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

ASIGNATURA: Testing de Software **CÓDIGO: 3347**

DOCENTE RESPONSABLE: Dr. Nazareno Aguirre

EQUIPO DOCENTE: Dr. Renzo Degiovanni, Dra. Valeria Bengolea, Dr. Pablo Ponzio

AÑO ACADÉMICO: 2016

REGIMEN DE LA ASIGNATURA: Cuatrimestral

RÉGIMEN DE CORRELATIVIDADES:

Para cursar:

Aprobada	***Regular***
-	Ingeniería de Software (c.3304)

Para rendir

Aprobada
Ingeniería de Software (c.3304)

CARGA HORARIA TOTAL: 8 horas semanales, 112 horas totales.

TEÓRICAS: 4 hs **PRÁCTICAS:** 4 hs **LABORATORIO:**

CARÁCTER DE LA ASIGNATURA: Optativa

A. CONTEXTUALIZACIÓN DE LA ASIGNATURA

Testing de Software será, durante el primer cuatrimestre de 2016, asignatura optativa de la Licenciatura en Ciencias de la Computación. La materia presenta los principios fundamentales para el *testing* sistemático de *software*, uso de herramientas de soporte al *testing*, e introducción a la automatización de este proceso.

B. OBJETIVOS PROPUESTOS

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.

C. CONTENIDOS BÁSICOS DEL PROGRAMA A DESARROLLAR

Calidad de Software. Validación y Verificación. Análisis estático y dinámico. Inspección manual de código. *Testing* manual. El *Testing* en el proceso de desarrollo de software. Procesos Agiles. El *testing* como especificación de programas. Clasificaciones básicas de tipos de *testing*.

Testing Unitario. *Testing* ad-hoc. Sistematización del *Testing* de Software. *Testing* Unitario. Introducción de *JUnit*. Casos de *tests* positivos y negativos. *Suites* de *Tests*. Datos compartidos para *tests*: SetUp y TearDown. Independencia de *tests*. *Tests* Parametrizados. Teorías.

Criterios de Evaluación de Suites de Test. Criterios de Cobertura. Enfoques tradicionales. *Testing* de Caja Negra. Clases de Equivalencias. Análisis de Valores Bordes. *Testing* de Caja Blanca. Cobertura de Sentencias. Cobertura de Decisión. Cobertura de Caminos. Grafos de Flujo de Control. Herramientas de medición de cobertura. *Testing* basado en Mutación. Herramientas de soporte para *Testing* de Mutación.

Dobles de Prueba. Tipos de dobles. Dummy. Stub. Spy. Fake. Mocks. Herramientas de soporte para dobles simples. Mocking para Bases de Datos.

Otros Tipos de Testing. *Testing* de Aceptación. *Testing* de Regresión. *Testing* Diferencial. *Testing* de Integración. *Testing* de Interfaces de usuario.

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. *Testing* basado en Especificaciones de Requisitos. *Testing* basado en Modelos. Localización y

Reparación de fallas usando *Testing*. Avances recientes relacionadas el *testing* automático.

D. FUNDAMENTACIÓN DE LOS CONTENIDOS

El *testing* consiste en ejecutar una pieza de código utilizando diversas entradas, y examinar si el resultado obtenido en cada ejecución es el esperado (no representa un error).

Es ampliamente aceptado que al incorporar el testing al proceso de desarrollo de software, de una forma controlada y sistemática, es posible detectar un gran número de errores en el software antes de ser lanzado para su uso, mejorando significativamente la calidad del producto final.

Por esta razón, el *testing* juega actualmente un rol muy importante en las metodologías modernas (ágiles) de desarrollo, ya sea como guía del desarrollo (por ejemplo, en *test driven development*), o como método de detección de errores durante el desarrollo.

Además, diversos estudios estiman que usualmente el costo del *testing* equivale a más de la mitad del presupuesto total para el desarrollo de un software.

Lo anterior muestra que el *testing* es un tema de suma relevancia práctica, y una herramienta esencial en la formación de un profesional de las Ciencias de la Computación.

En la materia se tratarán los principios que se consideran esenciales para realizar un el *testing* exitoso (con una alta tasa de detección de errores) en la práctica. Entre ellos se destacan las técnicas de construcción de *tests* con un alto grado de detección de errores, las herramientas de sistematización del *testing* (para facilitar la ejecución de los *tests*), la evaluación de la calidad de las suites de *tests* (criterios de cobertura, mutación, etc.) y el mejoramiento de las mismas, y la generación de casos de *tests* asistida por herramientas especializadas de *software*.

Se elegirán cuidadosamente los lenguajes de programación a utilizar durante el curso, en particular garantizando la disponibilidad de herramientas de *software* para asistir a las actividades de *testing*, con el objetivo de mostrar que estas herramientas alivian significativamente la carga de trabajo, y son esenciales en la construcción de *suites* de *tests* de calidad.

En la materia se hará énfasis en la aplicación práctica de los conceptos teóricos enseñados, usando para esto programas de simple a mediana complejidad que serán provistos por los docentes.

E. ACTIVIDADES A DESARROLLAR

Se pondrá especial énfasis en la aplicación de las principales técnicas y herramientas para la sistematización y automatización del testing de software. Las clases serán teórico-prácticas. Se fomentará el uso de lenguajes de programación para la implementación de soluciones a problemas concretos, que integren librerías que asistan el proceso de testing (los lenguajes elegidos cuentan, todos ellos, con herramientas de soporte).

La materia contará con tres trabajos prácticos obligatorios, cuya aprobación es requisito para la regularidad, que tienen como objetivo aplicar la teoría aprendida en programas reales de complejidad media intentando estimular al alumnado. Más específicamente, se espera que los alumnos sean capaces de desarrollar suites de tests (con o sin asistencia de herramientas automáticas) con una alta tasa de detección de errores, evaluar la calidad de dichas suites, y mejorarlas para incrementar su capacidad de detección de errores. Se buscará que los trabajos prácticos hagan evidentes lo compleja y costosa que puede ser la actividad de testing, y la necesidad de contar con herramientas que sistematicen. También se intentará mostrar como las herramientas automáticas pueden aliviar el esfuerzo requerido para la construcción de suites de test.

Como en la actualidad el testing es una parte central de las metodologías ágiles de desarrollo de software, se repasarán y ejercitarán temas relacionados a este tipo de metodologías, reforzando los conocimientos de los alumnos en Ingeniería de Software (algunos de ellos vistos con anterioridad en la materia Ingeniería de Software). Además, se busca afianzar los conocimientos de los alumnos en especificación formal de programas, centrales a las herramientas automáticas de asistencia al testing.

Se fomentará la lectura de material adicional y la auto organización 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.

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.

CLASES TEÓRICAS: **4** horas semanales
CLASES PRÁCTICAS: **4** horas semanales

F. NÓMINA DE TRABAJOS PRÁCTICOS

La materia contará con tres trabajos prácticos obligatorios, cada uno con una instancia de recuperación, que serán evaluados por los docentes de la materia y su aprobación es condición para la regularidad. El plazo para la resolución de cada uno de los trabajos prácticos es de dos semanas.

Además, se proveerá una serie de ejercicios adicionales (cuya resolución es opcional) que acompañarán cada una de las clases teórico-prácticas.

El objetivo de los trabajos prácticos obligatorios (cuya resolución deberá ser individual) es evaluar la aplicación de las técnicas aprendidas en situaciones concretas de tamaño mediano, y la correcta comprensión de los fundamentos teóricos subyacentes a las técnicas estudiadas.

G. HORARIOS DE CLASES:

El dictado de la asignatura estará compuesto por cuatro horas semanales de clases teóricas, y cuatro horas semanales de clases prácticas. Los horarios de las mismas serán definidos de común acuerdo con los alumnos, dado que la materia puede cursarse tanto en cuarto como quinto año. Los siguientes son horarios sugeridos.

Clases Teóricas:

Lunes de 16 a 18.
Viernes de 16 a 18

Clases Prácticas:

Miércoles de 18 a 20
Viernes de 18 a 20

HORARIO DE CLASES DE CONSULTAS: Las clases de consulta se darán semanalmente bajo demanda de los alumnos, en horario a convenir con los mismos.

H. MODALIDAD DE EVALUACIÓN:

Evaluaciones Parciales: La materia contará con tres trabajos prácticos obligatorios, cada uno con una instancia de recuperación, que serán evaluados por los docentes de la materia y su aprobación es condición para la regularidad. El plazo para la resolución de cada uno de los trabajos prácticos es de dos semanas.

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.

CONDICIONES DE REGULARIDAD: Aprobar los tres trabajos prácticos obligatorios.

CONDICIONES DE PROMOCIÓN: Aprobar trabajos prácticos, todos con nota mayor o igual a 6 (seis), y 7 (siete) en promedio.

PROGRAMA ANALÍTICO

A. CONTENIDOS

Son los mismos que pusimos anteriormente en el punto C de contenidos básicos.

B. CRONOGRAMA DE CLASES Y PARCIALES

Semana	Día/Fecha	Teóricos	Día/Fecha	Prácticos	Exámenes
1	14/3 al 18/3	Calidad de Software. Análisis estático y dinámico. Inspección manual de código. Testing manual.	14/3 al 18/3	Práctica 1:Testing Manual.	
2	21/3 al 25/3	Sistematización del Testing de Software. Testing Unitario.	21/3 al 25/3	Práctico 2: Testing Unitario. Sistematización del Testing de Software	
3	28/3 al 01/4	Testing en el proceso de desarrollo de software. Procesos Ágiles.	28/3 al 01/4	Práctica 3: Testing y Procesos de desarrollo.	
4	4/4 al 8/4	Criterios de Evaluación de Suites de Test: Caja Negra y Caja Blanca. Herramientas de Medición de Cobertura.	4/4 al 8/4	Práctica 4: Criterios de Evaluación de Suites de Test.	
5	11/4 al 15/4	Testing de Mutación	11/4 al 15/4	Práctica 5: Testing de Mutación. Herramientas Automáticas	Presentación Trabajo Práctico 1
6	18/4 al 22/4	Dobles de Prueba.	18/4 al 22/4	Práctico 6: Dobles de Prueba	
7	25/4 al 29/4	Testing de interfaces de Usuario	25/4 al 29/4	Práctico 7: Dobles de Prueba y Testing de Interfaces de Usuario	
8	2/5 al 6/5	Otros tipos de testing: Testing de Aceptación. Testing de Regresión.	2/5 al 6/5	Práctico 8: Otros Tipos de Testing	
9	9/5 al 13/5	Otros tipos de testing: Testing Diferencial. Testing de Integración	9/5 al 13/5	Práctico 8: Otros Tipos de Testing	Presentación Trabajo Práctico 2
10	16/5 al 20/5	Generación Aleatoria de Casos de Tests.	16/5 al 20/5	Práctico 9: Generación Automática de Casos de Test	
11	23/5 al 27/5	Generación de Casos de Tests basada en algoritmos genéticos.	23/5 al 27/5	Práctico 9: Generación Automática de Casos de Test	
12	30/5 al 3/6	Generación Exhaustiva Acotada.	30/5 al 3/6	Práctico 9: Generación Automática de Casos de Test	
13	6/6 al 10/6	Generación basada en *Constraint Solving*. Problemas de escalabilidad y aplicabilidad del testing automático	6/6 al 10/6	Práctico 9: Generación Automática de Casos de Test	Presentación Trabajo Práctico 3
14	13/6 al 17/6	Testing basado en especificaciones de Requisitos. Testing basado en Modelos. Localización y Reparación de fallas	13/6 al 17/6		Defensa Oral de Trabajos Prácticos

		usando testing. Avances recientes relacionados al testing automático.			

C. BIBLIOGRAFÍA

Obligatoria

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

C. Pacheco, S. Lahiri, M. Ernst, and T. Ball, *Feedback-directed random test generation*, ICSE 2007.

K. Claessen and J. Hughes, *QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs*, ICFP 2000.

G. Fraser and A. Arcuri, *Evolutionary Generation of Whole Test Suites*, QSIC 2011.

C. Boyapati, S. Khurshid, and D. Marinov, *Korat: Automated Testing Based on Java Predicates*, ISSTA 2002.

N. Tillmann and J. de Halleux, *Pex: White Box Test generation for .NET*, TAP 2008.

P. Ammann, J. Offutt. Introduction to Software Testing. Cambridge University Press. 2008.

De consulta

Tutoriales y manuales de las herramientas utilizadas.