Desarrollo de Software Basado en Componentes y Servicios (DSBCS)

M.I. Capel

Departamento de Lenguajes y Sistemas Informáticos
Email: manuelcapel@ugr.es
http://lsi.ugr.es/mcapel/

3 de octubre de 2018

Presentación de la asignatura

Información General

- Profesor: Manuel I. Capel
  email: manuelcapel@ugr.es
- Departamento: LSI
- Despacho: A-37 (3^a planta)

Tutorías

L	M	X	J	V
15.30-17.30	-	11.30-13.30	11.30-13.30	-

Cuadro: (*) se puede acordar una tutoría con el profesor fuera de este horario enviando un mensaje a manuelcapel@ugr.es

Coordenadas de la asignatura

DDBCS

Máster en Ingeniería Informática
Máster Universitario en Ingeniería Informática
Módulo: Tecnologías Informáticas 1
Materia: Sistemas Basados en Componentes y Servicios
Carácter: Obligatoria
Carga docente: 4 cr.
Grupos teoría: 1 (Jueves 16.00-17.30, 1.6)
Grupos prácticas: 1 (Martes 19.00-20.30, 2.2)

DDBCS

Teoría: 12 sesiones= 18,0 hrs (comienza: 4/10; termina: 17/01/2019)
Prácticas: 12 sesiones= 18,0 hrs (comienza: 9/10; termina: 15/01/2019)
+ 4 horas dedicadas a la defensa/exposición (15, 17/01/2019) de trabajos y sesión de evaluación(a fijar:del 28/01 al 8/02 de 2019)

Objetivos formativos

Objetivos (BOE 8 de junio de 2009)

- Comprender los modelos de componentes actuales
- Tendencias de desarrollo de software con énfasis en componentes y servicios
- Apreciar las ventajas que reporta basarse en componentes y servicios, especialmente, para validación
- Modelos formales esenciales de respaldo
- Arquitecturas software actuales, patrones y estilos, así como conocer su impacto en el desarrollo de software
- Planificar la evolución de un sistema software y evaluar el nivel de calidad que mantiene
- Fundamentos, herramientas y distribuciones libres disponibles del software de intermediación
- Arquitecturas de servicios y cómo aplicar las metodologías y tecnologías apropiadas en casos de aplicación

Objetivos formativos II

Capacidades que se han de adquirir

- Aplicar las arquitecturas más adecuadas, los componentes que las integran, las interfaces que se definen entre ellos, los patrones que supervisan su composición
- Diferenciar los paradigmas “Grid Computing” y “Cloud Computing”, sabiendo cuál aplicar en cada caso
- Obtener provecho de conductores de software inspirados en software intermediario para desarrollar aplicaciones y servicios específicos
- Diseñar y utilizar marcos de trabajo para la construcción de sistemas software distribuidos de calidad en diferentes dominios de aplicaciones
- Más información en:
  http://masteres.ugr.es/ing-informatica

Programa de Teoría

I. Desarrollo de software basado en componentes y servicios
   - Formalización de los sistemas abiertos y basados en componentes.
   - Técnicas de diseño y desarrollo basadas en componentización del software
   - Resolución de ejercicios

II. Servicios Web y Procesos de negocio
   - Limitaciones del software intermediario (middleware) convencional.
   - Servicios Web contemporáneos (WS 2.0)
   - Programación de SW
   - Desarrollo de software, basado en SW, para procesos de negocio
   - Composición de SW: orquestación y coreografía. Notaciones y lenguajes actuales

III. Sistemas Ubicuos e Inteligencia Ambiental
   - Introducción a la Computación Ubicua
   - Frameworks actuales para el desarrollo de sistemas ubicuos
   - Servicios colaborativos
   - Modelado ontológico con OWL
   - Casos de estudio

Seminarios

Tema

I Especificación de componentes software con UML/OCL

II Introducción al diseño/implementación/despliegue de servicios Web

III Introducción a la orquestación de servicios Web complejos: WS-BPEL

IV OWL y modelado semántico de ontologías para la Web Semántica

Programa de Prácticas

1. Programación de componentes-software distribuidos con el marco de trabajo JSF
2. Desarrollo de un servicio Web CRUD con persistencia de entidades e interfaz REST
3. Modelado de procesos de negocio propuestos como casos de estudio con BPEL 2.0
4. Desarrollo completo de una aplicación receptiva y adaptable para dispositivos móviles y su interfaz Web RESTful. Desarrollo del servicio y base de datos en la parte servidora

Práctica(*)	Fecha indicativa de comienzo
1	09/10/2018
2	16/10/2018
3	30/10/2018
4	13/11/2017

12 sesiones de prácticas + 1 sesión de defensa/exposición

(*) Fecha límite de entrega de prácticas: 11/01/2019 (23:00 hrs)

Modelo de evaluación: *sistema de evaluación continua*

Teórico (NET)

- varias pruebas de comprensión (*tests*) y entregas de ejercicios (*tareas*), sobre el desarrollo y los resultados, en Prado en las fechas establecidas para cada una

Práctico (NEP)

- asistencia a prácticas obligatoria (máximo 3 faltas justificadas) para ser calificado con el *sistema de evaluación continua* del aprendizaje
- se pueden realizar de forma individual o en grupos (2 máximo)
- entrega de resultados parciales, que serán calificados
- reuniones de seguimiento de las prácticas con el profesor
- la última práctica realizada se expondrá a toda la clase, en fecha y hora anunciada con anterioridad

Calificación:

$$0,5 \times NET + 0,5 \times NEP$$

Modelo de evaluación: *evaluación única* y convocatoria de septiembre

- **Evaluación:** examen teórico-práctico (10 puntos)
- **Evaluación en la convocatoria de septiembre para ambas modalidades:** Sólo se realizará evaluación con examen teórico/práctico

Documentación de la asignatura

Toda la documentación de la asignatura se gestiona en la plataforma Prado (https://pradoposgrado.ugr.es), así como:

- Notificación de resultados de pruebas y exámenes
- Entrega de ejercicios, trabajos y prácticas
- Comunicación entre alumno y profesor

Recomendaciones

Es importante:

- Utilizar las horas de tutoría
- Mantenerse informado de la marcha de la asignatura: acceder al sitio de la asignatura *Desarrollo de Sistemas de Software basados en Componentes y Servicios-1718*) a través de acceso identificado en *Prado* https://pradoposgrado.ugr.es/moodle/course/view.php?id=70
- Participar en clase
- Revisar la bibliografía, no limitarse sólo a leer las diapositivas

Bibliografía:


Aalst, W. v. d., Benatallah, B., Casati, F., Curbera, F., and Verbeek, H. (2007).
Business process management: Where business processes and web services meet (guest editorial).
Data & Knowledge Engineering, 61(1):1–5.


Armbrust, M. and etal. (2009).
Above the Clouds: A Berkeley View of Cloud Computing.
ACM.


Armbrust, M. and etal. (2010).
A view of cloud computing.
Communications of the ACM, 53:50–58.


Bell, M. (2010).
SOA Modeling Patterns for Service Oriented Discovery Analysis.
Wiley, Complementaria.


Capel, M. (2016).
Desarrollo de software ys sistemas basados en componentes y servicios.
Garceta, **Básica**.


Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J. (2010).
Documenting Software Architectures: Views and Beyond.
Addison-Wesley, second edition.


deSilva, L. and Balasubramaniam, D. (2012).
Controlling software architecture erosion: a survey.
Journal of Systems and Software, 85(01/2012):132–151.


Developer and Works (2009).
Cloud computing versus grid computing.
IBM.


Erl, T. (2004).
Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services.
Prentice-Hall.


Erl, T. (2008).
SOA. Principles of Service Design.
Prentice-Hall.


Fowler, M. (2003).
Patterns of enterprise application architecture.
Addison–Wesley, Boston, Mass.


Lungu, M. (2008).
Software architecture recovery.
University of Lugano, http://www.slideshare.net/mircea.lungu/software-architecture-recovery-in-five-questions-presentation, 2008 edition.


Maranzano, J., Rozsypal, S., Zimmerman, G., Warnken, G., Wirth, P., and Weiss, D. (2005).
Architecture reviews: Practice and experience.
IEEE Software, 22(2).


Marcs, E.D. and Bell, M. (2006).
Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology.
Wiley, **Básica**.


Osterwalder and Pigneur (2004).
An ontology for e-business models.
Butterworth-Heinemann, pages 65–97.


Silver, B. (2011).
BPMN Method and Style: with BPMN Implementer's Guide.
Cody-Cassidy Press, Complementaria.


Szyperski, C. (1998).
Component Software. Beyond Object-Oriented Programming.
Addison–Wesley.


Tang, A., Han, J., and Vasa, R. (2009).
Software architecture design reasoning: A case for improved methodology support.
IEEE Software, 26(2).


Taylor, H. (2009).
Event-driven Architecture: How SOA Enables the Real-time Enterprise.
Addison–Wesley, Complementaria.


Terra, R., Valente, M., Czarnecki, K., and Bigonha, R. (2012).
Recommending refactorings to reverse software architecture erosion.
In *16th European Conference on Software Maintenance and Reengineering*.


Verissimo, P. and Rodrigues, L. (2004).
Distributed Systems for System Architects.
Kluwer Academic.


von der Beeck, M. (2000).
Behaviour specifications: Equivalence and refinement notions.
In *Visuelle Verhaltensmodellierung Verteilter und Nebenlaeufiger Software–Systeme, 8. Workshop des Arbeitskreises GROOM der GI Fachgruppe 2.1.9 OO Software-Entwicklung*, pages 1–5, Paderborn, D. Universitaet Munster.


Woods, E. (2012).
Industrial architectural assessment using tara.
Journal of Systems and Software, 85(9):2034–2047.


zur Muehlen, M., Nickerson, J. V., and Swenson, K. D. (2005).
Developing web services choreography standards—the case of REST vs. SOAP.
Decission Support Systems, 40(1):9–29.

Texto


Garceta
Desarrollo de software y sistemas
basados en componentes y servicios
Manuel I. Capel Tuñón

Biblio complementaria