Maestría en Ciencias de la Información y las Comunicaciones

Énfasis: Ingeniería de Software

ESPACIO ACADÉMICO (Asignatura): **PATRONES Y ARQUITECTURAS DE SOFTWARE[1].**

- Obligatorio (X) : Básico (X) Complementario ( )
- Electivo ( ) : Intrínsecas ( ) Extrínsecas ( )

NÚMERO DE CREDITOS: Cuato (4)

TIPO DE CURSO: TEÓRICO: _____ PRACTICO:_____ TEO-PRAC: X

Alternativas metodológicas:
Clase Magistral (X), Seminario ( ), Seminario – Taller (X), Taller ( ), Prácticas (X), Proyectos tutorados ( ), Otro: _____

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.

PRERREQUISITO/ CONOCIMIENTOS PREVIOS: Modelado de software desde las perspectivas funcional, estructural y dinámica; programación orientada a objetos.

1. *Elaboró: Henry Alberto Diosa Ph.D. Profesor de planta.*

Programación del Contenido

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.

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.*

Competencias de formación

COMPETENCIAS 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.

COMPETENCIAS 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.

COMPETENCIAS 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.

Programa sintético

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 (Faζade, en el inglés).
   - Intermediario (*Proxy*, en el inglés).
   - Puente (*Bridge*, en el inglés).
   - Forzador de agregación (*Aggregate Enforcer*, 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).
   - Método Plantilla (*Template Method*, en el inglés)

Estrategias

Metodología Pedagógica y didáctica:

Se desarrollarán clases magistrales y conferencias sobre los temas programados en el curso.

Semanalmente se entregarán dos ejemplos que reflejen un patrón de diseño presentado en su conceptualización en sesiones presenciales por parte del profesor; el estudiante debe recuperar los modelos funcionales, estructurales y dinámicos del ejemplo, también debe lograr la ejecución del código fuente del ejemplo dado y presentar la sustentación personal presencial de la ingeniería inversa aplicada en las sesiones pactadas para tal fin.

Se evaluarán periódicamente las lecturas técnicas que el docente programe para el curso junto con exposiciones y ejercicios puntuales que sean asignados a los estudiantes.

El modelo pedagógico y didáctico utilizado se basa en el seminario-taller y en la ejercitación en Revisiones Técnicas Formales (RTF's) como buena práctica de los procesos de desarrollo de software.

	Horas			Horas profesor/semana	Horas Estudiante/semana	Total Horas Estudiante/semestre	Créditos
Tipo de Curso	TD	TC	TA	(TD + TC)	(TD + TC +TA)	X 16 semanas	
Teórico-Práctico	3	1	4	4	8	128	4

Trabajo Presencial Directo (TD): Trabajo de aula con trabajo de análisis grupal estilo seminario.

Trabajo Mediado Cooperativo (TC): Trabajo de tutoría del docente a pequeños grupos o de forma individual a los estudiantes.

Trabajo Autónomo (TA): Trabajo del estudiante sin presencia del docente, que se puede realizar en distintas instancias: en grupos de trabajo o en forma individual, en casa o en biblioteca, laboratorio...

Recursos

- 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.
- 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.

Bibliografía

Textos guías

- 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.

Textos complementarios

- 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

Organización / Tiempos

Se recomienda trabajar una unidad cada cuatro semanas, trabajar en pequeños grupos de estudiantes, utilizar Internet (aula virtual, correo institucional, portal web institucional, entre otros) para comunicarse con los estudiantes, para revisiones de avances y solución de preguntas (esto considerarlo entre las horas de trabajo cooperativo).

Semana/unidad temática	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1. **Introducción.**																
2. **Fundamentos de arquitecturas de software.**																
3. **Técnicas y patrones de concurrencia.**																
4. **Recuperación de diseño aplicado a ejemplos de patrones de diseño creacionales.**																
5. **Recuperación de diseño a ejemplos de patrones de diseño para el tratamiento de colecciones.**																
6. **Recuperación de diseño a ejemplos de patrones de diseño estructurales.**																
7. **Recuperación de diseño a ejemplos que aplican patrones de diseño comportamentales.**																
8. **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).**																
Examen final																

Evaluación

La evaluación se realizará teniendo en cuenta:

	TIPO DE EVALUACIÓN	FECHA	PORCENTAJE
TRES VALORACIONES	Tres controles de lectura	**Cada control se efectuará con una diferencia temporal de quince a veinte días.**	**20.00%**
TERCERA NOTA	Ejercicios puntuales para la valoración de conceptualización sobre algunas temáticas del curso.	**En el transcurso del semestre.**	**10.00%**
CUARTA NOTA	Recuperación de diseño a ejemplos con su respectiva sustentación. Evaluaciones puntuales sobre patrones de diseño.	**En el transcurso del semestre.**	**40.00%**
EXAMEN FINAL	Valoración de conceptualización de lo estudiado en el curso.	**Fin de semestre**	**30.00%**

ASPECTOS A EVALUAR DEL CURSO

1. Evaluación del desempeño docente: Se hará a través del modelo de evaluación docente que ha instaurado la Universidad Distrital
2. Evaluación de los aprendizajes de los estudiantes en sus dimensiones: Sustentación de la recuperación de diseños de software en las tres perspectivas básicas (funcional, estructural y dinámica). Se efectuarán diferentes hitos de revisión asociados a la revisión de temas técnicos y de investigación por medio de lecturas. Se valora y determina la suficiencia conceptual frente a temas relacionados con el uso consistente de las unidades linguísticas del metamodelo propio de la disciplina.
3. Autoevaluación: Por parte del docente a través del modelo instaurado por la Universidad Distrital.
4. Coevaluación del curso: Existirá una retroalimentación de los estudiantes sobre fortalezas y debilidades del curso al finalizar el semestre.