Model View Controller

Prof. Rafael Guterres Jeffman

Fecomércio RS | Senac

O que é MVC?

- Model-View-Controller é um padrão de desenvolvimento de software.

- Uma aplicação MVC é dividida em três partes interconectadas, de forma que cada parte é responsável por apenas uma parte do sistema (dados, controle, visualização).

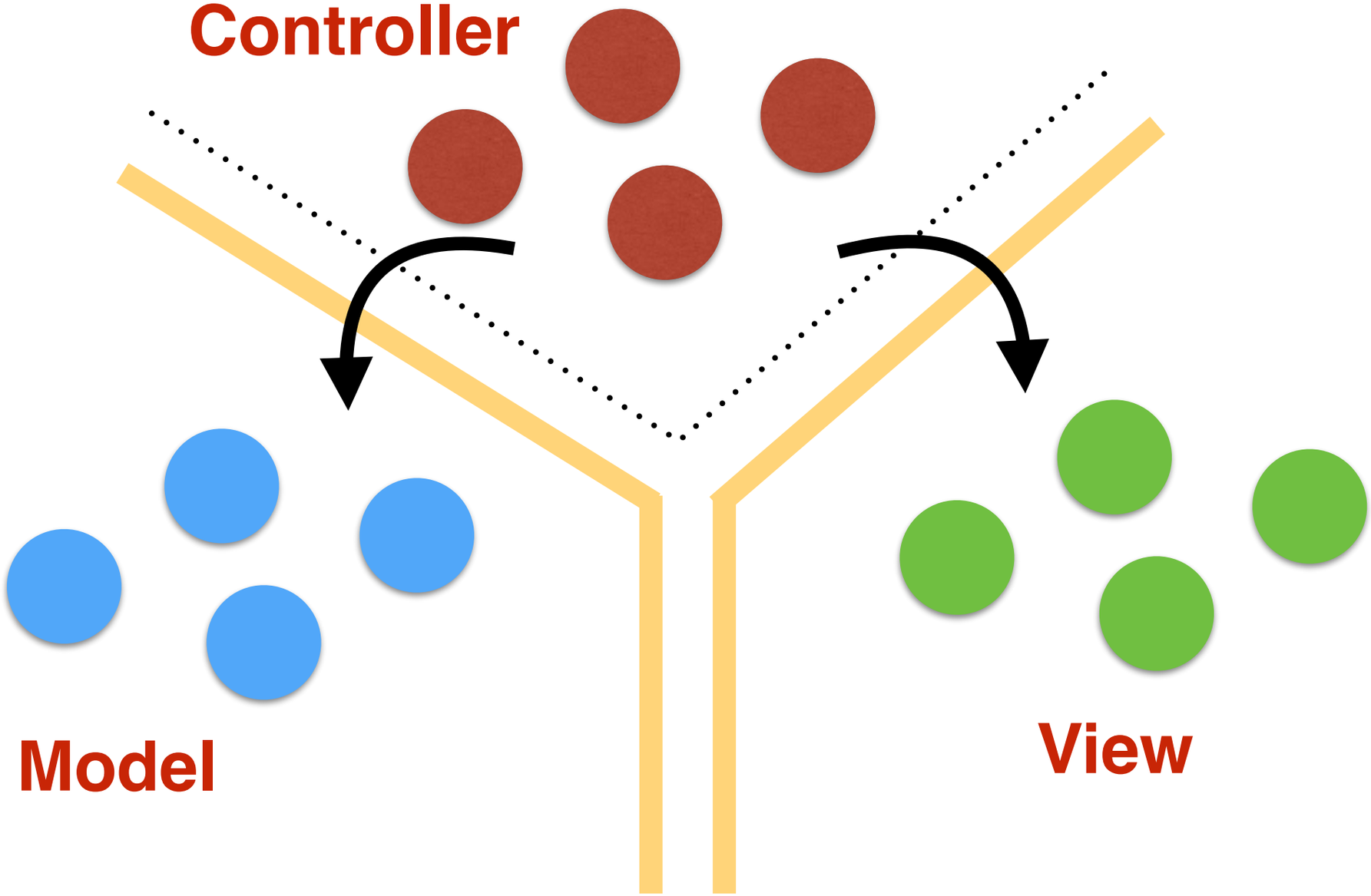
Vantagens

- Organização do código.
- Flexibilidade.

Desvantagens

- Complexidade
- Pode “aumentar” pequenas aplicações.

Comunicação no MVC



Model

- Modela e gerencia os dados do **domínio** da aplicação. Consiste em dados, regras de negócio, lógica e funções.

- Idealmente, é independente da aplicação, mas dependente do domínio.

- Não tem conhecimento da interface ou dos controles.

- Pode ser “ativo”, quando notifica os controles/interfaces de uma modificação; ou “passivo”, quando espera que os controles/interfaces requisitem as modificações.

- Modelos são independentes das interfaces.

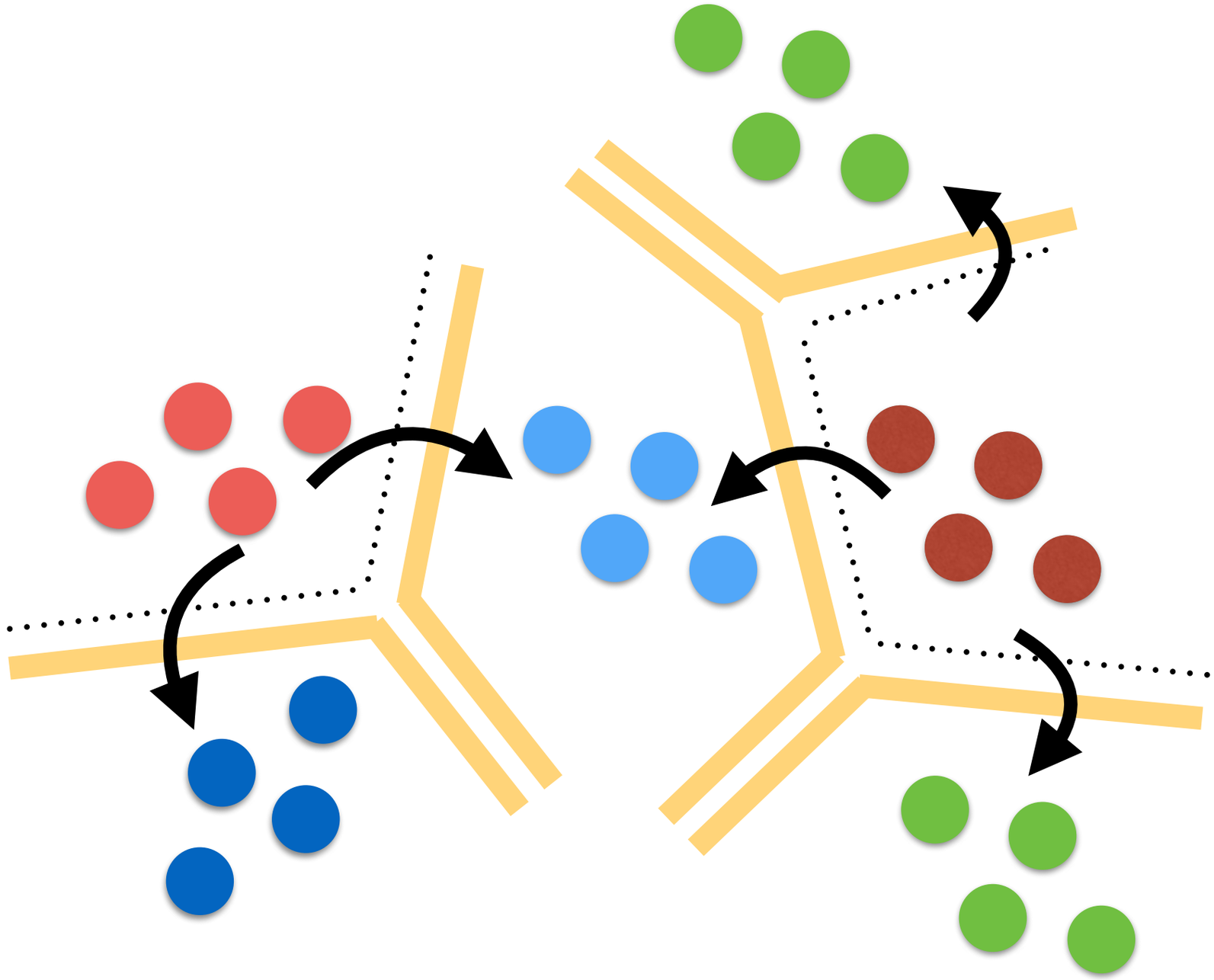
View

- Gerencia a visualização dos dados da aplicação.
- Uma mesma aplicação pode ter “views” diferentes.
- A entrada de dados também é parte da view.
- Os dados exibidos são passados pelos controles.
- As intervenções de usuários são passadas para os controles.
- Views são independentes dos modelos.

Controller

- Atua como mediador entre o modelo e a view.
- Conhece o modelo, e sabe como traduzir a informação de/para a View.
- Apesar da View e do Modelo serem independentes do Controle, o Controle depende tanto da View quanto do Modelo.
- Atua tanto na visualização da informação, quanto na alteração dos dados do modelo devido a iteração com o usuário.

Múltiplos MVC



Fecomércio RS | Senac

Exemplo em Java

- Model: classe Pessoa
- Controller: classe PessoaController
- View: classe ExtendedView

Pessoa

```
package mvc.model;

public class Pessoa {

    private String nome;
    private String telefone;

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
}
```

PessoaController

```
package mvc.controller;

import mvc.model.Pessoa;
import mvc.view.ExtendedView;

public class PessoaController {

    private ExtendedView view = new ExtendedView();

    private Pessoa pessoa;

    public void criaPessoa() {
        pessoa = new Pessoa();
        pessoa.setNome("Rafael");
        pessoa.setTelefone("555-1234");
    }

    public void mostraPessoa() {
        view.imprimePessoa(pessoa.getNome(), pessoa.getTelefone());
    }

}
```

ExtendedView

```
package mvc.view;

import static java.lang.System.out;

public class ExtendedView {
   public void imprimePessoa(String nome, String telefone)
   {
      out.println("Nome: " + nome);
      out.println("Telefone: " + telefone);
   }
}
```

Aplicação

```
package mvc.app;

import mvc.controller.PessoaController;

public class PessoaApp {
   public static void main(String[] args)
   {
      PessoaController controller = new PessoaController();
      controller.criaPessoa();
      controller.mostraPessoa();
   }
}
```

Ganhando com Herança

- “View” é na verdade uma “interface” para exibir os dados.

- A interface possui apenas um método para imprimir os dados de um contato.

- Podemos dessa forma ter diferentes “Views” para o mesmo modelo e controle.

interface PessoaView

```
package mvc.view;

public interface PessoaView {
   public void imprimePessoa(String nome, String telefone);
}
```

Modificação de ExtendedView

```
package mvc.view;

import static java.lang.System.out;

public class ExtendedView implements PessoaView {
   public void imprimePessoa(String nome, String telefone)
   {
      out.println("Nome: " + nome);
      out.println("Telefone: " + telefone);
   }
}
```

CompactView

```
package mvc.view;

import static java.lang.System.out;

public class CompactView implements PessoaView {

   public void imprimePessoa(String nome, String telefone) {
      out.println(String.format("Contato: %s - %s", nome, telefone));
   }

}
```

Modificação do Controller

```
package mvc.controller;

import mvc.model.Pessoa;
import mvc.view.*;

public class PessoaController {

     private PessoaView view;

     private Pessoa pessoa;

     public PessoaController() {
          this.view = new ExtendedView();
     }

     public void setView(PessoaView view) {
          this.view = view;
     }

     public void criaPessoa() {
          pessoa = new Pessoa();
          pessoa.setNome("Rafael");
          pessoa.setTelefone("555-1234");
     }

     public void mostraPessoa() {
          view.imprimePessoa(pessoa.getNome(), pessoa.getTelefone());
     }

}
```

Nova Aplicação

```
package mvc.app;

import mvc.controller.PessoaController;
import mvc.view.CompactView;
import mvc.view.ExtendedView;

public class PessoaApp {
    public static void main(String[] args)
    {
        PessoaController controller = new PessoaController();
        controller.criaPessoa();

        controller.setView(new ExtendedView());
        controller.mostraPessoa();

        controller.setView(new CompactView());
        controller.mostraPessoa();
    }
}
```