METODOS AGILES PARA APLICACIONES WEB

Año 2018

Carrera/ Plan:
Analista en TIC Plan 2017
Licenciatura en Informática Plan 2015
Licenciatura en Sistemas Plan 2015
Licenciatura en Informática Plan 2003-07/Plan 2012
Licenciatura en Sistemas Plan 2003-07/Plan 2012

Área: Ingeniería de Software, Bases de Datos y Sistemas de Información
Año: 3º año
Régimen de Cursada: Semestral
Carácter: Optativa
Correlativas:
Orientación a Objetos 1
Ingeniería de Software 1
Profesores: *Alejandra Garrido*
Hs semanales: *6hs*

FUNDAMENTACIÓN

Esta materia les permite a los alumnos profundizar los conocimientos adquiridos en el Paradigma de Orientación a Objetos y en conceptos básicos de Ingeniería de Software en el contexto de la construcción de aplicaciones complejas como es el caso de las aplicaciones web con procesos de negocio.

En particular abarcaremos el desarrollo y el mantenimiento de aplicaciones web con metodologías ágiles, enfatizando los valores y principios de las mismas y ejercitando sus prácticas. También cubriremos conceptos y técnicas relacionadas al desarrollo ágil que han surgido con énfasis en los últimos años como el concepto de deuda técnica y las técnicas de visual management.

Nos enfocaremos además en el desarrollo de software de calidad, tanto calidad interna del código para permitir su mantenibilidad y reuso, como calidad externa del software como son la usabilidad y accesibilidad, cubriendo tanto requisitos funcionales como no funcionales de las aplicaciones web. Dentro de los métodos y técnicas que se abordarán para permitir construir y mantener la calidad del software se cubrirán temas de testing de distintos niveles y refactoring.

A través de esta materia los alumnos se expondrán al desarrollando un sistema real en un grupo de trabajo, y utilizando una metodología ágil muy actual en el ámbito profesional.

Objetivos Generales:

- Presentar a los alumnos la problemática de construir y mantener una aplicación compleja como es el caso de las aplicaciones web con procesos de negocio, que respeten criterios de calidad interna y externa.
- Profundizar los conceptos de diseño y construcción de una aplicación cuando la complejidad se incrementa, como es el caso de las aplicaciones web, que conllevan una arquitectura de software particular.
- Presentar los valores, principios y prácticas de las metodologías ágiles para la construcción de aplicaciones complejas en el contexto de un equipo de desarrollo, así

como los principios y prácticas de métodos más específicos surgidos del movimiento ágil, como son “lean software development” y “visual management” de proyectos ágiles.
- Exponer el concepto de “technical debt” en el mantenimiento de software y presentar técnicas específicas para su abordaje a través de testing y refactoring. Profundizar estas técnicas no solo para la mejora de atributos de calidad interna, sino también para aquellos de calidad externa como son la usabilidad y accesibilidad.

CONTENIDOS MINIMOS

- Arquitecturas de software. Arquitecturas web.
- Valores, principios y prácticas de las metodologías ágiles. Métodos ágiles: XP y Scrum.
- Origen de Lean software development. Principios y prácticas Lean.
- Principios y herramientas de Visual Management en proyectos ágiles.
- Mantenimiento de software. Introducción al concepto de Technical Debt o Deuda Técnica. Tipos de Deuda Técnica. Abordaje de la Deuda Técnica.
- Testing. Tipos de tests: de aceptación, de unidad, de integración. Integración continua. Refactoring. Concepto de bad smell.
- Usabilidad en las aplicaciones web. Accesibilidad.
- Refactoring como técnica para mejorar la usabilidad de aplicaciones web.

PROGRAMA ANALÍTICO

I - Arquitecturas de software

1. Definición de arquitectura de software. Diferentes abordajes.
2. Arquitecturas de software más reconocidas: “pipes and filters”, “layered architectures”, “event-based”, “model-view-controller”.
3. Patrones arquitecturales. Blackboard system. Broker pattern. Microservices.
4. Arquitecturas web. Model-View-Controller en aplicaciones web.

II - Metodologías ágiles de desarrollo y prácticas relacionadas

5. Metodologías Agiles. Filosofía. Características fundamentales. Diferencias con las metodologías tradicionales de desarrollo de software.
6. Extreme Programming. Valores, principios y prácticas.
7. Scrum. Roles, reuniones y artefactos.
8. Lean software development. Principios y prácticas Lean. Visual management. Herramientas de visual management más reconocidas. Trello.

9. Métodos de desarrollo web que intentan integrarse con prácticas agiles. User Centered Design. Mockup Driven Development

III - Mantenimiento de software con métodos ágiles

10. Etapa de mantenimiento de software. Costo del mantenimiento
11. Technical debt o Deuda Técnica. Tipos de Deuda Técnica. Abordaje de la Deuda Técnica
12. Refactoring. Características. Proceso de refactoring. El concepto de bad smell. Límites en la preservación del comportamiento. Herramientas. Refactoring en distintos lenguajes y paradigmas de programación. Refactoring de modelos y de otros artefactos de software. Refactoring de arquitecturas. Refactoring to patterns. Refactoring para la mejora de atributos de calidad internos (flexibilidad, mantenibilidad).
13. Test Driven Development. Características. Tipos de Testing: de unidad, de integración, de aceptación, de regresión. Características de cada uno. Estrategias de testing de unidad. Framework de testing de unidad: familia XUnit.

IV - Usabilidad en aplicaciones web

14. Definición de Usabilidad. Factores de la usabilidad. Las 10 heurísticas de Nielsen. Métodos de evaluación de la usabilidad. Tests de usuario.
15. Accesibilidad en las aplicaciones web. Guidelines de W3C.
16. Refactoring como técnica para mejorar la usabilidad de aplicaciones web. Malos olores de usabilidad o o"usability smells". Identificación automática de usability smell a partir de eventos de interacción.

METODOLOGÍA DE ENSEÑANZA

Los conceptos teóricos son presentados en clases teóricas por el profesor, y se les ofrecerá material extra a los alumnos que permita profundizar los temas durante las clases teóricas, a partir de distintos artículos que disparen la inquietud de los alumnos y que motiven la participación en clase.

Durante las clases prácticas los alumnos irán desarrollando una aplicación web en grupo, que constará de diferentes instancias de entrega. El proyecto que los alumnos deben desarrollar profundizará la tecnología de orientación a objetos, utilizando frameworks orientados a objetos y patrones de diseño, y deberá demostrar su calidad interna y externa a partir de un conjunto de tests de aceptación y tests de usuario.

EVALUACIÓN

Para **aprobar la cursada** de la materia se requiere aprobar las distintas instancias de entrega del proyecto donde se desarrollará una aplicación web en forma grupal.

Para **aprobar la materia** se requiere aprobar un examen sobre los temas vistos en las clases teóricas.

BIBLIOGRAFÍA OBLIGATORIA

- Software Architecture: Perspectives on an Emerging Discipline. Mary Shaw, David Garlan. Pearson. 1996.
- Pattern Oriented Software Architecture. A system of patterns. Buschmann, Meunier, Rohnert, Sommerlad and Stal. Wiley. 1996.
- Extreme Programming Explained. Kent Beck. Addison Wesley. 2005.
- "The evaluation of accessibility, usability and user experience ". Helen Petrie and Nigel Bevan. The Universal Access Handbook, C Stepanidis (ed), CRC Press, 2009.
- "Lean Software Development". Dasari. Ravi Kumar. The PROJECT PERFECT White Paper Collection.
- "Visual Management – A General Overview". Algan Tezel, Lauri Koskela, Patricia Tzortzopoulos. 5th Int. Conference on Construction in the 21st Century (CITC-V). 2009.
- Refactoring: lmproving the Design of Existing Code. Fowler, Martin. Addison-Wesley, 1999.
- Simple Smalltalk Testing: With Patterns. Kent Beck. http://www.xprogramming.com/testfram.htm
- SUnit Explained. Stephane Ducasse. http://www.iam.unibe.ch/~ducasse/

BIBLIOGRAFÍA COMPLEMENTARIA

- Essential Scrum: A Practical Guide to the Most Popular Agile Process. Kenneth Rubin. Addison-Wesley. 2012.
- Design Patterns. Elements of Reusable Objects Oriented Software. Garnma, Helm, Johnson, Vlissides, Addison-Wesley, Professional Computing Series.
- Refactoring to Patterns. Joshua Kerievsky. Addison Wesley, 2004.
- Software Testing. A Craftsman's approach, 4th ed. Paul Jorgensen. CRC Press, 2013.
- "Seaside: A Flexible Environment for Building Dynamic Web Applications –". Ducasse, Lienhard and Renggli. IEEE Software 2007.
- "Service Oriented Architecture". Perrey and Lycett. Proceedings of the Symposium on Applications and the Internet Workshops, 2003.

- "Test-Driven Web Application Development in Java". Pipka. Lecture Notes in Computer Science, 2003, Volume 2591/2003, 378-393.

CRONOGRAMA DE CLASES Y EVALUACIONES

Inicio de clases estimado: Semana del 27/8

Clase	Contenidos/Actividades	Evaluaciones previstas
1	Arquitecturas de software. Abordajes. Principales arquitecturas de software	
2	Patrones arquitecturales. Arquitecturas web: MVC.	Presentación del proyecto seleccionado
3	Metodologías ágiles. XP y Scrum.	
4	Lean software development. Visual management	Presentación del “product backlog”
5	User Centered Design. Integración de UCD con métodos agiles. Mockup-Driven Development.	Planificación en Trello de los sprints de desarrollo
6	Mantenimiento de software. Concepto de deuda técnica. Introducción a Refactoring y Testing	
7	Refactoring de código. Code smells. Herramientas de refactoring. Refactoring para aplicaciones web.	Presentación de 1er release del proyecto
8	Testing. Test Driven Development. Framework XUnit. Testing de aplicaciones web	
9	Usabilidad. Métodos de evaluación de la usabilidad. User testing. Split testing.	Presentación de 2do release del proyecto
10	Refactoring para mejorar usabilidad. Usability smells. Accessibility smells.	Presentación de tests de usuario para el primer y segundo release
11	Accesibilidad. W3C Guidelines: WCAG	
12	Identificación de usability smells a partir de eventos de interacción	Presentación de distintas versiones del producto desarrollado para Split testing
13	Herramientas para la integración práctica de la evaluación y mejora de la usabilidad en el desarrollo web con métodos ágiles	
14	Integración final de temas	Presentación del producto refactorizado que demuestre la mejora en usabilidad

Contacto de la cátedra (mail, página, plataforma virtual de gestión de cursos):

- Docentes:
  - Dra. Alejandra Garrido: garrido@lifia.info.unlp.edu.ar

Firmas del/los profesores responsables:

Alejandra Garrido