Framework Aplicacional

Enquadramento

Um dos objectivos no desenvolvimento de software é implementar uma solução que resolva não só o problema em questão, mas que forneça também uma base flexível e extensível para o desenvolvimento de outras aplicações com características semelhantes.

Nesta perspectiva qualquer aplicação deve ser encarada como um conjunto de peças (**componentes**) que se integram dando forma ao produto final (**aplicação**).

Na identificação dos vários componentes há que distinguir entre aqueles que têm um carácter **genérico** e os que implementam **especificidades únicas da aplicação**.

Divididas as responsabilidades há que encontrar ***frameworks*** que forneçam uma base para a implementação da funcionalidade genérica pretendida.

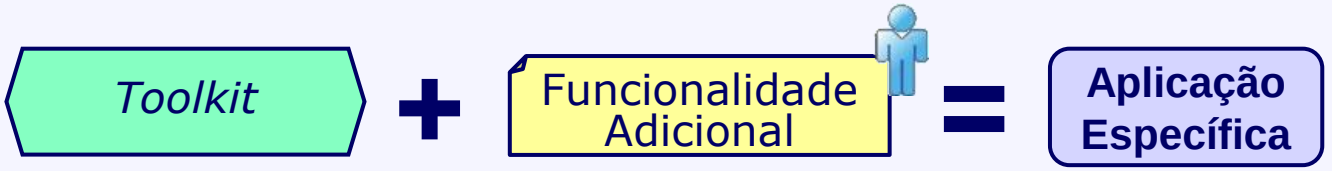
➔ Se não existir, terá que ser implementada.

Framework Aplicacional

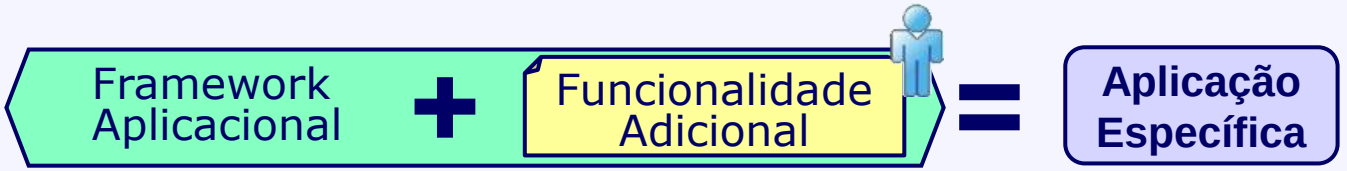
Framework $\neq$ **Biblioteca de classes**

(no livro *Design Patterns* é usada a designação ***Toolkit*** para biblioteca de classes)

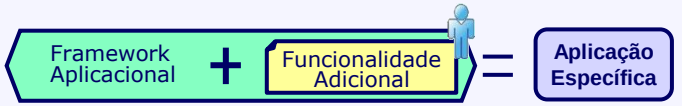
- Uma biblioteca implementa e oferece uma **funcionalidade completa**.



- Uma *framework* **estrutura os mecanismos essenciais** num determinado domínio.



Framework Aplicacional...



- A maioria da actividade tem origem na *framework*. **A *framework* chama os métodos na ordem apropriada à execução dos mecanismos essenciais**.

- O programador tem pouca influência na ordem pela qual são chamados os métodos implementados com a funcionalidade adicional.

Característica:

➔ **A *framework* controla o fluxo de execução da aplicação**.

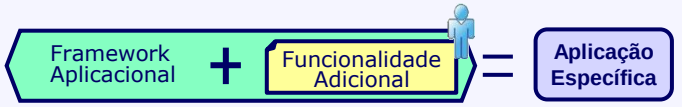
➔ Este fenómeno designa-se de **inversão de controlo**.

Papéis:

- ***Framework***: determina os métodos que devem ser chamados na ordem e no momento correcto.

- **Programador**: **redefine métodos** de modo a preencher os mecanismos da *framework* com a funcionalidade da aplicação.

Framework Aplicacional…



- Para construir uma aplicação o programador **estende** alguns dos tipos da *framework* e **implementa a funcionalidade adicional específica** da sua aplicação.

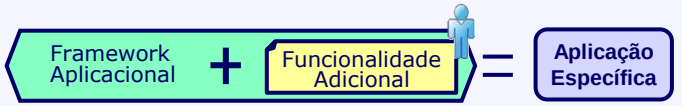
  Exemplo:
  - O *AWT* (*abstract window toolkit*) é uma *framework* de desenvolvimento no domínio das **interfaces gráficas do utilizador** (**GUI**);

  ➔ Um programador pode implementar novos componentes GUI estendo classes da *framework*, tais como, `Component` ou `Canvas`, adaptando o seu comportamento para obedecer à funcionalidade pretendida.

Característica:

➔ Fornece um conjunto de tipos que são reutilizados pelo programador para construir uma determinada aplicação, através de técnicas de **herança** (extensão e implementação) das classes e interfaces da *framework*.

Framework Aplicacional...



Definição:
Uma ***framework* aplicacional** é um conjunto de tipos cooperantes que definem um **desenho reutilizável para um sistema de software**, num determinado domínio de aplicações.

➔ Ao contrário de um padrão de desenho uma *framework* **não é o desenho genérico de uma solução para um problema recorrente**.

Um padrão de desenho:

- não é materializado nem depende de uma determinada tecnologia de desenvolvimento de software;
- não se aplica exclusivamente a um domínio de aplicações.

Uma *framework*:

- **estabelece** a arquitectura das aplicações construídas;
- **pode implementar parte** dos mecanismos essenciais às aplicações desse domínio;
- **pode aplicar os padrões de desenho** como peças de construção.

Conclusões

- O conceito de *framework* tornou-se vulgar e **essencial** em programação OO.
- As *frameworks* dão a **capacidade máxima reutilização** aos sistemas OO.
- A maioria do desenho e código de uma aplicação é influenciado pelas *frameworks* que usa.

> "***If applications are hard to design, and toolkits are harder, then frameworks are hardest of all***.",
> *Design Patterns (Gamma et al. 1995)*