FORMULARIO PARA LA PRESENTACIÓN DE PROGRAMAS DE ASIGNATURAS en el CONTEXTO DE PANDEMIA por Covid-19[1]

Año Lectivo: 2021

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**

MODALIDAD DE CURSADO: Presencial

DOCENTE RESPONSABLE:

Marcela Elena Daniele, Mag en Ingeniería de Software, PAD Simple

EQUIPO DOCENTE:

Marcelo Uva, Lic en Cs de la Computación, Ay 1ra Exclusivo

Ariel Arsaute, Lic en Cs de la Computación, Ay 1ra SemiExclusivo

Daniela Solivellas, Prof en Cs de la Computación, JTP SemiExclusiva

Franco Brusatti, Lic en Cs de la Computación, Ay 1ra SemiExclusivo

RÉGIMEN DE LA ASIGNATURA: Cuatrimestral

UBICACIÓN EN EL PLAN DE ESTUDIO: 3er año, Primer Cuatrimestre

RÉGIMEN DE CORRELATIVIDADES:

Asignaturas aprobadas:

Asignaturas regulares: 3325

CARÁCTER DE LA ASIGNATURA: Obligatoria

CARGA HORARIA TOTAL: 180 horas

Teóricas:	**60 hs**	**Prácticas:**	**70 hs**	**Teóricas -Prácticas:**	**…. hs**	**Laboratorio:**	**50 hs**

CARGA HORARIA SEMANAL: 12 horas (según el plan de estudio vigente)

Teóricas:	**5 hs**	**Prácticas:**	**4 hs**	**Teóricas -Prácticas:**	**…. hs**	**Laboratorio:**	**4 hs**

[1] Res. CS 120/2017 y Res. CD 049/2020

1. CONTEXTUALIZACIÓN DE LA ASIGNATURA

La asignatura se desarrolla en el 3er. año de las carreras de Computación, donde el futuro egresado afianza la noción del desarrollo de productos de software como una tarea de ingeniería que requiere de metodologías, técnicas y herramientas para el desarrollo de productos de software de calidad que resuelvan problemas de la humanidad. En esta asignatura, el estudiante aborda diferentes métodos para llevar a cabo el ciclo de vida de desarrollo de un software y experimenta el desarrollo de un proyecto real, pasando por todas las etapas del proceso, utilizando y aplicando los conocimientos adquiridos en las asignaturas anteriores como así también las asignaturas del mismo año, como la resolución de problemas algorítmicos y la programación. Los temas abordados en esta asignatura se complementan fundamentalmente con los contenidos de Ingeniería de Software (código 3304), asignatura del segundo cuatrimestre del mismo año.

2. 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.
- 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.
- Conocer e internalizar el uso de lenguajes gráficos de especificación para el modelado de sistemas.
- 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 de 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.
- Elegir herramientas de software adecuadas, con formación para adaptarse a los constantes cambios, a nuevas tecnologías y a diversos ámbitos de aplicación de los sistemas de software.

3. EJES TEMÁTICOS ESTRUCTURANTES DE LA ASIGNATURA Y ESPECIFICACIÓN DE CONTENIDOS

3.1. Contenidos mínimos 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). Ingeniería de requerimientos del Software. Diseño modular. Diseño detallado. Diseño OO. Patrones de Diseño. Prueba: Casos de prueba. Métodos de Prueba del Software. Prueba Funcional. Prueba Estructural. Una metodología tradicional de desarrollo de software orientada a objetos: Proceso Unificado: características, conceptos, etapas del ciclo de vida: Captura de Requerimientos. Análisis y Diseño, Implementación, Prueba. Una metodología de desarrollo Ágil: SCRUM.

3.2. Ejes temáticos o unidades

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. Definición y documentación de Requerimientos en diferentes metodologías ágiles y tradicionales. 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: Modelado de sistemas. Principios. Técnicas de Descripción de Requerimientos. Modelado estructural o estático y modelado dinámico. MOO. UML: Estereotipos. Valores etiquetados. Restricciones. Diagramas de clases: Clases: atributos, operaciones y responsabilidades. Relaciones. Interfaces. Paquetes. Diagrama de objetos: Objetos, Instancias, enlaces. Diagrama de Interacción: escenarios. Diagrama de casos de uso. Diagrama de actividades. Diagrama de estados. Eventos y señales. Modelado de nodos y componentes.

Unidad 4: Diseño de 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: Patrones de diseño. Introducción. Conceptos. Descripción. Utilización. Problema. Solución. Consecuencia. Catálogo de patrones de diseño. Categoría de patrones: creacionales, estructurales y de comportamiento.

Unidad 6: Prueba de software. Fundamentos teóricos. Principios de la prueba. Métodos de 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.

4. ACTIVIDADES A DESARROLLAR

4.1. Actividades en modalidad virtual

La totalidad de las horas de la asignatura se desarrolla con actividades en modalidad virtual.

CLASES TEÓRICAS: las clases teóricas se desarrollan semanalmente, entre 3 y 5 hs semanales. Se abordan los temas de las unidades del programa, a continuación se brinda la nómina. En algunos temas, previamente a las clases, se brinda material de lectura a los estudiantes para ser discutido y/o expuesto por ellos. Las clases teóricas se realizan por alguna de las plataformas Google Meet o Jitsi Meet (SIAL). Se realizan instancias de seguimiento teóricas que aportan a la evaluación del proceso, en los que los estudiantes responden preguntas teóricas, plantean preguntas, y/o participan de reuniones de discusión de los temas y aportan sus opiniones.

14/4. Primer Seguimiento teórico. lectura Capítulo 1 Pressman/Capítulo 1 O´Regan.
28/4. Segundo Seguimiento teórico UML.
19/5 y 20/5. Tercer Seguimiento teórico Diseño de SW. Patrones de Diseño
30/6. Cuarto Seguimiento teórico integrador.

Nómina de teóricos: Presentación. Introducción. Conceptos básicos de IS. Metodologías. Ciclo de vida en el desarrollo de un software. Ingeniería de Requerimientos del Software. Requerimientos funcionales y no funcionales. Análisis del Problema. SRS c/ más de un método. Modelado de sistemas. Lenguaje de Modelado Unificado (UML: Unified Modeling Language): Introducción. Clases. Relaciones. Diagramas de Clases. Diagramas de Objetos. Ingeniería Inversa y Directa. Diagramas de Casos de Uso. Diagramas de Actividades. Diagramas de Estados. Diagramas de Interacción: Diagrama de Secuencia. Una metodología tradicional: Proceso Unificado (PU). Una metodología Ágil: SCRUM. PU: Captura de Requerimientos: M Negocio + M Casos de Uso. Plantillas Genéricas para la descripción de Casos de Uso. SCRUM: Historias de usuario. Análisis y Diseño de Software. Diseño Funcional.

Diseño de Casos de Uso. Diseño Detallado. PU. Análisis y Diseño. UML. PU: Análisis. Diseño. Plantillas genéricas AyD. Pruebas de Software: Objetivos, Tipos de prueba, Estrategias. Pruebas de Unidad: Pruebas de Caja Negra o Funcional y Pruebas de Caja Blanca o Estructural.

CLASES PRÁCTICAS: P1. Identificación y explicitación de requerimientos funcionales y no funcionales. P2: UML. Diagrama de Clases. Modelo de Dominio. P3. UML. Diagrama de Objetos. P4. Ingeniería Directa e Ingeniería Inversa. P5. Trabajo práctico integrador grupal. Elegir una narrativa, listar RF y RNF, plantear los artefactos MNeg, MCU. Incluir DC, DO, DAct. y enviar por email. P6. Diagrama de Estados. P7. Diagrama de Secuencia. P8. Patrones de Diseño. P9. Pruebas de Software.

5. PROGRAMAS Y/O PROYECTOS PEDAGÓGICOS INNOVADORES E INCLUSIVOS

6. CRONOGRAMA TENTATIVO DE CLASES E INSTANCIAS EVALUATIVAS

6.1. Cronograma de clases e instancias evaluativas a realizar en la virtualidad.

Días y horarios: miércoles 13 a 16 hs, viernes 10-12 hs, martes y jueves 10-12 hs, 16-18 hs.

SEMANA	TEÓRICOS	PRÁCTICOS	TALLER
1	31-03. Presentación. Introducción y Conceptos básicos IS. Metodologías. Ciclo de vida en el desarrollo de un software. Leer Cap 1 O´Regan. Cap 1 Pressman.	Ejercicio modelo de cada práctica para resolver con estudiantes, Ejercicio para resolver individualmente por c/estudiante, mostrar la solución y discutir.	
2	7-04. Ingeniería de requerimientos. Análisis del Problema. SRS c/ más de un método Requerimientos funcionales y no funcionales.	6/4 -8/4. P1. Identificación y explicitación de requerimientos funcionales y no funcionales.	Presentación (Idea - GIT/GitHub)
3	14/4. Primer Seg teórico Cap 1 P/Cap 1 O´P. 14-04. UML. Introducción. Clases. Relaciones. Diagramas de Clases.	13/4. P1. Describir narrativa del problema del taller, explicitación de RF y RNF, analizar estructura del SRS 15/4. P2: UML. Diag Clases. Modelo de Dominio	Metodología (SCRUM - Product Discovery Meeting - SRS)
4	21-04. UML. Diagramas de Clases. Diagramas de Objetos. Ingeniería Inversa y Directa.	20/4. 22/4. P2: UML. Diagrama de Clases. Modelo de Dominio	Toolset (Sinatra Microframework - REST - Monolithic Architecture)
*5	28/4. Segundo Seg teórico UML. 28-04. UML. Diag Casos de Uso. Diag Actividades. Diag Estados. Diag Secuencia	27 y 29/4. P3. UML. Diagrama de Objetos.	KickStart (Docker Sinatra - ActiveRecord Pattern with Sequel)
6	04-5, 05-05. Ingeniería de Requerimientos. Metodología OO: PU. Metodología Ágil SCRUM. PU: Captura de Requerimientos: MN + MCU. Plantillas Genéricas p/desc de CU. SCRUM: Historias de usuario	4/5. P4. ID e II tomar un ejercicio de la P2, pasarlo a eclipse, exportar. Dar código para hacer DC 6/5. P5. TP integrador grupal. Elegir narrativa, RF y RNF, plantear artefactos MNeg, MCU. Incluir DC, DO, DAct..	Retrospectiva Sprint 1
7	711-5, 12-05. Análisis y Diseño de SW. Diseño Funcional. Diseño CU. Diseño Detallado.	11/5 -13 /5 P6. D Estados	Retrospectiva Sprint 1
8	19/5. Tercer Seg teórico Diseño SW. 19-05, 20-05. Patrones de Diseño	18/5. P7. D Secuencia 20/5 P8. Patrones de Diseño	Retrospectiva Sprint 1
9	26-05. Prueba de Software. Prueba estructural	27/5 P8. Patrones de Diseño	Retrospectiva Sprint 1 Retrospectiva Sprint 1
10	2/6. Prueba de Software. Prueba Funcional	1/ 6 - 2/6 - 3/6 P8. Patrones de Diseño	Retrospectiva Sprint 2
11	9-06. Prueba Funcional. PU: Modelo de Prueba.	10/6 P9. Pruebas de Software	Retrospectiva Sprint 2
12	16/6. Completar PU. Análisis y Diseño. PU: Análisis. Diseño. Plantillas genéricas AyD.	15/6 - 17/6. P9. Pruebas de Software	Retrospectiva Sprint 2
13	Implementación, Despliegue y Nodos.	22/6. P9. Pruebas de Software	Presentación/Defensa final
14	martes 29/6 Recuperatorios. miércoles 30/6 Cuarto Seg teórico integrador.		

Instancias Evaluativas

4/5. Parcial. UML: Diagrama de Clases. 4/6. Parcial. Patrones de Diseño. 23/6 Parcial. Prueba de software. 29/6 Recuperatorios de parciales.
25/6 Presentación/Defensa final del Proyecto Taller

7. BIBLIOGRAFÍA

7.1. Bibliografía obligatoria y de consulta.

- Pressman, Roger. Software Engineering: A Practitioner's Approach. 8va edition. McGraw Hill. 2015.
- Sommerville Ian. Ingeniería del Software, 7th edition, Addisson Wesley. Pearson Education, 2005.
- Ghezzi Carlo, Jazayeri M., Mandrioli D.. Fundamentals of Software Engineering. Prentice Hall, 1991.
- Gerard O' Regan. Concise Guide to Software Engineering From Fundamentals to Application Methods. Springer. 2017.
- Kendall y Kendall. Análisis y Diseño de Sistemas. Pearson Education. 2005.
- Pankaj Jalote. An Integrated Approach to Software Engineering. Springer. 2005.
- Booch Grady, Rumbaugh J., Jacobson Ivar. The Unified Modeling Language. Addison Wesley. 1999.
- 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.
- Paul Ammann and Jeff Offutt. Introduction to Software Testing. Cambridge University Press. 2nd Edition. 2016
- Paul C. Jorgensen. Software Testing: A Craftsman's Approach, Auerbach Publications; Fourth Edition. 2013
- Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.
- Grand Mark. Patterns in Java. Vol 1. A Catalog of Reusable Design Patterns Illustred with UML. J Wiley&Sons Inc. 1998.
- Highsmith Jim. Agile Software Development Ecosystems. Addison-Wesley. 2002. ISBN: 0 201-760-13-6.
- Jacobson Ivar, Booch Grady, Rumbaugh James. The Unified Software Development Process. Addison Wesley. 1999.
- Meyer Bertrand. Object Oriented Software Construction. Prentice Hall. 1997.
- OMG. Object Management Group. Unified Modeling Language Specification. http://www.omg.org .
- Pattern-Oriented Software Architecture. Volume 1: A System of Patterns. Frank Buschmann , Regine Meunier , Hans Rohnert , Peter Sommerlad , Michael Stal .
- Rational Unified Process. http://www.rational.com/rup/
- www.agilemanifesto.org - http://www.agile-spain.com/

NOTA: Se utilizan otros materiales de lectura complementarios extraídos de diferentes sitios web, manuales de las herramientas utilizadas, entre otros.

7.2. Plataformas/herramientas virtuales; materiales audiovisuales, enlaces, otros.

Se utiliza google meet y jitsi SIAL para clases y consultas.
Se usa un canal en Slack para informar, generar hilos de debate y discusión, compartir resultados, etc.
Todo el material se carga una carpeta en google drive que se comparte con los estudiantes. Además, todas las clases son grabadas y subidas a dicho repositorio.
También, las presentaciones utilizadas en las clases teóricas y las guías de práctico están disponibles en SIAL, en el aula virtual de la asignatura.

8. DÍA Y HORARIOS DE CLASES VIRTUALES y PRESENCIALES

ver cronograma detallado en 6.1

9. DÍA Y HORARIO DE CLASES DE CONSULTAS VIRTUALES y PRESENCIALES

Las consultas son virtuales y los días y horarios son acordados con los estudiantes. Todos los docentes brindan al menos un horario de consulta semanal, También responden consultas por mail como así también por whatsapp.

10. REQUISITOS PARA OBTENER LA REGULARIDAD Y LA PROMOCIÓN

Requisitos para la regularidad: El estudiante debe participar y aprobar las instancias evaluativas descriptas en 6,1. Debe aprobar los parciales o recuperatorios correspondientes, concluir el desarrollo del taller y presentarlo en la fecha prevista. Además, se realiza una valoración de todo el proceso de cursado y participación del estudiante en todas las actividades planteadas en la asignatura, tales como el práctico integrador grupal y los seguimientos teóricos.

Evaluación Final para estudiantes regulares: se evalúa con exámenes orales teóricos y prácticos.

Evaluación Final para estudiantes libres: el examen incluye el desarrollo de un proyecto de software (20 días aproximadamente), un examen práctico, y un exámenes oral teóricos.

11. CARACTERÍSTICAS, MODALIDAD Y CRITERIOS DE LAS INSTANCIAS EVALUATIVAS

Firma Profesor/a Responsable **Firma Secretario/a Académico/a**