



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

Clase 1

Diagramas del diseño

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

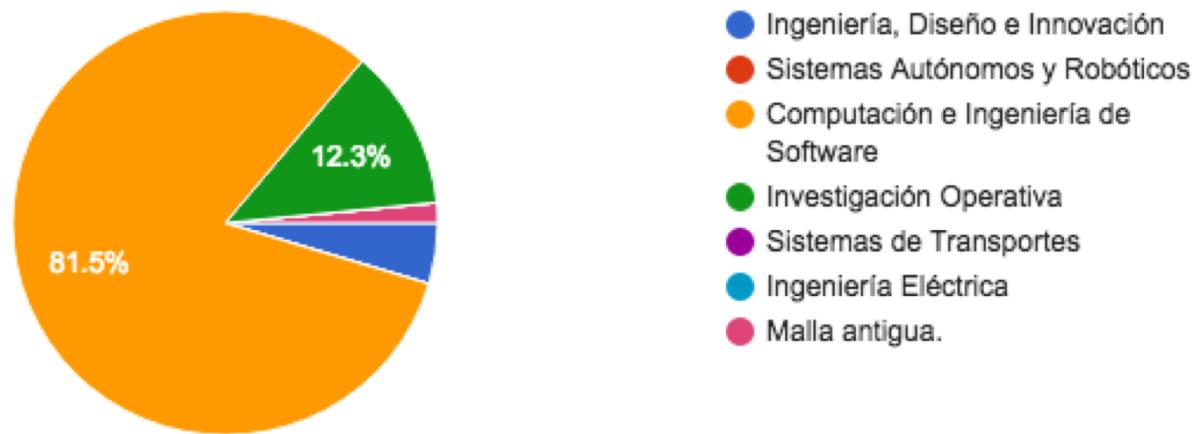
17 de agosto de 2018

Evaluaciones

- **Controles / Lecturas / Actividades:**
 - En horario de clases
 - Se notificarán con anticipación
- **Interrogaciones:**
 - I1: viernes 14 de septiembre
 - I2: viernes 9 de noviembre
 - **Examen** (fecha por definir)
- **Tareas:**
 - T1: viernes 31 de agosto
 - T2: viernes 5 de octubre
 - T3: viernes 9 de noviembre

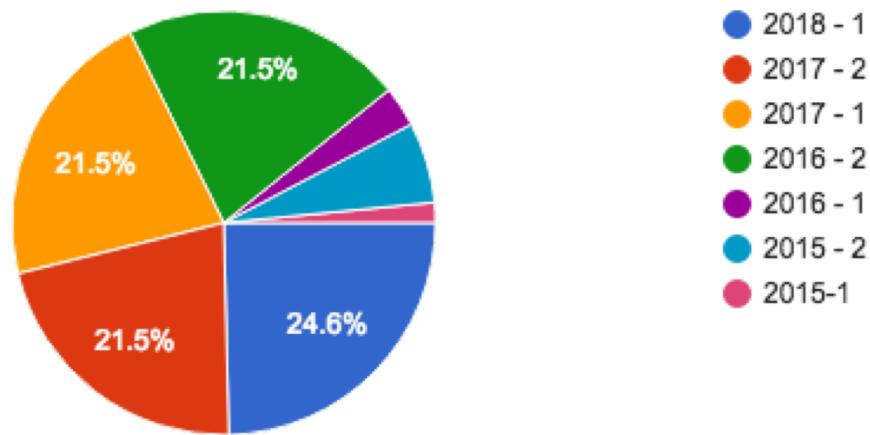
¿Qué major estás cursando?

65 responses

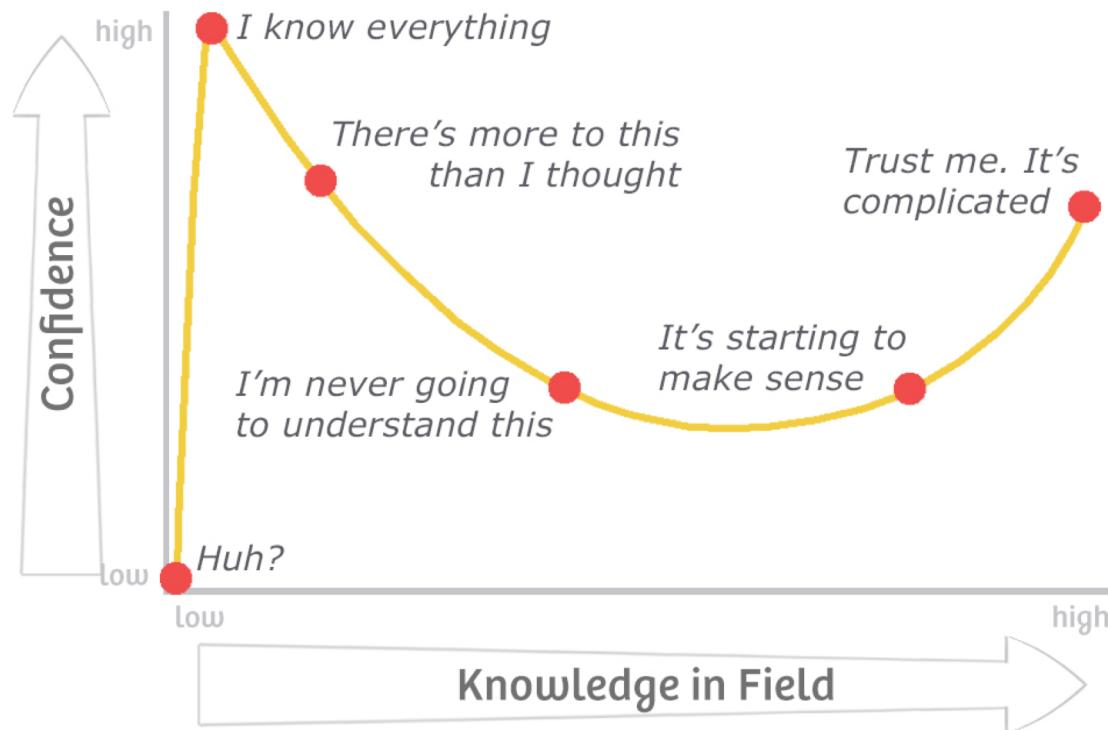


¿En qué semestre aprobaste Ingeniería de Software?

65 responses

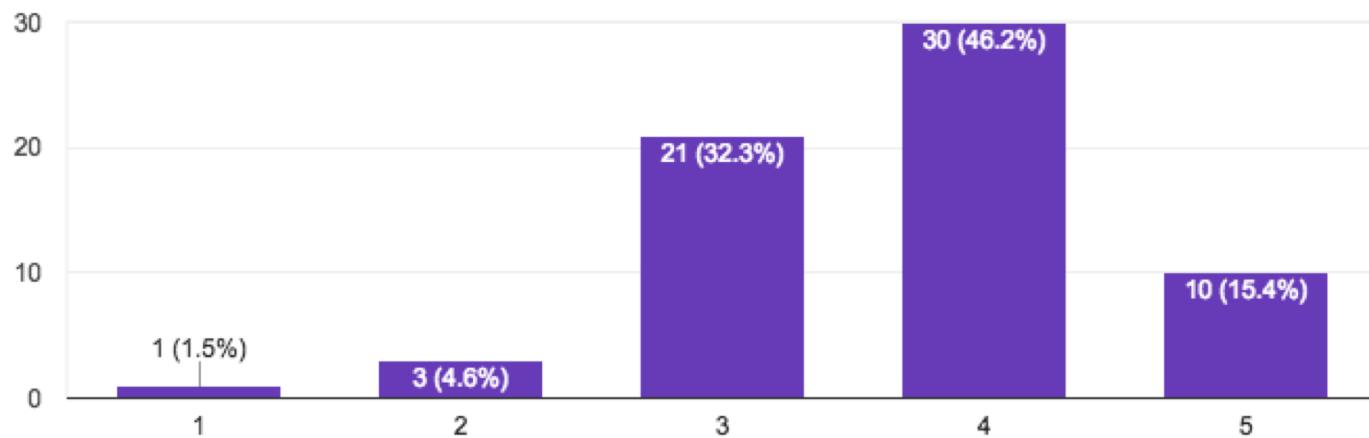


Efecto Dunning-Kruger



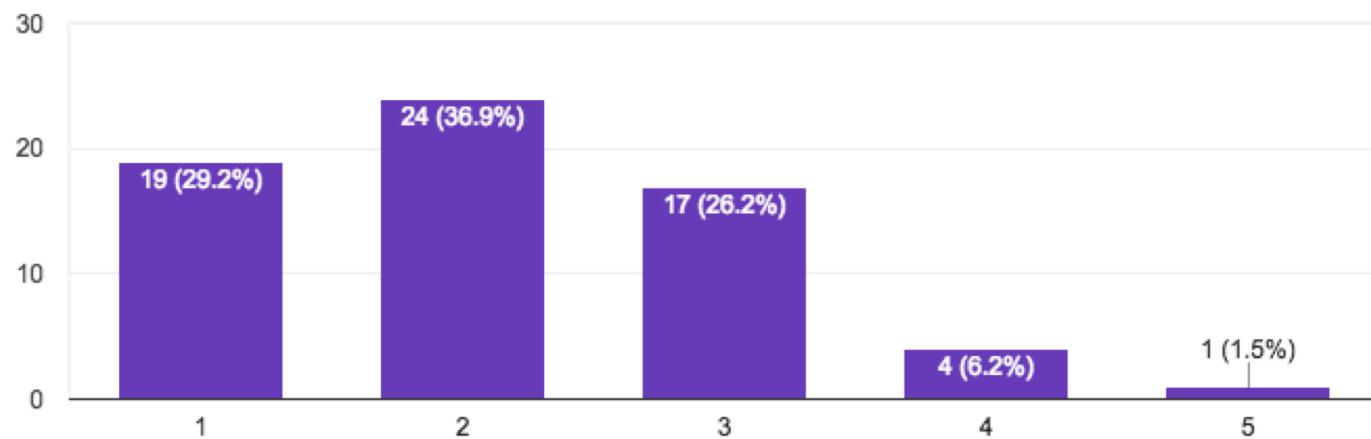
¿Cuál es tu nivel de experiencia con git?

65 responses



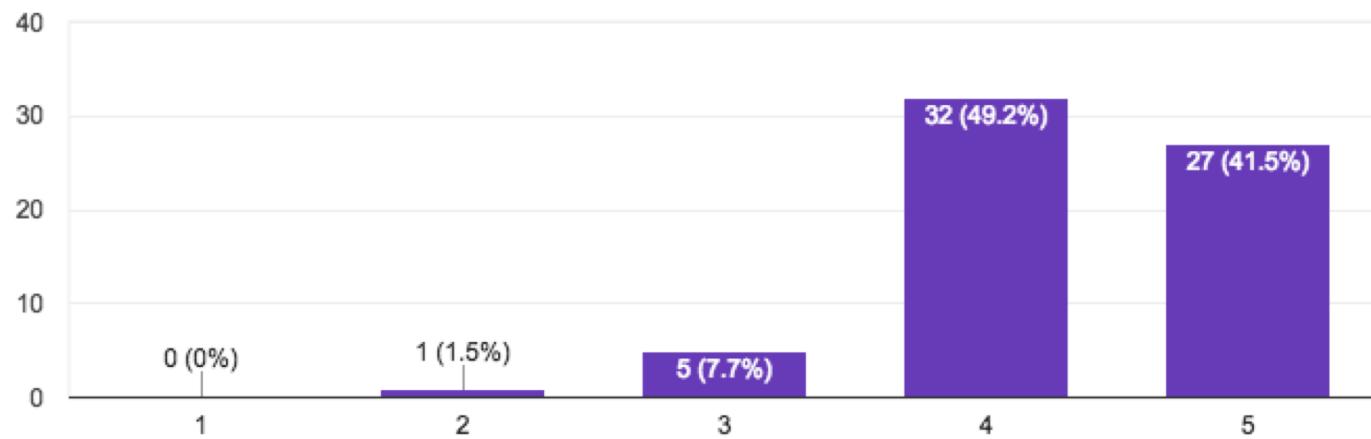
¿Cuál es tu nivel de experiencia con unit testing?

65 responses



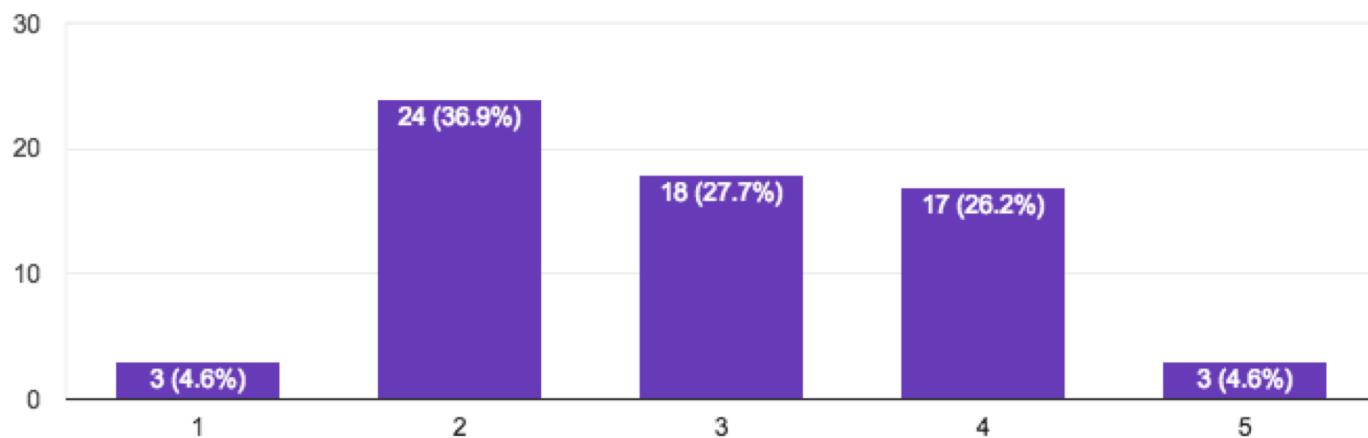
Python

65 responses



Ruby

65 responses



Diseño de software

¿Qué es el diseño de software?

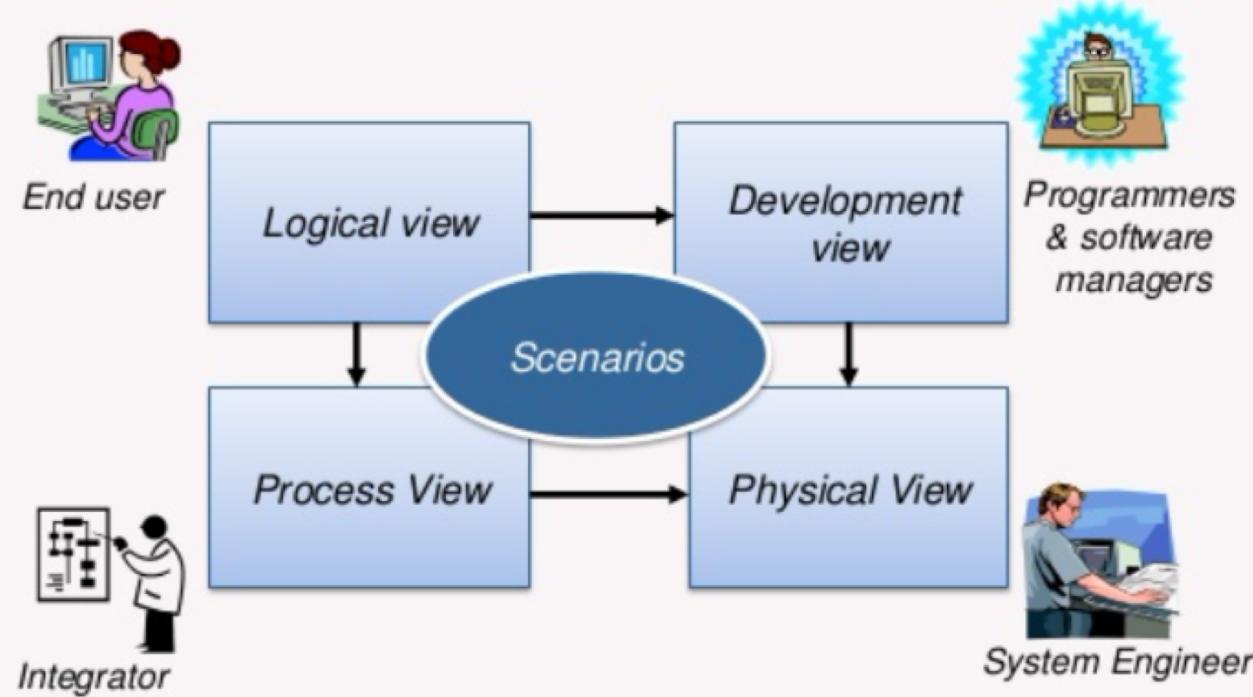
- Un concepto reciente (70 años)
- Traduce los requisitos a especificaciones técnicas
- Se divide en 4 áreas:
 - Diseño de componentes
 - Diseño de arquitectura
 - Diseño de clases/datos
 - Diseño de interfaces

Modelo 4 + 1

- Propuesto por Philippe Kruchten en 1995
- *Framework* para describir la arquitectura de un *software*, basado en el uso de múltiples vistas concurrentes.

Modelo 4 + 1

The 4+1 View model



- Describes software architecture using five concurrent views.

Modelo 4 + 1

- ¿Qué es una vista?

Es una representación del sistema completo, enfocada desde una perspectiva con consideraciones específicas.

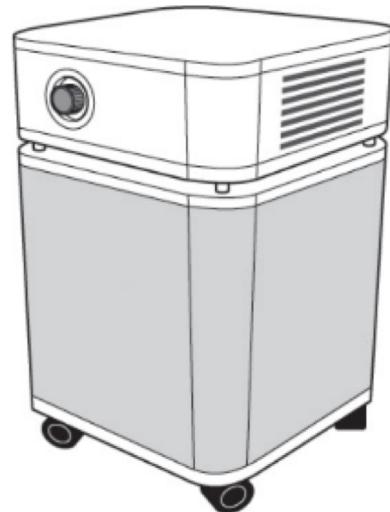
Unified Modeling Language 2.0

- UML: estándar para especificar un sistema
- 1º versión propuesta en 1997
- Ayuda a entender y explicar un sistema de manera consistente
- *The Elements of UML 2.0 Style*

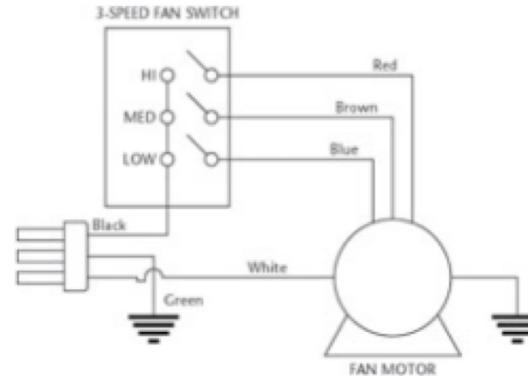
Unified Modeling Language 2.0

Lenguaje visual para describir artefactos de software

artefacto



lenguaje visual



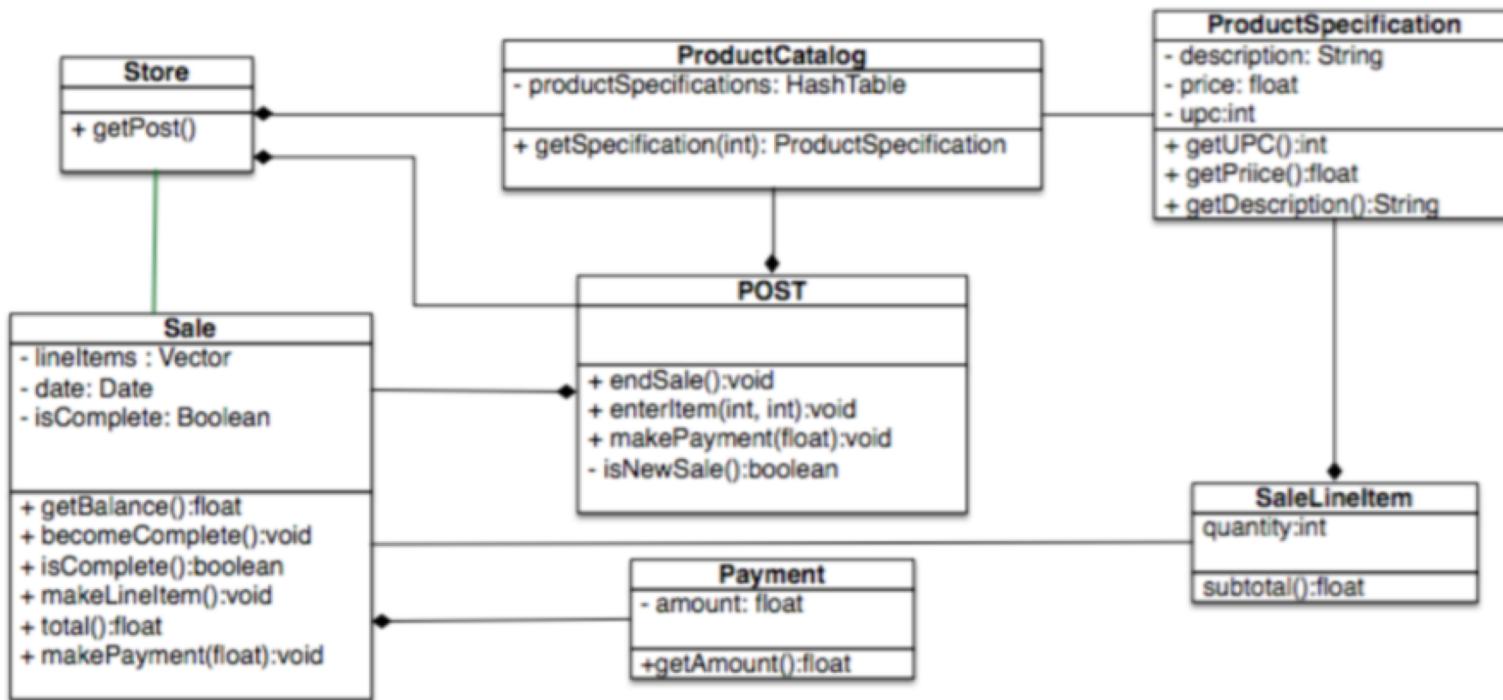
Unified Modeling Language 2.0

```
public class Payment {  
    private float amount;  
    public Payment (float cashTendered){this.amount = cashTendered;}  
    public float getAmount() { return amount; }  
}  
  
public class ProductCatalog{  
    private HashTable productSpecifications = new Hashtable();  
    public ProductCatalog(){  
        ProductSpecification ps = new ProductSpecification(100, 1, "product 1");  
        productSpecifications.put (new Integer(100), ps);  
        ps = new ProductSpecification(200, 1, "product 2");  
        productSpecifications.put (new Integer(200), ps);  
    }  
  
    public ProductSpecification getSpecification (int upc){  
        return (ProductSpecification)productSpecifications.get(new Integer(upc));  
    }  
}  
  
class POST {  
    private ProductCatalog productCatalog;  
    private Sale sale;  
    public POST(ProductCatalog catalog){productCatalog = catalog;}  
    public void endSale() {sale.becomeComplete();}  
    public void enterItem(int upc, int quantity){  
        if (isNewSale() ) sale = new Sale();  
        ProductSpecification spec = productCatalog.specification(upc);  
        sale.makeLineItem(spec, quantity);  
    }  
    public void makePayment(float cashTendered){  
        sale.makePayment(cashTendered);  
    }  
    private boolean isNewSale(){return (sale == null) ||(sale.isComplete() );}  
}  
  
public class ProductSpecification{  
    private int upc = 0;  
    private float price = 0;  
    private String description = "";  
    public ProductSpecification(int upc, float price, String description){  
        this.upc = upc;  
        this.price = price;  
        this.description = description;  
    }  
    public int getUPC {return upc; }  
    public float getPrice() {return price; }  
    public String getDescription() {return description; }  
}  
  
class Sale {  
    private Vector lineItems = new Vector();  
    private Date date = new Date();  
    private boolean isComplete = false;  
    private Payment payment;  
    public float getBalance () {return payment.getAmount() - total(); }  
    public void becomeComplete() {isComplete = true; }  
    public boolean isComplete() {return isComplete; }  
    public void makeLineItem(ProductSpecification spec, int quantity) {  
        lineItems.addElement(new SaleLineItem(spec, quantity));  
    }  
    public float total(){  
        float total = 0;  
        Enumeration e = lineItems.elements();  
        while(e.hasMoreElements() ){  
            total += ( (SaleItem) e.nextElement() ).subtotal();  
        }  
        return total;  
    }  
    public void makePayment (float cashTendered) {  
        payment = new Payment (cashTendered);  
    }  
}  
  
class SaleLineItem{  
    private int quantity;  
    private ProductSpecification productSpec;  
    public SaleLineItem (ProductSpecification spec, int quantity) {  
        this.productSpec = spec;  
        this.quantity = quantity;  
    }  
    public float subtotal() { return quantity* productSpec.getPrice(); }  
}  
  
class Store {  
    private ProductCatalog productCatalog = new ProductCatalog();  
    private POST post = new POST (productCatalog);  
    public POST getPOST() {return post; }  
}
```

Artefacto

Unified Modeling Language 2.0

Representación Visual del Artefacto



Vista lógica

- Interesados: Usuarios finales
- Consideraciones: Requisitos funcionales
 - ¿Qué es lo que el sistema debería proveer a sus usuarios?
- Diagramas:
 - Clases
 - Comunicación
 - Secuencia
 - Estado

Diagrama de clases

Modela las clases, sus relaciones, operaciones y atributos.

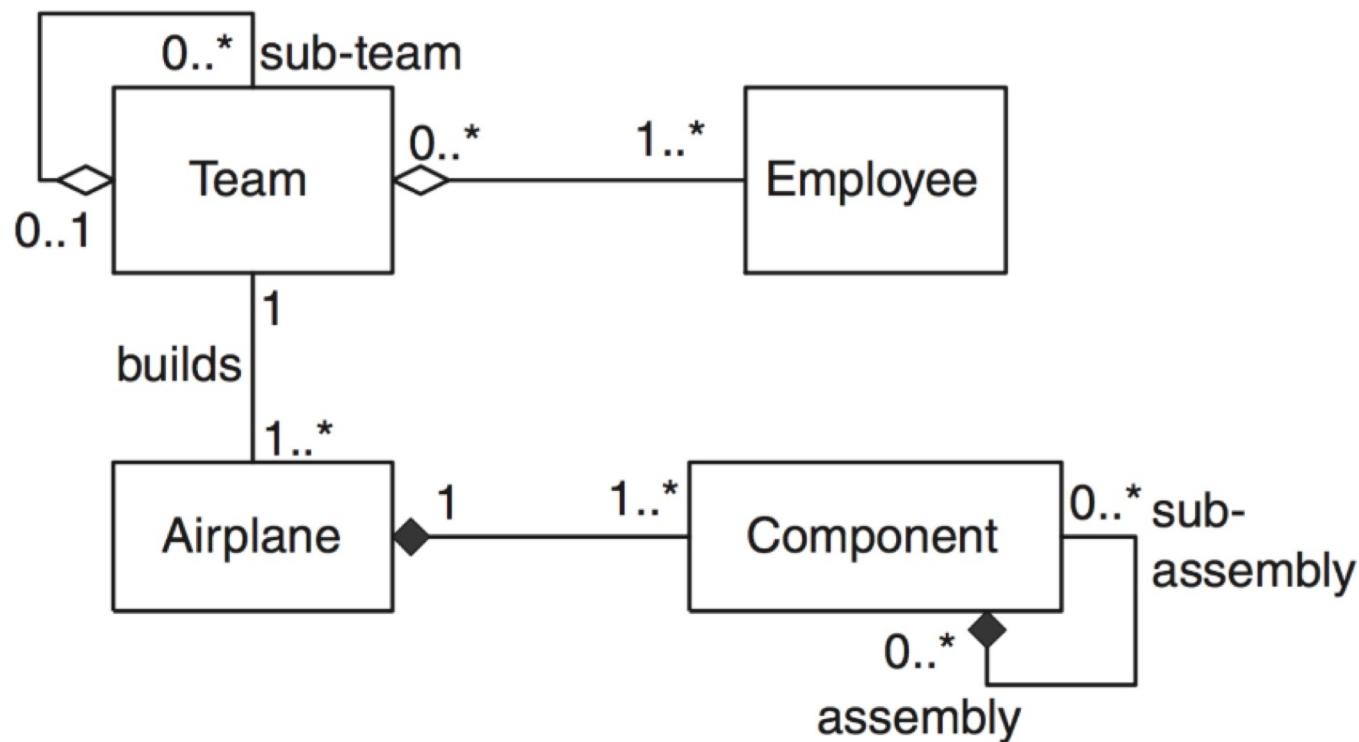


Diagrama de comunicación

Modela la comunicación y dirección de los mensajes entre clases.

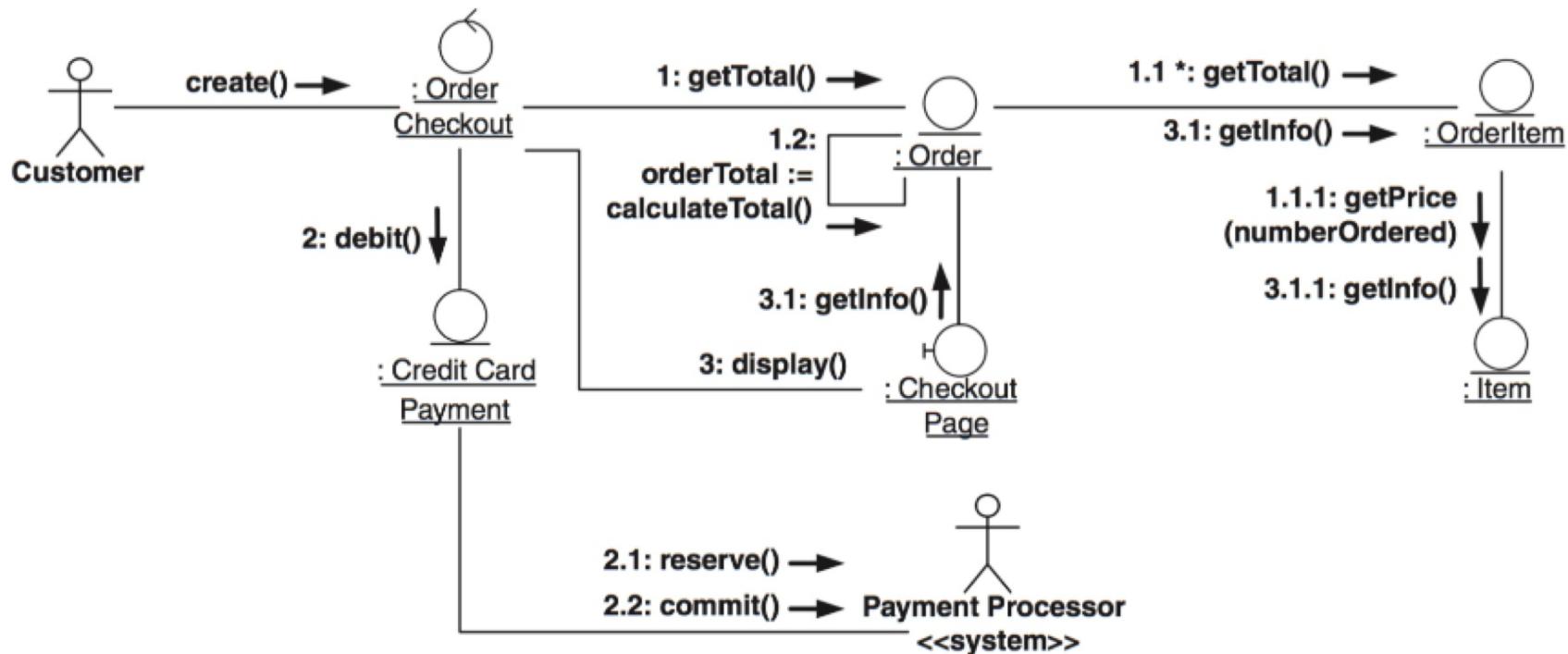


Figure 33. An instance-level UML communication diagram.

Diagrama de secuencia

Modela la interacción y orden en que las entidades se relacionan.

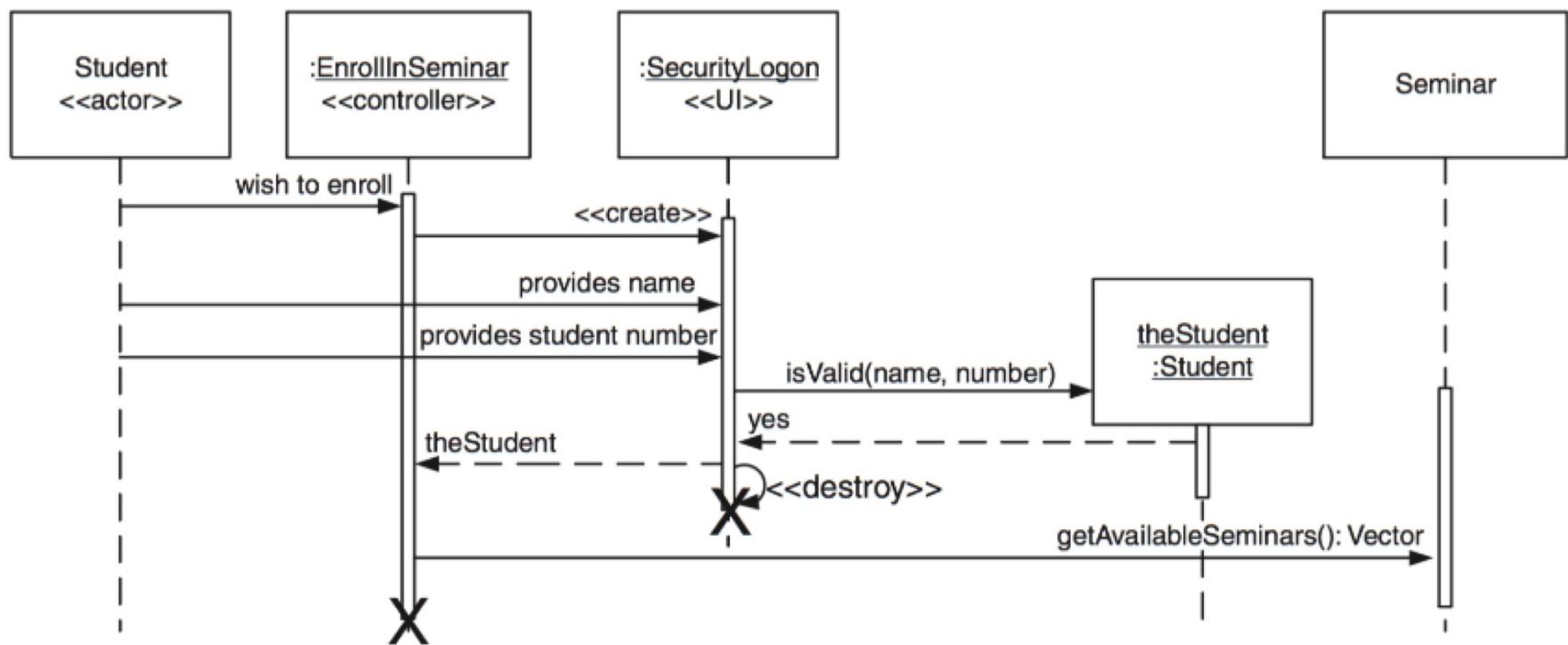
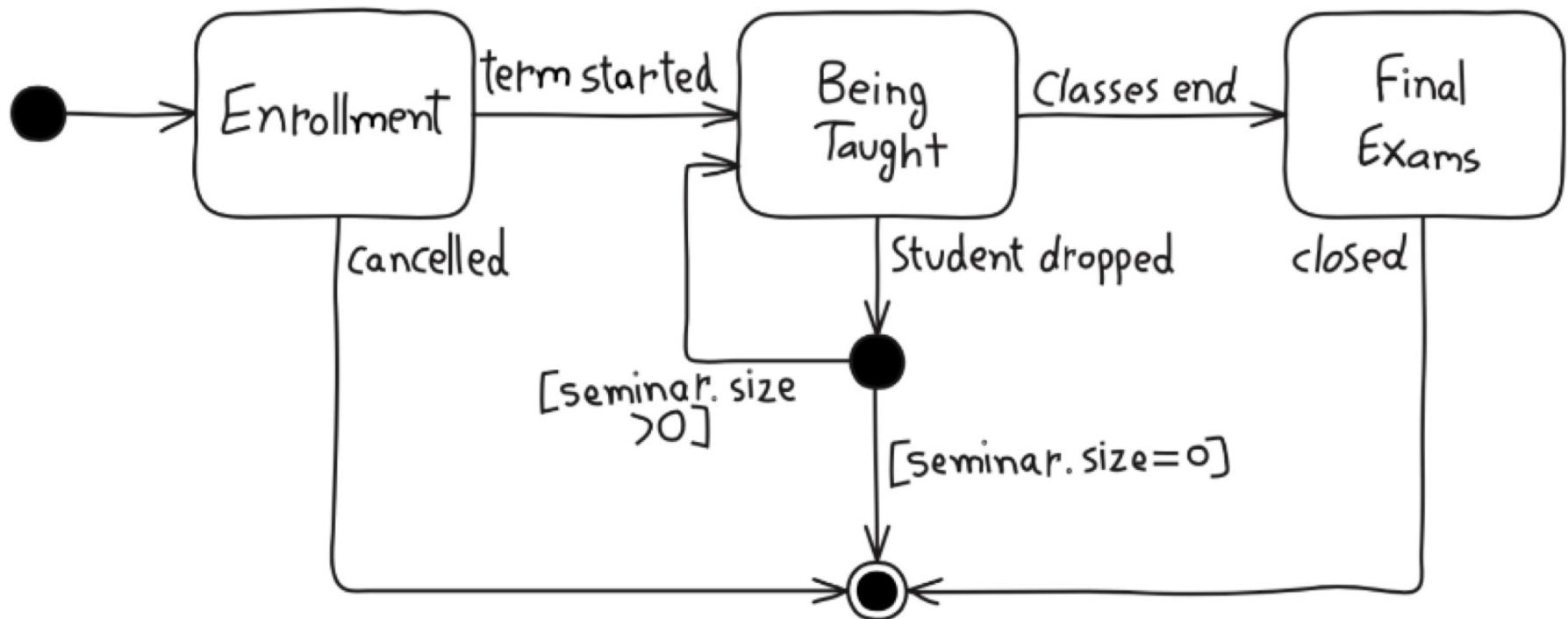


Diagrama de estado

Modela el comportamiento de las entidades dado distintos estados, basado en sus respuestas a eventos.

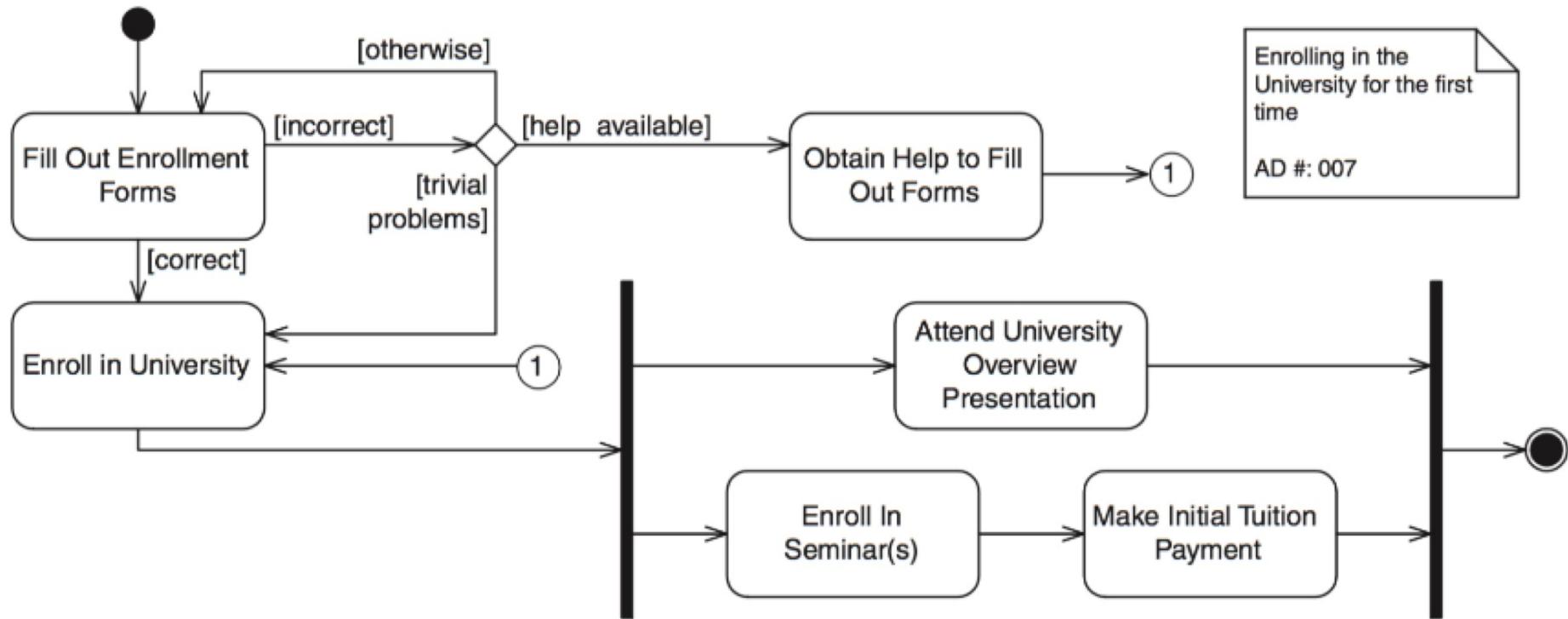


Vista de procesos

- Interesados: Analistas
- Consideraciones: Requisitos no funcionales (conurrencia, rendimiento, escalabilidad)
 - ¿Qué procesos y reglas tiene el sistema, y cómo interactúan los componentes?
- Diagramas:
 - Actividad

Diagrama de actividad (o flujo)

Modela procesos en base a reglas.



Vista de implementación

- Interesados: Desarrolladores y líderes de proyectos
- Consideraciones: Organización del *software*
 - ¿Cómo se organiza el *software* del sistema?
- Diagramas:
 - Componentes
 - Paquetes

Diagrama de componentes

Modela la dependencia entre los componentes del *software*.

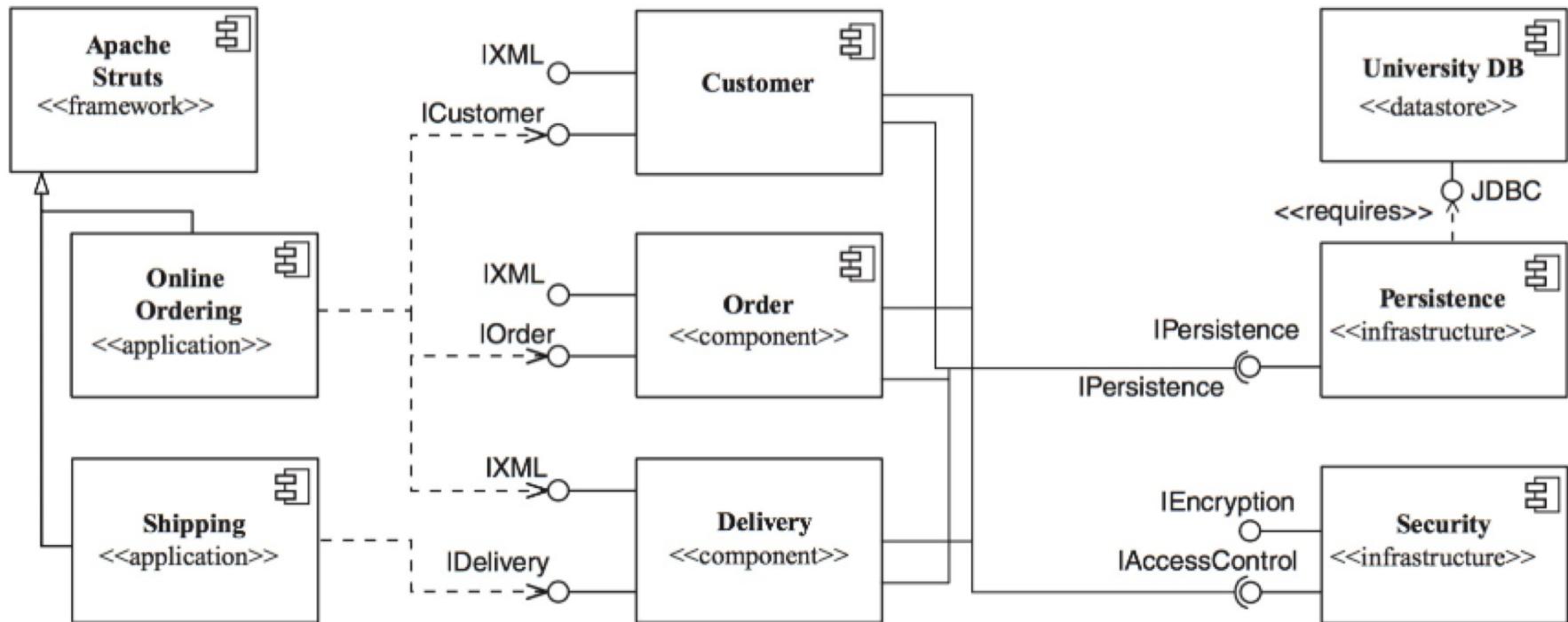
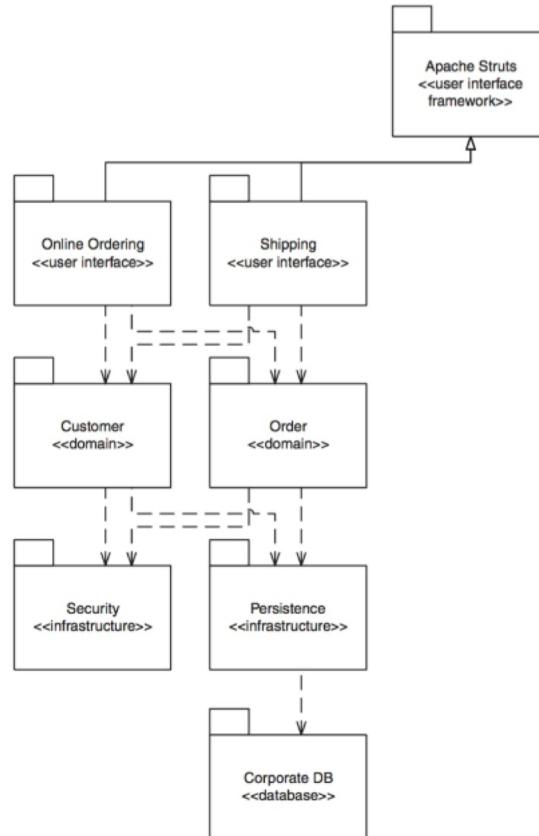


Diagrama de paquetes

Modela la dependencia entre los paquetes del *software*. Un paquete se puede ver como una carpeta o librería del proyecto.

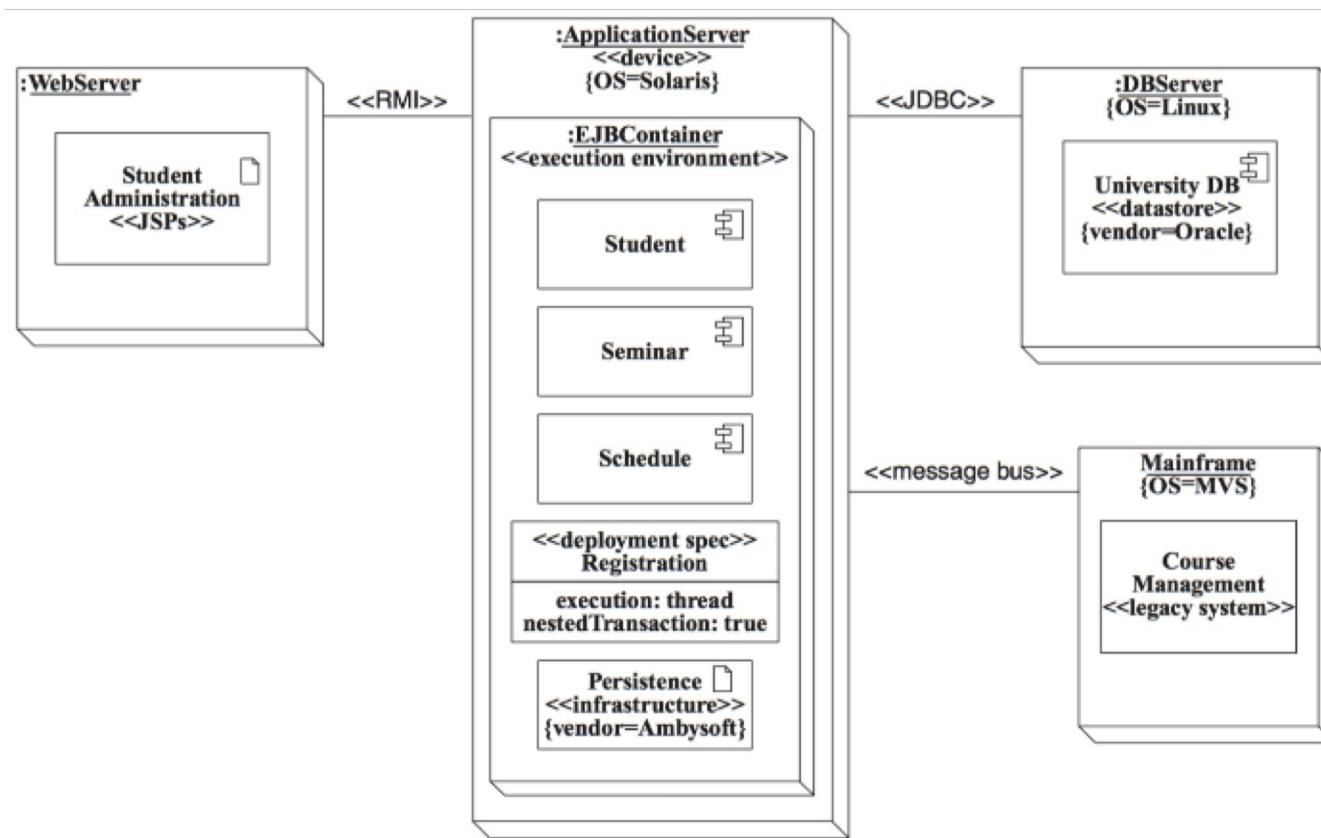


Vista física

- Interesados: Arquitectos de sistemas
- Consideraciones: Requisitos no funcionales
 - ¿Cómo es el ambiente de ejecución del sistema?
- Diagramas:
 - Despliegue

Diagrama de despliegue

Modela la configuración en tiempo de ejecución del *hardware*, y el *software* que se ejecuta.

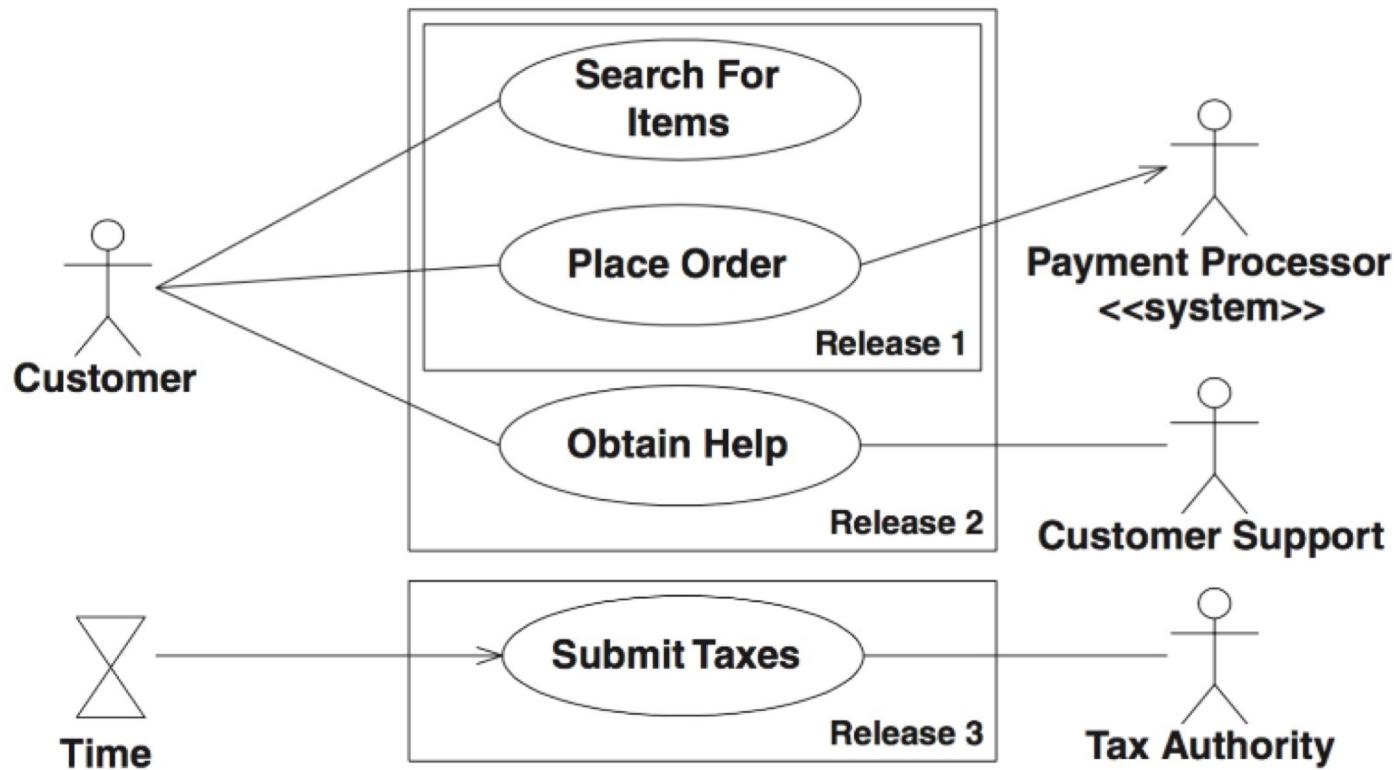


Vista de escenarios

- Interesados: Todos
- Consideraciones: Consistencia, validez
 - ¿Qué es lo que el sistema debería hacer?
- Diagramas:
 - Casos de uso

Diagrama de casos de uso

Modela la interacción entre los actores y las funcionalidades de un sistema.



Actividad 1



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

Clase 1

Diagramas del diseño

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

17 de agosto de 2018