Universidad Nacional de Rio Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales

FORMULARIO PARA LA PRESENTACIÓN DE PROGRAMAS DE ASIGNATURAS

Año Lectivo: 2023

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 versión 1 (vigente)

ASIGNATURA: Validación Verificación de Software **CÓDIGO:** 3308

MODALIDAD DE CURSADO: Presencial

DOCENTE RESPONSABLE: Valeria Susana Bengolea, Dra. en Ciencias de la Computación, Profesora Adjunta Exclusiva

EQUIPO DOCENTE: Valeria Susana Bengolea, Dra. en Ciencias de la Computación, Profesora Adjunta Exclusiva

RÉGIMEN DE LA ASIGNATURA: cuatrimestral

UBICACIÓN EN EL PLAN DE ESTUDIO: quinto año, segundo cuatrimestre

RÉGIMEN DE CORRELATIVIDADES:

Asignaturas aprobadas: Programación Avanzada (1948)

Asignaturas regulares: Análisis Comparativo de Lenguajes (1956)

CARÁCTER DE LA ASIGNATURA: Optativa

CARGA HORARIA TOTAL: 112 horas

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

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

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

1. CONTEXTUALIZACIÓN DE LA ASIGNATURA

Validación y Verificación de Software se presenta como asignatura optativa de la Licenciatura en Ciencias de la Computación. Según el plan de estudios de la Licenciatura en Ciencias de la Computación, pueden cursarse materias optativas en los años cuarto y quinto de la misma. Las correlatividades fijadas permiten que esta asignatura pueda ser cursada en cualquiera de estos años.

2. OBJETIVOS PROPUESTOS

El objetivo esencial de la materia es lograr que los alumnos se familiaricen con las técnicas más importantes de validación y verificación de software disponibles en la actualidad, que comprendan los fundamentos teóricos que hacen posibles la aplicación de estas técnicas en la práctica, y sepan aplicarlas.

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

3.1. Contenidos mínimos

Calidad del software. Confiabilidad de software. El problema de construir programas correctos.

Especificaciones de programas. Revisión de enfoques tradicionales a la validación y verificación de programas. Programación basada en contratos.

Testing. Conceptos fundamentales de *testing*, sus objetivos y principios. Pruebas unitarias. Automatización de testing usando *frameworks xUnit*. Criterios de cobertura. Testing Basado en Propiedades. El concepto de doble de prueba y su necesidad.

Testing y sus limitaciones. Problemas en la automatización de las técnicas tradicionales de validación y verificación de programas. Generación automática de casos de *tests*.

El problema de construir y razonar sobre programas concurrentes. El álgebra de procesos FSP. Propiedades de *safety* y *liveness* en FSP, y su análisis mediante LTSA.

Comprobación de modelos (*model checking*). Definiciones fundamentales. Lógicas temporales. Automatización en la verificación de propiedades temporales mediante el *model checking*.

La lógica como lenguaje formal y su expresividad. Lógica proposicional. Reducción del problema de la deducción a *SAT solving* proposicional. La lógica relacional. Descripción de diseños y propiedades estructurales mediante la lógica relacional. El lenguaje de especificaciones *Alloy*. La extensión *DynAlloy*.

3.2. Ejes temáticos o unidades

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. El *testing* como guía en la construcción de software. Pruebas unitarias: conceptos de prueba y granularidad. Automatización de testing usando frameworks xUnit. Estructuración y composición de tests de unidad. Construcción de suites de testing. Revisión de criterios y técnicas de diseño de casos de prueba. Criterios de cobertura: Testing de caja negra y caja blanca. Testing de Mutación. Herramientas de medición de cobertura. Independencia en unidades de test. Herramientas. *Debugging* y *Testing*. Testing basado en propiedades. Dobles de prueba. Importancia de los dobles de prueba. Tipos de dobles: *dummies, stubs, mocks*, etc. Herramientas de apoyo.

Necesidad de las especificaciones de programas. Revisión de enfoques tradicionales a la validación y verificación de programas. Testing y sus limitaciones. Problemas en la automatización de las técnicas tradicionales de validación y verificación de programas. Generación Automática de Tests: Técnicas y Herramientas novedosas de Generación Automática de Casos de Tests. Ejecución Simbólica. Problemas de escalabilidad y aplicabilidad del *testing* automático.

El problema de construir y razonar sobre programas concurrentes. Algunas técnicas clásicas para la verificación de programas concurrentes. Las álgebras de procesos. El álgebra de procesos FSP. Modelado de procesos concurrentes en FSP. FSP en el modelado de objetos compartidos y exclusión mutua. Diferentes modelos de interacción entre procesos. Propiedades de *safety* y *liveness* en FSP, y su análisis mediante LTSA.

Comprobación de modelos (*model checking*). Lógica modal. Definiciones fundamentales. Operadores modales y sus distintas interpretaciones. Estructuras de Kripke. Relación entre estructuras de Kripke y la semántica de programas. Lógica temporal lineal como una lógica modal. Descripción de propiedades relacionadas con el tiempo mediante la utilización de operadores modales.
Sintaxis y semántica formales de la lógica temporal lineal proposicional. Autómatas de Büchi y su relación con modelos de la lógica temporal lineal proposicional. Traducción de interpretaciones y fórmulas de la lógica temporal lineal proposicional a autómatas de Büchi. Reducción de la comprobación de modelos en lógica temporal lineal proposicional a un

problema en Teoría de Autómatas. Automatización en la verificación de propiedades temporales mediante el *model checking*. Ejemplos de su uso en la práctica. La concurrencia y las dificultades en la comprensión de programas concurrentes. Utilización de herramientas de *model checking*. Limitaciones del *model checking* en relación a la lógica temporal lineal proposicional.

La lógica como lenguaje formal y su expresividad. Lógica proposicional. Sus ventajas y sus limitaciones en poder expresivo. Reducción del problema de la deducción a SAT solving proposicional. Automatización de la deducción. Restricciones para la aplicación de SAT solving para deducción en lógica de primer orden. Representación de diseños de sistemas mediante lógica de primer orden. La lógica relacional. Descripción de diseños y propiedades estructurales mediante la lógica relacional. El poder de SAT solving de primer orden como una técnica de validación de sistemas. Limitaciones. La herramienta *Alloy*. Definición de diseños mediante signaturas y funciones de transformación de estados. Validación de diseños en *Alloy*. Utilización en la práctica. Ejemplos. *Alloy* y sus limitaciones para razonar sobre aspectos dinámicos de programas. La extensión *DynAlloy* y sus aplicaciones.

4. ACTIVIDADES A DESARROLLAR

CLASES TEÓRICAS: 2 clases teóricas presenciales de 2hs cada una. En las mismas, se utilizarán presentaciones y se realizarán demostraciones de uso de herramientas de validación y verificación de software utilizadas como parte de cada unidad. Estarán disponibles, en el aula virtual del curso las presentaciones utilizadas como material de estudio**.**

CLASES PRÁCTICAS: 2 clases prácticas presenciales de laboratorio de 2 horas cada una. Se resolverá de manera conjunta algunos de los ejercicios que se presenta en las guías de trabajo práctico y se ayudará a que el alumnado pueda completar el resto de los ejercicios de la guía. Se utilizarán las herramientas de validación y verificación de software propuestas en cada unidad. Las guías de trabajos prácticos estarán disponibles con anterioridad a la clase en el aula virtual del curso. Cada guía de trabajo práctico tendrá el código fuente asociado en un repositorio de código git.

CLASES DE TRABAJOS PRÁCTICOS DE LABORATORIO: Todas las clases prácticas requieren de computadoras (laboratorio informático).

OTRAS: La materia contará con dos trabajos prácticos (grupales) de entrega y aprobación obligatoria, que serán evaluados por la docente de la materia. Para el desarrollo de cada uno de estos trabajos se utilizará una plataforma de desarrollo y control de versiones usando *git*. Esto ayudará al seguimiento de parte de la docente de cada uno de los grupos de alumnos. Por otra parte, habrá una instancia de evaluación oral individual (defensa de trabajos prácticos) que se realizará la última semana de cursando. Se acordará horario con cada uno de los alumnos. El plazo para la resolución de cada uno de los trabajos prácticos es de dos semanas.

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 15/08 2hs	Teórico. Introducción. Metodología/herramientas a utilizar en la materia. Introducción al Análisis Automático. Presentación de algunas herramientas de análisis.
	Jueves 17/08 4hs	Teórico: Calidad en el desarrollo de software. Especificaciones en V&V. Diseño por Contrato. Invariante de representación.Testing: Conceptos Básicos. Test Unitarios
	Viernes 18/08 2hs	Práctico: Especificaciones en V&V. Diseño por Contrato. Invariante de representación. Testing Unitario
2	Martes 22/08 2hs	Teórico: . Test Parametrizados. Testing basado en Propiedades. Herramientas de Apoyo
	Jueves 24/08 4hs	Teórico: Testing basado en Propiedades, Herramientas de Apoyo (Continuación).
	Viernes 25/08 2hs	Práctico: Testing Unitario. Test Parametrizados.
3	Martes 29/08 2hs	Teórico: Criterios de Cobertura. Clasificación Caja Blanca. Testing de Mutación. Conceptos y herramientas avanzadas. Herramientas de medición para *testing* de unidad. Testing de Mutación. Herramientas.
	Jueves 31/08 4hs	Teórico: Criterios de Cobertura. Caja Negra. Herramientas de Apoyo al Testing de Caja Negra. Test basados en Propiedades. Generadores aleatorios de Entradas simples. Factorías para objetos complejos. Herramientas
	Viernes 1/09 2hs	Práctico: Criterio de Cobertura: Caja Blanca/Caja Negra Práctico: Testing basados en Propiedades
4	Martes 5/9 2hs	Teórico: Dobles de pruebas
	Jueves 7/9 4hs	Teórico: Generación Automática de Test. Técnicas y herramientas Generación Automática Aleatoria Práctico dobles + Introducción al práctico práctico Generación Automática Aleatoria
	Viernes 8/9 2hs	Práctico dobles + práctico práctico Generación Automática Aleatoria
5	Martes 12/9 2hs	*Teórico: Generación Automática de test. Técnicas y herramientas (Continuación) Generación basada en algoritmos genéticos*
	Jueves 14/9 4hs	Teórico: Generación Automática de test. Técnicas y herramientas (Continuación) Generación basada en búsqueda

		Práctico: Demostración de instalación/uso de herramientas.
	Viernes 15/9 2hs	Práctico : Generación automática de casos de test
6	Martes 19/09 2hs	Teórico: Ejecución Simbólica y Testing Automático
	Jueves 21/09 4hs	Asueto Día del Estudiante
	Viernes 22/09 2hs	***Introducción del primer trabajo práctico obligatorio*** ***Repaso General*** Práctico. Generación Automática de Test, repaso general para la resolución del Trabajo Práctico Obligatorio
7	Martes 26/09 2hs	Teórico: Introducción a la Verificación de sistemas Concurrentes. Propiedades de Sistemas Concurrentes. Introducción al Análisis de propiedades en FSP
	Jueves 28/09 4hs	Teórico: Sistemas Concurrentes y su Corrección. Modelos de Software. El Lenguaje FSP y la herramienta LTSA Teórico-Práctico: Resolución guiada de ejercicio
	Viernes 29/09 2hs	Práctico Concurrencia en Java, el lenguaje FSP y LTSA
8	Martes 3/10 2hs	Teórico: Lógicas Modales - Lógica Temporal Lineal
	Jueves 5/10 4hs	Teórico: Lógica Temporal Lineal (Continuación) LTL y la herramienta LTSA. Práctico: Lógica Temporal Lineal
	Viernes 6/10 2hs	Práctico: Lógica Temporal Lineal (Continuación)
9	Martes 10/10 2hs	Teórico: Principios de Model Checking, Autómatas de Buchi
	Jueves 12/10 4hs	Teórico: Principios de Model Checking, Autómatas de Buchi y la Verificación de propiedades temporales. Herramientas Práctico: Lógica Temporal Lineal (Continuación)
	Viernes 13/10 2hs	**Feriado puente**
10	Martes 17/10 2h	Teórico: La lógica como lenguaje formal y su expresividad. SAT Solving:
	Jueves 19/10 4hs	Teórico: Análisis Automático de especificación en Alloy. Práctico: Verificación automática de propiedades temporales Introducción de una herramienta de Model Checking
	Viernes 20/10 2hs	Resolución guiada de ejercicio práctico Model Checking Práctico: Verificación automática de propiedades temporales
11	Martes 24/10 2hs	Teórico: Análisis Automático de especificación en Alloy.
	Jueves 26/10 4hs	Práctico de Alloy

	Viernes 27/10 2hs	**Introducción del segundo trabajo práctico obligatorio.** Repaso general para la resolución del Trabajo Práctico Obligatorio
12	Martes 31/10 2hs	Teórico: Modelos Dinámicos de Software (DynAlloy)
	Jueves 2/11 4hs	Práctico: Verificación automática de propiedades temporales Practico Alloy/DynAlloy
	Viernes 3/11 2hs	Práctico: Verificación automática de propiedades temporales Practico Alloy/DynAlloy
13	Martes 7/11 2hs	Práctico: Alloy
	Jueves 9/11 4hs	Teórico: Satisfactibilidad booleana: Algoritmos . Alloy para generar tests a partir de especificaciones (Continuación)
	Viernes 10/11 2hs	Práctico integrador con los temas vistos en la segunda parte
14	Martes 14/11 2hs	Evaluación oral individual (Defensa de trabajos prácticos obligatorios) En esta instancia de evaluación cada alumno es citado de manera individual en un horario pre-determinado
	Jueves 16/11 4hs	Evaluación oral individual (Defensa de trabajos prácticos obligatorios) En esta instancia de evaluación cada alumno es citado de manera individual en un horario pre-determinado
	Viernes 17/11 2hs	Evaluación oral individual (Defensa de trabajos prácticos obligatorios) En esta instancia de evaluación cada alumno es citado de manera individual en un horario pre-determinado

*Teóricos, teóricos-prácticos, trabajos de laboratorios, salidas a campo, seminarios, talleres, coloquios, instancias evaluativas, consultas grupales y/o individuales, otras.

7. BIBLIOGRAFÍA

7.1. Bibliografía obligatoria y de consulta

- Glenford J. Myers, *The Art of Software Testing*, John Wiley & Sons, Inc., 2012.D.
- Paul Ammann, Jeff Offutt, Introduction to Software Testing, Cambridge University Press, 2015.
- Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanshi, Prentice Hall, 2009
- Feedback-directed random test generation, Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Ernst, and Thomas Ball (ICSE 2007).

- Korat: A Tool for Generating Structurally Complex Test Inputs, Aleksandar Milicevic, Sasa Misailovic, Darko Marinov, Sarfraz Khurshid, (ICSE 2007).

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

- Jeff Magee & Jeff Kramer. Concurrency: State Models & Java Programs (2nd edition). Wiley. 2006.

- Daniel Jackson, Software Abstractions: Logic, Language and Analysis (Revised versión), MIT Press, 2012.

- Christel Baier and Joost-Pieter Katoen, Principles of Model Checking, The MIT Press, 2008.

- Willem Visser, Klaus Havelund, Guillaume P. Brat, Seungjoon Park, Flavio Lerda: Model Checking Programs. Autom. Softw. Eng. 10(2): 203-232 (2003)

- Corina S. Pasareanu and Willem Visser, Symbolic Execution and Model Checking for Testing, Haifa Verification Conference, 2017

- Logic in Computer Science modelling and reasoning about systems Michael Huth and Mark Ryan (2nd edition).

- Tutoriales y manuales de las herramientas utilizadas.

7.2. Otros: materiales audiovisuales, enlaces, otros.

8. DÍA Y HORARIOS DE CLASES

El dictado de la asignatura estará compuesto por cuatro horas semanales de clases teóricas y 4 horas semanales de clases prácticas. Las clases se desarrollarán los martes de 14 a 16, jueves de 14 a 18 y los viernes de 13 a 15.

9. DÍA Y HORARIO DE CLASES DE CONSULTAS

Las clases de consulta se darán semanalmente bajo demanda de los alumnos, en horario a convenir con los mismos.

10. REQUISITOS PARA OBTENER LA REGULARIDAD Y LA PROMOCIÓN

Evaluaciones Parciales: La materia contará con dos trabajos prácticos obligatorios, con una instancia de evaluación oral (defensa) ha realizarse la última semana de la materia. Cada trabajo cuenta con una instancia de recuperación. La aprobación de dichos trabajos (y su instancia de evaluación oral) es condición para la regularidad. El plazo para la resolución de cada uno de los trabajos prácticos será de dos semanas.

Evaluación Final: El examen final para alumnos regulares se llevará a cabo mediante evaluación escrita. 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: Para regularizar la materia se deberá: aprobar los dos trabajos prácticos de programación obligatorios y la instancia de defensa oral con nota no menor a 5.

CONDICIONES DE PROMOCIÓN: Para ser promovido en la materia se deberá aprobar los trabajos prácticos y la instancia de defensa oral con promedio no menor a 7.

11. CARACTERÍSTICAS, MODALIDAD Y CRITERIOS DE LAS INSTANCIAS EVALUATIVAS

El objetivo de los trabajos prácticos obligatorios es poder 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. Permiten también detectar y corregir problemas tanto en las actividades de cálculo y deducción en matemática y lógica como en la construcción de programas, y así afianzar los conocimientos adquiridos en materias anteriores, de las áreas de matemática discreta, lógica, programación e ingeniería de software. El objetivo de la instancia oral individual es evaluar si el alumno ha comprendido los fundamentos teóricos subyacentes a cada una de las técnicas aplicadas en los trabajos prácticos así como también evaluar que el alumno ha tenido una participación activa en la resolución grupal de los trabajos.

La materia podrá rendirse en condición de libre.

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