procedimentos estão agrupados no objeto-transação Estrutura. Os procedimentos Procedimento-Normal e Verifica-Se-Exceção são excludentes.

```
Objeto-Regra-Comportamento      :  Estrutura-OK
Descrição                       :  Se estrutura estiver OK segue processamento normal.
Se                              :  Verifica-Estrutura-OK
Então                           :  Procedimento-Normal
Fim-Objeto-Regra-Comportamento
```

```
Objeto-Regra-Comportamento      :  Estrutura-NOK
Descrição                       :  Se estrutura não estiver OK pergunta se é exceção.
Se                              :  Não (Verifica-Estrutura-OK)
Então                           :  Verifica-Se-Exceção
Fim-Objeto-Regra-Comportamento
```

```
Objeto-Transação        :  Estrutura
Descrição               :  Verifica se estruturas OK
Conjunto-Procedimentos  :  Estrutura-OK
                           Estrutura-NOK
Fim-Objeto-Transação
```

onde a condição Verifica-Estrutura-OK devolve VERDADE se todos os valores estiverem dentro do especificado.

```
Objeto-Condição     :  Verifica-Estrutura-OK
Descrição           :  Checa se os dados entrados satisfazem a estrutura especificada
Se                  :  Estruturas estão recebendo valores do tipo especificado?
Fim-Objeto-Condição
```

Se os valores digitados não forem os especificados, verifica-se se houve erro de digitação ou se é uma exceção desejada. Se for um erro de digitação ele deve ser tratado normalmente. Se for uma exceção, ativa-se o mecanismo de tratamento de exceções. Estes dois procedimentos estão

agrupados no objeto transação Verifica-Se-Exceção.

```
Objeto-Regra-Comportamento      : Erro-Digitação
Descrição                       : Não é exceção e sim erro de digitação
Se                              : Não (É-Exceção)
Então                           : Trata-Erro
Fim-Objeto-Regra-Comportamento

Objeto-Regra-Comportamento      : Exceção-Desejada
Descrição                       : Realmente se deseja uma exceção
Se                              : É-Exceção
Então                           : Tratamento-Exceção
Fim-Objeto-Regra-Comportamento

Objeto-Condição  : É-Exceção
Descrição        : Verifica se é uma exceção
Se               : Confirma-Exceção
                   (a ação faz as devidas perguntas ao usuário e caso
                   ele confirme a exceção o objeto condição devolve VERDADE)
Fim-Objeto-Condição

Objeto-Transação      : Verifica-Se-Exceção
Descrição             : Verifica se é uma exceção ou erro de digitação
Conjunto-Procedimentos : Erro-Digitação
                         Exceção-Desejada
Fim-Objeto-Transação
```

O tratamento de exceções consiste de duas partes. A primeira ocorre se houver um tratamento específico para a exceção. Caso contrário é feito um tratamento default. Não especificaremos neste trabalho como será feita a especificação de um mecanismo específico para uma dada exceção.

```
Objeto-Regra-Comportamento      : Tratamento-Exceção
Descrição                       : Corpo do mecanismo de tratamento de eceções
Se                              : Não (Tem-Trat-Específico)
Então                           : Executa.Tratamento-Default
Fim-Objeto-Regra-Comportamento
```

O Tratamento default para o caso de exceções é composto de tres partes: a criação da instância excepcional, a criação de uma instância de VIOLAÇÃO e uma de EXCEÇÃO.

```
Objeto-Transação      : Tratamento-Default
Descrição             : Caso não tenha sido especificado um tratamento segue-se este procedim
Conjunto-Procedimentos : Marca-Objeto-Excepcional
                         Cria-Objeto-Violação
                         Cria-Objeto-Exceção
Fim-Objeto-Transação
```

A primeira parte, que consiste da criação de uma instância excepcional, possui duas regras de comportamento. Se for a inclusão de um atributo adicional executar-se-á o procedimento Marca-Atributo. Se for uma exceção sobre um objeto já especificado na estrutura, executa-se o procedimento Marca-Objeto. A condição Atributo-Adicional verifica qual das duas opções se deseja.

```
Objeto-Regra-Comportamento  : Marca-Atributo
Descrição                   : Cria uma instância exceção do tipo desejado
Se                          : Atributo-Adicional
Ação                        : Insere.Atributo-Adicional(Exceção)
Fim-Objeto-Regra-Comportamento
```

```
Objeto-Regra-Comportamento  : Marca-Objeto
Descrição                   : Cria uma instância exceção do tipo desejado
Se                          : Não (Atributo-Adicional)
Ação                        : Insere.Objeto(Exceção)
Fim-Objeto-Regra-Comportamento
```

```
Objeto-Condição  : Atributo-Adicional
Descrição        : Verifica se é a inclusão de um novo atributo diferente do especificado
Se               : Deseja-se incluir um novo atributo?
Fim-Objeto-Condição
```

```
Objeto-Transação      : Marca-Objeto-Excepcional
Descrição             : Cria uma instância excepcional
Conjunto-Procedimentos : Marca-Atributo
                         Marca-Objeto
Fim-Objeto-Transação
```

A segunda parte consiste da criação de uma instância de VIOLAÇÃO. Seus atributos devem receber o nome da entidade ou relacionamento na qual está ocorrendo a exceção, o atributo ou objeto excepcional (no caso da inclusão de um novo atributo este deve receber o valor ATRIBUTO-ADICIONAL) e a instância excepcional.

```
Objeto-Ação  : Cria-Objeto-Violação
Descrição    : Cria a instância da violação ocorrida
Sobre        : Violação
Ação         : Inserir.Violação(Tipo-Dado,Objeto-Dado, Instância)
Fim-Objeto-Ação
```

A última consiste na criação de uma instância de EXCEÇÕES. O conjunto de procedimentos que definem esta parte estão no objeto ação Cria-Objeto-Exceção.

```
Objeto-Ação          : Cria-Objeto-Exceção
Descrição            : Cria uma instância de exceção
Sobre                : Exceção
Ação                 : Inserir.Exceção(Instância, Objeto)
Fim-Objeto-Transação
```

O objeto Transação especifica as ações que compõem o método de criação.

```
Objeto-Transação     : Cria
Descrição            : Método para a criação de uma instância do objeto
Conjunto-Procedimentos : Dados-Objeto
                         Identificação
                         Estrutura
Fim-Objeto-Transação
```

Finalmente criamos uma meta-regra que irá comandar a sequência em que estas ações devem ser executadas e define as ações excludentes.

```
Objeto-Meta-Regra    : Meta-Regra-Cria
Descrição            : Cria-Instancia de objeto
Conjunto-Controle
Regras-Exclusivas    : Procedimento-Normal
                       Verifica-Se-Exceção
Regras-Exclusivas    : Trata-Erro
                       Tratamento-Exceções
Regras-Exclusivas    : Erro-Digitação
                       Exceção-Desejada
Fim-Objeto-Meta-Regra
```

Os outros métodos (modificação, remoção, consulta, seleção, impressão) podem ser definidos analogamente.

5 Conclusão

A criação de um mecanismo de exceções para tatar as ocorrências atípicas que porventura possam surgir durante o ciclo de vida de um sistema de informações é muito importante. Ele evita que o usuário do modelo tenha que se preocupar em definir todos os casos excepcionais que podem ocorrer dentro de seu universo.

Muitas vêzes torna-se necessário uma generalização maior do que a desejável apenas para tratar um caso. Um exemplo típico disto é a classificação dada aos pássaros. Em princípio todos os pássaros voam, mas existem os pinguins que são pássaros mas não conseguem voar. Em uma modelagem sem tratamento de exceções seria necessário omitir o fato que os pássaros voam pois senão os pinguins não poderiam ser incluídos. Por outro lado, poderia criar-se duas subclasses dentro de pássaros para tratar o caso dos pinguins. Uma de pássaros que voam e uma dos que não voam. Neste caso estaríamos fazendo o inverso, uma especialização desnecessária e cara.

Verificou-se que o uso de mecanismos de tratamento de exceções em linguagens de programação torna-as mais eficientes. Isto também deve ser possível em sistemas de informações.

Além disto, o uso de um mecanismo de exceções torna a modelagem mais inteligível ao apresentar os casos normais separados das exceções, permitindo que tratamentos específicos sejam dados a ambos.

Referências

[Ban89] Mônica Bandeira. *Modelo Estático/Dinâmico (E/D) de Dados.* Master's thesis, Universidade Federal de Pernambuco, December 1989.

[Bor85] Alexander Borgida. Features of languages for the development of information systems at the conceptual level. *IEEE Software*, January 1985.

[Cea88] L. Cardelli and J. Donahue et al. *Modula-3 Report.* Technical Report, Digital Systems Research Center, August 1988.

[Fon87] Décio Fonseca. *Une Mecanisme d'Activation et Controle de Declencheurs Orienteé Objets.* PhD thesis, Univeridade PARIS VI, 1987.

[Fon89] Décio Fonseca. *Projeto APOIO: um Ambiente para Concepção de Banco de Dados.* Technical Report, Universidade Federal de Pernambuco, 1989.

[Iss89] Valerie Issarny. *Le Traitement d'Exceptions: Aspects Theoriques et Pratiques.* Technical Report, INRIA, November 1989.