Formulario de Aprobación Curso de Posgrado 2010

Asignatura: Técnicas de Testing Combinatorio y de Mutación

Profesor de la asignatura [1]: Macario Polo Usaola; Doctor por la Universidad de Castilla-La Mancha; Profesor Titular de Universidad, área de Lenguajes y Sistemas Informáticos, destino en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha (campus de Ciudad Real), España.

Profesor Responsable Local [1]: Licenciada Mónica Wodzislawski, Grado 3, InCo, UdelaR

Otros docentes de la Facultad:
Beatriz Pérez, Magister en Informática, Grado 3, InCo, UdelaR

Docentes fuera de Facultad:

Instituto ó Unidad: Instituto de Computación

Departamento o Área: Ingeniería de Software

[1] Agregar CV si el curso se dicta por primera vez.
(Si el profesor de la asignatura no es docente de la Facultad se deberá designar un responsable local)

Fecha de inicio y finalización:
Horario y Salón:

Horas Presenciales: 20

Nº de Créditos: 2

Público objetivo y Cupos: Titulados universitarios en informática o áreas afines.

Objetivos:
Los asistentes a este curso aprenderán y pondrán en práctica de técnicas de testing de software. Se describen algunos conceptos fundamentales al comienzo, con el objetivo de recordar a los asistentes estas ideas básicas y nivelar sus conocimientos. A continuación se describen en profundidad técnicas específicas de testing, que se pondrán en práctica con diversas herramientas. En particular, se describen algoritmos de generación automática de casos de prueba, que los asistentes pondrán en funcionamiento utilizando una herramienta web desarrollada por Beatriz Pérez Lamancha y Macario Polo Usaola (http://161.67.140.42/CombTestWeb/).

Se continuará con una descripción de las pruebas utilizando mutación, técnica que se ha utilizado intensamente durante 30 años en investigación, pero que ahora comienza, cada vez más, a implantarse en la industria. Se utilizará para esto la herramienta Muclipse y la herramienta Bacterio, desarrollada por Pedro Reales Mateo y Macario Polo Usaola (http://alarcos.esi.uclm.es/testing/).

En definitiva, el objetivo principal del curso es la capacitación de los asistentes en técnicas avanzadas de testing de software.

Es de destacar que gran parte de los contenidos de este curso los imparte el ponente en el curso de doctorado Pruebas y Seguridad de Sistemas de Información, perteneciente al programa de doctorado Técnicas Informáticas Avanzadas, del Departamento de Tecnologías y Sistemas de Información de la Universidad de Castilla-La Mancha, que cuenta con mención de calidad de la Agencia Nacional de Evaluación de la Calidad.

Conocimientos previos exigidos: Los propios de la titulación universitaria aceptada.

Conocimientos previos recomendados: Ingeniería del software. Participación en proyectos de desarrollo o mantenimiento de software.

Metodología de enseñanza:
Deseablemente, las clases se impartirán en un laboratorio de computadores, con disponibilidad de un PC por cada alumno o dos alumnos. Cada PC tendrá instalado el entorno de desarrollo Eclipse 3.0 (recomiendo la versión Europa que tengo en la siguiente URL, pues es bastante estable y dispone de muchos plugins: http://www.inf-cr.uclm.es/www/mpolo/asig/0708/europa.zip), así como una máquina virtual Java 1.6.
El curso combina teoría, en forma de minicharlas o miniclases magistrales, con prácticas, ejemplos y casos de estudio utilizando herramientas.
La primera parte del curso (Técnicas de testing combinatorio) tiene una duración aproximada de 8-9 horas presenciales, completándose las 11-12 horas restantes con la segunda parte (Pruebas utilizando mutación). Se estima conveniente la resolución, por parte de los alumnos, de diversos supuestos prácticos sobre un proyecto de testing, fuera del horario presencial, con una carga horaria aproximada de 10 horas.

Forma de evaluación:
Examen tipo MO con un valor de un 20% y la realización de un trabajo con un valor de 80%. Se evalúa de 0 a 10 puntos.

---

0) **Temario:** Introducción.
1) Criterios de cobertura de artefactos software.
    a. Concepto
    b. Utilidad
    c. Criterios de cobertura para código fuente
        i. Sentencias
        ii. Condiciones
        iii. Decisiones
        iv. Condiciones y decisiones
        v. Modified Condition/Decision
    d. Criterios de cobertura para máquinas de estado
        i. Estados
        ii. Transiciones
        iii. Pares de transiciones
        iv. Caminos importantes
2) Valores de prueba.
    a. Propuesta de valores de prueba (clases de equivalencia, valores límite, conjetura de errores...)
    b. Each use (cada uso)
    c. Pairwise
    d. N-wise
3) Estrategias de combinación para casos de prueba.
    a. Estructura de un caso de prueba.
    b. All combinations

c. Un pairwise básico
d. Antirandom
e. Comb
f. AETG
g. Weighted AETG
h. Customized Pairwise
i. Algoritmos evolutivos
j. Prácticas de algoritmos con el sistema CTWeb.
    i. Carga de datos y ejecución de algoritmos
    ii. Modo "verbose"
    iii. Ejemplos de generación de casos de prueba, carga de datos, generación de configuraciones, etc.

4) Técnicas para generación de casos de prueba.
    a. A partir de máquinas de estado
    b. A partir de expresiones regulares
    c. A partir de requisitos descritos en forma de tablas de decisión
5) Pruebas utilizando mutación.
    a. Introducción.
    b. Conceptos importantes: Mutante, mutante vivo, mutante muerto, mutante equivalente, *mutation score*, matrices de muertos.
    c. Operadores de mutación.
    d. Fases del proceso y costes:
        i. Generación
        ii. Ejecución
        iii. Análisis de resultados
    e. Tipos de mutación: strong, weak, functional qualification, flexible weak mutation
    f. Algunas herramientas. Uso de Muclipse
    g. La herramienta Bacterio.
        i. Configuración. Tipos de mutación.
        ii. Generación de mutantes: operadores y algoritmos de generación.
        iii. Modos de ejecución: *Selected versions* y *Test case file (full tests, versions oriented, test case oriented).*
        iv. Análisis de resultados
        v. Testing exploratorio (beta)
6) Casos de prueba redundantes.
7) Recapitulación.

Bibliografía:
Nota: la bibliografía en forma de artículos de revistas está disponible para su reparto entre los asistentes, de acuerdo con las condiciones de copyright de sus respectivas editoriales.

- Grindal M, Offutt AJ and Andler SF. Combination testing strategies: a survey. *Software Testing, Verification and Reliability*, 15(167-199, 2005.
- Hamlet R. Testing programs with the help of a compiler. *IEEE Transactions on Software Engineering*, 3(4), 279-290, 1977.
- Myers GJ. *The art of software testing*, 2nd edition. John wiley & Sons, 2004.
- Polo M, Piattini M and García-Rodríguez I. Decreasing the cost of mutation testing with second-order mutants. *Software Testing, Verification and Reliability*, 19(2), 111-131, 2008.
- Polo M and Reales P. (2010). Mutation testing cost reduction techniques: a survey. *IEEE Software*, 27(3), 80-86.