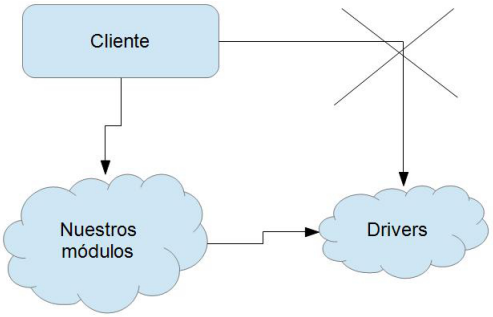
# Interacción con drivers. Patrón de diseño **Bridge** o **Handle/Body**.

Emilio Martín López.
Universidad Nacional de Rosario.

28 de Septiembre, 2012.

- Una aplicación debe interactuar con distintas implementaciones de un driver.
- Todos los drivers proveen escencialemnte la misma funcionalidad.
- Sin embargo algunos proveen más funcionalidad que otros.
- Además cada driver tiene su propia interfaz.
  Por ejemplo: un driver puede tener una subrutina que retorna la velocidad de un móvil en ese momento. Otro puede tener dos subrutinas: una que devuelve la posición y otra el tiempo.
- Se espera que el sistema pueda utilizar cualquiera de estos drivers.
- Incluso debe ser posible cambiar un driver por otro en tiempo de ejecución (uno por otro).
- En la aplicación existen diferentes módulos que usarán los drivers.
- Sin embargo estos módulos proveen más o menos la misma funcionalidad.
- Todos estos módulos deben usar la instancia actual del driver que haya sido elegido en ese momento.
- No queremos programar ningún driver; queremos usarlos al máximo posible.

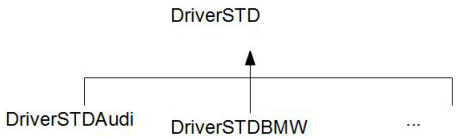
Ver GRAFICO1 el cliente no sabe de los driver simplemente usa los módulos que debemos programar.



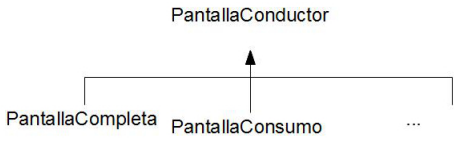

Es decir la idea est tener algo así:

| Cliente |
|---|
| Módulos para presentación |
| Driver |
| Placa |

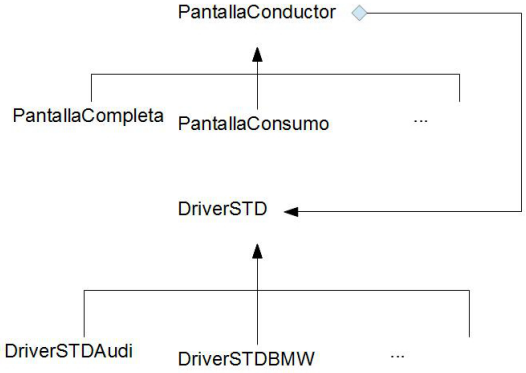
Ejemplo:
Auto → placa de colección de datos del auto → driver que provee esos datos → diferentes pantallas que muestran esos datos al conductor.
Item de cambio: drivers con distintas interfaces.
Volviendo al ejemplo del auto, las diferentes pantallas muestran más o menos datos al conductor. Supongamos que tenemos:
PantConsumo
PantCompleta
PantKm
⋮
Y por otro lado tenemos:
DriverAudi
DriverBMW
DriverFord
⋮
cada uno **con su propia interfaz**. Debemos tratar de definir una interfaz que sea una especie de **driver estándar DriverSTD**. Luego sus herederos tratarían con los diferentes drivers propietarios. Ver GRAFICO2. Los herederos tienen la misma interfaz. La interfaz de DriverSTD puede ser completamente distinta a la de todos los drivers.



Entonces la implementación de cada método de cada heredero se realiza en términos de la interfaz del driver correspondiente (por ejemplo, la implementación de DriverSTDAudi se hace haciendo llamadas la interfaz de DriverAudi).
Por otro lado definimos una jerarquía de pantallas. Ver GRAFICO3.




Esto lo compongo con lo visto anteriormente como muestra GRAFICO4.




La secuencia de ejecución es más o menos la siguiente:

1. El cliente crea una instnacia de heredero de PantConductor;
2. El cliente crea una instancia de un heredero de DriverSTD;
3. El cliente configura 1 con 2;
4. El ciente utiliza la interfaz de PantConductor.

Entonces mi jerarquía queda algo así:

| Cliente |
|---|
| Pantalla |
| DriverSTD |
| Driver |
| Placa |

La implementación de todos los herederos de PantConductor se realiza mediante llamadas a la interfaz de DriverSTD.
Este patrón de diseño se conoce como **Bridge** o **Handle/Body**.