Método Desarrollo Software Dirigido por Modelos

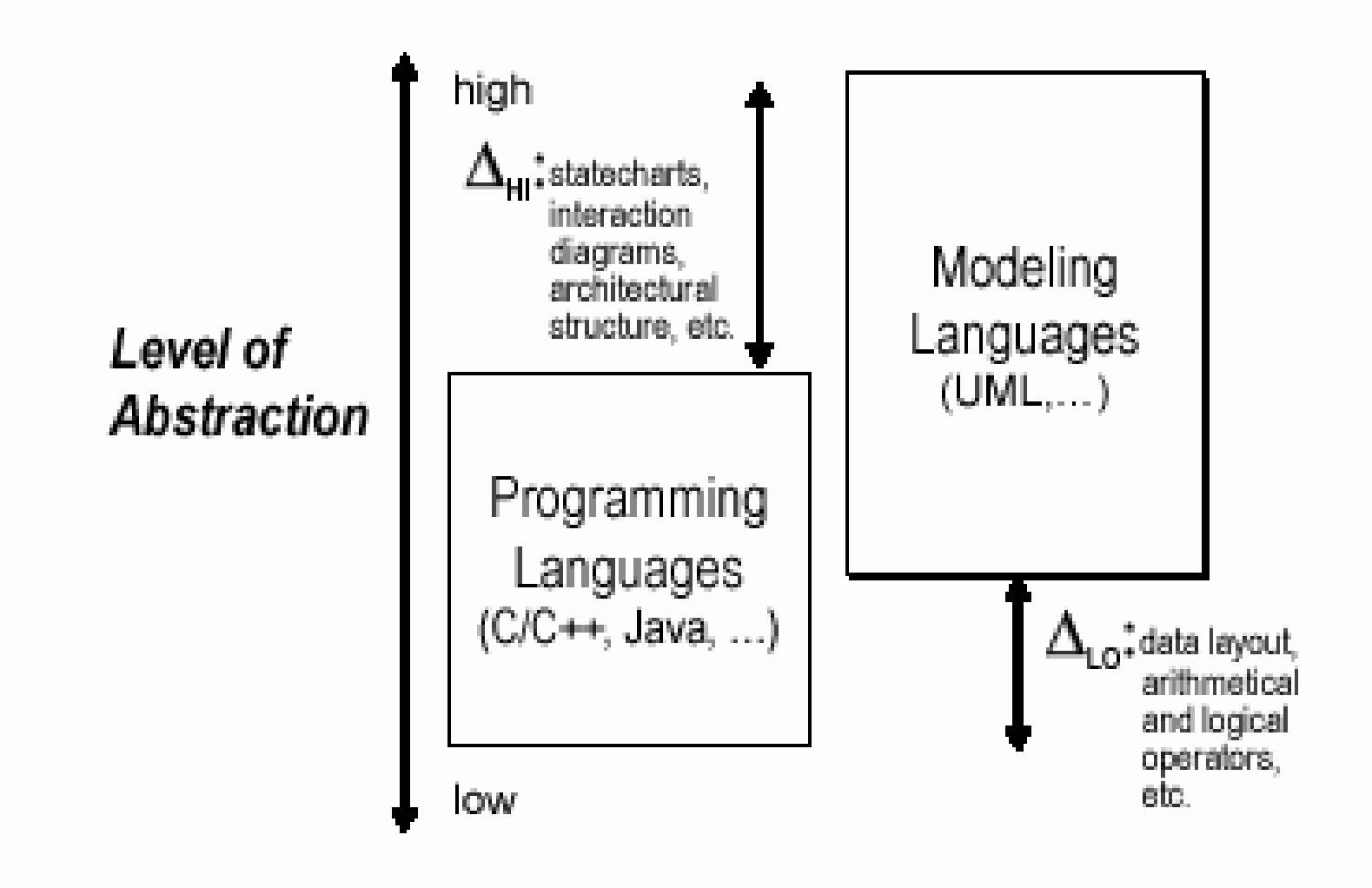
Diseño de Sistemas Software

Carlos Pérez
Santiago Meliá
Cristina Cachero

Necesidad del modelado

- Para la definición de la funcionalidad y arquitectura en la fase de diseño es necesario subir el nivel de abstracción de la implementación
- Permite la definición de una arquitectura temprana para que se valide la viabilidad tecnológica del proyecto
- Permite el paso del análisis al diseño de forma intuitiva, y posteriormente con procesos de automatización de diseño a implementación

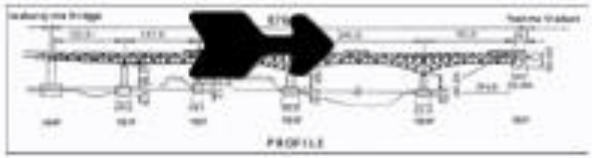
Modelado y lenguajes



La importancia de los modelos

- Before they build the real thing...

…they first build models…and then learn from them



Modelos de diseño

- Modelo de Diseño:
  - Modelo de Ingeniería que representa de forma reducida un sistema
- Propósito
  - Ayudar a comprender un problema complejo (o solución)
  - Comunicar ideas acerca de un problema o solución
  - Guiar la implementación

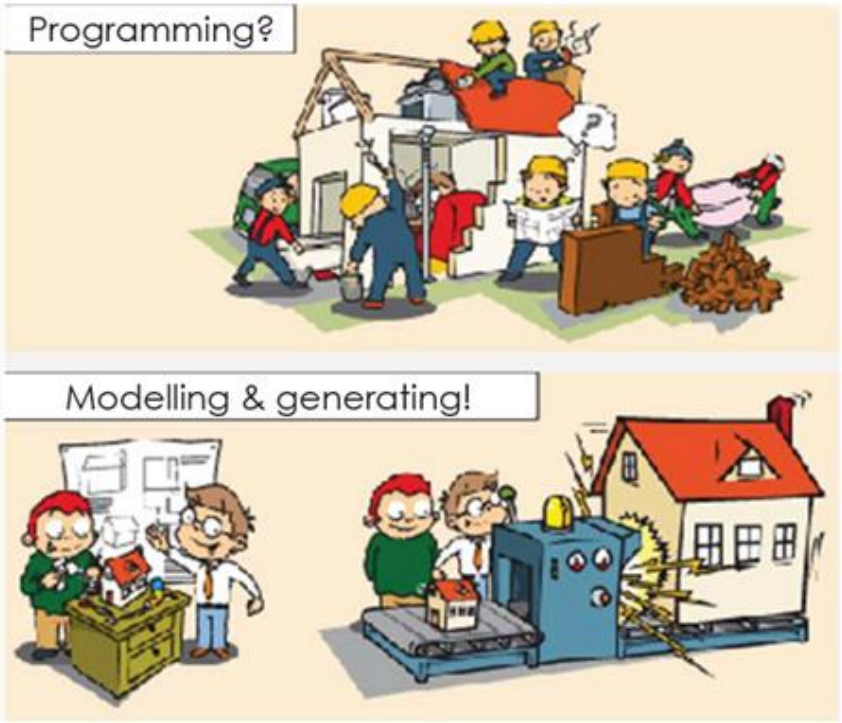
Características de los modelos

- Abstracto
  - Enfatiza los elementos importantes y oculta los irrelevantes
- Comprensible
  - Fácil de comprender por los observadores
- Preciso
  - Representa de forma fiel el sistema que modela
- Predictivo
  - Se pueden usar para deducir conclusiones sobre el sistema que modela
- Barato
  - Mucho más barato y sencillo de construir que el sistema que modela

Cómo se usan en diseño de software

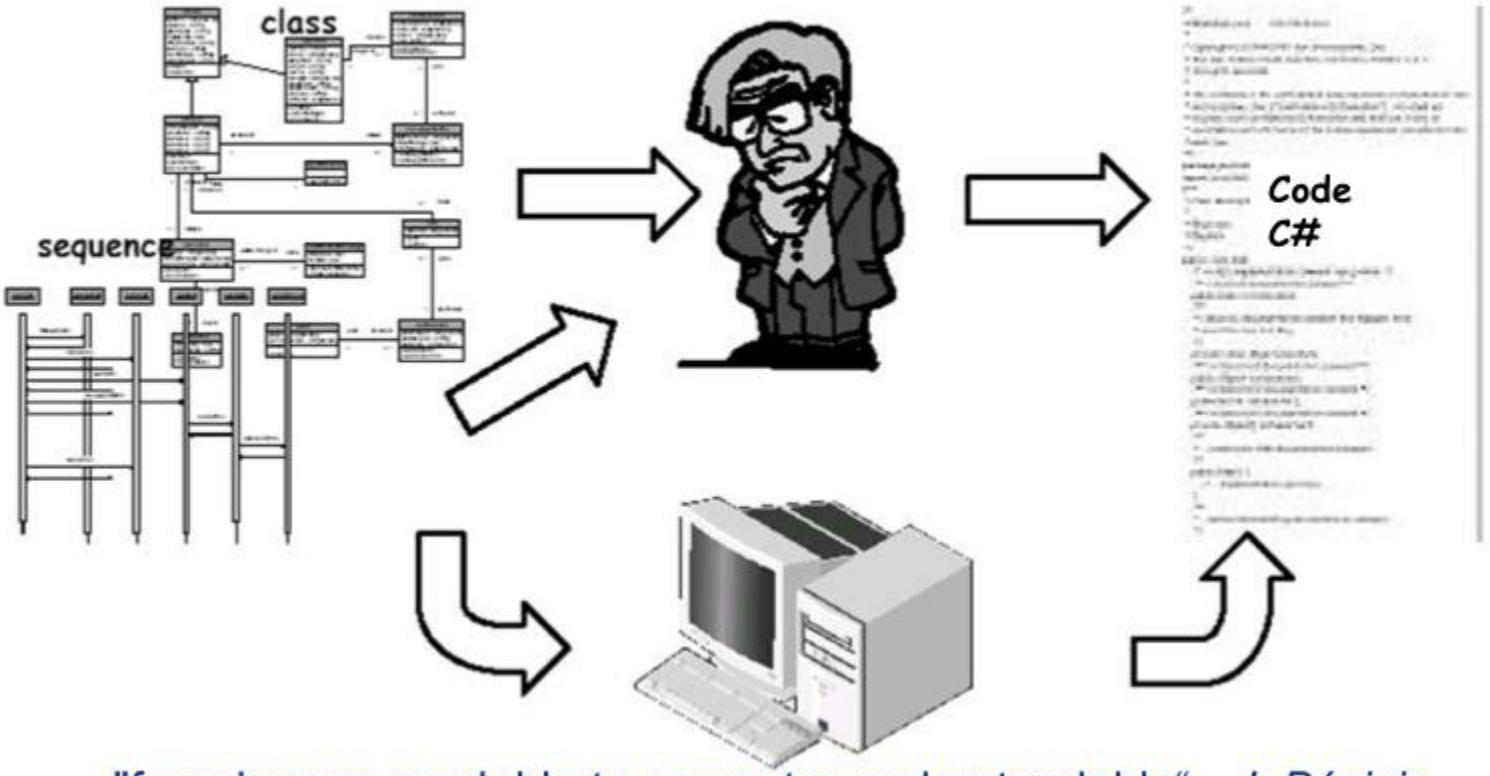
- Para detectar errores u omisiones en el diseño antes de comprometer recursos para la implementación
  - Analizar y experimentar
  - Investigar y comparar soluciones alternativas
  - Minimizar riesgos
- Para comunicarse con los «stakeholders»
  - Clientes, usuarios, implementadores, encargados de pruebas, documentadores, etc.
- Para guiar la implementación (Model-Driven Engineering)

Model-Driven Engineering



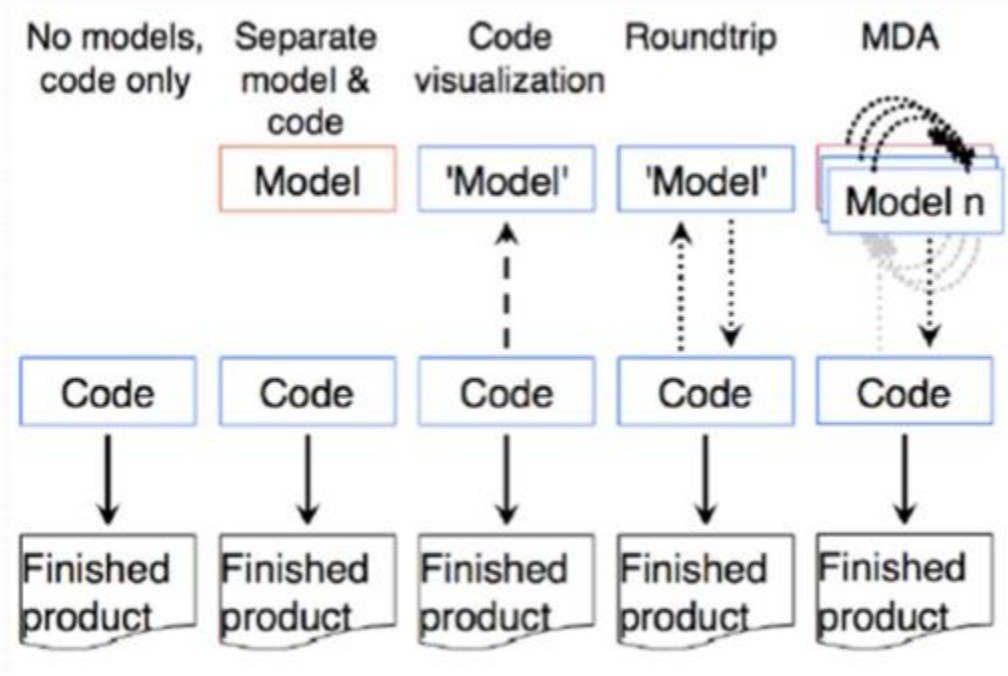
- Persigue la «industrialización» del desarrollo de software
- Utiliza los modelos para representar el problema y obtener el código final
- **El esfuerzo se centra en representar el problema, no tanto en la tecnología**

Evolución de los modelos



"from human-readable to computer-understandable“ *J. Bézivin*

Evolución de los modelos

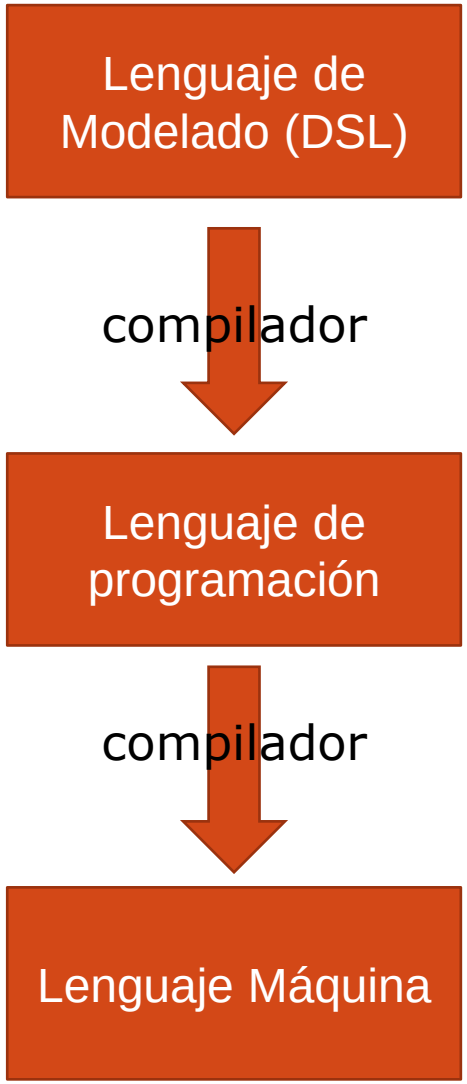


Metodologías MDE

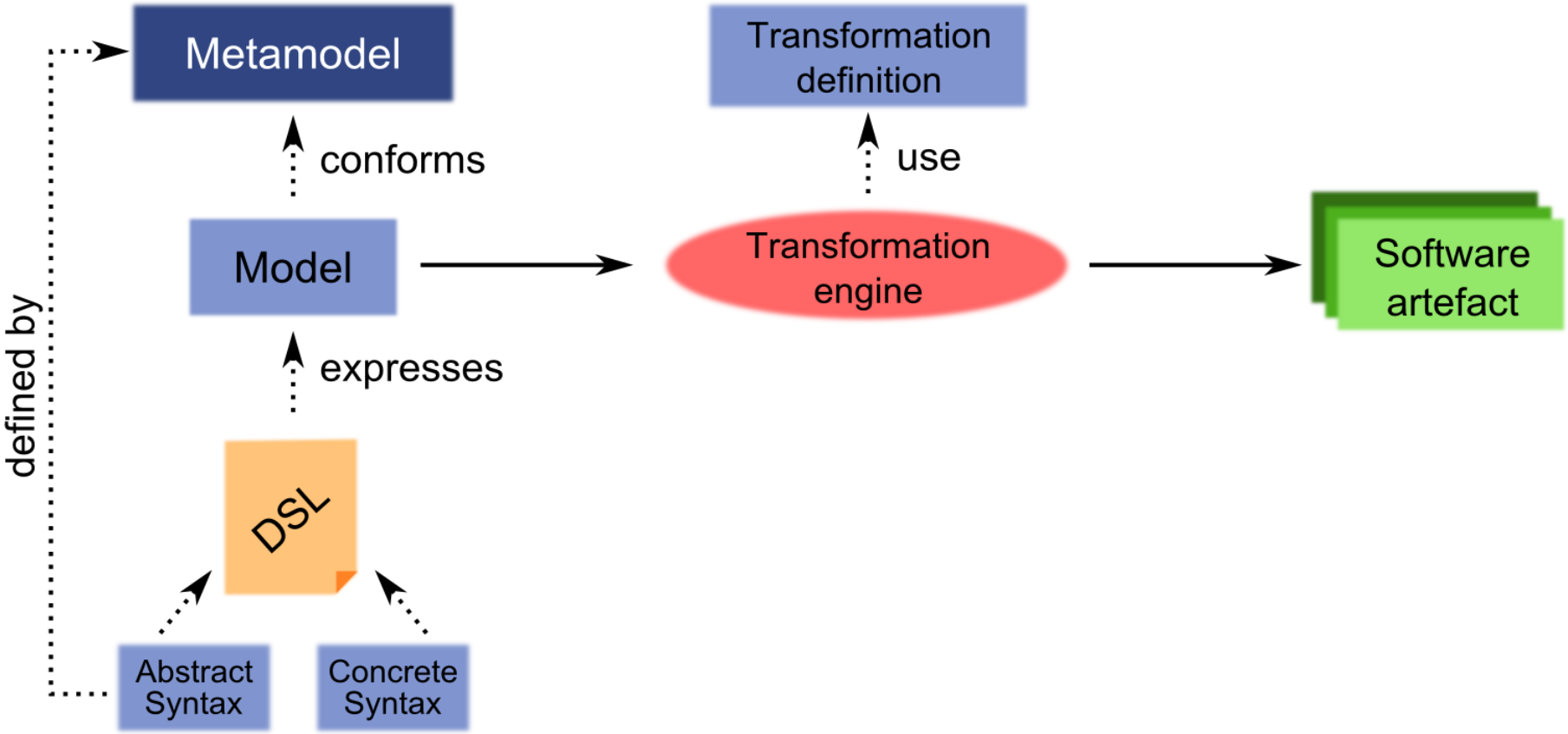
- Proporcionan:
  - Mejora la productividad (acelera el desarrollo de software)
  - Mejora la mantenibilidad
  - Reduce los errores al automatizar parte o todo el código obtenido
  - Permiten especificar con mayor precisión el modelado de un dominio concreto
- Inconvenientes:
  - Supone una mayor curva de aprendizaje para los desarrolladores
  - Coste de las herramienta MDE

Principios

- Principio general: Emplear para el desarrollo de software lenguajes lo más cercanos posible al lenguaje de dominio de la aplicación: LENGUAJES ESPECÍFICOS DE DOMINIO (DSL)
  - De más alto nivel de abstracción que los lenguajes de programación

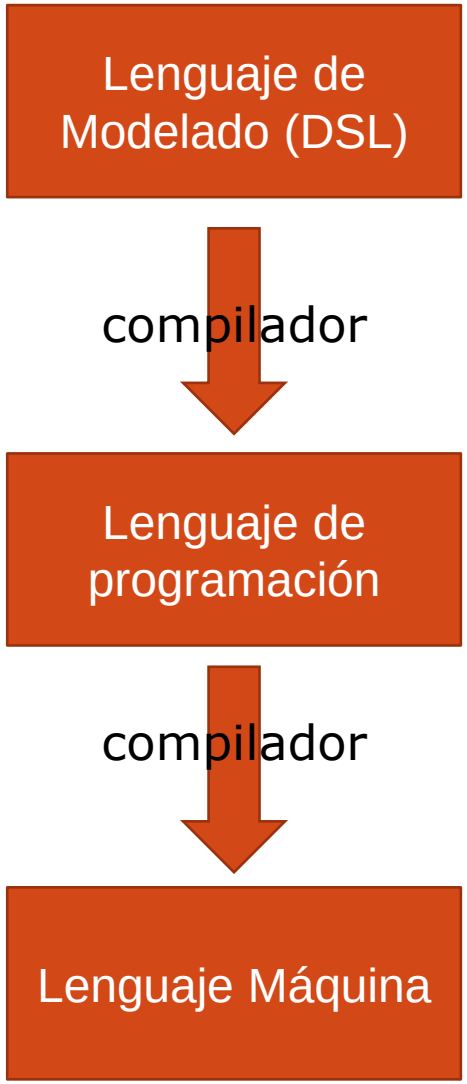


Proceso MDE

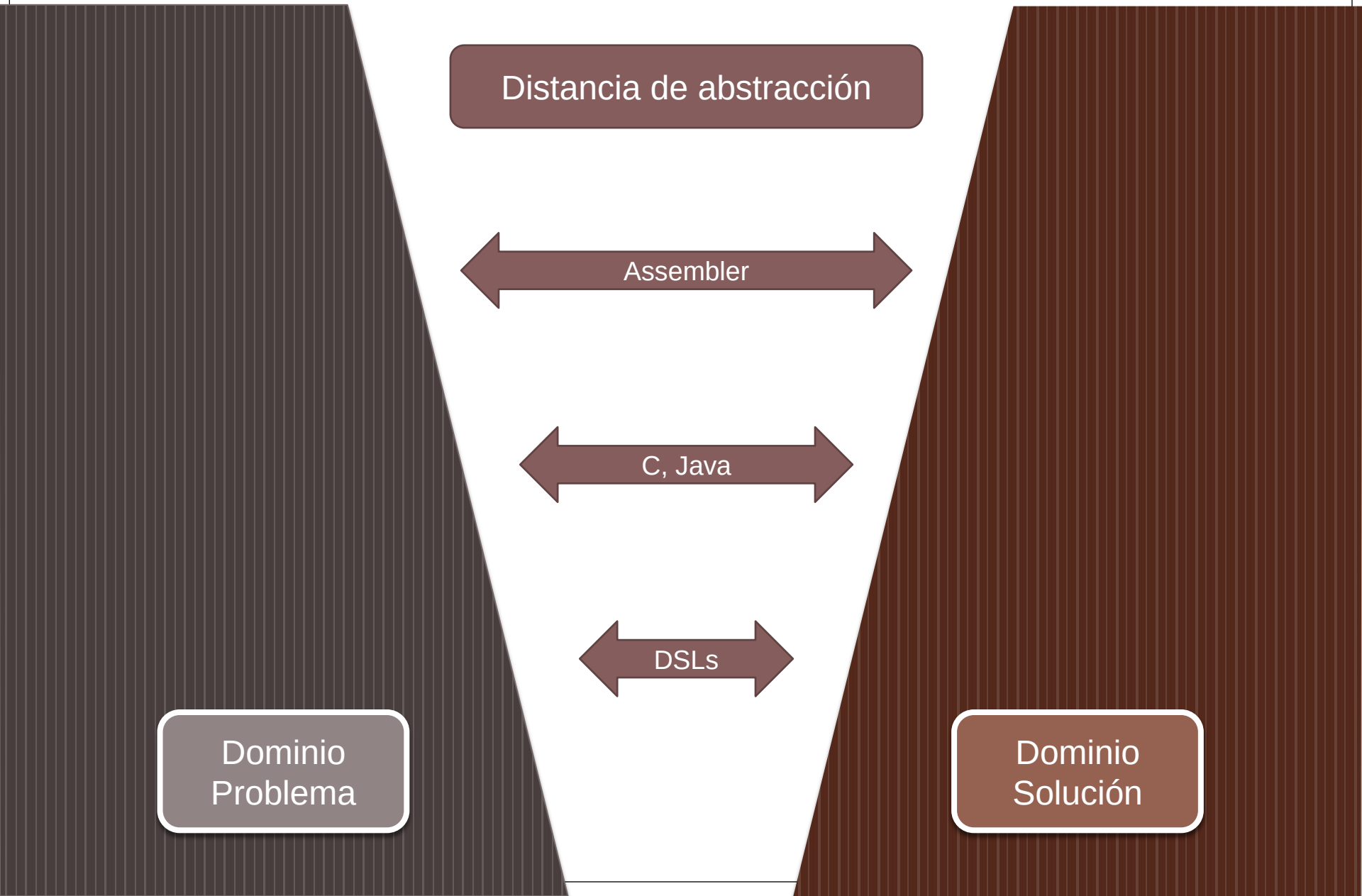

Metamodel
conforms
Model
expresses
DSL
Abstract Syntax
Concrete Syntax
defined by
Transformation definition
use
Transformation engine
Software artefact

Estrategias

- Mayores abstracciones (una instrucción equivale a más líneas de código)
- Evitación de redundancia
- Separación de preocupaciones
- Usa conceptos de dominio (mejor comunicación)



Estrategias: Reducción de la distancia problema-solución



Lenguajes Específicos de Dominio (DSL)

- Definición: Lenguaje formal procesable diseñado para abordar un aspecto determinado de un sistema software.
  - Aportan los conceptos de un dominio de aplicación al lenguaje de desarrollo
  - Permite identificar primitivas de alto nivel, al recoger conocimiento de expertos en el dominio

DSL vs GPL:

GPLs **vs** **DSLs**

DSL vs GPL: GPL's

- Java

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); }
}
```

- C

```
#include <stdio.h>
int main(void)
{
    printf("hello, world\n");
}
```

- Python

```
print("Hello world")
```

- Delphi

```
program ObjectPascalExample;
  type
    THelloWorld = object
      procedure Put;
    end;
  var
    HelloWorld: THelloWorld;
  procedure THelloWorld.Put;
  begin
    WriteLn('Hello, World!');
  end;
begin
   New(HelloWorld);
   HelloWorld.Put;
   Dispose(HelloWorld);
end.
```

DSL vs GPL: DSL's

- SQL

```
CREATE TABLE Employee (
  id INT NOT NULL IDENTITY (1,1) PRIMARY KEY,
  name VARCHAR(50),
  surname VARCHAR(50),
  address VARCHAR(255),
  city VARCHAR(60),
  telephone VARCHAR(15),
)
```

- HTML

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <p>Example</p>
  </body>
</html>
```

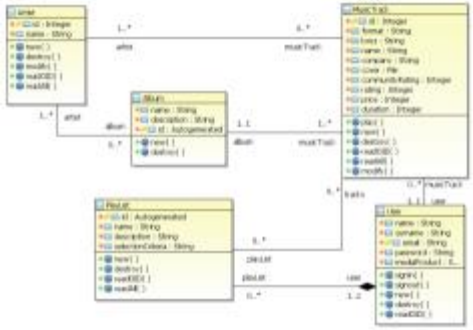
CSS

```
body {
  text-align: left;
  font-family: helvetica, sans-serif;
}
h1 {
  border: 1px solid #b4b9bf;
  font-size: 35px;}
```

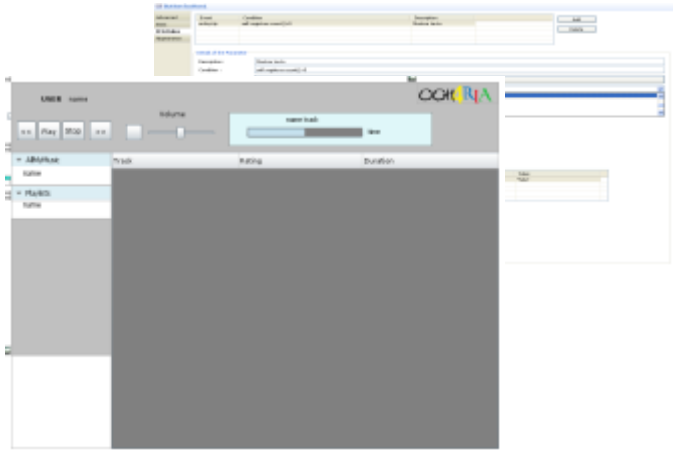
LATEX

```
\ifthenelse{\boolean{showcomments}}
        {\newcommand{\nb}[2]{
      \fcolorbox{gray}{yellow}{
    \bfseries\sffamily\scriptsize#1
                }
        {\sf\small\textit{#2}}
                }
\newcommand{\version}{\scriptsize$-$working$-$}
                }
        {\newcommand{\nb}[2]{}
         \newcommand{\version}{}
                }
```

Ejemplos de DSLs

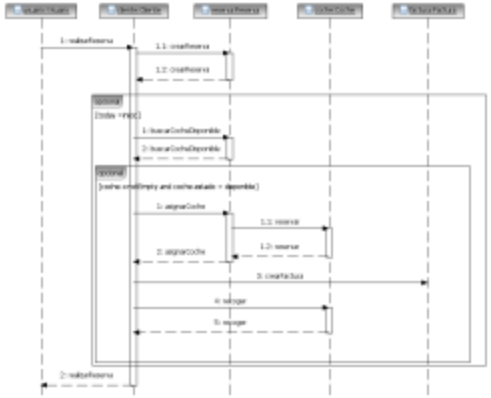


Modelo de Dominio



Modelado de Interfaz

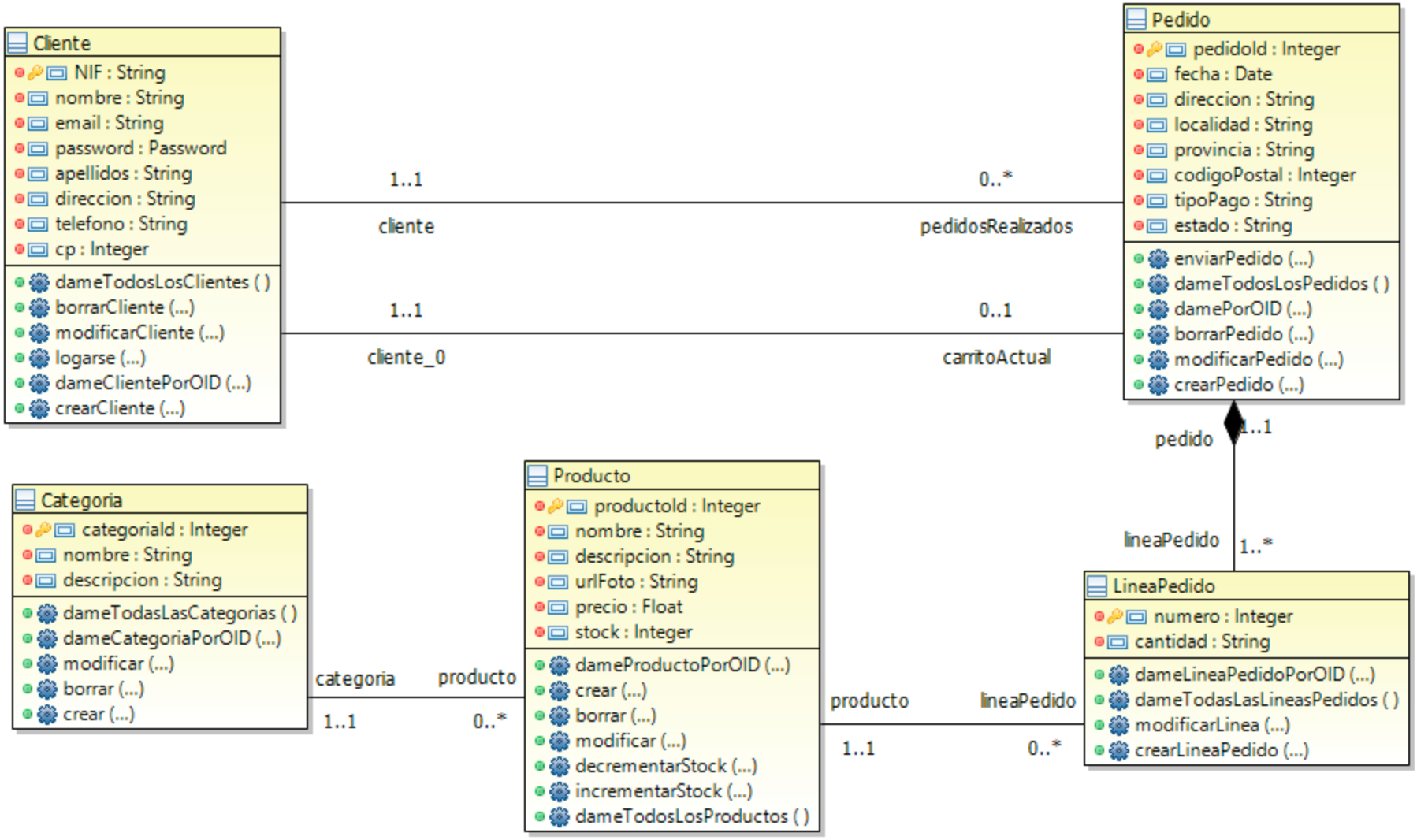
Modelo de Componentes



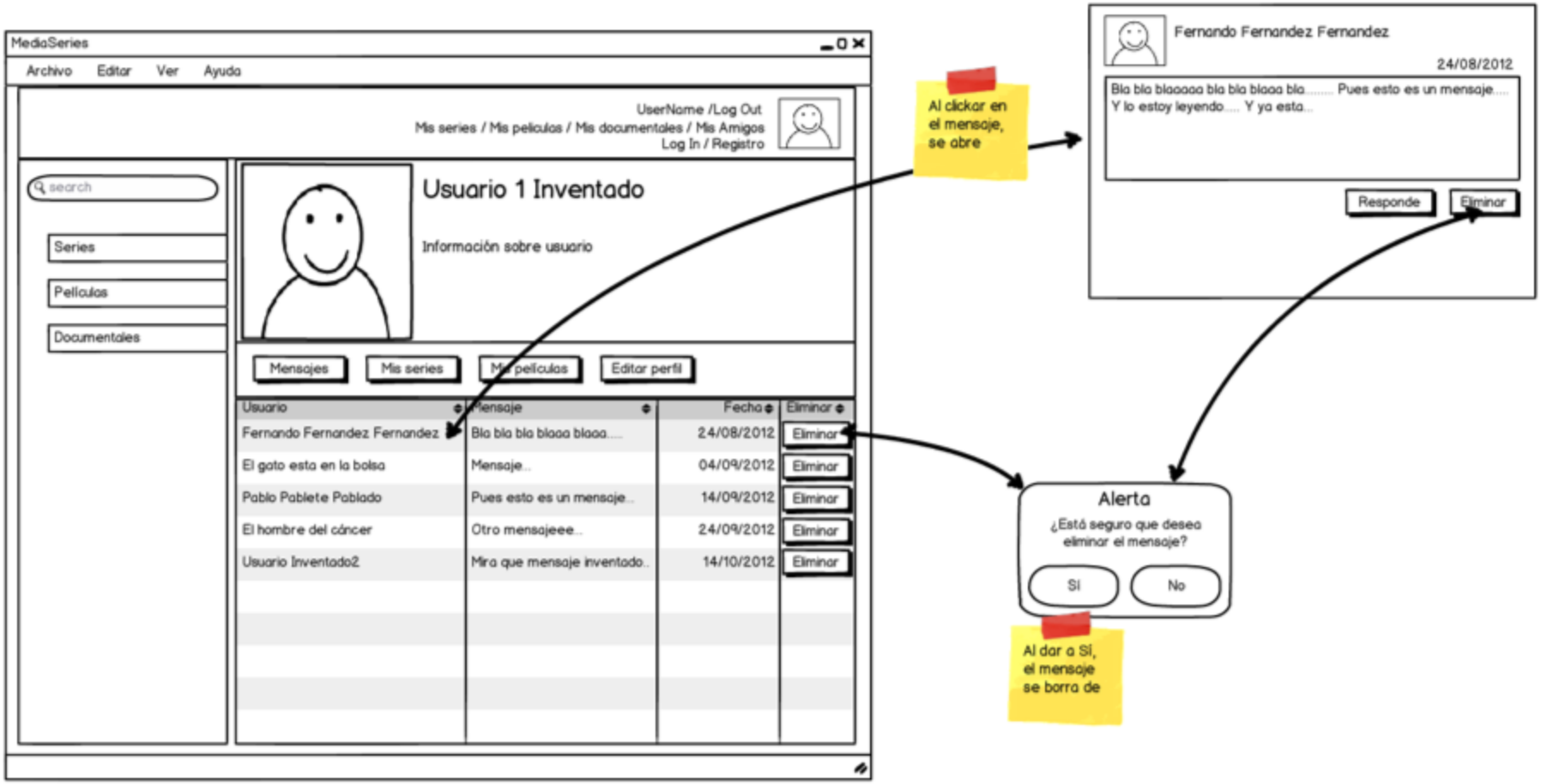
Modelo de Secuencia
para Operaciones Complejas

Modelo de
Arquitectura

Modelo de dominio

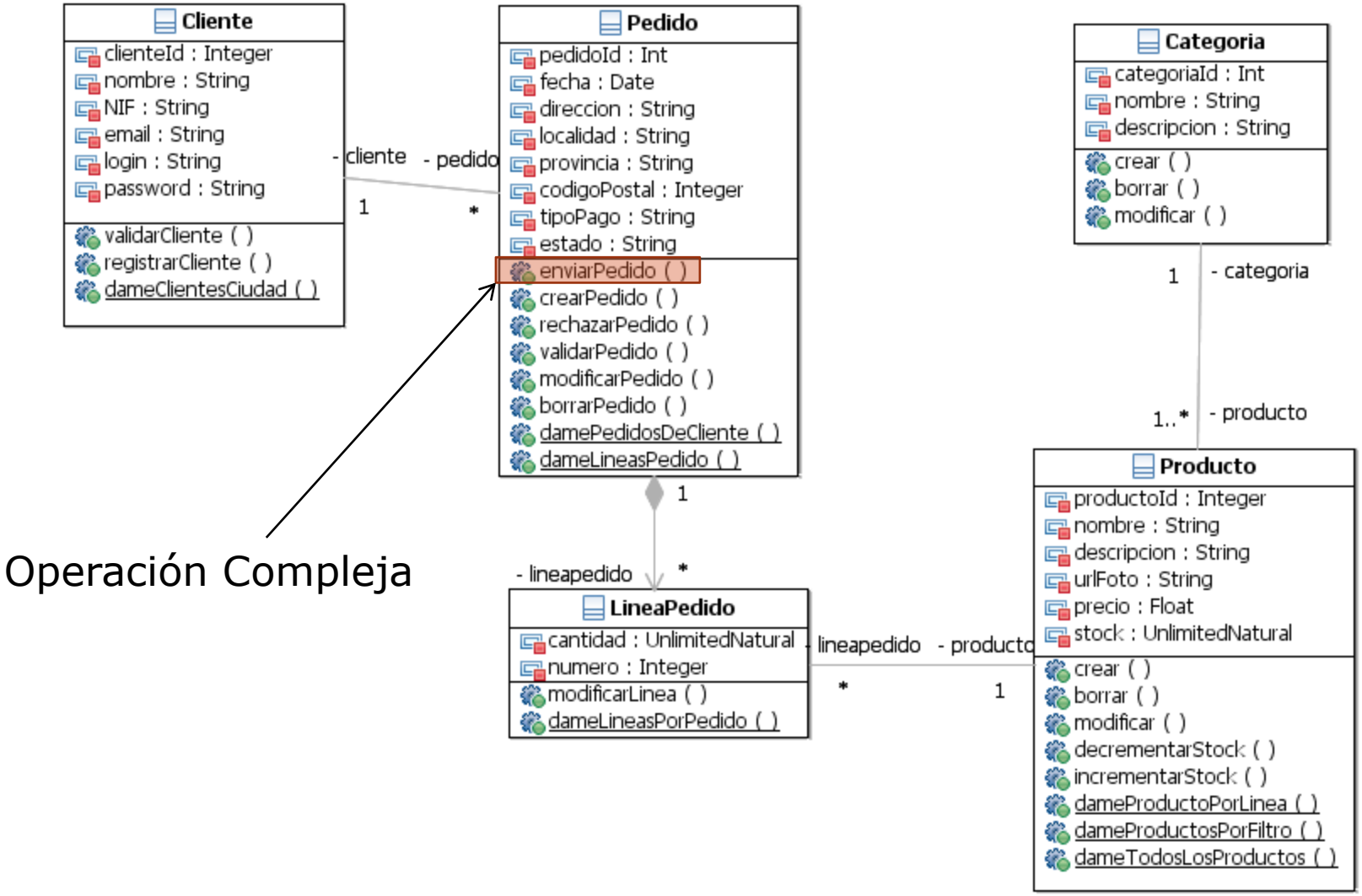


Modelado de interfaz

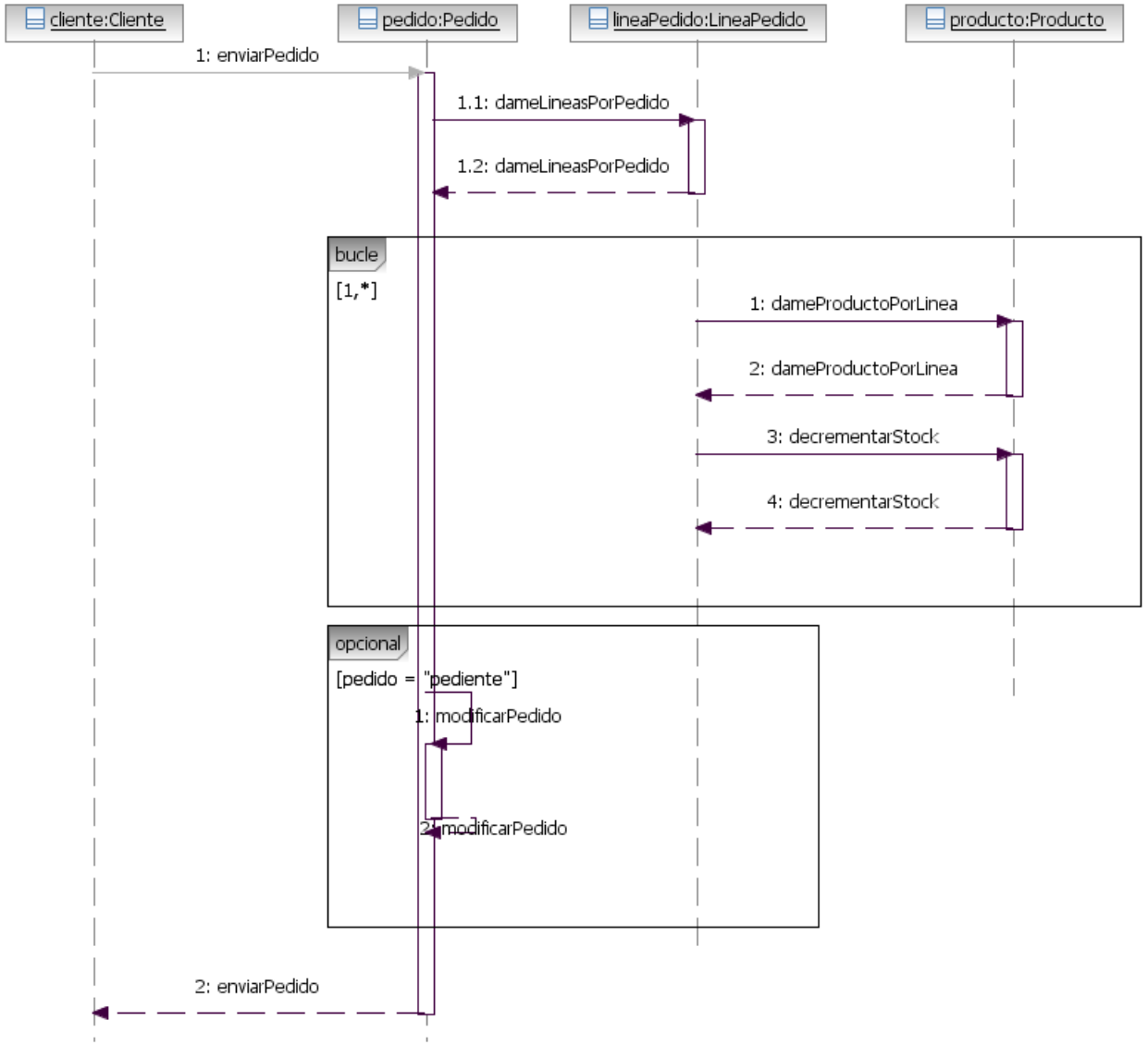


BALSAMIQ

Ejemplo de operación



Comportamiento de la operación



Aspectos a considerar

- Vamos a utilizar modelos centrados en la parte del problema (funcionales) que permiten obtener entre el 50% y el 80% del código de una aplicación
- El código restante se ha de modelar mediante diagramas de componentes y secuencia que expresen los aspectos complejos que no recoge el modelo
- A la hora de trabajar es necesario compatibilizar el código generado con el código manual

Modelo de dominio

- En el diseño orientado e objetos, el modelo de dominio es el elemento central para representar la lógica de negocio y la persistencia de las aplicaciones software [Turk 2003]. Entre sus beneficios están:
  - Facilita la detección de inconsistencias
  - Facilita la definición de restricciones en fases tempranas de diseño
  - Facilita la definición de un vocabulario común entre los desarrolladores
- Existen 2 tendencias: modelado de dominio orientado a objetos (p.ej. UML) y el modelado de dominio orientado a datos (p.ej. ER)

Modelo de dominio de OOH4RIA

- Vamos a utilizar un lenguaje específico de dominio basado en la metodología OOH4RIA que permita representar el modelo de dominio de forma no ambigua
- Permite acelerar el desarrollo generando el código de una lógica de negocio orientada a objetos y una persistencia relacional basándose en el framework Hibernate
- Permite recoger aspectos cercanos a la implementación manteniendo la apariencia de un diagrama de clases de UML
- Permite tanto una representación gráfica como una representación textual

Agile Model Driven Development

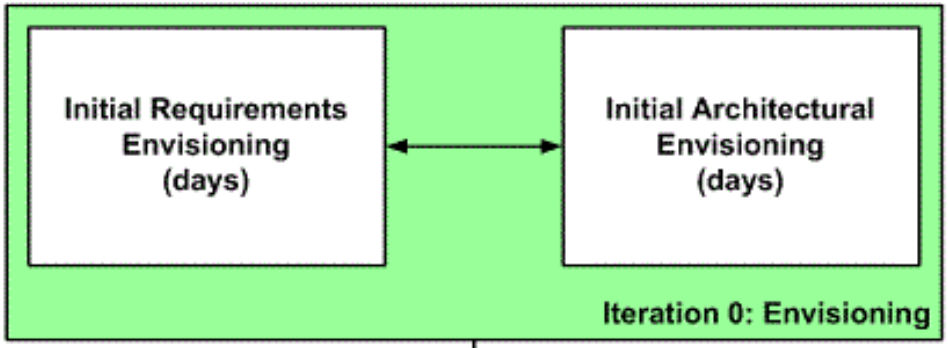
Agile Model Driven Development (AMDD)

- El proceso MDD require la creación de numerosos modelos antes de generar el código, y normalmente se sigue un enfoque en serie: análisis -> diseño -> generación de código

- Es posible seguir una aproximación ágil, creando solamente modelos que sean lo suficientemente buenos, y dividiendo el desarrollo en iteraciones

Ciclo de vida AMDD

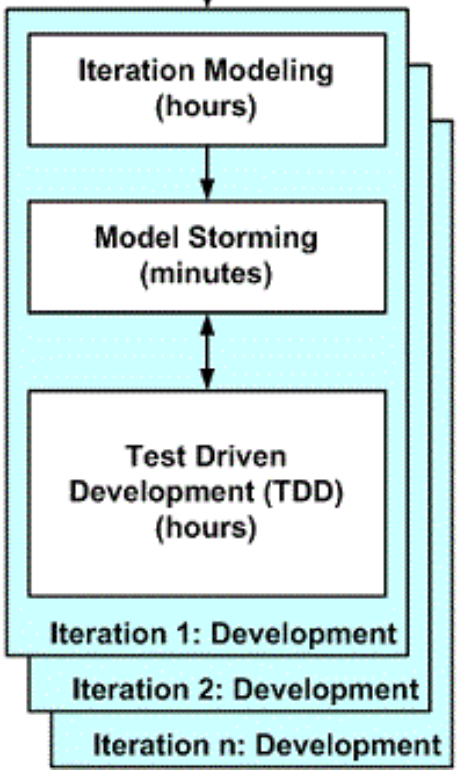
- Identify the high-level scope
- Identify initial "requirements stack"
- Identify an architectural vision



- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration

- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later

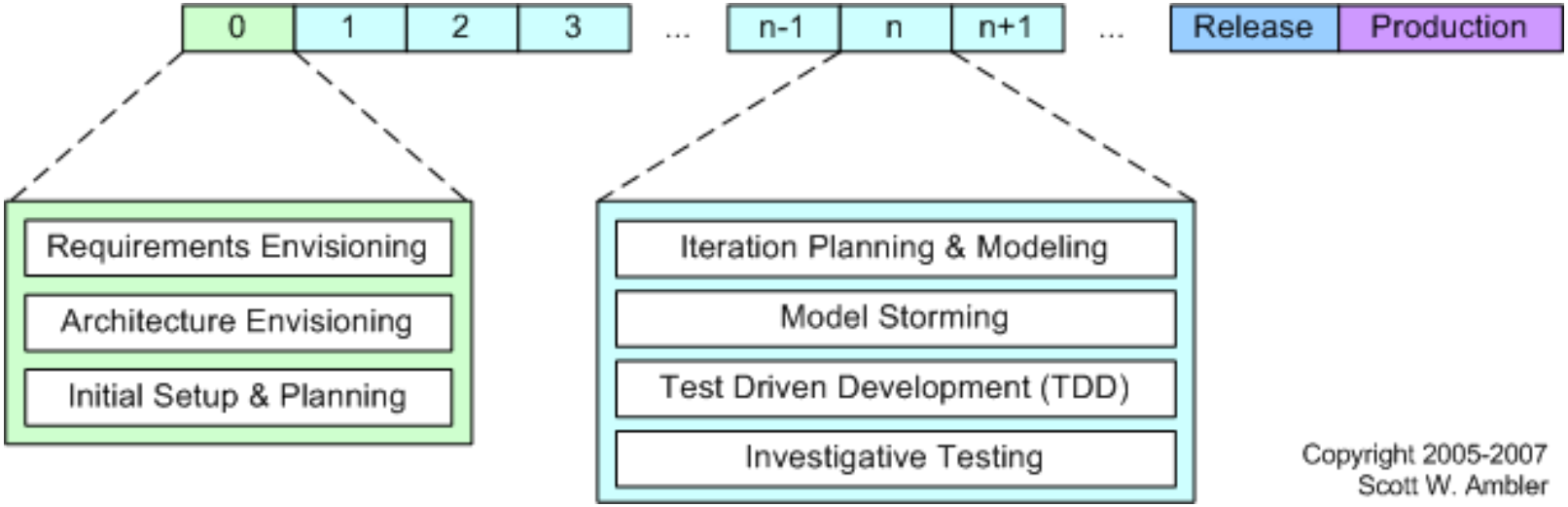
- Develop working software via a test-first approach
- Details captured in the form of executable specifications



Reviews (optional)

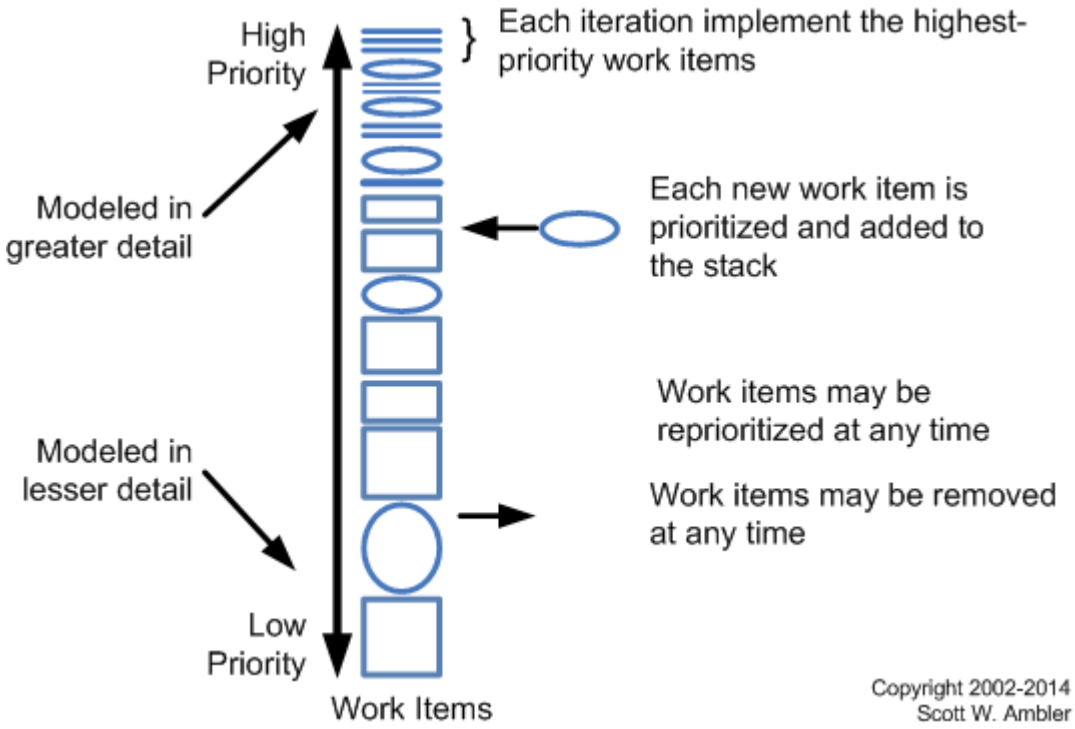
All Iterations (hours)

Ciclo de vida AMDD



Modelado en iteraciones

- En lugar de modelar todos los requisitos en una fase temprana, en cada iteración se modelan únicamente los más importantes



Bibliografía

- http://www.slideshare.net/zirrus/domainspecific-langauges. Javier Luis Cánovas Izquierdo. Feb. 2013.
- Introducción al Desarrollo Software Basado en Modelos. Juan de Lara.
- Desarrollo de Software Dirigido por Modelos. Jesús García Molina (Universidad de Murcia)
- Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development. Scott Ambler.