# O uso de técnicas de modelagem no projeto de frameworks 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. This work focuses on use of modeling techniques to framework design. A set of modeling techniques for the representation of class structure is proposed. This set extends the techniques used by currently object-oriented methodologies. The proposed notation, besides aiding the design process, can be used for better understanding of frameworks when building new applications.

**Keywords:** Software Engineering; object-oriented paradigm; software reuse; software development methodologies; object-oriented framework; patterns.

# 1 - Introdução

Um framework de aplicações orientado a objetos é uma estrutura de classes interrelacionadas, que constitui uma implementação inacabada, para um conjunto de aplicações de um domínio. Além de permitir a reutilização de um conjunto de classes, um framework minimiza o esforço de desenvolvimento de aplicações, por portar a definição da arquitetura das aplicações geradas a partir dele, como também por ter predefinido o fluxo de controle destas aplicações [TAL 95]. A figura 1 ilustra uma aplicação desenvolvida a partir de um framework. A parte sombreada corresponde ao framework - uma estrutura de classes que é reutilizada - e parte não-sombreada, corresponde ao que é produzido pelo usuário do framework no desenvolvimento de uma aplicação específica.

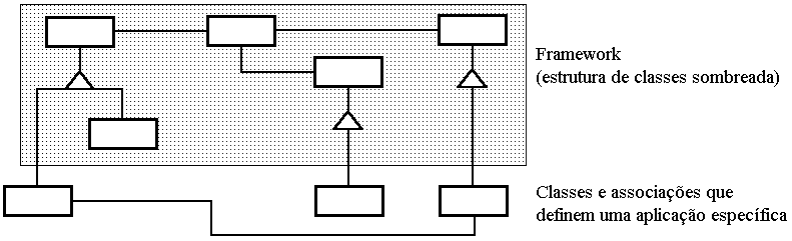


*Figura 1 - aplicação desenvolvida utilizando um framework*

Existem atualmente poucas metodologias voltadas ao desenvolvimento de frameworks. A metodologia "projeto dirigido por exemplo" (*example-driven design*) [JOH 93], é caracterizada pela ênfase à análise do maior número possível de aplicações do domínio tratado, já desenvolvidas, para a obtenção de conhecimento do domínio. Estabelece a seguinte seqüência de etapas para a produção de um framework:

1 - Análise do domínio (as aplicações existentes são a principal fonte de informações);

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

3 - Teste do framework através do seu uso no desenvolvimento dos exemplos (implementar, testar e avaliar cada exemplo usado na primeira etapa, utilizando para isto, o framework desenvolvido).

A metodologia, "projeto dirigido por pontos de flexibilização" (*hot spot driven design*) [PRE 95], enfatiza a busca de *hot spots*. Hot spots são as partes da estrutura de classes do framework (classes, métodos) que devem ser mantidas flexíveis, para possibilitar sua adaptação a diferentes aplicações do domínio. A metodologia estabelece basicamente a mesma seqüência de

etapas da metodologia "projeto dirigido por exemplo", porém sua ênfase é a identificação das partes da estrutura que variam em diferentes aplicações, e a seleção de soluções de projeto adequadas a estes casos.

Metodologias de desenvolvimento de frameworks ora propostas, descrevem como produzir frameworks, sem estabelecer um processo detalhado de construção de modelos que dirija o desenvolvimento [JOH 93] [TAL 94] [PRE 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* [PRE 95] são usados para descrever os aspectos de implementação do framework, cujo entendimento é necessário para o desenvolvimento de aplicações.

Este trabalho é voltado a aprofundar a discussão em torno do uso de técnicas de modelagem no projeto de frameworks. Um framework desenvolvido, é usado para exemplificar estas situações.

É proposto um conjunto de técnicas de modelagem para projeto de frameworks. Este conjunto está baseado em técnicas usadas por metodologias OOAD existentes, porém, com extensões sintáticas que permitem representar conceitos exclusivos de frameworks. Um framework desenvolvido, é usado para ilustrar como a notação de projeto proposta pode auxiliar (em termos de registro, classificação e organização de acesso às informações do projeto) no desenvolvimento, bem como na tarefa de aprender a usar o framework.

## 2 - O framework desenvolvido

Está sendo desenvolvido um pequeno framework para a geração de jogos de tabuleiro, FraG, construído como uma ferramenta didática, para o ensino de aspectos referentes a desenvolvimento e utilização de frameworks. FraG é voltado a generalizar um conjunto de jogos que tem em comum o uso de tabuleiro, podendo também utilizar dados, peões e outros elementos,

caracterizando um domínio de aplicações[1]. A atual versão foi implementada em Smalltalk, sob o framework MVC, que dá suporte aos aspectos de geração de interfaces gráficas, bem como à interação do usuário com a aplicação através destas interfaces.

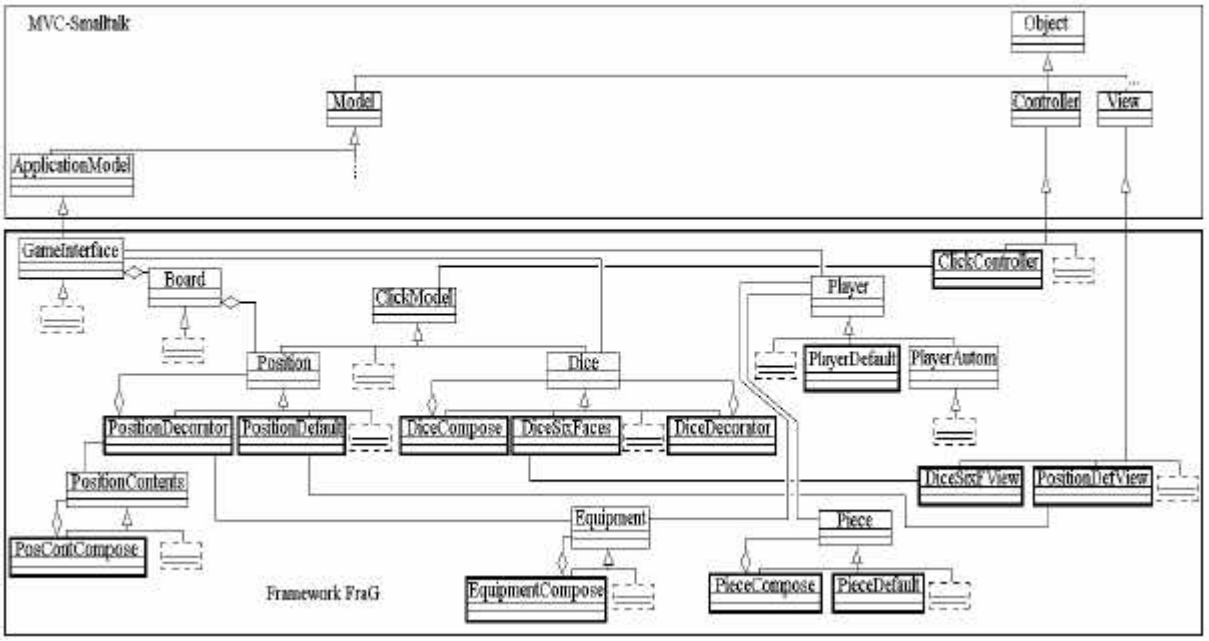


*Figura 2 - modelo de objetos do framework FraG*

A figura 2 apresenta um modelo de objetos que contém as principais classes do framework de jogos de tabuleiro[2]. A figura 3 apresenta a estrutura de classes de um exemplo de aplicação desenvolvida sob o framework - no caso um Jogo da Velha, que demanda a criação de apenas duas classes. A notação usada [SIL 96b] diferencia classes abstratas[3] (contorno com traço fino),

---

1 Exemplos de jogos: Jogo da Velha, Dama, Ludo Real, Gamão, Banco Imobiliário, RPG etc.

2 Foram omitidas as relações de atributos e métodos, e informações relativas às associações (tipo, cardinalidade etc.). Classes do framework sem superclasse explicitada, são subclasses de Model. Os nomes de algumas classes se referem a elementos do domínio: Dice (dado), Board (tabuleiro), Player (jogador), Piece (peão), Position (casa).

3 Consideram-se classes abstradas as classes que possuem pelo menos um método abstrato (com a assinatura definida, mas sem implementação) [JOH 91].

classes concretas (contorno com traço gosso) e classes a serem criadas pelo usuário do framework na geração de aplicações (contorno tracejado).

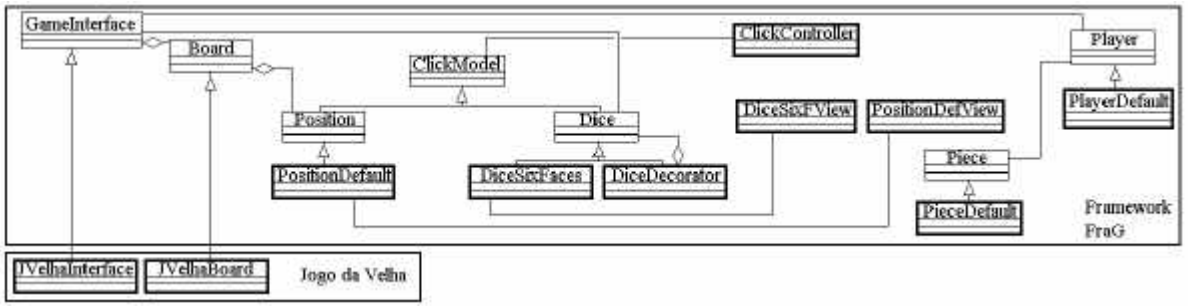


*Figura 3 - modelo de objetos do Jogo da Velha, desenvolvido sob o framework FraG*

## 3 - A necessidade de documentar o projeto de um framework

O processo de desenvolvimento de frameworks corresponde a uma evolução iterativa de sua estrutura de classes, que envolve atividades como identificação de classes, modelagem de cenários, identificação de estados de objetos etc. Este refinamento cíclico também é causado pela busca de adaptação da estrutura de classes, aos aspectos de generalidade, extensibilidade e flexibilidade. Estas atividades implicam no manuseio de uma grande quantidade de informações.

O uso de técnicas de modelagem gráficas no projeto, visa o registro, classificação e organização de acesso ao conjunto de informações referentes ao projeto do framework. Isto é útil para a etapa de desenvolvimento do framework (para orientar os ciclos de evolução), para a etapa de manutenção (o refinamento do framework, à medida em que se obtém novos conhecimentos do domínio), bem como para utilização do framework[4].

As metodologias de desenvolvimento de frameworks existentes estabelecem o processo de desenvolvimento em linhas gerais, sem se ater à definição de técnicas de modelagem. Alguns

---

[4] É possível que ao desenvolver uma aplicação, um usuário do framework crie desnecessariamente uma classe, ao invés invés de reutilizar uma classe disponível na estrutura do framework, por desconhecê-la. Tal situação foi observada por Campo ao submeter um grupo de usuários ao desenvolvimento de aplicações utilizando o framework HotDraw [CAM 97].

frameworks existentes também não apresentam uma descrição utilizando técnicas de modelagem baseadas em notação gráfica, o que torna difícil sua compreensão - e, conseqüentemente, sua utilização. Este é o caso, por exemplo, dos frameworks MVC [PAR 94a,b], e HotDraw [JOH 92], que tem sua documentação (distribuída com os frameworks) baseada em descrição textual, código do framework e código de exemplos de aplicações desenvolvidas a partir do framework. É uma tarefa complexa, por exemplo, aprender a desenvolver um editor gráfico, por mais simples que seja, utilizando o framework HotDraw, a partir de sua documentação. Esta dificuldade de compreender frameworks existentes, motivou Campo a desenvolver o Luthier, um mecanismo voltado à atividade de compreender um framework existente. Luthier é um framework projetado para suportar a construção de ferramentas visuais, para a análise dinâmica de programas orientados a objetos, baseado em meta-objetos [CAM 97].

Para suprir estas deficiências em termos de documentação, foi definido um conjunto de técnicas de modelagem para a documentação do projeto de frameworks (apresentado na tabela 1). Este conjunto é uma união de técnicas de modelagem utilizadas em metodologias OOAD (de análise e projeto orientadas a objetos, voltadas ao desenvolvimento de aplicações). Algumas técnicas foram estendidas de modo a abranger conceitos do contexto de frameworks, que não existem no contexto das metodologias OOAD, como classe a ser criada, por exemplo.

Um dicionário de dados complementa o conjunto de modelos. Contém uma descrição textual das classes, descrição dos atributos, descrição textual dos métodos e uma seção para a descrição de outras características não relacionadas diretamente a classes específicas, como a descrição dos *padrões* utilizados.

O primeiro princípio que norteou a definição das técnicas de modelagem foi a utilização de um mesmo conjunto de técnicas de modelagem ao longo de todo o processo de desenvolvimento de frameworks. Assim, o produto da análise está sujeito a alterações ao longo do projeto, facilitando a evolução iterativa da estrutura de classes do framework, ao longo de seu desenvolvimento. Além disto, não é estabelecida uma fronteira rígida entre estas etapas.

Um segundo princípio, é que a modelagem deve abranger as visões estática e dinâmica[5], havendo assim técnicas de modelagem voltadas para cada uma. Estas visões podem ser direcionadas ao sistema como um todo (conjunto de classes interrelacionadas) ou às classes separadamente. O conjunto de técnicas de modelagem definido cobre estes aspectos.

| | Sistema | Classes | Métodos |
|---|---|---|---|
| Modelagem estática | Modelo de objetos[6] | descrição das classes contida no modelo de objetos | assinatura dos métodos contida no modelo de objetos |
| Modelagem dinâmica | Cenários[7]<br>Modelo de ciclo-de-vida[8]<br>Diagrama de cooperação[9] | Statechart[10] | Diagrama SDL[11] |

*Tabela 1 - conjunto de técnicas de modelagem para o projeto de frameworks*

---

5 A modelagem estática é responsável pela descrição da estrutura de componentes de um sistema (descrição dos componentes e da sua organização). A modelagem dinâmica se refere a dois aspectos: descrição comportamental, que se preocupa com o fluxo de controle do sistema (seqüências de procedimentos), e descrição funcional, que se preocupa em descrever os fluxos de dados e suas transformações.

6 Notação proposta por Silva e Price [SIL 96b], baseada principalmente na notação de OMT [RUM 94], com diferenciação sintática de associação estática e dinâmica (como na notação de OOSE [JAC 92]) e com a diferenciação de tipos de classe (abstrata, concreta e a ser definida).

7 Notação proposta por Silva e Price [SIL 96b], baseada principalmente na notação de OOSE [JAC 92], que permite distinguir a origem dos objetos presentes em um cenário (instância de classe concreta do framework, instância de classe a ser definida pelo usuário ou a possibilidade de optar por um destes dois casos).

8 Utilizado pela metodologia Fusion [COL 94].

9 Técnica de modelagem proposta por Silva e Price [SIL 96b], que explicita todas as chamadas feitas a cada método e todas as chamadas a métodos feitas a partir cada método do sistema.

10 Utilizado pela metodologia OMT [RUM 94].

11 Utilizado pela metodologia OOSE [JAC 92].

Este conjunto de técnicas proposto se refere à documentação de projeto. A documentação fornecida ao usuário do framework pode incluir este mesmo conjunto de técnicas de modelagem, porém contendo apenas as classes que o usuário do framework precisa conhecer para produzir aplicações[12]. Além desta descrição, a documentação voltada ao usuário deve incluir um manual de utilização (como o cookbook de MVC [PAR 94a], por exemplo) e exemplos de aplicações desenvolvidas sob o framework (incluindo especificação de projeto e código fonte).

## 4 - Aspectos específicos da modelagem de frameworks

### 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 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.

---

12 Há duas razões de ordem prática para que a documentação fornecida ao usuário contenha apenas um subconjunto das classes que compõem o framework:

- quanto mais sumária for a descrição, menor será o esforço requerido para aprender a usar o framework [JOH 93];
- em termos de produção industrial de software, é inviável esperar que os produtores de software forneçam todo o código e documentação de projeto de um produto [SZY 96].

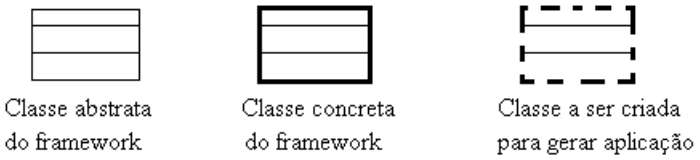


*Figura 4 - distinção gráfica das classes de um framework*

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.

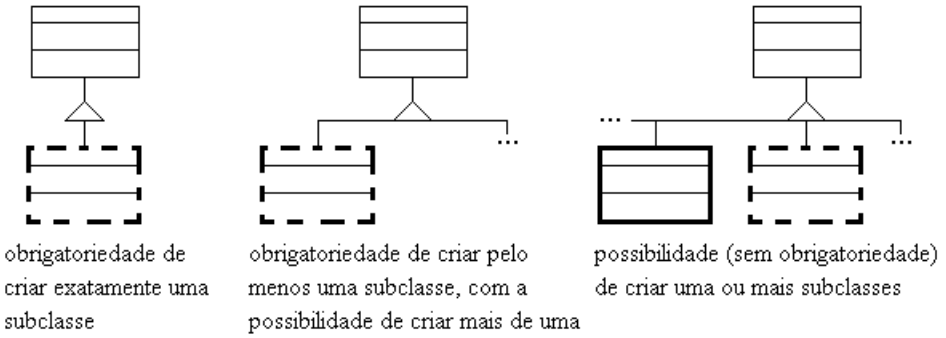


*Figura 5 - possíveis situações das subclasses criadas pelo usuário do framework*

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].

A notação que relaciona os métodos 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.

**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 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 6 contém uma 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).

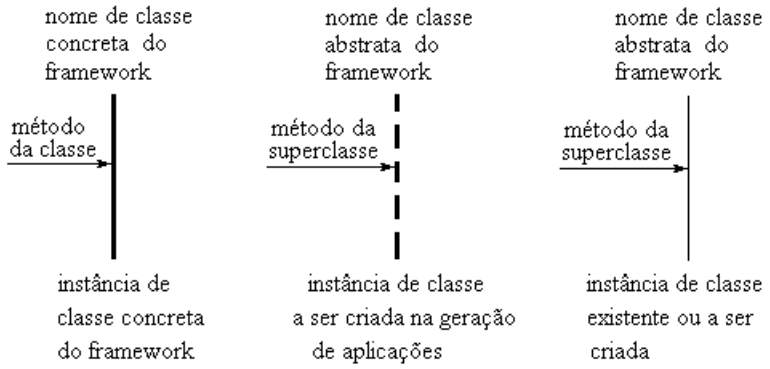


*Figura 6- diferenciação entre instâncias de classes do framework e da aplicação*

## 5 - Apoio da documentação ao aspecto de como usar o framework

A seguir são apresentados alguns exemplos de como a notação proposta pode auxiliar no processo de aprendizagem de como utilizar o framework. O framework FraG é utilizado para ilustrar as situações.

- **Questão**: a geração de uma aplicação ocorre a partir da produção de novas classes e da utilização de classes concretas presentes no framework. Então que classes devem ser desenvolvidas pelo usuário ? Que classes do framework podem ser reutilizadas ?

A partir do modelo de objetos do framework FraG (vide figura 2), observa-se para a geração de aplicações, a obrigatoriedade de produzir uma subclasse concreta das classes GameInterface e Board, e a possibilidade de produzir subclasses de outras classes como Position, Dice, Player etc., havendo nestes casos a possibilidade de usar as classes concretas existentes no framework. O primeiro esforço deve ser estudar as classes em que ocorre obrigatoriedade de geração de subclasses.

- **Questão**: ao gerar uma classe concreta para a aplicação (considerando o caso de subclasse de classe abstrata do framework), que métodos devem ser definidos e que métodos herdados podem ser reutilizados ?

A partir da notação da classe no modelo de objetos, descobrem-se quais métodos devem ser produzidos para cada classe (vide figura 7):

- métodos *abstratos* precisam ser definidos;

- métodos *template* fornecem uma estrutura de algoritmo definida, mas permitem flexibilidade através dos métodos hook chamados. A avaliação da necessidade ou convenência de produção de métodos deve ser estendida para os métodos hook.
- métodos *base* não precisam, mas podem ser sobrepostos.

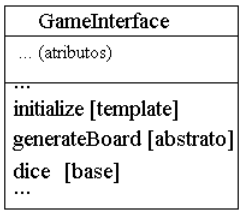


*Figura 7 - representação da classe GameInterface no modelo de objetos*

⌖ **Questão**: como descobrir as responsabilidades dos métodos a produzir ? Em que situação do processamento da aplicação eles atuam ?

Esta questão é complexa e sua resposta demanda a consulta a quase todos os modelos, como descrito a seguir.

→ O modelo de ciclo de vida descreve as seqüências de situações de processamento possíveis para o sistema, desde o início até o encerramento de uma execução, ou seja, o ciclo de vida de uma execução do sistema. O modelo de ciclo de vida do framework de jogos de tabuleiro está apresentado abaixo (expressão de mais alto nível de abstração).

*ciclo de vida* aplicaçãoEspecífica = inicialização **.** procedimento DeLance$^{+}$

*Figura 8 - expressão de mais alto nível, do modelo de ciclo de vida do framework de jogos*

→ As situações de processamento são detalhadas através de cenários. A figura 9 apresenta um trecho do cenário de inicialização do framework de jogos de tabuleiro[13].

13 Tracejado indica classe a ser criada; traço grosso contínuo, classe concreta do framework (caso que não ocorre neste cenário) e traço fino contínuo, indica que pode ser um dos casos anteriores [SIL 96b].

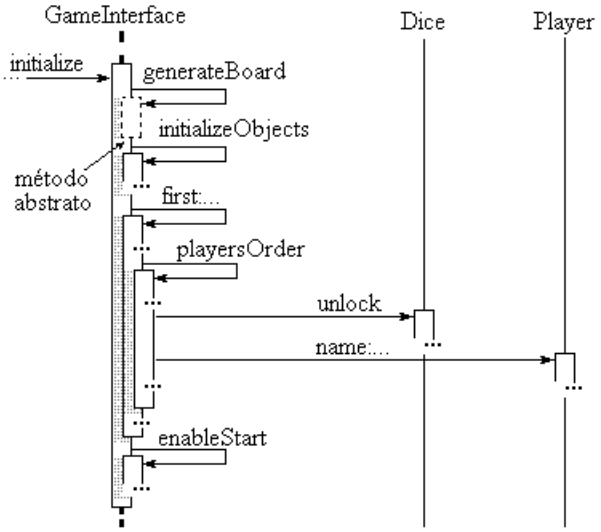


*Figura 9 - trecho do cenário de inicialização, do framework de jogos de tabuleiro*

→ O diagrama de cooperação apresenta toda a cooperação entre métodos (independente de situação de processamento específica, como no caso dos cenários). A figura 10 apresenta um diagrama de cooperação destacando apenas os métodos chamados pelo método "new:" da classe Board.

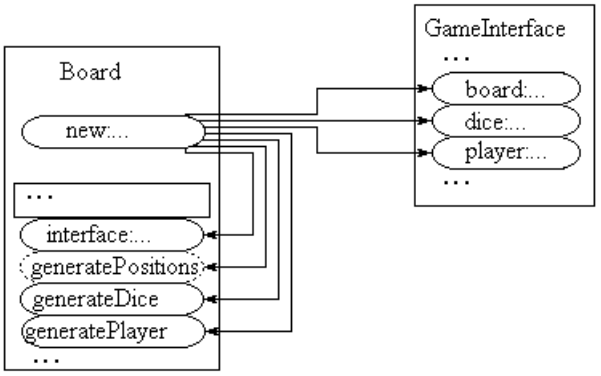


*Figura 10 - Diagrama de cooperação destacando os métodos chamados pelo método "new:" da classe Board*

→ Outras informações podem ser buscadas nos demais modelos:

- Statecharts e diagramas SDL detalham aspectos específicos de cada classe (evolução de estados de objetos, algoritmo de métodos etc);
- O dicionário de dados contém descrição textual das classes, atributos e métodos.

## 6 - Conclusão

A atividade de desenvolvimento de um framework ocorre de forma iterativa. Sua estrutura de classes é refinada ciclicamente em função da aquisição de informações, ou da busca de soluções de projeto mais adequadas. Foi proposto um conjunto de técnicas de modelagem para o projeto de frameworks, com a finalidade de complementar as metodologias de desenvolvimento de frameworks propostas atualmente. O conjunto de técnicas proposto visa auxiliar nos aspectos de registro, classificação e organização de acesso às informações do projeto, o que é útil para as fases de desenvolvimento (para apoiar a consulta e atualização de informações ao longo dos ciclos de desenvolvimento), de manutenção, como também para a utilização de frameworks.

Para auxiliar a tarefa complexa de manuseio de um conjunto de modelos que descreve um framework, estuda-se atualmente o desenvolvimento de ferramentas que permitam a edição, integração de diferentes modelos (com verificação de consistência) e uso de filtros de visualização, que permitam diferentes visões de um mesmo modelo.

## 7 - Bibliografia


[BEC 94] BECK, K., JOHNSON, R. **Patterns generate architectures**. (disponível por ftp anônimo em st.cs.uiuc.edu/pub).

[CAM 97] CAMPO, M. R. **Compreensão visual de frameworks através da introspeção de exemplos**. Porto Alegre: UFRGS/II/CPGCC, mar. 1997. Tese de Doutorado.

[COL 94] COLEMAN, D. *et al*. **Object-oriented development**: the Fusion method. Englewood Cliffs: Prentice Hall, 1994.

[FAY 96] FAYAD, M., CLINE, P. Aspects of software adaptability. **Communications of the ACM**. v.39, n.10. oct. 1996.

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

[JAC 92] JACOBSON, I. *et al*. **Object-oriented software engineering -** a use case driven approach. Reading: Addison-Wesley, 1992.

[JOH 92] JOHNSON, R. E. **Documenting frameworks using patterns**. In: Object-Oriented Programming Systems, Languages and Applications Conference - OOPSLA, 1992, Vancouver. **Proceedings...** Vancouver.

[JOH 93] JOHNSON, R. E. **How to design frameworks**. In: Object-Oriented Programming Systems, Languages and Applications Conference - OOPSLA, 1993, Washington. **Proceedings...** Washington.

[PAR 94a] PARCPLACE. **VisualWorks Cookbook**. ParcPlace Systems Inc.1994.

[PAR 94b] _____. **VisualWorks User's Guide**. ParcPlace Systems Inc.1994.

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

[RUM 94] RUMBAUGH, J. *et all*. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Editora Campus, 1994.

[SIL 96a] 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.

[SIL 96b] SILVA, R. P., PRICE, R. T. **Técnicas relacionadas ao desenvolvimento de frameworks**. Porto Alegre: UFRGS/II/CPGCC, dez. 1996. EQ 08.

[SZY 96] SZYPERSKI, C. **Component-oriented programming: a refined variation on object-oriented programming**. In: European Conference on Object-Oriented Programming - ECOOP, 1996, Linz. **Tutorial Notes** Linz.

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

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