	FORMATO DE SYLLABUS	Código: AA-FR-003	
	Macroproceso: Direccionamiento Estratégico	Versión: 01	
	Proceso: Autoevaluación y Acreditación	Fecha de Aprobación: 27/07/2023	

FACULTAD:		**FACULTAD DE INGENIERÍA**						
PROYECTO CURRICULAR:		Maestría en Ciencias de la Información y las Comunicaciones				**CÓDIGO PLAN DE ESTUDIOS:**		
I. IDENTIFICACIÓN DEL ESPACIO ACADÉMICO								
NOMBRE DEL ESPACIO ACADÉMICO: INGENIERÍA DE SOFTWARE I								
Código del espacio académico:		79501006	Número de créditos académicos:				4	
Distribución horas de trabajo:		HTD	48	HTC	16	HTA	128	
Tipo de espacio académico:		Asignatura	X	Cátedra				
NATURALEZA DEL ESPACIO ACADÉMICO:								
Obligatorio Básico	X	Obligatorio Complementario		Electivo Intrínseco		Electivo Extrínseco		
CARÁCTER DEL ESPACIO ACADÉMICO:								
Teórico		Práctico		Teórico-Práctico	X	Otros:		Cuál:_____
MODALIDAD DE OFERTA DEL ESPACIO ACADÉMICO:								
Presencial	X	Presencial con incorporación de TIC		Virtual		Otros:		Cuál:_____

II. SUGERENCIAS DE SABERES Y CONOCIMIENTOS PREVIOS

Para el buen desarrollo del curso se considera necesario que el estudiante tenga amplios conocimientos en informática, sistemas de información, bases de datos y programación entre otros.

III. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

La ingeniera de software, como disciplina que ha alcanzado un grado de madurez importante, brinda un cuerpo de conocimiento sólido para plantear y resolver problemas sobre sistemas computacionales, estudiar las diferentes áreas de este cuerpo de conocimientos fundamental para encarar formalmente problemas que impliquen manejo de información. En este orden de ideas, conceptos como: procesos de software, metodologías, ingeniería de requerimientos y diseño son fundamentales para encarar proyectos de software en sus fases iniciales.

En Ingeniería de Software I, se busca abordar los problemas antes mencionados, de modo que se permita además crear una línea que continúa en patrones y arquitecturas de software, con lo cual se busca cubrir el espectro de conocimiento del área por lo menos en sus conceptos más relevantes.

IV. OBJETIVOS DEL ESPACIO ACADÉMICO (GENERAL Y ESPECÍFICOS)

OBJETIVO GENERAL: Brindar las herramientas que permitan abordar proyectos de software, especialmente en sus etapas iniciales, esto a través de diferentes procesos, metodologías y modelos en los cuales se enfatizará especialmente en la fase de requerimientos y diseño. un espacio de prospectiva que permita realizar una continua revisión de áreas en la disciplina Ingeniería de Software.

OBJETIVOS ESPECÍFICOS:

• Proporcionar herramientas conceptuales alrededor de la importancia del desarrollo del software como resultado de una serie de transformaciones sociales y culturales que han impactado la sociedad.

• Propiciar el uso adecuado de estándares, técnicas y herramientas de modelado que permitan en el trabajo eficaz y la capacidad de analizarlos, de evaluarlos y de proponerlos, como alternativas de solución.

• Proporcionar los métodos enfocados a la elección de requerimientos que faciliten esta fase dentro del desarrollo del software y que permitan hacer una especificación confiable

• Proporcionar los métodos enfocados a la elección de requerimientos que faciliten esta fase dentro del desarrollo del software y que permiten hacer una especificación confiable.
• Realizar modelos conceptuales que puedan ser llevados a artefactos principalmente modelados en UML.

V. PROPÓSITOS DE FORMACIÓN Y DE APRENDIZAJE (PFA) DEL ESPACIO ACADÉMICO

Competencias	Dominio-Nivel	RA	Resultados de Aprendizaje
Generales	Reconocer el cuerpo de conocimiento de la ingeniería de software	RA01	
	Realizar reflexiones sobre las temáticas referidas a la ingeniería de software.	RA01	
Cognitivas (Saber)	Capacidad de discernir que conocimientos y herramientas tecnologías debe apropiar para la resolución de problemas particulares en el desarrollo del software.	RA02	
	Representar soluciones de problemas aplicando el modelamiento del mismo mediante la abstracción de lógica del negocio.	RA02	
	Identificar los diversos componentes de un sistema de información para distinguir el ámbito de su solución.	RA02	
Procedimentales / Instrumentales (Saber hacer)	Modelar, implementar y evaluar problemas cuya solución requiere el uso de las diferentes estructuras que maneja la ingeniería del software.	RA03	
	Modelar, implementar y evaluar problemas descomponiéndolos en subproblemas que permitan	RA03	
	Modelar, implementar y evaluar mecanismos para el manejo arquitectura del software estándares, métricas entre otras.	RA03	
	Situar históricamente los diferentes momentos en la evolución de la ingeniería del software, arquitectura, patrones, métricas, estándares entre otras.	RA03	
	Utilizar los sistemas computacionales como herramienta de posibles soluciones a problemas específicos.	RA03	
Actitudinales (Ser)	Actuar estratégicamente dentro de un grupo de trabajo para el desarrollo de proyectos	RA04	
	Actuar éticamente comprometido con el desarrollo de las actividades de la asignatura.	RA04	
	Comunicarse estratégicamente haciendo uso de la tecnología.	RA04	
	Actuar en contextos académicos y profesionales con un enfoque culto, ético y humanístico.	RA04	
	Interpretar la realidad y proponer nuevos argumentos para desarrollar soluciones innovadoras en contextos sociales	RA04	
	Presentar los trabajos de forma estética, ergonómica y conforme al contexto sociocultural al cual se destinan.	RA04	

VI. CONTENIDOS TEMÁTICOS

PROGRAMA SINTÉTICO

CAPÍTULO 1 Introducción la Ingeniería del Software
CAPÍTULO 2 Modelos de Proceso Software Tradicionales y estándares, Swebok
CAPÍTULO 3 Modelos de Proceso Software Modernos
CAPÍTULO 4 Introducción a los Requerimientos
CAPÍTULO 5 Proyectos de Software las 4p del se
CAPÍTULO 6 Metodologías de Desarrollo de Software

CAPÍTULO 6 Metodologías de Desarrollo de Software

CONTENIDO DETALLADO:

CAPÍTULO 1 Introducción la Ingeniería del Software

• Presenta una panorámica de la evolución del software y las principales características que este posee. Adicionalmente realiza una Introducción al concepto de Ingeniería del Software y sus principales componentes de estudio.

CAPÍTULO 2 Modelos de Proceso Software Tradicionales

• Presenta los diferentes tipos de modelos de proceso de desarrollo de software tradicionales, estableciendo sus fortalezas, características y estrategias de trabajo con cada uno de ellos. Entre ellos se trabajarán: CVC – Ciclo del Vida Clásico, CP – Construcción por Prototipos, RAD – Rapid Application Development, Métodos Iterativos como el Incremental y Espiral.

CAPÍTULO 3 Modelos de Proceso Software Modernos

• Presenta los diferentes tipos de modelos de proceso de desarrollo de software Modernos, en los cuales se cuenta con: MA – Modelos Agiles como XP (Extreme Programming) y FDD (Feature Driving Programming), RUP – Rational Unified Process (Introducción al modelado UML) y MSF – Microsoft Solution Framework.

CAPÍTULO 4 Gestión de Proyectos

• Hace énfasis en los métodos y técnicas para la Gestión de Proyectos Software.
• Las 4 P de desarrollo de software
• Importancia del arquitecto de software

CAPÍTULO 5 Introducción Arquitectura del Software

• Introducción arquitectura del software
• Presenta una panorámica de la evolución de la ASW características que posee. Adicionalmente realiza una Introducción a los tipos de arquitectura de Software.

CAPÍTULO 6 Introducción a los requerimientos

VII. ESTRATEGIAS DE ENSEÑANZA QUE FAVORECEN EL APRENDIZAJE

Tradicional		Basado en Proyectos	X	Basado en Tecnología	X
Basado en Problemas	X	Colaborativo		Experimental	
Aprendizaje Activo	X	Autodirigido		Centrado en el estudiante	

VIII. EVALUACIÓN

Resultados de aprendizaje (RA) a ser evaluados:	Resultados de aprendizaje asociados a las evaluaciones					
	Actividades Entregables	Talleres	Parciales	Informes de proyecto final	Proyecto final	Exposiciones
RA01	X					
RA02			X			
RA03		X	X			
RA04					X	
RA05						
RA06						
RA07						
RA08						
RA09						
Tipo de evaluación**	EOP	EHP	EE		EBP	
Porcentaje de evaluación (%)	10%	10%	50%		30%	
Trabajo Individual (I) o Grupal (G)	I	G	I		G	

Tipo de nota	0-5	0-5	0-5	0-5	0-5	0-5

IX. MEDIOS Y RECURSOS EDUCATIVOS

A continuación, se describirá cada uno de los recursos propuestos acordes con el modelo que se debe diligenciar:

• Aula normal con pizarrón para sesiones de cátedra y sesiones de discusión y trabajo en grupoApoyo tecnológico Proyector de multimedia, aula virtual.
• Material físico como documentos, libros, revistas entre otros.
• Material virtual: conferencias en la web, documentos virtuales.

X. PRÁCTICAS ACADÉMICAS - SALIDAS DE CAMPO

XI. BIBLIOGRAFÍA

• Craig Larman. “Applying UML and Patterns 2nd Edition”. Prentice Hall. 2002.
• Bernd Bruegge, Allen h. Dutoit. “Ingeniería de Software Orientado a Objetos”. Prentice Hall. 2002.
• Perdita Stevens, Rob Pooley. “Utilización de UML en Ingeniería del Software con Objetos y Componentes”. Addison Wesley. 2002.
• Grady Booch, James Rumbaugh, Ivar Jacobson. “The Unified Modeling Language User Guide”. Addison-Wesley. 1999.
• Ramirez, A. "Introducción a los Patrones de Diseño." Creative Commons. Agosto 2004.
• J. y Rodríguez, G. "Patrones de Interacción: Una Solución para el Diseño de la Retroalimentación Visual de Sistemas Interactivos". CIC. 2002.
• C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, y S. Angel, "A Pattern Language", Oxford University Press, New York, 1977
• P. Coad, D. North y M. Mayfield, "Object Models Strategies, Patterns, and Applications", Yourdon Press, Prentice Hall, 1995.
• J.O. Coplien, "Generative Pattern Languages: An emerging direction of software design", C++ Report, julio-agosto 1994.
• J.O. Coplien, "Advanced C++ Programming Styles and Idioms", Addison-Wesley, 1992.
• E. Gamma, R. Helm, R. Johnson y J. Vlissides, "Design Patterns Elements of Reusable ObjectOriented Software", Addison-Wesley, 1995.
• R. Helm, "Patterns in Practice", Proceedings OOPSLA’95, ACM SigPlan Notices vol.30, No. 10, octubre 1995.
• D.C. Schmidt, "Using Design Patterns to Develope Reusable OO Comunication Software",CACM Vol.38, No. 10, 1995.
• H.A. Schmid, "Crating the Architecture of a Manufacturing Framework by Design Patterns", Memorias de OOPSLA’95.
• H.Huni, R. Johnson, R. Engel,"A Framework for Network Protocol Software", Memorias de OOPSLA’95.G. Booch, "Designing an Application Frameworks", Dr. Dobb’s Journal, Febrero de 1994.
• R.E. Johnson, "Documenting Frameworks using Patterns", Memorias de OOPSLA’92. D.B. Lange, Y. Nakamura, "Interactive Visualization of Design Patterns”, Memorias de OOPSLA’95.
• R. Helm y E. Gamma, "Patterns for Resusable O-O Software", Dr. Dobb’s Sourcebook, marzo-abril de 1995.
• Eds. J.O. Coplien y D.C. Schmidt, "Pattern Languages of Program Design", AddisonWesley 1995., "ectures, and projects
• W.J. Brown, R.C. Malveau, H.W. "Skip" MacCormick III y T.J. Mowbrayin AntiPatterns, Refactoring Software, ArchitCrisis", John Wiley & Sons, 1998.A85

XII. SEGUIMIENTO Y ACTUALIZACIÓN DEL SYLLABUS

Fecha revisión por Consejo Curricular:			
Fecha aprobación por Consejo Curricular:		Número de acta:	

**Tipo de Evaluación	Abreviatura
1. Evaluación de habilidad	EHP
2. Evaluación basada en p	EBP
3. Evaluación oral o prese	EOP

4. Evaluación escrita	EE
5. Evaluación formativa	EF
6. Evaluación de desempe	ED