



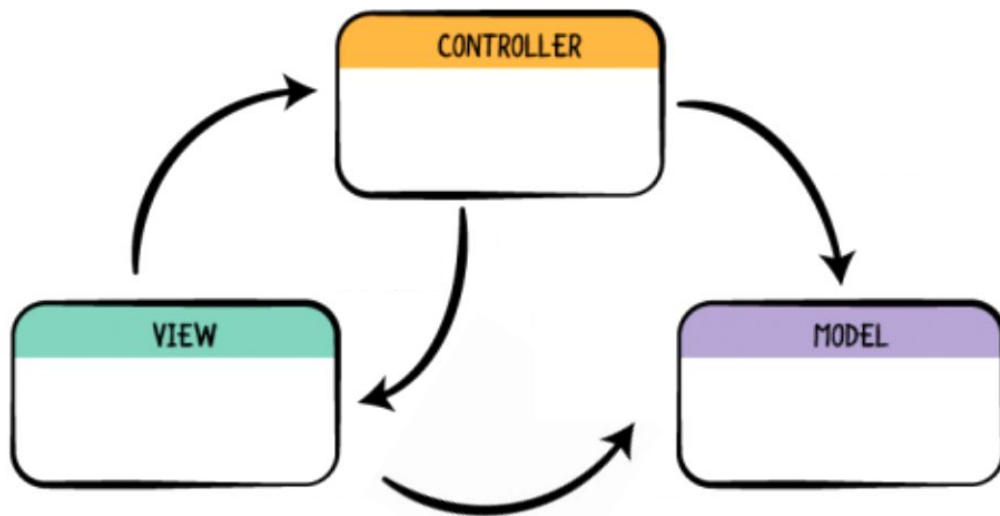
Conformidade Arquitetural

Trabalho Prático 10

Christian Marlon Souza Couto
Luana Almeida Martins

Objetivo

- Identificar desvios em relação à arquitetura planejada.



Solução

- Fornecer regras de conformidade;
- Analisar as dependências entre pacotes;
- Relatar tipos de dependências não permitidos conforme as regras estabelecidas.

Terminologia

- CAN - indica uma dependência permitida;
- CANNOT - indica uma dependência não permitida;
- MUST - indica uma dependência obrigatória.

Tipos de dependências

- Declaração de atributos;
- Criação de objetos;
- Extensão de classes;
- Implementação de interfaces.

Terminologia para desvios

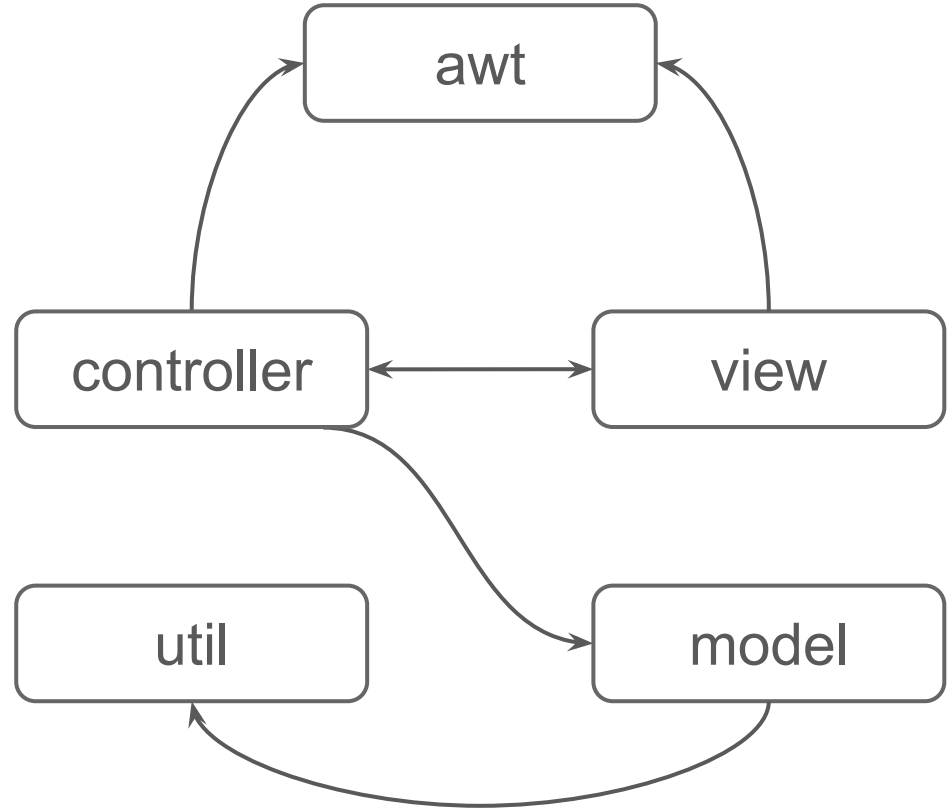
- Cinza - dependência permitida, porém não existente (CAN);
- Amarelo - indica uma dependência não permitida, porém existente (CANNOT);
- Vermelho - indica uma dependência obrigatória, porém não cumprida (MUST).

Exemplo

view MUST model

controller CANNOT awt

controller CAN util

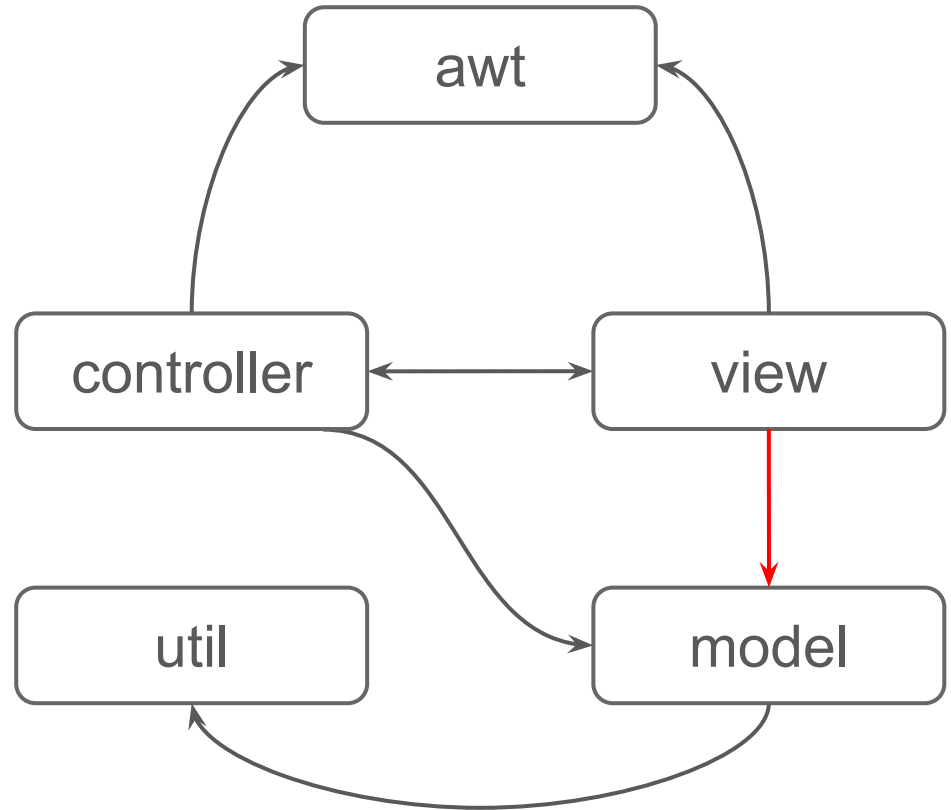


Exemplo

view **MUST** model

controller CANNOT awt

controller CAN util

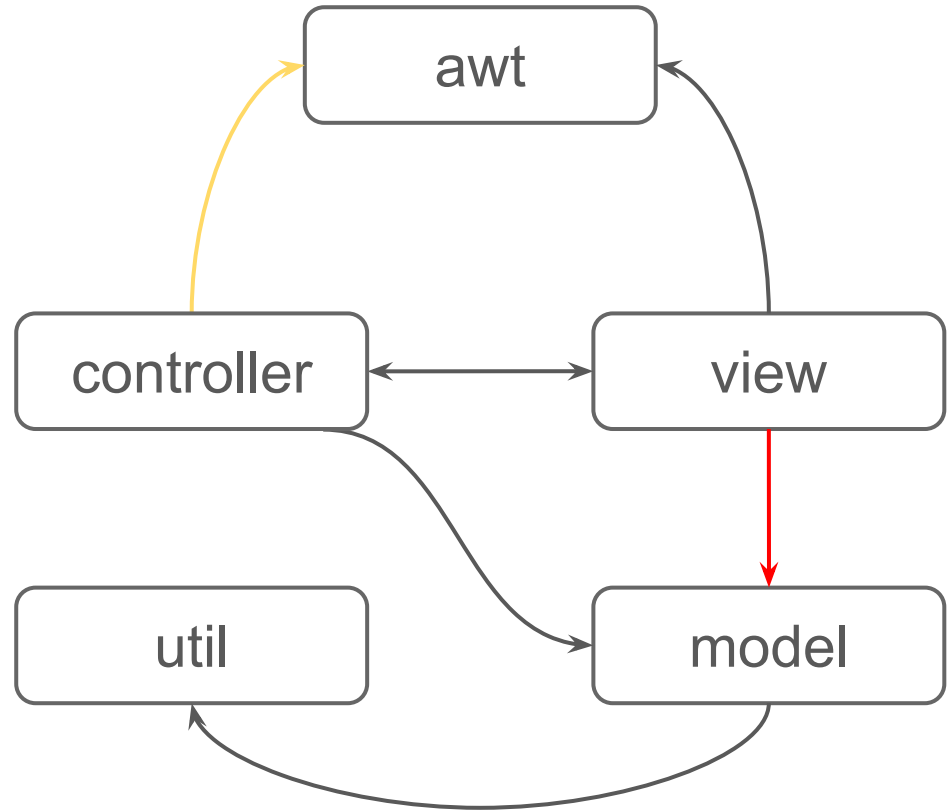


Exemplo

view **MUST** model

controller **CANNOT** awt

controller CAN util

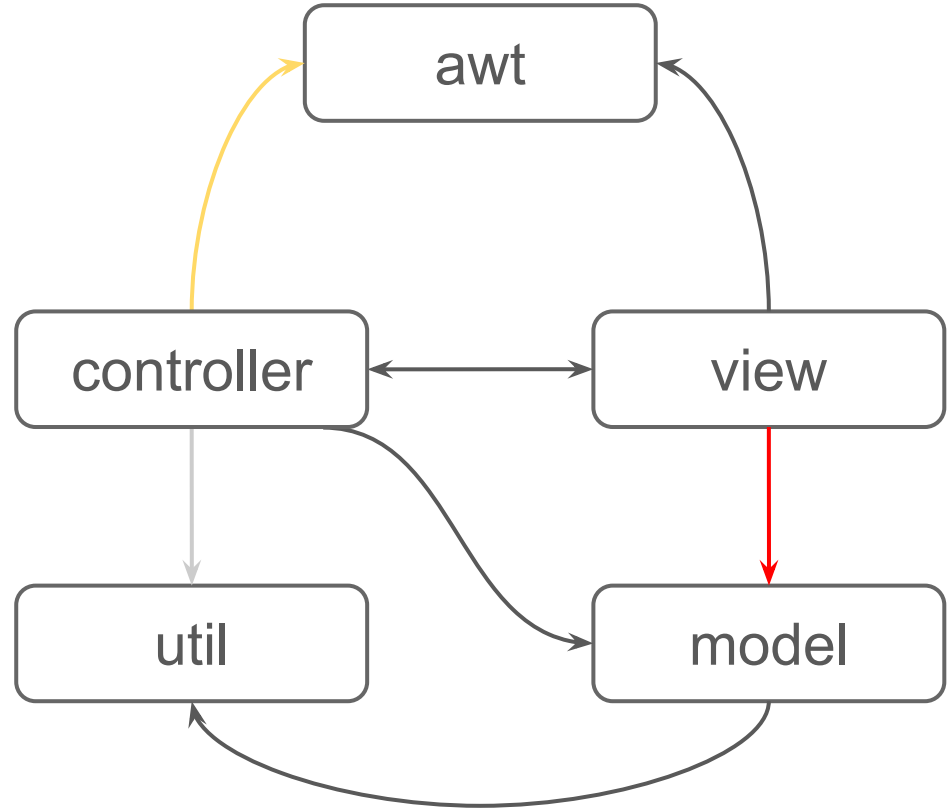


Exemplo

view **MUST** model

controller **CANNOT** awt

controller CAN util



Exemplo

The screenshot shows an IDE window titled "Métodos" with a table of dependencies. The table has two columns: "Pacote" and "depende de". The rows are:

Pacote	depende de
view	model
controller	awt
controller	util

Below the table, a dialog box titled "Informação de conformidade" is displayed. It contains an information icon and the text: "A classe ControllerPrincipal.controller PODE MAS NAO depende de util". An "OK" button is at the bottom right of the dialog.

Implementação

