



**UTPL**  
*La Universidad Católica de Loja*

**Modalidad Abierta y a Distancia**



# Modelado de Sistemas

## Guía didáctica

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas



**Departamento de Ciencias de la Computación y  
Electrónica**

**Sección departamental de Ingeniería de Software y  
Gestión de Tecnologías de la Información**

---

## Modelado de Sistemas

***Guía didáctica***

**Autor:**

Daniel Alejandro Guamán Coronel



TURI \_ 2013

**Asesoría virtual**  
[www.utpl.edu.ec](http://www.utpl.edu.ec)

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

Índice

Primer bimestre

Segundo bimestre

Solucionario

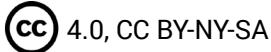
Referencias bibliográficas

## Modelado de Sistemas

Guía didáctica

Daniel Alejandro Guamán Coronel

Universidad Técnica Particular de Loja



## Diagramación y diseño digital:

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

[www.ediloja.com.ec](http://www.ediloja.com.ec)

[edilojainfo@ediloja.com.ec](mailto:edilojainfo@ediloja.com.ec)

Loja-Ecuador

ISBN digital - 978-9942-25-686-7



La versión digital ha sido acreditada bajo la licencia Creative Commons 4.0, CC BY-NY-SA: Reconocimiento-No comercial-Compartir igual; la cual permite: copiar, distribuir y comunicar públicamente la obra, mientras se reconozca la autoría original, no se utilice con fines comerciales y se permiten obras derivadas, siempre que mantenga la misma licencia al ser divulgada. <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>

23 de abril, 2020

Índice

# Índice

<b>1. Datos de información.....</b>	<b>8</b>
1.1. Presentación. Orientaciones de la asignatura .....	8
1.2. Competencias genéricas de la UTPL.....	8
1.3. Competencias específicas de la carrera .....	8
1.4. Problemática que aborda la asignatura en el marco del proyecto.....	9
<b>2. Metodología de aprendizaje.....</b>	<b>9</b>
<b>3. Orientaciones didácticas por resultados de aprendizaje .....</b>	<b>11</b>
<b>Primer bimestre.....</b>	<b>11</b>
Resultado de aprendizaje 1 .....	11
Contenidos, recursos y actividades de aprendizaje.....	11
<b>Semana 1 .....</b>	<b>11</b>
<b>Unidad 1. Introducción a modelado.....</b>	<b>12</b>
1.1. ¿Por qué modelar? .....	12
1.2. La importancia de modelar.....	14
Actividades de aprendizaje recomendadas .....	17
Resultado de aprendizaje 2 .....	18
Contenidos, recursos y actividades de aprendizaje.....	18
<b>Semana 2 .....</b>	<b>18</b>
1.3. Principios de modelado .....	18
1.4. Modelado orientado a objetos .....	20
Actividades de aprendizaje recomendadas .....	21
Autoevaluación 1 .....	22
Resultado de aprendizaje 3 .....	28
Contenidos, recursos y actividades de aprendizaje.....	28

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

<b>Semana 3 .....</b>	<b>28</b>
<b>Unidad 2. UML y conceptos generales .....</b>	<b>28</b>
2.1. Visión general de UML .....	29
2.2. Modelo conceptual de UML .....	30
2.3. Arquitectura.....	31
Actividades de aprendizaje recomendadas .....	32
<b>Semana 4 .....</b>	<b>33</b>
2.4. Ciclo de vida del desarrollo de software.....	33
2.5. Vistas UML .....	34
Actividades de aprendizaje recomendadas .....	36
Autoevaluación 2 .....	37
Resultado de aprendizaje 4 .....	41
Contenidos, recursos y actividades de aprendizaje.....	41
<b>Semana 5 .....</b>	<b>41</b>
2.6. Modelado estructural básico .....	41
<b>Unidad 3. Casos de uso y diagramas de caso de uso.....</b>	<b>45</b>
3.1. Diagrama .....	45
Actividades de aprendizaje recomendadas .....	46
Autoevaluación 3 .....	47
<b>Semana 6 .....</b>	<b>51</b>
3.2. Casos de uso.....	51
3.3. Diagramas de caso de uso .....	53
Actividades de aprendizaje recomendadas .....	55
Resultado de aprendizaje 6 .....	57
Contenidos, recursos y actividades de aprendizaje.....	57
<b>Semana 7 .....</b>	<b>57</b>
<b>Unidad 4. Diagramas de Interacción y diagrama de actividades ...</b>	<b>57</b>
4.1. Diagramas de interacción o de secuencia .....	58
4.2. Diagrama de actividades.....	66

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

Índice	
Primer bimestre	
Segundo bimestre	
Solucionario	
Referencias bibliográficas	
<b>Actividades de aprendizaje recomendadas .....</b>	69
<b>Autoevaluación 4 .....</b>	70
<b>Actividades finales del bimestre.....</b>	74
<b>Semana 8 .....</b>	74
 <b>Segundo bimestre .....</b>	 77
<b>Resultado de aprendizaje 3 .....</b>	77
<b>Contenidos, recursos y actividades de aprendizaje.....</b>	77
<b>Semana 9 .....</b>	77
<b>Unidad 5. Relaciones y mecanismos comunes.....</b>	78
<b>5.1. Relaciones .....</b>	78
<b>5.2. Mecanismos comunes .....</b>	84
<b>Actividades de aprendizaje recomendadas .....</b>	89
<b>Resultado de aprendizaje 5 .....</b>	92
<b>Contenidos, recursos y actividades de aprendizaje.....</b>	92
<b>Semana 10 .....</b>	92
<b>Unidad 6. Diagrama de clases.....</b>	92
<b>6.1. Diagrama de clase .....</b>	92
<b>6.2. Relaciones entre clases.....</b>	95
<b>Actividades de aprendizaje recomendada .....</b>	99
<b>Autoevaluación 5 .....</b>	100
<b>Resultado de aprendizaje de 4 .....</b>	108
<b>Contenidos, recursos y actividades de aprendizaje.....</b>	108
<b>Semana 11 .....</b>	108
<b>6.3. Consideraciones al momento de diseñar diagrama de clases.....</b>	108
<b>Actividades de aprendizaje recomendadas .....</b>	113
<b>Autoevaluación 6 .....</b>	114
<b>Resultado de aprendizaje 6.....</b>	122
<b>Contenidos, recursos y actividades de aprendizaje.....</b>	122

<b>Semana 12 .....</b>	<b>122</b>
<b>    Unidad 7. Interfaces, Paquetes, Componentes.....</b>	<b>122</b>
7.1. Interfaces .....	122
7.2. Paquetes.....	126
Actividades de aprendizaje recomendadas .....	131
Autoevaluación 7 .....	132
<b>Semana 13 .....</b>	<b>136</b>
7.3. Componentes .....	136
Actividades de aprendizaje recomendadas .....	140
Resultado de aprendizaje 7 .....	140
Contenidos, recursos y actividades de aprendizaje.....	140
<b>Semana 14 .....</b>	<b>140</b>
<b>    Unidad 8. Diagrama de despliegue y modelado arquitectónico ....</b>	<b>141</b>
8.1. Despliegue y diagrama de despliegue .....	142
8.2. Modelado arquitectónico .....	148
Actividades de aprendizaje recomendadas .....	151
Autoevaluación 8 .....	152
Resultado de aprendizaje 2 .....	156
Contenidos, recursos y actividades de aprendizaje.....	156
<b>Semana 15 .....</b>	<b>156</b>
Actividades de aprendizaje recomendadas .....	168
Actividades finales del bimestre .....	169
<b>Semana 16 .....</b>	<b>169</b>
<b>    4. Solucionario .....</b>	<b>171</b>
<b>    5. Referencias bibliográficas .....</b>	<b>181</b>

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



## 1. Datos de información

### 1.1. Presentación. Orientaciones de la asignatura



### 1.2. Competencias genéricas de la UTPL

- Organización y planificación del tiempo.

### 1.3. Competencias específicas de la carrera

- Diseñar aplicaciones de software que mediante técnicas avanzadas de modelado permitan solucionar los requerimientos del cliente, utilizando estándares de la industria.

- Modelar procesos de negocio utilizando técnicas y marcos de referencia para identificar problemas, oportunidades de mejora y proponer alternativas que permitan dar soporte a la estrategia del negocio.

#### 1.4. Problemática que aborda la asignatura en el marco del proyecto

- Capacidad de análisis, entendimiento y comprensión de un problema dentro de un contexto determinado.
- Necesidad de documentar una solución software haciendo uso de marcos de referencia, estándares como ISO e IEEE, nomenclatura, elementos y estereotipos UML.
- Proponer soluciones de software (de escritorio, móviles o web) haciendo uso de diagramas, vistas, puntos de vista para exponerlos a los clientes o involucrados.
- Proponer soluciones arquitectónicas de software candidatas ante un problema o necesidad.



---

## 2. Metodología de aprendizaje

---

En la presente asignatura se utilizará la metodología de aprendizaje basado en problemas. Bajo este contexto, se requiere que el estudiante haga uso de los conocimientos adquiridos hasta el

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

momento, como parte de su formación académica y/o formación profesional dentro de su lugar de trabajo o como parte de las actividades de auto estudio diarias.

El modelado se utiliza para representar de forma documentada, estandarizada (textual y visual) y estilizada, una solución o posibles soluciones a problemas que, haciendo uso de hardware y/o software, pueden ser implementadas a través de diversas tecnologías y lenguajes de programación. Estas soluciones, desde el punto de vista de su estructura y comportamiento en tiempo de ejecución, deben permitir el análisis, evaluación, mantenimiento y evolución.

En el contexto de la vida real, existen problemas que, para su automatización mediante la creación o implementación de software, deben hacer uso de frameworks (marcos de referencia), modelos y metodologías de desarrollo, donde las Ciencias de la Computación y los profesionales formados en esta área pueden apoyarse en actividades como análisis, identificación de requisitos e integración de posibles soluciones a dichos problemas.

En la presente guía didáctica se abordan los aspectos básicos relacionados al modelado, para que conozca y aplique de forma correcta los documentos, elementos y nomenclatura que se consideran útiles en el diseño e implementación de software. Tanto la documentación, diseño e implementación pueden conducirle a investigar y aplicar conceptos avanzados, utilizando para ello bibliografía básica y complementaria impresa o en línea.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



---

### 3. Orientaciones didácticas por resultados de aprendizaje

---



Primer bimestre

**Resultado de aprendizaje 1** | Determina los requisitos de datos para una aplicación.

#### Contenidos, recursos y actividades de aprendizaje

---



Semana 1

---

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



## Unidad 1. Introducción a modelado

Las unidades 1 y 2 hacen referencia a los conceptos básicos para entender y comprender el tema de modelado y respondernos a preguntas del ¿por qué? y ¿para qué? Para captar las ideas principales que ayudarán en la representación textual y gráfica (usando diagramas) de la(s) solución(es) a un problema que se resuelve con software, es necesario leer y comprender. Por ello, le invito a iniciar el estudio de los temas correspondientes al primer bimestre, revisando los fundamentos necesarios de Modelado. Al finalizar el mismo usted debería estar en capacidad de responder a la pregunta: ¿por qué modelamos?

### 1.1. ¿Por qué modelar?

Una concepción errónea en una persona con conocimientos en desarrollo de software, es que cuando un cliente le solicita la automatización de un proceso manual haciendo uso de software para dar solución a un problema, considerado mediano o grande, el ingeniero de software muchas de las veces no toma en cuenta los modelos de desarrollo, las metodologías de desarrollo (tradicionales o no tradicionales) y las fases del ciclo de vida de desarrollo de software que son parte de las metodologías (análisis, diseño, codificación, pruebas, implementación, mantenimiento). Se enfoca netamente en su codificación, utilizando para ello diversos lenguajes de programación y tecnologías que permiten la construcción e implementación de la solución. Análogamente, como

si un arquitecto quisiera construir una casa, pero sin planos, sería imposible, ¿verdad?

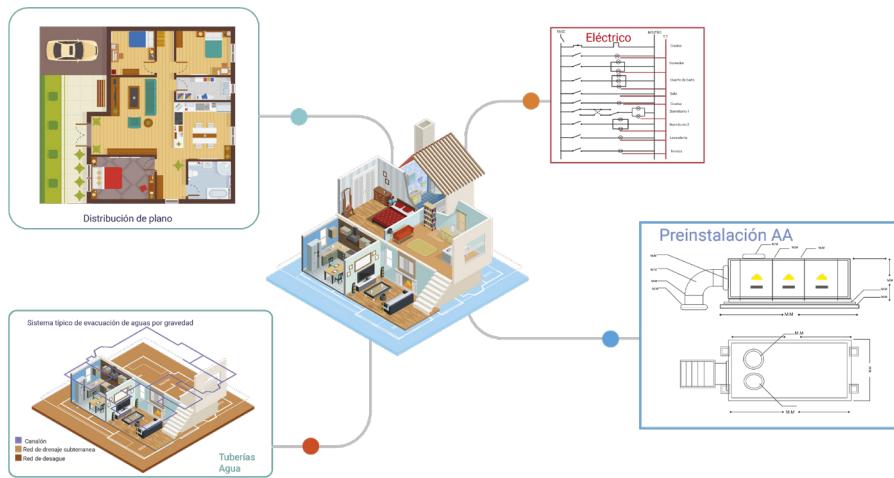
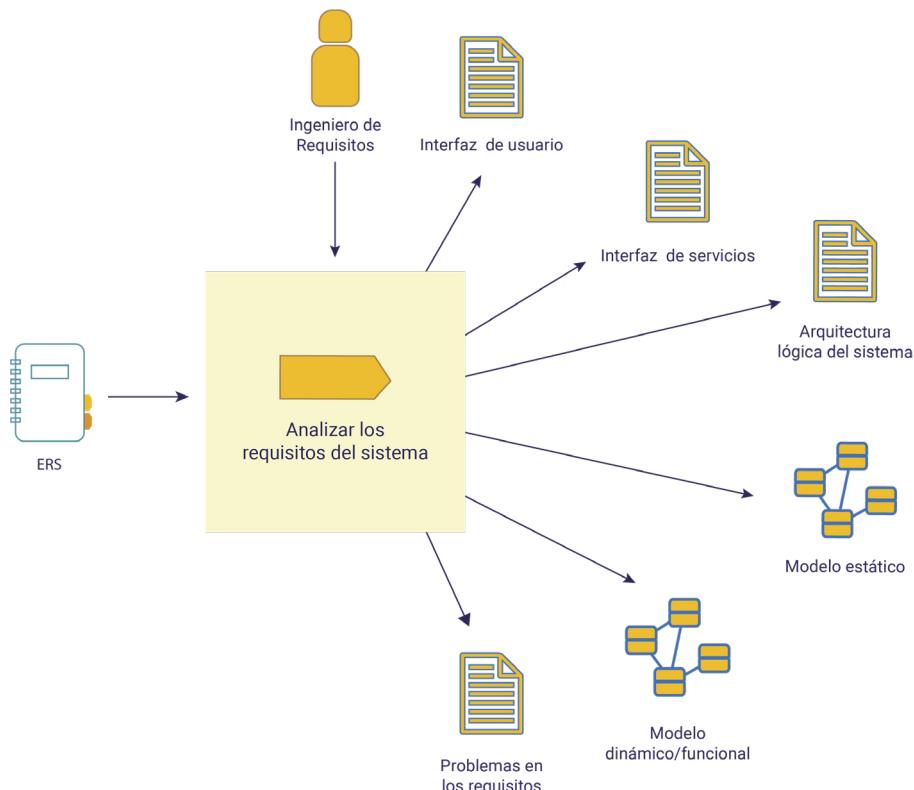


Figura 1. Analogía modelar vs diseños o planos de una casa.

Pero, ¿cuáles son los requisitos identificados como parte de la fase de análisis?, ¿realizó alguna observación directa, entrevista, consulta a los usuarios de la organización para redactar las necesidades que posteriormente se transforman en requisitos?, ¿qué análisis permitió elaborar o proponer un diseño que se vea reflejado en su codificación e implementación?, ¿existe la documentación o entregables requeridos como resultados de la fase de análisis, diseño, codificación e implementación que permita posteriormente el mantenimiento o evolución del software?, ¿qué modelos o representaciones de alto nivel se han levantado para codificar e implementar el producto software? Todas estas son preguntas que como Ingenieros deberíamos contestar al momento de proponer soluciones software. Es por ello que, antes de una implementación, se debe analizar, documentar, diseñar, refining modelos o diagramas que permitan tener una visión clara del problema, para proponer una o varias alternativas de solución antes de su implementación en sí.

## 1.2. La importancia de modelar

Los modelos son utilizados con diferentes propósitos. Una de ellos en el campo de Ciencias de la Computación para representar de forma textual o gráfica una propuesta de solución aceptable, acorde a un contexto o dominio de aplicación. Para lograr ello, los ingenieros de software a partir de la ERS (Especificación de requisitos de software) recopilan datos, los organizan en información y lo formalizan en conocimiento para representar de forma general lo que se requiere del sistema.



*Figura 2. Diagrama usado para representar la visión general del sistema a partir del ERS*

Usted, como Ingeniero en Tecnologías de la Información, debe ser capaz de analizar, diseñar y construir proyectos y productos software utilizando: experiencia de implementaciones anteriores, documentación base (estándares ISO e IEEE), marcos de referencia (frameworks) y componentes/elementos estandarizados, que a través de su integración permitan que la solución propuesta se ajuste a las necesidades de los clientes donde se tomen en cuenta aspectos como recursos, tiempo y costos.

Como se menciona en el texto básico y bibliografía complementaria, los objetivos que se alcanzan frente al desarrollo de software mediante los modelos son:

- a. Capturar y especificar con precisión los requisitos y el conocimiento dentro de un contexto, para que todos los involucrados (stakeholders) puedan entenderlos y aceptarlos.
- b. Pensar en el diseño (lógico y físico) del software aplicando un proceso o metodología de desarrollo.
- c. Ayudar a visualizar el software tal como es o como se espera que sea (interfaz de usuario).
- d. Permitir especificar la estructura o el comportamiento de un sistema (modelo estático, modelo dinámico/funcional).
- e. Brindar una plantilla o documento de análisis del sistema que guie en la construcción de un sistema.
- f. Documentar las decisiones que se han tomado.
- g. Generar productos de trabajo utilizables.

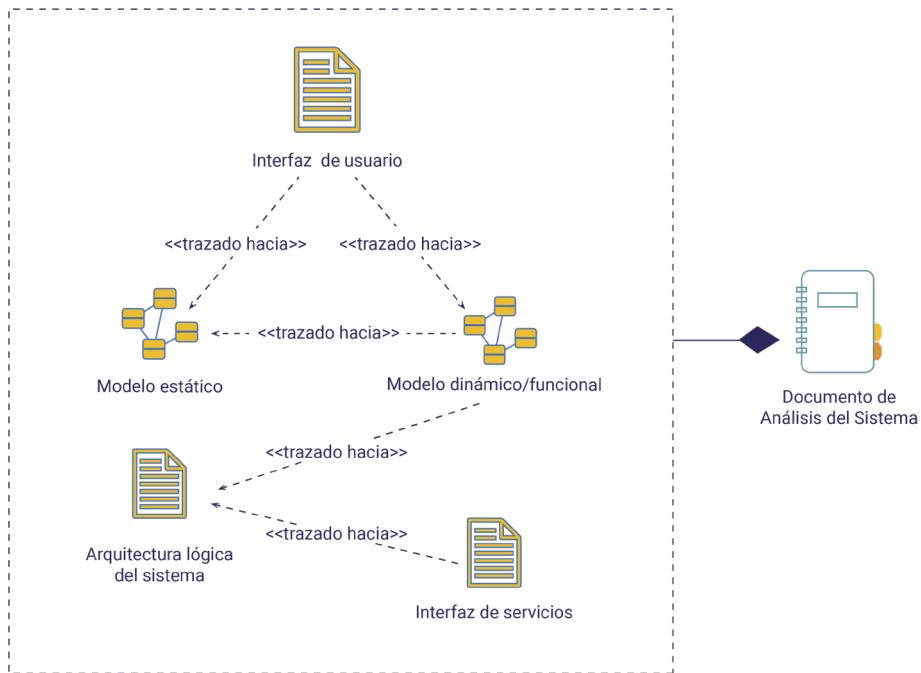


Figura 3. Componentes del documento de análisis del sistema

Una vez que ha estudiado la parte conceptual en esta semana 1, lo que debe tener claro es que para proponer una solución se debe entender bien la problemática o necesidad que por lo general lo dan los clientes o usuarios que requieren la automatización de un servicio. En esta fase Ud. está adoptando el rol de analista, por lo que debe hacer uso de técnicas para captar lo principal que le solicita un cliente.



## Actividades de aprendizaje recomendadas

### Lecturas recomendadas:

- (Unidad 1.2) [En el recurso educativo abierto \(Ingeniería de Software\)](#), por favor revise el Tema 1. Introducción a la Ingeniería del Software, donde identificará la evolución del término Ingeniería de Software, entender como se analiza la resolución de un problema, las metodologías y vistas que son parte del modelo 4+1 view. (Páginas, 19-22, 36-42, 44, 57, 59-67, 137-146)
- (Unidad 1.2) Tome el texto básico y revise por favor en el capítulo 1 los temas denominados ***¿Por qué modelamos? y la importancia de modelar.***
- Complemente la revisión del texto básico con la siguiente bibliografía complementaria, donde de forma breve se expone la importancia de modelado.

**Ejercicio 1.** Con base en la revisión de la bibliografía y recursos educativos abiertos sugeridos para su estudio y los que usted puede adicionar, por favor responda las seis preguntas que se proponen respecto a modelado:

1. **¿Por qué y para qué modelar?**
2. **¿Por qué es importante el tema de modelado?, ¿qué problemas resuelve?**
3. **¿Se puede usar el modelado solamente cuando se requiere construir un producto software? Argumente su respuesta.**

4. ¿Qué se puede modelar?. Proponga sus propios ejemplos (al menos 4).
5. En temas de software, ¿qué tipos de sistemas (pequeños, medianos, grandes) se podrían modelar? ¿Por qué?
6. Para modelar, ¿se puede utilizar una representación formal o informal?, ¿cuándo utilizaría cada uno de ellos y por qué?

### Resultado de aprendizaje 2

Identifica las partes esenciales de un sistema software que se podrían representar utilizando los diferentes tipos de modelado.

## Contenidos, recursos y actividades de aprendizaje



### Semana 2

Una vez que ha dado respuesta a las preguntas propuestas en el ejercicio 1 y conoce la importancia y objetivos de modelar, es necesario que estudie y analice los cuatro principios básicos de modelado que se proponen el texto básico.

#### 1.3. Principios de modelado

Cada principio de modelado tiene un objetivo que cumplir, el cual depende del contexto o ámbito de aplicación, del recurso humano

(a quien se expondrá dicho modelo como propuesta de solución) y de la formalidad y nivel de precisión que se requiera. Siguiendo los principios de modelado y tomando en cuenta la arquitectura MOF (Meta Modeling and Meta-Object Facility), se puede realizar una aproximación de la solución a un problema dentro de un contexto, de forma individual o integral, incluyendo aspectos de software o hardware.

Por ejemplo, cuando realicemos el modelado de clases siempre partiremos desde el nivel más bajo M0 Layer, el cual se relaciona con un problema en específico. Entendido el problema, podemos empezar a realizar el modelado de usuario haciendo uso de UML. En este caso representado por M1, donde en su forma normal, por ejemplo, para la representación de Clases se utilizan abstracciones como “Video”, para la representación de atributos de la clase se puede tener “title”, y como tipos de dato asociado a los atributos por ejemplo se puede definir “String”.

Sin embargo, si deseamos conocer el metamodelo del modelo podemos observar que una clase como tal se descompone o tiene su metamodelo o partes más internas (PrimitiveDataType, Classifier, Association, Attribute, Parameter, Operation) que el usuario no las visualiza pero que nosotros como ingenieros debemos conocerla. Para el presente curso usaremos los niveles M0 y M1 y con el uso de diagramas podremos representar o diseñar propuestas de solución de software comprensibles.

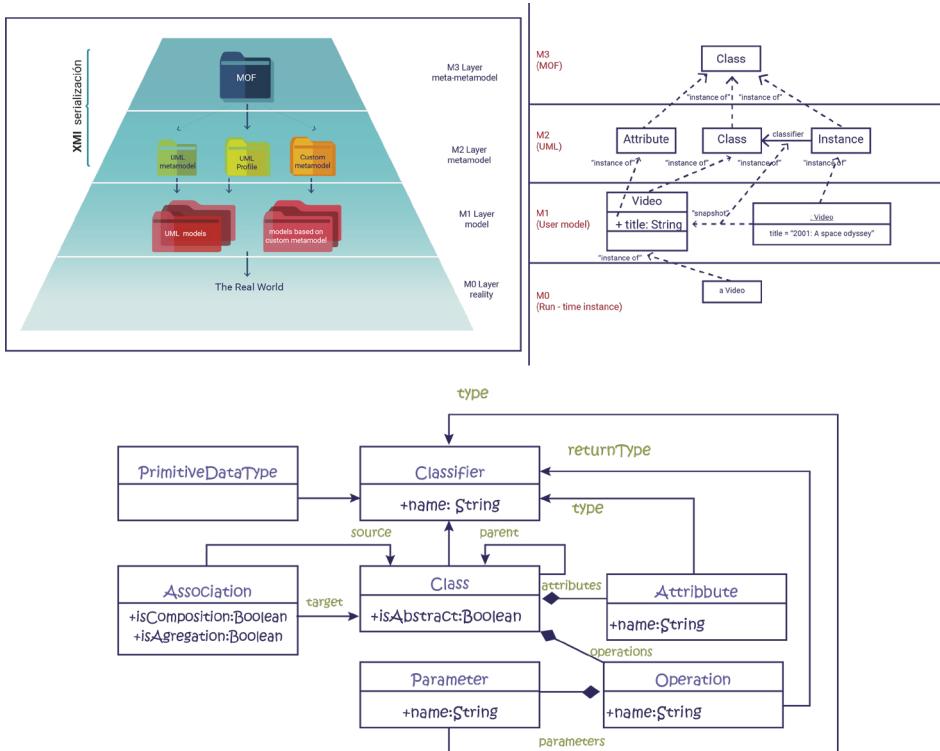


Figura 4. Meta-Object Facility (MOF), Modelo, Metamodelo y Metametamodelo de una Clase

Cabe mencionar que, asociado a cada principio, existe documentación básica, estandarizada y de referencia (ISO e IEEE), nomenclatura como UML y BPMN, que es usada para documentar los modelos y procesos de software.

#### 1.4. Modelado orientado a objetos

En esta sección es necesario recordar conceptos y principios ya conocidos del paradigma Orientado a Objetos. Conceptos tales como: clases, asociaciones, objetos, enlaces, atributos, operaciones y métodos. Principios tales como: abstracción, encapsulación,

generalización, polimorfismo, con la finalidad de que cuando abordemos más adelante este tema a fondo lo tengamos claro.

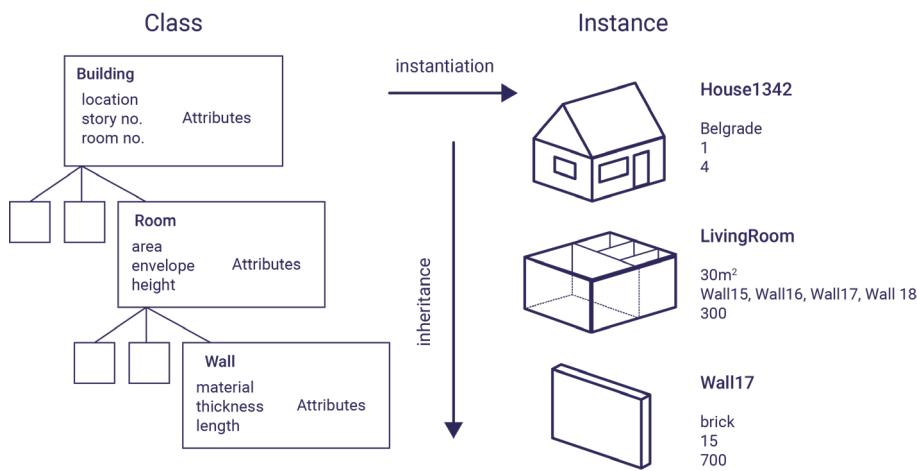


Figura 5. Ejemplo Clases, atributos, instanciación (instantiation), herencia (inheritance), objetos (instance)

No olvide que el éxito en el presente curso requiere que complemente el estudio con bibliografía adicional disponible en línea en formato digital. Con el desarrollo de los siguientes temas que se abordan de manera amplia en el texto básico, identificará que los modelos tienen dos aspectos principales: información semántica (metametamodelo-metamodelo), y presentación visual (modelo), los cuales se explicarán conforme avance su estudio



### Actividades de aprendizaje recomendadas

Una vez que ha estudiado los conceptos relacionados a la Unidad, le invito a desarrollar la Autoevaluación con el fin de evaluar los conocimientos adquiridos hasta el momento.



## Autoevaluación 1

Una vez que ha estudiado los conceptos relacionados a la Unidad, le invito a desarrollar la Autoevaluación con el fin de evaluar los conocimientos adquiridos hasta el momento.

1. ¿Cuál es la mejor alternativa con respecto al concepto de modelo?
  - a. El modelo corresponde a la codificación a nivel de código de un problema desde un punto de vista.
  - b. El modelo es similar al metamodelo, metamodelo y captura los aspectos más importantes de lo que se requiere representar acorde a un punto de vista.
  - c. El modelo de un sistema software es realizado utilizando lenguaje HTML.
  - d. El modelo captura los aspectos más importantes de lo que se requiere representar acorde a un punto de vista.

2. En base a las siguientes alternativas, seleccione la(s) que responda(n) a la pregunta ¿Para qué sirven los modelos?
  - a. Para capturar las necesidades de los clientes y clasificarlas en requisitos.
  - b. Para pensar acerca del diseño de un sistema de software (diagramas de alto nivel) más que en su codificación en sí.
  - c. Para generar productos de trabajo usables y reutilizables.
  - d. Para generar y explorar múltiples soluciones desde el punto de vista económico.
3. ¿Cuál de las siguientes opciones representan a los dos aspectos principales de un modelo?
  - a. Información Semántica e Información del Lenguaje
  - b. Información Semántica e Información Sintáctica
  - c. Información Semántica y Presentación Visual
  - d. Información Semántica y Presentación Modelada
4. ¿Qué tipo de información de una aplicación de software capturan los modelos?
  - a. Visual
  - b. Semántica
  - c. Léxica
  - d. Todas las anteriores

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

5. La calidad del software puede ser percibida desde dos puntos de vista, ¿cuál es la mejor alternativa?
  - a. Vista de desarrollador y Vista de alta gerencia
  - b. Vista de construcción y vista del producto
  - c. Vista de Proceso y Vista de Despliegue
  - d. Vista de Escenarios y Vista Lógica
6. ¿Cuáles son considerados tipos de producto software?
  - a. Software de sistemas
  - b. Software de tiempo real
  - c. Software empotrado
  - d. Software de gestión
7. Dentro de las categorías de aplicaciones web constan:
  - a. Informativas
  - b. Interactivas
  - c. Transaccionales
  - d. Workflow
8. ¿Cuál es el resultado que se obtiene de la recolección de análisis y requisitos?
  - a. Modelo del dominio de la solución
  - b. Modelo del dominio del problema
  - c. Modelo en UML
  - d. Todas las anteriores

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

9. ¿Qué involucra el modelo del dominio del problema?
  - a. Descripción de aspectos relevantes del sistema para el problema dentro del contexto.
  - b. Comprensión del entorno donde funcionará el sistema
  - c. Formulación y análisis del problema con el cliente.
  - d. Uso de técnicas como entrevista, observación directa, encuesta con el cliente.
  
10. ¿Qué involucra el modelo del dominio de la solución?
  - a. Comprensión de los sistemas a construir.
  - b. Evaluar diferentes posibles soluciones.
  - c. Participación de un equipo de desarrollo para la construcción del sistema.
  - d. Gestión del Producto software

[Ir al solucionario](#)

## Lecturas recomendadas

- (Unidad 1.3) En el texto básico revise por favor en el capítulo 1 el tema denominado **Principios de modelado**.
- (Unidad 1.4) Para revisar los conceptos y principios del paradigma orientado a objetos puede hacerlo en el texto básico en el capítulo 1 en la sección **Modelado Orientado a objetos** o hacer uso de bibliografía complementaria y recursos educativos abiertos para abordar este tema.
- (Unidad 1.4) Para comprender conceptos de meta-metamodelo meta-modelo y modelo le invito a buscar en la web y revisar estos conceptos en los siguientes enlaces donde se expone acerca de modelado.
- Conceptos y semántica de metamodelo, escrito en inglés.
- Conceptos y definiciones y ejemplos de metamodelo, escrito en español.
- Presentación de conceptos y explicación de las capas de meta-metamodelo, metamodelo, modelo y su representación con UML.
- Video explicativo de Modelo y Meta-modelo de clases, explicación en inglés y representación gráfica en UML.

**Ejercicio 2.** En sus propias palabras, y siguiendo el formato que se propone, mediante un pequeño ejemplo realice una breve exposición acerca de cada principio de modelado.

a. Principio 1:

- Características
- Ventajas/Desventajas
- Ejemplo

b. Principio 2:

- Características
- Ventajas/Desventajas
- Ejemplo

c. Principio 3:

- Características
- Ventajas/Desventajas
- Ejemplo

d. Principio 4:

- Características
- Ventajas/Desventajas
- Ejemplo

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

## Resultado de aprendizaje 3

Utiliza UML como lenguaje de construcción de modelos para modelar componentes de un sistema de información.

### Contenidos, recursos y actividades de aprendizaje



#### Semana 3



## Unidad 2. UML y conceptos generales

La presente unidad aborda las temáticas relacionadas con la presentación de UML y conceptos generales que se usan dentro de modelado. Empezamos con el tema de UML (Unified Modeling Language, por sus siglas en inglés), considerado como lenguaje o notación gráfica soportada por un metamodelo único que permite diseñar y describir sistemas de software, especialmente los que se implementan bajo el paradigma de orientación a objetos. Otra definición menciona que UML es un lenguaje visual utilizado para modelamiento y comunicación acerca de un sistema que usa diagramas y soportado por texto que permite la descripción del diagrama.

## 2.1. Visión general de UML

Si usted recuerda, en asignaturas que involucran codificación de software, como profesional dentro de la rama de Ciencias de la Computación o en su lugar de trabajo, ha utilizado documentación estandarizada o formal para comunicar de forma visual y textual algunos aspectos de un proceso o sistema. Esta actividad que involucra partir de un modelo UML para luego codificarlo se conoce como Ingeniería Directa (Forward Engineering), concepto que le invito a revisar en el texto básico o bibliografía complementaria. Por otro lado, el concepto de Ingeniería Inversa (Reverse Engineering) implica la construcción del modelo UML que se encuentra implementado en el código. Por favor revisemos estos conceptos, establezcamos sus diferencias y piense en qué escenarios podría utilizar cualquiera de los dos o su combinación.

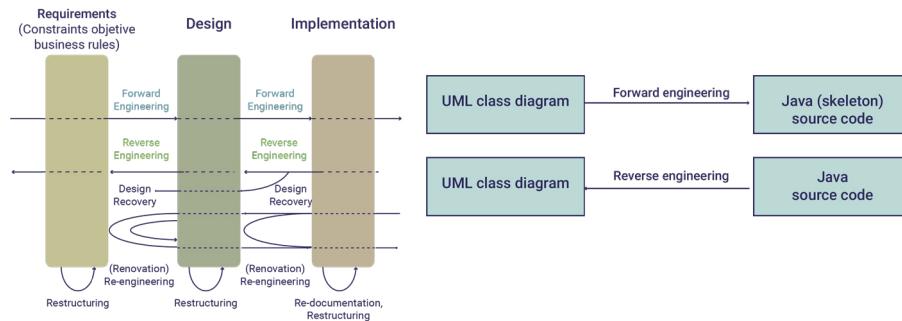


Figura 6. Diferencias entre Forward Engineering and Reverse Engineering

Una vez que puede dar respuestas a preguntas tales como ¿qué es UML?, ¿para qué sirve?, ¿cómo y cuando se lo utiliza?, es primordial que usted identifique los tres bloques principales del modelo conceptual, los cuales son: 1) bloques de construcción básicos de UML, 2) las reglas UML que permiten unir dichos bloques y 3) mecanismos comunes en UML.

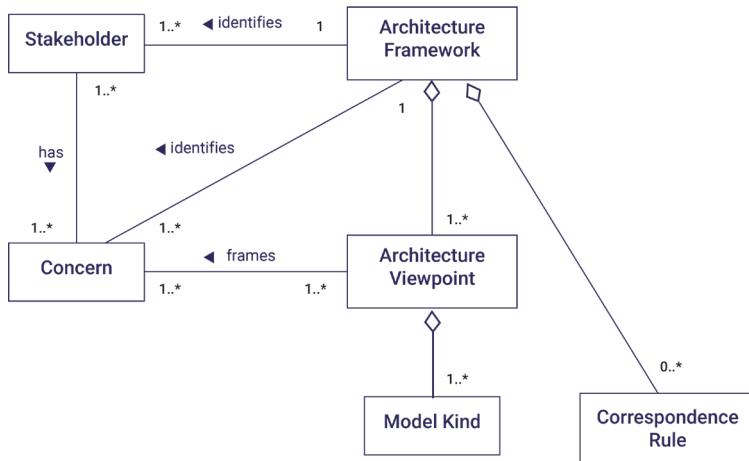


Figura 7. Figura 7 ISO/IEC/IEEE 42010 - Modelo conceptual de Descripción Arquitectónica

Las reglas y mecanismos comunes con los conceptos que se proponen revisar en el texto básico serán útiles para realizar la parte práctica, replíquelos y representelos usando como herramienta CASE de modelado StarUML. Es hora de abordar de forma breve conceptos básicos de arquitectura software, cuyo objetivo es estudiar el conjunto de decisiones que permiten el diseño, construcción y posterior implementación y evolución de los sistemas de software (en forma más ampliada los estudiará en la unidad 8 de la presente guía didáctica).

## 2.3. Arquitectura

La arquitectura software, es quizá la más importante que se puede utilizar para gestionar las diferentes vistas, puntos de vista y con ello controlar el desarrollo iterativo e incremental de un sistema a lo largo de su ciclo de vida. A la arquitectura software se la considera como la disciplina que permite cerrar la brecha entre la fase de requisitos y la fase de implementación, las cuales forman parte del ciclo de vida del software.

Bajo este contexto, por ejemplo, el framework propuesto por Krutchen es el modelo de vistas 4+1, el mismo que permite vistas tales como: vista lógica, vista de desarrollo, vista de proceso, vista física y vista de escenarios representar de forma visual y documentada las características y modelos principales del software.



Figura 8. Vistas del modelo 4+1 (se incluye diagramas y roles)



## Actividades de aprendizaje recomendadas

- (Unidad 2.1) Retome la lectura en el texto básico en el capítulo 2 y revise los aspectos relacionados a **UML y su visión general**.
- (Unidad 2.2) Para comprender mejor los tres bloques del modelo conceptual, es recomendable que revise de forma breve los subtemas que se proponen en el texto básico en el capítulo 2. Retome la lectura en el texto básico y revise los aspectos relacionados a **UML y su modelo conceptual**.
- (Unidad 2.3) Retome la lectura en el texto básico y revise los aspectos relacionados a UML el tema **introductorio de Arquitectura**.
- (Unidad 2.3) Para revisar la definición y elementos que forman parte de 4+1 view le invito a revisar los siguientes enlaces:
  - [Explicación en inglés de las vistas que forman parte del modelo para organizar los diagramas y los elementos UML que se usan](#).
  - [Explicación en español de las vistas que forman parte del modelo y los diagramas que intervienen.](#) (páginas 25-30).



## Semana 4

Espero que con los conocimientos adquiridos hasta el momento vaya desarrollando la tarea propuesta como parte de la praxis profesional (cátedra integradora/proceso práctico de aprendizaje), analizando el problema, las necesidades de los clientes y documentando los requisitos antes de empezar a diseñar los diagramas como parte de la documentación. ¡Buena suerte!

### 2.4. Ciclo de vida del desarrollo de software

En el texto básico, en el apartado relacionado con el ciclo de vida de desarrollo del software, se menciona que UML es utilizado de manera independiente al proceso de desarrollo que se utilice, sin embargo se sugiere utilizarlo en procesos como:

- Proceso racional unificado (RUP, Rational Unified Process por sus siglas en inglés).
- Centrado en la arquitectura.
- Iterativo e incremental.
- Proceso ágil.

Revise brevemente las fases que son parte del ciclo de desarrollo de software, las metodologías y modelos/procesos que se pueden usar para el desarrollo de software. El estudio de este tema, desde el punto de vista teórico y de implementación, lo revisará con profundidad en asignaturas de ciclos superiores.

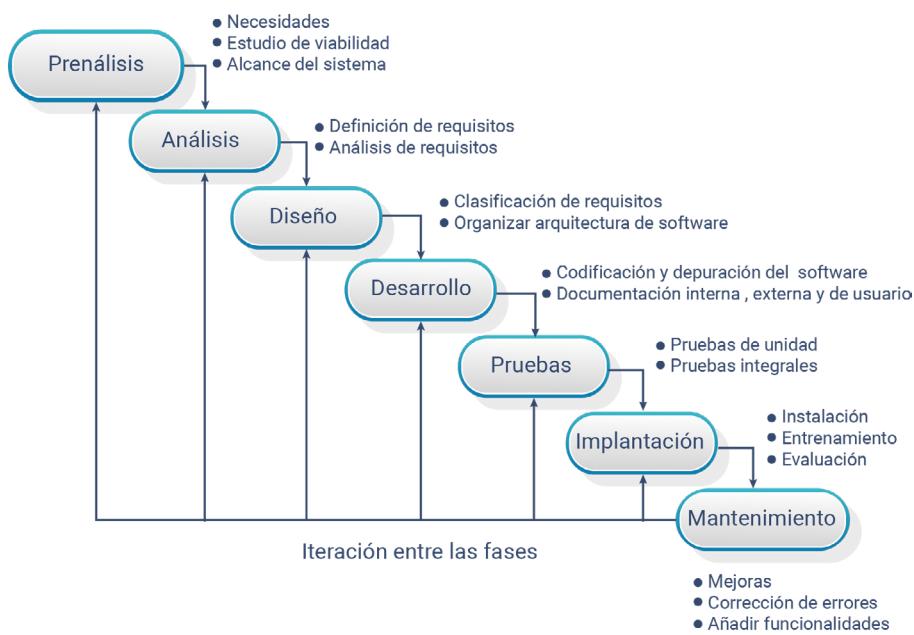


Figura 9. Fases del ciclo de desarrollo de software

Una vez que ha revisado los temas relacionados con UML, modelo conceptual, aspectos básicos e introductorios de arquitectura software y las fases del ciclo de vida de desarrollo del software, revisaremos el concepto de vistas y puntos de vista.

## 2.5. Vistas UML

Si bien en el texto básico se menciona de forma breve el tema de vistas, es necesario incluirlo en el presente curso, ya que es una forma de presentar de manera organizada los diferentes diagramas que se corresponden a distintas representaciones UML que de forma documentada y visual apoyan el desarrollo de software. Entre los diagramas asociados a modelado se tiene por ejemplo:

- Modelado de información (modelado de relaciones entre entidades y diagrama de clases).

- Modelado de comportamiento (diagramas de estado, análisis de casos de uso, diagramas de interacción, modos de fallo y análisis de efectos y análisis de árboles de fallas).
- Modelado arquitectónico (patrones arquitectónicos y diagramas de componentes).
- Modelado de dominios (enfoques de ingeniería de dominios).
- Modelado de empresas (procesos de negocio, organizaciones, objetivos y flujo de trabajo).
- Modelado de sistemas embebidos (programación en tiempo real, análisis y protocolos de interfaz).

Para efectos de estudio en la presente asignatura, tomaremos como referencia la representación de vista que se utilizan para organizar y presentar los conceptos de uno o varios tipos de diagramas UML con el fin de proporcionar una notación visual como lo exponen Rumbaugh, Jacobson, y Booch (2004).

Tabla 1. Vistas y diagramas UML.

<b>Major Area</b>	<b>View</b>	<b>Diagram</b>	<b>Main Concepts or UML Elements</b>
structural	static view	class diagram	association, class, dependency, generalization, interface, realization
	design view	internal structure	connector, interface, part, port, provided interface, role, required interface
		collaboration diagram	connector, collaboration, collaboration use, role
		component diagram	component, dependency, port, provided interface, realization, required interface, subsystem
	use case view	use case diagram	actor, association, extend, include, use case, use case generalization

Major Area	View	Diagram	Main Concepts or UML Elements
dynamic	state machine view	state machine diagram	completion transition, do activity, effect, event, region, state, transition, trigger
	activity view	activity diagram	action, activity, control flow, control node, data flow, exception, expansion region, fork, join, object node, pin
	interaction view	sequence diagram	occurrence specification, execution specification, interaction, interaction fragment, interaction operand, lifeline, message, signal
		communication diagram	collaboration, guard condition, message, role, sequence number
physical	deployment view	deployment diagram	artifact, dependency, manifestation, node
model management	model management view	package diagram	import, model, package
	profile	package diagram	constraint, profile, stereotype, tagged value

Fuente: Rumbaugh, et al. (2004)

Tenga en cuenta la representación y distribución a nivel de área (*major area*), vista (*view*), diagrama (*diagram*) y principales conceptos (*main concepts*) expuestos en la Tabla 3, ya que estos servirán de apoyo y referencia para ubicar al diagrama objeto de estudio dentro de cada una de las unidades del presente curso.



### Actividades de aprendizaje recomendadas

Con base en las siguientes preguntas, seleccione la mejor alternativa o alternativas que se proponen para cada una de ella.



## Autoevaluación 2

En base a las siguientes preguntas, seleccione la mejor alternativa o alternativas que se proponen para cada una de ella.

1. ¿Cómo se conoce a la representación de un sistema completo desde la perspectiva de un conjunto relacionado de intereses o problemas relativos a un sistema?
  - a. Vista.
  - b. Punto de Vista.
  - c. Punto de Vista estructural.
  - d. Punto de Vista dinámico.
2. ¿Cómo se conoce a la descripción arquitectónica organizada dentro de una o más vistas?
  - a. Vista.
  - b. Punto de Vista.
  - c. Punto de vista estructural.
  - d. Punto de vista dinámico.
3. Seleccione la mejor alternativa de solución. Las convenciones para construir y usar una vista se usan dentro de:
  - a. Vista.
  - b. Punto de Vista.
  - c. Punto de vista estructural.
  - d. Punto de vista dinámico

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

4. Seleccione la mejor alternativa de solución. La organización estructural, componentes y sus interacciones se exponen en:
  - a. Vista.
  - b. Punto de vista estructural.
  - c. Punto de vista dinámico.
  - d. Punto de vista físico.
5. Seleccione la mejor alternativa de solución. Las acciones dinámicas “de” y “dentro” de un sistema (ordenamiento, sincronización, etc.) corresponden a:
  - a. Vista.
  - b. Punto de vista estructural.
  - c. Punto de vista dinámico.
  - d. Punto de vista físico.
6. Seleccione la mejor alternativa de solución. Las interconexiones de comunicaciones físicas y su ordenamiento entre los componentes del sistema se exponen en:
  - a. Vista.
  - b. Punto de vista estructural.
  - c. Punto de vista dinámico.
  - d. Punto de vista físico.
7. ¿Cómo se considera a 4+1 de Krutchén?
  - a. Metodología de referencia.
  - b. Modelo de referencia.
  - c. Lenguaje Unificado de referencia
  - d. Framework o marco de referencia

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

8. Las vistas que se propone en 4+1 son:

- a. Vista Lógica, Vista de Secuencia, Vista de Proceso, Vista Física, Escenarios.
- b. Vista Lógica, Vista de Desarrollo, Vista de Clases, Vista Física, Escenarios.
- c. Vista Lógica, Vista de Desarrollo, Vista de Proceso, Vista Física, Escenarios.
- d. Vista Lógica, Vista de Desarrollo, Vista de Proceso, Vista Secuencia, Escenarios.

9. ¿Qué diagramas incluye la vista de desarrollo?

- a. Secuencia, Comunicación.
- b. Clases, Objetos.
- c. Componentes, Paquetes.
- d. Despliegue, Topología de red (No UML).

10. ¿Qué diagramas incluye la vista lógica?

- a. Clases y Objetos.
- b. Componentes, Paquetes.
- c. Secuencia, Comunicación..
- d. Despliegue, Topología de red (No UML).

Ir al solucionario

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

**Ejercicio 3.** Exponga al menos cuatro propósitos donde UML como lenguaje se puede utilizar.

**Ejercicio 4.** Exponga al menos cinco áreas o dominios donde se podría utilizar UML.

**Ejercicio 5:** Mediante un esquema u organigrama represente los tres elementos principales del modelo conceptual.

**Ejercicio 6:** Utilizando el modelo de vistas de arquitectura 4+1 ubique o clasifique los diagramas que corresponden y se utilizan en cada vista.

**Ejercicio 7:** Ubique las actividades que se realizan en cada una de las fases del ciclo de vida de desarrollo del software.

### Bibliografía y lecturas recomendadas

- (Unidad 2.4) En el texto básico revise en el capítulo 1 los aspectos relacionados a ¿cómo se utiliza UML en el ciclo de vida de desarrollo de sistemas?. Recuerde utilizar estrategias de lectura como subrayado o anotaciones.
- (Unidad 2.4) [Para reforzar los conceptos y validar otros ejemplos, le invito a revisar el recurso educativo abierto \(Ingeniería de Software\) Tema 2. Lenguaje Unificado de Modelado \(UML\)](#),

## Resultado de aprendizaje 4

Diseña representaciones para facilitar la comunicación, discusión, exploración y validación de la especificación y diseño de un sistema.

### Contenidos, recursos y actividades de aprendizaje



#### Semana 5

Ahora, revisemos la sección correspondiente al Modelado Estructural Básico, que comprende el estudio de subtemas como clases, introducción, términos y conceptos, atributos, operaciones, responsabilidad y técnicas comunes de modelado.

#### 2.6. Modelado estructural básico

Ahora que ya se encuentra en capacidad de responder a preguntas tales como ¿por qué y para qué se modela?, ¿cuáles son algunas de las vistas, diagramas y principales conceptos del modelado? ¿cómo y cuando utilizar UML para tareas de modelado? y otras cuestiones desarrolladas en las secciones anteriores, está en capacidad de iniciar el estudio referente a modelado estructural básico teniendo en cuenta lo siguiente:

- Para llegar a modelar algo siempre debe existir necesidades dadas por los clientes, quienes desean automatizar o solucionar un proceso mediante un sistema o solución software.

- Dichas necesidades deben ser analizadas y en el análisis se convierten en uno o varios requisitos, los cuales pueden clasificarse como funcionales y/o no funcionales.<sup>1</sup>
- Por lo general, requisitos funcionales son aquellos que deben ser modelados y son visibles en la implementación del sistema y los requisitos no funcionales son considerados externos, que no necesariamente se visualizan en la interfaz gráfica de usuario pero que están ligados al cumplimiento de la calidad del software a través de los atributos de calidad.
- Los requisitos pueden ser documentados y/o modelados formalmente. En ambos casos es necesario utilizar estándares tales como ISO/IEC e IEEE, definidos para software (por ejemplo, ISO/IEC 9126, ISO/IEC/IEEE 42010, ISO/IEC 12207, ISO/IEC/IEEE 29148).<sup>2</sup>

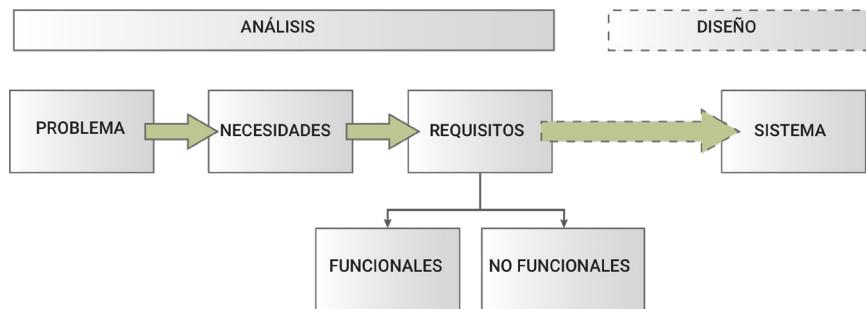
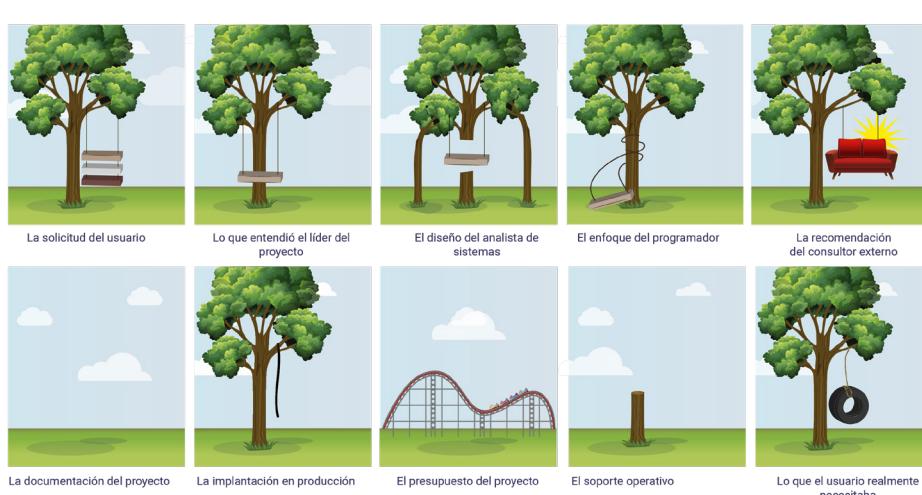


Figura 10. Fases previas al diseño de una propuesta de solución

- Los requisitos formalizados deben ser entendidos, validados y aceptados por el cliente. Por ello, el analista en la fase de levantamiento de requisitos y análisis (los cuales forman

<sup>1</sup> Se recomienda revisar en bibliografía base o complementaria de forma breve los conceptos de requisitos funcionales y no funcionales.

<sup>2</sup> Se recomienda revisar en bibliografía base o complementaria de forma breve los estándares ISO/IEC propuestos y pensar en su uso como parte del desarrollo de software.



*Figura 11. Desde la solicitud del usuario hasta lo que necesita.*

Es necesario en este punto conocer los roles de personas que intervienen dentro del desarrollo de software (analistas de negocio, analistas de sistemas, arquitectos de software, documentadores, desarrolladores, control de calidad, administrador de base de datos, responsable de release, líder de equipo de desarrollo, entre otros). Esto se puede observar en la Figura 12, donde personas dependiendo de sus capacidades forman parte del equipo de trabajo que gestionará el proyecto y construirá el producto de software.

Con los enfoques ágiles los roles existen, lo que cambia es su distribución o acoplamiento dependiendo de las fases, actividades y tareas que se asignen.

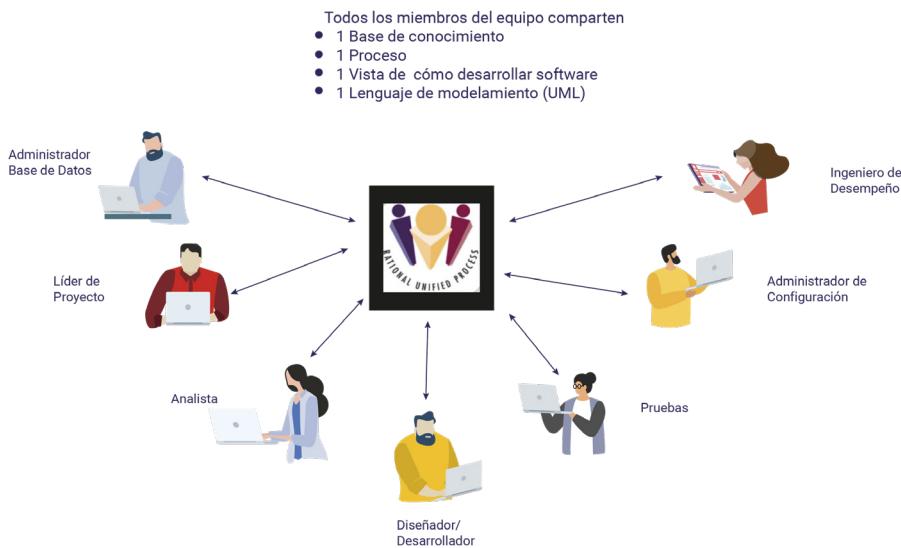


Figura 12. Roles de personas dentro de RUP.

Recuerde que por el momento estamos enfocados en la adquisición de los conceptos teóricos, unos conocidos y otros nuevos, que posteriormente los usaremos en la práctica. Algo que debe tener claro es que los diagramas son utilizados como apoyo para representar formalmente y de manera estilizada los requisitos dados por los clientes. Dependiendo del diagrama objeto de estudio, es necesario conocer los elementos UML básicos para su representación (nomenclatura, formas de los elementos, asociaciones, estereotipos, propiedades).



## Unidad 3. Casos de uso y diagramas de caso de uso

Antes de iniciar con el estudio referente a los casos de uso, es necesario preguntarse, ¿cuál es el insumo de entrada para elaborar los casos de uso?. Muy bien, la respuesta es la documentación que contiene la especificación de los requisitos (requisitos funcionales y requisitos no funcionales). Cuando el cliente nos solicita dar una solución de software, siempre debemos partir del entendimiento y comprensión del problema o las necesidades de los clientes, que transformados en requisitos son los que se pueden especificar, describir y modelar, con el objetivo de que con ellas podamos proponer una o varias soluciones automatizadas a un problema dentro de un contexto.

La correcta identificación, clasificación y redacción de los requisitos funcionales y no funcionales, ayudarán en la definición de los actores, los casos de uso, las relaciones y otros elementos y artefactos que se usan para representar una solución con el uso de Diagramas. Antes de iniciar la explicación de casos de uso, es necesario responder a las preguntas: ¿qué es un diagrama?, ¿para qué sirven? y ¿cómo se los diseña?. Estas preguntas serán resueltas conforme avanza con su autoestudio.

### 3.1. Diagrama

El concepto de diagrama, explicado y comparado con otros tales como: sistema, subsistema, modelo y vista, los puede revisar a profundidad en el texto básico y bibliografía complementaria, con la finalidad de que establezca similitudes y diferencias.

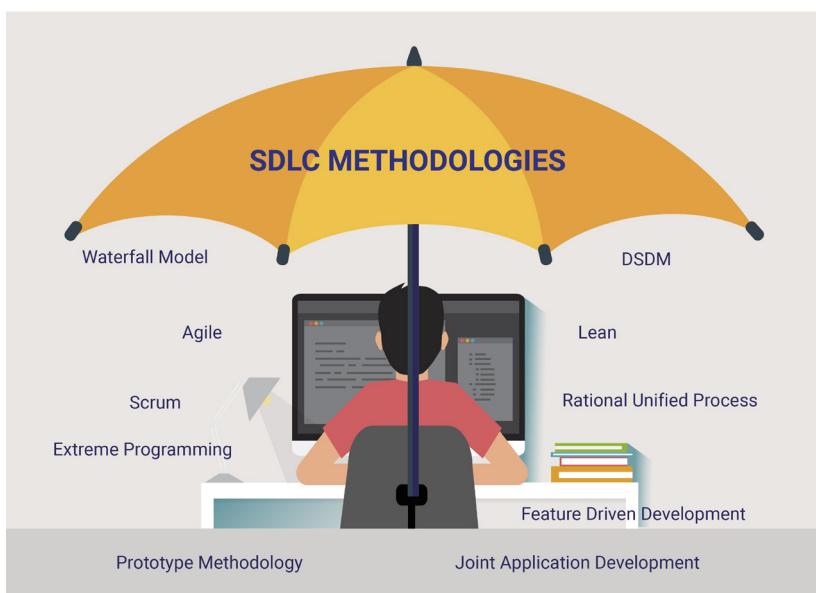


Figura 13. Metodologías usadas en el desarrollo de Software (SDLC: Software Development Life Cycle).



### Actividades de aprendizaje recomendadas



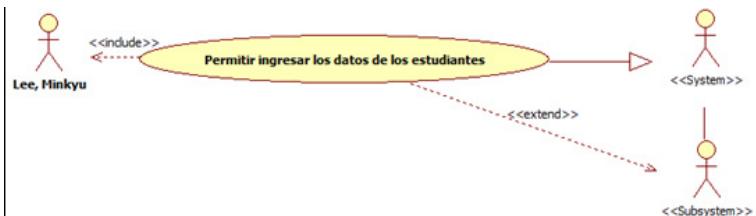
## Autoevaluación 3

Para evaluar sus conocimientos, por favor seleccione la mejor alternativa para cada una de las preguntas que se proponen en la Autoevaluación 3.

1. ¿Cuáles son los elementos que forman parte de un diagrama de caso de uso? Seleccione las opciones que considere correctas.
  - a. Actores.
  - b. Casos de uso.
  - c. Ámbito del Sistema o Boundary.
  - d. Relaciones.
  
2. ¿Cuál es la mejor alternativa respecto al concepto de casos de uso?
  - a. Técnica para a través de un lenguaje de programación presentar los requisitos.
  - b. Técnica para diseñar la interacción cliente – servidor.
  - c. Técnica para modelar las entidades que forman parte de la solución.
  - d. Técnica para capturar información respecto de los servicios que proporciona a su entorno.

3. ¿Cuál es la mejor alternativa respecto a la utilización de los casos de uso?
  - a. Proporcionan una descripción general de todos o parte de los requisitos de uso para un sistema u organización.
  - b. Los casos de uso son usados para comunicar el alcance de un proyecto de desarrollo.
  - c. Muestran las relaciones entre actores y casos de uso dentro de un sistema.
  - d. Todos los anteriores.
4. ¿Cómo se conoce al conjunto de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software?
  - a. Metodología de desarrollo.
  - b. Proceso de Software.
  - c. Casos de Uso.
  - d. Requisitos.
5. ¿Cuál de las siguientes opciones se consideran como modelos de desarrollo de software?
  - a. Cascada, Espiral, Iterativo.
  - b. Ágil, Prototipo, Incremental.
  - c. Desarrollo rápido de aplicaciones.
  - d. Todos los anteriores.
6. ¿Cuál de las siguientes opciones se consideran como procesos de desarrollo de software?
  - a. RUP (Rational Unified Process).
  - b. Métrica V3, PMBOK.
  - c. SCRUM, CMMI.
  - d. Todas las anteriores.

7. ¿Cuáles se consideran como técnicas para capturar requisitos? Seleccione las que considere correctas.
- Entrevistas.
  - Escenarios.
  - Obser.
  - vación Directa.
  - Prototipado.
8. Seleccione Verdadero o Falso a la siguiente pregunta. ¿Las fases del ciclo de vida de desarrollo de software son parte de los Procesos de desarrollo?
- Verdadero.
  - Falso.
9. Las fases nombradas como Inicio, Elaboración, Construcción y Transición pertenecen a:
- Cascada.
  - Ágil.
  - RUP.
  - XP (eXtreme Programming).
10. ¿Cuál es el error en el siguiente diagrama de caso de uso?



- La relación de <<include>> no es la correcta.
- El nombre del caso de uso no es el correcto.
- La relación de <<exclude>> no es la correcta.
- No se puede relacionar dos actores entre sí.

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

## Lecturas recomendadas

- (Unidad 2.6) Para entender lo relacionado con los requerimientos, su forma de identificación y documentación, le invito a revisar el video cuyo nombre es Análisis de requerimientos.
- (Unidad 2.6) Para comprender y asociar los conceptos de la unidad, le invito a que realice la lectura en el texto básico el capítulo 2, y revise los aspectos relacionados a Modelado Estructural Básico el mismo que comprende el tema de clases, términos, conceptos, técnicas comunes para modelar así como sugerencias y consejos de modelado.
- (Unidad 3.1) En el texto básico, en el capítulo 7 referente a Diagramas, es necesario que identifique y clasifique los conceptos de Diagramas, técnicas comunes de modelado, sugerencias y consejos.
- (Unidad 3.1) Para representar los diagramas una herramienta CASE alternativa es STARUML.
  - El siguiente enlace hace referencia a la guía de StarUML a través de la cual se puede documentar y graficar de forma estandarizada los diagramas.



## Semana 6

### 3.2. Casos de uso

Similar a los conceptos estudiados anteriormente, con base en un problema o una necesidad dada por un cliente se identifican los requisitos que pueden ser implementados en el sistema. Estos requisitos son insumos necesarios para diseñar casos de uso, donde existen dos aspectos a tener en cuenta. El primero, relacionado con la especificación del caso (descripción textual de los requisitos identificados) y el segundo con la representación visual con el diagrama de casos de uso. Estos últimos se representan utilizando UML como lenguaje que permite modelar la interacción entre los elementos del diagrama tales como Actores, las Relaciones entre los actores y los Casos de Uso dentro de un Contexto (ámbito del sistema).

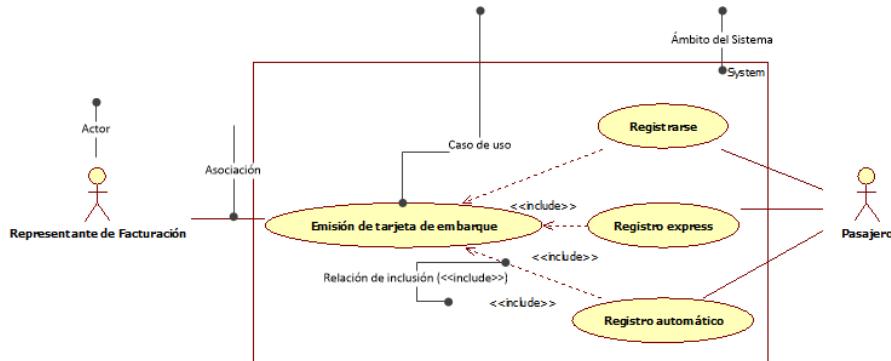


Figura 14. Elementos de diagrama de caso de uso

En la presente unidad se exponen los conceptos teóricos referentes a casos de uso y su representación gráfica a través de UML.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Gráficamente, en un diagrama de caso usan estereotipos y elementos UML definidos, como los que se expondrán mas adelante. Algunas de las recomendaciones relacionadas con el diagrama y descripción de casos de uso, las encontrará en el texto básico y bibliografía complementaria como (Fowler, 2004) donde se sugieren algunos aspectos a tener en cuenta para su elaboración y redacción (ver Figura 15). Le recomiendo revisarlos para de esta forma empezar a modelar de forma correcta.

Use Case Section	Comment
Use Case Name	Start with a verb.
Scope	The system under design.
Level	“user-goal” or “subfunction”
Primary actor	Calls on the system to deliver its services.
Stakeholders and Interests	Who cares about this use case, and what do they want?
Preconditions	What must be true on start, and worth telling the reader?
Success Guarantee	What must be true on successful completion, and worth telling the reader.
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenarios of success or failure.
Special Requirements	Related non-functional requirements.
Technology and Data Variations List	Varying I/O methods and data formats.
Frequency of Occurrence	Influences investigation, testing, and timing of implementation.
Miscellaneous	Such as open issues.

Figura 15. Sugerencias para diseñar casos de uso. Fuente: Fowler (2004).

Recuerde utilizar y revisar bibliografía complementaria como Ambler (2005) en el apartado denominado “*Use-Case Guidelines*”, “*Actor Guidelines*”, “*Relationship Guidelines*” en donde encontrará escrito en idioma inglés algunas sugerencias que se brindan para diseñar diagramas de casos de uso entre las que destacan:

- Los nombres de caso de uso deben iniciar con un verbo fuerte.
- Coloque su (s) actor (es) principal (es) en la esquina superior izquierda del diagrama.
- Los nombres de los actores deben ser escritos en singular y relacionados con el dominio o ámbito del sistema.
- Cada actor se puede asociar con uno o más casos de uso, no hay actores sin relaciones.
- No debe permitir que los actores interactúen entre sí, sino que se relacionen con un caso de uso, sistema, subsistema o componente.

### 3.3. Diagramas de caso de uso

Cuando se trabaja con diagramas o especificación de caso de uso, normalmente se sugiere utilizar lenguaje natural (*como nos expresamos formalmente para explicar el funcionamiento o comportamiento de algo*) para expresar lo que se realizaría con el sistema, teniendo en cuenta que pueden existir excepciones, errores o flujos de eventos excepcionales.

En el siguiente apartado se representa de forma visual y utilizando nomenclatura y elementos UML los diagramas de casos de uso los cuales muestran casos de uso, actores y relaciones.

Un escenario se considera como una secuencia de pasos que describen una interacción entre un actor y el sistema u otro actor. Por lo tanto, un caso de uso es un conjunto de escenarios unidos por una meta de usuario común. Un actor se considera como el papel en el que un usuario o sistema externo cumple para interactuar con el sistema que se está modelando, la interacción y uso entre el actor y el caso de uso se representa a través de una asociación, lo que permite indicar que el actor se comunica con el sistema y participa

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

en el caso de uso. Una asociación en UML, se muestra como una línea sólida continua o discontinua entre un actor y un caso de uso sirve para relacionar un caso de uso con múltiples actores u otros casos de uso. Dos de las relaciones o asociaciones que se utilizan en un diagrama de caso de uso son inclusión (“*include*”) y extensión (“*extend*”).

En el texto básico encontrará las definiciones y representación UML para cada una de ellas. Sin embargo, se puede mencionar que “*include*” indica que un caso de uso base incluirá o generará otro caso de uso necesario y de forma obligatoria, mientras que “*extend*” indica que el caso de uso se extenderá (o se insertará en..) y aumentará el caso de uso base de forma opcional. En el caso de las relaciones, es muy importante tener en cuenta la dirección de las flechas y el tipo de línea (continua o discontinua) las cuales indican inclusión o extensión.

En el diagrama de la figura 16, se muestra un ejemplo de diagrama de caso de uso con sus elementos y nomenclatura UML, donde, por ejemplo, el caso de uso Seleccionar ruta incluye Seleccionar Fecha. Observe la dirección de la flecha y la forma que se usa para mostrar la inclusión o exclusión. Lo que se encuentra dentro de “<<.....>>”, para el caso de la asociación y de la representación de actores (sistemas o subsistemas) se considera como estereotipo. Podemos observar tres actores Cliente, Sistema de Boletos Aéreos, Servicio de PayPal. Además, existen comentarios nombrados como actor, caso de uso e include. Los comentarios pueden ser usados para describir alguna restricción o aspecto a considerar de los elementos UML que usted considere, en este caso es solo explicativo. Con el diagrama se observa que el Cliente, utilizando el sistema, podrá Seleccionar ruta, Seleccionar fecha y Pagar, el sistema de boletos, considerado como un servicio web, por ejemplo, actualizará la disponibilidad de asientos, y para el pago se usará un servicio web provisto por parte de PayPal.

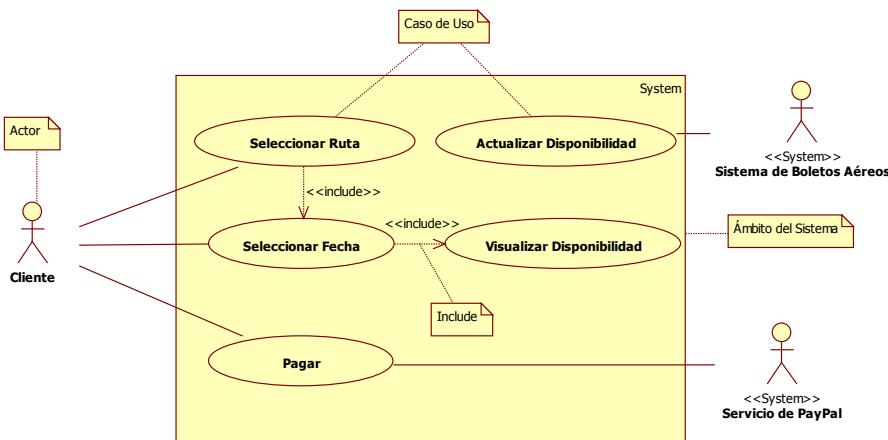


Figura 16. Diagrama de Casos de Uso

Los detalles detrás de cada elemento en el diagrama de casos de uso pueden ser expresados de forma textual o utilizando otras técnicas de modelado UML. Los diagramas de casos de uso y sus detalles asociados para un sistema específico expresan los requisitos funcionales del sistema. Sin embargo, UML no proporciona ninguna orientación explícita sobre cómo capturar los detalles del texto, sino que se centra más en la notación.

Al finalizar el estudio de la unidad 3 en esta semana, donde se han revisado conceptos como diagramas, casos de uso y su representación gráfica a través de diagramas de caso de uso Ud. como parte de su formación



### Actividades de aprendizaje recomendadas

#### Ejercicio 8

Proponga en un pequeño caso, describalos y represente usando UML donde se incluyan los elementos que intervienen en un diagrama de casos de uso.

## Ejercicio 9

Con base al siguiente escenario, represente las alternativas de solución usando diagramas de caso de uso.

**Proceso de préstamo de libros:** en la biblioteca de UTPL un estudiante requiere tomar prestado un libro. Para ello, el estudiante ha ubicado el libro y ha consultado al bibliotecario la disponibilidad de este, por lo que el bibliotecario ingresa al sistema de préstamos de libros, verifica disponibilidad, ingresa el número de identificación del estudiante, se recuperan los datos de matrícula del estudiante para registrar el préstamo. En caso de que no se recuperen los datos del estudiante, es porque no se encuentra matriculado en el periodo académico actual, el bibliotecario llena los datos adicionales para el préstamo como fecha de préstamo, estado del libro, quién presta el libro y registra el préstamo.

### Bibliografía y lecturas recomendadas

- (Unidad 3.2) Para conocer más sobre los casos de uso, le invito a que revise en el texto básico el capítulo 17 correspondiente a Casos de Uso.
- (Unidad 3.3) Le invito a retomar la lectura en el texto básico en el capítulo 18 denominado Diagramas de Casos de Uso, en el cual se exponen los aspectos teóricos y prácticos de los casos de uso.
- (Unidad 3.3) [Revise los elementos UML que se usan en los diagramas de casos de uso utilizando StarUML](#).
- (Unidad 3.3) [Revise lo que se expone en el recurso educativo abierto \(Ingeniería de Software\) en el tema 5 referente a casos de uso, contexto y requisitos del sistema. Páginas 61, 62, 63, páginas 73 a la página 88.](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

## Resultado de aprendizaje 6

Selecciona los componentes que debe incluir en un proceso para solucionar un problema específico.

### Contenidos, recursos y actividades de aprendizaje



Semana 7



#### Unidad 4. Diagramas de Interacción y diagrama de actividades

Continuando con nuestro estudio, en el texto básico se exponen los temas relacionados a diagramas de interacción (secuencia y comportamiento), los cuales son una técnica para representar el modelado dinámico de un sistema. Una interacción a menudo modela la ejecución de una operación, caso de uso u otra entidad de comportamiento específica.

#### 4.1. Diagramas de interacción o de secuencia

En la presente asignatura se abordarán los diagramas de secuencia, ya que estos apoyan en la exposición gráfica de un escenario particular, de un caso de uso y los eventos e intercambio de mensajes que generan los actores externos con un orden especificado. Normalmente los diagramas de secuencia se usan durante las fases de análisis y diseño para documentar y comprender el flujo lógico del sistema.

Como lo recomiendan algunos autores, un diagrama de secuencia debe utilizarse cuando:

- Se tenga una necesidad de exponer a alguien, cómo colabora un grupo de clases u objetos dentro de un caso de uso particular.
- Se requiera visualizar dicha colaboración desde el punto de vista de funcionamiento del sistema.
- Quiera observar el comportamiento de varios objetos dentro de un solo caso de uso (diagrama de estado) o en muchos casos de uso o en muchos hilos (diagrama de actividad).
- Desea describir el flujo de mensajes, eventos, acciones entre objetos.
- Desea mostrar procesos concurrentes, activaciones y secuencias de tiempo que no se representan fácilmente en otros diagramas.

Generalmente, los diagramas de secuencia no son utilizados como documentación obligatoria en el desarrollo de los sistemas, sino como instrumentos para perfeccionar sus habilidades, como rol de analista y documentar ciertas interacciones del sistema. Le invito a

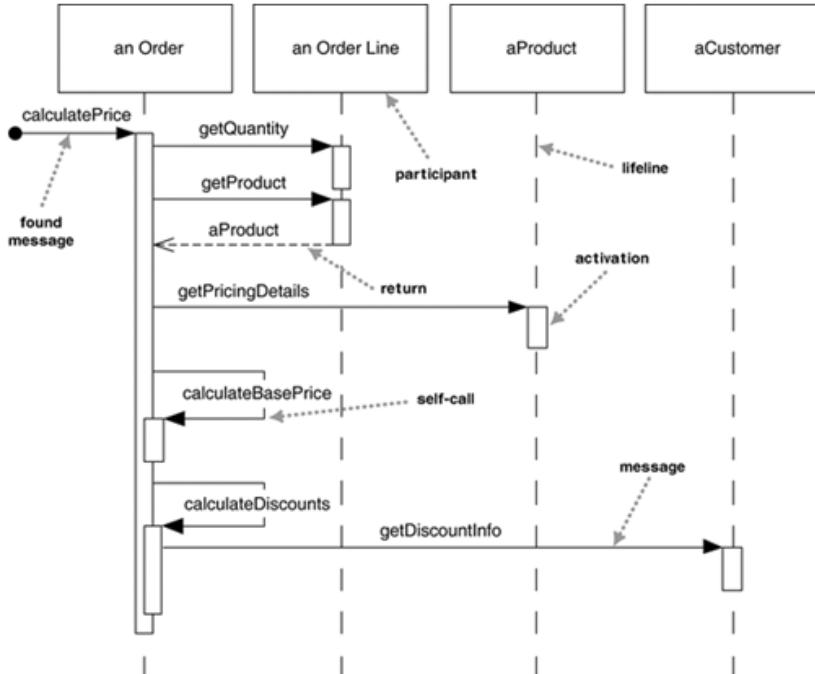


Figura 17. Elementos de un diagrama de secuencia. Fuente: Fowler (2004).

Explicando los elementos que forman parte de un diagrama de secuencia, los ejes en este tipo de diagramas se encuentran distribuidos de forma horizontal y vertical. En el eje horizontal se muestran los participantes y en el eje vertical se representan la línea de vida de cada participante.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Los participantes son objetos o clases que actúan en el diagrama. Con base en la figura 17 se puede observar que en un diagrama de secuencia dentro de los rectángulos se representa a los participantes. Los participantes pueden tener tres representaciones como las que se muestran en la Tabla 4 Representación de elementos UML que son parte de un diagrama de interacción.

Tabla 2. Representación de elementos UML que son parte de un diagrama de interacción.

Definición	Ejemplo	Explicación
		Se hace referencia a un Objeto de una clase en particular, el objeto seguido de ":" luego el nombre de la clase de la cual se crea el Objeto. <>:<>
		Se hace referencia a Objetos anónimos ubicando solamente el nombre de la clase de la cual se crearían uno o varios objetos de la clase. Se ubica solamente ":" seguido del nombre de la Clase. <<:Clase>>
		Se ubica objetos anónimos es decir no se conoce a la clase que pertenecen. <>

Fuente: elaboración propia.

Los mensajes permiten la comunicación (llamadas de método) entre objetos participantes indicadas por tipos de flechas que indican los tipos de mensajes. Dependiendo del tipo de flecha se crea el elemento activador (ver Tabla 5).

Tabla 3. Uso de flechas y explicación de elementos UML que intervienen en un diagrama de interacción.

Tipo de Flecha	Descripción	Explicación
<pre> sequenceDiagram     participant Juan as Juan:Persona     participant Persona as :Persona     Juan-&gt;&gt;Persona: 1 : Sincrono()   </pre>	Mensaje síncrono	Permite la activación de otro objeto, de la cual se espera una respuesta.
<pre> sequenceDiagram     participant Juan as Juan:Persona     participant Persona as :Persona     Juan-&gt;&gt;Persona: 1 : Asincrono()   </pre>	Mensaje asíncrono	Permite pasar un mensaje de un objeto A a uno B sin necesidad de esperar una respuesta.
<pre> sequenceDiagram     participant Juan as Juan:Persona     participant Persona as :Persona     Juan-&gt;&gt;Persona: 1 : Envio()     Persona--&gt;&gt;Juan: 2 : Respuesta   </pre>	Mensaje respuesta	La respuesta desde otro objeto es a través de línea punteada.

Tipo de Flecha	Descripción	Explicación
		Sobre las flechas se ubican el nombre del mensaje (Crear) y los argumentos (objPersona), por ejemplo Crear(objPersona), en este caso un argumento objPersona, pero tambien puede ser Crear (idPersona, nombres). Los argumentos dentro de un mensaje son opcionales y se utilizan para representar de mejor manera la creación de objetos.
		<i>*En StarUML los argumentos del mensaje se visualizan al hacer doble click sobre el mensaje o en Propiedades – Detail – Arguments del mensaje seleccionado.</i>

Fuente: elaboración propia.

Las líneas de vida se encuentran representadas por una línea punteada vertical que se traza desde el centro del elemento que representa a los participantes y exponen la vida útil de cada uno de ellos durante el escenario modelado.

Una activación se representa como un cuadro rectangular alargado que se ubica sobre la línea de vida del objeto, dibujado cuando el método de un objeto está en la pila. En la figura 18 se muestran dos elementos de activación en los participantes Controlador y Modelo. El mensaje nombrado con el número 4, se conoce como autoestímulo para indicar que un objeto se llama asimismo.

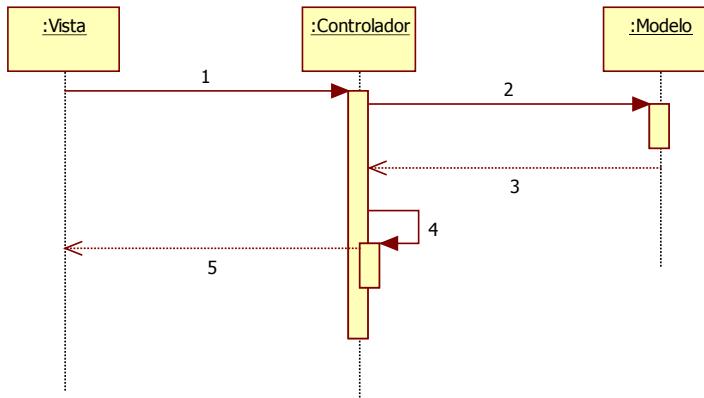


Figura 18. Representación de activación

La línea de vida de los objetos tienen dos estados creación y eliminación. La creación es representada cuando a partir de un objeto se llama a otro o también se lo representa con un mensaje de “new” o “create” escrito sobre la flecha que hace referencia al objeto. Cuando se crea un objeto, gráficamente este se ubica debajo de los otros objetos. La eliminación es marcada por una “X” o con el mensaje “destroy” y se muestra al final de la línea de vida de un objeto (ver figura 19).

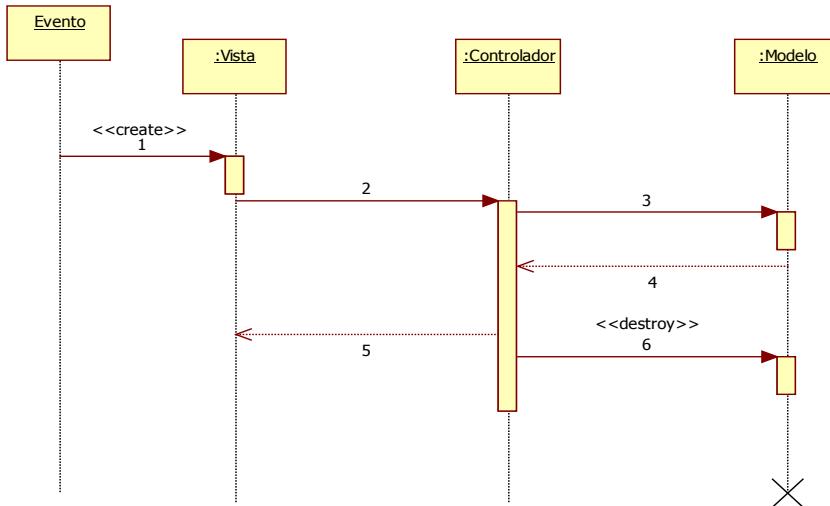


Figura 19. Representación de líneas de vida y estado de los objetos.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

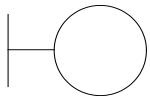
El fusionar y complementar los elementos que se utilizan en un diagrama de secuencia y representarlos de forma gráfica es una tarea que requiere capacidad de análisis, tiempo y uso de buenas prácticas. Por ello, recuerde utilizar y revisar bibliografía complementaria escrita en idioma inglés como (Ambler, 2005) en el apartado denominado “*UML Sequence Diagrams*”, “*Guidelines for Lifelines*”, “*Messages Guidelines*” y “*Guidelines for Return Values*”, lo cual le puede ayudar a comprender mejor la unidad referente a diagramas de secuencia.

Existen otros elementos que forman parte de un diagrama de secuencia, tales como alternativas (*alt*), opciones (*op*) y bucles (*loops*), donde para cada una de ellas se utiliza *Combined Fragment*, *Interaction Operand* y *Frame*. En la presente guía no se realiza una explicación a fondo de estos elementos. Sin embargo, le animo a que los revise en el texto básico, ya que su representación se deriva de los elementos básicos de un diagrama de secuencia y estos se ven mejor a nivel de implementación, utilizando cualquier lenguaje de programación.

Para realizar la codificación de una funcionalidad en cualquier lenguaje de programación, se toma como base un modelo que tiene asociado un metamodelo. Por lo tanto, ¿puede usarse el código para describir parte de un sistema (funcionalidades) y reemplazar el diseño representado con cualquier diagrama? Desde mi punto de vista esto sería adecuado y en la práctica ocurre. Escribir el código ajustado a los requisitos funcionales y no funcionales analizados y tomando en consideración los insumos de análisis, diseño arquitectónico, utilizando buenas prácticas, desde luego que ayudará a que éste sea descriptivo, expresivo y legible y con ello no será necesario diseñar diagramas como el de interacción para describir el comportamiento de ciertas clases, objetos, métodos y su funcionalidad.

En otras herramientas de modelado, la notación gráfica que se utiliza para representar un diagrama de secuencia (que se clasifica como parte de *Analysis Model*) y que se usa para explicar un caso de uso, es como el ejemplo que se expone en la figura tomada de (Larman, 2012). Donde los elementos que se muestran corresponden a Actor, Objeto entidad, Objeto borde, Objeto control (ver tabla 6).

Tabla 4. Elementos UML diagrama de secuencia.

Nombre	Elemento UML
Actor	 <b>Customer</b>
Objeto Entidad ( <i>Entity</i> )	 <b>:Order</b>
Objeto Borde ( <i>Boundary</i> )	 <b>:OrderPage</b>
Objeto Control ( <i>Control</i> )	 <b>:OrderCheckout</b>

Un Objeto entidad (*Entity*) representa la información persistente sobre el cual el sistema debe dar seguimiento. Un Objeto borde

(*Boundary*) representa la interacción entre los actores y el sistema. Un Objeto control se encargan de realizar los casos de uso. Estos objetos se visualizan en la figura 20.

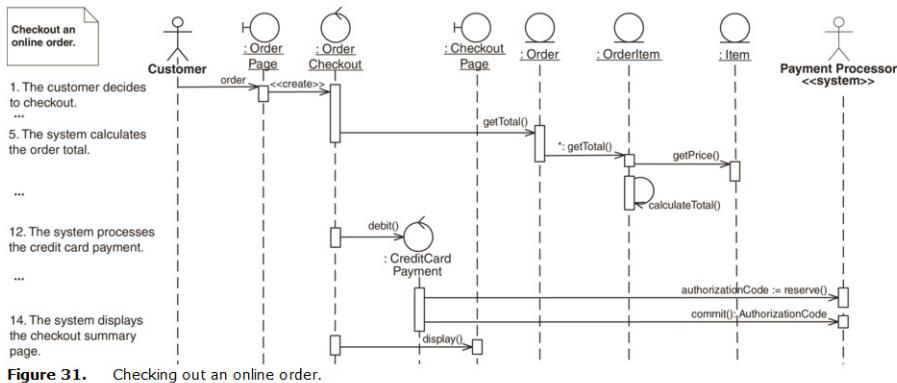


Figura 20. Representación de objeto entidad, borde y control en un diagrama de secuencia. Fuente: Larman (2012)

Como conclusión, se recuerda que un diagrama de secuencia no es usado para suplir la codificación completa que se realiza en su implementación mediante un lenguaje de programación. Los diagramas de secuencia, así como los casos de uso, poseen lógica agnóstica, esto quiere decir que su implementación puede hacerse usando cualquier lenguaje de programación. Finalmente, los diagramas de secuencia sirven para la visualización de muchos objetos/clases y su interacción a la vez en la misma página desde el punto de vista de diseño o de modelamiento.

## 4.2. Diagrama de actividades

¿Ha comprendido hasta el momento el tema de diagramas de secuencia? La idea dentro de un proyecto de software no es que deben realizar todos los diagramas, especialmente de interacción en su totalidad, a menos que sea necesario documentarlos o utilizarlos como instrumentos de explicación del sistema en tiempo real (en

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

reuniones de revisión o socialización, diseño o de desarrollo de los sistemas), pero siempre tomando en cuenta lo que ello implica desde el punto de vista de costo, tiempo y uso de recursos.

A los diagramas de actividad, en el texto básico y bibliografía complementaria se lo define como un gráfico donde se muestran los flujos de mensajes, lógica y acciones que muestran el flujo de control (y opcionalmente datos) donde la ejecución de los pasos puede ser simultánea y secuencial.

Le sugiero revisar en el texto básico los conceptos, elementos, uso, técnicas comunes, sugerencias, consejos y pensar en posibles escenarios de aplicación de los diagramas de actividades. Es necesario enfatizar que entre los elementos que son parte de un diagrama de actividad constan: nodo inicial, bifurcación (fork), acción, decisión, join, merge, nodo final o cierre de la actividad, expuestos en la figura 21.

Una diferencia entre un diagrama de actividad y los diagramas de flujo es que estos últimos muestran la lógica para un solo método (declaraciones condicionales, bucles, entre otros) mientras que los diagramas de actividad muestran el flujo entre los objetos. Como parte del presente curso no se exponen muchos conceptos, gráficas o ejemplos de este tipo de diagramas debido a que en el texto básico y material complementario que se cargará a la plataforma virtual (EVA) los podrá encontrar. Es importante indicar que los diagramas de actividad, no muestran todos los detalles del cálculo de un procedimiento específico que realiza una funcionalidad o el sistema, sino que sirven para mostrar el flujo de actividades que participan en un caso de uso en particular.

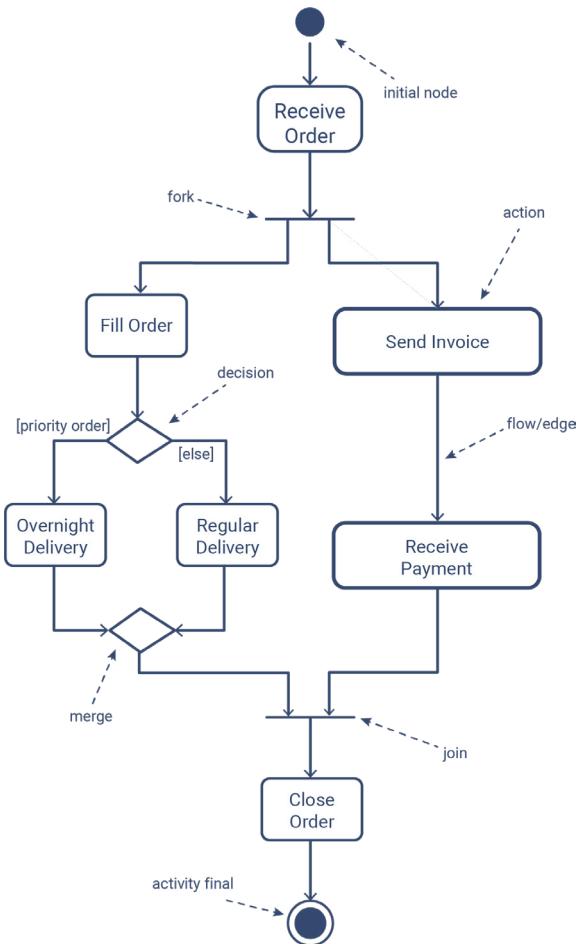


Figura 21. Elementos de un diagrama de actividades. Fuente: Fowler (2004).

Con base en la revisión de la bibliografía básica y complementaria, usted identificará que los diagramas de actividad son considerados como una técnica para describir la lógica o secuencia del procedimiento, el proceso de negocio y el flujo de trabajo, por lo tanto juegan un papel similar a los *diagramas de flujo* o *diagrama de procesos*. La principal diferencia entre ellos es con respecto a la notación del diagrama y que admiten una representación o comportamiento paralelo.

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

Similar a los apartados anteriores, es necesario que utilice y revise bibliografía complementaria escrita en idioma inglés (Ambler, 2005) en el apartado denominado “*UML Activity diagram*”, “*General Guidelines*”, “*Activity Guidelines*”, “*Decision Point and Guard Guidelines*”, “*Parallel Flow Guidelines*”, “*Activity Partition Guidelines*”, “*Action-Object Guidelines*” los cuales le orientarán para documentar y diagramar adecuadamente una actividad.



### Actividades de aprendizaje recomendadas



## Autoevaluación 4

Procedamos a resolver la Autoevaluación y con ello detectar temas a reforzar y mejorar respecto a conceptos de modelado y su representación usando UML. Por favor seleccione para cada pregunta las alternativas que se proponen.

1. ¿Qué permiten describir los diagramas de secuencia?
  - a. El flujo de mensajes, eventos y acciones entre objetos.
  - b. El flujo de sentencias, algoritmos y eventos entre objetos.
  - c. El flujo de activaciones desde la UI (User Interface) hasta la base de datos.
  - d. Todas las anteriores.
  
2. ¿En qué fase se usan los diagramas de secuencia para documentar y comprender el flujo lógico del sistema?
  - a. Análisis y codificación.
  - b. Codificación y pruebas.
  - c. Pruebas e implementación.
  - d. Análisis y diseño.
  
3. ¿Cuáles son los elementos principales de un diagrama de secuencia?
  - a. Participantes, mensajes, ejes (horizontal, vertical).
  - b. Actores, casos, ámbito del sistema.
  - c. Entidades, mensajes, participantes.
  - d. Participantes, casos, ejes.

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

4. Los diagramas de actividad y diagramas de secuencia son parte de la vista:

- a. Estática.
- b. Dinámica.
- c. Física.
- d. De Procesos.

5. ¿Cómo se conoce a los diagramas que muestran el flujo paso a paso de un proceso, flujo de control o flujo de datos?

- a. Diagrama de casos de uso.
- b. Diagrama de secuencia.
- c. Diagrama de actividades.
- d. Diagrama dinámico.

6. ¿Cómo se conoce al concepto que muestra un conjunto de acciones, el flujo entre ellas y los valores producidos o consumidos?

- a. Flujo.
- b. Tarea.
- c. Secuencia.
- d. Actividad.

7. Los diagramas de actividad se emplean para especificar:

- a. Una operación compleja.
- b. Un proceso de negocio (business process) o flujo de trabajo (workflow).
- c. El proceso de negocio asociado a un caso de uso.
- d. Todos los anteriores.

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

8. La sintaxis que se utiliza para representar los objetos dentro de un diagrama de secuencia son:
  - a. <objectname>:<classname>.
  - b. :<classname>.
  - c. object.
  - d. Todas las anteriores.
9. ¿Qué elemento se asocia a la siguiente definición? “se consideran como llamada a métodos desde un objeto a otro”.
  - a. participantes.
  - b. línea de vida.
  - c. activador.
  - d. mensaje.
10. La línea de vida de los objetos es utilizada para:
  - a. Representar la creación de objetos.
  - b. Representar la eliminación o destrucción de objetos.
  - c. Ubicar sobre ella una caja gruesa que simbolice activación.
  - d. Representar la salida o llegada de un mensaje.

[Ir al solucionario](#)

## Ejercicio 10

Con un diagrama de secuencia y actividades represente el siguiente escenario, considere todos los pasos y actividades posibles.

**Escenario:** usted, como estudiante UTPL, hace uso de la plataforma educativa, para ello ingresa al enlace que le permite el acceso a visualizar sus materias que está cursando en el periodo académico actual. Por tanto, va a representar en los diagramas todo el proceso para visualizar un recurso de esta asignatura en la plataforma educativa.

### Bibliografía y lecturas recomendadas

- (Unidad 4.1) [La guía para usar los diagramas de secuencia en StarUML](#) lo puede visualizar en el enlace. No olvide revisar su creación, los elementos del estandar UML y su representación a través de la herramienta de modelado StarUML.
- (Unidad 4.1) En el texto básico revise el capítulo 19 que expone los conceptos relacionados a **Diagramas de Interacción**.
- (Unidad 4.1) [Otras definiciones y representaciones de los diagramas de actividad](#) lo puede encontrar en el recurso educativo abierto (*Ingeniería de Software*), para ello revise el Tema 10. Comportamiento del sistema. Páginas 44 a 78.
- (Unidad 4.2) Para conocer su forma de representación y usos le invito a revisar en el texto básico el capítulo 20 que hace referencia a los conceptos de **Diagramas de actividades**.
- (Unidad 4.2) [El uso de diagramas de secuencia utilizando StarUML](#) lo puede encontrar en el enlace que se propone.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

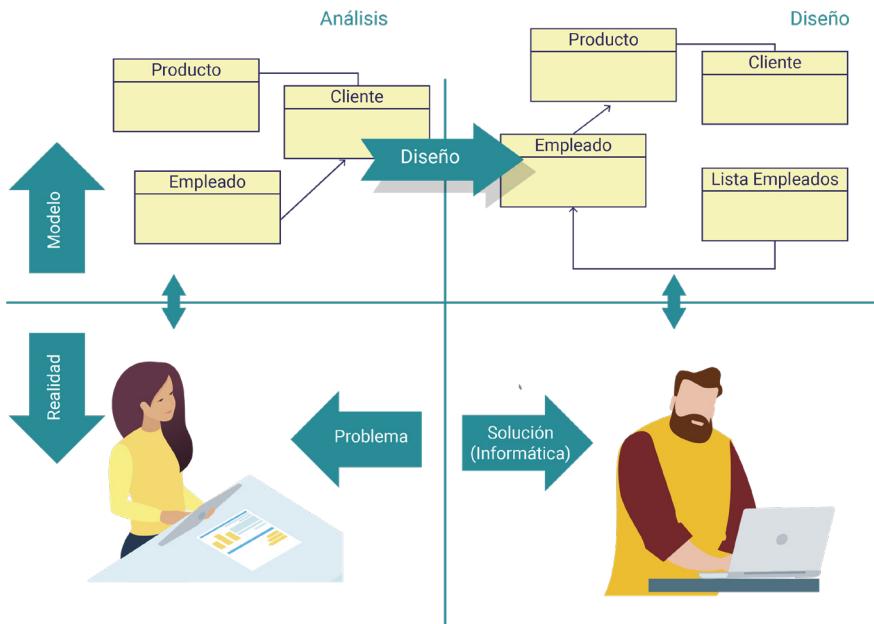


## Actividades finales del bimestre



### Semana 8

Hemos cubierto el estudio de las unidades planificadas para este Primer Bimestre. ¿Cómo ha resultado el estudio de cada una de las unidades hasta el momento?, me imagino un poco cansado, estresante y con la dificultad normal que exige una Ingeniería. Pero bueno, conforme vaya asociando los conceptos teóricos con la práctica usando herramientas de modelado, su uso resultará comprensible ya que la idea de utilizar un vocabulario común, estereotipos, nomenclatura definida y estandarizada como UML, para representar formalmente una solución a un problema, en este caso de software. Recuerde que los estándares están definidos y es responsabilidad de los Ingenieros de Software, quienes con un rol definido dentro de un proyecto de software, son los encargados del análisis, diseño y construcción de software.



Fuente: Tomado de: [enlace web](#)

Le invito a que desarrolle las actividades que se planifican durante este Primer Bimestre, las cuales son formativas, sumativas (calificadas) a lo largo de su preparación y auto-aprendizaje. Dichas actividades planificadas son Foro, Chat y Cuestionarios que se han configurado en la plataforma académica virtual. Asimismo, por favor en la medida de lo posible, realice los ejercicios y cuestionarios propuestos como parte del presente curso, que son sumativos y sirven de preparación para la evaluación final.

Si aún existen dudas, por favor revise el texto básico o contacte con su docente/tutor por los distintos medios como correo electrónico, plataforma académica virtual (EVA) o línea telefónica.

## Actividad 1

Revise y complete los ejercicios que se proponen en los siguientes enlace:

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

## Ejemplo de modelado en UML.

### UML ejemplo sencillo sobre Modelado de un Proyecto

De ser posible vuelva a representar los diagramas y complete los que se solicitan.

Hasta el momento, y como parte de la tarea que forma parte del praxis profesional (cátedra integradora/proceso práctico de aprendizaje), es necesario que haya entendido el problema, identificado los requisitos funcionales, no funcionales y atributos de calidad. Desde el punto de vista de representación gráfica con UML que haya documentado de forma correcta, usando una herramienta CASE los diagramas, elementos y nomenclatura asociada a los diagramas y especificación de casos de uso, diagramas de actividades, secuencia y estados y haya definido las entidades que formarán parte del diagrama de clases, paquetes, componentes, despliegue y arquitectura candidata.

Con estas indicaciones finalizamos este Primer Bimestre, espero que la resolución de la evaluación presencial donde se evalúan aspectos teóricos y prácticos sean exitosos. Por ahora, tome un descanso y en poco tiempo le animo a que con la misma dedicación y empeño iniciemos el estudio de las unidades que corresponden al Segundo Bimestre. ¡Buena suerte!

### Actividad 2

- Desarrolle el pequeño banco de preguntas que se cargará en la plataforma educativa para su auto-estudio previo a la evaluación presencial.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



## Segundo bimestre

### Resultado de aprendizaje 3

Utiliza UML como lenguaje de construcción de modelos para modelar componentes de un sistema de información.

## Contenidos, recursos y actividades de aprendizaje

Al iniciar este segundo bimestre, es necesario tener claros los conceptos, elementos y estereotipos UML que le permiten implementar diagramas (de caso de uso, secuencia, actividades) utilizando técnicas de análisis, una metodología y siguiendo un proceso de desarrollo de software acorde a un contexto dado. Ampliemos y reforcemos la parte teórica y práctica que le ayudarán en el tema de diagramas. Si aún existen dudas, por favor revise el texto básico o contacte con su docente/tutor por los distintos medios, como correo electrónico, plataforma virtual (EVA) o línea telefónica.



### Semana 9



## Unidad 5. Relaciones y mecanismos comunes

### 5.1. Relaciones

El tema de relaciones y sus distintas formas de uso es primordial para todos los diagramas, especialmente cuando se requiere trabajar con clases. Por ello, es necesario iniciar recordando cómo identificar clases a partir de un problema, ya que esto lo explicaremos a detalle en la Unidad 6. Por ejemplo, una clase es un sustantivo y debe analizar si a partir de dicha clase se pueden crear al menos un objeto o un conjunto de ellos. Ahora, se sugiere revisar a detalle los procedimientos, tips y mejores prácticas para identificar a más de clases, atributos y métodos con lo que empezaremos el estudio para posteriormente representarlos en un diagrama de clases.

Está claro que un actor, un caso de uso, un objeto, una entidad por si sola o sin relaciones, no dice nada, especialmente cuando queremos diseñar un diagrama. Es por ello que se deben definir y representar las relaciones que pueden existir haciendo uso de estereotipos, nomenclatura y que representados en UML que puedan solucionar un problema que posteriormente debe ser implementada con cualquier lenguaje de programación orientado a objetos. Para este fin, aparte del texto básico, se sugiere revisar la bibliografía complementaria como (Rumbaugh et al., 2004), especialmente en el capítulo denominado “Static View” con lo que damos inicio al estudio de esta unidad.

Para el tema de relaciones utilizando UML hay que tener en cuenta que:

- Existen tres tipos de relaciones que se consideran importantes: dependencia, generalización y asociación (fuerte-composición y débil-agregación).
- Cuando se trabaja con relaciones (asociación especialmente) se debe considerar aspectos como multiplicidad, intercambio de mensajes y nomenclatura, que al momento de codificar utilizando un lenguaje de programación tiene su forma de implementación.
- Cuando se representa una relación con UML, se debe tener cuidado con la forma y dirección de las flechas que relacionan casos de uso, clases, paquetes, interfaces, componentes, entre otros.

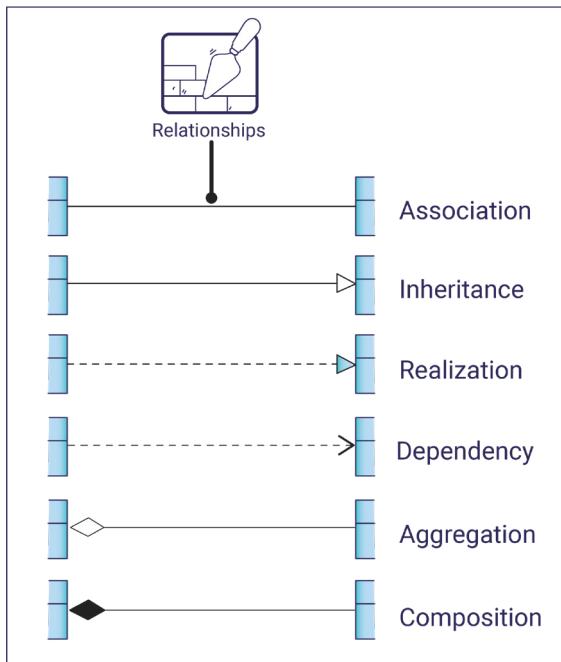


Figura 22. Simbología utilizada para representar relaciones

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Espero que la revisión de los conceptos y la explicación gráfica expuesta en el texto básico sean apoyo suficiente para comprender la implementación de diagramas con UML. Es necesario tener en cuenta que para un modelado de clases, que es lo que estudiaremos en la siguiente unidad, se tenga en cuenta que tres relaciones son consideradas principales, entre ellas constan:

1. **Dependencia:** se considera como una relación que indica que una clase utiliza datos/información de otra clase u objeto en una sola dirección (la que indica la flecha) pero no de forma inversa.
2. **Generalización:** es una relación en donde intervienen una superclase, clase abstracta o clase padre que tienen propiedades comunes con clases más específicas consideradas subclases o clases hijas.
3. **Asociación:** la asociación representa instancia de variables dirigidas desde una clase origen hacia la clase destino. Aspectos como el nombre de la propiedad junto con la multiplicidad gráficamente se ubican en el final objetivo de la asociación. Una asociación define un tipo de enlace y las características de sus enlaces, y un enlace es una instancia de una asociación. Como parte de la asociación se tiene Agregación y Composición.
  - a. **Agregación:** la agregación es una relación de parte entera entre un agregado, el todo y sus partes. Esta relación se conoce a menudo como una relación tiene-un, porque el conjunto tiene sus partes
  - b. **Composición:** la composición, también conocida como agregación compuesta, es una relación de toda la parte entre un compuesto (el todo) y sus partes, en el que las partes deben pertenecer sólo a un todo y el conjunto es

responsable de crear y destruir sus partes cuando se crea o se destruye. Esta relación se conoce a menudo como una relación contiene-uno, porque el todo contiene sus partes.

Como se indica en el texto básico y como se comentó anteriormente, es muy importante cuando se representa un diagrama con UML la dirección de las flechas, que permiten relacionar las clases, casos de uso, actividades. Sin embargo, hay que tener en cuenta algunos otros aspectos, especialmente en la terminación o final de asociación.

- **Final de Asociación:** es un punto final de la línea dibujada sobre la flecha que representa una asociación y que conecta a una clase. Un final de asociación puede incluir cualquiera de los siguientes elementos para expresar más detalles sobre cómo la clase se relaciona con otra clase o clases de la asociación: nombre de rol, flecha de navegación, especificación de multiplicidad, símbolo de agregación o composición, Índice.
  - a. **Nombre de rol:** es opcional e indica el rol que una clase juega en relación con las otras clases de la asociación. Visualmente, el nombre de rol se muestra cerca del final de una asociación para una clase.
  - b. **Flecha de navegación:** la navegación es opcional e indica si una clase puede ser referenciada de las otras clases en una asociación. La navegación se muestra como una flecha adjunta a una asociación que apunta hacia la clase en cuestión. Si no hay flechas, se supone que las asociaciones son navegables en todas las direcciones, y todas las clases involucradas en la asociación pueden referirse entre sí.

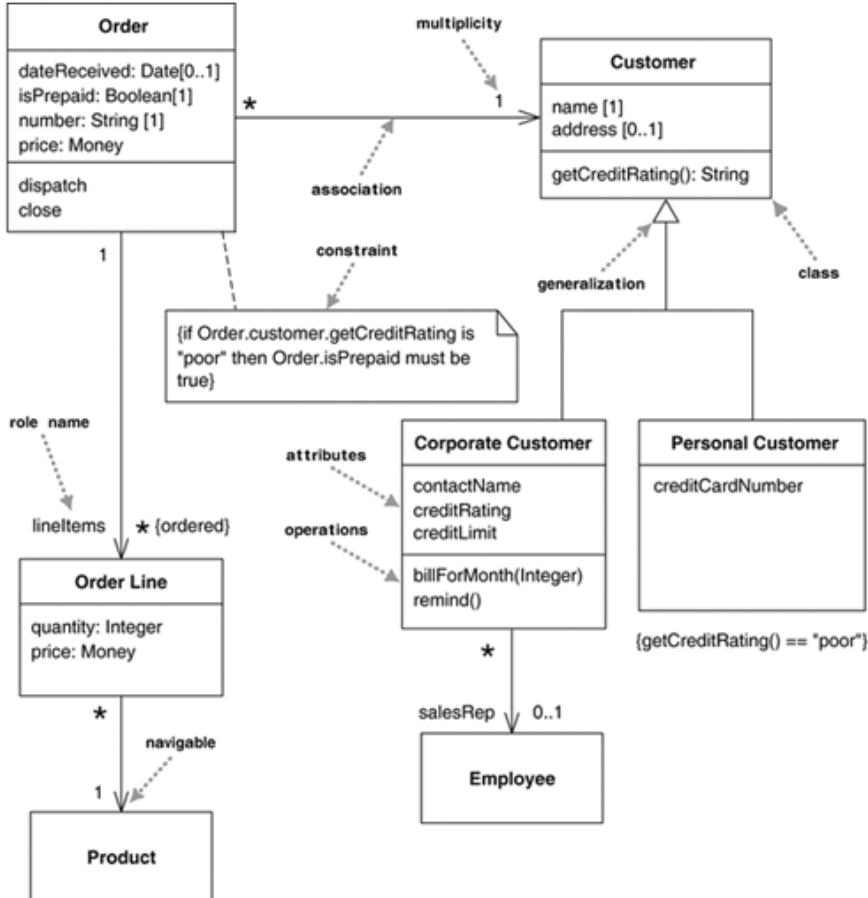


Figura 23. Elementos de diagrama de clases y relaciones. Fuente: Fowler (2004).



Figura 24. Ejemplo Asociación de clases.

En la figura 24 se indica que un Team Developer está formado por 20 Desarrolladores de Software. El nombre cerca de la punta de flecha es el nombre de la variable de instancia. El número cerca de la punta de flecha nos dice cuántas referencias se tienen hacia la clases. En el momento que requiera implementar dichas clases y relaciones a nivel de codificación java por ejemplo tendríamos:

```

Public class TeamDeveloper{
private Desarrollador arrDev[20];
}
  
```

Si se modifica la representación UML anterior por la de la figura 25 tendríamos.



Figura 25. Ejemplo Asociación Clases.

Por lo tanto, a nivel de código java se tendrá:

```

Public class TeamDeveloper{
private Vector arrDev;
}
  
```

Respecto a la codificación java de los ejemplos anteriores, su representación se realiza de dicha forma porque no se tiene un tamaño definido del arreglo (Vector arrDev). Recuerde asociar los conceptos estudiados con su representación en UML y revisar el texto básico y bibliografía complementaria, para su posterior implementación con cualquier lenguaje de programación, que es el punto final donde las clases se utilizan.

NOTA: se realiza una breve explicación de las relaciones entre clases y su codificación en java por si lo necesitase en algún momento.

## 5.2. Mecanismos comunes

Por convención, en diagramas UML donde intervengan relaciones o asociaciones tales como casos de uso con actores, clases con subclases, actividades con tareas, procesos, paquetes, componentes, nodos, entre otros, las líneas o flechas que unen dos o mas elementos deben estar alineados vertical u horizontalmente. No se permite tener una representación que desde el punto de vista visual se vea distorsionado, incomprendible, que se cruzan las flechas sobre los elementos evitando su visualización correcta, entre otros. Además, con respecto al tamaño de los elementos y

su alineación (horizontal y vertical), es fundamental que los ubique todos acorde sus necesidades similar a cuando diseña una GUI (Graphical User Interface) en cualquier lenguaje de programación. Revise los siguientes ejemplos de diagramas expuestos gráficamente como incorrectos y correctos.

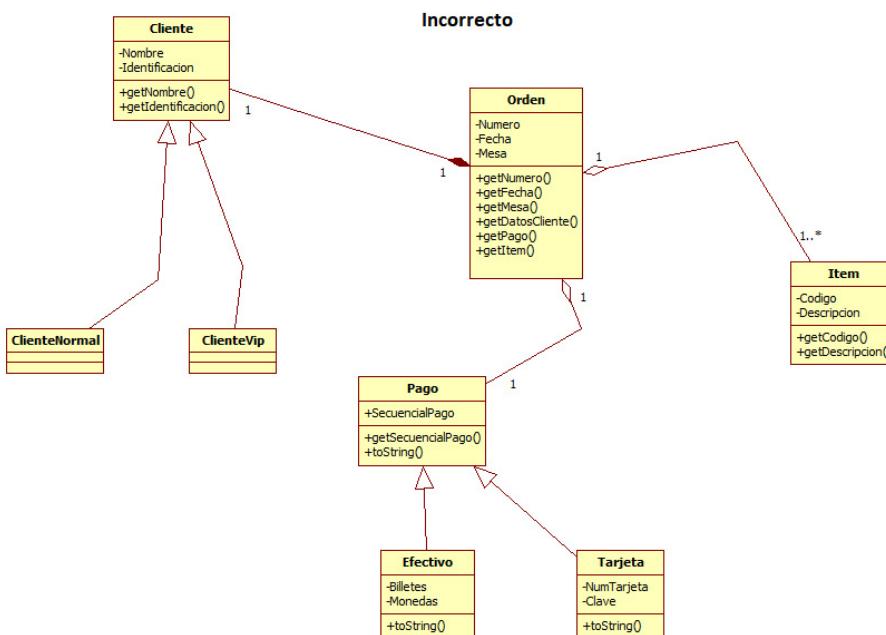


Figura 26. Incorrecta representación y relación entre clases con UML.

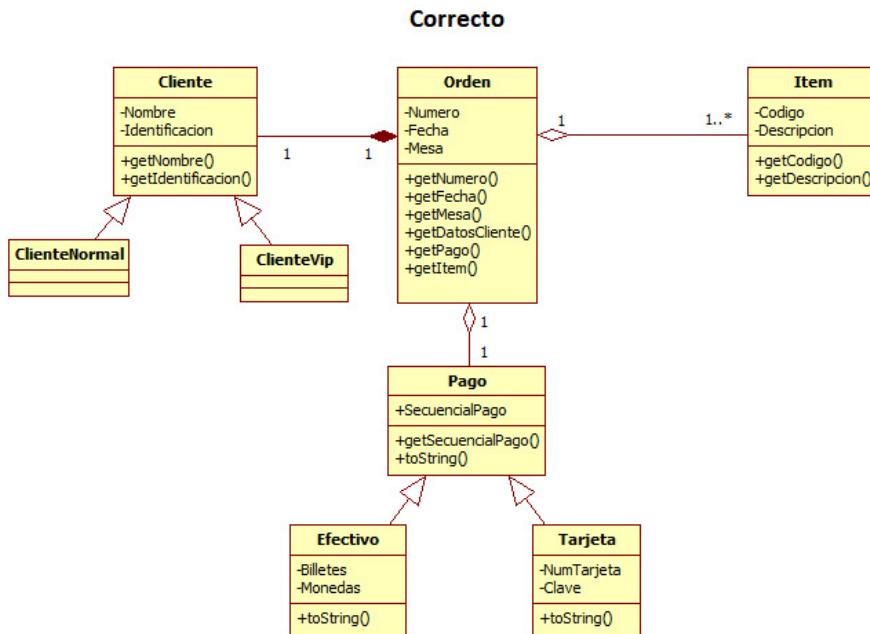


Figura 27. Correcta representación y relación entre clases con UML.

Para lograr que dos elementos UML se asocien con línea rectilíneas utilice la propiedad de StarUML como la que se expone a continuación. Donde primeramente debe marcar la asociación luego va a *Format, Line Style* y finalmente selecciona *Rectlinear* u *Oblique*.

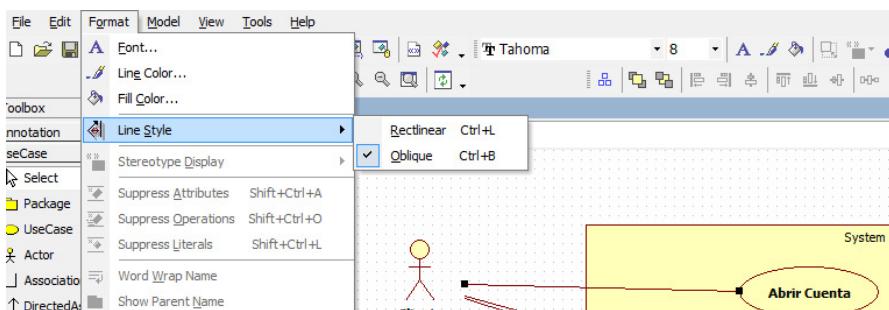


Figura 28. Interfaz Gráfica de StarUML como herramienta de modelado.

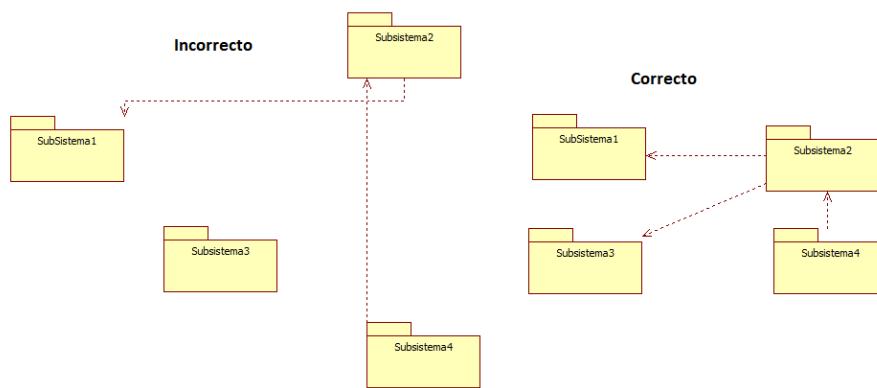


Figura 29. Correcta e incorrecta representación de un diagrama de paquetes.

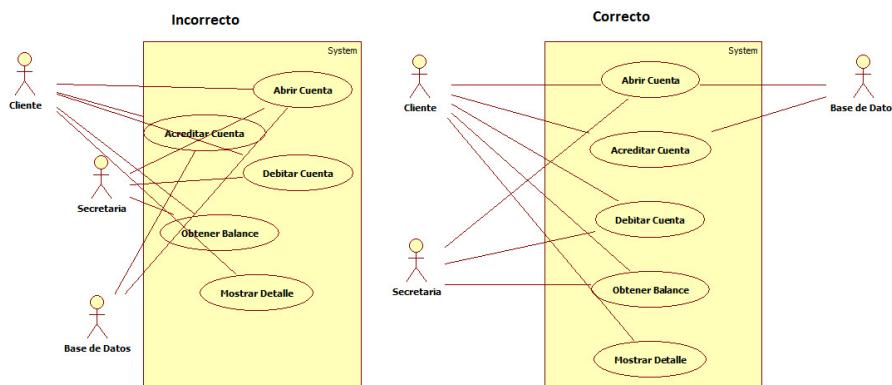


Figura 30. Correcta e incorrecta representación de los diagramas de caso de uso.

¿Está claro el tema referente a relaciones, su definición, simbología, tips y sugerencias que se pueden usar en un diagrama desde el punto de vista de análisis?. Recuerde utilizar y revisar bibliografía complementaria como (Ambler, 2005) en el apartado denominado “Relationship Guidelines” donde encontrará, escrito en idioma inglés, algunas sugerencias referentes al tema de relaciones entre clases.

En la bibliografía encontrará, por ejemplo, que se debe modelar las relaciones horizontalmente, modelar las relaciones entre dos clases

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

solamente cuando exista colaboración entre ellas, indicar siempre la multiplicidad existente entre las clases ya que siempre que exista una relación fuerte o débil existirá una multiplicidad, entre otras.

Hasta el momento, debe tener en cuenta que, con base en las necesidades dadas por el cliente, posterior análisis y documentación de los requisitos, se puede identificar las clases, subclases, relaciones y otras propiedades que permitirán diseñar un diagrama de clases acorde al contexto. Este diseño se lo puede representar de forma textual o gráfica utilizando conceptos y elementos de UML como los estudiados previamente. Pero ¿cuándo o para qué diagramar?, aunque ya lo conoce, revisemos los siguientes aspectos sugeridos en bibliografía complementaria.

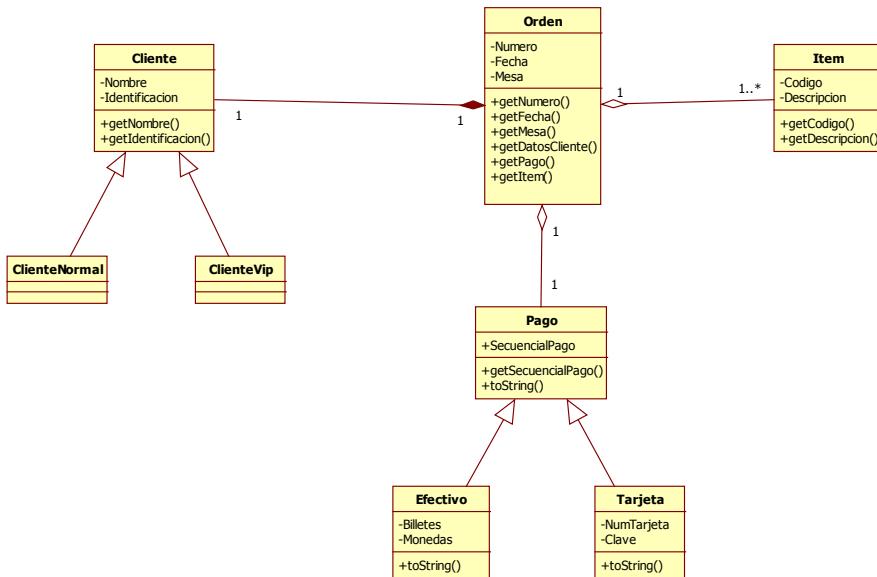
1. Utilizar diagramas cuando varias personas necesitan entender la estructura de una parte particular del diseño de la solución software.
2. Se realiza un diagrama cuando dos o más personas no están de acuerdo en cómo se debe diseñar un elemento en particular y se requiera consenso del equipo.
3. Los diagramas se utilizan cuando se requiera representar una idea del diseño antes que su implementación a través de código.
4. Dibujar diagramas cuando se requiera explicar la estructura de alguna parte del código a otra persona.
5. Dibujar diagramas los necesarios como una buena práctica de documentación del proyecto haya o no solicitado el cliente.



## Actividades de aprendizaje recomendadas

### Ejercicio 10

En base al siguiente diagrama representado en StarUML, por favor modifíquelo (desde el punto de vista visual), ajústelo, complételo para que desde el punto de vista de análisis se asocie y cumpla con los conceptos y especificaciones estudiadas hasta ahora.



Luego de realizar el ejercicio, ¿solamente con el diagrama UML se puede entender de qué trata el problema? Una forma de explicar las clases que forman parte del ejercicio podría ser:

- Un **Cliente** realiza un consumo o compra de un producto en una cafetería, restaurante u otro, la **Orden** de compra se componen de varios **Item** de productos. La **Orden** debe ser pagada y para ello se realiza un **Pago** que puede ser en

**Efectivo o con Tarjeta** de crédito. Si es en **Efectivo** el pago se puede realizar con billetes o monedas.

### Ejercicio 11

11.1. ¿Cuál es el elemento UML que se utiliza para representar: una Clase (Diagrama de clases), Interface (Diagrama de Clases), Componente (Diagrama de Componentes), Nodo (Diagrama de Despliegue), Estado (Diagrama de Estados), Nota (Diagrama de casos de Uso)? Utilice StarUML como herramienta de modelado para representar gráficamente dichos elementos.

#### Elemento UML

Clase	
Interface	
Componente	
Nodo	
Estado	
Nota	

11.2. ¿Cuál es el símbolo de relación (*tipo de línea y flecha*) que se utiliza para representar el concepto de una relación entre clases? Utilice StarUML como herramienta de modelado para graficar dichos conceptos.

#### StarUML como herramienta de modelado

Concepto	Representación Gráfica
Dependencia	
Asociación	
Generalización/Herencia	
Realización	
Agregación	
Composición	

- 11.3. Utilice las técnicas de modelado para identificar clases, atributos, métodos, relaciones (si fuese posible) y representarlo con UML para el siguiente planteamiento.

La Biblioteca UTPL requiere implementar una solución para realizar actividades de registro, y visualización de material bibliográfico y préstamos de dicho material a personas. Conociendo que como Biblioteca se tiene que libro, revista y tesis son los tipos de material bibliográfico. Los atributos de material bibliográfico que se registrarán en el sistema son código, autor, título, año, estatus (se refiere a Activo o Inactivo). Libro tiene como atributo editorial. Revista tiene como atributo número, Tesis tiene como atributo código. Alumno y Docente son los tipos de Persona que pueden solicitar el préstamo de material bibliográfico. El sistema almacenará de una Persona su identificación, nombres, correo, teléfono. Del alumno se registrará el id de matrícula y de docente se registrará el código docente.

### Bibliografía y lecturas recomendadas

- (Unidad 5.1) Retome la lectura en el texto básico y revise el capítulo 5 que expone los **conceptos de Relaciones, términos, conceptos y técnicas de modelado que apoyarán en la representación textual y gráficas en modelado**.
- (Unidad 5.2) Para complementar conceptos, nomenclatura, simbología y técnicas de modelado para utilizarlo con StarUML, se sugiere revisar el texto básico en capítulo **6 Mecanismos Comunes**, que incluye aspectos como nomenclatura, simbología y técnicas de modelado.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

## Resultado de aprendizaje 5

Crea modelos que apoyen el proceso de análisis de sistemas.

### Contenidos, recursos y actividades de aprendizaje



#### Semana 10



## Unidad 6. Diagrama de clases

### 6.1. Diagrama de clase

El **diagrama de clases** pertenece al área principal estructural y corresponde a la vista estática. Por ello, es necesario tener en cuenta los siguientes puntos cuando se desea elaborar un diagrama de clases:

- Como parte del análisis, para un diagrama de clases se debe primeramente identificar las clases o entidades que formarán parte de la solución, analizando las cosas que son importantes, utilizando conceptos como abstracción y sabiendo que una clase representa a un conjunto de objetos

donde cada uno de ellos tiene propiedades que posteriormente se pueden diseñar, modelar e implementar a través de código.

- En el proceso de identificación de clases, objetos, atributos, métodos y relaciones, debe existir un refinamiento que se lo realiza de forma iterativa, con el fin de mejorar la definición de cada concepto que serán implementadas en el software para cumplir con los requisitos del cliente y con atributos de calidad.
- Para realizar un diagrama de clases, se necesita tener conocimientos básicos del paradigma de programación orientado a objetos, por ello se recomienda haber estudiado o volver a revisar los conceptos tales como: clases, atributos, métodos, relaciones, objetos, visibilidad los cuales se utilizarán para dar inicio con el estudio de diagramas estructurales.
- Para representar un diagrama de clases, se lo puede hacer desde el punto de vista de análisis y desde el punto de vista de diseño. En el primero caso, aspectos como tipos de dato, atributos y métodos no es obligatorio, en el segundo caso se necesita ubicar a detalle los atributos y métodos para cada clase, ya que será la base para la codificación en cualquier lenguaje de programación.

Es conocido que UML provee tres compartimentos dentro de un rectángulo para representar una clase. El primero contiene el nombre de la clase, el segundo donde se ubican los atributos y el tercero para los métodos u operaciones. El nombre de las clases por defecto en StarUML se muestra marcadas con “Negrita” como lo visualizará en los ejemplos de la guía y el texto básico.

Es necesario que para la implementación de los conceptos teóricos utilice la herramienta CASE de modelado sugerida. Con dicha herramienta, un diagrama de clases UML es representado con los siguientes elementos (ver figura 31). Una clase se muestra como un

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

rectángulo de contorno sólido con tres compartimentos estándar separados por líneas horizontales. El compartimiento superior muestra el nombre de la clase, el segundo compartimento opcional muestra una lista de atributos y el tercer compartimiento opcional muestra una lista de operaciones. En el caso de que la clase Persona sea abstracta su visualización es usando formato Cursiva (Italica) *Persona*.

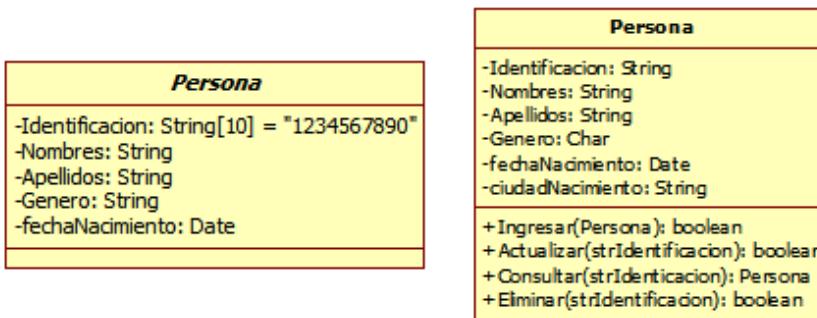


Figura 31. Representación de clase en StarUML

La representación de los atributos de la clase Persona siguen la siguiente Simbología:

StarUML como herramienta de modelado

Simbología	Nombre	Ejemplo
visibility name: type [count] = default value	Definición de atributos de clase	-Identificacion: String[10] = "1234567890", donde "-" corresponde a visibilidad, Identificacion es el nombre del atributo, String es el tipo de dato, [10] es el tamaño definido para la variable y "1234567890" es el valor por defecto.
visibility name (parameters):return type	Definición de métodos de clase	+ Actualizar (strIdentificacion):boolean, donde "+" corresponde a la visibilidad, Actualizar es el nombre del método, strIdentificación son los parámetros que se envía al método y boolean es el tipo de dato que retorna el método.

## 6.2. Relaciones entre clases

Para representar las clases y sus relaciones, es necesario que se realice un análisis detallado del problema o necesidades de los clientes. En el análisis se debe en primer lugar identificar los requisitos los mismos que pueden clasificarse como funcionales y no funcionales. Con base en los requisitos funcionales, se pueden identificar los subsistemas, componentes y funcionalidades que puede formar parte de la solución software. Posterior a ello, se debe proceder a identificar entidades (clases) realizando una abstracción, utilizando técnicas y estrategias como que una clase sea la contenedora o la creadora de nuevos objetos de otras clases.

Recordar que el identificar clases es un proceso iterativo, que debe realizarlo algunas veces hasta refinar y decidir cuáles serán las clases y las relaciones entre ellas. En el caso de las relaciones, es fundamental que conozca los conceptos y representación usando UML referentes a generalización y asociación (expuestos anteriormente), los cuales serán de ayuda para representar la relación entre dos o más clases y claro sin olvidar el sentido o dirección que deben tener las flechas y los estereotipos a utilizar. A continuación, se exponen dichos conceptos.

- **Generalización:** se explica como una relación de herencia. Herencia entre clases y a través de implementación de interface.
- **Asociación:** se considera como una relación de uso. Dentro de la asociación existe conceptos como dependencia, agregación, composición.

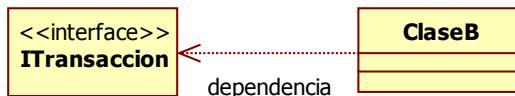
- **Agregación:** Significa “*es parte de*”, simbolizado por un diamante blanco.



- **Composición:** Significa “*está compuesto de*”, simbolizado por un diamante negro, las partes viven y mueren conjuntamente.



- **Dependencia:** Significa “*usa temporalmente*”, simbolizado por línea punteada, a menudo es un detalle de implementación, no una parte intrínseca del estado de ese objeto.



Retomando el ejemplo de la Institución Educativa XYZ del Ejercicio 8, la representación de clases utilizando StarUML sería como la que se muestra en la figura 32.

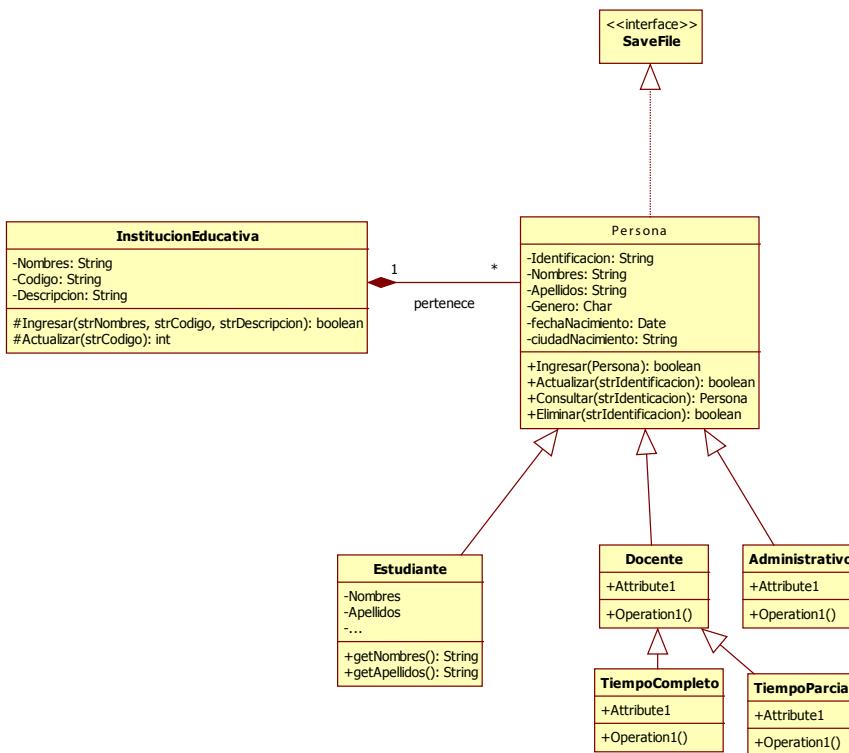


Figura 32. Representación de clases y sus relaciones en StarUML

Como se puede observar en la Figura 32, en el método Ingresar de la clase InstitucionEducativa se envia como parámetros todos los atributos de la clase separados por coma, esto es una forma de representar. A diferencia del método Ingresar de la clase Persona, donde se envía como Parámetro Persona, lo cual implica que sería un objeto de dicha clase que contiene todos los atributos de la clase en mención. Por cada atributo se expone el tipo de dato soportado (integer, String, Char, etc.) así como su visibilidad (público, privado, protegido) y si nos fijamos en la parte de los métodos también se usa visibilidad, así tambien el o los atributo(s) que se pasaría al método y el tipo de dato que se espera como retorno.

En caso de que el tipo de dato de retorno del método sea boolean, esto indica que el método retornará true o false y en caso de que

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

el tipo de dato sea entero se espera un valor de retorno que puede ser 1 o 0 u otro número que luego, al momento de interpretarlo con lógica a nivel de programación, permitirá alertar la acción realizada a través de un mensaje que el usuario visualizará en su interfáz gráfica.

Asimismo, se puede observar que las clases Estudiante, Docente y Administrativo utilizan la simbología de herencia donde las flechas apuntan hacia arriba a la clase padre, y para efectos de explicación no se han ubicado ni los atributos y métodos que corresponden.

Como parte de la generalización se usa una interface denominada SaveFile donde para relacionar estas clases se usa una línea punteada o el simbolo de realización hacia la clase de la cual se implementará una interface, por lo tanto la jerarquía se dibujan de arriba hacia abajo. Recuerde los términos interface e interfaz son completamente diferentes le invito a revisarlos en el texto básico o bibliografía complementaria.

Como parte de la asociación, existe el concepto de multiplicidad, nombre y navegabilidad, los cuales se explican a continuación.

- **Multiplicidad:** representado al inicio o final de la asociación donde por ejemplo:

Multiplicidad	Descripción
*	Significa 0, 1 o muchos
1	Significa exactamente 1
2..4	Significa inclusive entre 2 y 4
1..*	Significa 1 o muchos

- **Nombre:** el nombre que se ubica sobre la línea de asociación, se usa especialmente en asociaciones, en el ejemplo “pertenece” es un nombre. Algo fundamental es que el nombre permitirá leer de la clase A a B y viceversa y el significado

debe ser el mismo. Muchas personas pertenecen a una InstitucionEducativa y a una InstitucionEducativa se pertenece muchas Personas. En el caso de que no se pueda leer de forma adecuada, se recomienda ubicar el rol de la clase A o B con respecto al diagrama, esto se ubica al nivel donde se ubica la multiplicidad.

- **Navegabilidad:** relacionada con la dirección de las flechas, algo que es muy importante cuando se representa asociación, dependencia, realización o generalización entre clases.



### Actividades de aprendizaje recomendada

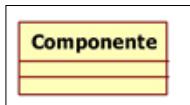


## Autoevaluación 5

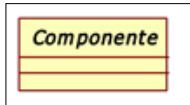
Es hora de evaluar los conocimientos adquiridos en la presente unidad, para ello por favor realice la siguiente Autoevaluación que le permitirá asociar una o varias respuestas a una pregunta propuesta.

1. ¿Cuál de las siguientes alternativas en UML representa a una clase abstracta?

- a. Alternativa 1



- b. Alternativa 2



- c. Alternativa 3



- d. Todas las anteriores

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

2. ¿Cuáles de las siguientes representaciones UML corresponden a asociación, agregación y composición? Seleccione todas las opciones que aplican.

a. Alternativa 1 

b. Alternativa 2 

c. Alternativa 3 

d. Alternativa 4 

3. ¿Cuál de las siguientes representaciones UML corresponden a generalización, dependencia, realización? Seleccione la alternativa y ubique el texto según corresponda.

a. Alternativa 1 

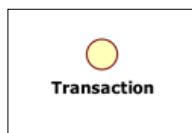
b. Alternativa 2 

c. Alternativa 3 

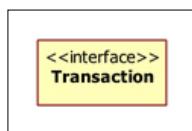
d. Alternativa 4

4. ¿Qué elemento UML representa a una Interface?

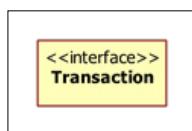
a. Alternativa 1



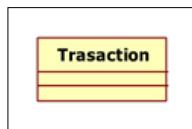
b. Alternativa 2



c. Alternativa 3



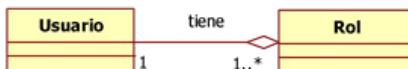
d. Alternativa 4



5. ¿Qué elementos UML se muestran en la siguiente figura?, Seleccione todas las que aplica.

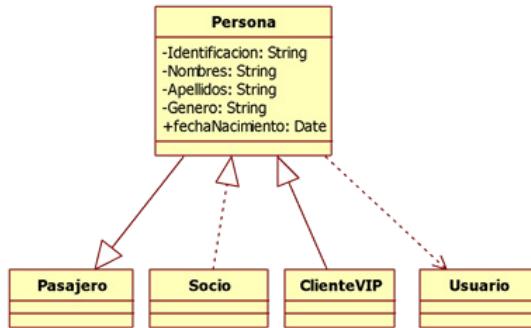


- a. Multiplicidad
  - b. Nombre de rol
  - c. Asociación
  - d. Generalización
6. ¿Cuál es la mejor opción de lectura en la siguiente representación?



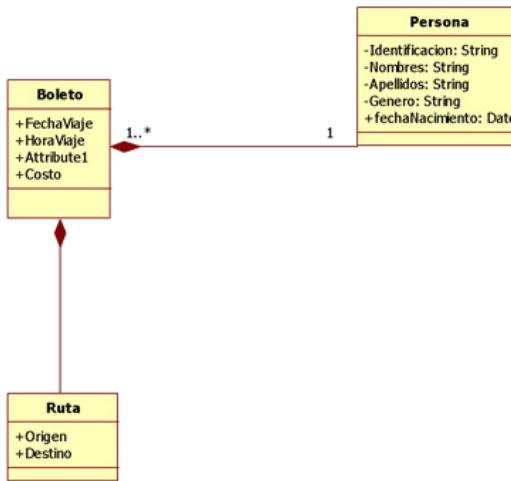
- a. Un rol tiene un Usuario
- b. Un usuario tiene uno o muchos roles
- c. Uno o muchos roles tienen un usuario
- d. Un usuario tiene un rol

7. Si deseamos representar gráficamente con UML que una subclase hereda de Persona, ¿cuál sería dicha clase?



- a. Pasajero
- b. Socio
- c. ClienteVIP
- d. Usuario

8. En el siguiente ejemplo de UML, ¿qué se desea representar?



- a. Las subclases de boleto son Persona y Ruta
- b. El boleto se asocia con Persona y se deriva de Ruta
- c. El boleto se compone de Persona y Ruta
- d. El boleto depende de Persona y Ruta

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

9. ¿Cuál es la mejor opción con respecto a una interface?
  - a. Los elementos de una interface son nombre, atributos, métodos
  - b. Los elementos de una interface son nombre, atributos
  - c. Los elementos de una interface son nombre, métodos.
  - d. Los métodos en una interface no se implementan en la interface sino en la clase.
10. Cuando se tiene un modelo UML y se requiere generar el código a partir de dicho modelo estamos hablando de usar
  - a. Ingeniería Inversa
  - b. Arquitectura Inversa
  - c. Top Down
  - d. Ingeniería Directa

[Ir al solucionario](#)

## Lecturas recomendadas

- (Unidad 6.1) Para profundizar en aspectos del diagrama de clases, su representación en UML, nomenclatura utilizada, le invito a que en el texto básico revise en el capítulo 8 los conceptos referentes a **Diagramas de clases y sus sub unidades**.
- (Unidad 6.2) [Revise por favor el recurso educativo abierto \(Ingeniería de Software\)](#) en el Tema 8. Estructura del sistema, en el se exponen aspectos de clases y diagrama de clases. Revise el enlace que hace referencia a [diagrama de clases y su representación a través de StarUML](#)

## Ejercicio 12

“La Institución Educativa XYZ requiere desarrollar una solución software que permita automatizar el proceso de Gestión Académica. Una de sus funcionalidades corresponde al Registro de Estudiantes, para ello se requiere que a través de una interfaz (formulario) se solicite al cliente datos como identificación, nombres, apellidos, género, fecha de nacimiento, ciudad de nacimiento y que estos datos se almacenen en una base de datos.....”(continúa).

Si bien en el enunciado del Ejercicio 8 resulta sencillo identificar clases que podrían participar en la solución final, también se puede observar en la descripción de la necesidad dada por el cliente que no se mencionan aspectos tales como visibilidad, relaciones u otros, éstos por lo general son aspectos que un analista debe llevar a cabo. Veamos cómo podríamos empezar a identificar y extraer las clases y demás conceptos para un diagrama de clases.

Las posibles clases derivadas del Ejercicio serían:

**InstitucionEducativa y Persona.**

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

- Una definición para la clase InstitucionEducativa sería:

**Clase:** InstitucionEducativa

**Atributos:** Nombre, Descripcion

**Métodos:** Ingresar, Actualizar

- Una definición para la clase Persona sería:

**Clase:** Persona

**Atributos:** Identificacion, Nombres, Apellidos, Genero, fechaNacimiento, ciudadNacimiento

**Métodos:** Ingresar, Actualizar, Consultar, Eliminar

Los objetos que se derivarían de la clase InstitucionEducativa serían por ejemplo Universidad Técnica Particular de Loja, Colegio Eugenio Espejo, Colegio Borja, Universidad de Guayaquil, etc., y los objetos que se derivarían de la clase Persona serían Juan, Pedro, Ana. Recuerde, esto lo revisó y analizó en la unidad anterior en el primer bimestre, ahora lo que se pide como parte del ejercicio es que represente las clases usando nomenclatura UML y definiendo la visibilidad y propiedad para cada atributo y método.

## Resultado de aprendizaje de 4

Diseña representaciones para facilitar la comunicación, discusión, exploración y validación de la especificación y diseño de un sistema.

### Contenidos, recursos y actividades de aprendizaje



#### Semana 11

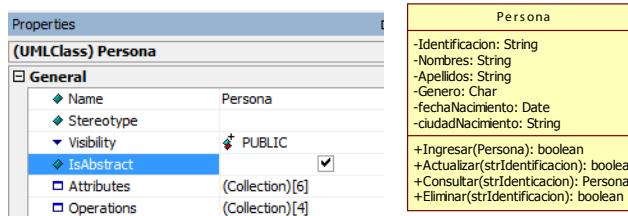
### 6.3. Consideraciones al momento de diseñar diagrama de clases

El diagrama de clases, como su nombre lo indica, muestra las clases que forman parte de un sistema, sus operaciones, atributos y relaciones. Estas se utilizan para:

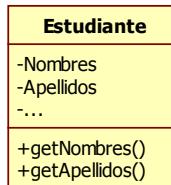
- Explorar conceptos de dominio en forma de un modelo de dominio.
- Analizar requisitos en forma de un modelo de análisis conceptual.
- Representar el diseño detallado de software orientado o basado en objetos.

Se sugiere que cuando se represente las clases utilizando UML, tenga en cuenta los siguientes puntos:

- a. Centrar el nombre de las clases (Identificando cuando sea una clase y clase abstracta la cual se representa con cursiva). (ver clase Persona)



- b. Justificar a la izquierda atributos, operaciones.  
 c. Incluir una elipsis (...) al final de una lista incompleta (ver clase Estudiante)



Para evaluar los conceptos de las unidades estudiadas hasta el momento, se propone resolver los ejercicios que se proponen en esta unidad, sin olvidarse de tomar en cuenta algunos de los tips que han sido extraídos del texto básico y bibliografía complementaria para diagramar o modelar, entre los que destacan:

- Empiece desarrollando modelos considerados simples, utilizando herramientas de modelado o en papel, para familiarizarse con los estereotipos, elementos y propiedades que luego puedan ser analizados a detalle y posteriormente codificados.
- Trate de entender y diseñar el modelo de manera simple, codificandolo de manera conjunta con un lenguaje de programación donde analice y visualice cada una de las propiedades de dicho modelo.

- No trate de diseñar diagramas para todo, concentrese netamente en la solución específica a proponer y en los objetivos que aporta cada diagrama.
- No se requiere cantidad de modelos, sino calidad por ello no trate de realizar todos, solo diseñe los específicos acorde al problema a solucionar.
- Utilice los diagramas de clase para representar las entidades y relaciones que dan solución a un problema sabiendo que estos se enfocan exclusivamente en la estructura de un sistema mas no en su comportamiento.

Le recuerdo que, existen lineamientos dados por autores como (Ambler, 2005), donde dentro del apartado denominado “*Class Style Guidelines*” expone escrito en idioma inglés algunas sugerencias para identificar clases, entre las que destacan (ver figura 33).

- Utilizar terminología común para nombre de clases: Esto significa que debe nombrarlas acorde al contexto del problema con la finalidad de entenderlas mejor.
- Utilizar sustantivos singulares completos: Por ejemplo, no utilizar para una clase Empleado Emp como abreviatura de una clase o por ejemplo si la clase es ClienteVIP usar CVP como abreviatura ya que de esta forma no es comprensible para las personas que revisarán el modelo que son otros analistas, desarrolladores u otro actor que participe del desarrollo de software.
- Se sugiere utilizar singular en lugar de plural para el nombre de clases, atributos y métodos, por ejemplo Alumno en vez de Alumnos, Universidad en lugar de Universidades.
- Los nombres de operaciones deben tener verbos fuertes: Las operaciones implementan la funcionalidad de un objeto, por

lo tanto estos deberían ser nombrados de forma en la que comuniquen efectivamente dicha funcionalidad.

- Tener en cuenta que no se utilizan espacios entre los nombres de las operaciones y que la primera letra es en minúsculas y la segunda en Mayúsculas en caso de nombres compuestos de dos palabras (fijarse en la diferencia y descripción en cada columna).

>The name was shortened because the term “TheObject” did not any value

#### Sugerencias para escritura de clase

Initial Name	Good Analysis Name	Good Design Name	Issue
Open Acc	Open Account	openAccount()	An abbreviation was replaced with the full word to make it clear what is meant.
Mailing Label Print	Print Mailing Label	printMailingLabel()	The verb was moved to the beginning of the name to make it active.
purchaseparkingpass()	Purchase Parking Pass	purchaseParkingPass()	Mixed case was applied to increase the readability of the design-level name
Save the Object	Save	save()	

Figura 33. Sugerencias para escritura de clase.

Tomado de: Ambler (2005)

- Asigne los atributos con nombres basados en Dominios: se debe utilizar una descripción completa del nombre dado al atributo con el fin de que la representación de éste sea obvia y entendible. En la figura 34 se muestra una convención Java que permite el nombrado de atributos donde además se considera que no se deben ubicar caracteres especiales y que para unir dos palabras se utilice “\_” como por ejemplo fechaNacimiento o fecha\_Nacimiento pero no fecha-Nacimiento o fecha#Nacimiento.

## Convención Java escritura de atributos

Initial Name	Good Analysis Name	Good Design Name	Issue
fName	First Name	firstName	Do not use abbreviations in attribute names.
firstName	First Name	firstName	Capitalizing the second word of the design name makes the attribute name easier to read.
personFirstName	First Name	firstName	This depends on the context of the attribute, but if this is an attribute of the "Person" class, the including "person" merely lengthens the name without providing any value.
nameLast	Last Name	lastName	This name "lastName" was not consistent with "firstName" (and it sounded strange anyway).
hTTPConnection	HTTP Connection	httpConnection	The abbreviation for the design name should be in all lowercase.
firstNameString	First Name	firstName	Indicating the type of the attribute, in this case "string", couples the attribute name to its type. If the type changes, perhaps because you decide to reimplement this attribute as an instance of the class "NameString," then you will need to rename the attribute.
OrderItemCollection	Order Items	orderItems	The second version of the design name is shorter.

Figura 34. Convención Java escritura de atributos. Fuente: Ambler (2005)

- Adicional a lo expuesto anteriormente, se puede mencionar que las clases, las cuales definen un tipo de objeto y las características de sus objetos, se pueden describir en

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

varios niveles de precisión y concreción. En primeras etapas del diseño, el modelo captura los aspectos más lógicos y generales del problema y en etapas posteriores, el modelo se refinará y se captura decisiones de diseño y detalles de implementación, con esto se logrará proporcionar una solución más acorde al problema o necesidad del cliente.



### Actividades de aprendizaje recomendadas

Muy bien, es momento de que los conocimientos respecto a los diagramas de clases y la nomenclatura UML sean evaluados, para ello le invito a que seleccione la mejor alternativa a las siguientes preguntas propuestas.

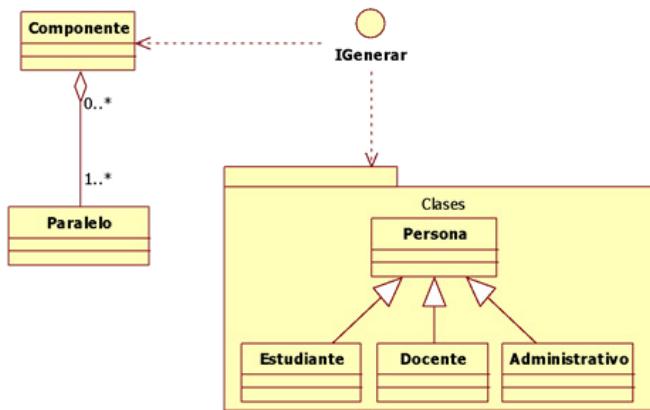


## Autoevaluación 6

Muy bien, es momento de que los conocimientos respecto a los diagramas de clases y la nomenclatura UML sean evaluados, para ello le invito a que seleccione la mejor alternativa a las siguientes preguntas propuestas.

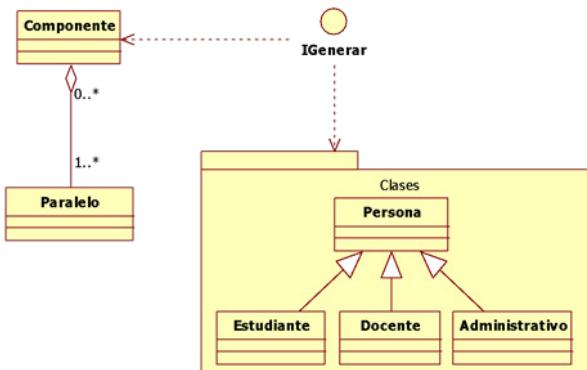
1. Una clase en su representación formal con UML está compuesta por tres compartimentos distribuidos ordenadamente como:
  - a. Métodos, Atributos, Clase.
  - b. Atributos, Método, Clase.
  - c. Atributos, Clase, Métodos.
  - d. Clase, Atributos, Métodos.
  
2. Si “Periodo Académico” se considera una clase. La mejor alternativa para nombrarla en UML sería:
  - a. Periodo-Académico.
  - b. Per\_Acad.
  - c. PeriodoAcademico.
  - d. PeriodoAcadémico.
  
3. La asociación que permite representar el siguiente enunciado “una clase X se compone de una Clase Y o una clase Z y que se representa con un diamante negro en la terminación de la asociación” se conoce como:
  - a. Agregación.
  - b. Herencia.
  - c. Dependencia.
  - d. Composición.

4. La dependencia que en su concepto menciona que es “usada temporalmente” es hace uso de
- Una o varias interfaces sin relacionarse con ninguna clase.
  - Una o varias interfaces solamente.
  - Una o varias interfaces y su relación con una o varias clases.
  - Ninguna de las anteriores.
5. En base a la siguiente representación UML. ¿Qué asociaciones UML se muestran?



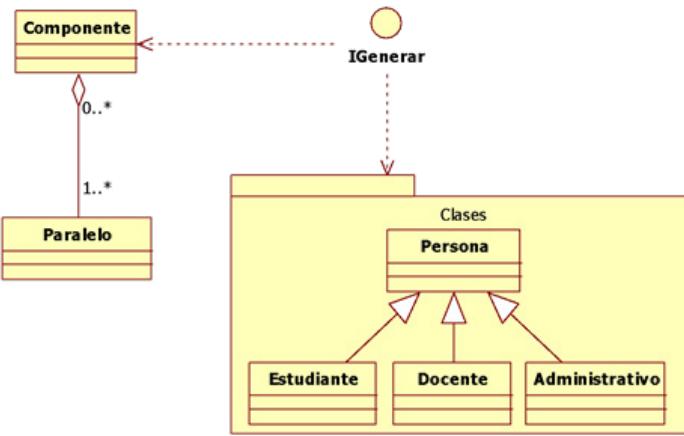
- Composición.
- Agregación.
- Dependencia.
- Herencia.

6. En base a la representación gráfica UML, ¿qué significa IGenerar?



- a. Una interfaz gráfica de Usuario.
- b. Una interface que tiene una dependencia de la clase Componente y del Paquete Clases.
- c. Una interface que realiza la clase Componente y el Paquete Clases.
- d. Una interface gráfica de Usuario que tiene una dependencia de la clase Componente y del Paquete Clases.

7. En base a la representación gráfica UML, ¿Qué nombre tiene la clase abstracta?



- a. Componente.
  - b. Persona.
  - c. IGenerar.
  - d. No existe clase abstracta.
8. Si “Dirección Uno” es un atributo de una clase identificado en la fase de Análisis, ¿cuál es la mejor alternativa para representarla en UML desde el punto de vista de diseño?
- a. dirección Uno.
  - b. direcciónUno:String.
  - c. direccionUno: String.
  - d. DireccionUno: String.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

9. Para representar la multiplicidad de relaciones entre clases en java, cuando es 1 a muchos (1..\*), se puede hacer uso de estructuras de datos como:
- a. Enteros.
  - b. Arreglos.
  - c. ArrayList.
  - d. List.
10. ¿Cuál podría ser la mejor opción para ubicar los roles al inicio y fin de las asociaciones entre las clases Componente y Paralelo y que su lectura en lenguaje natural sea el adecuado?
- a. 1 o muchos Paralelos “se configuran para” ningún o muchos componentes.
  - b. Ningún o muchos componentes “se ofertan para” 1 o muchos paralelos.
  - c. 1 o muchos paralelos tienen 0 o muchos componentes y viceversa.
  - d. Todas las opciones son correctas.

[Ir al solucionario](#)

**Lecturas recomendadas:**

- (Unidad 6.3) [Revise por favor el recurso educativo abierto \(Ingeniería de Software\)](#) en el Tema 8. Estructura del sistema, el cual le ayudará a reforzar los conocimientos del tema de clases. Páginas 75 a 90.

Teniendo en cuenta lo expuesto hasta el momento y apoyado en el texto básico, proceda por favor a resolver el siguiente ejercicio, donde se requiere que identifique clases, atributos, métodos que podrían implementarse con programación, trate de ir representándolos en StarUML.

**Ejercicio 13**

- 13.1. ¿Qué es un diagrama de clases y para qué sirve?
- 13.2. ¿En base a qué requisitos se diseña un diagrama de clases?
- 13.3. ¿Qué estereotipos, relaciones y nomenclatura se utilizan en un diagrama de clases?, proponga un modelo.
- 13.4. ¿Cuál es la simbología que se usa para representar asociación, dependencia, herencia, agregación, composición?, represéntelos a través de un diagrama de clases.
- 13.5. En un diagrama de clases, ¿cómo se identifica que una clase sea abstracta?, explíquelo y represente gráficamente.
- 13.6. Realice un ejemplo de diagrama de clases que contenga clase abstracta, herencia, composición, agregación y codifiquelo utilizando un lenguaje de programación.

**Ejercicio 14**

La Guardería “Mis primeros pasos” de la ciudad de Loja desea implantar un sistema que se ajuste a los requerimientos de la

institución y por ello le solicita su apoyo con el análisis para el desarrollo de un prototipo de una de las funcionalidades que el sistema debe tener. Las funcionalidades del sistema son:

Inscripción, Pagos o pensiones, Recursos humanos y Reportes. La funcionalidad del prototipo en la que se requiere su apoyo tiene que cumplir con los requerimientos de *Ingreso* y *Consulta* de niños con sus representantes.

Para cubrir el proceso de inscripción, el sistema solicitará a través de una interfaz gráfica:

### **Del niño**

- Nombres
- Fecha de nacimiento
- Días de nacimiento: serán calculados en base a la fecha actual y la fecha de nacimiento
- Tipo de Sangre
- Alergias: la opción de selección será SI o NO.
- Vacunas: la opción de selección corresponde al mes de vacuna por ejemplo enero, febrero, marzo, y demás.
- Medicamentos: la opción de selección será SI o NO.

### **Del representante**

- Nombres
- Fecha de nacimiento
- Correo electrónico
- Número de fono fijo en caso de emergencia
- Número de celular en caso de emergencia

El prototipo permitirá asociar a cada niño con su representante y seleccionar los días (*cualquier día(s) de la semana*) que el niño asistirá a la guardería, El sistema finaliza el proceso de inscripción registrando la fecha y hora de admisión del niño, la misma que

será obtenida desde el sistema. Lo que se solicita es identificar las clases, atributos y métodos y representarlas con UML utilizando las buenas prácticas propuestas por los autores en la bibliografía básica y complementaria.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

## Resultado de aprendizaje 6

Selecciona los componentes que debe incluir en un proceso para solucionar un problema específico.

### Contenidos, recursos y actividades de aprendizaje



Semana 12



## Unidad 7. Interfaces, Paquetes, Componentes

Las clases se pueden relacionar con Interfaces, las clases y la relación entre ellas representadas a través de diagramas pueden ser agrupados dentro de paquetes y componentes, es por ello que iniciemos el estudio referente a Interface. Tener claro que el término Interfaz es distinto a Interface.

### 7.1. Interfaces

Usted conoce que las clases pueden asociarse e intercambiar mensajes con otras clases y/o interfaces. Estas pueden estar

contenidas en paquetes y/o componentes los cuales se utilizan para organizar los elementos del modelo. En esta unidad revisaremos estos conceptos que sirven de apoyo al momento de realizar un modelo y que posteriormente son utilizados en diagramas específicos. Una **Interface** se puede considerar como un contrato o servicio, similar a las interfaces de programación de aplicaciones (API, por sus siglas en inglés), que tiene operaciones o implementaciones de lógica interna (como una caja blanca sobre la que no sabemos que existe en su interior), sin atributos, ni asociaciones, ni instancias pero que son proporcionadas o solicitadas por otra interface, componente o sistema para su uso o consumo. En la Figura 35 se expone la representación UML de interfaces con los elementos que intervienen.

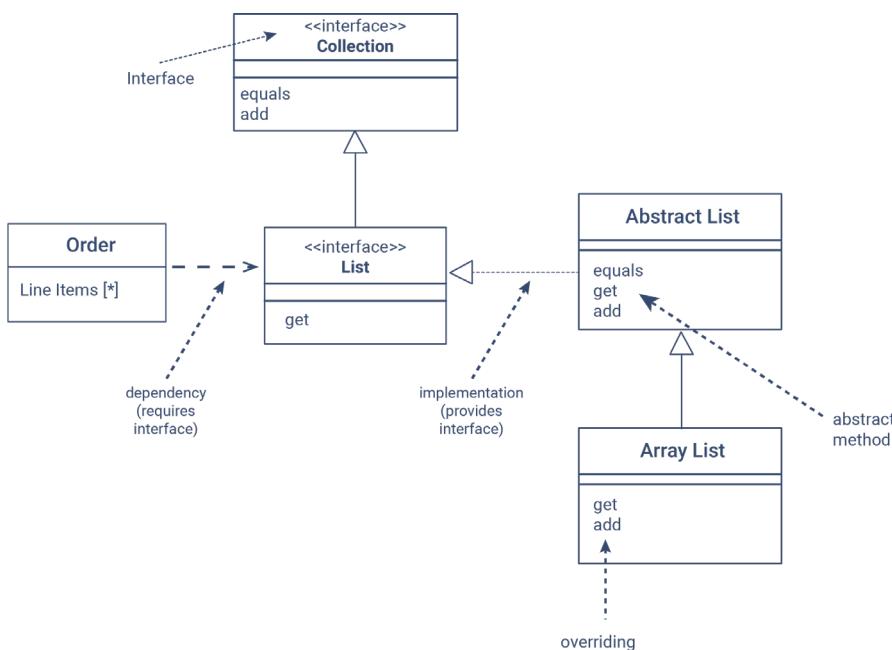


Figura 35. Representación en UML de interfaces y sus elementos. Fuente: Fowler (2004).

Índice

Primer bimestre

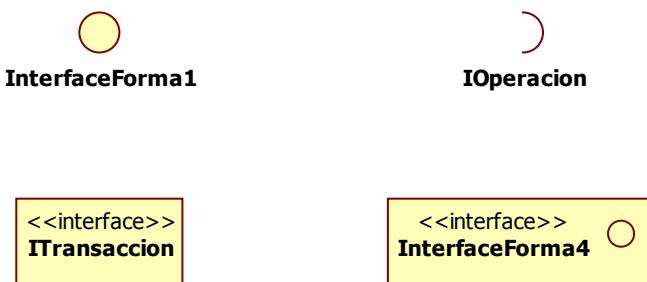
Segundo bimestre

Solucionario

Referencias bibliográficas

En el texto básico se menciona que una interface puede ser de dos tipos: interface proporcionada e interface requerida. En ambos casos, su definición y representación por medio de UML es diferente. Por ello, se sugiere a más de los conceptos, identificar las diferencias a nivel de representación UML que permiten su comunicación con paquetes y componentes. Para ello le invito a revisar en el texto básico la representación gráfica y textual de una interface, la cual empieza con una letra “I”, por ejemplo “Interface” – ISemantica – IProtocolo – IFinanciera – IContexto, etc. En StarUML las Interfaces pueden ser representadas gráficamente en UML de diferentes formas como se muestran a continuación.

Tabla 5. Formas de representar una interfaz con UML.



Para poder añadir una interface en StarUML debemos arrastrar y soltar el elemento que tiene el símbolo  . Para poder visualizar la misma de diferentes formas debemos hacer click derecho sobre la figura de la interface para poder desplegar las diferentes formas. Luego hacemos click en *Format*, luego en *Stereotype Display* y finalmente podemos variar las formas para comprobar su representación (*None*, *Textual*, *Iconic*, *Decoration*) como se expone en la figura 36.

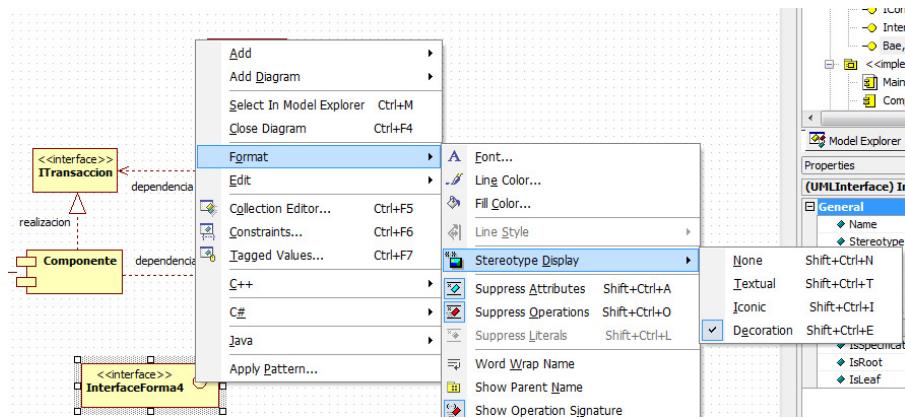


Figura 36. Formas para cambiar la visualización de una interface.

Luego de revisar la bibliografía, usted ha identificado que una interface se representa en UML o a través de lenguaje de programación como una clase signada con la palabra reservada “*interface*”. Otra forma de representar con UML una interface es por medio de un pequeño círculo con el nombre *interface* situado cerca del símbolo y sin mostrar las operaciones de la interface (dependiendo si es interface proporcionada o requerida). Recuerde utilizar y revisar bibliografía complementaria (Ambler, 2005), en el apartado denominado “*Guidelines for UML Interfaces*”, donde encontrará escrito en idioma inglés algunas sugerencias, definición e implementación de Interfaces. Un ejemplo donde se visualiza los tipos de Interfaces (provista y requerida), clases, relaciones, paquetes y componentes se expone en la Figura 37.

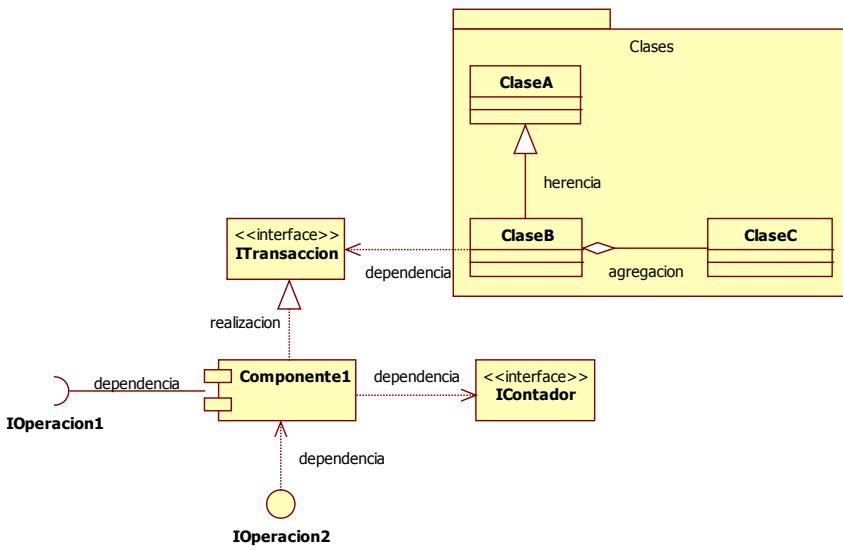


Figura 37. Clases, interface, paquetes, componentes y sus relaciones.

## 7.2. Paquetes

Los Paquetes permiten agrupar/organizar y relacionar elementos del modelo lógico (clases, relaciones y casos de uso), especialmente cuando la solución que se propone contiene muchos elementos del modelo y su explicación resulta compleja.

Visualmente los paquetes se pueden representar en forma de carpetas de archivos o directorios y se pueden diseñar en cualquier diagrama UML, aunque su especificación está dada en los diagramas de paquetes, similar a cómo lo haría en java u otro lenguaje de programación donde dentro de un paquete ubica clases, librerías, UI (User Interface), reglas de negocio, acceso a datos, modelo, controlador, css (hoja de estilos), archivos (usados por la aplicación o usados como salida resultante de ejecución de la aplicación).

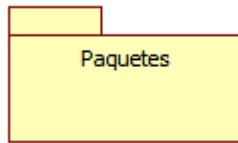
Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas



Es necesario mencionar que cuando se utilizan paquetes y sus relaciones, se revise la nomenclatura, representación de los elementos y estereotipos y dirección de las flechas que permiten relacionar/conectar un paquete con otro. Similar a como se realiza con las clases e interfaces. Además, es necesario que los paquetes tengan un nombre único, ya que no se puede tener dos paquetes con el mismo nombre por convención y posterior implementación. En la, se muestran tres paquetes denominados como Presentación, Negocio y Datos (similar a una arquitectura en layers) y dentro de cada uno de ellos contiene a otros paquetes. Para relacionar paquetes se utiliza como relación dependencia, implementacion, import/access o merge como estereotipos UML para su representación. En la se muestra la relación de dependencia, donde un paquete depende de otro o de una clase(s) en particular.

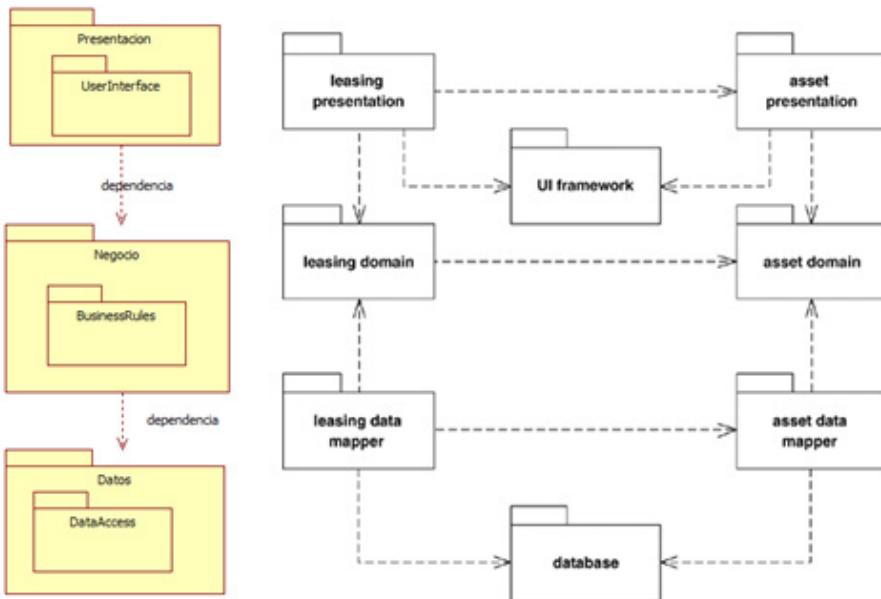


Figura 38. Representación de paquetes y sus relaciones. Fuente: Fowler (2004).

Los paquetes se consideran agrupadores de un grupo de elementos básicos (clases y casos de uso). Los nombres que se ubican a los paquetes son únicos, no se puede tener dos paquetes con el mismo nombre y como se mencionó anteriormente un paquete puede contener otros paquetes creando así una jerarquía de paquetes. Las relaciones utilizadas dentro de un diagrama de paquetes son dependencia, implementación, import/access, merge.

- **Dependencia:** Se usa cuando el paquete A depende del paquete B, si A contiene una clase que depende de una clase en B. La representación gráfica es:

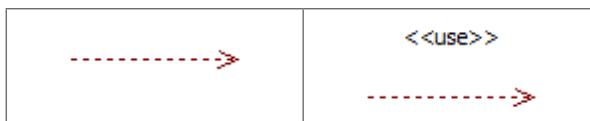


Figura 39. Simbología usada para representar dependencia.

- El estereotipo que se utiliza es, aunque no es necesario utilizarla, ya que por defecto se entiende al usar la representación a través de la línea punteada. En la dependencia es entre paquetes y corresponde a la distribución de “3-layer lógica”, donde en el paquete Presentación contiene la Interfaz gráfica de usuario, el paquete de Negocio contiene los cálculos o reglas para los mismos, y en el paquete de Datos contiene el acceso y comunicación a los datos que se almacenan en la base de datos.
- Implementación:** usado cuando para una implementación existen muchos caminos con objetivos diferentes. Su representación gráfica es como se muestra a continuación, donde la Puerta de enlace o Gateway del nivel Padre o BaseDatos es similar pero con implementaciones diferentes para cada motor de base de datos.

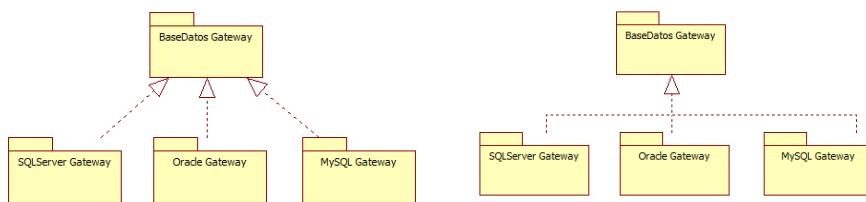


Figura 40. Formas de representar gráficamente la relación de implementación.

- Import/access:** para hacer uso de las relaciones de Import/Access, es necesario conocer como se relacionan las clases y como las clases son requeridas por otros paquetes. **Import** usa como estereotipo de relación `<>` que indica que los elementos importados dentro del paquete de importación es público. En el caso de **Access** que hace uso del estereotipo `<>`, indica que los elementos importados dentro del paquete de importación son privados (ver Figura 41).

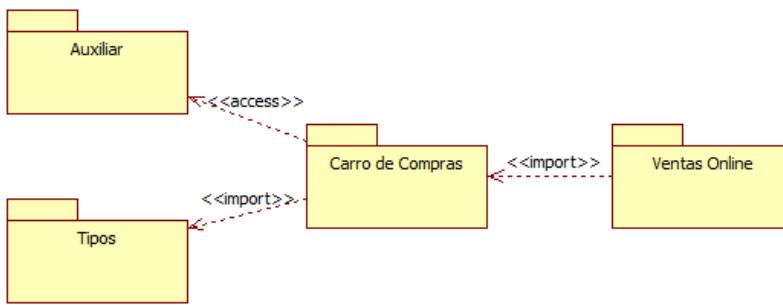


Figura 41. Ejemplo de Import y Access a nivel de paquetes.

En base a la representación de paquetes y su relación en UML, analice las siguientes preguntas y respuestas.

- a. ¿Son los elementos de **Tipos** accesibles desde **Ventas Online**? - SI
- b. ¿Son los elementos de **Auxiliar** accesibles desde **Ventas Online**? - NO
- c. ¿Son los elementos de **Tipos** accesibles desde **Carro de Compras**? - SI
- d. ¿Son los elementos de **Auxiliar** accesibles desde **Carro de Compras**? - SI
  
- **Merge:** Indica que el contenido (clases y sus relaciones) de dos o más paquetes pueden ser combinados en uno solo, representando a una suma o unión. Por ejemplo, si unimos Paquete A + Paquete B el paquete resultante es A' o B', dependiendo del paquete si es emisor o receptor.

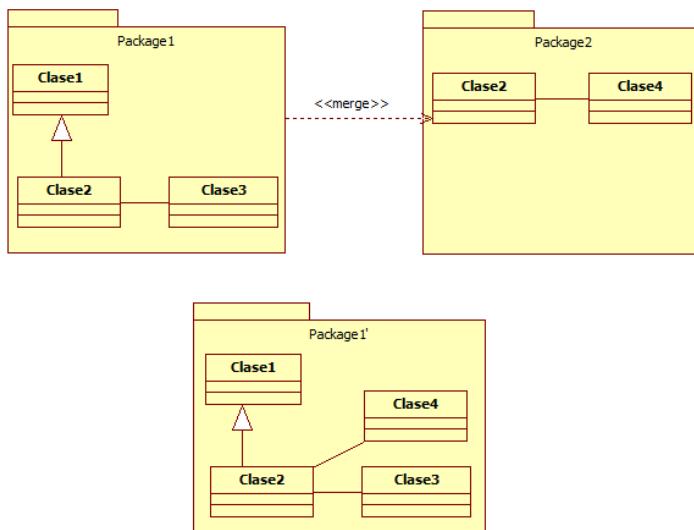


Figura 42. Representación de Merge de Paquetes y su contenido.

Finalmente, recuerde que un diagrama de paquetes que se usa básicamente para dos cosas: agrupar un diagrama de clases y agrupar casos de uso. Muy bien, es hora de realizar el siguiente ejercicio donde los conceptos teóricos referentes a interfaces y paquetes se proponen como parte de su auto-estudio. Aspectos teóricos y representación gráfica a través de UML como clases, interfaces, paquetes y componentes van de la mano, por ello una vez que conoce la parte teórica y de representación de interfaces y paquetes es hora que utilizando StarUML realice o replique los ejemplos prácticos del texto básico y propuestos en el curso.



### Actividades de aprendizaje recomendadas

Para evaluar los conocimientos adquiridos respecto a Interfaces, paquetes y componentes, es necesario realizar la presente Autoevaluación con el fin de conocer el avance y comprensión de los conceptos que son usados en la práctica.



## Autoevaluación 7

1. ¿De qué tipo puede ser representada una interface en UML?
  - a. Provista.
  - b. Requerida.
  - c. Java.
  - d. Todas las anteriores.
2. ¿Cuándo se representa una interface en UML, cuál es la estructura que ella posee?
  - a. Nombre de Interface, Atributos, Métodos.
  - b. Nombre de Interface, Atributos.
  - c. Nombre de Interface, Métodos.
  - d. Atributos, Métodos.
3. ¿Cuál es la mejor opción respecto a los métodos que se definen en una Interface en UML?
  - a. Se implementan con su lógica en la Interface definida.
  - b. Se implementan con su lógica en la Clase que lo requiere.
  - c. Los métodos pueden tener uno o varios parámetros.
  - d. Todas las opciones son correctas.
4. ¿Cuáles son las asociaciones que se usan en UML para relacionar interfaces con componentes?
  - a. Agregación.
  - b. Composición.
  - c. Realización.
  - d. Dependencia.

5. ¿Cuál es la mejor alternativa respecto a los paquetes y su uso?

- a. Los paquetes en UML se usan para representar que internamente existen clases y sus relaciones.
- b. Las relaciones entre paquetes usan estereotipos como import, access.
- c. El símbolo de la flecha para asociar paquetes es a través de una línea punteada.
- d. Todas las anteriores

6. ¿Cómo se comunican o relacionan los componentes?

- a. Generalización.
- b. Composición.
- c. Agregación.
- d. Interfaces.

7. ¿Cuál es la mejor alternativa referente a componentes?

- a. Un componente puede contener a uno o muchos componentes.
- b. Un componente puede ser expuesto como una vista Caja Negra.
- c. Un componente puede ser expuesto como una vista Caja Blanca.
- d. Un componente a nivel de implementación es una funcionalidad

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

8. ¿Cuál es la mejor alternativa que se puede ubicar en el espacio en blanco para para el siguiente enunciado? “El cliente expone que debe existir \_\_\_\_\_ que permita registrar los datos de un estudiante”
- Un componente.
  - Una funcionalidad.
  - Un paquete.
  - Una interface.
9. ¿Cuál es la mejor alternativa que se puede ubicar en el espacio en blanco para el siguiente enunciado? Uno(a) de \_\_\_\_\_ que forma parte del sistema se denomina “Gestionar Personal”. Este contiene a funcionalidades como Registrar Personal, Buscar Personal, Editar Personal y Dar de baja al Personal, es decir hacer el CRUD correspondiente”
- las funcionalidades.
  - los paquetes.
  - los componentes.
  - las interfaces provistas.
10. ¿Cuál(es) de las siguientes opciones se consideran correctas?
- Un componente a nivel de diseño contiene a uno o más componentes.
  - Un componente a nivel de diseño y codificación contiene clases.
  - El modelado de componentes se lo realiza en la fase de implementación.
  - El modelado de componentes comienza posterior a la codificación completa del sistema.

Ir al solucionario

## Lecturas recomendadas

- (Unidad 7.1) En el texto básico revise los **conceptos de Interface, Interface solicitada e interface proporcionada**, su representación visual y textual expuestos en el Capítulo 11.
- (Unidad 7.2) Para revisar su conceptualización, características, uso y representación con UML, le invito a revisar en el texto básico el capítulo 12 correspondiente a **Paquetes**.
- (Unidad 7.2) [Para ampliar el estudio de paquetes, le invito a revisar el recurso educativo abierto \(Ingeniería de Software\) Tema 9. Arquitectura Lógica del Sistema. Revisar ejemplos desde páginas 3 a 47.](#)

### Ejercicio 15:

- 15.1. ¿Cómo se representa gráficamente usando UML una Interface requerida y una Interface provista?
- 15.2. Con base en los conceptos estudiados, ¿para qué sirve una interface?, proponga un ejemplo de su uso.
- 15.3. Represente usando una herramienta CASE de modelado, la comunicación de una interface con una o varias clases.
- 15.4. Mencione al menos 4 escenarios donde utilizaría Interfaces.
- 15.5. ¿Cuál es la diferencia entre una interface y una clase desde el punto de vista de su concepto y su representación con UML?

### Ejercicio 16. Responda a las siguientes inquietudes

- 16.1. ¿Para qué sirven y cómo se representa los paquetes en UML?
- 16.2. ¿Qué contiene o puede contener un paquete?, ¿Clases, relaciones, paquetes, componentes, interfaces, etc?



## Semana 13

### 7.3. Componentes

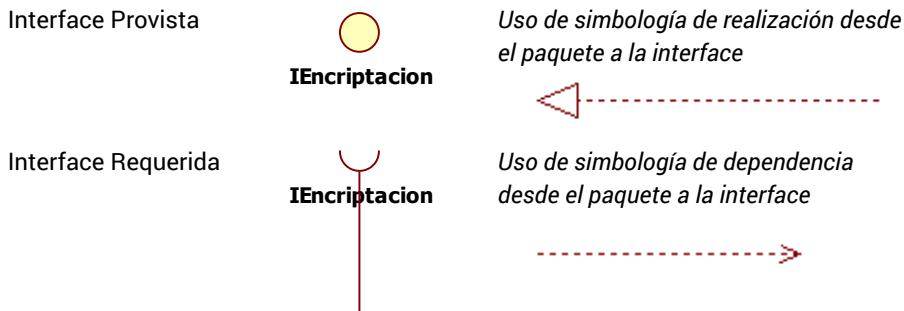
Los **Componentes** comparados con algunos elementos UML expuestos anteriormente, tienen algo en común. Por ejemplo en el uso de estereotipos, nomenclatura y relaciones que permiten su representación y comunicación. Previo a la revisión del texto básico, es necesario mencionar que en representación UML los componentes se conectan a través de interfaces requeridas o proporcionadas y asociaciones. Además, los componentes son utilizados para representar de forma lógica o física piezas modulares de un sistema cuyo comportamiento visible externamente se puede describir mucho más concisamente que su implementación.

Es necesario recalcar que los elementos UML pueden variar en su representación gráfica de una versión o herramienta CASE a otra, pero su significado y uso es el mismo. En este caso se muestra la simbología de componente en UML1 en la izquierda y UML2 a la derecha ambos soportados por StarUML.



*Figura 43. Formas de representar con nomenclatura UML a los componentes.*

Recordar que para la comunicación de los componentes es necesario usar interfaces provistas y requeridas representadas con UML como se muestra a continuación.



¿Revisó el texto básico?, ¿está de acuerdo con lo que se menciona a continuación respecto a componentes? “**Un Componente** puede contener uno o más componentes y a uno o muchos paquetes y éstos paquetes contienen una o muchas clases relacionadas. Por lo que las dependencias entre los componentes suelen ser un subconjunto de las dependencias entre los paquetes”. Muy bien sus opiniones, identificación de conceptos similares y diferencias son importantes al momento de aplicarlas en un diagrama.

Un componente se define también como una unidad modular con interfaces bien definidas y puede exponerse como dos vistas, *Caja blanca* y *Caja negra*.

- **Vista Caja Negra:** se muestran solamente interfaces provistas e interfaces requeridas.
- **Vista Caja Blanca:** muestra la estructura interna de las interfaces y/o su estructura interna

En el ejemplo de la figura 44, se expone un diagrama de componentes, donde intervienen Gestión Académica y Gestión Financiera. Estos por medio de sus interfaces hacen uso de un Componente denominado Seguridad y Persistencia, los cuales permiten realizar el registro de datos sobre una base de datos. Note la representación en dicha figura donde se observa interfaces requeridas e interfaces proporcionadas especialmente.

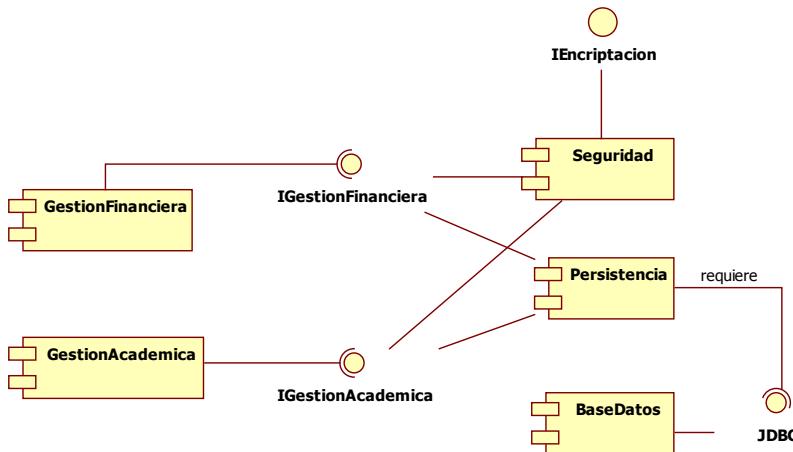


Figura 44. Figura 44. Representación de componentes e interfaces en StarUML

Es necesario comentar que los componentes no dependen directamente de otros componentes, sino de las interfaces que soportan éstos y que al igual que los demás elementos en UML, hacen uso de su representación y estructura como se muestra en la figura 45. Por ello, le invito a replicarlos a los ejemplos de la figura con StarUML y así comprender mejor los conceptos de componente, puerto, parte y conector.

Revisando bibliografía complementaria, en (Ambler, 2005) se mencionan algunas consideraciones para definición y uso de los componentes “*UML Component Diagrams*”, entre los que constan:

- Aplicar nombres descriptivos a componentes arquitectónicos.
- Aplicar convenciones de nomenclatura específicas del entorno a los componentes de diseño detallado.
- Aplicar estereotipos textuales coherentes.
- Evite modelar datos y componentes de la interfaz de usuario.
- Mostrar solamente interfaces relevantes.

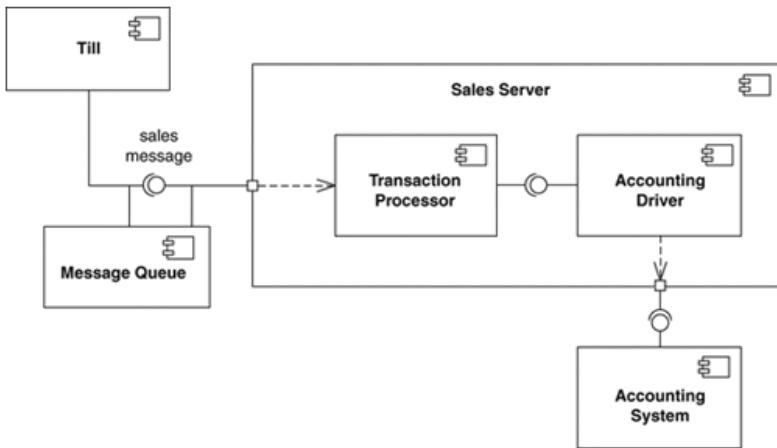


Figura 45. Componente que contiene otros componentes. Fuente: Fowler (2004).

Como se observa en la figura 45, un componente puede contener otro componente y para su comunicación hace uso de interfaces. El modelado de componentes es un tipo especializado de modelado estructural que se usa durante las actividades de diseño para representar una aproximación a la implementación y despliegue de un sistema software. Normalmente, el modelado de componentes comienza una vez que el análisis y diseño del sistema por medio de diagramas como casos de uso, diagrama de clases, modelo entidad relación, diagrama de procesos, se encuentren bastante completo, según lo determine el proceso de desarrollo de su sistema.

Espero que hasta el momento vaya asociando los conceptos teóricos con la práctica utilizando para ello herramienta CASE y nomenclatura UML



## Actividades de aprendizaje recomendadas

- (Unidad 7.3) Le invito a revisar el texto básico el capítulo 15 donde encontrará información relacionada a conceptos, características y representación gráfica de **Componentes**.
- (Unidad 7.3) Para ampliar el estudio de componentes, le invito a revisar el recurso educativo (Ingeniería de Software) abierto Tema 11. Arquitectura Física del Sistema.

### Resultado de aprendizaje 7

Identifica la utilidad de los diagramas respecto a su representación y exposición desde el punto de vista teórico/práctico a través de un modelo.

## Contenidos, recursos y actividades de aprendizaje



### Semana 14

Como avances en el desarrollo de su tarea que forma parte de la praxis profesional (cátedra integradora/proceso práctico de aprendizaje) hasta el momento debe haber realizado los diagramas correspondientes a clases, interfaces, paquetes y componentes.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Recuerde usar elementos, nomenclatura y estereotipos de forma correcta que se explican en el presente curso, texto básico y bibliografía complementaria. En caso de tener alguna duda utilice los distintos medios de comunicación para contactarse con su docente/tutor y solventar las mismas. Una vez que estamos en la capacidad de analizar un problema (requisitos funcionales y no funcionales) dentro de un contexto, proponer una o varias soluciones a dicho problema, presentar documentación arquitectónica haciendo uso de los distintos diagramas (casos de uso, actividades, estado, secuencia, clases, paquetes, componentes) y representarlos gráficamente haciendo uso de UML, es hora de estudiar algunas formas para representar el diagrama de despliegue.



## **Unidad 8. Diagrama de despliegue y modelado arquitectónico**

En un diagrama de despliegue usando UML, se muestran los nodos (por lo general hardware, dispositivos), los componentes de software que se encuentran desplegados o se ejecutan en dichos nodos y el “middleware” o comunicación (protocolos, tecnologías de comunicación) utilizados para conectar dichos nodos.

Los conceptos estudiados en unidades anteriores y en otras asignaturas, son necesarios en esta unidad 8, en la cual se debe representar gráficamente cómo desde el punto de vista lógico y físico, como parte de la arquitectura de un sistema, se puede desplegar una solución software.

## 8.1. Despliegue y diagrama de despliegue

El despliegue representa la relación existente entre los elementos lógicos (componentes y clases), físicos de un sistema (Nodos, dispositivos) y los activos de tecnologías de información asociados a ellos (Artefactos).

### Nodos

Un nodo representa un dispositivo o recurso computacional y un ambiente de ejecución los cuales generalmente tienen memoria o capacidad de procesamiento y almacenamiento. Los dispositivos conocidos también como hardware o recursos computacionales pueden ser: CPU, impresora, laptop, dispositivo móvil, entre otros.

Los ambientes de ejecución pueden ser un servidor de base de datos, servidor de aplicaciones, servidor web, servidor de correo, máquinas virtuales, contenedores, y demás. Los nodos pueden ser físicamente conectados a través de canales de comunicación tales como cables, wireless, bluetooth haciendo uso de tecnología o protocolos de comunicación. Para representar la conexión entre dos o más nodos con UML se hace uso de una relación de asociación.

Gráficamente, un nodo o instancia de nodo, en StarUML se representa como un cubo estilizado como se expone en la Figura 46, donde en su interior puede contener componentes o artefactos.

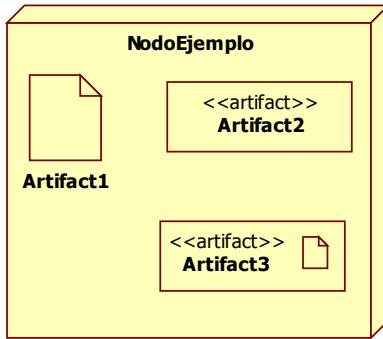


Figura 46. Representación en StarUML de un Nodo y sus artefactos

En otro ejemplo tomado de (Rumbaugh et al., 2004), se visualizan los nodos BankServer (representado como servidor) y ATMKiosk (representado como un cliente quien hace petición al servidor) (ver Figura 47). En la figura se puede observar que un tipo de nodo en UML se lo nombra usando nomenclatura bajo el formato `<>:<>`, en la figura se expone como `<>:<>`. Una instancia de nodo se muestra como un cubo que en su interior contiene una cadena de texto subrayada con su nombre y tipo de nodo. Las asociaciones entre nodos representan rutas de comunicación ( communication link ) y pueden tener estereotipos para distinguir diferentes tipos de caminos. En la figura se observa los elementos tales como interface, componente, relaciones de dependencia y estereotipos los cuales puede replicarlos.

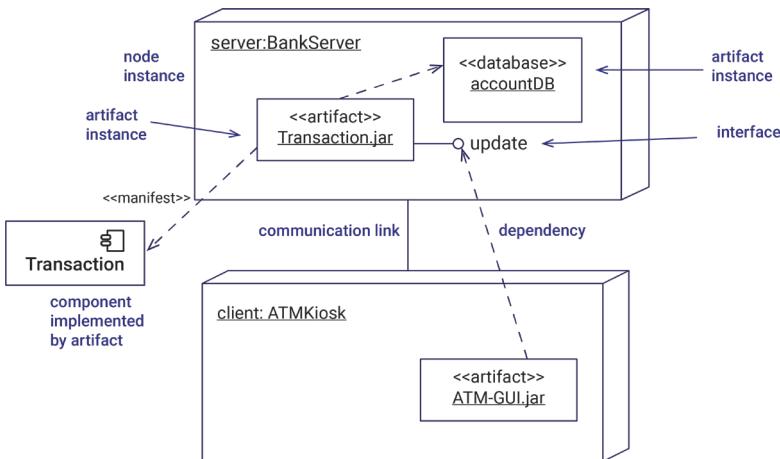


Figura 47. Ejemplo de despliegue. Fuente: Rumbaugh et al.(2004).

Los artefactos (`<>`) son elementos de información creados durante el desarrollo del software o cuando el sistema se ejecuta. Un ejemplo de `<>` sería cuando Ud. realiza una aplicación cliente y al empaquetar o crear el .war, .jar, .exe de su aplicación se requiere la inclusión de librerías (propias del lenguaje o externas) para el correcto funcionamiento de la aplicación, de lo contrario existirán errores en su ejecución. De igual manera si utiliza archivos con extensión .css, hojas de estilo o librerías (.dll). Estos, al momento de desplegar la aplicación web en un servidor, deben estar incluidos de lo contrario existirán errores al momento de interactuar con la interfaz gráfica de la aplicación.

Entre algunos de los considerados artefactos constan: archivos fuente, archivos scripts, archivos ejecutables, tablas de base de datos, archivos de configuración con extensión xml, csv, txt, doc, etc. En UML la representación es similar a una clase pero con el nombre de estereotipo `<>` y para su asociación con un nodo u otro artefacto se utiliza como relación el símbolo de dependencia.

Espero que, basándose en el estudio, haciendo uso de la bibliografía correspondiente, tenga en cuenta que en un diagrama de despliegue

intervienen conceptos ya conocidos tales como relaciones, componentes, paquetes, interfaces, estereotipos los que se usan para documentar una solución.

Los diagramas de despliegue muestran el diseño físico de un sistema, revelando las piezas de software que se ejecutan en distintas partes o nodos de hardware. Estos se consideran simples y útiles cuando se requiera mostrar una propuesta de solución software hacia los clientes, para ello es necesario se utilice nomenclatura básica y necesaria. Un ejemplo del uso de dicha nomenclatura es como la que se muestra en la figura tomada de (Fowler, 2004).

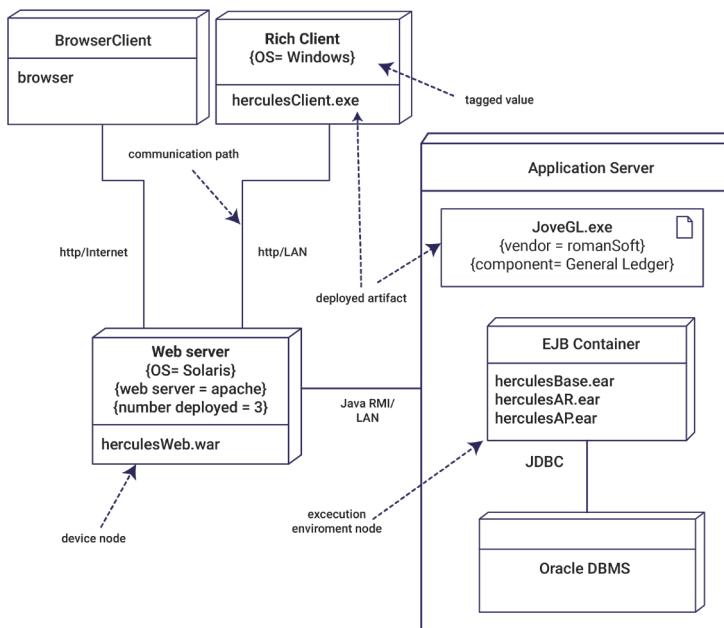


Figura 48. Ejemplo de despliegue. Fuente: Fowler (2004).

Como puede observar en la Figura 48, existen cuatro nodos nombrados como *Application Server*, *Rich Client*, *BrowserClient* y *Web server*, los cuales para su comunicación hacen uso de protocolos tales como http (internet y LAN) o tecnologías como RMI (Java

Remote Method Invocation), JDBC(Java Database Connectivity). Cada nodo contiene en su interior artefactos deployados que corresponden a aplicaciones instaladas o archivos que se ejecutan en el nodo. Se puede observar además que si desea explicar los detalles, en este caso se expone el sistema operativo que está instalado en el Servidor y algunas aplicaciones relevantes para el funcionamiento de la aplicación tales como los que se muestran en *Web server, Rich Client y Application Server*.

Otro ejemplo de diagrama de despliegue es el tomado de (Ambler, 2005), donde se muestra el despliegue de un sistema Web. En dicho diagrama, se usan cuatro nodos que se comunican vía RMI, JDBC o mensajes a través de un bus de servicios. Nótese la diferencia con la figura anterior que dentro de EJBContainer se tiene componentes ( Student, Seminar, Schedule ) que se ejecutan (ver Figura 49). Otro nodo donde en su interior se muestra un componente es :DBServer y el componente ( University DB ) hace referencia a una base de datos Oracle

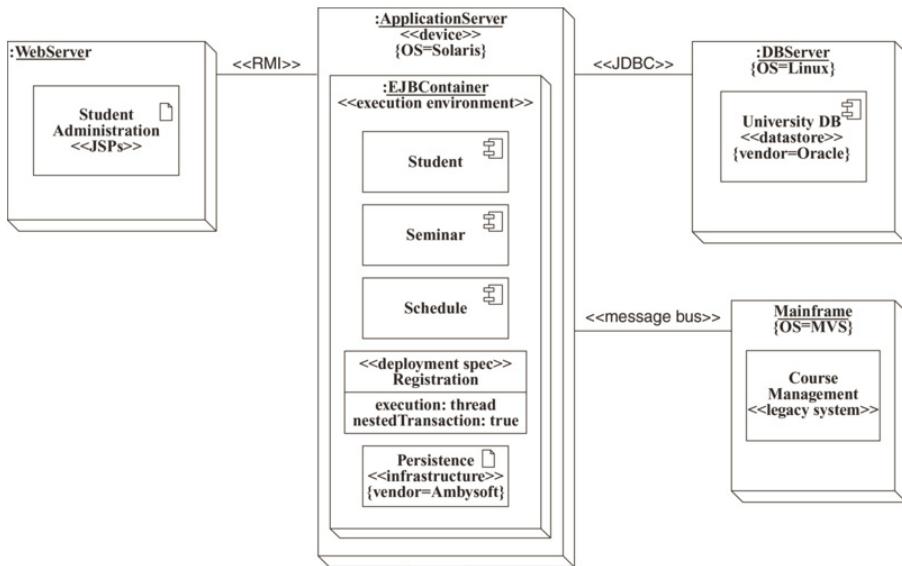


Figura 49. Ejemplo de despliegue. Fuente: Ambler (2005).

¿Sirven como referencia los diagramas de despliegue mostrados como ejemplos? Le invito a replicarlos en su equipo de trabajo utilizando una herramientas CASE de modelado. Seguramente usted también revisó el texto básico donde se encuentra un ejemplo que contiene representación gráfica y textual de un diagrama de despliegue, pero en este caso de una aplicación cliente, si aún no lo ha hecho le invito a revisarlo ya que complementará a los ejemplos expuestos previamente.

Es necesario mencionar que el diagrama de despliegue es un tipo especializado de modelado estructural, que apoya la representación del entorno de implementación (hardware, software) de un sistema de cualquier tipo (cliente, móvil, web o su combinación) y uso de tecnologías. A diferencia del modelo o diagrama de componentes de un sistema, un modelo de despliegue muestra los recursos externos que requieren esos componentes para cumplir su objetivo.

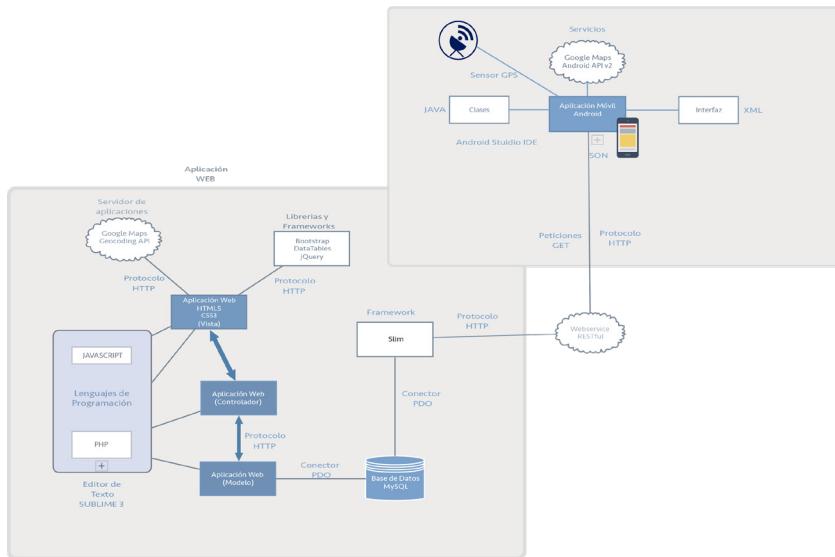


Figura 50. Forma de representar el despliegue de una aplicación

Por lo general, el diagrama de despliegue se utiliza durante la fase de diseño con el fin de determinar cómo las actividades de implementación haciendo uso de estándares y buenas prácticas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

permiten el cumplimiento de atributos de calidad como seguridad, rendimiento, mantenibilidad, usabilidad, entre otros atributos de calidad necesarios en los sistemas.

Una vez que por medio del estudio de las unidades de Primer y Segundo Bimestre asocia los conceptos con la práctica, diseñando los diagramas que son útiles para proponer una arquitectura candidata ante un problema que puede ser resuelto con software, es necesario estudiar los conceptos básicos de modelado arquitectónico, entre ellos constan: estilos arquitectónicos, patrones arquitectónicos y patrones de diseño principalmente.

## 8.2. Modelado arquitectónico

Para comprender el término modelado arquitectónico, empecemos haciendo una analogía con la construcción de un edificio, diseñar y ensamblar un auto u otro concepto donde a través de vistas lógicas (partes internas) y vistas físicas (aspectos externos y/o hardware) se expongan los elementos y su combinación para cumplir con el objetivo deseado dentro de un contexto. Hay que tener siempre presente que una gestión del proyecto, gestión de riesgos e inversión a utilizar (recursos, tiempo y dinero), serán necesarias siempre y cuando el objetivo para la construcción e implementación del producto software lo requieran.

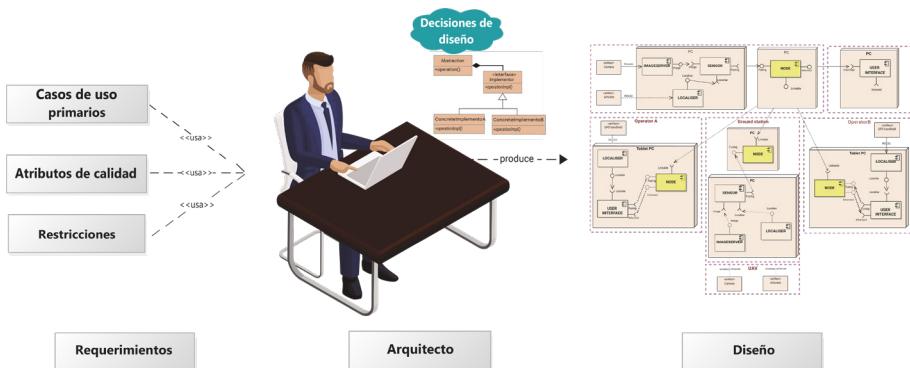


Figura 51. El arquitecto software toma como entrada los requerimientos que influyen en una arquitectura y producen un diseño arquitectónico

Considerando que un buen análisis es necesario para realizar el diseño e implementación, la arquitectura de un sistema abarca dimensiones como la arquitectura lógica y la arquitectura de despliegue:

- La arquitectura lógica (*logical architecture*), que describe el sistema en términos de su organización conceptual, de funcionamiento interno o interacción, para ello se soporta en diagramas ya conocidos como casos de uso, clases, paquetes, interfaces, subsistemas, marcos de trabajo. En su forma de representación similar a 3 “layers” o capas lógicas.
- La arquitectura de despliegue o física (*physical architecture*), que describe el sistema en términos de la asignación de procesos a las unidades hardware de procesamiento que funcionan localmente o en cloud y la configuración de la red (componentes, nodos, dispositivos, protocolos o interfaces de comunicación interna y externa del sistema), similar a la representación de “tiers” o capas físicas.

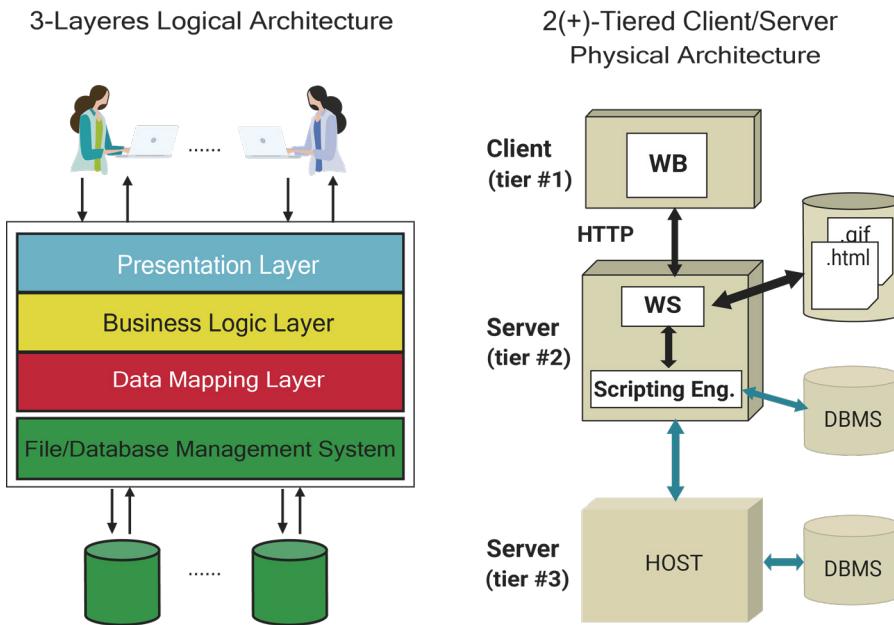


Figura 52. Arquitectura lógica vs Arquitectura Física.

El estudio y análisis de términos tales como arquitectura software, elementos arquitectónicos, patrones arquitectónicos, patrones de diseño, atributos de calidad y su combinación son fundamentales en este contexto, por lo que le animo a revisarlos en la bibliografía básica y complementaria.

### 8.2.1. Arquitectura software

La arquitectura software desempeña un papel esencial durante el ciclo de vida de desarrollo de software sirviendo como puente entre los **requisitos, derivados del análisis e implementación para proponer un diseño**. Dentro de los equipos de desarrollo de software, la arquitectura software se considera un medio de comunicación que permite representar en alto nivel la estructura interna/externa (*arquitectura lógica*) que debe ser implementada (*arquitectura física*) para el correcto funcionamiento del sistema.

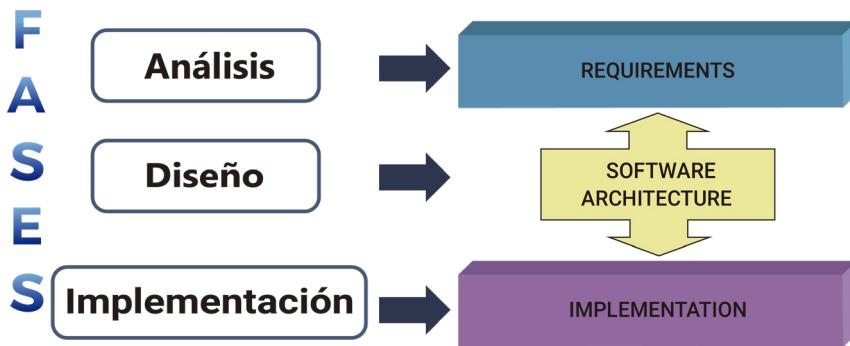


Figura 53. Fases necesarias para definir una arquitectura.



### Actividades de aprendizaje recomendadas



## Autoevaluación 8

1. ¿Cuáles de los siguientes elementos forman parte de un diagrama de despliegue? Seleccione las opciones de respuesta que apliquen.
  - a. Nodos que contienen componentes y/o artefactos.
  - b. Nodos que representan dispositivos hardware.
  - c. Artefactos.
  - d. Canales de Comunicación.
  
2. ¿Cuáles de los siguientes elementos se consideran parte de un Nodo? Seleccione las opciones de respuesta que apliquen.
  - a. Dispositivo Móvil (Android - IOS).
  - b. Laptop.
  - c. Servidor.
  - d. Cloud.
  
3. ¿Cuáles de las siguientes opciones se consideran parte de un artefacto? Seleccione las opciones de respuesta que apliquen.
  - a. Un programa con extensión .jar.
  - b. Un programa con extensión .exe.
  - c. Un programa con extensión .war.
  - d. Un archivo con extensión .ear.

4. Asocie el concepto al atributo de calidad que corresponda.  
“Evalúa la capacidad (en relación al funcionamiento y tiempo de ejecución) que tiene el software para usar recursos”.
- Rendimiento.
  - Confiabilidad.
  - Usabilidad.
  - Funcionalidad.
5. Asocie el concepto al atributo de calidad que corresponda:  
“Estima el esfuerzo que debe realizar el usuario para lograr adaptarse al software”.
- Rendimiento.
  - Confiabilidad.
  - Usabilidad.
  - Funcionalidad.
6. Asocie el concepto al atributo de calidad que corresponda:  
“Valora la capacidad y esfuerzo para realizar modificaciones o corregir errores en el software”.
- Rendimiento.
  - Confiabilidad.
  - Usabilidad.
  - Mantenibilidad.
7. En el contexto de Software, un Patrón se define como una regla que consta de 3 partes. ¿Cuáles son dichas partes?
- Sistema, Subsistema, Componente.
  - Contexto, Subsistema, Problema.
  - Contexto, Problema y Solución.
  - Contexto, Problema y Componente.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

8. ¿Es correcta o no la definición expuesta por Medvidovic y Taylor para Patrón Arquitectónico a la que definen como: “un conjunto de decisiones de diseño arquitectónico que se aplican a un problema de diseño recurrente y se parametrizan para tener en cuenta los diferentes contextos de desarrollo de software en los que ese problema aparece”?
  - a. Verdadero.
  - b. Falso.
9. Cuando se habla de las ventajas de documentar la arquitectura se plantea el hecho de que favorece la comunicación con los participantes ¿Por qué motivo sucede esto?
  - a. Porque es una presentación de alto nivel del sistema.
  - b. Porque se elabora es una etapa temprana del desarrollo.
  - c. Porque muestra cómo se organiza el sistema.
10. ¿Cuál de las siguientes características debe implementarse en la arquitectura del sistema para garantizar la disponibilidad?
  - a. Activos críticos del sistema protegidos.
  - b. Operaciones críticas del sistema implementadas y ubicadas en algún componente individual.
  - c. Incluir componentes redundantes

[Ir al solucionario](#)

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

## Lecturas recomendadas

- (Unidad 8.1) Para profundizar en el tema de los diagramas de despliegue en el texto básico revise los capítulos 27 y 31 correspondiente a **Despliegue y Diagrama de Despliegue**.
- (Unidad 8.1) Con el fin de identificar conceptos de arquitectura, diferencias y semejanzas con diagramas le invito a revisar en el texto básico el capítulo correspondiente a **Modelado Arquitectónico**.
- (Unidad 8.2) [Le invito a revisar el recurso educativo abierto, donde se exponen los diagramas que más se usan en un diseño arquitectónico, conceptos de estilos y patrones arquitectónicos.](#)
- (Unidad 8.2.1) [Le invito a revisar el recurso educativo abierto \(Ingeniería de Software\) en el Tema 5. Principios del diseño de software. En el documento revise las páginas 5, 6, 7, 12, 14 y 15.](#)

**Resultado de aprendizaje 2**

Identifica las partes esenciales de un sistema utilizando los diferentes tipos de modelado.

**Contenidos, recursos y actividades de aprendizaje****Semana 15****8.2.2. Estilos arquitectónicos**

La construcción de software depende en gran medida de una arquitectura software, que tenga como soporte estructural un estilo arquitectónico, patrones arquitectónicos y patrones de diseño los cuales se pueden aplicar de forma individual o combinada dentro de un contexto, utilizando un vocabulario formal de diseño. Según (Monroe, Kompanek, Melton, y Garlan, 1997) los estilos arquitectónicos determinan un conjunto de reglas de diseño que identifican tipos de componentes, conectores y patrones que se pueden utilizar para constituir un sistema o subsistema de software, junto con restricciones locales y globales de composición.

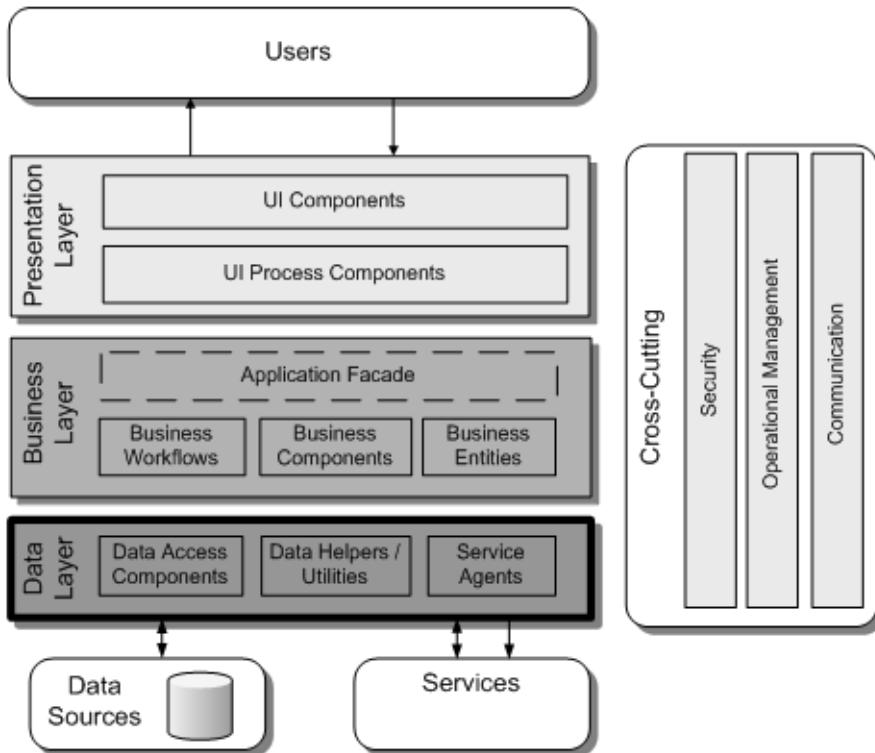


Figura 54. Arquitectura típica de una aplicación software. Fuente: [enlace web](#)

Los estilos arquitectónicos para Medvidovic y Taylor (2010), se consideran como una colección de decisiones de diseño arquitectónico que son aplicables en un contexto de desarrollo dado, limitan las decisiones de diseño que son específicas del sistema dentro de su contexto y recaban cualidades beneficiosas de cada sistema resultante.

Entre los beneficios que tiene el utilizar estilos arquitectónicos, Medvidovic y Taylor, (2010) exponen los siguientes:

- **Reutilización:** a nivel de diseño, soluciones bien entendidas aplicadas a nuevos problemas. A nivel de codificación, implementaciones compartidas de aspectos invariables de un estilo.

- Comprensión de la organización del sistema.
- **Interoperabilidad:** compatible con la estandarización de estilos.
- **Estilo-especificidad:** desde el punto de vista del análisis e implementación habilitado por el espacio de diseño restringido correspondiente a experiencias o implementaciones previas utilizando un estilo o su combinación.

Los estilos arquitectónicos como lo mencionan Klein et al. (1999) y Bass, Klein, y Bachmann (2000), se diferencian por su objetivo y características de diseño e implementación, permitiendo la creación de arquitecturas con propiedades diferentes asociando el estilo con atributos de calidad. Entre los tipos de estilos arquitectónicos constan:

- Estilos tradicionales influenciados por el lenguaje de programación (Procedural, orientado a objetos)
- Capas (Máquinas virtuales, cliente-servidor [*Client-server*])
- Flujo de datos (Procesamiento por lotes, tuberías y filtros [*Pipes and filters*])
- Memoria compartida (*Blackboard*, basado en reglas)
- Interpreter (Intérprete, código móvil)
- Invocación implícita (Basado en eventos [*Event-driven*], Llamada/Retorno [*Publish-subscribe*])
- *Peer-to-peer*
- Estilos derivados o Heterogéneos [*REST*]



Figura 55. Estilos arquitectónicos, una analogía.

### 8.2.3. Patrones arquitectónicos

El *Software Engineering Institute* (SEI), define un patrón arquitectónico como una descripción de elementos y tipos de relaciones junto con una serie de restricciones sobre cómo estos son usados. Para (Buschmann et al., 1996) un patrón se define como una regla que consta de 3 partes: **contexto, problema y solución**. En (Buschmann et al., 1996) se expone que un patrón arquitectónico expresa una organización o esquema estructural fundamental para los sistemas de software, debido a que presentan soluciones puntuales a problemas particulares o recurrentes que pueden ocurrir en un contexto de diseño específico, definiendo para ello estrategias de implementación que incluye componentes, conectores, reglas y pautas para organizar las relaciones entre ellos.

Un patrón arquitectónico se define como un conjunto de decisiones de diseño arquitectónico que se aplican a un problema de diseño recurrente, y se parametrizan para tener en cuenta los diferentes

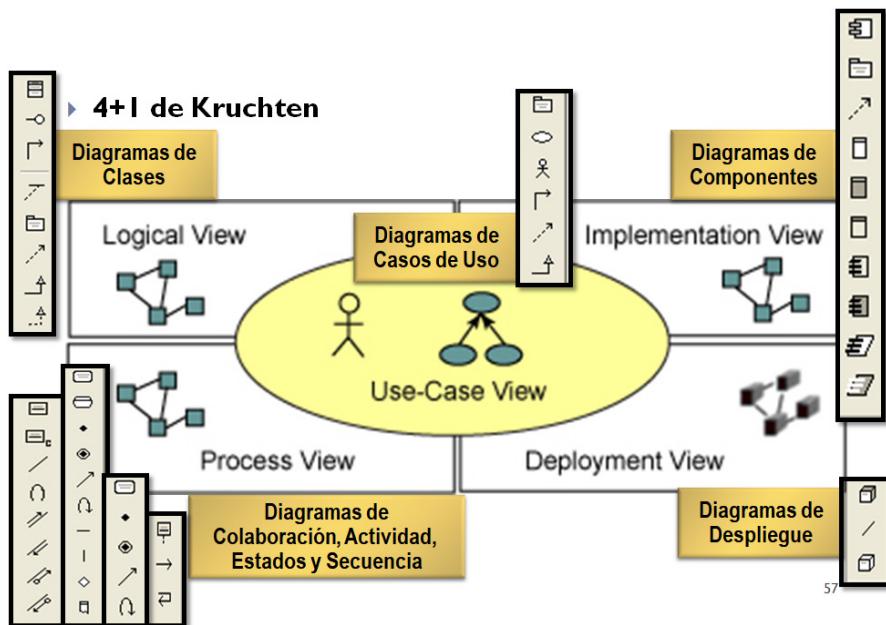


Figura 56. 4+1 View propuesto por Kruchten, diagramas y elementos UML usados.

El estándar ANSI/IEEE 1471-2000 (Hilliard, 2000) define vista, punto de vista y vistas arquitectónicas como:

- Vista: representación de un sistema completo desde la perspectiva de un conjunto relacionado de preocupaciones. Una vista conforma un punto de vista.
- Punto de vista: especificación de las convenciones para construir y usar una vista. Los tipos de elementos, las relaciones, así como la meta-information para describir la vista son descritas por los puntos de vista.
- Vistas arquitectónicas: representación de un sistema desde la perspectiva de un conjunto de asuntos de interés.

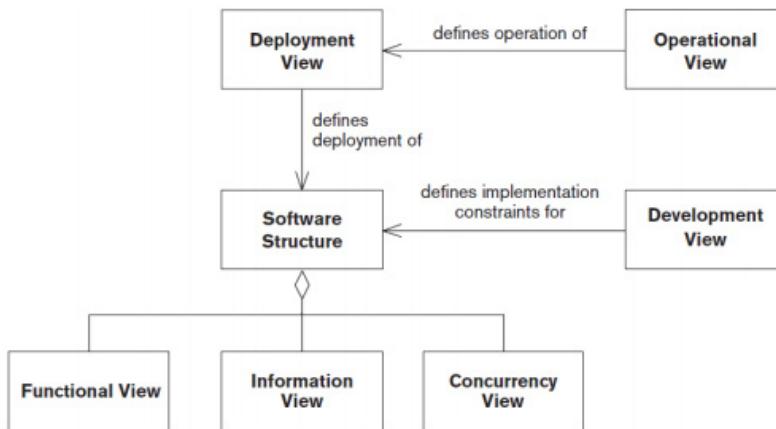


Figura 57. Estándar ANSI/IEEE 1471-2000 derivado del estándar ISO/IEC/IEEE 42010.

Entre los tipos de patrones arquitectónicos constan: Capas<sup>1</sup>, Blackboard<sup>2</sup>, Modelo-Vista-Controlador (MVC)<sup>3</sup>, Broker, Presentation-Abstraction-Control, Microkernel y Reflection (ver Figura 57)

<sup>1</sup> <https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/system-architecture/architectural-patterns>

<sup>2</sup> <http://hillside.net/plop/plop97/Proceedings/lalande.pdf>

<sup>3</sup> <https://msdn.microsoft.com/en-us/library/ff649643.aspx>

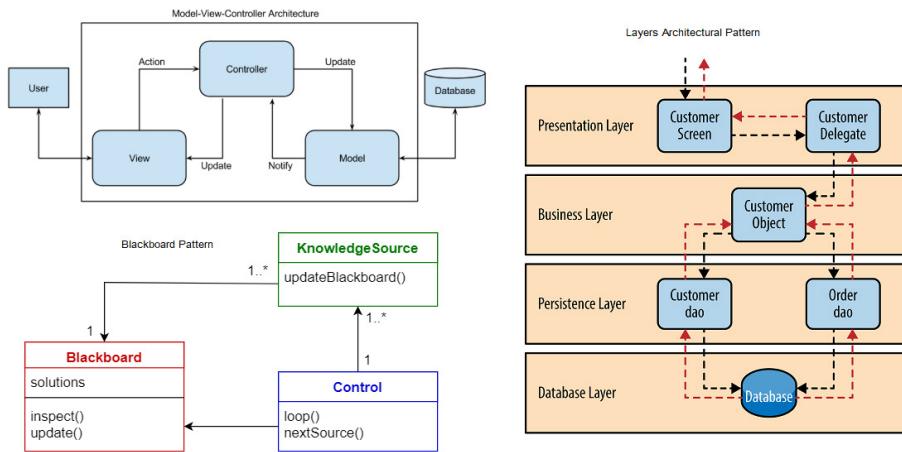


Figura 58. Representación de patrones arquitectónicos.

#### 8.2.4. Patrones de diseño

Según Tichy (1997), los patrones de diseño son utilizados por los ingenieros de software para el análisis, diseño e implementación de abstracciones específicas de datos, funciones e interconexiones que a través de un vocabulario y plantillas base de diseño permiten el cumplimiento de atributos de calidad y reutilización a nivel de implementación.

El mismo autor menciona que un patrón de diseño describe una familia de soluciones para un problema de diseño de software. Las soluciones involucran el uso de uno o varios elementos tales como interfaces, clases, objetos, métodos, funciones, procesos, hilos, relaciones entre los elementos como asociación, herencia, delegación, invocación, creación y una descripción de su comportamiento (Tichy, 1997).

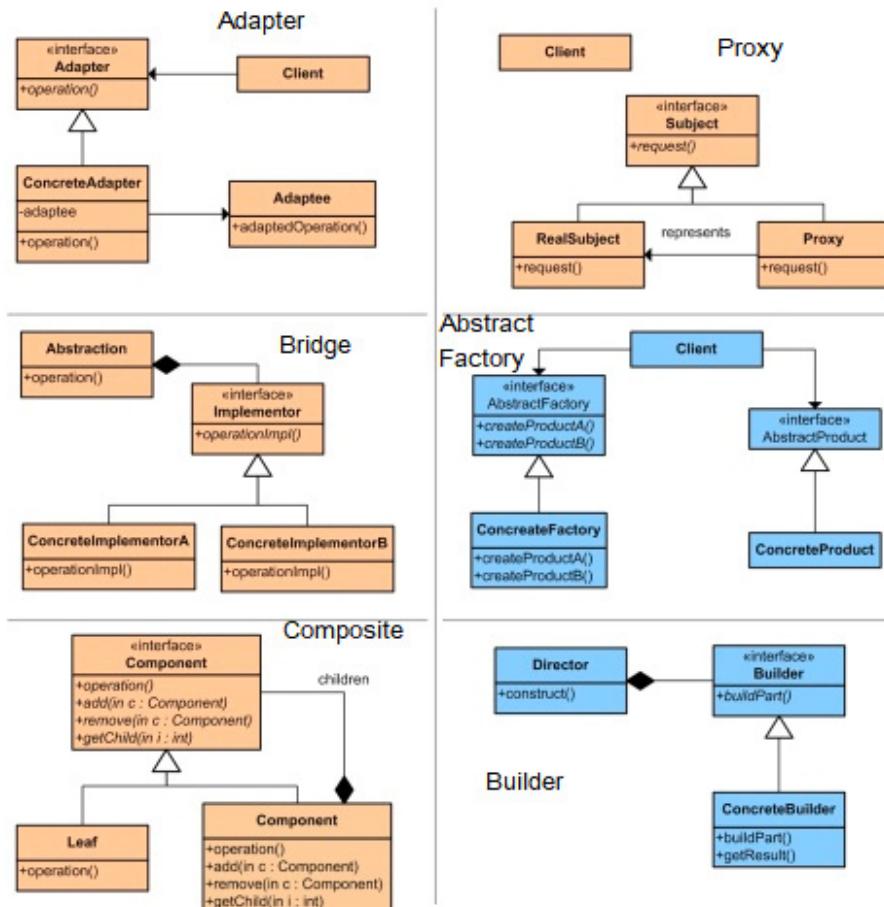


Figura 59. Patrones de diseño y su representación en UML

Acorde con Lea (1994), en su trabajo denominado “*Christopher Alexander: An introduction for object-oriented designers*”, expone una descripción de cualidades necesarias de los patrones de diseño, entre las que destacan:

- **Encapsulación y abstracción:** el patrón de diseño debe encapsular un problema y proveer una solución concreta dentro de un contexto particular.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

- **Extensión y variabilidad:** los patrones deben tener la capacidad de trabajar e interoperar con otros patrones para dar solución al problema.
- **Generación y composición:** cada patrón posterior a su aplicación genera un resultado, el cual debe coincidir con el contexto inicial de dicho patrón. Esta subsistencia de patrones podría ser aplicado progresivamente con el propósito de otorgar una solución integral a un problema.
- **Equilibrio:** cada patrón de diseño debe ofrecer un equilibrio entre las consecuencias y restricciones de su implementación para solucionar el problema dentro del contexto.

Los patrones de diseño se clasifican en tres grupos: (i) Creacionales, (ii) Estructurales, (iii) Comportamiento. Gamma, Helm, Johnson, y Vlissides (1994), en su libro conceptualizan, clasifican y representan de forma práctica los patrones de diseño. En la Tabla 8 se expone de forma resumida el propósito de cada patrón acorde a su tipo.

Tabla 6. Clasificación y especificación de Patrones de Diseño.

Tipo	Patrón de diseño	Propósito
Creacionales	Abstract Factory	Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.
	Builder	Separa la construcción de un objeto complejo de su representación de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
	Factory Method	Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
	Prototype	Especifica los tipos de objetos a crear por medio de una instancia prototípica y crea nuevos objetos copiando dicho prototipo.
	Singleton	Garantiza que una clase tenga solo una instancia proporcionando un punto de acceso global a ella.

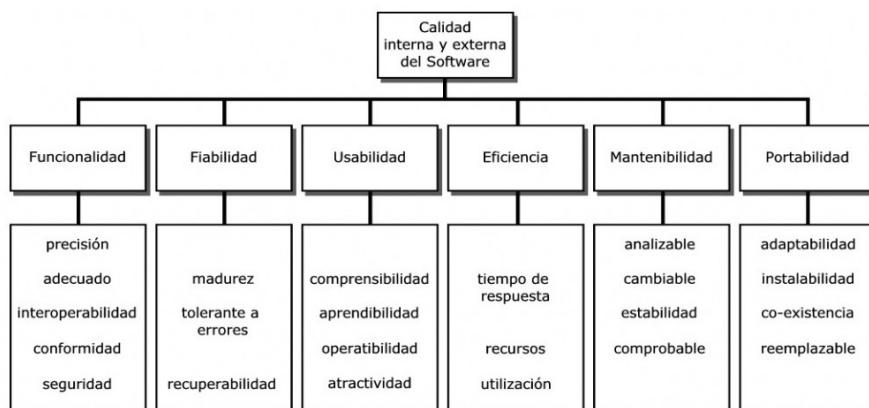
<b>Tipo</b>	<b>Patrón de diseño</b>	<b>Propósito</b>
Estructurales	Adapter	Convierte la interface de una clase en otra interface para que las clases cooperen de forma de que no existan clases con interfaces incompatibles.
	Bridge	Desacopla una abstracción de su implementación de modo que ambas puedan variar de forma independiente.
	Composite	Compone objetos en estructuras de árbol para su representación. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
	Decorator	Asigna responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
	Facade	Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.
Comportamiento	Command	Encapsula una petición en un objeto, permitiendo parametrizar a los clientes con diferentes peticiones, hacer cola o llevar un registro de las peticiones, y poder deshacer las operaciones.
	Interpreter	Define una representación de su gramática junto con un intérprete que usa la misma para interpretar sentencias del lenguaje.
	Iterator	Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
	Mediator	Define un objeto que encapsula la forma que interactúan una serie de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente y permite variar la interacción entre ellos de forma independiente.
	Observer	Define entre objetos una independencia de uno a muchos de forma que cuando un objeto cambie de estado se notifique y actualice automáticamente todos los objetos que dependan de él.
	Strategy	Define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

Fuente: adaptado de “Design patterns: elements of reusable object-oriented languages and systems” (Gamma, et. al., 1994).

Los patrones de diseño dependiendo del contexto y del problema a solucionar, sugieren una implementación (a través de lenguajes y tecnologías de programación) usando buenas prácticas de diseño y programación, con el fin de mejorar atributos de calidad, los mismos que son aplicados sobre distintos tipos de aplicaciones (cliente, web, móviles, servicios).

### 8.2.5. Atributos de calidad de software

Acorde a O'Brien, Merson, y Bass, (2007), la calidad del software es el grado con el que un sistema, componente o proceso cumple los requisitos y necesidades específicas del cliente, dichas necesidades exigen el establecimiento de métricas que permitan cuantificar y establecer niveles mínimos y máximos de calidad. Esto implica que para mejorar la calidad del software se debe poner atención a los atributos de calidad. Para M. Barbacci, Klein, Longstaff, y Weinstock, (1995), un atributo de calidad se define como la propiedad de un producto o servicio del cual su calidad será evaluada por la parte interesada. Para el SEI , los atributos de calidad tales como rendimiento, seguridad, capacidad de cambio, confiabilidad, usabilidad tienen influencia significativa en la arquitectura software.



Bass et al. (s/f) definen un atributo de calidad como una propiedad medible y comprobable de un sistema, que es usado para

indicar que tan bien el sistema satisface las necesidades de los interesados. Algunos de los atributos de calidad de software que los autores mencionan y se proponen en la norma ISO/IEC 9126-1 son: rendimiento, confiabilidad, usabilidad, funcionalidad, mantenibilidad y portabilidad los cuales se detallan brevemente a continuación:

- **Rendimiento:** evalúa la capacidad (en relación al funcionamiento y tiempo de ejecución) que tiene el software para usar recursos.
- **Confiabilidad:** capacidad del software de mantener su nivel de ejecución sobre condiciones regulares en un lapso determinado de tiempo, además precisa que los resultados de la funcionalidad de software sean exactos.
- **Usabilidad:** estima el esfuerzo que debe realizar el usuario para lograr adaptarse al software.
- **Funcionalidad:** determina la capacidad que tiene el software para cumplir los requisitos del cliente.
- **Mantenibilidad:** valora la capacidad y esfuerzo para realizar modificaciones o corregir errores en el software.
- **Portabilidad:** define la capacidad del software para ser transferido de un ambiente a otro, sin presentar problemas de despliegue.

M. Barbacci et al., (1995) en su trabajo exponen definiciones para atributos de calidad como Rendimiento, Confianza, Seguridad, Protección y su relación con la arquitectura y patrones utilizando para ello una taxonomía que permite identificar las preocupaciones desde el punto de vista de software. Los factores específicos de ciertos atributos, los métodos y técnicas que se utilizan para evaluar su cumplimiento a través de principios son expuestos por los mismos autores en M. R. Barbacci, Klein, y Weinstock (1997).



## Actividades de aprendizaje recomendadas

**Ejercicio 17.** Proponga un ejemplo en el cual se evidencie a través de representaciones gráficas y elementos UML, el uso de al menos 3 estilos arquitectónicos, 3 patrones arquitectónicos, 3 patrones de diseño (en un solo ejemplo todo lo solicitado o un ejemplo para cada uno).

**Ejercicio 18.** Proponga un ejemplo en el cual se evidencie a través de representaciones gráficas y elementos UML, el uso de un estilo arquitectónico que contenga un patrón arquitectónico y 3 patrones de diseño, similar al diseño de una arquitectura software.

## Bibliografía y lecturas recomendadas

- (Unidad 8.2.2) Una explicación acerca de los [estilos arquitectónicos](#) y [patrones arquitectónicos](#) se muestran en los enlaces propuestos. Le invito a revisar y complementar el estudio con la guía y el texto básico.
- (Unidad 8.2.3) Le invito a revisar en el texto básico el capítulo 29 relacionado a Patrones y Frameworks.
- (Unidad 8.2.3) [En este enlace](#) encontrará definiciones de [patrones arquitectónicos](#), características y su representación gráfica haciendo uso de elementos UML.
- (Unidad 8.2.3) [Una explicación breve y con representación gráfica de los principales patrones arquitectónicos](#) podrá encontrar en este enlace.

- (Unidad 8.2.4) Los patrones de diseño, una explicación textual y a través de UML lo puede revisar en el presente recurso educativo.
- (unidad 8.2.5) “Microsoft Application Architecture Guide” escrita por Meier et al. en relación con los atributos de calidad los define como factores generales que afectan el comportamiento del sistema en tiempo de ejecución, diseño y experiencia del usuario. Le invito a revisar estos y otros conceptos relacionados con arquitectura y patrones especialmente cuando lleve a cabo actividades de implementación.



### Actividades finales del bimestre



### Semana 16

¡Muy bien!, hemos completado el estudio en lo referente a la asignatura de Modelado dentro del presente periodo académico. Para apoyar a su comprensión y aprendizaje se han seleccionado e incluido una serie de conceptos básicos y necesarios para que sirvan de ayuda al momento de analizar, diseñar y proponer una solución que de forma textual y gráfica pueda ser presentada a los Stakeholders dependiendo del rol que desempeñe dentro de un Proyecto Software (Analista, Desarrollador, Arquitecto de Soluciones, Analista de Negocio u otro) para construir un Producto Software.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Se ha propuesto como herramienta CASE para modelado *StarUML*, donde los elementos UML son los mismos y mejorados en herramientas de pago tales como *Enterprise Architect*, *Rational Rose*, entre otras. Sin embargo, el conocer algunos tips, técnicas y buenas prácticas para modelado hará que su uso sea el adecuado al momento de representar gráficamente una solución a un problema previo a su implementación.

### **Actividad 1:**

Revise el resumen de los temas del segundo bimestre que se proponen en el recurso educativo abierto que tiene por nombre [Resumen 4+1, Estilos y Patrones](#)

### **Actividad 2**

Desarrolle el pequeño banco de preguntas que se cargará en la plataforma educativa para su auto estudio previo a la evaluación presencial.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



## 4. Solucionario

### Primer bimestre

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
1	d	Si revisa la definición de modelo, que es lo que en el presente curso usamos, lo que se modela siempre son los aspectos más importantes recabados siempre partiendo de un análisis previo lo cual le ayudan a representar textual o gráficamente los requisitos acordes a unas necesidades dadas por el cliente
2	a, b, c, d	Todas las opciones propuestas son correctas cuando vamos a modelar, es importante tener claro su uso, ventajas y desventajas.
3	c	El modelo que se usa para representar gráficamente una solución, siempre tiene asociado información semántica que puede ser usada por DSL, ATL, UML.
4	b	La información semántica es la que se representa a través del uso de modelos que contienen elementos que posteriormente serán estudiados y entendidos por los involucrados.
5	b	La calidad es un aspecto importante dentro del software, por lo que dentro de la vista de construcción (todas las fases del desarrollo) y desde la vista del producto (mantenimiento y despliegue) debe ser tomada en cuenta.
6	a, b, c, d	Los literales expuestos son tipos de sistemas, existen algunos otros pero cada uno cumple con su finalidad y propósito y ha sido construido siguiendo unos lineamientos de diseño y especificación para dar solución a problemas puntuales.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
7	a, b, c, d	Las aplicaciones Web son aquellas que por lo general se despliegan haciendo uso del internet. Las informativas se consideran aquellas que exponen datos e información resultante de un análisis previo, las interactivas se consideran aquella en la que se puede manipular los datos o información, las transaccionales se pueden considerar a las bancarias o donde existe un procesamiento y workflow donde se ve involucrado un proceso que pasado del manual al semiautomatizado apoya en la resolución de un problema.
8	b	Cuando se analiza las necesidades de los usuarios y se los transforma en requisitos, estamos entendiendo el problema, por lo que aún no tenemos claro una situación de diseño de solución final y solamente tenemos el Modelo del dominio del problema
9	a, b, c, d	Para entender el problema al cual debemos buscarle una solución, justamente se necesita de todas las opciones propuestas.
10	a, b, c, d	El dominio de la solución tiene en cuenta aspectos técnicos y de procedimientos (como los marcados en la opción A, B, C y D) a utilizar en la implementación de los modelos aprobados por los Stakeholders.

Ir a la  
autoevaluación

<b>Autoevaluación 2</b>		
<b>Pregunta</b>	<b>Respuesta</b>	<b>Retroalimentación</b>
1	a	El concepto de vista está relacionado al concepto del problema y la solución a aplicar, el concepto es tomado de forma literal de la bibliografía.
2	b	El punto de vista es como un problema o alternativa de solución que cada involucrado lo interpreta con su equipo de trabajo de forma diferente. Para ello hace uso de una o varias vistas.
3	b	Las convenciones para construir y usar una Vista se conocen como Punto de Vista.
4	b	El punto de vista estructural se usa en tiempo de diseño donde a través de herramientas CASE se representan los componentes y sus interacciones.
5	c	El punto de vista dinámico se usa para representar en tiempo la ejecución el funcionamiento de una solución implementada dentro de un sistema.
6	d	Toda implementación requiere del uso de recursos hardware y software para su funcionamiento, a ello le llamamos Punto de vista físico.
7	d	El modelo 4+1 de Krutchen se considera como un Framework o marco de referencia donde se distribuyen las vistas asociadas a una representación o uso que puede ser usado por los Stakeholders.
8	c	Las vistas que propone Krutchen y sus definiciones son Vista Lógica, Vista de Desarrollo, Vista de Proceso, Vista Física, Escenarios.
9	c	Si revisa cada una de las vistas propuestas por el modelo 4+1 observará que dentro de la vista de desarrollo se encuentran la de Componentes y Paquetes, estas justamente son usadas por los desarrolladores quienes implementan la solución.
10	a	Los diagramas que forman parte de la vista lógica son Clases y Objetos, resultantes del análisis y que sirven para ser usadas para la estructuración interna de la solución software.

[Ir a la autoevaluación](#)

<b>Autoevaluación 3</b>		
<b>Pregunta</b>	<b>Respuesta</b>	<b>Retroalimentación</b>
1	a, b, c, d	Para representar usando UML un caso de uso se debe hacer uso de elementos como actores, casos de uso, ámbito del sistema y relaciones
2	d	Los casos de uso son usados luego de que se ha analizado el problema y se recopila las necesidades y requisitos validados por los clientes, por ende, es una técnica para capturar información respecto de los servicios que proporciona a su entorno.
3	d	Otro de las características con respecto a los casos de uso es que se derivan de los requisitos sirven para determinar el alcance de una funcionalidad específica y sus relaciones.
4	b	Un proceso se considera como pasos a seguir, dentro del ámbito de software constan el conjunto de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos
5	d	Los modelos de desarrollo contienen fases, actividades y tareas que apoyan el desarrollo de software, algunas de ellas son las opciones propuestas.
6	d	Los procesos de desarrollo es algo mucho más amplio, estos contienen o definen un modelo de desarrollo. Entre las más conocidas constan las propuestas como opciones.
7	a, b, c, d	Una de las actividades del analista es recabar necesidades y documentar los requisitos de los clientes, por lo que entre las técnicas que se usan constan las propuestas.
8	b	Las fases del ciclo de vida de desarrollo de software como Análisis, Diseño, Codificación, Pruebas e Implementación son usadas por los modelos de desarrollo.
9	c	Las fases expuestas corresponden a Rational Unified Process (RUP)
10	a, b, c, d	El diagrama contiene errores desde su nomenclatura hasta su representación como se indica en todo los literales de respuesta.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 4		
Pregunta	Respuesta	Retroalimentación
1	a	Los diagramas de secuencia son usados para describir el flujo de mensajes, eventos y acciones entre objetos, por lo general son usados para representar una funcionalidad en concreto y dentro de los diagramas se deben realizar los más representativos.
2	d	Los diagramas de secuencia se documentan en las fases de análisis y diseño previo a una codificación, pruebas o implementación.
3	a	Cada diagrama UML tiene elementos principales que se usan para representar una posible solución a un problema, en el caso de los diagramas de secuencia se encuentran Participantes, mensajes, ejes (horizontal, vertical)
4	b	La vista dinámica se usa para representar una funcionalidad en tiempo de ejecución, por lo que los diagramas de actividad y diagramas de secuencia se usan en esta vista.
5	c	Los diagramas de actividades son usados para mostrar el flujo paso a paso de un proceso, flujo de control o flujo de datos.
6	d	Al conjunto de acciones, el flujo entre ellas y los valores producidos o consumidos se lo conoce como Actividad.
7	d	Las opciones referentes a los diagramas de actividad son algunas en las que puede verse involucrado su uso y representación.
8	d	Existen 3 formas que se pueden usar para representar los objetos dentro de un diagrama de secuencia, por definición se hace uso de estereotipos como los que se proponen en las opciones.
9	d	La definición que se propone como texto (mensaje) corresponde a uno de los elementos dentro de un diagrama de secuencia
10	a, b, c, d	Entre los elementos que se usan en un diagrama de secuencia se encuentran las líneas de vida de los objetos la misma que se usa para representar creación o destrucción de objetos, así como otras capacidades.

Ir a la  
autoevaluación



Autoevaluación 5		
Pregunta	Respuesta	Retroalimentación
1	b	A nivel de representación UML y a nivel de codificación, el concepto de clase abstracta tiene su propia representación. En UML se usa letra cursiva lo que permite cualquier herramienta CASE.
2	a, b, c, d	En los ejemplos se trata de representar una Asociación en los literales A, B, C, D. Más específicamente una agregación en el literal B y composición en el literal C
3	a, b, c	Es necesario conocer las asociaciones UML representadas como generalización (literal A), dependencia (literal B), y realización (literal C).
4	a, b, c	Dependiendo del elemento UML que se usa para representar una interface, las opciones que se usan para representar son A, B, C.
5	a, b, c	Algunos de los elementos UML que se usan en la representación de clases son multiplicidad, nombre de rol, asociación.
6	b	Cuando se relaciona dos o más clases lo que normalmente realiza el analista o quien modela las clases es leerlas, la lectura acorde al rol debe poder hacerse de izquierda a derecha, así como de derecha a izquierda.
7	c	Siempre que se modela hay que tener en cuenta el elemento UML a utilizar y en el caso de la asociación la dirección de la flecha desde la clase origen hacia el destino.
8	c	Recuerde que cuando se usa relaciones entre clases, elementos y sus relaciones lo que debemos aprender es a leer de izquierda a derecha como de derecha a izquierda.
9	c, d	Cuando se trabaja con clases, una interface solamente se usa para definir nombre, métodos, pero no los implementa, sino que la clase que hace llamado o uso de la interface las implementa.
10	d	La Ingeniería Directa corresponde a cuando desarrollamos un modelo y en base a este se procede a su codificación e implementación usando cualquier lenguaje y tecnología de programación.

<b>Autoevaluación 6</b>		
<b>Pregunta</b>	<b>Respuesta</b>	<b>Retroalimentación</b>
1	d	En la mayoría de bibliografía se puede observar que los tres compartimentos principales para representar una clase están distribuidos en el orden de Clase, atributos, métodos.
2	c	Acorde a las convenciones para representar y nombrar clases, la mejor opción para palabras combinadas es usar guion bajo o las palabras juntas siempre y cuando expresen en su totalidad lo que trata dicha clase.
3	d	Cuando se representa clases con UML hay que tratar de leer dichas clases, para ello el uso de las relaciones también tienen asociados conceptos o nomenclatura para expresar lo que trata la clase.
4	c	Cuando se requiere representar con UML una agregación, esta puede representarse usando interfaces y clases.
5	b, c, d	La idea de representar con UML el diseño de clases es que se visualicen los elementos a usar, en este caso el diagrama muestra agregación entre Paralelo y Componente, Dependencia entre la Interface IGenerar, Herencia dentro del paquete de Clases.
6	b	Una de las formas para representar una Interface es a través del símbolo y nomenclatura que en UML se puede usar.
7	d	En el diagrama UML que se muestra no existe una clase abstracta la que se representa con letra cursiva.
8	c	Cuando se representa a nivel de análisis la extracción de clases, se debe tener en cuenta su forma de escritura y representación, por lo que se usa la primera palabra en minúscula luego seguido por dos puntos y el tipo de dato de dicho atributo.
9	b, c, d	A nivel de UML no hay inconveniente de representar la multiplicidad, pero a nivel de codificación, se debe usar una estructura de dato como Arreglos, ArrayList o List que permiten el almacenamiento temporal de 1 o muchos datos.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

### Autoevaluación 6

Pregunta	Respuesta	Retroalimentación
10	a, b	Se plantea el escenario donde si relacionamos dos clases debemos tener un mensaje y rol para las clases, por lo que las opciones A y B serían las correctas. Recuerde, debemos modelar y la lectura debe ser el adecuado entre dichas clases.

Ir a la  
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 7		
Pregunta	Respuesta	Retroalimentación
1	a, b	En UML existe dos tipos de interface que se pueden representar estas son Provista y Requerida, cada una con significado y uso diferentes en implementación
2	c	Una interface en UML es usada netamente para representar su nombre y los métodos que implementará la clase que se relacione dicha interface.
3	b, c	Las interfaces implementan su lógica en la Clase que lo requiere, la interface puede tener métodos y por ende esta puede recibir uno o varios parámetros.
4	c, d	Realización y Dependencia son dos de las relaciones que a nivel de diseño con UML se usan para asociarse con Interfaces.
5	d	Todas las opciones que se presentan con respecto a los paquetes son correctas ya que se usan para representar que internamente existen clases y sus relaciones, usan estereotipos como import, Access y el símbolo de la flecha para asociar paquetes es a través de una línea punteada.
6	d	Para relacionar los componentes se puede hacer uso de asociación e interfaces provistas y requeridas.
7	a, b, c	Las tres opciones A, B y C son correctas respecto a los componentes, su definición y posible uso.
8	b	Por lo general haciendo uso de un sistema se registran los datos hacia una base de datos. Al registro como algo puntual se lo conoce como funcionalidad del sistema.
9	c	Un componente contiene a una o muchas funcionalidades de un sistema. Especialmente los sistemas realizan CRUD que significa Inserción, Lectura o búsqueda, Actualización y Borrado lógico o físico de los datos de una base de datos.
10	a, b	Con respecto a componentes, un componente a nivel de diseño contiene a uno o más componentes y a nivel de diseño y codificación contiene clases.

Ir a la  
autoevaluación



<b>Autoevaluación 8</b>		
<b>Pregunta</b>	<b>Respuesta</b>	<b>Retroalimentación</b>
1	a, b, c, d	Un diagrama de despliegue muestra como la aplicación se distribuirá físicamente para que los clientes puedan hacer uso de la misma, para ello se requiere de nodos que contienen componentes y/o artefactos, nodos que representan dispositivos hardware, artefactos y canales de comunicación.
2	a, b, c, d	Si revisamos la definición de nodo, este se considera todo dispositivo por lo general hardware que permite una interacción con los sistemas.
3	a, b, c, d	Si revisamos la definición de artefacto, todas las opciones propuestas se consideran como uno de ellos.
4	a	Los atributos de calidad se usan para mejorar la calidad en el software, el rendimiento está asociado al tiempo de respuesta por ejemplo cuando queremos consultar registros desde la base de datos.
5	c	Uno de los atributos que tiene relación con la facilidad de uso de un tipo de aplicación es Usabilidad, que al igual que los demás atributos de calidad es usada para mejorar la calidad en el software.
6	d	Todo sistema o aplicación debe ser mantenible lo que involucra adicionar nuevas funcionalidades y/o corregir bugs existentes, el atributo mantenibilidad es el que apoya en este objetivo.
7	c	En el contexto de Software un Patrón se define como una regla que consta de 3 partes conocidas como Contexto, Problema y Solución.
8	a	La respuesta es verdadera con respecto a la definición dada por los autores para patrón Arquitectónico
9	a	Una de las ventajas de documentar la arquitectura es Porque es una presentación formal de alto nivel del sistema.
10	c	Para garantizar la disponibilidad en una arquitectura software se debe considerar la inclusión de componentes redundantes los cuales permiten que su uso e implementación con otros servicios o componentes sea el adecuado.



## 5. Referencias bibliográficas

Ambler, S.W. (2005). *The Elements of UML™ 2.0 Style*. United States of America: Cambridge University Press.

Texto escrito en idioma inglés que proporciona ejemplos prácticos, consejos y buenas prácticas para modelado.

Blanco, C, Hernández, J., López, P, Ruiz, F. (2011). *Ingeniería de Software I*. Cantabria: OCW Universidad de Cantabria.  
Recuperado de <https://ocw.unican.es/course/view.php?id=169&section=2>

Recurso educativo abierto que contiene conceptos y ejemplos relacionados a Ingeniería de Software.

Bruegge, B., Dutoit, A. (2009). *Object-oriented Software Engineering: using UML, Patterns and Java*. Múnich, Alemania: Editorial Pearson Education.

Texto escrito en idioma inglés que presenta definiciones y ejemplos prácticos utilizando UML. Se expone con base en los casos de resolución y diseño de diagramas.

*Enterprise Architect* (<https://sparxsystems.com/>)

*Flowchart* (<https://www.draw.io/>)

Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. United States of America: Addison-Wesley Professional.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Explicación detallada de los temas de UML, sus conceptos y representación.

Guamán, D., (2018). *Guía didáctica de Modelado de Sistemas*. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

El propósito de esta guía es presentar una recopilación de temas que involucran aspectos de Modelado. La información presentada guiará a los estudiantes en la revisión esquematizada de los contenidos y complementará la bibliografía proporcionada.

Jacobson, I., Booch, G., y Rumbaugh, J., (2006). *El Lenguaje Unificado de Modelado*. (2<sup>a</sup> edición), México: Editorial Pearson Education.

Los autores del texto exponen conceptos, terminología, glosario de términos que son utilizados al momento de modelar con UML y su representación a través de diagramas que forman parte de vistas estructurales y dinámicas. Para el análisis y diseño de dichos diagramas en el texto se proponen técnicas y recomendaciones para su uso e implementación.

Larman, C. (2012). *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development*. India: Pearson Education India.

Artículo que expone la relación entre UML y los patrones de diseño y patrones arquitectónicos.

*LucidChart* (<https://www.lucidchart.com/pages/es>)

[Materiales de clase programa Ingeniería de Software]. (s/f).

Universidad de Salamanca Recuperado de <http://ocw.usal.es/ensenanzas-tecnicas/ingenieria-del-software/materiales-de-clase/>

Índice

Primer  
bimestre

Segundo  
bimestre

Solucionario

Referencias  
bibliográficas

Recurso educativo abierto que contiene conceptos de Análisis orientado a objetos, principios del diseño del software, diseño orientado a objetos.

Rumbaugh, J.; Jacobson, I. & Booch, G., (2004). *The Unified Modeling Language Reference Manual*, Boston, Massachusetts: Addison-Wesley.

Recurso escrito en inglés que hace referencia al uso de UML y su representación en diagramas.

Universidad de Salamanca. (s/f). *Ingeniería de Software*. Recuperado de <http://ocw.usal.es/ensenanzas-tecnicas/ingenieria-del-software/materiales-de-clase/>

Recurso educativo abierto que contiene conceptos de Análisis orientado a objetos, principios del diseño del software, diseño orientado a objetos.

### Enlaces web

The Unified Modeling Language, <https://www.uml-diagrams.org/>

StarUML 5.0 User Guide, [http://staruml.sourceforge.net/docs/user-guide\(en\)/toc.html](http://staruml.sourceforge.net/docs/user-guide(en)/toc.html)

StarUML (<http://staruml.io/>)

LucidChart (<https://www.lucidchart.com/pages/es>)

Flowchart (<https://www.draw.io/>)

VisualParadigmOnline (<https://online.visual-paradigm.com/es/diagrams/features/uml-tool/>)

Enterprise Architect (<https://sparxsystems.com/>)