Asignatura	**Ingeniería del Software II**
Código	**IS884**
Créditos	**4**
Intensidad semanal	**6 Horas**
Requisitos	**IS714 Ingeniería del Software I**

Justificación	Después de establecer los requerimientos de un producto de software, es necesario plantear el sistema solución, el cual debe cumplir con todos los requerimientos. El diseño es una labor de gran responsabilidad y requiere de conocimientos en muchos temas de la informática como las comunicaciones, el hardware, sistemas operativos, bases de datos, lenguajes y herramientas de desarrollo de software, arquitecturas de software, patrones de diseño, etc. El gobierno nacional, por medio del ICFES y de COLCIENCIAS ha expresado su interés en fomentar el tema del diseño en ingeniería, pues la falta de desarrollo de este conocimiento en el país se considera una falencia. Luego de desarrollado un producto, sigue la etapa de mantenimiento que es responsable de mantener el producto operando por varios años. Es un área importante para los usuarios de software y se requiere que el ingeniero de sistemas y computación sea competente en ella.
Competencias Previas	• Conocer el SWEBOK. • Conocer sobre levantamiento de requerimientos. • Conocer sobre especificación de requerimientos (casos de uso). • Conocer sobre modelamiento UML de los requerimientos funcionales.

	• Conocer sobre requerimientos no funcionales • Conocer sobre Teoría General de Sistemas • Conocer las tecnologías que se usan en la construcción de sistemas de software: hardware, comunicaciones, bases de datos, lenguajes de programación, tecnologías WEB. • Competencia para leer documentos sobre esta asignatura en inglés.
Objetivo general	Desarrollar las habilidades y competencias en los estudiantes relacionadas con el diseño de software. Desarrollar habilidades y competencias en los estudiantes relacionadas con el mantenimiento de software.
Objetivos Específicos	1. El estudiante conocerá las arquitecturas tradicionales y sus atributos. 2. El estudiante conocerá los enfoques arquitectónicos para lograr los diferentes requerimientos no funcionales que se suelen solicitar. 3. El estudiante obtendrá la capacidad de plantear la arquitectura correcta. 4. El estudiante adquirirá la capacidad de modelar el diseño detallado y documentarlo para que pueda ser construído por los implementadores. 5. El estudiante estará en capacidad de plantear estrategias para el mantenimiento del software, especialmente de mantenimiento preventivo.
Metodología	El profesor orientará la asignatura con clases magistrales y planteará lecturas y temas de investigación. Se discutirán los documentos de apoyo, lecturas planteadas y temas de investigación en clase. Se realizarán tres grandes evaluaciones: Parcial 1, Parcial 2 y Final. Cada una de estas evaluaciones se puede dividir en varias evaluaciones de menor valor (quices, trabajos, exposiciones, etc.)

	teniendo en cuenta la autonomía de cada profesor. ESTRATEGIAS DE APRENDIZAJE: Constructivismo: Se construirán nuevos conocimientos teniendo como base los conocimientos que el estudiante ya ha construido. Aprendizaje Significativo: Se hará énfasis en los conocimientos más significativos e importantes.
Competencias Genéricas	Requeridas: - Lectura en Inglés. - Comprensión de lectura. - Redacción. - Pensamiento sistémico. A adquirir: - Capacidad de Síntesis. - Comprensión del proceso de la ingeniería. - Aplicación de conocimientos científicos y técnicos a la resolución de problemas.
Competencias específicas	REQUERIDAS: Cognitivas: 1. Inglés. 2. Programación de computadores. 3. Comunicaciones y redes de datos. 4. Bases de datos. 5. Sistemas operativos. 6. Herramientas de desarrollo IDE. (Ambientes Integrados de Desarrollo).

	7. Cómo funcionan las empresas, sus procesos y necesidades. 8. Probabilidades. Procedimentales/Instrumentales: 1. Modelamiento Orientado a Objetos. 2. Proceso de Requerimientos de Software. 3. Topologías y tecnologías de red. 4. Conceptos de Internet y Web. Actitudinales: 1. Pensamiento lógico matemático para afrontar los problemas. 2. Actitud creativa, curiosidad por explorar nuevas posibilidades. 3. Método científico. A ADQUIRIR: Cognitivas: 1. Conocer muchas arquitecturas, sus ventajas y desventajas. 2. Conocer las arquitecturas necesarias para obtener requerimientos no funcionales de desempeño, escalabilidad, seguridad, comunicabilidad, entre otros. 3. Conocer los conceptos fundamentales que dirigen el diseño. Procedimentales/Instrumentales: 1. Modelamiento UML del diseño: Subsistemas, paquetes, componentes, interfaces hombre-máquina.

	2. Metodología de diseño ADD (Attribute Driven Design) del SEI. 3. Implementación de un modelo OO mediante bases de datos. Actitudinales: 1. Pensamiento sistémico para crear arquitecturas que cumplan requerimientos. 2. Aplicación del pensamiento probabilístico al diseño. 3. Enfoque hacia la calidad del diseño.

Contenido de la asignatura	
Unidad 1	**INTRODUCCIÓN AL DISEÑO** 1. ¿Qué es diseño? 1.1 Conceptos de TGS 1.2 El problema del análisis 1.3 El problema de la síntesis 1.4 El problema de la caja negra 1.5 Los requerimientos no funcionales y su papel 2 Conceptos fundamentales del diseño 2.1 Modularidad 2.2 Abstracción 2.3 Acoplamiento 2.4 Cohesión 2.5 Ocultamiento de información
Unidad 2	**DISEÑO ARQUITECTÓNICO** 1 Decisiones del diseño arquitectónico 2 Vistas arquitectónicas 3 Patrones arquitectónicos 3.1 MVC 3.2 Capas 3.3 Repositorio centralizado

	3.4 Cliente-servidor 3.5 Tuberías y filtros 4 Arquitecturas de aplicación 4.1 Procesamiento de transacciones 4.2 Sistemas de información 4.3 Procesamiento de lenguaje 5 Modelamiento UML de los sistemas/subsistemas
Unidad 3	**INGENIERÍA DEL SOFTWARE BASADA EN COMPONENTES CBSE** 1 Introducción 2 Modelos de componentes 3 Proceso de la CBSE 4 CBSE y reutilización 5 Composición de componentes 6 Modelamiento UML de los componentes
Unidad 4	**REUTILIZACIÓN DE SOFTWARE** 1 Introducción 2 Panorama de la reutilización 3 Frameworks de aplicación 4 Líneas de productos de software
Unidad 5	**INGENIERÍA DEL SOFTWARE DISTRIBUÍDO** 1 Introducción 2 Conflictos en los sistemas distribuídos 2.1 Modelos de interacción 2.2 Middleware 3 Computación Cliente-Servidor 4 Patrones arquitectónicos para sistemas distribuídos 4.1 Arquitecturas maestro-esclavo 4.2 Arquitecturas cliente-servidor de dos niveles 4.3 Arquitecturas cliente-servidor multinivel 4.4 Arquitecturas de componentes distribuídos 4.5 Arquitecturas peer-to-peer 5 Modelamiento UML de arquitecturas distribuídas 6 El Software como servicio
Unidad 6	**ARQUITECTURAS ORIENTADAS A SERVICIOS** 1 Introducción 2 Servicios como componentes de reutilización

	3 Ingeniería de servicio 3.1 Identificación de candidatos a servicio 3.2 Diseño de interfaces del servicio 3.3 Implementación y despliegue del servicio 3.4 Servicios de sistemas heredados 4 Desarrollo de software con servicios 4.1 Diseño e implementación del flujo de trabajo 4.2 Pruebas del servicio
Unidad 7	**DISEÑO DE INTERFACES DE USUARIO** 1 Factores humanos 2 Principios de diseño 2.1 Familiaridad del usuario 2.2 Consistencia 2.3 Sorpresa mínima 2.4 Recuperabilidad 2.5 Guía al usuario 2.6 Diversidad de usuarios 3 Temas de diseño 3.1 Interacción con el usuario 3.1.1 Manipulación directa 3.1.2 Selección de menú 3.1.3 Llenar formulario 3.1.4 Lenguaje de comandos 3.1.5 Lenguaje natural 3.2 Ventajas y desventajas de los estilos 4 Presentación de la información 5 Proceso de diseño de interfaces de usuario 5.1 Análisis de usuarios 5.2 Prototipado de interfaces de usuario 5.3 Evaluación de la interfaz
Unidad 8	**MANTENIMIENTO DE SOFTWARE** 1 Razones para realizar el mantenimiento 2 La garantía del software 3 El mantenimiento correctivo 4 El mantenimiento preventivo (Refactoring)

Texto Guía	SOM11

Referencia	***Bibliografía***
SOM11	Sommerville, Ian. Software Engineering. 9ª Ed. 2011. Pearson Education. ISBN 978-607-32-0603-7.
SOM06	Sommerville, Ian. Software Engineering. 8ª Ed. 2006. Pearson Education. ISBN 978-0-321-31379-9.
PRE01	Pressman, Roger. Software Engineerion: A Practitioner's Approach. 5ª Ed. Mc Graw Hill Higher Education. 2001. ISBN 0-07-365578-3.
BOO07	Booch, Grady et Al. Object Oriented Analysis and Design with Applications. 3ª Ed. Addison Wesley. 2007. ISBN 0-201-89551-X.
GOR06	Gorton, Ian. Essential Software Architecture. 2006. Springer Verlag Berlin Heidelberg. 2006. ISBN 973-3-540-28713-1.
FOW02	Fowler, Martin. Refactoring: Improving the Design of Existing Code. Addison Wesley. 2002. ISBN 0-201-48567-2.