CONSTRUÇÃO DE PADRÕES PARA A MODELAGEM DE ASPECTOS TEMPORAIS

Eduardo Kroth
Universidade de Santa Cruz do Sul
Departamento de Informática
Santa Cruz do Sul – RS - Brasil
kroth@dinf.unisc.br

Eugênio de Oliveira Simonetto
Pontifícia Universidade Católica do Rio Grande do Sul
FACI – Departamento de Informática
Uruguaiana – RS - Brasil
eugenio@pucrs.campus2.br

Carlos Poll
Universidade de Santa Cruz do Sul
Departamento de Informática
Santa Cruz do Sul – RS - Brasil
poll@unisc.com.br

RESUMO

O conceito de padrões juntamente com a modelagem temporal dos dados são dois tópicos que vem atraindo interesses independentes dos construtores de software nos últimos anos. O primeiro, porque segue uma filosofia de construção de sistemas baseada na reutilização de classes em orientação à objetos. E o segundo, porque possibilita modelar toda evolução temporal de uma informação. Padrões são um pequeno conjunto de classes que apresentam soluções semelhantes para contextos diferentes. A modelagem de aspectos temporais concentra as suas atenções sobre o armazenamento e o tratamento, em um modelo de dados, de toda a sua evolução temporal, ou seja, como a informação se comporta com o correr do tempo. Baseado nestes conceitos, o objetivo do trabalho é apresentar um padrão para os aspectos temporais dos sistemas, com as atenções voltadas para a modelagem das linhas de tempo de validade e de transação.

Palavras-Chaves: Padrões, Modelagem Temporal, Orientação à Objetos, Engenharia de Software

INTRODUÇÃO

Ao longo do tempo, os desenvolvedores de software perceberam que a maioria dos softwares existentes no mundo, apesar de apresentarem contextos diferentes, possuíam algo em comum. Através dessa percepção, buscou-se uma maneira pela qual estes problemas comuns pudessem apresentar soluções semelhantes e serem reaproveitadas de uma maneira eficaz e produtiva, bem como resolver o domínio do problema [Mil95]. O surgimento do conceito *padrões* junto ao paradigma de Orientação à Objetos foi uma maneira encontrada pelos projetistas para o reaproveitamento de partes de software, pois os padrões proporcionam a solução dos domínios comuns [Sil97].

Outro fato importante no desenvolvimento de software está relacionado com a modelagem dos aspectos temporais dos sistemas, ou seja, a forma de representar as propriedades estáticas (modelo de dados) e dinâmicas (modelos de transações) dos sistemas de informação. A modelagem de aspectos temporais se concentra nas propriedades dinâmicas onde os valores das propriedades variam com o tempo.

Este trabalho propõe o desenvolvimento de um padrão que atenda os problemas referentes a representação dos aspectos referentes a temporalidade dos dados em sistemas de informação.

O trabalho está organizado da seguinte maneira: na seção 2 é é apresentado o modelo de padrões para a modelagem temporal dos dados, na seção 3 é apresentada uma aplicação do modelo de padrões para fins de validação do trabalho. Por fim, na seção 4, são apresentadas as conclusões.

2. O MODELO PROPOSTO

Nesta seção, apresentaremos o resultado dos estudos realizados atendendo a proposta inicial de construir um padrão para aspectos temporais.

Serão abordados os problemas encontrados para se chegar a uma proposta de solução, onde o padrão proposto deve conceder uma liberdade ao usuário na escolha das classes que achar necessária para a sua implementação, ou seja, de acordo com a maneira de como irá armazenar os seus dados e a forma pela qual irá representá-los.

2.1 - Descrição do Padrão

Um modelo de padrão deve conter alguns tópicos que o descrevam detalhadamente para uma melhor compreensão deste[Gam94]. Nesta descrição, será apresentado primeiramente o nome do padrão, responsável pela sua identificação dentro de um dicionário de padrões. São apresentados também, alguns problemas encontrados em casos do mundo real, os quais nos possibilitaram levantar os tópicos necessários para definição do modelo a ser apresentado, bem como a solução definida para tais.

Nome : Temporal

Contexto : o padrão deve possibilitar ao usuário deste, a liberdade de escolha na modelagem de seu sistema. O padrão deve possibilitar a modelagem dos tempos de validade e de transação de forma separada ou com ambos os tempos. O usuário deve ter também, a possibilidade de modelar o tempo na forma única (ponto no tempo) ou através de dois pontos de tempo (limites de tempo), bem como, modelar o tempo através de horas, datas, ou ambos os tempos.

Problema: Um dos principais motivos que levaram a construção do padrão para aspectos temporais é que, informações temporais, tais como, valores e características de evolução temporal estão presentes em grande número de aplicações do mundo real [Ede98, Sno99], sendo que, os problemas de temporalidade apresentam soluções semelhantes. Como o conceito de padrões visa apresentar uma solução para os problemas que aparecem freqüentemente em domínio de problemas específicos.

Um motivo pelo qual se optou pela construção deste padrão, foi o de que representações de aspectos temporais são necessárias em sistemas de informação, sejam eles acadêmicos, comerciais, científicos ou outros tipos de sistemas que não tratam a informação referente ao tempo, para representar os dados que variam com o correr deste [Sno99]. Alguns exemplos de informação temporal são preço, endereço e salário. Elas são consideradas informações temporais porque sofrem constantes mudanças nos seus valores e, estas mudanças estão diretamente ligadas ao tempo.

Alguns dos problemas encontrados nos sistemas convencionais, é quanto ao número indefinido de datas que um sistema está sujeito, como também, as diferentes forma que os projetistas modelam suas informações temporais (através de horas ou datas, ponto no tempo ou dois pontos no tempo) . Outros problemas que surgem na modelagem do tempo em sistemas são:

- um mesmo sistema pode possuir várias datas que não possuem relação entre si, ou seja, cada data é representada de forma única, não indicando pontos correlatos (inicial e final). Ex.: data da compra: 12/10/98 e data da venda 15/10/98;
- um sistema em que se modela datas apenas com dois pontos (inicial e final). Ex.: data de validade para uma promoção comercial: data inicial = 10/10/98 e data final = 15/10/98;
- um mesmo sistema pode possuir várias datas onde algumas possuindo relação (indicam o inicio e o fim do tempo da informação) e outras datas, apenas referenciando um ponto na linha do tempo. Ex.: data de entrega da amostra: 09/10/98 (não possui relação com as demais data, pois sua representação é de forma única), data de inicio da análise: 10/10/98, data de fim da análise : 15/10/98 (possui relação com a data inicial de análise);

- alguns sistemas representando o tempo somente através de datas e outros através de horas;
- alguns sistemas representando ambos os tempos (datas e horas);
- representação do tempo de validade ou transação somente com um ponto no tempo;
- representação do tempo de validade ou transação com dois pontos de tempo;
- alguns representando apenas o tempo de validade e outros apenas o tempo de transação;
- alguns sistemas que queiram representar ambos os tempos (transação e validade).

Solução: Baseando-se nos problemas citados e nos conhecimentos oferecidos pela literatura de modelagem temporal, este trabalho propõe a seguinte solução:

O modelo de objetos é constituído de quatro classes onde duas delas estão para modelar os tempos de transação e duas para modelar os tempos de validade. Para cada tipo de representação de tempo existe classe com objetivos distintos. Uma classe trata do problema com um único ponto no tempo e a outra classe trata dos problemas com dois pontos no tempo. Além disto, os atributos desta classes devem atender a dois tipos de granularidade de tempo: data (dia, mês e ano) e horário (hora, minuto e segundo).

Esta solução pode ser utilizada por qualquer sistema que queira fazer uso do tempo para modelar informações de características temporais. Para construção do modelo de classes e métodos foram pesquisados em alguns sistemas do mundo real tópicos que apresentassem problemas com relação ao aspecto temporal, e concluiu-se que o modelo terá as seguintes definições:

Comentario [LdI1]: em Contexto, explicar os problemas encontrados. Esta parte descrita serve para a Solucao

- *Definição das classes* : o modelo de classes a ser construído deve atender aos diferentes tipos de modelo de dados (banco de dado temporais) existentes. Estes modelos de dados podem ser do tipo tempo de validade, tempo de transação ou bitemporal [Sno99]. Sendo assim o modelo de objetos é constituído de quatro classes para poder atender ao tipo de banco de dados que o projetista optar. São elas :

- Duas classes contendo um ponto no tempo, uma para modelar o tempo de validade e outra para modelar o tempo de transação, e;
- Duas classes contendo limites de tempo (dois pontos de tempo), uma para modelar o tempo de validade e outra para modelar o tempo de transação.

- *Definição dos atributos* : verificou-se que alguns dos sistemas faziam referência a um ponto no tempo e outros a dois pontos no tempo (tempo inicial e tempo final). Alguns modelos representam o tempo através de datas (dia/mês/ano) , outros através de datas e horas. Sendo assim as classes do padrão terão como atributos datas representando dia, mês e ano e horas representando horas, minutos e segundos.

- *Definição dos métodos* : os métodos foram definidos com base na Teoria dos Intervalos de Allen [All83] que trabalham basicamente sobre intervalos de tempo .

Comentario [LdI2]: ?

Na teoria de Allen, os eventos são representados por intervalos de tempo relacionados entre si através de relacionamentos temporais, descritos por predicados de uma linguagem lógica temporal. Os intervalos de tempo podem ser relacionados por meio de uma série de relações temporais, e suas inversas. Um conjunto de treze relações permitem expressar qualquer relacionamento entre dois intervalos. De acordo com a teoria de Allen, o mundo é descrito por asserções temporais sobre o que se sabe sobre o passado, o presente e o futuro.

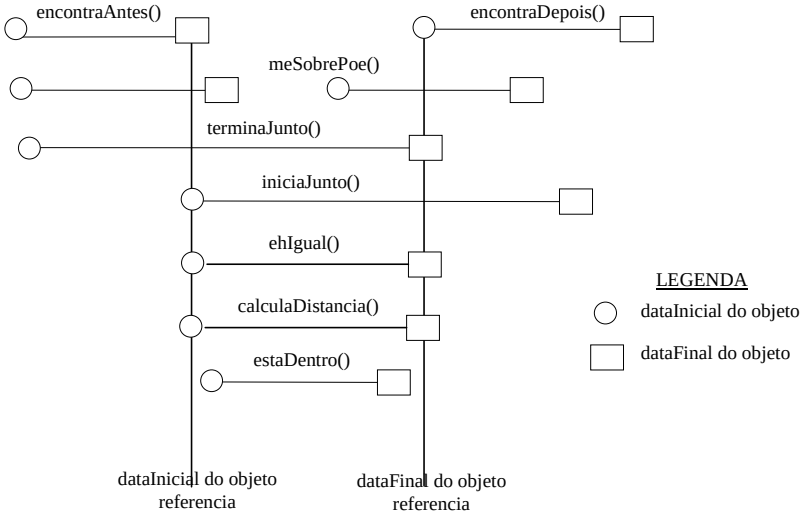

Figura 2.1 - Linha temporal demonstrando as funções de Allen

Na teoria de Allen, os eventos são representados por intervalos de tempo relacionados entre si através de relacionamentos temporais, descritos por predicados de uma linguagem lógica temporal. Os intervalos de tempo podem ser relacionados por meio de uma série de relações temporais, e suas inversas. Um conjunto de treze relações permitem expressar qualquer relacionamento entre dois intervalos. De acordo com esta teoria, o mundo é descrito por asserções temporais sobre o que se sabe sobre o passado, o presente e o futuro.

Após definido que o modelo de objetos é constituído de quatro classes elas foram assim denominadas :

- *ValidadeComposta* e *TransaçãoComposta* : nome das classes que modelam dois pontos de tempo (limites de tempo);
- *ValidaÚnica* e *TransaçãoÚnica*: nome das classes que modela apenas um ponto no tempo.

Classe ValidadeComposta - esta classe foi criada para modelar os tempos de validade de uma informação sendo constituída de dois atributos do tipo *Date* para modelar as datas (dia/mês/ano) de validade de uma informação e de dois atributos também do tipo *Time* para modelar as horas (horas, minutos, segundos) de validade de uma informação. Os atributos são definidos da seguinte maneira:

- dataValidadeInicial : representa o início de validade de uma informação na forma de data (dia/mês/ano inicial);
- dataValidadeFinal : representa o final da validade desta mesma informação na forma de data (dia/mês/ano final);
- horaValidadeInicial : representa o início da validade de uma informação na forma de horas (horas, minutos, segundos inicial);
- horaValidadeFinal : representa o final da validade desta mesma informação na forma de horas (horas, minutos, segundos final).

Todos os métodos de todas as classes que serão descritos a seguir , recebem os seguintes parâmetros:

Objeto de referência : será passado como parâmetro os tempos de inicio e fim de validade de um determinado objeto, sendo que, este (o objeto) será utilizado como referência para execução do método. Este objeto de referência pode ser qualquer objeto existente no modelo de dados utilizado, bem como um novo objeto a ser criado somente com as tempos de validade inicial e final.

Vetor de objetos : junto com objeto de referência é passado como parâmetro todos os outros objetos do modelo de dados através do vetor de objeto. Este vetor é passado como parâmetro para que os tempos dos objetos contidos nele possam ser comparados com os tempos do objeto de referência.

Sobre estes dois parâmetros que os métodos serão executados, onde os tempos dos objetos contidos no vetor de objetos serão comparados com os tempos do objetos de referência, retornando somente os valores válidos, de acordo com cada método.

	ID	Descrição	Preço	Validade inicial	Validade final
	0001	Nescau	2,20	10/05/98	15/07/98
	0001	Nescau	2,30	16/10/98	18/10/98
Objeto referência →	0001	Nescau	2,60	19/10/98	31/10/98
	0001	Nescau	2,30	01/11/98	03/11/98
	0001	Nescau	2,60	04/11/98	Null

TABELA 2.2 – Evolução temporal de informações com mesmo id através de pontos no tempo de validade

Descrição dos métodos da Classe Validade Composta

A tabela 2.2 é composta por dois atributos adicionais (validade inicial e validade final) junto a uma informação com características evolutivas. Ela será utilizada para exemplificar os métodos que serão executados sobre objetos que possuem o mesmo identificador externo, no caso da tabela de exemplo será o *ID* do objeto. Estes métodos são descritos a seguir:

a) ***meuPassado*** *(objeto de referência, vetor de objetos)* : Através dele será permitido recuperar todos os objetos do seu passado, ou seja, todos os objetos que estão vinculados pelo mesmo ID e que possuíram validade anterior ao tempo inicial do objeto de referência. Ex.: Através deste método, o objeto referência a ser enviado seria o que está marcado na tabela 2.2. Os valores a serem retornados por este método são aqueles com o tempo de validade anterior a do objeto passado como parâmetro.

b) ***meuFuturo*** *(objeto de referência, vetor de objetos)* : seguindo o mesmo procedimento do método anterior, os objetos a serem retornados agora, terão o tempo de validade superior ao tempo do objeto passado como parâmetro. No caso da tabela 2.2, os objetos resultantes serão os com tempo de validade inicial igual a 01/11/98 e 04/11/98.

c) ***meuPresente*** *(objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos válidos no momento. O usuário informará os dois pontos de tempo (validade inicial e validade final) . Estes pontos são transformados em um objeto, e, este será passado como objeto referência junto com o vetor de objetos obtendo como resultado os valores válidos naquele momento(valores que estão dentro dos pontos). Exemplificando este método através da tabela 2.2. Um determinado usuário quer saber quais os objetos válido no período de 18/10/98 a 01/11/98, o presente método retornará para este usuário como valor válido, o objeto marcado na tabela 2.2;

d) ***calculaDistância*** *(objeto de referência)* : este método calculará o tempo válido decorrido dentre os dois pontos(tempo inicial e final) do tempo de validade de um objeto.

Comentario [u3]: Referenciar a tabela

Comentario [u4]: Falta bibliografia

Os próximos métodos estaoDentro(), terminaJunto(), iniciaJunto(), encontraAntes(), encontraDepois(), ehIgual e meSobrepoe() serão exemplificados através da tabela 2.3 baseada no Sistema Acadêmico da Universidade de Santa Cruz do Sul onde construiu-se uma tabela de dados para explicar e exemplificar melhor os métodos.

Matric_aluno	...	Dt_início_abono	Dt_fim_abono	...	Cód_justif_abono
11111		10/10/98	15/10/98		0034
22222		13/10/98	21/10/98		1234
77777		13/10/98	17/10/98		3452
99999		18/10/98	21/10/98		1234
44444		12/10/98	14/10/98		0034

TABELA 2.3 - Evolução temporal de informações através de pontos de tempo de validade

e) ***estaoDentro*** *(objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que se encontram dentro dos dois pontos de tempo do objeto de referência. Ex: Suponhamos que um usuário deseje saber a quais alunos foram concedidos o abono dentro do período de 11/10/98 a 16/10/98. Ele envia estes tempos como o objeto de referência juntamente com todos os outros da tabela através do vetor de objetos, e, o método recuperará os alunos 11111 e 44444 por estes se encontrarem dentro do intervalo;

f) ***terminaJunto*** *(objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tiverem a sua validade final igual a validade final do objeto de referência. Ex: Se desejarmos saber quais os abonos que tem a sua validade terminada no dia 21/10/98, este método recuperará todos os alunos cujo seus abonos terminam nesta data, no caso da tabela 2.3, serão os alunos 22222 e 99999;

g) ***iniciaJunto*** *(objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tenham a sua validade inicial igual a validade inicial do objeto de referência. Ex: Se desejarmos saber quais os abonos tem a sua validade iniciada no dia 13/10/98, este método recuperará todos os alunos cujo seus abonos iniciam nesta data, no caso da tabela 2.3, serão os alunos 22222 e 77777;

h) ***encontraAntes*** *(objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tenham a sua validade final igual a validade inicial do objeto de referência. Ex.: Suponhamos que o objeto de referência possua a data de validade inicial igual a 21/10/98, os objetos resultantes deste método serão 22222 e 99999 (conforme tabela 2.3) porque possuem a sua data de validade final igual a data de validade inicial do objeto referência;

i) ***encontraDepois*** *(objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tenham a sua validade inicial igual a validade final do objeto de referência. Ex.: Se a data de validade final do objeto de referência for igual a 12/10/98, o objeto resultante deste método será (conforme tabela 2.3) 44444 porque possui a data de validade inicial igual a data de validade final do objeto referência;

j) ***ehIgual*** *(objeto de referência, vetor de objetos)* : este método permite capturar todos objetos cujos tempos de validade inicial e final coincidem com os tempos inicial e final do objeto referência. Ex.: Sendo os tempos de validade inicial igual a 10/10/98 e a data de validade final igual a 15/10/98 do objeto de referência, o objeto resultante deste método será o (conforma tabela 2.3)11111 porque as seus tempos de validade inicial e final coincidem com as do objeto de referência;

k) ***meSobrepoe*** *(objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que sobrepõe o objeto de referencia seja pelo tempo de validade inicial ou final. Ex.: Sendo a data de validade inicial igual a 16/10/98 e a data de validade final igual a 19/10/98 do objeto de referência, os objetos resultantes deste método serão(conforme tabela 2.3) 77777 por

que a sua validade final termina dentro dos pontos do objeto de referência e o 99999 porque este inicia a sua validade dentro dos pontos do objeto de referência.

Classe TransaçãoComposta - esta classe foi criada para modelar os tempos de transação de uma informação sendo constituída de 2 dois atributos do tipo *DATE* para modelar as datas (dia/mês/ano) de transação de uma informação e de 2 atributos também do tipo *TIME* para modelar as horas (horas, minutos, segundos) de transação de uma informação. Os atributos foram definidos da seguinte maneira:

- dataTransacaoInicial : representa o início da transação de uma informação na forma de data (dia/mês/ano inicial);
- dataTransacaoFinal : representa o final da transação desta mesma informação na forma de data (dia/mês/ano final);
- horaTransacaoInicial : representa o início da transação de uma informação na forma de horas (horas, minutos, segundos inicial);
- horaTransacaoFinal : representa o final da transação desta mesma informação na forma de horas (horas, minutos, segundos final).

Id	Descrição	Preço	Transação_inicial	Transação_final
0001	Nescau	2,30	01/10/97	31/12/98
0001	Nescau	2,40	31/12/98	01/06/98
0001	Nescau	2,50	01/06/98	NULL

TABELA 2.4 - Evolução temporal de informações de mesmo *id* através de dois pontos de tempo de transação

Descrição dos Métodos da Classe TransaçãoComposta

Estes métodos possuem a mesma funcionalidade dos métodos da classe validade composta, porém, serão executados apenas sobre os tempos de transação.

Funcionalidade dos métodos: todos os métodos (com exceção do método meuPresente() que recebe como parâmetro apenas o vetor de objetos) recebem como parâmetro um objeto de referência e um vetor de objetos contendo todos os objetos que possuem os tempos de transação junto a uma informação temporal, sendo que a funcionalidade, dos métodos será baseada sobre o objeto de referência. Os tempos de transação do objeto de referência serão comparadas com os respectivos tempos de transação dos objetos existentes no vetor.

a) *meuPassado (objeto de referência, vetor de objetos)* : este método permitirá recuperar todos os valores de um mesmo objeto anteriores ao atual. Ex.: enviando a linha marcada da tabela5.5 como o objeto de referência, o método retornaria os últimos valores alterados. Neste caso, os valores são R$ 2,30 e R$ 2,40.
b) *meuPresente (vetor de objetos)* : este método permite recuperar todos os objetos cujos tempos da transação final sejam iguais a NULL. Isto permitirá ao usuário saber quais os objetos cujo tempo de transação ainda não foi finalizado. No caso da tabela 2.4 o objeto resultante deste método será a linha marcada.
c) *calculaDistancia (objeto de referência, vetor de objetos)* : este método calculará o tempo decorrido dentre os dois pontos(tempo inicial e tempo final) do tempo de transação de um objeto.
d) *estaoDentro (objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que foram alterados e se encontram dentro dos dois pontos do tempo de transação do objeto de referência.

e) *terminaJunto (objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tiveram a sua alteração finalizada(tempos de transação final iguais) junto com objeto de referência.
f) *iniciaJunto (objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tiveram seus valores alterados no mesmo tempo (tempos de transação iniciais iguais) que o objeto de referência.
g) *encontraAntes (objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tenham o seu tempo de transação final igual ao tempo de transação inicial do objeto de referência.
h) *encontraDepois (objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que tenham o tempo de transação inicial igual ao tempo de transação final do objeto de referência.
i) *ehIgual (objeto de referência, vetor de objetos)* : este método permite recuperar todos objetos cujos tempos de transação inicial e final coincidem com os tempos de transação inicial e final do objeto referência.
j) *meSobrepoe (objeto de referência, vetor de objetos)* : este método permite recuperar todos os objetos que sobrepõe o objeto de referencia seja pelo tempo de transação inicial ou final.
k) *buscaTempoSistema()*: este método não possui relação alguma com os métodos anteriores, pois, sua função é apenas buscar o tempo do sistema, tanto para o tempo de transação inicial como final, sempre que ocorrer uma alteração de valor de um objeto.

Classe ValidadeÚnica - esta classe foi criada para modelar o tempo de validade de uma informação na forma única sendo constituída de 2 atributos, um para modelar apenas data (dia/mês/ano) e um para modelar hora (horas, minutos, segundos. Os atributos foram definidos da seguinte maneira :

- dataValidade : representa o dia em que uma informação foi válida na forma de data (dia/mês/ano);
- horaValidade : representa a hora em que uma informação foi válida na forma de hora (horas, minutos, segundos).

Este será utilizado como objeto de referência para exemplificar os métodos

Cód_reserva	Cód_sala	Cód_estagiário	Dia_hora_reserva
0001	01	01	10/10/98 13:30h
0002	01	01	10/10/98 17:00h
0003	02	01	10/10/98 17:00h
0004	03	02	11/10/98 9:00h
0005	03	02	12/10/98 8:00h

TABELA 2.5 - Representação temporal através de ponto no tempo

Descrição dos Métodos da Classe ValidadeÚnica

A tabela 2.5 apresenta um caso de uma informação temporal do tipo reserva, representada por um único ponto no tempo para exemplificar os métodos da classe.

a) *iniciaDepois (objeto de referencia, vetor de objetos)* : este método permite recuperar todos os objetos cujo tempo de validade seja superior ao tempo de validade do objeto de referência. Ex.: enviando como objeto de referência a linha marcada na tabela 2.5 (0002), o que este método retornará são 0004 e 0005;
b) *iniciaAntes (objeto de referencia, vetor de objetos)* : este método permite recuperar todos os objetos cujo tempo de validade seja inferior ao tempo de validade do objeto de referência. Ex.: seguindo o mesmo exemplo do método VemDepois, o que este método retornará será 0001;

c) *saoIguais (objeto de referencia, vetor de objetos)* : este método permite recuperar todos os objetos cujo tempo de validade seja igual ao tempo de validade do objeto de referência. Ex.: Enviando o mesmo objeto marcado na tabela 2.5 o retorno será 0003.

Classe Transação Única : esta classe foi criada para modelar o tempo de transação de uma informação na forma única sendo constituída de 2 atributos do tipo *DATE* um para modelar a data de alteração de uma informação (dia/mês/ano) e um para modelar a hora (horas, minutos, segundos). Seus atributos foram definidos da seguinte maneira :

- dataTransaçao : representa o dia em que uma informação foi alterada na forma de data (dia/mês/ano);
- horaTransaçao : representa a hora em que uma informação foi alterada na forma de hora (horas, minutos, segundos).

Descrição dos Métodos da Classe TransaçãoÚnica

Estes métodos possuem a mesma funcionalidade dos métodos da classe Validade Única, porém atuarão somente sobre o tempo de transação:

a) *iniciaDepois (objeto de referencia, vetor de objetos)* : este método permite recuperar todos os objetos cujo tempo de transação seja superior ao tempo de transação do objeto de referência. Através deste método o usuário poderá recuperar todos os valores alterados após a data passada com referência;
b) *iniciaAntes (objeto de referencia, vetor de objetos)* : este método permite recuperar todos os objetos cujo tempo de transação seja inferior ao tempo de transação do objeto de referência.Com este método será possível recuperar todos os valores anteriores ao objeto de referência;
c) *saoIgual (objeto de referencia, vetor de objetos)* : este método permite recuperar todos os objetos cujo tempo de transação seja igual ao tempo de transação do objeto de referência. Este método permitira saber quais os objetos foram alterados na mesma data;
d) *buscaTempoSistema()* : este método tem a função de buscar no sistema o tempo sempre que se deseja registrar a alteração de uma informação.

3. VALIDAÇÃO DO PADRÃO

Nesta seção, apresentaremos uma aplicação utilizando o padrão para comprovar a sua validade. Para isso, foi utilizado um caso do mundo real que apresenta soluções semelhantes e, que os aspectos temporais se encontram presentes. Para comprovar a validade do padrão, foi desenvolvido um exemplo de aplicação utilizando a classe Produtos, a qual é encontrada na maioria dos sistemas de informações comerciais e industriais.

Aplicação Tipo Preços - Temporal

A aplicação terá como base uma típica classe de um sistema de informação de supermercados; a classe Produtos. Esta classe possui uma propriedade de característica evolutiva (propriedade dinâmica), normalmente denominada Preço. Esta propriedade é considerada dinâmica, porque sofre constantes mudanças nos seus valores.

A aplicação utilizará somente as classes de transação composta para modelar os tempos de início e fim da alteração e a classe de Validade composta para modelar os tempos em que a informação terá validade.

FUNCIONALIDADE: os supermercados realizam durante os fins de semana promoção de alguns produtos para atrair seus clientes. Para um maior controle sobre esta mudanças de preço é importante registrar os tempos em que os valores desta propriedade são alteradas bem como os tempos em que ela vigorou ou vigorará. Sendo assim, incorporou-se juntamente com as propriedades básicas da classe Produtos quatro informações adicionais (tempos de validade e transação) na modelagem (ver tabela 5.1).

idProduto	Código do produto
preço	Valor do produto (propriedade denominada dinâmica)
data_inicial do vigor	Data que passa a valer o preço
data_final do vigor	Data que este valor deixa de valer
data da alteração	Data em que se iniciou a alteração deste preço

TABELA 5.1 Dicionário de dados resumido da classe preços

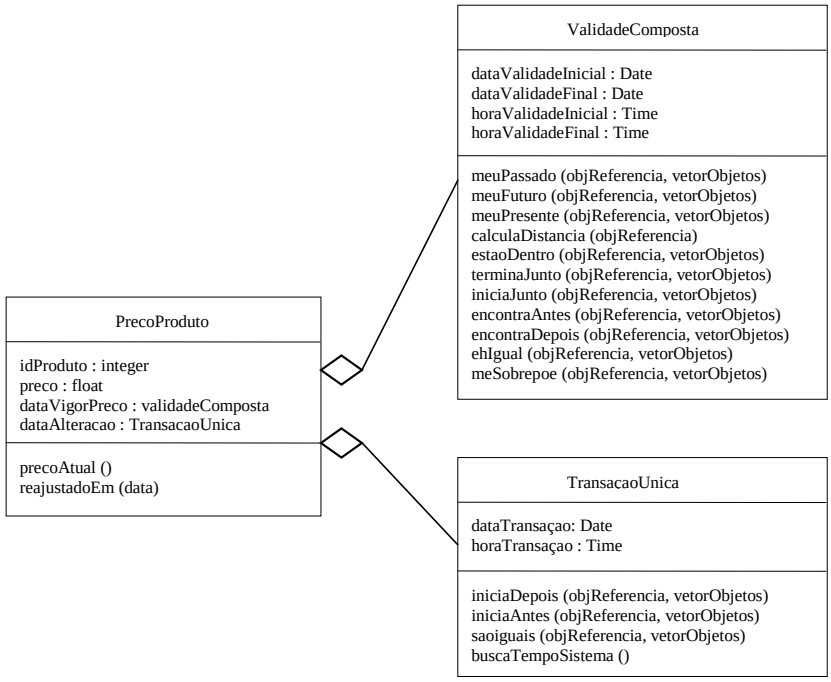

FIGURA 3.1 - Exemplo de aplicação do padrão Temporal

3.1 – Implementação do Padrão

A implementação do modelo foi desenvolvida utilizando-se a linguagem de programação Java (JDK1.1.5) por ela ser uma linguagem orientada a objetos. Como o trabalho propôs criar um padrão para aspectos temporais, ele trabalha exclusivamente com o tempo, sendo assim, verificou-se que e a linguagem de programação Java disponibiliza vários pacotes (*packages*) de classes que são de fundamental importância para a implementação deste padrão. Dentre os pacotes que a linguagem fornece se encontra o pacote *java.util*. Este pacote apresenta todas as características necessárias para que a implementação se transformasse em uma solução global, não apresentando características de qualquer aplicação e possibilitando a sua reutilização para outras aplicações.

Para fins de exemplo, a seguir apresentaremos o código de um dos métodos do padrão Temporal:

```
public Enumeration MeuPassado(Validade_Composta data_objeto_referencia,
Hashtable objetos)
{     Vector vetor_de_resposta=new Vector();
      Validade_Composta objeto_comparacao;
      Enumeration e=objetos.elements();
      Enumeration k=objetos.keys();
      if (e!=null)
      while (e.hasMoreElements())
        {       objeto_comparacao=(Validade_Composta) e.nextElement();
                 if (objeto_comparacao.data_final!=null)
                   if(objeto_comparacao.data_final.before
                        (data_objeto_refencia.data_inicial))
                        vetor_de_resposta.addElement(k.nextElement());
                   else
                        k.nextElement();
                else
                     k.nextElement();
        }
        return vetor_de_resposta.elements();
 }
```

4. Conclusões

O principal mérito do trabalho apresentado, é o de propor uma maneira de resolver um problema genérico referente à modelagem temporal de dados, que é o de manipular as funções de posição de objetos na linha do tempo. As funções de posição, são baseadas na teoria de Allen, a qual abrange todas possibilidades previsíveis de representação de objetos na linha do tempo.

A modelagem de objetos, que representa a solução proposta, foi desenvolvida utilizando-se os conceitos de padrões (*patterns*), onde se estrutura modelos de objetos que satisfaçam a solução do problema citado. A criação das classes temporais segue a linguagem de padrões, possibilitando o reuso destas classes através da associação do tipo agregação por composição. Sendo assim, o projetista define qual classe do padrão Temporal é mais adequada ao seu problema. A classe escolhida é inserida no problema como um tipo de objeto para o atributo em questão.

O trabalho foi implementado e testado usando os problemas que originaram a pesquisa (Sistemas de Automação de Supermercados e Sistemas de Controle Acadêmico da Universidade de Santa Cruz do Sul), onde o modelo apresentou resultados satisfatórios.

Bibliografia

[All83] ALLEN, J.F. **Maintaining Knowledge About Temporal Intervals.** Communications of the ACM, NY, v.26, n.11, p.832-843, Nov. 1983.

[Ede98] EDELWEISS, N. **Bancos de Dados Temporais.** Belo Horizonte: JAI'98 - UFMG, 1998.

[Fow97] FOWLER, M. **Analysis Patterns: Reusable Object Models**. Massachussets: Addison Wesley Longman, 1997.

[Gam94] GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. Massachussets, Addison Wesley Publishing Company, 1994.

[Mil95] MILI Hafedt. et al. **Reusing Software: Issues and Research Directions**. IEEE Transactions on Software Engineering, vol 21, nº. 6, junho 1995.

[Sil97] SILVA, Ricardo Pereira e; PRICE, Roberto Tom. **Em Direção a uma Metodologia para o Desenvolvimento de Frameworks de Aplicação Orientados a Objetos**. In: 11° SBES. **Anais...** Fortaleza-CE: UFCE,1997.

[Sno99] Snodgrass, R.T. **Developing Time-Oriented Databases Applications in SQL**. San Francisco - CA, Morgan Kaufman Publishers, 1999.

[Sou95] SOUKUP, J. *Implementing Patterns*. In: **Pattern Languages of Program Design** – Massachussets, editado por Coplien e Schmidt, Addison Wesley Publishing Company, 1995.