Lo mismo pero con orientación a objetos…

- La base de datos ya NO es la reina de la fiesta

```
Client c = new Client("1234567", "Manuel", ...);
Vehicle v = new Vehicle("1241-CKJ", "seat", "almera");
c.addVehicle( v );
WorkOrder workOrder = new WorkOrder(v, "Pre-revisión ITV");


Mechanic m = findMechanic("Luis", "Pérez");
workOrder.assignMechanic( m );

Intervention i = new Intervention(workOrder, m);
SparePart r = findSparePart("DKJ.1244");
new Sustitution( i, r, 2 );

Long number = getNextInvoiceNumer();
Invoice f = new Invoice( number, workOrder );

CreditCard visa = new CreditCard("visa", "1214-2154-2221", ...);
Cash cash = new Cash();
c.addPaymentMean( visa );
c.addPaymentMean( cash );

double amount = f.getAmount();
new Charge(f, cash, 100.0);
new Charge(f, visa, amount - 100.0);
```

```
Client c = new Client("1234567", "Manuel", ...);
Vehicle v = new Vehicle("1241-CKJ", "seat", "almera");
c.addVehicle( v );
WorkOrder workOrder = new WorkOrder(v, "Pre-revisión ITV");
```

Un cliente nuevo llega…

```
Mechanic m = findMechanic("Luis", "Pérez");
workOrder.assignMechanic( m );
```

La orden de trabajo se asigna al mecánico

```
Intervention i = new Intervention(workOrder, m);
SparePart r = findSparePart("DKJ.1244");
new Sustitution( i, r, 2 );
```

El mecánico la marca como finalizada

```
Long number = getNextInvoiceNumer();
Invoice f = new Invoice( number, workOrder );
```

Se le crea la factura

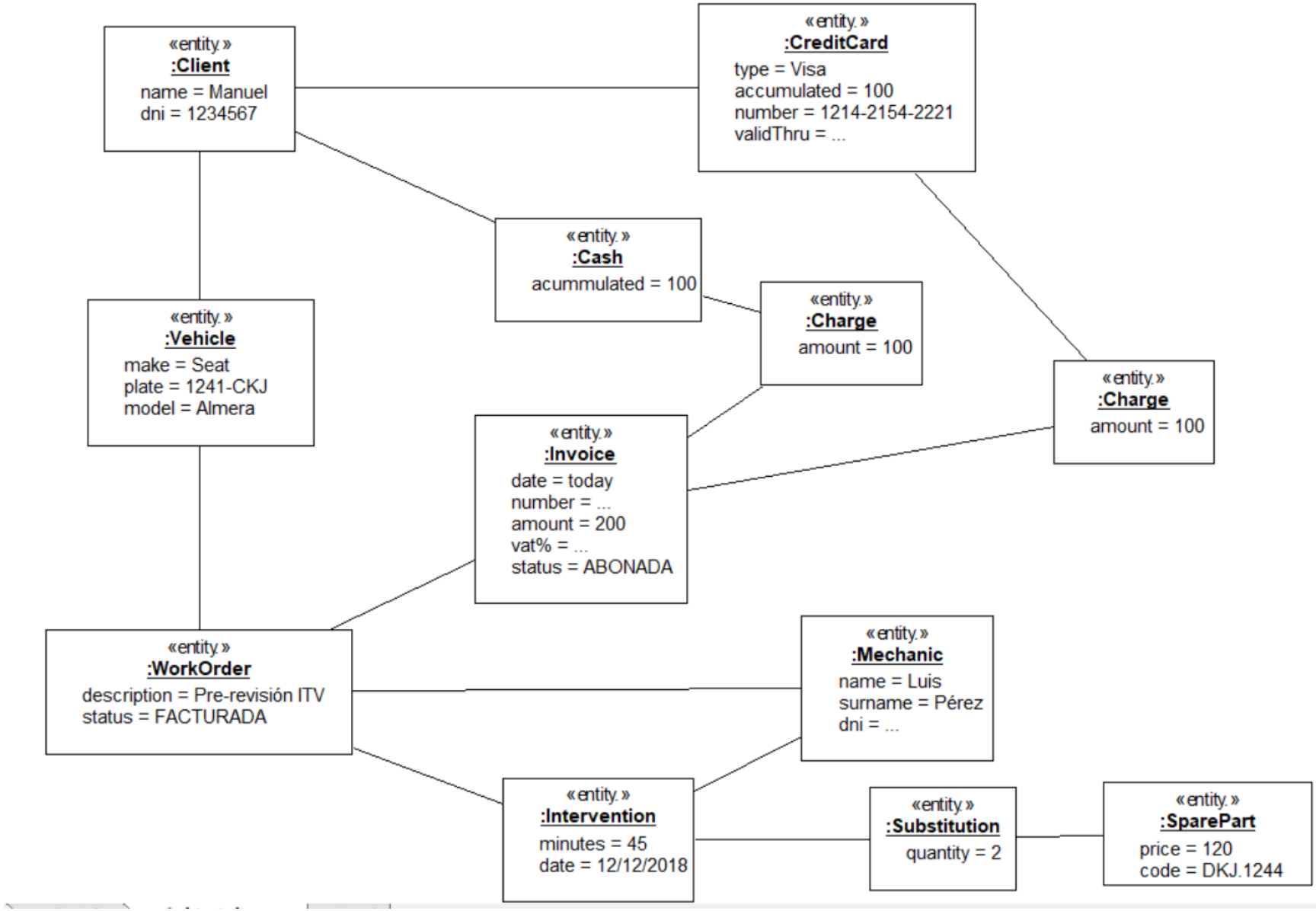
```
CreditCard visa = new CreditCard("visa", "1214-2154-2221", ...);
Cash cash = new Cash();
c.addPaymentMean( visa );
c.addPaymentMean( cash );

double amount = f.getAmount();
new Charge(f, cash, 100.0);
new Charge(f, visa, amount - 100.0);
```

Cuando el cliente pasa a recoger el vehículo…

… registra medios de pago…

… y paga con ellos


«entity.»
:Client
name = Manuel
dni = 1234567
«entity.»
:CreditCard
type = Visa
accumulated = 100
number = 1214-2154-2221
validThru = ...
«entity.»
:Cash
acummulated = 100
«entity.»
:Charge
amount = 100
«entity.»
:Vehicle
make = Seat
plate = 1241-CKJ
model = Almera
«entity.»
:Charge
amount = 100
«entity.»
:Invoice
date = today
number = ...
amount = 200
vat% = ...
status = ABONADA
«entity.»
:WorkOrder
description = Pre-revisión ITV
status = FACTURADA
«entity.»
:Mechanic
name = Luis
surname = Pérez
dni = ...
«entity.»
:Intervention
minutes = 45
date = 12/12/2018
«entity.»
:Substitution
quantity = 2
«entity.»
:SparePart
price = 120
code = DKJ.1244

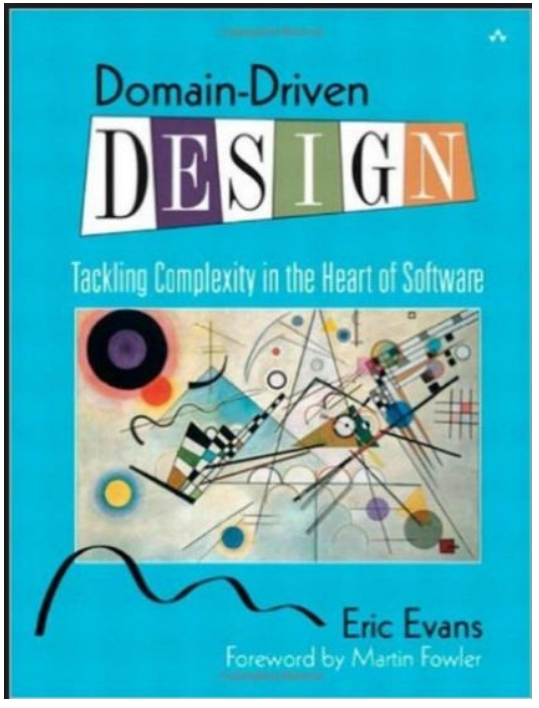
- ¿Es sencillo entender este código?
- ¿Qué hay en memoria?
- ¿Qué ocurrirá si el taller sigue funcionando, los clientes viniendo, los mecánicos reparando, etc…?
- ¿Y si se apaga el ordenador?

La orientación a objetos y la persistencia no se llevan bien.

- La persistencia es muy intrusiva…
- … y me desbarata el diseño limpio del código que me daría la orientación a objetos

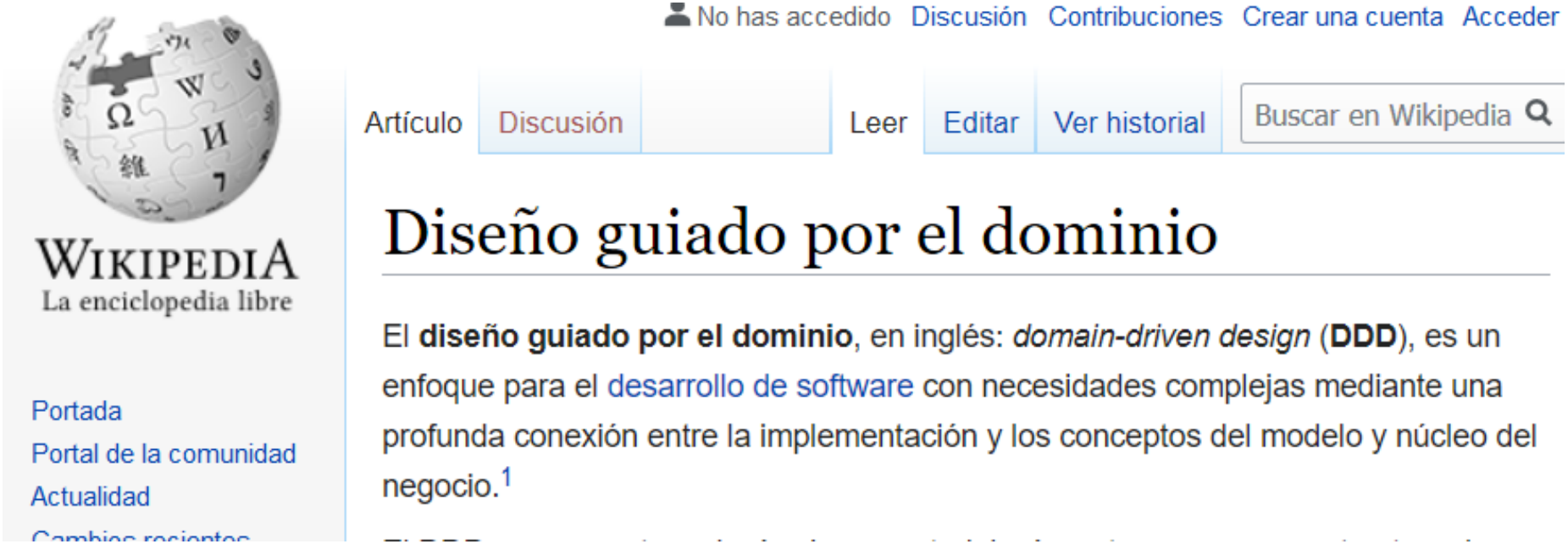
Patrón "modelo de dominio"

- Sin embargo quiero esa forma de programar
  - Es limpia
  - Más fácil de entender
  - Más fácil de mantener → más barato
  - Es reflejo directo del modelo del dominio



- Domain Driven Design
  - recomendado

DDD...


No has accedido Discusión Contribuciones Crear una cuenta Acceder
WIKIPEDIA
La enciclopedia libre
Artículo Discusión Leer Editar Ver historial
Buscar en Wikipedia
Diseño guiado por el dominio
El **diseño guiado por el dominio**, en inglés: *domain-driven design* (**DDD**), es un enfoque para el desarrollo de software con necesidades complejas mediante una profunda conexión entre la implementación y los conceptos del modelo y núcleo del negocio.[1]
Portada
Portal de la comunidad
Actualidad

Por esto…

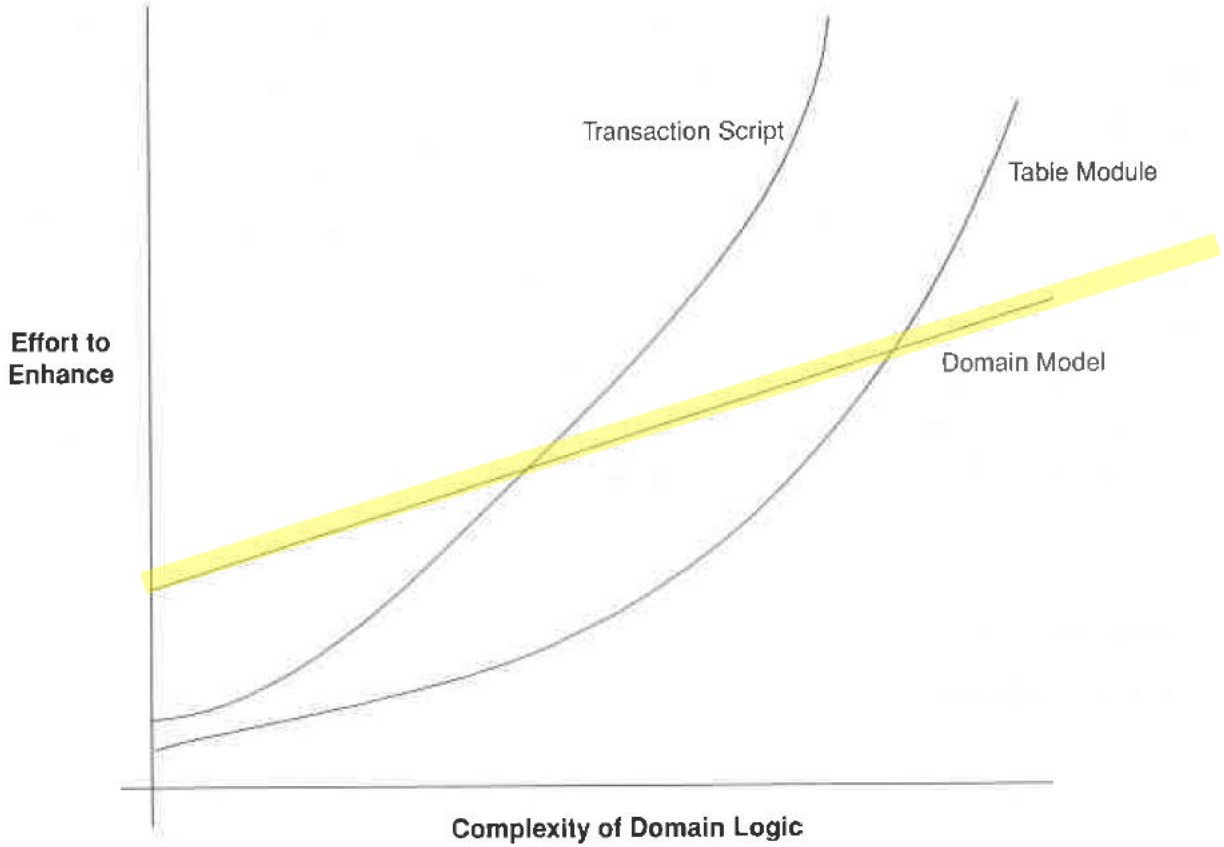


Figure 2.4 *A sense of the relationships between complexity and effort for different domain logic styles.*

Imagen tomada de (pg 29) ->