MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
PROPUESTA DE CURSO DE POSGRADO

1- DATOS GENERALES DE LA ACTIVIDAD CURRICULAR	
1.1 Título del Curso	Tópicos Avanzados en Pruebas de Software
1.2 Área temática[1]	Ingeniería de Software

2- COMPOSICION DEL EQUIPO DOCENTE	
2.1 Responsable a cargo de la actividad curricular	Dr. Andrés Pablo Flores
2.2 Docentes	

3- CARGA HORARIA					
Carga horaria teórica	30				
Carga horaria práctica	30				
Carga horaria total	60				
Distribución horaria semanal	Lu	Ma	Mie	Jue	Vie
Fecha de inicio sugerida					

4- BREVE RESUMEN DE CONTENIDOS (hasta 400 palabras)

La naturaleza evolutiva del hardware y software propicia cambios en la forma en la que se desarrollan actualmente las aplicaciones software, sumado a ello se presenta la reducción de costos en hardware cada vez más potente, que estimula a un mejor aprovechamiento por parte del software que lo controla. Esto genera la evolución a nuevos paradigmas mejor preparados para afrontar una necesidad de eficacia en rendimiento y costo para los procesos de construcción de aplicaciones software. Tales paradigmas se basan en extensiones de las estrategias existentes o bien desarrollando nuevos enfoques, donde los aspectos de estandarización juegan un rol de suma importancia para solucionar requerimientos de integración de aplicaciones distribuidas a distintos niveles: locales a una organización incluso pequeña, o bien escalando a multinacionales cuyos requerimientos adoptan la forma de componentes software o servicios que explotan la web como infraestructura de comunicación.

Ante situaciones como las descriptas, surge la necesidad de asegurar una confiabilidad en el software que se desarrolla, lo cual se relaciona directamente con correctitud del software y garantía de calidad, y sumado a ello se deben considerar los costos de corregir y actualizar un software de escritorio de la propia organización (denominado in-house), dado que tales costos se incrementan incluso superando la proporción de distribución que se modela para las aplicaciones. Por lo tanto ante cada nuevo paradigma que se instala como solución al desarrollo de sistemas cada vez más complejos, las técnicas para asegurar la correctitud del software y garantía de calidad deben adaptarse y evolucionar en consecuencia.

Así el Testing del Software se ajusta ante el advenimiento de cada nuevo paradigma en función de las estrategias que actualmente resultan exitosas y desarrollando nuevos enfoques que resuelvan los aspectos que aún no han sido adecuadamente cubiertos. Por otro lado ante una gran diversidad de estrategias de Verificación y Validación de Software, resulta de suma importancia no solamente inspeccionar detalladamente los fundamentos, estrategias, heurísticas, y aspectos de calidad sobre los que influye, sino también adquirir la habilidad para discernir el uso adecuado de cada una de ellas de acuerdo a la planificación que se realice para el testing y el contexto en el cual se aplica.

[1] Corresponde a uno de los siguientes tópicos: Algoritmos y Lenguajes; Teoría de la Computación; Ingeniería de Software, Bases de Datos y Sistemas de Información; Arquitecturas, Sistemas Operativos y Redes.

5- CONOCIMIENTOS PREVIOS REQUERIDOS

6- OBJETIVOS

La asignatura tendrá por objetivo el comprender y aplicar - en lo posible - los conceptos y técnicas adecuados para comprobar que el software cumple con sus especificaciones y satisface las expectativas de los usuarios.
Para esto se cubrirán las técnicas de testing y análisis clasificadas como Caja Blanca y Caja Negra, partiendo de las técnicas para software tradicional (incluyendo Orientación a Objetos), continuando con sistemas en la Web y móviles, y luego para los paradigmas basado en componentes Software, orientado a Servicios y basado en Cloud.

7- CONTENIDOS (organizados en unidades, ejes, módulos, otros)

Unidad 1:
Verificación de Software: objetivos y requerimientos – Testing de unidad: estructural y funcional – Testing basado en Estado – Testing basado en Defectos – Metaheuristicas: algorítmos genéticos y búsqueda Tabú – Testing de Integración – Testing de Sistema – Testing de Regresión – Testing de Aceptación – Automatización de Testing.

Unidad II
Fundamentos de Testing para Aplicaciones Web – Modelo Cliente-Servidor – Testing de Unidad – Cubrimientos Funcionales: Partición por Nivel, Análisis de Valores Límite, Test basado en Estado, Usabilidad – Cubrimientos Estructurales: Modelo Navegacional, Test de Scripts – Testing de Sistema: Performance, Configuración, Seguridad – Testing de Aplicaciones Móviles – Automatización de Testing Web.

Unidad III
Fundamentos de Software basado en Componentes – Testing para Componentes: Rol del Usuario y Proveedor – Cubrimientos para Testing de Unidad e Integración para Componentes – Testing de Regresión para Componentes – Metadatos para Testing de Componentes – Estrategias de Self-Testing – Automatización de Testing para Componentes.

Unidad IV
Fundamentos de Aplicaciones orientadas a Servicios – Testing para Servicios Web – Cubrimientos para Testing de Unidad e Integración para Servicios – Testing en Cloud: SaaS, PaaS, IaaS, TaaS, Elasticidad – Automatización de Testing para Servicios.

Unidad V
Gestión de Testing – Planificación – Estándares de Testing – Documentación – Calidad de Producto Software – Métricas para Facilidad de Testing – Aseguramiento de la Calidad – Gestión de la Configuración.

8- PROPUESTA DIDÁCTICA (metodología de trabajo de clases teóricas y prácticas)

Las actividades se desarrollarán según el cronograma estándar establecido para el curso. Las horas asignadas a cada encuentro se dedicarán a:
- Introducción y discusión de contenidos teórico-prácticos
- Desarrollo de ejercicios de acuerdo a trabajos prácticos específicos según cada unidad del temario, que permiten cubrir los aspectos de prueba de software así como generar discusión y formación de criterios.
- Talleres en los que se podrá evaluar y/o usar técnicas específicas (ej. cubrimientos para componentes, estándares aseguramiento de calidad de verificación y validación, etc.). En estos talleres se utilizarán herramientas para prueba de software, tales como JUnit para programas Java.
- Actividades grupales de exposición de temas en las que se profundizarán aspectos teórico-prácticos (ej., casos de estudio desarrollados grupalmente).

9- MODALIDAD DE EVALUACIÓN Y CONDICIONES DE ACREDITACIÓN[2]

Mediante elaboración de trabajo individual o en grupos de no más de 3 personas que profundice aspectos teóricos y/o prácticos tratados en el curso.
Opcionalmente, podrá instrumentarse un examen final escrito sobre contenidos teóricos y/o prácticos

10- BIBLIOGRAFÍA DE LECTURA OBLIGATORIA CORRESPONDIENTE A CADA UNIDAD Y GENERAL

[2] Son condiciones mínimas para la aprobación de todos los cursos: cumplir con un mínimo del 80% de asistencia a las clases, realizar las tareas y aprobar las evaluaciones que se hayan propuesto en el programa, con una calificación no menor a 7 (puntos). Los trabajos de evaluación pautados y la calificación de los alumnos deberán realizarse dentro de los 60 días posteriores a la finalización del curso.

Binder, Robert V. (2000). Testing Object Oriented Systems - Models, Patterns and Tools. Addison-Wesley
Myers, G. 2004. The art of software testing, 2nd edition. John Wiley & Sons Inc.
Astels, D. 2003. Test-Driven Development: A Practical Guide. Pearson Education Inc.
Nguyen, H., Johnson, B., Hackett, M. 2003. Testing Applications on the Web: Test Planning for Mobile and Internet-Based Systems, Second Edition. Wiley Publishing, Inc.
Stottlemyer, Diane. 2001. Automated Web Testing Toolkit Expert Methods for Testing and Managing Web Applications. John Wiley & Sons, Inc.
Subraya, B. M. 2006. Integrated Approach to Web Performance Testing: A Practitioner's Guide. IRM Press.
Harty, Julian. 2010. A Practical Guide to Testing Wireless Smartphone Applications (Synthesis Lectures on Mobile and Pervasive Computing) 1st Edition. Morgan & Claypool Publishers.
Beydeda, S., Gruhn, V. 2005. Testing Commercial-off-the-Shelf Components and Systems. Springer-Verlag.
Hans-Gerhard Gross. 2005. Component-Based Software Testing with UML. Springer.
Gao, J., Tsao, J., Wu, Y. 2003. Testing and Quality Assurance for Component-Based Software. Artech House Inc.
Thomas, Erl. 2005. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR.
Baresi, L., Di Nitto, E. 2007. Test and Analysis of Web Services. Springer-Verlag.
Kankanamge, Charitha. 2012. Web Services Testing with soapUI. Packt Publishing.
Blokland, K., Mengerink, J., Pol, M. 2013. Testing Cloud Services – How to Test SaaS, PaaS & IaaS. Rocky Nook Inc.
Tilley, S., Parveen, T. 2012. Software Testing in the Cloud – Migration and Execution. Springer.
ISTQB. Glossary of terms used in Software Testing. Version 1.0. 'Glossary Working Party', International Software Testing Qualification Board. 2004.
IEEE Std 829-1998. IEEE Standard for Software Test Documentation. 1998.
SEI. The Capability Maturity Model – CMU/SEI-93-TR-24:
http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html
SEI. Key Practices of CMM – CMU/SEI-93-TR-
25:http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.025.html
ISO/IEC. Software Engineering — Guide to the Software Engineering Body of Knowledge (SWEBOK).ISO/IEC TR-19759, 2005.
BCS SIGIST. Standard for Software Component Testing. Working Draft 3.4. British Computer Society Specialist Interest Group in Software Testing (BCS SIGIST). 2001.
ISO/IEC 25051. Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing. 2006.
Delamaro M, Maldonado J, Mathur A. Interface Mutation: An Approach for Integration Testing. IEEE Transactions on Software Engineering; 27(3):228–247. 2001.
Orso A, Do H, Rothermel G, Harrold MJ, Rosenblum D. Using Component Metadata to Regression Test Component-based Software. Software Testing, Verification and Reliability May 2007; 17:61–94. http://www.interscience.wiley.com.
Ghosh S, Mathur AP. Interface Mutation. Software Testing, Verification and Reliability (11):227–247. 2001. http://www.interscience.wiley.com
Jaffar-Ur Rehman M, Jabeen F, Bertolino A, Polini A. Testing Software Components for Integration: a Survey of Issues and Techniques. Software Testing, Verification and Reliability 17(2):95–133. 2007. http://www.interscience.wiley.com .
Glover, F. Tabu search: A tutorial. Interfaces 20(4), 74–94. 1990.
Díaz, E., J. Tuya, et al. Pruebas automáticas de cobertura de software mediante una herramienta basada en Búsqueda Tabú. VIII Jornadas de Ingeniería del Software y Bases de Datos, Alicante, Spain. 2003.
Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer. 1996.
Jans, R., Degraeve, Z. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. European Journal of Operational Research 177(3), 1855–1875. 2007.
Pargas, R. P., M. J. Harrold, et al. Test-Data Generation Using Genetic Algorithms. Software Testing, Verification and Reliability (9): 263-282. 1999. http://www.interscience.wiley.com

11- INFRAESTRUCTURA E INSUMOS REQUERIDOS[3]

[3] Deberá constar aquí si la realización del curso requiere contar con instalaciones especiales (laboratorio, sala de informática, equipamiento audiovisual, etc). Explicitar si se estima que el curso debe tener un

12 – OTRA INFORMACIÓN RELEVANTE

número máximo determinado de asistentes para poder ser dictado.