

**Planificación de la Asignatura:** Ingeniería de Software II

**Fecha:** 23/10/2024 13:02

**Código:** L1336

**Carrera:** Licenciatura en Bioinformática

**Departamento Académico:** Informática

**Docente a cargo:**

**Correo del docente a cargo:** victor.valotto@uner.edu.ar

**Régimen de Dictado:** Cuatrimestral 2º Cuatrimestre

**Carga Horaria Semanal:** 5 horas semanales

**Carga Horaria Total:** 70 horas

---

**Contenidos Mínimos:**

Arquitectura de Software. Diseño y Patrones. Modelado de Software, UML. Pruebas de Software. Tareas del proceso de Pruebas. Niveles y Técnicas. Aseguramiento y gestión de la calidad. Administración y control de Proyectos. Estimación de Proyectos de Software.

**Competencias Genéricas:**

CT2. Concepción, diseño y desarrollo de proyectos de la disciplina Bioinformática. Nivel de Dominio 2

CT3. Gestión, planificación, ejecución y control de proyectos de la disciplina Bioinformática. Nivel de Dominio 2

CT4. Utilización de técnicas y herramientas de aplicación en la disciplina Bioinformática. Nivel de Dominio 2

CS1. Fundamentos para el desempeño en equipos de trabajo. Nivel de Dominio 1.

CS2. Fundamentos para una comunicación efectiva. Nivel de Dominio 1.

CS3. Fundamentos para una actuación profesional ética y responsable. Nivel de Dominio 1.

CS4. Fundamentos para evaluar y actuar en relación con el impacto social de su actividad profesional en el contexto global y local. Nivel de Dominio 1.

CS5. Fundamentos para el aprendizaje continuo y autónomo. Nivel de Dominio 1.

CS6. Fundamentos para el desarrollo de una actitud profesional emprendedora. Nivel de Dominio 1.

**Competencias Específicas:**

CE 3 Desarrollar estudios en metodologías estadísticas, matemáticas y computacionales para analizar el genoma y la expresión génica. Dominio 2.

CE 4 Desarrollar estudios de modelización de los mecanismos de regulación de la expresión génica. Dominio 2.

CE 5 Generar estrategias para la modelización de epidemias que permitan analizar la evolución de las mismas en los diferentes espacios sociales tendientes a la elaboración de planes y proyectos que permitan elaborar políticas de salud destinadas a prevenir sus consecuencias sociales. Dominio 2.

CE 7 Participar en estudios de cambio global y pérdida de biodiversidad mediante la creación de modelos que incorporen variables para evaluar los posibles efectos de tales modificaciones. Dominio 2.

CE 10 Aportar soluciones mediante la realización de simulaciones en campos de neurociencia computacional y cristalografía macromolecular computacional. Dominio 2.

**Argumentación de aportes marcados en la matriz de competencias:****Competencias Genéricas Tecnológicas**

Concepción, diseño y desarrollo de proyectos de la disciplina Bioinformática: Esta competencia está directamente relacionada con la Ingeniería de Software, ya que abarca todo el ciclo de vida del desarrollo de software, desde la concepción y el diseño hasta la implementación. Los principios y prácticas de la Ingeniería de Software son fundamentales para el desarrollo eficaz de aplicaciones y sistemas bioinformáticos, asegurando que los proyectos sean escalables, mantenibles y cumplan con los requerimientos de los usuarios finales.

Gestión, planificación, ejecución y control de proyectos de la disciplina Bioinformática: La Ingeniería de Software aporta metodologías y herramientas para la gestión de proyectos, como metodologías ágiles, que son esenciales para la planificación, ejecución y control de proyectos de software. Estas habilidades son críticas para liderar proyectos de bioinformática, asegurando que se completen a tiempo, dentro del presupuesto y con la calidad esperada.

Utilización de técnicas y herramientas de aplicación en la disciplina Bioinformática: La Ingeniería de Software enseña sobre herramientas específicas de desarrollo (como entornos de desarrollo integrado, sistemas de control de versiones, herramientas de testing) y técnicas (como programación orientada a objetos, desarrollo dirigido por pruebas, integración continua) que son aplicables y a menudo necesarias en proyectos bioinformáticos. Esta competencia se relaciona con conocer y aplicar las herramientas adecuadas para el desarrollo y mantenimiento de software en bioinformática.

#### Competencias Genéricas Sociales, Políticas y Actitudinales

Fundamentos para el desempeño en equipos de trabajo: La materia enseña a los bioinformáticos a trabajar eficazmente en equipos que a menudo incluyen biólogos, matemáticos y científicos de la computación, desarrollando una colaboración interdisciplinaria efectiva.

Fundamentos para una comunicación efectiva: La comunicación es clave en la Bioinformática, donde se debe explicar conceptos complejos a un público diverso. La gestión de proyectos proporciona habilidades para articular problemas y soluciones tanto a especialistas como a no especialistas.

Fundamentos para una actuación profesional ética y responsable: La materia resalta la importancia de la ética en la gestión de datos biológicos, incluida la privacidad de los datos personales y la confidencialidad, preparando a los bioinformáticos para manejar datos sensibles con integridad.

Fundamentos para evaluar y actuar en relación con el impacto social de su actividad profesional en el contexto global y local: En Bioinformática, es fundamental entender el impacto de los avances tecnológicos en la sociedad. La gestión de proyectos enseña a los estudiantes a considerar las implicaciones sociales de sus proyectos y a buscar beneficios globales y locales.

Fundamentos para el aprendizaje continuo y autónomo: Dado que la Bioinformática es un campo de rápida evolución, la gestión de proyectos ayuda a los estudiantes a desarrollar habilidades para el aprendizaje

autónomo necesario para mantenerse al día con las nuevas tecnologías y métodos de análisis.

Fundamentos para el desarrollo de una actitud profesional emprendedora: La gestión de proyectos fomenta una mentalidad emprendedora, animando a los bioinformáticos a buscar oportunidades innovadoras y a llevar ideas del concepto a la realidad.

#### Competencias Específicas

Desarrollar estudios en metodologías estadísticas, matemáticas y computacionales para analizar el genoma y la expresión génica: Esta competencia se vincula directamente con la Ingeniería de Software, ya que el análisis de genomas y la expresión génica requieren de herramientas computacionales avanzadas, algoritmos específicos y software capaz de manejar grandes volúmenes de datos, todos aspectos en los cuales los principios de la Ingeniería de Software son fundamentales.

Desarrollar estudios de modelización de los mecanismos de regulación de la expresión génica: Similar a la anterior, esta competencia requiere del desarrollo y aplicación de software especializado para modelar procesos biológicos complejos, lo cual implica una sólida comprensión de la Ingeniería de Software para diseñar soluciones efectivas y eficientes.

Generar estrategias para la modelización de epidemias que permitan analizar la evolución de las mismas en los diferentes espacios sociales tendientes a la elaboración de planes y proyectos que permitan elaborar políticas de salud destinadas a prevenir sus consecuencias sociales: Aunque esta competencia es más amplia y multidisciplinaria, la parte de modelización de epidemias y análisis de su evolución implica el uso de software y modelos computacionales, donde la Ingeniería de Software juega un papel clave en el desarrollo de herramientas adecuadas para estos análisis.

Participar en el desarrollo y la implementación de la tecnología de GeneChips, expresión génica, mapeo, rastreo de polimorfismos, descubrimiento de genes y desarrollo de algoritmos diagnósticos: Esta competencia está directamente relacionada con la Ingeniería de Software ya que implica el desarrollo de algoritmos y software para el manejo de datos complejos y específicos de la bioinformática, como los generados por tecnologías de GeneChips y análisis de expresión génica.

Aportar soluciones mediante la realización de simulaciones en campos de neurociencia computacional y cristalografía macromolecular computacional: La simulación en estos campos requiere de software especializado que es capaz de realizar cálculos complejos y modelar sistemas biológicos, lo cual implica una

aplicación directa de los principios de la Ingeniería de Software para su desarrollo y optimización

---

**Correlativas Regulares para cursar:**

Ingeniería de Software I

**Correlativas Aprobadas para cursar:**

No posee

**Correlativas Aprobadas para promocionar o rendir el examen final:**

Primer año

Bases de Datos

Ingeniería de Software I

**Insercion de la Asignatura en el plan de Estudios:**

La presente materia constituye una segunda parte y complementa los contenidos relacionados con el Área de Conocimiento de la Ingeniería de Software, iniciada ya con la materia Ingeniería de Software I. Por lo tanto los conceptos fundamentales involucrados en la planificación de Ingeniería de Software I son aplicables a Ingeniería de Software II.

**Objetivo General:**

Transmitir al alumno la importancia de la especificación en el proceso de desarrollo de software. Estudio de la problemática que plantea el desarrollo de grandes sistemas de software, profundizar en los aspectos de especificación. Introducir al alumno en los formalismos y herramientas de especificación y especialmente en los de especificación del paradigma orientación a objetos.

- Dotar al alumno de conocimientos para que sea capaz de realizar especificaciones en modelos.
- Dotar al alumno de conocimientos sobre procesos de ingeniería del software haciendo hincapié en los métodos iterativos e incrementales. Mostrar el papel de las especificaciones en dichos procesos.
- Que el alumno logre comprender la importancia de las buenas prácticas y que toda construcción y mantenimiento de un producto de software con calidad debe ser soportado por procesos de ingeniería.

**Objetivos Particulares:**

- Describir las actividades técnicas e ingenieriles que se llevan a cabo en el ciclo de vida de un producto software.
- Describir los problemas, principios, métodos y tecnologías asociadas con la Ingeniería del Software.
- Introducir a la visión de arquitectura de software como base del proceso de desarrollo de los grandes sistemas.
- Comprender los fundamentos del diseño de sistemas software.
- Aplicar de forma práctica los conceptos teóricos sobre el desarrollo estructurado y orientado a objetos.
- Comprender las técnicas y métodos de planificación de proyectos de software.
- Comprender la importancia de las técnicas de estimación de software.
- Comprender el alcance de las actividades de aseguramiento de la calidad y administración de la configuración.
- Realizar un proyecto en grupo, aplicando los principios introducidos en la parte teórica de la asignatura.



**Programa Analítico:**

- **Módulo 1 - Diseño Arquitectural:** ¿Qué es la arquitectura de software? Decisiones de diseño. Estilos y modelos arquitectónicos. Modelo de Referencia y patrones de arquitectura. Sistemas distribuidos desde el punto de vista de arquitectura. Arquitectura de capas. Concepto de Servicios. Atributos de calidad. Especificación de los escenarios de calidad. Vista de arquitectura.
- **Módulo 2 - Diseño Detallado:** ¿Qué es el modelado? Fundamentos de UML. Elementos y diagramas. Paquetes, diagramas de actividad, diagramas de interacción (secuencia y colaboración), diagramas de clases, diagramas de transición de estados, diagramas de componentes y diagrama de despliegue.
- **Módulo 3 - Mantenimiento del Software:** Concepto de Cambio, impacto del cambio en la organización. Características y problemas. Políticas y administración del cambio. Proceso de Administración de cambios, roles y responsabilidades. Análisis de Impacto. Trazabilidad. Administración de requerimientos, seguimiento y control de cambios.
- **Módulo 4 - Testing de Software:** Conceptos. ¿Qué es testear software? Axioma del Testing. Taxonomías de Testing (Tipos de Testing). El proceso de testing. Planificación de Testing. Diseño de Caso de Testing. Testing de caja blanca y testing de caja negra. Estrategias de Testing: Particiones de Equivalencias y Valores Límites. Diseño de casos de pruebas funcionales. Axiomas del proceso de Testing.
- **Módulo 5 - Gestión de Proyectos:** Definiciones de Proyecto. Procesos de Administración de Proyectos. Organización de proyectos. Roles y Responsabilidades. Alcance. Ciclo de Vida. WBS. Programación de actividades. Diagrama de Gantt: tareas, asignaciones, hitos, holguras, camino crítico. Administración de Riesgos: Identificación del riesgo, análisis del riesgo, exposición del riesgo, mitigación y contingencia. Planificación de la comunicación. Plan de comunicación. Tipos y herramientas de comunicación. Control y seguimiento: Analizar datos del proyecto. Analizar variaciones. Cierre. Indicadores de gestión. Acciones adaptativas. Cierre del proyecto. Lecciones aprendidas.
- **Módulo 6 - Estimación de Proyectos:** ¿Porque estimar?. Métricas de Tamaño: Líneas de Código, Costeo del Proyecto, Puntos de Función, Puntos de Caso de Uso, StoryPoints. Métricas de esfuerzo y costo: COCOMO.
- **Módulo 7 - Gestión de la calidad :** Concepto de Calidad. Los hacedores de la Calidad: Juran, Deming y

Crosby. Total Quality Management. Problemas asociados a la calidad. Planificación de la Calidad, Aseguramiento de la Calidad, Control de la Calidad. Atributos de calidad. Cuantificando los atributos de Calidad. Costos de la Calidad. Roles y Responsabilidades del área de QA. Revisiones. Tipos de Revisiones. Revisiones Formales. Proceso. Administración de la Configuración. Problemas: en el desarrollo de software. Identificación de los elementos de configuración. Conceptos. Elementos de Configuración, Línea Base, Hitos y Entregables. Almacenado de los elementos de configuración. Gestión de Versiones, conceptos. Desarrollo concurrente. Recomendaciones. Gestión de entregas. Procedimientos. Tipos y clasificaciones de las entregas. Gestión de Entornos. Tipos de entornos.

**Metodología Didáctica:**

Las instancias para el desarrollo de la asignatura serán clases teóricas, prácticas. Para el desarrollo de cada una de las partes en que se divide la asignatura, se partirá de elementos simples sobre los que se estructurará el conocimiento.

Se continuará con el enfoque ya desarrollado en Ingeniería de Software I, donde se propone que los estudiantes trabajen en un esquema de autoestudio durante las horas asignadas a la cátedra.

Posteriormente, habrá una instancia final en cada módulo, correspondiente a una actividad de coloquio dirigida por el docente a cargo con el objetivo de mejorar conceptos, estimular el trabajo grupal, enfatizar el análisis crítico y participar en actividades basadas en juegos para la autoevaluación. Esta visión tiene una perspectiva de aula invertida.

Se intentará apuntar a un método de enseñanza que es tanto interactivo como reflexivo, adaptándose a la diversidad de estilos de aprendizaje de nuestros estudiantes. Cada semana, los estudiantes incursionarán en nuevos aspectos de la Ingeniería de Software a través de videos educativos breves y concisos, diseñados para introducir conceptos clave de manera efectiva. Estos videos serán el punto de partida para una exploración más profunda, complementados con una variedad de materiales adicionales como artículos relevantes, estudios de caso y recursos en línea.

Para garantizar una comprensión sólida y activa de estos conceptos, cada video estará acompañado de un cuestionario corto de múltiples opciones. Estos cuestionarios, diseñados para ser tanto desafiantes como informativos, servirán para reforzar el aprendizaje y proporcionarán una retroalimentación inmediata sobre la comprensión de los temas. Esta estrategia formativa permitirá a los estudiantes autoevaluar su progreso y a los docentes identificar y abordar áreas que puedan necesitar atención adicional.

Respecto a las actividades prácticas, éstas se compondrán de situaciones y/o escenarios que se pueden presentar en proyectos reales, donde en general lo que dispara son elementos de análisis y discusiones para las posibles resoluciones a los problemas.

**Formación Práctica:**

La realización de las actividades prácticas se centrarán en los módulos relacionados con el Producto de Software principalmente.

Con excepción del primer ejercicio, el resto está basado en un Caso de Estudio propuesto por la cátedra.

Los ejercicios apuntan a ir evolucionando en sus especificaciones de manera que los alumnos vayan proveyendo soluciones parciales en cada ejercicio. Estas soluciones son el producto de la realización de los ejercicios, los cuales están basados en la elaboración de modelos para cada una de las fases propuestas en los ejercicios, y serán presentadas por cada grupo o alumno.

El desarrollo de estas presentaciones serán realizadas con la asistencia de herramientas de software recomendadas por la cátedra, con recomendaciones del uso de herramientas gratuitas o de uso académico.

Estos trabajos tienen por objetivo generar discusiones y debates sobre el criterio de uso de las diferentes técnicas, métodos y herramientas ante diferentes situaciones.

Estos aspectos se desarrollarán durante los horarios de clases, para revisar los resultados y analizar los mismos, durante horas anexas que los alumnos utilizarán según la necesidad individual.

La principal fuente para la generación de los prácticos corresponden a los dos libros bases de la bibliografía.

**Listado de Actividades de Formación Práctica:**

- Ejercicio 1: Definición de Arquitectura:
- Ejercicio 2: Diseño detallado – Colaboración de Objetos.
- Ejercicio 3: Diseño detallado – Clases y Componentes del Sistema.
- Ejercicio 4: Diseño de Casos de Prueba.
- Ejercicio 5: Dimensionamiento del Sistema.

**Intensidad de la formación práctica**

Detalle de la carga horaria total prevista para cada una de las siguientes actividades:

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 1: 0 horas

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 2: 30 horas

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 3: 0 horas

Horas totales de actividades de formación práctica: 30 horas

**Metodología de Evaluación Durante el cursado:**

Se hará un seguimiento de los estudiantes a lo largo del dictado de la materia, a través los trabajos prácticos.

Para la evaluación de los conocimientos adquiridos los estudiantes deberán realizar una exposición sobre un contenido definido por la cátedra, seguido de preguntas relacionadas con dichos contenidos.

**Metodología de Evaluación en Exámenes Finales:**

Tanto alumnos libres como regulares serán evaluados de acuerdo a un examen escrito.

**Condiciones de Regularidad :**

Para la evaluación de los conocimientos adquiridos los estudiantes deberán realizar una exposición sobre un contenido definido por la cátedra, seguido de preguntas relacionadas con dichos contenidos.

Habrà una instancia de recuperación en caso que la evaluación de esta presentación sea inferior a 6(seis) aprobado. Si la evaluación es de 8(ocho) o superior la condición del estudiante es promocionado. En cualquier otro caso, es decir, entre 6 inclusive o menor a 8 la condición de regular.

**Cronograma de parciales durante el primer Cuatrimestre:**

---

**Cronograma de parciales durante el segundo Cuatrimestre:**

**Primer Examen Parcial:** 29 de Octubre de 2024

**Recuperatorio 01:** 05 de Noviembre de 2024



**Bibliografía Principal:**

Ingeniería del Software, un enfoque práctico. Roger Pressman. MCGRAW-HILL, 2005, Edición Número 6.

Ingeniería del Software, Ian Sommerville. PEARSON ADDISON WESLEY, 2005, Séptima edición.

**Bibliografía Complementaria:**

The Project Manager's Guide to Software Engineering Best Practices, Mark J. Christensen, Richard H. Thayer, IEEE Press, 2001.

The Software Engineering Book of Knowledge, IEEE, 2004.

Software Engineering, Editado por Merlin Dorfman y Richard H. Thayer, IEEE Press, 1997..

Software Engineering Project Management, Editado por Richard H. Thayer, IEEE Press, 2000.  
Segunda Edición.

Software Architecture in Practice, Ediciones 2, Len Bass, Paul Clements, Rick Kazman, Addison -Wesley, 2003.

Documenting Software Architecture, Paul Clemens, Felix Bachman, Addison –Wesley, 2002.

A Guide to Software Configuration Management. Alexis Leon, Artech House, 2000.

Configuration Management Principles and Practice. Jonassen Hass, Anne Mette , Addison -Wesley, 2002.

Software Configuration Management Patterns: Effective Teamwork, Practical Integration. Stephen Berzuck, Brad Appleton, , Addison -Wesley, 2002

Practical Guide to Software Quality Management, Second Edition - John W. Horch, 2003.

Metrics and Models in Software Quality Engineering, Second Edition, Stephen H.Kan, Addison Wesley , 2002

Patrones de Diseño. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Addison-Wesley, 2005.

Object Oriented Software Construction. Bertrand Meyer, Prentice Hall, 1997

UML y patrones.: 2da Edicion. Craig Larman. Prentice Hall, 2002.

El lenguaje unificado de modelado. Grady Booch, James Rumbaugh, Iva Jacobson, Prentice Hall , 1999.

**Equipo de Cátedra:**

Profesor Adjunto Dedicación Simple

Jefe de Trabajos Prácticos Dedicación Simple

Primer Cuatrimestre: Dictado de la materia Ingeniería de Software I

Asignación:

Profesor Adjunto: 10 hrs por semana

Jefe de Trabajos Prácticos: 6 hrs por semana

Segundo Cuatrimestre: Dictado de la materia Ingeniería de Software II

Asignación:

Profesor Adjunto: 10 hrs por semana

Jefe de Trabajos Prácticos: 6 hrs por semana

**Actividades de Investigación Gestión y Extensión:**

Sin Actividades

---

**Requisitos de admisión para alumnos oyentes:**

Sin Requisitos

---

**Infraestructura, equipamiento y recursos necesarios:**

Laboratorios de computación

**Otros:**

Sin información