**Formato Syllabus**

|  |  |
| --- | --- |
| **Nombre Corto** | Teoría de la computación |
| **Nombre Largo** | Teoría de la computación |
| **Descripción** | Introducción a máquinas de estado, autómatas y redes de Petri. Introducción a la teoría de computabilidad y paradigmas de programación. Este último tiene un fuerte componente práctico |
| **Condiciones** | Programación avanzada Lógica y Matemáticas Discretas |

Objetivos de Formación

|  |
| --- |
| Proveer a los estudiantes los conocimientos básicos que serán aplicados en cursos posteriores de la carrera:   * Programación reactiva/basada en eventos, la cual se aplica en implementación de interfaces gráficas * Programación orientada a aspectos y mixins, los cuales se aplican en el desarrollo de servicios web * Programación de flujo de datos y funcional, que se utiliza para procesamiento de datos * Máquinas de estado, que se usa en diseño de ciertos tipos software * Expresiones regulares, que se usa en muchos problemas de reconocimiento de strings con cierta estructura * Redes de Petri, que son el modelo de ejecución de BPMN * Teoría de la computabilidad, que es la base a todo lo anterior |

Contenidos Temáticos

|  |
| --- |
| 1. Máquinas de estado y aplicaciones 2. Teoría de la computabilidad 3. Paradigmas de programación |

Resultados Disciplinares Esperados a nivel de programa, relacionados con la asignatura

|  |
| --- |
| Explicar la teoría de los autómatas (1)  Entender las máquinas de Turing y Von Neuman (2)  Explicar los conceptos y tecnologías en los que se basan los diferentes lenguajes de programación (2)  Escoger una estrategia de solución a un problema algorítmico. (1)  Decidir Cuando Un Problema Es Soluble Con Un Algoritmo. (1)  Capacidad Para Aprender Autónomamente Y Aplicar Los Diferentes Modelos (Paradigmas De Uso Común Y No Común) De Programación Que Están Asociados Con Los Diferentes Lenguajes De Programación. (2)  Implementar, Probar Y Depurar Programas Que Utilicen Los Diferentes Modelos De Programación En Los Lenguajes Correspondientes (1) |

Competencias Transversales

|  |
| --- |
| 2.1 (1) Conocer diferentes modelos o estrategias para representar un problema (modelos conceptuales, cualitativos, cuantitativos, etc.) (1)  3.3 (2) Sintetizar (en español) la idea de un tema técnico tomando como fuente varios artículos científicos escritos en inglés (2) |

Resultados de Aprendizaje Esperados (RAE)

|  |
| --- |
|  |

Estrategias Pedagógicas

|  |
| --- |
| * Clases expositivas para explicar la teoría * Clases prácticas para aplicar la teoría. El trabajo en algunas de dichas clases aporta a talleres que serán evaluados |

Evaluación

|  |
| --- |
| Parcial 1. Máquinas de estado y aplicaciones. Teoría de la computabilidad (30%)  Parcial 2. Paradigmas de programación (30%)  Taller 1. Expresiones regulares (10%)  Taller 2. Programación funcional y de flujo de datos (10%)  Taller 3. Programación reactiva/basada en eventos (10%)  Taller 4. Programación orientada a aspectos y mixins (10%) |

Recursos Bibliográficos

|  |
| --- |
| Básicos   * Bainomugisha, E., Carreton, A. L., Cutsem, T. van, Mostinckx, S., & Meuter, W. de. (2013). A survey on reactive programming. ACM Computing Surveys (CSUR), 45(4), 52. https://doi.org/10.1145/2501654.2501666 * Bracha, G., Cook, W., Bracha, G., & Cook, W. (1990). Mixin-based inheritance. ACM SIGPLAN Notices, 25, 303–311. https://doi.org/10.1145/97946.97982 * Cutland, N. (1980). Computability: An Introduction to Recursive Function Theory (1 edition). Cambridge Eng. ; New York: Cambridge University Press. * Filman, R., & Friedman, D. (2000). Aspect-Oriented Programming is Quantification and Obliviousness. Retrieved from citeseer.ist.psu.edu/filman00aspectoriented.html * Friedl, J. E. F. (2006). Mastering Regular Expressions (Third edition). Sebastapol, CA: O’Reilly Media. * Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). Introduction to Automata Theory, Languages, and Computation (3 edition). Boston: Pearson. * Pierce, B. C. (2002). Types and programming languages. Cambridge, MA, USA: MIT Press. * Reisig, W. (2013). Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Retrieved from https://www.springer.com/gp/book/9783642332777   Complementarios   * Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., & Griswold, W. (2001). An Overview of AspectJ. Lecture Notes in Computer Science, 2072, 327–355. * Kiczales, G., & Irwin, J. (1997). Aspect-Oriented Programming. Proc. of ECOOP 1997. Retrieved from http://www.cs.ubc.ca/~gregor/papers/kiczales-ECOOP1997-AOP.pdf * Kiczales, Gregor, Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., & Griswold, W. G. (2001). Getting started with ASPECTJ. Communications of the ACM, 44(10), 59–65. * ReactiveX. (n.d.). Retrieved August 26, 2019, from http://reactivex.io/ |

Tabla de Contenidos

|  |
| --- |
| 1. Máquinas de estado y aplicaciones  Máquinas de estado finitas  Autómatas finitos deterministas  Autómatas finitos no deterministas  Expresiones regulares  Autómatas push-down  Redes de Petri  2. Teoría de la computabilidad  Máquinas de Turing  Problema de la detención (The halting problem)  3. Paradigmas de programación  Programación funcional  Cálculo Lambda  Dataflow programming  Programación reactiva / basada en eventos  Programación orientada a aspectos  Mixins |

Comentarios

|  |
| --- |
|  |