FACULTAD DE INFORMATICA

UNIVERSIDAD NACIONAL DE LA PLATA

Tópicos de Ingeniería de Software II

Carrera: **Maestría en Ingeniería de Software**

Año: **2021**
Duración: 108 hs. totales
Profesor a cargo: ***Matías Urbieta***
Docentes: ***Andres Diaz Pace, Julián Grigera, Claudia Pons***

OBJETIVOS GENERALES:

Brindar a los asistentes técnicas y conceptos avanzados del desarrollo de Software, que permiten construir software robusto, mantenible, extensible y reutilizable. Se abordará el diseño de software desde una perspectiva arquitectural y de atributos de calidad, con énfasis en la reutilización de software a partir del concepto de frameworks. Se abordarán patrones de arquitectura, con un foco en el ámbito de aplicaciones Web. El diseño e implementación de aplicaciones interactivas será abordada a través de técnicas de maquetado y diseño de experiencia de usuario. Por otro lado se introducirán conceptos de inteligencia artificial y cómo puede aplicarse al campo de la ingeniería de software.

CONTENIDOS MÍNIMOS:

- Nociones de arquitectura de software y atributos de calidad
- Conceptos avanzados del diseño orientado a objetos y frameworks de desarrollo.
- Arquitecturas web y diseño de aplicaciones Web.
- Usabilidad, experiencia del usuario y estrategias de testing y mejora incremental
- Introducción a la Inteligencia Artificial, y su impacto sobre la Ingeniería de Software.

PROGRAMA

Arquitectura y Atributos de Calidad

- Definición de arquitectura. Rol en el desarrollo de software.
- Atributos de calidad como conductores del diseño arquitectónico.
- Estilos arquitectónicos más comunes.
- Captura de decisiones arquitectónicas.
- Vistas de módulos, componentes y conectores, y despliegue.

Conceptos de frameworks

- Tipos de frameworks. Características.
- Principio de inversión de control. Hotspots.
- Composición vs. Herencia en frameworks. Ejemplos.
- Relación con arquitecturas de referencia.

Arquitecturas Web

- Frameworks tradicionales multicapa. Model-View-Controller. Principios.
- Frameworks de aplicaciones Single-Page.
- Arquitecturas basadas en Microservicios.
- Nociones de infraestructura para puesta en producción.
- Contenedores y virtualización.

Usabilidad y UX en el proceso de desarrollo

- Relevamiento de requerimientos de interacción y experiencia de usuario (UX).
- Herramientas de maquetado y prototipado.
- Requerimientos en el contexto de metodologías ágiles.
- Patrones y buenas prácticas de diseño UX

Tests de usabilidad

- Pruebas de usuario remotas y análisis de resultados.
- Captura y reparación automatizadas de problemas de usabilidad web.
- Testing multivariado y A/B testing.

Agile + UX

- Técnicas de relevamiento y especificación orientadas a las tareas.
- Requerimientos de interacción temprano e iterativo en el ciclo de desarrollo ágil.
- Proceso de mejora incremental de la experiencia del usuario durante un desarrollo ágil, basado en las técnicas de refactoring y testing.

Introducción a la Inteligencia Artificial.

- Fundamentos. IA simbólica vs. IA no simbólica.
- IA no simbólica: Aprendizaje automático o machine learning. Aprendizaje supervisado vs. Aprendizaje no supervisado. Árboles de decisión. Deep Learning (Redes Neuronales Artificiales). Tipos de redes neuronales.
- Aplicaciones de la Inteligencia Artificial en el campo de la Ingeniería de Software.

ACTIVIDADES EXPERIMENTALES Y DE INVESTIGACION

Actividades prácticas:

Desarrollo de trabajos prácticos parciales luego de cada eje temático de la materia. Estos trabajos están pensados para ser iniciados en clases prácticas estilo taller, apuntando a una puesta en común, que luego continuará individualmente cada alumno.

Actividad 1: luego del dictado de los primeros temas, se trabaja sobre un caso de estudio para que los alumnos puedan realizar un análisis de sus principales atributos de calidad y un (re-)diseño arquitectural del mismo, para luego implementarlo sobre alguna de las tecnologías a utilizar en el curso (por ej., Java). Además, se trabajará sobre la generación de diagramas de arquitectura basados en UML.

Actividad 2: al finalizar con los contenidos de UX + Usabilidad en el ciclo de desarrollo, se trabajará sobre un prototipo de aplicación desde el punto de vista del diseño de interacción. Los alumnos crearán un prototipo con las funcionalidades principales de la aplicación, empezando por el estudio de los potenciales usuarios, pasando por prototipos de baja fidelidad, ajustándolo en al menos dos iteraciones de pruebas de usuario.

Actividad 3: mediante esta actividad el alumno pondrá en práctica los conceptos teóricos aprendidos acerca de cómo diseñar e implementar un agente inteligente utilizando Redes Neuronales Artificiales RNAs a través de un framework.

El alumno logrará comprender el concepto de RNA con sus diferentes configuraciones de parámetros e hiperparámetros, incluyendo hiperparámetros relacionados con la estructura de la RNA e hiperparámetros relacionados con el entrenamiento. El alumno también será capaz de evaluar la calidad de la RNA para realizar su tarea.

Trabajo Final: el trabajo final consiste en la elección por parte del alumno de alguna de las temáticas abordadas durante las 3 actividades prácticas intermedias para ser ampliada. El desarrollo de este trabajo involucra el estudio de la literatura actual sumado a la extensión

del trabajo de la actividad intermedia, con un grado medio de dificultad. Tiene un plazo máximo de entrega de 3 meses luego de terminada la cursada.

Investigación:

Los alumnos analizarán artículos de investigación de la bibliografía complementaria con el objetivo de profundizar en los temas abordados. Eventualmente, los alumnos podrán plantear mejoras o enfoques alternativos para las problemáticas discutidas en los artículos.

METODOLOGIA DE EVALUACION

La evaluación de la materia es a través de trabajos prácticos parciales y de un trabajo práctico final integrador. La calificación final del alumno es en base al promedio de los mismos. Los trabajos prácticos parciales son de pequeña envergadura y tienen una función de poner rápidamente en práctica los conocimientos que se van abordando. Estos son realizados y aprobados durante la cursada de la materia.

El trabajo final integrador está basado en la elección por parte del alumno de alguna de las temáticas abordadas durante el curso y el estudio de la literatura actual; puede consistir en una extensión de alguno de los trabajos prácticos parciales y tiene un plazo máximo de entrega de 3 meses luego de la última clase. Los trabajos prácticos se realizan usando el lenguaje de modelado y diferentes lenguajes de programación para su implementación, tales como JavaScript, Smalltalk, Java, Python, etc. qué son los más apropiados de acuerdo con estos objetivos de aprendizaje/actualización.

COMPETENCIAS A DESARROLLAR EN RELACION CON EL OBJETIVO DE LA CARRERA

C.1- Manejar y aplicar tecnologías actuales para el desarrollo de sistemas de software, incluyendo métodos, lenguajes, arquitecturas, frameworks y herramientas.

C.3- Gestionar, planificar y controlar proyectos de software de distinta envergadura.

C.4- Definir parámetros de calidad tanto interna como externa de un producto software, y establecer procesos de evaluación y mejora que atiendan la satisfacción de todos los involucrados (el cliente, los usuarios y su experiencia, y el equipo de desarrollo).

C.6- Tener capacidad de analizar el estado del arte en los distintos aspectos de la ingeniería de software, así como producir conocimiento científico en el área.

BIBLIOGRAFÍA BÁSICA

- Cervantes, H., Kazman, R. Designing Software Architectures: A Practical Approach. Addison-Wesley. 2016
- Fayad, M. E., Schmidt, D. C., & Johnson, R. E. Implementing application frameworks: object-oriented frameworks at work. John Wiley & Sons, Inc..
- Fayad, M. E., Schmidt, D. C., & Johnson, R. E. (1999). Building application frameworks: object-oriented foundations of framework design. John Wiley & Sons, Inc.
- Antani, V., & Stefanov, S. Object-Oriented JavaScript. Packt Publishing Ltd. 2017.
- Design Patterns: Elements of Reusable Object-Oriented Software. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Addison Wesley, 1994.
- S. Russell y P. Norvig. Artificial Intelligence. A Modern Approach.Prentice Hall, 3ra edición. 2010.
- Nielsen, Jakob. Designing web usability: The practice of simplicity. New riders publishing, 1999.
- Krug, Steve. Don't make me think!: Web & Mobile Usability: Das intuitive Web. MITP-Verlags GmbH & Co. KG, 2018.

BIBLIOGRAFÍA COMPLEMENTARIA

- Firmenich S., Bosetti G., Rossi G., Winckler M., Barbieri T. (2016) Abstracting and Structuring Web Contents for Supporting Personal Web Experiences. In: Bozzon A., Cudre-Maroux P., Pautasso C. (eds) Web Engineering. ICWE 2016. Lecture Notes in Computer Science, vol 9671. Springer,
- Grigera J., Garrido, A., Rivero, J.M., Rossi, G. (2017). Automatic detection of usability smells in web applications. Int. J. Hum. Comput. Stud. 97: 129-148 (2017)
- Firmenich S., Garrido A., Grigera J., Rivero M., Rossi G. Usability improvement through A/B testing and refactoring. Software Quality Journal 27(1): 203-240 (2019)
- Bouvin, N. O. (2019, September). From NoteCards to Notebooks: There and Back Again. In Proceedings of the 30th ACM Conference on Hypertext and Social Media (pp. 19-28).
- Martin Fowler: Domain-Specific Languages. The Addison-Wesley signature series, Addison-Wesley 2011, ISBN 978-0-321-71294-3, pp. I-XXVIII, 1-597
- Wolfgang Ertel:Introduction to Artificial Intelligence.Springer. ISBN 978-3-319-58486-7
- Matias Urbieta, Nahime Torres, José Matías Rivero, Gustavo Rossi, Francisco José Domínguez Mayo: Improving Mockup-Based Requirement Specification with End-User Annotations. XP 2018: 19-34
- Angular 6 by Example: Get Up and Running with Angular by Building Modern Real-world Web Apps, 3rd Edition Kevin D. Hennessy
- Designing with Data: Improving the User Experience with A/B Testing. Rochelle King, Elizabeth Churchill, Caitlin Tan. O'Reilly Media.2017
- Krug, Steve. Rocket surgery made easy: The do-it-yourself guide to finding and fixing usability problems. New Riders, 2009.
- Goodfellow, Ian; Bengio, Yoshua y Courville, Aaron (2016). Deeplearning. MIT Press.
- Keras: the Python deep learning API https://keras.io/
- Shai Shalev-Shwartz y Shai Ben-David. (Ed.). (2014).Understanding Machine Learning: From Theory to Algorithms. New York. UnitedState: Cambridge University Press.

- Exploring architecture blueprints for prioritizing critical code anomalies: Experiences and tool support. E Guimaraes, S Vidal, A Garcia, JA Diaz Pace, C Marcos. Software: Practice and Experience, 48 (5), 1077-1106. Wiley. (2018)
- High-Level Design Stories in Architecture-Centric Agile Development. J. A Diaz-Pace, A. J. Bianchi. 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), 137-144, IEEE. (2019)
- Assisting requirements analysts to find latent concerns with REAssistant. A Rago, C Marcos, J. A. Diaz-Pace. Automated Software Engineering, 23 (2), 219-252. Springer. (2016)