NOMBRE DE LA CARRERA	INGENIERÍA INFORMÁTICA EMPRESARIAL
NOMBRE DEL MÓDULO	INGENIERÍA DE SOFTWARE
NÚMERO DE CRÉDITOS (EXPRESADOS EN SCT-CHILE)	8 SCT – 216 Chile: Total horas de trabajo presencial: 144 Total horas de trabajo autónomo: 72
ÁREA DE CONOCIMIENTO	Ingeniería y Tecnología: Ingeniería de la Información
SEMESTRE	QUINTO y SEXTO

PREREQUISITOS	Análisis y Diseño Lógico de Sistemas
UNIDAD RESPONSABLE DE LA CONSTRUCCIÓN DEL SYLLABUS	ESCUELA DE INGENIERÍA INFORMÁTICA EMPRESARIAL
COMPETENCIAS DEL PERFIL DE EGRESO AL QUE CONTRIBUYE ESTE MÓDULO Y NIVEL DE LOGRO DE CADA UNA DE ELLAS.	1. Innovar en el ámbito de la gestión con apoyo de las Tecnologías de Información para mejorar la rentabilidad, eficiencia y productividad de las organizaciones. 2. Participar activamente en equipos de trabajo multidisciplinarios responsables de la innovación para mejorar la rentabilidad, eficiencia y productividad en las organizaciones.
APRENDIZAJES	Evalúa variables que definen un problema de tecnologías de información y gestión Captura adecuadamente requerimientos de información Acoge opiniones y comentarios respecto de su trabajo Expone temáticas de la disciplina ante diversas audiencias

<table>
<tr>
<td>UNIDADES DE APRENDIZAJES Y SABERES ESENCIALES</td>
<td>
2.1.Unidad I: Introducción: ¿Qué es un software y cómo se desarrolla?
<table>
<tr><th>SABER (conocimientos, recursos cognitivos)</th><th>SABER HACER (Procedimientos, recursos procedimentales)</th><th>SABER SER/CONVIVIR (Actitudes, recursos actitudinales)</th></tr>
<tr><td>Comprender los diferentes modelos de arquitectura de software orientados a concebir diseños de solución eficientes y reutilizables

Conocer (identificar) el ciclo de vida de desarrollo de un sistema de software y aplicar modelos de referencia en función del tipo de organización</td><td>Analizar sistemas de software utilizando lenguajes de modelamiento

Crear (diseñar) sistemas de software utilizando lenguajes de modelamiento

Aplicar los diferentes modelos de arquitectura de software orientados a concebir diseños de solución eficientes y reutilizables

Analizar (estimar) los esfuerzos requeridos para el desarrollo de un sistema de software y transformarlo en requerimientos de recursos</td><td></td></tr>
</table>
Unidad II: Análisis
<table>
<tr><th>SABER (conocimientos, recursos cognitivos)</th><th>SABER HACER (Procedimientos, recursos procedimentales)</th><th>SABER SER/CONVIVIR (Actitudes, recursos actitudinales)</th></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Conocer métodos para capturar requerimientos específicos a partir de entrevistas, reuniones, conversaciones</td><td>Capturar requerimientos de un sistema de información para estimar esfuerzos y costos

Acoger observaciones/dudas asociadas a los esfuerzos estimados</td><td>Participar de captura de requerimientos de un sistema de información para estimar esfuerzos y costos

Apreciar las observaciones o dudas asociadas a una propuesta planteada</td></tr>
</table>

Unidad III: Diseño lógico
<table>
<tr><td>SABER (conocimientos, recursos cognitivos)</td><td>SABER HACER (Procedimientos, recursos procedimentales)</td><td>SABER SER/CONVIVIR (Actitudes, recursos actitudinales)</td></tr>
<tr><td></td><td>Explicar los esfuerzos y costos estimados en distintos formatos (informes escritos, posters, presentaciones orales, videos, etc.)</td><td>Defender propuestas de solución en distintos formatos (informes escritos, posters, presentaciones orales, videos, etc.)</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>METODOLOGÍA A UTILIZAR</td><td>Clases Expositivas Participativas.

Las clases serán de carácter expositivas y participativas, en las que el docente hará una exposición inicial respecto del objetivo de la clase correspondiente y de los conceptos asociados, posteriormente los alumnos en grupos de trabajo</td></tr>
</table>

	constituidos por el docente, trabajarán en los avances de proyecto que el docente proponga y/o que emerjan de las vivencias de los propios alumnos. Laboratorios en Universidad. Las ayudantías se llevarán a cabo en los laboratorios de la carrera bajo la super-visión de los ayudantes, donde se aplicarán los conceptos y casos desarrollados en clase con el apoyo de herramientas de software. Evaluaciones. Se realizarán 4 pruebas teóricas, varias tareas y 6 avances de proyecto. Avances de proyecto. Se exigirán informes y exposiciones de avance de proyecto que desarrollen los alumnos, que permitan verificar el logro de las capacidades comprometidas en las distintas unidades de aprendizaje, donde el expositor será un miembro del grupo a elección del docente.
EVALUACIÓN DE APRENDIZAJES	Componentes y procedimientos evaluativos del módulo (ponderaciones). Evaluaciones. Prueba parcial 1: 15 %. Prueba parcial 2: 15 %. Prueba parcial 3: 15 %. Prueba parcial 4: 15 %. Promedio de tareas: 10 %. Promedio de avances de proyectos: 30 %.

	Donde: Las tareas se promedian y del resultado se extrea el 10 % para la nota final (cantidad de tareas será según el avance del curso). Los seis avances de proyectos se promedian y del resultado se extrae el 30 % para la nota final. Por lo tanto la nota final se calcula de la siguiente manera: Nota Final = $PP1 * 0,15 + PP2 * 0,15 + PP3 * 0,15 + PP4 * 0,15 + \bar{x}$ Tareas $* 0,1 +$ $\bar{x}$ Avances de proyectos. $* 0,3$ Se exige la aprobación del módulo con nota 4.0 o más en la nota final; en caso contrario (reprobación) existirá una instancia de recuperación (Acumulativa Opcional) que incluirá toda la materia del curso, donde la nota de presentación (nota final) es del 70 % y la prueba Acumulativa Opcional es del 30 %. Para esta prueba se exigirá nota mínima según reglamento del estudiante. También se puede faltar a una prueba durante el curso, donde la prueba de recuperación será la misma Acumulativa Opcional. Para las otras inasistencias se evaluará con la nota mínima.
REQUERIMIENTOS ESPECIALES	Sin requerimientos
BIBLIOGRAFÍA	Bibliografía Básica: 1. R. Pressman, Ingeniería del Software: Un enfoque práctico, vol. 6. McGraw-Hill, 2005. 2. Sommerville, Ingeniería del Software, vol. 7. Pearson Educación S.A., 2005.

	Bibliografía Complemetaria: 1. K. Kendall and J. Kendall, Análisis y diseño de sistemas., vol. 8. Pearson Educación S.A., 2011. 2. P. Deitel and H. Deitel, Cómo programar en java., vol. 7. Pearson Edu-cación, 2008. 3. C. Larman, UML y patrones: Introducción al análisis y diseño orientado a objetos. Pearson Educación, 1999. 4. M. Fowler and K. Scott, UML gota a gota. Pearson Educación, 2000. 5. J. Pavón Mestras, "El patrón modelo-vista-controlador (mvc)," 2008-2009. 6. R. Alarcón, Diseño orientado a objetos con UML. Grupo Eidos., 2000. 7. R. Pressman, Ingeniería del software un enfoque práctico., vol. 5. McGraw-Hill, 2001. 8. "Real time intelligence complex event processing." http://www.complexevents.com/. 166 A. Lozano Tello, Iniciación a la programación utilizando lenguajes vi-suales orientados a eventos. Ed.Bellisco Ediciones Técnicas y Científicas., 2001. 9. D. Fuller Padilla, "Apuntes de taller de ingeniería de software", 2003. 10. L. Guerrero, "Rational unified process.." 11. J. Palacio, Gestión de Proyectos Scrum Manager., vol. 2.5.1. Scrum Ma-nager., 2015.