****

***TÉCNICO SUPERIOR EN DESARROLLO DE SOFTWARE***

**Unidad Curricular:** **Ingeniería de Software ii**

**Ubicación en el Diseño Curricular:** 2º año

**Formato curricular:** materia

**Carga horaria semanal:** 4 hs. cátedra semanales

**Régimen de cursado:** Anual

**Profesora:** Susana Isabel Arce

**Año lectivo:** 2019

**Plan:** 2120/16

**FUNDAMENTACIÓN**

Esta unidad curricular tiene la finalidad de introducir al estudiante en el trabajo de Ingeniería, llevando a cabo proyectos con la utilización de métodos y la ayuda de herramientas propias de su ámbito de acción. Reconocer, diferenciar y validar las necesidades que justifican sus proyectos. Documentar escenarios y propuestas que respondan a dichas necesidades contemplando las demandas de artefactos requeridos por las empresas de la región.

**PROPÓSITOS**

El desarrollo de esta unidad curricular tiene como finalidad afianzar en los estudiantes
la capacidad de diseñar artefactos de software que resuelvan problemas planteados.
Analizar críticamente la eficiencia y mantenibilidad de diseños alternativos. Diseñar las
estructuras de datos e interfaces que las mismas utilicen. Analizar y discutir su
eficiencia y escalabilidad. A partir del diseño, analizar clases de equivalencia y diseñar
esquemas de prueba. Documentar el diseño.

**OBJETIVOS**

• Interpretar críticamente las especificaciones recibidas.
•Interpretar la arquitectura del sistema en el cual se inserta la asignación.
•Aplicar patrones de diseño si corresponde.
•Diseñar la solución.
•Representar el diseño.
•Verificar el diseño.

**CONTENIDOS****Principios generales de diseño**: descomposición, desacoplamiento, cohesión,
reusabilidad, portabilidad, testeabilidad, flexibilidad, escalabilidad. Patrones de diseño.
Arquitecturas de software: concepto de vistas, arquitecturas distribuidas, “pipe-andfilter”, “model-view-controller”.
**Diseño orientado a objetos**. Diseño estructurado. Diseño orientado al reuso de
componentes, incorporación de elementos disponibles al diseño. Diseño de interfaces
con el usuario. Prototipos rápidos.
Diseño orientado a objetos, encapsulamiento y ocultamiento de información, separación entre comportamiento e implementación, clases y subclases, herencia (sustitución), polimorfismo (subtipos vs. herencia), jerarquías de clases, clases colección y protocolos de iteración.
**Distinción entre validación y verificación**. Enfoques estáticos y dinámicos.
Fundamentos de “testing”, testeo de caja negra y de caja blanca. Pruebas funcionales:
generación de casos o datos de prueba, clases de equivalencia. Pruebas estructurales:
pruebas estáticas, pruebas dinámicas, cobertura de la prueba. Otro tipo de objetivos:
verificación de usabilidad, confiabilidad, seguridad. Registro de fallas, seguimiento de
fallas e informes técnicos.
Prueba unitaria, de integración, validación y prueba del sistema. Desarrollo conducido
por el testeo. Refactorización del código. Testeo de regresión. Verificación y validación
de artefactos que no constituyen código: documentación, archivos de ayuda, material de capacitación. Inspecciones, revisiones cruzadas, auditorías.
Herramientas para desarrollo en ambientes web. “Frameworks” de aplicaciones y su
utilización.
*Práctica Formativa*Como parte de la forma de adquirir estos aprendizajes y demostración práctica de los
resultados alcanzados, los estudiantes tienen que realizar en un mínimo del 33%, las
siguientes actividades:
Diseñar artefactos de software (clases, objetos, métodos, algoritmos, tablas) que
resuelvan problemas planteados. Analizar críticamente la eficiencia y mantenibilidad de diseños alternativos.
Relacionar situaciones con patrones de diseño. Analizar diversos tipos de arquitectura
de sistemas de software, discutiendo sus propiedades de calidad (escalabilidad,
portabilidad, seguridad, mantenibilidad). Construir prototipos rápidos con herramientas
sencillas.
Planificar y diseñar casos y conjuntos de datos para prueba de artefactos dados,
respondiendo a objetivos y requisitos de cobertura. Implementar pruebas de programas
y pequeños sistemas utilizando herramientas y creando ambientes necesarios, realizar
los procesos y revisar los resultados para generar informes de fallas.
Desarrollar proyectos grupales durante los cuales se simulen condiciones similares a las del trabajo profesional y en los que cada uno aporte componentes que deben integrarse en el producto final.

**METODOLOGÍAS DE TRABAJO**

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes.

Facilitar el contacto directo con materiales e instrumentos, al llevar a cabo actividades prácticas, para contribuir a la formación de las competencias para el trabajo experimental como: identificación, manejo y control de variables y datos relevantes, planteamiento de hipótesis, trabajo en equipo.

Fomentar actividades grupales que propicien la comunicación, el intercambio argumentado de ideas, la reflexión, la integración y la colaboración de y entre los estudiantes.

Observar y analizar fenómenos y problemáticas propias del campo ocupacional. Ejemplo: el proyecto que se realizará durante el curso.

**ACTIVIDADES**

Producir diagramas de clase a partir de problemas correspondientes a diversos dominios. Analizar y discutir sus propiedades y corrección. Representar situaciones determinadas utilizando diagramas UML u otras técnicas. Analizar y discutir sus características y defectos. Modelizar y especificar casos de usos a partir de descripciones de situaciones realistas. Documentar escenarios. Revisar documentos de requerimientos de software utilizando buenas prácticas para determinar su calidad. Realizar revisiones cruzadas de especificaciones. Utilizar las herramientas de soporte en todas las actividades que realicen.

**EVALUACIÓN**

Prácticas de laboratorio

Entrega de reporte de Tópicos investigados

Exposición de temas

Defensa de proyectos.

**BIBLIOGRAFÍA**

BORRERO, L. Tecnologías de la Información En Internet. Editorial Norma Colombia. 2003

PRESSMAN, R. S. Ingeniería del Softwuare – Un enfoque práctico. 7ma. Edición McGraw Hill. Interamericana Editores. 2010

WEITZENFELD, A. Ingeniería de software orientada a objetos. Cengage learning editores. México. 2005.

 Prof. Prog. Susana Isabel Arce