

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

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

**Código:** L1332

**Carrera:** Licenciatura en Bioinformática

**Departamento Académico:** Informática

**Docente a cargo:**

**Correo del docente a cargo:** vvalotto@ingenieria.uner.edu.ar

**Régimen de Dictado:** Cuatrimestral 1º Cuatrimestre

**Carga Horaria Semanal:** 4 horas semanales

**Carga Horaria Total:** 56 horas

---

**Contenidos Mínimos:**

El proceso de software. Ciclos de Vida y Diseño del Software. Especificación y análisis de Requerimientos.  
Diseño Centrado en el Usuario. Gráficas por computadora, interfaces orientadas al usuario.

**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:**

Base de Datos

**Correlativas Aprobadas para cursar:**

No posee

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

Primer año

Algoritmos y Estructuras de Datos

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

Partiendo desde una posición muy general, son muy pocas las personas que dudan de la necesidad del software en la vida diaria de todos nosotros. Lo podemos ver cotidianamente. Más aún, los negocios sin la ayuda del software, entre otras cosas, probablemente no puedan alcanzar los objetivos. A pesar de esta expansión e inserción en los aspectos más simples de nuestra vida, como por ejemplo buscar donde comprar un libro o juguete, hasta mucho más complejos como el control de los equipos de medicina, sigue habiendo una idea o mejor dicho una “manera de ver” al software como un producto desarrollado por el hombre con mucho de arte y de inspiración.

Por otro lado, hay muchas evidencias por los estudios realizados y por los innumerables fracasos, que el software es una actividad cara y que en general siempre tiene los problemas de tener muchos defectos, no tener las funciones que se pensaban y que tener un software terminado a tiempo es una esperanza que alguna vez podrá ocurrir. Construir un software, no es hacer un programa. Son cuestiones bastante diferentes. Un software es un elemento económicamente productivo, tiene asociados conceptos de productividad y uso social, y debe ser visto como cualquier otro bien que es producido por el hombre, a pesar de ser (o parecer) intangible.

Ahora bien, ¿Qué producto creado por el hombre puede ser considerado exitoso y útil para que el resto de la sociedad se beneficie de él, si no puede ser producido de manera extendida y mantener las mismas funcionalidades y funcionamiento originales y/o mejoradas? Esto se logra con cierto grado de invariabilidad en la manera de hacerlo, con aceptación de los compradores y de los usuarios y con el entendimiento de que debe existir una determinada secuenciación en la “fabricación de un producto de software”.

Por lo tanto con esta visión, estamos en presencia de un producto de ingeniería, donde existen las etapas más básicas de un proceso de ingeniería: concepción y entendimiento, diseño, construcción, pruebas del producto y puesta en funcionamiento. En definitiva todas aquellas profesiones, y principalmente aquellas que se valen del software como ayuda para la elaboración y uso de sus productos, que poseen algún grado de interés o involucramiento deben comprender los alcances conceptuales de esta disciplina para potenciar la mejora en su concreción y así proveer mejores soluciones a la sociedad. Precisamente a la Bionformática podemos conceptualizarla como la construcción de herramientas adaptables para procesar, administrar, analizar y visualizar la información biológica, la que no solo requieren de un conocimiento establecido de la ciencia subyacente, sino también la capacidad de escribir programas eficientes, o mejor dicho Productos de Software.

Esta materia en conjunto con “Ingeniera de Software II” deben apuntar desde el inicio en el desarrollo de capacidades sobre los alumnos para que los futuros profesionales puedan enfrentar las diferentes situaciones laborales, ya sea de índole científica o comercial, con un enfoque en la generación de productos

de calidad basado en un proceso de desarrollo con calidad. Es por ello que el contenido y los objetivos están basados en conceptos y buenas prácticas adoptadas por la industria del software, en el que se incluyen principios y técnicas que ayudarán a los alumnos a adquirir un conjunto de herramientas del conocimiento que puedan proveer los resultados que se esperan.

El software, por todo lo enunciado en los primeros párrafos, tiene una implicancia económica suprema en nuestra sociedad, y más aún dentro de una profesión donde constituye junto con las ciencias biológicas los basamentos de su concepción. Por ello esta materia tiene dentro de cada uno de los contenidos, la esencia para que la construcción de un producto de software tenga un abordaje donde se puede ahorrar tiempo y esfuerzo haciendo lo correcto, en el momento adecuado y de la manera correcta.

Ambas materias “Ingeniería de Software I” e “Ingeniería de Software II” van más allá de la codificación para examinar el ciclo de vida del proyecto. Es una guía través del proceso de desarrollo de aplicaciones de principio a fin, sin ocuparse de una metodología en particular, paradigma de programación o plataforma tecnológica.

Ambos cursos se centrará en introducir las diferentes áreas de conocimiento que comprende la ingeniería de software, basado en el cuerpo de conocimiento de la ingeniería de software (SWEBOK) propuesto por la IEEE.

En particular el alcance de los contenidos de “Ingeniería de Software I” abarcará principalmente la introducción el procesos de desarrollo de un producto de software, y a la modelización de la necesidades de quien finalmente usará el producto de software como herramienta.

**Objetivo General:**

Presentar los conceptos fundamentales, junto con un enfoque de ingeniería a la hora de enfrentarse a una solución basada en software como herramienta para las actividades relacionadas con la bioinformática. Esta asignatura está centrada la propuesta de ofrecer al futuro profesional la visión de que el software es un producto que tiene un proceso de construcción que determina la calidad y el éxito de su utilidad posterior.

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.
- Presentar situaciones reales y donde la solución a problemas no son solo técnicas sino son principalmente de uso social.

**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.
- Presentar la importancia de los requisitos en el ciclo de vida del software.
- Introducir a las técnicas básicas de elicitación, documentación, especificación y prototipado de los requisitos de un sistema software.
- Demostrar la ventajas del uso de estándares en el modelado, y la necesidad de presentar diferentes visiones del mismo productos según las diferentes necesidades.
- Realizar un proyecto en grupo, aplicando los principios introducidos en la parte teórica de la asignatura.

**Programa Analítico:**

## Programa:

- Módulo 1. El producto de Software: ¿Qué es el software? Características. ¿Qué es Ingeniería de Software? Diferencias con otras disciplinas. Proceso de Software. Costos y Desafíos. Código de Ética. Problemática actual. Evidencia de la problemática.
- Módulo 2. La Ingeniería del Software: Conceptos: Programa, Producto, Sistema. Tipos de Sistemas. ¿Qué es ingeniería? Ingeniería de Sistemas y de Software. Proceso de Ingeniería. Áreas de Aplicación Cuerpo de Conocimiento de la Ingeniería de Software.
- Módulo 3. El Proceso del Software: Modelos de procesos: Cascada, Code-and-Fix. Evolutivo, Desarrollo en Espiral, Desarrollo Incremental, Basado en componentes. Áreas de proceso del Desarrollo de Software. RUP. Desarrollo Ágil: motivación y características. Selección del ciclo de vida.
- Módulo 4 – Requerimientos Funcionales: Proceso y producto. Comprender los fracasos del software. El problema de los malos requerimientos. Concepto de requerimientos. Definiciones, niveles y tipos de requerimientos. La ingeniería de requerimientos. El ciclo de requerimientos. El analista de requerimientos. Proceso de elicitación de requerimientos. Los stakeholders, los usuarios. Actividades de elicitación. Técnicas para entender el problema. Análisis de requerimientos. Técnicas de modelado del problema. Clasificación y priorización. Modelado con Casos de Uso. Actores, Casos de Uso, escenarios. Historias de Usuario. Especificación de requerimientos.
- Módulo 5 – Requerimientos No Funcionales. Atributos de Calidad. Restricciones. Los atributos de calidad y la arquitectura del software, aspectos transversales. Escenarios de Atributos de Calidad. Clasificación de los Atributos de Calidad. Priorización y conflictos. Sistemas en Tiempo Real y sistemas Críticos.
- Módulo 6 – Diseño Centrado en el Usuario. Interacción del Usuario y presentación de la Información. Proceso de Diseño de la Interfaz de Usuario. Técnicas de Análisis de usuario. Prototipado de la Interfaz. Aspectos transversales en el diseño de interfaz de usuario. Usabilidad. Estándares y Guías. Diseño de Interfaz de Usuario Web.

**Metodología Didáctica:**

Las instancias para el desarrollo de la asignatura se dividen en dos tipos de enfoques de trabajo con los estudiantes: un bloque de actividades con una visión teórica y otro bloque dirigido a actividades prácticas. Cada módulo de la asignatura comenzará con elementos simples sobre los cuales se estructurará el conocimiento.

En cuanto al enfoque teórico de los módulos I y II, se utilizará una orientación de clase expositiva. A partir del módulo III, 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 y aprovechando las ventajas de la modalidad híbrida. 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, estas se compondrán de situaciones y escenarios que se presentan en proyectos reales. Estas actividades estimularán el análisis y la discusión para encontrar posibles soluciones 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:**

Serán realizadas una cantidad de ejercicios por cada módulo, con una carga horaria para los alumnos estimada que comprende las horas de cátedra y horas individuales extra cátedra.

- Trabajo Practico 1: Procesos y Ciclos de Vida: 4 hrs.
- Trabajo Practico 2: Clasificación de Requerimientos: 2 hrs.
- Trabajo Práctico 3: Modelado con Casos de Uso: 6 hrs.
- Trabajo Práctico 4: Clasificación de los Atributos de Calidad: 2 hrs.
- Trabajo Práctico 5: Especificación de Requerimientos No Funcionales: 6 hrs.
- Trabajo Practico 6: Diseño de Interfaz de Usuario: 4 hrs.

**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 y coloquios.

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:**

**Primer Examen Parcial:** 04 de Junio de 2024

**Recuperatorio 01:** 11 de Junio de 2024

---

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

**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:**

Bibliografía de Consulta:

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.

Writing Better Requirements Alexander, Ian F., and Richard Stevens, Addison-Wesley, 2002.

Writing Effective Use Cases. Cockburn, Alistair. Boston, MA: Addison-Wesley, 2000.

Software Requirements, Second Edition Karl E. Wiegers. Microsoft Press, 2003.

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

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

User Interface Design – A Software Engineering Perspective. Soren Lauesen. Pearson – Addison Wesley

The Elements of User Experience, 2nd Edition – Jesse James Garret – New Riders – 2011

Designing Web Interfaces, Bill Scott & Theresa Neil, O Reilly, 2009

Don't Make Me Think! 2nd Edition. Steve Krug. New Riders, 2006.

**Equipo de Cátedra:**

Segundo Cuatrimestre: Dictado de la materia Ingeniería de Software I .

Profesor Adjunto Dedicación Simple

Jefe de Trabajos Prácticos Dedicación Simple

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