TEC | Tecnológico de Costa Rica

Programa del curso IC-6821

Diseño de Software

Escuela de Computación
Carrera de Ingeniería en Computación, Plan 411.

TEC | Tecnológico de Costa Rica

I parte: Aspectos relativos al plan de estudios

1 Datos generales

Nombre del curso:	Diseño de Software
Código:	IC-6821
Tipo de curso:	Teórico-Práctico
Electivo o no:	No
Nº de créditos:	4
Nº horas de clase por semana:	4
Nº horas extraclase por semana:	8
Ubicación en el plan de estudios:	Curso del V Semestre del Bachillerato de Ingeniería en Computación
Requisitos:	IC-5821 Requerimientos de Software
Correquisitos:	Ninguno.
El curso es requisito de:	IC-6831 Aseguramiento de la Calidad del Software
Asistencia:	Obligatoria
Suficiencia:	No
Posibilidad de reconocimiento:	Sí
Vigencia del programa:	I Semestre de 2018

TEC | Tecnológico de Costa Rica

2 Descripción general

En este curso se estudia el diseño de software, dentro del contexto del proceso de desarrollo de software. Se espera que al finalizar el curso el estudiante sea capaz de diseñar software mediante un proceso de diseño sistemático.

3 Objetivos

Objetivo General

Valorar un conjunto de métodos, técnicas y herramientas para el diseño y la especificación de un producto de software.

Objetivos Específicos

Al finalizar el curso el estudiante estará en capacidad de:

1. Aplicar técnicas y herramientas orientadas a objetos para la modelación del diseño de software.
2. Documentar la toma de decisiones durante la etapa de diseño del software.
3. Comprender los diferentes niveles de abstracción en que deben expresarse las soluciones de problemas de diseño.
4. Desarrollar destrezas para diseñar la arquitectura de software
5. Analizar aspectos de las tecnologías actuales y de las tendencias tecnológicas que influyen en los diseños del software.

4 Contenidos

1. Introducción/vistazo al diseño y arquitectura de software

1.1. De los requerimientos al diseño.
1.2. Pasos para diseñar software
1.3. Artefactos del diseño de software
1.4. El rol del arquitecto de software

2. El diseño de interfaces de usuario

2.1. El concepto de utilizabilidad (“usability”)
2.2. Técnicas para diseñar interfaces de usuario (diseño en papel, reutilización de conceptos y diseños conocidos, diseño con usuarios, etc.)
2.3. Diseño de la parte funcional de la interfaz, considerando diferentes tecnologías, eficiencia y eficacia
2.4. La importancia de la validación de datos
2.5. Tipos de componentes para el diseño de interfaces de usuario
2.6. Software para el diseño de interfaces de usuario

2.7. Guías de diseño gráfico de la interfaz, considerando diferentes tecnologías
2.8. Consideraciones para el diseño de reportes

3. Principios de diseño
3.1. Principio de divide y conquista
3.2. Principio de incrementar la cohesión
3.3. Principio de reducción de acoplamiento
3.4. Principio de mantenimiento del nivel de abstracción alto
3.5. Principio de incrementar la reusabilidad y reusar lo existente
3.6. Principio del diseño para la flexibilidad
3.7. Principio de anticipar la obsolecencia
3.8. Principio de diseñar para la portabilidad
3.9. Principio del Diseño para pruebas ("testabilidad")
3.10. Principio del diseño defensivo

4. El diseño de la arquitectura del software
4.1. El entendimiento del problema
4.2. Identificación de elementos y sus relaciones
4.3. Descomposición del sistema
4.4. Diseño con componentes
4.5. Diseño con servicios
4.6. Diseño de la integración entre sistemas
4.7. El papel de los atributos de calidad (requerimientos no funcionales) en la especificación de la arquitectura
4.8. Especificación de la arquitectura de software (artefactos)
4.9. Estilos y patrones arquitectónicos
4.10. Tendencias tecnológicas en arquitectura de software
4.11. Arquitectura de aplicaciones empresariales

5. El diseño detallado del software
5.1. Artefactos del diseño detallado
5.2. Diseño de los casos de uso
5.3. Diseño de los componentes y de los servicios
5.4. Diseño de clases
5.5. Aplicación de patrones de diseño en el diseño de clases

6. El proceso de revisión del diseño
6.1. En relación con los requerimientos funcionales

6.2. En relación con los atributos de calidad del software (requerimientos no funcionales)
6.3. En relación con los requerimientos tecnológicos.
6.4. Revisión y Control del diseño durante el proceso de construcción de software.

7. **Tendencias en el diseño de software**

II parte: Aspectos operativos

5 Metodología de enseñanza y aprendizaje

Exposición magistral de los temas y análisis de casos de estudio, ejercicios y proyectos prácticos para afianzar los conocimientos, desarrollar habilidades y destrezas del trabajo en equipo.

6 Evaluación

La evaluación consiste en múltiples tareas cortas programadas que cubren los contenidos vistos en clase y un proyecto de programación de tamaño considerable que será desarrollado durante el semestre y que incorpora los diferentes conceptos del curso

Rubro	**Porcentaje**
Trabajos Individuales	20%
Participación en clase	10%
Estudios de Caso (Formato conferencia)	20%
Proyectos Prácticos	50% distribuido: 5% por presentación de avance 15% presentación final 10% Diseño de Interfaz 10% Diseño de Clases 5% Arquitectura de Software
Total	100%

Cronograma de Actividades

Actividad	Semanas
Introducción/vistazo al diseño y arquitectura de software	0.5
El diseño de interfaces de usuario	3.5
Principios de diseño	1
El diseño de la arquitectura del software	5
El diseño detallado del software	4
El proceso de revisión del diseño	1
Tendencias en el diseño de software	0.5

7 Bibliografía

Obligatoria

Primer: Agile Model-driven development with UML 2.0. Cambridge University Press.

Liskov, B. (2001). *Program Development in Java: Abstraction, Specification, and Object-oriented Design.* Addison-Wesley.

Braude, E. J. (2004). *Software Design: from programming to architecture.* John Wiley & Sons.

Lethbridge, T. C., & Laganiere, R. (2004). *Object-Oriented Software Engineering: Practical Software Development using UML and Java* (2nd edition ed.). McGrawHill.

Albin, S. (2003). *The Art of Software Architecture: Design Methods and Techniques.* John Wiley & Sons.

Booch. *Análisis y Diseño Orientado a Objetos Con Aplicaciones.* Addison-Wesley.

Budgen, D. (2003). *Software Design* (2nd. Edition ed.). Addison-Wesley.

Fowler, M. (1997). *UML Gota a Gota.* Addison-Wesley,.

IEEE Standard for Software Reviews. (1997).

Gorton, I. (2006). *Essential Software Architecture.* Springer-Verlag.

Jacobson, Booch, & Rumbaugh. (2000). *El Proceso Unificado de Desarrollo de Software.* Addison-Wesley.

Jacobson, Booch, & Rumbaugh. (1999). *El Lenguaje Unificado de Modelado.* Addison-Wesley.

Nielsen. (1993). *Usability Engineering.* Morgan Kaufmann.

Nielsen. (2000). *Designing Web Usability.* New Riders.

Sommerville. (2000). *Ingeniería de Software.* Addison-Wesley.

Fairley, R. (1988). *Ingeniería de Software.* McGraw-Hill/Interamericana de México.

Pressman. *Ingeniería del Software.* McGraw- Hill / interamericana de México.

Rumbaugh, J. (1991). *Objetct-Oriented Modeling and Design.* Prentice Hall.

Larman, C. (2003). *UML y Patrones* (2da edición ed.). Prentice Hall.

(s.f.). Obtenido de www.microsoft.com

(s.f.). Obtenido de www.sun.com

Adicional

No tiene Bibliografía adicional.

8 Profesor

Ing Jonathan Solis Parajeles.

Tel: 87245688

Email: josolis@itcr.ac.cr