Patrones de Diseño

Builder
(Constructor)

Luis Osvaldo Llanes Nava

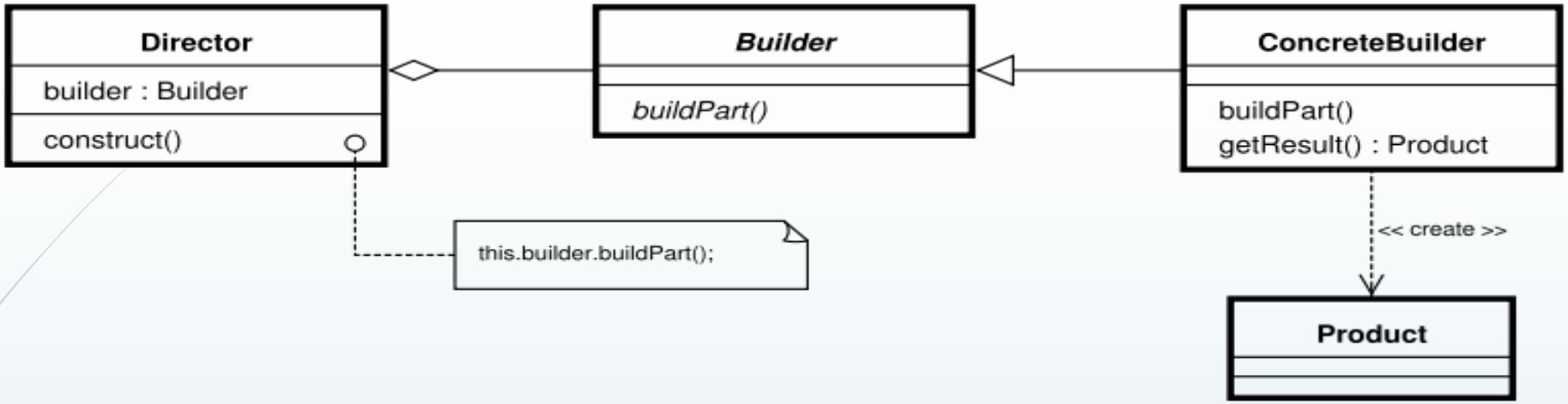
Problema y aplicación

Un único proceso de construcción debe ser capaz de construir distintos objetos complejos.

- Nuestro sistema trata con objetos complejos pero el número de configuraciones es limitada.
- El algoritmo de creación del objeto complejo puede independizarse de las partes que lo componen y del ensamblado de las mismas

Relaciones con otros Diseños

- Abstract Factory es similar en que también se puede construir objetos complejos
- Un Builder suele construir un Composite
- Se relaciona con el Factory Method



Director: Se encarga de construir un objeto utilizando el Constructor (Builder).

Builder: Interfaz abstracta que permite la creación de objetos.

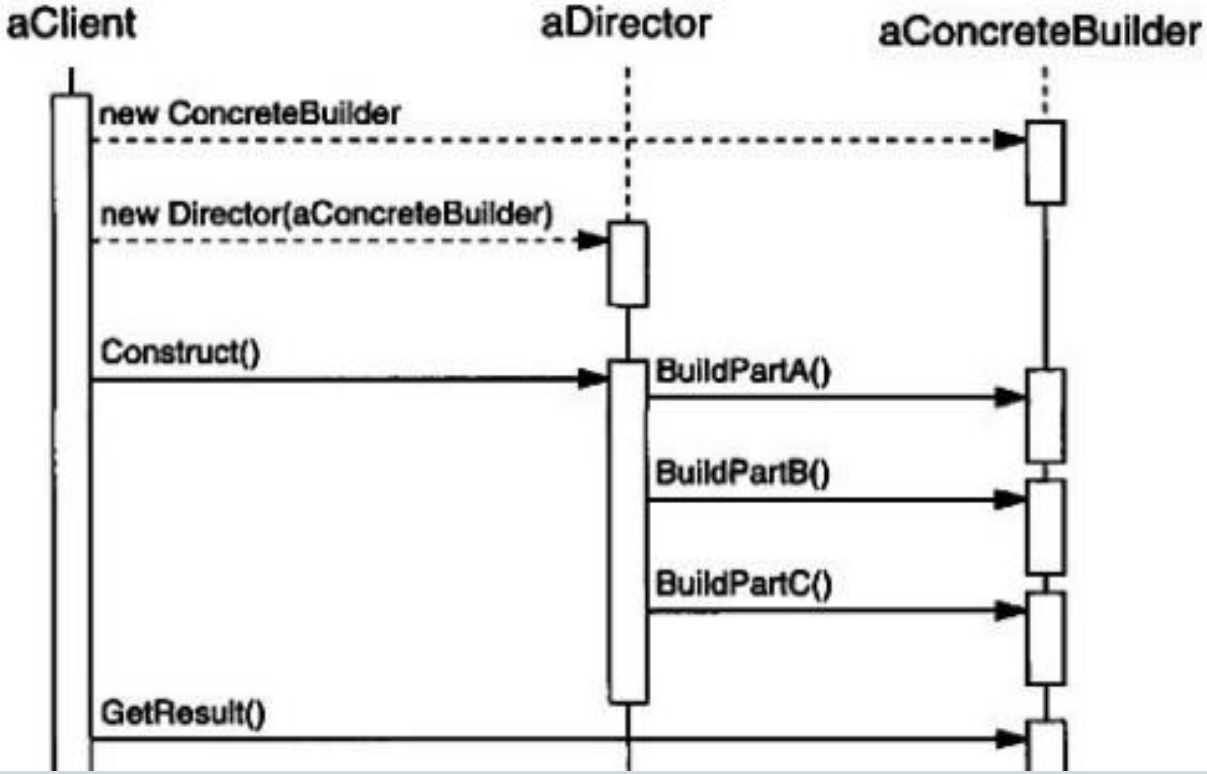
Concrete Builder: Implementación concreta del Builder definida para cada uno de los tipos.

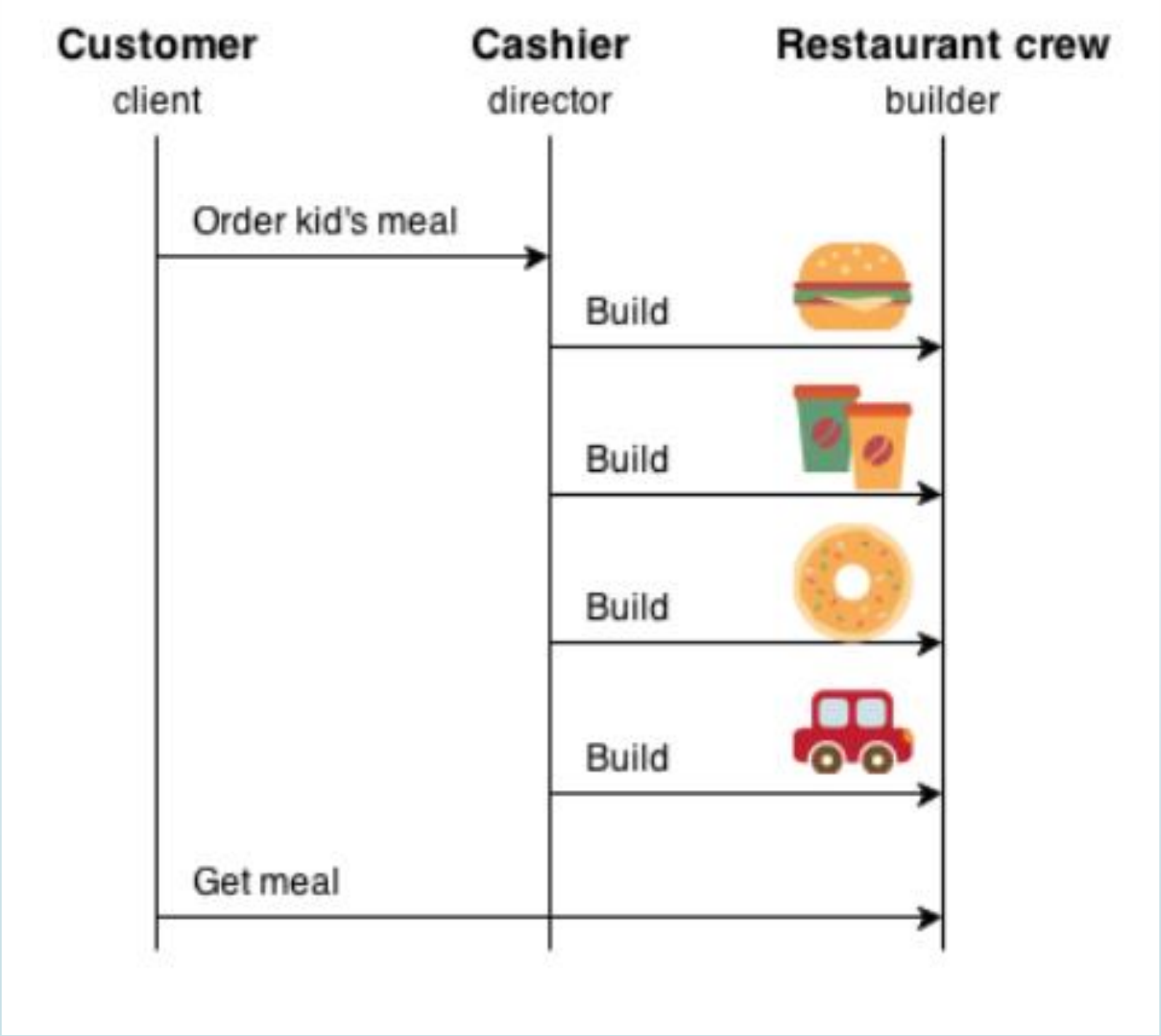
Product: Objeto que se ha construido tras el proceso definido por el patrón.

Collaborations

- The client creates the Director object and configures it with the desired Builder object.
- Director notifies the builder whenever a part of the product should be built.
- Builder handles requests from the director and adds parts to the product.
- The client retrieves the product from the builder.

The following interaction diagram illustrates how Builder and Director cooperate with a client.




Customer
client
Cashier
director
Restaurant crew
builder
Order kid's meal
Build
Build
Build
Build
Get meal

Consecuencias

- POSITIVAS:
  - Reduce el acoplamiento.
  - Permite variar la representación interna del objeto, respetando la clase builder. Es decir, conseguimos independizar la construcción de la representación.
- NEGATIVAS:
  - Introduce complejidad en los programas.