FACULTAD DE CIENCIAS EXACTAS, INGENIERIA Y AGRIMENSURA
U.N.R.

PROGRAMA ANALITICO DE LA ASIGNATURA: Ingeniería de Software II

Código R-421

PLAN DE ESTUDIOS: 2010
CARRERA: Licenciatura en Ciencias de la Computación
DEPARTAMENTO: Cs. de la Computación (ECEN)
PROFESOR: Maximiliano Cristiá

2013 **HASTA AÑO**

TENTATIVO ~~**DEFINITIVO**~~ **DE EXAMEN**

PROGRAMA

~~**ANUAL**~~ ~~**SEMESTRAL**~~ **CUATRIMESTRAL**

Táchese lo que no corresponda.

OBSERVACIONES:

PRESUPUESTO HORARIO SEMANAL PROMEDIO		
TEORÍA:	4	1
PRÁCTICA:	4	2
LABORATORIO:	1	3
TOTAL ASIGNADO:	9	4 1+2+3
DEDICACIÓN DEL ALUMNO FUERA DE CLASE:	8	5
PRESUPUESTO TOTAL:	17	6 5+4
PROGRAMA BASADO EN SEMANAS ÚTILES :	15	7
HORAS TOTALES ASIGNADAS:	135	7x4
HORAS TOTALES PRESUPUESTAS:	255	7x6

OBJETIVOS: (qué debe saber el alumno al concluir el curso)

(1) Debe manejar con fluidez el principio de Ocultamiento de la Información para el diseño de sistemas de software y la metodología de diseño sugerida por David L. Parnas.
(2) Debe entender claramente las semejanzas, diferencias y alcances del Diseño Basado en Ocultamiento de la Información (DBOI), el Diseño Basado en Tipos Abstractos de Datos (DTAD) y el Diseño Orientado a Objetos (DOO).
(3)Debe ser capaz de diseñar sistemas de software utilizando apropiadamente DBOI, DTAD y DOO.
(4) Debe utilizar con fluidez los patrones de diseño dentro del dominio del DOO.
(5) Debe conocer los principales estilos arquitectónicos para la construcción de sistemas de software de tamaño industrial, sus ventajas y desventajas relativas.
(6) Debe ser capaz de seleccionar la arquitectura más conveniente para cada sistema.
(7) El alumno debe ser capaz de aplicar las principales técnicas de testing de software.

UBICACIÓN EN LA CARRERA Y CARACTERISTICAS GENERALES:

Materia obligatoria del 4º año de la carrera.
Está orientada a que el alumno se introduzca en la complejidad de los grandes sistemas de software y que comprenda que ésta puede ser dominada definiendo una Arquitectura y un Diseño antes de comenzar a programar.

MATERIAS RELACIONADAS:

Previas: R-411 Ingeniería de Software I
Simultáneas recomendadas: - - -
Posteriores: - - -

.....	….....	.....	.....
Firma Profesor	**Fecha**	**Aprob. Escuela**	**Fecha**

Aprobado en reunión de Consejo Académico de fecha: .....

CONTENIDO TEMATICO

Ordenar temas utilizando codificación decimal

1. Unidad I: Diseño de software

1.1. Introducción

1.1.1. El diseño como fase del ciclo de desarrollo del sistema.
1.1.2. Diferencias entre modelo funcional y diseño.
1.1.3. Relación entre el modelo funcional y el diseño.
1.1.4. Relación entre los métodos formales y el diseño de software.

1.2. Problemas con el diseño funcional o estructurado

1.3. Diseño basado en ocultación de la información (DBOI)

1.3.1. Diseño para el cambio; la metodología de David L. Parnas; ítem con probabilidad de cambio; componentes, interfaz e implementación; abstracción y encapsulamiento.
1.3.2. Un ejemplo de DBOI; documentación del diseño; lenguaje 2MIL; limitaciones del DBOI.

1.4. Diseño Basado en Tipos Abstractos de Datos (DTAD)

1.5. Diseño Orientado a Objetos (DOO)

1.5.1. El concepto de herencia

1.6. Documentación de diseño

1.6.1. Ítem con probabilidad de cambio; interfaces; estructura de módulos; guía de módulos; estructura de uso; estructura de procesos; estructura de objetos; estructura de herencia.

1.7. Reingeniería de software

1.7.1. Reconstrucción del diseño a partir del código fuente; modificación del diseño para incorporar metodologías más modernas; cambios a la implementación para adecuarla al nuevo diseño.

1.8. Comentarios sobre diseño de software concurrente y de tiempo rea

1.9. Comentarios sobre diseño centrado en el usuario

2. Unidad II: Patrones de diseño

2.1. Introducción

2.1.1. Herencia de clase y herencia de interfaces; herencia y composición de objetos.

2.2. Estudio y aplicación de algunos patrones de diseño

2.2.1. Composite; Abstract Factory; Command; Iterator; Visitor; Bridge; Decorator; Strategy.

3. Unidad III: Arquitecturas de software

3.1. Vocabulario, conceptos y problemas

3.1.1. Diferencias entre arquitectura y diseño; componentes y conectores; estilos arquitectónicos.

3.2. Estilos arquitectónicos

3.2.1. Un catálogo incompleto de estilo arquitectónicos.
3.2.2. Invocación Implícita.
3.2.3. Tubos y filtros.
3.2.4. Sistemas Estratificados.
3.2.5. Control de procesos
3.2.6. Blackboard Systems.

3.2.7. Cliente/Servidor de Tres Capas.

4. Unidad IV: Testing de software

4.1. Introducción al testing de software

4.1.1. Verificación y Validación (V&V); definición de testing y vocabulario básico; el proceso de testing; testing de distintos aspectos de un software; las dos metodologías clásicas de testing.

4.2. Testing estructural basado en flujo de control

4.2.1. Grafo de flujo de control de un programa; criterios de testing estructural basado en flujo de control: cubrimiento de sentencias, cubrimiento de flechas, cubrimiento de condiciones, criterio de valores de verdad, criterio de condiciones múltiples.

4.3. Testing basado en especificaciones Z

4.3.1. Introducción: programas y especificaciones, casos de prueba exitosos, espacio válido de entrada, funciones de refinamiento y abstracción, el proceso de testing basado en especificaciones formales.

4.3.2. Tácticas de testing funcional: formal normal disyuntiva, particiones estándar, propagación de subdominios, mutación de especificaciones, causa-efecto, otras tácticas.

BIBLIOGRAFIA

a) Adecuada al programa. Ordenada por temas y con su codificación de biblioteca, incluidas las publicaciones de la Cátedra con su código de publicación.

5. Unidad I

5.1. Introducción

5.1.1. Shaw, M., Garlan, D., Software architecture: perspectives on an emerging discipline, Prentice Hall, Upper Saddle River, 1996.

5.1.2. Sommerville, I., Software Engineering, Addison-Wesley, Harlow, 1995.

5.1.3. Bass, L., Clements, P., Kazman, R., Software architecture in practice 2nd Edition, Addison-Wesley, Reading, 2002.

5.1.4. Ghezzi, C., Jazayeri, M. y Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, Upper Saddle River, 1991.

5.1.5. Maximiliano Cristia; Comentarios sobre Diseño de Software; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

5.2. Problemas con el diseño funcional o estructurado

5.2.1. Sommerville, I., Software Engineering, Addison-Wesley, Harlow, 1995.

5.2.2. Ghezzi, C., Jazayeri, M. y Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, Upper Saddle River, 1991.

5.2.3. Maximiliano Cristia; Comentarios sobre Diseño de Software; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

5.3. DBOI

5.3.1. Maximiliano Cristia; Comentarios sobre Diseño de Software; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

5.3.2. Ghezzi, C., Jazayeri, M. y Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, Upper Saddle River, 1991.

5.3.3. Parnas, D.L., "On the criteria to be used in decomposing systems into modules", Communications of the ACM, 15(12): 1053-1058, diciembre 1972.

5.3.4. Parnas, D.L., "Designing software for ease of extension and contraction", IEEE Transactions on Software Engineering, 5(2): 128-137, marzo 1979.

5.4. DTAD

5.4.1. Maximiliano Cristia; Comentarios sobre Diseño de Software; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

5.4.2. Ghezzi, C., Jazayeri, M. y Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, Upper Saddle River, 1991.

5.5. DOO

5.5.1. Maximiliano Cristia; Comentarios sobre Diseño de Software; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

5.5.2. Ghezzi, C., Jazayeri, M. y Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, Upper Saddle River, 1991.

5.6. Documentación de diseño

5.6.1. Maximiliano Cristia; Comentarios sobre Diseño de Software; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

5.6.2. Ghezzi, C., Jazayeri, M. y Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, Upper Saddle River, 1991.

5.6.3. Paul Clements, David Garlan, Len Bass, Judith Stafford, Robert Nord, James Ivers y Reed Little; Documenting Software Architectures: Views and Beyond; 2002; ISBN 0201703726; Pearson Education.

5.7. Reingeniería de software y diseño de software concurrente y de tiempo real

5.7.1. Sommerville, I., Software Engineering, Addison-Wesley, Harlow, 1995.

5.7.2. Bass, L., Clements, P., Kazman, R., Software architecture in practice 2nd Edition, Addison-Wesley, Reading, 2002.

5.7.3. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stad; Pattern-Oriented Software Architecture: A System of Patterns; John Wiley Press; 1996; ISBN 0-471-95869-7.

6. Unidad II

6.1.1. Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides; Patrones de diseño; Addison Wesley; 2003.

7. Unidad III

7.1. Vocabulario, conceptos y problemas

7.1.1. Bass, L., Clements, P., Kazman, R., Software architecture in practice 2nd Edition, Addison-Wesley, Reading, 2002.

7.1.2. Shaw, M., Garlan, D., Software architecture: perspectives on an emerging discipline, Prentice Hall, Upper Saddle River, 1996.

7.1.3. Shaw, M., Garlan, D., "Formulations and Formalisms in Software Architecture", Computer Science Today: Recent Trends and Developments, Springer-Verlag, 1995.

7.2. Estilos arquitectónicos

7.2.1. Shaw, M., Garlan, D., Software architecture: perspectives on an emerging discipline, Prentice Hall, Upper Saddle River, 1996.

7.2.2. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stad; Pattern-Oriented Software Architecture: A System of Patterns; John Wiley Press; 1996; ISBN 0-471-95869-7.

7.2.3. Berson, A., Client/Server Architecture, McGraw-Hill, 1992.

7.2.4. Nii, H.P., "Blackboard systems", AI Magazine, 7(3):38-53 y 7(4):82-107, julio 1986.

7.2.5. Garlan, D., Kaiser, G., Notkin, D., "Using tool abstraction to compose systems", IEEE Computer, 25(6):30-38, junio 1992.

7.2.6. Paul Clements, David Garlan, Len Bass, Judith Stafford, Robert Nord, James Ivers y Reed Little; Documenting Software Architectures: Views and Beyond; 2002; ISBN 0201703726; Pearson Education.
7.2.7. Maximiliano Cristiá; Un catálogo incompleto de estilos arquitectónicos; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

8. Unidad IV

8.1. Introducción al testing de software

8.1.1. Maximiliano Cristiá; Introducción al testing de software; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

8.2. Testing estructural

8.2.1. Sommerville, I., Software Engineering, Addison-Wesley, Harlow, 1995.
8.2.2. Ghezzi, C., Jazayeri, M. y Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall, Upper Saddle River, 1991.
8.2.3. Maximiliano Cristiá; Introducción al testing estructural; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

8.3. Testing basado en modelos

8.3.1. Hörcher, H., Peleska, J., "Using formal specifications to support software testing", Software Quality Journal, 4: 309-327, 1995.
8.3.2. Stocks, P., "Applying formal methods to software testing", Tesis doctoral, Department of Computer Science, University of Queensland, 1993.
8.3.3. Maximiliano Cristiá; Testing basado en especificaciones Z; apunte de clase; www.fceia.unr.edu.ar/ingsoft.

b) Complementaria para profundización o extensión de temas.

9. Unidad III

9.1.1. Abowd, G., Allen, R., Garlan, G., "Formalizing style to understand descriptions pf software architecture", ACM Transactions on Software Engineering and Methodology, 4(4): 319-364, octubre 1995.
9.1.2. Allen, R., "A Formal Approach to Software Architecture", Tesis doctoral, Carnegie Mellon School of Computer Science, 1997.
9.1.3. Bass, L., Clements, P., Kazman, R., Software architecture in practice, Addison-Wesley, Reading, 1998. Páginas 75-91, 267-284.
9.1.4. Garlan, D., Monroe, R., Wile, D., "ACME: An Architecture Description Interchange Language", Proceedings of CASCON'97, noviembre 1997.
9.1.5. Moriconi, M., Riemenschneider, R., "Introduction to SADL 1.0. A languaje for specifying architecture hierachies", Computer Science Laboratory, SRI International, marzo 1997.
9.1.6. Reiss, S., "Connecting tools using message passing in the Field environment", IEEE Software, 7(4):57-66, julio 1990.
9.1.7. Shaw, M., Garlan, D., Software architecture: perspectives on an emerging discipline, Prentice Hall, Upper Saddle River, 1996. Páginas 43-51, 82-96, 165-172.

10. Unidad IV

10.1.1. Richards Adrion, W., Branstad, M. A., Cherniavsky, J. C., "Validation, verification, and testing of computer software", ACM Computing Surveys, 11(2): 159-192, junio 1982.

10.1.2. Stocks, P., Carringtong, D., "A framework for specification-based testing", IEEE Transactions on Software Engineering, 22(11): 777-793, noviembre 1996.
10.1.3. Rapps, S., Weyuker, E.J., "Data flow analysis techniques for test data selection", Proceedings of IEEE International Conference on Software Engineering, 272-278, septiembre 1982.
10.1.4. Tai, K., "Theory of fault-based predicate testing for computer programs", IEEE Transactions on Software Engineering, 22(8): 552-562, agosto 1996.