Universidad de Talca
Facultad de Economía y Negocios.
Escuela de Ingeniería Informática Empresarial.

SYLLABUS MÓDULO ANÁLISIS Y DISEÑO DE SOFTWARE

Docente: Sergio Antonio Baltierra Valenzuela.

1. Identificación del módulo.

Nombre.	Detalle.
Nº de Créditos.	Trabajo en Clases: 2 bloques teóricos semanales. Trabajo en Laboratorio con Ayudantía: 2 bloques práctico semanal. Trabajo no Presencial: 2 bloques de trabajo semanal. Total Semanal: 6 bloques. Número de Semanas: 36 semanas. Total Módulo: 216 bloques. Total Créditos: 8 créditos ECTS.
Nivel	
Requisitos	Programación Orientada a Objetos.
Responsable de confección de syllabus.	Escuela de Ingeniería Informática Empresarial.
Docentes que imparten el módulo.	Sergio Antonio Baltierra Valenzuela.
Contribución a la Formación	Este Módulo, pertenece a la formación disciplinaria del plan de estudios de la carrera de Ingeniería en Informática Empresarial y se sitúa en el dominio del desarrollo de sistemas de información.

Subcompetencias a desarrollar	Capacidad para analizar y diseñar software complejos utilizando la ingeniería de software. Aplicar los diferentes modelos de arquitectura orientados a concebir diseños de solución eficientes y reutilizables. Identificar el ciclo de vida de desarrollo de un sistema de software y aplicar modelos de referencia en función del tipo de organización. Estimar los esfuerzos requeridos para el desarrollo de un sistema de software y transformarlo en requerimientos de recursos.

2. Unidades de aprendizaje.

2.1. Unidad I: Introducción: ¿Qué es un software y cómo se desarrolla?

Recorrido de Aprendizaje:	¿Qué es un software? Ingeniería de software. Proceso de software.
Capacidades a lograr:	Identificar que es un software y que etapas llevan a su desarrollo.
Estrategias y procedimientos metodológicos:	Se hará uso de los métodos de aprendizaje por recepción (para dar a conocer conceptos básicos) y el método de aprendizaje por descubrimiento (durante el desarrollo de un proyecto y tareas).
Productos:	Presentación de conceptos básicos de la ingeniería de software. Descripción del proyecto a concretar.

2.2. Unidad II: Análisis.

Recorrido de Aprendizaje:	Definición y alcances del proyecto de desarrollo de software. Requerimientos del software. • Requerimientos del stakeholder.. • Requerimientos del software. • Requerimientos de diseño. • Requerimientos funcionales. • Requerimientos no funcionales. • Otros tipos de requerimientos. Captura o toma de requerimientos. Herramientas para la captura de requerimientos. • Entrevista. Análisis de los requerimientos. • Descubrimiento de requerimientos. • Clasificación y organización de requerimientos. • Ordenación por prioridades y negociación de requerimientos. • Documentación de los requerimientos. Validación de requerimientos.
Capacidades a lograr:	Adquirir y desarrollar la capacidad de análisis.
Estrategias y procedimientos metodológicos:	Se hará uso de los métodos de aprendizaje por recepción (para dar a conocer conceptos básicos) y el método de aprendizaje de aprendizaje por descubrimiento (durante el desarrollo de un proyecto y tareas).
Productos:	Presentación de conceptos básicos de la ingeniería de software. Descripción del proyecto a concretar.

2.3. Unidad III: Diseño lógico.

Recorrido de Aprendizaje:	UML: Introducción. Diagrama de casos de uso. Diagrama de clases. Diagrama de paquetes. Diagrama de interacción. • Diagrama de Secuencia. • Diagrama de Colaboración. Diagrama de Estado. Diagrama de Actividades. Diagrama Físico. • Diagrama de Componentes. • Diagrama de Implantación.
Capacidades a lograr:	Entender la importancia de los lenguajes de modelado y su estandarización. Documentar el proceso de diseño. Capacidad de diseñar software de nivel específico al genérico. Entender el problema a resolver.
Estrategias y procedimientos metodológicos:	Se hará uso de los métodos de aprendizaje por recepción (para dar a conocer conceptos básicos) y el método de aprendizaje por descubrimiento (durante el desarrollo de un proyecto y tareas).

Productos:

En base al proyecto de desarrollo de software modelar el análisis utilizando UML.

Aplicar una herramienta de apoyo para la diagramación y documentación.

Exposición y defensa del modelado.

2.4. Unidad IV: Arquitectura de Software (Diseño Arquitectónico).

Recorrido de Aprendizaje:	Introducción. Estilos arquitectónicos. Patrones Arquitectónicos. UML para el diseño arquitectónico. Tipos de arquitectura de software: • Modelo Cliente Servidor (N capas). • Modelo Vista Controlador. • Modelo de software intermedio. • Modelo Orientado a Servicios (SOA). • Modelo Dirigido por Eventos (EDA).
Capacidades a lograr:	Entender la importancia de arquitectura a la hora del diseño de software. Usar herramientas para el diseño arquitectónico. Identificar y aplicar correctamente los patrones de arquitectura y de Software. Conocer la arquitectura de software que actualmente se usan en el mercado.

Estrategias y procedimientos metodológicos:	Se hará uso de los métodos de aprendizaje por recepción (para dar a conocer conceptos básicos) y el método de aprendizaje por descubrimiento (durante el desarrollo de un proyecto y tareas).
Productos:	Establecer la arquitectura de software óptima para el diseño del proyecto de desarrollo de software. Usar herramientas de diseño arquitectónico. Documentación, exposición y defensa del modelado.

2.5. Unidad V: Ciclo de Vida del Desarrollo del Software.

Recorrido de Aprendizaje:	Introducción. Roles de un equipo de desarrollo de software. • Administrador del proyecto. • Analista. • Diseñadores. • Programadores. • Téster. • Documentador. Modelos de Desarrollo del Software. • Modelo cascada. • Modelo iterativo e incremental. • Modelo ágil.
Capacidades a lograr:	Saber qué modelo de desarrollo de software es útil en un proyecto. Saber qué roles usar en un ciclo de vida de desarrollo de software.

Estrategias y procedimientos metodológicos:	Se hará uso de los métodos de aprendizaje por recepción (para dar a conocer conceptos básicos) y el método de aprendizaje por descubrimiento (durante el desarrollo de un proyecto y tareas).
Productos:	Establecer un modelo para el proyecto de software. Establecer roles en el proyecto de desarrollo de software. Documentación, exposición y defensa del ciclo de vida de desarrollo de software escogido.

2.6. Unidad VI: Estimación del esfuerzo y costo para un proyecto de desarrollo de software.

Recorrido de Aprendizaje:

Introducción.

Técnicas de estimación.

Recursos

- Recursos humanos.
- Recursos de software y hardware.

Técnicas de descomposición

- Dimensionamiento del problema.
- Estimación basado en problema.
- Estimación basada en proceso.
- Estimación con casos de uso.

Modelos algoritmos de costes.

- El modelo de COCOMO.
- Modelos algorítmicos de costes en la planificación.

La decisión de comprar o desarrollar.

Capacidades a lograr:	Saber cuánto cobrar por el desarrollo de un software. Saber cuánto cuesta desarrollar un software. Saber qué recursos se necesita para desarrollar un software.
Estrategias y procedimientos metodológicos:	Se hará uso de los métodos de aprendizaje por recepción (para dar a conocer conceptos básicos) y el método de aprendizaje por descubrimiento (durante el desarrollo de un proyecto y tareas).
Productos:	Establecer el costo en moneda nacional del proyecto de desarrollo de software. Establecer cuánto cobrar en moneda nacional por un proyecto de desarrollo de software.

3. Definición de metodología, evaluación, exigencias y bibliografía.

Metodología.

Clases Expositivas Participativas.

Las clases serán de carácter expositivas y participativas, en las que el docente hará una exposición inicial respecto del objetivo de la clase correspondiente y de los conceptos asociados, posteriormente los alumnos en grupos de trabajo constituidos por el docente, trabajarán en los avances de proyecto que el docente proponga y/o que emerjan de las vivencias de los propios alumnos.

Laboratorios en Universidad.

Las ayudantías se llevarán a cabo en los laboratorios de la carrera bajo la supervisión de los ayudantes, donde se aplicarán los conceptos y casos desarrollados en clase con el apoyo de herramientas de software.

Evaluaciones.

Se realizarán 4 pruebas teóricas, varias tareas y 6 avances de proyecto.

Avances de proyecto. Se exigirán informes y exposiciones de avance de proyecto que desarrollen los alumnos, que permitan verificar el logro de las capacidades comprometidas en las distintas unidades de aprendizaje, donde el expositor será un miembro del grupo a elección del docente.

Evaluaciones.	Componentes y procedimientos evaluativos del módulo (ponderaciones). **Evaluaciones.** Prueba parcial 1: 15 %. Prueba parcial 2: 15 %. Prueba parcial 3: 15 %. Prueba parcial 4: 15 %. Promedio de tareas: 10 %. Promedio de avances de proyectos: 30 %. Donde: Las tareas se promedian y del resultado se extrea el 10 % para la nota final (cantidad de tareas será según el avance del curso). Los seis avances de proyectos se promedian y del resultado se extrae el 30 % para la nota final. Por lo tanto la nota final se calcula de la siguiente manera: $$\text{Nota Final} = PP1 * 0{,}15 + PP2 * 0{,}15 + PP3 * 0{,}15 + PP4 * 0{,}15 + \bar{x}\ \text{Tareas} * 0{,}1 + \bar{x}\ \text{Avances de proyectos.} * 0{,}3$$
Exigencias.	Se exige la aprobación del módulo con nota 4.0 o más en la nota final; en caso contrario (reprobación) existirá una instancia de recuperación (Acumulativa Opcional) que incluirá toda la materia del curso, donde la nota de presentación (nota final) es del 70 % y la prueba Acumulativa Opcional es del 30 %. Para esta prueba se exigirá nota mínima según reglamento del estudiante. También se puede faltar a una prueba durante el curso, donde la prueba de recuperación será la misma Acumulativa Opcional. Para las otras inasistencias se evaluará con la nota mínima.

Bibliografía.

LIBRO GUIA

R. Pressman, Ingeniería del Software: Un enfoque práctico, vol. 6. McGraw-Hill, 2005.

I. Sommerville, Ingeniería del Software, vol. 7. Pearson Educación S.A., 2005.

MATERIAL DE REFERENCIA

K. Kendall and J. Kendall, Análisis y diseño de sistemas., vol. 8. Pearson Educación S.A., 2011.

P. Deitel and H. Deitel, Cómo programar en java., vol. 7. Pearson Educación, 2008.

C. Larman, UML y patrones: Introducción al análisis y diseño orientado a objetos. Pearson Educación, 1999.

M. Fowler and K. Scott, UML gota a gota. Pearson Educación, 2000.

J. Pavón Mestras, "El patrón modelo-vista-controlador (mvc)," 2008-2009.

R. Alarcón, Diseño orientado a objetos con UML. Grupo Eidos., 2000.

R. Pressman, Ingeniería del software un enfoque práctico., vol. 5. McGraw-Hill, 2001.

"Real time intelligence complex event processing." http://www.complexevents.com/. 166

A. Lozano Tello, Iniciación a la programación utilizando lenguajes visuales orientados a eventos. Ed.Bellisco Ediciones Técnicas y Científicas., 2001.

D. Fuller Padilla, "Apuntes de taller de ingeniería de software", 2003.

L. Guerrero, "Rational unified process.."

J. Palacio, Gestión de Proyectos Scrum Manager., vol. 2.5.1. Scrum Manager., 2015.