ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

CONTENIDO DE CURSO

INGENIERÍA DE SOFTWARE II

SOFG1003

A. IDIOMA DE ELABORACIÓN

Español

B. DESCRIPCIÓN DEL CURSO

El curso permite a los estudiantes desarrollar un sistema de software de mediana complejidad, para lo cual el estudiante debe desarrollar habilidades técnicas y no técnicas. En el primer grupo de habilidades, se encuentran las buenas prácticas de programación y el desarrollo de varios tipos de pruebas, a fin de garantizar la funcionalidad y confiabilidad del sistema desarrollado. El segundo grupo de habilidades incluye el trabajo en equipo, el aprendizaje continuo (lenguajes de programación y/o herramientas requeridas por el cliente), y la comunicación oral y escrita.

C. CONOCIMIENTOS PREVIOS DEL CURSO

El estudiante que se registra en la materia debe tener la capacidad de:

- Diferenciar entre requerimientos funcionales y no funcionales
- Diseñar un sistema de software
- Programar en cualquier lenguaje de programación

D. OBJETIVO GENERAL

Desarrollar un sistema de software usando una metodología de desarrollo y herramientas para obtener un producto final debidamente probado y versionado, con código legible y fácil de mantener.

E. OBJETIVOS DE APRENDIZAJE DEL CURSO

El estudiante al finalizar el curso estará en capacidad de:

1	Usar una herramienta de control de código fuente para la gestión del desarrollo y versionamiento de un sistema de software de complejidad moderada desarrollado en equipo de mínimo 3 personas.
2	Usar un estándar de codificación para mejorar la calidad del código en un proyecto dado.
3	Desarrollar un conjunto de casos de prueba para un segmento de código de tamaño mediano con la finalidad de implementar adecuadamente las pruebas.
4	Analizar el impacto de una petición de cambio a un producto de software existente para determinar la prioridad y factibilidad del mismo.

F. ESTRATEGIAS DE APRENDIZAJE

Aprendizaje asistido por el profesor	✔
Aprendizaje cooperativo/colaborativo:	✔
Aprendizaje de prácticas de aplicación y experimentación:	✔
Aprendizaje autónomo:	✔

G. EVALUACIÓN DEL CURSO

Actividades de Evaluación	DIAGNÓSTICA	FORMATIVA	SUMATIVA
Exámenes			✔
Lecciones		✔	
Tareas		✔	
Proyectos			✔
Laboratorio/Experimental		✔	
Participación en Clase			
Visitas			
Otras			

H. PROGRAMA DEL CURSO

UNIDADES y SUBUNIDADES	Horas Docencia
1. Herramientas y ambientes de desarrollo de software.	6
1.1. Administración de la configuración del software y control de versiones.	
1.2. Administración de liberaciones de código	
1.3. Integración continua: conceptos y herramientas	
2. Construcción de software.	6
2.1. Prácticas de codificación	
2.2. Estándares de codificación	
3. Verificación y validación del software.	24
3.1. Verificación y validación: Conceptos	
3.2. Inspecciones, revisiones, auditorías	
3.3. Tipos de pruebas: Interfaz humano computador, usabilidad, confiabilidad, seguridad, cumplimiento de las especificaciones.	
3.4. Técnicas de pruebas: caja blanca y caja negra	
3.5. Creación del plan de pruebas y generación de casos de prueba	
3.6. Herramientas para automatizar pruebas	
3.7. Pruebas unitarias (repaso y uso)	
3.8. Pruebas de integración y del sistema	
3.9. Pruebas de regresión	
3.10. Seguimiento de defectos	
3.11. Limitaciones de las prueba en dominios particulares (sistemas paralelos o sistemas críticos para la seguridad)	
4. Evolución de software.	6
4.1. Evolución del software	
4.2. Características de un software mantenible	
4.3. Reingeniería de sistemas	
5. Confiabilidad del software	6
5.1. Confiabilidad del software: conceptos	
5.2. Confiabilidad del software, confiabilidad del sistema, y comportamiento de fallas.	
5.3. Fallas: conceptos y técnicas que se pueden aplicar para minimizarlas	

I. REFERENCIAS BIBLIOGRÁFICAS

BÁSICA	1. Jorgensen, Paul A.. (INGRESAR Año Publicación). Software testing. A craftsman's approach. (Fourth edition). INGRESAR Lugar Publicación: INGRESAR Editorial. ISBN-10: 1466560681, ISBN-13: 9781466560680

J. DESCRIPCIÓN DE UNIDADES

1. *Herramientas y ambientes de desarrollo de software.*

Introducción a la unidad

En esta unidad los estudiantes conocerán y usarán una herramienta para administrar las diferentes versiones de su producto de software.

Subunidades

1.1. Administración de la configuración del software y control de versiones.
1.2. Administración de liberaciones de código
1.3. Integración continua: conceptos y herramientas

Objetivos de Aprendizaje

1.1. Usar una herramienta de control de código fuente para la gestión del desarrollo y versionamiento de un sistema de software.

Actividades

1.1. Clase magistral

Taller en clases 1 - herramientas para administrar versiones
Taller en clases 2 - revisar herramientas para pruebas

1.2. Trabajo autónomo

Trabajo Autónomo 1:Usar las herramientas en el desarrollo de sus proyectos de software

2. *Construcción de software.*

Introducción a la unidad

En esta unidad los estudiantes codificarán el producto de software usando un estándar de codificación y estrategias para integrar los diferentes componentes desarrollados.

Subunidades

2.1. Prácticas de codificación
2.2. Estándares de codificación

Objetivos de Aprendizaje

2.1. Usar un estándar de codificación para mejorar la calidad del código de un sistema de software.

Actividades

2.1. Clase magistral

Taller 1: Definición del estándar de codificación a usar en cada grupo de trabajo
Taller 2: Usar el estándar mientras codifican en clases.

2.2. Trabajo Autónomo

Trabajo Autónomo 1: Usar un estándar de codificación en el desarrollo de sus proyectos de software

J. DESCRIPCIÓN DE UNIDADES

3. *Verificación y validación del software.*

Introducción a la unidad

En esta unidad, los estudiantes deben verificar y asegurar la calidad de su producto de software por medio de un conjunto de pruebas.

Subunidades

3.1. Verificación y validación: Conceptos
3.2. Inspecciones, revisiones, auditorías
3.3. Tipos de pruebas: Interfaz humano computador, usabilidad, confiabilidad, seguridad, cumplimiento de las especificaciones.
3.4. Técnicas de pruebas: caja blanca y caja negra
3.5. Creación del plan de pruebas y generación de casos de prueba
3.6. Herramientas para automatizar pruebas
3.7. Pruebas unitarias (repaso y uso)
3.8. Pruebas de integración y del sistema
3.9. Pruebas de regresión
3.10. Seguimiento de defectos
3.11. Limitaciones de las prueba en dominios particulares (sistemas paralelos o sistemas críticos para la seguridad)

Objetivos de Aprendizaje

3.1. Explicar la diferencia entre verificar y validar
3.2. Realizar una inspección de un segmento de código.
3.3. Desarrollar un conjunto de casos de prueba para un segmento de código de tamaño mediano con la finalidad de implementar adecuadamente las pruebas.

Actividades

3.1. Clase magistral

Taller 1: Inspecciones de código
Talleres 2 y 3: Pruebas de sistema
Talleres 4 y 5: Pruebas de caja blanca y unitarias
Taller 6: Pruebas de integración
Taller 7 y 8: Codificar y probar (pruebas unitarias, de integración, de regresión)

3.2. Trabajo Autónomo

Trabajo Autónomo 1: inspeccionar el código desarrollado
Trabajo Autónomo 2: Realizar pruebas unitarias para su proyecto
Trabajo Autónomo 3: Realizar pruebas de integración para su proyecto
Trabajo Autónomo 4: Realizar pruebas unitarias, de integración y de regresión para su proyecto

4. *Evolución de software.*

Introducción a la unidad

En esta unidad los estudiantes conocerán sobre la evolución de productos de software, los tipos de mantenimiento que existen y estrategias para realizar los cambios requeridos por el software.

Subunidades

4.1. Evolución del software

J. DESCRIPCIÓN DE UNIDADES

4.2. Características de un software mantenible
4.3. Reingeniería de sistemas

Objetivos de Aprendizaje

4.1. Analizar el impacto de una petición de cambio a un producto de software existente para determinar la prioridad y factibilidad del mismo.

Actividades

4.1. Clase magistral

Taller 1: Cambios en el software y análisis de impacto
Talleres 2: Refactorización de código
Talleres 3: Reingeniería de sistemas

4.2. Trabajo Autónomo

Trabajo Autónomo 1: Refactorizar su código

5. *Confiabilidad del software*

Introducción a la unidad

En esta unidad los estudiantes aprenderán conceptos relacionados a la confiabilidad del software, así como técnicas para minimizar las fallas en el mismo.

Subunidades

5.1. Confiabilidad del software: conceptos
5.2. Confiabilidad del software, confiabilidad del sistema, y comportamiento de fallas.
5.3. Fallas: conceptos y técnicas que se pueden aplicar para minimizarlas

Objetivos de Aprendizaje

5.1. Explicar los problemas que existen en alcanzar muy altos niveles de confiabilidad
5.2. Listar las alternativas que permiten minimizar las fallas en cada etapa del ciclo de vida del software.

Actividades

5.1. Clase magistral

Taller 1: Para una situación dada, identificar causas de falla posibles y cómo atacarlas.

5.2. Trabajo Autónomo

Trabajo Autónomo 1: identificar causas de falla posibles y cómo atacarlas – para su proyecto

K. RESPONSABLES DE LA ELABORACIÓN DEL CONTENIDO DE CURSO

Profesor	Correo	Participación
VILLAVICENCIO CABEZAS MONICA KATIUSKA	mvillavi@espol.edu.ec	Coordinador de materia
MONSALVE ARTEAGA CARLOS TEODORO	monsalve@espol.edu.ec	Colaborador
FALCONES MONTESDEOCA CRUZ MARIA	cfalcone@espol.edu.ec	Colaborador