



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 0

Presentación del curso

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

10 de agosto de 2018

1. Aspectos Administrativos

- Curso y horario
- Ayudantes
- Objetivos del curso
- Contenidos del curso
- Canales de comunicación
- Evaluaciones

2. Introducción - Repaso

Curso y horario

Profesor	Rodrigo Saffie
Correo	rasaffie@uc.cl
Horario de clases	Viernes M4-M5 (14:00 – 16:50)
Sala de clases	C403
Requisitos	IIC2143

Ayudantes

David Galemiri	dagalemiri@uc.cl
Nicolás Gebauer	negebauer@uc.cl
Harold Müller	hlmuller@uc.cl
Oscar Ríos	orrios@uc.cl

Objetivos del curso

- Realizar diseño y programación orientados a objetos con pericia
- Evaluar la calidad de un diseño, utilizando criterios teóricos y prácticos
- Analizar software para mejorar su eficiencia, confiabilidad y mantenibilidad
- Utilizar patrones de diseño en el desarrollo de software

Contenidos del curso

- ¿Qué es el diseño detallado?
- Principios fundamentales del diseño detallado
- Patrones de diseño
- *Frameworks* y librerías
- *Code Smells*
- Métricas de calidad
- *Testing*
- *Refactoring*
- Ingeniería reversa

Canales de comunicación

- **GitHub:** <https://github.com/IIC2113-2018-2>
- **Siding**

Evaluaciones

- **Controles / Lecturas / Actividades:**
 - En horario de clases
 - Se notificarán con anticipación
- **Interrogaciones:**
 - 2 interrogaciones en horario de clases (fechas por definir)
 - 1 examen (fecha por definir)
- **Tareas:**
 - 3 tareas (fechas por definir)

¿Qué es el diseño detallado de software?

¿Qué es el software?

“Se conoce como *software* al soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados *hardware*.”

[Wikipedia, 2018]

¿Qué es el software?

“Software is: (1) instructions (computer programs) that when executed provide desired features, function and performance; (2) data structures that enable the programs to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.”

[R. S. Pressman, 2009, *Software Engineering: A Practitioner’s Approach*]

¿Qué es el software?

- No se manufactura, se desarrolla

“[...] software systems continually change, new software systems are built from the old ones, and [...] all must interoperate and cooperate with each other”

[Dayani-Fard, 1999]

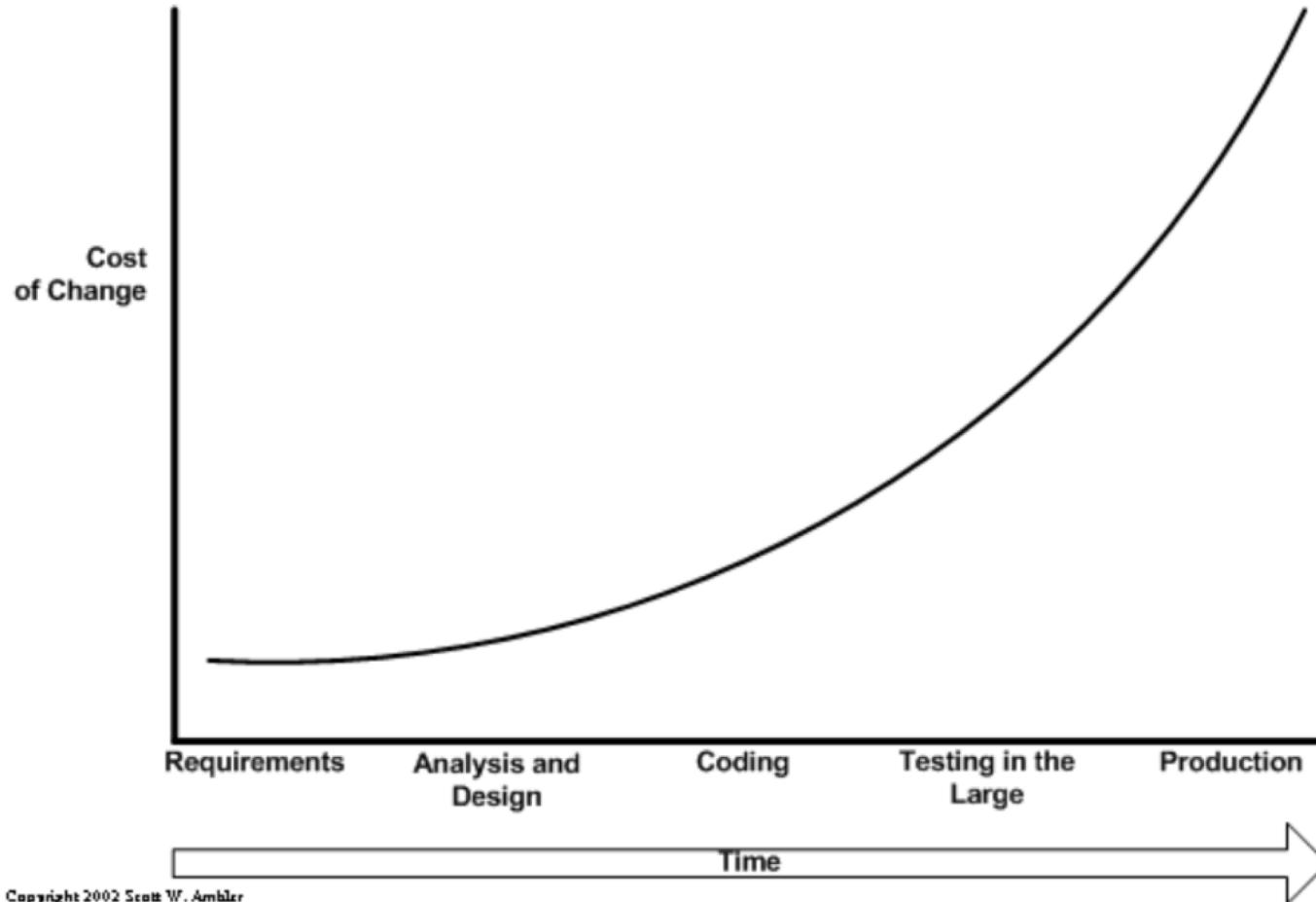
Etapas del desarrollo de software

- Análisis
- Planificación
- Diseño
- Construcción
- Implementación
- Mantención

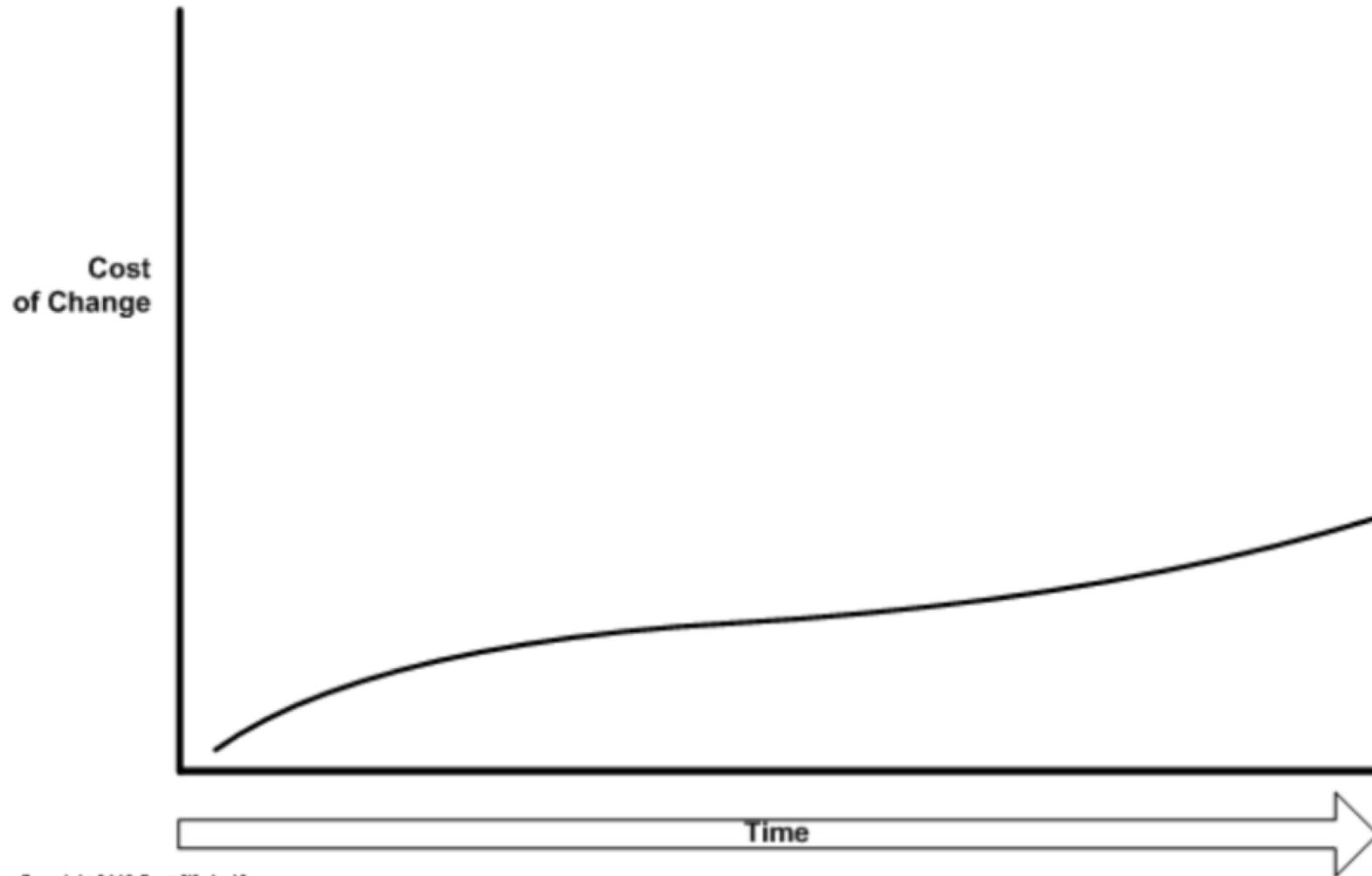
Metodologías de desarrollo

- Cascada
- Espiral
- Iterativo
- Incremental
- Prototipado
- Ágil:
 - XP
 - SCRUM

Costo tradicional



Costo ágil



¿Qué es el software?

- No se manufactura, se desarrolla
- No se fatiga, se vuelve obsoleto

Legacy software

“Es un sistema informático que ha quedado anticuado pero que sigue siendo utilizado y no se quiere (o no se puede) reemplazar o actualizar de forma sencilla”

[Wikipedia, 2018]

Puede:

- No presentar documentación
- Tener código incomprensible
- No presentar control de cambios
- Entre otros...

<https://github.com/Droogans/unmaintainable-code>

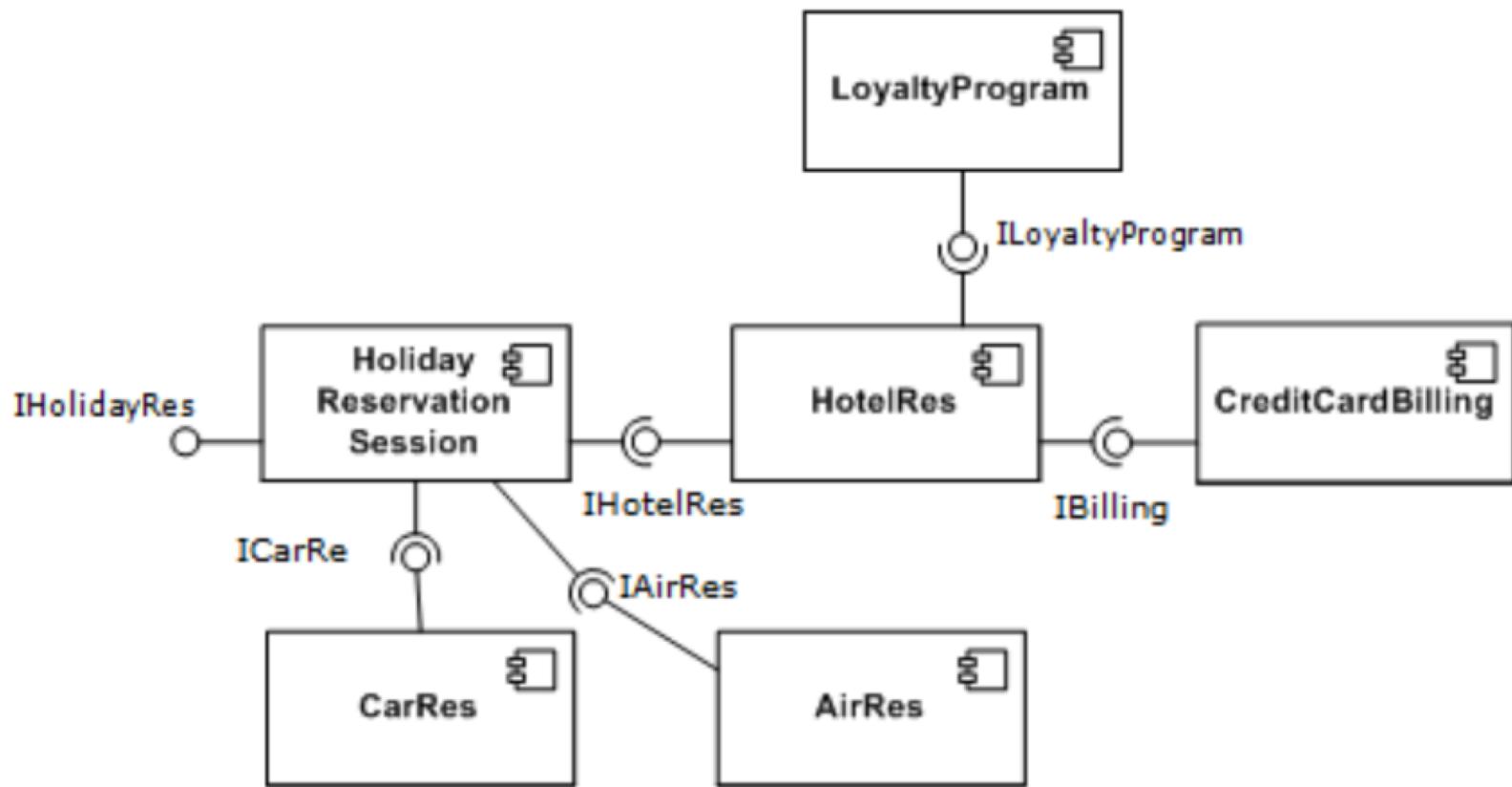
Diseño de software

¿Qué es el diseño de software?

- Un concepto reciente (70 años)
- Traduce los requisitos a especificaciones técnicas
- Se divide en 4 áreas:
 - Diseño de componentes
 - Diseño de arquitectura
 - Diseño de clases/datos
 - Diseño de interfaces

Diseño de componentes

Descripción de los componentes del sistema



Diseño de componentes

¿Qué es un componente?

Depende del punto de vista:

- Vista orientada a objetos
- Vista tradicional
- Vista orientada a procesos

Diseño de componentes

Vista orientada a objetos

Es un conjunto de clases que colaboran:

- Incluyen atributos y operaciones relevantes
- Definen interfaces para la comunicación

Diseño de componentes

Vista tradicional

Un modulo es un componente funcional del sistema:

- Tiene lógica del proceso
- Datos y estructuras para ejecutar lógica
- Una interfaz para ser invocado

Diseño de componentes

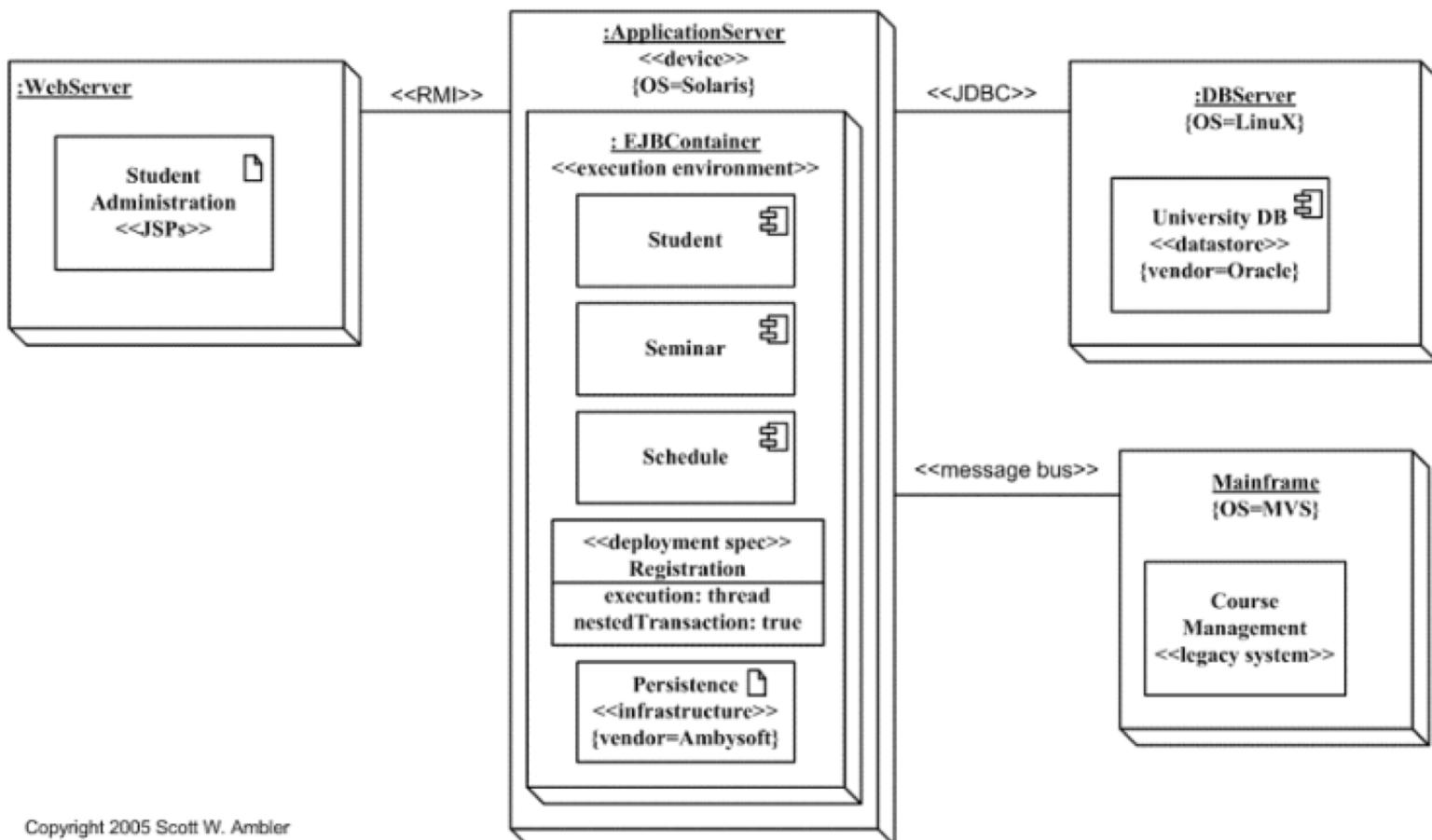
Vista orientada a procesos

Componentes que resuelven necesidades recurrentes:

- Reutilizables
- Especializados y descritos completamente
- Patrones de diseño

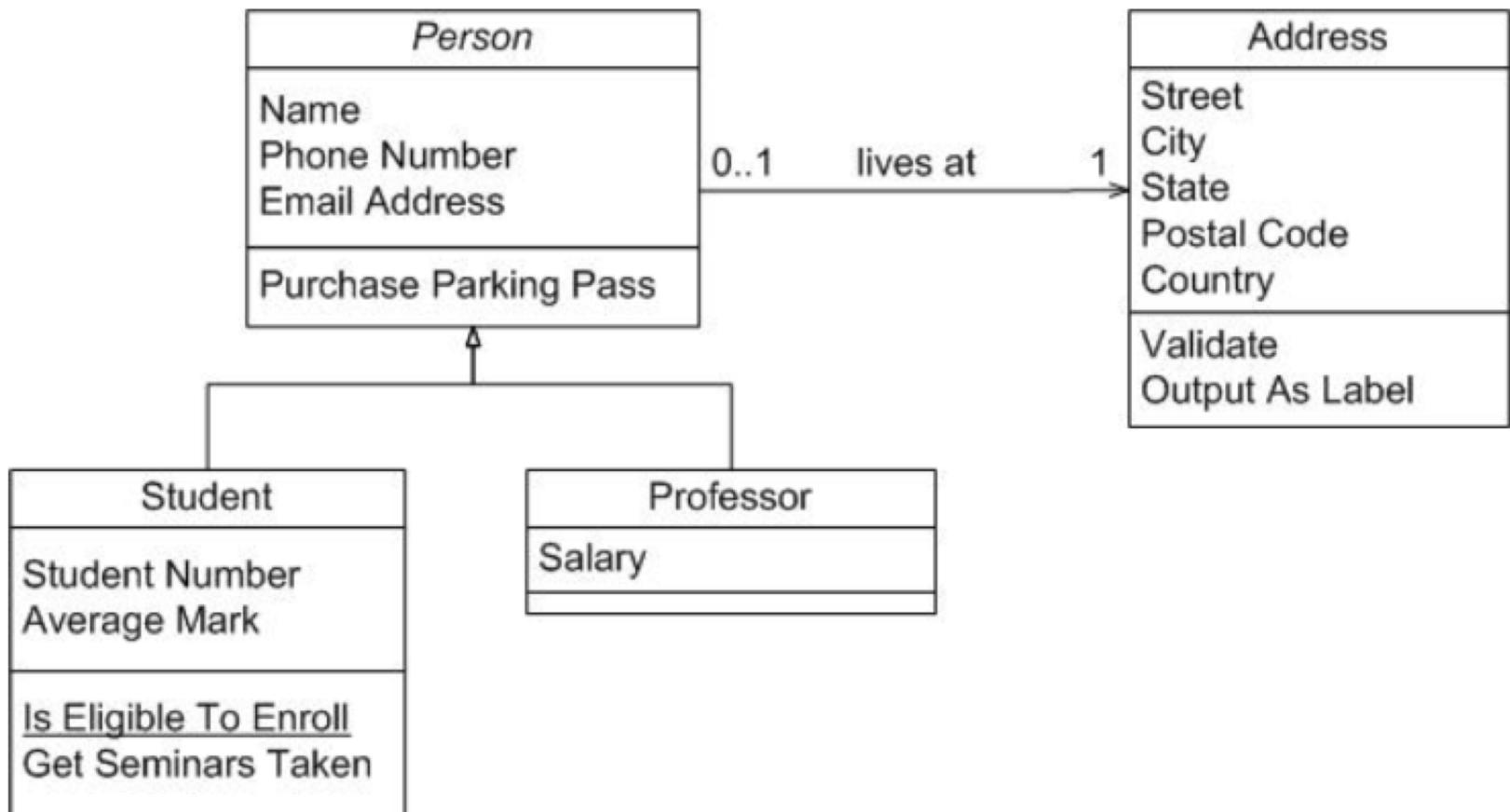
Diseño de arquitectura

Relaciones entre los componentes del sistema



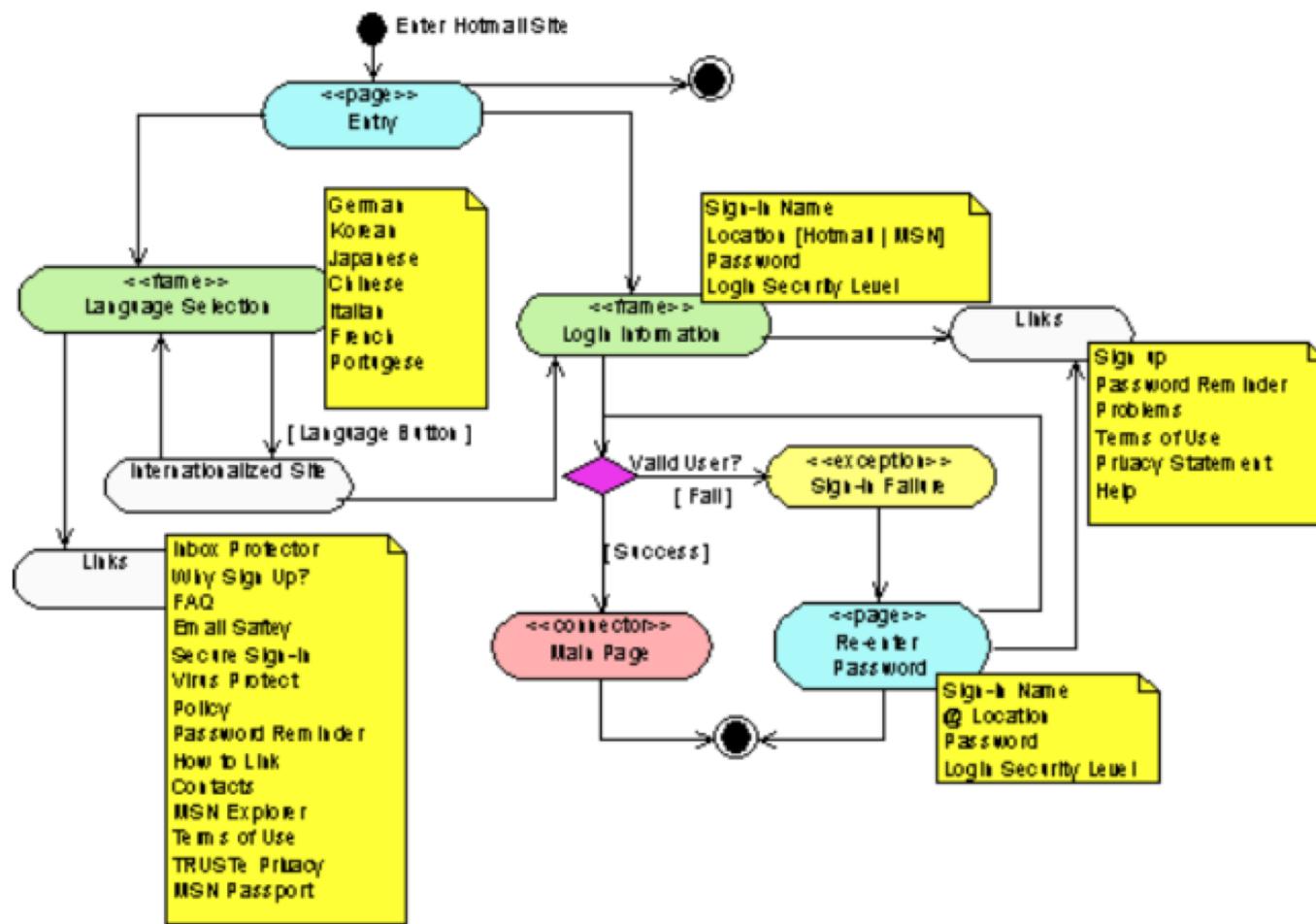
Diseño de clases/datos

Esquema de clases y sus relaciones



Diseño de interfaces

Comunicación con sistemas externos (y humanos)



Diseño de software

¿Qué cualidades debe tener la descripción del diseño de un software?

- Sin ambigüedades, precisa y objetiva
- Convenciones claras para los involucrados
- Modificable, con menor costo que la construcción

Diseño de software

¿Quiénes están interesados en la descripción del diseño?

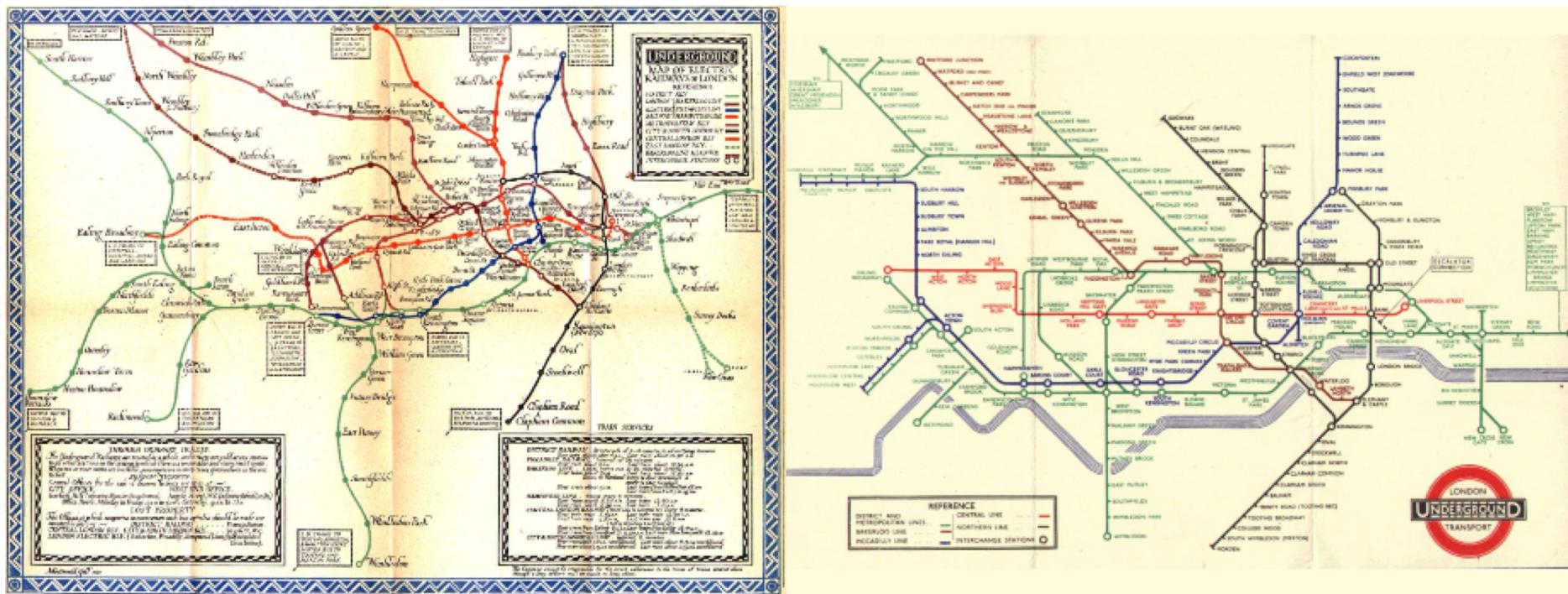
- Desarrolladores
- Líderes de proyectos
- Arquitectos de sistemas
- Analistas
- Clientes
- Usuarios

Principios del diseño de software

- Abstracción
- Ocultamiento
- Cohesión
- Acoplamiento

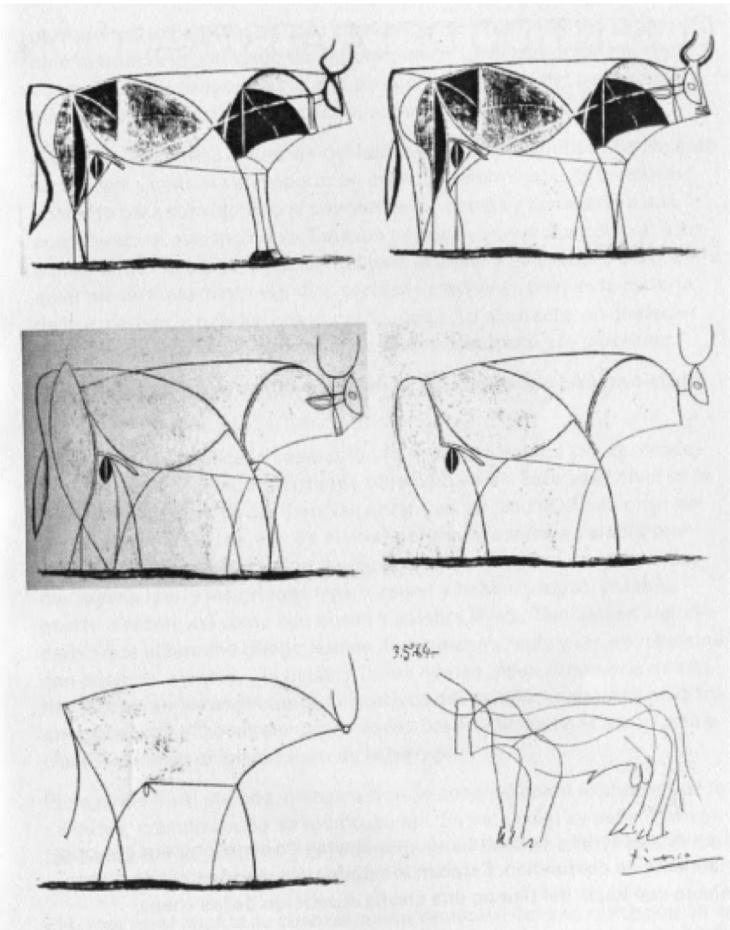
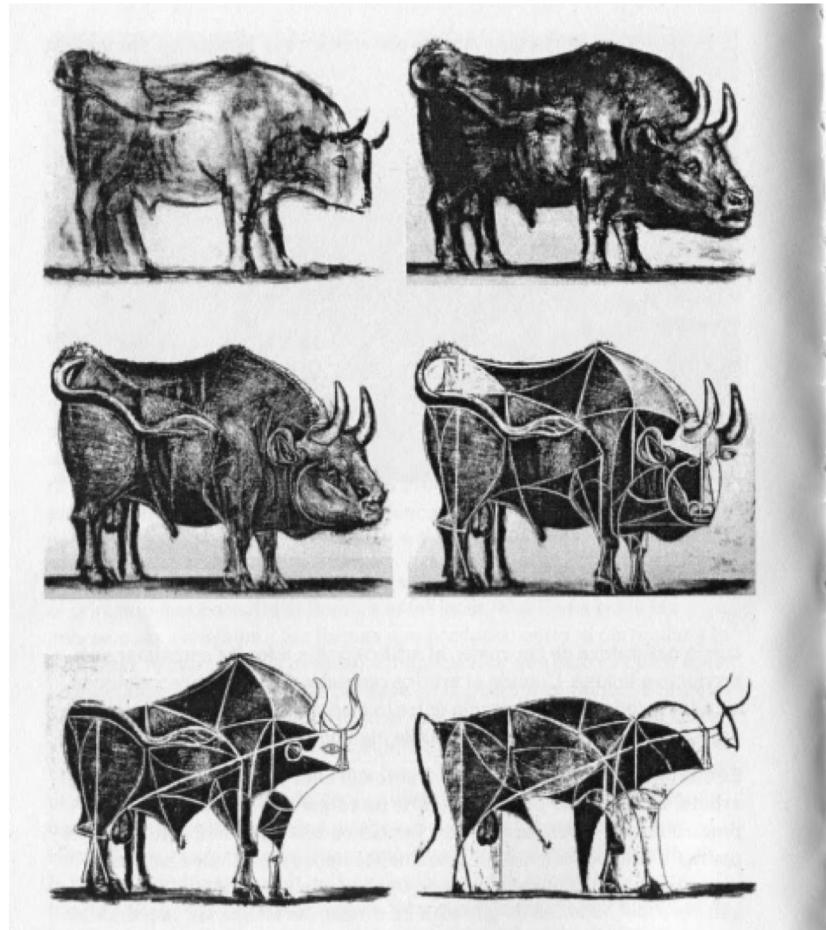
Abstracción

Rescatar información relevante dado un contexto



Abstracción

Rescatar información relevante dado un contexto



Ocultamiento

No exponer información o lógica innecesaria

Ejemplos

- Servicios Web
- Librerías
- Módulos con modificadores de acceso (*public, private*)

Ocultamiento

Beneficios

- Independencia
- Mantenibilidad
- Reusabilidad
- Extensibilidad

Cohesión

Medida de cuán relacionados están los datos, responsabilidades y métodos de una clase

Beneficios

- Reduce complejidad
- Aumenta mantenibilidad

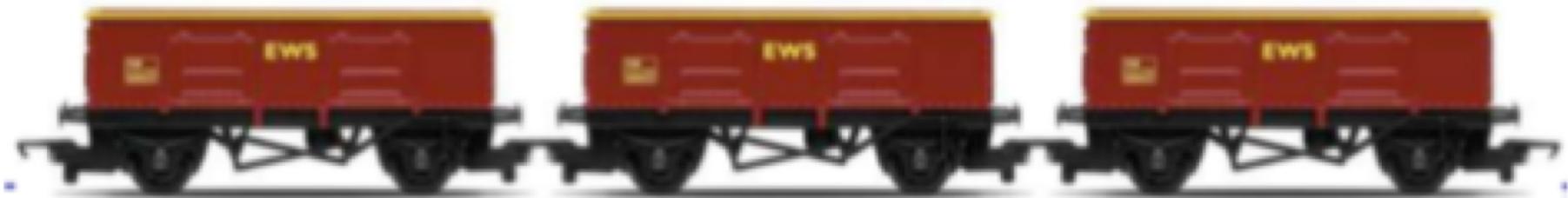
Acoplamiento

Medida de cuán conectados están dos subsistemas o clases

```
class Warehouse
  def sale_price(item)
    (item.price - item.rebate) * @vat
  end
end
```

Bajo Acoplamiento, Alta Cohesión

Cohesión y Acoplamiento en un tren



- Carros acoplados mediante interfaz simple y pequeña
- Separación en carros permite separar contenidos cohesionados

¿Por qué alta cohesión y bajo acoplamiento?

- Interfaces simples
- Comunicación simple
- Cambios afectan a sectores limitados del código
- Aumenta reusabilidad
- Aumenta extensibilidad



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 0

Presentación del curso

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

10 de agosto de 2018