ESPECIALIZACIÓN EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN

MAESTRÍA EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN

1. Nombre de la actividad curricular: Pruebas de Software

2. Año Académico: 2020

3. Docente: Nazareno Aguirre

4. Fundamentación

Entre los aspectos que hacen a la calidad de los productos de software, la calidad funcional, o más precisamente el grado con que el software cumple con los requisitos funcionales establecidos para el mismo, es de fundamental importancia. Uno de los mayores desafíos de la ingeniería de software es, precisamente, brindar herramientas metodológicas y de análisis efectivas, que contribuyan a brindar garantías de calidad funcional. Existen numerosas técnicas, manuales, semi-automáticas y automáticas, que apuntan a garantizar la calidad funcional del software mediante variedades de análisis dinámicos y estáticos. Sin embargo, principalmente debido a razones de escalabilidad, el mecanismo más empleado en la práctica para dar garantías de correcto funcionamiento de software es el *testing*. El *testing*, que esencialmente consiste en ejecutar el software bajo análisis en un número de escenarios de ejecución particulares, es una actividad sumamente costosa en tiempo, y que demanda ingenio y dedicación por parte del o los ingenieros a cargo de la misma. En particular, cuando se necesita garantizar buenos niveles de calidad a través de testing, éste debe llevarse adelante de manera sistemática y metódica, y cumplir con ciertos criterios de adecuidad. Más aún, con la creciente aplicación de técnicas de testing a diferentes niveles de abstracción y en diferentes etapas del proceso de desarrollo, resulta fundamental, desde una perspectiva práctica, comprender los fundamentos de testing, las clasificaciones y diferentes formas de esta técnica en el desarrollo de software, y técnicas y metodologías fundamentales del área, en la actualidad.

Luego, y debido a la importancia que el testing tiene como técnica general para el análisis y la provisión de garantías de calidad de software, conocer los principios fundamentales del testing en todas sus formas, es crucial hoy en día para la formación y actualización de profesionales de Informática. Además, debido al alto costo que las actividades de validación y verificación demandan (en las cuales se aplican estas técnicas de testing), conocer el amplio espectro de herramientas de automatización de testing es esencial como parte del cuerpo de conocimiento de los mencionados profesionales.

5. Objetivos

El objetivo general del curso es conseguir que los alumnos comprendan los fundamentos de validación y verificación de software en general, del testing como mecanismo específico para estas tareas, y las técnicas y metodologías más importantes vinculadas al testing de software.

Los objetivos específicos de este curso se resumen en los siguientes:

- Que los cursantes consigan una formación sólida sobre los elementos que hacen a la calidad de software, con énfasis en calidad funcional.
- Que los cursantes comprendan los fundamentos del testing, incluyendo los conceptos fundamentales del área, las clasificaciones usuales y diferentes formas de testing en la práctica, y técnicas y metodologías para llevar adelante actividades de validación y verificación usando este enfoque.

- Que los alumnos identifiquen los elementos centrales para la aplicación de testing en la práctica, y se familiaricen tanto con las metodologías para su empleo, como con herramientas de testing automático.

6. Contenidos

1. 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*?

2. El testing como actividad de validación y verificación. Conceptos fundamentales de testing, sus objetivos y principios. Etapas del testing. Elementos constitutivos de las pruebas de software, y su importancia. El *testing* como especificación. Niveles de testing de acuerdo a la actividad del software.

3. Pruebas unitarias: conceptos de prueba y granularidad. Automatización de testing usando frameworks xUnit. Estructuración y composición de tests de unidad. Construcción de suites de tests. Generalización de tests. Tests parametrizados y teorías.

4. Criterios de adecuidad y técnicas de diseño de casos de prueba. Testing de caja negra y caja blanca. Testing de mutación. Criterios de cobertura. Principales criterios de caja negra (particionado del espacio de entradas, testing exhaustivo acotado, criterio de consultas booleanas) y caja blanca (cobertura de sentencias, cobertura de ramas, cobertura de ciclos, cobertura de caminos).

5. Independencia en unidades de test. El concepto de doble (objeto simulado) y su necesidad. Razones para la simulación de objetos: no determinismo, eficiencia, desarrollo independiente. Simulación de entradas (resp. salidas) directas e indirectas. Tipos de dobles: dummies, stubs, mocks, etc. Herramientas de apoyo.

6. La necesidad de las especificaciones de programas. El problema del oráculo en testing. Soluciones a la ausencia de especificaciones. Testing diferencial y testing de regresión.

7. Testing en el proceso de desarrollo de software. Testing desacoplado del desarrollo de software y sus limitaciones. Testing como etapa de desarrollo. Prácticas de incorporación de testing a la codificación. Test-first y Test-driven development.

8. Tests como documentación de requisitos. Desarrollo guiado por comportamiento (behavior-driven development) y su conexión con testing de aceptación. Descripción de comportamientos y escenarios legibles. Mapping de comportamientos en tests. Herramientas.

9. Gestión del testing. El plan de testing y su manejo. Trazabilidad de defectos y tests. Clasificación de defectos. Priorización en testing y depuración. Reportes de testing: reportes diarios de ejecución, reportes de integración y aceptación. Reportes de defectos. Gestión de testing e integración continua. Herramientas.

10. Otras formas de testing. Validación y testing de sistema. Testing en sistemas con características no deterministas. Testing de desempeño y carga. Oráculos vinculados a desempeño. Análisis de tiempo de respuesta y robustez. Stress testing.

7. Metodología de Enseñanza y Formación práctica

Se pondrá especial énfasis en la aplicación de las principales técnicas de testing de software en problemas concretos. Las clases serán por lo tanto teórico-prácticas. Se fomentará el uso de herramientas de software para la asistencia en el proceso de validación y verificación mediante testing (las técnicas elegidas cuentan, todas ellas, con herramientas de soporte).

Durante el desarrollo del curso se brindarán trabajos prácticos, cuya resolución y aprobación será requisito para la aprobación del curso. Los mismos tienen por objetivo lograr que los alumnos puedan consolidar la teoría aprendida y aplicarla en la resolución, mediante la asistencia de programas de computadora, de problemas de validación y verificación de software a través de testing. Se buscará que los problemas a resolver en los trabajos prácticos hagan evidentes las bondades y limitaciones de las distintas técnicas de presentadas. Se intentará utilizar ejemplos y problemas interesantes, en los cuales las fallas sean difíciles de reconocer, intentando estimular al alumnado.

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.

Además, se proveerá una serie de ejercicios adicionales de menor dificultad y longitud que los de los trabajos prácticos, que acompañarán cada una de las clases teórico-prácticas.

El curso tendrá un único examen, con una recuperación. El examen será teórico-práctico, y abarcará todos los contenidos de la asignatura.

8. Carga horaria total

Carga horaria teórica	Carga horaria práctica	Carga horaria total
36 horas	24 horas	60 horas

9. Modalidad de Evaluación

La evaluación del desempeño de los alumnos en el curso se realizará mediante trabajos prácticos, y un único examen teórico-práctico, que abarcará la totalidad de los contenidos del curso. El examen contará con una recuperación.

En lo que respecta a los trabajos prácticos, los alumnos podrán consultar durante su desarrollo al docente, en relación a cualquier aspecto vinculado a su resolución. Los alumnos podrán conformar equipos de trabajo, para la resolución de trabajos prácticos. El examen, en cambio, será de resolución individual.

10. Requisitos de aprobación y promoción

La calificación se expresará en escala numérica de cero (0) a diez (10) sin decimales. Para la promoción se requerirá la norma mínima de siete (7). (Extraído de la Ordenanza N° 1313)

11. Infraestructura y equipamiento

La infraestructura y ámbitos a utilizar en el dictado son los siguientes:

1. Campus virtual: el material bibliográfico del curso, las presentaciones y los enunciados de las ejercitaciones y trabajos prácticos serán distribuidos a través de una plataforma de campus virtual. Adicionalmente, se utilizarán otras herramientas para facilitar la colaboración y comunicación (canales de comunicación Slack, repositorios de código (Github, Bitbucket), repositorios de imágenes ejecutables (docker).
2. Aulas: las clases teóricas se desarrollan en un aula con capacidad suficiente para los alumnos del curso, equipo de proyección y acceso a internet mediante conexión wifi. Todo el equipamiento mencionado será empleado en el dictado de las clases teóricas.
3. Laboratorio: adicionalmente, y debido a las características del curso, se requerirá disponibilidad de equipos de computación para realizar la ejercitación semanal, y los trabajos prácticos requeridos para la aprobación del curso.

12. Bibliografía

A. Leitner, I. Ciupa, B. Meyer, M. Howard, Reconciling Manual and Automated Testing: the AutoTest

Experience, in Proc. of HICSS 2007, IEEE CS, 2007.

B. Dutertre, L. de Moura, The YICES SMT Solver, 2006.

Boyapati C., Khurshid S., Marinov D., Korat: automated testing based on Java predicates, in Proc. of ISSTA 2002, ACM Press, 2002.

C. Barrett, R. Sebastiani, S. Seshia, C. Tinelli, Satisfiability Modulo Theories, Handbook of Satisfiability, IOS Press, 2009.

C. Kaner, J. Bach, B. Pettichord, Lessons Learned in Software Testing, Wiley, 2001.

C. Pacheco, S. Lahiri, M. Ernst, T. Ball, Feedback-Directed Random Test Generation, in Proc. of ICSE 2007, IEEE CS, 2007.

C. Pacheco, Shuvendu K. Lahiri, M. D. Ernst, and T. Ball. Feedback-directed random test generation, en Proceedings of the 29th International Conference on Software Engineering (ICSE), 2007.

D. Jackson, M. Vaziri, Finding bugs with a constraint solver, in Proc. of ISSTA 2000, ACM, Press, 2000.

D. Jackson, Software Abstractions, MIT Press, 2006.

D. Shao, S. Khurshid, D. Perry, Whispec: white-box testing of libraries using declarative specifications, in Proc. of LCSD 2007, ACM Press, 2007.

E. Weyuker, B. Jeng, Analyzing Partition Testing Strategies, Trans. Software Eng. 17(7), IEEE Press, 1991.

G. Dennis, F. Chang, D. Jackson, Modular Verification of Code with SAT, in Proc. of ISSTA 2006, ACM Press, 2006.

G. Fraser, A. Arcuri, EvoSuite: automatic test suite generation for object-oriented software, in Proc. of ESEC/FSE 2011, ACM Press, 2011.

H. Czemerinski, V. Braberman, S. Uchitel, "Behaviour Abstraction Coverage as Black-Box Adequacy Criteria", Proceedings of IEEE International Conference on Software Testing, Verification and Validation (ICST), 2013.

H. Zhu, P. Hall, J. May, Software Unit Test Coverage and Adequacy, Computing Surveys 29(4), ACM Press, 1997.

J. Dolby, M. Vaziri, F. Tip, Finding Bugs Efficiently with a SAT Solver, in Proc. of ESEC/FSE 2007, ACM Press, 2007.

J. Siddiqui, S. Khurshid, Pkorat: Parallel Generation of Structurally Complex Test Inputs, Proceedings of IEEE International Conference on Software Testing, Verification and Validation (ICST), 2009.

J.Belt, Robby, X.Deng, Sireum/TopiLDP: A Lightweight Semi-Decision Procedure for Optimizing Symbolic Execution-based Analyses, in Proc. of ESEC/FSE 2009, ACM Press, 2009.

J.Galeotti,N.Rosner, C.López Pombo, M.Frias, Analysis of invariants for efficient bounded verification, in Proc. of ISSTA 2010, ACM Press, 2010.

L. de Moura, N. Bjorner, Z3: An Efficient SMT Solver, in Proc. Of TACAS 2008, LNCS 4963, Springer, 2008.

L. Liu , B. Meyer, B. Schoeller, Using contracts and boolean queries to improve the quality of automatic test generation, in Proc. of TAP 2007, LNCS 4454, Springer, 2007.

M. Gligoric, T. Gvero, V. Jagannath, S. Khurshid, V. Kuncak,D. Marinov, Test Generation through Programming in UDITA, in Proc. of ICSE 2010, ACM Press, 2010.

M. Wynne, A. Hellesoy, The Cucumber Book, Behaviour-Driven Development for Testers and Developers, Pragmatic Bookshelf, 2012.

N. Tillmann, J. de Halleux, Pex: White Box Test Generation for .NET, in Proc. of TAP 2008, LNCS 4966, Springer, 2008.

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

P. Chalin, J. Kiniry, G. Leavens, E. Poll, Beyond Assertions: Advanced Specification and Verification with JML and ESC/Java2, in Proc. of FMCO 2005, LNCS 4111, Springer, 2006.

R. Martin, The Clean Coder, A Code of Conduct for Professional Programmers, Prentice-Hall, 2011.

R. Sharma, M. Gligoric, A. Arcuri, G. Fraser, D. Marinov, Testing container classes: random or systematic?, in Proc. of FASE 2011, LNCS 6603, Springer, 2011.

S. A. Khalek. Systematic Testing Using Test Summaries: Effective and Efficient Testing of Relational Applications. Ph.D. Thesis. University of Texas at Austin, 2011.

S. Khurshid, D. Marinov, TestEra: Specification-Based Testing of Java Programs Using SAT, Autom. Soft. Eng. 11(4), Kluwer Academic, 2004.

W. Visser, C. Pasareanu, R. Pelánek, Test Input Generation for Java Containers using StateMatching, in Proc. of ISSTA 2006, ACM Press, 2006.

X. Deng, Robby, J. Hatcliff, Kiasan/KUnit: Automatic Test Case Generation and Analysis Feedback for Open Object-oriented Systems, in Proc. of MUTATION 2007,IEEE CS, 2007.