PROGRAMA DE CURSO

Código	Nombre			
CC6401	Taller de metodologías ágiles de desarrollo de software			
Nombre en Inglés				
Workshop of Agile Software Development Methods				
SCT	Unidades Docentes	Horas de Cátedra	Horas Docencia Auxiliar	Horas de Trabajo Personal
6	10	3,5	0	6,5
Requisitos			Carácter del Curso	
CC4401			Taller de Investigación Aplicada	

Resultados de Aprendizaje

Los ingenieros de software estamos enfrentados a una actividad laboral eminentemente heurística, es decir, en donde todos los días se están resolviendo problemas distintos a los que se ha enfrentado anteriormente. De hecho, esto define a un buen ingeniero de software: el automatizar sistemáticamente procesos que son similares a la experiencia anterior.

Por ende un ingeniero de software está llamado a estar permanentemente navegando en aguas desconocidas, y su mejor herramienta para una productividad sostenible es el avanzar de manera sistemática creando colaborativamente (con sus clientes, usuarios, profesionales de especialidades complementarias y colegas) nuevo conocimiento a la vez que se generan nuevas soluciones informáticas. Esto es lo que da origen a la Agilidad.

La industria del software está dominada por un paradigma tradicional en donde se asimila el desarrollo de software a un trabajo repetitivo tradicional, sin tomar en cuenta la incertidumbre inherente a esta labor.

En este taller los alumnos y alumnas aprenderán a detectar los problemas que el paradigma tradicional ha generado en la industria del software, e investigando que soluciones metodológicas han sido desarrolladas al alero de las metodologías ágiles, además de desarrollar contenido (artículos, videos, etc.) donde se capture lo aprendido. Muchas de estas soluciones son de reciente aparición en la industria (no más de 1 o 2 años) por lo que se requiere la capacidad de entender este contenido en el idioma de origen, esencialmente inglés.

Posteriormente se realizarán proyectos los más auténticamente reales posible de desarrollo de software, incorporando prácticas de:

- Entendimiento Compartido y Diseño Colaborativo a través de Pensamiento Visual usando baja tecnología
- Gestión de Flujo de trabajo usando Kanban
- Resolución de problemas de negocio complejos de forma incremental usando Scrum
- Desarrollo incremental de software con calidad intrínseca usando Extreme Programming
- Validación de Innovaciones usando Lean Startup
- Y nuevas metodologías que ocupen otros espacios de trabajo de la ingeniería de software

Metodología Docente	Evaluación General

Diseño colaborativo de alto rendimiento mediante herramientas visuales de Baja tecnología
Para lograr una comunicación efectiva de los modelos mentales de cada alumno se usarán técnicas de Pensamiento Visual (Visual Thinking) usando técnicas del libro Gamestorming, las que han sido recopiladas desde las prácticas de las empresas más innovadoras del mundo, en particular las startups de Silicon Valley.
 	
Investigación de problemas de la industria del software
Los alumnos deberán conocer de la forma más directa posible la realidad de la industria de software local (y si es posible mundial), detectar los desafíos más relevantes y explorar las soluciones que para dichos desafíos han surgido de la Agilidad.
 	
Desarrollo de proyectos auténticos de desarrollo de software
A partir de las motivaciones de desarrollo profesional los alumnos irán desarrollando de forma individual o en equipo pequeños experimentos prototipos de sus Propuestas Únicas de Valor (sean estas de emprendimiento o innovación) que serán retroalimentadas por el equipo docente y validadas en el mercado mediante experimentos exposición temprana a la industria.
 	
Mentoring mediante Aprendizaje Cognitivo
El modelo a aplicar será de Aprendizaje Cognitivo, en donde los docentes modelan previamente las destrezas a los alumnos, para luego acompañar su desarrollo (coaching) hasta que el alumno posea autonomía en la ejecución de la habilidad.
 	
Trabajo en equipo
En general se replicará el modelo usado en las startups actuales de “co-work”, en donde proyectos distintos comparten un espacio físico y de servicios común y comparten sus experiencias generando sinergias de aprendizaje y de habilidades.
	• Proceso (30%)
	  ○ reflexión y auto evaluación de como en este proyecto se pudo o no aplicar los principios ágiles que se han descubierto en el taller
	• Producto (40%)
	  ○ Periódicamente los alumnos deberán presentar sus avances específicos en la evolución del producto, los que serán evaluados y retroalimentados por el equipo docente
	• Trabajo en Equipo
	  ○ Coevaluación entre los miembros del grupo en relación a los aportes de cada integrante con respecto a los aportes:
	  ▪ Entendimiento del Problema de Negocio a resolver
	  ▪ Trabajo en Equipo
	  ▪ Aporte Técnico

Unidades Temáticas

Número	Nombre de la Unidad	Duración en Semanas
1	Introducción Teórica	1

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Introducción principios básicos de Pensamiento Visual para el Entendimiento Compartido, para comunicarse efectivamente • Inicio de exploración de la realidad de la industria del software	• Trabajo en equipo: capacidad de desarrollar de forma efectiva diseños colaborativos usando Pensamiento Visual • Plan de exploración de la industria del software	Ver bibliografía

Número	Nombre de la Unidad	Duración en Semanas
2	Exploración de desafíos en la industria y soluciones	6

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Definición de iniciativa a realizar • Delineamiento de primer producto mínimo viable incorporando los nuevos enfoques estudiados por loa alumnos • Estudio de casos relevantes del ecosistema emprendedor internacional (principalmente EEUU y Europa) • Desarrollo de presentación de la idea en forma concisa (“ëlevator pitch”)	• Conocimiento de primera mano acerca de las implicaciones de aplicar en la práctica los enfoques estudiados • Comunicación Oral, Lectura/Escritura en Inglés: capacidad de presentar la idea de negocio de forma clara y cautivadora, en castellano e inglés • Trabajo en Equipo: capacidad de diseñar en forma colaborativa planes incrementales de exploración de ideas	• Ver bibliografía

Número	Nombre de la Unidad	Duración en Semanas
3	Desarrollo de Proyecto de Software	8

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Gestión de Flujo de trabajo usando Kanban • Resolución de problemas de negocio complejos de forma incremental usando Scrum • Desarrollo incremental de software con calidad intrínseca usando Extreme Programming • Validación de Innovaciones usando Lean Startup • Y nuevas metodologías que ocupen otros espacios de trabajo de la ingeniería de software	• Aprendizaje auténtico: Conocimiento de primera mano acerca de las implicaciones de aplicar en la práctica los enfoques ágiles en el desarrollo de software	• Ver bibliografía

Bibliografía

Comunicación y diseño colaborativo de alto rendimiento mediante Pensamiento Visual para el Entendimiento Compartido	• Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers
	  ○ Dave Gray, Sunni Brown, James Macanufo
	• Visual Meetings: How Graphics, Sticky Notes and Idea Mapping Can Transform Group Productivity
	  ○ David Sibbet
	• Innovation Games: Creating Breakthrough Products Through Collaborative Play
	  ○ by Luke Hohmann
	• Rapid Problem Solving with Post-It Notes
	  ○ David Straker

Principios Ágiles	• Lean Software Development: An Agile Toolkit
	  ○ Mary y Tom Poppendieck
	• Kanban: Successful Evolutionary Change for Your Technology Business
	  ○ David J. Anderson
	• Personal Kanban: Mapping Work | Navigating Life
	  ○ Jim benson, Toninanne de Maria
	• The People's Scrum: Agile Ideas for Revolutionary Transformation
	  ○ Tobias Mayer

Bibliografía	

Desarrollo Ágil	• Extreme Programming Explained
	  ○ Kent Beck
	• Clean Code: A Handbook of Agile Software Craftsmanship [Paperback]
	  ○ Robert C. Martin
	• The Clean Coder: A Code of Conduct for Professional Programmers
	  ○ Robert C. Martin
	• Growing Object-Oriented Software, Guided by Tests
	  ○ Steve Freeman, Nat Pryce
	• Continuous delivery and DevOps: A Quickstart Guide
	  ○ Paul Swartout
	• Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing
	  ○ Gojko Adzic

Vigencia desde:	Marzo de 2009
Elaborado por:	Agustín Villena Moya