

# Framework

## O que é um Framework?

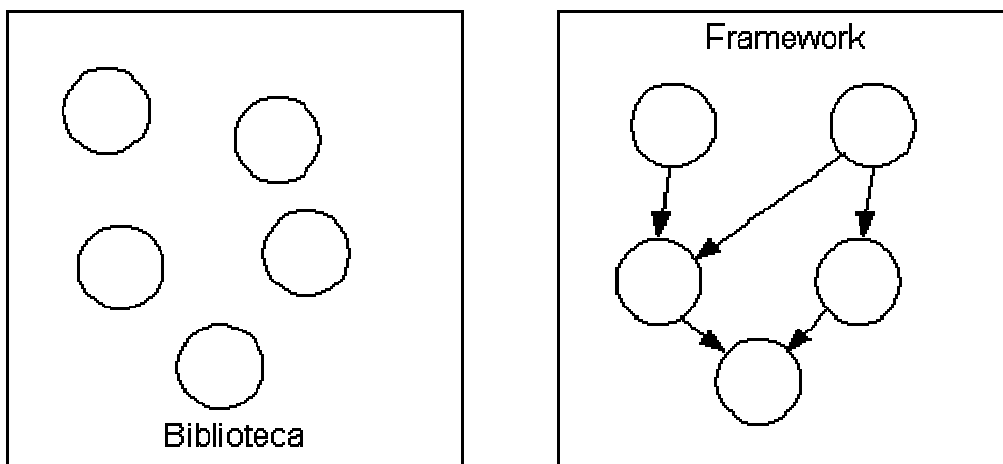
Um framework captura a funcionalidade comum a várias aplicações.

As aplicações devem ter algo razoavelmente grande em comum: pertencem a um mesmo domínio de problema, há várias definições de frameworks, a definição que usamos foca quatro características principais de um framework Orientado a Objeto:

"Um framework provê uma solução para uma família de problemas semelhantes, usando um conjunto de classes e interfaces que mostra como decompor a família de problemas, e como 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". Observe que um framework é uma aplicação quase completa.

Diferenças entre um Framework e uma Biblioteca de Classes OO, numa biblioteca de classes, cada classe é única e independente das outras, num framework, as dependências/colaborações estão embutidas (wired-in interconnections).

Com Frameworks, as aplicações criam as colaborações:



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 a Hollywood Principle, é o framework que chama o código da aplicação que trata das particularidades dessa aplicação.

Framework = Upside-down library

## Para que serve um framework?

A utilização de um framework por um desenvolvedor torna-se útil no momento em que você constrói ou utiliza certo componente em mais de uma vez. A reutilização de códigos que o framework proporciona é fantástica. Um ótimo conselho é sempre construir os códigos os mais genéricos possíveis, mesmo que isto custe algum tempo a mais, mas quando você for precisar novamente desta ferramenta criada, terá em suas mãos e poderá reutilizar o código já anteriormente desenvolvido. Quanto mais reutilização de códigos utilizarem, mais produtividade você ganha.

Um ótimo exemplo é um formulário de login, muitos sites utilizam, por que você não cria um formulário genérico e reutiliza da mesma forma para os sites que for desenvolver. Afinal login aqui e em qualquer lugar do mundo será o mesmo. Como acima citei, faça um formulário genérico, use todas as possibilidades que um formulário de login possui, como por exemplo:

- O próprio login.
- O lembrete de senha.
- E o cadastro da pessoa.

Com isto será possível você abranger cerca de 98% dos forms que for criar. O que você não precisa em um projeto, descarte, mas tente sempre ter o máximo de funcionalidade que o sistema for prover. Assim você consegue manter um repositório de funções rico.

## Construir um framework?

Não há necessidade, pois os frameworks da internet são ricos, e possuem muitas funcionalidades que podemos utilizar em nossos projetos. Existem frameworks para todas as linguagens, por exemplo, para Ruby existe o Rails, para PHP existe o CakePHP, Zend Framework, para Java existem frameworks específicos para cada tipo de aplicação, em Java script existe um muito bom chamado Prototype.

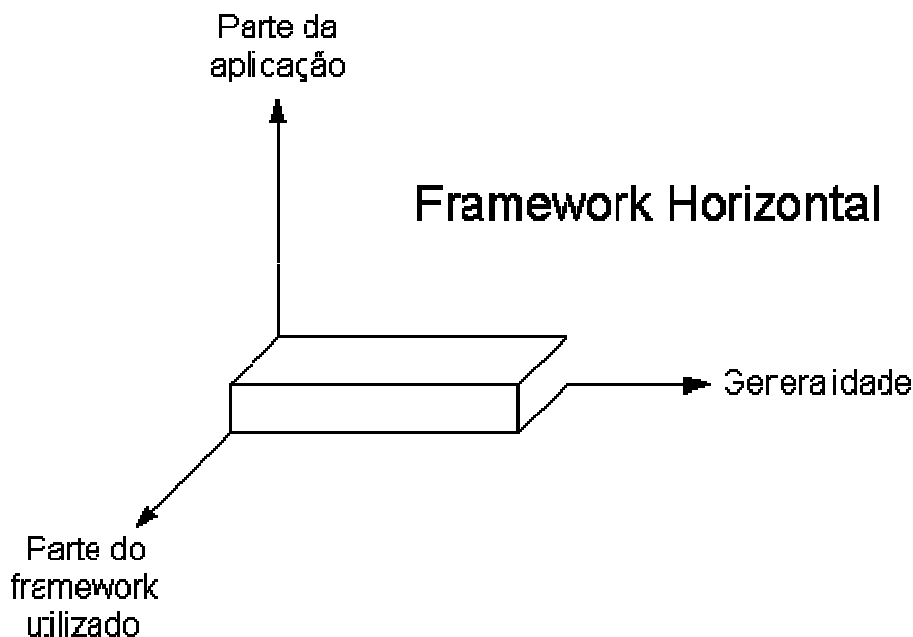
---

### Como o framework é usado:

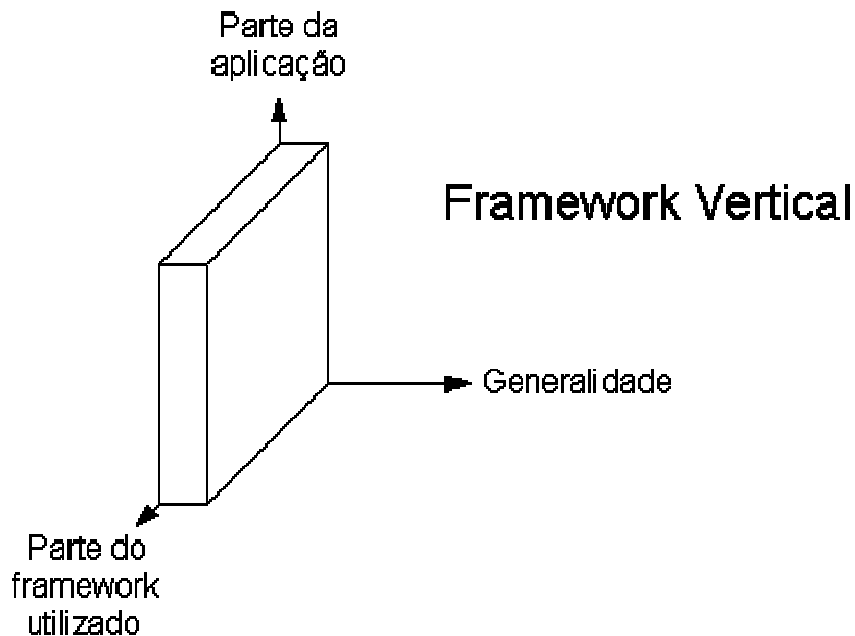
Como introduzir as particularidades da aplicação? Inheritance-focused, também chamado de White-box ou architecture-driven estende ou modifica funcionalidade pela definição de subclasses com override de métodos Composition-focused também chamado de Black-box ou data-driven, usa a funcionalidade já presente no framework. As coisas internas do framework não podem ser vistas ou alteradas deve-se usar as interfaces fornecidas, as instâncias e composições feitas, determinam as particularidades da aplicação, no limite, o framework se torna Component-Oriented (COFW).

### Onde o framework é usado:

- Framework de suporte provê serviços de nível de sistema operacional (e não de aplicação).
- Acesso a arquivos.
- Computação distribuída.
- Device drives.
- Resolve apenas uma fatia do problema da aplicação.
- Ex: framework para construção de interface GUI.

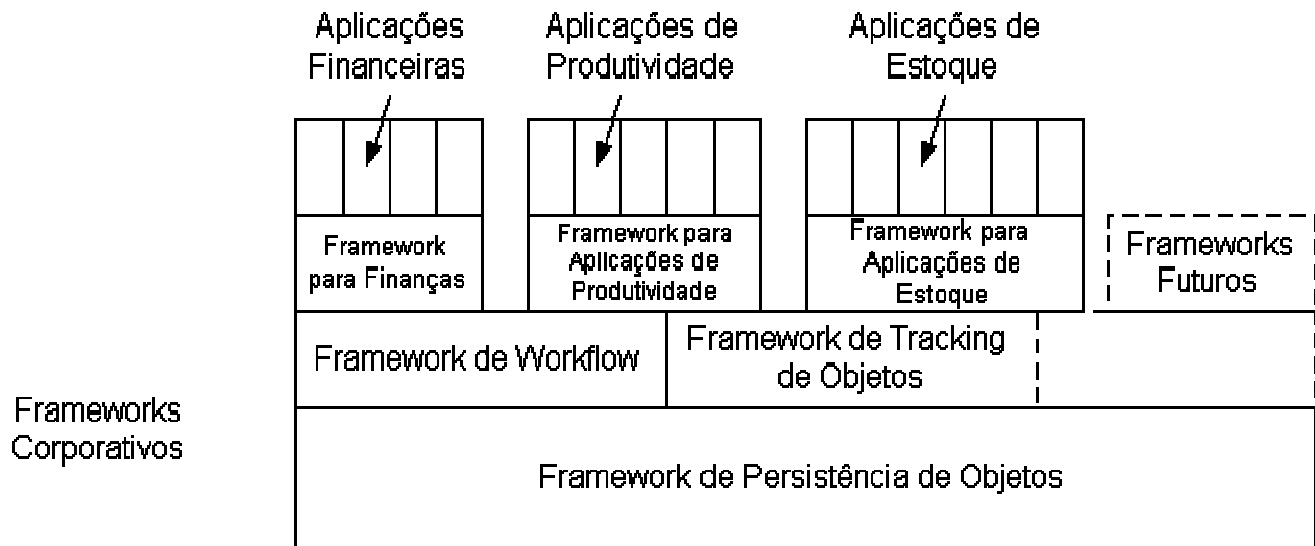


- Framework de domínio, também chamado de framework vertical encapsula conhecimento ("expertise") aplicável a aplicações pertencendo a um domínio particular de problema, resolve boa parte da aplicação.
- Exemplo: Framework para construir aplicações de controle de manufatura.



Estruturas de Frameworks:

Uma empresa pode usar vários frameworks para construir suas aplicações alguns podem ser horizontais (abaixo, na figura) e outros verticais (acima, na figura).



**Vantagens no uso de frameworks:**

Se o framework estiver pronto, os benefícios são claros em termos de redução de custos, redução de time-to-market.

Motivos: Maximização de re-uso (análise, design, código, testes) desenvolvedores se concentram em adicionar valor em vez de reinventar a roda, menos manutenção, fatoração de aspectos comuns a várias aplicações, uso de herança permite corrigir todas as aplicações com a troca de uma classe-mãe, mas cuidado com o "Fragile Base Class Problem" onde a troca da classe-mãe quebra as filhas, estabilização, melhor do código (menos defeitos) devido ao uso em várias aplicações, melhor consistência e compatibilidade entre aplicações alavancagem do conhecimento de especialistas. Framework oferece uma forma de empacotar o conhecimento de especialistas sobre domínios de problemas assim, não se perde o conhecimento com a saída de especialistas e o conhecimento pode ser usado/estudado sem a presença do especialista.

**Desvantagens no uso de frameworks:**

Construir um framework é complexo, re-uso não vem sozinho: deve ser planejado é mais complexo e demora mais fazer uma aplicação tendo que construir um framework em vez de fazer a aplicação do zero, benefícios é realizado em longo prazo.