Em direção a uma metodologia para o desenvolvimento de frameworks de aplicação orientados a objetos

Ricardo Pereira e Silva Eng., M.Sc.* Roberto Tom Price Eng., M.Sc., D.Phil

Universidade Federal do Rio Grande do Sul - Instituto de informática
Caixa postal 15064 - Porto Alegre - RS - Brasil

* Universidade Federal de Santa Catarina - Depto. de Informática e de Estatística
Caixa Postal 476 - Florianópolis - SC - Brasil

e-mail: {ricardo, tomprice}@inf.ufrgs.br

Abstract

The analisys of some framework development methodologies, shows the lack of modeling techniques and a detailed development process. In contrast, this features are contained in most object-oriented application development methodologies. Starting with analisys of these methodologies, is discussed how to adapt their modeling techniques to be used in frameworks development. The perceived need is to extend the static modeling techniques with class and method categorization, and to stress the use of colaborations for the dynamic modeling.

Keywords: Software Engineering; software development methodologies; object-oriented framework;

1 - Introdução

A reutilização de componentes de software em larga escala é um dos argumentos a favor da abordagem de orientação a objetos. Em muitos casos, constitui uma perspectiva frustrada, pois a reutilização não é característica inerente da orientação a objetos, mas deve ser obtida a partir do uso de técnicas que produzam software reutilizável [Johnson 88] [Pree 95]. Os frameworks orientados a objetos[1] são estruturas de classes interrelacionadas, que permitem não apenas reutilização de classes, mas minimizam o esforço para o desenvolvimento de aplicações - por conterem o protocolo de controle da aplicação (a definição da arquitetura), liberando o desenvolvedor de software desta preocupação. Os frameworks invertem a ótica do reuso de classes, da abordagem bottom-up[2] para a abordagem top-down. A lógica de desenvolvimento parte da visão global da aplicação, já definida no framework, em direção aos detalhes da aplicação específica, que são definidos pelo usuário do framework. Assim, a implementação de uma aplicação a partir do framework é feita pela adaptação de sua estrutura de classes, fazendo com que esta que inclua as particularidades da aplicação [Deutsch 89] [Taligent 95].

Frameworks promovem reutilização de software em larga escala. Em contrapartida, produzir um framework é uma tarefa muito mais complexa, que produzir aplicações específicas. O início do desenvolvimento de um framework demanda profundo conhecimento do domínio de aplicações tratado. Isto requer a existência prévia de aplicações. O desenvolvimento de um framework demanda a generalização das características de um domínio de aplicações, para capacitá-lo a cobrir diferentes aplicações deste domínio. Além disso, o processo de desenvolvimento de frameworks precisa ser iterativo e incremental: precisa ser refinado ao longo de ciclos de desenvolvimento, e deve ter a capacidade de assimilar novas informações do domínio, capturadas ao longo da utilização do framework. A aceitação de um framework requer o seu teste no desenvolvimento de diferentes aplicações. Todos estes aspectos exigem um esforço de desenvolvimento bastante superior ao necessário, para o desenvolvimento de uma aplicação específica [Johnson 93].

Metodologias de desenvolvimento de frameworks ora propostas, descrevem como produzir frameworks, sem estabelecer um processo detalhado de construção de modelos que dirija o desenvolvimento [Johnson 93] [Taligent 94] [Pree 95]. Também não estabelecem um conjunto de mecanismos de descrição que contenham o projeto do framework - a modelagem do domínio de aplicações. A documentação proposta pelos autores de metodologias abrange fundamentalmente a questão de como utilizar os frameworks. Mecanismos como *cookbooks*, contratos e *patterns* [Pree 95] são usados para descrever os aspectos de implementação do framework, cujo entendimento é necessário para o desenvolvimento de aplicações.

Várias metodologias de desenvolvimento de aplicações vem sendo propostas, tendo como base a abordagem de orientação a objetos[3]. Caracterizam-se por um conjunto de técnicas de modelagem e de um processo de desenvolvimento. Admitem o uso de ferramentas para auxílio ao processo de desenvolvimento de aplicações, que podem ser integradas em ambientes de desenvolvimento. Por outro lado, as metodologias OOAD em geral, se atêm a produzir uma aplicação específica, sem se preocupar com o desenvolvimento de classes de objetos reutilizáveis,

---

1 Também chamados frameworks de aplicação orientados a objetos, ou simplesmente, frameworks.

2 Desenvolvimento bottom-up é o que ocorre quando classes de objetos de uma biblioteca são interligadas, para a contrução de uma aplicação.

3 Metodologias de desenvolvimento de aplicações orientadas a objetos, que abrangem as etapas de análise e projeto serão tratadas daqui por diante como metodologias OOAD.

ou em buscar subsídios em bibliotecas de classes. Usam o encapsulamento de dados da orientação a objetos basicamente como um mecanismo de estruturação da especificação[4] em uma abordagem de desenvolvimento top-down.

No presente trabalho é desenvolvida uma análise de como aspectos de metodologias OOAD podem ser aproveitados, para o preenchimento das lacunas presentes nas metodologias de desenvolvimento de frameworks. Inicialmente são apresentadas as características de frameworks e duas metodologias de desenvolvimento de frameworks: projeto dirigido por exemplo (example-driven design) [Johnson 93] e projeto dirigido por hot spot (hot spot[5] driven design) [Pree 95]. A seguir, uma descrição sumária de três metodologias OOAD[6]: OMT [Rumbaugh 94], OOSE [Jacobson 92] e Fusion [Coleman 94]. Finalmente, é discutido como as técnicas de modelagem presentes nas metodologias OOAD poderiam ser adaptadas para seu uso no desenvolvimento de frameworks.

2 - Frameworks de aplicação orientados a objetos

Um framework de aplicação orientado a objetos é uma estrutura de classes interrelacionadas, que constitui uma implementação inacabada, para um conjunto de aplicações de um domínio. Rebecca Wirfs-Brock e Ralph Jonhson [Wirfs-Brock 90] propõem a seguinte definição para framework: "uma coleção de classes concretas e abstratas e as interfaces entre elas, e é o projeto de um subsistema". Em outra publicação, Ralph Jonhson acrescenta: "não apenas classes, mas a forma como as instâncias das classes colaboram" [Jonhson 93].

A diferença fundamental entre um framework e a reutilização de classes de uma biblioteca, é que neste caso são usados componentes isolados, cabendo ao desenvolvedor estabelecer sua interligação, e no caso do framework, é procedida a reutilização de um conjunto de classes interrelacionadas - interrelacionamento estabelecido no projeto do framework. As figuras abaixo ilustram esta diferença. A figura 1 contém uma aplicação que reutiliza classes de uma biblioteca de classes - usando a notação de OMT. Em cinza, estão representadas as classes reutilizadas e em preto, o que é definido pelo desenvolvedor da aplicação, ou seja, outras classes e as associações entre classes. A figura 2 contém uma aplicação desenvolvida a partir de um framework. A diferença entre os dois casos é que na aplicação da figura 1, o desenvolvedor deve estabelecer as associações entre classes e no caso da utilização do framework, como ilustrado na figura 2, as associações são predefinidas, cabendo ao desenvolvedor da aplicação conectar as classes desenvolvidas de uma forma predeterminada pelo framework (na figura 2 como na figura 1, está representado em cinza o que é predefinido, e em preto, o que é acrescentado pelo desenvolvedor da aplicação).

---

4 Em função das vantagens do tipo abstrato de dados, em relação aos módulos da abordagem funcional. Esta comparação é tratada em [Meyer 88].

5 A expressão hot-spot foi mantida em Inglês e significa "parte importante", significado este não obtido a partir de sua tradução literal.

6 Estas metodologias OOAD se destacam como estando entre as que apresentam mais características desejáveis em relação a outras avaliadas, em publicações que procedem avaliações de metodologias, como [Fowler 94], [Hodgson 94], [Monarchi 92], [Pastor 94], [Perez 95].

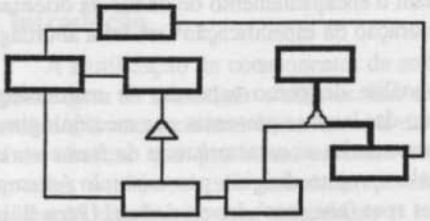

Figura 1 - uma aplicação orientada a objetos com reutilização de classes

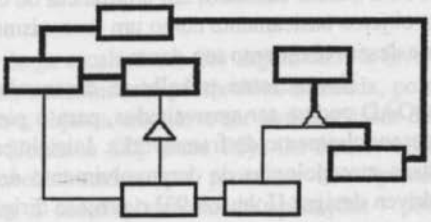

Figura 2 - uma aplicação orientada a objetos a partir de um framework

Dois aspectos caracterizam um framework [Taligent 95]:

- Os frameworks fornecem infraestrutura e projeto: frameworks portam infraestrutura de projeto disponibilizada ao desenvolvedor da aplicação, o que reduz a quantidade de código a ser desenvolvida, testada e depurada. As interconexões preestabelecidas definem a arquitetura da aplicação, liberando o desenvolvedor desta responsabilidade. O código escrito pelo desenvolvedor visa estender ou particularizar o comportamento do framework, de forma a moldá-lo a uma necessidade específica.
- Os frameworks "chamam", não são "chamados": um papel do framework é fornecer o fluxo de controle da aplicação. Assim, em tempo de execução, as instâncias das classes desenvolvidas esperam ser chamadas pelas instâncias das classes concretas do framework.

De acordo com a forma como deve ser utilizado, um framework pode se classificar como dirigido a arquitetura ou dirigido a dados. No primeiro caso, a aplicação deve ser gerada a partir da criação de subclasses das classes do framework. No segundo caso, a partir de diferentes combinações de objetos, gerados pela instanciação das classes presentes no framework. Frameworks dirigidos a arquitetura são mais difíceis de usar, pois, para a geração das subclasses exigem um profundo conhecimento do seu projeto, bem como um esforço no desenvolvimento de código. Frameworks dirigidos a dado são mais fáceis de usar, porém são menos flexíveis. Uma outra abordagem seria uma combinação dos dois casos, onde o framework apresentaria uma base dirigida a arquitetura e uma camada dirigida a dado. Com isto, possibilita a geração de aplicações a partir da combinação de objetos, mas permite a geração de subclasses [Taligent 94].

As metodologias OOAD são compostas de conjuntos de técnicas de modelagem e um processo de desenvolvimento. As metodologias de desenvolvimento de frameworks descritas a seguir, estabelecem como produzir frameworks, sem se ater a modelos. Assim, fica subentendido que um procedimento de desenvolvimento de frameworks, pela necessidade de documentação, deve adotar uma metodologia OOAD como base de trabalho, alterando seus procedimentos em função das especificidades do desenvolvimento de frameworks. Com isto, as metodologias OOAD podem vir a complementar as metodologias de desenvolvimento de frameworks.

Projeto dirigido por exemplo [Johnson 93]

Johnson estabelece que o desenvolvimento de um framework para um domínio de aplicação é decorrente de um processo de aprendizado a respeito deste domínio, que se processa concretamente a partir do desenvolvimento de aplicações ou do estudo de aplicações desenvolvidas. Em função das pessoas pensarem de forma concreta, ao invés de abstrata, a abstração do domínio - que é o próprio framework - é obtida a partir da generalização de situações concretas - as aplicações. Abstrações são obtidas de uma forma bottom-up, a partir do

exame de exemplos concretos: aspectos semelhantes de diferentes aplicações são fatorados, dando origem a classes abstratas que agrupam as semelhanças, cabendo às classes concretas do nível hierárquico inferior, a especialização, para satisfazer cada caso.

O processo de generalização ocorre a partir da busca de elementos que recebem nomes diferentes, mas são a mesma coisa; recorrendo à parametrização para eliminar diferenças; particionando elementos para tentar obter componentes similares; agrupando elementos similares em classes.

O processo de desenvolvimento segundo o projeto dirigido por exemplo, atravessa as etapas de análise, projeto e teste:

1 - Analisar o domínio de aplicação:
- aprender as abstrações já conhecidas;
- coletar exemplos de programas que poderiam ser desenvolvidos a partir do framework (no mínimo, quatro);
- avaliar a significância de cada exemplo no contexto do domínio;

2 - Projetar uma hierarquia de classes que possa ser especializada para abranger os exemplos (um framework);

3 - Testar o framework usando-o para desenvolver os exemplos (implementar, testar e avaliar cada exemplo usado na primeira etapa, utilizando para isto, o framework desenvolvido).

Observe-se que o passo 2 corresponde à construção de uma especificação, como procedido em metodologias OOAD.

Projeto dirigido por hot spot [Pree 95]

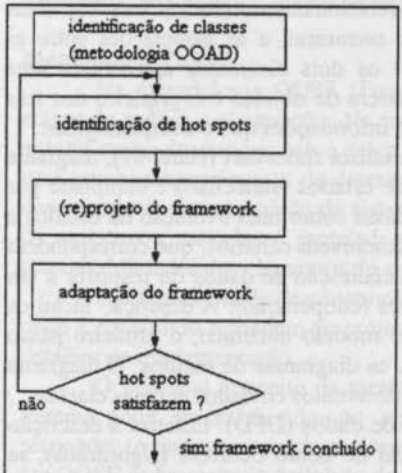

Figura 3 - as etapas do projeto dirigido por hot spot para o desenvolvimento de um framework - extraído de [Pree 95]

Uma aplicação orientada a objetos é completamente definida. Um framework, ao contrário possui partes propositalmente não definidas - o que lhe dá a capacidade de ser flexível e se moldar a diferentes aplicações. Os hot spots são as partes do framework mantidas flexíveis. A essência da metodologia é identificar os hot spots na estrutura de classes de um domínio, e a partir disto construir o framework. Pree propõe a seguinte seqüência de procedimentos:

- A primeira etapa, a identificação de classes, consiste em definir uma estrutura de classes - semelhante ao que é feito no desenvolvimento de uma aplicação orientada a objetos. Isto demanda a aplicação de uma metodologia OOAD.
- Na segunda etapa são identificados os hot spots. É preciso identificar que aspectos diferem de aplicação para aplicação e definir o grau de flexibilidade que deve ser mantido em cada caso.
- A terceira etapa, o projeto do framework (ou a reelaboração do projeto), consiste em modificar a estrutura de classes inicialmente definida, de modo que possa comportar a flexibilidade requerida. Nesta etapa se definirá o que o usuário do framework deve fazer para gerar uma aplicação - que subclasses deve definir, a que classes

deve fornecer parâmetros para a criação de objetos, quais as combinações de objetos possíveis.

- A quarta etapa consiste num refinamento da estrutura de classes do framework. Após isto o framework é avaliado (a questão é: os hot spots definidos dão ao framework o grau de flexibilidade requerido para gerar aplicações ?). Caso não seja considerado satisfatório, retorna-se à etapa de identificação de hot spots.

3 - Metodologias de desenvolvimento de aplicações orientadas a objetos

Metodologia OMT

A metodologia OMT [Rumbaugh 94] se propõe a gerar uma especificação de sistema, utilizando diferentes modelos de representação, tendo cada um maior ou menor importância, em função de aspectos como tipo de aplicação e linguagem de programação usada. Preconiza a construção de um modelo do domínio de aplicação tratado (análise) e na posterior adição a este modelo, dos detalhes de implementação (projeto). A mesma descrição do domínio do problema será usada ao longo da análise e projeto, sendo enriquecida em informações ao longo desta evolução.

São utilizados três elementos de modelagem para descrever um sistema: o modelo de objetos, que descreve a estrutura estática do sistema, ou seja, as classes e suas associações; o modelo dinâmico[7], que descreve a seqüência de interações entre os objetos do sistema e o modelo funcional, que descreve as transformações de valores dos dados procedidas pelo sistema.

O modelo de objetos estabelecido pela metodologia OMT é definido pelos autores, como uma tentativa de aperfeiçoamento do modelo de entidade-relacionamento (ER) [Chen 76]. Seus principais elementos são as classes, com sua organização estrutural, e as associações entre as classes, representadas como traços interligando classes - os dois elementos apresentam uma correspondência com os elementos do diagrama ER. O modelo de objetos é o primeiro dos três modelos a ser construído. Os demais modelos porém, geram informações que o complementam.

O modelo dinâmico descreve aspectos de controle e utiliza statechart [Harel 87], diagrama de eventos e diagrama de fluxo de eventos. Um diagrama de estados (statechart) é composto por estados e transições, e modela o comportamento de uma classe como uma evolução de estados a partir da ocorrência de eventos. Os diagramas de eventos descrevem cenários, que correspondem às situações a que o sistema pode ser submetido (como inicialização de dados ou resposta a um comando do usuário) e que explicita a interação entre classes (cooperação). A descrição inclui os eventos e a ordem em que estes ocorrem. Na geração do modelo dinâmico, o primeiro passo consiste em gerar os diagramas de eventos e a partir deles, os diagramas de estados. O diagrama de fluxo de eventos concentra em um diagrama, o conjunto de eventos enviados entre as classes.

O modelo funcional se utiliza do diagrama de fluxo de dados (DFD). Enfatiza a descrição de cálculos e transformações de dados, sem a preocupação de como ocorrem (algoritmos), se ocorrem (procedimentos de decisão), em que ordem ocorrem (seqüenciamento) ou quem realiza (alocação de tarefas aos elementos do sistema). Um processo do DFD constituirá um método de

---

7 No contexto de OMT (pela nomenclatura que usa), a modelagem dinâmica se atém à descrição comportamental (controle). No restante do texto, modelagem dinâmica se refere a dois aspectos: descrição comportamental, que se preocupa com o fluxo de controle do sistema, e descrição funcional, que se preocupa em descrever os fluxos de dados e suas transformações.

classe ou um fragmento de método. O DFD produz uma descrição da aplicação como um todo, sem referência às classes.

A metodologia estabelece uma seqüência de etapas para elaborar uma descrição de sistema - análise, projeto e implementação. O conjunto de modelos (de objetos, dinâmico e funcional) é desenvolvido na etapa de análise, sem a definição dos métodos de classe. Ao longo do projeto os mesmos modelos são refinados visando completar a especificação do sistema tratado.

A análise parte de um enunciado do problema e gera uma descrição para o sistema composta pelos três modelos do método. Neste primeiro esforço o objetivo é entender e descrever o problema estabelecido. A ênfase da descrição produzida durante a análise é o que o sistema deve fazer e não, como o sistema deve fazer. Os objetos presentes na descrição da análise são elementos e conceitos do domínio da aplicação e não, conceitos da solução computacional.

A etapa de projeto é dividida em duas subetapas, o projeto do sistema e o projeto dos objetos - desenvolvidas cronologicamente nesta ordem. Na etapa de projeto do sistema são definidos os subsistemas (segundo um critério basicamente funcional). Cada subsistema engloba classes do sistema que compartilham algumas propriedades comuns - funcionalidade, localização física etc. Um subsistema consiste em um pacote de classes interrelacionadas, com uma interface bem definida com outros subsistemas.

O projeto dos objetos consiste na evolução dos modelos gerados na análise. Complementa a descrição daquilo que o sistema deve fazer (definida na análise) com o modo como o sistema deve fazer. As classes provenientes da análise são complementadas com métodos e novos atributos; novas classes pertencentes ao domínio da solução são acrescentadas aos modelos.

É desenvolvido um dicionário de dados contendo a descrição das classes, de seus atributos e métodos, ao longo das etapas de análise e projeto.

Metodologia OOSE

Na metodologia OOSE [Jacobson 92] o desenvolvimento de uma aplicação passa pelas etapas de análise e construção. Na análise é elaborado o modelo de requisitos, que apresenta uma especificação do sistema sob a ótica de um observador externo, e o modelo de análise, que faz uma primeira aproximação da descrição interna do sistema. Os autores estabelecem que a análise deve produzir uma descrição de sistema independente da realidade de implementação. A definição de atributos e métodos é protelada para o projeto - a prioridade da análise é definir papéis e responsabilidades dos elementos do sistema.

A etapa chamada de construção se divide em projeto e implementação. A etapa de projeto, onde é construído o modelo de projeto, visa "formalizar"[8] o modelo de análise e completá-lo com detalhes de implementação.

O principal conceito da metodologia é o *use case*[9]. *Use cases* são as situações a que o sistema pode ser submetido, ou seja, as situações de processamento a que o sistema deve responder (o que corresponde aos cenários de OMT). As várias etapas de descrição de um sistema em OOSE se baseiam na definição de *use cases*.

Na análise são construídos os modelos de requisitos e de análise. A construção do modelo de requisitos é iniciada com a identificação dos atores, ou seja, os elementos que interagem com o

---

8 Os modelos de requisitos e de análise estão muito escorados em descrição textual, sendo assim bastante informais.

9 A expressão "use cases" pode ser traduzida como casos de uso ou como situações de utilização. No presente texto será mantida a expressão em Inglês, por ser característica da metodologia.

sistema. Identificados os atores, são definidos os *use cases* a partir da busca de todas as situações de interação entre atores e sistema. Atores e *use cases* são agrupados em um diagrama sintaticamente simples e é procedida uma descrição textual de cada *use case*, sob a ótica de um observador externo ao sistema. O modelo de requisitos inclui uma descrição informal da interface e um primeiro modelo de objetos, com as classes do domínio do problema identificadas, associações e herança.

O elemento central do modelo de análise é um conjunto de diagramas de modelo de objetos, sendo desenvolvido um diagrama para cada *use case*. O modelo de objetos é composto por três tipos de classes: as classes entidade, que são as classes do domínio do problema; as classes de interface, responsáveis pela comunicação entre o sistema e o meio externo; e as classes de controle, que concentram funcionalidades específicas de *use cases*, que não se deseje[10] alocar em outras classes. OOSE divide as associações do modelo de objetos em dois tipos: as associações de conhecimento que indicam manutenção de referência do objeto associado, e as associações de comunicação, que indicam interação em tempo de execução (cooperação). A associação entre classes não é constante nos vários diagramas (duas classes associadas em um *use case*, podem não estar associadas em outro, por exemplo). O conjunto de informações que define uma classe pode estar distribuído por vários diagramas. O modelo de análise é completado por descrição textual associada a cada diagrama do modelo de objetos (descrição das classes e dos *use cases*). Cabe salientar que OOSE concentra poucas informações específicas às classes no modelo de análise: não há preocupação em definir atributos e métodos[11].

O modelo de projeto é mais formal que os modelos desenvolvidos durante a etapa de análise. É composto pelo modelo de blocos, diagramas de interação, especificação da interface dos blocos e diagramas SDL. A primeira atividade da etapa de projeto é o desenvolvimento do modelo de blocos, que é a evolução do modelo de objetos da análise (em função de fatores referentes à implementação, novas classes podem incluídas na especificação). Os diagramas de interação (semelhantes aos diagramas de eventos usados em OMT) formalizam a descrição de *use cases* desenvolvida na análise. O projeto dos blocos, fase seguinte da etapa projeto, inicia com a definição da interface de cada classe. Isto consiste em relacionar atributos e a assinatura de métodos das classes, na sintaxe da linguagem de programação adotada para a implementação (um header de C++, por exemplo). No projeto do comportamento dos módulos, o diagrama SDL é usado para descrever cada classe, funcional e comportamentalmente.

Metodologia Fusion

Na metodologia Fusion [Coleman 94] o desenvolvimento de uma aplicação passa pelas etapas de análise, projeto e implementação. Na análise é construído o modelo de objetos e o modelo de interface. O modelo de objetos de Fusion é muito semelhante ao modelo de objetos de OMT, ou seja, com uma semântica muito próxima aos diagramas ER. O modelo de interface contém a modelagem dinâmica desenvolvida na etapa de análise. Durante a análise não existe a noção de método de classe, mas de operação do sistema. São identificadas as possíveis interações

---

10 A expressão *desejo* quer significar estilo de modelagem, ou seja, qualquer sistema descrito usando classes de controle pode também ser descrito sem que seja usado este tipo de classe. Isto é semelhante à opção de concentrar informações ou não, nas associações de OMT.

11 Atributos necessariamente aparecem na descrição textual, mas sem a preocupação de identificar todos os atributos de uma classe. OOSE recomenda definir papéis e responsabilidades das classes, ao invés de métodos específicos, durante a análise.

do sistema com seu meio externo - as "operações" - de uma forma semelhante à construção do modelo de requisitos de OOSE. Após isto, as operações são descritas. Na modelagem das operações durante a análise, o sistema é sempre visto como um todo (ao invés de um conjunto de objetos que interagem). Durante o projeto é produzida uma descrição do conjunto de classes - são definidos novos atributos e o conjunto de métodos. O projeto utiliza modelos diferentes dos modelos da análise. Para a implementação, Fusion prevê o uso de uma linguagem orientada a objetos.

Fusion supõe o início do processo de desenvolvimento da especificação a partir de uma descrição textual dos requisitos. A análise inicia com a construção do modelo de objetos. São considerados dois tipos de atributos: os atributos dado, que armazenam informações inerentes à classe, e os atributos objeto, que armazenam a referência a outros objetos. Durante a construção do modelo de objetos apenas os atributos dado são incluídos na descrição.

A tarefa seguinte da etapa de análise, é a identificação e descrição das interações do sistema com o meio externo. O modelo de interface utiliza o conceito de operação para descrever estas interações. A descrição de uma operação envolve o sistema, agentes[12], e eventos trocados entre eles. Segundo a metodologia, uma operação do sistema se constitui de um evento de entrada (originado por um agente) e do efeito que ele pode ter. O modelo de interface é composto pelos modelos de operação e de ciclo de vida. O modelo de operação é uma descrição textual para cada operação, em formato estruturado. Cada operação é descrita nos seguintes termos: nome da operação, descrição textual da operação, relação dos dados utilizados, relação dos possíveis eventos de saída originados pela operação, definição da precondição para que a operação ocorra (predicado) e o estado do sistema resultante da ocorrência da operação. O modelo de operação não apresenta uma visão temporal da seqüência de passos para a execução da operação.

O modelo de ciclo de vida descreve as seqüências de operações possíveis para o sistema, ou seja, o ciclo de vida de uma execução do sistema - também porta a informação de temporalidade das etapas de uma operação, ausente no modelo de operação.

Durante a análise a ênfase é a identificação e detalhamento das operações, sob a ótica de um observador externo. Na etapa de projeto estas informações são estendidas para o conjunto de classes, originando os métodos, os atributos objeto e novos atributos dado. O grafo de interação de objetos é o primeiro modelo gerado na etapa de projeto. Este diagrama descreve a interação entre objetos para a realização de uma operação do sistema. Cada operação identificada origina um grafo de interação de objetos - que admite estruturação. Um dos objetos envolvidos na interação é definido como coordenador da operação (decisão de projeto). Cabe ao coordenador iniciar a operação. Os outros objetos envolvidos na operação são chamados colaboradores.

A atividade seguinte é a construção dos grafos de visibilidade, que concentram a informação das referências a outros objetos que uma classe mantém - e que origina os atributos objeto. Os grafos de visibilidade organizam a informação para cada classe do sistema: para cada classe é construído um grafo de visibilidade.

A terceira atividade da etapa de projeto é a construção das descrições de classe. É gerada uma descrição para cada classe do sistema. Como o modelo anterior, organiza por classes, uma informação disponível no restante da especificação. A descrição de cada classe apresenta um formato semelhante a um header de C++ e contém: nome da classe, superclasses (caso haja), atributos e assinatura dos métodos.

---

12 Os agentes de Fusion são equivalentes aos atores de OOSE.

Os grafos de herança descrevem estrutura de herança das classes do sistema descrito. A sintaxe destes grafos é exatamente a sintaxe de herança do modelo de objetos.

Um dicionário de dados é construído ao longo das etapas de análise e projeto, descrevendo de forma textual, classes, atributos, métodos e conceitos presentes na modelagem, como predicados, eventos, agentes, etc.

Quadro resumo dos das técnicas de modelagem das metodologias OOAD[13] apresentadas

	visão estática	visão dinâmica	
		classes	sistema
OMT	modelo de objetos	statecharts	diagramas de eventos (cenários), DFD
OOSE	modelo de objetos, especificação da interface	diagramas SDL	diagramas de interação (cenários)
Fusion	modelo de objetos, descrição de classe, grafos de visibilidade grafos de herança		grafos de interação de objetos (cenários), modelo de interface

Tabela 1 - resumo dos das técnicas de modelagem das metodologias OOAD apresentadas

4 - Uso de técnicas de modelagem para o desenvolvimento de frameworks

Uma especificação construída sob a ótica da orientação a objetos é a descrição de um conjunto de classes - as classes individualmente e suas associações. Isto se aplica tanto a especificação de uma aplicação quanto a de um framework. A especificação de um sistema deve abranger uma descrição estática e uma descrição dinâmica. A descrição dinâmica apresenta dois aspectos: a descrição comportamental, que se preocupa com o fluxo de controle do sistema, ou seja, a seqüência de execução, e a descrição funcional, que se preocupa em descrever os fluxos de dados e suas transformações entre unidades funcionais. Em um sistema baseado em orientação a objetos as técnicas de modelagem também devem ter a capacidade de descrever as classes individualmente e o sistema como um todo - caso este que consiste em descrever o interrelacionamento entre as classes. O conjunto de técnicas de modelagem das metodologias OOAD apresentadas abrange estes aspectos.

Analisa-se a seguir como as técnicas de modelagem das metodologias OOAD podem ser usadas para o desenvolvimento de frameworks. Não se entra no mérito de comparar técnicas de modelagem ou processos de desenvolvimento de diferentes metodologias OOAD, ou comparar metodologias de desenvolvimento de frameworks. A ênfase adotada é avaliar que aspectos das técnicas de modelagem devem ser adaptados para seu uso no desenvolvimento de frameworks. A análise é desenvolvida inicialmente sobre aspectos de modelagem estática e a seguir, de modelagem dinâmica. As adaptações de notação sugeridas são dirigidas à etapa de projeto, no ciclo de vida do framework.

---

13 Excluídas descrições textuais como dicionário de dados, e modelos textuais de OOSE (da etapa de análise).

Modelagem estática

As classes abstratas de um framework são repositórios de conceitos gerais do domínio de aplicação. Além disto, a geração de aplicações a partir de um framework pode ser baseada em criação de subclasses de classes abstratas. Com isto, é importante que em um modelo de objetos de um framework haja distinção gráfica entre classes abstratas e classes concretas.

O projeto de um framework estabelece a flexibilidade permitida ao usuário para a geração de aplicações. Assim, a representação de classes deve distinguir graficamente as classes que devem (ou que podem) ser acrescentadas pelo usuário, para o desenvolvimento de uma aplicação.

A notação de modelo de objetos de metodologias OOAD nem sempre adota a distinção entre classe abstrata e concreta (das três metodologias OOAD apresentadas, apenas Fusion o faz). Além disto, não existe o conceito de *classe a acrescentar* no contexto do desenvolvimento de aplicações orientadas a objetos - não sendo portanto, tratado pelas metodologias OOAD. Estes fatores forçam que se adapte a notação do modelo de objetos das metodologias OOAD, para que estas possam ser usadas no desenvolvimento de frameworks.

A figura 4 sugere uma notação para a diferenciação entre as classes abstratas do framework, classes concretas do framework e classes a serem criadas pelo usuário do framework na geração de aplicações.

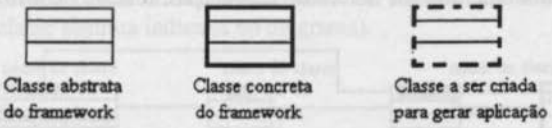

Figura 4 - distinção gráfica das classes de um framework

Esta diferenciação na notação além de mostrar onde é possível a extensão da estrutura de classes, permite distinguir a obrigatoriedade ou não da criação de classes - e no caso da obrigatoriedade, se é possível a inclusão de mais de uma classe. A criação de classes por parte do usuário (subclasses de classes do framework), poderá se enquadrar em uma das situações apresentadas na figura 5. No primeiro caso o projeto prevê que o usuário do framework deve produzir exatamente uma subclasse de uma classe abstrata - a estrutura de controle projetada para o framework não comporta mais de uma subclasse. No segundo caso o usuário do framework deve produzir pelo menos uma subclasse, mas tem a possibilidade de gerar mais subclasses da classe abstrata. O terceiro caso é parecido com o segundo, exceto pelo fato do framework incluir na estrutura, uma ou mais subclasses concretas. Assim, deixa de existir a obrigatoriedade do usuário produzir uma subclasse.

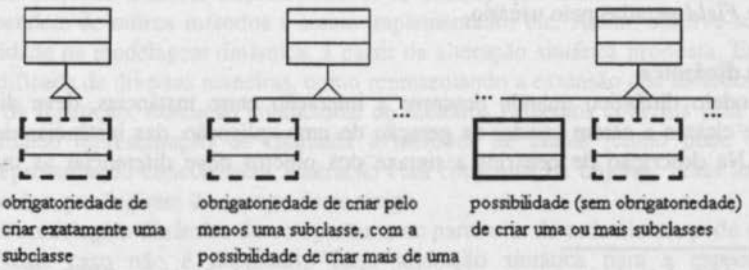

Figura 5 - possíveis situações das subclasses criadas pelo usuário do framework

Um método de uma classe abstrata pode ser classificado como [Johnson 91]:

- abstrato: um método que não é completamente definido na classe abstrata; deve ser definido em uma subclasse;
- template: é definido em termos de outros métodos (métodos hook);
- base: é completamente definido.

Na execução de um método template ocorre a chamada de pelo menos um método hook. O método hook por sua vez, pode ser um método abstrato, base ou template [Pree 95].

No contexto de um framework, um método de uma classe abstrata pode ser deixado propositalmente incompleto para que sua definição seja acabada na geração de uma aplicação. Este é o caso por exemplo, de um método abstrato pertencente a uma classe abstrata do framework, que deve ser sobreposto na subclasse gerada pelo usuário, no desenvolvimento de uma aplicação. A notação que relaciona os métodos[14] de uma classe abstrata de um framework deve explicitar esta classificação (abstrato, template ou base), pois é uma informação relevante do projeto: aspectos de implementação mantidos flexíveis, e que devem ser definidos na geração de aplicações. As metodologias OOAD não possuem uma notação para esta classificação por não ser um aspecto relevante no contexto do desenvolvimento de aplicações específicas.

O exemplo abaixo apresenta uma parte do modelo de objetos de um framework para a visualização de hiperdocumentos. Os nomes de associação, relação de atributos e métodos foram omitidos.

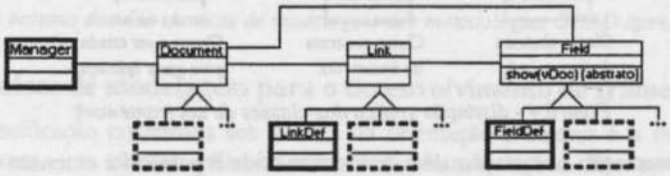

Figura 6 - um exemplo de modelo de objetos de um framework

O uso da notação sugerida torna claro que para a geração de uma aplicação, o usuário precisa gerar pelo menos uma subclasse de *Document*, e pode gerar subclasses de *Link* e *Field*. Esta informação não poderia ser obtida diretamente de um modelo de objetos produzido segundo a sintaxe de uma metodologia OOAD. Isto demonstra que a distinção de classes proposta, produz um aumento de expressividade para o modelo de objetos.

A identificação do tipo de método nas classes abstratas do framework também explicita aspectos necessários para a geração de aplicações. O método *show* da classe *Field*, por exemplo, que é classificado como abstrato no diagrama acima, deve ser sobreposto em todas subclasses concretas de *Field* geradas pelo usuário.

Modelagem dinâmica

O modelo dinâmico quando descreve a interação entre instâncias, deve diferenciar as instâncias de classes a serem criadas na geração de uma aplicação, das instâncias de classes do framework. Na descrição de cenários a sintaxe dos objetos deve diferenciar as instâncias das

[14] Em OMT a relação dos métodos é posta no símbolo de classe, do modelo de objetos; OOSE e Fusion relacionam métodos em um modelo à parte ("especificação de interface" e "descrição de classe", respectivamente).

classes do framework, das instâncias das classes a serem criadas. Existem três situações possíveis a descrever:

- o objeto do diagrama é uma instância de classe do framework;
- o objeto do diagrama é uma instância de classe definida pelo usuário;
- o objeto do diagrama pode ser instância de classe do framework ou de classe criada pelo usuário.

A notação das metodologias OOAD não permite esta diferenciação, pois no contexto de uma aplicação específica, os objetos dos cenários são sempre instâncias de classes concretas da aplicação. Assim para que se possa distinguir a origem dos objetos presentes em um cenário, é necessário uma alteração da sintaxe de representação dos objetos. A figura 7 contém uma sugestão de notação para a diferenciação das instâncias. O primeiro caso é semelhante ao que se observa em um cenário de aplicação: o objeto é identificado com o nome da classe a que pertence, e pode ser chamado para a execução dos métodos desta classe (ou métodos concretos de superclasses). Nos dois últimos casos a classe do objeto no diagrama será definida na geração de uma aplicação. Assim, o objeto é identificado com um nome de classe abstrata, indicando que corresponderá a uma instância de uma subclasse. Os métodos das chamadas ao objeto são métodos da classe abstrata. No último caso é indicado que o objeto pode ser uma instância de classe do framework, ou de classe criada pelo usuário. As classes das duas situações possuem mesma superclasse (a classe abstrata indicada no diagrama).

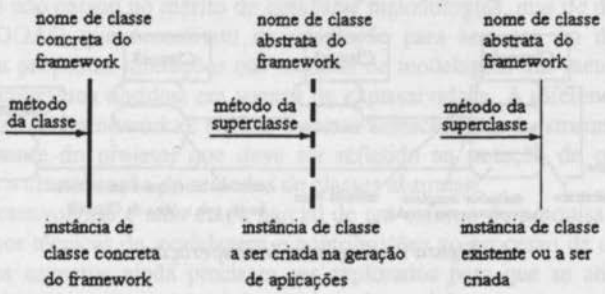

Figura 7 - diferenciação entre instâncias de classes do framework e da aplicação

A figura 8 apresenta um cenário do framework para a visualização de hiperdocumentos (do exemplo anterior).

A notação adotada torna evidentes aspectos das classes a serem produzidas na geração de aplicações: os métodos a serem implementados, os clientes que usam estes métodos, se estes métodos dependem de outros métodos a serem implementados etc. Assim, observa-se um ganho de expressividade na modelagem dinâmica, a partir da alteração sintática proposta. Esta notação pode ser modificada de diversas maneiras, como representando a extensão dos métodos, adotando mecanismos de repetição, execução condicional de métodos (aspectos cobertos pela notação de OOSE), incluindo representação de chamada a métodos de classe (como pode ocorrer em Smalltalk), representando concorrência, interação com conjuntos de objetos. Estes aspectos não foram explorados, por fugirem do escopo deste artigo.

Para a modelagem dinâmica do comportamento particular de cada classe, pode ser adotado statechart. Neste caso não é necessário fazer alteração sintática para a especificação de frameworks. A descrição de uma classe concreta do framework é equivalente à descrição de classe

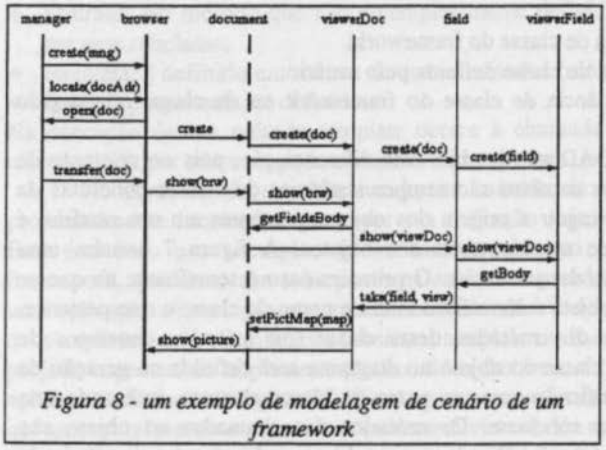

Figura 8 - um exemplo de modelagem de cenário de um framework

de uma aplicação. Como a notação dos statecharts admite estruturação, um estado de uma classe abstrata do framework pode corresponder a um conjunto de estados da classe concreta de uma aplicação, sendo que o diagrama da superclasse sempre está contido no diagrama da subclasse.

Uma notação adicional à descrição de cenários, é necessária para explicitar a dependência existente entre métodos de classe (cooperação). A explicitação da classificação dos métodos mostraria claramente os aspectos de projeto a serem respeitados na definição do algoritmo de métodos das classes a serem criadas. A figura 9 apresenta uma sugestão de diagrama de cooperação.

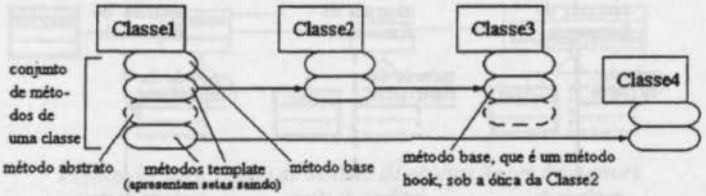

Figura 9 - diagrama de cooperação

Aspectos adicionais

A avaliação da possibilidade das metodologias OOAD serem usadas para o desenvolvimento de frameworks, originou o presente estudo, ainda em andamento [Silva 96]. Os acréscimos sugeridos às técnicas de modelagem geram um ganho de expressividade, como demonstrado nos exemplos. Esta contribuição porém, não é suficiente para a definição de um procedimento completo de desenvolvimento de frameworks. Outros aspectos ainda precisam ser investigados.

O projeto de frameworks envolve conceitos não explorados suficientemente pelas metodologias OOAD. Um primeiro é a necessidade de mecanismos que levem mais diretamente à descrição do domínio. As metodologias OOAD de um modo geral, determinam "identificar classes", "identificar estruturas", mas sem estabelecer como, assim como não se aprofundam na questão da fatorização da estrutura de classes . Um possível caminho para transpor esta limitação é o estudo de técnicas de Análise de Domínio que levem a uma sistematização dos procedimentos de identificação de classes e estruturas. Uma outra possibilidade a explorar, é a aplicação de técnicas de Engenharia Reversa que sistematizem o processo de extrair uma descrição orientada a objetos de uma aplicação já desenvolvida.

Em termos de modelagem dinâmica, as técnicas de modelagem das metodologias OOAD em geral, não descrevem eficientemente a interação com conjuntos de objetos de uma classe. As notações são pouco claras para expressar se a interação envolve todo o conjunto, parte do conjunto, ou um objeto específico. Técnicas de modelagem que explicitem a cadeia de métodos envolvidos na execução de um determinado método - diagramas de cooperação - podem auxiliar a fatoração da estrutura de classes, pela identificação de semelhanças funcionais.

A utilização de *design patterns* no desenvolvimento de frameworks tende a minimizar o esforço de desenvolvimento, por dispor soluções para problemas comuns de projeto [Gamma 94]. É preciso definir precisamente o papel dos design patterns no processo de desenvolvimento de frameworks - o estágio do processo em que devem ser usados e a forma utilizá-los para gerar um ganho de qualidade no desenvolvimento.

5 - Conclusão

Analisando metodologias de desenvolvimento de frameworks propostas, observa-se a ausência de técnicas de modelagem e de um processo de desenvolvimento detalhado. Tais características fazem parte de metodologias OOAD. Assim, a partir da análise de metodologias OOAD e de metodologias de desenvolvimento de frameworks, discutiu-se como adaptar mecanismos das metodologias OOAD para seu uso no desenvolvimento de frameworks.

A discussão não entrou no mérito de comparar metodologias, mas de destacar os aspectos de metodologias OOAD que necessitam de adaptação para seu uso no desenvolvimento de frameworks. Foram propostas alterações nas técnicas de modelagem das metodologias OOAD e demonstrados os acréscimos obtidos, em termos de expressividade. A diferenciação entre classes abstratas e concretas do framework, e classes a serem acrescentadas à estrutura do framework, é um aspecto importante do projeto, que deve ser refletido na notação de projeto utilizada. O mesmo ocorre com a classificação de métodos de classes abstratas.

O estudo desenvolvido é uma etapa parcial de um esforço de pesquisa ora em curso, que visa analisar e propor técnicas de modelagem e contribuições ao processo de desenvolvimento de frameworks. Outros aspectos ainda precisam ser explorados para que se atinja esta meta, tais como a definição de um conjunto ótimo de técnicas de modelagem para a especificação de um framework, assim como de um processo de desenvolvimento; uso de metodologias de captura de características do domínio; utilização de design patterns e outras abordagens.

6 - Bibliografia

[Chen 76] CHEN, P. P. S. **The entity-relationship model - toward a unified view of data**. In: ACM Transactios on Database Systems 1. mar. 1976.

[Coleman 94] COLEMAN, D. *et all*. **Object-oriented development: the Fusion method**. Prentice Hall, 1994.

[Deutsch 89] DEUTSCH, P. **Frameworks and reuse in the smalltalk 80 system**. In: Software reusability. Ted Biggerstaff Ed., ACM Press, 1989.

[Fowler 94]. FOWLER, M. **Describing and comparing object-oriented analisys and design methods**. In: Object development methods. Edited: Carmichael. New York: SIGS Books, 1994.

[Gamma 94] GAMMA, E. *et all.* **Design patterns**: elements of reusable object-oriented software. Addison-Wesley, 1994.

[Harel 87] HAREL, D. *et all.* **Statecharts: a visual formalism for complex systems**. In: Science of Computer Progamming 8. 1987.

[Hodgson 94] HODGSON, R. **Contemplating the universe of methods**. In: Object development methods. Edited: Carmichael. New York: SIGS Books, 1994.

[Jacobson 92] JACOBSON, I. *et all.* **Object-oriented software engineering - a use case driven approach**. Addison-Wesley, 1992.

[Johnson 88] JOHNSON, R. E., FOOTE, B. **Designing reusable classes**. In: Journal of object-oriented programming. jun./jul. 1988.

[Johnson 91] JOHNSON, R. E., RUSSO, V. F. **Reusing object-oriented designs**. University of Ilinois tech report UIUCDCS91-1696, 1991.

[Johnson 93] JOHNSON, R. E. **How to design frameworks**. In: Notes of OOPSLA 93 tutorial. 1993.

[Meyer 88] MEYER, B. **Object-oriented software construction**. Englewood Cliffs: Prentice Hall, 1988.

[Monarchi 92] MONARCHI, D. E., PUHR, G. **A research typology for object-oriented analysis and design**. In: Communications of the ACM. v.35, n.9. sep. 1992.

[Pastor 94] PASTOR, E. A. **Estudo comparativo de metodologias de desenvolvimento de software**. UFRGS/II/CPGCC. Porto Alegre, 1994. Trabalho individual.

[Perez 95] PEREZ, M., ANTONETI, J. **A qualitative comparison of object-oriented methodologies**. In: Proceedings of 5th. International Symposium on Systems Research, Informatics and Cybernetics. Baden: aug. 1995.

[Pree 95] PREE, W. **Design patterns for object oriented software development**. Addison-Wesley, 1995.

[Rumbaugh 94] RUMBAUGH, J. *et all.* **Modelagem e projetos baseados em objetos**. Editora Campus, 1994.

[Silva 96] SILVA, R. P. **Avaliação de metodologias de análise e projeto orientadas a objetos voltadas ao desenvolvimento de aplicações, sob a ótica de sua utilização no desenvolvimento de frameworks orientados a objetos**. Porto Alegre: UFRGS/II/CPGCC, jul. 1996. TI 556.

[Taligent 94] TALIGENT. **Building object-oriented frameworks**. Taligent Inc. white paper, 1994.

[Taligent 95] TALIGENT. **Leveraging object-oriented frameworks**. Taligent Inc. white paper, 1995.

[Wirfs-Brock 90] WIRFS-BROCK, R., JOHNSON, R. E. **Surveying current research in object-oriented design**. Communications of the ACM. v.33, n.9. sep. 1990.