PROGRAMA DE ASIGNATURA	
ASIGNATURA: Ingeniería del Software II	**AÑO:** 2024
CARACTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 5° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
REGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTACIÓN Y OBJETIVOS

El curso introduce metodologías y técnicas avanzadas para la construcción de software confiable y seguro. Los temas tratados a lo largo del curso brindan el conocimiento fundamental y las herramientas para asegurar que el software que será parte de sistemas de alta complejidad, del cual pueden depender vidas humanas o respondan a misiones críticas, brinde un servicio correcto y efectivo.

Objetivos:
Al finalizar la materia los/as estudiantes estarán en condiciones de:
• comprender la problemática de los sistemas críticos (incluyendo sistemas concurrentes y de tiempo real) y los requerimientos fundamentales que estos deben satisfacer;
• elaborar modelos operacionales de estos tipos de modelos en lenguajes formales;
• expresar formalmente los requerimientos de estos sistemas complejos;
• seleccionar y manipular las herramientas y técnicas adecuadas para hacer los distintos tipos de análisis y verificación de modelos y especificaciones;
• comprender los fundamentos matemáticos y algorítmicos detrás de las distintas herramientas de análisis y verificación.

CONTENIDO

I. El problema de la correción del software

(1) Definición de sistemas críticos, (2) Limitaciones del testing y la simulación, (3) Discusiones sobre verificación.

II. Programación concurrente

(1) Definición de sistemas reactivos, (2) Interacción entre procesos, (3) Los problemas de la concurrencia, (4) Semántica de los programas concurrentes, (5) Interleaving y no determinismo, (6) Razonamiento sobre programas concurrentes, (7) La necesidad de abstraer para modelar, (8) El lenguaje de modelado FSP: sintaxis y semántica, (9) La herramienta LTSA.

III. Sincronización de procesos concurrentes

(1) Recursos compartidos: interferencia y exclusión mutua, (2) Detección de errores, (3) Monitores, sincronización condicional e invariantes del monitor, (4) Semáforos y su invariante, (5) Buffers acotados, (6) Bisimulación como equivalencia de procesos, (7) Comunicación mediante pasaje de mensajes, (8) Pasaje sincrónico de mensajes, (9) Recepción selectiva, (10) Pasaje asincrónico de mensajes, (11) Rendezvous. (12) Transacciones distribuidas.

IV. Propidades de los sistemas concurrentes

(1) Categorías de propiedades: alcanzabilidad, safety, liveness, y fairness, (2) Necesidad de la categorización de propiedades, (3) Propiedades como conjuntos de trazas, (4) Lenguajes ω-regulares, (5) Formalización de las propiedades de safety y liveness, (6) Otras propiedades, (7) Análisis automatizado de propiedades usando FSP: deadlock, safety y liveness.

V. Lógicas temporales

(1) Limitaciones de los métodos previos y de las lógicas usuales, (2) Lógicas modales, (3) Introducción a las lógicas temporales, (4) La lógica temporal lineal LTL, (5) Sintaxis y semántica,

(6) Operadores derivados y leyes, (7) Especificación de propiedades con LTL: Safety y Liveness, (8) Fairness: incondicional, débil y fuerte, (9) Otros tipos propiedades en LTL.

VI. Model checking

(1) El modelo de un sistema, (2) Autómatas de Büchi: definición y uso para presentar programas y propiedades, (3) Model Checking de propiedades LTL con enfoque en la teoría de autómatas, (4) Herramientas de model checking, (5) El model checking de propiedades descritas en LTL Spin, (6) Promela: modelado y análisis, (7) El model checker de propiedades descritas en CTL (computational tree logic) SMV, (8) El model checker de propiedades de tiempo Uppaal, (9) Otros model checkers.

VII. Especificaciones de sistemas

(1) Características de los lenguajes de especificación, (2) Las lógicas como lenguajes de especificación, (3) Lógica proposicional: Sintaxis, semántica y poder expresivo, (4) SAT solving en la lógica proposicional: ventajas y desventajas, (5) Lógica de primer orden: Sintaxis, semántica y poder expresivo, (6) SAT solving en la lógica de primer orden, (7) El álgebra relacional. Sintaxis, Semántica y Axiomas.

VIII. El lenguaje de especificación Alloy

(1) Sintaxis del lenguaje Alloy, (2) Características de Alloy, (3) Uso de Alloy para la resolución de problemas con restricciones (constraint solving), (4) Modelos de ejecuciones, (5) Uso de Alloy para verificar refinamientos, (6) Análisis de especificaciones en Alloy: Cotas, cuantificadores no acotados, axiomas de generación.

IX. Algoritmos para verificar satisfactibilidad en lógica proposicional

(1) Algoritmos simples: Tablas de verdad y argumentos semánticos, (2) Algoritmos avanzados, (3) Tablas de verdad revisadas, (4) Conversión a forma normal conjuntiva, (5) Regla de resolución clausal, (6) Propagación de restricciones booleanas, (7) El algoritmo de Davis, Putnam, Logemann & Loveland, (8) Cláusulas de Horn, (9) Linealidad de la resolución en la lógica de Horn, (10) La lógica de Horn como base de la programación lógica y los demostradores automáticos de teoremas.

X. Testing

(1) Definición del testing basado en modelos, (2) Testing con modelos formales, (3) El proceso de testing formal, (4) Conformidad correccion y exhaustividad, (5) La teoría de conformidad de testing basada en entradas y salidas (ioco: Input/Output Conformance Testing), (6) Extensión con tiempo y canales de la teoría ioco, (7) Definición de cubrimiento semántico.

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

[1] J. Magee y J. Kramer. Concurrency: State Models & Java Programs, 2nd edition. Wiley 2006.
[2] C. Baier and J.-P. Katoen. Principles of Model Checking. MIT Press, 2008.
[3] D. Jackson. Software Abstractions: Logic, Language, and Analysis (Revised Edition). MIT Press, 2011.
[4] A.R. Bradley y Z. Manna. The Calculus of Computation: Decision Procedures with Applications to Verification. Springer, 2007.

BIBLIOGRAFÍA COMPLEMENTARIA

[5] G. J. Holzmann. The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, 2003.
[6] B. Alpern y F. Scneider. Defining Liveness. Information Processing Letter 21:181-185. 1985
[7] B. Alpern y F. Scneider. Recognizing Safety and Liveness. Distributed Computing 2 (3): 117-126. 1987.
[8] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, P. Schnoebelen. Systems

and Software Verification Model-Checking Techniques and Tools. Springer, 2001.
[9] E.M. Clarke, O. Grumberg, D. Peled. Model Checking. MIT Press, 1999.
[10] P. Jalote. An Integrated Approach to Software Engineering, Third Edition. Springer. 2005.
[11] M. Müller-Olm, D. Schmidt, B. Steffen. Model Checking: A Tutorial Introduction. En A. Cortesi, G. Filé (Eds.), Procs. Of SAS'99, LNCS 1694, pp. 330-354. Springer 1999.
[12] J. Tretmans. A formal Approach to Conformance Testing. PhD Thesis. Univeristeit Twente, The Netherlands, 1992.

EVALUACIÓN

FORMAS DE EVALUACIÓN

La materia consta de dos evaluaciones parciales y la elaboración de un trabajo práctico con múltiples instancias de evaluación.

El trabajo práctico, determinante para la obtención de la regularidad, es un trabajo de investigación integral que comprende el desarrollo de un breve manuscrito asociado y una presentación oral, que se complementa con las actividades de la materia y orientan a la consolidación de la elaboración y diseminación de trabajos de investigación y desarrollo en el área.

Las dos evaluaciones parciales complementan al trabajo práctico en lo que respecta a la promoción de la materia.

REGULARIDAD

De acuerdo a lo establecido en la Ordenanza 4/2011, para obtener la regularidad, el/la estudiante deberá:
- aprobar al menos el 60 % de los Trabajos Prácticos o de Laboratorio.

PROMOCIÓN

De acuerdo a lo establecido en la Ordenanza 4/2011, para obtener la promoción, el/la estudiante deberá:
- aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete),
- aprobar todos los Trabajos Prácticos.