Universidade Estadual de Maringá (UEM)

Departamento de Informática (DIN)

Introdução a Frameworks

Prof. Dr. Edson A. Oliveira Junior

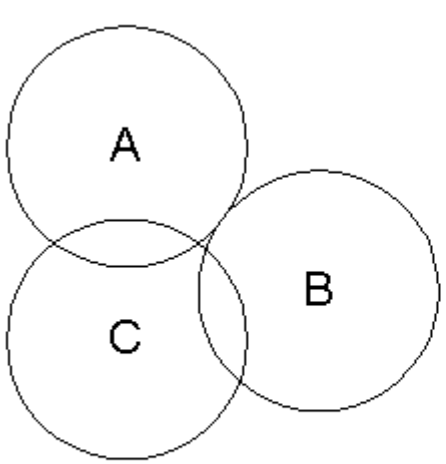
edson@din.uem.br

Roteiro

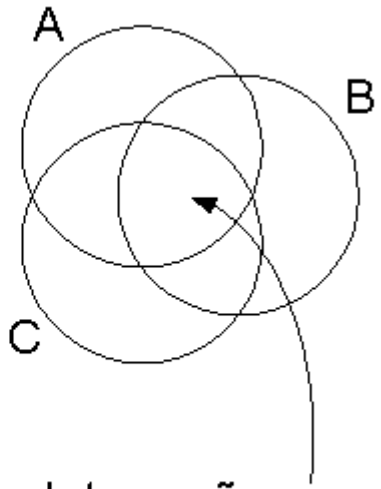
- O que é um framework?
- Características de um framework
- Frameworks x bibliotecas OO
- Padrões x Frameworks
- Tipos de frameworks
- Classificação de frameworks

O que é um Framework?

- Um framework captura a funcionalidade comum a várias aplicações
- As aplicações devem ter algo razoavelmente grande em comum: pertencem a um mesmo domínio de problema



Impossível criar Framework



Interseção grande
Possível criar Framework

O que é um Framework?

- Um framework é uma aplicação quase completa, mas com pedaços faltando
- Em um framework o trabalho consiste em prover os pedaços que são específicos para sua aplicação.

O que é um Framework?

- Quatro características principais de um framework (Orientado a Objeto):
  - Um framework provê uma solução para uma família de problemas semelhantes
  - Usa um conjunto de classes e interfaces que mostra como decompor a família de problemas
  - E como os objetos dessas classes colaboram para cumprir suas responsabilidades
  - O conjunto de classes deve ser flexível e extensível para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades de cada aplicação.

O que é um Framework?

- Projeto de software reutilizável que descreve como um sistema é decomposto em um conjunto de objetos ou componentes
- Orientado a objetos: classes abstratas e concretas com um relacionamento de colaboração, para ser estendidas em aplicações específicas.

Outras definições de Framework

- Projeto reusável de um sistema ou parte dele representado por um conjunto de classes abstratas que cooperam entre si.
- Esqueleto de uma aplicação que pode ser customizado por um desenvolvedor de aplicações.
- Conjunto de classes projetadas para trabalhar juntas e resolver um problema.
- Conjunto integrado de componentes de software de domínio específico que pode ser reusado para criar aplicações.

Características de um Framework

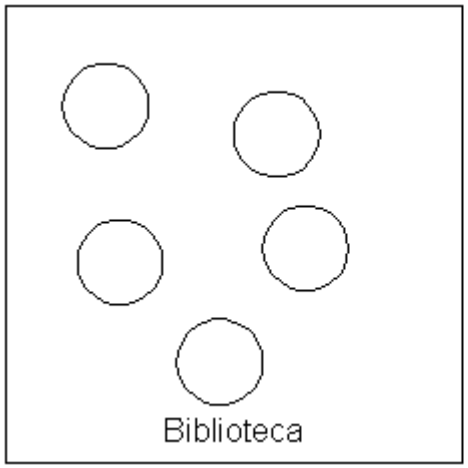
- Um framework deve ser reusável
  - Bem documentado
  - Fácil de usar
- Deve ser extensível
  - O framework contém funcionalidade abstrata (sem implementação) que deve ser completada.

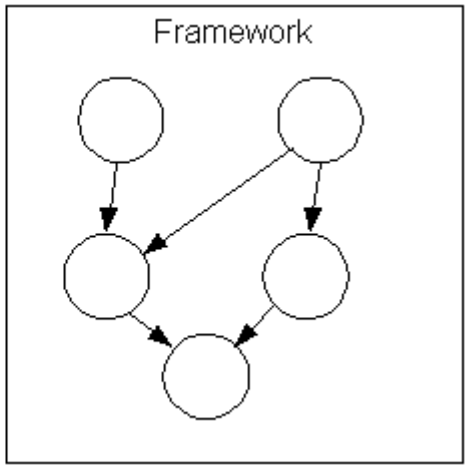
Características de um Framework

- Deve ser de uso seguro
  - O desenvolvedor de aplicações não pode destruir o framework
- Deve ser eficiente
  - Devido a seu uso em muitas situações, algumas das quais poderão necessitar de eficiência
- Deve ser completo
  - Para endereçar o domínio do problema pretendido

Framework x Bibliotecas

- Numa biblioteca de classes, cada classe é única e independente das outras
- Num framework, as dependências/colaborações estão embutidas (wired-in interconnections)
- Com biblioteca, as aplicações criam as colaborações

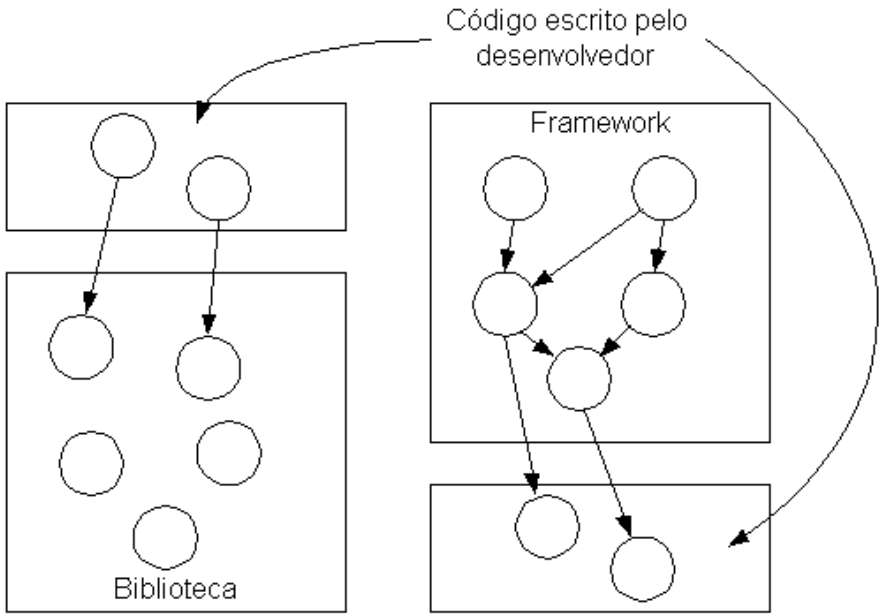




Framework x Bibliotecas

- Vê-se portanto que um framework impõe um modelo de colaboração (o resultado da análise e design) ao qual você deve se adaptar
  - Já que a comunicação entre objetos já está definida, o projetista de aplicações não precisa saber quando chamar cada método: é o framework que faz isso
- Não se pode embutir conhecimento do domínio (análise + design) numa biblioteca de classes
- O framework é usado de acordo com o Hollywood Principle ("Don't call us, we'll call you")
  - É o framework que chama o código da aplicação (que trata das particularidades dessa aplicação)
  - Framework = Upside-down library

Framework x Bibliotecas



- Exemplo do Hollywood Principle
  - Modelo de eventos em Java/AWT
  - AWT é um framework
  - No código abaixo, mouseClicked() e mousePressed() são chamados pelo framework (AWT)

Framework x Bibliotecas

```
public class MeuMouseListener implements MouseListener {
    public void mouseClicked(MouseEvent event) {
        ...
    }
    public void mousePressed(MouseEvent event) {
        ...
    }
    ...
}
...
    MeuMouseListener mouseListener = new MeuMouseListener();
    JButton meuBotão = new JButton("clique aqui");
    // O seguinte método estabelece a interação entre o objeto
    // meuBotão e o objeto mouseListener
    meuBotão.addMouseListener(mouseListener);
```

Framework x Bibliotecas

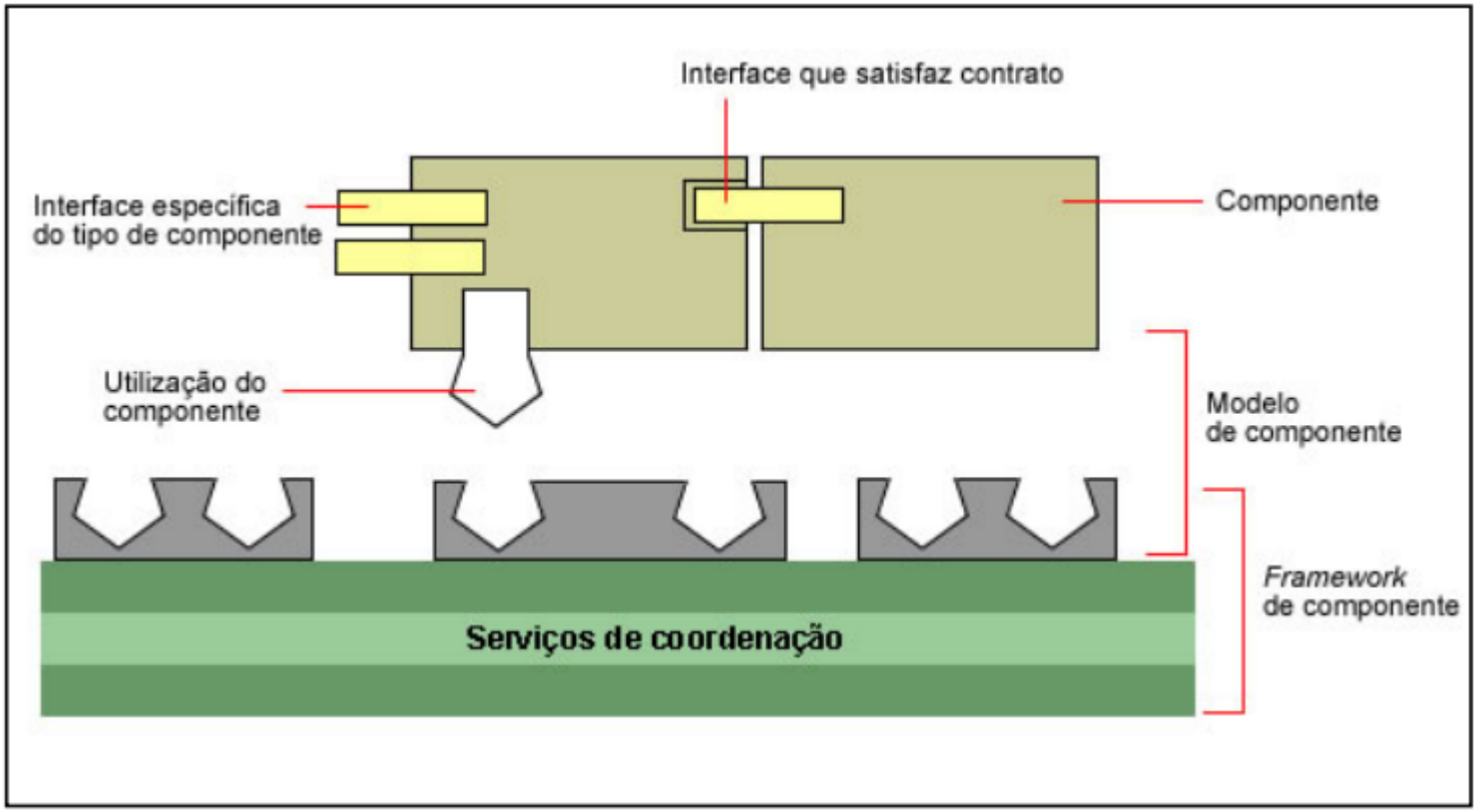
- Conjunto de classes instanciadas pelo cliente
- Cliente chama funções
- Fluxo de controle não pré-definido
- Interação não pré-definida
- Não tem comportamento "default"

- Cuida da personalização através de subclasses
- Chama funções do cliente
- Controla o fluxo de execução
- Define a interação dos objetos
- Tem comportamento "default"

Padrões x Frameworks

- Design patterns são mais abstratos do que frameworks
  - Um framework inclui código, um design pattern não (só um exemplo do uso de um pattern)
  - Devido à presença de código, um framework pode ser estudado a nível de código, executado, e reusado diretamente
- Design patterns são elementos arquiteturais menores do que frameworks
  - Um framework típico contém vários design patterns mas o contrário nunca ocorre
  - Exemplo: Design patterns são frequentemente usados para documentar frameworks
- Design patterns são menos especializados do que frameworks
  - Frameworks sempre têm um domínio de aplicação particular enquanto design patterns não ditam uma arquitetura de aplicação particular

Framework de componentes



Tipos de Framework

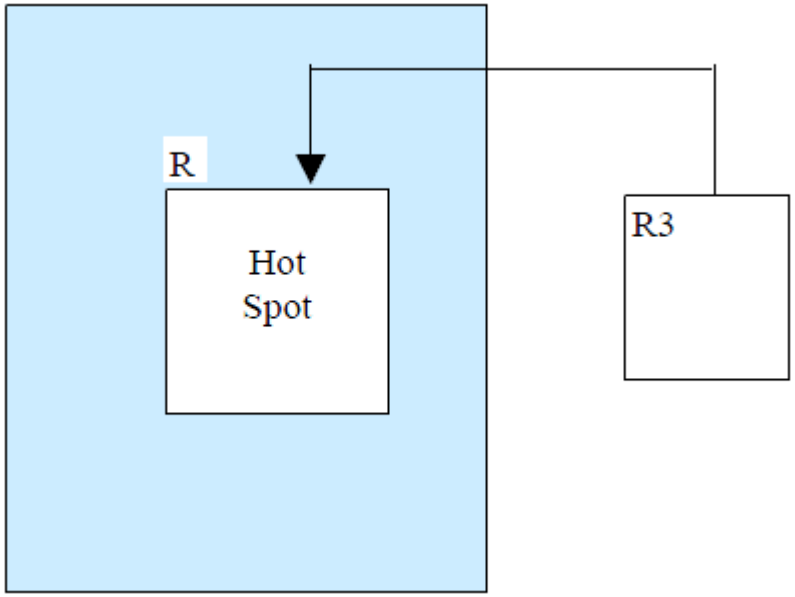
- Framework caixa branca:
  - Reuso por herança
  - Deve-se entender detalhes de como o framework funciona
- Framework caixa preta:
  - Reuso por composição
  - Deve-se entender apenas a interface do cliente

Tipos de Framework

- Aspectos variáveis de um domínio de aplicação
- Hot-Spots (pontos de especialização)
  - Representam as partes do framework de aplicação que são específicas de sistemas individuais
  - São projetados para serem genéricos - podem ser adaptados às necessidades da aplicação
- Frozen-Spots (pontos fixos)
  - Definem a arquitetura geral de um sistema de software - seus componentes básicos e os relacionamentos entre eles
  - Permanecem fixos em todas as instanciações do framework de aplicação

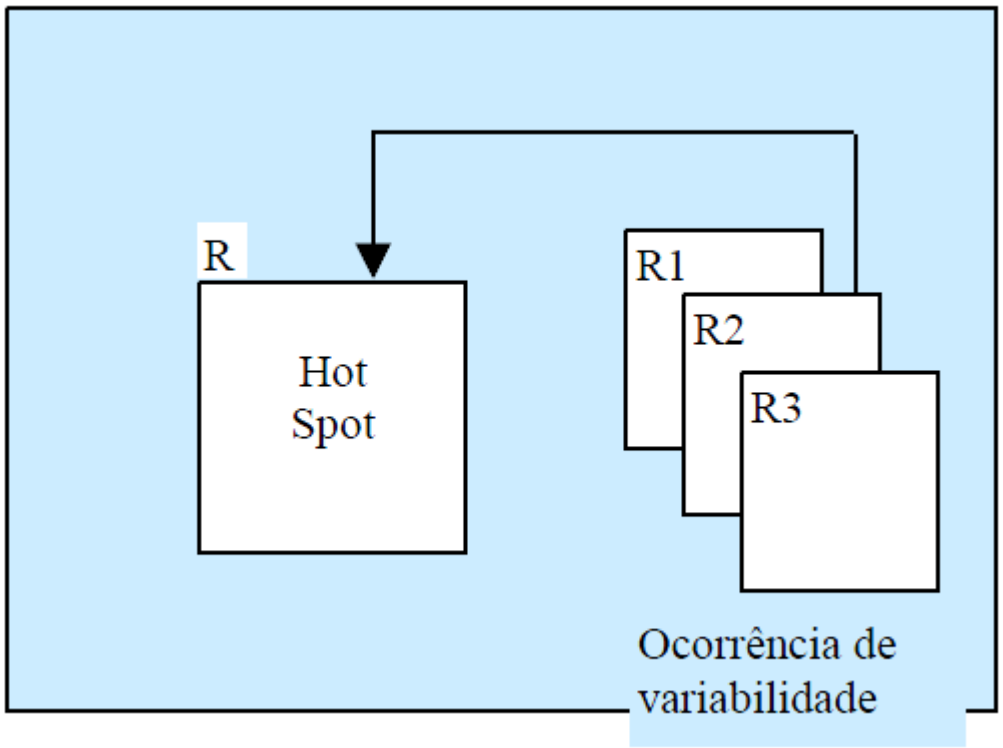
Hot-spots

- Caixa Branca



Hot-spots

- Caixa Preta



Classificação de Frameworks

- Divididos em 3 tipos
  - De infraestrutura de sistemas
  - De integração e comunicação
  - De aplicações gerais

Classificação de Frameworks

- Frameworks de Infraestrutura
  - Simplificam o desenvolvimento da infraestrutura de sistemas portáveis e eficientes.
  - Exemplos: sistemas operacionais, comunicação, interfaces com o usuário e ferramentas de processamento de linguagem
  - Em geral são usados internamente em uma organização de software e não são vendidos a clientes diretamente.

Classificação de Frameworks

- Frameworks de Integração (Middleware)
  - Usados em geral para integrar aplicações e componentes distribuídos.
  - Projetados para melhorar a habilidade de desenvolvedores em modularizar, reutilizar e estender sua infra-estrutura de software para funcionar “seamlessly” em um ambiente distribuído
  - Exemplos: Object Request Broker (ORB), middleware orientado a mensagens e bases de dados transacionais

Classificação de Frameworks

- Frameworks de aplicação organizacional
  - Voltados a domínios de aplicação mais amplos e são a pedra fundamental para atividades de negócios das empresas.
  - Exemplos: telecomunicações, aviação, manufatura e engenharia financeira.
  - São mais caros para desenvolver ou comprar, mas podem dar um retorno substancial do investimento, já que permitem o desenvolvimento de aplicações e produtos diretamente

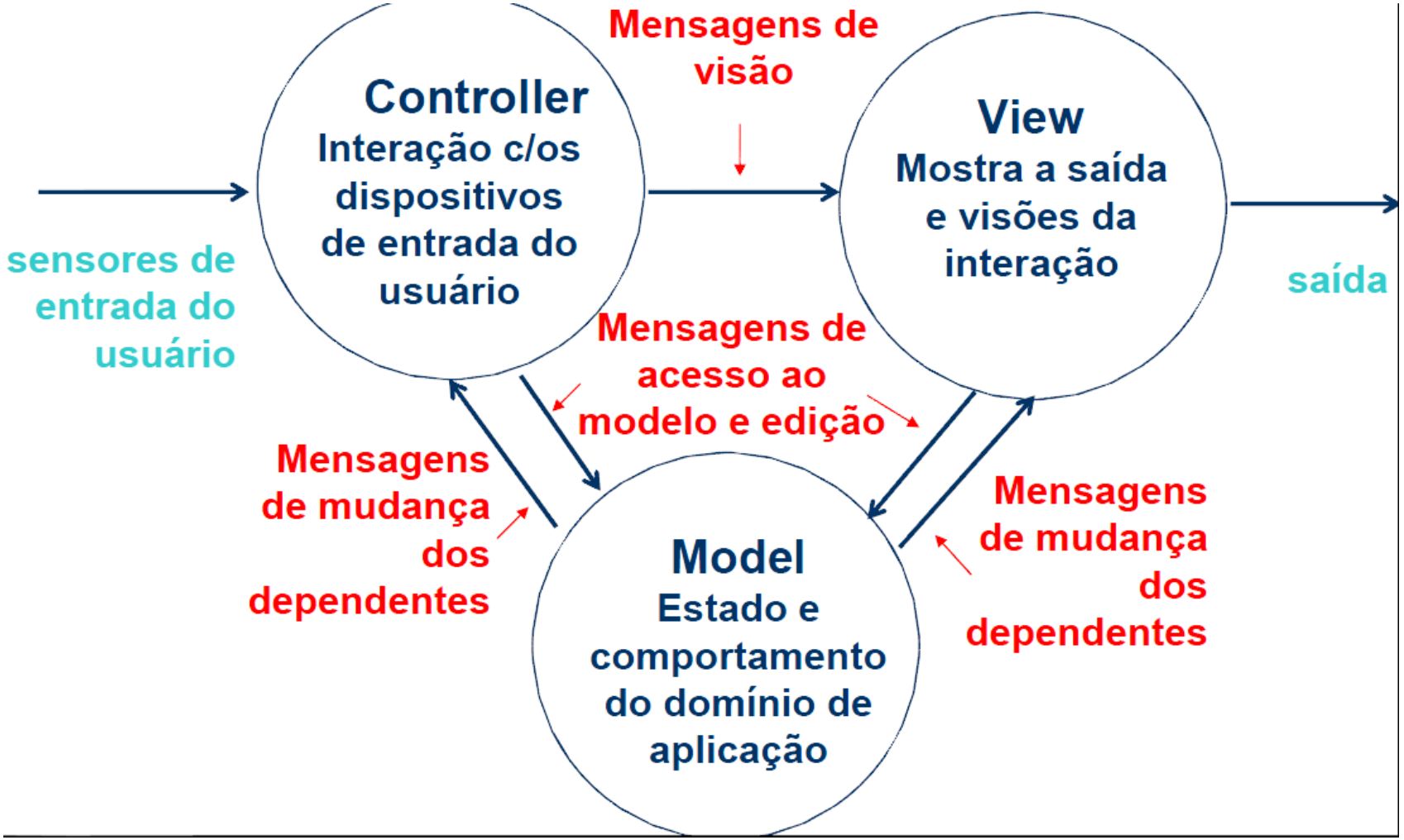
Exemplos de Frameworks

- A maioria dos frameworks existentes é para domínios técnicos tais como interfaces com o usuário ou distribuição
  - MacApp: framework para aplicações Macintosh
  - Lisa Toolkit
  - Smalltalk - Model View Controller (MVC)
- A maioria dos frameworks para aplicações específicas não é de domínio público
  - ET++, Interviews, ACE
  - Microsoft Foundation Classes (MFC) e DCOM
  - JavaSoft's RMI
  - Implementações do OMG's CORBA

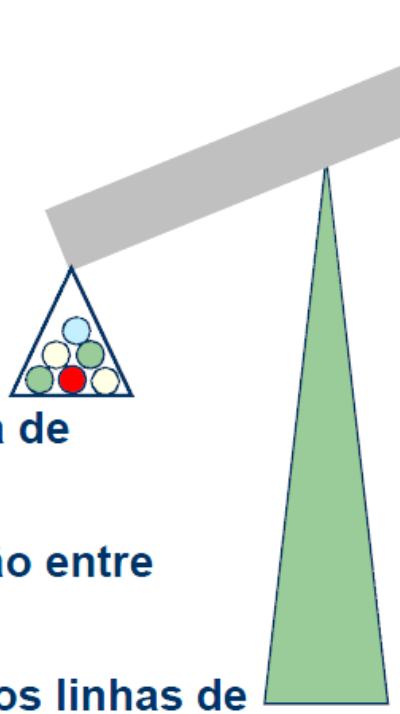
Model-View-Controller (MVC)

- Primeiro framework largamente utilizado
- Implementado em Smalltalk-80
- Usado pelo Smalltalk como interface com o usuário
- Mostrou a adequação da orientação a objetos para implementação de interfaces gráficas com o usuário

Model-View-Controller (MVC)



Benefícios x Custos



Benefícios

- Economia a longo prazo
- Aproveitamento da experiência de especialistas do domínio
- Maior consistência e integração entre aplicações
- Redução da manutenção: menos linhas de código nas aplicações, reparos no framework se propagam pelas aplicações
- Melhora da produtividade: os programadores podem se concentrar nas características específicas de suas aplicações

Custos

- Mais esforço para construção e aprendizado
- Programas mais difíceis de depurar
- Necessária documentação de manutenção e apoio

Atividades Extra-Classe

[Não vale nota ☺]

- Leia os artigos indicados abaixo e reflita:
  - **Quais os principais aspectos considerados no desenvolvimento de frameworks OO?**
    - "Building Application Frameworks: Object-Oriented Foundations of Framework Design" - http://dl.acm.org/citation.cfm?id=326112
  - **Como é possivel projetar frameworks Oo a partir de Linhas de Produto de Software?**
    - "Recovering Object-Oriented Framework for Software Product Line Reengineering" - http://link.springer.com/chapter/10.1007/978-3-642-21347-2_10

Atividades Extra-Classe

[Não vale nota ☺]

- Qual a principal diferença entre frameworks, componentes, bibliotecas e API?

Material Complementar

- Slides sobre "Reuso de Software" do livro "Engenharia de Software" de Ian Sommerville
  - https://www.slideshare.net/software-engineering-book/ch15-software-reuse

Referências

MALDONADO, J. C. et. Al. Padrões e Frameworks de Software (Notas didáticas), ICMC-USP, 2010.

GIMENES, I. M. S.; HUZITA, E. H. M. Desenvolvimento Baseado em Componentes: Conceitos e Técnicas. Editora Ciência Moderna, 2005.