

FRAMEWORK

Problema:

- Construir a simulação do comportamento de caixas de supermercado.

Problema:

- Construir a simulação do comportamento de caixas de supermercado.

```
for( ; time < fim ; ) {
    e = lista.getPrimeiro();
    time = e.getTime();
    switch( e.evento ) {
        case SIT_1 : trataSit_1();
                    geranovoevento();
                    break;
        case SIT_2: trataSit_2();
                    geranovoevento();
                    break;
        ....
    }
}
```

Problema:

- Construir a simulação do comportamento de caixas de supermercado.

```
for( ; time < fim ; ) {
    e = lista.getPrimeiro();
    time = e.getTime();
    e.executa();
    e.geraNovoEvento();
}
```

```
class evento {
    ...
    virtual void executa() = 0;
    virtual void geraNovoEvento();
}
```

Motivação

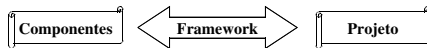
- **Reutilização de código**
 - Economia de tempo e dinheiro
- **Reutilização é uma tarefa complexa e proporcional ao tamanho do código a ser reutilizado**
- **O POO oferece recursos para implementação de frameworks**
- **Vantagens**
 - ganho de tempo e credibilidade no código

O que é um framework

- “Um framework é um conjunto de classes que constitui um design abstrato para soluções de uma família de problemas” – Johnson e Foote – 1988
- “Um framework é um conjunto de objetos que colaboram com o objetivo de cumprir um conjunto de responsabilidades para uma aplicação ou um domínio de um subsistema.” – Johnson – 1991
- “Uma arquitetura desenvolvida com o objetivo de se obter a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização.” - Mattsson – 1996

O que é um framework

- **Frameworks:**
 - estão exatamente no meio do conceito de técnicas de reuso. São mais abstratos e flexíveis que **componentes** mas mais concretos e mais fáceis de reutilizar que apenas uma **informação de projeto**
 - Mais complexos que componentes
 - Menos flexível e menos provável de ser aplicável a casos gerais



O que é um framework?

- **Inversão de fluxo**
 - O framework tenta capturar o fluxo de controle de aplicação dentro de um domínio. Este fluxo pode ser especializado mais tarde por uma aplicação específica.
 - **Caso normal:** em uma aplicação o programador define quando cada serviço deve ser chamado.
 - **Framework:** o algoritmo principal é dado pela estrutura do framework, sendo este o responsável por invocar serviços.

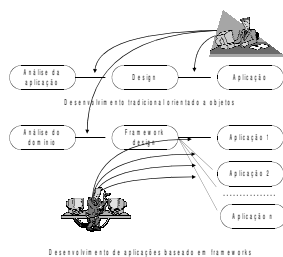
O que é um framework?

- **Inversão de fluxo**
 - O framework tenta capturar o fluxo de controle de aplicação dentro de um domínio. Este fluxo pode ser especializado mais tarde por uma aplicação específica
- **Genéricos dentro de um domínio**
 - Os frameworks são adaptáveis a classes de aplicações por customização e extensão da estrutura que eles provêem

O que é um framework?

- **Pontos de flexibilização: HOT SPOTS**
 - Os hot spots expressam aspectos do domínio do framework que não podem ser antecipados, ou seja, são dependentes da aplicação.
 - Não é função dos desenvolvedores de um framework compor os hot spots cobertos no domínio das aplicações.
- A flexibilização permite o uso do framework para o desenvolvimento de inúmeras aplicações no domínio modelado pelo framework

Desenvolvimento de framework



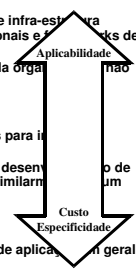
Objetivo

- Manter o conhecimento da organização sobre o domínio da aplicação na organização.
- Reutilização
- Otimizar generalização e a especificação

Benefícios

- **Modularidade:** encapsulamento de implementações flexíveis através de uma interface estável
 - ajuda prover qualidade de software localizando o impacto das mudanças no design e na implementação
 - esta localização reduz o esforço necessário para compreensão e manutenção de software
- **Reuso:** as interfaces estáveis propiciam reuso pois definem componentes genéricos que podem ser reaplicáveis para criar novas aplicações
- **Extensibilidade:** uma instância do framework é uma nova aplicação desenvolvida

Classificação por escopo

- **System infrastructure frameworks:**
 - simplificam o desenvolvimento de sistemas de infra-estrutura para portáteis e eficientes como sistemas operacionais e frameworks de comunicação.
 - são usados internamente como um software da organização e não são vendidos para clientes diretamente
 - **Middleware integration frameworks:**
 - estes frameworks são frequentemente usados para integrar aplicações distribuídas e componentes
 - são projetados para aumentar a habilidade de desenvolver software de infra-estrutura para trabalharem similarmente em um ambiente distribuído
 - **Enterprise application frameworks:**
 - estes frameworks são voltados para domínio de aplicações em geral e são primordiais para atividades de negócio
- 
- Um diagrama de classificação por escopo. No topo, uma seta apontando para cima indica 'Aplicabilidade' e 'Não'. No fundo, uma seta apontando para baixo indica 'Custo' e 'Especificidade'.

Classificação por técnicas de extensão

- **Whitebox:**
 - Desenvolvimento de funcionalidades. Baseado em linguagens OO como herança (reuso de código) e amarração dinâmica (extensão).
 - Desenvolvedor vê tudo.
- **Blackbox:**
 - Uso de componentes prontos. Extensão pela definição interfaces para componentes que podem ser conectados ao framework.
 - O reuso das funcionalidades são feitas: 1) definindo os componentes que podem se adaptar a interface e 2) integrando estes componentes.
 - Desenvolvedor não vê nada.
- **Graybox:**
 - criado para evitar as desvantagens presentes nos frameworks whitebox e blackbox. Um bom graybox framework tem bastante flexibilidade e extensibilidade e também tem a habilidade de esconder informações não necessárias para desenvolvedores da aplicação.

Vantagens

- **Código**
 - Menor (na aplicação)
 - Parcialmente desenvolvido
 - Confiável (depurado)
 - Manutenibilidade
- **Reutilização:**
 - Código
 - Design

Desvantagens

- **Modelagem:** necessidade de experiência no domínio da aplicação.
- **Desenvolvimento de documentação** para o usuário do framework
- **Garantir integridade de versões**
 - Frameworks amadurecem, e as aplicações?...
- **O processo de depuração** pode se tornar complicado
 - (Porém uma vez feito o código esta pronto)
- **Flexibilidade vs. Generalização**
 - Compromisso com desempenho

Vantagens e Desvantagens

- **Vantagens:**
 - Pouco esforço de desenvolvimento da aplicação
 - Integração
 - Eficiência (para casos genéricos)
 - Uma vez desenvolvido o núcleo básico facilidade de desenvolvimento de aplicações
- **Desvantagem:**
 - Curva de aprendizagem
 - Pouca padronização
 - Manutenção do framework e da aplicação
 - Pontapé inicial demorado

Framework vs. Biblioteca de classes

- **Biblioteca de classes**
 - São um conjunto de classes relacionadas que tem funcionalidades de propósito geral
 - Suas classes não são relacionadas a um domínio de aplicação específica
 - Classes de bibliotecas podem ser utilizadas individualmente
- **Framework**
 - As classes são relacionadas entre si e as instâncias se dão em conjunto (ou de forma coordenada)

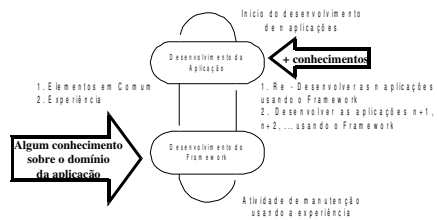
Framework vs. Programa OO

- **Programa OO:**
 - Descreve a execução de um programa que atende a todos os requisitos de uma especificação.
- **Framework:**
 - Captura as funcionalidades de diversas aplicações no domínio, mas não é executável, já que não cobre o comportamento de uma aplicação específica

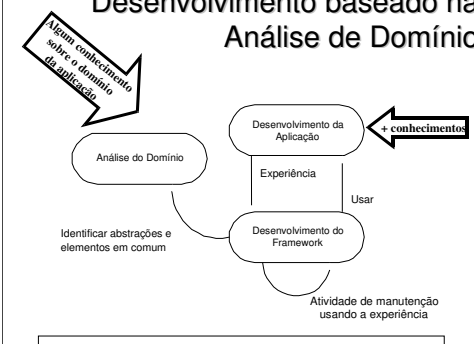
Processo de desenvolvimento

- **O desenvolvimento de um framework pode ser baseado:**
 - Na experiência dos desenvolvedores
 - Na compreensão dos problemas do domínio de aplicação
 - Empregando *design patterns*

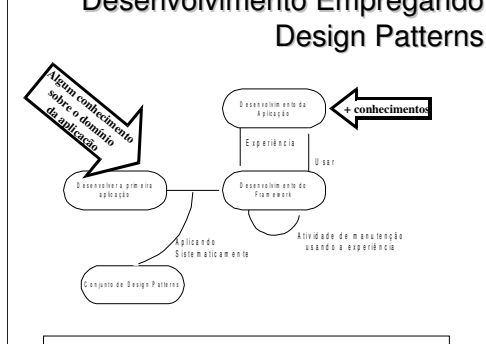
Desenvolvimento baseado na experiência



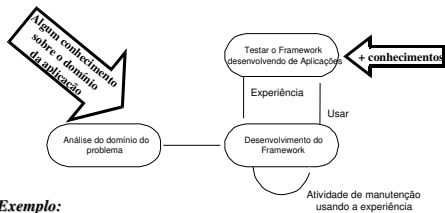
Desenvolvimento baseado na Análise de Domínio



Desenvolvimento Empregando Design Patterns



Desenvolvimento: Caso Geral



Exemplo:

Gustavo Lermen: TC
Desenvolvimento de um framework com concorrência para alinhamento de seqüências de DNA

Diretrizes de desenvolvimento

- Evitar todo esforço possível do usuário
 - O usuário, no caso é a pessoa responsável por desenvolver aplicações
 - Idealmente o usuário é especialista na aplicação, talvez não tanto em programação....
- Identificar quais as classes e operações do framework o usuário terá que utilizar
- A quantidade de código a ser escrita pelo usuário deve ser a mínima possível

Instanciação

- Conectando componentes já existentes backbox
 - reutiliza a interface do framework
 - reutiliza regras para a conexão dos componentes
- Criando novas sub-classes concretas whitebox
 - as sub-classes são bem acopladas à super-classe
 - é necessário ter um maior conhecimento das classes abstratas
- Estendendo as classes abstratas
 - adicionando novas operações e variáveis
 - maneira mais difícil porém mais poderosa do uso de frameworks

Documentação

- Documentação deve se adaptar a diferentes públicos
 - **Portfolio**
 - descrição as capacidades do framework e sua aplicabilidade.
 - **Guia de Usuário.**
 - guia para exploração do framework, explicitando hot spots e comportamento básico da execução. Descrição das semânticas esperadas aos hot spots.
 - Exemplos
 - **Desenvolvimento**
 - descrição elaborada, contendo os algoritmo abstratos e o modelo de colaboração possibilitando a manutenção do próprio framework.

Documentação

| | Propósito do Framework | Como Utilizar | Aplicações de Exemplo | Design do Framework |
|----------------|------------------------|---------------|-----------------------|---------------------|
| Decisão de Uso | X | - | - | - |
| Utilização | X | X | X | - |
| Manutenção | X | X | X | X |

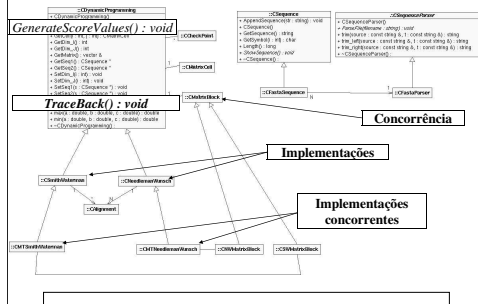
Conclusão

- É importante ter em mente:
 - o domínio do problema que um framework endereça
 - a estrutura interna de um framework
 - como o framework deve ser usado
- “Quanto mais customizado um componente é, mais ele se adapta a uma situação em particular mas mais esforço é necessário para aprender a usá-lo e para usá-lo efetivamente.”
- **Compromisso**

Exemplo Sequenciamento de DNA

- Definir uma estrutura para suporte ao sequenciamento de DNA

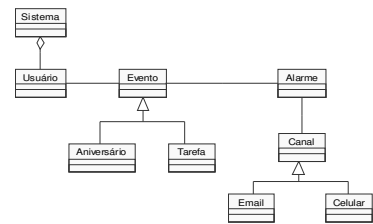
Exemplo Sequenciamento de DNA



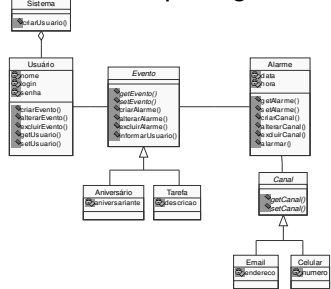
Exemplo Agenda Eletrônica

- Enunciado:
 - O sistema de agenda pode ter vários usuários
 - Os usuários selecionam o tipo de evento para o qual ele deseja ser alarmado
 - Ao selecionar um evento o usuário escolhe o alarme(data e hora) que deseja e como ele deve ser avisado
 - O usuário é avisado através dos canais de comunicação do sistema com o usuário
- Ex. de evento: tarefa, aniversário
- Ex. de canal: celular, email

Exemplo Agenda Eletrônica



Exemplo Agenda Eletrônica



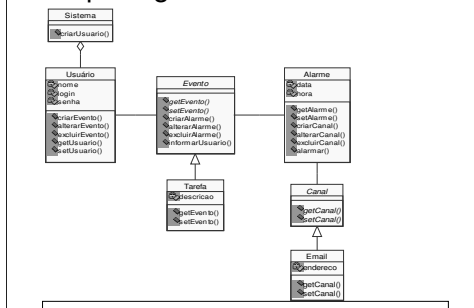
Exemplo Agenda Eletrônica

- Whitebox**
- Processo de desenvolvimento:**
 - baseado na análise do domínio e na análise de aplicações já existentes
- Instanciação:**
 - estendendo as classes abstratas: adicionando novas operações e variáveis
- Documentação:**
 - Propósito do framework:
 - "O framework se destina à aplicações no domínio de agenda eletrônica. Tais aplicações são usadas para acionar um alarme e avisar ao usuário da aplicação sobre um evento quando chegar a hora e a data desejada."

Exemplo Agenda Eletrônica

- Como utilizar o framework:
 - "Para utilizar o framework é necessário instanciar as classes abstratas *Evento* e *Canal* e implementar alguns métodos (métodos abstratos e outros caso necessário) e adicionar algum atributo para armazenar a informação recebida. Além disso é necessário modificar a implementação dos métodos *criarEvento()* da classe *Usuário* e *criarCanal()* da classe *Alarme*."
- Propósito das aplicações: Uma aplicação exemplo
 - criação da classe *Tarefa* por herança a partir da classe *Evento*
 - criação do atributo *descricao* para a classe *Tarefa*
 - implementação dos métodos *setEvento* e *getEvento*
 - criação da classe *Email* por herança a partir da classe *Canal*
 - criação do atributo *endereco* para a classe *Email*
 - implementação dos métodos *getCanal* e *setCanal*

Exemplo Agenda Eletrônica



Exemplo Agenda Eletrônica

- Design do framework:

