Instituto de computação

Técnicas de Programação Avançada

TCC-00175

Profs.: Anselmo Montenegro

www.ic.uff.br/~anselmo

Conteúdo:Introdução a Frameworks para Aplicações

Baseado em Building Application Frameworks
Mohamed E. Fayad et Al
Wiley

Roteiro

Introdução
O que são Frameworks para Aplicações
Benefícios
Uma visão geral de Frameworks utilizados
Classificação dos Frameworks para Aplicações
Pontos fortes e fracos
Reuso de Componentes vs Reuso de Projetos
Frameworks para Aplicações vs outras técnicas de reuso

Roteiro

Como usar Frameworks para Aplicações
Aprendizado de Frameworks para Aplicações
Aspectos sobre Desenvolvimento de Frameworks para Aplicações

Introdução

Reuso de software tem sido um dos objetivos da Engenharia de Software por décadas

Reuso não é algo simples

Aos primeiros esforços voltados para reuso resultaram em pequenos componentes tipo caixa preta reutilizáveis

Introdução

O surgimento do paradigma orientado a objetos permitiu que componentes reutilizáveis maiores se tornassem disponíveis

Isto levou a definição de Frameworks para Aplicações Orientados a Objetos

Frameworks para Aplicações tem sido usado para muitos domínios diferentes

O uso de Framework para Aplicações tem permitido uma maior reutilização e uma disponibilização mais rápida de aplicações no mercado

O que são Frameworks para Aplicações

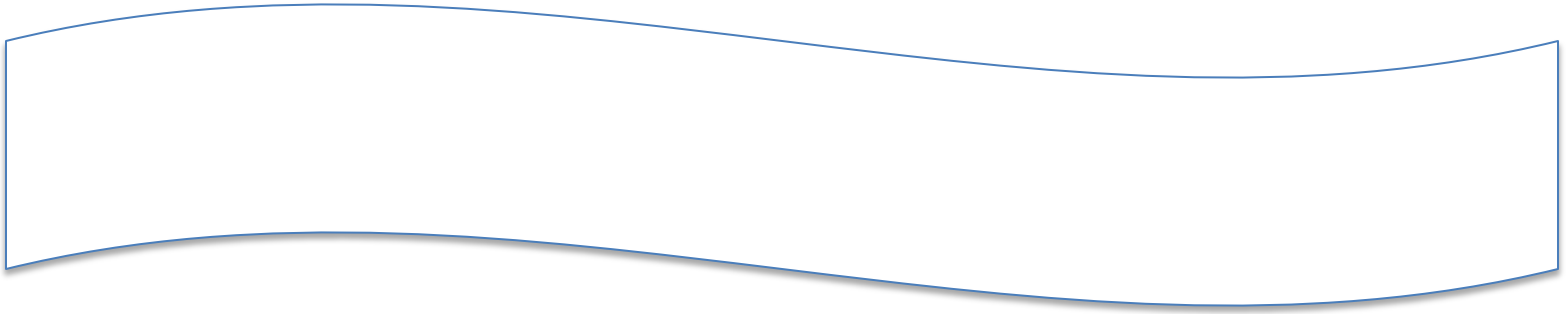
- Frameworks podem ser entendidos como uma técnica de reuso
- Tem sido utilizados há um bom tempo mas muitos projetos de Frameworks não tem sido bem sucedidos
- Há também a falta de uma metodologia orientada a objetos que indique como usar um framework

O que são Frameworks para Aplicações

- Existem diferentes formas de se definir um Framework

O que são Frameworks para Aplicações

- Existem diferentes formas de se definir um Framework



O que são Frameworks para Aplicações

A maioria dos frameworks comerciais tratam de interfaces, persistência e distribuição

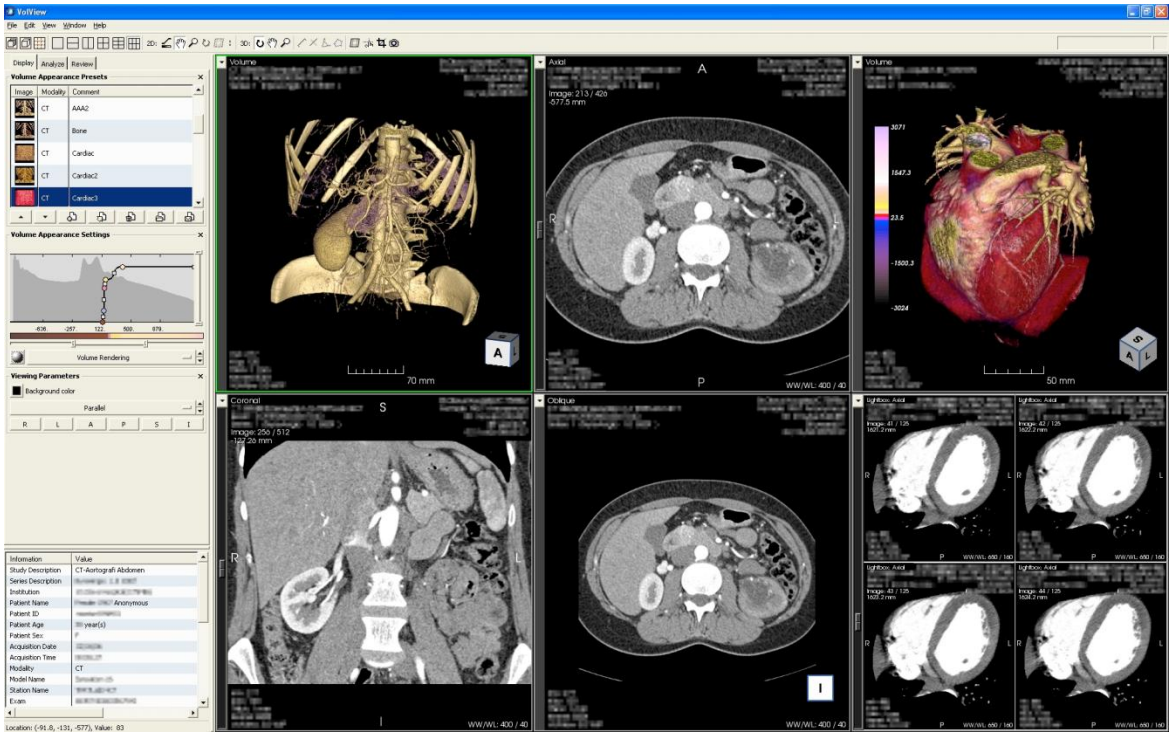
A maioria dos frameworks para aplicações específicas são proprietários

O que são Frameworks para Aplicações

- Exemplos de frameworks:
  - Hibernate – framework para mapeamento objeto-relacionalhttp://hibernate.org/
  - Struts – framework para desenvolvimento de camada controladora em uma estrutura Model 2(similar ao MVC) http://struts.apache.org/
  - Ruby on Rails- framework que facilita o desenvolvimento de aplicativos web orientados a banco de dados (usa também o MVC) http://rubyonrails.org/
  - VTK (Visualization Tookit) – framework para desenvolvimento de aplicações gráficas http://www.vtk.org/

O que são Frameworks para Aplicações

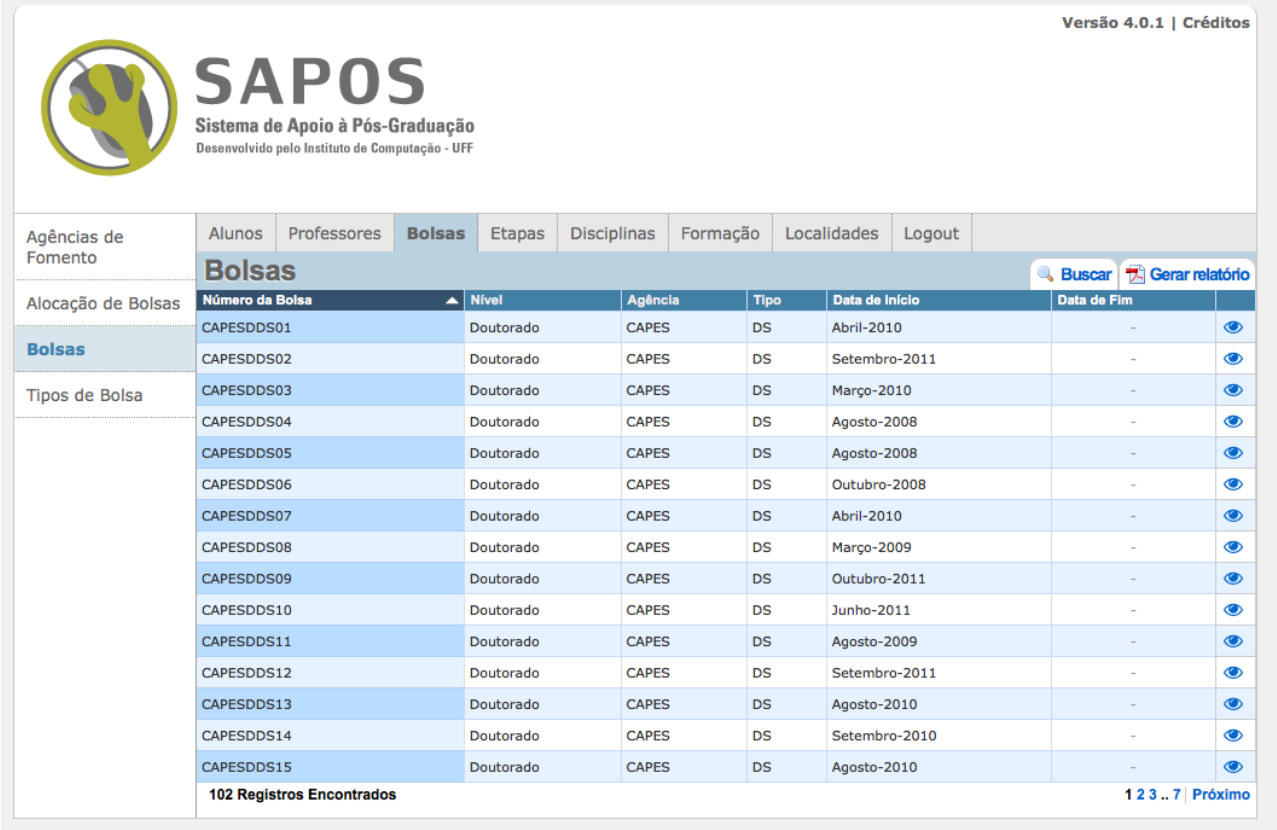
- Aplicações desenvolvidas com frameworks:



- http://www.vtk.org/

O que são Frameworks para Aplicações

- Aplicações desenvolvidas com frameworks:



- https://sel.ic.uff.br/sapos/

O que são Frameworks para Aplicações

- Um framework é um projeto de um sistema, reutilizável que descreve como o sistema é decomposto em um conjunto de objetos que interagem

- Às vezes o sistema é uma aplicação inteira ou um subsistema

O que são Frameworks para Aplicações

- O framework descreve:
  - os objetos componentes e suas interações
  - a interface de cada objeto e o fluxo de controle entre eles
  - como as responsabilidades do sistema são mapeadas nos objetos

O que são Frameworks para Aplicações

- A parte mais importante de um framework é o modo em que ele é dividido em componentes
- Frameworks também reutilizam implementações, mas isso é menos importante que:
  - o reuso das interfaces internas do sistema;
  - e como as funções são divididas em seus componentes

O que são Frameworks para Aplicações

> " O projeto de alto nível é o principal conteúdo intelectual de um software e os frameworks são uma das formas de reutilizá-los"

O que são Frameworks para Aplicações

- Características:
  - são tipicamente implementados em linguagens O.O
  - cada objeto é implementado como uma classe abstrata
  - os frameworks utilizam as classes abstratas como projeto de seus componentes pois definem a interface do seu esqueleto que podem ser estendidos para implementar tais componentes

O que são Frameworks para Aplicações

- Características:
  - são tipicamente implementados em linguagens O.O
  - cada componente é implementado como uma classe abstrata
  - os frameworks utilizam as classes abstratas como projeto de seus componentes pois definem a interface do seu esqueleto que podem ser estendidos para implementar tais componentes

O que são Frameworks para Aplicações

- Alguns sistemas O.O como a linguagem Java, o modelo COM (Common Object Models) e Corba (Common Object Request Broker Architecture) separam interfaces de classes

- Tais sistemas (usando interfaces) capturam apenas o aspecto estático de uma interface

O que são Frameworks para Aplicações

- Um framework possui um modelo colaborativo e um padrão de interação
- Por isso, frameworks em Java contém são descritos tanto por interfaces como classes abstratas
- As classes abstratas provêem parte da implementação de suas subclasses
- Exemplo: usando o padrão Template Method

O que são Frameworks para Aplicações

- Frameworks usufruem das 3 principais características de linguagens O.O:
  - Abstração de dados
  - Polimorfismo
  - Herança

O que são Frameworks para Aplicações

- Outra característica muito importante de um framework é a inversão de controle:
  - Controle direto – o desenvolvedor reutiliza componentes de um a biblioteca escrevendo o programa principal e chamando os componentes da biblioteca
  - Controle inverso- o programa principal é reutilizado e o código do desenvolvedor é chamado pelo código do framework

Benefícios

- Modularidade – frameworks encapsulam detalhes de implementação voláteis em interfaces estáveis
- Reusabilidade – as interfaces estáveis definem componentes genéricos que podem ser reaplicados para gerar novas aplicações.

Benefícios

- Extensibilidade – proveêm métodos hook (ganchos) que permitem a aplicação estender suas interfaces estáveis

- Os métodos hook desacoplam as interfaces estáveis e comportamentos de um domínio de aplicação de variações que podem surgir da instanciação de um aplicação para um contexto específico

Benefícios

- Inversão de controle – a arquitetura de um framework permite que os passos de processamento da aplicação sejam customizados por objetos tratadores de eventos que são invocados pelo mecanismo reativo do framework

- Em outras palavras, quando um evento ocorre, o tratador reage invocando métodos hook em objetos pre-registrados que executam ações específicas de processamento de eventos

- Exemplos de eventos: mensagens de janela, pacotes chegando de portas de comunicação

Classificação de Frameworks de Aplicação segundo o Escopo

- Frameworks para infraestrutura – facilitam o desenvolvimento de software básico portável e eficiente tais como S.O.'s, infraestrutura para comunicação, interface com o usuário e processamento de linguagem. São usados por organizações de software internamente e não são vendidos

- Frameworks de integração – são usados comumente para integrar aplicações e componentes distribuídos. Frameworks de integração Middleware são usados para que os desenvolvedores modularizem, reutilizem, estendam sua infraestrutura de software para trabalhar sem emendas em um ambiente distribuído. Exemplos: ORB frameworks, message-oriented middleware e bancos de dados transacionais

Classificação de Frameworks de Aplicação segundo o Escopo

- Enterprise Application Frameworks – frameworks que tratam de vastos domínios de aplicação. São caros de desenvolver e apesar de agilizarem o desenvolvimento de aplicativos para o usuário final, seu retorno é pequeno quanto comparado a frameworks de infra-estrutura e middleware de integração

Classificação de Frameworks de Aplicação black-box vs white-box vs gray-box

- White-box frameworks – dependem fortemente de características de linguagens O.O como herança e ligação dinâmica para alcançar extensibilidade. A funcionalidade existente é reutilizada e estendida via:

  (1) herança de classes base;
  (2) sobrescrita de métodos hook usando padrões como o template method

- O uso de frameworks do tipo white-box requer conhecimento intimo da estrutura interna do framework, em particular da estrutura de heranças que define os componentes

Classificação de Frameworks de Aplicação black-box vs white-box vs gray-box

- Black-box frameworks – suporta extensibilidade pela definição de interfaces que podem ser acopladas no framework via composição de objetos:
  (1) definição de componentes que respeitam a interface
  (2) integração no framework usando padrões como o Strategy

- Black-box frameworks são estruturados usando composição e delegação. Logo, black-box frameworks são mais fáceis de usar, já que requerem menos conhecimento da estrutura interna do framework. São , entretanto, mais difíceis de desenvolver já que é necessário desenvolver interfaces e hooks que prevejam a grande variedade de casos de uso

Classificação de Frameworks de Aplicação black-box vs white-box vs gray-box

- Gray-box frameworks – projetados para eliminar as limitação de frameworks white-box e black-box. Um gray-box framework possui boa flexibilidade e extensibilidade além de esconder informação desnecessária do desenvolvedor de aplicações

Limitações e fraquezas de Frameworks de Aplicação

- Manutenibilidade
  - Os requisitos de frameworks também mudam
  - Modificações e adaptações podem ocorrer em nível funcional(a funcionalidade não satisfaz os requisitos dos desenvolvedores) e nível não-funcional(aspectos qualitativos com reusabilidade e portabilidade)
  - Os desenvolvedores da aplicação dependem dos desenvolvedores dos frameworks

Limitações e fraquezas de Frameworks de Aplicação

- Validação e remoção de falhas
  - Difícil validar componentes abstratos
  - Dificuldade de depuração devido a inversão de controle o fluxo alternar entre os componentes da infra-estrutura do framework independente de aplicações e as callbacks específicas da aplicação

Limitações e fraquezas de Frameworks de Aplicação

- Eficiência – a generalidade e flexibilidade de um framework em geral reduzem a eficiência

- Em geral isto ocorre devido ao uso de ligação dinâmica de algumas linguagens O.O como C++ e Java

- Isto acarreta em:
  - (1) aumento no armazenamento (necessidade de ponteiros para tabelas virtuais)
  - (2) degradação do desempenho (overhead para invocar métodos ligados dinamicamente)
  - (3) inabilidade para armazenar objetos em memória compartilhada

Reuso: Componentes vs Designs

- A tecnologia de reuso ideal deveria prover componentes facilmente conectáveis para construir um novo sistema
- O desenvolvedor não precisa saber como o componente é implementado e a especificação é facilmente compreendida
- Isto ocorre com sistemas elétricos
- Em sistemas de software isto não é fácil de obter

Framework de aplicações vs outras formas de reuso

- O ideal de uma técnica de reuso pode ser vislumbrada com um componente que serve as necessidades de um desenvolvedor sem a necessidade de customização ou aprendizado
- Componentes com esta característica são difíceis de ser reaproveitados no futuro
- Quanto mais customizável um componente maior a probabilidade dele poder ser usado no futuro, porém, maior seu trabalho para se uso
- Exemplos de componentes: Enterprise Java Beans, Webservices, etc

Framework de aplicações vs outras formas de reuso

- Frameworks podem ser vistos como componentes no sentido de que vendedores os vendem como produtos
- Por outro lado, frameworks são mais customizáveis do que componentes
- É mais razoável entender frameworks e componentes como tecnologias distintas que colaboram entre si

Framework de aplicações vs outras formas de reuso

- Frameworks fornecem um contexto no qual um componente pode ser reutilizado
- Um framework provê um modo padrão para componentes tratarem erros, trocarem dados e invocarem operações uns sobre os outros
- Frameworks tornam mais fácil a criação de novos componentes a partir de componentes menores (exemplo, uma janela de diálogo a partir de widgets)
- Também provêem as especificações para um novo componente e um template para sua implementação

Framework de aplicações vs outras formas de reuso

- Frameworks são similares a outras técnicas de reuso como templates e schemas, porém a diferença é que frameworks são expressos em linguagens de programação

Como usar Frameworks

- O desenvolvimento baseado em um framework requer que a aplicação se encaixe no modelo do framework
- O projeto de uma aplicação baseada em um framework deve ser criado a partir do projeto do framework

Como usar Frameworks

- Uma aplicação baseada em framework tem 3 partes:
  - O framework
  - As subclasses concretas que herdam das classes abstratas do framework
  - Outros elementos

Como usar Frameworks

- Os demais elementos incluem:
  - Scripts que indicam que classes concretas serão utilizadas e como serão interconectadas
  - Objetos que não tem relação como framework
  - Objetos do framework que não são chamados pelo framework

Obs.: Objetos que são chamados pelo framework devem participar do modelo colaborativo do framework e logo são parte do framework

Como usar Frameworks

- Modos de utilizar um framework em função da dificuldade:
  - Conectar componentes existentes
  - Definir novas subclasses concretas
  - Extende-lo modificando as subclasses abstratas, criando novas operações e adicionando variáveis membro

Mais difícil

Como usar Frameworks

- Uso por conexão de componentes:
  - Não altera o framework
  - Não é preciso criar subclasses concretas
  - Reutiliza as interfaces e regras do framework para conectar componentes
  - Funciona como Lego
- Nem todos os frameworks funcionam assim

Como usar Frameworks

- Uso por definição de subclasses
  - Requer a criação de subclasses concretas
  - Requer mais conhecimento sobre as classes abstratas
  - O programador dever conhecer as interfaces do framework em detalhes

Como usar Frameworks

- Uso por extensão e modificação das classes abstratas
  - Modo mais difícil de usar um framework
  - Mais poderoso entretanto
  - Mudanças a classes abstratas podem quebrar a consistência de classes concretas

Como usar Frameworks

- Blackbox framework:
  - Programadores usam o framework conectando componentes sem ter que olhar para suas implementações
  - São mais fáceis de aprender a usar

Como usar Frameworks

- Whitebox framework:
  - Baseados em herança
  - Requerem mais conhecimento por parte do programador
  - Mais poderosos na mão de experts

Como usar Frameworks

- Graybox frameworks:
  - Blackbox e whitebox frameworks formam um espectro
  - É comum um framework ser usado como blackbox em um momento e depois ser estendido em outro momento
  - Graybox frameworks compartilham características de whitebox e blackbox frameworks

Como aprender a usar um Framework

- Frameworks são mais difíceis de aprender que uma biblioteca
- É necessário aprender todas as classes de um framework
- As classes mais importantes são abstratas o que torna o processo de aprendizado mais difícil
- Frameworks são mais fáceis de aprender se existe uma boa documentação
- A melhor forma de aprender um framework é por exemplo

Como aprender a usar um Framework

- Frameworks são mais difíceis de aprender que uma biblioteca
- É necessário aprender todas as classes de um framework
- As classes mais importantes são abstratas o que torna o processo de aprendizado mais difícil
- Frameworks são mais fáceis de aprender se existe uma boa documentação
- A melhor forma de aprender um framework é por exemplos

Como aprender a usar um Framework

- Exemplos são concretos e mais fáceis de compreender que o framework como um todo
- Exemplos resolvem problemas específicos e você pode estudar sua execução para aprender o fluxo de controle dentro de um framework

Aspectos sobre desenvolvimento de frameworks

- O desenvolvimento de frameworks requer iteração
- Porque iteração é necessário?
- Em geral, no caso de projetos, as iterações ocorrem porque os projetistas não sabem o que fazer a princípio, o que pode ser uma falha na etapa de análise
- Mesmo projetistas experientes recorrem a muitas iterações quando estão projetando frameworks

Aspectos sobre desenvolvimento de frameworks

- Etapas no projeto de um framework:

1. Análise do domínio
2. Coleta de exemplos
3. Geração de uma versão whitebox que implementa os exemplos
4. Uso do framework para construir aplicações
5. Determinação dos pontos fracos que são as partes difíceis de modificar
6. Melhorias no framework tornando-o mais blackbox
7. Estabilização do framework
8. Produção

Aspectos sobre desenvolvimento de frameworks

- Razões para iteração:
  - Falhas na análise do domínio descobertas quando um sistema é construído.
  - Frameworks tornam explicitas a partes do design que são prováveis de mudarem e a única forma de aprender o que se modifica é via experiência
  - Frameworks são abstrações cujo projeto depende dos exemplos originais. Quanto mais exemplos maior a chance de generalidade e reuso. Uma melhor notação para frameworks deve levar a mais passos de iteração na fase de projeto

Aspectos sobre desenvolvimento de frameworks

- Erros comuns: começar a usar o framework enquanto o projeto está em mutação
- Por outro lado, o uso de um framework é a única forma de descobrir o que está errado nele
- Dica: utilizar frameworks para projetos pilos para garantir que ele esteja suficientemente flexível e geral
- Se não estiverem, tais projeto serão casos de teste importantes para o desenvolvedores

Aspectos sobre desenvolvimento de frameworks

- É difícil criar frameworks em prazos devido a necessidade de se conhecer em profundidade o domínio da aplicação e as diversas iterações

- O projeto de um framework não deve jamais estar no caminho crítico de um projeto importante

Aspectos sobre desenvolvimento de frameworks

- Frameworks devem ser desenvolvidos por grupos de pesquisa ou de desenvolvimento avançado
- Por outro lado, o projeto de um frameworks deve estar fortemente atrelado aos desenvolvedores de aplicações porque são eles que conhecem o domínio de aplicação

Desenvolvimento de frameworks baseado em Hot-spots

- Um framework define uma linguagem de alto nível com a qual aplicações dentro de um domínio são criadas via especialização (ou adaptação)
- A especialização ocorre em pontos predefinidos de refinamento denominados Hot-spots
- Especialização pode ocorrer através de composição (blackbox) ou herança (whitebox)

Desenvolvimento de frameworks baseado em Hot-spots

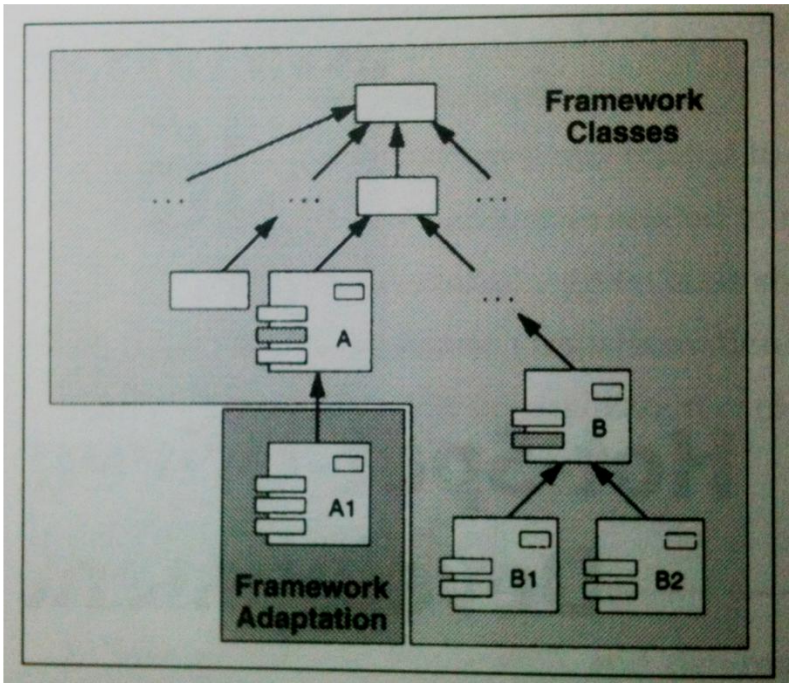
- Whitebox e blackbox são nomeclaturas que descrevem atributos de um framework
- Tais atributos categorizam seus Hot-spots
- Um framework não é jamais puramente whitebox ou blackbox

Desenvolvimento de frameworks baseado em Hot-spots

- Quando dizemos whitebox framework de fato queremos mencionar o aspecto whitebox de um framework
- O aspecto whitebox de um framework diz respeito as classes incompletas, isto é que contém métodos sem implementação padrão significativa

Hot-spots baseados em herança e interfaces

- Na figura ao lado, o método abstrato em cinza da classe A deve ser sobrescrito
- Os métodos abstratos formam neste caso o hot-spot
- Programadores modificam o comportamento de frameworks whitebox usando herança e sobrescrita de métodos



Hot-spots baseados em herança e interfaces

- Interfaces da linguagem Java representam classes abstratas puras consistindo apenas de métodos abstratos
- Interfaces permitem a separação de hierarquias de tipo e de classes
- A característica whitebox de um framework não muda com o uso de interfaces

Hot-spots baseados em composição

- Blackbox frameworks oferecem componentes prontos para adaptação
- Modificações são feitas simplesmente por composição
- Os hot-spots também correspondem a métodos no caso de blackbox frameworks
- Porém aqueles que especializam o framework o fazem via componentes de software inteiros

Hot-spots baseados em composição

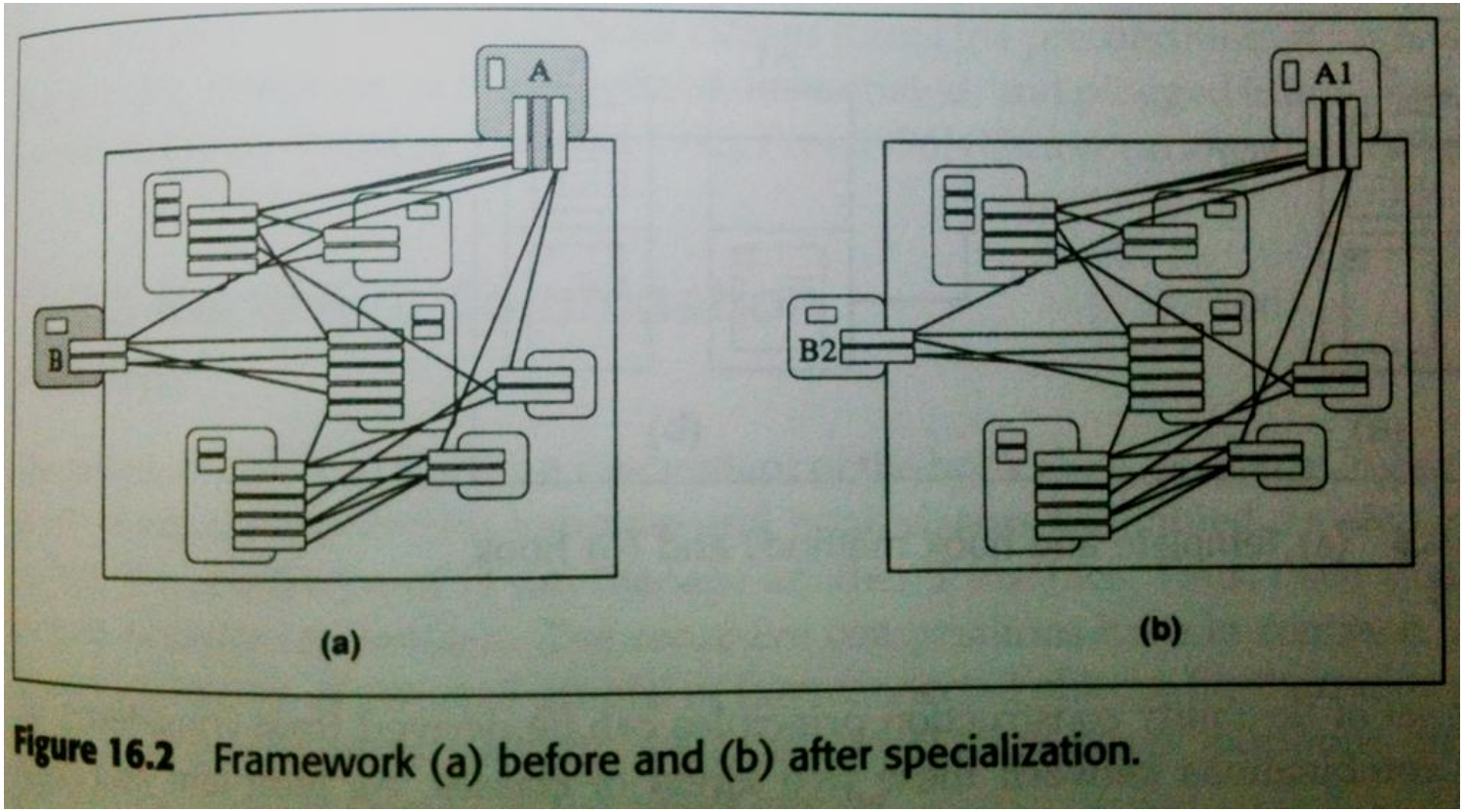


Figure 16.2 Framework (a) before and (b) after specialization.

Hooks como elementos construtores de Hot-spots

- Os métodos em uma classe de um framework podem ser categorizados em dois tipos: template methods e hooks
- Hooks podem ser vistos como repositórios ou hot-spots flexíveis que são invocados por métodos mais complexos
- Os métodos mais complexos que invocam os hooks são normalmente denominados template methods
- Template methods definem um comportamento abstrato, um fluxo de controle genérico ou a interação entre objetos

Hooks como elementos construtores de Hot-spots

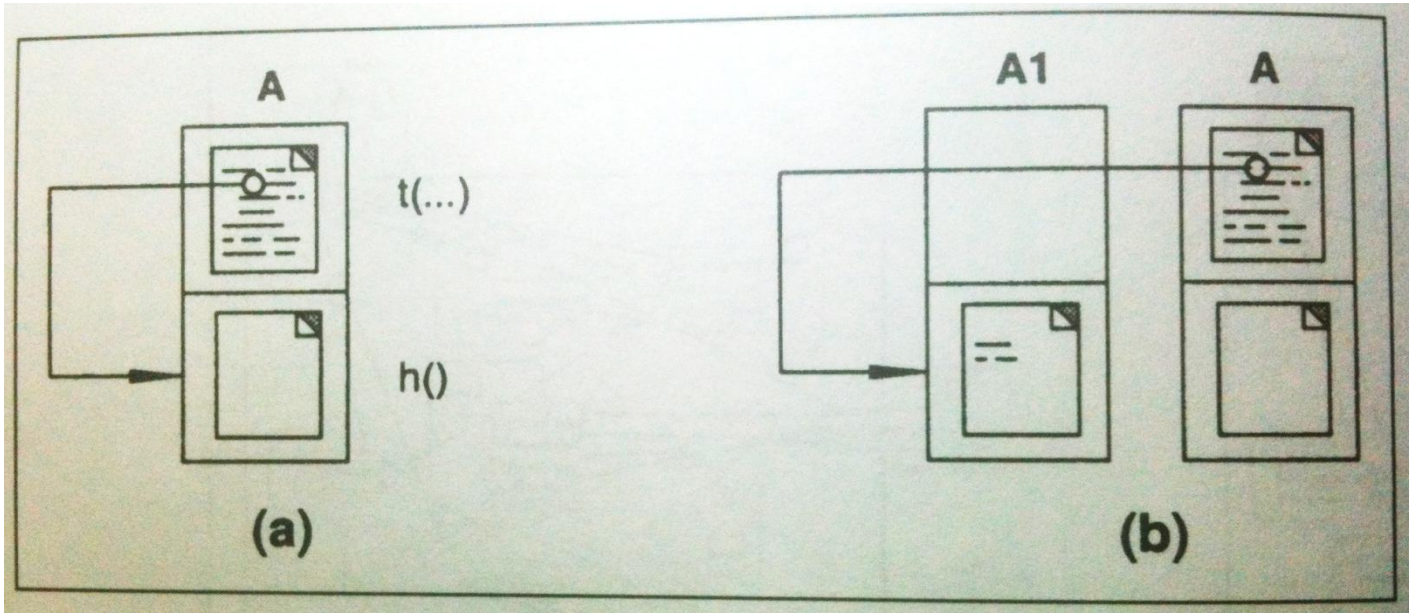
- Hooks são usados para para modificar o comportamento de um objeto sem tocar no código fonte da classe correspondente

- Isto é feito sobrescrevendo os métodos hook

Hooks como elementos construtores de Hot-spots

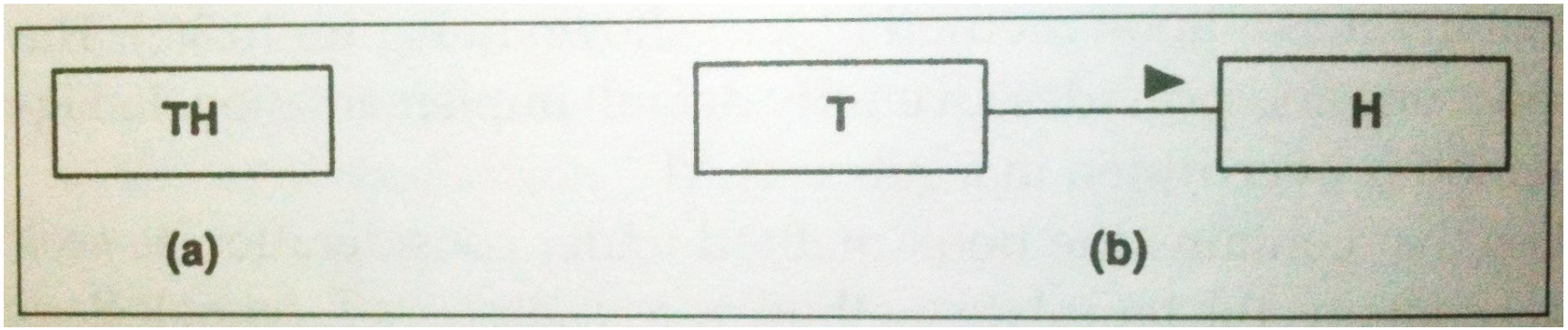
- O método t() é um template method que invoca o hook h() abstrato que provê uma implementação vazia (Figura (a))

- Na figura (b) o hook é sobrescrito na subclasse A1



Unificação vs Separação

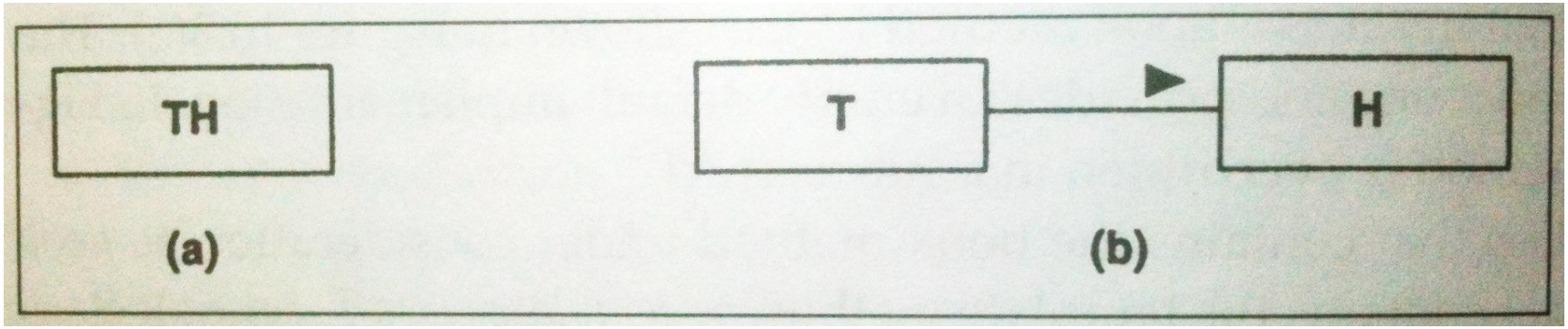
- Denominaremos H a classe que contém o hook method h()
- Denominaremos T a classe que contém o template method t()



Unificação vs Separação

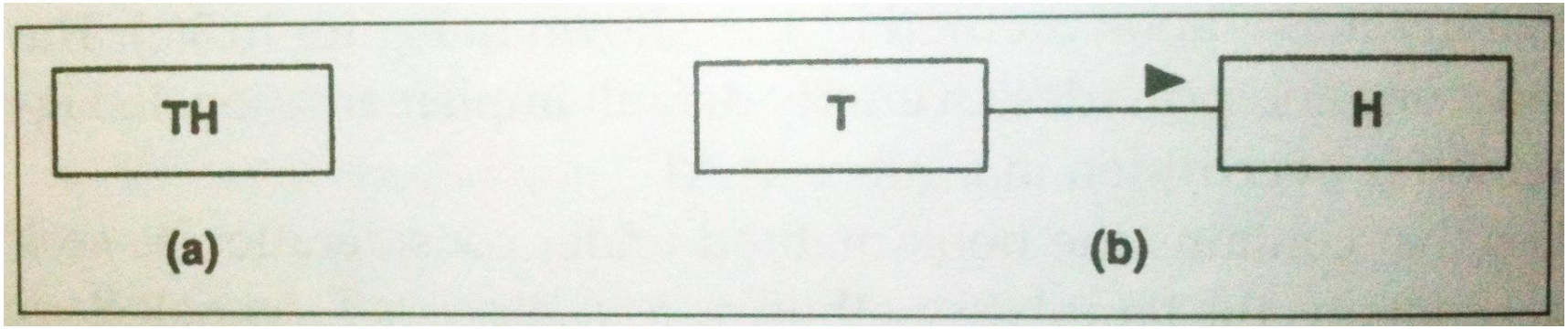
- Se as classes T e H são unificadas em um única, então a adaptação (especialização) só pode ocorrer via herança

- Separar as classes T e H é equivalente a acoplar (abstratamente) objetos destas classes de tal modo que o comportamento de T possa ser modificado por composição, isto é, plugando objetos específicos H



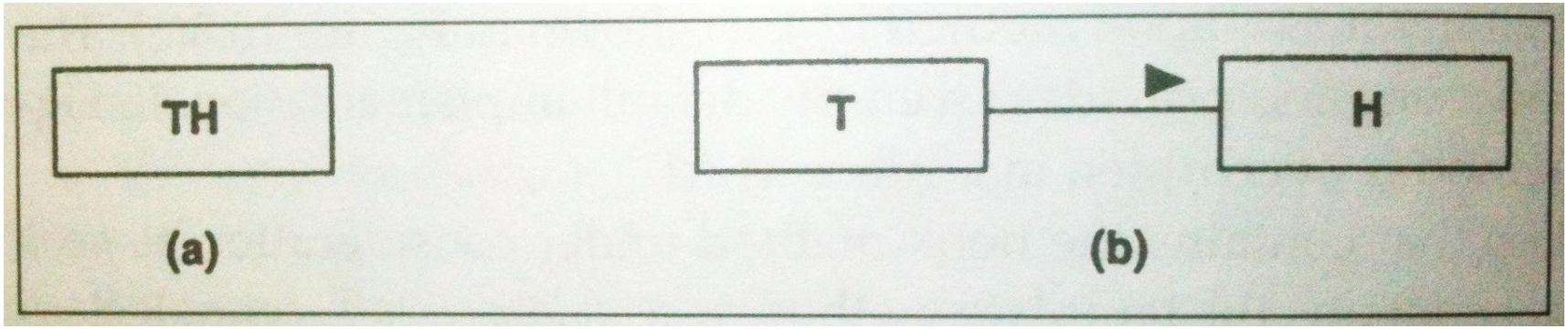
Unificação vs Separação

- A associação direcional entre T e H indica que um objeto de T está relacionado um objeto de H
- T deve enviar mensagens para H para invocar os hook methods



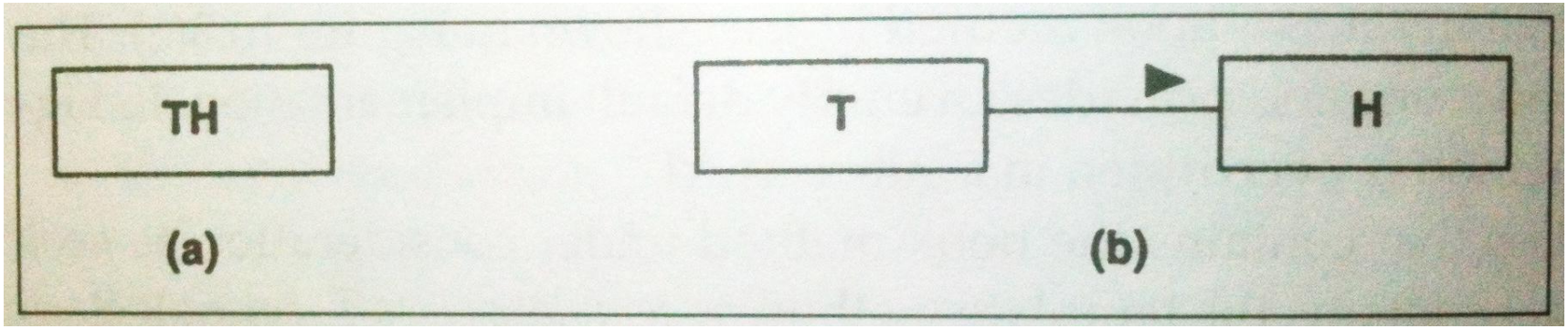
Unificação vs Separação

- Como implementar a associação entre T e H?
- Mantendo em T uma variável de instância do tipo de H
- Outra possibilidade é usar variáveis globais (ruim) ou passar uma referência para H temporariamente como parâmetro para um método de T



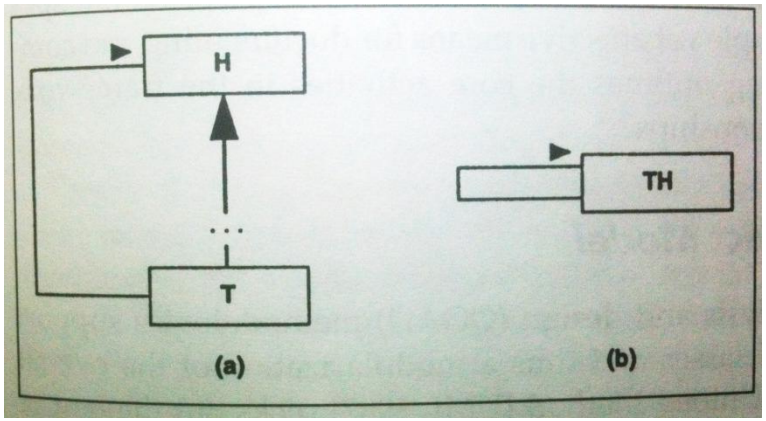
Unificação vs Separação

- A separação entre T e H é uma precondição para adaptação em tempo de execução
- Subclasses de H são definidas, instanciadas e plugadas em objeto de tipo T enquanto a aplicação está em execução



Padrões de combinação recursiva

- Uma classe template T pode ser filha de uma classe hook H
- Neste caso, T como filha de H pode gerenciar outra instancia de T
- Tal padrão é denominado composições recursivas
- Composições recursivas permitem criar grafos direcionados de objetos interconectados que podem ser plugados em uma instância de uma classe template T



Hooks como padrões de projetos

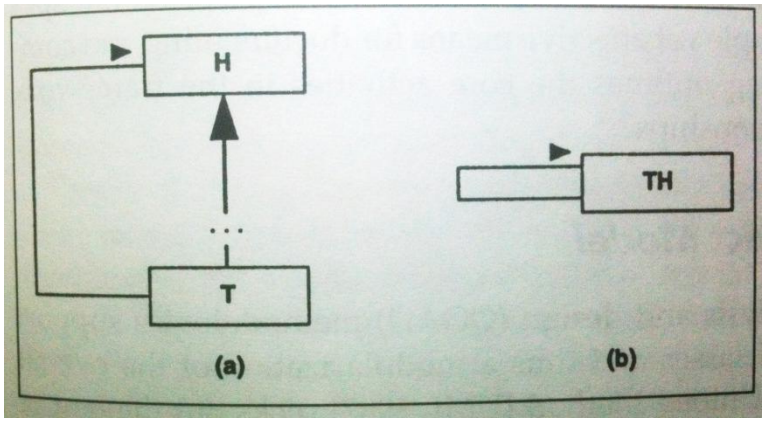
- Muitos padrões de projetos (quase todos) tipificam semânticas para hooks
- Eles representam como implementar sub-sistemas de hot-spots
- Alguns se baseiam no principio de construção por separação: Abstract Factory, Builder, Command, Interpreter, Observer, Prototype, State e Strategy
- Outros em ambos padrões de construção unificação e separação: Template Method e Bridge
- A semântica é expressa tipicamente no nome do hook method (por exemplo, no command, o método é denominado execute())

Multiplicidades no padrão de construção por separação e padrões de projeto

- Podemos enxergar o padrão Composite como o padrão da figura (a) onde existe uma relação 1:* entre T e H

- Da mesma forma o padrão Decorator é a construção da figura (a) com relação 1:1 entre T e H

- Observando as características do hot-spot podemos escolher um padrão de projeto adequado



Processo de desenvolvimento baseado em Hot-spots

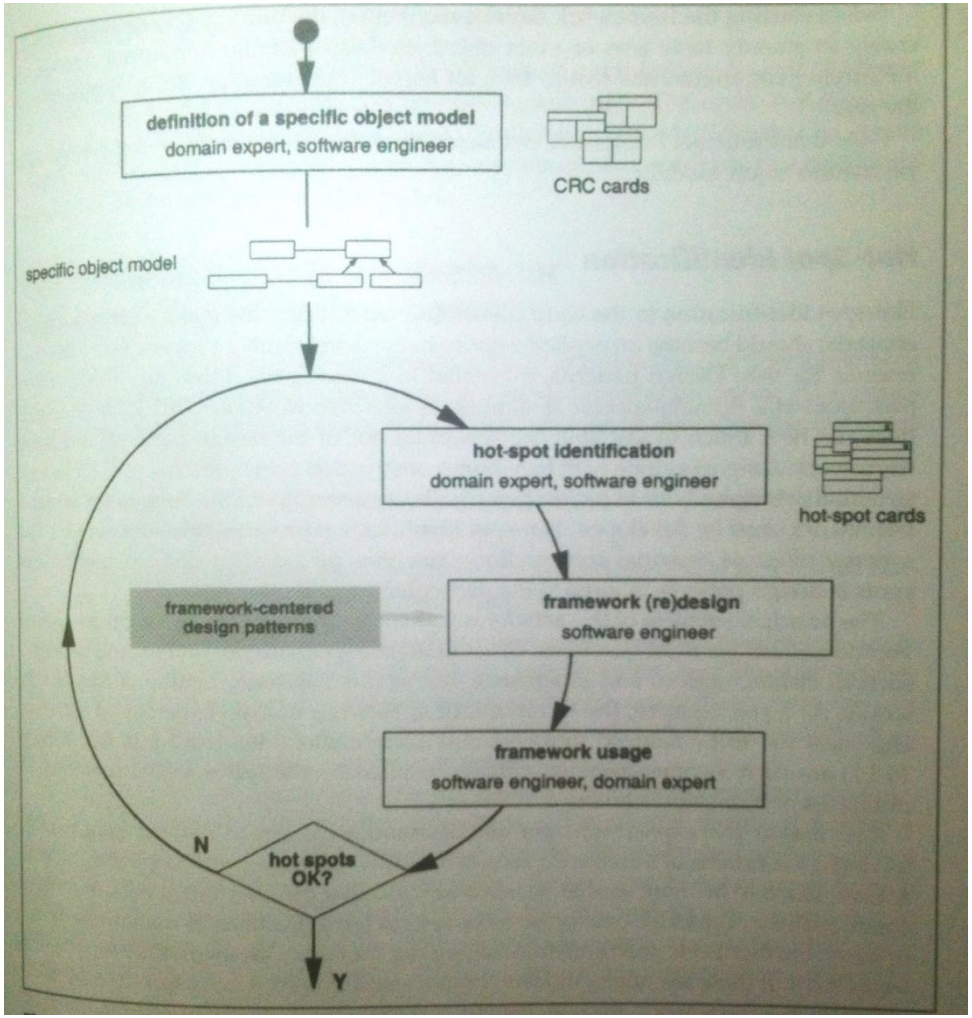
- O termo desenvolvimento de um framework expressa tanto o projeto inicial como a evolução do framework
- A qualidade de um framework é diretamente relacionada a flexibilidade requerida pelo domínio
- Logo, identificar hot-spots é parte crucial do processo

Processo de desenvolvimento baseado em Hot-spots

- Etapas:

1. Definição de um modelo orientado a objetos específico (especialista do domínio e engenheiro de software) UML
2. Identificação de hot-spots (especialista do domínio e engenheiro de software)
3. (Re)Projeto do Framework (engenheiro de software)
4. Uso do framework (especialista do domínio e engenheiro de software)
5. Se os hot-spots não estão adequados vá para 2
6. Fim

Processo de desenvolvimento baseado em Hot-spots



Definição do modelo de objetos

- A identificação das classes e objetos e suas relações e colaborações pode ser feita usando UML
- Inicialmente, é necessário adquirir o conhecimento específico do domínio do problema
- Os engenheiros de software interagem com os especialistas do domínio para tal fim
- No caso da modelagem de frameworks, ajuda muito ter se possível dois ou mais modelos O.O de duas aplicações similares a fim de identificar similaridades

Identificação de Hot-spots

- Identificar hot-spots cedo no processo de desenvolvimento de frameworks é muito importante
- A razão para isso é a de que as hot-spots irão indicar que padrão de construção e que padrões de projeto serão utilizados para implementar a especialização das classes abstratas que compõem o framework
- Identificar hot-spots consiste em identificar pontos de flexibilização do domínio de aplicação
- Isto deve ser feito em conjunto com especialistas do domínio

Identificação de Hot-spots

- Em geral, especialistas de domínio não conhecem conceitos sobre classes, objetos e frameworks
- É necessário utilizar uma ferramenta parque os especialistas juntamente como os engenheiros de software possam se comunicar para identificar os hot-spots
- Uma dessas ferramentas são os *hot-spot cards*

Reprojetar o framework

- Após identificar os hot-spots, os engenheiros de software devem revisitar e modificar o modelo para ganhar a flexibilidade dos hot-spots
- Padrões de construção de frameworks (unificação e separação das classes template e hook) ajudam nessa atividade

Uso do framework

- Somente o uso do framework é que vai apontar suas fraquezas
- Isso leva a um processo de especialização que ocorre diversas vezes
- Logo, o desenvolvimento de um framework pode requer um número indeterminado de iterações do processo de desenvolvimento

Hot-spots cards para captura de requisitos de flexibilização

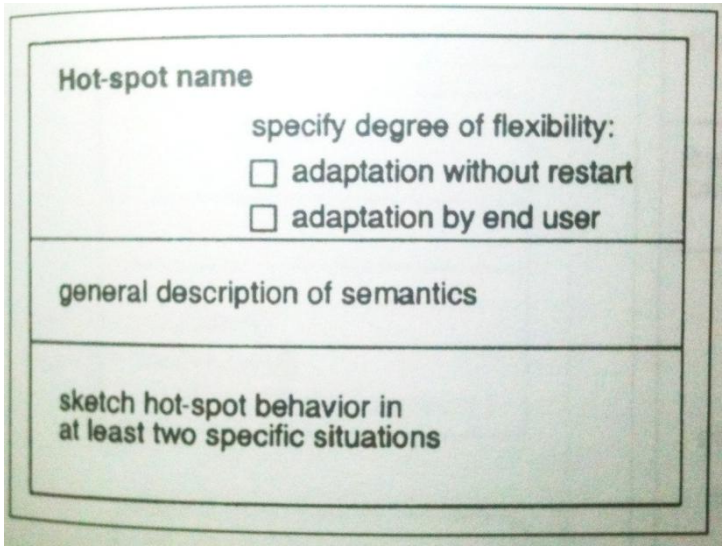
- Especialistas de domínio podem facilmente pensar em termos de funcionalidade de software
- Eles também sabem como tais funcionalidades suportam vários processos do negócio
- Deve-se efetuar as seguintes questões aos especialistas:
  - Que aspectos diferem de uma aplicação para aplicação no domínio de interesse? A resposta é uma lista de hot-spots
  - Qual o grau de flexibilidade desejado para estes hot-spots. O comportamento deve ser modificado em tempo de execução, e/ou pelo usuário final?

Hot-spots cards para captura de requisitos de flexibilização

- Na maioria das vezes, tais questionamentos falham
- Nestes casos, os engenheiros devem se abstrair de cenários particulares e procurar por similaridades e requisitos de hot-spot
- Hot-spot cards são uma ferramenta para iterativamente identificar hot-spots

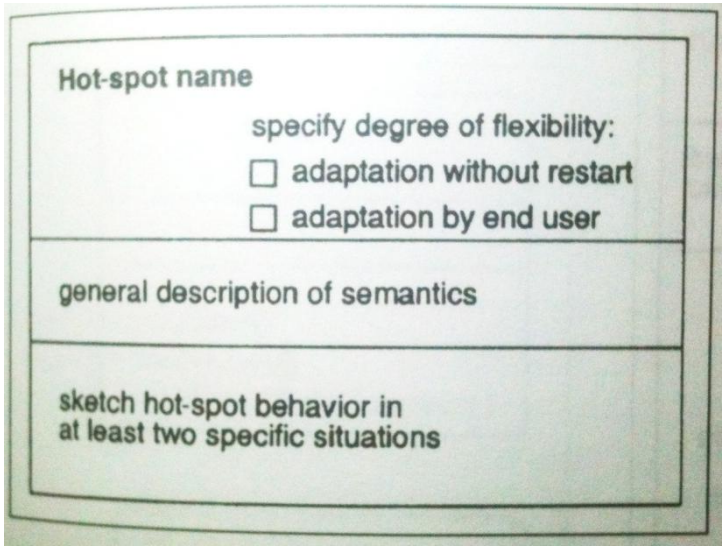
Hot-spots cards para captura de requisitos de flexibilização

- Um hot-spot card possui:
  - Nome
  - Descrição resumida da funcionalidade
  - Grau de flexibilidade
  - Esboço do comportamento para dois casos específicos, pelo menos



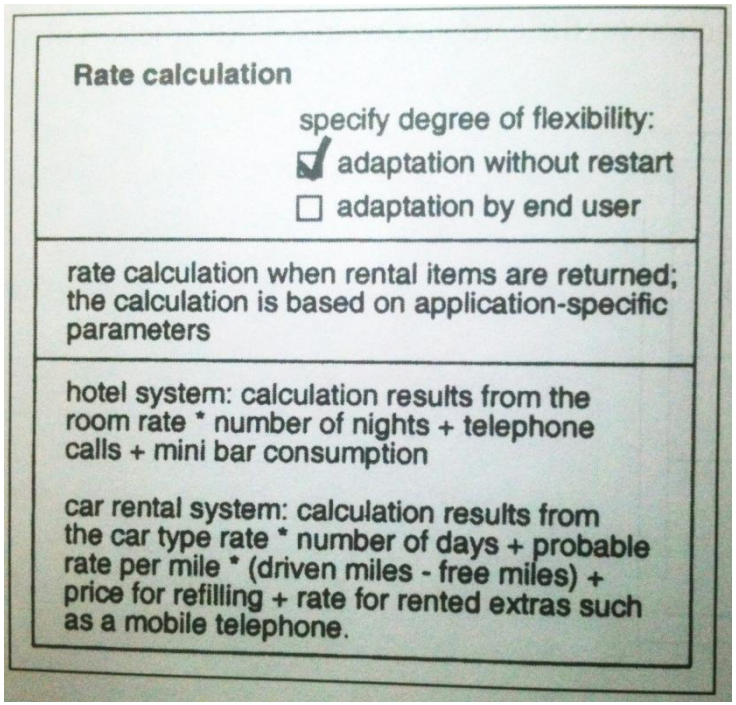
Hot-spots cards para captura de requisitos de flexibilização

- Um hot-spot card possui:
  - Nome
  - Descrição resumida da funcionalidade
  - Grau de flexibilidade
  - Esboço do comportamento para dois casos específicos, pelo menos



Hot-spots cards e padrões essenciais de construção

- Exemplo para um framework para sistemas de software para aluguel (veículos, imóveis, etc.)
- O cálculo da taxa ao retornar o item é uma funcionalidade que deve ser flexibilizada (hot-spot)



Rate calculation

specify degree of flexibility:
- ☑ adaptation without restart
- ☐ adaptation by end user

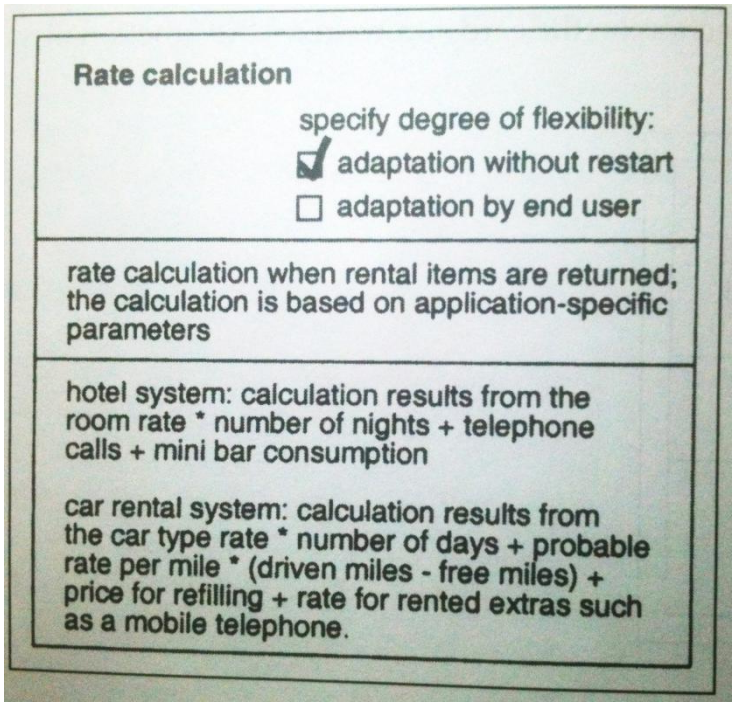
rate calculation when rental items are returned; the calculation is based on application-specific parameters

hotel system: calculation results from the room rate * number of nights + telephone calls + mini bar consumption

car rental system: calculation results from the car type rate * number of days + probable rate per mile * (driven miles - free miles) + price for refilling + rate for rented extras such as a mobile telephone.

Hot-spots cards e padrões essenciais de construção

- Hot-spots funcionais correspondem proximamente a hook methods e hook classes
- Um hot-spot card contém informação sobre a semântica do hot-spot e seu grau de flexibilidade
- Porém nãoindica diretamente em que classe/subsistema o hot-spot pertence



Hot-spots cards e padrões essenciais de construção

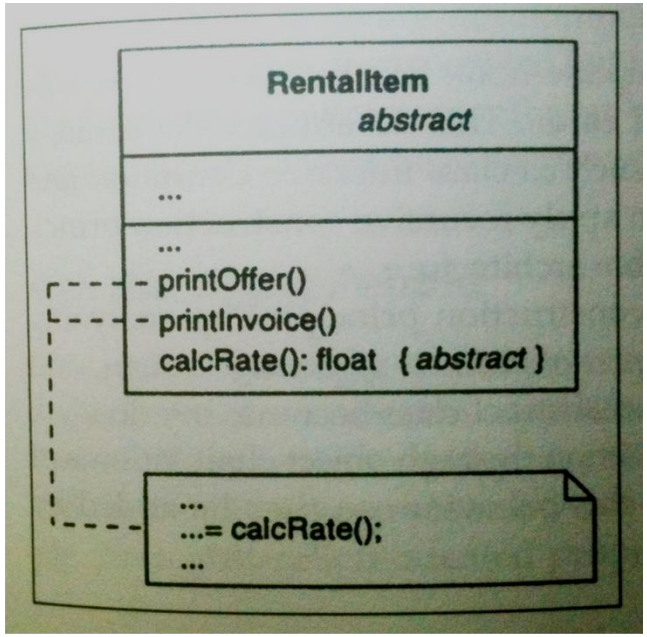
- A granularidade de um hot-spot pode indicar se um ou mais métodos devem ser associados a uma classe template de forma unificada ou separada
- Um hot-spot que possui somente um ponto de variação é dito elementar e não pode ser decomposto em unidades menores
- Dois hot-spots são ortogonais se as alternativas de um aspecto podem ser trocadas independentemente de outro aspecto

Hot-spots cards e padrões essenciais de construção

- Existe uma relação entre o grau de flexibilidade do hot-spot e padrões de construção
- Se nenhuma das checkboxs do hot-spot card (sem reinicilização e sem controle do usuário final) forem selecionadas então apenas um método hook extra deve ser incluído na classe apropriada
- A semântica do hot-spot é em geral suficiente para indicar em que classe o método hook deve ser adicionado

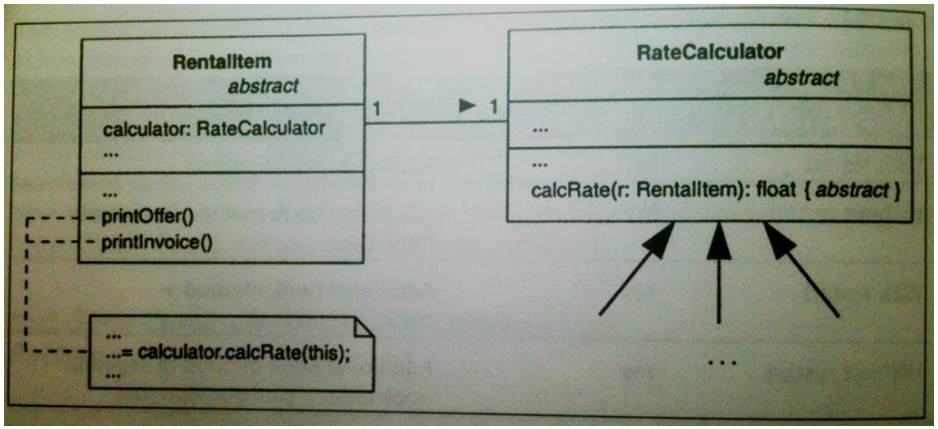
Hot-spots cards e padrões essenciais de construção

- Suponha a classe abstrata RentalItem especificada no modelo O.O do problema
- O hot-spot Rate Calculation da origem ao hook calcRate() em RentalItem
- Os métodos printOffer() e printInvoice() são métodos template existentes na mesma classe RentalItem



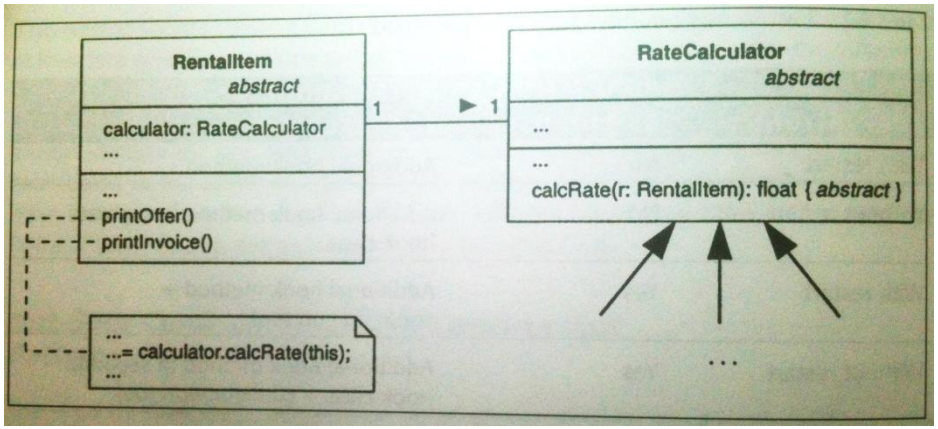
Hot-spots cards e padrões essenciais de construção

- Se a adaptação deve ocorrer sem reiniciar a aplicação então deve-se usar o padrão de construção por separação
- O método calcRate() deve ser colocado em outra classe RateCalculator. Instâncias da nova classe são plugadas a RentalItem para mudar seu comportamento
- Que padrão de projeto é este?????



Hot-spots cards e padrões essenciais de construção

- O modo como os objetos de RateCalculator pode ser alternados determina se o usuário pode ou não fazer a alteração
- Uma possibilidade é armazenar a informação a respeito de motor de calculo de taxa em um arquivo de configuração
- Outra possibilidade é criar uma ferramenta de configuração na interface com o usuário



Hot-spots cards e padrões essenciais de construção

Table 16.1 Transformation Rules for a Hot-Spot Card

ADAPTATION . . .	ADAPTATION BY END USER	OBJECT MODEL TRANSFORMATION
With restart	No	Additional hook method
Without restart	No	Additional hook method in separate hook class
With restart	Yes	Additional hook method + configuration tool
Without restart	Yes	Additional hook method in separate hook class + configuration tool

Combinações Template-Hook recursivas

- Hot-spot cards levam a padrões de unificação/separação das classes template e hook
- Entretanto, não dizem nada sobre o uso de padrões de construção recursivos
- O modelo de objetos entretanto pode dar dicas sobre a possibilidade de uso de recursividade
- Relações como parte-de, consiste-de, gerencia, é-dono-de, por exemplo, apontam para o padrão Composite
- Outra situação é aquela em que a classe abstrata fica sobrecarregada. Neste caso o uso do padrão Chain-of-responsability é uma possibilidade

Dicas para mineração de hot-spots

- Examinar o processo de manutenção de software legado ou software existente a partir do qual se gerará uma nova versão
- Investigar cenários e casos de uso. Casos de uso, quando em bom número, permitem identificar semelhanças e pontos de variação
- Questionar as pessoas corretas (com bom poder de abstração)

Padrões de Projeto

Bibliografia

- Mohamed E. Fayad (Editor), Douglas C. Schmidt (Editor), Ralph E. Johnson (Editor). Building Application Frameworks: Object-Oriented Foundations of Framework Design. Wiley; 1 edition (September 13, 1999). Capítulos 1, 15 e 16.

- As imagens dessa apresentação foram obtidas do livro acima.