Métodos formales de validación de sistemas

BREVE DESCRIPTOR:

La asignatura muestra las distintas fases que se suceden en el desarrollo formal de sistemas complejos: especificación del sistema, comprobación de que la especificación cumple los requisitos y validación, mediante *testing*, de que el sistema desarrollado es correcto con respecto a la especificación.

REQUISITOS:
Ingeniería del Software: nivel alto;

OBJETIVOS:
El principal objetivo de la asignatura es que los alumnos conozcan las distintas fases que deben acometerse durante el desarrollo formal de un sistema complejo. Por ello, los alumnos aprenderán:

- distintos formalismos para especificar formalmente sistemas complejos, con un énfasis especial en formalismos gráficos y
- los principios básicos de la comprobación de modelos y del testing formal de sistemas así como las principales herramientas para llevarlos a cabo.

CONTENIDOS TEMÁTICOS:
Tema 1: Introducción al desarrollo de sistemas software.
Tema 2: Especificación formal de sistemas software.
Tema 3: Comprobación de modelos.
Tema 4: Testing formal de sistemas.

BIBLIOGRAFÍA:

- C. Baier and J.-P. Katoen; *Principles of Model Checking*; MIT Press, 2008;
- E. Clarke, O. Grumberg and D. Peled; *Model Checking*; MIT Press, 2000;
- M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. Talcott; *All About Maude - A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic*; Springer, 2007;
- R. M. Hierons, J. P. Bowen and M. Harman; *Formal Methods and Testing*; Springer, 2008;
- B. O'Sullivan, J. Goerzen and D. B. Stewart; *Real World Haskell*; O'Reilly Media, Inc., 2008;
- M. Utting and B. Legeard; *Practical Model-Based Testing: A Tools Approach*; Morgan-Kaufmann, 2007;

OTRA INFORMACIÓN RELEVANTE
EVALUACIÓN:

Evaluación continua. Los alumnos asistirán al menos al 90% de las clases, prepararán las lecturas que se propongan durante el curso, y que se discutirán en clase, y presentarán, organizados en grupos de tres alumnos, un trabajo. Siendo P1 el porcentaje, en tanto por ciento, de las clases a las que un alumno ha asistido, P2 la nota obtenida por participación en clase y T la valoración del trabajo y de su presentación, (tanto P2 como T toman valores entre 0 y 10), la nota final vendrá dada por 0,015* P1 + 0,2 * P2 + 0,65 * T.

ACTIVIDADES DOCENTES
Clases teóricas:
Presenciales

ACTIVIDADES DOCENTES
Clases prácticas:
Presenciales, parte de ellas a realizar en laboratorio.

Discusión adicional

A continuación se incluyen comentarios, por parte de los miembros de la comisión encargada de redactar esta ficha, sobre los contenidos de la asignatura. Estos comentarios no han sido incluidos en la ficha docente pero pueden ser tenidos en cuenta a la hora de planificar los contenidos concretos de la asignatura.

Manuel:

He estado buscando definiciones y una muy concisa y que resume los puntos esenciales es:

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It is normally the responsibility of software testers as part of the software development lifecycle.

http://en.wikipedia.org/wiki/Verification_and_validation_%28software%29

Utilizar algo similar nos permite no centrarnos en la validación (que se realiza al final del proceso de desarrollo) sino empezar en la especificación formal de los requisitos y tratar la *verificación*[1] de sistemas como parte de la asignatura.

Cubrir en esta asignatura las tres etapas "formales" en el desarrollo de software (especificación, verificación, validación) nos permite realizar un temario donde el profesor puede poner más peso en una de las partes (si no apareciera en el temario, en teoría, no se podría tratar) teniendo en cuenta su *expertise* y los intereses de los alumnos.

Adrián:

- Una parte interesante del model checking, al menos para mi, son las lógicas modales. Creo que sin meternos en unos detalles muy profundos se podría explicar lo que es una lógica modal y un par de ejemplos interesantes (las temporales y las de conocimiento, por ejemplo), y ponerlo como objetivo/tema.

- No creo que sea necesario decir que haremos un énfasis especial en formalismos gráficos. Aunque entiendo que a los alumnos les puede parecer más amigable ver "dibujos" el generador de casos de Haskell, Quickcheck, es muy interesante, podemos aprovechar que han visto Haskell durante la carrera, y además hay versiones para lenguajes como Java (aunque creo que es de pago) o el model checker es bastante rápido y razonablemente sencillo de usar. Resumiendo, creo que podemos ser generales en la ficha (lenguajes declarativos vs formalismos gráficos) y luego, como Manuel dice, que cada se centre en lo que quiera, aunque intente dar una visión general.

[1] Al principio me despistaba mucho el uso de este término dado que en software engineering, usualmente, se considera que verificación es (com)probar que la especificación cumple los requisitos… vamos, model checking….

- No tengo claro qué pretendemos con el model checking, ¿que lo sepan usar y que además lo entiendan? Donde entender quiere decir que sepan del automata de Büchi y todo lo demás, no que sepan que "se comprueban todos los estados".

Ricardo:

En cuanto al contenido, la ficha que envias Manuel es bastante abierta porque permite hablar desde algebras de procesos, sistemas de especificacion como Maude, o logicas modales como dice Adrian. En la parte de testing, tambien queda abierto a sistemas como QuickCheck que es una herramienta de mucho exito en la vida real (hay una version comercial para Erlang, aparte de la de Haskell, C y Java). Tambien podria ser util en la parte de verificacion hablar de demostradores asistidos como Isabelle o Coq, que se estan empleando actualmente como herramientas no solo de verificacion sino tambien de certificacion.