UNIVERSIDAD NACIONAL DE RÍO CUARTO
FACULTAD DE CIENCIAS EXACTAS, FÍSICO-QUÍMICAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

CARRERA/S: ANALISTA EN COMPUTACIÓN. PROFESORADO Y LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN.

PLAN DE ESTUDIOS: 1999

ASIGNATURA: Análisis y Diseño de Sistemas **CÓDIGO: 3303**

DOCENTE RESPONSABLE: Mg. Marcela Daniele

EQUIPO DOCENTE: Lic. Marcelo Uva, Lic. Ariel Arsaute, Lic. Franco Brusatti, Prof. Daniela Solivellas

AÑO ACADÉMICO: 2019

REGIMEN DE LA ASIGNATURA: cuatrimestral

RÉGIMEN DE CORRELATIVIDADES: (para cursado)

Aprobada	*Regular*
	3325

CARGA HORARIA TOTAL: 180 **TEÓRICO:** 60 hs **PRÁCTICAS y LABORATORIO:** 120 hs

CARÁCTER DE LA ASIGNATURA: Obligatorio

A. CONTEXTUALIZACIÓN DE LA ASIGNATURA

La asignatura se desarrolla en el 3er. año de las carreras de: Analista en Computación, Profesorado en Ciencias de la Computación y Licenciatura en Ciencias de la Computación.

B. OBJETIVOS PROPUESTOS

Con el cursado de esta asignatura se espera que el estudiante pueda:

- Comprender conceptos básicos de Ingeniería de Software, diferentes métodos de desarrollo de software y su evolución (MOO, MDD).
- Adquirir el conocimiento sobre las etapas del ciclo de vida de desarrollo de un software para obtener un producto de calidad. Instanciar en diferentes métodos de desarrollo de software.
- Utilizar lenguajes gráficos de especificación para el modelado de sistemas.
- Utilizar diversas herramientas de software de modelado e incentivar su uso y comparación.
- Introducirse en conceptos de diseño, en la construcción de modelos genéricos para solucionar problemas con similares características, e instanciar problemas reales sobre dichos modelos.
- Utilizar los patrones de diseño adecuados para modelar soluciones a problemas recurrentes.
- Desarrollar habilidades y actitudes que favorezcan el trabajo colaborativo y el aprendizaje continuo.
- Comprender la importancia de la prueba del software, conocer y aplicar diferentes técnicas y herramientas.
- Ser capaz de elegir las herramientas de software más adecuadas, de acuerdo al tipo de sistema, con la suficiente formación para adaptarse a los constantes cambios, a las nuevas tecnologías y a los diversos ámbitos de aplicación de los sistemas de software

C. CONTENIDOS BÁSICOS DEL PROGRAMA A DESARROLLAR

Conceptos básicos de Ingeniería de software. Modelos de desarrollo de Software. Metodologías de desarrollo de software: tradicionales y ágiles. Modelado orientado a objetos. Desarrollo dirigido por modelos. Modelado Gráfico: UML (Unified Modeling Language) Diagramas de Clases. Clases. Diagrama de Objetos. Diagrama de Interacción. Diagrama de Casos de Uso. Diagrama de Actividades, de Estados, de Componentes y de Despliegue. Ingeniería de requerimientos del Software. Etapas del ciclo de desarrollo de un software. Una metodología de desarrollo de software orientada a objetos: Proceso Unificado: Características, Conceptos, etapas del ciclo de vida. Captura de Requerimientos. Análisis, Diseño e implementación del Software. Una metodología de desarrollo Agil: SCRUM. Diseño modular.

Diseño detallado. Diseño OO. Patrones de Diseño. Prueba: Casos de prueba. Métodos de Prueba del Software. Prueba Funcional. Prueba Estructural.

D. FUNDAMENTACIÓN DE LOS CONTENIDOS

Hacia fines de los años 60 se produce la llamada "crisis del software", caracterizada por síntomas en el software como, carencia de fiabilidad, necesidad de mantenimiento permanente, retrasos en las entregas y costos superiores a los presupuestados y difícil de mantener o adaptarse a los continuos cambios. A consecuencia de esta crisis surge "Ingeniería de Software". Muchas fueron las discusiones por una definición apropiada de este término, pero se logró el acuerdo común de que el desarrollo de software debe verse como el desarrollo de un producto de ingeniería que requiere planeamiento, análisis, diseño, implementación, prueba y mantenimiento. La idea básica consistió en observar el sistema de software a construir como un producto complejo, y a su proceso de construcción como un trabajo ingenieril, basado en metodologías, técnicas, teorías y herramientas.

Es por ello que para desarrollar un proyecto de software es necesario seguir un proceso o metodología de software, que indica la secuencia de actividades a seguir para completar el ciclo de vida de desarrollo de un software.

E. ACTIVIDADES A DESARROLLAR

Se desarrollan las clases teóricas a todos los estudiantes que cursan la materia, con un total de entre 3 y 5 horas semanales. Se divide el total de estudiantes en dos comisiones prácticas, asistiendo a clases dos veces por semana, 2 hs. cada una. Además, los estudiantes realizan un proyecto de software de forma grupal, y asisten a una clase- taller semanal de 2 horas.

CLASES TEÓRICAS: Presencial en aula, 60 Horas totales

CLASES PRÁCTICAS y TALLER: Presencial y semi presencial en laboratorio, 120 Horas totales

F. NÓMINA DE TRABAJOS PRÁCTICOS

Metodologías de desarrollo de software.
Análisis y Especificación de Requerimientos.
Descripción de funcionalidades con PU y SCRUM
Prueba Funcional. Prueba Estructural.
UML. Diagramas de Clases.
UML. Diagramas de Objetos.
UML. Diagramas de Actividades.
UML. Diagramas de Estados.
UML. Diagramas de Secuencia
Ingeniería Directa. Ingeniería Inversa.
Patrones de Diseño. Implementación

G. HORARIOS DE CLASES:

TEORICOS miércoles 13 a 16 hs y viernes de 10 a 12 hs (alternando con Taller)

TALLER: viernes 10 a 12 hs.

PRACTICOS Comisión 1: Martes-Jueves 10 a 12 hs. **Comisión 2**: Martes-Jueves 16 a 18 hs.

HORARIO DE CLASES DE CONSULTAS (estos horarios se pueden modificar de acuerdo a la coordinación de los estudiantes, y se publican los nuevos horarios

Responsable: Lunes 12 hs.

Auxiliares: Martes 13 hs, Jueves 9 hs. Viernes 12 hs

H. MODALIDAD DE EVALUACIÓN:

EVALUACION PARCIAL: 2 exámenes parciales escritos sobre el práctico de la materia, con sus respectivos recuperatorios. Un proyecto – taller grupal.

EVALUACIÓN FINAL: escritos u orales sobre la teoría y práctica de la materia.

CONDICIONES DE REGULARIDAD: Aprobar dos exámenes parciales y el proyecto final.

CONDICIONES DE PROMOCIÓN: No posee.

I. PROGRAMA ANALÍTICO

Unidad 1: **Ingeniería de software. Introducción**. Historia. Definición. El Ciclo de Vida del software. Atributos del software. Metodologías tradicionales. Metodologías ágiles. Modelos de desarrollo de Software: Construcción de Prototipos, Procesos Evolutivos, Incremental, Espiral, Ensamblaje de Componentes, Desarrollo Concurrente, Métodos Formales, Orientado a Objetos. Diseño por Contratos. Proceso Unificado. SCRUM. XP. Tipos de Sistemas.

Unidad 2: **Ingeniería de requerimientos del Software**. Requerimientos funcionales y no funcionales. Requerimientos de usuario y de sistema. Documentación de requerimientos -SRS. Reingeniería del Software: Ingeniería Inversa e Ingeniería Directa. Proceso Unificado: Características, Conceptos, Artefactos, Actividades, Trabajadores, Flujos de Trabajo y Fases. Captura de Requerimientos. Modelo de Negocio. Procesos de negocio. Glosario de términos. Reglas de negocio. Modelo de Casos de Uso. Modelo de Análisis. Clases de análisis. SCRUM: Características. Conceptos. Historias de usuario. Roles.

Unidad 3: La importancia de modelar. Principios del modelado. Modelado orientado a objetos. Técnicas de Descripción aplicadas a Requerimientos. Modelado estructural o estático y modelado dinámico. UML: Estereotipos. Valores etiquetados. Restricciones. Diagramas de Clases. Clases: atributos, operaciones y responsabilidades. Relaciones. Interfaces. Paquetes. Instancias. Diagrama de Objetos. Diagrama de Interacción. Diagrama de Casos de Uso. Eventos y señales. Diagrama de Actividades. Máquinas de Estado. Componentes. Despliegue.

Unidad 4: Diseño e implementación del Software. Conceptos de diseño. Abstracción. Refinamiento. Modularidad. Arquitectura del software. Estructura de programa. Ocultamiento de información. Diseño modular. Cohesión y Acoplamiento. Diseño detallado. Diseño OO.

Proceso Unificado: Modelo de Diseño: artefactos, actividades y trabajadores. Modelo de Implementación. Nodos y Componentes. Modelo de Prueba. Casos de prueba. Procedimiento de prueba. Plan de prueba.

Unidad 5: Métodos de Prueba del Software. Fundamentos teóricos. Principios de la Prueba. Prueba Funcional. Análisis del Valor Límite. Clases de Equivalencia. Tablas de Decisión. Prueba Estructural. Cobertura de Sentencia. Cobertura de Arco. Cobertura de Condición. Cobertura de Camino. Complejidad Ciclomática. Herramientas.

Unidad 6: Patrones de Diseño. Introducción. Conceptos. Descripción. Utilización. Problema. Solución. Consecuencia. Catálogo de Patrones de Diseño. Categoría de Patrones: creacionales, estructurales y de comportamiento.

CRONOGRAMA DE CLASES Y PARCIALES

FECHA	TEÓRICOS	FECHA	PRÁCTICOS
Miércoles 13-03	Presentación. Introducción. Conceptos básicos de IS, 4 Ps. Metodologías. Ciclo de vida en el desarrollo de un software	Martes 19/3	1. UML. Diagramas de Clases.
Viernes 15-03	UML. Introducción. Clases. Relaciones. Diagramas de Clases.	JUEVES 21-3	1. UML. Diagramas de Clases.
Miércoles 20-03	UML. Introducción. Clases. Relaciones. Diagramas de Clases. Diagramas de Objetos. Ingeniería Inversa y Directa.	MARTES 26-3 JUEVES 28-03 VIERNES 29-3	1. UML. Diagramas de Clases. 1. UML. Diagramas de Clases. P-T 3. Anal y Espec de Requer
Miércoles 27-03	Análisis del Problema. SRS c/ más de un método. Ingeniería de requerimientos del Software. Requerimientos funcionales y no funcionales.	JUEVES 04-04 Martes 9/4	2. UML. Diagramas de Objetos 4 Prueba
Miércoles 03-04	Prueba. Funcional y Estructural. PU: Modelo de Prueba.	Jueves 11-4 Viernes 12-4 Martes 16-04	4 Prueba 4 Prueba 5. Ejercicios de Repaso 1er Parcial
Miércoles 10-04	UML. Diagramas de Casos de Uso. UML. Diagramas de Actividades. Diagramas de Estados	MARTES 23-4 JUEVES 25-04	6. Diagrama de Casos de Uso 7. UML. Diagrama de Actividades
Miércoles 17-04	PRIMER PARCIAL	MARTES 30-4 JUEVES 2-5	7. UML. Diagrama de Actividades 8. UML. Diagrama de Estados
Miércoles 24-04 (feriado 1/5)	Una metodología de d**esarrollo OO: Proceso Unificado.** Una metodología Agil: SCRUM PU: Captura de Requerimientos: M Negocio + M Casos de Uso. Plantillas Genéricas p/desc de CU	MARTES 07-05 JUEVES 09-05	8. UML. Diagrama de Estados
Miércoles 08-05	UML. Diagramas de Interacción: Diagrama de Secuencia. A y Diseño de Software. Diseño Funcional. Diseño de Casos de Uso. Diseño Detallado. (Jalote). PU: Análisis. Diseño. Plantillas genéricas para A y Diseño.	MARTES 14-05 JUEVES 16-05	9. UML. Diagrama de Secuencia
Miércoles 15-05	PATRONES DE DISEÑO	MARTES 21-5 JUEVES 23-05	9. UML. Diagrama de Secuencia 10. Patrones de Diseño
Miércoles 22-05	PATRONES DE DISEÑO	MARTES 28-5	10. Patrones de Diseño.
Miércoles 29-05	Implementación. Conceptos. Nodo y Componente. Modelo de Implementación.	Jueves 30-5 Martes 04-06	10. Patrones de Diseño. 11. Ejercicios de Repaso 2º parcial.
Miércoles 05-06	SEGUNDO PARCIAL	Jueves 6-6 Martes 11 06	11. Ejercicios de Repaso 2º parcial 11. Ejercicios de Repaso 2º parcial

Miércoles 12-06	UML: Diagrama de Despliegue. Diagrama de Artefactos.	Jueves 13-6 Martes 18-06	Práctica 10. Ejercicios de Repaso 2º parcial

Miércoles 17/4	PRIMER PARCIAL
Miércoles 05/6	SEGUNDO PARCIAL
A coordinar con estudiantes	RECUPERATORIO 1º PARCIAL y 2º PARCIAL
Miércoles 19/6	PRESENTACION FINAL DEL PROYECTO

BIBLIOGRAFÍA

1. Pressman, Roger. Software Engineering: A Practitioner's Approach. 8va edition. McGraw Hill. 2015.
2. Sommerville Ian. Ingeniería del Software, 7th edition, Addisson Wesley. Pearson Education, 2005. ISBN: 978-84-7829-074-1
3. Ghezzi Carlo, Jazayeri M., Mandrioli D.. Fundamentals of Software Engineering. Prentice Hall, 1991.
4. Kendall y Kendall. Análisis y Diseño de Sistemas. Pearson Education. 2005.
5. Pankaj Jalote. An Integrated Approach to Software Engineering. Springer. 2005.
6. Booch Grady, Rumbaugh J., Jacobson Ivar. The Unified Modeling Language. Addison Wesley. 1999.
7. Booch Grady. Object-oriented analisys and desing with applications. Benjamin Cummins, 1994. Versión español: Análisis y Diseño Orientado a Objetos: Con Aplicaciones. Addison Wesley. 1996.
8. Paul Ammann and Jeff Offutt. Introduction to Software Testing. Cambridge University Press. 2008
9. Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.
10. Grand Mark. Patterns in Java. Volume 1. A Catalog of Reusable Design Patterns Illustred with UML. John Wiley & Sons Inc. 1998.
11. Highsmith Jim. Agile Software Development Ecosystems. Addison-Wesley. 2002. ISBN: 0 201-760-13-6.
12. Jacobson Ivar, Booch Grady, Rumbaugh James. The Unified Software Development Process. Addison Wesley. 1999.
13. Meyer Bertrand. Object Oriented Software Construction. Prentice Hall. 1997.
14. OMG. Object Management Group. Unified Modeling Language Specification. http://www.omg.org .
15. Pattern-Oriented Software Architecture. Volume 1: A System of Patterns. Frank Buschmann , Regine Meunier , Hans Rohnert , Peter Sommerlad , Michael Stal .
16. Rational Unified Process. http://www.rational.com/rup/
17. www.agilemanifesto.org - http://www.agile-spain.com/

NOTA: En cada una de las teorías se indican los capítulos de lectura del material presentado en la bibliografía.

Profesora Responsable: Mg. Marcela Daniele. _____