Resumo

Os Sistemas Gerenciadores de Banco de Dados Orientados a Objetos (SGBDOO) têm se firmado como alternativa para aplicações que necessitem manipular dados não convencionais. Devido à riqueza de conceitos, os projetos de aplicações que utilizam SGBDs OO são bem mais complexos do que no caso dos relacionais. O objetivo deste trabalho é investigar a adequação da metodologia OMT (Object Modeling Technique) a ambientes de SGBD OO. Para tanto, utilizamos o OMT para projetar uma aplicação multimídia e relatamos o resultado dessa experiência.

Palavras-chave

Metodologia Orientada a Objetos, Banco de Dados Orientado a Objetos, Projeto Portinari

Abstract

The Object-Oriented Database Management Systems (OODBMS) have become established as alternative to applications that must manipulate non-conventional data. Because of the wealth of concepts, application designs that use OODBMS are far more complex than relational systems. The aim of this project is to investigate the applicatility of the OMT methodology to OODBMS. Therefore, we have used the OMT to design a multimedia application and herein report the result of this experiment.

Keywords

Object-Oriented Methodology, Object-Oriented Database, Portinari Project

Utilizando a metodologia OMT para uma Aplicação de Banco de Dados Orientado a Objetos

Fabrícia C. V. Amaral José Renato L. Milet Rosana S. G. Lanzelotte

PUC-Rio - Departamento de Informática
E-mail:fabricia,milet,rosana@inf.puc-rio.br

Resumo

Os Sistemas Gerenciadores de Banco de Dados Orientados a Objetos (SGBDOO) têm se firmado como alternativa para aplicações que necessitem manipular dados não convencionais. Devido à riqueza de conceitos, os projetos de aplicações que utilizam SGBDs OO são bem mais complexos do que no caso dos relacionais. O objetivo deste trabalho é investigar a adequação da metodologia OMT (Object Modeling Technique) a ambientes de SGBD OO. Para tanto, utilizamos o OMT para projetar uma aplicação multimídia e relatamos o resultado dessa experiência.

1 Introdução

As metodologias para a construção de sistemas sofrem influências advindas das ferramentas utilizadas para implementar tais sistemas. Por exemplo, o aparecimento dos sistemas gerenciadores de bancos de dados (SGBDs) provocou o surgimento de metodologias para o projeto de banco de dados [Teo82] que passaram a fazer parte das metodologias de análise e projeto de sistemas.

Com a utilização das linguagens orientadas a objetos (OO) surgiram as metodologias OO para se modelar aplicações que seriam implementadas com estas linguagens. Quando são analisadas tais metodologias, constata-se que usualmente estão voltadas a projeto de aplicações com linguagens de programação OO (C++, SmallTalk) que são independentes dos sistemas de banco de dados. A figura 1 mostra como ferramentas de implementação influenciaram metodologias de construção de sistemas. Como se observa na figura 1, as metodologias OO ainda não incorporam conceitos encontrados em SGBDs OO.

Ao avaliarmos os SGBDs OO observa-se que além dos ítens de dados eles possibilitam a especificação de métodos associados aos objetos. Para o projeto de banco de dados OO as metodologias OO até então propostas podem ser utilizadas. Entretanto, torna-se necessário

verificar a adequação dessas metodologias aos requisitos específicos dos SGBDs OO. As metodologias de projeto de BDs tradicionais não são suficientes, pois o objetivo principal do próprio processo de projeto é formular as estruturas do banco de dados.

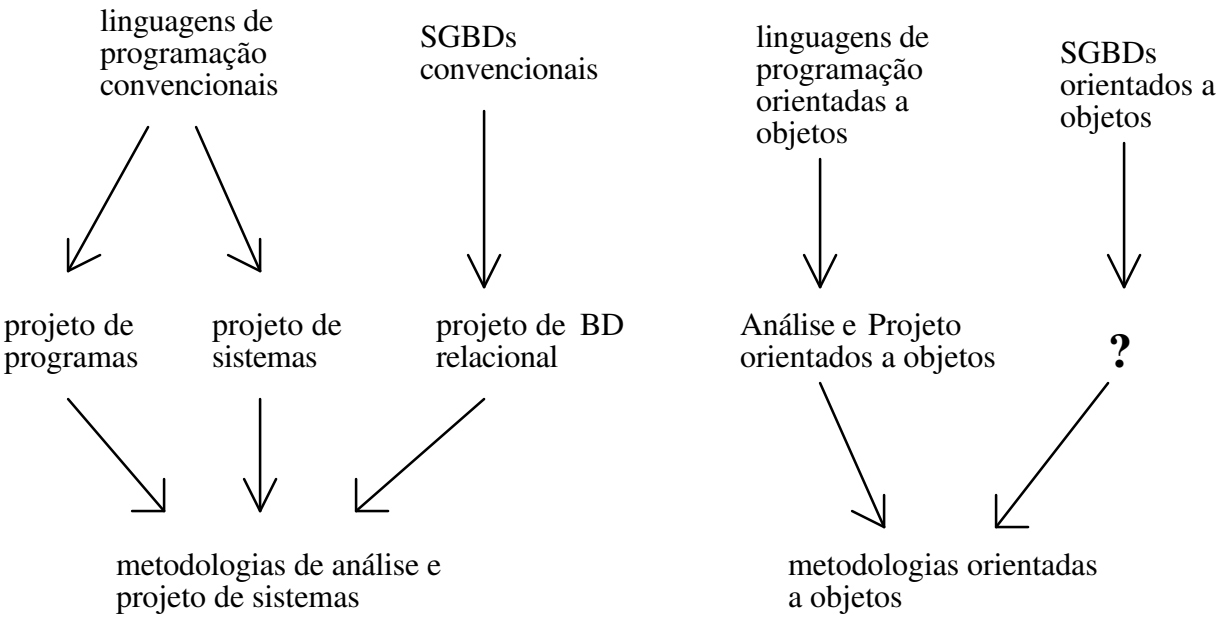


Figura 1: Evolução das metodologias

Um exemplo de uma metodologia representativa da abordagem OO é o OMT - Object Modeling Technique [Rum91], o qual é objeto de estudo neste trabalho.

O objetivo deste trabalho é investigar a adequação da metodologia OMT ao desenvolvimento de uma aplicação utilizando um SGBDOO. Pretende-se observar as lacunas da metodologia, quando utilizada para especificar tal sistema. A aplicação será implementada no âmbito do Projeto OPUS [Jab92] da PUC-Rio utilizando o SGBDOO O_2 [Ban88] em ambiente UNIX e estação de trabalho SUN.

Na seção 2 será descrita a aplicação no contexto do Projeto Portinari [Lan93] e o OMT será introduzido; na seção 3 apresenta-se o desenvolvimento da aplicação usando o OMT e comentários sobre sua adequação à fase de análise; na seção 4 é discutida a fase de projeto; na seção 5 será apresentada um guia de utilização do OMT; e a seção 6 conclui com algumas críticas e sugestões sobre a metodologia adotada.

2 A Aplicação e a Metodologia

A aplicação a ser modelada foi escolhida pelo interesse em se investigar a manipulação de dados multimídia - objetos do Projeto Portinari - utilizando-se o paradigma OO.

2.1 O Projeto Portinari

O Projeto Portinari teve início em 1979, na Pontifícia Universidade Católica (PUC) do Rio de Janeiro. Sua documentação reúne informações relacionadas às obras do pintor brasileiro Cândido Portinari (1903-1962), que servirão de plataforma à confecção do Catálogo Raisonné, e também dados que retratam diversos aspectos sócio-culturais do país neste século. Tendo como perspectiva disponibilizar a imensa quantidade dessas informações, tornou-se fundamental a construção de uma base de dados. Como o projeto tem um aspecto multimídia, fez-se necessário a escolha de um banco de dados que desse suporte a esta característica. Uma aplicação hipermídia do Projeto Portinari já foi implementada em ambiente de SGBD relacional [Mar93].

2.2 Especificação da Aplicação

A aplicação alvo do presente trabalho denominada CAD_Parede, poderá auxiliar museus ou exposições e mostras de arte para simular, por exemplo, as disposições possíveis de um ou mais quadros em uma parede. O usuário define uma parede a ser criada fornecendo suas dimensões, escala e cor e poderá manipular os quadros sobre essa parede. Deverá fazer parte deste projeto o armazenamento prévio das informações das obras (texto e imagem). A aplicação tornará disponível as obras cadastradas que poderão ser colocadas na parede. O usuário poderá também arrastar ou tirar estes quadros. A parede, com ou sem quadros, poderá ser armazenada para posterior recuperação.

2.3 A Escolha da Metodologia OO

Considerando-se que a aplicação descrita na seção anterior será implementada utilizando-se o paradigma OO, a escolha de uma metodologia de análise e projeto deverá seguir a mesma abordagem. Dentre as diversas metodologias OO, foram analisadas as metodologias de Booch [Boo91] e Rumbaugh - OMT [Rum91].

Booch dá maior relevância à fase de projeto do que a fase de análise. Ele mostra também como os modelos de projetos ficariam implementados em diversas linguagens de programação. A notação empregada no modelo de objetos é de difícil leitura. Os elementos dos diagramas são descritos textualmente .

A metodologia OMT enfatiza a fase de análise. Os modelos adotados nesta fase são semelhantes aos diagramas de entidades e relacionamentos (ER) [Chen76] e de Fluxo de Dados [You89] e possuem notação para representar os diversos aspectos (estrutural, comportamental e funcional) da aplicação. A maior parte das informações do sistema são apresentadas gráficamente, requerendo pouca informação textual para suplementá-las. Rumbaugh mostra o mapeamento de seus modelos para uma implementação em banco de dados.

Pelos motivos apresentados, o OMT foi o escolhido para o presente trabalho.

3 A Utilização da Metodologia OMT

A metodologia OMT compreende três modelos: *modelo de objetos*, *modelo dinâmico* e *modelo funcional*. Cada modelo representa uma visão diferente da aplicação que se quer modelar, ao mesmo tempo em que podem ser considerados complementares quanto a seus conteúdos. Estes modelos são construídos inicialmente com base na análise do domínio da aplicação e posteriormente, na fase de projeto, são complementados com detalhes referentes a requisitos de implementação.

3.1 O Modelo de Objetos

O modelo de objetos do OMT descreve a estrutura estática dos objetos em um sistema do mundo real. Este modelo mostra os objetos relevantes ao domínio da aplicação, seus relacionamentos e os atributos e operações que caracterizam cada classe de objetos.

A abordagem tradicionalmente mais utilizada para modelar informações tem sido o modelo de Entidades e Relacionamentos (ER). Na essência, o modelo OMT de objetos estende o modelo ER, combinando os conceitos de orientação a objetos (classe e herança) com os conceitos de modelagem de informação (entidades e associações). A figura 2 mostra o modelo de objetos da aplicação CAD_Parede cujo dicionário de dados apresenta-se a seguir, contendo apenas as classes obtidas com base na descrição do problema durante a fase de análise:

Parte do Dicionário de Dados - Descrição das Classes e Associações

Quadro: entidade que descreve os quadros.

Quadro-Pendurado: entidade que descreve os quadros que estão em alguma parede.

Parede: entidade que descreve a parede criada pelo usuário.

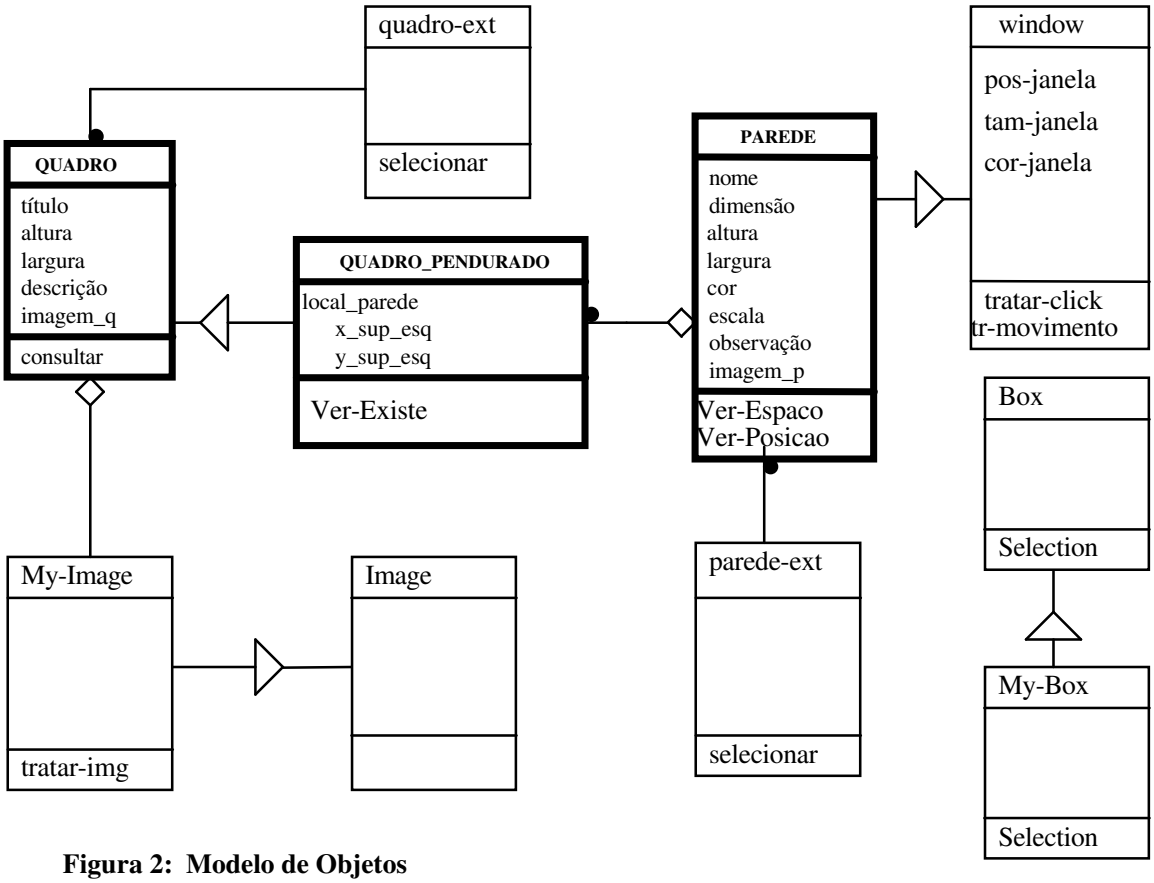


Figura 2: Modelo de Objetos

Como se vê na Figura 2, o modelo de objetos é um grafo, composto de nodos correspondentes a classes e arestas, que representam associações, generalizações e agregações, como se explica a seguir.

Uma *classe de objeto* descreve um grupo de objetos com propriedades similares (atributos), comportamento comum (operações), relacionamentos comuns com outros objetos, e semântica comum. Foi possível identificar as classes QUADRO e PAREDE como essenciais ao domínio da aplicação. Observando-se o modelo verifica-se que a notação OMT que representa uma classe é uma caixa retangular dividida em três partes, contendo: nome da classe, atributos dos objetos e operações aplicadas por ou para objetos da classe. De forma menos direta, tornou-se necessário criar uma terceira classe denominada QUADRO_PENDURADO. Esta classe, no modelo, aparece relacionando-se com a classe QUADRO via *generalização* e com a classe PAREDE via *agregação*, que serão explicadas a seguir.

Generalização é uma forma poderosa de abstração para compartilhamento de similaridades entre classes porém preservando suas diferenças. A classe QUADRO, que está sendo refinada, é chamada *superclasse* e a classe QUADRO_PENDURADO, que é a versão refinada da classe QUADRO, é chamada *sub-classe*. A classe QUADRO_PENDURADO herda todas as

propriedades (atributos e operações) de sua superclasse e adiciona suas próprias propriedades (atributo local_parede e operações tirar_quadro e arrastar_quadro). A notação OMT utilizada para representar esta generalização (herança) é um triângulo na aresta que liga a superclasse às suas subclasses.

Agregação é um relacionamento "parte-de" na qual objetos representando componentes de alguma coisa são associados com um objeto que representa toda a reunião. No exemplo, a classe QUADRO_PENDURADO é componente da classe PAREDE. A notação OMT utilizada é um pequeno losango na aresta junto à classe que reúne os componentes.

Associações são os meios para estabelecimento de relacionamentos entre objetos e classes. São os análogos ao relacionamento do modelo ER. As associações são notadas no OMT por arestas que ligam objetos ou classes.

Resumidamente, o diagrama apresentado pode ser lido da seguinte forma: QUADRO_PENDURADO é um QUADRO e faz parte de uma parede, e PAREDE é composta de zero ou mais QUADROS_PENDURADOS. A multiplicidade (zero ou mais) é representada por um círculo cheio na extremidade da associação. As classes WINDOW, IMAGE, MY_IMAGE, BOX, MY_BOX, PAREDE_EXT e QUADRO_EXT serão discutidas no projeto dos objetos, seção 4.2.

3.2 Modelo Dinâmico

O modelo de objetos capturou somente a estrutura estática do sistema. Os aspectos dinâmicos são representados no *modelo dinâmico* e estão relacionados à seqüência de operações que ocorre nos objetos ao longo do tempo, provocado por estímulos de um objeto a outro ou por interações com o usuário da aplicação. Exemplos de interação:

Usuário entra operação *criar parede*
Sistema solicita *escala, dimensão e cor*
Usuário entra *escala, dimensão e cor*

No exemplo acima, quando o usuário **entra operação criar parede**, dizemos que um *evento* ocorreu a partir de um estímulo do usuário e provocou uma *mudança de estado*; o sistema, que aguardava a entrada da **operação criar** passa a **solicitar escala, dimensão e cor**.

A seqüência de interações do exemplo acima chama-se *cenário* e é a primeira etapa para construção do modelo dinâmico. O cenário permite visualizar a seqüência de eventos que deve ocorrer na aplicação e ajuda a evitar que passos importantes sejam esquecidos.

Em seguida representa-se, em um diagrama chamado *event trace* , os objetos que enviam e recebem os eventos. Este diagrama (figura 3) mostra cada objeto como uma linha vertical e cada evento como uma seta horizontal do objeto que envia para o que recebe.

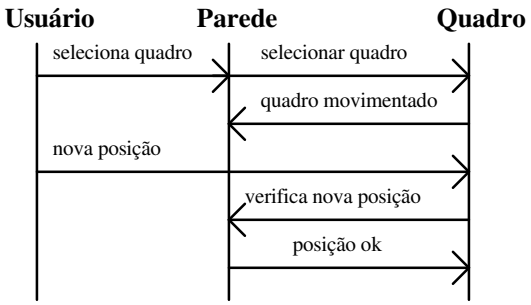


Figura 3: Event trace

À partir do event trace é possível grupar todos os eventos de acordo com os objetos dos quais eles partiram e onde eles chegaram através do *diagrama de fluxo de eventos*. Isto permite obter uma visão geral do comportamento do sistema como um todo.

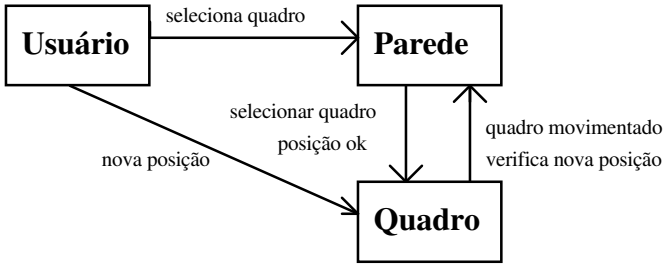


Figura 4: Diagrama de Fluxo de Evento

A partir do event trace, são construídos os *diagramas de estados.* Para cada classe, existirá um diagrama de estados, que captura os diversos estados que um objeto da classe pode assumir. O conjunto de todos os diagramas de estados constitui o modelo dinâmico. Um diagrama de estados, que utiliza a representação gráfica de máquinas de estado finito, é um grafo cujos nós são *estados* e cujos arcos são *eventos*; especifica a seqüência de estados provocada por uma seqüência de eventos.

A figura 5 mostra o diagrama de estados da classe Parede.

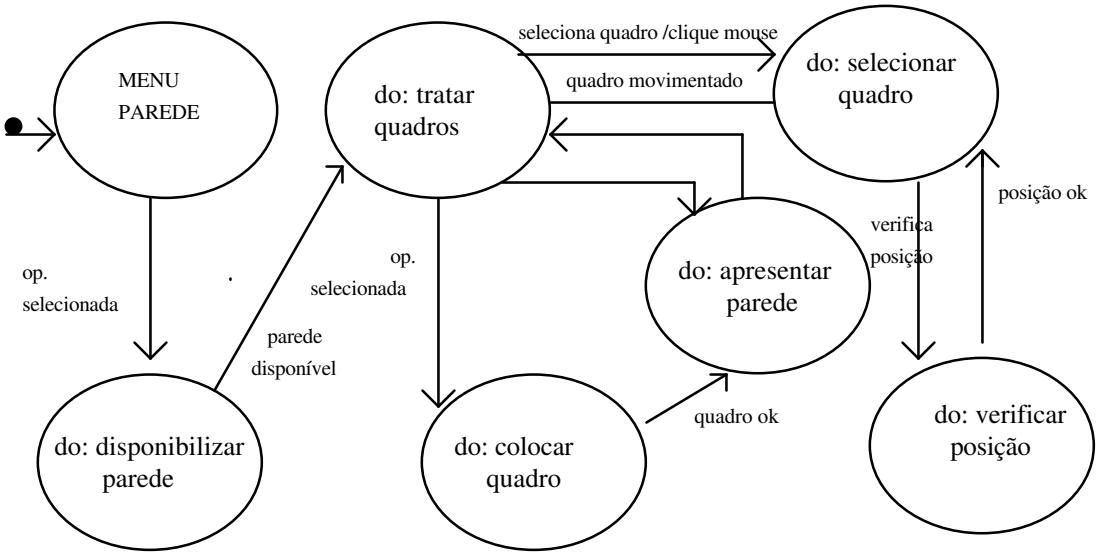


Figura 5: Diagrama de Estado da Classe Parede

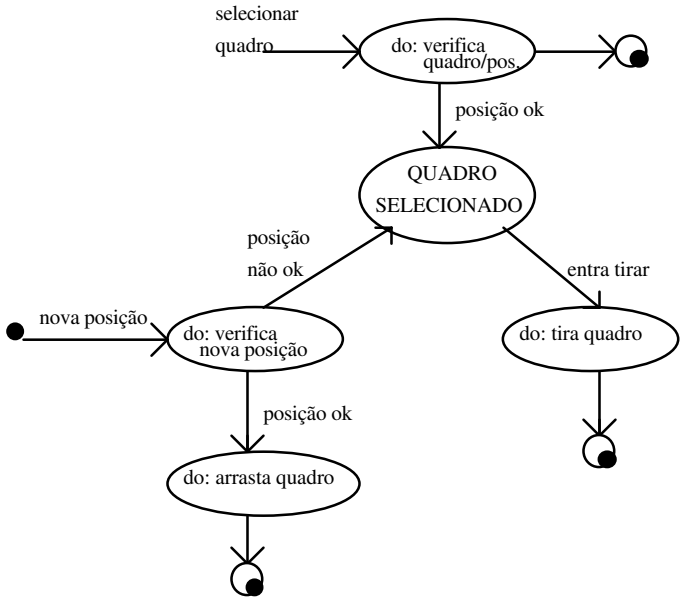


Figura 6: Diagrama de Estado da classe Quadro

Os estados são desenhados como figuras arredondadas e os eventos são arcos dirigidos ligando o estado atual a um novo estado alvo.

Cada estado pode ter, associado a ele, uma *atividade* a ser executada. Atividades são operações que tomam tempo para se completar e são representadas pela notação **do:** no diagrama; enquanto a *ação*, associada a um evento, é uma operação instantânea e indicada pela barra (**/**) após o nome do evento. Uma ação também pode ser associada à entrada e saída de um estado.

Na figura 5, **seleciona quadro /clique mouse** é um exemplo de evento com uma ação associada. Os estados iniciais e finais podem ser observados no diagrama através de um ponto cheio na entrada e de um ponto cheio dentro de um círculo, na saída.

No exemplo da aplicação é possível observar que há estados associados a objetos de classes diferentes. O estado onde se encontra a atividade **selecionar quadro** (figura 5) torna-se um evento de mesmo nome no diagrama da classe **quadro** (figura 6), caracterizando troca de eventos entre objetos de classes diferentes, sem uma indicacão específica na notação utilizada.

O modelo dinâmico mostra-se bastante adequado para representar a sequência de eventos das interfaces de usuário, como é o caso da aplicação Portinari. Dessa maneira, não é preciso representar a interface de usuário no modelo funcional, como acontecia no passado.

3.3 Modelo Funcional

O modelo funcional descreve aqueles aspectos de um sistema relacionados com transformações de valores e restrições nestas transformações. O modelo funcional é representado por diagramas de fluxo de dados (DFD).

Diagramas de Fluxo de Dados mostram a transformação dos valores sem considerar quando ou como as funções são executadas. As funções são invocadas no modelo funcional como ações no modelo dinâmico e são mostradas como operações nos objetos do modelo de objetos.

O modelo funcional pode ser construído em camadas hierarquicamente distribuídas. Uma camada de nível mais alto pode ser decomposta em processos mais simples. A figura 7 mostra o diagrama de fluxo de dados de nível superior para o exemplo CAD_Parede:

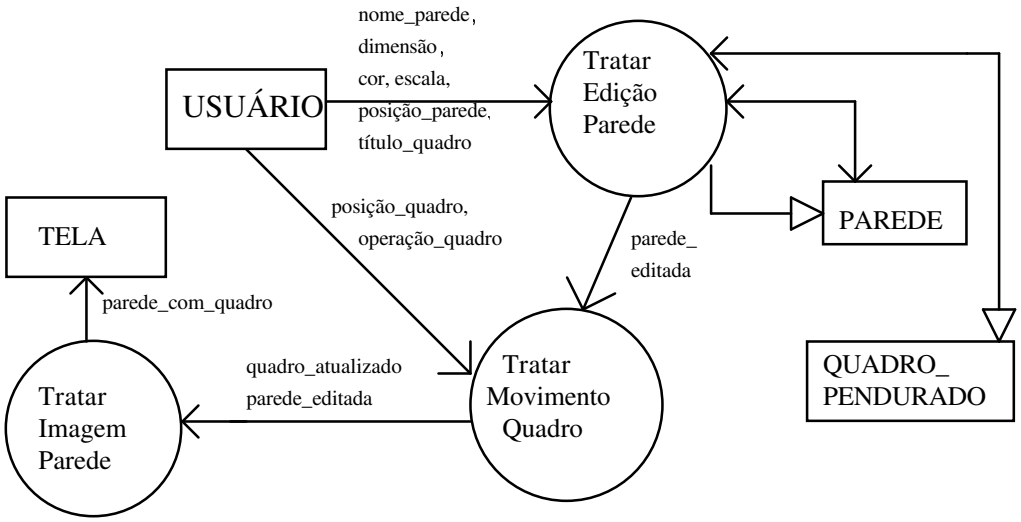


Figura 7: Modelo Funcional

Observa-se no exemplo que uma notação adicional foi introduzida em relação ao DFD tradicional. O fluxo de dados que gera um objeto é indicado por um triângulo vazio em sua extremidade. O processo **Tratar Edição Parede** poderá criar novo objeto na classe PAREDE. Por outro lado, a operação de atualização de atributos é indicada por uma ponta de seta.

Os processos de nível mais alto poderão ser expandidos em novos DFDs. Naturalmente, a recursão irá parar e os processos atômicos deverão ser descritos em linguagem natural, equações matemáticas ou por outros meios. Isto não será mostrado no exemplo; mas para melhor entendimento, as funções de nível mais alto serão descritas a seguir:

Processo Tratar Edição Parede: este processo poderá criar uma parede nova através dos dados de entrada escala, dimensão e cor, carregar ou gravar com o nome-parede fornecido, ou criar um objeto na classe QUADRO_PENDURADO quando a operação for colocar.

Processo Tratar Movimentação Quadro: este processo verifica se o quadro selecionado poderá ser arrastado para a nova posição desejada pelo usuário, atualizando seu valor na classe QUADRO_PENDURADO ou eliminado o objeto desta classe se a operação for tirar.

Processo Tratar Imagem Parede: Este processo fará o "refresh" da parede com os quadros nas suas novas posições.

No exemplo, do processo **Tratar Edição Parede** deriva-se a operação **colocar** para a classe PAREDE. O processo **Tratar Movimentação Quadro** gera as operações **tirar-quadro** e **arrastar-quadro** na classe QUADRO_PENDURADO.

O modelo funcional, além dos DFDs, contém as restrições de integridade do sistema relativas às transformações nos objetos.

Restrições:

- uma parede não pode conter o mesmo quadro mais de uma vez;
- um quadro não pode sobrepor-se a outro quadro na parede.

3.4 A Adequação do OMT para a fase de Análise

Tendo concluido a fase de análise da aplicação, apresenta-se algumas observações e críticas ao uso do OMT para esta atividade.

O processo de modelagem na fase de análise foi feito de maneira dinâmica. À medida em que cada modelo vai sendo construído, acréscimos podem ser feitos nos outros modelos já desenhados. Eles representam visões complementares entre si e seus elementos também fazem parte dos outros modelos, porém em níveis de detalhe diferentes.

O modelo de objetos, em geral, requer múltiplas versões, resultado do processo iterativo que ocorre na formalização dos modelos dinâmico e funcional. Por vezes torna-se necessário acrescentar atributos, especializar classes e incluir operações. As operações das classes no modelo de objetos do CAD_PAREDE foram obtidas do modelo dinâmico originadas dos eventos e do modelo funcional originadas dos processos. Alterações de requisitos da aplicação que ocasionalmente possam ocorrer, na maioria das vezes, são mudanças nas funções. Estas mudanças constituirão de adição ou alteração das operações, deixando a estrutura básica dos objetos inalterada.

No modelo de objetos, a notação utilizada para a descrição das classes, apesar de ser expressiva e de fácil leitura, não contém um diferenciador para alguns tipos de classes que surgiram após a fase de análise e que não faziam parte da descrição do problema.

No modelo dinâmico, à medida em que os diagramas de estados vão sendo construídos, poderá ser necessário complementar os demais diagramas com detalhes que ainda não tenham sido considerados, cuidando para que o modelo dinâmico como um todo guarde a consistência entre seus diagramas.

No modelo dinâmico a notação oferece meios de se representar eventos, ações e atividades para cada classe. Porém, a troca de mensagens entre objetos de classes diferentes não é muito clara. Observando-se os diagramas de estados para as classes PAREDE e para a classe QUADRO, a atividade *do: selecionar quadro* na classe PAREDE gera o evento denominado *seleção de quadro* na classe QUADRO.

No OMT, o modelo funcional é construído utilizando-se a mesma abordagem da análise estruturada tradicional. A metodologia não mostra como se relacionam os diagramas de estados e o DFD, não estando definido no modelo como os processos representam as atividades do modelo dinâmico. A construção do DFD, na maioria das vezes é feita baseada somente na decomposição funcional.

4 A Fase de Projeto

Nesta fase define-se como será a arquitetura do sistema, identificando-se as partes componentes ou subsistemas, alocação de recursos de hardware, software de controle, gerenciamento de dados e indicadores que possam ser tomados como referência para as decisões de implementação, tais como performance e uso de espaço de memória.

4.1 O Projeto do Sistema

O sistema gerenciador de banco de dados utilizado é o O_2 cujo paradigma é orientado a objetos. O O_2 tem também uma linguagem de programação embutida, o O_2C, utilizado para a implementação dos métodos.

A arquitetura do sistema é do tipo interface interativa. Portanto, o controle de sequência da aplicação, orientada por eventos, está capturado pelo modelo dinâmico do OMT.

O software de controle externo é direcionado a eventos com a utilização do sistema de interface de usuário X Window. O O_2Look é uma interface de usuário gráfica para o SGBDOO O_2, opera sob o Sistema X Window e faz uso do gerenciador gráfico Motif.

4.2 O Projeto dos Objetos

Nesta etapa, devem ser especificadas novas classes, que são úteis para a implementação.

4.2.1 Classes de Implementação

As classes discutidas na seção 3.1 não são suficientes para a implementação da interface com o usuário. Por este motivo, foi incluído no projeto a classe WINDOW e foram definidas todas as operações que ela deverá ter para tratar a manipulação do *mouse*. Como o usuário trabalhará sempre sobre a parede, conclui-se que a classe PAREDE deverá herdar as propriedades da classe WINDOW e escreve-se novamente os métodos que deverão dar tratamento diferente a seus objetos.

A classe IMAGE é uma classe pré-definida pelo O_2. Na presente aplicação, definiu-se uma especialização desta, a classe MY_IMAGE, que possibilita definir métodos específicos de manipulação da imagem das obras.

A classe BOX também é uma classe pré-definida pelo O_2. Esta classe contém métodos para construção de diálogos com o usuário final. A classe BOX é suficiente para especificação de diálogos simples. O formato de apresentação da tela pode ser definido pelo programador. Para este projeto, a classe MY_BOX especializará a classe BOX para que sejam definidas as telas de diálogo com o usuário.

As classes PAREDE_EXT e QUADRO_EXT foram criadas para permitir a manipulação do conjunto de objetos e também serem nomeadas com o objetivo de se tornarem persistentes. Este procedimento é um requisito do O_2.

4.2.2 Definição das Operações

O modelo de objetos da fase de análise não precisa mostrar as operações. O projetista deve converter as ações e atividades do modelo dinâmico e os processos do modelo funcional em

operações vinculadas às classes no modelo de objetos. Fazendo-se esta conversão, mapeia-se a estrutura lógica do modelo de análise na organização física do programa.

Observando-se o diagrama de estados da classe PAREDE, o estado ***Colocar-Quadro*** irá produzir na classe QUADRO-EXT uma **operação selecionar,** que tornará disponível ao usuário todo o conjunto para que se faça uma seleção, retornando em seguida o quadro escolhido. Esta interação com o usuário necessitará da **operação selection** na classe MY-BOX, que define novamente a **operação selection** da classe BOX, para prover a interface. Na classe QUADRO-PENDURADO deverá ter a **operação ver-existe** que verifique se aquele quadro já não foi alocado àquela parede. Na classe PAREDE para este evento, a **operação ver-posição** verificará se o quadro desejado não se sobrepõe a outro e a **operação ver-espaço** calculará se existe espaço disponível na parede.

Neste ponto, o projetista deverá projetar os algoritmos para implementar as operações.

5 Um Guia de Uso do OMT

Esta seção utiliza os conceitos do OMT introduzidos nas seções anteriores durante a modelagem da aplicação CAD_Parede e descreve os passos da metodologia que foram seguidos para produzir cada um dos diagramas já apresentados. Com a prática na construção dos diagramas, alguns desses passos poderão ser realizados simultaneamente.

5.1 A Fase de Análise

A fase de análise do OMT parte da especificação da aplicação, investigando-se seus requisitos, restrições e realizando os procedimentos necessários ao completo entendimento do enunciado do problema.

Partindo da especificação da aplicação na seção 2.2, observa-se que um usuário deverá interagir com o CAD_Parede para definir uma parede fornecendo as informações necessárias para que o sistema possa criá-la e em seguida selecionar os quadros que deverão ser pendurados, arrastados ou tirados da parede, que poderá ser armazenada.

Com base nesses requisitos, os modelos: de objetos, dinâmico e funcional do OMT deverão ser construídos.

5.1.1 O Modelo de Objetos

Para a construção do modelo de objetos procura-se identificar, na especificação da aplicação, os objetos que deverão compor as classes do domínio da aplicação, observando-se que tipo de relação eles mantêm entre si. Cada um desses objetos e seus atributos são definidos em um

dicionário de dados. O diagrama de classes e o dicionário de dados formam o modelo de objetos. A seguir, estão descritos os procedimentos para a construção do modelo de objetos do OMT que foram seguidos no desenvolvimento da aplicação CAD_Parede.

Identificar objetos e classes. Com base no enunciado da aplicação, algumas possíveis classes de objetos podem ser identificadas: *museus, exposições, mostras de arte, quadros, parede, usuário, obras (texto e imagem)*. Dessas, somente *obra* (como um sinônimo de *quadro*) e *parede* são os objetos do domínio da aplicação. Usuário é apenas a pessoa que interage com o sistema e por isso faz parte do modelo dinâmico da aplicação.

Preparar um dicionário de dados. Além da descrição das classes no contexto da aplicação, as associações e atributos também devem fazer parte do dicionário de dados. O dicionário de dados das classes da aplicação CAD_Parede encontra-se no anexo 1.

Identificar associações (incluindo agregações) entre classes. No enunciado da seção 2.2 é possível observar uma associação *um para muitos* entre *parede* e *obra*. Uma outra interpretação que cabe é a de que *obra é parte de parede*, indicando uma associação de agregação entre a classe *parede* e a classe *obra*. Os passos seguintes ajudam a ter um melhor entendimento do domínio da aplicação, que está sendo representado no modelo de objetos, através da identificação dos atributos das classes e da investigação de suas possíveis especializações.

Identificar atributos. À medida em que as classes vão sendo identificadas, é possível listar alguns de seus atributos. Os atributos da classe *parede* são: *nome*, *dimensões* e *cor*. A classe *obra* utiliza um subconjunto de atributos, extraído do Projeto Portinari para a aplicação CAD_Parede, que são: *título*, *dimensões* e *fotografia* da obra. Para esses atributos, *quadro* mostrou-se uma denominação de classe mais apropriada do que *obra*. Um quadro que estiver colocado numa parede deverá conter a informação da posição que ele ocupa. Portanto, o atributo *posição* deverá ser acrescentado à classe *quadro*. A figura 2, na seção 2.1 mostra o modelo de objetos com esses atributos.

Organizar e simplificar classes de objeto usando herança. O uso de herança identificando aspectos comuns entre as classes vai refletir no compartilhamento de suas estruturas. A classe *parede* pode ser especializada em *parede sem quadro* e *parede com quadro* de acordo com o estado de conter ou não quadros. A classe *quadro* pode ser especializada em *quadro pendurado*, representando o estado de um quadro quando está colocado em uma parede. Com isso, a relação

de agregação, que havia inicialmente entre as classes *parede* e *quadro*, passa a ser entre suas especializações *parede com quadro* e *quadro pendurado*; isto é, *um ou mais quadros pendurados fazem parte de uma parede com quadro*, como se pode ver na figura 8.

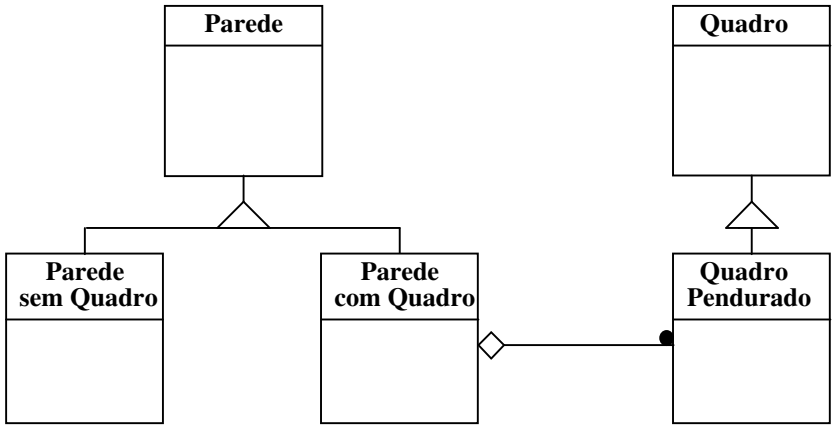


Figura 8: O uso de herança nas classes da aplicação CAD_Parede

Nesta figura é possível observar que faz mais sentido à classe *quadro pendurado* conter o atributo *posição*, que indica sua localização na parede.

Verificar os caminhos de acesso para as consultas. Neste momento deverá ser verificado se o modelo de objetos possui os caminhos necessários através das associações para atender às consultas que deverão ser formuladas pelos usuários. Uma consulta que está especificada para a aplicação CAD_Parede é a recuperação de uma parede com os seus quadros. O caminho de acesso para obter esta consulta poderá ser obtida com a relação de agregação entre as classes *parede com quadro* e *quadro pendurado*, como está mostrado no modelo de objetos da figura 8.

Iterar e refinar o modelo. O modelo de objetos após o processo de refinamento é o da figura 2, enfocando somente as classes desenhadas com as linhas mais grossas. Neste processo não foram mantidas as classes especializadas de *parede* pelo fato de não haver necessidade de compartilhamento de seus atributos e operações, para esta aplicação. A classe *quadro pendurado* é um quadro com o acréscimo de atributos, relacionados com sua localização na parede. Neste momento o tipo *coordinate_type* foi criado para conter as coordenadas das dimensões tanto de parede como de quadro e também para ser o tipo do atributo *posição*. Verificou-se que o OMT não apresenta uma notação específica para representar a definição de tipos.

Grupar as classes em módulos. Para aplicações muito grandes, a metodologia OMT sugere que o conjunto de classes do modelo de objetos possa ser dividido em subconjuntos menores, de acordo com algum critério que os identifique, para permitir uma visualização melhor do modelo como um todo. A aplicação CAD_Parede não é complexa o bastante para que suas classes necessitem ser agrupadas em módulos.

5.1.2 O modelo Dinâmico

As trocas de eventos que ocorrem em uma aplicação são registradas no modelo dinâmico do OMT. Este modelo é composto por um conjunto de diagramas de estado. Os passos necessários à construção desses diagramas é mostrado a seguir, para a aplicação CAD_Parede.

Preparando um cenário. Os cenários contêm os diálogos mais comuns entre o usuário externo e o sistema. Este primeiro passo ajuda a organizar esses diálogos, sem a preocupação em serem completos e nem sobre a maneira como as informações são obtidas. O anexo 2 mostra um cenário do CAD_Parede. É possível observar nesse diagrama as condições de exceção que ocorrem nos diálogos.

Identificando Eventos. Com base no cenário construído no passo anterior, pode-se observar tanto eventos trocados entre um usuário externo e o sistema (por exemplo, *opção criar parede)* quanto eventos entre objetos. Uma vez identificados, esses eventos são organizados nos diagramas *event-trace*. Estes diagramas podem ser mais completos do que os cenários e ajudam a visualizar os eventos de entrada e saída de cada classe de objetos. O event-trace da aplicação CAD_Parede está mostrado no anexo 2. Os eventos de entrada e saída para cada objeto podem ser grupados no diagrama de fluxo de eventos dando uma visão mais espacial dos eventos entre as classes de objetos. O *event trace* poderia ser usado para representar as assinaturas correspondentes à passagem de parâmetros entre as operações das classes às quais elas pertencem.

Construindo um Diagrama de Estados. Um diagrama de estados é construído para cada classe da aplicação, podendo-se optar em construir diagramas somente para aquelas consideradas mais importantes devido ao número de eventos enviados e recebidos. Na aplicação CAD_Parede, foram construídos diagramas de estados para as classes: parede, quadro e quadro pendurado, e para o objeto CAD_Parede. Estes diagramas estão mostrados no anexo 2. Com o objetivo de simplificar o modelo, a classe CAD_Parede engloba o conjunto de classes de interface necessárias à aplicação e inclui: janelas, menus e imagens. A sua definição consta da seção 4.2.1. O diagrama de estados

da classe *parede* encontra-se no anexo 2. Destes diagramas reúnem as operações relacionadas com a parede. Nestas figuras, é possível observar a correspondência dos eventos nos diagramas *event trace* e os diagramas de estados.

Checando eventos entre os objetos. Após a construção dos diagramas de estados, esses diagramas são verificados entre si, procurando-se observar a coerência entre os eventos através dos diagramas, que devem estar relacionados com o diagrama event trace ou o cenário. Após a construção dos diagramas de estados, que juntos formam o modelo dinâmico da aplicação, é possível identificar as operações sobra cada classe de objetos. O diagrama *event trace* poderia ser utilizado para que fossem acrescentadas as assinaturas através dos parâmetros trocados entre os objetos. Estes parâmetros podem ajudar a verificar se os eventos estão sendo associados às classes corretamente. Esse procedimento pode ajudar a eliminarem-se dúvidas sobre que eventos estão relacionados com que classes.

5.1.3 O modelo Funcional

O modelo funcional mostra as transformações dos fluxos de informações que ocorrem em um sistema sem representar quando ocorrem, pois este aspecto de sequência no tempo foram considerados no modelo dinâmico.

Identificar valores de entrada e saída. O primeiro passo para a construção do modelo funcional procura identificar as informações que entram e saem do sistema. Estas informações, em geral, são os parâmetros mostrados no event trace. O anexo 3, mostra a representação dessa troca de informações para a aplicação CAD_Parede. Ela contém o usuário como um agente externo e a aplicação CAD_Parede, desenhados em um retângulo e as informações representadas pelas setas indicando o sentido de entrada ou saída das informações do sistema.

Construíndo diagramas de fluxo de dados. O diagrama de fluxo de dados é construído em camadas ou níveis, isto é, um processo num nível mais alto pode ser decomposto em vários processos, e assim por diante, mantendo-se a correspondência dos fluxos de entrada e saída, para cada camada de processos. Porém, em cada um desses níveis é possível que surjam novos fluxos internos que ligam os processos do mesmo nível. O anexo 3 mostra o diagrama de fluxo de dados da aplicação CAD_Parede. Nesses diagramas, é possível observar que o modelo funcional da figura 7 foi decomposto em mais um nível. Um diagrama de fluxo de dados especifica somente dependência entre operações e não a sequência entre elas. Este sequenciamento é parte do modelo dinâmico.

Descrevendo as funções. Cada processo do diagrama de fluxo de dados deverá ser descrito de forma procedural ou declarativa, podendo-se utlizar linguagem natural, pseudocódigo ou outra forma mais apropriada. Estas descrições devem enfatizar o que o processo faz e não como faz. A seção 3.3 mostra uma descrição das funções representadas pelos processos.

Identificando restrições entre objetos. As restrições podem ser sobre dois objetos ao mesmo tempo ou em tempos diferentes. As restrições na aplicação CAD_Parede são: uma parede não pode conter dois quadros iguais; e um quadro não pode estar sobre outro quadro na parede. Estas restrições pode ser observadas no modelo dinâmico seguindo-se o evento *pendurar quadro*.

5.2 Adicionando Operações

Com os modelos de objetos, dinâmico e funcional construídos, as operações podem ser obtidas observando-se os eventos no modelo dinâmico e as funções no modelo funcional. Eventuais consultas também poderão ser traduzidas em operações, com base no modelo de objetos. Por exemplo, obter uma parede que tenha sido solicitada pode ser feita executando-se uma query. Na aplicação CAD_Parede pode-se listar as operações das classes baseando-se nas atividades contidas nos diagramas de estados correspondentes. Porém, as atividades de uma classe podem ser eventos em uma outra. Neste caso verifica-se que atividade este evento gera nesta outra classe, que correponderá à operação e a classe que estará associada a esta operação. Por exemplo, a atividade solicita criar parede na classe CAD_Parede é um evento no diagrama de estados da classe parede que deve executar a operação criar parede e será esta a operação e a classe que deverão ser consideradas. A seguir, foram listadas as operações associadas com cada classe da aplicação.

Operações da classe CAD_Parede

- solicita informações parede
- selecionar parede
- mostrar parede
- solicita confirmação
- solicita posição

Operações da Classe Parede

- criar parede
- apagar parede
- obtém lista parede

- obtém parede
- valida posição
- pendurar quadro
- desvincula (quadro pendurado) da parede

Operações da Classe Quadro Pendurado

- verifica quadro pendurado
- posicionar quadro
- retirar quadro

Operações da Classe Quadro

- lista de quadros
- obtém quadro

O conjunto dessas operações é analisado sobre a melhor organização entre elas e suas respectivas classes. Algumas dessas operações juntas corresponderão a um processo do modelo funcional. Por exemplo, o processo *mover quadro* no modelo funcional faz parte das operações *valida posição* e *pendurar quadro.*

5.3 A Fase de Projeto e Implementação

Os modelos desenvolvidos durante a fase da análise continuam a ser trabalhados na fase de projeto, para atender aos requisitos necessários à implementação em um ambiente computacional definido. As etapas desta fase estão descritas na seção 4. Nesta seção serão acrescentadas as considerações de projeto voltadas para o SGBDOO O_2.

O modelo de objetos do OMT não captura todos os conceitos de um SGBDOO como o O_2. A fase de projeto, conforme foi visto na seção 4, propõe-se a fornecer a definição completa das classes, algoritmos dos métodos e interfaces, necessários para se implementar a aplicação. Nesta fase descreve-se o esquema do SGBDOO O_2 que consiste de:

a) Definição das Classes, incluindo a especificação de tipos, métodos e relacionamentos entre as classes;
b) Definição dos objetos nomeados, valores nomeados e tipos nomeados;
c) Definição da aplicação;
d) Definição das funções.

Ao definirem-se alguns destes ítens observam-se algumas limitações no OMT que refletem-se na fase de implementação:

- No modelo do O_2 tem-se o conceito de classes e tipos. A classe descreve a estrutura do objeto e um conjunto de métodos. Uma instância de uma classe é o objeto. O objeto tem uma identidade que lhe permite ser compartilhado por outros objetos. O aspecto do dado de um objeto é chamado valor e a estrutura do valor é denominado tipo. O tipo não tem identidade. Nota-se no OMT que não se pode distinguir entidades modeladas como classes de entidades modeladas como tipos no modelo de objetos.

- Ao se fazer uma aplicação usando um SGBDOO, um novo enfoque aparece em relação a implementação utilizando uma linguagem de programação: o conceito de persistência. O modelo de persistência de um SGBDOO define a maneira pela qual os dados persistentes e transientes são criados. No SGBDOO O_2 a persistência é associada ao conceito de nomes. Tudo que é nomeado persiste e parte de nome persistente é persistente. Portanto para que o conjunto de objetos de uma classe seja persistente é necessário declarar-se um nome associado a esta classe. O OMT não distingue, no modelo de objetos, as classes cujas instâncias devam permanecer após a execução do programa, das classes cujos objetos são voláteis. O mesmo se aplica a tipos.

- Num ambiente orientado a objetos é de se supor a existência de biblioteca de classes pré-programadas que poderão ser reutilizadas nas aplicações. Isto requer uma notação específica para as classes pré-definidas no SGBDOO O_2, das quais se fará reuso.

6 Conclusão

No presente trabalho descreveu-se a utilização da metodologia OMT para implementar uma aplicação utilizando o SGBDOO O_2. Foram identificadas as classes e relacionamentos com base na análise do domínio da aplicação. Em seguida, construiu-se o modelo dinâmico para representar a troca de eventos entre os objetos. Através das ações e atividades do modelo dinâmico foram acrescentados, no modelo de objetos, as operações associadas aos métodos e suas assinaturas. Posteriormente, construiu-se o modelo funcional no qual descrevem-se as transformações de valores decorrentes das operações do modelo de objetos.

Observa-se, já na fase de análise, que o modelo dinâmico assumiu maior importância que o modelo funcional, pois a sequência de eventos das interfaces com o usuário é mostrada no diagrama de estados que faz parte daquele modelo. Entretanto tornou-se difícil perceber estados e

eventos compartilhados por duas classes de objetos distintos. Outra dificuldade encontrada foi em associar-se claramente os processos do modelo funcional às ações e atividades do modelo dinâmico, pois o modelo dinâmico foi construído para cada classe e o modelo funcional é feito para o sistema como um todo.

Nas fases de projeto e implementação foram definidas novas classes para possibilitarem interface com o usuário e atender aos requisitos do SGBDOO O_2.

A aplicação aqui descrita encontra-se em fase de conclusão. Como trabalho futuro, pretende-se aprofundar as extensões ao OMT no sentido de:

- Utilizar o *event trace* para especificar o conjunto de métodos de cada classe;
- Integrar melhor as visões do sistema proporcionadas pelos modelos dinâmico e funcional;
- Propor extensões à metodologia OMT para se adequarem a projetos de aplicações em SGBDs OO.

Referências

[Ama94] F. C. V. Amaral, "OMT_2 : Utilizando uma Metodologia Orientada a Objetos para Projeto de Aplicações com SGBDs Orientados a Objetos", Dissertação de Mestrado em desenvolvimento, Pontifícia Universidade Católica - Rio de Janeiro, 1994.

[Ban88] F. Bancilhon, G. Barbedette, V. Benzaken, C. Delobel, S. Gamerman, C. Lecluse, P. Pfeffer, P. Richard, F. Velez, "The Design and Implementation of O_2 an Object-Oriented Database System", Advances in Object-Oriented Database Systems - Proceedings of the Second International Workshop on Object-Oriented Database Systems, K. Dittrich (ed.) Bad Munster, FRG, Setembro 1988.

[Boo91] G. Booch, "Object-Oriented Design", Redwood City, Calif.: Benjamin/Cummings, 1991.

[Chen76] P. P. S. Chen, "The entity-relationship model - toward a unified view of data", ACM Transactions on Database System, março 1976.

[Jab92] S. Jablonski, A. M. Brito, A. Ribenboim, R.S.G. Lanzelotte, "Projeto OPUS: Evoluindo para plataformas de SGBDOO", VII Simpósio Brasileiro de Banco de Dados, maio 1992.

[Lan93] R. S. G. Lanzelotte, M. P. Marques, M. C. S. G. Penna, J. C. Portinari, F. D. Ruiz, D. Schwabe, "The Portinari Project: Science and Art team up together to help Cultural

Projects", International Conference on Hypermedia and Interactivity in Museums, setembro 1993.

[Mar93] M. Marques, R. S. G. Lanzelotte, "Combining Hypermedia and Databases - The Portinari Project", VIII Simpósio Brasileiro de Banco de Dados, maio 1993.

[Rum91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, "Object-Oriented Modeling and Design", Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[Teo82] T.J.Teorey, J.P.Fry; Design of Database Structures, Prentice Hall, Englewood Cliffs, New Jersey,1992.

[You89] E. Yourdon, "Modern Structured Analisys", Englewood Cliffs, New Jersey, Yordon Press, 1979.

Anexos:

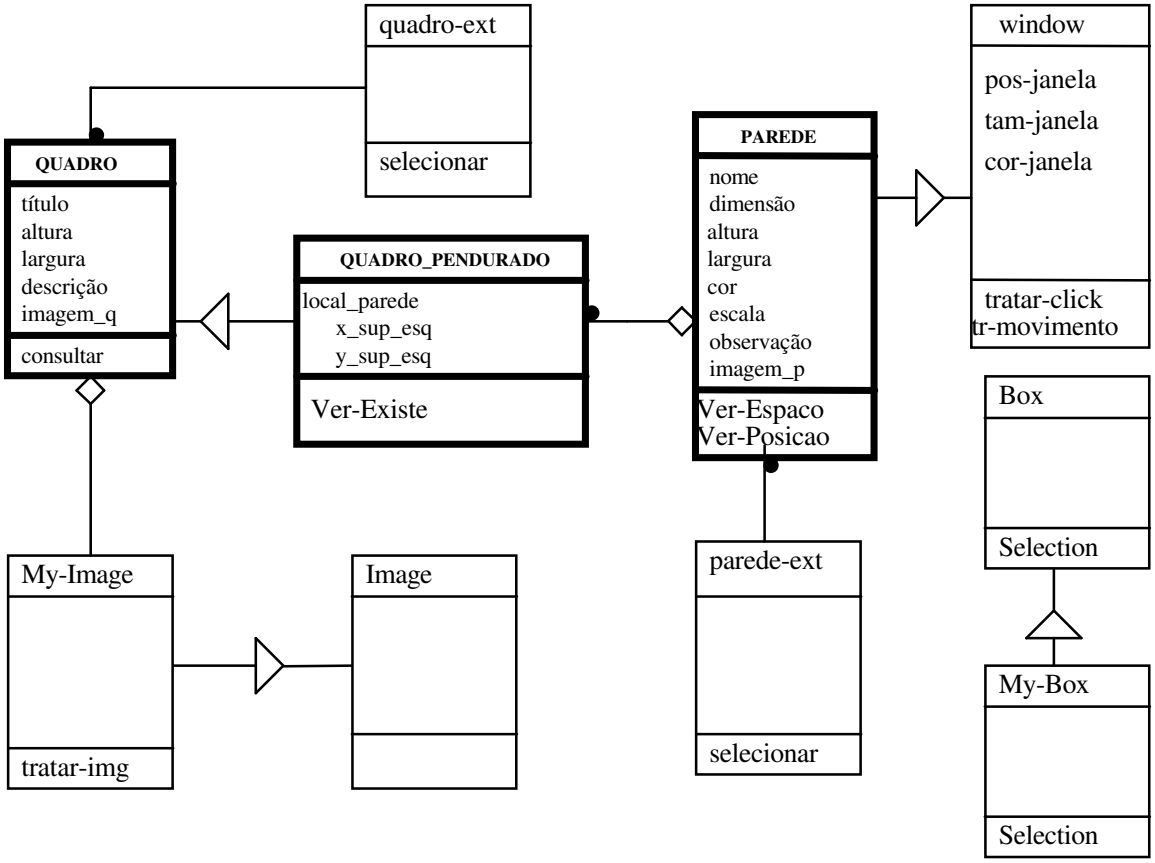
1 - O Modelo de Objetos

2 - O Modelo Dinâmico

3 - O Modelo Funcional

Anexo 1 - O Modelo de Objetos da Aplicação CAD_Parede

Modelo de Objetos da Aplicação CAD_Parede



Dicionário de Dados - Descrição das Classes, Associações e Atributos

Classes e Associações

Quadro: entidade que descreve os quadros.

Quadro-Pendurado: entidade que descreve os quadros que estão em alguma parede.

Parede: entidade que descreve a parede criada pelo usuário.

Atributos da Classe Parede

nome: é o nome de uma parede definida pelo usuário.
dimensão: altura e largura reais de uma parede.
cor: nome da cor da parede.
escala: a relação entre o tamanho original da parede e o tamanho na tela.
observação: espaço para comentários do usuário.
imagem_p: a fotografia da parede definida.

Atributos da Classe Quadro

título: o nome do quadro.
altura: o comprimento do quadro no lado vertical.
largura: o comprimento do quadro no lado horizontal.
descrição: contém uma descrição sobre o quadro.
imagem_q: contém a fotografia do quadro.

Atributos da Classe Quadro Pendurado

local_parede: a posição que o quadro ocupa na parede tomando-se como referência o seu canto superior esquerdo.

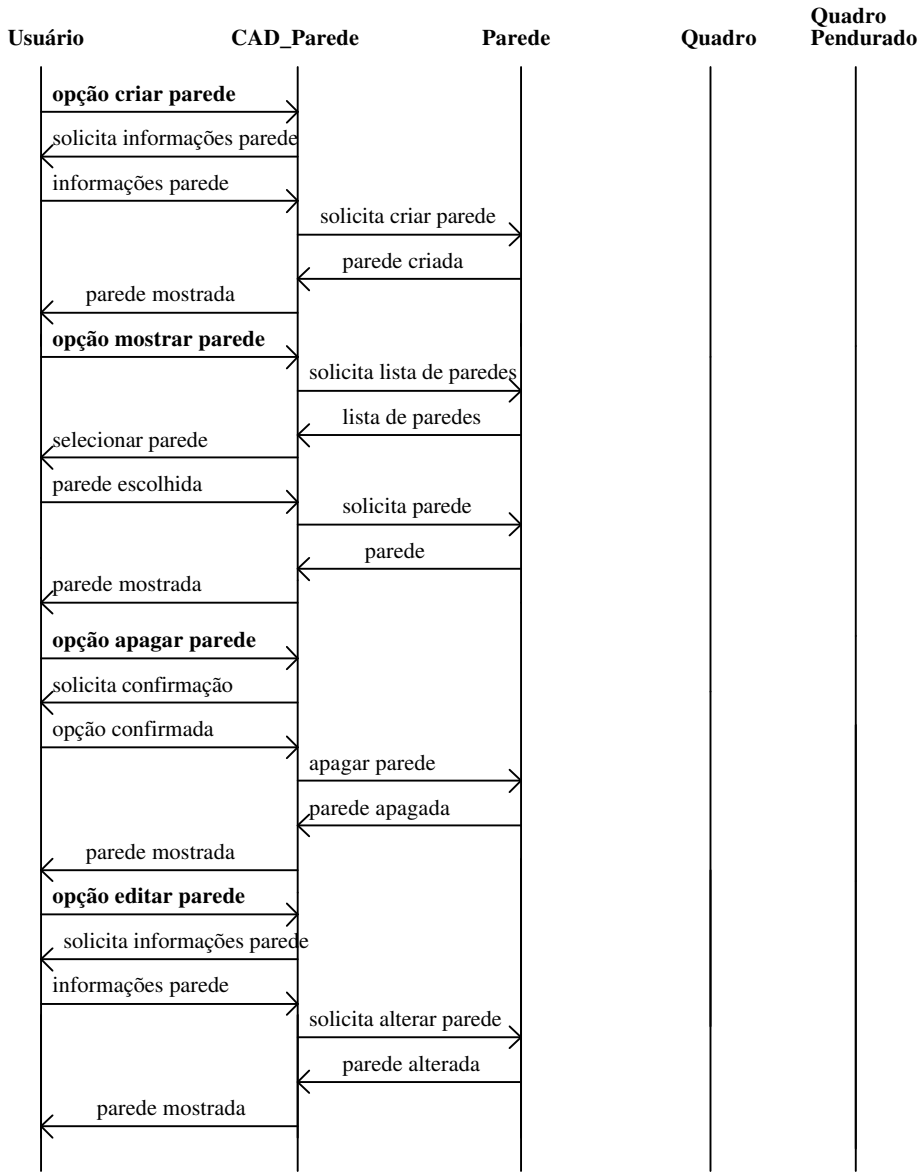
Anexo 2: O Modelo Dinâmico da Aplicação CAD_Parede

Um Cenário da Aplicação CAD_Parede

O CAD_Parede solicita ao usuário a opção desejada
O Usuário escolhe a opção criar parede
O CAD_Parede solicita escala, dimensões, cor, título
O Usuário preenche escala, dimensões, cor, título
O CAD_Parede verifica os dados informados pelo usuário e cria a parede
ou solicita nova entrada caso contrário
O CAD_Parede mostra a parede
O Usuário escolhe a opção pendurar quadro
O CAD_Parede mostra a lista de quadros
O Usuário seleciona um quadro
O CAD_Parede mostra o quadro selecionado
O Usuário move o quadro para a parede na posição desejada
O CAD_Parede valida a posição do quadro na parede;
Se posição válida, mensagem OK; senão, mensagem
"posição inválida - posicione novamente ou cancele".
O Usuário escolhe opção mover quadro pendurado na parede, movendo o quadro para a
nova posição
O CAD_Parede valida a posição do quadro ...
O Usuário escolhe a opção tirar quadro pendurado
O CAD_Parede solicita a confirmação
O Usuário confirma; ou o usuário cancela a opção apagar
O Usuário escolhe a opção mostrar parede
O CAD_Parede mostra a lista de paredes
O Usuário seleciona uma parede
O CAD_Parede mostra a parede

O Usuário escolhe a opção alterar parede
O CAD_Parede mostra as informações da parede e solicita escala, dimensões e cor ...

Diagrama Event Trace da Aplicação CAD_Parede



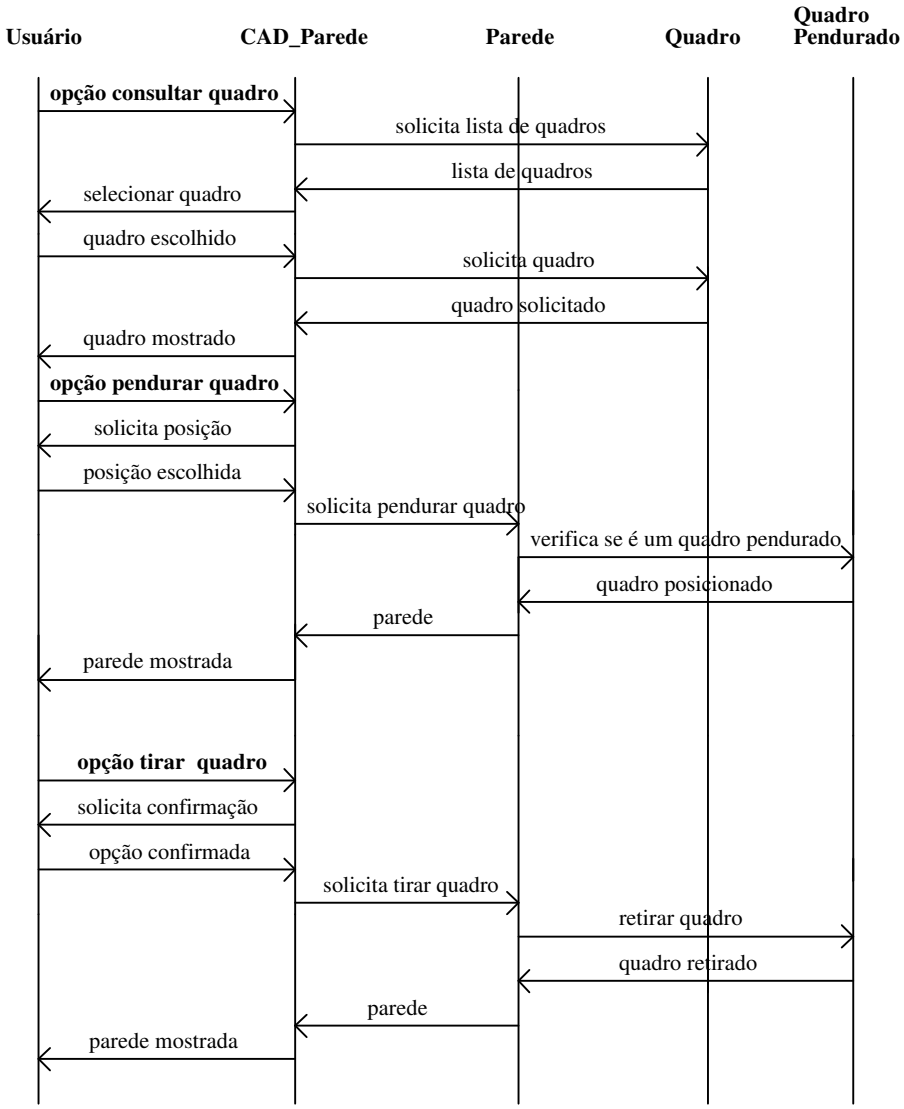

Usuário
CAD_Parede
Parede
Quadro
Quadro Pendurado
opção consultar quadro
solicita lista de quadros
lista de quadros
selecionar quadro
quadro escolhido
solicita quadro
quadro solicitado
quadro mostrado
opção pendurar quadro
solicita posição
posição escolhida
solicita pendurar quadro
verifica se é um quadro pendurado
quadro posicionado
parede
parede mostrada
opção tirar quadro
solicita confirmação
opção confirmada
solicita tirar quadro
retirar quadro
quadro retirado
parede
parede mostrada

Diagrama de Estados da Classe CAD_Parede

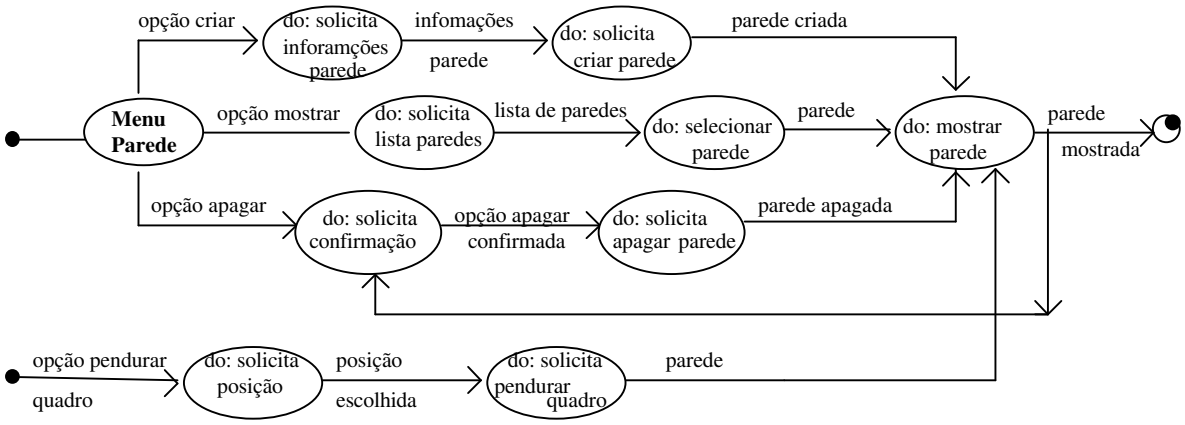


Diagrama de Estados da Classe Parede

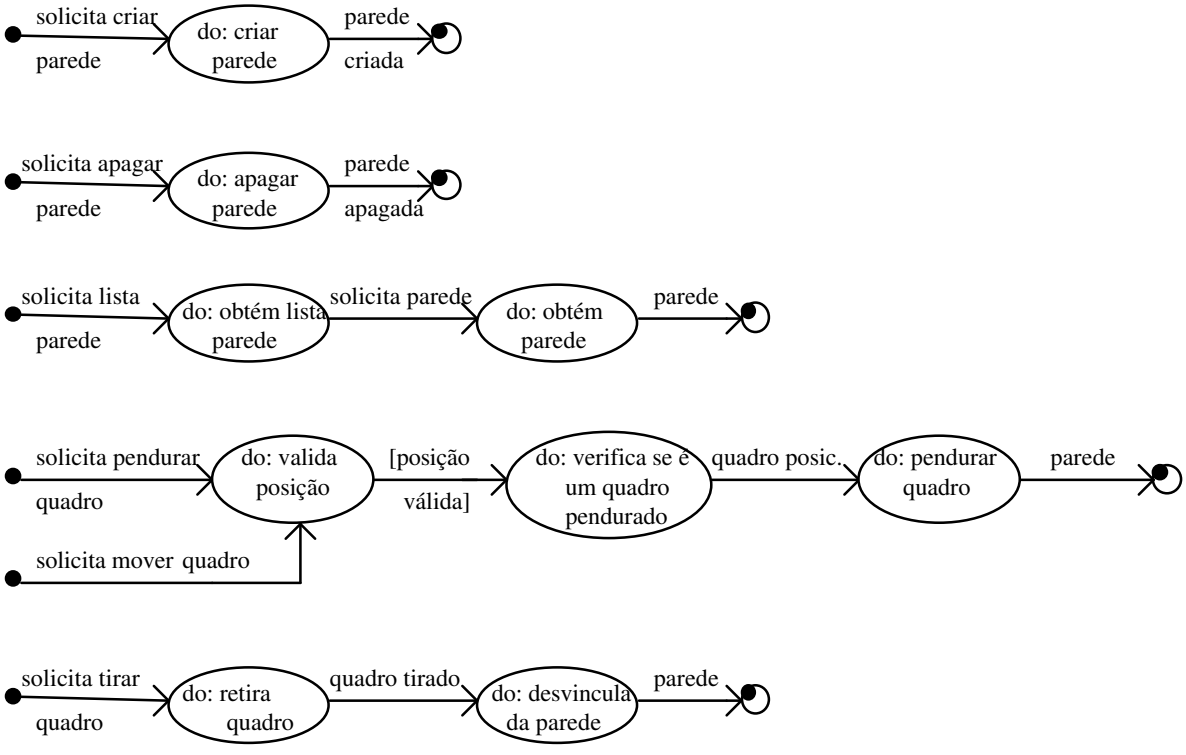


Diagrama de Estados da Classe Quadro Pendurado

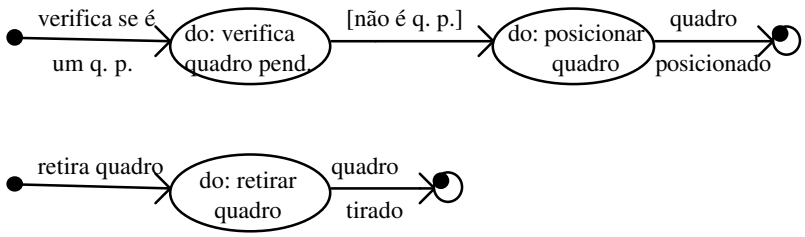
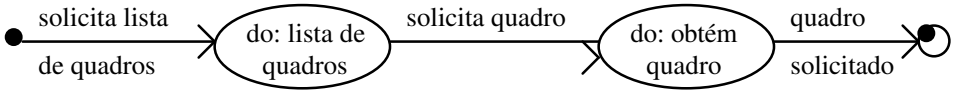
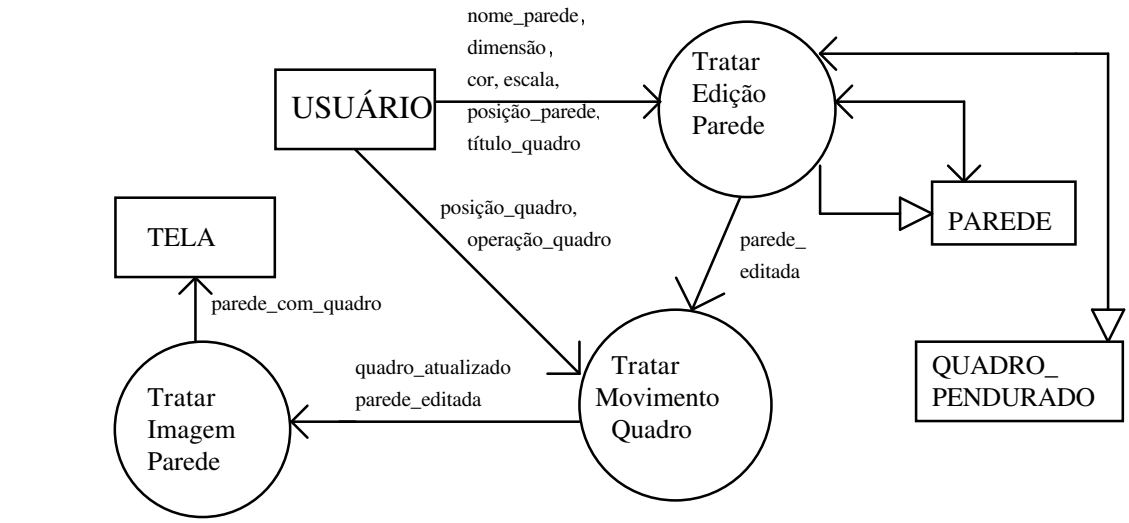


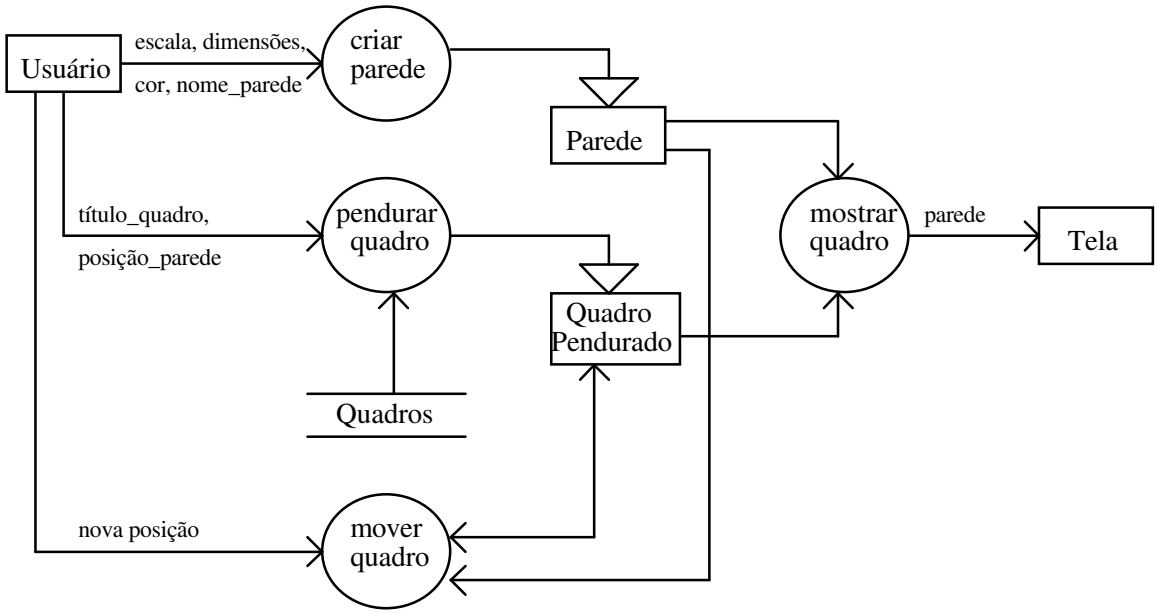
Diagrama de Estados da Classe Quadro



Anexo 3: O Modelo Funcional da Aplicação CAD_Parede

Modelo Funcional da Aplicação CAD_Parede





Descrição dos Processos do Modelo Funcional

Processo Tratar Edição Parede: este processo poderá criar uma parede nova através dos dados de entrada escala, dimensão e cor, carregar ou gravar com o nome-parede fornecido, ou criar um objeto na classe QUADRO_PENDURADO quando a operação for colocar.

Processo Tratar Movimentação Quadro: este processo verifica se o quadro selecionado poderá ser arrastado para a nova posição desejada pelo usuário, atualizando seu valor na classe QUADRO_PENDURADO ou eliminado o objeto desta classe se a operação for tirar.

Processo Tratar Imagem Parede: Este processo fará o "refresh" da parede com os quadros nas suas novas posições.