FORMULARIO PARA LA PRESENTACIÓN DE PROGRAMAS DE ASIGNATURAS
Año Lectivo: 2024

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

CARRERA/S: Licenciatura en Ciencias de la Computación
PLAN DE ESTUDIOS: 1999 (vigente)
ASIGNATURA: Testing de Software **CÓDIGO: 3347**
MODALIDAD DE CURSADO: Presencial
DOCENTE RESPONSABLE: Dra. Valeria Bengolea. Prof. Adjunto Simple
EQUIPO DOCENTE: Dra. Valeria Bengolea. Prof. Adjunto Simple
RÉGIMEN DE LA ASIGNATURA: Cuatrimestral
UBICACIÓN EN EL PLAN DE ESTUDIO: 2024. Primer Cuatrimestre
RÉGIMEN DE CORRELATIVIDADES:

Asignaturas aprobadas: Ingeniería de Software. Cod. 3304

Asignaturas regulares: Ingeniería de Software. Cod. 3304

CARÁCTER DE LA ASIGNATURA: Optativa
CARGA HORARIA TOTAL: 112 horas

Teóricas:	**56 hs**	**Prácticas:**	**56 hs**	**Teóricas -Prácticas:**	**…. hs**	**Laboratorio:**	**…. hs**

CARGA HORARIA SEMANAL: 4 horas (según el plan de estudio vigente)

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

1. CONTEXTUALIZACIÓN DE LA ASIGNATURA

Testing de Software se dictará como asignatura optativa de la carrera Licenciatura en Ciencias de la Computación durante el segundo cuatrimestre. La materia presenta los fundamentos del testing sistemático de software, resalta la importancia del testing en las metodologías de desarrollo de software actuales (metodologías ágiles), e intenta mostrar cómo el testing sistemático usualmente se relaciona con un incremento de la calidad del software producido. Además, se presentan en la materia enfoques del estado del arte para automatizar diversas tareas asociadas al testing.

2. OBJETIVOS PROPUESTOS

El testing de software es un tema de suma relevancia práctica para los estudiantes de Ciencias de la Computación. El objetivo es que los estudiantes comprendan la importancia del testing para evaluar y mejorar la corrección funcional del software, y de esta manera aportar al desarrollo de software de calidad.

Los estudiantes aprenderán a desarrollar tests de calidad para programas del mundo real. Entre otros criterios, esto incluye el diseño de tests con buenas capacidades para detectar fallas en el software. Para esto se verán en la materia los criterios de cobertura que dan mejores resultados en la práctica.

Se espera que los estudiantes puedan automatizar en la mayor medida posible los tests producidos, con el objetivo de reducir el costo del testing y posibilitar el testing de regresión (para detectar fallas que aparecen a medida que el programa evoluciona).

Finalmente, se espera que los estudiantes conozcan técnicas y herramientas actuales de generación automática de test que pueden ser utilizadas en la práctica para mejorar la calidad del software desarrollado.

El objetivo esencial de la materia es lograr que los alumnos se familiaricen con nociones básicas referidas al testing de software, que comprendan la importancia de sistematizar esta tarea, conozcan técnicas actuales que ayudan a la automatización de este proceso, y además sepan aplicar estas técnicas utilizando herramientas en la práctica.

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

3.1. Contenidos mínimos

Calidad de Software. Validación vs. Verificación. El rol de la validación y la verificación en la calidad del software. Confiabilidad de software. El problema de construir programas correctos. ¿Qué es un programa correcto? Análisis Estático vs Análisis Dinámico El testing como actividad de validación y verificación.

¿Por qué testeamos el software?. Fallas, errores y defectos. Testing y debugging. Objetivos del testing. Actividades del testing. Modelo de defecto y falla RIPR (Reachability, Infection, Propagation, Revelation).

Automatización de tests unitarios. Framework JUnit. Scripts de test. Suites. Aserciones. Fixtures compartidos y globales. Clasificación en categorías. Tests parametrizados.

Criterios de Cobertura: Particionado del espacio de entradas: Criterios de cobertura de bloques. Todas las combinaciones. Cada selección posible. Cobertura de pares. Cobertura de tuplas. Cobertura de bloque base y múltiples bloques base. Cobertura de grafos. Cobertura de grafos para código fuente. Grafo de Flujo de Control. Cobertura de expresiones lógicas. Predicados y cláusulas. Determinante de un predicado. Criterios de cobertura lógica. Cobertura de predicados y cláusulas. Cobertura combinatoria. Cobertura de cláusulas activas (tradicionalmente llamado MCDC). Testing basado en la sintaxis. Mutación.

Dobles de prueba. Entrada/salida indirecta. Tipos de dobles. Stubs. Mocks. Drivers.

Tests basados en propiedades. Propiedades. Generación automática aleatoria de entradas. Herramientas. Generación Automática de Tests.

3.2. Ejes temáticos o unidades

Unidad 1. Calidad de Software. Validación vs Verificación. Definiciones, diferencias. El rol de la validación y la verificación en la calidad del software. Confiabilidad de software. El problema de construir programas correctos. ¿Qué es un programa correcto? Análisis Estático vs Análisis Dinámico El testing como actividad de validación y verificación. Conceptos fundamentales de testing, sus objetivos y principios.

Unidad 2: Motivación: ¿Por qué testeamos el software?. Fallas, errores y defectos. Testing y debugging. Objetivos del testing. Actividades del testing. Modelo de defecto y falla RIPR (Reachability, Infection, Propagation, Revelation). Testing unitario. Testing de módulo. Testing de integración. Testing de sistema. Testing de aceptación. Criterios de cobertura. Ventajas. Otros tipos de testing. Testing de regresión. Testing diferencial.

Unidad 3: Automatización de tests unitarios. Framework JUnit. Scripts de test. Suites. Aserciones. Fixtures compartidos y globales. Clasificación en categorías. Tests parametrizados.

Unidad 3: Criterios de Cobertura: Particionado del espacio de entradas. Características, particiones y bloques. Enfoques para determinar características basadas en la interfaz y en la funcionalidad. Testing Combinatorial. Criterios de cobertura de bloques. Todas las combinaciones. Cada selección posible. Cobertura de pares. Cobertura de tuplas. Cobertura de bloque base y múltiples bloques base. Restricciones entre diferentes características. Subsunción entre criterios de cobertura de bloques. Aplicaciones a testing de unidad. Herramientas

Unidad 4: Criterios de Cobertura: Cobertura de grafos. Definiciones: grafos, caminos, caminos de test. Caminos simples y primos. Criterios de cobertura de grafos. Cobertura de nodos y aristas. Cobertura de pares de aristas. Cobertura de caminos. Cobertura de caminos primarios. Subsunción entre criterios de cobertura de grafos. Cobertura de grafos para código fuente. Grafo de Flujo de Control.

Unidad 5: Criterios de Cobertura: Cobertura de expresiones lógicas. Predicados y cláusulas. Determinante de un predicado. Criterios de cobertura lógica. Cobertura de predicados y cláusulas. Cobertura combinatoria. Cobertura de cláusulas activas (tradicionalmente llamado MCDC). Cobertura general, restringida y correlacionada de cláusulas activas. Cobertura de cláusulas inactivas. Subsunción de criterios de cobertura lógica. Aplicaciones a testing de unidad.

Unidad 6: Criterios de Cobertura: Testing basado en la sintaxis. Generación de entradas basada en gramáticas. Gramáticas como reconocedores y generadores. Cobertura de gramáticas. Cobertura de símbolos terminales y de producciones. Cobertura de todas las cadenas. Mutación. Operadores de mutación. Generación de entradas válidas e inválidas basada en mutación. Cobertura de mutantes (matar mutantes). Mutación para código fuente. Ventajas y problemas. Mutación débil y fuerte. Mutantes equivalentes. Operadores de mutación para código fuente. Aplicaciones a testing de unidad. Herramientas.

Unidad 7: Dobles de prueba. Entrada/salida indirecta. Tipos de dobles. Stubs. Mocks. Drivers. Estrategias de integración: bottom-up y top-down. Herramientas.

Unidad 8 Tests basados en propiedades. Propiedades. AAAA (*Assume, Arrange, Act. Assert)*. Generación automática aleatoria de entradas. Herramientas.

Unidad 9. Generación Automática de Tests. Generación Aleatoria de Casos de Tests. Generación de Casos de Tests basada en algoritmos genéticos. Generación Exhaustiva acotada de Casos de Tests. Generación basada en Constraint Solving (SMT Solving). Herramientas que asisten los diferentes procesos de generación automática. Medición de calidad de las Test Suites generadas automáticamente. Problemas de escalabilidad y aplicabilidad del testing automático

4. ACTIVIDADES A DESARROLLAR

CLASES TEÓRICAS: 4 horas presenciales semanales.

En las clases teóricas se presentarán los fundamentos teóricos del testing de software.

CLASES PRÁCTICAS: 4 horas presenciales semanales.

Las clases prácticas estarán acompañadas de guías de ejercicios (de resolución obligatoria) para llevar a la práctica los conceptos teóricos abordados.

Se fomentará el uso de lenguajes de programación para la implementación de soluciones a problemas de testing concretos. En lo posible, se proveerán bibliotecas para promover la automatización del testing. En las guías prácticas, se ejercitarán los conceptos teóricos mediante ejercicios de programación (codificación de tests, desarrollo de software, uso de herramientas de testing, etc.).

Se fomentará la lectura de material adicional y la autoorganización de los alumnos en sus actividades. Además, se dejará en manos de los alumnos la instalación y manejo de las herramientas de software utilizadas en la asignatura, como una manera de estimular la práctica en cuestiones más técnicas (no necesariamente ligadas a los tópicos que la asignatura abarca), y la experiencia en la utilización de herramientas nuevas.

CLASES DE TRABAJOS PRÁCTICOS DE LABORATORIO: --

OTRAS: --

5. PROGRAMAS Y/O PROYECTOS PEDAGÓGICOS INNOVADORES E INCLUSIVOS

6. CRONOGRAMA TENTATIVO DE CLASES E INSTANCIAS EVALUATIVAS

Semana	Clase/Horas	Actividad: tipo y descripción*
1	Martes 13/08 2hs	Teoría. Presentación Modalidad. Introducción: Calidad de Software.
	Jueves 15/08 2hs	Teoría: ¿Por qué testeamos el software?

	Viernes 16/08 4hs	Teoría (Continuación): ¿Por qué testeamos el software? Práctica: Defectos, Fallas y Errores.
2	Martes 20/08 2hs	Teoría: Automatización de Test. Frameworks. Fixtures compartidos y globales. Test Parametrizados.
	Jueves 22/08 2hs	Práctica: Automatización de tests, fixtures y test parametrizados: JUnit
	Viernes 23/08 4hs	Práctico: Automatización de tests, fixtures y test parametrizados: JUnit
3	Martes 27/08 2hs	Teoría: Criterios de Cobertura. Requisitos de Tes. Subsunción de criterios. Criterio de Cobertura. Particionado del espacio de entradas.
	Jueves 29/08 2hs	Teoría (Continuación): Particionado del espacio de entradas. Herramientas.
	Viernes 30/08 4hs	Práctico: Particionado del espacio de entradas. Herramientas.
4	Martes 3/9 2hs	Práctico: Particionado del espacio de entradas. Herramientas.
	Jueves 5/9 2hs	Práctico: Particionado del espacio de entradas. Herramientas
	Viernes 6/9 4hs	Criterios de Cobertura: Cobertura de grafos. Cobertura de grafos para código fuente. Grafo de Flujo de Control.
5	Martes 10/9 2hs	Práctico: Criterios de Cobertura: Cobertura de grafos. Cobertura de grafos para código fuente. Grafo de Flujo de Control.
	Jueves 12/9 2hs	Práctico: Criterios de Cobertura: Cobertura de grafos. Cobertura de grafos para código fuente. Grafo de Flujo de Control.
	Viernes 13/9 4hs	Teórico: Criterios de Cobertura: Cobertura de expresiones lógicas
6	Martes 17/09 2hs	Práctico: Criterios de Cobertura: Cobertura de expresiones lógicas
	Jueves 19/09 2hs	Práctico: Criterios de Cobertura: Cobertura de expresiones lógicas
	Viernes 20/09 4hs	Criterios de Cobertura: Testing basado en la sintaxis. Testing de Mutación.
7	Martes 24/09 2hs	Práctico: Testing de Mutación. Herramientas automáticas.

	Jueves 26/09 2hs	Práctico: Testing de Mutación. Herramientas automáticas.
	Viernes 27/09 4hs	Práctico: Testing de Mutación. Herramientas automáticas. Teórico: Dobles de Pruebas Frameworks
8	Martes 1/10 2hs	Práctico: Dobles de Pruebas Frameworks
	Jueves 3/10 2hs	Práctico: Dobles de Pruebas Frameworks
	Viernes 4/10 4hs	Teoría: Testing Basado en Propiedades. Generación aleatoria de entradas. Herramientas Práctico: Testing Basado en Propiedades
9	Martes 8/10 2hs	Práctico: Testing Basado en Propiedades
	Jueves 10/10 2hs	Práctico: Testing Basado en Propiedades
	Viernes 11/10 4hs	Feriado puente. Día del respeto a la diversidad cultural.
10	Martes 15/10 2h	Práctico: Testing Basado en Propiedades:
	Jueves 17/10 2hs	Teoría: Generación Automática de Test. Generación Aleatoria Testing de Regresión. Contratos. Especificaciones operativas. RepOK. Criterios de cobertura como métricas para medir la calidad de las suites.
	Viernes 18/10 4hs	Teoría (Continuación) Generación Aleatoria de Test Práctica: Generación Aleatoria Testing de Regresión. Demo práctica de uso de herramientas
11	Martes 22/10 2hs	Teoría: Generación Automática de Test. Generación basada en algoritmos genéticos. Revisión de conceptos relacionados a algoritmos genéticos.
	Jueves 24/10 4hs	Práctica: Generación Automática de Test (incluyendo las técnicas vistas)
	Viernes 25/10 4hs	Teoría: Generación basada en Constraint Solving (SMT Solving) Práctica: Generación Automática de Test (incluyendo las técnicas vistas) **Presentación Trabajo Práctico Obligatorio**
12	Martes 29/10 2hs	Teoría: Generación Exhaustiva Acotada de entradas. Especificaciones. repOk.

	Jueves 31/10 2hs	Práctica: Generación Exhaustiva Acotada de entradas. Especificaciones. repOk.
	Viernes 1/11 4hs	Teoría: Revisión de conceptos necesarios para la resolución del trabajo práctico. Práctica: Generación Exhaustiva Acotada de entradas. Especificaciones. repOk.
13	Martes 5/11 2hs	Práctica: Resolución de ejercicios práctico + Trabajo práctico obligatorio
	Jueves 7/11 2hs	Práctica: Resolución de ejercicios práctico + Trabajo práctico obligatorio
	Viernes 8/11 4hs	**Fecha límite para la entrega del Trabajo Práctico Obligatorio**
14	Martes 12/11 2hs	Evaluación oral individual En esta instancia de evaluación cada alumno es citado de manera individual en un horario predeterminado
	Jueves 14/11 2hs	**Fecha límite para la entrega del Trabajo Práctico Obligatorio (Recuperatorio)** Evaluación oral individual En esta instancia de evaluación cada alumno es citado de manera individual en un horario predeterminado
	Viernes 15/11 4hs	Evaluación oral individual En esta instancia de evaluación cada alumno es citado de manera individual en un horario predeterminado

7. BIBLIOGRAFÍA

7.1. Bibliografía obligatoria y de consulta

Obligatoria

Paul Ammann, Jeff Offutt.Introduction to Software Testing (2nd edition). Cambridge University Press. 2017.

De consulta

Joshua Bloch, Effective Java, Second Edition, 2000

Glenford J. Myers, The Art of Software Testing, John Wiley & Sons, Inc. 2012.

B. Meyer, “Object Oriented Software Construction” second edition, Addison-Wesley, 2000.

B. Liskov, “Program Development in Java, Abstraction, Specification and Object-Oriented Design”, Addison-Wesley, 2001.

7.2. Otros: materiales audiovisuales, enlaces, otros.

Slack (comunicaciones, material, etc.)

Google Classroom (comunicaciones, material, videos grabados de las clases en vivo)

Github (templates de programación con tests para autocorrección, repositorio de código para interacción entre estudiantes y docentes)

Tutoriales y manuales de las herramientas utilizadas.

8. DÍA Y HORARIOS DE CLASES

Clases Teórico-Prácticas:
martes de 14 a 16.
jueves de 16 a 18.
Viernes de 14 a 18.

9. DÍA Y HORARIO DE CLASES DE CONSULTAS

Las clases de consulta se dictarán semanalmente, en horario a convenir con los estudiantes.

10. REQUISITOS PARA OBTENER LA REGULARIDAD Y LA PROMOCIÓN

CONDICIONES DE REGULARIDAD: Aprobar las guías prácticas obligatorias (de entrega grupal), un trabajo práctico grupal y una defensa oral individual cada uno con nota cinco o mayor.

CONDICIONES DE PROMOCIÓN: Aprobar las guías prácticas (con nota cinco o mayor). Aprobar el trabajo práctico grupal y la defensa oral con notas mayor o igual a 7.

11. CARACTERÍSTICAS, MODALIDAD Y CRITERIOS DE LAS INSTANCIAS EVALUATIVAS

Trabajos prácticos obligatorios:

La materia contará con una guía de ejercicios prácticos por cada una de las unidades de la asignatura. Los estudiantes podrán resolver estas guías prácticas durante las clases prácticas de la asignatura con asistencia del docente. La resolución de estas guías prácticas será grupal y obligatoria. Se utilizará github classroom para lograr un trabajo grupal colaborativo, permitiendo por otra parte la interacción continua con el docente. El objetivo de estas guías prácticas será evaluar la aplicación de las técnicas aprendidas y la correcta comprensión de los fundamentos teóricos subyacentes a las técnicas estudiadas. **La resolución obligatoria de estas guías mediante el uso de repositorio de código colaborativos tiene como objetivo fundamental mantener un seguimiento continuo del proceso de enseñanza-aprendizaje**.

La materia contará además con una trabajo práctico grupal con una instancia de evaluación oral (defensa) individual. El plazo para la resolución del trabajo práctico es de dos semanas. El trabajo práctico obligatorio contará con una instancia de recuperación. El objetivo de este trabajo es evaluar la aplicación de las técnicas aprendidas en programas concretos de tamaño mediano, y la correcta comprensión de los fundamentos teóricos subyacentes a las técnicas estudiadas.

Evaluación Final: El examen final para alumnos regulares se llevará a cabo mediante evaluación oral, si el número de alumnos evaluados así lo permite. Abarcará la totalidad de los contenidos de la asignatura, verificando que los alumnos hayan adquirido los conocimientos teóricos y puedan aplicarlos en casos concretos en la práctica.

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