Curso Académico 2011-12

454 PROGRAMACION CONCURRENTE

Ficha Docente

ASIGNATURA

Nombre de asignatura (Código GeA): 454 PROGRAMACION CONCURRENTE (106168)
Créditos: 9

PLAN/ES DONDE SE IMPARTE

Titulación: INGENIERIA EN INFORMATICA
Plan: 36098 - INGENIERO EN INFORMATICA
Curso: **Ciclo:** 2
Carácter: OPTATIVA
Duración/es: Anual (actas en Jun. y Sep.)
Idioma/s en que se imparte:

PROFESOR COORDINADOR

Nombre	Departamento	Centro	Correo electrónico	Teléfono

PROFESORADO

Nombre	Departamento	Centro	Correo electrónico	Teléfono
MARTINEZ TORRES, RAFAEL	Sistemas Informáticos y Computación	Facultad de Informática	rmartine@fdi.ucm.es	91394 7555

SINOPSIS

REQUISITOS:

Para la parte teórica, matemática discreta, lógica, verificación y derivación de programas secuenciales, fundamentos de algoritmia. Para la parte del laboratorio, cierta experiencia en el manejo de lenguajes como C/C++, y/o Java, y/o C#

OBJETIVOS:

Estudio de modelos teóricos de la programación concurrente de forma que se puedan reconocer y sintetizar algoritmos adecuadamente sincronizados en un pseudo-lenguaje que pueda ser traducido sin mucho esfuerzo a programas en un lenguaje al uso como C/C++, Java, C#... Asímilar ciertos patrones recurrentes de programación distribuida de utilidad para la resolución de problemas concretos.

CONTENIDOS TEMÁTICOS:

1. Primer Cuatrimestre: Memoria compartida. Presentado un pseudo-lenguaje imperativo con extensiones para la ejecución simultánea de procesos, se formula con precisión el problema de la interferencia de procesos debido a la diferencia de velocidades y en base a ello se da un método sistemático basado en predicados invariantes para la correcta sincronización de los mismos mediante primitivas a nivel HW, semáforos o monitores.

    - Semántica axiomática de un pseudo lenguaje concurrente (O'Wicky-Gray)
    - Verificación y derivación formal de algoritmos concurrentes.
    - Primitivas de sincronización disponibles (HW, semáforos, monitores) .
    - Algunos problemas clásicos de concurrencia.
    - Pequeña práctica de memoria compartida.

2. Segundo Cuatrimestre: Paso de Mensajes.En la comunicación asíncrona los procesos intercambian mensajes mediante un medio de transmisión -el canal- y consecuentemente el pseudo-lenguaje se extiende para dar cobertura a las acciones de comunicación: emisión no bloqueante y recepción bloqueante . El estado del canal pasa así a ser una variable más a considerar en el cálculo. Algunas dificultades inherentes a este modelo se simplifican en la variante de comunicación síncrona, donde tanto emisor y receptor quedan bloqueados en el acto de comunicación.

    - Semántica axiomática de un pseudo lenguaje de paso de ménsajes asíncrono/síncrono.
    - Verificación de pequeños algoritmos distribuidos.
    - Algunos patrones recurrentes de interacción en programación distribuida (cliente/servidor, sistólicos, sondeo y prueba, difusión...)
    - Pequeña práctica de paso de mensajes.

ACTIVIDADES DOCENTES:

Enseñanza presencial de teoría y prácticas de laboratorio.

EVALUACIÓN:

Primer Cuatrimestre: Prácticas con plazos de entrega. Examen escrito.
Segundo Cuatrimestre: Prácticas con plazos de entrega. Examen escrito (coincidente con la convocatoria de junio).
Todos los exámenes son escritos y se realizarán en el aula.
La NOTA FINAL será la media aritmética de los dos exámenes de los cuatrimestres siempre que se obtenga un mínimo de 4 puntos en cada uno de ellos.
Alternativamente la que se obtenga en el final de Junio o Septiembre. Las prácticas sirven en su caso para apoyar la nota del examen.
Examenes finales en Junio y Septiembre.

BIBLIOGRAFÍA BÁSICA:

Greg Andrews; Concurrent Programming: Principles and Practice; Benjamin/Cummings, 1991;

BREVE DESCRIPTOR:

Estudio de los problemas inherentes a la programación concurrente en sus dos modalidades, memoria compartida y paso de mensajes. Primitivas para la programación concurrente. Esquemas de interacción en programación distribuida. Lenguajes y formalismos para la especificación e implementación de procesos concurrentes.

OTRA INFORMACIÓN RELEVANTE:

Aunque no se excluyen pequeñas prácticas didácticas en el laboratorio, el curso es predominantemente teórico y ha de servir como soporte a otras asignaturas donde la concurrencia sea parte del problema concreto a resolver y no objeto de estudio en si mismo. El alumno no ha de esperar un curso intensivo sobre "threads" o "sockets" en C o Java, ya que todos los algoritmos se formulan en un pseudo lenguaje imperativo, optando por la generalidad y sacrificando la aplicabilidad.