MODELAGEM DINÂMICA OO: UMA ANÁLISE COMPARATIVA DE TÉCNICAS

Guillermo Bustos Reinoso
Carlos A. Heuser

UFRGS/Instituto de Informática
Caixa Postal 15064
91501-970 Porto Alegre RS
e-mail: {gbustos|heuser}@inf.ufgrs.br

Abstract

In object-oriented systems modeling three perspectives are considered: the static or structural perspective, the dynamic perspective and the functional perspective. Modeling of the static perspective uses established concepts developed in the area of semantic data modeling. However, there are several different proposals for dynamic modeling of OO systems. This papers reviews some representative approaches and compares them by several criteria.

Keywords: OO modeling, OO analysis, dynamic Modeling

1 INTRODUÇÃO

A *Análise Orientada a Objetos* (AOO) é definida como a construção de modelos do domínio do problema, identificando e especificando um conjunto de objetos semânticos[1], que colaboram e comportam-se conforme os requisitos estabelecidos para o sistema de objetos [Monarchi& Puhr92]. A AOO é centrada na construção de modelos.

Nas diversas técnicas de modelagem orientada a objetos, dois aspectos ortogonais ou dimensões são para descrever um sistema: a dimensão *estrutural* dos objetos e a dimensão *dinâmica* do comportamento. A dimensão estrutural centra-se no aspecto estático ou passivo do sistema de objetos e relaciona-se com a estrutura estática dos mesmos. A dimensão dinâmica, por sua vez, tem a ver com o aspecto dinâmico ou ativo do sistema, descrevendo o comportamento do sistema de objetos.

Para modelar a dinâmica de um sistema de objetos, é necessário considerar também o aspecto estrutural. Na dimensão estrutural, são usados conceitos já estabelecidos como, a identidade dos objetos constituintes, sua classificação, seu encapsulamento (incluindo tanto atributos como operações), suas associações estáticas específicas e seus construtores de agregação estrutural (clusters ou semelhantes), muitos deles originários da modelagem semântica de dados.

Já na dimensão dinâmica, há diversas propostas na literatura, baseadas em diferentes conceitos. Este artigo revisa as técnicas de modelagem na dimensão dinâmica, fazendo uma comparação criteriosa entre as mesmas. Para isto, na seção 2 são apresentados os conceitos básicos da modelagem nesta dimensão. A seguir, são descritos os enfoques alternativos e as estratégias de abordagem para a construção dos modelos dinâmicos. Na seção 3, são estudadas brevemente várias técnicas de modelagem dinâmica usadas na metodologias de AOO. Exemplos e notações para cada caso também são apresentados. Finalmente, na seção 4, é feito e realizada a comparação entre as diferentes propostas.

[1] Estes objetos semânticos são aqueles cujo significado é determinado pelo domínio específico da aplicação.

2 Modelagem Dinâmica na Orientação a Objetos

A modelagem dinâmica OO e a modelagem dinâmica fora deste paradigma diferem no objeto a ser modelado. Fora da OO, modela-se o comportamento do sistema como um todo. Na OO, modela-se o comportamento dos objetos e, através da colaboração entre eles, o comportamento do sistema.

Assim, o comportamento de um sistema orientado a objetos está fundamentalmente determinado pela colaboração ou interação do comportamento dos objetos constituintes e, por sua vez, esta colaboração depende do conjunto de comportamentos individuais exibidos por estes mesmos objetos. Assim, o comportamento de um sistema pode ser reduzido, em uma primeira aproximação, ao comportamento individual de seus objetos.

2.1 Conceitos Básicos

O modelo básico de comportamento de um objeto é a máquina de estados finitos (MEF). Uma MEF é uma máquina hipotética que possui um conjunto finito de estados. Em qualquer instante do tempo, encontra-se em um único estado. A notação mais comum para representar uma MEF é um diagrama de transição de estados, que utiliza os conceitos básicos de estado, regra de transição (ou simplesmente transição), evento e ação.

Os *estados* correspondem a estágios dentro do ciclo de vida do objeto, e podem ser definidos como *passivos* ou *ativos* [deChampeaux93]. Um estado é ativo quando está determinado por um subconjunto de valores específicos do domínio dos atributos do objeto. No decorrer do tempo, nada muda no objeto enquanto estiver neste tipo de estado. Já um estado é dito *ativo* quando o objeto está envolvido em algum tipo de processo. Esta propriedade dos estados ativos torna mais difícil definir quando se "entra" ou "sai" deles[2]. No caso de um estado ativo, uma mudança do valor de um ou mais atributos pode determinar uma mudança de comportamento do objeto que possui tais atributos, o que necessariamente deve ser refletido como o passo de um estado a outro estado. As *transições* representam, neste contexto, mudanças nos valores dos atributos, implicando assim em mudanças de estados. Estas transições são produzidas como um reflexo a incidentes ou fatos acontecidos dentro ou fora do próprio objeto. Neste último caso, os fatos podem ser gerados por outros objetos ou ainda podem ser gerados no exterior do sistema. Estes fatos são conhecidos como *eventos*.

As *ações* são atividades ou operações que devem ser realizadas pelo objeto, tais como cálculos, geração de eventos para outro(s) objeto(s) do sistema ou agente(s) externo(s) ao mesmo, modificação de atributos ou de outros objetos. As ações podem ser definidas como ocorrendo durante uma transição (que corresponde ao modelo *Mealy* da MEF com estados passivos) ou durante a permanência em um estado (que corresponde ao modelo *Moore* da MEF com estados ativos) [Davis93].

Modelar o aspecto dinâmico de um sistema orientado a objetos implica em identificar os eventos que produzem as transições, as sequências dos eventos, os estados que definem o contexto para estes eventos, e a organização temporal destes eventos e estados. Assim, torna-se necessário transpor os limites da MEF dos objetos para refletir a interação ou colaboração entre eles, considerando o fluxo de eventos entre os objetos, para conseguir uma adequada representação do sistema. Isto mostra a importância de um conceito adicional para a modelagem nesta di-

---

[2] Este tipo de estado geralmente é expresso em linguagem natural por meio de verbos na forma de gerúndio (p. ex. *calculando*). É interessante destacar que é possível emular um estado ativo usando um estado passivo e uma transição sobre o mesmo estado (vide [deChampeaux93] para maiores detalhes).

mensão: a forma em que a interação entre os objetos é realizada, isto é, quais os objetos que participam, como os objetos agem na interação, e qual a natureza de tal interação. A modelagem dinâmica deve, portanto, tratar com dois níveis de abstração:

- um nível de *colaboração entre os objetos*, que lida com o controle no sistema, e a troca de mensagens e os protocolos de comunicação entre os objetos; e
- um nível de *comportamento do objeto*, que lida com os estados, transições, eventos e ações, todos internos a cada objeto.

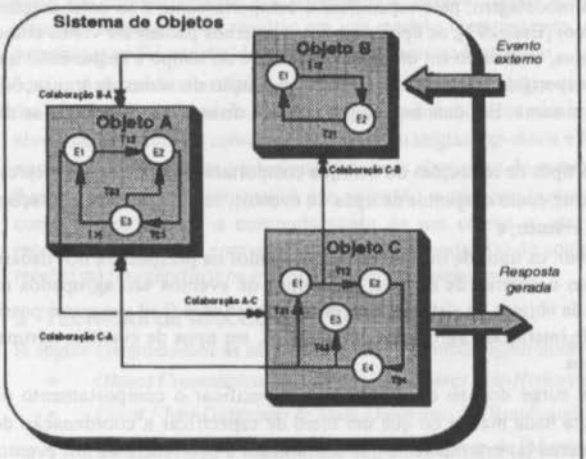

Figura 1 - Níveis de colaboração e de comportamento do objeto na modelagem dinâmica.

A figura 1 representa graficamente esta característica da modelagem dinâmica. O nível de comportamento do objeto diz relação com os estados e transições do objeto. Por exemplo, os estados *E1* e *E2*, representados por círculos no *Objeto B* (representado, por sua vez, pela caixa sombreada), e as transições *T12* e *T21* entre estes mesmos estados, representadas pelas setas internas. O nível de colaboração entre os objetos corresponde aos eventos que fluem entre os objetos (representados pelas setas com ponta dupla), como por exemplo as *colaborações B-A* e *C-B*, que podem originar-se a partir de um estado ativo (*E1* do *Objeto B*) no primeiro caso, ou de uma transição (*T12* do *Objeto C*) no último caso. Também neste nível interessam os eventos externos ao sistema e as respostas geradas para o exterior do sistema, como os mostrados na figura (representados por setas grossas), que afetam os *Objetos B* e *C*, respectivamente.

Finalmente, um conceito adicional é necessário para a modelagem dinâmica. Trata-se da hierarquização dos modelos através de mecanismos de particionamento e/ou agregação. Mecanismos desta natureza são úteis, porque fornecem abstrações dos modelos, permitindo ocultar detalhes irrelevantes nos distintos níveis dentro da hierarquia.

2.2 Os Enfoques Alternativos

Há três enfoques alternativos para realizar a modelagem dinâmica OO: comportamental (*behavioral*), dirigido por eventos (*event-driven*) e baseado em regras (*rule-based*).

No enfoque *comportamental*, a modelagem dinâmica é realizada a partir da identificação de interações típicas do sistema com os agentes externos, atores ou usuários (papéis de usuários na realidade). Isto define o comportamento externo que deve exibir o sistema como um todo. Considerando este comportamento global esperado do sistema, expresso em termos de um conjunto de tipos de interação, é possível identificar posteriormente os objetos e suas colaborações ao interior do sistema. Cada tipo de interação define um modo específico de uso do sistema,

uma sequência especial de transações relacionadas realizadas por um usuário e o sistema. O conjunto de tipos de interação deve especificar todos os modos existentes de uso do sistema. Diversas técnicas de análise usam este enfoque, entre elas estão as de [Gibson90] e [Rubin&Goldberg92], que utilizam *scripts* de atividades para as interações, e a proposta de [Jacobson93], que introduz o *use-case model* para este mesmo fim.

Uma outra forma de iniciar a modelagem dinâmica é através dos *eventos* específicos. O enfoque dirigido por eventos consiste na identificação de um conjunto de tipos de eventos, não necessariamente no início da modelagem, para especificar o comportamento e as colaborações entre os objetos. De acordo com [Essink91], os tipos de eventos externos podem ser vistos como *inputs* para o sistema de objetos, ocorrendo em um ponto específico do tempo e requerendo um "processamento de evento" específico que consiste em uma execução de séries de transações nos objetos constituintes do sistema. Em dois aspectos o enfoque de eventos diferencia-se do comportamental:

1. *nível de abstração*: os tipos de interações do enfoque comportamental podem ser expressos mais detalhadamente como conjuntos de tipos de eventos, sendo assim as interações mais abstratas que os eventos; e
2. *critério de agrupamento*: os tipos de interações são agrupados na perspectiva dos usuários ou atores que usam o sistema de objetos, já os tipos de eventos são agrupados na perspectiva dos tipos de objetos do sistema. Conforme o anterior, é fácil ver que é possível mapear os tipos de interações, agrupados por usuários, em tipos de eventos, agrupados por tipos de objetos.

Um terceiro enfoque surge do uso de *regras* para especificar o comportamento do sistema de objetos. Uma regra nada mais é do que um meio de especificar a coordenação de sinais (mensagens geradas interna ou externamente que comunicam a ocorrência de um evento) e operações, definindo assim as condições necessárias para a invocação das operações [Tsalgatidou& Loucopoulos91]. Na modelagem dinâmica, as regras assumem geralmente a seguinte sintaxe:

WHEN <sinal>
[**IF** <pré-condição>]
THEN <parte ação>

onde a parte **WHEN** é compulsória, implicando que quando o *<sinal>* é verdadeiro, a regra pode ser considerada. A parte opcional **IF** da *<pré-condição>*, quando verdadeira, indica que a parte do **THEN** pode ser executada. A *<parte ação>* corresponde à geração de um sinal (mensagem) composto por uma operação e parâmetros opcionais [D'Haenens91]. As regras podem ser válidas para o sistema como um todo, ou para classes, ou ainda para instâncias destas.

Algumas técnicas, ainda pouco desenvolvidas, que usam este enfoque são as de [D'Haenens91], que propõe um método de desenvolvimento usando regras, modelagem ER e orientação a objetos, e a proposta de [Tsalgatidou&Loucopoulos91] que usa o conceito de *fatos* para a modelagem estática e regras representadas graficamente com redes de Petri para o aspecto dinâmico.

2.3 Alternativas para o processo de modelagem

Como em outras áreas da engenharia de software há basicamente três alternativas para executar o processo de modelagem: *top-down*, *bottom-up* e *middle-out*.

Entende-se como estratégia *top-down* à construção de modelos dinâmicos através de um refinamento sucessivo, em que cada refinamento acrescenta novos detalhes ao modelo que eram

abstratos no refinamento anterior. Em OO, parte-se da modelagem da colaboração entre objetos, para a modelagem do comportamento dos objetos. Este processo resulta geralmente, porém não sempre, em um modelo hierarquizado, em que cada nível ou camada, no sentido do topo ao fundo da hierarquia, é mais detalhado, e portanto, menos abstrato.

De forma complementar, uma estratégia *bottom-up* é a construção de modelos dinâmicos através de um agregação sucessiva, em que cada agregação acrescenta maior abstração ao modelo, escondendo detalhes que eram considerados na agregação anterior. Em OO, isso significa partir do comportamento dos objetos para, depois, passar a colaboração entre objetos. Este processo também pode resultar em um modelo hierarquizado, de tal forma que é impossível determinar se foi construído de forma *top-down* ou *bottom-up*.

A terceira alternativa é a *middle-out*. Esta estratégia consiste na identificação "linear" dos elementos do modelo, i.e. a construção dos modelos não segue o sentido vertical entre os níveis da hierarquia, (como acontece nas estratégias *top-down* e *bottom-up*), senão o sentido horizontal, acrescentando e relacionando novos elementos do mesmo nível de abstração e detalhe. Assim, o modelo é construído por extensão e não por hierarquização. A modelagem pode começar por definir o comportamento de um objeto e, através dos eventos gerados (ou recebidos), continuar com a construção do(s) modelo(s) de comportamento do(s) objeto(s) que recebe(m) (ou gera(m)) os eventos, e assim sucessivamente.

3 Técnicas de Modelagem Dinâmica

A seguir consideramos as seguintes técnicas de modelagem dinâmica:

- *Object Communication Diagram & Object Life-History Diagrams* de [Yourdon94],
- *Event Flow Diagrams & State Diagrams* de [Rumbaugh91],
- *State Transition Diagrams & Event Diagram* de [Martin&Odell93],
- *State Model & Object Communication Model* de [Shlaer&Mellor92],
- *State Nets & Object-Interaction Model* de [Embley92], e
- *Object/Behavior Diagrams* de [Kappel&Schrefl91].

Como exemplo específico para mostrar as diferentes representações[3] sugeridas pelas técnicas é usada uma porção do problema de organização de conferências da IFIP [Olle82]. É considerado o *artigo* como uma classe, cujos objetos apresentam o seguinte conjunto mínimo de estados: *{recebido, em avaliação, aceito, rejeitado}*. Adicionalmente, é usada a interação dos objetos desta classe com os de outras classes, como por exemplo *autores* e *avaliadores* (*referees*) ou agentes externos ao sistema, como por exemplo o *comitê de programa*.

3.1 *Object Communication Diagram & Object Life-History Diagrams*

A proposta de [Yourdon94] corresponde a uma evolução da proposta de [Coad&Yourdon92]. Nela são abordados ambos os níveis da modelagem dinâmica: para o nível de colaboração entre os objetos usa o *object communication diagram*, e para o nível de comportamento do objeto usa os *object life-history diagrams*. O enfoque utilizado é dirigido por eventos. A ordem na abordagem sugere que esta é uma técnica de estratégia *bottom-up*.

No *object communication diagram* são especificadas as comunicações (conexões de mensagens) entre os objetos e as classes. Isto é realizado após a modelagem dos estados e operações de cada objeto. A seqüência das mensagens pode ser indicada acrescentando numeração

[3] As notações utilizadas em cada técnica revista são tão fieis quanto possível às propostas pelos respectivos autores.

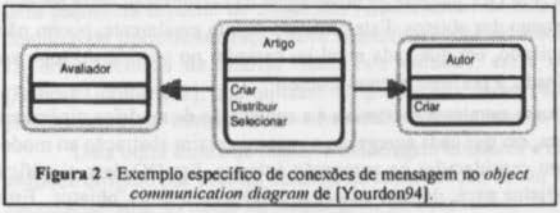

Figura 2 - Exemplo específico de conexões de mensagem no *object communication diagram* de [Yourdon94].

às mensagens. A figura 2 mostra exemplos de conexões de mensagem entre três objetos das classes *Artigo*, *Autor* e *Avaliador* (representadas pelas caixas de cantos arredondados). A mensagem da direita origina-se em uma instância da classe *Artigo* (indicado pela seta que sai da borda externa da classe que representa justamente as instâncias daquela classe), e tem como destino a classe *Autor* (indicado pela seta que chega a borda interior que representa a classe). A conexão da esquerda é entre instâncias.

Figura 3 - Exemplo de um *object life-history diagram* de [Yourdon94], para o objeto *Artigo*.

Para o caso dos *object life-history diagrams*, o procedimento da técnica consiste essencialmente na identificação de estados (passivos) dos objetos nas diferentes classes. Para isto, os autores indicam que deve examinar-se os valores potenciais dos atributos para determinar se implicam mudanças no comportamento do objeto. Os *object life-history diagrams* resultantes (que correspondem a uma MEF do modelo Mealy) utilizam representações para estados (caixas com nome), transições (setas que indicam o sentido da mudança de estado), condições (anotadas em cada transição acima da linha horizontal) e ações (anotadas sob a linha horizontal). A figura 3 mostra a representação resultante para o exemplo padrão.

3.2 *Event Flow Diagrams & State Diagrams*

Na proposta de [Rumbaugh91], a modelagem dinâmica é tratada nos dois níveis: o nível de colaboração entre os objetos usando os *event flow diagrams*, e o nível de comportamento do objeto usando os *state diagrams*. Ambas as ferramentas fazem parte do *dynamic model*. Esta é uma técnica que adota um enfoque dirigido por eventos e uma estratégia *top-down*.

A modelagem do aspecto dinâmico é realizada em duas fases: na primeira são considerados cenários de interação entre objetos, esta interação é mapeada nos *event flow diagrams*; e na segunda são construídos *state diagrams* para cada objeto de comportamento não-trivial, considerando a troca de eventos definida previamente.

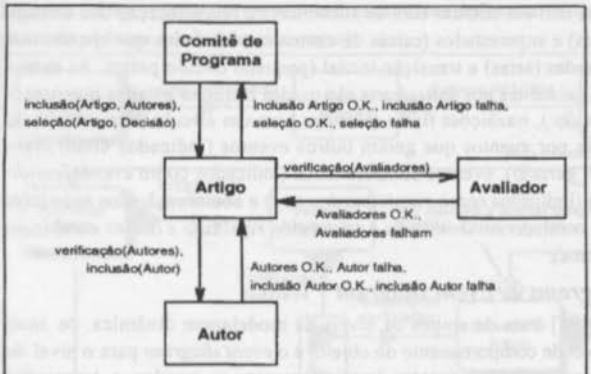

Figura 4 - Exemplo específico de um *event flow diagram* de [Rumbaugh91].

Os *event flow diagrams* permitem representar a interação entre os objetos em termos de eventos, a partir de cenários de interação típica entre os objetos e entre os objetos e os agentes externos.

Os *event flow diagrams* não indicam seqüência, apenas indicam a emissão e recepção de todos os eventos possíveis entre os objetos, e entre estes e os agentes externos. Os *event flow diagrams* utilizam caixas para representar os objetos e os agentes externos (não são diferenciados), e utilizam setas para representar o fluxo dos eventos. A figura 4 mostra a representação resultante para a interação dos objetos *Artigo*, *Autor*, *Avaliador* e o agente externo *Comitê de Programa*. Na figura, os eventos (que também incluem erros) podem ser indicados com parâmetros, assim por exemplo o evento *inclusão(Autor)*, entre as classes *Artigo* e *Autor*, tem como parâmetro os valores dos atributos para um novo objeto da classe *Autor*.

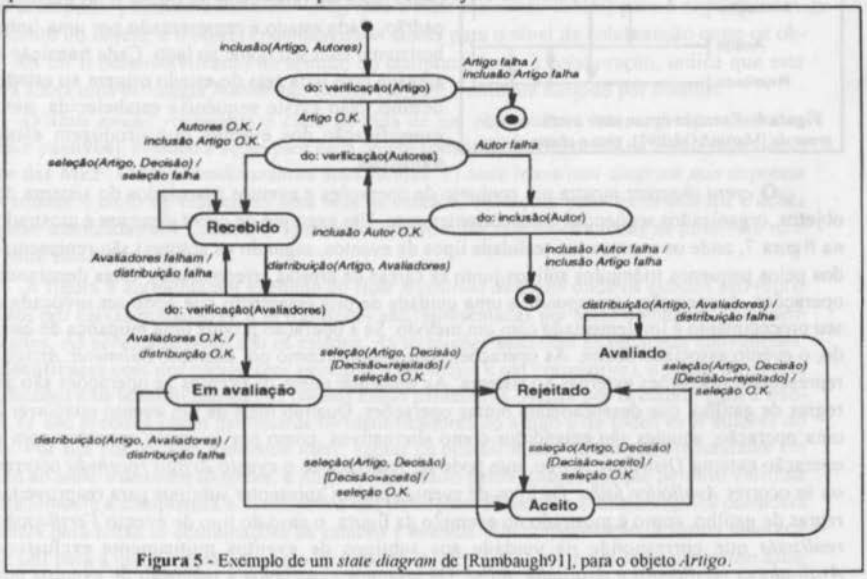

Figura 5 - Exemplo de um *state diagram* de [Rumbaugh91], para o objeto *Artigo*.

Por sua vez, os *state diagrams* propostos correspondem a uma extensão (combinando os modelos Mealy e Moore das MEF) dos *statecharts* de [Harel87]. Como estes últimos permitem concorrência, esta é possível ao interior de um objeto. A figura 5 mostra um *state diagram* para o exemplo padrão. Nele pode observar-se algumas das propriedades e extensões em relação aos

statecharts. Os *state diagrams* têm em comum com os *statecharts* a representação dos estados (caixas de cantos arredondados) e superestados (caixas de cantos arredondados que contêm outros estados), transições nomeadas (setas) e transição inicial (pequeno círculo preto). As extensões e notações especiais não existentes nos *statecharts* são usadas naqueles estados que possuem atividades (indicados com *do:*), transições finais (círculos com um círculo preto menor no interior), transições disparadas por eventos que geram outros eventos (indicados como *evento_que_dispara/evento_que_é_gerado*), eventos condicionados (indicados como *evento[condição]*), eventos parametrizados (indicados como *evento(parâmetro)*) e combinação dos anteriores (na figura as transições entre o estado *em avaliação* e os estados *rejeitado* e *aceito* combinam todas as notações para os eventos).

3.3 *State Transition Diagrams & Event Diagram*

A proposta de [Martin&Odell93] trata de ambos os níveis da modelagem dinâmica: os *state transition diagrams* para o nível de comportamento do objeto, e o *event diagram* para o nível de colaboração. Contudo, o *event diagram* contém implicitamente os estados e transições mostrados no *state transition diagram*, assim este último é utilizado mais como um complemento de documentação do que como uma ferramenta em si mesma. Esta técnica trabalha por excelência sob o enfoque dirigido por eventos e opta por uma estratégia *middle-out*.

Os *state transition diagrams* mostram os estados (passivos) dos objetos e todas as possíveis transições que podem ocorrer entre eles. Utilizam um esquema extremamente simples como pode ser observado na figura 6 do exemplo padrão. Cada estado é representado por uma linha horizontal com seu nome ao lado. Cada transição é anotada com uma seta do estado origem ao estado destino. Não existe seqüência estabelecida, nem especificação dos eventos que produzem estas transições.

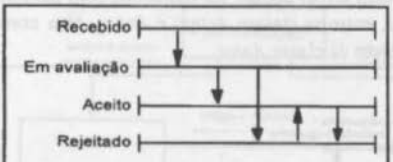

Figura 6 - Exemplo de um *state transition diagram* de [Martin&Odell93], para o objeto *Artigo*.

O *event diagram* mostra um conjunto de operações e eventos associados do sistema de objetos, organizados seqüencial e concorrentemente. Um exemplo de *event diagram* é mostrado na figura 7, onde os eventos (na realidade tipos de eventos, segundo os autores) são representados pelos pequenos triângulos sólidos junto às caixas de arestas arredondadas, estas denotando operações. A operação corresponde a uma unidade de processamento que pode ser invocada e seu procedimento é implementado com um método. Se a operação produz uma mudança de estado, o evento associado ocorre. As operações com sombra (como por exemplo *Submeter Artigo*) representam operações externas ao sistema. As setas que unem os eventos às operações são as regras de gatilho, que desencadeiam outras operações. Quando mais de um evento concorrer a uma operação, aqueles são entendidos como alternativos, como por exemplo acontece com a operação externa *Distribuir Artigo*, que pode ser realizada se o evento *Artigo recebido* ocorrer *ou* se ocorrer *Avaliador falha*. Os tipos de eventos podem apresentar subtipos para reaproveitar regras de gatilho, como é mostrado no exemplo da figura: o caso do tipo de evento *Verificação realizada* que corresponde na verdade aos subtipos de eventos mutuamente exclusivos *Avaliadores verificados* e *Avaliador falha*. Os losangos associados à recepção de eventos por parte de uma operação (como acontece com a operação *Selecionar Artigo* e os eventos *rejeitado* e *aceito* do exemplo da figura), indicam condições de controle (neste caso anotadas diretamente

no diagrama da figura) que devem ser verificadas antes de invocar a operação (no exemplo: só o artigo aceito pode ser rejeitado e vice-versa).

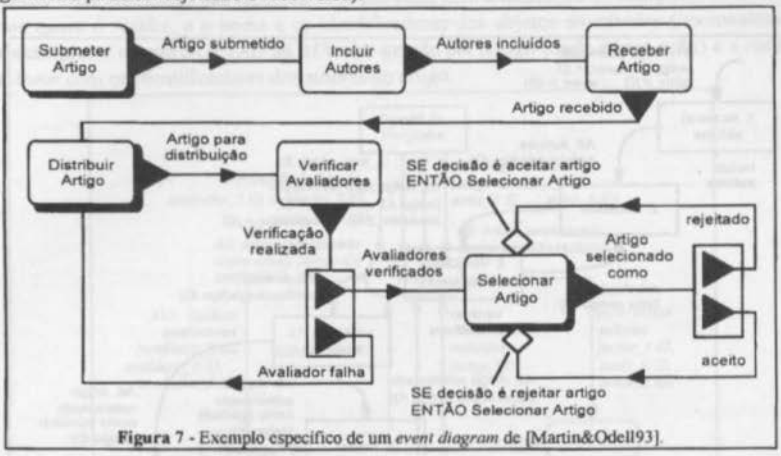

Figura 7 - Exemplo específico de um *event diagram* de [Martin&Odell93].

3.4 *State Model & Object Communication Model*

A proposta de [Shlaer&Mellor92] para a modelagem dinâmica consiste em dois modelos: o *state model* (composto pelos *state transition diagrams* e *state transition tables*) para o nível de comportamento do objeto, e o *object communication model* para o nível de colaboração entre os objetos. Ao ser o desenvolvimento no sentido do comportamento à colaboração, indica que esta técnica adota uma estratégia *bottom-up*. Utiliza também o enfoque dirigido por eventos.

O *state model* representa o ciclo de vida de um objeto. Para isto, representa estados (ativos e passivos), eventos e ações para cada objeto constituinte do sistema na forma do modelo Moore das MEF. O *state model* assume duas formas: 1) *state transition diagram* que expressa graficamente o ciclo de vida como uma rede de estados, transições entre estes estados e ações que estão associadas aos estados, e 2) *state transition table* que mostra todas as possíveis relações entre estados e eventos.

A figura 8 apresenta um exemplo de *state transition diagram* onde os estados são representados por caixas numeradas, e as transições são representadas por setas nomeadas que unem os estados. As ações aparecem sob os estados. As transições assumem as seguintes convenções: são identificadas com um código (por exemplo *A* de *Artigo* e um correlativo), o nome do evento (significado) e os identificadores dos objetos como parâmetros. Assim por exemplo, para a transição *A1* são precisos como parâmetros os identificadores do artigo e de todos os *n* autores do artigo. Por sua vez, o *state transition table*, apesar de possuir muita informação redundante em relação ao *state transition diagram*, é julgado necessário pelos autores porque permite verificar mais facilmente a completeza e consistência das transições. Preencher a tabela implica considera os efeitos para todas as combinações de estados e eventos. Em compensação, a forma tabular é pouco útil para a compreensão do ciclo de vida do objeto. A tabela 1 é um *state transition table* que representa a situação da figura 8. As filas mostram os estados e as colunas os eventos. As entradas da tabela são preenchidas conforme as seguintes alternativas: 1) número de estado a que conduz a transição da coluna a partir do estado da fila; 2) indicação de *não pode ocorrer*, se esta combinação estado-evento não pode ocorrer na realidade do sistema, adicionalmente é

recomendada incluir uma referência para uma nota explicativa; e 3) indicação de *evento ignorado*, quando o objeto não reage à recepção do evento.

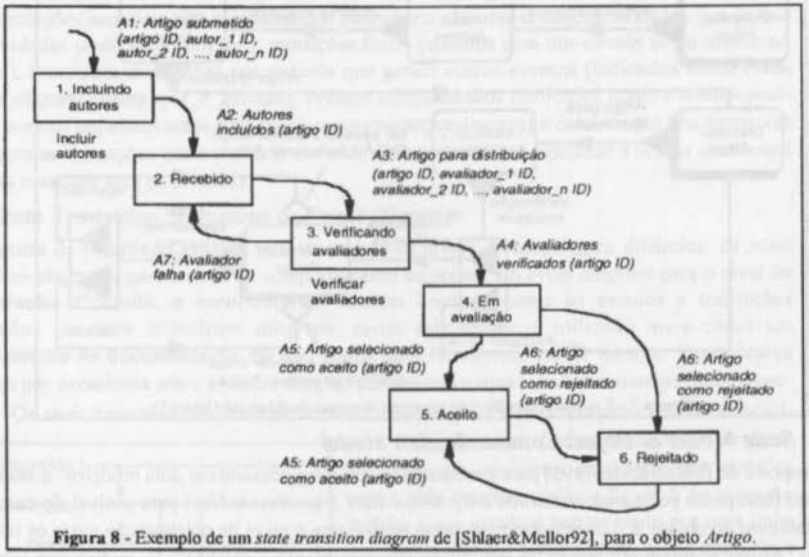

Figura 8 - Exemplo de um *state transition diagram* de [Shlaer&Mellor92], para o objeto *Artigo*.

	A1: Artigo submetido	A2: Autores incluídos	A3: Artigo para distribuição	A4: Avaliadores verificados	A5: Artigo selecionado como aceito	A6: Artigo selecionado como rejeitado	A7: Avaliador falha
1. Incluindo autores	não pode ocorrer [1]	2	não pode ocorrer [8]	não pode ocorrer [8]	não pode ocorrer [8]	não pode ocorrer [8]	não pode ocorrer [8]
2. Recebido	não pode ocorrer [1]	evento ignorado	3	não pode ocorrer [4]	não pode ocorrer [6]	não pode ocorrer [6]	não pode ocorrer [4]
3. Verificando avaliadores	não pode ocorrer [1]	não pode ocorrer [2]	não pode ocorrer [3]	4	não pode ocorrer [6]	não pode ocorrer [6]	2
4. Em avaliação	não pode ocorrer [1]	não pode ocorrer [2]	não pode ocorrer [3]	evento ignorado	5	6	não pode ocorrer [7]
5. Aceito	não pode ocorrer [1]	não pode ocorrer [2]	não pode ocorrer [3]	não pode ocorrer [5]	evento ignorado	6	não pode ocorrer [5]
6. Rejeitado	não pode ocorrer [1]	não pode ocorrer [2]	não pode ocorrer [3]	não pode ocorrer [5]	5	evento ignorado	não pode ocorrer [5]

Notas:
[1] Artigo já foi submetido
[2] Artigo já foi recebido
[3] Artigo já foi para distribuição
[4] Artigo não foi para distribuição
[5] Artigo já foi avaliado
[6] Artigo não foi avaliado
[7] Artigo já foi distribuído
[8] Artigo não foi recebido

Tabela 1 - Exemplo de um *state transition table* de [Shlaer&Mellor92], para o objeto *Artigo*.

O *object communication model* expressa a comunicação de eventos entre os *state models* e os agentes externos ao sistema. Para isto, é necessário ter modelado previamente os comportamentos independentes dos objetos em termos dos *state transition diagrams* e *state transition tables*. Assim, eventos recebidos e eventos gerados (ações) em um *state model* aparecem como entradas e saidas nos objetos do *object communication model*. A figura 9 mostra um *object communication model* em que os *state models* são representados por caixas de cantos arredondados (por exemplo, *Artigo* representa o *state model* da figura 8 e da tabela 1). Cada agente externo é

representado por uma caixa simples (como *Comitê de Programa* na figura). Os eventos que fluem entre estes elementos são representados por setas com a indicação do código do evento, conforme quem o recebe, e o nome e os identificadores dos objetos envolvidos (opcionalmente). Por exemplo, o evento *AU1* (*AU* de *AUtor*) é gerado por *Artigo* (onde é uma ação) e é recebido por *Autor* com os identificadores dos autores do artigo.

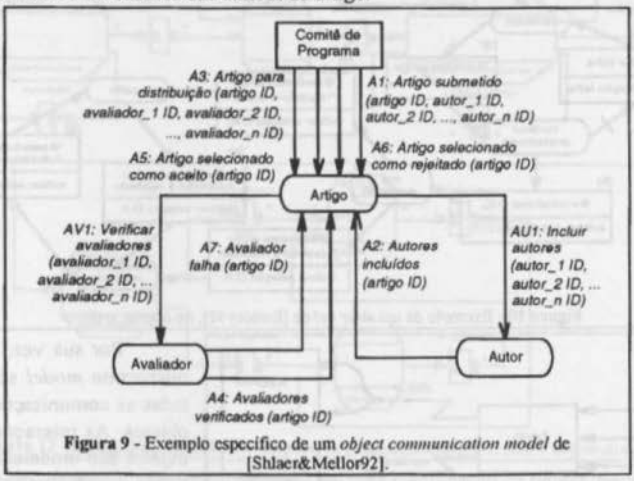

Figura 9 - Exemplo específico de um *object communication model* de [Shlaer&Mellor92].

3.5 *State Nets & Object-Interaction Model*

A proposta de [Embley92] contempla ambos os níveis da modelagem dinâmica: os *state nets* (que fazem parte do *object-behavior model*) para o nível de comportamento do objeto, e o *object-interaction model* para o nível de colaboração entre os objetos. Esta técnica assume um enfoque dirigido por eventos e uma estratégia *bottom-up*.

Os *state nets* modelam o comportamento de cada objeto em forma independente e assumindo o modelo Mealy das MEF. Estas redes são construídas a partir da identificação dos estados (ativos e passivos), e a definição das transições e as exceções. As transições nos *state nets* possuem um identificador, um *trigger*[4] e uma ação. Podem existir transições iniciais e finais em um *state net*. Os estados podem ser múltiplos, tanto anteriores a uma transição quanto posteriores a ela, neste último caso pode ou não existir opção de escolha entre os estados. Existe também a possibilidade de uma transição ser disparada produzindo uma seqüência paralela de ativação de estados e transições, permitindo assim a concorrência ao interior do objeto. A figura 10 mostra um *state net* possível para o objeto *Artigo*. Esta rede representa os estados como caixas de cantos arredondados e as transições como caixas bipartidas. O sentido da transição é indicado por setas. No alto da transição existe um identificador numérico único dentro do *state net*. Cada transição é definida por um *trigger* na metade superior e uma ou mais ações na metade inferior. A parte do *trigger* que é precedida por um @ corresponde a um evento. Assim, no caso da transição de número *[6]* lê-se "quando se seleciona artigo e a decisão é aceito, se deve indicar seleção O.K.", gerando a mudança do estado *em avaliação* para o estado *aceito*. As setas que possuem um traço perpendicular representam um *path* para uma transição que modela um com-

[4] O *trigger* pode ser baseado em eventos, mas não necessariamente.

portamento de exceção, geralmente indicando situações de erro, como por exemplo quando um avaliador falha na verificação para avaliar um artigo (transição de número *[4]*).

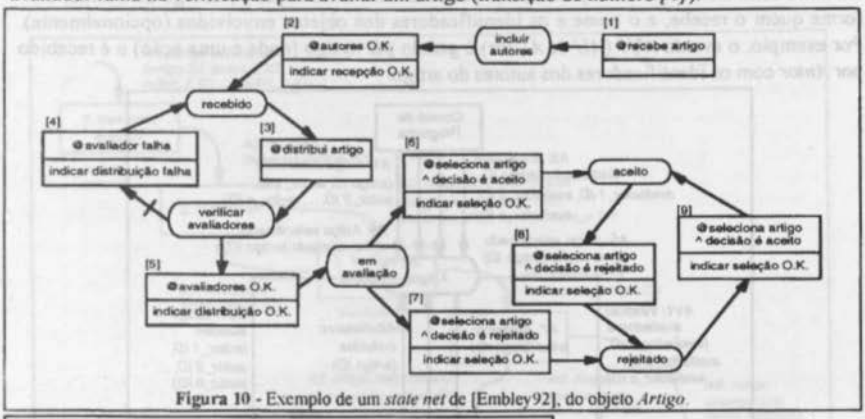

Figura 10 - Exemplo de um *state net* de [Embley92], do objeto *Artigo*.

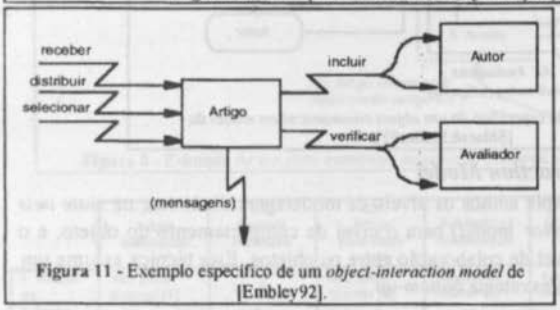

Figura 11 - Exemplo específico de um *object-interaction model* de [Embley92].

Por sua vez, no *object-interaction model* são apresentadas as comunicações entre os objetos. As interações entre os objetos são modeladas após definido o comportamento destes objetos usando os *state nets*. As interações podem ser através de atividades (ações) ou listas de objetos, podem ser sincronas ou assincronas (usando repositórios neste último caso), múltiplas (várias origens e/ou vários destinos), ter objetos de origem e/ou destino especificados, ser bidirecionais ou ser contínuas, entre outras propriedades. A figura 11 mostra um exemplo de interação entre os objetos (representada pela setas quebradas) dos objetos *Artigo*, *Autor* e *Avaliador* (representados pelas caixas). A comunicação *incluir*, entre *Artigo* e *Autor*, e a comunicação *verificar*, entre *Artigo* e *Avaliador*, são múltiplas, i.e. vários objetos destinos recebem a mensagem (no exemplo, vários *autores* de um *artigo* e vários *avaliadores* do *artigo*). As mensagens *receber*, *distribuir* e *selecionar* são geradas por agentes externos anônimos. O destino da lista de objetos *(mensagens)* também é externo ao modelo.

3.6 *Object/Behavior Diagrams*

A proposta de [Kappel&Schrefl91] trata dos niveis da modelagem dinâmica da seguinte maneira: usa os *behavior diagrams* para o nivel de comportamento do objeto, e uma variação destes, os *activity specification diagrams*, para o nível de colaboração entre os objetos. O enfoque dirigido por eventos é o adotado por esta técnica. Apesar de utilizar refinamento sucessivo para o comportamento dos objetos (o que implicaria, pelo menos parcialmente, uma estratégia *top-down*) e derivar as colaborações após a definição do comportamento (o que indicaria uma estratégia *bottom-up*), esta proposta constrói o modelo dinâmico no sentido horizontal no nível das colaborações, o que implica uma estratégia *middle-out*. Contudo, esta

proposta está mais centrada na diagramação de sistemas orientados a objetos e não apresenta, portanto, procedimentos explícitos para a modelagem.

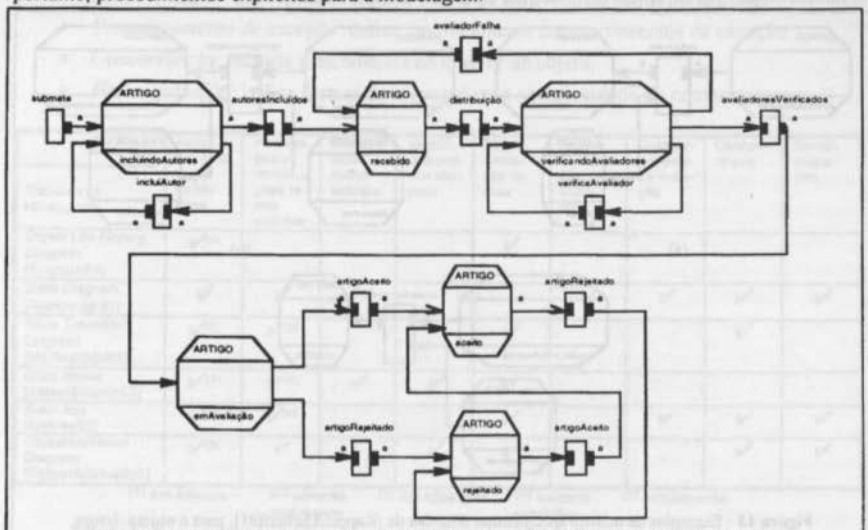

Figura 12 - Exemplo de um *behavior diagram* de [Kappel&Schrefl91], para o objeto *Artigo*.

Os *object/behavior diagrams* consistem em uma combinação de *object diagrams*, baseados nos conceitos da modelagem semântica de dados, que descrevem as propriedades estruturais dos objetos, e os *behavior diagrams*, que baseiam-se nas redes de Petri [Heuser90]. Os *behavior diagrams* são utilizados para representar o comportamento de objetos, modelando o ciclo de vida das instâncias de um tipo de objeto. Um *behavior diagram* consiste em um conjunto de estados (ativos e passivos) do objeto e um conjunto de atividades. Os estados são representados pelas caixas maiores[5], com o nome do objeto na parte superior e o nome do estado na parte inferior; e as atividades são representadas pelas caixas menores que unem os estados com setas (vide figura 12). Cada atividade possui *input* e *output ports*, por onde fluem *tokens* (denotados por *a* para o caso de *Artigo* da figura 12) associados às instâncias do tipo do objeto. Estes *ports* aparecem como pequenas caixas à esquerda da atividade na borda interna (*input*), e à direita da mesma na borda externa (*output*).

É através das atividades com *ports* que é representada a interação dos objetos. Assim, os *activity specification diagrams* mostram as atividades (que nos *behavior diagrams* aparecem apenas dando seqüência aos estados no ciclo de vida do objeto), como relacionadas com outros ciclos de vida. Estes *activity specification diagrams* correspondem às operações dos objetos. A operação é definida para as instâncias do tipo do objeto do *primary port*, que liga os pré e pós-estados (pré e pós-condições) do objeto. Os restantes *inputs ports* representam parâmetros formais da operação e seus pré-estados podem ser interpretados como pré-condições destes parâmetros. O valor de retorno da operação é representado por um *return port* e seus pós-

[5] Esta notação para os objctos é derivada dos *object diagrams* onde é usada a composição gráfica para representar alguns relacionamentos estáticos (vide [Kappel&Schrefl91]).

estados podem ser interpretados como pós-condições que o valor retornado deve satisfazer. Os restantes *output ports* representam efeitos colaterais da operação.

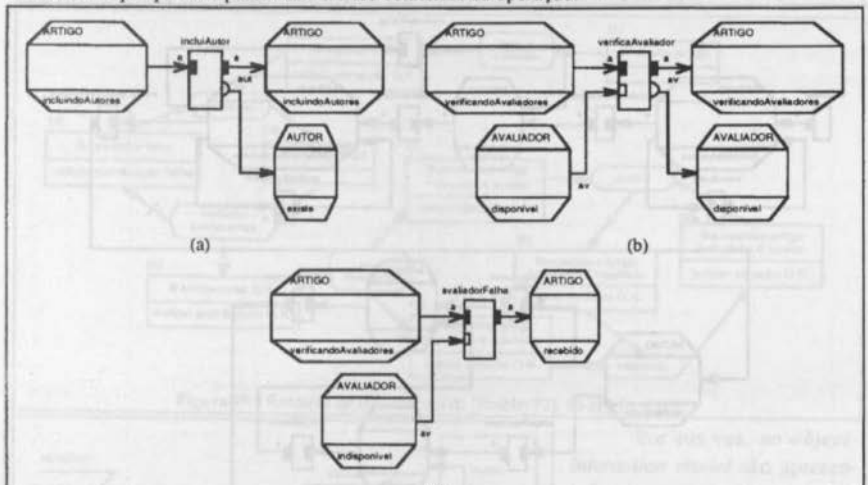

Figura 13 - Exemplos de *activity specification diagram* de [Kappel&Schrefl91], para o objeto *Artigo*: (a) incluiAutor, (b) verificaAvaliador, e (c) avaliadorFalha.

As notações para todos estes *ports* podem ser observadas na figura 13. Nela aparecem as três relações entre o objeto *Artigo* com os objetos *Autor* e *Avaliador*. No *activity specification diagram* da figura 13(a), os *primary ports* são representados por *ports* pretos e o *return port* é representado pelo pequeno semicírculo. Este diagrama deve ser interpretado como a inclusão de novos autores enquanto o artigo está no estado *incluindoAutores*. No caso do *activity specification diagram* da figura 13(b), o *input port* para o pré-estado de *Avaliador* é representado pela caixinha branca e as linhas descontinuas indicam que ocorre apenas uma "leitura". Este diagrama corresponde à situação em que um *avaliador* é encarregado de relatar um *artigo*. O caso da figura 13(c) acontece quando o *avaliador* correspondente não está disponivel e portanto falha na sua tentativa de avaliação. Em todos estes *activity specification diagrams* existe um *token* para cada tipo de objeto que flui através da atividade: *a* para *Artigo*, *aut* para *Autor*, e *av* para *Avaliador*.

4 Análise Comparativa das Técnicas

Esta seção analisa de forma comparativa as técnicas apresentadas segundo critérios listados a seguir. A tabela 2 compara as técnicas no que refere a modelagem do comportamento do objeto. Já a tabela 3 considera a modelagem da colaboração entre os objetos.

Na tabela 2 os critérios utilizados para a comparação das técnicas de modelagem são os seguintes:

- *Ações em estados e transições*: define ações tanto em estados quanto em transições.
- *Parâmetros e condições nos eventos*: define eventos parametrizados e sujeitos a condições adicionais para ativar as transições.
- *Eventos sobre outros estados*: verifica os efeitos da ocorrência dos eventos em todos os estados possíveis.

- *Identificadores dos objetos*: indica se é necessário especificar identificadores.
- *Representação*: usa representação na forma de uma rede ou estruturadamente.
- *Comportamento de exceção*: define explicitamente comportamentos de exceção.
- *Concorrência*: permite concorrência ao interior do objeto.
- *Hierarquização*: indica formas de hierarquizar a complexidade do comportamento.

CRITÉRIOS / TÉCNICAS DE MODELAGEM	Ações em estados e transições	Parâmetros e condições nos eventos	Eventos sobre outros estados	Identificadores dos objetos	Representação de rede	Representação estruturada	Comportamento de exceção	Concorrência	Hierarquização
Object Life-History Diagram [Yourdon94]	✔(3)				✔				
State Diagram [Rumbaugh91]	✔	✔	✔(5)			✔	✔	✔	✔
State Transition Diagram [Martin&Odell93]	✔(3)	✔(4)			✔			✔	
State Model [Shlaer&Mellor92]	✔(1)	✔(2)	✔	✔	✔				
State Net [Embley92]	✔(3)	✔(4)			✔		✔	✔	✔
Object/Behavior Diagram [Kappel&Schrefl91]	✔(3)	✔		✔	✔			✔	✔

(1) em estados (2) somente parâmetros (3) em transições (4) somente condições (5) parcialmente

Tabela 2 - Comparativo de técnicas para o nível de comportamento do objeto na modelagem dinâmica.

CRITÉRIOS / TÉCNICAS DE MODELAGEM	Comunicação classe-instância	Mensagens assíncronas	Múltiplos origens e destinos	Qualificação das mensagens	Seqüenciação das mensagens	Mensagens de exceção	Identificadores de objetos	Agentes externos	Hierarquização
Object Communication Diagram [Yourdon94]	✔		✔		✔				✔(1)
Event Flow Diagram [Rumbaugh91]						✔		✔	
Event Diagram [Martin&Odell93]			✔		✔			✔	✔
Object Communication Model [Shlaer&Mellor92]			✔	✔			✔	✔	✔(1)
Object-Interaction Model [Embley92]	✔	✔	✔	✔		✔		✔	✔
Object/Behavior Diagram [Kappel&Schrefl91]		✔	✔(2)	✔			✔		

(1) derivada da modelagem estática (2) pelos *tokens*

Tabela 3 - Comparativo de técnicas para o nível de colaboração entre os objetos na modelagem dinâmica.

Na tabela 3, os critérios utilizados para as técnicas de modelagem são:

- *Comunicação classe-instância*: permite comunicação entre classes, entre classes e instâncias, e entre instâncias.
- *Mensagens assíncronas*: é possível ter mensagens assíncronas entre os objetos.
- *Múltiplos origens e destinos*: mensagens geradas e/ou recebidas por múltiplas instâncias.
- *Qualificação das mensagens*: especifica objetos origens e/ou destinos.
- *Seqüenciação das mensagens*: dá uma seqüência para a emissão das mensagens.

- *Mensagens de exceção*: define explicitamente mensagens de exceção.
- *Identificadores dos objetos*: para aqueles envolvidos na mensagem.
- *Agentes externos*: permite interação explícita com agentes externos ao sistema.
- *Hierarquização*: hierarquiza a complexidade da colaboração entre os objetos.

4 Conclusões e Comentários

Foi apresentada uma visão geral da modelagem dinâmica OO, através da definição dos conceitos básicos, os enfoques alternativos e as estratégias de abordagem. As técnicas mais representativas para modelar o aspecto dinâmico sob a orientação a objetos foram brevemente descritas e contrastadas.

O enfoque dirigido por eventos é usado mais frequentemente, pela simplicidade de associar os eventos aos objetos. O enfoque comportamental poderia ser adotado como um passo prévio à identificação dos eventos específicos para os objetos. Em relação ao processo de modelagem, não foram identificadas alternativas preferidas.

O estado da arte descrito pelas técnicas, enfoques e estratégias, mostra a relativa imaturidade da modelagem dinâmica na orientação a objetos: as propostas para modelar o aspecto dinâmico tem sido todas adaptadas de fora do paradigma, trazendo com isto algumas dificuldades. A maior delas surge da integração dos aspectos estáticos e dinâmicos, tanto fora como dentro do paradigma da orientação a objetos. Existe uma tendência, devido principalmente à grande complexidade que podem atingir os sistemas de objetos, a tratar os aspectos estáticos e dinâmicos com relativa independência. As técnicas vistas tratam este problema dividindo: fornecem ferramentas específicas para a dimensão dinâmica e outras para a dimensão estática

Observando as representações das diferentes técnicas, vê-se que não existe maior diversidade de modelos e notações: são todas baseadas em redes de transição de estados (a partir da concepção das MEF) com variações menores, sendo a maior delas a apresentada com os *statecharts*. A única exceção corresponde às redes de Petri.

Considerando os critérios gerais usados nas tabelas comparativas, pode indicar-se a técnica dos *state diagrams* de [Rumbaugh91] como a mais completa no nível de comportamento do objeto (a extensão aos *statecharts* explica parcialmente esta completeza). Quanto ao nível de colaboração entre os objetos, a técnica do *object-interaction model* de [Embley92] é a mais completa, já que fornece um poderoso conjunto de elementos de modelagem e notações especiais. Considerando os escassos critérios de comparação e aplicação utilizados, apenas é possível indicar a proposta mais completa para os diferentes níveis da modelagem dinâmica, o que não necessariamente a torna a "melhor" técnica.

Finalmente, algumas linhas de pesquisas que podem ser visualizadas a partir deste trabalho são as seguintes:

- técnicas de modelagem especificamente para a orientação a objetos,
- mecanismos para o desenvolvimento por refinamento sucessivo dos modelos dinâmicos, tanto para o nível de colaboração quanto o de comportamento,
- técnicas integralmente dinâmicas orientadas a objetos, que incluam os aspectos estáticos como casos particulares dos aspectos dinâmicos, para resolver alguns dos problemas de integração existentes até hoje, e
- introduzir em forma complementar, representações tabelares e/ou baseadas em regras para facilitar a consistência e a completeza dos modelos dinâmicos.

Referências Bibliográficas

[Coad&Yourdon92] Coad, P. & E. Yourdon. *Análise Baseada em Objetos*. Rio de Janeiro: Campus. 1992.

[Davis93] Davis, A. M. *Software Requirements: Objects, Functions, & States*. Englewood Cliffs: Prentice-Hall, 1993.

[deChampeaux93] de Champeaux, D.; D. Lea & P. Faure. *Object-Oriented System Development*. Reading: Addison-Wesley, 1993.

[D'Haenens91] D'Haenens, I.; F. Van Assche; E. Halpin & B. Karakostas. Experiences With Rule-Based Dynamic Modeling. In: *Dynamic Modelling of Information Systems*, H. G. Sol & K. M. van Hee (eds.), North-Holland: Amsterdam, pp.289-301, 1991.

[Embley92] Embley, D.; B. Kurtz & S. Woodfield. *Object-Oriented System Analysis: A Model Driven Approach*. Englewood Cliffs: Prentice-Hall, 1992.

[Essink91] Essink, L. Dynamic Modelling: An Example of an Event-Driven Approach. In: *Dynamic Modelling of Information Systems*, H. G. Sol & K. M. van Hee (eds.), North-Holland: Amsterdam, pp.89-119, 1991.

[Gibson90] Gibson, E. Objects - Born and Bred. *Byte*, Peterborough, v.15, n.10, pp.245-254, Oct. 1990.

[Harel87] Harel, D. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, Amsterdam, v.8, n.3, pp.231-274, Jun. 1987.

[Heuser90] Heuser, C. *Modelagem Conceitual de Sistemas: Redes de Petri*. Kapelusz: Buenos Aires, 1990.

[Jacobson93] Jacobson, I.; M. Christerson; P. Jonsson & G. Övergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. 4th revised printing. Wokingham: Addison-Wesley, 1993.

[Kappel&Schrefl91] Kappel, G. & M. Schrefl. Using an Object-Oriented Diagram Technique for the Design of Information Systems. In: *Dynamic Modelling of Information Systems*, H. G. Sol & K. M. van Hee (eds.), North-Holland: Amsterdam, pp.121-164, 1991.

[Martin&Odell93] Martin, J. & J. Odell. *Principles of Object-Oriented Analysis and Design*. Englewood Cliffs: Preintice-Hall, 1993.

[Monarchi&Puhr92] Monarchi, D. & G. Puhr. A Research Typology for Object-Oriented Analysis and Design. *Communications of the ACM*, New York, v.35, n.9, pp.35-47, Sep. 1992.

[Olle82] Olle, T. W. Comparative Review of Information Systems Design Methodologies: Problem Definition. In: IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodlogies, 1982, Noordwijkerhout. *Proceedings*... Amsterdam: North-Holland, pp.8-9, 1982.

[Rubin&Goldberg92] Rubin, K. & A. Goldberg. Object Behavior Analysis. *Communications of the ACM*, New York, v.35, n.9, pp.48-62, Sep. 1992.

[Rumbaugh91] Rumbaugh, J.; M. Blaha; W. Premerlani; F. Eddy & W. Lorensen. *Object-Oriented Modeling and Design*. Englewood Cliffs: Prentice-Hall, 1991.

[Shlaer&Mellor92] Shlaer, S. & S. Mellor. *Object Life Cycles: Modeling the World in States*. Englewood Cliffs: Prentice-Hall, 1992.

[Tsalgatidou&Loucopoulos91] Tsalgatidou, A. & P. Loucopoulos. An Object-Oriented Rule-Based Approach to the Dynamic Modelling of Information Systems. In: *Dynamic Modelling of Information Systems*, H. G. Sol & K. M. van Hee (eds.), North-Holland: Amsterdam, pp.165-188, 1991.

[Yourdon94] Yourdon, E. *Object-Oriented Systems Design: A Integrated Approach*. Englewood Cliffs: Prentice-Hall, 1994.