	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: PATRONES Y ARQUITECTURAS DE SOFTWARE								
Código del espacio académico:		79502007	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		Presencial con incorporación de TIC		Virtual		Otros:		Cuál:_____

II. SUGERENCIAS DE SABERES Y CONOCIMIENTOS PREVIOS

Modelado de software desde las perspectivas funcional, estructural y dinámica; programación orientada a objetos.

III. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

El cuerpo de conocimiento que grandes organizaciones de Ingenieros como la ACM (Association for Computing Machinery) y la IEEE (The Institute of Electrical and Electronics Engineers) han promulgado, da importancia pragmática a las temáticas sobre patrones de diseño y arquitecturas de software en el área de conocimiento de diseño.

Es así como en el énfasis de Ingeniería de Software se ha establecido que la asignatura aquí presentada aborde la revisión conceptual y práctica del tema arquitectural junto con la recuperación de diseño basada en ejemplos aplicados de patrones. Este enfoque permite al futuro magíster de la Universidad Distrital contar con herramientas de uso práctico que sean coherentes con su formación previa y que apoyen la recuperación de diseños de software detallados y de alto nivel de abstracción dotados de solidez y elegancia. El ejercicio de recuperación de diseño permitirá madurar conceptos abordados en la ingeniería directa estudiada y aplicada en el curso de Ingenieria de Software II.

De otra parte, uno de los problemas más trabajado en las últimas décadas ha sido el manejo de concurrencia de procesos computacionales. En este sentido, se propone el estudio y analisis de un conjunto de técnicas que en algunos casos se han denominado paradigmas de programación concurrente, relevantes en la formación de todo investigador que se vea enfrentado al diseño de soluciones de software donde el uso de arquitecturas de multiprocesadores es ya común. Por lo anterior, este curso revisa e intenta que el estudiante conceptualice de manera sólida los principales asuntos tratados en el ámbito de la programación concurrente.

IV. OBJETIVOS DEL ESPACIO ACADÉMICO (GENERAL Y ESPECÍFICOS)

OBJETIVO GENERAL: Establecer bases conceptuales y prácticas en las áreas de modelado arquitectural de software, paradigmas de programación concurrente y recuperación de diseños detallados de software con base en ejemplos que usan patrones de diseño bien conocidos; de estos últimos, se tendrán en cuenta aquellos ampliamente utilizados, como los denominados Patrones de la Pandilla de los Cuatro

denominados Patrones de la Pandilla de los Cuatro.

OBJETIVOS ESPECÍFICOS

• Aplicar y ejercitar los conocimientos de diseño detallado de software que fueron adquiridos en los cursos de Ingeniería de Software I e Ingenieria de Software II.
• Ejercitar en la comprensión de patrones de diseño con base en el ejercicio práctico de recuperación de diseño para la elaboración de modelos funcionales basados en casos de uso, diseños estructurales y dinámicos alrededor de ejemplos donde se implementen soluciones que usan los patrones GoF.
• Introducir conceptos básicos del modelado formal y semiformal de arquitecturas de software a través de una revisión generalista de Lenguajes de Descripción Arquitectural y la revisión detallada de uno de ellos.
• Revisar los conceptos fundamentales en el área de conocimiento de arquitecturas de software.
• Diferenciar conceptualmente el modelado arquitectural del modelado detallado en tiempo de diseño.
• Estudiar y cimentar los conceptos fundamentales de los paradigmas de programación para manejar concurrencia

V. PROPÓSITOS DE FORMACIÓN Y DE APRENDIZAJE (PFA) DEL ESPACIO ACADÉMICO			
Competencias	**Dominio-Nivel**	**RA**	**Resultados de Aprendizaje**
De contexto	Aportar de manera desinteresada a un equipo de estudio.		
	Argumentar de manera sólida y consistente la razón de los modelos de soluciones recurrentes en software usando patrones de diseño ampliamente utilizados, representados usando unidades lingüísticas de UML.		
	Cumplir con acuerdos establecidos en un grupo de manera responsable y con alta calidad.		
Básicas Científicas (Cognitivas)	Diferenciar conceptualmente modelo, metamodelo y meta-metamodelo en el diseño de software.		
	Diferenciar conceptualmente las técnicas de programación concurrente en casos puntuales de desarrollo de software.		
	Disertar sobre temas técnicos de software con la capacidad de comunicar claramente la intencionalidad de la exposición a un auditorio idóneo en el asunto.		
	Abordar de manera autónoma la lectura de artículos de investigación o divulgación de resultados respecto a paradigmas de programación concurrente y especificación formal de arquitecturas de software.		
	Aplicar de manera idónea la ingeniería inversa a programas orientados a objetos.		
Laborales	Integrarse de manera efectiva en equipos de diseño de software con capacidad de efectuar Revisiones Técnicas Formales (RTF's).		
	Identificar las situaciones problémicas en el diseño de software donde un patrón de diseño puede ser el más adecuado.		
	Capacidad de aplicar técnicas de concurrencia en un lenguaje de programación específico.		
VI. CONTENIDOS TEMÁTICOS			

1. Introducción [2 h.]

• Presentación del profesor, objetivos, contenido, metodología y formas de evaluación del curso.
• Arquitectura de sistemas vs. arquitecturas de software.
• Conceptos básicos de patrones de software en tiempo de diseño.
• Marcos de trabajo vs. patrones de diseño.
• Categorización de los patrones de software de "Gangs of Four"

2. Fundamentos de arquitecturas de software [10 h.]

• Conceptos básicos
• Niveles de abstracción arquitecturales.
• Taxonomía de estilos arquitecturales.
• Representación arquitectural y lenguajes de descripción arquitectural
• Aspectos de investigación en arquitecturas de software.
• Introducción al modelado formal: The CHAM (Chemical Abstract Machine)
• Introducción a la especificación formal de arquitecturas de software: Modelado basado en un enfoque axiomático (Lenguaje Z) y modelado basado en un enfoque de álgebra de procesos (CSP de Hoare).

3. Técnicas y patrones de concurrencia [8 h.]

• Paradigmas de programación concurrente.
(a) Busy-Wait
(b) Semáforos
(c) Monitores
(d) Paso de mensajes
• Patrones de diseño para control de concurrencia
(a) Sección crítica
(b) Bloqueo ordenado consistente
(c) Suspensión vigilada o condicionada
(d) Bloqueo de lectura-escritura

4. Recuperación de diseño aplicado a ejemplos de patrones de diseño creacionales o de construcción de objetos[5 h.]

• Método factoría (Factory Method, en el inglés).
• Única ejemplificación (Singleton, en el inglés).
• Factoría Abstracta (Abstract Factory, en el inglés).
• Prototipo (Prototype, en el inglés)
• Constructor (Builder, en el inglés)

5. Recuperación de diseño aplicado a ejemplos de patrones de diseño para el tratamiento de colecciones [6 h.]

• Tratamiento uniforme a objetos compuestos (Composite, en el inglés).
• Iterador (Iterator, en el inglés) .
• De información intrínseca y extrínseca a objetos (Flyweight, en el inglés).
• Visitador (Visitor, en el inglés).

6. Recuperación de diseño aplicado a ejemplos de patrones de diseño estructurales [7 h.]

• Decorador (Decorator, en el inglés).
• Adaptador (Adapter, en el inglés).
• Cadena de responsabilidad (Chain of responsibility, en el inglés).
• Fachada (Facade, en el inglés)

• Fachada (Façade, en el inglés).
• Intermediario (Proxy, en el inglés).
• Puente (Bridge, en el inglés).

7. Recuperación ded diseño aplicado a ejemplos de patrones de diseño comportamentales [10 h.]

• Comando (Command, en el inglés).
• Mediador (Mediator, en el inglés).
• Memorizador (Memento, en el inglés).
• Observador (Observer, en el inglés).
• Intepretador (Intepreter, en el inglés).
• Estado (State, en el inglés).
• Estrategia (Strategy, en el inglés).

VII. ESTRATEGIAS DE ENSEÑANZA QUE FAVORECEN EL APRENDIZAJE

Tradicional		Basado en Proyectos		Basado en Tecnología	
Basado en Problemas		Colaborativo		Experimental	
Aprendizaje Activo		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
RA04			X			
RA05						
RA06						
RA07						
RA08						
RA09						
Tipo de evaluación**						
Porcentaje de evaluación (%)	20%	10%	30%			40%
Trabajo Individual (I) o Grupal (G)						
Tipo de nota	0-5	0-5	0-5	0-5	0-5	0-5

IX. MEDIOS Y RECURSOS EDUCATIVOS

• Salón de conferencias.
• Ayudas audiovisuales.
• Herramienta de modelado conforme a UML en sus últimas revisiones.
• Programa para diseñar presentaciones .
• Acceso a Internet .
• Suscripción a revistas técnicas nacionales e internacionales sobre Ingeniería de Software y Ciencias de la Computación.
• IDE para programación en JAVA o mínimamente la máquina virtual en su última versión estable. Otros lenguajes de programación orientados a objetos podrán ser utilizados.
• Textos técnicos sobre Ingeniería de Software+A70.
• Acceso a bibliotecas digitales y con formato en copia dura.

Esta asignatura cuenta con un aula virtual. En este sitio puede encontrar el estudiante:
• Este documento de especificación de contenidos programáticos y demás aspectos del denominado Syllabus de la asignatura.

• Material de apoyo audiovisual para las conferencias esenciales de modelado de software.
• Documentos de apoyo técnico.
• Programación de controles de lectura.
• Especificación de ejercicios puntuales.
• Código fuente de los ejercicios de recuperación de diseño del curso.
• Especificación del examen final.

X. PRÁCTICAS ACADÉMICAS - SALIDAS DE CAMPO

XI. BIBLIOGRAFÍA

Básicas:

• GAMMA ERICH, R. HELM, R. J., AND VLISSIDES, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.
• KUCHANA, P. "Software Architecture Design Patterns in Java". CRC Press, 2004.
• BASS, LEN; CLEMENS, PAUL AND KAZMAN, RICK. "Software Architecture in Practice". SEI Series in Software Engineering. Addison-Wesley. Third Edition. 2015.
• Taylor, Richard N.; Medvidovic, Nenad; Dashofy, Eric M. "Software Architecture. Foundations, Theory and Practice". John Wiley & Sons Inc. 2010.
• Buschmann, Frank; Meunier, Regine; Ronhert, Hans; Sommerlad, Peter and Stal, Maichael. "Pattern-Oriented Software Architecture. A system of Patterns". Volume I. 2000.
• Buschmann, Frank; Meunier, Regine; Ronhert, Hans; Sommerlad, Peter and Stal, Maichael. "Pattern-Oriented Software Architecture. A system of Patterns". Volume II. 2000.
• Clemens, Paul; Bachmann, Felix; Bass, Len; Garlan, David; Ivers, James; Little, Reed; Merson, Paulo; Nord, Robert and Stafford, Judith. "Documenting Software Architectures. Views and Beyond". Second Edition. SEI Series in Software Engineering. Addison-Wesley. 2011.
• Balachandran Pillai, Anand. "Software Architecture with Python". Packt Publishing Ltd. 2017.
• Diosa, Henry Alberto. "Sistema de Información soporte a sistemas de abastecimiento y seguridad alimentaria. Arquitectura de referenci prescriptiva". Editorial UD. 2019.

Complementarias:

• BERRY, G., AND BOUDOL, G. "THE CHEMICAL ABSTRACT MACHINE". THEORETICAL COMPUTER SCI. 96 (1992), 217–248.
• COMMITTEE, I. C. S. P. P. GUIDE TO THE SOFTWARE ENGINEERING BODY OF KNOWLEDGE. TECH. REP., IEEE, 2004.
• OMG UNIFIED MODELING LANGUAGE. OBJECT MANAGEMENT GROUP. VERSIÓN 2.5. 2015.
• ON COMPUTING CURRICULA: IEEE COMPUTER SOCIETY, T. J. T. F., AND FOR COMPUTING MACHINERY, A. CURRICULUM GUIDELINES FOR UNDERGRADUATE DEGREE PROGRAMS IN SOFTWARE ENGINEERING. TECH. REP., IEEE AND ACM, 2004.
• TUCKER, A. B., AND SELECTED CONTRIBUTORS. COMPUTER SCIENCE HANDBOOK. SECOND EDITION. CHAPMAN AND HALL/CRC IN COOPERATION WITH ACM, 2004.
• DIOSA HENRY ALBERTO. "LINEAMIENTOS PARA EL DISEÑO DE PROGRAMAS DE FORMACIÓN EN INGENIERÍA DE SOFTWARE". EDITORIAL UD. 2018

REVISTAS:

• IEEE Transactions on Software Engineering.
• IEEE Software Engineering.
• Publicaciones de Elsevier en "Notes of Computer Science".
• PUBLICACIONES DE ACM EN CIENCIAS DE LA COMPUTACIÓN E INGENIERÍA DE SOFTWARE.

DIRECCIONES DE INTERNET:

• https://www.omg.org/
• https://www.w3.org/
• https://www.uml.org/
• http://arquisoft.udistrital.edu.co
• https://sparxsystems.com

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