

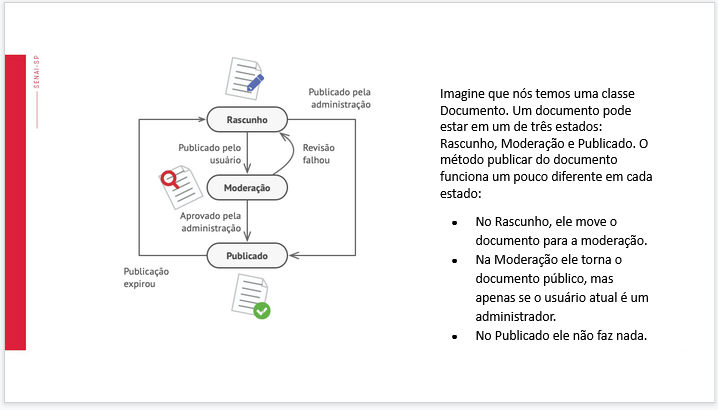
ideia principal é que, em qualquer dado momento, há um número *finito* de *estados* que um programa possa estar. Dentro de qualquer estado único, o programa se comporta de forma diferente, e o programa pode ser trocado de um estado para outro instantaneamente. Contudo, dependendo do estado atual, o programa pode ou não trocar para outros estados. Essas regras de troca, chamadas *transições*, também são finitas e pré determinadas.

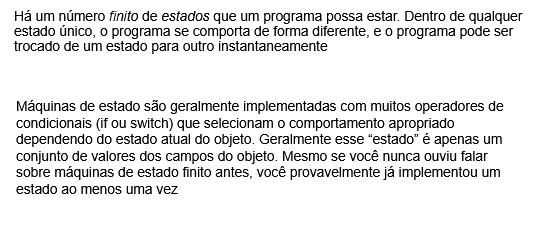
Você também pode aplicar essa abordagem para objetos. Imagine que nós temos uma classe Documento. Um documento pode estar em um de três estados: Rascunho, Moderação e Publicado. O método publicar do documento funciona um pouco diferente em cada estado:

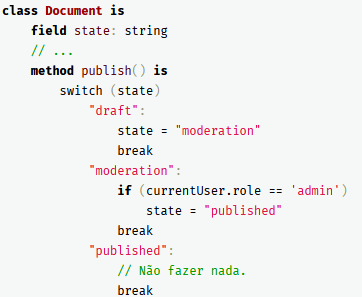
* No Rascunho, ele move o documento para a moderação.
* Na Moderação ele torna o documento público, mas apenas se o usuário atual é um administrador.
* No Publicado ele não faz nada.

Máquinas de estado são geralmente implementadas com muitos operadores de condicionais (if ou switch) que selecionam o comportamento apropriado dependendo do estado atual do objeto. Geralmente esse “estado” é apenas um conjunto de valores dos campos do objeto. Mesmo se você nunca ouviu falar sobre máquinas de estado finito antes, você provavelmente já implementou um estado ao menos uma vez. A seguinte estrutura de código lembra alguma coisa para você?

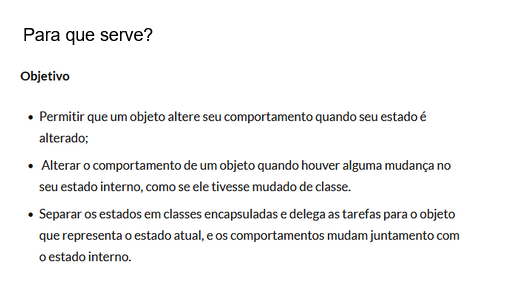
A maior fraqueza de uma máquina de estados baseada em condicionais se revela quando começamos a adicionar mais e mais estados e comportamentos baseados em estados para a classe Documento. A maioria dos métodos irá conter condicionais monstruosas que selecionam o comportamento apropriado de um método de acordo com o estado atual. Um código como esse é muito difícil de se fazer manutenção porque qualquer mudança na lógica de transição pode necessitar de mudanças de condicionais de estado em todos os métodos.

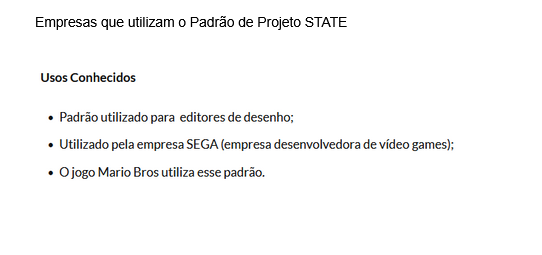












SITES DE REFERÊNCIA

<https://www.thiengo.com.br/padrao-de-projeto-state-estado>

<https://padroesdeprojetoifc.wordpress.com/2016/11/23/state/>

<https://refactoring.guru/pt-br/design-patterns/state>