UNIVERSIDAD
NACIONAL DE SAN CRISTÓBAL
DE HUAMANGA

Real, Pontificia y Nacional
1677

Book of Syllabi

Ciencia de la Computación

– 2024-II–

: May 23, 2025

Equipo de Trabajo

Ernesto Cuadros-Vargas (Editor)
Orador distinguido para la *Association of Computing Machinery* (ACM)
Miembro del Directorio de Gobernadores de la Sociedad de Computación del IEEE (2020-2023)
Miembro del *Steering Committee* de *ACM/IEEE-CS Computing Curricula 2020 (CS2020)*
Miembro del *Steering Committee de ACM/IEEE-CS Computing Curricula for Computer Science (CS2013)*
Presidente de la Sociedad Peruana de Computación (SPC) 2001-2007, 2009
email: *ecuadros@spc.org.pe*
http://socios.spc.org.pe/ecuadros

Contents

First Semester **5**
1.1 CS100. Introduction to Computer Science 5
1.2 CS111. Introduction to Programming 19
1.3 CS1D1. Discrete Structures I 27
1.4 MA100. Mathematics I 31

Second Semester **34**
2.1 CS112. Computer Science I 34
2.2 CS1D2. Discrete Structures II 41
2.3 MA101. Math II 45

Third Semester **49**
3.1 CS113. Computer Science II 49
3.2 CS221. Computer Systems Architecture 53
3.3 CS2B1. Platform Based Development 60
3.4 MA102. Calculus I 64

Fourth Semester **68**
4.1 CS210. Algorithms and Data Structures 68
4.2 CS211. Theory of Computation 71
4.3 CS271. Data Management 74
4.4 CS2S1. Operating systems 79
4.5 MA201. Calculus II 87
4.6 MA203. Statistics and Probabilities 90

Fifth Semester **92**
5.1 CS212. Analysis and Design of Algorithms 92
5.2 CS272. Databases II 96
5.3 CS291. Software Engineering I 100
5.4 CS342. Compilers 106

Sixth Semester **111**
6.1 CS231. Networking and Communication 111
6.2 CS261. Artificial Intelligence 115
6.3 CS292. Software Engineering II 124
6.4 CS311. Competitive Programming 130
6.5 CS312. Advanced Data Structures 134
6.6 CS393. Information systems 138
6.7 MA307. Mathematics applied to computing 140

Seventh Semester **143**
7.1 CS2H1. User Experience (UX) . . . 143
7.2 CS391. Software Engineering III . . . 149
7.3 CS3I1. Computer Security . . . 155
7.4 CS251. Computer graphics . . . 165
7.5 CS262. Machine learning . . . 172

Eighth Semester **174**
8.1 CS281. Computing in Society . . . 174
8.2 CS3P1. Parallel and Distributed Computing . . . 183
8.3 CS361. Computational Vision . . . 189

Ninth Semester **191**
9.1 CS370. Big Data . . . 191
9.2 CB309. Bioinformatics . . . 194
9.3 CS369. Topics in Artificial Intelligence . . . 198
9.4 CS351. Topics in Computer Graphics . . . 204
9.5 CS392. Tópicos en Ingeniería de Software . . . 206

Tenth Semester **211**
10.1 CS365. Evolutionary Computing . . . 211
10.2 CS3P2. Cloud Computing . . . 213
10.3 CS3P3. Internet of Things . . . 216
10.4 FG211. Professional Ethics . . . 222

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS100. Introduction to Computer Science (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS100. Introduction to Computer Science
2.2 Semester	:	1^{st} Semester.
2.3 Credits	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course serves as the foundation for understanding the fundamental concepts of computational thinking applicable across various professions.

The course provides, starting from ground zero, a panoramic view of: introductory computational thinking, data storage, computer architecture, operating systems, networks and the Internet, algorithms, sorting methods, software engineering, databases, data structures, software engineering, computer graphics, artificial intelligence among others.

Designed as an introductory course to Computer Science, the concepts are presented in a playful manner and using an Active Learning methodology. Throughout the course, active audience participation is encouraged, akin to a theatrical performance.

The related knowledge areas covered are directly aligned with the Computing Curricula ACM/IEEE-CS.

The course **does not require** any prior knowledge in computer handling topics and can be taken by student from any field.

5. GOALS

- Introduce the fundamental concepts of Computational Thinking and Computer Science to students from any professional background.
- Develop their ability to abstract.
- Understand how Computational Thinking is applied in each of their professions.
- Apply advanced concepts in a simplified manner in any career.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Computational Thinking (Part I) (4)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• General course instructions. • Explanation of the evaluation system. • Definition of Computing. • Computing as a Human-Computer dyad. • Distortions in the definition of computing. • Computing as the automation of abstraction. • Computing and Engineering: similarities and differences. • Algorithmic problem-solving. • Dynamics: Understanding the execution of an algorithm at human speed.	• Apply the fundamental concepts of computing in real-life situations. [Usar] • Identify distortions of Computing in real-life situations. [Usar] • Clearly identify at least 3 contexts of using the word"Engineer" in English. [Evaluar] • Identify the limitations of humans in solving computational problems. [Usar]
Readings : [BB19]	

Unit 2: Computational Thinking. Part II (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Binary vs. decimal numbering. • Character representation: the ASCII table. • Internal representation of colors. • Understanding an image from the inside. • Binary search. • Computational complexity of an algorithm.	• Apply various numbering systems to real-world problems. [Usar] • Understand the internal representation of characters in the ASCII and UTF-8 tables. [Familiarizarse] • Understand the representation of colors in an image. [Familiarizarse] • Apply Divide and Conquer algorithmic strategy. [Usar] • Determine basic analysis of algorithmic complexity. [Usar]
Readings : [BB19]	

Unit 3: Lógica digital y sistemas digitales (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Revisión e historia de la Arquitectura de Computadores. • Lógica combinacional vs. secuencial/Arreglos de puertas de campo programables como bloque fundamental de construcción lógico combinacional-secuencial. • Multiples representaciones / Capas de interpretación (El hardware es solo otra capa)	• Describir el avance paulatino de los componentes de la tecnología de computación, desde los tubos de vacío hasta VLSI, desde las arquitecturas mainframe a las arquitecturas en escala warehouse [Familiarizarse] • Comprender que la tendencia de las arquitecturas modernas de computadores es hacia núcleos múltiples y que el paralelismo es inherente en todos los sistemas de hardware [Familiarizarse] • Explicar las implicancias de los límites de potencia para mejoras adicionales en el rendimiento de los procesadores y también en el aprovechamiento del paralelismo [Familiarizarse] • Relacionar las varias representaciones equivalentes de la funcionalidad de un computador, incluyendo expresiones y puertas lógicas, y ser capaces de utilizar expresiones matemáticas para describir las funciones de circuitos combinacionales y secuenciales sencillos [Familiarizarse] • Diseñar los componentes básicos de construcción de un computador: unidad aritmético lógica (a nivel de puertas lógicas), unidad central de procesamiento (a nivel de registros de transferencia), memoria (a nivel de registros de transferencia) [Usar]
Readings : [BB19]	

Unit 4: Representación de programas (2)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Programas que tienen otros programas como entrada tales como interpretes, compiladores, revisores de tipos y generadores de documentación. • Arboles de sintaxis abstracta, para contrastar la sintaxis correcta. • Estructuras de datos que representan código para ejecución, traducción o transmisión.	• Explicar como programas que procesan otros programas tratan a los otros programas como su entrada de datos [Familiarizarse] • Describir un árbol de sintaxis abstracto para un lenguaje pequeño [Usar] • Describir los beneficios de tener representaciones de programas que no sean cadenas de código fuente [Familiarizarse] • Escribir un programa para procesar alguna representación de código para algún propósito, tales como un interprete, una expresión optimizada, o un generador de documentación [Usar]
Readings : [BB19]	

Unit 5: Criptografía (2)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Terminología básica de criptografía cubriendo las nociones relacionadas con los diferentes socios (comunicación), canal seguro / inseguro, los atacantes y sus capacidades, cifrado, descifrado, llaves y sus características, firmas. • Apoyo a la infraestructura de clave pública para la firma digital y el cifrado y sus desafíos.	• Describir el propósito de la Criptografía y listar formas en las cuales es usada en comunicación de datos [Familiarizarse] • Explicar como los protocolos de intercambio de claves trabajan y como es que pueden fallar [Familiarizarse]
Readings : [BB19]	

Unit 6: Organización y Arquitectura del Sistema de Memoria (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Sistemas de Almacenamiento y su Tecnología. • Jerarquía de Memoria: importancia de la localización temporal y espacial. • Organización y Operaciones de la Memoria Principal. • Latencia, ciclos de tiempo, ancho de banda e intercalación. • Memorias caché (Mapeo de direcciones, Tamaño de bloques, Reemplazo y Politicas de almacenamiento) • Multiprocesador coherencia cache / Usando el sistema de memoria para las operaciones de sincronización de memoria / atómica inter-core. • Memoria virtual (tabla de página, TLB) • Manejo de Errores y confiabilidad. • Error de codificación, compresión de datos y la integridad de datos.	• Identifique las principales tecnologías de memoria (Por ejemplo: SRAM, DRAM, Flash,Disco Magnetico) y su relación costo beneficio [Familiarizarse] • Explique el efecto del retardo de la memoria en tiempo de ejecución [Familiarizarse] • Describa como el uso de jerarquía de memoria (caché, memoria virtual) es aplicado para reducir el retardo efectivo en la memoria [Familiarizarse] • Describa como el uso de jerarquía de memoria (caché, memoria virtual) es aplicado para reducir el retardo efectivo en la memoria [Familiarizarse] • Explique el efecto del retardo de la memoria en tiempo de ejecución [Familiarizarse]
Readings : [BB19]	

Unit 7: Visión general de Sistemas Operativos (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Papel y el propósito del sistema operativo. • Funcionalidad de un sistema operativo típico. • Los mecanismos de apoyo modelos cliente-servidor, dispositivos de mano. • Cuestiones de diseño (eficiencia, robustez, flexibilidad, portabilidad, seguridad, compatibilidad) • Influencias de seguridad, creación de redes, multimedia, sistemas de ventanas.	• Explicar los objetivos y funciones de un sistema operativo moderno [Familiarizarse] • Analizar las ventajas y desventajas inherentes en el diseño de un sistema operativo [Usar] • Describir las funciones de un sistema operativo contemporaneo respecto a conveniencia, eficiencia, y su habilidad para evolucionar [Familiarizarse] • Discutir acerca de sistemas operativos cliente-servidor, en red, distribuidos y cómo se diferencian de los sistemas operativos de un solo usuario [Familiarizarse] • Identificar amenazas potenciales a sistemas operativos y las características del diseño de seguridad para protegerse de ellos [Familiarizarse]
Readings : [BB19]	

Unit 8: Introducción a redes (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Organización de la Internet (proveedores de servicios de Internet, proveedores de contenido, etc) • Técnicas de Switching (por ejemplo, de circuitos, de paquetes) • Piezas físicas de una red, incluidos hosts, routers, switches, ISPs, inalámbrico, LAN, punto de acceso y firewalls. • Principios de capas (encapsulación, multiplexación) • Roles de las diferentes capas (aplicación, transporte, red, enlace de datos, física)	• Articular la organización de la Internet [Familiarizarse] • Listar y definir la terminología de red apropiada [Familiarizarse] • Describir la estructura en capas de una arquitectura típica en red [Familiarizarse] • Identificar los diferentes tipos de complejidad en una red (bordes, núcleo, etc.) [Familiarizarse]
Readings : [BB19]	

Unit 9: Entrega confiable de datos (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Control de errores (técnicas de retransmisión, temporizadores) • El control de flujo (agradecimientos, ventana deslizante) • Problemas de rendimiento (pipelining) • TCP	• Describir el funcionamiento de los protocolos de entrega fiables [Familiarizarse] • Listar los factores que afectan al rendimiento de los protocolos de entrega fiables [Familiarizarse] • Diseñar e implementar un protocolo confiable simple [Usar]
Readings : [BB19]	

Unit 10: Análisis Básico (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Diferencias entre el mejor, el esperado y el peor caso de un algoritmo. • Análisis asintótico de complejidad de cotas superior y esperada. • Definición formal de la Notación Big O. • Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial. • Medidas empíricas de desempeño. • Compensación entre espacio y tiempo en los algoritmos. • Uso de la notación Big O.	• Explique a que se refiere con "mejor", "esperado" y "peor" caso de comportamiento de un algoritmo [Familiarizarse] • En el contexto de a algoritmos específicos, identifique las características de data y/o otras condiciones o suposiciones que lleven a diferentes comportamientos [Evaluar] • Determine informalmente el tiempo y el espacio de complejidad de simples algoritmos [Usar] • Indique la definición formal de Big O [Familiarizarse] • Lista y contraste de clases estándares de complejidad [Familiarizarse] • Realizar estúdios empíricos para validar una hipótesis sobre runtime stemming desde un análisis matemático Ejecute algoritmos con entrada de varios tamaños y compare el desempeño [Evaluar] • Da ejemplos que ilustran las compensaciones entre espacio y tiempo que se dan en los algoritmos [Familiarizarse] • Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Usar] • Explicar el uso de la notación theta grande, omega grande y o pequeña para describir la cantidad de trabajo hecho por un algoritmo [Familiarizarse]
Readings : [BB19]	

Unit 11: Algoritmos y Estructuras de Datos fundamentales (8)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo. • Algoritmos de búsqueda secuencial y binaria. • Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción) • Algoritmos de ordenamiento con peor caso o caso promedio en O(N lg N) (Quicksort, Heapsort, Mergesort) • Tablas Hash, incluyendo estratégias para evitar y resolver colisiones. • Árboles de búsqueda binaria: – Operaciones comunes en árboles de búsqueda binaria como seleccionar el mínimo, máximo, insertar, eliminar, recorrido en árboles. • Grafos y algoritmos en grafos: – Representación de grafos (ej., lista de adyacencia, matriz de adyacencia) – Recorrido en profundidad y amplitud • Montículos (Heaps) • Grafos y algoritmos en grafos: – Algoritmos de la ruta más corta (algoritmos de Dijkstra y Floyd) – Árbol de expansión mínima (algoritmos de Prim y Kruskal) • Búsqueda de patrones y algoritmos de cadenas/texto (ej. búsqueda de subcadena, búsqueda de expresiones regulares, algoritmos de subsecuencia común más larga)	• Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse] • Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Usar] • Implementar algoritmos numéricos básicos [Usar] • Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar] • Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Usar] • Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse] • Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, busqueda y hashing [Familiarizarse] • Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] • Explicar como el balanceamiento del arbol afecta la eficiencia de varias operaciones de un arbol de búsqueda binaria [Familiarizarse] • Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección,y para implementar el algoritmo en un contexto en específico [Evaluar]
Readings : [BB19]	

Unit 12: Sistemas de Bases de Datos (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Enfoque y Evolución de Sistemas de Bases de Datos. • Componentes del Sistema de Bases de Datos. • Diseño de las funciones principales de un DBMS. • Arquitectura de base de datos e independencia de datos. • Uso de un lenguaje de consulta declarativa. • Sistemas de apoyo a contenido estructurado y / o corriente. • Enfoques para la gestión de grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce).	• Describe los enfoques principales para almacenar y procesar larges volúmenes de datos [Familiarizarse] • Explica las características que distinguen un esquema de base de datos de aquellos basados en la programación de archivos de datos [Familiarizarse] • Describe los componentes de un sistema de bases datos y da ejemplos de su uso [Familiarizarse] • Cita las metas básicas, funciones y modelos de un sistema de bases de datos [Familiarizarse] • Describe los componentes de un sistema de bases datos y da ejemplos de su uso [Familiarizarse] • Identifica las funciones principales de un SGBD y describe sus roles en un sistema de bases de datos [Familiarizarse] • Explica las características que distinguen un esquema de base de datos de aquellos basados en la programación de archivos de datos [Familiarizarse] • Usa un lenguaje de consulta declarativo para recoger información de una base de datos [Usar] • Describe las capacidades que las bases de datos brindan al apoyar estructuras y/o la secuencia de flujo de datos, ejm. texto [Familiarizarse]
Readings : [BB19]	

Unit 13: Programación orientada a objetos (4)

Competences Expected: 1,2,6

Topics	**Learning Outcomes**

• Diseño orientado a objetos:
  – Descomposicion en objetos que almacenan estados y poseen comportamiento
  – Diseño basado en jerarquia de clases para modelamiento
• Definición de las categorías, campos, métodos y constructores.
• Las subclases, herencia y método de alteración temporal.
• Asignación dinámica: definición de método de llamada.
• Subtipificación:
  – Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos.
  – Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos.
  – Relación entre subtipos y la herencia.
• Lenguajes orientados a objetos para la encapsulación:
  – privacidad y la visibilidad de miembros de la clase
  – Interfaces revelan único método de firmas
  – clases base abstractas
• Uso de coleccion de clases, iteradores, y otros componentes de la libreria estandar.
	• Diseñar e implementar una clase [Usar]
	• Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar]
	• Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar]
	• Comparar y contrastar (1) el enfoque procedurar/funcional- definiendo una función por cada operación con el cuerdo de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar]
	• Explicar la relación entre la herencia orientada a objetos (codigo compartido y *overriding*) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse]
	• Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar]
	• Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, selecionar la forma mas natural por cada lenguaje [Usar]

Readings : [BB19]	

Unit 14: Procesos de Software (4)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Consideraciones a nivel de sistemas, ejem., la interacción del software con su entorno. • Introducción a modelos del proceso de software (e.g., cascada, incremental, agil): – Actividades con ciclos de vida de software. • Programación a gran escala versus programación individual. • Evaluación de modelos de proceso de software. • Conceptos de calidad de software. • Mejoramiento de procesos. • Modelos de madurez de procesos de software. • Mediciones del proceso de software.	• Describir cómo la programación en grandes equipos difiere de esfuerzos individuales con respecto a la comprensión de una gran base de código, lectura de código, comprensión de las construcciones, y comprensión de contexto de cambios [Familiarizarse] • Describir las ventajas y desventajas relativas entre varios modelos importantes de procesos (por ejemplo, la cascada, iterativo y ágil) [Familiarizarse] • Diferenciar entre las fases de desarrollo de software [Familiarizarse] • Describir cómo la programación en grandes equipos difiere de esfuerzos individuales con respecto a la comprensión de una gran base de código, lectura de código, comprensión de las construcciones, y comprensión de contexto de cambios [Familiarizarse] • Explicar el papel de los modelos de madurez de procesos en la mejora de procesos [Familiarizarse] • Comparar varios modelos comunes de procesos con respecto a su valor para el desarrollo de las clases particulares de sistemas de software, teniendo en cuenta diferentes aspectos tales como, estabilidad de los requisitos, tamaño y características no funcionales [Usar] • Definir la calidad del software y describir el papel de las actividades de aseguramiento de la calidad en el proceso de software [Familiarizarse]
Readings : [BB19]	

Unit 15: Cuestiones fundamentales (2)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Descripción general de los problemas de Inteligencia Artificial, ejemplos recientes de aplicaciones de Inteligencia artificial. • ¿Qué es comportamiento inteligente? – El Test de Turing – Razonamiento Racional versus No Racional	• Describir el test de Turing y el experimento pensado cuarto chino" (*Chinese Room*) [Familiarizarse]
Readings : [BB19]	

Unit 16: Estrategias de búsquedas básicas (1)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Espacios de Problemas (estados, metas y operadores), solución de problemas mediante búsqueda. • Factored representation (factoring state hacia variables) • Uninformed search (breadth-first, depth-first, depth-first with iterative deepening) • Heurísticas y búsqueda informada (hill-climbing, generic best-first, A*) • El espacio y el tiempo de la eficiencia de búsqueda. • Dos jugadores juegos (introducción a la búsqueda minimax). • Satisfacción de restricciones (backtracking y métodos de búsqueda local).	• Formula el espacio eficiente de un problema para un caso expresado en lenguaje natural (ejm. Inglés) en términos de estados de inicio y final, así como sus operadores [Usar] • Describe el rol de las heurísticas y describe los intercambios entre completitud, óptimo, complejidad de tiempo, y complejidad de espacio [Familiarizarse] • Describe el problema de la explosión combinatoria del espacio de búsqueda y sus consecuencias [Familiarizarse] • Selecciona e implementa un apropiado algoritmo de búsqueda no informado para un problema, y describe sus complejidades de tiempo y espacio [Usar] • Selecciona e implementa un apropiado algoritmo de búsqueda no informado para un problema, y describe sus complejidades de tiempo y espacio [Usar] • Evalúa si una heurística dada para un determinado problema es admisible/puede garantizar una solución óptima [Evaluar] • Compara y contrasta tópicos de búsqueda básica con temas jugabilidad de juegos [Familiarizarse]
Readings : [BB19]	

Unit 17: Aprendizaje Automático Básico (1)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Definición y ejemplos de la extensa variedad de tareas de aprendizaje de máquina, incluida la clasificación. • Aprendizaje inductivo • Aprendizaje simple basado en estadísticas, como el clasificador ingenuo de Bayes, árboles de decisión. • El problema exceso de ajuste. • Medicion clasificada con exactitud.	• Listar las diferencias entre los tres principales tipos de aprendizaje: supervisado, no supervisado y por refuerzo [Familiarizarse] • Identificar ejemplos de tareas de clasificación, considerando las características de entrada disponibles y las salidas a ser predecidas [Familiarizarse] • Describir el sobre ajuste (*overfitting*) en el contexto de un problema [Familiarizarse]
Readings : [BB19]	

Unit 18: Conceptos Fundamentales (2)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Aplicaciones multimedia, incluyendo interfaces de usuario, edición de audio y vídeo, motores de juego, cad, visualización, realidad virtual. • Digitalización de datos analógicos, la resolución y los límites de la percepción humana, por ejemplo, los píxeles de la pantalla visual, puntos para impresoras láser y muestras de audio • El uso de las API estándar para la construcción de interfaces de usuario y visualización de formatos multimedia estándar • Formatos estándar, incluyendo formatos sin pérdidas y con pérdidas. • Modelos de color sustractivo Aditivo y (CMYK y RGB) y por qué estos proporcionan una gama de colores. • Soluciones de compensación entre el almacenamiento de datos y los datos re-computing es personalizado por vectores y raster en representaciones de imágenes. • Animación como una secuencia de imágenes fijas. • Almacenamiento doble.	• Identificar usos comunes de presentaciones digitales de humanos (por ejemplo, computación gráfica,sonido) [Familiarizarse] • Explicar en términos generales cómo las señales analógicas pueden ser representadas por muestras discretas, por ejemplo,cómo las imagenes pueden ser representadas por pixeles [Familiarizarse] • Explicar cómo las limitaciones en la percepción humana afectan la selección de la representación digital de señales analógicas [Usar] • Describir las diferencias entre técnicas de compresión de imágenes con pérdida y sin pérdida ejemplificando cómo se reflejan en formatos de archivos de imágenes conocidos como JPG, PNG, MP3, MP4, y GIF [Familiarizarse] • Describir modelos de color y su uso en los dispositivos de visualización de gráficos [Familiarizarse] • Describir las ventajas y desventajas entre el almacenamiento de información vs almacenar suficiente información para reproducir la información, como en la diferencia entre el vector y la representación de la trama [Familiarizarse] • Describir los procesos básico de la producción de movimiento continuo a partir de una secuencia de cuadros discretos(algunas veces llamado it flicker fusion) [Familiarizarse] • Describir cómo el doble buffer puede eliminar el parpadeo de la animación [Familiarizarse]
Readings : [BB19]	

Unit 19: Rendering Básico (2)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Renderizado en la naturaleza, por ejemplo, la emisión y dispersión de la luz y su relación con la integración numérica. • Renderizado Fordward and Backward (i.e., *ray-casting* y rasterización) • Representación poligonal • Radiometría básica, triángulos similares y modelos de proyecciones • Afinamiento y Transformaciones de Sistemas de coordenadas • *Ray tracing* • Visibilidad y oclusión, incluyendo soluciones a este problema, como el almacenamiento en búfer de profundidad, algoritmo del pintor, y el trazado de rayos. • Representación de la ecuación de adelante hacia atrás. • Rasterización triangular simple. • Mapeo de texturas, incluyendo minificación y magnificación (e.g., MIP-mapping trilineal) • Aplicación de la representación de estructuras de datos espaciales. • Muestreo y anti-aliasing. • Gráficos en escena y la canalización de gráficos.	• Discutir el problema de transporte de la luz y su relación con la integración numérica, es decir, se emite luz, dispersa alrededor de la escena, y es medida por el ojo [Familiarizarse] • Describir la tubería básica gráficos y cómo el factor de representación va hacia adelante y atrás en esta [Familiarizarse] • Crear un programa para visualizar modelos 3D de imagenes gráficas simples [Usar] • Derivar la perspectiva lineal de triángulos semejantes por conversión de puntos (x,y,z) a puntos (x/z, y/z, 1) [Usar] • Obtener puntos en 2-dimensiones y 3-dimensiones por aplicación de transformaciones afín [Usar] • Aplicar sistema de coordenadas de 3-dimensiones y los cambios necesarios para extender las operaciones de transformación 2D para manejar las transformaciones en 3D [Usar] • Explicar la dualidad de rastreo de rayos/rasterización para el problema de visibilidad [Familiarizarse] • Implementar simples procedimientos que realicen la transformación y las operaciones de recorte de imágenes simples en 2 dimensiones [Usar] • Calcular las necesidades de espacio en base a la resolución y codificación de color [Evaluar] • Calcular los requisitos de tiempo sobre la base de las frecuencias de actualización, técnicas de rasterización [Evaluar]
Readings : [BB19]	

Unit 20: Closure Class: How Does a Search Engine Like Google Work? (2)	
Competences Expected: 1,2,6	
Topics	**Learning Outcomes**
• Problem analysis • The index does not grow linearly with the size of the indexed information. • Response time does not depend on the size of the "database." • Response time does not depend on the number of occurrences found. • Combining various data structures to reach a solution. • Analyzing the scalability of the solution.	• Understand the principles under which a search engine is created [Usar] • Correctly apply data structures to solve the problem [Usar] • Apply concepts related to algorithmic complexity in a search engine [Usar].
Readings : [BB19]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[BB19] J. Glenn Brookshear and Dennis Brylow. *Computer Science: An Overview*. Ed. by PEARSON. Global Edition. Pearson, 2019. URL: `http://www.pearsonhighered.com/brookshear`.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS111. Introduction to Programming (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS111. Introduction to Programming
2.2 Semester	:	1^{st} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the first course in the sequence of introductory courses to Computer Science.This course is intended to cover the concepts outlined by the Computing Curricula ACM/IEEE-CS 2013. Programming is one of the pillars of Computer Science; any professional of the area, will need to program to materialize their models and proposals. This course introduces participants to the fundamental concepts of this art. Topics include data types, control structures, functions, lists, recursion, and the mechanics of execution, testing, and debugging.

5. GOALS

- Introduce the fundamental concepts of programming.
- Develop the ability of abstraction using programming language

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Historia (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Pre-historia – El mundo antes de 1946. • Historia del hardware, software, redes. • Pioneros de la Computación. • Historia de Internet.	• Identificar importantes tendencias en la historia del campo de la computación [Familiarizarse] • Identificar las contribuciones de varios pioneros en el campo de la computación [Familiarizarse] • Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse] • Comparar la vida diaria antes y después de la llegada de los ordenadores personales y el Internet [Evaluar]
Readings : [BB19], [Gut13], [Zel10]	

Unit 2: Sistemas de tipos básicos (2)	
Competences Expected:	
Topics	**Learning Outcomes**

• Tipos como conjunto de valores junto con un conjunto de operaciones.
  – Tipos primitivos (p.e. números, booleanos)
  – Composición de tipos construídos de otros tipos (p.e., registros, uniones, arreglos, listas, funciones, referencias)
• Asociación de tipos de variables, argumentos, resultados y campos.
• Tipo de seguridad y los errores causados por el uso de valores de manera incompatible dadas sus tipos previstos.
	• Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse]
	• Para un lenguaje con sistema de tipos estático, describir las operaciones que están prohibidas de forma estática, como pasar el tipo incorrecto de valor a una función o método [Familiarizarse]
	• Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse]
	• Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar]
	• Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar]
	• Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar]

Readings : [Gut13], [Zel10]	

Unit 3: Conceptos Fundamentales de Programación (9)	
Competences Expected:	
Topics	**Learning Outcomes**
• Sintaxis y semántica básica de un lenguaje de alto nivel. • Variables y tipos de datos primitivos (ej., numeros, caracteres, booleanos) • Expresiones y asignaciones. • Operaciones básicas I/O incluyendo archivos I/O. • Estructuras de control condicional e iterativas. • Paso de funciones y parámetros. • Concepto de recursividad.	• Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar] • Identifica y describe el uso de tipos de datos primitivos [Familiarizarse] • Escribe programas que usan tipos de datos primitivos [Usar] • Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar] • Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar] • Escribe un programa que usa E/S de archivos para brindar persistencia a través de ejecuciones múltiples [Usar] • Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Familiarizarse] • Describe el concepto de recursividad y da ejemplos de su uso [Evaluar] • Identifica el caso base y el caso general de un problema basado en recursividad [Familiarizarse]
Readings : [Gut13], [Zel10]	

Unit 4: Análisis Básico (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Diferencias entre el mejor, el esperado y el peor caso de un algoritmo. • Definición formal de la Notación Big O. • Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial. • Uso de la notación Big O. • Análisis de algoritmos iterativos y recursivos.	• Explique a que se refiere con "mejor", "esperado" y "peor" caso de comportamiento de un algoritmo [Familiarizarse] • En el contexto de a algoritmos específicos, identifique las características de data y/o otras condiciones o suposiciones que lleven a diferentes comportamientos [Familiarizarse] • Indique la definición formal de Big O [Familiarizarse] • Use la notación formal de la Big O para dar límites superiores asintóticos en la complejidad de tiempo y espacio de los algoritmos [Usar] • Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Usar]
Readings : [Gut13], [Zel10]	

Unit 5: Algoritmos y Estructuras de Datos fundamentales (8)

Competences Expected:

Topics	**Learning Outcomes**
• Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo. • Algoritmos de búsqueda secuencial y binaria. • Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción) • Algoritmos de ordenamiento con peor caso o caso promedio en O(N lg N) (Quicksort, Heapsort, Mergesort) • Tablas Hash, incluyendo estratégias para evitar y resolver colisiones. • Árboles de búsqueda binaria: – Operaciones comunes en árboles de búsqueda binaria como seleccionar el mínimo, máximo, insertar, eliminar, recorrido en árboles. • Grafos y algoritmos en grafos: – Representación de grafos (ej., lista de adyacencia, matriz de adyacencia) – Recorrido en profundidad y amplitud • Montículos (Heaps) • Grafos y algoritmos en grafos: – Algoritmos de la ruta más corta (algoritmos de Dijkstra y Floyd) – Árbol de expansión mínima (algoritmos de Prim y Kruskal) • Búsqueda de patrones y algoritmos de cadenas/texto (ej. búsqueda de subcadena, búsqueda de expresiones regulares, algoritmos de subsecuencia común más larga)	• Implementar algoritmos numéricos básicos [Usar] • Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar] • Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Usar] • Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse] • Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, busqueda y hashing [Familiarizarse] • Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] • Explicar como el balanceamiento del arbol afecta la eficiencia de varias operaciones de un arbol de búsqueda binaria [Familiarizarse] • Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Usar] • Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección,y para implementar el algoritmo en un contexto en específico [Evaluar] • Describir la propiedad del heap y el uso de heaps como una implementación de colas de prioridad [Familiarizarse] • Resolver problemas usando algoritmos de grafos, incluyendo camino más corto de una sola fuente y camino más corto de todos los pares, y como mínimo un algoritmo de arbol de expansion minima [Usar] • Trazar y/o implementar un algoritmo de comparación de string [Usar]

Readings : [Gut13], [Zel10]

Unit 6: Programación orientada a objetos (4)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Object poriented languages and encapsulation: Privacy y visibility of class members. • Definición de las categorías, campos, métodos y constructores. • Subclasses and inheritance. • Asignación dinámica: definición de método de llamada.	• Diseñar e implementar una clase [Usar] • Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Familiarizarse] • Comparar y contrastar (1) el enfoque procedurar/funcional- definiendo una función por cada operación con el cuerdo de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Familiarizarse] • Explicar la relación entre la herencia orientada a objetos (codigo compartido y *overriding*) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse] • Use encapsulation to create objects [Familiarizarse].
Readings : [Gut13], [Zel10]	

Unit 7: Algoritmos y Diseño (9)	
Competences Expected:	
Topics	**Learning Outcomes**
• Conceptos y propiedades de los algoritmos – Comparación informal de la eficiencia de los algoritmos (ej., conteo de operaciones) • Rol de los algoritmos en el proceso de solución de problemas • Estrategias de solución de problemas – Funciones matemáticas iterativas y recursivas – Recorrido iterativo y recursivo en estructura de datos – Estrategias Divide y Conquistar • Conceptos y principios fundamentales de diseño – Abstracción – Descomposición de Program – Encapsulamiento y camuflaje de información – Separación de comportamiento y aplicación	• Discute la importancia de los algoritmos en el proceso de solución de un problema [Familiarizarse] • Discute como un problema puede ser resuelto por múltiples algoritmos, cada uno con propiedades diferentes [Familiarizarse] • Crea algoritmos para resolver problemas simples [Usar] • Usa un lenguaje de programación para implementar, probar, y depurar algoritmos para resolver problemas simples [Usar] • Implementa, prueba, y depura funciones recursivas simples y sus procedimientos [Usar] • Determina si una solución iterativa o recursiva es la más apropiada para un problema [Evaluar] • Implementa un algoritmo de divide y vencerás para resolver un problema [Usar] • Aplica técnicas de descomposición para dividir un programa en partes más pequeñas [Usar] • Identifica los componentes de datos y el comportamiento de mútiples tipos de datos abstractos [Usar] • Implementa un tipo de dato abstracto coherente, con la menor pérdida de acoplamiento entre componentes y comportamientos [Usar] • Identifica las fortalezas y las debilidades relativas entre múltiples diseños e implementaciones de un problema [Evaluar]
Readings : [Gut13], [Zel10]	

Unit 8: Métodos de Desarrollo (1)	
Competences Expected:	
Topics	**Learning Outcomes**
• Entornos modernos de programación: – Búsqueda de código. – Programación usando libreria de componentes y sus APIs.	• Construir y depurar programas que utilizan las bibliotecas estándar disponibles con un lenguaje de programación elegido [Familiarizarse]
Readings : [Gut13], [Zel10]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Zel10] John Zelle. *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates Inc, 2010.

[Gut13] John V Guttag. . *Introduction To Computation And Programming Using Python*. MIT Press, 2013.

[BB19] J. Glenn Brookshear and Dennis Brylow. *Computer Science: An Overview*. Ed. by PEARSON. Global Edition. Pearson, 2019. URL: `http://www.pearsonhighered.com/brookshear`.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS1D1. Discrete Structures I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS1D1. Discrete Structures I
2.2 Semester	:	1^{st} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Discrete structures provide the theoretical foundations necessary for computation. These fundamentals are not only useful to develop computation from a theoretical point of view as it happens in the course of computational theory, but also is useful for the practice of computing; In particular in applications such as verification, cryptography, formal methods, etc.

5. GOALS

- Apply Properly concepts of finite mathematics (sets, relations, functions) to represent data of real problems.
- Model real situations described in natural language, using propositional logic and predicate logic.
- Determine the abstract properties of binary relations.
- Choose the most appropriate demonstration method to determine the veracity of a proposal and construct correct mathematical arguments.
- Interpret mathematical solutions to a problem and determine their reliability, advantages and disadvantages.
- Express the operation of a simple electronic circuit using Boolean algebra.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Funciones, relaciones y conjuntos (22)

Competences Expected:

Topics	Learning Outcomes
• Conjuntos: – Diagramas de Venn – Unión, intersección, complemento – Producto Cartesiano – Potencia de conjuntos – Cardinalidad de Conjuntos finitos • Relations: – Reflexivity, simmetry, transitivity – Equivalence relations – Partial order relations and sets – Extremal elements of a partially ordered sets • Funciones: – Suryecciones, inyecciones, biyecciones – Inversas – Composición	• Explicar con ejemplos la terminología básica de funciones, relaciones y conjuntos [Evaluar] • Realizar las operaciones asociadas con conjuntos, funciones y relaciones [Evaluar] • Relacionar ejemplos prácticos para conjuntos funciones o modelos de relación apropiados e interpretar la asociación de operaciones y terminología en contexto [Evaluar]

Readings : [Gri03], [Ros07], [Vel06]

Unit 2: Lógica básica (14)

Competences Expected:

Topics	Learning Outcomes
• Lógica proposicional. • Conectores lógicos. • Tablas de verdad. • Forma normal (conjuntiva y disyuntiva) • Validación de fórmula bien formada. • Reglas de inferencia proposicional (conceptos de modus ponens y modus tollens) • Logica de predicados: – Cuantificación universal y existencial • Limitaciones de la lógica proposicional y de predicados (ej. problemas de expresividad)	• Convertir declaraciones lógicas desde el lenguaje informal a expresiones de lógica proposicional y de predicados [Usar] • Aplicar métodos formales de simbolismo proposicional y lógica de predicados, como el cálculo de la validez de formulas y cálculo de formas normales [Usar] • Usar reglas de inferencia para construir demostraciones en lógica proposicional y de predicados [Usar] • Describir como la lógica simbólica puede ser usada para modelar situaciones o aplicaciones de la vida real, incluidos aquellos planteados en el contexto computacional como análisis de software (ejm. programas correctores), consulta de base de datos y algoritmos [Familiarizarse] • Aplicar métodos formales de simbolismo proposicional y lógica de predicados, como el cálculo de la validez de formulas y cálculo de formas normales [Usar] • Describir las fortalezas y limitaciones de la lógica proposicional y de predicados [Usar]

Readings : [Ros07], [Gri03], [Vel06]

Unit 3: Técnicas de demostración (14)	
Competences Expected:	
Topics	**Learning Outcomes**
• Nociones de implicancia, equivalencia, conversión, inversa, contrapositivo, negación, y contradicción • Estructura de pruebas matemáticas. • Demostración directa. • Refutar por contraejemplo. • Demostracción por contradicción. • Inducción sobre números naturales. • Inducción estructural. • Inducción leve y fuerte (Ej. Primer y Segundo principio de la inducción) • Definiciones matemáticas recursivas. • Conjuntos bien ordenados.	• Identificar la técnica de demostración utilizada en una demostración dada [Evaluar] • Describir la estructura básica de cada técnica de demostración (demostración directa, demostración por contradicción e inducción) descritas en esta unidad [Usar] • Aplicar las técnicas de demostración (demostración directa, demostración por contradicción e inducción) correctamente en la construcción de un argumento solido [Usar] • Determine que tipo de demostración es la mejor para un problema dado [Evaluar] • Explicar el paralelismo entre ideas matemáticas y/o inducción estructural para la recursión y definir estructuras recursivamente [Familiarizarse] • Explicar la relación entre inducción fuerte y débil y dar ejemplos del apropiado uso de cada uno [Evaluar] • Enunciar el principio del buen-orden y su relación con la inducción matemática [Familiarizarse]
Readings : [Ros07], [**Vel06**], [Sch12], [Vel06]	

Unit 4: Data Representation (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Numerical representation: sign-magnitude, floating point. • Representation of other objects: sets, relations, functions.	• Explain numerical representations such as sign-magnitude and floating point. [Evaluar]. • Carry out arithmetic operations using different kinds of representations. [Evaluar]. • Explain the floating point standard IEEE-754 [Familiarizarse].
Readings : [Ros07], [Gri03], [Vel06]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Gri03] R. Grimaldi. *Discrete and Combinatorial Mathematics: An Applied Introduction*. 5 ed. Pearson, 2003.

[Vel06] Daniel J. Velleman. *How to Prove It: A Structured Approach*. Ed. by Cambridge University Pres. 2nd. 2006.

[Ros07] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. 7 ed. 2007.

[Sch12] Edward R. Scheinerman. *Mathematics: A Discrete Introduction*. 3 ed. 2012.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE
MA100. Mathematics I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	MA100. Mathematics I
2.2 Semester	:	1^{st} Semester.
2.3 Credits	:	5
2.4 Horas	:	2 HT; 6 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE
The course aims to develop in students the skills to deal with models in science and engineering related to single variable differential calculus skills. In the course it is studied and applied concepts related to calculation limits, derivatives and integrals of real and vector functions of single real variables to be used as base and support for the study of new contents and subjects. Also seeks to achieve reasoning capabilities and applicability to interact with real-world problems by providing a mathematical basis for further professional development activities.

5. GOALS

- .
- .
- .

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• . • .	• . • .
Readings : [Ste12], [ión14]	

Unit 2: (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• . • . • . • . • . • .	• . • . • . • . • . • .
Readings : [Ste12], [ión14]	

Unit 3: (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• . • . • . • . • .	• . • . • . • . • . • . • . • . • . • .
Readings : [Ste12], [ión14]	

Unit 4: (22)	
Competences Expected:	
Topics	**Learning Outcomes**
• . • . • . • .	• . • . • . • . • . • . • . • . • . • . • .
Readings : [Ste12], [ión14]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Ste12] James Stewart. *Calculus*. 7th. 2012.

[ión14] ROn Larson ión. *Calculus*. 10th. 2014.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS112. Computer Science I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS112. Computer Science I
2.2 Semester	:	2^{nd} Semester.
2.3 Credits	:	5
2.4 Horas	:	2 HT; 6 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS111. Introduction to Programming. (1^{st} Sem) CS111. Introduction to Programming. (1^{st} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the second course in the sequence of introductory courses in computer science. The course will introduce students in the various topics of the area of computing such as: Algorithms, Data Structures, Software Engineering, etc.

5. GOALS

- Introduce the student to the foundations of the object orientation paradigm, allowing the assimilation of concepts necessary to develop information systems.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Familiarity)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Overview of Programming Languages (1)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Brief review of programming paradigms. • Comparison between functional and imperative programming. • History of programming languages (emphasis on C and C++).	• Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse]
Readings : [Str13], [Jos19], [Dei17]	

Unit 2: Máquinas virtuales (2)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• The concept of a virtual machine. • Tipos de virtualización (incluyendo Hardware / Software, OS, Servidor, Servicio, Red) . • Intermediate languages.	• Explicar el concepto de memoria virtual y la forma cómo se realiza en hadware y software [Familiarizarse] • Diferenciar emulacion y el aislamiento [Familiarizarse] • Evaluar virtualización de compensaciones [Evaluar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 3: Sistemas de tipos básicos (6)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Type systems in programming languages. • Declaration models (linking, visibility, scope, and lifetime). • Overview of type checking.	• Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse] • Para un lenguaje con sistema de tipos estático, describir las operaciones que están prohibidas de forma estática, como pasar el tipo incorrecto de valor a una función o método [Familiarizarse] • Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse] • Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar] • Dar un ejemplo de un programa que no verifique tipos en un lenguaje particular y sin embargo no tenga error cuando es ejecutado [Familiarizarse] • Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar] • Explicar como las reglas de tipificación definen el conjunto de operaciones que legales para un tipo [Familiarizarse] • Escribir las reglas de tipo que rigen el uso de un particular tipo compuesto [Usar] • Explicar por qué indecidibilidad requiere sistemas de tipo para conservadoramente aproximar el comportamiento de un programa [Familiarizarse] • Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar] • Discutir las diferencias entre, genéricos (*generics*), subtipo y sobrecarga [Familiarizarse] • Explicar múltiples beneficios y limitaciones de tipificación estática en escritura, mantenimiento y depuración de un software [Familiarizarse]
Readings : [Str13], [Jos19], [Dei17]	

Unit 4: Conceptos Fundamentales de Programación (6)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Diseño orientado a objetos: – Descomposicion en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquia de clases para modelamiento • Variables and data types. • Expressions and operators. • Conditional statements.	• Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar] • Identifica y describe el uso de tipos de datos primitivos [Familiarizarse] • Escribe programas que usan tipos de datos primitivos [Usar] • Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar] • Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar] • Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Evaluar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 5: Functions (6)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Paso de funciones y parámetros. • Parameter passing. • Function overloading. • Fundamentals of recursion. • Function templates.	• Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar] • Understand and apply the concept of parameter passing to a function, both by value and by reference. [Usar] • Identify and apply the concept of function overloading. [Usar] • Describe el concepto de recursividad y da ejemplos de su uso [Familiarizarse] • Design, implement, and apply the concept of templates to create generic functions. [Usar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 6: Arrays, Pointers, and Memory Management (8)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Array definition. • Multidimensional arrays. • Pointer fundamentals. • Dynamic memory management (new/delete, stack vs. heap). • Smart pointers (unique_ptr, shared_ptr, weak_ptr). • Advanced pointer concepts (pointers to pointers, pointers to functions).	• Understand and implement one-dimensional arrays. [Familiarizarse] • Design and apply the concept of multidimensional arrays. [Usar] • Understand and apply the concept of references and pointers. [Familiarizarse] • Understand, apply, and evaluate the relationship between pointers and arrays. [Evaluar] • Understand and implement dynamic memory management. Differentiate between heap and stack memory regions. [Evaluar] • Design, implement, and evaluate concepts like pointer-to-pointer, pointer-to-function, among other advanced pointer concepts. [Evaluar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 7: Working with Arrays and Pointers (5)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Arrays as function arguments. • Character arrays and pointers. • Pointers and 2-dimensional arrays. • Pointers and multidimensional arrays.	• Demonstrate the use of pointers with different types of arrays. [Usar] • Demonstrate the layout of an array in memory and how pointers are manipulated within those memory spaces. [Usar] • Demonstrate the use of pointer arithmetic with arrays.[Usar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 8: Pointers and Dynamic Memory (5)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Pointers and dynamic memory - stack vs heap. • Pointers as return values from a function in C/C++. • Pointers to functions in C/C++. • Pointers to functions and callbacks. • Memory leaks in C/C++.	• Show the memory structure within a program and understand how the compiler allocates elements on the stack and heap.[Usar] • Demonstrate the use of functions and operators for dynamic memory allocation and deallocation.[Usar] • Understand the implications of returning pointers from functions. [Usar] • Use pointers to functions as parameters. [Usar] • Understand the implications of dynamic memory usage and memory leaks. [Usar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 9: Pointers and Classes (5)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Pointers to class members - attributes. • Pointers to class members - methods and calls to method pointers. • Pointers to class members - static methods and calls to static method pointers. • Pointers to classes - example with linked list management.	• Understand the use of pointers to different elements of a class. [Usar] • Understand the use of pointers to static members of a class. [Usar] • Introduce the node structure and its use in a simple data structure. [Usar] • Introduce data structures, showing a simple implementation of linked lists.[Usar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 10: Programación orientada a objetos (8)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Diseño orientado a objetos: – Descomposicion en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquia de clases para modelamiento • Lenguajes orientados a objetos para la encapsulación: – privacidad y la visibilidad de miembros de la clase – Interfaces revelan único método de firmas – clases base abstractas • Definición de las categorías, campos, métodos y constructores. • Las subclases, herencia y método de alteración temporal. • Subtipificación: – Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos. – Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos. – Relación entre subtipos y la herencia. • Uso de coleccion de clases, iteradores, y otros componentes de la libreria estandar. • Asignación dinámica: definición de método de llamada.	• Diseñar e implementar una clase [Usar] • Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar] • Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar] • Comparar y contrastar (1) el enfoque procedurar/funcional- definiendo una función por cada operación con el cuerdo de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar] • Explicar la relación entre la herencia orientada a objetos (codigo compartido y *overriding*) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse] • Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar] • Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, selecionar la forma mas natural por cada lenguaje [Usar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 11: Templates and STL (6)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Class templates. • Basic concepts of the Standard Template Library (STL) including: vector, list, stack, queue.	• Understand the concepts of class templates. [Familiarizarse] • Implement and create new generic data types. [Usar] • Understand the basic structures of the STL. [Familiarizarse] • Use basic data structures like stack, queue, list, and vector from the STL. [Usar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 12: Operator Overloading (4)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Operator overloading definition.	• Understand the concepts of operator overloading. [Familiarizarse] • Implement the overloading of allowed operators in the programming language. [Usar]
Readings : [Str13], [Jos19], [Dei17]	

Unit 13: File Handling (4)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• File input and output (I/O).	• Understand the concepts of file manipulation. [Familiarizarse] • Create programs to read and write files. [Usar]
Readings : [Str13], [Jos19], [Dei17]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Str13] Bjarne Stroustrup. *The C++ Programming Language*. 4th. Addison-Wesley, 2013.

[Dei17] Deitel & Deitel. *C++17 - The Complete Guide*. 10th. Pearson, 2017.

[Jos19] Nicolai M. Josuttis. *C++17 - The Complete Guide*. 1st. 2019.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS1D2. Discrete Structures II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS1D2. Discrete Structures II
2.2 Semester	:	2^{nd} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS1D1. Discrete Structures I. (1^{st} Sem) CS1D1. Discrete Structures I. (1^{st} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

In order to understand the advanced computational techniques, the students must have a strong knowledge of the Various discrete structures, structures that will be implemented and used in the laboratory in the programming language..

5. GOALS

- That the student is able to model computer science problems using graphs and trees related to data structures.
- That the student applies efficient travel strategies to be able to search data in an optimal way.
- That the student uses the various counting techniques to solve computational problems.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Familiarity)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Familiarity)

7. TOPICS

Unit 1: Digital Logic and Data Representation (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Reticles: Types and properties. • Boolean algebras. • Boolean Functions and Expressions. • Representation of Boolean Functions: Normal Disjunctive and Conjunctive Form. • Logical gates. • Circuit Minimization.	• Explain the importance of Boolean algebra as a unification of set theory and propositional logic [Evaluar]. • Explain the algebraic structures of reticulum and its types [Evaluar]. • Explain the relationship between the reticulum and the ordinate set and the wise use to show that a set is a reticulum [Evaluar]. • Explain the properties that satisfies a Boolean algebra [Evaluar]. • Demonstrate if a terna formed by a set and two internal operations is or not Boolean algebra [Evaluar]. • Find the canonical forms of a Boolean function [Evaluar]. • Represent a Boolean function as a Boolean circuit using logic gates [Evaluar]. • Minimize a Boolean function. [Evaluar].
Readings : [Ros07], [Gri03]	

Unit 2: Fundamentos de conteo (40)	
Competences Expected:	
Topics	**Learning Outcomes**
• Técnicas de Conteo: – Conteo y cardinalidad de un conjunto – Regla de la suma y producto – Principio de inclusión-exclusión – Progresión geométrica y aritmética • Principio de las casillas. • Permutaciones y combinaciones: – Definiciones básicas – Identidad de Pascal – Teorema del binomio • Resolviendo relaciones de recurrencia: – Un ejemplo de una relación de recurrencia simple, como los números de Fibonacci – Otras ejemplos, mostrando una variedad de soluciones • Aritmetica modular basica	• Aplicar argumentos de conteo, incluyendo las reglas del producto y de la suma, principio de inclusión-exclusión y progresiones aritméticas/geométricas [Familiarizarse] • Aplicar el principio de las casillas en el contexto de una demostración formal [Familiarizarse] • Calcular permutaciones y combinaciones en un conjunto, e interpreta su significado en el contexto de una aplicación en particular [Familiarizarse] • Mapear aplicaciones del mundo real a formalismos de conteo adecuados, como el determinar el número de formas de acomodar a un conjunto de personas alrededor de una mesa, sujeto a restricciones en la disposición de los asientos, o en el número de maneras de determinar ciertas manos en juegos de cartas (ejm. una casa llena) [Familiarizarse] • Resolver una variedad de relaciones de recurrencia básicas [Familiarizarse] • Analizar un problema para determinar las relaciones de recurrencia implícitas [Familiarizarse] • Realizar cálculos que involucran aritmética modular [Familiarizarse]
Readings : [Gri97]	

Unit 3: Árboles y Grafos (40)	
Competences Expected:	
Topics	**Learning Outcomes**

• Árboles.
  – Propiedades
  – Estrategias de recorrido
• Grafos no dirigidos
• Grafos dirigidos
• Grafos ponderados
• Arboles de expansion/bosques.
• Isomorfismo en grafos.
	• Ilustrar mediante ejemplos la terminología básica de teoría de grafos, y de alguna de las propiedades y casos especiales de cada tipo de grafos/árboles [Familiarizarse]
	• Demostrar diversos métodos de recorrer árboles y grafos, incluyendo recorridos pre, post e inorden de árboles [Familiarizarse]
	• Modelar una variedad de problemas del mundo real en ciencia de la computación usando formas adecuadas de grafos y árboles, como son la representación de una topología de red o la organización jerárquica de un sistema de archivos [Familiarizarse]
	• Demuestrar como los conceptos de grafos y árboles aparecen en estructuras de datos, algoritmos, técnicas de prueba (inducción estructurada), y conteos [Familiarizarse]
	• Explicar como construir un árbol de expansión de un grafo [Familiarizarse]
	• Determinar si dos grafos son isomorfos [Familiarizarse]

Readings : [Joh99]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Gri97] R. Grimaldi. *Matemáticas Discretas y Combinatoria.* Addison Wesley Iberoamericana, 1997.

[Joh99] Richard Johnsonbaugh. *Matemáticas Discretas.* Prentice Hall, México, 1999.

[Gri03] R. Grimaldi. *Discrete and Combinatorial Mathematics: An Applied Introduction.* 5 ed. Pearson, 2003.

[Ros07] Kenneth H. Rosen. *Discrete Mathematics and Its Applications.* 7 ed. 2007.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

MA101. Math II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	MA101. Math II
2.2 Semester	:	2^{nd} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	MA100. Mathematics I. (1^{st} Sem) MA100. Mathematics I. (1^{st} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The course develops in students the skills to deal with models of science and engineering skills. In the first part of the course a study of the functions of several variables, partial derivatives, multiple integrals and an introduction to vector fields is performed. Then the student will use the basic concepts of calculus to model and solve ordinary differential equations using techniques such as Laplace transforms and Fourier series.

5. GOALS

- Apply derivation rules and partial differentation in functions of several variables.
- Apply techniques for calculating multiple integrals.
- Understand and use the concepts of vector calculus.
- Understand the importance of series.
- Identify and solve differential equations of the first order and their applications in chemical and physical problems.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Multi-Variable Function Differential (24)	
Competences Expected:	
Topics	**Learning Outcomes**
• Concept of multi-variable functions. • Directional Derivates • Tangent line, normal plane to curve line and tangent plane, normal line to a curve plan. Know to calculate their equations. • Concept of extreme value and conditional extreme value of multi-variable functions • Applications problems such as modeling total production of an economic system, speed of sound through the ocean, thickener optimization, etc.	• Understand the concept of multi-variable functions. • Master the concept and calculation method of the direction derivative and gradient of the guide. • Master the calculation method of the first order and second order partial derivative of composite functions. • Master the calculation method of the partial derivatives for implicit functions. • Understand tangent line, normal plane to curve line and tangent plane, normal line to a curve plan. Know to calculate their equations. • Learn the concept of extreme value and conditional extreme value of multi-variable functions; know to find out the binary function extreme value. • Be able to solve simple applications problems.
Readings : [Ste12], [Zil13]	

Unit 2: Multi-Variable function Integral (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Double integral, triple integral and nature of the multiple integral. • Method of double integral • Line Integral • The Divergence, Rotation and Laplacian	• Understand the double integral, triple integral, and understand the nature of the multiple integral. • Master the calculation method of double integral (Cartesian coordinates, polar coordinates) the triple integral (Cartesian coordinates, cylindrical coordinates, spherical coordinates). • Understand the concept of line Integral, their properties and relationships. • Know to calculate the line integral. • Master the calculation the rotational, divergence and Laplacian.
Readings : [Ste12], [Zil13]	

Unit 3: Series (24)	
Competences Expected:	
Topics	**Learning Outcomes**
• Convergent series • Taylor and McLaurin series • Orthogonal functions	• Master to calculation if series is convergent, and if convergent, find the sum of the series trying to find the radius of convergence and the interval of convergence of a power series. • Represent a function as a power series and find the Taylor and McLaurin Series to estimate function values to a desired accuracy. • Understand the concepts of orthogonal functions and the expansion of a given function f to find its Fourier series.
Readings : [Ste12], [Zil13]	

Unit 4: Ordinary Differential Equations (30)	
Competences Expected:	
Topics	**Learning Outcomes**
• Concept of differential equations • Methods to resolve differential equations • Methods to resolve the secod order linear differential equations • Higher order linear ordinary differential equations • Applications problems using Laplace transforms	• Understand differential equations, solutions, order, general solution, initial conditions and special solutions etc. • Master the calculation method for variables separable equation and first order linear equations. Known to solve homogeneous equation and Bernoulli (Bernoulli) equations; understand variable substitution to solve the equation. • Master to solve total differential equations. • Be able to use reduced order method to solve equations. • Understand the structure of the second order linear differential equation. • Master calculation method for the constant coefficient homogeneous linear differential equations; and understand calculation method for the higher order homogeneous linear differential equations. • Know to apply the differential equation calculation method to solve simple geometric and physic application problems. • Solve properly certain types of differential equations using Laplace transforms.
Readings : [Ste12], [Zil13]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Ste12] James Stewart. *Calculus*. 7th. CENGAGE Learning, 2012.

[Zil13] Dennis G. Zill. *Differential equations with Boundary value problems*. 8th. CENGAGE Learning, 2013.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS113. Computer Science II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS113. Computer Science II
2.2 Semester	:	3^{rd} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS112. Computer Science I. (2^{nd} Sem) CS112. Computer Science I. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the third course in the sequence of introductory courses in computer science. This course is intended to cover Concepts indicated by the Computing Curriculum IEEE (c) -ACM 2001, under the functional-first approach. The object-oriented paradigm allows us to combat complexity by making models from abstractions of the problem elements and using techniques such as encapsulation, modularity, polymorphism and inheritance. The Dominion of these topics will enable participants to provide computational solutions to design problems simple of the real world.

5. GOALS

- Introduce the student in the fundaments of the paradigm of object orientation, allowing the assimilation of concepts necessary to develop an information system

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

3) Communicate effectively in a variety of professional contexts.. (Usage)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

7. TOPICS

Unit 1: Advanced STL (8)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Associative Containers (std::set, std::map, std::unordered_set, std::unordered_map). • Adapters (std::stack, std::queue, std::priority_queue). • Advanced STL Algorithms. • Functors and Predicates.	• Understand the use of associative containers. [Usar] • Implement programs that use STL adapters. [Usar] • Apply advanced STL algorithms. [Usar] • Use functors and predicates with the STL. [Usar]
Readings : [**Stroustrup2013**], [MJo19], [**Deitel17**]	

Unit 2: Advanced Templates (7)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Template Metaprogramming. • SFINAE (Substitution Failure Is Not An Error). • Perfect Forwarding.	• Apply template metaprogramming to solve complex problems. [Usar] • Understand and use SFINAE for template selection. [Usar] • Use Perfect Forwarding for efficient argument passing. [Usar]
Readings : [**Stroustrup2013**], [MJo19], [**Deitel17**]	

Unit 3: Variadic Templates (12)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Introduction to Variadic Templates. • Variadic Template Functions. • Variadic Template Methods. • Variadic Template Classes. • Classes inheriting from variable lists of variadic templates. • Example: Implementing a custom tuple class.	• Understand the concept of variadic templates. [Familiarizarse] • Implement variadic functions. [Usar] • Design classes with variadic methods. [Usar] • Create classes with variadic templates. [Usar] • Implement inheritance with variable lists of variadic templates. [Usar] • Apply variadic templates to solve real-world problems. [Usar]
Readings : [**Stroustrup2013**], [MJo19], [**Deitel17**]	

Unit 4: Move Semantics and Rvalue References (5)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Lvalues and Rvalues. • Rvalue References. • Move Semantics. • Move Constructors and Move Assignment Operators. • Perfect Forwarding (review).	• Explain move semantics and its purpose in C++. [Familiarizarse] • Define and use rvalue references. [Usar] • Analyze the performance implications of using move semantics. [Evaluar] • Implement move constructors and move assignment operators for custom classes. [Usar] • Apply move semantics to optimize resource management in C++ programs. [Usar]
Readings : [**Stroustrup2013**], [MJo19], [**Deitel17**]	

Unit 5: Design Patterns (Creational and Structural) (6)	
Competences Expected: 1	
Topics	**Learning Outcomes**
• Singleton, Factory, Builder. • Adapter, Decorator, Facade.	• Understand and apply creational design patterns: Singleton, Factory, Builder. [Usar] • Understand and apply structural design patterns: Adapter, Decorator, Facade. [Usar]
Readings : [**Stroustrup2013**], [MJo19], [**Deitel17**]	

Unit 6: Functors (3)	
Competences Expected: 1,3	
Topics	**Learning Outcomes**
• Definition of Functors. • Functors and Templates. • Passing Functors to Functions using parameters. • Passing Functors to Functions using templates. • Passing Functors to Classes using parameters. • Passing Functors to Classes using templates. • Examples and Applications.	• Introduction to Functors. [Usar] • Using Functors as parameters to functions and classes. [Usar] • Using Functors in functions and classes through templates. [Usar]
Readings : [**Stroustrup2013**], [MJo19], [**Deitel17**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[MJo19] Nicolai M.Josuttis. *C++17-The Complete Guide*. 1st. 2019.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS221. Computer Systems Architecture (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS221. Computer Systems Architecture
2.2 Semester	:	3^{rd} Semester.
2.3 Credits	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS1D2. Discrete Structures II. (2^{nd} Sem)

CS1D2. Discrete Structures II. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

A computer scientist must have a solid knowledge of the organization and design principles of diverse computer systems, by understanding the limitations of modern systems they could propose next-gen paradigms. This course teaches the basics and principles of Computer Architecture. This class addreses digital logic design, basics of Computer Architecture and processor design (Instruction Set architecture, microarchitecture, out-of-order execution, branch prediction), execution paradigms (superscalar, dataflow, VLIW, SIMD, GPUs, systolic, multithreading) and memory system organization.

5. GOALS

- Provide a first approach in Computer Architecture.
- Study the design and evolution of computer architectures, which lead to modern approaches and implementations in computing systems.
- Provide fine-grained details of computer hardware, and its relation with software execution.
- Implement a simple microprocessor using Verilog language.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Lógica digital y sistemas digitales (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Revisión e historia de la Arquitectura de Computadores. • Lógica combinacional vs. secuencial/Arreglos de puertas de campo programables como bloque fundamental de construcción lógico combinacional-secuencial. • Multiples representaciones / Capas de interpretación (El hardware es solo otra capa) • Herramientas de diseño asistidas por computadora que procesan hardware y representaciones arquitecturales. • Registrar transferencia notación / Hardware lenguage descriptivo (Verilog/VHDL) • Restriccion física (Retrasos de Entrada, fan-in, fan-out, energia/poder)	• Describir el avance paulatino de los componentes de la tecnología de computación, desde los tubos de vacío hasta VLSI, desde las arquitecturas mainframe a las arquitecturas en escala warehouse [Familiarizarse] • Comprender que la tendencia de las arquitecturas modernas de computadores es hacia núcleos múltiples y que el paralelismo es inherente en todos los sistemas de hardware [Usar] • Explicar las implicancias de los límites de potencia para mejoras adicionales en el rendimiento de los procesadores y también en el aprovechamiento del paralelismo [Usar] • Relacionar las varias representaciones equivalentes de la funcionalidad de un computador, incluyendo expresiones y puertas lógicas, y ser capaces de utilizar expresiones matemáticas para describir las funciones de circuitos combinacionales y secuenciales sencillos [Familiarizarse] • Diseñar los componentes básicos de construcción de un computador: unidad aritmético lógica (a nivel de puertas lógicas), unidad central de procesamiento (a nivel de registros de transferencia), memoria (a nivel de registros de transferencia) [Usar] • Usar herramientas CAD para capturar, sistetizar, y simular bloques de construcción (como ALUs, registros, movimiento entre registros) de un computador simple [Familiarizarse] • Evaluar el comportamiento de un diagrama de tiempos y funcional de un procesador simple implementado a nivel de circuitos lógicos [Evaluar]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

Unit 2: Representación de datos a nivel máquina (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Bits, Bytes y Words. • Representacion de datos numérica y bases numéricas. • Sistemas de punto flotante y punto fijo. • Representaciones con signo y complemento a 2. • Representación de información no numérica (códigos de caracteres, información gráfica) • Representación de registros y arreglos.	• Explicar porqué en computación todo es datos, inclusive las instrucciones [Evaluar] • Explicar las razones de usar formatos alternativos para representar datos numéricos [Familiarizarse] • Describir cómo los enteros negativos se almacenan con representaciones de bit de signo y complemento a 2 [Usar] • Explicar cómo las representaciones de tamaño fijo afectan en la exactitud y la precisión [Usar] • Describir la representación interna de datos no numéricos como caracteres, cadenas, registros y arreglos [Usar] • Convertir datos numéricos de un formato a otro [Usar]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

Unit 3: Organización de la Máquina a Nivel Ensamblador (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Organización Básica de la Máquina de Von Neumann. • Unidad de Control. • Paquetes de instrucciones y tipos (manipulación de información, control, I/O) • Assembler / Programación en Lenguaje de Máquina. • Formato de instrucciones. • Modos de direccionamiento. • Llamada a subrutinas y mecanismos de retorno. • I/O e Interrupciones. • Montículo (Heap) vs. Estático vs. Pila vs. Segmentos de código.	• Explicar la organización de la maquina clásica de von Neumann y sus principales unidades funcionales [Familiarizarse] • Describir cómo se ejecuta una instrucción en una máquina de von Neumann con extensión para hebras, sincronización multiproceso y ejecucion SIMD (máquina vectorial) [Familiarizarse] • Describir el paralelismo a nivel de instrucciones y sus peligros, y cómo es esto tratado en pipelines de proceso típicos [Familiarizarse] • Resumir cómo se representan las instrucciones, tanto a nivel de máquina bajo el contexto de un ensamblador simbólico [Familiarizarse] • Demostrar cómo se mapean los patrones de lenguajes de alto nivel en notaciones en lenguaje ensamblador o en código máquina [Usar] • Explicar los diferentes formatos de instrucciones, así como el direccionamiento por instrucción, y comparar formatos de tamaño fijo y variable [Usar] • Explicar como las llamadas a subrutinas son manejadas a nivel de ensamblador [Usar] • Explicar los conceptos básicos de interrupciones y operaciones de entrada y salida (I/O) [Familiarizarse] • Escribir segmentos de programa simples en lenguaje ensamblador [Usar] • Ilustrar cómo los bloques constructores fundamentales en lenguajes de alto nivel son implementados a nivel de lenguaje máquina [Usar]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

Unit 4: Organización funcional (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Implementación de rutas de datos simples, incluyendo la canalización de instrucciones, detección de riesgos y la resolución. • Control de unidades: Realización Cableada vs Realización Microprogramada. • Instruccion (Pipelining) • Introducción al paralelismo al nivel de instrucción (PNI)	• Comparar implementaciones alternativas de ruta de datos [Evaluar] • Discutir el concepto de puntos de control y la generación de señales de control usando implementaciones a nivel de circuito o microprogramadas [Familiarizarse] • Explicar el paralelismo a nivel de instrucciones básicas usando pipelining y los mayores riesgos que pueden ocurrir [Usar] • Diseñar e implementar un procesador completo, incluyendo ruta de datos y control [Usar] • Calcular la cantidad promedio de ciclos por instrucción de una implementación con procesador y sistema de memoria determinados [Evaluar]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

Unit 5: Organización y Arquitectura del Sistema de Memoria (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Sistemas de Almacenamiento y su Tecnología. • Jerarquía de Memoria: importancia de la localización temporal y espacial. • Organización y Operaciones de la Memoria Principal. • Latencia, ciclos de tiempo, ancho de banda e intercalación. • Memorias caché (Mapeo de direcciones, Tamaño de bloques, Reemplazo y Politicas de almacenamiento) • Multiprocesador coherencia cache / Usando el sistema de memoria para las operaciones de sincronización de memoria / atómica inter-core. • Memoria virtual (tabla de página, TLB) • Manejo de Errores y confiabilidad. • Error de codificación, compresión de datos y la integridad de datos.	• Identifique las principales tecnologías de memoria (Por ejemplo: SRAM, DRAM, Flash,Disco Magnetico) y su relación costo beneficio [Familiarizarse] • Explique el efecto del retardo de la memoria en tiempo de ejecución [Familiarizarse] • Describa como el uso de jerarquía de memoria (caché, memoria virtual) es aplicado para reducir el retardo efectivo en la memoria [Usar] • Describa como el uso de jerarquía de memoria (caché, memoria virtual) es aplicado para reducir el retardo efectivo en la memoria [Usar] • Explique el efecto del retardo de la memoria en tiempo de ejecución [Usar] • Calcule el tiempo de acceso promedio a memoria bajo varias configuraciones de caché y memoria y para diversas combinaciones de instrucciones y referencias a datos [Evaluar]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

Unit 6: Interfaz y comunicación (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Fundamentos de I/O: Handshaking, Bbuffering, I/O programadas, interrupciones dirigidas de I/O. • Interrumpir estructuras: interrumpir reconocimiento, vectorizado y priorizado. • Almacenamiento externo, organización fisica y discos. • Buses: Protocoles de bus, arbitraje, acceso directo a memoria (DMA). • Introducción a Redes: comunicación de redes como otra capa de acceso remoto. • Soporte Multimedia. • Arquitecturas RAID.	• Explicar como las interrupciones son aplicadas para implementar control de entrada-salida y transferencia de datos [Familiarizarse] • Identificar diversos tipos de buses en un sistema computacional [Familiarizarse] • Describir el acceso a datos desde una unidad de disco magnético [Usar] • Comparar organizaciones de red conocidas como organizaciones en bus/Ethernet, en anillo y organizaciones conmutadas versus ruteadas [Evaluar] • Identificar las interfaces entre capas necesarios para el acceso y presentación multimedia, desde la captura de la imagen en almacenamiento remoto, a través del transporte por una red de comunicaciones, hasta la puesta en la memoria local y la presentación final en una pantalla gráfica [Familiarizarse] • Describir las ventajas y limitaciones de las arquitecturas RAID [Familiarizarse]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

Unit 7: Multiprocesamiento y arquitecturas alternativas (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Ley potencial. • Ejemplos de juego de instrucciones y arquitecturas SIMD y MIMD. • Redes de interconexión (Hypercube, Shuffle-exchange, Mesh, Crossbar) • Sistemas de memoria de multiprocesador compartido y consistencia de memoria. • Coherencia de cache multiprocesador.	• Discutir el concepto de procesamiento paralelo mas allá del clásico modelo de von Neumann [Evaluar] • Describir diferentes arquitecturas paralelas como SIMD y MIMD [Familiarizarse] • Explicar el concepto de redes de interconexión y mostrar diferentes enfoques [Usar] • Discutir los principales cuidados en los sistemas de multiprocesamiento presentes con respecto a la gestión de memoria y describir como son tratados [Familiarizarse] • Describir las diferencias entre conectores electricos en paralelo backplane, interconexión memoria procesador y memoria remota via red, sus implicaciones para la latencia de acceso y el impacto en el rendimiento de un programa [Evaluar]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

Unit 8: Mejoras de rendimiento (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Arquitectura superescalar. • Predicción de ramificación, Ejecución especulativa, Ejecución fuera de orden. • Prefetching. • Procesadores vectoriales y GPU's • Soporte de hardware para multiprocesamiento. • Escalabilidad. • Arquitecturas alternativas, como VLIW / EPIC y aceleradores y otros tipos de procesadores de propósito especial.	• Describir las arquitecturas superescalares y sus ventajas [Familiarizarse] • Explicar el concepto de predicción de bifurcaciones y su utilidad [Usar] • Caracterizar los costos y beneficios de la precarga prefetching [Evaluar] • Explicar la ejecución especulativa e identifique las condiciones que la justifican [Evaluar] • Discutir las ventajas de rendimiento ofrecida en una arquitectura de multihebras junto con los factores que hacen dificil dar el maximo beneficio de estas [Evaluar] • Describir la importancia de la escalabilidad en el rendimiento [Evaluar]
Readings : [HH12], [PP05], [PH04], [JAs07], [HP06], [Par05], [Sta10], [PCh06]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[PH04] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. 3rd ed. San Mateo, CA: Morgan Kaufman, 2004.

[Par05] Behrooz Parhami. *Computer Architecture: From Microprocessors to Supercomputers*. New York: Oxford Univ. Press, 2005.

[PP05] Yale N Patt and Sanjay J Patel. *Introduction to Computing Systems*. 2nd. McGraw Hill, 2005.

[HP06] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. 4th. San Mateo, CA: Morgan Kaufman, 2006.

[PCh06] Pong P.Chu. *RTL Hardware Design Using VHDL*. 1st. Wiley-Interscience, 2006.

[JAs07] Peter J.Ashenden. *Digital Design (Verilog): An Embedded Systems Approach Using Verilog*. Morgan Kaufmann, 2007.

[Sta10] William Stalings. *Computer Organization and Architecture: Designing for Performance*. 8th. Upper Saddle River, NJ: Prentice Hall, 2010.

[HH12] David Harris and Sarah Harris. *Digital Design and Computer Architecture*. 2nd. Morgan Kaufmann, 2012.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS2B1. Platform Based Development (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS2B1. Platform Based Development
2.2 Semester	:	3^{rd} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS112. Computer Science I. (2^{nd} Sem) CS112. Computer Science I. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The world has changed due to the use of fabric and related technologies, rapid, timely and personalized access to the information, through web technology, ubiquitous and pervasive; they have changed the way we do things, how do we think? and how does the industry develop? Web technologies, ubiquitous and pervasive are based on the development of web services, web applications and mobile applications, which are necessary to understand the architecture, design, and implementation of web services, web applications and mobile applications.

5. GOALS

- That the student is able to design and implement services, web applications using tools and languages such as HTML, CSS, JavaScript (including AJAX), back-end scripting and a database, at an intermediate level.
- That the student is able to develop mobile applications, administration of web servers in a Unix system and an introduction to web security, at an intermediate level.

6. COMPETENCES

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

3) Communicate effectively in a variety of professional contexts.. (Usage)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Usage)

7. TOPICS

Unit 1: Introducción (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Visión general de plataformas (ejemplo, Web, Mobil, Juegos, Industrial) • Programacíon a través de APIs específicos. • Visión general de lenguajes de plataforma (ejemplo, Objective C, HTML5) • Pogramacíón bajo restricciónes de plataforma.	• Describir cómo el desarrollo basado en plataforma difiere de la programación de proposito general [Familiarizarse] • Listar las características de lenguajes de plataforma [Familiarizarse] • Escribir y ejecutar un programa simple basado en plataforma [Familiarizarse] • Listar las ventajas y desventajas de la programación con restricciones de plataforma [Familiarizarse]
Readings : [**fielding2000fielding**], [**grove2009web**], [**annuzzi2013introduction**], [**Cornez2015**]	

Unit 2: Plataformas web (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Lenguajes de programación web (e.g., HTML5, Javascript, PHP, CSS) • • Web Platform constraints: Client-Server, Stateless-Stateful, Cache, Uniform Interface, Layered System, Code on Demand, ReST. • Restricción de plataformas web. • Software como servicio. • Estándares web.	• Diseñar e implementar una aplicación web sencilla [Familiarizarse] • Describir las limitaciones que la web pone a los desarrolladores [Familiarizarse] • Comparar y contrastar la programación web con la programación de proposito general [Familiarizarse] • Describir las diferencias entre software como un servicio y productos de software tradicionales [Familiarizarse] • Discutir cómo los estándares de web impactan el desarrollo de software [Familiarizarse] • Revise una aplicación web existente con un estándar web actual [Familiarizarse]
Readings : [**fielding2000fielding**]	

Unit 3: Desarrollo de servicios y aplicaciones web (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• Describe, identify and debug issues related to web application development • Design and development of interactive web applications using HTML5 and Python • Use MySQL for data management and manipulate MySQL with Python • Design and development of asynchronous web applications using Ajax techniques • Using dynamic client side Javascript scripting language and server side python scripting language with Ajax • Apply XML / JSON technologies for data management with Ajax • Use framework, services and Ajax web APIs and apply design patterns to web application development	• Server-side python scripting language: variables, data types, operations, strings, functions, control statements, arrays, files and directory access, maintain state. [Usar] • Web programming approach using embedded python. [Usar] • Accessing and Manipulating MySQL. [Usar] • The Ajax web application development approach. [Usar] • DOM and CSS used in JavaScript. [Usar] • Asynchronous Content Update Technologies. [Usar] • XMLHttpRequest objects use to communicate between clients and servers. [Usar] • XML and JSON. [Usar] • XSLT and XPath as mechanisms for transforming XML documents. [Usar] • Web services and APIs (especially Google Maps). [Usar] • Macros Ajax for the development of contemporary web applications. [Usar] • Design patterns used in web applications. [Usar]
Readings : [**freeman2011head**]	

Unit 4: Plataformas móviles (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Lenguajes de Programación para Móviles. • Design Principles: Segregation of Interfaces, Single Responsability, Separation of concerns, Dependency Inversion. • Desafíos con mobilidad y comunicación inalámbrica. • Aplicaciones Location-aware. • Rendimiento / Compensación de Potencia. • Restricciones de las Plataformas Móviles. • Tecnologías Emergentes.	• Diseñar e implementar una aplicación móvil para una plataforma móvil dada [Familiarizarse] • Discutir las limitaciones que las plataformas móviles ponen a los desarrolladores [Familiarizarse] • Discutir el rendimiento vs perdida de potencia [Familiarizarse] • Compare y contraste la programación móvil con la programación de proposito general [Familiarizarse]
Readings : [**martin2017clean**], [**annuzzi2013introduction**]	

Unit 5: Mobile Applications for Android Handheld Systems (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• The Android Platform • The Android Development Environment • Application Fundamentals • The Activity Class • The Intent Class • Permissions • The Fragment Class • User Interface Classes • User Notifications • The BroadcastReceiver Class • Threads, AsyncTask & Handlers • Alarms • Networking (http class) • Multi-touch & Gestures • Sensors • Location & Maps	• Students identify necessary software and install it on their personal computers. • Students perform various tasks to familiarize themselves with the Android platform and Environment for development. [Usar] • Students build applications that trace the lifecycle callback methods emitted by the Android platform and demonstrate the behavior of Android when device configuration changes (for example, when the device moves from vertical to horizontal and vice versa). [Usar] • Students build applications that require starting multiple activities through both standard and custom methods. [Usar] • Students build applications that require standard and custom permissions. [Usar] • Students build an application that uses a single code base, but creates different user interfaces depending on the screen size of a device. [Usar] • Students construct a to-do list manager using the user interface elements discussed in class. The application allows users to create new items and to display them in a ListView. [Usar] • Students build an application that uses location information to collect latitude, length of places they visit. [Usar]
Readings : [**annuzzi2013introduction**], [**Cornez2015**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

MA102. Calculus I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	MA102. Calculus I
2.2 Semester	:	3^{rd} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	MA100. Mathematics I. (1^{st} Sem) MA100. Mathematics I. (1^{st} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course introduces the first concepts of linear algebra as well as numerical methods with an emphasis on problem solving with the Scilab open source libe package. Mathematical theory is limited to fundamentals, while effective application for problem solving is privileged. In each subject, a few methods of relevance for engineering are taught. Knowledge of these methods prepares students for the search for more advanced alternatives, if required.

5. GOALS

- Ability to apply knowledge about Mathematics.
- Ability to apply engineering knowledge.
- Ability to apply the modern knowledge, techniques, skills and tools of modern engineering to the practice of engineering

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Introduction (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Importance of linear algebra and numerical methods. Examples.	• Be able to understand the basic concepts and importance of Linear Algebra and Numerical Methods.
Readings : [AR14], [CC15]	

Unit 2: Linear Algebra (14)	
Competences Expected:	
Topics	**Learning Outcomes**
• Elementary matrix algebra and determinants • Null space and exact solutions of systems of linear equations Ax=b: – Tridiagonal and triangular systems and Gaussian elimination with and without pivoting. – LU factorization and Crout algorithm. • Basics on eigenvalues and eigenvectors: – Characteristic polynomials. – Algebraic and geometric multiplicities. • Least squares estimation. • Linear transformations.	• Understanding the basics concepts of Linear Algebra. • Solve properly linear transformations problems.
Readings : [AR14], [CC15]	

Unit 3: Numerical methods (22)	
Competences Expected:	
Topics	**Learning Outcomes**
• Basics on solutions of systems of linear equations Ax=b: Jacobi and Gauss Seidel methods. • Application of matrix factorizations to the solution of linear systems (singular value decomposition, QR, Cholesky) Numerical computation of null space, rank and condition number. • Root finding: – Bisection. – Fixed-point iteration. – Newton-Raphson methods. • Basics on interpolation: – Newton and Lagrange polynomial interpolations – Spline interpolation • Basics on numerical differentiation and Taylor approximation • Basics on numerical integration: – Trapezium, midpoint and Simpson rule – Gaussian quadrature • Basics on numerical solutions to ODEs: – Finite differences; Euler and Runge-Kutta methods – Converting higher order ODEs into a system of low order ODEs – Runge-Kutta methods for systems of equations – Single shooting method • Short introduction to optimization techniques: overview on linear programming, bounded linear systems, quadratic programming, gradient descent.	• Understanding the basics concepts of Numerical Methods. • Applying the most frequent methods for the resolution of mathematical problems. • Implementing and applying numerical algorithms for the solution of mathematical problems using the Scilab open-source computational package. • Applying Scilab for the solution of mathematical problems and for plotting graphs.
Readings : [AR14], [CC15]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[AR14] H. Anton and C. Rorres. *Elementary Linear Algebra, Applications Version.* 11th. Wiley, 2014.

[CC15] S.C. Chapra and R.P. Canale. *Numerical Methods for Engineers,* 7th. Vol. 1. McGraw-Hill, 12015.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS210. Algorithms and Data Structures (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS210. Algorithms and Data Structures
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS113. Computer Science II. (3^{rd} Sem) CS113. Computer Science II. (3^{rd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The theoretical foundation of all branches of computing rests on algorithms and data structures, this course will provide participants with an introduction to these topics, thus forming a basis that will serve for the following courses in the career.

5. GOALS

- Make the student understand the importance of algorithms for solving problems.
- Introduce the student to the field of application of data structures.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Graphs (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Graph Concept • Directed Graphs and Non-directed Graphs. • Using Graphs. • Measurement of efficiency ,in time and space. • Adjacency matrices. • Tag adjacent matrices. • Adjacency Lists. • Implementation of graphs using adjacency matrices. • Graph Implementation using adjacency lists • Insertion, search and deletion of nodes and edges. • Graph search algorithms.	• Acquire Dexterity to Perform Correct Implementation. [Usar] • Develop knowledge to decide when it is better to use one implementation technique than another. [Usar]
Readings : [Cor+09], [Fag+14], [Knu97], [Knu98]	

Unit 2: Scatter Matrices (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Initial concepts. • Dense Matrices • Measurement of Efficiency in Time and Space • Static scatter vs. dynamic matrix creation. • Insert, search, and delete methods.	• Understand the use and implementation of scatter matrices.[Evaluar]
Readings : [Cor+09], [Fag+14], [Knu97], [Knu98]	

Unit 3: Balanced Trees (16)	
Competences Expected:	
Topics	**Learning Outcomes**
• AVL Trees. • Measurement of Efficiency. • Simple and Composite Rotations • Insertion, deletion and search. • Trees B , B+ B* y Patricia.	• Understand the basic functions of these complex structures in order to acquire the capacity for their implementation. [Evaluar]
Readings : [Cor+09], [Fag+14], [Knu97], [Knu98]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. 3rd. Addison-Wesley Professional, 1997.

[Knu98] Donald E. Knuth. *The art of computer programming, volume 3:Sorting and searching*. 2nd. Addison-Wesley Professional, 1998.

[Cor+09] Thomas H. Cormen et al. *Introduction to Algorithms*. Third Edition. ISBN: 978-0-262-53305-8. MIT Press, 2009.

[Fag+14] José Fager et al. *Estructura de datos*. First Edition. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIN), 2014.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS211. Theory of Computation (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS211. Theory of Computation
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS1D2. Discrete Structures II. (2^{nd} Sem) CS1D2. Discrete Structures II. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course emphasizes formal languages, computer models and computability, as well as the fundamentals of computational complexity and complete NP problems.

5. GOALS

- That the student learn the fundamental concepts of the theory of formal languages.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Computabilidad y complejidad básica de autómatas (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Máquinas de estado finito. • Expresiones regulares. • Problema de la parada. • Gramáticas libres de contexto. • Introducción a las clases P y NP y al problema P vs. NP. • Introducción y ejemplos de problemas NP- Completos y a clases NP-Completos. • Máquinas de Turing, o un modelo formal equivalente de computación universal. • Máquinas de Turing no determinísticas. • Jerarquía de Chomsky. • La tesis de Church-Turing. • Computabilidad. • Teorema de Rice. • Ejemplos de funciones no computables. • Implicaciones de la no-computabilidad.	• Discute el concepto de máquina de estado finito [Evaluar] • Diseñe una máquina de estado finito determinista para aceptar un determinado lenguaje [Evaluar] • Genere una expresión regular para representar un lenguaje específico [Evaluar] • Explique porque el problema de la parada no tiene solucion algorítmica [Evaluar] • Diseñe una gramática libre de contexto para representar un lenguaje especificado [Evaluar] • Define las clases P y NP [Evaluar] • Explique el significado de NP-Completitud [Evaluar] • Explica la tesis de Church-Turing y su importancia [Familiarizarse] • Explica el teorema de Rice y su importancia [Familiarizarse] • Da ejemplos de funciones no computables [Familiarizarse] • Demuestra que un problema es no computable al reducir un problema clásico no computable en base a él [Familiarizarse]
Readings : [Mar10], [Lin11], [Sip12]	

Unit 2: Complejidad Computacional Avanzada (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Revisión de las clases P y NP; introducir spacio P y EXP. • Jerarquía polimonial. • NP completitud (Teorema de Cook). • Problemas NP completos clásicos. • Técnicas de reducción.	• Define la clase P-Space y su relación con la clase EXP [Evaluar] • Define la clase P-Space y su relación con la clase EXP [Evaluar] • Explique el significado de NP-Completo (También aparece en AL / Automata Básico, Computalidad y Complejidad) [Evaluar] • Muestre ejemplos de problemas clásicos en NP - Completo [Evaluar] • Pruebe que un problema es NP- Completo reduciendo un problema conocido como NP-Completo [Evaluar]
Readings : [Mar10], [Lin11], [Sip12], [HU13]	

Unit 3: Teoría y Computabilidad Avanzada de Autómatas (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Conjuntos y Lenguajes: – Lenguajes Regulares. – Revisión de autómatas finitos determinísticos (Deterministic Finite Automata DFAs) – Autómata finito no determinístico (Nondeterministic Finite Automata NFAs) – Equivalencia de DFAs y NFAs. – Revisión de expresiones regulares; su equivalencia con autómatas finitos. – Propiedades de cierre. – Probando no-regularidad de lenguajes, a través del lema de bombeo (Pumping Lemma) o medios alternativos. • Lenguajes libres de contexto: – Autómatas de pila (Push-down automata (PDAs) – Relación entre PDA y gramáticas libres de contexto. – Propiedades de los lenguajes libres de contexto.	• Determina la ubicación de un lenguaje en la jerarquía de Chomsky (regular, libre de contexto, enumerable recursivamente) [Evaluar] • Convierte entre notaciones igualmente poderosas para un lenguaje, incluyendo entre estas AFDs, AFNDs, expresiones regulares, y entre AP y GLCs [Evaluar]
Readings : [HU13], [Bro93]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Bro93] J. Glenn Brookshear. *Teoría de la Computación*. Addison Wesley Iberoamericana, 1993.

[Mar10] John Martin. *Introduction to Languages and the Theory of Computation*. 4th. McGraw-Hill, 2010.

[Lin11] Peter Linz. *An Introduction to Formal Languages and Automata*. 5th. Jones & Bartlett Learning, 2011.

[Sip12] Michael Sipser. *Introduction to the Theory of Computation*. 3rd. Cengage Learning, 2012.

[HU13] John E. Hopcroft and Jeffrey D. Ullman. *Introducción a la Teoría de Autómatas, Lenguajes y Computación*. Pearson Education, 2013.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS271. Data Management (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS271. Data Management
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- CS112. Computer Science I. (2^{nd} Sem)
- CS1D2. Discrete Structures II. (2^{nd} Sem)
- CS112. Computer Science I. (2^{nd} Sem)
- CS1D2. Discrete Structures II. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Information management (IM) plays a major role in almost all areas where computers are used. This area includes the capture, digitization, representation, organization, transformation and presentation of information; Algorithms to improve the efficiency and effectiveness of accessing and updating stored information, data modeling and abstraction, and physical file storage techniques.It also covers information security, privacy, integrity and protection in a shared environment. Students need to be able to develop conceptual and physical data models, determine which (IM) methods and techniques are appropriate for a given problem, and be able to select and implement an appropriate IM solution that reflects all applicable restrictions, including Scalability and usability.

5. GOALS

- That the student learn to represent information in a database prioritizing the efficiency in the recovery of the same.
- That the student learn the fundamental concepts of the management of databases. This includes the design of databases, database languages and the realization of databases.
- Discuss the database model with the base in relational algebra, relational calculus and the study of SQL statements.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

3) Communicate effectively in a variety of professional contexts.. (Usage)

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Usage)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Usage)

7. TOPICS

Unit 1: Sistemas de Bases de Datos (14)	
Competences Expected:	
Topics	**Learning Outcomes**
• Enfoque y Evolución de Sistemas de Bases de Datos. • Componentes del Sistema de Bases de Datos. • Diseño de las funciones principales de un DBMS. • Arquitectura de base de datos e independencia de datos. • Uso de un lenguaje de consulta declarativa. • Sistemas de apoyo a contenido estructurado y / o corriente. • Enfoques para la gestión de grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce).	• Explica las características que distinguen un esquema de base de datos de aquellos basados en la programación de archivos de datos [Usar] • Describe los componentes de un sistema de bases datos y da ejemplos de su uso [Usar] • Cita las metas básicas, funciones y modelos de un sistema de bases de datos [Usar] • Describe los componentes de un sistema de bases datos y da ejemplos de su uso [Usar] • Identifica las funciones principales de un SGBD y describe sus roles en un sistema de bases de datos [Usar] • Explica las características que distinguen un esquema de base de datos de aquellos basados en la programación de archivos de datos [Usar] • Usa un lenguaje de consulta declarativo para recoger información de una base de datos [Usar] • Describe las capacidades que las bases de datos brindan al apoyar estructuras y/o la secuencia de flujo de datos, ejm. texto [Usar] • Describe los enfoques principales para almacenar y procesar larges volúmenes de datos [Usar]
Readings : [RC04], [EN04], [RG03], [ER15], [CJ11], [KS02]	

Unit 2: Modelado de datos (14)	
Competences Expected:	
Topics	**Learning Outcomes**
• Modelado de datos • Modelos conceptuales (e.g., entidad-relación, diagramas UML) • Modelos de hoja de cálculo • Modelos Relacionales. • Modelos orientados a objetos. • Modelos de datos semi-estructurados (expresados usando DTD o XML Schema, por ejemplo)	• Compare y contrasta modelos apropiados de datos, incluyendo estructuras sus estructuras internas, para diversos tipos de datos [Usar] • Describe los conceptos en notación de modelos (ejm. Diagramas Entidad-Relación o UML) y cómo deben de ser usados [Usar] • Define la terminología fundamental a ser usada en un modelo relacional de datos [Usar] • Describe los principios básicos del modelo relacional de datos [Usar] • Aplica los conceptos de modelado y la notación de un modelo relacional de datos [Usar] • Describe los principios básicos del modelo relacional de datos [Usar] • Describe los principios básicos del modelo relacional de datos [Usar] • Da una semi estructura equivalente (ejm. en DTD o Esquema XML) para un esquema relacional dado [Usar]
Readings : [SW04], [EN04], [KS02]	

Unit 3: Indexación (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• El impacto de indices en el rendimiento de consultas. • La estructura basica de un indice. • Mantener un buffer de datos en memoria. • Creando indices con SQL. • Indexando texto. • Indexando la web (e.g., web crawling)	• Generar un archivo índice para una colección de recursos [Usar] • Explicar la función de un índice invertido en la localización de un documento en una colección [Usar] • Explicar cómo rechazar y detener palabras que afectan a la indexación [Usar] • Identificar los índices adecuados para determinado el esquema relacional y el conjunto de consultas [Usar] • Estimar el tiempo para recuperar información, cuando son usados los índices comparado con cuando no son usados [Usar] • Describir los desafíos claves en el rastreo web, por ejemplo, la detección de documentos duplicados, la determinación de la frontera de rastreo [Usar]
Readings : [WM01], [RG03], [ER15], [CJ11], [KS02]	

Unit 4: Bases de Datos Relacionales (14)	
Competences Expected:	
Topics	**Learning Outcomes**
• Mapeo de esquemas conceptuales a esquemas relacionales. • Entidad y integridad referencial. • Algebra relacional y calculo relacional. • Diseño de bases de datos relacionales. • Dependencia funcional. • Descomposición de un esquema. • Llaves candidatas, SuperLlaves y cierre de un conjunto de atributos. • Formas Normales (BCNF) • Dependencias multi-valoradas (4NF) • Uniendo dependencias (PJNF, 5NF) • Teoría de la representación.	• Prepara un esquema relacional de un modelo conceptual desarrollado usando el modelo entidad-relación [Usar] • Explica y demuestra los conceptos de restricciones de integridad de la entidad e integridad referencial (incluyendo la definición del concepto de clave foránea) [Usar] • Demuestra el uso de las operaciones de álgebra relacional de la teoría matemática de conjuntos (unión, intersección, diferencia, y producto Cartesiano) y de las operaciones de álgebra relacional desarrolladas específicamente para las bases de datos relacionales (selección (restringida), proyección, unión y división) [Usar] • Escribe consultas en cálculo relacional de tuplas [Usar] • Escribe consultas en cálculo relacional de tuplas [Usar] • Determina la dependencia funcional entre dos o más atributos que son subconjunto de una relación [Usar] • Conecta restricciones expresadas como clave primaria y foránea, con dependencias funcionales [Usar] • Calcula la cerradura de un conjunto de atributos dado dependencias funcionales [Usar] • Determina si un conjunto de atributos forma una superclave y/o una clave candidata de una relación dada dependencias funcionales [Usar] • Evalua una descomposición propuesta, a fin de determinar si tiene una unión sin pérdidas o preservación de dependencias [Usar] • Describe las propiedades de la FNBC, FNUP (forma normal unión de proyecto), 5FN [Usar] • Explica el impacto de la normalización en la eficacia de las operaciones de una base de datos especialmente en la optimización de consultas [Usar] • Describe que es una dependencia de multi valor y cual es el tipo de restricciones que especifica [Usar]
Readings : [WM01], [RG03], [ER15], [CJ11], [KS02]	

Unit 5: Lenguajes de Consulta (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Visión general de lenguajes de base de datos. • SQL (definición de datos, formulacion de consultas, sublenguaje update, restricciones, integridad) • Selecciones • Proyecciones • Select-project-join • Agregaciones y agrupaciones. • Subconsultas. • Entornos QBE de cuarta generación. • Diferentes maneras de invocar las consultas no procedimentales en lenguajes convencionales. • Introducción a otros lenguajes importantes de consulta (por ejemplo, XPATH, SPARQL) • Procedimientos almacenados.	• Crear un esquema relacional de bases de datos en SQL que incorpora restricciones clave y restricciones de integridad de entidad e integridad referencial [Usar] • Usar SQL para crear tablas y devuelve (SELECT) la información de una base de datos [Usar] • Evaluar un conjunto de estrategias de procesamiento de consultas y selecciona la estrategia óptima [Usar] • Crear una consulta no-procedimental al llenar plantillas de relacines para construir un ejemplo del resultado de una consulta requerida [Usar] • Adicionar consultas orientadas a objetos en un lenguaje stand-alone como C++ o Java (ejm. SELECT ColMethod() FROM Objeto) [Usar] • Escribe un procedimiento almacenado que trata con parámetros y con algo de flujo de control de tal forma que tenga funcionalidad [Usar]
Readings : [Die01], [EN04], [Cel05], [KS02]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Die01] Suzanne W Dietrich. *Understanding Relational Database Query Languages, First Edition*. Prentice Hall, 2001.

[WM01] Mark Whitehorn and Bill Marklyn. *Inside Relational Databases, Second Edition*. Springer, 2001.

[KS02] Henry F. Korth and Abraham Silberschatz. *Fundamentos de Base de Datos*. McGraw-Hill, 2002.

[RG03] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. 3rd. McGraw-Hill, 2003.

[EN04] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems, Fourth Edition*. Addison Wesley, 2004.

[RC04] Peter Rob and Carlos Coronel. *Database Systems: Design, Implementation and Management, Sixth Edition*. Morgan Kaufmann, 2004.

[SW04] Graeme Simsion and Graham Witt. *Data Modeling Essentials, Third Edition*. Morgan Kaufmann, 2004.

[Cel05] Joe Celko. *Joe Celko's SQL Programming Style*. Elsevier, 2005.

[CJ11] Date C.J. *SQL and Relational Theory: How to Write Accurate SQL Code*. O'Reilly Media, 2011.

[ER15] Jim Webber Emil Eifrem and Ian Robinson. *Graph Databases*. 2nd. O'Reilly Media, 2015.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS2S1. Operating systems (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS2S1. Operating systems
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS221. Computer Systems Architecture. (3^{rd} Sem) CS221. Computer Systems Architecture. (3^{rd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

An Operating System (OS) manages the computing resources to complete the execution of multiple applications and their associated processes. This course teaches the design of modern operating systems; and introduces their fundamental concepts covering multiple-program execution, scheduling, memory management, file systems, and security. Also, the course includes programming activities on a minimal operating system to solve problems and extend its functionality. Notice that these activities require much time to complete. However, working on them provides valuable insight into operating systems.

5. GOALS

- Study the design of modern operating systems.
- Provide a practical experience by designing and implementing a minimal operating system.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Familiarity)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Assessment)

7. TOPICS

Unit 1: Visión general de Sistemas Operativos (3)	
Competences Expected:	
Topics	**Learning Outcomes**
• Papel y el propósito del sistema operativo. • Funcionalidad de un sistema operativo típico. • Los mecanismos de apoyo modelos cliente-servidor, dispositivos de mano. • Cuestiones de diseño (eficiencia, robustez, flexibilidad, portabilidad, seguridad, compatibilidad) • Influencias de seguridad, creación de redes, multimedia, sistemas de ventanas.	• Explicar los objetivos y funciones de un sistema operativo moderno [Familiarizarse] • Analizar las ventajas y desventajas inherentes en el diseño de un sistema operativo [Evaluar] • Describir las funciones de un sistema operativo contemporaneo respecto a conveniencia, eficiencia, y su habilidad para evolucionar [Familiarizarse] • Discutir acerca de sistemas operativos cliente-servidor, en red, distribuidos y cómo se diferencian de los sistemas operativos de un solo usuario [Familiarizarse] • Identificar amenazas potenciales a sistemas operativos y las características del diseño de seguridad para protegerse de ellos [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 2: Principios de Sistemas Operativos (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Métodos de estructuración (monolítico, capas, modular, los modelos micro-kernel) • Abstracciones, procesos y recursos. • Los conceptos de interfaces de programa de aplicación (API) • La evolución de las técnicas de hardware / software y las necesidades de aplicación • Organización de dispositivos. • Interrupciones: métodos e implementaciones. • Concepto de usuario de estado / sistema y la protección, la transición al modo kernel.	• Explicar el concepto de una capa lógica [Familiarizarse] • Explicar los beneficios de construir capas abstractas en forma jerárquica [Familiarizarse] • Describir el valor de la API y *middleware* [Familiarizarse] • Describir como los recursos computacionales son usados por aplicaciones de software y administradas por el software del sistema [Familiarizarse] • Contrastar el modo *kernel* y modo usuario en un sistema operativo [Evaluar] • Discutir las ventajas y desventajas del uso de procesamiento interrumpido [Familiarizarse] • Explicar el concepto de una capa lógica [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 3: Concurrencia (9)	
Competences Expected:	
Topics	**Learning Outcomes**
• Diagramas de estado. • Estructuras (lista preparada, bloques de control de procesos, y así sucesivamente) • Despacho y cambio de contexto. • El papel de las interrupciones. • Gestionar el acceso a los objetos del sistema operativo atómica. • La implementación de primitivas de sincronización. • Cuestiones multiprocesador (spin-locks, reentrada)	• Describir la necesidad de concurrencia en el marco de un sistema operativo [Familiarizarse] • Demostrar los potenciales problemas de tiempo de ejecución derivados de la operación simultánea de muchas tareas diferentes [Usar] • Resumir el rango de mecanismos que pueden ser usados a nivel del sistema operativo para realizar sistemas concurrentes y describir los beneficios de cada uno [Familiarizarse] • Explicar los diferentes estados por los que una tarea debe pasar y las estructuras de datos necesarias para el manejo de varias tareas [Familiarizarse] • Resumir las técnicas para lograr sicronización en un sistema operativo(por ejemplo, describir como implementar semáforos usando primitivas del sistema operativo.) [Familiarizarse] • Describir las razones para usar interruptores, despacho, y cambio de contexto para soportar concurrencia en un sistema operativo [Familiarizarse] • Crear diagramas de estado y transición para los dominios de problemas simples [Usar]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 4: Planificación y despacho (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Planificación preventiva y no preferente. • Planificadores y políticas. • Procesos y subprocesos. • Plazos y cuestiones en tiempo real.	• Comparar y contrastar los algoritmos comunes que se utilizan tanto para un programa preferente y no preferente de las tareas en los sistemas operativos, como la comparación de prioridad, el rendimiento, y los esquemas de distribución equitativa [Evaluar] • Describir las relaciones entre los algoritmos de planificación y dominios de aplicación [Familiarizarse] • Discutir los tipos de planeamiento de procesos *scheduling* de corto, a mediano, a largo plazo y I/O [Familiarizarse] • Describir las diferencias entre procesos y hebras [Familiarizarse] • Comparar y contrastar enfoques estáticos y dinámicos para *scheduling* en tiempo real [Evaluar] • Hablar sobre la necesidad de tiempos límites de *scheduling* [Familiarizarse] • Identificar formas en que la lógica expresada en algoritmos de planificación son de aplicación a otros ámbitos, tales como I/O del disco, la programación de disco de red, programación de proyectos y problemas más allá de la computación [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 5: Manejo de memoria (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Revisión de la memoria física y hardware de gestión de memoria. • Conjuntos de trabajo y thrashing. • El almacenamiento en caché	• Explicar la jerarquía de la memoria y costo-rendimiento de intercambio [Familiarizarse] • Resumir los principios de memoria virtual tal como se aplica para el almacenamiento en cache y paginación [Familiarizarse] • Evaluar las ventajas y desventajas en términos del tamaño de memoria (memoria principal, memoria caché, memoria axiliar) y la velocidad del procesador [Evaluar] • Defiende las diferentes formas de asignar memoria a las tareas, citando las ventajas relativas de cada uno [Familiarizarse] • Describir el motivo y el uso de memoria caché (rendimiento y proximidad, dimensión diferente de como los caches complican el aislamiento y abstracción en VM) [Familiarizarse] • Estudiar los conceptos de *thrashing*, tanto en términos de las razones por las que se produce y las técnicas usadas para el reconocimiento y manejo del problema [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 6: Seguridad y protección (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Visión general de la seguridad del sistema . • Política / mecanismo de separación. • Métodos de seguridad y dispositivos. • Protección, control de acceso y autenticación. • Las copias de seguridad.	• Articular la necesidad para la protección y seguridad en un sistema operativo [Familiarizarse] • Resumir las caracteristicas y limitaciones de un sistema operativo usado para proporcionar protección y seguridad [Familiarizarse] • Explicar el mecanismo disponible en un OS para controlar los accesos a los recursos [Familiarizarse] • Realizar tareas de administración de sistemas sencillas de acuerdo a una política de seguridad, por ejemplo la creación de cuentas, el establecimiento de permisos, aplicación de parches y organización de backups regulares [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 7: Máquinas virtuales (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Tipos de virtualización (incluyendo Hardware / Software, OS, Servidor, Servicio, Red) • Paginación y la memoria virtual. • Sistemas de archivos virtuales. • Los Hypervisor. • Virtualización portátil; emulación vs aislamiento. • Costo de la virtualización.	• Explicar el concepto de memoria virtual y la forma cómo se realiza en hadware y software [Familiarizarse] • Diferenciar emulacion y el aislamiento [Familiarizarse] • Evaluar virtualización de compensaciones [Evaluar] • Discutir sobre hipervisores y la necesidad para ellos en conjunto con diferentes tipos de hipervisores [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 8: Manejo de dispositivos (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Características de los dispositivos serie y paralelo. • Haciendo de abstracción de dispositivos. • Estrategias de buffering. • Acceso directo a memoria. • La recuperación de fallos.	• Explique la diferencia clave entre dispositivos seriales y paralelos e identificar las condiciones en las cuales cada uno es apropiado [Familiarizarse] • Identificar los requerimientos para recuperación de errores [Familiarizarse] • Explique *buffering* y describir las estrategias para su aplicación [Familiarizarse] • Diferenciar los mecanismos utilizados en la interconexión de un rango de dispositivos (incluyendo dispositivos portátiles, redes, multimedia) a un ordenador y explicar las implicaciones de éstas para el diseño de un sistema operativo [Familiarizarse] • Describir las ventajas y desventajas de acceso directo a memoria y discutir las cirscunstancias en cuales se justifica su uso [Familiarizarse] • Identificar la relación entre el hardware físico y los dispositivos virtuales mantenidos por el sistema operativo [Familiarizarse] • Implementar un controlador de dispositivo simple para una gama de posibles equipos [Usar]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 9: Sistema de archivos (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Archivos: los datos, metadatos, operaciones, organización, amortiguadores, secuenciales, no secuencial. • Directorios: contenido y estructura. • Los sistemas de archivos: partición, montar sistemas de archivos / desmontar, virtuales. • Técnicas estándar de implementación . • Archivos asignados en memoria. • Sistemas de archivos de propósito especial. • Naming, búsqueda, acceso, copias de seguridad. • La bitacora y los sistemas de archivos estructurados (log)	• Describir las decisiones que deben tomarse en el diseño de sistemas de archivos [Familiarizarse] • Comparar y contrastar los diferentes enfoques para la organización de archivos, el reconocimiento de las fortalezas y debilidades de cada uno. [Evaluar] • Resumir cómo el desarrollo de hadware ha dado lugar a cambios en las prioridades para el diseño y la gestión de sistemas de archivos [Familiarizarse] • Resumir el uso de diarios y como los sistemas de archivos de registro estructurado mejora la tolerancia a fallos [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 10: Sistemas empotrados y de tiempo real (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Proceso y programación de tareas. • Los requisitos de gestión de memoria / disco en un entorno en tiempo real. • Los fracasos, los riesgos y la recuperación. • Preocupaciones especiales en sistemas de tiempo real.	• Describir que hace a un sistema un sistema en tiempo real [Familiarizarse] • Explicar la presencia y describir las características de latencia en sistemas de tiempo real [Familiarizarse] • Resumir los problemas especiales que los sistemas en tiempo real presentan, incluyendo el riesgo, y cómo se tratan estos problemas [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 11: Tolerancia a fallas (3)	
Competences Expected:	
Topics	**Learning Outcomes**
• Conceptos fundamentales: sistemas fiables y disponibles. • Redundancia espacial y temporal. • Los métodos utilizados para implementar la tolerancia a fallos. • Los ejemplos de los mecanismos del sistema operativo para la detección, recuperación, reinicie para implementar la tolerancia a fallos, el uso de estas técnicas para los servicios propios del sistema operativo.	• Explicar la importancia de los términos tolerancia a fallos, fiabilidad y disponibilidad [Familiarizarse] • Explicar en términos generales la gama de métodos para implementar la tolerancia a fallos en un sistema operativo [Familiarizarse] • Explicar cómo un sistema operativo puede continar funcionando después de que ocurra una falla [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

Unit 12: Evaluación del desempeño de sistemas (3)	
Competences Expected:	
Topics	**Learning Outcomes**
• ¿Por qué el rendimiento del sistema debe ser evaluado? • ¿Qué se va a evaluar? • Sistemas de políticas de rendimiento, por ejemplo, el almacenamiento en caché, de paginación, la programación, la gestión de memoria, y la seguridad. • Modelos de evaluación: analítica, simulación, o de implementación específico determinista. • Cómo recoger los datos de evaluación (perfiles y mecanismos de localización)	• Describir las medidas de rendimiento utilizados para determinar cómo el sistema funciona [Familiarizarse] • Explicar los principales modelos de evaluación utilizados para evaluar un sistema [Familiarizarse]
Readings : [Avi12], [Sta05], [Tan06], [Tan01], [AD14]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Tan01] Andrew S. Tanenbaum. *Modern Operating Systems, 4/E.* Prentice Hall, 2001.

[Sta05] William Stallings. *Operating Systems: Internals and Design Principles, 5/E.* Prentice Hall, 2005.

[Tan06] Andrew S. Tanenbaum. *Operating Systems Design and Implementation, 3/E.* Prentice Hall, 2006.

[Avi12] Greg Gagne Avi Silberschatz Peter Baer Galvin. *Operating System Concepts, 9/E.* John Wiley & Sons, Inc., 2012.

[AD14] Thomas Anderson and Michael Dahlin. *Operating Systems: Principles and Practice.* 2nd. Recursive Books, 2014.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

MA201. Calculus II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	MA201. Calculus II
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- MA101. Math II. (2^{nd} Sem)
- MA102. Calculus I. (3^{rd} Sem)
- MA101. Math II. (2^{nd} Sem)
- MA102. Calculus I. (3^{rd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Es una extensión de los cursos de Análisis Matemático I y Análisis Matemático II, tomando en cuenta dos o más variables, indispensables para aquellas materias que requieren trabajar con geometría en curvas y superficies, así como en procesos de búsqueda de puntos extremos.

5. GOALS

- Diferenciar e integrar funciones vectoriales de variable real, entender y manejar el concepto de parametrización. Describir una curva en forma paramétrica.
- Describir, analizar, diseñar y formular modelos continuos que dependen de más de una variable.
- Establecer relaciones entre diferenciación e integración y aplicar el cálculo diferencial e integral ala resolución de problemas geométricos y de optimización.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• R^3 como espacio euclídeo y álgebra . • Superficies básicas en el espacio.	• Manejar el álgebra vectorial en R^3[Usar]. • Identificar tipos de superficies en el espacio [Usar]. • Graficar superficies básicas [Usar].
Readings : [Apó73], [Sim95]	

Unit 2: (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Funciones vectoriales de variable real. Reparametrizaciones • Diferenciación e integración • Velocidad, aceleración , curvatura, torsión	• Describir las diferentes carácteristicas de una curva [Usar].
Readings : [Apó73], [Sim95]	

Unit 3: (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Curvas de nivel • Límites y continuidad • Diferenciación	• Graficar campos escalares • Discutir la existencia de un límite y la continuidad de un campo escalar [Usar]. • Calcular derivadas parciales y totales [Usar].
Readings : [Apó73], [Bar76], [Sim95]	

Unit 4: (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Máximos y mínimos • Multiplicadores de Lagrange	• Interpretar la noción de gradiente en curvas de nivel y en superficies de nivel [Usar]. • Usar técnicas para hallar extremos [Usar].
Readings : [Apó73], [Sim95], [Bar76]	

Unit 5: (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Integración de Riemann • Integración sobre regiones • Cambio de coordenadas • Aplicaciones	• Reconocer regiones de integración adecuadas [Usar]. • Realizar cambios de coordenadas adecuados [Usar]. • Aplicar la integración múltiple a problemas [Usar].
Readings : [Apó73]	

Unit 6: (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Integrales de linea • Campos conservativos • Integrales de superficie	• Calcular la integral de linea de campos vectoriales[Usar]. • Reconocer campos conservativos[Usar]. • Hallar funciones potenciales de campos conservativos [Usar]. • Hallar integrales de superficies y aplicarlas [Usar].
Readings : [Apó73]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Apó73] Tom M Apóstol. *Calculus*. Vol. II. Editorial Reverté, 1973.

[Bar76] Robert G. Bartle. *The Elements of Real Analysis*. Wiley; 2 edition, 1976.

[Sim95] George F Simmons. *Calculus With Analytic Geometry*. McGraw-Hill Science/Engineering, 1995.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

MA203. Statistics and Probabilities (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	MA203. Statistics and Probabilities
2.2 Semester	:	4^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	MA100. Mathematics I. (1^{st} Sem) MA100. Mathematics I. (1^{st} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

It provides an introduction to probability theory and statistical inference with applications, needs in data analysis, design of random models and decision making.

5. GOALS

- An ability to design and conduct experiments, as well as to analyze and interpret data.
- An ability to identify, formulate, and solve real problems.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Variable Type (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Variable Type: Continuous, discrete	• Classify the relevant variables identified according to their type: continuous (interval and ratio), categorical (nominal, ordinal, dichotomous). • Identify the relevant variables of a system using a process approach.
Readings : [MRo14], [Men14]	

Unit 2: Descriptive Statistics (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Central Tendency (Mean, median, mode) • Dispersion (Range, standard deviation, quartile) • Graphics: histogram, boxplot, etc.: Communication ability.	• Use central tendency measures and dispersion measures to describe the data gathered. • Use graphics to communicate the characteristics of the data gathered.
Readings : [MRo14], [Men14]	

Unit 3: Inferential Statistics (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Determination of the sample size • Confidence interval • Type I and type II error • Distribution type • Hypothesis test (t-student, means, proportions and ANOVA) • Relationships between variables: correlation, regression.	• Propose questions and hypotheses of interest. • Analyze the data gathered using different statistical tools to answer questions of interest. • Draw conclusions based on the analysis performed.
Readings : [MRo14], [Men14]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[MRo14] Sheldon M.Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. 5th. Academic Press, 2014.

[Men14] Beaver Mendenhall. *Introducción a la probabilidad y estadística*. 13th. Cengage Learning, 2014.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS212. Analysis and Design of Algorithms (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS212. Analysis and Design of Algorithms
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- CS210. Algorithms and Data Structures. (4^{th} Sem)
- CS211. Theory of Computation. (4^{th} Sem)
- CS210. Algorithms and Data Structures. (4^{th} Sem)
- CS211. Theory of Computation. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

An algorithm is, essentially, a well-defined set of rules or instructions that allow solving a computational problem. The theoretical study of the performance of the algorithms and the resources used by them, usually time and space, allows us to evaluate if an algorithm is suitable for solving a specific problem, comparing it with other algorithms for the same problem or even delimiting the boundary between Viable and impossible. This matter is so important that even Donald E. Knuth defined Computer Science as the study of algorithms. This course will present the most common techniques used in the analysis and design of efficient algorithms, with the purpose of learning the fundamental principles of the design, implementation and analysis of algorithms for the solution of computational problems

5. GOALS

- Develop the ability to evaluate the complexity and quality of algorithms proposed for a given problem.
- Study the most representative, introductory algorithms of the most important classes of problems treated in computation.
- Develop the ability to solve algorithmic problems using the fundamental principles of algorithm design learneds.
- Be able to answer the following questions when a new algorithm is presented: How good is the performance ?, Is there a better way to solve the problem?

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Análisis Básico (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Diferencias entre el mejor, el esperado y el peor caso de un algoritmo. • Análisis asintótico de complejidad de cotas superior y esperada. • Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial. • Asymptotic Notation • Análisis de algoritmos iterativos y recursivos. • Inductive proofs and correctness of algorithms • Algunas versiones del Teorema Maestro.	• Explique a que se refiere con "mejor", "esperado" y "peor" caso de comportamiento de un algoritmo [Evaluar] • Determine informalmente el tiempo y el espacio de complejidad de simples algoritmos [Evaluar] • Lista y contraste de clases estándares de complejidad [Evaluar] • Explicar el uso de la notación theta grande, omega grande y o pequeña para describir la cantidad de trabajo hecho por un algoritmo [Evaluar] • Analyze worst-case running times of algorithms using asymptotic analysis [Evaluar] • Usar relaciones recurrentes para determinar el tiempo de complejidad de algoritmos recursivamente definidos [Evaluar] • Resuelve relaciones de recurrencia básicas, por ejemplo. usando alguna forma del Teorema Maestro [Evaluar] • Argue the correctness of algorithms using inductive proofs [Evaluar]
Readings : [KT05], [DPV06], [Cor+09], [SF13], [Knu97]	

Unit 2: Estrategias Algorítmicas (30)	
Competences Expected:	
Topics	**Learning Outcomes**
• Algoritmos de fuerza bruta. • Algoritmos voraces. • Divide y vencerás. • Programación Dinámica.	• Para cada una de las estrategias (fuerza bruta, algoritmo goloso, divide y vencerás, recursividad en reversa y programación dinámica), identifica un ejemplo práctico en el cual se pueda aplicar [Evaluar] • Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima [Evaluar] • Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima [Evaluar] • Usa programación dinámica para resolver un problema determinado [Evaluar] • Determina el enfoque algorítmico adecuado para un problema [Evaluar]
Readings : [KT05], [DPV06], [Cor+09], [Als99]	

Unit 3: Algoritmos y Estructuras de Datos fundamentales (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Grafos y algoritmos en grafos: – Algoritmos de la ruta más corta (algoritmos de Dijkstra y Floyd) – Árbol de expansión mínima (algoritmos de Prim y Kruskal) • Cache oblivious algorithms • Number theory and cryptography	• Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] • Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Evaluar] • Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección,y para implementar el algoritmo en un contexto en específico [Evaluar] • Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Evaluar]
Readings : [KT05], [DPV06], [Cor+09], [SW11], [GT09]	

Unit 4: Computabilidad y complejidad básica de autómatas (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Introducción a las clases P y NP y al problema P vs. NP. • Introducción y ejemplos de problemas NP- Completos y a clases NP-Completos. • Reductions	• Define las clases P y NP [Familiarizarse] • Explique el significado de NP-Completitud [Familiarizarse]
Readings : [KT05], [DPV06], [Cor+09]	

Unit 5: Estructuras de Datos Avanzadas y Análisis de Algoritmos (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Grafos (ej. Ordenamiento Topológico, encontrando componentes puertemente conectados) • Algoritmos aleatorios. • Análisis amortizado. • Análisis Probabilístico. • Approximation Algorithms • Linear Programming	• Entender el mapeamento de problemas del mundo real a soluciones algorítmicas (ejemplo, problemas de grafos, programas lineares,etc) [Familiarizarse] • Seleccionar y aplicar técnicas avanzadas de análisis (ejemplo, amortizado, probabilistico,etc) para algoritmos [Usar]
Readings : [KT05], [DPV06], [Cor+09], [Tar83], [Raw92]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Tar83] Robert Endre Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, 1983.

[Raw92] G.J.E. Rawlins. *Compared to What?: An Introduction to the Analysis of Algorithms*. Computer Science Press, 1992.

[Knu97] D.E. Knuth. *The Art of Computer Programming: Fundamental algorithms*. v. 1. Addison-Wesley, 1997.

[Als99] H. Alsuwaiyel. *Algorithms: Design Techniques and Analysis*. World Scientific, 1999.

[KT05] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[DPV06] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill Education, 2006.

[Cor+09] Thomas H. Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009.

[GT09] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis and Internet Examples*. 2nd. John Wiley & Sons, Inc., 2009.

[SW11] R. Sedgewick and K. Wayne. *Algorithms*. Pearson Education, 2011.

[SF13] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Pearson Education, 2013.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS272. Databases II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS272. Databases II
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS271. Data Management. (4^{th} Sem) CS271. Data Management. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Information Management (IM) plays a leading role in almost every area where computers are used. This area includes the capture, digitization, representation, organization, transformation and presentation of information; Algorithms to improve the efficiency and effectiveness of access and update of stored information, data modeling and abstraction, and physical file storage techniques.

It also covers information security, privacy, integrity and protection in a shared environment. Students need to be able to develop conceptual and physical data models, determine which IM methods and techniques are appropriate for a given problem, and be able to select and implement an appropriate IM solution that reflects all applicable constraints, including scalability and Usability.

5. GOALS

- To make the student understand the different applications that the databases have, in the different areas of knowledge.
- Show appropriate ways of storing information based on their various approaches and their subsequent retrieval of information.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Assessment)

7. TOPICS

Unit 1: Diseño Físico de Bases de Datos (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Almacenamiento y estructura de archivos. • Archivos indexados. • Archivos Hash. • Archivos de Firma. • Árboles B. • Archivos con índice denso. • Archivos con registros de tamaño variable. • Eficiencia y Afinación de Bases de Datos.	• Explica los conceptos de registro, tipos de registro, y archivos, así como las diversas técnicas para colocar registros de archivos en un disco [Usar] • Da ejemplos de la aplicación de índices primario, secundario y de agrupamiento [Usar] • Distingue entre un índice no denso y uno denso [Usar] • Implementa índices de multinivel dinámicos usando árboles-B [Usar] • Explica la teoría y la aplicación de técnicas de hash internas y externas [Usar] • Usa técnicas de hasp para facilitar la expansión de archivos dinámicos [Usar] • Describe las relaciones entre hashing, compresión, y búsquedas eficientes en bases de datos [Usar] • Evalúa el costo y beneficio de diversos esquemas de hashing [Usar] • Explica como el diseño físico de una base de datos afecta la eficiencia de las transacciones en ésta [Usar]
Readings : [Bur04], [Cel05]	

Unit 2: Procesamiento de Transacciones (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Transacciones. • Fallo y recuperación. • Control concurente. • Interacción de gestión de transacciones con el almacenamiento, especialmente en almacenamiento.	• Crear una transacción mediante la incorporación de SQL en un programa de aplicación [Usar] • Explicar el concepto de confimaciones implicitas [Usar] • Describir los problemas especificos para la ejecución de una transacción eficiente [Usar] • Explicar cuando y porqué se necesita un *rollback*, y cómo registrar todo asegura un *rollback* adecuado [Usar] • Explicar el efecto de diferentes niveles de aislamiento sobre los mecanismos de control de concurrencia [Usar] • Elejir el nivel de aislamiento adecuado para la aplicación de un protocolo de transacción especificado [Usar] • Identificar los límites apropiados de la transacción en programas de aplicación [Usar]
Readings : [Phi97], [Ram04]	

Unit 3: Almacenamiento y Recuperación de Información (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Documentos, publicación electrónica, markup, y lenguajes markup. • Tries, archivos invertidos, Árboles PAT, archivos de firma, indexación. • Análisis Morfológico, stemming, frases, stop lists. • Distribuciones de frecuencia de términos, incertidumbre, fuzificación (fuzzyness), ponderación. • Espacio vectorial, probabilidad, lógica, y modelos avanzados. • Necesidad de Información , Relevancia, evaluación, efectividad. • Thesauri, ontologías, clasificación y categorización, metadata. • Información bibliográfica, bibliometría, citaciones. • Enrutamiento y filtrado. • Búsqueda multimedia. • Información de resumen y visualización. • Búsqueda por facetas (por ejemplo, el uso de citas, palabras clave, esquemas de clasificación). • Librerías digitales. • Digitalización, almacenamiento, intercambio, objetos digitales, composición y paquetes. • Metadata y catalogación. • Nombramiento, repositorios, archivos • Archivamiento y preservación, integrdad • Espacios (Conceptual, geográfico, 2/3D, Realidad virtual) • Arquitecturas (agentes, autobuses, envolturas / mediadores), de interoperabilidad. • Servicios (búsqueda, de unión, de navegación, y así sucesivamente). • Gestión de derechos de propiedad intelectual, la privacidad y la protección (marcas de agua).	• Explica los conceptos básicos de almacenamiento y recuperación de la información [Usar] • Describe que temas son específicos para una recuperación de la información eficiente [Usar] • Da aplicaciones de estrategias alternativas de búsqueda y explica porqué una estrategia en particular es apropiada para una aplicación [Usar] • Diseña e implementa un sistema de almacenamiento y recuperación de la información o librería digital de tamaño pequeño a mediano [Usar] • Describe algunas de las soluciones técnicas a los problemas relacionados al archivamiento y preservación de la información en una librería digital [Usar]
Readings : [Pet98], [Ram04]	

Unit 4: Bases de Datos Distribuidas (36)	
Competences Expected:	
Topics	**Learning Outcomes**
• DBMS Distribuidas – Almacenamiento de datos distribuido – Procesamiento de consultas distribuido – Modelo de transacciones distribuidas – Soluciones homogéneas y heterogéneas – Bases de datos distribuidas cliente-servidor • Parallel DBMS – Arquitecturas paralelas DBMS: memoria compartida, disco compartido, nada compratido; – Aceleracion y ampliación, por ejemplo, el uso del modelo de procesamiento MapReduce – Replicacion de informacion y modelos de consistencia debil	• Explicar las técnicas usadas para la fragmentación de datos, replicación, y la asignación durante el proceso de diseño de base de datos distribuida [Usar] • Evaluar estrategias simples para la ejecución de una consulta distribuida para seleccionar una estrategia que minimise la cantidad de transferencia de datos [Usar] • Explicar como el protocolo de dos fases de *commit* es usado para resolver problemas de transacciones que acceden a bases de datos almacenadas en múltiples nodos [Usar] • Describir el control concurrente distribuido basados en técnicas de copia distinguidos y el método de votación. [Usar] • Describir los tres niveles del software en el modelo cliente servidor [Usar]
Readings : [M T99]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Phi97] Eric Newcomer Philip A. Bernstein. *Principles of Transaction Processing, First Edition*. Morgan Kaufmann, 1997.

[Pet98] Julita Vassileva Peter Brusilovsky Alfred Kobsa. *Adaptive Hypertext and Hypermedia, First Edition*. Springer, 1998.

[M T99] Patrick Valduriez M. Tamer Ozsu. *Principles of Distributed Database Systems, Second Edition*. Prentice Hall, 1999.

[Bur04] Donald K. Burleson. *Physical Database Design Using Oracle*. CRC Press, 2004.

[Ram04] Shamkant B. Navathe Ramez Elmasri. *Fundamentals of Database Systems, Fourth Edition*. Addison Wesley, 2004.

[Cel05] Joe Celko. *Joe Celko's SQL Programming Style*. Elsevier, 2005.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS291. Software Engineering I (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS291. Software Engineering I
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- CS113. Computer Science II. (3^{rd} Sem)
- CS271. Data Management. (4^{th} Sem)
- CS113. Computer Science II. (3^{rd} Sem)
- CS271. Data Management. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The aim of developing software, except for extremely simple applications, requires the execution of a well-defined development process. Professionals in this area require a high degree of knowledge of the different models and development process, so that they are able to choose the most suitable for each development project. On the other hand, the development of medium and large-scale systems requires the use of pattern and component libraries and the mastery of techniques related to component-based design

5. GOALS

- Provide the student with a theoretical and practical framework for the development of software under quality standards.
- Familiarize the student with the software modeling and construction processes through the use of CASE tools.
- Students should be able to select architectures and ad-hoc technology platforms for deployment scenarios
- Applying component-based modeling to ensure variables such as quality, cost, and time-to-market in development processes.
- Provide students with best practices for software verification and validation.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

3) Communicate effectively in a variety of professional contexts.. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Ingeniería de Requisitos (18)

Competences Expected:

Topics	**Learning Outcomes**
• Al describir los requisitos funcionales utilizando, por ejemplo, los casos de uso o historias de los usuarios. • Propiedades de requisitos, incluyendo la consistencia, validez, integridad y viabilidad. • Requisitos de software elicitatión. • Descripción de datos del sistema utilizando, por ejemplo, los diagramas de clases o diagramas entidad-relación. • Requisitos no funcionales y su relación con la calidad del software. • Evaluación y uso de especificaciones de requisitos. • Requisitos de las técnicas de modelado de análisis. • La aceptabilidad de las consideraciones de certeza/incertidumbre sobre el comportamiento del software/sistema. • Prototipos. • Conceptos básicos de la especificación formal de requisitos. • Especificación de requisitos. • Validación de requisitos. • Rastreo de requisitos.	• Enumerar los componentes clave de un caso de uso o una descripción similar de algún comportamiento que es requerido para un sistema [Evaluar] • Describir cómo el proceso de ingeniería de requisitos apoya la obtención y validación de los requisitos de comportamiento [Evaluar] • Interpretar un modelo de requisitos dada por un sistema de software simple [Evaluar] • Describir los retos fundamentales y técnicas comunes que se utilizan para la obtención de requisitos [Evaluar] • Enumerar los componentes clave de un modelo de datos (por ejemplo, diagramas de clases o diagramas ER) [Evaluar] • Identificar los requisitos funcionales y no funcionales en una especificación de requisitos dada por un sistema de software [Evaluar] • Realizar una revisión de un conjunto de requisitos de software para determinar la calidad de los requisitos con respecto a las características de los buenos requisitos [Evaluar] • Aplicar elementos clave y métodos comunes para la obtención y el análisis para producir un conjunto de requisitos de software para un sistema de software de tamaño medio [Evaluar] • Comparar los métodos ágiles y el dirigido por planes para la especificación y validación de requisitos y describir los beneficios y riesgos asociados con cada uno [Evaluar] • Usar un método común, no formal para modelar y especificar los requisitos para un sistema de software de tamaño medio [Evaluar] • Traducir al lenguaje natural una especificación de requisitos de software (por ejemplo, un contrato de componentes de software) escrito en un lenguaje de especificación formal [Evaluar] • Crear un prototipo de un sistema de software para reducir el riesgo en los requisitos [Evaluar] • Diferenciar entre el rastreo (*tracing*) hacia adelante y hacia atrás y explicar su papel en el proceso de validación de requisitos [Evaluar]

Readings : [ES14], [HF03]

Unit 2: Diseño de Software (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Principios de diseño del sistema: niveles de abstracción (diseño arquitectónico y el diseño detallado), separación de intereses, ocultamiento de información, de acoplamiento y de cohesión, de reutilización de estructuras estándar. • Diseño de paradigmas tales como diseño estructurado (descomposición funcional de arriba hacia abajo), el análisis orientado a objetos y diseño, orientado a eventos de diseño, diseño de nivel de componente, centrado datos estructurada, orientada a aspectos, orientado a la función, orientado al servicio. • Modelos estructurales y de comportamiento de los diseños de software. • Diseño de patrones. • Relaciones entre los requisitos y diseños: La transformación de modelos, el diseño de los contratos, invariantes. • Conceptos de arquitectura de software y arquitecturas estándar (por ejemplo, cliente-servidor, n-capas, transforman centrados, tubos y filtros). • El uso de componentes de diseño: seleccion de componentes,diseño,adaptacion y componentes de ensamblaje, componentes y patrones, componentes y objetos(por ejemplo,construir una GUI usando un standar widget set) • Calidad del diseño interno, y modelos para: eficiencia y desempeño, redundancia y tolerancia a fallos, trazavilidad de los requerimientos. • Calidad del diseño interno, y modelos para: eficiencia y desempeño, redundancia y tolerancia a fallos, trazavilidad de los requerimientos. • Medición y análisis de la calidad de un diseño. • Compensasiones entre diferentes aspectos de la calidad. • Aaplicaciones en frameworks. • Middleware: El paradigma de la orientacion a objetos con middleware, requerimientos para correr y clasificar objetos, monitores de procesamiento de transacciones y el sistema de flujo de trabajo. • Principales diseños de seguridad y codificación(cross-reference IAS/Principles of securre design). – Principio de privilegios mínimos – Principio de falla segura por defecto – Principio de aceptabilidad psicológica	• Formular los principios de diseño, incluyendo la separación de problemas, ocultación de información, acoplamiento y cohesión, y la encapsulación [Familiarizarse] • Usar un paradigma de diseño para diseñar un sistema de software básico y explicar cómo los principios de diseño del sistema se han aplicado en este diseño [Usar] • Construir modelos del diseño de un sistema de software simple los cuales son apropiado para el paradigma utilizado para diseñarlo [Usar] • En el contexto de un paradigma de diseño simple, describir uno o más patrones de diseño que podrían ser aplicables al diseño de un sistema de software simple [Familiarizarse] • Para un sistema simple adecuado para una situación dada, discutir y seleccionar un paradigma de diseño apropiado [Usar] • Crear modelos apropiados para la estructura y el comportamiento de los productos de software desde la especificaciones de requisitos [Usar] • Explicar las relaciones entre los requisitos para un producto de software y su diseño, utilizando los modelos apropiados [Evaluar] • Para el diseño de un sistema de software simple dentro del contexto de un único paradigma de diseño, describir la arquitectura de software de ese sistema [Familiarizarse] • Dado un diseño de alto nivel, identificar la arquitectura de software mediante la diferenciación entre las arquitecturas comunes de software, tales como 3 capas (*3-tier*), *pipe-and-filter*, y cliente-servidor [Familiarizarse] • Investigar el impacto de la selección arquitecturas de software en el diseño de un sistema simple [Evaluar] • Aplicar ejemplos simples de patrones en un diseño de software [Usar] • Describir una manera de refactorar y discutir cuando esto debe ser aplicado [Familiarizarse] • Seleccionar componentes adecuados para el uso en un diseño de un producto de software [Usar] • Explicar cómo los componentes deben ser adaptados para ser usados en el diseño de un producto de software [Familiarizarse] • Diseñar un contrato para un típico componente de software pequeño para el uso de un dado sistema [Usar] • Discutir y seleccionar la arquitectura de software

Unit 3: Construcción de Software (24)	
Competences Expected:	
Topics	**Learning Outcomes**
• Prácticas de codificación: técnicas, idiomas/patrones, mecanismos para construcción de programas de calidad: – Prácticas de codificación defensive – Prácticas de codificación segura – Utilizando mecanismos de manejo de excepciones para hacer el programa más robusto, tolerante a fallas • Normas de codificación. • Estrategias de integración. • Desarrollando contexto: ""campo verde"" frente a la base de código existente : – Análisis de cambio impacto – Cambio de actualización • Los problemas de seguridad potenciales en los programas : – Buffer y otros tipos de desbordamientos – Condiciones elemento Race – Inicialización incorrecta, incluyendo la elección de los privilegios – Entrada Comprobación – Suponiendo éxito y corrección – La validación de las hipótesis	• Describir técnicas, lenguajes de codificación y mecanismos de implementación para conseguir las propiedades deseadas, tales como la confiabilidad, la eficiencia y la robustez [Evaluar] • Construir código robusto utilizando los mecanismos de manejo de excepciones [Evaluar] • Describir la codificación segura y prácticas de codificación de defensa [Evaluar] • Seleccionar y utilizar un estándar de codificación definido en un pequeño proyecto de software [Evaluar] • Comparar y contrastar las estrategias de integración incluyendo: de arriba hacia abajo (*top-down*), de abajo hacia arriba (*bottom-up*), y la integración Sándwich [Evaluar] • Describir el proceso de analizar e implementar los cambios a la base de código desarrollado para un proyecto específico [Evaluar] • Describir el proceso de analizar e implementar los cambios a la base de código desarrollado para un proyecto específico [Evaluar] • Reescribir un programa sencillo para eliminar vulnerabilidades comunes, tales como desbordamientos de búffer, desbordamientos de enteros y condiciones de carrera [Evaluar] • Escribir un componente de software que realiza alguna tarea no trivial y es resistente a errores en la entrada y en tiempo de ejecución [Evaluar]
Readings : [ES14], [HF03]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[HF03] Brian Lyons Hans-Erik Eriksson Magnus Penker and Davis Fado. *UML 2 Toolkit*. 2nd. Wiley, Oct. 2003.

[ES14] Bert Bates Eric Freeman Elisabeth Robson and Kathy Sierra. *Head First Design Patterns*. 2nd. O'Reilly Media, Inc, July 2014.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS342. Compilers (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS342. Compilers
2.2 Semester	:	5^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS211. Theory of Computation. (4^{th} Sem) CS211. Theory of Computation. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

That the student knows and understands the concepts and fundamental principles of the theory of compilation to realize the construction of a compiler

5. GOALS

- Know the basic techniques used during the process of intermediate generation, optimization and code generation.
- Learning to implement small compilers.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Representación de programas (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Programas que tienen otros programas como entrada tales como interpretes, compiladores, revisores de tipos y generadores de documentación. • Arboles de sintaxis abstracta, para contrastar la sintaxis correcta. • Estructuras de datos que representan código para ejecución, traducción o transmisión. • Compilación en tiempo just-in time y re-compilación dinámica. • Otras características comunes de las máquinas virtuales, tales como carga de clases, hilos y seguridad.	• Explicar como programas que procesan otros programas tratan a los otros programas como su entrada de datos [Familiarizarse] • Describir un árbol de sintaxis abstracto para un lenguaje pequeño [Familiarizarse] • Describir los beneficios de tener representaciones de programas que no sean cadenas de código fuente [Familiarizarse] • Escribir un programa para procesar alguna representación de código para algún propósito, tales como un interprete, una expresión optimizada, o un generador de documentación [Familiarizarse] • Explicar el uso de metadatos en las representaciones de tiempo de ejecución de objetos y registros de activación, tales como los punteros de la clase, las longitudes de arreglos, direcciones de retorno, y punteros de *frame* [Familiarizarse] • Discutir las ventajas, desventajas y dificultades del término (*just-in-time*) y recompilación automática [Familiarizarse] • Identificar los servicios proporcionados por los sistemas de tiempo de ejecución en lenguajes modernos [Familiarizarse]
Readings : [Lou04b]	

Unit 2: Traducción y ejecución de lenguajes (10)

Competences Expected:

Topics	Learning Outcomes
• Interpretación vs. compilación a código nativo vs. compilación de representación portable intermedia. • Pipeline de traducción de lenguajes: análisis, revisión opcional de tipos, traducción, enlazamiento, ejecución: – Ejecución como código nativo o con una máquina virtual – Alternativas como carga dinámica y codificación dinámica de código (o "just-in-time"") • Representación en tiempo de ejecución de construcción del lenguaje núcleo tales como objetos (tablas de métodos) y funciones de primera clase (cerradas) • Ejecución en tiempo real de asignación de memoria: pila de llamdas, montículo, datos estáticos: – Implementación de bucles, recursividad y llamadas de cola • Gestión de memoria: – Gestión manual de memoria: asignación, limpieza y reuso de la pila de memoria – Gestión automática de memoria: recolección de datos no utilizados (*garbage colletion*) como una técnica automática usando la noción de accesibilidad	• Distinguir una definición de un lenguaje de una implementación particular de un lenguaje (compilador vs interprete, tiempo de ejecución de la representación de los objetos de datos, etc) [Evaluar] • Distinguir sintaxis y parseo de la semantica y la evaluación [Evaluar] • Bosqueje una representación de bajo nivel de tiempo de ejecución de construcciones del lenguaje base, tales como objetos o cierres (*closures*) [Evaluar] • Explicar cómo las implementaciones de los lenguajes de programación tipicamente organizan la memoria en datos globales, texto, *heap*, y secciones de pila y cómo las características tales como recursión y administración de memoria son mapeados a esté modelo de memoria [Evaluar] • Identificar y corregir las pérdidas de memoria y punteros desreferenciados [Evaluar] • Discutir los beneficios y limitaciones de la recolección de basura (*garbage collection*), incluyendo la noción de accesibilidad [Evaluar]
Readings : [Aho+11], [Lou04a], [App02], [TS98]	

Unit 3: Análisis de sintaxis (10)

Competences Expected:

Topics	Learning Outcomes
• Exploración (análisis léxico) usando expresiones regulares. • Estratégias de análisis incluyendo técnicas de arriba a abajo (top-down) (p.e. descenso recursivo, análisis temprano o LL) y de abajo a arriba (bottom-up) (ej, 'llamadas hacia atrás - bracktracking, o LR); rol de las gramáticas libres de contexto. • Generación de exploradores (scanners) y analizadores a partir de especificaciones declarativas.	• Usar gramáticas formales para especificar la sintaxis de los lenguajes [Evaluar] • Usar herramientas declarativas para generar parseadores y escáneres [Evaluar] • Identificar las características clave en las definiciones de sintaxis: ambiguedad, asociatividad, precedencia [Evaluar]
Readings : [Aho+11], [Lou04a], [App02], [TS98]	

Unit 4: Análisis semántico de compiladores (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Representaciones de programas de alto nivel tales como árboles de sintaxis abstractas. • Alcance y resolución de vínculos. • Revisión de tipos. • Especificaciones declarativas tales como gramáticas atribuídas.	• Implementar analizadores sensibles al contexto y estáticos a nivel de fuente, tales como, verificadores de tipos o resolvedores de identificadores para identificar las ocurrencias de vinculo [Evaluar] • Describir analizadores semanticos usando una gramatica con atributos [Evaluar]
Readings : [Aho+11], [Lou04a], [App02], [TS98]	

Unit 5: Generación de código (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Llamadas a procedimientos y métodos en envío. • Compilación separada; vinculación. • Selección de instrucciones. • Calendarización de instrucciones. • Asignación de registros. • Optimización por rendija (peephole)	• Identificar todos los pasos esenciales para convertir automáticamente código fuente en código emsamblador o otros lenguajes de bajo nivel [Evaluar] • Generar código de bajo nivel para llamadas a funciones en lenguajes modernos [Evaluar] • Discutir por qué la compilación separada requiere convenciones de llamadas uniformes [Evaluar] • Discutir por qué la compilación separada limita la optimización debido a efectos de llamadas desconocidas [Evaluar] • Discutir oportunidades para optimización introducida por la traducción y enfoques para alcanzar la optimización, tales como la selección de la instrucción, planificación de instruccion, asignación de registros y optimización de tipo mirilla (*peephole optimization*) [Evaluar]
Readings : [Aho+11], [Lou04a], [App02], [TS98]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[TS98] Bernard Teufel and Stephanie Schmidt. *Fundamentos de Compiladores*. Addison Wesley Iberoamericana, 1998.

[App02] A. W. Appel. *Modern compiler implementation in Java*. 2.a edición. Cambridge University Press, 2002.

[Lou04a] Kenneth C. Louden. *Compiler Construction: Principles and Practice*. Thomson, 2004.

[Lou04b] Kenneth C. Louden. *Lenguajes de Programacion*. Thomson, 2004.

[Aho+11] Alfred Aho et al. *Compilers Principles Techniques And Tools*. 2nd. ISBN:10-970-26-1133-4. Pearson, 2011.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS231. Networking and Communication (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS231. Networking and Communication
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS2S1. Operating systems . (4^{th} Sem) CS2S1. Operating systems . (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The ever-growing development of communication and information technologies means that there is a marked tendency to establish more computer networks that allow better information management..

In this second course, participants will be introduced to the problems of communication between computers, through the study and implementation of communication protocols such as TCP / IP and the implementation of software on these protocols

5. GOALS

- That the student implements and / or modifies a data communication protocols.
- That the student master the data transmission techniques used by the existing network protocols.
- That the student knows the latest trends in networks that are being applied on the Internet.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Familiarity)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Assessment)

7. TOPICS

Unit 1: Introducción a redes (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Organización de la Internet (proveedores de servicios de Internet, proveedores de contenido, etc) • Técnicas de Switching (por ejemplo, de circuitos, de paquetes) • Piezas físicas de una red, incluidos hosts, routers, switches, ISPs, inalámbrico, LAN, punto de acceso y firewalls. • Principios de capas (encapsulación, multiplexación) • Roles de las diferentes capas (aplicación, transporte, red, enlace de datos, física)	• Articular la organización de la Internet [Familiarizarse] • Listar y definir la terminología de red apropiada [Familiarizarse] • Describir la estructura en capas de una arquitectura típica en red [Familiarizarse] • Identificar los diferentes tipos de complejidad en una red (bordes, núcleo, etc.) [Familiarizarse]
Readings : [KR13]	

Unit 2: Aplicaciones en red (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Esquemas de denominación y dirección (DNS, direcciones IP, identificadores de recursos uniformes, etc) • Las aplicaciones distribuidas (cliente / servidor, peer-to-peer, nube, etc) • HTTP como protocolo de capa de aplicación . • Multiplexación con TCP y UDP • API de Socket	• Listar las diferencias y las relaciones entre los nombres y direcciones en una red [Familiarizarse] • Definir los principios detrás de esquemas de denominación y ubicación del recurso [Familiarizarse] • Implementar una aplicación simple cliente-servidor basada en *sockets* [Usar]
Readings : [KR13]	

Unit 3: Entrega confiable de datos (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Control de errores (técnicas de retransmisión, temporizadores) • El control de flujo (agradecimientos, ventana deslizante) • Problemas de rendimiento (pipelining) • TCP	• Describir el funcionamiento de los protocolos de entrega fiables [Familiarizarse] • Listar los factores que afectan al rendimiento de los protocolos de entrega fiables [Familiarizarse] • Diseñar e implementar un protocolo confiable simple [Usar]
Readings : [KR13]	

Unit 4: Ruteo y reenvío (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Enrutamiento vs reenvío . • Enrutamiento estático . • Protocolo de Internet (IP) • Problemas de escalabilidad (direccionamiento jerárquico)	• Describir la organización de la capa de red [Familiarizarse] • Describir cómo los paquetes se envían en una red IP [Familiarizarse] • Listar las ventajas de escalabilidad de direccionamiento jerárquico [Familiarizarse]
Readings : [KR13]	

Unit 5: Redes de área local (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Problemas de Acceso Múltiple. • Enfoques comunes a Acceso múltiple (exponencial backoff, multiplexación por división de tiempo, etc) • Redes de área local . • Ethernet . • Switching .	• Describir como los paquetes son enviados en una red Ethernet [Familiarizarse] • Describir las diferencias entre IP y Ethernet [Familiarizarse] • Describir las etapas usadas en un enfoque común para el problema de múltiples accesos [Familiarizarse]
Readings : [KR13]	

Unit 6: Asignación de recursos (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Necesidad de asignación de recursos . • Asignación fija (TDM, FDM, WDM) versus la asignación dinámica . • De extremo a extremo frente a las red de enfoque asistida . • Justicia. • Principios del control de congestión. • Enfoques para la congestión (por ejemplo, redes de distribución de contenidos)	• Describir como los recursos pueden ser almacenados en la red [Familiarizarse] • Describir los problemas de congestión en una red grande [Familiarizarse] • Comparar y contrastar las técnicas de almacenamiento estático y dinámico [Familiarizarse] • Comparar y contrastar los enfoques actuales de la congestión [Familiarizarse]
Readings : [KR13]	

Unit 7: Celulares (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Principios de redes celulares. • Redes 802.11 • Problemas en el apoyo a los nodos móviles (agente local)	• Describir la organización de una red inalambrica [Familiarizarse] • Describir como las redes inalámbricas soportan usuarios móviles [Familiarizarse]
Readings : [KR13], [Cha16]	

Unit 8: Redes sociales (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Panorama de las redes sociales. • Ejemplo plataformas de redes sociales. • Estructura de los grafos de redes sociales. • Análisis de redes sociales.	• Discutir los principios fundamentales(como pertenencia, confianza) de una red social [Familiarizarse] • Describir como redes sociales existentes operan [Familiarizarse] • Construir un grafo de una red social a partir de datos de la red [Usar] • Analizar una red social para determinar quienes son las personas importantes [Usar] • Evaluar una determinada interpretación de una pregunta de red social con los datos asociados [Familiarizarse]
Readings : [KR13], [Kad11]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Kad11] Charles Kadushin. *Understanding Social Networks: Theories, Concepts, And Findings*. Oxford University Press, Usa; 1 edition, 2011.

[KR13] J.F. Kurose and K.W. Ross. *Computer Networking: A Top-down Approach*. 7th. Always learning. Pearson, 2013.

[Cha16] Paresh Chayapathi Rajendra; Syed F. Hassan; Shah. *Network Functions Virtualization (NFV) with a Touch of SDN*. Addison-Wesley Professional; 1 edition, 2016.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS261. Artificial Intelligence (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS261. Artificial Intelligence
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	MA203. Statistics and Probabilities. (4^{th} Sem) MA203. Statistics and Probabilities. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Research in Artificial Intelligence has led to the development of numerous relevant tonic, aimed at the automation of human intelligence, giving a panoramic view of different algorithms that simulate the different aspects of the behavior and the intelligence of the human being.

5. GOALS

- Evaluate the possibilities of simulation of intelligence, for which the techniques of knowledge modeling will be studied.
- Build a notion of intelligence that later supports the tasks of your simulation.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Familiarity)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Familiarity)

7. TOPICS

Unit 1: Cuestiones fundamentales (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Descripción general de los problemas de Inteligencia Artificial, ejemplos recientes de aplicaciones de Inteligencia artificial. • ¿Qué es comportamiento inteligente? – El Test de Turing – Razonamiento Racional versus No Racional • Características del Problema: – Observable completamente versus observable parcialmente – Individual versus multi-agente – Deterministico versus estocástico – Estático versus dinámico – Discreto versus continuo • Naturaleza de agentes: – Autónomo versus semi-autónomo – Reflexivo, basado en objetivos, y basado en utilidad – La importancia en percepción e interacciones con el entorno • Cuestiones filosóficas y éticas.	• Describir el test de Turing y el experimento pensado cuarto chino" (*Chinese Room*) [Usar] • Determinando las caracteristicas de un problema dado que la Inteligencia Artificial deberian resolver [Usar]
Readings : [De 06], [Pon+14]	

Unit 2: Agentes (2)

Competences Expected:

Topics	**Learning Outcomes**

• Definición de Agentes
• Arquitectura de agentes (Ej. reactivo, en capa, cognitivo)
• Teoría de agentes
• Racionalidad, teoría de juegos:
  – Agentes de decisión teórica
  – Procesos de decisión de Markov (MDP)
• Agentes de Software, asistentes personales, y acceso a información:
  – Agentes colaborativos
  – Agentes de recolección de información
  – Agentes creíbles (carácter sintético, modelamiento de emociones en agentes)
• Agentes de aprendizaje
• Sistemas Multi-agente
  – Agentes Colaborativos
  – Equipos de Agentes
  – Agentes Competitivos (ej., subastas, votaciones)
  – Sistemas de enjambre y modelos biológicamente inspirados
	• Lista las características que definen un agente inteligente [Usar]
	• Describe y contrasta las arquitecturas de agente estándares [Usar]
	• Describe las aplicaciones de teoría de agentes para dominios como agentes de software, asistentes personales, y agentes creibles [Usar]
	• Describe los paradigmas primarios usados por agentes de aprendizaje [Usar]
	• Demuestra mediante ejemplos adecuados como los sistemas multi-agente soportan interacción entre agentes [Usar]

Readings : [Nil01], [RN03], [Pon+14]

Unit 3: Estrategias de búsquedas básicas (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Espacios de Problemas (estados, metas y operadores), solución de problemas mediante búsqueda. • Factored representation (factoring state hacia variables) • Uninformed search (breadth-first, depth-first, depth-first with iterative deepening) • Heurísticas y búsqueda informada (hill-climbing, generic best-first, A*) • El espacio y el tiempo de la eficiencia de búsqueda. • Dos jugadores juegos (introducción a la búsqueda minimax). • Satisfacción de restricciones (backtracking y métodos de búsqueda local).	• Formula el espacio eficiente de un problema para un caso expresado en lenguaje natural (ejm. Inglés) en términos de estados de inicio y final, así como sus operadores [Usar] • Describe el rol de las heurísticas y describe los intercambios entre completitud, óptimo, complejidad de tiempo, y complejidad de espacio [Usar] • Describe el problema de la explosión combinatoria del espacio de búsqueda y sus consecuencias [Usar] • Compara y contrasta tópicos de búsqueda básica con temas jugabilidad de juegos [Usar]
Readings : [Nil01], [Pon+14]	

Unit 4: Búsqueda Avanzada (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Búsqueda estocástica: – Simulated annealing – Algoritmos genéticos – Búsqueda de árbol Monte-Carlo • Construcción de árboles de búsqueda, espacio de búsqueda dinámico, explosión combinatoria del espacio de búsqueda. • Implementación de búsqueda A *, búsqueda en haz. • Búsqueda Minimax, poda alfa-beta. • Búsqueda Expectimax (MDP-Solving) y los nodos de azar.	• Diseñar e implementar una solución a un problema con algoritmo genético [Usar] • Diseñar e implementar un esquema de recocido simulado (*simulated annealing*) para evitar mínimos locales en un problema [Usar] • Diseñar e implementar una búsqueda A* y búsqueda en haz (*beam search*) para solucionar un problema [Usar] • Aplicar búsqueda minimax con poda alfa-beta para simplifiar el espacio de búsqueda en un juego con dos jugadores [Usar] • Comparar y contrastar los algoritmos genéticos con técnicas clásicas de búsqueda [Usar] • Comparar y contrastar la aplicabilidad de varias heurísticas de búsqueda, para un determinado problema [Usar]
Readings : [Gol89], [Nil01], [RN03], [Pon+14]	

Unit 5: Razonamiento Bajo Incertidumbre (18)

Competences Expected:

Topics	**Learning Outcomes**

• Revisión de Probabilidad Básica
• Variables aleatorias y distribuciones de probabilidad:
  – Axiomas de probabilidad
  – Inferencia probabilística
  – Regla de Bayes
• Independecia Condicional
• Representaciones del conocimiento:
  – Redes bayesianas
  ∗ Inferencia exacta y su complejidad
  ∗ Métodos de Muestreo aleatorio (Monte Carlo) (p.e. Muestreo de Gibbs)
  – Redes Markov
  – Modelos de probabilidad relacional
  – Modelos ocultos de Markov
	• Aplicar la regla de Bayes para determinar el cumplimiento de una hipótesis [Usar]
	• Explicar cómo al tener independencia condicional permite una gran eficiencia en sistemas probabilísticos [Usar]
	• Identificar ejemplos de representación de conocimiento para razonamiento bajo incertidumbre [Usar]
	• Indicar la complejidad de la inferencia exacta. Identificar métodos para inferencia aproximada [Usar]

Readings : [KF09], [RN03]

Unit 6: Aprendizaje Automático Básico (4)

Competences Expected:

Topics	**Learning Outcomes**
• Definición y ejemplos de la extensa variedad de tareas de aprendizaje de máquina, incluida la clasificación. • Aprendizaje inductivo • Aprendizaje simple basado en estadísticas, como el clasificador ingenuo de Bayes, árboles de decisión. • El problema exceso de ajuste. • Medicion clasificada con exactitud.	• Listar las diferencias entre los tres principales tipos de aprendizaje: supervisado, no supervisado y por refuerzo [Usar] • Identificar ejemplos de tareas de clasificación, considerando las características de entrada disponibles y las salidas a ser predecidas [Usar] • Explicar la diferencia entre aprendizaje inductivo y deductivo [Usar] • Describir el sobre ajuste (*overfitting*) en el contexto de un problema [Usar] • Aplicar un algoritmo de aprendizaje estadístico simple como el Clasificador Naive Bayesiano e un problema de clasificacion y medirla precisión del clasificador [Usar]

Readings : [Mit98], [RN03], [Pon+14]

Unit 7: Aprendizaje de máquina avanzado (20)

Competences Expected:

Topics	**Learning Outcomes**

• Definición y ejemplos de una amplia variedad de tareas de aprendizaje de máquina
• Aprendizaje general basado en estadística, estimación de parámetros (máxima probabilidad)
• Programación lógica inductiva (*Inductive logic programming ILP*)
• Aprendizaje supervisado
  – Aprendizaje basado en árboles de decisión
  – Aprendizaje basado en redes neuronales
  – Aprendizaje basado en máquinas de soporte vectorial (*Support vector machines SVMs*)
• Aprendizaje y *clustering* no supervisado
  – EM
  – K-means
  – Mapas auto-organizados
• Aprendizaje semi-supervisado.
• Aprendizaje de modelos gráficos
• Evaluación del desempeño (tal como cross-validation, area bajo la curva ROC)
• Aplicación de algoritmos Machine Learning para Minería de datos.
	• Explica las diferencias entre los tres estilos de aprendizaje: supervisado, por refuerzo y no supervisado [Usar]
	• Implementa algoritmos simples para el aprendizaje supervisado, aprendizaje por refuerzo, y aprendizaje no supervisado [Usar]
	• Determina cuál de los tres estilos de aprendizaje es el apropiado para el dominio de un problema en particular [Usar]
	• Compara y contrasta cada una de las siguientes técnicas, dando ejemplo de cuando una estrategia es la mejor: árboles de decisión, redes neuronales, y redes bayesianas [Usar]
	• Evalúa el rendimiento de un sistema de aprendizaje simple en un conjunto de datos reales [Usar]
	• Describe el estado del arte en la teoría del aprendizaje, incluyendo sus logros y limitantes [Usar]
	• Explica el problema del sobreajuste, conjuntamente con técnicas para determinar y manejar el problema [Usar]

Readings : [RN03], [KF09], [Mur12]

Unit 8: Procesamiento del Lenguaje Natural (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Gramaticas determinísticas y estocásticas • Algoritmos de parseo – Gramáticas libres de contexto (CFGs) y cuadros de parseo (e.g. Cocke-Younger-Kasami CYK) – CFGs probabilísticos y ponderados CYK • Representación del significado / Semántica – Representación de conocimiento basado en lógica – Roles semánticos – Representaciones temporales – Creencias, deseos e intenciones • Metodos basados en el corpus • N-gramas y Modelos ocultos de Markov (HMMs) • Suavizado y back-off • Ejemplos de uso: POS etiquetado y morfologia • Recuperación de la información: – Modelo de espacio vectorial ∗ TF & IDF – Precision y cobertura • Extracción de información • Traducción de lenguaje • Clasificación y categorización de texto: – Modelo de bolsa de palabras	• Define y contrasta gramáticas de tipo estocásticas y determinísticas, dando ejemplos y demostrando como adecuar cada una de ellas [Usar] • Simula, aplica, o implementa algoritmos clásicos y estocásticos para el parseo de un lenguaje natural [Usar] • Identifica los retos de la representación del significado [Usar] • Lista las ventajas de usar corpus estándares. Identifica ejemplos de corpus actuales para una variedad de tareas de PLN [Usar] • Identifica técnicas para la recuperación de la información, traducción de lenguajes, y clasificación de textos [Usar]
Readings : [Nil01], [RN03], [Pon+14]	

Unit 9: Visión y percepción por computador (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Visión Computacional – Adquisición de imágenes, representación, procesamiento y propiedades – Representación de formas, reconocimiento y segmentación de objetos – Análisis de movimiento • Modularidad en reconocimiento. • Enfoques de reconocimiento de patrones – Algoritmos de clasificación y medidas de calidad de la clasificación. – Técnicas estadísticas.	• Resumir la importancia del reconocimiento de imagenes y objetos en Inteligencia Artificial (AI) e indicar varias aplicaciones significativas de esta tecnologia [Usar] • Listar al menos tres aproximaciones de segmentación de imágenes, tales como algoritmos de limites (thresholding), basado en el borde y basado en regiones, junto con sus características definitorias, fortalezas y debilidades [Usar] • Implementar reconocimiento de objetos en 2d basados en la representación del contorno y/o regiones basadas en formas [Usar] • Proporcionar al menos dos ejemplos de transformación de una fuente de datos de un dominio sensorial a otro, ejemplo, datos táctiles interpretados como imágenes en 2d de una sola banda [Usar] • Implementar un algoritmo para la extracción de caracteristicas en información real, ejemplo, un detector de bordes o esquinas para imágenes o vectores de coeficientes de Fourier describiendo una pequeña porción de señal de audio [Usar] • Implementar un algoritmo de clasificación que segmenta percepciones de entrada en categorias de salida y evalua cuantitativamente la clasificación resultante [Usar] • Evaluar el desempeño de la función de extracción subyacente, en relación con al menos una aproximación alternativa posible (ya sea implementado o no) en su contribución a la tarea de clasificación (8) anterior [Usar]
Readings : [Nil01], [RN03], [Pon+14]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Gol89] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[Mit98] M. Mitchell. *An introduction to genetic algorithms*. The MIT press, 1998.

[Nil01] Nils Nilsson. *Inteligencia Artificial: Una nueva visión*. McGraw-Hill, 2001.

[RN03] Stuart Russell and Peter Norvig. *Inteligencia Artifical: Un enfoque moderno*. Prentice Hall, 2003.

[De 06] L.N. De Castro. *Fundamentals of natural computing: basic concepts, algorithms, and applications*. CRC Press, 2006.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[Pon+14] Julio Ponce-Gallegos et al. *Inteligencia Artificial*. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS292. Software Engineering II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS292. Software Engineering II
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS291. Software Engineering I. (5^{th} Sem) CS291. Software Engineering I. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The topics of this course extend the ideas of software design and development from the introduction sequence to programming to encompass the problems encountered in large-scale projects. It is a broader and more complete view of Software Engineering appreciated from a Project point of view.

5. GOALS

- Enable students to be part of and define software development teams facing real-world problems.
- familiarize the students with the process of administering a software project in such a way as to be able to create, improve and use tools and metrics that allow them to carry out the estimation and monitoring of a software project
- Create, evaluate and execute a test plan for medium-sized code segments, Distinguish between different types of tests, lay the foundation for creating, improve test procedures and tools for these purposes
- Select with justification an appropriate set of tools to support the development of a range of software products.
- Create, improve and use existing patterns for software maintenance. Disclose features and design patterns for software reuse.
- Identify and discuss different specialized systems, create, improve and use specialized standards for the design, implementation, maintenance and testing of specialized systems.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

3) Communicate effectively in a variety of professional contexts.. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Herramientas y Entornos (12)	
Competences Expected:	
Topics	**Learning Outcomes**

• Administración de configuración de software y control de versiones.
• Administración de despliegues.
• Análisis de requerimientos y herramientas para modelado del diseño.
• Herramientas de *testing* incluyendo herramientas de análisis estático y dinámico.
• Entornos de programación que automatizan el proceso de construcción de partes de programa (ejem., construcciones automatizadas)
  – Integración continua.
• Mecanismos y conceptos de herramientas de integración.
	• Administración de configuración de software y control de versiones. [Usar]
	• Administración de despliegues. [Usar]
	• Análisis de requerimientos y herramientas para modelado del diseño. [Usar]
	• Herramientas de *testing* incluyendo herramientas de análisis estático y dinámico. [Usar]
	• Entornos de programación que automatizan el proceso de construcción de partes de programa (ejem., construcciones automatizadas)
	  – Integración continua.
	[Usar]
	• Mecanismos y conceptos de herramientas de integración. [Usar]

Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 2: Verificación y Validación de Software (12)

Competences Expected:

Topics	**Learning Outcomes**
• Verificación y validación de conceptos. • Inspecciones, revisiones, auditorias. • Tipos de pruebas, incluyendo la interfas humano computador, usabildiad, confiabilidad , seguridad,desempeño para la especificación. • Fundamentos de testeo: – Pruebas de Unit, integración, validación y de Sistema – Creación de plan de pruebas y generación de casos de test – Técnicas de test de caja negra y caja blanca – Test de regresión y automatización de pruebas • Seguimiento de defectos. • Limitaciones de testeo en dominios particulares, tales como sistemas paralelos o críticos en cuanto a seguridad. • Enfoques estáticos y enfoques dinámicos para la verificación. • Desarrollo basado en pruebas. • Plan de Validación, documentación para validación. • Pruebas Orientadas a Objetos, Sistema de Pruebas. • Verificación y validación de artefactos no codificados (documentación, archivos de ayuda, materiales de entrenamiento) • Logeo fallido, error crítico y apoyo técnico para dichas actividades. • Estimación fallida y terminación de las pruebas que incluye la envios por defecto.	• Distinguir entre la validación y verificación del programa [Usar] • Describir el papel que las herramientas pueden desempeñar en la validación de software [Usar] • Realizar, como parte de una actividad de equipo, una inspección de un segmento de código de tamaño medio [Usar] • Describir y distinguir entre diferentes tipos y niveles de pruebas (unitaria, integracion, sistemas y aceptacion) [Usar] • Describir tecnicas para identificar casos de prueba representativos para integracion, regresion y pruebas del sistema [Usar] • Crear y documentar un conjunto de pruebas para un segmento de código de mediano tamaño [Usar] • Describir cómo seleccionar buenas pruebas de regresión y automatizarlas [Usar] • Utilizar una herramienta de seguimiento de defectos para manejar defectos de software en un pequeño proyecto de software [Usar] • Discutir las limitaciones de las pruebas en un dominio particular [Usar] • Evaluar un banco de pruebas (*a test suite*) para un segmento de código de tamaño medio [Usar] • Comparar los enfoques estáticos y dinámicos para la verificación [Usar] • Identificar los principios fundamentales de los métodos de desarrollo basado en pruebas y explicar el papel de las pruebas automatizadas en estos métodos [Usar] • Discutir las limitaciones de las pruebas en un dominio particular [Usar] • Describir las técnicas para la verificación y validación de los artefactos de no código [Usar] • Describir los enfoques para la estimación de fallos [Usar] • Estimar el número de fallos en una pequeña aplicación de software basada en la densidad de defectos y siembra de errores [Usar] • Realizar una inspección o revisión del de código fuente de un software para un proyecto de software de tamaño pequeño o mediano [Usar]

Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]

Unit 3: Evolución de Software (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Desarrollo de Software en el contexto de código grande pre existente – Cambios de software – Preocupaciones y ubicación de preocupaciones – *Refactoring* • Evolución de Software. • Características de Software mantenible. • Sistemas de Reingeniería. • Reuso de Software. – Segmentos de código – Bibliotecas y *frameworks* – Componentes – Líneas de Producto	• Identificar los problemas principales asociados con la evolución del software y explicar su impacto en el ciclo de vida del software [Usar] • Estimar el impacto del cambio de requerimientos en productos existentes de tamaño medio [Usar] • Usar refactorización en el proceso de modificación de un componente de sosftware [Usar] • Estudiar los desafios de mejorar sistemas en un entorno cambiante [Usar] • Perfilar los procesos de pruebas de regresión y su rol en el manejo de versiones [Usar] • Estudiar las ventajas y desventajas de diferentes tipos de niveles de confiabilidad [Usar]
Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 4: Gestión de Proyectos de Software (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• La participación del equipo: – Procesos elemento del equipo, incluyendo responsabilidades de tarea, la estructura de reuniones y horario de trabajo – Roles y responsabilidades en un equipo de software – Equipo de resolución de conflictos – Los riesgos asociados con los equipos virtuales (comunicación, la percepción, la estructura) • Estimación de esfuerzo (a nivel personal) • Riesgo. – El papel del riesgo en el ciclo de vida – Categorías elemento de riesgo, incluyendo la seguridad, la seguridad, mercado, finanzas, tecnología, las personas, la calidad, la estructura y el proceso de • Gestión de equipos: – Organización de equipo y la toma de decisiones – Roles de identificación y asignación – Individual y el desempeño del equipo de evaluación • Gestión de proyectos: – Programación y seguimiento de elementos – Herramientas de gestión de proyectos – Análisis de Costo/Beneficio • Software de medición y técnicas de estimación. • Aseguramiento de la calidad del software y el rol de las mediciones. • Riesgo. – Identificación de riesgos y gestión. – Análisis riesgo y evaluación. – La tolerancia al riesgo (por ejemplo, riesgo adverso, riesgo neutral, la búsqueda de riesgo) – Planificación de Riesgo • En todo el sistema de aproximación al riesgo, incluyendo riesgos asociados con herramientas.	• Discutir los comportamientos comunes que contribuyen al buen funcionamiento de un equipo [Usar] • Crear y seguir un programa para una reunión del equipo [Usar] • Identificar y justificar las funciones necesarias en un equipo de desarrollo de software [Usar] • Entender las fuentes, obstáculos y beneficios potenciales de un conflicto de equipo [Usar] • Aplicar una estrategia de resolución de conflictos en un ambiente de equipo [Usar] • Utilizar un método ad hoc para estimar el esfuerzo de desarrollo del software (ejemplo, tiempo) y comparar con el esfuerzo actual requerido [Usar] • Listar varios ejemplos de los riesgos del software [Usar] • Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Usar] • Describir las diferentes categorías de riesgo en los sistemas de software [Usar] • Demostrar a través de la colaboración de proyectos de equipo los elementos centrales de la contrucción de equipos y gestión de equipos [Usar]
Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the

different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Blu92] Bruce I. Blum. *Software Engineering: A Holistic View*. 7th. Oxford University Press US, May 1992.

[Mon96] Carlo Montangero. *Software Process Technology*. Springer, Sept. 1996.

[Con00] R Conradi. *Software Process Technology*. Springer, Mar. 2000.

[WK00] Yingxu Wang and Graham King. *Software Engineering Processes: Principles and Applications*. CRC Press, Apr. 2000.

[Amb01] Vincenzo Ambriola. *Software Process Technology*. Springer, July 2001.

[PS01] John W. Priest and Jose M. Sanchez. *Product Development and Design for Manufacturing*. Marcel Dekker, Jan. 2001.

[WA02] Daniel R. Windle and L. Rene Abreo. *Software Requirements Using the Unified Process*. Prentice Hall, Aug. 2002.

[Oqu03] Flavio Oquendo. *Software Process Technology*. Springer, Sept. 2003.

[Key04] Jessica Keyes. *Software Configuration Management*. CRC Press, Feb. 2004.

[Pre04] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. 6th. McGraw-Hill, Mar. 2004.

[Sch04] Stephen R Schach. *Object-Oriented and Classical Software Engineering*. McGraw-Hill, Jan. 2004.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS311. Competitive Programming (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS311. Competitive Programming
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS212. Analysis and Design of Algorithms. (5^{th} Sem) CS212. Analysis and Design of Algorithms. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Competitive Programming combines problem-solving challenges with the fun of competing with others. It teaches participants to think faster and develop problem-solving skills that are in high demand in the industry. This course will teach you to solve algorithmic problems quickly by combining theory of algorithms and data structures with practice solving problems.

5. GOALS

- That the student uses techniques of data structures and complex algorithms..
- That the student apply the concepts learned for the application on a real problem.
- That the student investigate the possibility of creating a new algorithm and / or new technique to solve a real problem.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Introduction (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Introduction to Competetive Programming • Computacional model • Runtime and space complexity • Recurrence and recursion • Divide and conquer	• Identify and learn how to use the resources in the Random Access Machine (RAM) computacional model. [Usar] • Compute the runtime and space complexity for written algorithms. [Usar] • Compute the recurrence relations for recursive algorithms. [Usar] • Solve problems related to searching and sorting. [Usar] • Learning to select the right algorithms for divide-and-conquer problems. [Usar] • Design new algorithms for real-world problem solving.[Usar]
Readings : [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

Unit 2: Data structure (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Arrays and strings problems • Linked lists problems • Stacks and queues problems • Trees problems • Hash tables problems • Heaps problems	• Recognize different data structures, their complexities, uses and restrictions.[Usar] • Identify the type of data structure appropriate to the resolution of the problem. [Usar] • Recognize types of problems associated with operations on data structures such as searching, inserting, deleting and updating.[Usar]
Readings : [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

Unit 3: Algorithmic Design Paradigms (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Brute force • Divide and conquer • Backtracking • Greedy • Dynamic Programming	• Learning the different algorithhmic design paradigms.[Usar] • Learning to select the right algorithms for different problems applying different algorithhmic design paradigms.[Usar]
Readings : [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

Unit 4: Graphs (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Graphs transversal • Graphs aplications • Shortest path • Networks and flows	• Identify problems classified as graph problems. [Usar] • Learn how to select the right algorithms for network problems (transversal, MST, shortest-path, network and flows). [Usar]
Readings : [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

Unit 5: Advanced topics (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Number theory • Probabilities and combinations • String algorithms (tries, string hashing, z-algorithm) • Geometric algorithms	• Learning to select the right algorithms for problems in number theory and mathematics as they are important in competitive programming. [Usar] • Learning to select the right algorithms for problems about probabilities and combinations, strings and computational geometry. [Usar]
Readings : [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

Unit 6: Domain specific problems (20)	
Competences Expected:	
Topics	**Learning Outcomes**
• Latency and throughput • Parallelism • Networks • Storage • High availability • Caching • Proxies • Load balancers • Key-value stores • Replicating and sharing • Leader election • Rate limiting • Logging and monitoring	• Learning to design systems for different domain-specific problems by applying knowledge about networks, distributed computing, high availability, storage and system architecture.[Usar]
Readings : [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Mig03] Steve Skiena Miguel A. Revilla. *Programming Challenges: The Programming Contest Training Manual*. Springer, May 2003.

[Cor+09] T. H. Cormen et al. *Introduction to Algorithms*. MIT Press, 2009.

[ALP12] A. Aziz, T.H. Lee, and A. Prakash. *Elements of Programming Interviews: The Insiders' Guide*. ElementsOfProgrammingInterviews.com, 2012. URL: `https://books.google.com.pe/books?id=y6FLBQAAQBAJ`.

[Hal13] Steven Halim. *Competitive Programming*. 3 rd. Lulu, 2013.

[Laa17] Antti Laaksonen. *Guide to Competitive Programming: Learning and Improving Algorithms Through Contests*. Stringer, 2017.

[Kul19] Alexander S. Kulikov. *Learning Algorithms Through Programming and Puzzle Solving*. Active Learning Technologies, 2019.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS312. Advanced Data Structures (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS312. Advanced Data Structures
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS212. Analysis and Design of Algorithms. (5^{th} Sem) CS212. Analysis and Design of Algorithms. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Los algoritmos y estructuras de datos son una parte fundamental de la ciencia de la computación que nos permiten organizar la información de una manera más eficiente, por lo que es importante para todo profesional del área tener una sólida formación en este aspecto.

En el curso de estructuras de datos avanzadas nuestro objetivo es que el alumno conozca y analize estructuras complejas, como los Métodos de Acceso Multidimensional, Métodos de Acceso Espacio-Temporal y Métodos de Acceso Métrico, etc.

5. GOALS

- Que el alumno entienda, diseñe, implemente, aplique y proponga estructuras de datos innovadoras para solucionar problemas relacionados al tratamiento de datos multidimensionales, recuperación de información por similitud, motores de búsqueda y otros problemas computacionales.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Familiarity)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Familiarity)

7. TOPICS

Unit 1: Multidimensional Data (16)	
Competences Expected: 1,2	
Topics	**Learning Outcomes**
• Introducción al curso. • Introducción a datos multidimensionales. • Maldición de la dimensionalidad.	• Introducir la transcendencia de la representación multidimensional de datos. [Usar] • Entender la complejidad de lidiar con datos multidimensional y de alta dimensión.[Usar] • Entender la maldición de la dimensionalidad, y su impacto en el indizado de grandes volúmenes de datos.[Usar] • Presentar y discutir aplicaciones reales de datos multidimensionales en motores de búsqueda.[Usar]
Readings : [Cua+04], [Knu07a], [Knu07b], [Gam+94]	

Unit 2: Multidimensional Acces Data Structures (16)	
Competences Expected: 1,2	
Topics	**Learning Outcomes**
• Introducción a estructuras de datos espaciales. • Estructuras espaciales, Quadtree,Octree y visualización. • Kd-Tree. • Introducción a R-Tress. • R tree (Guttman). • R+ tree. • R* tree. • Variación R*-tree y relación con paginación y tamaño de bloques. • X-tree.	• Introducir los fundamentos teóricos de estructuras de datos espaciales. • Entender los beneficios y limitaciones deestructuras de datos espaciales basadas en árbol. • Implementar diferentes estructuras de datos para el indizado de grandes volumenes de datos. • Entender los fundamentos e implementar estrategias de búsqueda como vecinos mas próximos y búsquedas por rango.
Readings : [Cua+04], [Knu07a], [Knu07b], [Gam+94]	

Unit 3: Approximate Access Methods (20)	
Competences Expected: 1,2	
Topics	**Learning Outcomes**
• Métodos de Acceso Métrico para distancias discretas • Métodos de Acceso Métrico para distancias continuas	• Que el alumno entienda conozca e implemente algunos métodos de acceso métrico[Usar] • Que el alumno entienda la importancia de estos Métodos de Acceso para la Recuperación de Información por Similitud[Usar]
Readings : [Cua+04], [Knu07a], [Knu07b], [Gam+94]	

Unit 4: Métodos de Acceso Aproximados (20)	
Competences Expected: 1,2	
Topics	**Learning Outcomes**
• Space Filling Curves: Hilbert curve y Z-order • Proyecciones y complejidad. • Locally sensitive hashing (LSH)	• Entender, conocer e implementar algunos métodos de acceso aproximados. • Entender la importancia de estos métodos de Acceso para la recuperación de información por similitud en entornos donde la escalabilidad sea una factor muy importante.
Readings : [Sam06], [PI06], [Zez+07]	

Unit 5: Clustering (8)	
Competences Expected: 1,2	
Topics	**Learning Outcomes**
• Introducción a Clustering. • Kmeans y DBScan. • Clustering Applications. • Clustering Ensemble.	• Introducir los fundamentos teóricos para el clustering de datos multidimensionales. • Implementar diferentes estrategias para el clustering de datos multidimensionales, como basados en partición, en jerarquía o en densidad. • Entender los fundamentos, aplicaciones e implementar ensambles de métodos de clustering. • Implementar ensambles de métodos de clustering con datos reales.
Readings : [Cua+04], [Knu07a], [Knu07b], [Gam+94]	

Unit 6: Temporal Data Structures (8)	
Competences Expected: 1,2	
Topics	**Learning Outcomes**
• Introducción a Estructuras de datos temporales. • Versionando la estructura de Datos. • Persistencia • Retroactividad	• Introducir los fundamentos teóricos de estructuras de datos temporales. • Entender, discutir e implementar Persistencia y sus tipos. • Entender, discutir e implementar Retroactividad y sus tipos. • Entender y discutir los beneficios y limitaciones entre persistencia y retroactividad.
Readings : [Cua+04], [Knu07a], [Knu07b], [Gam+94]	

Unit 7: Final Talks (8)	
Competences Expected: 1,2	
Topics	**Learning Outcomes**
• Seminarios de trabajo de investigación.	• Investigar sobre nuevos métodos para el indizado de grandes volumenes de datos complejos. • Presentar y dirigir la discusión sobre métodos para indizados de Big Data investigado.
Readings : [Cua+04], [Knu07a], [Knu07b], [Gam+94]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Gam+94] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Computing Series. ISBN-10: 0201633612. Addison-Wesley Professional, Nov. 1994.

[Cua+04] Ernesto Cuadros-Vargas et al. "Implementing data structures: An incremental approach". http://socios.spc.org.pe/ecuadros/cursos/pdfs/. 2004.

[PI06] Trevor Darrell PGregory Shakhnarovich and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. 1st. ISBN 0-262-19547-X. MIT Press, Mar. 2006.

[Sam06] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Illustrated. Elsevier/Morgan Kaufmann, Aug. 2006. URL: `http://books.google.com.pe/books?id=vO-NRRKHG84C`.

[Knu07a] Donald Ervin Knuth. *The Art of Computer Programming, Fundamental Algorithms*. 3rd. Vol. I. 0-201-89683-4. Addison-Wesley, Feb. 2007.

[Knu07b] Donald Ervin Knuth. *The Art of Computer Programming, Sorting and Searching*. 2nd. Vol. II. 0-201-89685-0. Addison-Wesley, Feb. 2007.

[Zez+07] Pavel Zezula et al. *Similarity Search: The Metric Space Approach*. 1st. ISBN-10: 0387291466. Springer, Nov. 2007.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS393. Information systems (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS393. Information systems
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS291. Software Engineering I. (5^{th} Sem) CS291. Software Engineering I. (5^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Analyze techniques for the correct implementation of scalable, robust, reliable and efficient information systems in organizations.

5. GOALS

- Implement correctly (scalable, robust, reliable and efficient) Information Systems in organizations.

6. COMPETENCES

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Introduction (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Introduction to information management. • Software for information management. • Technology for information management.	• Correctly apply technology for information management [Evaluar]
Readings : [Som17], [PM15], [LL17]	

Unit 2: Strategy (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Strategy for information management. • Strategy for knowledge management • Strategy for information system.	• Apply and evaluate correctly management strategies [Evaluar]
Readings : [Som17], [PM15]	

Unit 3: Implementation (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Management Information Systems Development. • Change management • Information Architecture	• Implement and correctly evaluate implementation strategies [Evaluar]
Readings : [Som17], [PM15]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[PM15] Roger S. Pressman and Bruce Maxim. *Software Engineering: A Practitioner's Approach*. 8th. McGraw-Hill, Jan. 2015.

[LL17] Kenneth C. Laudon and Jane P. Laudon. *Management Information Systems: Managing the Digital Firm*. 15th. Pearson, Mar. 2017.

[Som17] Ian Sommerville. *Software Engineering*. 10th. Pearson, Mar. 2017.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

MA307. Mathematics applied to computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	MA307. Mathematics applied to computing
2.2 Semester	:	6^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	MA201. Calculus II. (4^{th} Sem) MA201. Calculus II. (4^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Este curso es importante porque desarrolla tópicos del Álgebra Lineal y de Ecuaciones Diferenciales Ordinarias útiles en todas aquellas áreas de la ciencia de la computación donde se trabaja con sistemas lineales y sistemas dinámicos.

5. GOALS

- Que el alumno tenga la base matemática para el modelamiento de sistemas lineales y sistemas dinámicos necesarios en el Área de Computación Gráfica e Inteligencia Artificial.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: (0)	
Competences Expected:	
Topics	**Learning Outcomes**
• Espacios vectoriales. • Independencia, base y dimensión. • Dimensiones y ortogonalidad de los cuatro subespacios. • Aproximaciones por mínimos cuadrados. • Proyecciones • Bases ortogonales y Gram-Schmidt	• Identificar espacios generados por vectores linealmente independientes[Usar] • Construir conjuntos de vectores ortogonales[Usar] • Aproximar funciones por polinomios trigonométricos[Usar]
Readings : [Str03], [Apó73]	

Unit 2: (0)	
Competences Expected:	
Topics	**Learning Outcomes**
• Concepto de transformación lineal. • Matriz de una transformación lineal. • Cambio de base. • Diagonalización y pseudoinversa	• Determinar el núcleo y la imagen de una transformación[Usar] • Construir la matriz de una transformación[Usar] • Determinar la matriz de cambio de base[Usar]
Readings : [Str03], [Apó73]	

Unit 3: (0)	
Competences Expected:	
Topics	**Learning Outcomes**
• Diagonalización de una matriz • Matrices simétricas • Matrices definidas positivas • Matrices similares • La descomposición de valor singular	• Encontrar la representación diagonal de una matriz[Usar] • Determinar la similaridad entre matrices[Usar] • Reducir una forma cuadrática real a diagonal[Usar]
Readings : [Str03], [Apó73]	

Unit 4: (0)	
Competences Expected:	
Topics	**Learning Outcomes**
• Exponencial de una matriz • Teoremas de existencia y unicidad para sistemas lineales homogéneos con coeficientes constantes • Sistemas lineales no homogéneas con coeficientes constantes.	• Hallar la solución general de un sistema lineal no homogéneo[Usar] • Resolver problemas donde intervengan sistemas de ecuaciones diferenciales[Usar]
Readings : [Zil02], [Apó73]	

Unit 5: (0)	
Competences Expected:	
Topics	**Learning Outcomes**
• Sistemas dinámicos • El teorema fundamental • Existencia y unicidad • El flujo de una ecuación diferencial	• Discutir la existencia y la unicidad de una ecuación diferencial[Usar] • Analizar la continuidad de las soluciones[Usar] • Estudiar la prolongación de una solución[Usar]
Readings : [HS74]	

Unit 6: (0)	
Competences Expected:	
Topics	**Learning Outcomes**
• Estabilidad • Funciones de Liapunov • Sistemas gradientes	• Analizar la estabilidad de una solución[Usar] • Hallar la función de Liapunov para puntos de equilibrio[Usar] • Trazar el retrato de fase un flujo gradiente[Usar]
Readings : [Zil02], [HS74]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Apó73] Tom M Apóstol. *Calculus Vol II*. Editorial Reverté, 1973.

[HS74] Morris W. Hirsh and Stephen Smale. *Differential Equatons, Dynamical Systems, and Linear Álgebra*. Academia Press, 1974.

[Zil02] Dennis G. Zill. *Ecuaciones Diferenciales con Problemas de Valores en la Frontera*. Thomson Learning, 2002.

[Str03] Gilbert Strang. *Introduction to Linear Algebra, 3rd edición*. Wellesley-Cambridge Press, 2003.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS2H1. User Experience (UX) (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS2H1. User Experience (UX)
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS393. Information systems. (6^{th} Sem) CS393. Information systems. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Language has been one of the most significant creations of humanity. From body language and gesture, through verbal and written communication, to iconic symbolic codes and others, it has made possible complex interactions Among humans and facilitated considerably the communication of information. With the invention of automatic and semi-automatic devices, including computers, The need for languages or interfaces to be able to interact with them, has gained great importance. The utility of the software, coupled with user satisfaction and increased productivity, depends on the effectiveness of the User-Computer Interface. So much so, that often the interface is the most important factor in the success and failure of any computer system. The design and implementation of appropriate Human-Computer Interfaces, which in addition to complying with the technical requirements and the transactional logic of the application, consider the subtle psychological implications, sciences and user facilities, It consumes a good part of the life cycle of a software project, and requires specialized skills, both for the construction of the same, and for the performance of usability tests.

5. GOALS

- Know and apply criteria of usability and accessibility to the design and construction of human-computer interfaces, always looking for technology to adapt to people and not people to technology.
- That the student has a vision focused on the user experience by applying appropriate conceptual and technological approaches.
- Understand how emerging technology makes possible new styles of interaction.
- Determine the basic requirements at the interface level, hardware and software for the construction of immersive environments.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Familiarity)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)

3) Communicate effectively in a variety of professional contexts.. (Usage)

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Familiarity)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Familiarity)

7. TOPICS

Unit 1: Fundamentos (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Contextos para IHC (cualquiera relacionado con una interfaz de usuario, p.e., página web, aplicaciones de negocios, aplicaciones móviles y juegos) • Heurística de usabilidad y los principios de pruebas de usabilidad. • Procesos para desarrollo centrado en usuarios, p.e., enfoque inicial en usuarios, pruebas empíricas, diseño iterativo. • Principios del buen diseño y buenos diseñadores; ventajas y desventajas de ingeniería. • Diferentes medidas para evaluación, p.e., utilidad, eficiencia, facilidad de aprendizaje, satisfacción de usuario.	• Discutir por qué el desarrollo de software centrado en el hombre es importante [Familiarizarse] • Define un proceso de diseño centralizado en el usario que de forma explícita considere el hecho que un usuario no es como un desarrollador o como sus conocimientos [Familiarizarse] • Resumir los preceptos básicos de la interacción psicológica y social [Familiarizarse] • Desarrollar y usar un vocabulario conceptual para analizar la interación humana con el software: disponibilidad, modelo conceptual, retroalimentación, y demás [Familiarizarse]
Readings : [Dix+04], [Sto+05], [RS11]	

Unit 2: Factores Humanos (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Modelos cognoscitivos que informan diseño de interacciones, p.e., atención, percepción y reconocimiento, movimiento, memoria, golfos de expectativa y ejecución. • Capacidades físicas que informan diseño de interacción, p.e. percepción del color, ergonomía. • Accesibilidad, p.e., interfaces para poblaciones con diferentes habilidades (p.e., invidentes, discapacitados) • Interfaces para grupos de población de diferentes edades (p.e., niños, mayores de 80)	• Crear y dirigir una simple pruebga de usabilidad para una aplicación existente de software [Familiarizarse]
Readings : [Dix+04], [Sto+05], [RS11], [Mat11], [Nor04]	

Unit 3: Diseño y Testing centrados en el usuario (16)

Competences Expected:

Topics	**Learning Outcomes**
• Enfoque y características del proceso de diseño. • Requerimientos de funcionalidad y usabilidad. • Técnicas de recolección de requerimientos, ej. entrevistas, encuentas, etnografía e investigación contextual. • Técnicas y herramientas para el análisis y presentación de requerimientos ej. reportes, personas. • Análisis de tareas, incluidos los aspectos cualitativos de la generación de modelos de análisis de tareas. • Consideración de IHC como una disciplina de diseño: – Sketching – Diseño participativo – Sketching – Diseño participativo • Técnicas de creación de prototipos y herramientas, ej.bosquejos, *storyboards*, prototipos de baja fidelidad, esquemas de página. • Prototipos de baja fidelidad (papel) • Técnicas de evaluación cuantitativa ej. evaluación Keystroke-level. • Evaluación sin usuarios, usando ambas técnicas cualitativas y cuantitativas. Ej. Revisión estructurada, GOMS, análisis basado en expertos, heurísticas, lineamientos y estándar. • Evaluación con usuarios. Ej. Observación, Método de pensamiento en voz alta, entrevistas, encuentas, experimentación. • Desafíos para la evaluación efectiva, por ejemplo, toma de muestras, la generalización. • Reportar los resultados de las evaluaciones. • Internacionalización, diseño para usuarios de otras culturas, intercultural.	• Llevar a cabo una evaluación cuantitativa y discutir / informar sobre los resultados [Familiarizarse] • Para un grupo de usuarios determinado, realizar y documentar un análisis de sus necesidades [Familiarizarse] • Discutir al menos un standard nacional o internacional de diseño de interfaz de usuario [Familiarizarse] • Explicar cómo el diseño centrado en el usuario complementa a otros modelos de proceso software [Familiarizarse] • Utilizar *lo-fi* (baja fidelidad) técnicas de prototipado para recopilar y reportar, las respuestas del usuario [Usar] • Elegir los métodos adecuados para apoyar el desarrollo de una específica interfaz de usuario [Evaluar] • Utilizar una variedad de técnicas para evaluar una interfaz de usuario dada [Evaluar] • Comparar las limitaciones y beneficios de los diferentes métodos de evaluación [Evaluar]
Readings : [Dix+04], [Sto+05], [RS11], [Mat11], [Bux07]	

Unit 4: Diseño de Interacción (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Principios de interfaces gráficas de usuario (GUIs) • Elementos de diseño visual (disposición, color, fuentes, etiquetado) • Manejo de fallas humanas/sistema. • Estándares de interfaz de usuario. • Presentación de información: navegación, representación, manipulación. • Técnicas de animación de interfaz (ej. grafo de escena) • Clases Widget y bibliotecas. • Internacionalización, diseño para usuarios de otras culturas, intercultural. • Elección de estilos de interacción y técnicas de interacción.	• Crear una aplicación simple, junto con la ayuda y la documentación, que soporta una interfaz gráfica de usuario [Usar]
Readings : [Dix+04], [Sto+05], [RS11], [Joh10], [Mat11], [LS06]	

Unit 5: Nuevas Tecnologías Interactivas (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• Elección de estilos de interacción y técnicas de interacción. • Enfoques para el diseño, implementación y evaluación de la interacción sin mouse – Interfaces táctiles y multitáctiles. – Interfaces compartidas, incorporadas y grandes – Nuevas modalidades de entrada (tales como datos de sensores y localización) – Nuevas ventanas, por ejemplo, iPhone, Android – Reconocimiento de voz y procesamiento del lenguaje natural – Interfaces utilizables y tangibles – Interacción persuasiva y emoción – Tecnologías de interacción ubicuas y contextuales (Ubicomp) – Inferencia bayesiana (por ejemplo, texto predictivo, orientación guiada) – Visualización e interacción de ambiente / periféricos • Salida: – Sonido – Visualización estereoscópica – Forzar la simulación de retroalimentación, dispositivos hápticos • Arquitectura de Sistemas: – Motores de Juego – Relidad Aumentada móvil – Simuladores de vuelo – CAVEs – Imágenes médicas	• Describe cuando son adecuadas las interfaces sin uso de ratón [Familiarizarse] • Comprende las posibilidades de interacción que van más allá de las interfaces de ratón y puntero [Familiarizarse] • Discute las ventajas (y desventajas) de las interfaces no basadas en ratón [Usar] • Describir el modelo óptico realizado por un sistema de gráficos por computadora para sintetizar una visión estereoscópica [Familiarizarse] • Describir los principios de las diferentes tecnologias de seguimiento de espectador [Familiarizarse] • Determinar los requerimientos básicos en interfaz, software, hardware, y cofiguraciones de software de un sistema VR para una aplicación específica [Evaluar]
Readings : [Dix+04], [Sto+05], [RS11], [WW11], [Mat11]	

Unit 6: Colaboración y Comunicación (8)	
Competences Expected:	
Topics	**Learning Outcomes**
• La comunicación asíncrona en grupo, por ejemplo, el correo electrónico, foros, redes sociales. • Medios de comunicación social, informática social, y el análisis de redes sociales. • Colaboración en línea, espacios "inteligentes" y aspectos de coordinación social de tecnologías de flujo de trabajo. • Comunidades en línea. • Personajes de Software y agentes inteligentes, mundos virtuales y avatares. • Psicología Social	• Describir la diferencia entre la comunicación sincrónica y asincrónica [Familiarizarse] • Comparar los problemas de IHC en la interacción individual con la interacción del grupo [Familiarizarse] • Discuta varias problemas de interés social planteados por el software colaborativo [Usar] • Discutir los problemas de IHC en software que personifica la intención humana [Evaluar]
Readings : [Dix+04], [Sto+05], [RS11]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Dix+04] Alan Dix et al. *Human-computer Interaction*. 3 ed. Prentice-Hall, Inc, 2004.

[Nor04] Donald A. Norman. *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Book, 2004.

[Sto+05] D. Stone et al. *User Interface Design and Evaluation*. Morgan Kaufmann Series in Interactive Technologies, 2005.

[LS06] M. Leavitt and B. Shneiderman. *Research-Based Web Design & Usability Guidelines*. Health and Human Services Dept, 2006.

[Bux07] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., 2007.

[Joh10] Jeff Johnson. *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. 3 ed. Morgan Kaufmann Publishers Inc., 2010.

[Mat11] Lukas Mathis. *Designed for Use: Create Usable Interfaces for Applications and the Web*. Pragmatic Bookshelf, 2011.

[RS11] Y. Rogers and J Sharp H. & Preece. *Interaction Design: Beyond Human-Computer Interaction*. 3 ed. John Wiley and Sons Ltd, 2011.

[WW11] D. Wigdor and D. Wixon. *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann Publishers Inc, 2011.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS391. Software Engineering III (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS391. Software Engineering III
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS292. Software Engineering II. (6^{th} Sem) CS292. Software Engineering II. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Software development requires the use of best development practices, IT project management, equipment management And efficient and rational use of quality assurance frameworks, these elements are key and transversal during the whole productive process. The construction of software contemplates the implementation and use of processes, methods, models and tools that allow to achieve the realization of the quality attributes of a product.

5. GOALS

- Understand and implement the fundamental concepts of project management and software equipment management.
- Understand the fundamentals of project management, including its definition, scope, and need for project management in the modern organization.
- Students have to understand the fundamental concepts of CMMI, PSP, TSP to be adopted in software projects.
- Describe and understand quality assurance models as a key framework for the success of IT projects.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)

3) Communicate effectively in a variety of professional contexts.. (Usage)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Assessment)

7. TOPICS

Unit 1: Evolución de Software (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Desarrollo de Software en el contexto de código grande pre existente – Cambios de software – Preocupaciones y ubicación de preocupaciones – *Refactoring* • Evolución de Software. • Características de Software mantenible. • Sistemas de Reingeniería. • Reuso de Software. – Segmentos de código – Bibliotecas y *frameworks* – Componentes – Líneas de Producto	• Identificar los problemas principales asociados con la evolución del software y explicar su impacto en el ciclo de vida del software [Familiarizarse] • Estimar el impacto del cambio de requerimientos en productos existentes de tamaño medio [Usar] • Usar refactorización en el proceso de modificación de un componente de sosftware [Usar] • Estudiar los desafios de mejorar sistemas en un entorno cambiante [Familiarizarse] • Perfilar los procesos de pruebas de regresión y su rol en el manejo de versiones [Familiarizarse] • Estudiar las ventajas y desventajas de diferentes tipos de niveles de confiabilidad [Familiarizarse]
Readings : [PM15], [Som17]	

Unit 2: Gestión de Proyectos de Software (10)

Competences Expected:

Topics	Learning Outcomes
• La participación del equipo: – Procesos elemento del equipo, incluyendo responsabilidades de tarea, la estructura de reuniones y horario de trabajo – Roles y responsabilidades en un equipo de software – Equipo de resolución de conflictos – Los riesgos asociados con los equipos virtuales (comunicación, la percepción, la estructura) • Estimación de esfuerzo (a nivel personal) • Riesgo. – El papel del riesgo en el ciclo de vida – Categorías elemento de riesgo, incluyendo la seguridad, la seguridad, mercado, finanzas, tecnología, las personas, la calidad, la estructura y el proceso de • Gestión de equipos: – Organización de equipo y la toma de decisiones – Roles de identificación y asignación – Individual y el desempeño del equipo de evaluación • Gestión de proyectos: – Programación y seguimiento de elementos – Herramientas de gestión de proyectos – Análisis de Costo/Beneficio	• Discutir los comportamientos comunes que contribuyen al buen funcionamiento de un equipo [Familiarizarse] • Crear y seguir un programa para una reunión del equipo [Usar] • Identificar y justificar las funciones necesarias en un equipo de desarrollo de software [Usar] • Entender las fuentes, obstáculos y beneficios potenciales de un conflicto de equipo [Usar] • Aplicar una estrategia de resolución de conflictos en un ambiente de equipo [Usar] • Utilizar un método ad hoc para estimar el esfuerzo de desarrollo del software (ejemplo, tiempo) y comparar con el esfuerzo actual requerido [Usar] • Listar varios ejemplos de los riesgos del software [Familiarizarse] • Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Familiarizarse] • Describir las diferentes categorías de riesgo en los sistemas de software [Familiarizarse] • Demostrar a través de la colaboración de proyectos de equipo los elementos centrales de la contrucción de equipos y gestión de equipos [Usar] • Describir como la elección de modelos de procesos afectan la estructura organizacional de equipos y procesos de toma de decisiones [Familiarizarse] • Crear un equipo mediante la identificación de los roles apropiados y la asignación de funciones a los miembros del equipo [Usar] • Evaluar y retroalimentar a los equipos e individuos sobre su desempeño en un ambiente de equipo [Usar] • Usando un software particular procesar, describir los aspectos de un proyecto que encesita ser planeado y monitoreado, (ejemplo, estimar el tamaño y esfuerzo, un horario, reasignación de recursos, control de configuración, gestión de cambios, identificación de riesgos en un proyecto y gestión) [Familiarizarse]

Readings : [PM15], [Som17]

Unit 3: Gestión de Proyectos de Software (8)	
Competences Expected:	
Topics	**Learning Outcomes**

• Software de medición y técnicas de estimación.
• Aseguramiento de la calidad del software y el rol de las mediciones.
• Riesgo.
  – Identificación de riesgos y gestión.
  – Análisis riesgo y evaluación.
  – La tolerancia al riesgo (por ejemplo, riesgo adverso, riesgo neutral, la búsqueda de riesgo)
  – Planificación de Riesgo
• En todo el sistema de aproximación al riesgo, incluyendo riesgos asociados con herramientas.
	• Realizar el seguimiento del progreso de alguna etapa de un proyecto que utiliza métricas de proyectos apropiados [Usar]
	• Comparar las técnicas simples de tamaño de software y estimación de costos [Usar]
	• Usar una herramienta de gestión de proyectos para ayudar en la asignación y rastreo de tareas en un proyecto de desarrollo de software [Usar]
	• Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Evaluar]
	• Identificar riesgos y describir enfoques para manejar riesgos (evitar, aceptar, tranferir, mitigar) y caracterizar fortalezas y defectos para cada uno [Familiarizarse]
	• Explicar cómo el riesgo afecta las decisiones en el proceso de desarrollo de software [Usar]
	• Identificar los riesgos de seguridad para un sistema de software [Usar]
	• Demostrar un enfoque sistemático para la tarea de identificar los peligros y riesgos en una situación particular [Usar]
	• Aplicar los principios básicos del manejo de riesgos en una variedad de escenarios simples incluyendo una situación de seguridad [Usar]
	• Dirigir un análisis de costo/beneficio para el enfoque de mitigación de riesgos [Usar]
	• Identificar y analizar alguno de los riesgos para un sistema entero que surgen de aspectos distintos del software [Usar]

Readings : [PM15], [Som17]	

Unit 4: Procesos de Software (12)

Competences Expected:

Topics	Learning Outcomes
• Consideraciones a nivel de sistemas, ejem., la interacción del software con su entorno. • Introducción a modelos del proceso de software (e.g., cascada, incremental, agil): – Actividades con ciclos de vida de software. • Programación a gran escala versus programación individual. • Evaluación de modelos de proceso de software. • Conceptos de calidad de software. • Mejoramiento de procesos. • Modelos de madurez de procesos de software. • Mediciones del proceso de software.	• Describir cómo la programación en grandes equipos difiere de esfuerzos individuales con respecto a la comprensión de una gran base de código, lectura de código, comprensión de las construcciones, y comprensión de contexto de cambios [Usar] • Describir las ventajas y desventajas relativas entre varios modelos importantes de procesos (por ejemplo, la cascada, iterativo y ágil) [Usar] • Describir las ventajas y desventajas relativas entre varios modelos importantes de procesos (por ejemplo, la cascada, iterativo y ágil) [Usar] • Diferenciar entre las fases de desarrollo de software [Usar] • Describir cómo la programación en grandes equipos difiere de esfuerzos individuales con respecto a la comprensión de una gran base de código, lectura de código, comprensión de las construcciones, y comprensión de contexto de cambios [Usar] • Explicar el papel de los modelos de madurez de procesos en la mejora de procesos [Usar] • Comparar varios modelos comunes de procesos con respecto a su valor para el desarrollo de las clases particulares de sistemas de software, teniendo en cuenta diferentes aspectos tales como, estabilidad de los requisitos, tamaño y características no funcionales [Usar] • Definir la calidad del software y describir el papel de las actividades de aseguramiento de la calidad en el proceso de software [Usar] • Describir el objetivo y similitudes fundamentales entre los enfoques de mejora de procesos [Usar] • Comparar varios modelos comunes de procesos con respecto a su valor para el desarrollo de las clases particulares de sistemas de software, teniendo en cuenta diferentes aspectos tales como, estabilidad de los requisitos, tamaño y características no funcionales [Usar] • Evaluar un esfuerzo de desarrollo y recomendar cambios potenciales al participar en la mejora de procesos (usando un modelo como PSP) o involucración en una retrospectiva de un proyecto [Usar] • Explicar el papel de los modelos de madurez de procesos en la mejora de procesos [Usar] • Describir varias métricas de procesos para la evaluación y el control de un proyecto [Usar] • Usar las medidas en proyecto para describir el estado actual de un proyecto [Usar]

Readings : [PM15], [Som17]

Unit 5: Estándares ISO/IEC (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• ISO 9001:2001. • ISO 9000-3. • ISO/IEC 9126. • ISO/IEC 12207. • ISO/IEC 15939. • ISO/IEC 14598. • ISO/IEC 15504-SPICE. • IT Mark. • SCRUM. • SQuaRE. • CISQ.	• Learn and apply correctly standards and international standards . [Usar]
Readings : [Som17], [PM15]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[PM15] Roger S. Pressman and Bruce Maxim. *Software Engineering: A Practitioner's Approach*. 8th. McGraw-Hill, Jan. 2015.

[Som17] Ian Sommerville. *Software Engineering*. 10th. Pearson, Mar. 2017.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS3I1. Computer Security (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3I1. Computer Security
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS231. Networking and Communication. (6^{th} Sem) CS231. Networking and Communication. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Nowadays, information is one of the most valuable assets in any organization. This course is oriented to be able to provide the student with the security elements oriented to protect the Information of the organization and mainly to be able to foresee the possible problems related to this heading. This subject involves the development of a preventive attitude on the part of the student in all areas related to software development.

5. GOALS

- Discuss at an intermediate intermediate level the fundamentals of Computer Security.
- Provide different aspects of the malicious code.
- That the student knows the concepts of cryptography and security in computer networks.
- Discuss and analyze together with the student the aspects of Internet Security.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Assessment)

7. TOPICS

Unit 1: Fundamentos y Conceptos en Seguridad (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• CIA (Confidencialidad, Integridad, Disponibilidad) • Conceptos de riesgo, amenazas, vulnerabilidades, y los tipos de ataque . • Autenticación y autorización, control de acceso (vs. obligatoria discrecional) • Concepto de la confianza y la honradez . • Ética (revelación responsable)	• Analizar las ventajas y desventajas de equilibrar las propiedades clave de seguridad(Confidenciabilidad, Integridad, Disponibilidad) [Familiarizarse] • Describir los conceptos de riesgo, amenazas, vulnerabilidades y vectores de ataque(incluyendo el hecho de que no existe tal cosa como la seguridad perfecta) [Familiarizarse] • Explicar los conceptos de autentificación, autorización, control de acceso [Familiarizarse] • Explicar el concepto de confianza y confiabilidad [Familiarizarse] • Reconocer de que hay problemas éticos más importantes que considerar en seguridad computacional, incluyendo problemas éticos asociados a arreglar o no arreglar vulnerabilidades y revelar o no revelar vulnerabilidades [Familiarizarse]
Readings : [WL14]	

Unit 2: Principios de Diseño Seguro (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• Menor privilegio y aislamiento. • Valores predeterminados a prueba de fallos. • Diseño abierto. • La seguridad de extremo a extremo. • La defensa en profundidad (por ejemplo, la programación defensiva, defensa en capas) • Diseño de seguridad. • Las tensiones entre la seguridad y otros objetivos de diseño. • Mediación completa. • El uso de componentes de seguridad vetados. • Economía del mecanismo (la reducción de la base informática de confianza, minimizar la superficie de ataque) • Seguridad utilizable. • Componibilidad de seguridad. • Prevención, detección y disuasión.	• Describir el principio de privilegios mínimos y el aislamiento que se aplican al diseño del sistema [Familiarizarse] • Resumir el principio de prueba de fallos y negar por defecto [Familiarizarse] • Discutir las implicaciones de depender de diseño abierto o secreto de diseño para la seguridad [Familiarizarse] • Explicar los objetivos de seguridad de datos de extremo a extremo [Familiarizarse] • Discutir los beneficios de tener múltiples capas de defensas [Familiarizarse] • Por cada etapa en el ciclo de vida de un producto, describir que consideraciones de seguridad deberian ser evaluadas [Familiarizarse] • Describir el costo y ventajas y desventajas asociadas con el diseño de seguridad de un producto. [Familiarizarse] • Describir el concepto de mediación y el principio de mediación completa [Familiarizarse] • Conocer los componentes estándar para las operaciones de seguridad, en lugar de reinventar las operaciones fundamentales [Familiarizarse] • Explicar el concepto de computación confiable incluyendo base informática confiable y de la superficie de ataque y el principio de minimización de base informática confiable [Familiarizarse] • Discutir la importancia de la usabilidad en el diseño de mecanismos de seguridad [Familiarizarse] • Describir problemas de seguridad que surgen en los límites entre varios componentes [Familiarizarse] • Identificar los diferentes roles de mecanismos de prevención y mecanismos de eliminación/disuación [Familiarizarse]
Readings : [WL14]	

Unit 3: Programación Defensiva (25)	
Competences Expected:	
Topics	**Learning Outcomes**

• Validación de datos de entrada y sanitización
• Elección del lenguaje de programación y lenguajes con tipos de datos seguro.
• Ejemplos de validación de entrada de datos y sanitización de errores.
  – Desbordamiento de búfer
  – Errores enteros
  – Inyección SQL
  – Vulnerabilidad XSS
• Las condiciones de carrera.
• Manejo correcto de las excepciones y comportamientos inesperados.
• Uso correcto de los componentes de terceros.
• Desplegar eficazmente las actualizaciones de seguridad.
• Información de control de flujo.
• Generando correctamente el azar con fines de seguridad.
• Mecanismos para la detección y mitigación de datos de entrada y errores de sanitización.
• Fuzzing
• El análisis estático y análisis dinámico.
• Programa de verificación.
• Soporte del sistema operativo (por ejemplo, la asignación al azar del espacio de direcciones, canarios)
• El soporte de hardware (por ejemplo, el DEP, TPM)
	• Explicar por que la validación de entrada y desinfección de datos es necesario en el frente del control contencioso del canal de entrada [Usar]
	• Explicar por que uno deberia escoger para desallorrar un programa en un lenguaje tipo seguro como Java, en contraste con un lenguaje de programación no seguro como C/C++ [Usar]
	• Clasificar los errores de validación de entrada común, y escribir correctamente el código de validación de entrada [Usar]
	• Demostrar el uso de un lenguaje de programación de alto nivel cómo prevenir una condición de competencia que ocurran y cómo manejar una excepción [Usar]
	• Demostrar la identificación y el manejo elegante de las condiciones de error [Familiarizarse]
	• Explique los riesgos de mal uso de las interfaces con código de terceros y cómo utilizar correctamente el código de terceros [Familiarizarse]
	• Discutir la necesidad de actualizar el software para corregir las vulnerabilidades de seguridad y la gestión del ciclo de vida de la corrección [Familiarizarse]

Readings : [WL14]	

Unit 4: Ataques y Amenazas (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• Atacante metas, capacidades y motivaciones (como economía sumergida, el espionaje digital, la guerra cibernética, las amenazas internas, hacktivismo, las amenazas persistentes avanzadas) • Los ejemplos de malware (por ejemplo, virus, gusanos, spyware, botnets, troyanos o rootkits) • Denegación de Servicio (DoS) y Denegación de Servicio Distribuida (DDoS) • Ingeniería social (por ejemplo, perscando) • Los ataques a la privacidad y el anonimato . • El malware / comunicaciones no deseadas, tales como canales encubiertos y esteganografía.	• Describir tipos de ataques similares en contra de un sistema en particular [Familiarizarse] • Discutir los limitantes de las medidas en contra del malware (ejm. detección basada en firmas, detección de comportamiento) [Familiarizarse] • Identificar las instancias de los ataques de ingeniería social y de los ataques de negación de servicios [Familiarizarse] • Discutir como los ataques de negación de servicos puede ser identificados y reducido [Familiarizarse] • Describir los riesgos de la privacidad y del anonimato en aplicaciones comunmente usadas [Familiarizarse] • Discutir los conceptos de conversión de canales y otros procedimientos de filtrado de datos [Familiarizarse]
Readings : [WL14]	

Unit 5: Seguridad de Red (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• Red de amenazas y tipos de ataques específicos (por ejemplo, la denegación de servicio, spoofing, olfateando y la redirección del tráfico, el hombre en el medio, ataques integridad de los mensajes, los ataques de enrutamiento, y el análisis de tráfico) • El uso de cifrado de datos y seguridad de la red . • Arquitecturas para redes seguras (por ejemplo, los canales seguros, los protocolos de enrutamiento seguro, DNS seguro, VPN, protocolos de comunicación anónimos, aislamiento) • Los mecanismos de defensa y contramedidas (por ejemplo, monitoreo de red, detección de intrusos, firewalls, suplantación de identidad y protección DoS, honeypots, seguimientos) • Seguridad para redes inalámbricas, celulares . • Otras redes no cableadas (por ejemplo, ad hoc, sensor, y redes vehiculares) • Resistencia a la censura. • Gestión de la seguridad operativa de la red (por ejemplo, control de acceso a la red configure)	• Describir las diferentes categorías de amenazas y ataques en redes [Familiarizarse] • Describir las arquitecturas de criptografía de clave pública y privada y cómo las ICP brindan apoyo a la seguridad en redes [Familiarizarse] • Describir ventajas y limitaciones de las tecnologías de seguridad en cada capa de una torre de red [Familiarizarse] • Identificar los adecuados mecanismos de defensa y sus limitaciones dada una amenaza de red [Usar]
Readings : [WL14]	

Unit 6: Criptografía (25)

Competences Expected:

Topics	Learning Outcomes
• Terminología básica de criptografía cubriendo las nociones relacionadas con los diferentes socios (comunicación), canal seguro / inseguro, los atacantes y sus capacidades, cifrado, descifrado, llaves y sus características, firmas. • Tipos de cifrado (por ejemplo, cifrado César, cifrado affine), junto con los métodos de ataque típicas como el análisis de frecuencia. • Apoyo a la infraestructura de clave pública para la firma digital y el cifrado y sus desafíos. • Criptografía de clave simétrica: – El secreto perfecto y el cojín de una sola vez – Modos de funcionamiento para la seguridad semántica y encriptación autenticada (por ejemplo, cifrar-entonces-MAC, OCB, GCM) – Integridad de los mensajes (por ejemplo, CMAC, HMAC) • La criptografía de clave pública: – Permutación de trampilla, por ejemplo, RSA – Cifrado de clave pública, por ejemplo, el cifrado RSA, cifrado El Gamal – Las firmas digitales – Infraestructura de clave pública (PKI) y certificados – Supuestos de dureza, por ejemplo, Diffie-Hellman, factoring entero • Protocolos de intercambio de claves autenticadas, por ejemplo, TLS . • Primitivas criptográficas: – generadores pseudo-aleatorios y cifrados de flujo – cifrados de bloque (permutaciones pseudo-aleatorios), por ejemplo, AES – funciones de pseudo-aleatorios – funciones de hash, por ejemplo, SHA2, resistencia colisión – códigos de autenticación de mensaje – funciones derivaciones clave	• Describir el propósito de la Criptografía y listar formas en las cuales es usada en comunicación de datos [Familiarizarse] • Definir los siguientes términos: Cifrado, Criptoanálisis, Algorítmo Criptográfico, y Criptología y describe dos métodos básicos (cifrados) para transformar texto plano en un texto cifrado [Familiarizarse] • Discutir la importancia de los números primos en criptografía y explicar su uso en algoritmos criptográficos [Familiarizarse] • Ilustrar como medir la entropía y como generar aleatoriedad criptográfica [Usar] • Usa primitivas de clave pública y sus aplicaciones [Usar] • Explicar como los protocolos de intercambio de claves trabajan y como es que pueden fallar [Familiarizarse] • Discutir protocolos criptográficos y sus propiedades [Familiarizarse]

Readings : [WL14]

Unit 7: Seguridad en la Web (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• Modelo de seguridad Web – Modelo de seguridad del navegador incluida la política de mismo origen – Los límites de confianza de cliente-servidor, por ejemplo, no pueden depender de la ejecución segura en el cliente • Gestión de sesiones, la autenticación: – Single Sign-On – HTTPS y certificados • Vulnerabilidades de las aplicaciones y defensas : – Inyección SQL – XSS – CSRF • Seguridad del lado del cliente : – Política de seguridad Cookies – Extensiones de seguridad HTTP, por ejemplo HSTS – Plugins, extensiones y aplicaciones web – Seguimiento de los usuarios Web • Herramientas de seguridad del lado del servidor, por ejemplo, los cortafuegos de aplicación Web (WAFS) y fuzzers	• Describe el modelo de seguridad de los navegadores incluyendo las políticas del mismo origen y modelos de amenazas en seguridad web [Familiarizarse] • Discutir los conceptos de sesiones web, canales de comunicación seguros tales como Seguridad en la Capa de Transporte(TLS) y la importancia de certificados de seguridad, autenticación incluyendo inicio de sesión único, como OAuth y Lenguaje de Marcado para Confirmaciones de Seguridad($SAML$) [Familiarizarse] • Investigar los tipos comunes de vulnerabilidades y ataques en las aplicaciones web, y defensas contra ellos [Familiarizarse] • Utilice las funciones de seguridad del lado del cliente [Usar]
Readings : [WL14]	

Unit 8: Seguridad de plataformas (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• Integridad de código y firma de código. • Arranque seguro, arranque medido, y la raíz de confianza. • Testimonio. • TPM y coprocesadores seguros. • Las amenazas de seguridad de los periféricos, por ejemplo, DMA, IOMMU. • Ataques físicos: troyanos de hardware, sondas de memoria, ataques de arranque en frío. • Seguridad de dispositivos integrados, por ejemplo, dispositivos médicos, automóviles. • Ruta confiable.	• Explica el concepto de integridad de código y firma de códigos, así como el alcance al cual se aplica [Familiarizarse] • Discute los conceptos del origen de la confidencialidad y el de los procesos de arranque y carga segura [Familiarizarse] • Describe los mecanismos de arresto remoto de la integridad de un sistema [Familiarizarse] • Resume las metas y las primitivas claves de los modelos de plataforma confiable (TPM) [Familiarizarse] • Identifica las amenazas de conectar periféricos en un dispositivo [Familiarizarse] • Identifica ataques físicos y sus medidas de control [Familiarizarse] • Identifica ataques en plataformas con hardware que no son del tipo PC [Familiarizarse] • Discute los conceptos y la importancia de ruta confiable [Familiarizarse]
Readings : [WL14]	

Unit 9: Investigación digital (Digital Forensics) (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• Principios básicos y metodologías de análisis digital forensico. • Diseñar sistemas con necesidades forenses en mente. • Reglas de Evidencia - conceptos generales y las diferencias entre las jurisdicciones y la Cadena de Custodia. • Búsqueda y captura de comprobación: requisitos legales y de procedimiento. • Métodos y normas de evidencia digital. • Las técnicas y los estándares para la conservación de los datos. • Cuestiones legales y reportes incluyendo el trabajo como perito. • Investigación digital de los sistema de archivos. • Los forenses de aplicación. • Investigación digital en la web. • Investigación digital en redes. • Investigación digital en dispositivos móviles. • Ataques al computador/red/sistema. • Detección e investigación de ataque. • Contra investigación digital.	• Describe qué es una investigación digital, las fuentes de evidencia digital, y los límites de técnicas forenses [Familiarizarse] • Explica como diseñar software de apoyo a técnicas forenses [Familiarizarse] • Describe los requisitos legales para usar datos recuperados [Familiarizarse] • Describe qué es una investigación digital, las fuentes de evidencia digital, y los límites de técnicas forenses [Familiarizarse] • Describe como se realiza la recolección de datos y el adecuado almacenamiento de los datos originales y de la copia forense [Familiarizarse] • Realiza recolección de datos en un disco duro [Usar] • Describe la responsabilidad y obligación de una persona mientras testifica como un examinador forense [Familiarizarse] • Recupera datos basados en un determinado término de búsqueda en una imagen del sistema [Usar] • Reconstruye el historial de una aplicación a partir de los artefactos de la aplicación [Familiarizarse] • Reconstruye el historial de navegación web de los artefactos web [Familiarizarse] • Captura e interpreta el tráfico de red [Familiarizarse] • Discute los retos asociados con técnicas forenses de dispositivos móviles [Familiarizarse]
Readings : [WL14]	

Unit 10: Seguridad en Ingeniería de Software (25)	
Competences Expected:	
Topics	**Learning Outcomes**
• La construcción de la seguridad en el ciclo de vida de desarrollo de software. • Principios y patrones de diseño seguros. • Especificaciones de software seguros y requisitos. • Prácticas de desarrollo de software de seguros. • Asegure probar el proceso de las pruebas de que se cumplan los requisitos de seguridad (incluyendo análisis estático y dinámico)	• Describir los requisitos para la integración de la seguridad en el SDL [Familiarizarse] • Aplicar los conceptos de los principios de diseño para mecanismos de protección, los principios para seguridad de software (Viega and McGraw) y los principios de diseño de seguridad (Morrie Gasser) en un proyecto de desarrollo de software [Familiarizarse] • Desarrollar especificaciones para un esfuerzo de desarrollo de software que especifica completamente los requisitos funcionales y se identifican las rutas de ejecución esperadas [Familiarizarse]
Readings : [WL14]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[WL14] Stallings. W and Brown. L. *Computer Security: Principles and Practice*. Pearson Education, Limited, 2014.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS251. Computer graphics (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS251. Computer graphics
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- CS312. Advanced Data Structures . (6^{th} Sem)
- MA307. Mathematics applied to computing . (6^{th} Sem)
- CS312. Advanced Data Structures . (6^{th} Sem)
- MA307. Mathematics applied to computing . (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

It offers an introduction to the area of Computer Graphics, which is an important part of Computer Science. The purpose of this course is to investigate the fundamental principles, techniques and tools for this area.

5. GOALS

- Bring students to concepts and techniques used in complex 3-D graphics applications.
- Give the student the necessary tools to determine which graphics software and which platform are best suited to develop a specific application.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Conceptos Fundamentales (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Aplicaciones multimedia, incluyendo interfaces de usuario, edición de audio y vídeo, motores de juego, cad, visualización, realidad virtual. • Soluciones de compensación entre el almacenamiento de datos y los datos re-computing es personalizado por vectores y raster en representaciones de imágenes. • Modelos de color sustractivo Aditivo y (CMYK y RGB) y por qué estos proporcionan una gama de colores. • Animación como una secuencia de imágenes fijas.	• Explicar en términos generales cómo las señales analógicas pueden ser representadas por muestras discretas, por ejemplo,cómo las imagenes pueden ser representadas por pixeles [Familiarizarse] • Describir modelos de color y su uso en los dispositivos de visualización de gráficos [Familiarizarse] • Describir las ventajas y desventajas entre el almacenamiento de información vs almacenar suficiente información para reproducir la información, como en la diferencia entre el vector y la representación de la trama [Familiarizarse] • Describir los procesos básico de la producción de movimiento continuo a partir de una secuencia de cuadros discretos(algunas veces llamado it flicker fusion) [Familiarizarse]
Readings : [HB90]	

Unit 2: Rendering Básico (12)

Competences Expected:

Topics	**Learning Outcomes**
• Renderizado en la naturaleza, por ejemplo, la emisión y dispersión de la luz y su relación con la integración numérica. • Renderizado Fordward and Backward (i.e., *ray-casting* y rasterización) • Radiometría básica, triángulos similares y modelos de proyecciones • Afinamiento y Transformaciones de Sistemas de coordenadas • *Ray tracing* • Visibilidad y oclusión, incluyendo soluciones a este problema, como el almacenamiento en búfer de profundidad, algoritmo del pintor, y el trazado de rayos. • Rasterización triangular simple. • Renderización con una API basada en shader. • Aplicación de la representación de estructuras de datos espaciales. • Muestreo y anti-aliasing. • Renderizado Fordward and Backward (i.e., *ray-casting* y rasterización)	• Discutir el problema de transporte de la luz y su relación con la integración numérica, es decir, se emite luz, dispersa alrededor de la escena, y es medida por el ojo [Familiarizarse] • Describir la tubería básica gráficos y cómo el factor de representación va hacia adelante y atrás en esta [Familiarizarse] • Crear un programa para visualizar modelos 3D de imagenes gráficas simples [Usar] • Obtener puntos en 2-dimensiones y 3-dimensiones por aplicación de transformaciones afín [Usar] • Aplicar sistema de coordenadas de 3-dimensiones y los cambios necesarios para extender las operaciones de transformación 2D para manejar las transformaciones en 3D [Usar] • Contrastar la renderización hacia adelanate *forward* y hacia atras *backward* [Evaluar] • Explicar el concepto y las aplicaciones de mapeo de texturas, muestreo y el *anti-aliasing* [Familiarizarse] • Explicar la dualidad de rastreo de rayos/rasterización para el problema de visibilidad [Familiarizarse] • Implementar un sencillo renderizador en tiempo real utilizando una API de rasterización (por ejemplo, OpenGL) utilizando buffers de vértices y *shaders* [Usar] • Calcular las necesidades de espacio en base a la resolución y codificación de color [Evaluar] • Calcular los requisitos de tiempo sobre la base de las frecuencias de actualización, técnicas de rasterización [Evaluar]

Readings : [HB90], [Hug+13], [Wol11], [Shr+13]

Unit 3: Programación de Sistemas Interactivos (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Manejo de eventos e interacción de usuario. • Enfoques para el diseño, implementación y evaluación de la interacción sin mouse – Interfaces táctiles y multitáctiles. – Interfaces compartidas, incorporadas y grandes – Nuevas modalidades de entrada (tales como datos de sensores y localización) – Nuevas ventanas, por ejemplo, iPhone, Android – Reconocimiento de voz y procesamiento del lenguaje natural – Interfaces utilizables y tangibles – Interacción persuasiva y emoción – Tecnologías de interacción ubicuas y contextuales (Ubicomp) – Inferencia bayesiana (por ejemplo, texto predictivo, orientación guiada) – Visualización e interacción de ambiente / periféricos	• Discute las ventajas (y desventajas) de las interfaces no basadas en ratón [Evaluar]
Readings : [HB90]	

Unit 4: Modelado Geométrico (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Operaciones geométricas básicas como cálculo de intersección y pruebas de proximidad. • Volúmenes, voxels y representaciones basadas en puntos. • Curvas polinomiales y Superficies paramétricas. • Representación ímplicita de curvas y superficies. • Técnicas de aproximación, tales como curvas polinómicas, curvas Bezier, curvas spline y superficies, y base racional no uniforme (NURB) espinas, y el método de ajuste de nivel. • Técnicas de superficie de representación incluyendo teselación, la representación de malla, carenado malla, y las técnicas de generación de mallas, como la triangulación de Delaunay, marchando cubos. • Técnicas de subdivisión espacial. • Modelos procedimentales como fractales, modelamiento generativo y sistemas L. • Modelos deformables de forma libre y elásticamente deformables. • Subdivisión de superficies. • Modelado multiresolución. • Reconstrucción. • Representación de Geometría Sólida Constructiva (GSC)	• Representar curvas y superficies utilizando formas tanto implícitas y paramétricas [Usar] • Crear modelos poliédrico simples por teselación de superficies [Usar] • Generar un modelo fractal o terreno usando un método de procedimiento [Usar] • Generar una malla de un conjunto de puntos adquiridos por un scaner laser [Usar] • Construct modelos de geometría sólida constructiva a partir de simples primitivas, tales como cubos y superficies cuádricas [Usar] • Contrastar métodos de modelización con respecto a espacio y tiempo de complejidad y calidad de imagen [Evaluar]
Readings : [HB90], [Shr+13]	

Unit 5: Renderizado Avanzado (6)	
Competences Expected:	
Topics	**Learning Outcomes**
• Tiempo (desenfoque de movimiento), la posición del objetivo (enfoque), y la frecuencia continua (color) y su impacto en la representación. • Mapeo de Sombras. • Selectiva de oclusión. • Disperción de la Superficie. • Renderizado no fotorealistico. • Arquitectura del GPU. • Sistemas visuales humanos incluida la adaptación a la luz, la sensibilidad al ruido, y la fusión de parpadeo.	• Demostrar como un algoritmo calcula una solución a la ecuación de renderización [Evaluar] • Demostrar las propiedades de un algoritmo de renderización, por ejemplo, completo, consistente, e imparcial [Evaluar] • Implementar un algoritmo no trivial de sombreado(por ejemplo, sombreado caricaturizado(*toon shading*), mapas de sombras en cascada(*cascaded shadow maps*)) bajo una APi de rasterización [Usar] • Discutir como una técnica artística particular puede ser implementada en un renderizador [Familiarizarse] • Explicar como reconocer las técnicas gráficas usadas para crear una imagen en particular [Familiarizarse]
Readings : [HB90], [Hug+13], [Wol11], [Shr+13]	

Unit 6: Animación por computadora (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Cinématica directa e inversa. • Detección de colisiones y respuesta. • Animación procedimental empleando ruido, reglas (boids/crowds) y sistemas de partículas. • Algoritmos Skinning. • Movimientos basado en la física, incluyendo la dinámica del cuerpo rígido, sistemas de partículas físicas, redes de masa-muelle de tela y la carne y el pelo. • Animación de Cuadros Principales • Splines • Estructuras de datos para rotaciones, como cuaterniones. • Animación de Cámara. • Captura de Movimiento.	• Calcular la orientación de partes articuladas de un modelo de una localizacion y orientación usando un enfoque de cinemática inversa [Usar] • Implementar el método de interpolación *spline* para producir las posiciones y orientaciones en medio [Usar] • Implementar algoritmos para el modelamiento físico de partículas dinámicas usando simplemente la mecánica de Newton, por ejemplo Witkin & Kass , serpientes y gusanos, Euler simpléctica, Stormer/Verlet, o métodos de punto medio de Euler [Usar] • Discutir las ideas básicas detrás de algunos métodos para dinámica de fluidos para el modelamiento de trayectorias balísticas, por ejemplo salpicaduras, polvo, fuego, o humo [Familiarizarse] • Usar el software de animación común para construir formas orgánicas simples usando *metaball* y el esqueleto [Usar]
Readings : [HB90], [Shr+13]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[HB90] Donald Hearn and Pauline Baker. *Computer Graphics in C*. Prentice Hall, 1990.

[Wol11] David Wolff. *OpenGL 4.0 Shading Language Cookbook*. Packt Publishing, 2011.

[Hug+13] John F. Hughes et al. *Computer Graphics - Principles and Practice 3rd Edition*. Addison-Wesley, 2013.

[Shr+13] Dave Shreiner et al. *OpenGL, Programming Guide, Eighth Edition*. Addison-Wesley, 2013.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS262. Machine learning (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS262. Machine learning
2.2 Semester	:	7^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS261. Artificial Intelligence. (6^{th} Sem) CS261. Artificial Intelligence. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Write justification for this course here ...

5. GOALS

- Write your first goal here.
- Write your second goal here.
- Just in case you need more goals write them here

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Familiarity)

7. TOPICS

Unit 1: title for the unit goes here (5)	
Competences Expected:	
Topics	**Learning Outcomes**
• Topic1 • Topic2 • Topic3	• Learning outcome1 [Levelforthislearningoutcome]. • Apply computing in complex problems [Usar]. • Create a search engine [Evaluar]. • Study data structures [Familiarizarse].
Readings : [**Bibitem1**], [**Bibitem2**]	

Unit 2: another unit goes here (1)	
Competences Expected:	
Topics	**Learning Outcomes**
• Topic1	• Learning outcome xyz [Levelforthislearningoutcome].
Readings : [**Bibitem3**], [**Bibitem1**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS281. Computing in Society (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS281. Computing in Society
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	2
2.4 Horas	:	2 HT;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Ofrece una visión amplia de los aspectos éticos y profesionales relacionados con la computación. Los tópicos que se incluyen abarcan los aspectos éticos, sociales y políticos. Las dimensiones morales de la computación. Los métodos y herramientas de análisis. Administración de los recursos computacionales. Seguridad y control de los sistemas computacionales. Responsabilidades profesionales y éticas. Propiedad intelectual.

5. GOALS

- Hacer que el alumno entienda la importancia del cuidado y la ética en la transferencia y uso de la información.
- Inculcar en el alumno que las tendencias de mejoramiento de la tecnología, no debe ser llevada a degradar la moral de la sociedad.

6. COMPETENCES

3) Communicate effectively in a variety of professional contexts.. (Familiarity)

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Usage)

7. TOPICS

Unit 1: Historia (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Pre-historia – El mundo antes de 1946. • Historia del hardware, software, redes. • Pioneros de la Computación. • Historia de Internet.	• Identificar importantes tendencias en la historia del campo de la computación [Familiarizarse] • Identificar las contribuciones de varios pioneros en el campo de la computación [Familiarizarse] • Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse] • Comparar la vida diaria antes y después de la llegada de los ordenadores personales y el Internet [Familiarizarse]
Readings : [LL04], [McL00]	

Unit 2: Contexto Social (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Implicancias sociales de la computación en un mundo conectado en red. • Impacto de los medios sociales en el individualismo, colectivismo y en la cultura. • Crecimiento y control de la Internet • A menudo se refiere como la brecha digital, las diferencias en el acceso a los recursos de la tecnología digital y sus ramificaciones resultantes para el género, la clase, la etnia, la geografía, y/o los países subdesarrollados. • Los problemas de accesibilidad, incluyendo los requisitos legales. • Computación consciente del contexto.	• Describir las formas positivas y negativas en las que la tecnología computacional (redes, computación móvil, *cloud computing*) altera los modos de interacción social en el plano personal [Familiarizarse] • Identificar los supuestos y valores incorporados en el hardware y el software de diseño de los desarrolladores, especialmente lo que se refiere a la facilidad de uso para diversas poblaciones incluyendo minorías poblaciones y los discapacitados [Usar] • Interpretar el contexto social de un determinado diseño y su aplicación [Evaluar] • Evaluar la eficacia de un diseño y aplicación dada a partir de datos empíricos [Familiarizarse] • Resumir las implicaciones de los medios sociales en el individualismo frente al colectivismo y la cultura [Familiarizarse] • Discuta cómo el acceso a Internet sirve como una fuerza liberadora para las personas que viven bajo las formas opresivas de gobierno; explicar la utilización los límites al acceso a Internet como herramientas de represión política y social [Familiarizarse] • Analizar los pros y los contras de la dependencia de la computación en la implementación de la democracia (por ejemplo, prestación de servicios sociales, votación electrónica) [Familiarizarse] • Describir el impacto de la escasa representación de las diversas poblaciones en la profesión (por ejemplo, la cultura de la industria, la diversidad de productos) [Usar] • Explicar las consecuencias de la sensibilidad al contexto en los sistemas de computación ubicua [Familiarizarse]
Readings : [LL04], [McL00]	

Unit 3: Herramientas de Análisis (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Argumentación ética. • Teorías éticas y toma de decisiones. • Suposiciones morales y valores.	• Evaluar las posiciones de las partes interesadas en una situación dada [Familiarizarse] • Analizar errores lógicos básicos en una discusión [Usar] • Analizar un argumento para identificar premisas y la conclusión [Familiarizarse] • Ilustrar el uso de ejemplo y analogía en el argumento ético [Familiarizarse] • Evaluar compensaciones éticos / sociales en las decisiones técnicas [Familiarizarse]
Readings : [LL04], [McL00]	

Unit 4: Ética Profesional (4)

Competences Expected:

Topics	Learning Outcomes
• Community values and the laws by which we live. • La naturaleza del profesionalismo incluido el cuidado, la atención y la disciplina, la responsabilidad fiduciaria y mentoría. • Mantenerse al día como profesional de computación en términos de familiaridad, herramientas, habilidades, marco legal y profesional, así como la capacidad de autoevaluarse y avances en el campo de la computación. • La certificación profesional, códigos de ética, conducta y práctica, como la ACM / IEEE-CS, SE, AITP, IFIP y las sociedades internacionales. • Rendición de cuentas, la responsabilidad y la confiabilidad (por ejemplo, la corrección de software, fiabilidad y seguridad, así como la confidencialidad ética de los profesionales de seguridad cibernética) • El papel del profesional de de computación en las políticas públicas. • Mantenimiento de la conciencia en relación a las consecuencias. • Disidencia ética y la denuncia de irregularidades. • La relación entre la cultura regional y dilemas éticos. • Tratar con el acoso y la discriminación. • Formas de credenciamiento profesional. • Políticas de uso aceptable para la computación en el lugar de trabajo. • Ergonomía y entornos de trabajo computacionales saludables. • Consideraciones a tiempos de entrega de mercado vs estándares de calidad profesional.	• Identificar los problemas éticos que se plantean en el desarrollo de software y determinar cómo abordarlos técnica y éticamente [Usar] • Explicar la responsabilidad ética de velar por la corrección de software, confiabilidad y seguridad [Evaluar] • Describir los mecanismos que normalmente existen para que profesional se mantenga al día [Familiarizarse] • Describir las fortalezas y debilidades de códigos profesionales relevantes como expresiones de profesionalismo y guías para la toma de decisiones [Familiarizarse] • Analizar un problema mundial de computación, observando el papel de los profesionales y funcionarios del gobierno en el manejo de este problema [Familiarizarse] • Evaluar los códigos de ética profesional de la ACM, la Sociedad de Computación de la IEEE, y otras organizaciones [Familiarizarse] • Describir las formas en que los profesionales pueden contribuir a las políticas públicas [Familiarizarse] • Describir las consecuencias de la conducta profesional inadecuada [Usar] • Identificar las etapas progresivas en un incidente de denuncia de irregularidades [Usar] • Identificar ejemplos de cómo interactúa la cultura regional con dilemas éticos [Familiarizarse] • Investigar las formas de acoso, discriminación y formas de ayuda [Usar] • Examine las diversas formas de acreditación de profesionales [Usar] • Explicar la relación entre la ergonomía en los ambientes y la salud de las personas de computación [Usar] • Desarrollar un uso del computador/política de uso aceptable con medidas coercitivas [Familiarizarse] • Describir los problemas asociados con la presión de la industrias para centrarse en el tiempo de comercialización en comparación con la aplicación de normas de calidad profesional [Usar]

Readings : [LL04], [McL00], [Edi09a], [Edi09b], [Edi10]

Unit 5: Propiedad Intelectual (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Fundamentos filosóficos de propiedad intelectual. • Derechos de propiedad intelectual. • Propiedad intelectual digital intangible (IDIP). • Fundamentos legales para protección de la propiedad intelectual. • Gestión de derechos digitales. • Copyrights, patentes, secretos de comercio, marcas registradas. • Plagiarismo. • Fundamentos del movimiento Open Source. • Piratería de Software.	• Discute la racionalidad de la protección legal de la propiedad intelectual [Evaluar] • Discute las bases filosóficas de la propiedad intelectual [Familiarizarse] • Decribe la legislación orientada a los delitos de derechos de autor digitales [Evaluar] • Critica la legislación orientada a los delitos digitales de derechos de autor [Familiarizarse] • Identifica ejemplos contemporáneos de propiedad intelectual digital intangible [Evaluar] • Justifica el uso de material con derechos de autor [Evaluar] [Familiarizarse] • Evalúa los asuntos éticos inherentes a diversos mecanismos de detección de plagio [Familiarizarse] • Interpreta el intento y la implementación de licencias de software [Familiarizarse] • Discute asuntos que involucran la seguridad de patentes en software [Familiarizarse] • Caracteriza y contrasta los conceptos de derechos de autor, patentes y de marcas comerciales [Familiarizarse] • Identifica los objetivos del movimiento de software libre [Evaluar] • Identifica los objetivos del movimiento de software libre [Familiarizarse]
Readings : [LL04], [McL00], [Edi09a], [Edi09b], [Edi10]	

Unit 6: Privacidad y Libertades Civiles (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Fundamentos filosóficos de derechos de privacidad. • Fundamentos legales de protección de privacidad. • Implicaciones de privacidad de recopilación de datos generalizada de bases de datos transaccionales, almacenes de datos, sistemas de vigilancia y la computación en la nube. • Ramificaciones de privacidad diferencial. • Soluciones basadas en la tecnología para la protección de la privacidad. • Legislación de privacidad en áreas de práctica. • Libertades civiles y diferencias culturales. • Libertad de expresión y sus limitaciones.	• Discute las bases filosóficas para la protección legal de la privacidad personal [Familiarizarse] • Evalúa soluciones para amenazas a la privacidad en bases de datos transaccionales y almacenes de datos [Familiarizarse] • Describe los roles de la recolección de datos en la implementación de sistemas de vigilancia intrusiva (ejm. RFID, reconocimiento de rostro, cobro electrónico, computación móvil) [Familiarizarse] • Describe las ramificaciones de la privacidad diferenciada [Familiarizarse] • Investica el impacto de soluciones tecnológicas a los problemas de privacidad [Familiarizarse] • Critica la intención, el valor potencial y la implementación de las diversas formas de legislación en privacidad [Familiarizarse] • Identifica estrategias que permitan la apropiada libertad de expresión [Familiarizarse]
Readings : [LL04], [McL00], [Edi09a], [Edi09b], [Edi10]	

Unit 7: Políticas de seguridad, Leyes y crímenes computacionales (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Ejemplos de delitos informáticos y reparación legal para delincuentes informáticos. • Ingenieria social, robo de identidad y recuperación. • Tópicos relacionados al uso de acceso indebido y las infracciones y materia de seguridad. • Motivaciones y ramificaciones del ciberterrorismo y el hacking criminal, cracking. • Efectos de malware, como virus, worms y Trojan horses. • Estrategias de prevención de Crimen. • Políticas de Seguridad.	• Listar ejemplos clásicos de delitos informáticos y incidentes de ingeniería social con impacto social [Familiarizarse] • Indentificar leyes que se aplican a delitos informáticos [Familiarizarse] • Describir la motivación y ramificaciones de cybert-errorismo y hackeo criminal [Familiarizarse] • Examinar los problemas éticos y legales relacionados con el mal uso de accesos y diversas violaciones en la seguridad [Familiarizarse] • Discutir el rol del profesional en seguridad y los problemas que están envueltos [Familiarizarse] • Investigar medidas que puedan ser consideradas por personas y organizaciones incluyendo al gobierno para prevenir o mitigar efectos indeseables de los delitos informáticos y robo de identidad [Familiarizarse] • Escribir una politica de seguridad de una empresa, la cual incluye procedimientos para administrar contraseñas y monitorizar a los empleados [Familiarizarse]
Readings : [LL04], [McL00], [Edi09a], [Edi09b], [Edi10]	

Unit 8: Economía de la Computación (2)	
Competences Expected:	
Topics	**Learning Outcomes**
• Monopolio y sus implicaciones económicas. • Efecto del suministro de mano de obra calificada y la demanda sober la calidad de los productos de computación. • Estrategias de precio en el dominio de la computación. • El fenomeno del desarrollo de software outsourcing y off-shoring; impactos en el empleo y la economia. • Consecuencias de la globalización para la profesión de Ciencias de la Computación. • Diferencias en acceso a recursos de computación y el posible efecto de los mismos. • Analisis costo/beneficio de trabajos con consideraciones para manufactura, hardware, software e implicaciones de ingeniería. • Costo estimado versus costo actual in relacion al costo total. • Emprendimiento: perspectivas y entrampamientos. • Efectos de red o economias de escala del lado de la demanda. • El uso de la ingeniería económica para hacer frente a las finanzas.	• Resumir los fundamentos para los esfuerzos antimonopolio [Familiarizarse] • Identificar diversas maneras en que la industria de la tecnología de la información está afectada por la escasez de la oferta de trabajo [Familiarizarse] • Identificar la evolución de la estrategía de precios para el cálculo de los bienes y servicios [Familiarizarse] • Discutir los beneficios, los inconvenientes y las implicaciones de *off-shoring* y *outsourcing* [Familiarizarse] • Investigar y defender maneras de tratar las limitaciones en el acceso a la computación. [Usar] • Describir los beneficios económicos de efectos de la red [Usar]
Readings : [LL04], [McL00], [Edi09a], [Edi09b], [Edi10]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[McL00] Raymond McLeod Jr. *Sistemas de Información Gerencial*. Prentice Hall, 2000.

[LL04] Kenneth C. Laudon and Jane P. Laudon. *Sistemas de Información Gerencial*. Prentice Hall, 2004.

[Edi09a] Datamation Ediciones, ed. *Revista Datamation MC Ediciones*. 2009.

[Edi09b] Datamation Ediciones, ed. *Understanding the Digital Economy*. 2009.

[Edi10] Datamation Ediciones, ed. *Financial Times Mastering Information Management*. 2010.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS3P1. Parallel and Distributed Computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3P1. Parallel and Distributed Computing
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- CS212. Analysis and Design of Algorithms. (5^{th} Sem)
- CS231. Networking and Communication. (6^{th} Sem)
- CS212. Analysis and Design of Algorithms. (5^{th} Sem)
- CS231. Networking and Communication. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The last decade has brought explosive growth in computing with multiprocessors, including Multi-core processors and distributed data centers. As a result, computing parallel and distributed has become a widely elective subject to be one of the main components in the mesh studies in computer science undergraduate. Both parallel and distributed computing the simultaneous execution of multiple processes, whose operations have the potential to intercalar in a complex way. Parallel and distributed computing builds on foundations in many areas, including understanding the fundamental concepts of systems, such as: concurrency and parallel execution, consistency in state / memory manipulation, and latency. The communication and coordination between processes has its foundations in the passage of messages and models of shared memory of computing and algorithmic concepts like atomicity, consensus and conditional waiting.Achieving acceleration in practice requires an understanding of parallel algorithms, strategies for decomposition problem, systems architecture, implementation strategies and analysis of performance. Distributed systems highlight the problems of security and tolerance to Failures, emphasize the maintenance of the replicated state and introduce additional problems in the field of computer networks.

5. GOALS

- That the student is able to create parallel applications of medium complexity by efficiently leveraging machines with multiple cores.
- That the student is able to compare sequential and parallel applications.
- That the student is able to convert, when the situation warrants, sequential applications to parallel efficiently

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Fundamentos de paralelismo (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Procesamiento Simultáneo Múltiple. • Metas del Paralelismo (ej. rendimineto) frente a Concurrencia (ej. control de acceso a recursos compartidos) • Paralelismo, comunicación, y coordinación: – Paralelismo, comunicación, y coordinación – Necedidad de Sincronización • Errores de Programación ausentes en programación secuencial: – Tipos de Datos (lectura/escritura simultánea o escritura/escritura compartida) – Tipos de Nivél más alto (interleavings violating program intention, no determinismo no deseado) – Falta de vida/progreso (deadlock, starvation)	• Distinguir el uso de recursos computacionales para una respuesta mas rápida para administrar el acceso eficiente a un recurso compartido [Familiarizarse] • Distinguir múltiples estructuras de programación suficientes para la sincronización que pueden ser interimplementables pero tienen ventajas complementarias [Familiarizarse] • Distinguir datos de carrera (*data races*) a partir de carreras de mas alto nivel [Familiarizarse]
Readings : [Pac11], [Mat14], [**quinnz**], [Geo10]	

Unit 2: Arquitecturas paralelas (12)	
Competences Expected:	
Topics	**Learning Outcomes**

• Procesadores mutlinúcleo.
• Memoria compartida vs memoria distribuida.
• Multiprocesamiento simétrico.
• SIMD, procesamiento de vectores.
• GPU, coprocesamiento.
• Taxonomia de Flynn.
• Soporte a nivel de instrucciones para programación paralela.
  – Instrucciones atómicas como Compare/Set (Comparar / Establecer)
• Problemas de Memoria:
  – Caches multiprocesador y coherencia de cache
  – Acceso a Memoria no uniforme (NUMA)
• Topologías.
  – Interconecciones
  – Clusters
  – Compartir recursos (p.e., buses e interconexiones)
	• Explicar las diferencias entre memoria distribuida y memoria compartida [Evaluar]
	• Describir la arquitectura SMP y observar sus principales caracteristicas [Evaluar]
	• Distinguir los tipos de tareas que son adecuadas para máquinas SIMD [Usar]
	• Describir las ventajas y limitaciones de GPUs vs CPUs [Usar]
	• Explicar las caracteristicas de cada clasificación en la taxonomía de Flynn [Usar]
	• Describir los desafíos para mantener la coherencia de la caché [Familiarizarse]
	• Describir los desafíos para mantener la coherencia de la caché [Familiarizarse]

Readings : [Pac11], [KH13], [SK10], [Geo10]	

Unit 3: Descomposición en paralelo (18)	
Competences Expected:	
Topics	**Learning Outcomes**

• Necesidad de Comunicación y coordinación/sincronización.
• Independencia y Particionamiento.
• Conocimiento Básico del Concepto de Descomposición Paralela.
• Decomposición basada en tareas:
  – Implementación de estrategias como hebras
• Descomposición de Información Paralela
  – Estrategias como SIMD y MapReduce
• Actores y Procesos Reactivos (solicitud de gestores)
	• Explicar por qué la sincronización es necesaria en un programa paralelo especifico [Usar]
	• Identificar oportunidades para particionar un programa serial en módulos paralelos independientes [Familiarizarse]
	• Escribir un algoritmo paralelo correcto y escalable [Usar]
	• Paralelizar un algoritmo mediante la aplicación de descomposición basada en tareas [Usar]
	• Paralelizar un algoritmo mediante la aplicación de descomposición datos en paralelo [Usar]
	• Escribir un programa usando actores y/o procesos reactivos [Usar]

Readings : [Pac11], [Mat14], [Qui03], [Geo10]	

Unit 4: Comunicación y coordinación (18)

Competences Expected:

Topics	**Learning Outcomes**
• Memoria Compartida. • La consistencia, y su papel en los lenguaje de programación garantias para los programas de carrera libre. • Pasos de Mensaje: – Mensajes Punto a Punto versus multicast (o basados en eventos) – Estilos para enviar y recibir mensajes Blocking vs non-blocking – Buffering de mensajes • Atomicidad: – Especificar y probar atomicidad y requerimientos de seguridad – Granularidad de accesos atómicos y actualizaciones, y uso de estructuras como secciones críticas o transacciones para describirlas – Exclusión mutua usando bloques, semáforos, monitores o estructuras relacionadas ∗ Potencial para fallas y bloqueos (*deadlock*) (causas, condiciones, prevención) – Composición ∗ Componiendo acciones atómicas granulares más grandes usando sincronización ∗ Transacciones, incluyendo enfoques optimistas y conservadores • Consensos: – (Ciclicos) barerras, contadores y estructuras relacionadas • Acciones condicionales: – Espera condicional (p.e., empleando variables de condición)	• Usar exclusión mútua para evitar una condición de carrera [Usar] • Dar un ejemplo de una ordenación de accesos entre actividades concurrentes (por ejemplo, un programa con condición de carrera) que no son secuencialmente consistentes [Familiarizarse] • Dar un ejemplo de un escenario en el que el bloqueo de mensajes enviados pueden dar *deadlock* [Usar] • Explicar cuándo y por qué mensajes de multidifusión (*multicast*) o basado en eventos puede ser preferible a otras alternativas [Familiarizarse] • Escribir un programa que termine correctamente cuando todo el conjunto de procesos concurrentes hayan sido completados [Usar] • Dar un ejemplo de una ordenación de accesos entre actividades concurrentes (por ejemplo, un programa con condición de carrera) que no son secuencialmente consistentes [Familiarizarse] • Usar semaforos o variables de condición para bloquear hebras hasta una necesaria precondición de mantenga [Usar]
Readings : [Pac11], [Mat14], [Qui03], [Geo10]	

Unit 5: Análisis y programación de algoritmos paralelos (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Caminos críticos, el trabajo y la duración y la relación con la ley de Amdahl. • Aceleración y escalabilidad. • Naturalmente (vergonzosamente) algoritmos paralelos. • Patrones Algoritmicos paralelos (divide-y-conquista, map/reduce, amos-trabajadores, otros) – Algortimos específicos (p.e., MergeSort paralelo) • Algoritmos de grafos paralelo (por ejemplo, la ruta más corta en paralelo, árbol de expansión paralela) • Cálculos de matriz paralelas. • Productor-consumidor y algoritmos paralelos segmentados. • Ejemplos de algoritmos paralelos no-escalables.	• Definir: camino crítico, trabajo y *span* [Familiarizarse] • Calcular el trabajo y el *span* y determinar el camino crítico con respecto a un diagrama de ejecución paralela. [Usar] • Definir *speed-up* y explicar la noción de escalabilidad de un algoritmo en este sentido [Familiarizarse] • Identificar tareas independientes en un programa que debe ser paralelizado [Usar] • Representar características de una carga de trabajo que permita o evite que sea naturalmente paralelizable [Familiarizarse] • Implementar un algoritmo dividir y conquistar paralelo (y/o algoritmo de un grafo) y medir empiricamente su desempeño relativo a su analogo secuencial [Usar] • Descomponer un problema (por ejemplo, contar el número de ocurrencias de una palabra en un documento) via operaciones *map* y *reduce* [Usar] • Proporcionar un ejemplo de un problema que se corresponda con el paradigma productor-consumidor [Usar] • Dar ejemplos de problemas donde el uso de *pipelining* sería un medio eficaz para la paralelización [Usar] • Implementar un algoritmo de matriz paralela [Usar] • Identificar los problemas que surgen en los algoritmos del tipo productor-consumidor y los mecanismos que pueden utilizarse para superar dichos problemas [Usar]
Readings : [Mat14], [Qui03], [Geo10]	

Unit 6: Desempeño en paralelo (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Equilibrio de carga. • La medición del desempeño. • Programación y contención. • Evaluación de la comunicación de arriba. • Gestión de datos: – Costos de comunicación no uniforme debidos a proximidad – Efectos de Cache (p.e., false sharing) – Manteniendo localidad espacial • Consumo de energía y gestión.	• Detectar y corregir un desbalanceo de carga [Usar] • Calcular las implicaciones de la ley de Amdahl para un algoritmo paralelo particular [Usar] • Describir como la distribuición/disposición de datos puede afectar a los costos de comunicación de un algoritmo [Familiarizarse] • Detectar y corregir una instancia de uso compartido falso (*false sharing*) [Usar] • Explicar el impacto de la planificación en el desempeño paralelo [Familiarizarse] • Explicar el impacto en el desempeño de la localidad de datos [Familiarizarse] • Explicar el impacto y los puntos de equilibrio relacionados al uso de energía en el desempeño paralelo [Familiarizarse]
Readings : [Pac11], [Mat14], [KH13], [SK10], [Geo10]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Qui03] Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. 1st. McGraw-Hill Education Group, 2003.

[Geo10] Gerhard Wellein Georg Hager. *Introduction to High Performance Computing for Scientists and Engineers (Chapman and Hall/CRC Computational Science)*. Ed. by CRC Press. 1st. 2010.

[SK10] Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. 1st. Addison-Wesley Professional, 2010.

[Pac11] Peter S. Pacheco. *An Introduction to Parallel Programming*. 1st. Morgan Kaufmann, 2011.

[KH13] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013.

[Mat14] Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014. URL: `http://heather.cs.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf`.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS361. Computational Vision (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS361. Computational Vision
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS262. Machine learning. (7^{th} Sem) CS262. Machine learning. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Provee una serie de herramientas para resolver problemas que son difíciles de solucionar con los métodos algorítmicos tradicionales. Incluyendo heurísticas, planeamiento, formalismos en la representación del conocimiento y del razonamiento, técnicas de aprendizaje en máquinas, técnicas aplicables a los problemas de acción y reacción: asi como el aprendizaje de lenguaje natural, visión artificial y robótica entre otros.

5. GOALS

- Realizar algún curso avanzado de Inteligencia Artificial sugerido por el curriculo de la ACM/IEEE.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: (60)	
Competences Expected:	
Topics	**Learning Outcomes**
• CS360. Inteligencia Artificial • CS361. Razonamiento automatizado • CS362. Sistemas Basados en Conocimiento • CS363. Aprendizaje de Maquina [RN03],[Hay99] • CS364. Sistemas de Planeamiento • CS365. Procesamiento de Lenguaje Natural • CS366. Agentes • CS367. Robótica • CS368. Computación Simbólica • CS369. Algoritmos Genéticos [Gol89]	• Profundizar en diversas técnicas relacionadas a la Inteligencia Artificial [Usar]
Readings : [RN03], [Hay99], [Gol89]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Gol89] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[Hay99] Simon Haykin. *Neural networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[RN03] Stuart Russell and Peter Norvig. *Inteligencia Artifical: Un enfoque moderno*. Prentice Hall, 2003.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS370. Big Data (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS370. Big Data
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- CS272. Databases II. (5^{th} Sem)
- CS3P1. Parallel and Distributed Computing . (8^{th} Sem)
- CS272. Databases II. (5^{th} Sem)
- CS3P1. Parallel and Distributed Computing . (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Nowadays, knowing scalable approaches to processing and storing large volumes of information (terabytes, petabytes and even exabytes) is fundamental in computer science courses. Every day, every hour, every minute generates a large amount of information which needs to be processed, stored, analyzed.

5. GOALS

- That the student is able to create parallel applications to process large volumes of information
- That the student is able to compare the alternatives for the processing of big data
- That the student is able to propose architectures for a scalable application

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Introducción a Big Data (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Overview on Cloud Computing • Distributed File System Overview • Overview of the MapReduce programming model	• Explain the concept of Cloud Computing from the point of view of Big Data[Familiarizarse] • Explain the concept of Distributed File System [Familiarizarse] • Explain the concept of the MapReduce programming model[Familiarizarse]
Readings : [Cou+11]	

Unit 2: Hadoop (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Hadoop overview. • History. • Hadoop Structure. • HDFS, Hadoop Distributed File System. • Programming Model MapReduce	• Understand and explain the Hadoop suite [Familiarizarse] • Implement solutions using the MapReduce programming model. [Usar] • Understand how data is saved in the HDFS. [Familiarizarse]
Readings : [HDF11], [BVS13]	

Unit 3: Procesamiento de Grafos en larga escala (10)	
Competences Expected:	
Topics	**Learning Outcomes**
• Pregel: A System for Large-scale Graph Processing. • Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. • Apache Giraph is an iterative graph processing system built for high scalability.	• Understand and explain the architecture of the Pregel project. [Familiarizarse] • Understand the GraphLab project architecture. [Familiarizarse] • Understand the architecture of the Giraph project. [Familiarizarse] • Implement solutions using Pregel, GraphLab or Giraph. [Usar]
Readings : [Low+12], [Mal+10], [Bal+08]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Bal+08] Shumeet Baluja et al. "Video Suggestion and Discovery for Youtube: Taking Random Walks Through the View Graph". In: *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. Beijing, China: ACM, 2008, pp. 895–904. DOI: 10.1145/1367497.1367618. URL: http://doi.acm.org/10.1145/1367497.1367618.

[Mal+10] Grzegorz Malewicz et al. "Pregel: A System for Large-scale Graph Processing". In: SIGMOD '10 (2010), pp. 135–146. DOI: 10.1145/1807167.1807184. URL: http://doi.acm.org/10.1145/1807167.1807184.

[Cou+11] George Coulouris et al. *Distributed Systems: Concepts and Design*. 5th. USA: Addison-Wesley Publishing Company, 2011.

[HDF11] Kai Hwang, Jack Dongarra, and Geoffrey C. Fox. *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[Low+12] Yucheng Low et al. "Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud". In: *Proc. VLDB Endow.* 5.8 (Apr. 2012), pp. 716–727. DOI: 10.14778/2212351.2212354. URL: http://dx.doi.org/10.14778/2212351.2212354.

[BVS13] Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. *Mastering Cloud Computing: Foundations and Applications Programming*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CB309. Bioinformatics (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CB309. Bioinformatics
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	2
2.4 Horas	:	1 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	

- CS212. Analysis and Design of Algorithms. (5^{th} Sem)
- MA307. Mathematics applied to computing . (6^{th} Sem)
- CS212. Analysis and Design of Algorithms. (5^{th} Sem)
- MA307. Mathematics applied to computing . (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The use of computational methods in the biological sciences has become one of the key tools for the field of molecular biology, being a fundamental part of research in this area.
In Molecular Biology, there are several applications that involve both DNA, protein analysis or sequencing of the human genome, which depend on computational methods. Many of these problems are really complex and deal with large data sets.
This course can be used to see concrete use cases of several areas of knowledge of Computer Science such as Programming Languages (PL), Algorithms and Complexity (AL), Probabilities and Statistics, Information Management (IM), Intelligent Systems (IS).

5. GOALS

- That the student has a solid knowledge of molecular biological problems that challenge computing.
- That the student is able to abstract the essence of the various biological problems to pose solutions using their knowledge of Computer Science

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Usage)

7. TOPICS

Unit 1: Introduction to Molecular Biology (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Review of organic chemistry: molecules and macromolecules, sugars, nucleic acids, nucleotides, RNA, DNA, proteins, amino acids and levels of structure in proteins. • The Dogma of Life: From DNA to Proteins, Transcription, Translation, Protein Synthesis. • Genome study: Maps and sequences, specific techniques	• Achive a general knowledge of the most important topics in Molecular Biology. [Familiarizarse] • Understand that biological problems are a challenge to the computational world. [Evaluar]
Readings : [CB00], [SM97]	

Unit 2: Sequence Comparison (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Sequences of nucleotides and amino acid sequences. • Sequence alignment, paired alignment problem, exhaustive search, Dynamic programming, global alignment, local alignment, gaps penalty • Comparison of multiple sequences: sum of pairs, complexity analysis by dynamic programming, alignment heuristics, star algorithm, progressive alignment algorithms.	• Understand and solve the problem of aligning a pair of sequences. [Usar] • Understand and solve the problem of multiple sequence alignment. [Usar] • Know the various algorithms for aligning existing sequences in the literature . [Familiarizarse]
Readings : [CB00], [SM97], [Pev00]	

Unit 3: Phylogenetic Trees (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Phylogeny: Introduction and phylogenetic relations • Phylogenetic trees: definition, type of trees, problem of search and reconstruction of trees • Reconstruction methods: parsimony methods, distance methods, maximum likelihood methods, confidence of reconstructed trees	• Understand the concept of phylogeny, phylogenetic trees and the methodological difference between biology and molecular biology. [Familiarizarse] • Understand the problem of the reconstruction of phylogenetic trees, to know and apply the main algorithms for the reconstruction of phylogenetic trees. [Evaluar]
Readings : [CB00], [SM97], [Pev00]	

Unit 4: DNA Sequence Assembling (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Biological basis: ideal case, difficulties, alternative methods for DNA sequencing • Formal Assembly Models: Shortest Common Superstring, Reconstruction, Multicontig • Algorithms for sequence assembly: representation of overlaps, paths to create superstrings, voracious algorithm, acyclic graphs. • Assembly heuristics: search for overlays, ordering fragments, alignments and consensus.	• Understand the computational challenge of the Sequence Assembly problem. [Familiarizarse] • Understand the principle of formal model for assembly. [Evaluar] • Know the main heuristics for the problem of assembjale of DNA sequences[Usar]
Readings : [SM97], [Alu06]	

Unit 5: Secondary and tertiary structures (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Molecular structures: primary, secondary, tertiary, quaternary. • Prediction of secondary structures of RNA: formal model, pair energy, structures with independent bases, solution with Dynamic Programming, structures with loops. • *Protein folding*: Estructuras en proteinas, problema de protein folding. • *Protein Threading*: Definitions, Branch Bound Algorithm, Branch Bound for protein threading. • *Structural Alignment*: Definitions, DALI algorithm	• Know the protein structures and the necessity of computational methods for the prediction of the geometry. [Familiarizarse] • Know the algorithms for solving prediction problems of secondary structures RNA, and structures in proteins. [Evaluar]
Readings : [SM97], [CB00], [Alu06]	

Unit 6: Probabilistic Models in Molecular Biology (4)	
Competences Expected:	
Topics	**Learning Outcomes**
• Probability: Random Variables, Markov Chains, Metropoli-Hasting Algorithm, Markov Random Fields, and Gibbs Sampler, Maximum Likelihood. • Hidden Markov Models (HMM), parameter estimation, Viterbi algorithm and Baul-Welch method, Application in paired and multiple alignments, Motifs detection in proteins, in eukaryotic DNA, in sequences families. • Probabilistic phylogeny: probabilistic models of evolution, likelihood of alignments, likelihood for inference, comparison of probailistic and non-probabilistic methods	• Review concepts of Probabilistic Models and understand their importance in Computational Molecular Biology. [Evaluar] • Know and apply Hidden Markov Models for various analyzes in Molecular Biology.. [Usar] • Know the application of probabilistic models in Phylogeny and to compare them with non-probabilistic models[Evaluar]
Readings : [Dur+98], [CB00], [Alu06], [Kro+94]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Kro+94] Anders Krogh et al. "Hidden Markov Models in Computational Biology, Applications to Protein Modeling". In: *J Mol. Biol* 235 (1994), pp. 1501–1531.

[SM97] João Carlos Setubal and João Meidanis. *Introduction to computational molecular biology.* Boston: PWS Publishing Company, 1997, pp. I–XIII, 1–296.

[Dur+98] R. Durbin et al. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998, p. 357.

[CB00] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction.* 279 pages. John Wiley & Sons Ltd., 2000.

[Pev00] Pavel A. Pevzner. *Computational Molecular Biology: an Algorithmic Approach.* Cambridge, Massachusetts: The MIT Press, 2000.

[Alu06] Srinivas Aluru, ed. *Handbook of Computational Molecular Biology.* Computer and Information Science Series. Boca Raton, FL: Chapman & Hall, CRC, 2006.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS369. Topics in Artificial Intelligence (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS369. Topics in Artificial Intelligence
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS261. Artificial Intelligence. (6^{th} Sem) CS261. Artificial Intelligence. (6^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

La Computación Evolutiva comprende un conjunto de metodologías de búsqueda y optimización cuya base primordial es el Paradigma Neodarwiniano que agrupa la Herencia Genética (Mendel), el Seleccionismo (Weismann) y la Evolución de las Especies (Darwin) que, cuando llevadas a implementaciones computacionales, ofrecen una herramienta poderosa de optimización global para una determinada función objetivo. Son bastante robustos cuando se supone la existencia de muchos óptimos locales. De esta forma, estos algoritmos pueden aplicarse en diversos problemas de optimización.

5. GOALS

- Que el alumno sea capaz de entender y aplicar el Paradigma Neodarwiniano para solucionar problemas complejos de optimización.
- Entendimiento a detalle del principio, fundamentos teóricos, funcionamiento, implementación, intepretación de resultados y operación de los algoritmos de la Computación Evolutiva más populares y utilizados por la comunidad científica y profesional.
- Conocimiento del estado del arte en Computación Evolutiva
- Capacidad de tratar un problema real de optimización utilizando Computación Evolutiva

6. COMPETENCES

a) An ability to apply knowledge of mathematics, science. (Usage)

b) An ability to design and conduct experiments, as well as to analyze and interpret data. (Usage)

i) An ability to use the techniques, skills, and modern computing tools necessary for computing practice. (Usage)

j) Apply the mathematical basis, principles of algorithms and the theory of Computer Science in the modeling and design of computational systems in such a way as to demonstrate understanding of the equilibrium points involved in the chosen option. (Usage)

7. TOPICS

Unit 1: Introducción a la Optimización (4)	
Competences Expected: a,b	
Topics	**Learning Outcomes**
• Definiciones de Optimización: principio de estabilidad, optimización global. • Optimización Clásica: Definición del problema de optimización, concepto de convexidad, optimización numérica y combinatoria. • Técnicas de optimización clásica: optimización lineal, algoritmo simplex, optimización no lineal, algoritmos *steepest descent, conjugate gradient*, algoritmos de búsqueda, programación dinámica, • Heurísticas: definición, *Tabu search*, *Hill Climbing Simulated Annealing*, *Evolutionary Algorithms*	• Entender los principios básicos de la optimización • Entender e implementar algoritmos básicos de Optimización aplicados a problemas *benchmark*. • Entender la necesidad de uso de heurísticas
Readings : [Wei09], [RBK12]	

Unit 2: Computación Evolutiva: Conceptos básicos (8)	
Competences Expected: a,b,i	
Topics	**Learning Outcomes**
• Computación Evolutiva: definiciones • Ideas precursoras: El origen de las ideas, L'Eclerc, Lamarck, Darwin, Weismann, Mendel, Baldwin, Paradigma Neodarwiniano • Conceptos básicos de Computación Evolutiva: genes, cromosomas, individuos, población. • Paradigmas de la Computación Evolutiva: Programación Evolutiva, Estrategias Evolutivas, Algoritmos Genéticos, *Learning Classifier Systems*, Programación Genética.	• Entender los principios básicos que rigen la computación evolutiva • Conocer el contexto en que surgió la computación evolutiva.
Readings : [RBK12], [Wei09], [Fog95], [**koza98**], [Mit04], [Mic96]	

Unit 3: Algoritmo Genético Canónico (8)	
Competences Expected: a,b,i	
Topics	**Learning Outcomes**
• Algoritmo Genético: definición, componentes. • Algoritmo Genético Canónico: procedimiento elemental, ciclo de un AG, representación (codificación binaria, real a binario, decodificación binario a real), inicialización de la población, evaluación y aptitud, selección (proporcional, torneo), operadores genéticos (cruces, mutaciones), el dilema *exploiting-exploring*, ajustes en la aptitud, ajustes en la selección. • Monitoreo de un AG: curvas *best-so-far, online, offline* • Convergencia • Teoría de *Schemata*: Máscaras, esquemas, definiciones y propiedades, *Schemata theorem*: impacto de la selección, cruce de 1 punto y mutación, teorema fundamental de los algoritmos genéticos, hipótesis de los bloques constructores.	• Entender los algoritmos genéticos tradicionales. • Analizar y evaluar ventajas y desventajas del modelo genético tradicional. • Implementar un ejemplo de algoritmo genético tradicional y analizar su comportamiento.
Readings : [RBK12], [Hol75], [Gol89], [Mit04], [Mic96]	

Unit 4: Algoritmos Evolutivos en Optimización Numérica (8)	
Competences Expected: a,b,i	
Topics	**Learning Outcomes**
• Problemas con restricciones: definiciones, espacios válido e inválido. • Tratamiento de las restricciones: Penalización, reparación, uso de codificadores, operadores especializados. • Uso de codificación real: binario vs. real, algoritmo evolutivo con codificación real. • Modelo GENOCOP: tratamiento de restricciones lineales, inicialización, operadores, inicialización, modelo GENOCOP III para restricciones no lineales: reparación de individuos.	• Comprensión de las formas de tratar problemas de optimización con restricciones. • Entender y analizar los algoritmos evolutivos con codificación real. • Evaluar la aplicación de computación evolutiva en problemas de optimización numérica
Readings : [RBK12], [Mic96], [Mic00], [SC00]	

Unit 5: Algoritmos Evolutivos en Optimización Combinatoria (8)	
Competences Expected: a,b,i	
Topics	**Learning Outcomes**
• Espacios discretos y finitos • Algoritmos Evolutivos discretos: definición, modelo discreto generalizado • Algoritmos Evolutivos de orden: representación de soluciones, operadores de orden: cruces, mutaciones • Aplicaciones: *Quadratic assignment Problem* – QAP, *Travelling Salesman Problem* – TSP • Problemas de Planificación: variables típicas, características, representación, codificadores, evaluación de una planificación.	• Comprender e identificar el uso de Computación Evolutiva en problemas de optimización combinatoria • Evaluar la aplicación de computación evolutiva en problemas reales discretos
Readings : [RBK12], [Mit04], [Cru03]	

Unit 6: Paralelización y Multi objetivos (8)	
Competences Expected: a,b,i,j	
Topics	**Learning Outcomes**
• PEA – Algoritmos Evolutivos en Paralelo: arquitecturas de paralelización, arquitecturas *master-slave*, *coarse-grained*, *fine-grained* e híbridas • Análisis de la ejecución de una implementación *master-slave*. • Optimización de Multiples Objetivos: Definición formal, criterio de Pareto, Algoritmos Evolutivos Multi Objetivos (MOEA) sin uso de Pareto, MOEA con uso de Pareto: MOGA, NSGA, NPGA, NPGA2, PESA, SPEA, SPEA-II, Algoritmo Microgenético. • MOEA – Métricas de desempeño, investigación futura	• Comprender y analizar la capacidad de paralelización de los modelos evolutivos • Analizar la aplicabilidad de Computación Evolutiva en problemas de múltiples objetivos • Implementación de modelos paralelos y multiobjetivo
Readings : [RBK12], [Can00], [Coe07]	

Unit 7: Algoritmos Genéticos Avanzados (16)	
Competences Expected: a,b,i,j	
Topics	**Learning Outcomes**
• HEA – Algoritmos Evolutivos Híbridos: Por qué hibridizar?, formas de hibridización, búsqueda local y aprendizaje. • GP – Programación Genética: definición, representación, ciclo de la GP. • CA – Algoritmos Culturales: Evolución Cultural, componentes, procedimiento, espacio de creencia, operadores culturales. • CoEv – Coevolución: carácteristicas, modelo competitivo, modelo cooperativo. • DE – Evolución Diferencial: inicialización, operaciones, selección, DE vs. GA, variantes de DE, *Dynamic DE* • QIEA – Algoritmos Evolutivos con Inspiración Quántica: Computación quántica, algoritmos con inspiración quántica, QIEA-**B**, QIEA-**R**	• Reconocer y analizar la necesidad de usar Algoritmos Evolutivos más avanzados • Implementación de modelos avanzados de computación evolutiva
Readings : [RBK12], [El-+06], [Koz92], [**Reynolds94**], [SP95], [Cru07]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Hol75] John Henry Holland. *Adaptation in Natural and Artificial Systems*. first. Ann Arbor, Michigan: University of Michigan Press, 1975.

[SP95] Rainer Storn and Kenneth Price. *Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. Tech. rep. TR-95-012. Berkeley, California: International Computer Science Institute, Mar. 1995.

[Mic96] Zibgniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[Can00] Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[Mic00] Zbigniew Michalewicz. "Introduction to constraint-handling tecniques, Decoders, Repair algorithms, Constraint-preserving operators". In: *Evolutionary Computation 2, Advanced Algorithms and Operators* (2000), pp. 38–40, 49–55, 56–61, 62–68.

[SC00] Alice E. Smith and David W. Coit. "Penalty functions". In: *Evolutionary Computation 2, Advanced Algorithms and Operators* (2000), pp. 41–48.

[Cru03] André Vargas Abs da Cruz. "Otimização de planejamento com restrições de precedência usando algoritmos genéticos e co-evolução cooperativa". MA thesis. Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Feb. 2003. URL: http://www2.dbd.puc-rio.br/pergamum/biblioteca/php/mostrateses.php?open=1&arqtese=5000066121_03_Indice.html.

[Mit04] Melanie Mitchell. *An Introduction to Genetic Algorithms: Complex Adaptative Systems*. The MIT Press, 2004.

[El-+06] Tarek A. El-Mihoub et al. "Hybrid Genetic Algorithms: A Review". In: *Engineering Letters* 13.2 (Aug. 2006). URL: www.engineeringletters.com/issues_v13/issue.../EL_13_2_11.pdf.

[Coe07] Carlos A. Coello Coello. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. 2nd Edition. Springer, Sept. 2007.

[Cru07] André Abs da Cruz. "Algoritmos Evolutivos com Inspiração Quântica para Problemas com Representação Numérica". (In Portuguese). PhD thesis. Rio de Janeiro, Brasil: Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Mar. 2007.

[Wei09] Thomas Weise. *Global Optimization Algorithms - Theory and Application*. http://www.it-weise.de. 2009.

[RBK12] Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, eds. *Handbook of Natural Computing*. 1st. Springer Publishing Company, Incorporated, 2012.

[Fog95] David B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. New York: The Institute of Electrical and Electronic Engineers, 1995.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley Publishing Co., 1989.

[Koz92] John R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. Cambridge, Massachusetts: The MIT Press, 1992.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS351. Topics in Computer Graphics (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS351. Topics in Computer Graphics
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS251. Computer graphics . (7^{th} Sem) CS251. Computer graphics . (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

In this course you can delve into any of the topics Mentioned in the area of Graphics Computing (Graphics and Visual Computing - GV).

This course is designed to perform some advanced course suggested by the ACM / IEEE curriculum. [Hug+13; HB90]

5. GOALS

- That the student uses computer techniques Graphs that involve complex data structures and algorithms.
- That the student apply the concepts learned to create an application about a real problem.
- That the student investigate the possibility of creating a new algorithm and / or new technique to solve a real problem

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Advanced Topics on Computer Graphics (0)	
Competences Expected:	
Topics	**Learning Outcomes**
• CS355. Advanced Computer Graphics • CS356. Computer animation • CS313. Geometric Algorithms • CS357. visualization • CS358. Virtual reality • CS359. Genetic algorithms	• Advanced Topics on Computer Graphics
Readings : [**Soars022S**], [**Soars022W**], [**Soars022T**], [**Cambridge06**], [**MacGrew99**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[HB90] Donald Hearn and Pauline Baker. *Computer Graphics in C*. Prentice Hall, 1990.

[Hug+13] John F. Hughes et al. *Computer Graphics - Principles and Practice 3rd Edition*. Addison-Wesley, 2013.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS392. Tópicos en Ingeniería de Software (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS392. Tópicos en Ingeniería de Software
2.2 Semester	:	9^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS391. Software Engineering III. (7^{th} Sem) CS391. Software Engineering III. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

El desarrollo de software requiere del uso de mejores prácticas de desarrollo, gestión de proyectos de TI, manejo de equipos y uso eficiente y racional de frameworks de aseguramiento de la calidad y de Gobierno de Portfolios, estos elemento son pieza clave y transversal para el éxito del proceso productivo.

Este curso explora el diseño, selección, implementación y gestión de soluciones TI en las Organizaciones. El foco está en las aplicaciones y la infraestructura y su aplicación en el negocio.

5. GOALS

- Entender una variedad de frameworks para el análisis de arquitectura empresarial y la toma de decisiones
- Utilizar técnicas para la evaluación y gestión del riesgo en el portfolio de la empresa
- Evaluar y planificar la integración de tecnologías emergentes
- Entender el papel y el potencial de las TI para a apoyar la gestión de procesos empresariales
- Entender los difentes enfoques para modelar y mejorar los procesos de negocio
- Describir y comprender modelos de aseguramiento de la calidad como marco clave para el éxitos de los proyectos de TI.
- Comprender y aplicar el framework de IT Governance como elemento clave para la gestión del portfolio de aplicaciones Empresariales

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)

3) Communicate effectively in a variety of professional contexts.. (Usage)

5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Assessment)

7. TOPICS

Unit 1: Diseño de Software (18)

Competences Expected:

Topics	Learning Outcomes
• Principios de diseño del sistema: niveles de abstracción (diseño arquitectónico y el diseño detallado), separación de intereses, ocultamiento de información, de acoplamiento y de cohesión, de reutilización de estructuras estándar. • Diseño de paradigmas tales como diseño estructurado (descomposición funcional de arriba hacia abajo), el análisis orientado a objetos y diseño, orientado a eventos de diseño, diseño de nivel de componente, centrado datos estructurada, orientada a aspectos, orientado a la función, orientado al servicio. • Modelos estructurales y de comportamiento de los diseños de software. • Diseño de patrones. • Relaciones entre los requisitos y diseños: La transformación de modelos, el diseño de los contratos, invariantes. • Conceptos de arquitectura de software y arquitecturas estándar (por ejemplo, cliente-servidor, n-capas, transforman centrados, tubos y filtros). • El uso de componentes de diseño: seleccion de componentes,diseño,adaptacion y componentes de ensamblaje, componentes y patrones, componentes y objetos(por ejemplo,construir una GUI usando un standar widget set) • Calidad del diseño interno, y modelos para: eficiencia y desempeño, redundancia y tolerancia a fallos, trazavilidad de los requerimientos. • Calidad del diseño interno, y modelos para: eficiencia y desempeño, redundancia y tolerancia a fallos, trazavilidad de los requerimientos. • Medición y análisis de la calidad de un diseño. • Compensasiones entre diferentes aspectos de la calidad. • Aaplicaciones en frameworks. • Middleware: El paradigma de la orientacion a objetos con middleware, requerimientos para correr y clasificar objetos, monitores de procesamiento de transacciones y el sistema de flujo de trabajo. • Principales diseños de seguridad y codificación(cross-reference IAS/Principles of securre design). – Principio de privilegios mínimos – Principio de falla segura por defecto – Principio de aceptabilidad psicológica	• Formular los principios de diseño, incluyendo la separación de problemas, ocultación de información, acoplamiento y cohesión, y la encapsulación [Usar] • Usar un paradigma de diseño para diseñar un sistema de software básico y explicar cómo los principios de diseño del sistema se han aplicado en este diseño [Usar] • Construir modelos del diseño de un sistema de software simple los cuales son apropiado para el paradigma utilizado para diseñarlo [Usar] • En el contexto de un paradigma de diseño simple, describir uno o más patrones de diseño que podrían ser aplicables al diseño de un sistema de software simple [Usar] • Para un sistema simple adecuado para una situación dada, discutir y seleccionar un paradigma de diseño apropiado [Usar] • Crear modelos apropiados para la estructura y el comportamiento de los productos de software desde la especificaciones de requisitos [Usar] • Explicar las relaciones entre los requisitos para un producto de software y su diseño, utilizando los modelos apropiados [Usar] • Para el diseño de un sistema de software simple dentro del contexto de un único paradigma de diseño, describir la arquitectura de software de ese sistema [Usar] • Dado un diseño de alto nivel, identificar la arquitectura de software mediante la diferenciación entre las arquitecturas comunes de software, tales como 3 capas (*3-tier*), *pipe-and-filter*, y cliente-servidor [Usar] • Investigar el impacto de la selección arquitecturas de software en el diseño de un sistema simple [Usar] • Aplicar ejemplos simples de patrones en un diseño de software [Usar] • Describir una manera de refactorar y discutir cuando esto debe ser aplicado [Usar] • Seleccionar componentes adecuados para el uso en un diseño de un producto de software [Usar] • Explicar cómo los componentes deben ser adaptados para ser usados en el diseño de un producto de software [Usar] • Diseñar un contrato para un típico componente de software pequeño para el uso de un dado sistema [Usar] • Discutir y seleccionar la arquitectura de software adecuada para un sistema de software simple para un dado escenario [Usar]

Unit 2: Gestión de Proyectos de Software (14)

Competences Expected:

Topics	Learning Outcomes
• La participación del equipo: – Procesos elemento del equipo, incluyendo responsabilidades de tarea, la estructura de reuniones y horario de trabajo – Roles y responsabilidades en un equipo de software – Equipo de resolución de conflictos – Los riesgos asociados con los equipos virtuales (comunicación, la percepción, la estructura) • Estimación de esfuerzo (a nivel personal) • Riesgo. – El papel del riesgo en el ciclo de vida – Categorías elemento de riesgo, incluyendo la seguridad, la seguridad, mercado, finanzas, tecnología, las personas, la calidad, la estructura y el proceso de • Gestión de equipos: – Organización de equipo y la toma de decisiones – Roles de identificación y asignación – Individual y el desempeño del equipo de evaluación • Gestión de proyectos: – Programación y seguimiento de elementos – Herramientas de gestión de proyectos – Análisis de Costo/Beneficio • Software de medición y técnicas de estimación. • Aseguramiento de la calidad del software y el rol de las mediciones. • Riesgo. – El papel del riesgo en el ciclo de vida – Categorías elemento de riesgo, incluyendo la seguridad, la seguridad, mercado, finanzas, tecnología, las personas, la calidad, la estructura y el proceso de • En todo el sistema de aproximación al riesgo, incluyendo riesgos asociados con herramientas.	• Discutir los comportamientos comunes que contribuyen al buen funcionamiento de un equipo [Usar] • Crear y seguir un programa para una reunión del equipo [Usar] • Identificar y justificar las funciones necesarias en un equipo de desarrollo de software [Usar] • Entender las fuentes, obstáculos y beneficios potenciales de un conflicto de equipo [Usar] • Aplicar una estrategia de resolución de conflictos en un ambiente de equipo [Usar] • Utilizar un método ad hoc para estimar el esfuerzo de desarrollo del software (ejemplo, tiempo) y comparar con el esfuerzo actual requerido [Usar] • Listar varios ejemplos de los riesgos del software [Usar] • Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Usar] • Describir las diferentes categorías de riesgo en los sistemas de software [Usar] • Demostrar a través de la colaboración de proyectos de equipo los elementos centrales de la contrucción de equipos y gestión de equipos [Usar] • Describir como la elección de modelos de procesos afectan la estructura organizacional de equipos y procesos de toma de decisiones [Usar] • Crear un equipo mediante la identificación de los roles apropiados y la asignación de funciones a los miembros del equipo [Usar] • Evaluar y retroalimentar a los equipos e individuos sobre su desempeño en un ambiente de equipo [Usar] • Usando un software particular procesar, describir los aspectos de un proyecto que encesita ser planeado y monitoreado, (ejemplo, estimar el tamaño y esfuerzo, un horario, reasignación de recursos, control de configuración, gestión de cambios, identificación de riesgos en un proyecto y gestión) [Usar] • Realizar el seguimiento del progreso de alguna etapa de un proyecto que utiliza métricas de proyectos apropiados [Usar] • Comparar las técnicas simples de tamaño de software y estimación de costos [Usar] • Usar una herramienta de gestión de proyectos para ayudar en la asignación y rastreo de tareas en un proyecto de desarrollo de software [Usar] • Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Usar]

Unit 3: (14)	
Competences Expected:	
Topics	**Learning Outcomes**
• Administración del servicio como práctica. • Ciclo de vida del servicio. • Definiciones y conceptos genéricos. • Modelos y principios claves. • Procesos. • Tecnología y arquitectura. • Competencia y entrenamiento.	• Utilizar y aplicar correctamente ITIL en el proceso de software. [Usar]
Readings : [Som17], [PM15]	

Unit 4: (14)	
Competences Expected:	
Topics	**Learning Outcomes**
• Fundamentos e Introducción. • Frameworks de Control y IT Governance.	• Utilizar y aplicar correctamente COBIT en el proceso de software. [Usar]
Readings : [Som17], [PM15]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[PM15] Roger S. Pressman and Bruce Maxim. *Software Engineering: A Practitioner's Approach*. 8th. McGraw-Hill, Jan. 2015.

[Som17] Ian Sommerville. *Software Engineering*. 10th. Pearson, Mar. 2017.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS365. Evolutionary Computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS365. Evolutionary Computing
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS262. Machine learning. (7^{th} Sem) CS262. Machine learning. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Write justification for this course here ...

5. GOALS

- Write your first goal here.
- Write your second goal here.
- Just in case you need more goals write them here

6. COMPETENCES

a) An ability to apply knowledge of mathematics, science. (Familiarity)

7. TOPICS

Unit 1: title for the unit goes here (5)	
Competences Expected: a	
Topics	**Learning Outcomes**
• Topic1 • Topic2 • Topic3	• Learning outcome1 [Levelforthislearningoutcome]. • Apply computing in complex problems [Usar]. • Create a search engine [Evaluar]. • Study data structures [Familiarizarse].
Readings : [**Bibitem1**], [**Bibitem2**]	

Unit 2: another unit goes here (1)	
Competences Expected:	
Topics	**Learning Outcomes**
• Topic1	• Learning outcome xyz [Levelforthislearningoutcome].
Readings : [**Bibitem3**], [**Bibitem1**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE
CS3P2. Cloud Computing (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3P2. Cloud Computing
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS370. Big Data. (9^{th} Sem) CS370. Big Data. (9^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE
To understand advanced computational techniques, students must have a strong knowledge of various discrete structures, structures that will be implemented and used in the laboratory with the programming language.

5. GOALS

- Students will be able to model computer science problems using graphs and trees related to data structures.
- Students will efficiently apply traversal strategies to search for data optimally.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Theoretical Foundations of Cloud Computing (12)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Introduction to Cloud Computing • Cloud Computing Service Models • Cloud Computing Deployment Models • Infrastructure and Data Centers • Research Trends in Cloud Computing	• Understand the concepts related to Cloud Computing. • Understand the infrastructure and components of a Data Center. • Understand service models and deployment types in Cloud Computing. • Be familiar with research trends in the area of Cloud Computing.
Readings : [**aboveTheCloud**], [**surveySecurity**], [**mobileCloud**]	

Unit 2: Data Processing (15)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Introduction to the Hadoop framework. • Hadoop Distributed File System. • Introduction to the MapReduce programming model. • Introduction to the Spark framework.	• Understand the concepts related to the Hadoop framework. • Understand the concepts related to the Hadoop Distributed File System. • Understand and apply the MapReduce programming model. • Understand the concepts related to the Spark framework.
Readings : [**mapreduce**], [**spark**], [**yarn**]	

Unit 3: Virtualization, Containerization (15)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Introduction to Containerization. • Evolution of Containerization. • Differences between Containerization and Virtualization.	• Understand the concept of Containerization. • Create and use containers. • Understand the differences between Containerization and Virtualization.
Readings : [**CborgOmegaKubernetes**], [**borg**], [**ContainerizationPaaSCloud**], [**VirtualizationContainerization**]	

Unit 4: Trends in Cloud Computing (12)	
Competences Expected: 1,6	
Topics	**Learning Outcomes**
• Autoscaling. • Infrastructure as Code. • Serverless Computing.	• Understand different forms of autoscaling. • Use different tools for Infrastructure as Code in the cloud. • Understand the Serverless Computing paradigm.
Readings : [**Cormen2009**], [**Preparata**], [**Berg**]	

Unit 5: Distributed Systems (15)	
Competences Expected:	
Topics	**Learning Outcomes**
• Distributed System Faults • Distributed Algorithms • Distributed System Architectures • Distributed Services • Core Distributed System Concepts	• Distinguish between different types of distributed system faults [Familiarizarse] • Explain the challenges of distributed systems [Familiarizarse] • Write distributed algorithms [Usar] • Measure the performance of distributed systems [Usar] • Explain the rationale behind different distributed system designs [Familiarizarse] • Implement a distributed system [Usar] • Explain the trade-offs in distributed system design [Familiarizarse] • Describe different distributed system architectures [Familiarizarse] • Give examples of distributed systems [Usar]
Readings : [Cou+11]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Cou+11] George Coulouris et al. *Distributed Systems: Concepts and Design*. 5th. USA: Addison-Wesley Publishing Company, 2011.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

CS3P3. Internet of Things (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS3P3. Internet of Things
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	1 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS3P1. Parallel and Distributed Computing . (8^{th} Sem) CS3P1. Parallel and Distributed Computing . (8^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The last decade has an explosive growth in multiprocessor computing, including multi-core processors and distributed data centers. As a result, parallel and distributed computing has evolved from a broadly elective subject to be one of the major components in mesh studies in undergraduate computer science. Both parallel computing and distribution involve the simultaneous execution of multiple processes on different devices that change position.

5. GOALS

- That the student is able to create parallel applications of medium complexity by efficiently taking advantage of different mobile devices.

6. COMPETENCES

1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TOPICS

Unit 1: Fundamentos de paralelismo (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Procesamiento Simultáneo Múltiple. • Metas del Paralelismo (ej. rendimineto) frente a Concurrencia (ej. control de acceso a recursos compartidos) • Paralelismo, comunicación, y coordinación: – Paralelismo, comunicación, y coordinación – Necedidad de Sincronización • Errores de Programación ausentes en programación secuencial: – Tipos de Datos (lectura/escritura simultánea o escritura/escritura compartida) – Tipos de Nivél más alto (interleavings violating program intention, no determinismo no deseado) – Falta de vida/progreso (deadlock, starvation)	• Distinguir el uso de recursos computacionales para una respuesta mas rápida para administrar el acceso eficiente a un recurso compartido [Familiarizarse] • Distinguir múltiples estructuras de programación suficientes para la sincronización que pueden ser interimplementables pero tienen ventajas complementarias [Familiarizarse] • Distinguir datos de carrera (*data races*) a partir de carreras de mas alto nivel [Familiarizarse]
Readings : [Pac11], [Mat14], [Qui03]	

Unit 2: Arquitecturas paralelas (12)	
Competences Expected:	
Topics	**Learning Outcomes**

• Procesadores mutlinúcleo.
• Memoria compartida vs memoria distribuida.
• Multiprocesamiento simétrico.
• SIMD, procesamiento de vectores.
• GPU, coprocesamiento.
• Taxonomia de Flynn.
• Soporte a nivel de instrucciones para programación paralela.
  – Instrucciones atómicas como Compare/Set (Comparar / Establecer)
• Problemas de Memoria:
  – Caches multiprocesador y coherencia de cache
  – Acceso a Memoria no uniforme (NUMA)
• Topologías.
  – Interconecciones
  – Clusters
  – Compartir recursos (p.e., buses e interconexiones)
	• Explicar las diferencias entre memoria distribuida y memoria compartida [Evaluar]
	• Describir la arquitectura SMP y observar sus principales caracteristicas [Evaluar]
	• Distinguir los tipos de tareas que son adecuadas para máquinas SIMD [Usar]
	• Describir las ventajas y limitaciones de GPUs vs CPUs [Usar]
	• Explicar las caracteristicas de cada clasificación en la taxonomía de Flynn [Usar]
	• Describir los desafíos para mantener la coherencia de la caché [Familiarizarse]
	• Describir los desafíos para mantener la coherencia de la caché [Familiarizarse]

Readings : [Pac11], [KH13], [SK10]	

Unit 3: Descomposición en paralelo (18)	
Competences Expected:	
Topics	**Learning Outcomes**

• Necesidad de Comunicación y coordinación/sincronización.
• Independencia y Particionamiento.
• Conocimiento Básico del Concepto de Descomposición Paralela.
• Decomposición basada en tareas:
  – Implementación de estrategias como hebras
• Descomposición de Información Paralela
  – Estrategias como SIMD y MapReduce
• Actores y Procesos Reactivos (solicitud de gestores)
	• Explicar por qué la sincronización es necesaria en un programa paralelo especifico [Usar]
	• Identificar oportunidades para particionar un programa serial en módulos paralelos independientes [Familiarizarse]
	• Escribir un algoritmo paralelo correcto y escalable [Usar]
	• Paralelizar un algoritmo mediante la aplicación de descomposición basada en tareas [Usar]
	• Paralelizar un algoritmo mediante la aplicación de descomposición datos en paralelo [Usar]
	• Escribir un programa usando actores y/o procesos reactivos [Usar]

Readings : [Pac11], [Mat14], [Qui03]	

Unit 4: Comunicación y coordinación (18)

Competences Expected:

Topics	**Learning Outcomes**
• Memoria Compartida. • La consistencia, y su papel en los lenguaje de programación garantias para los programas de carrera libre. • Pasos de Mensaje: – Mensajes Punto a Punto versus multicast (o basados en eventos) – Estilos para enviar y recibir mensajes Blocking vs non-blocking – Buffering de mensajes • Atomicidad: – Especificar y probar atomicidad y requerimientos de seguridad – Granularidad de accesos atómicos y actualizaciones, y uso de estructuras como secciones críticas o transacciones para describirlas – Exclusión mutua usando bloques, semáforos, monitores o estructuras relacionadas ∗ Potencial para fallas y bloqueos (*deadlock*) (causas, condiciones, prevención) – Composición ∗ Componiendo acciones atómicas granulares más grandes usando sincronización ∗ Transacciones, incluyendo enfoques optimistas y conservadores • Consensos: – (Ciclicos) barerras, contadores y estructuras relacionadas • Acciones condicionales: – Espera condicional (p.e., empleando variables de condición)	• Usar exclusión mútua para evitar una condición de carrera [Usar] • Dar un ejemplo de una ordenación de accesos entre actividades concurrentes (por ejemplo, un programa con condición de carrera) que no son secuencialmente consistentes [Familiarizarse] • Dar un ejemplo de un escenario en el que el bloqueo de mensajes enviados pueden dar *deadlock* [Usar] • Explicar cuándo y por qué mensajes de multidifusión (*multicast*) o basado en eventos puede ser preferible a otras alternativas [Familiarizarse] • Escribir un programa que termine correctamente cuando todo el conjunto de procesos concurrentes hayan sido completados [Usar] • Dar un ejemplo de una ordenación de accesos entre actividades concurrentes (por ejemplo, un programa con condición de carrera) que no son secuencialmente consistentes [Familiarizarse] • Usar semaforos o variables de condición para bloquear hebras hasta una necesaria precondición de mantenga [Usar]
Readings : [Pac11], [Mat14], [Qui03]	

Unit 5: Análisis y programación de algoritmos paralelos (18)

Competences Expected:

Topics	**Learning Outcomes**
• Caminos críticos, el trabajo y la duración y la relación con la ley de Amdahl. • Aceleración y escalabilidad. • Naturalmente (vergonzosamente) algoritmos paralelos. • Patrones Algoritmicos paralelos (divide-y-conquista, map/reduce, amos-trabajadores, otros) – Algortimos específicos (p.e., MergeSort paralelo) • Algoritmos de grafos paralelo (por ejemplo, la ruta más corta en paralelo, árbol de expansión paralela) • Cálculos de matriz paralelas. • Productor-consumidor y algoritmos paralelos segmentados. • Ejemplos de algoritmos paralelos no-escalables.	• Definir: camino crítico, trabajo y *span* [Familiarizarse] • Calcular el trabajo y el *span* y determinar el camino crítico con respecto a un diagrama de ejecución paralela. [Usar] • Definir *speed-up* y explicar la noción de escalabilidad de un algoritmo en este sentido [Familiarizarse] • Identificar tareas independientes en un programa que debe ser paralelizado [Usar] • Representar características de una carga de trabajo que permita o evite que sea naturalmente paralelizable [Familiarizarse] • Implementar un algoritmo dividir y conquistar paralelo (y/o algoritmo de un grafo) y medir empiricamente su desempeño relativo a su analogo secuencial [Usar] • Descomponer un problema (por ejemplo, contar el número de ocurrencias de una palabra en un documento) via operaciones *map* y *reduce* [Usar] • Proporcionar un ejemplo de un problema que se corresponda con el paradigma productor-consumidor [Usar] • Dar ejemplos de problemas donde el uso de *pipelining* sería un medio eficaz para la paralelización [Usar] • Implementar un algoritmo de matriz paralela [Usar] • Identificar los problemas que surgen en los algoritmos del tipo productor-consumidor y los mecanismos que pueden utilizarse para superar dichos problemas [Usar]
Readings : [Mat14], [Qui03]	

Unit 6: Desempeño en paralelo (18)	
Competences Expected:	
Topics	**Learning Outcomes**
• Equilibrio de carga. • La medición del desempeño. • Programación y contención. • Evaluación de la comunicación de arriba. • Gestión de datos: – Costos de comunicación no uniforme debidos a proximidad – Efectos de Cache (p.e., false sharing) – Manteniendo localidad espacial • Consumo de energía y gestión.	• Detectar y corregir un desbalanceo de carga [Usar] • Calcular las implicaciones de la ley de Amdahl para un algoritmo paralelo particular [Usar] • Describir como la distribuición/disposición de datos puede afectar a los costos de comunicación de un algoritmo [Familiarizarse] • Detectar y corregir una instancia de uso compartido falso (*false sharing*) [Usar] • Explicar el impacto de la planificación en el desempeño paralelo [Familiarizarse] • Explicar el impacto en el desempeño de la localidad de datos [Familiarizarse] • Explicar el impacto y los puntos de equilibrio relacionados al uso de energía en el desempeño paralelo [Familiarizarse]
Readings : [Pac11], [Mat14], [KH13], [SK10]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Qui03] Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. 1st. McGraw-Hill Education Group, 2003.

[SK10] Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. 1st. Addison-Wesley Professional, 2010.

[Pac11] Peter S. Pacheco. *An Introduction to Parallel Programming*. 1st. Morgan Kaufmann, 2011.

[KH13] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013.

[Mat14] Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014. URL: `http://heather.cs.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf`.

San Cristobal of Huamanga National University (UNSCH)
School of Computer Science
Syllabus 2024-II

1. COURSE

FG211. Professional Ethics (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	FG211. Professional Ethics
2.2 Semester	:	10^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

La ética es una parte constitutiva inherente al ser humano, y como tal debe plasmarse en el actuar cotidiano y profesional de la persona humana. Es indispensable que la persona asuma su rol activo en la sociedad pues los sistemas económico-industrial, político y social no siempre están en función de valores y principios, siendo éstos en realidad los pilares sobre los que debería basarse todo el actuar de los profesionales.

5. GOALS

- Que el alumno amplíe sus propios criterios personales de discernimiento moral en el quehacer profesional, de forma que no sólo tome en cuenta los criterios técnicos pertinentes sino que incorpore a sí mismo cuestionamientos de orden moral y se adhiera a una ética profesional correcta, de forma que sea capaz de aportar positivamente en el desarrollo económico y social de la ciudad, región, país y comunidad global.[Usar]

6. COMPETENCES

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Usage)

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (Usage)

7. TOPICS

Unit 1: (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Ser profesional y ser moral. • La objetividad moral y la formulación de principios morales. • El profesional y sus valores. • La conciencia moral de la persona. • El aporte de la DSI en el quehacer profesional. • El bien común y el principio de subsidiaridad. • Principios morales y propiedad privada. • Justicia: Algunos conceptos básicos.	• Presentar al alumno la importancia de tener principios y valores en la sociedad actual.[Usar] • Presentar algunos de los principios de podrían contribuir en la sociedad de ser aplicados y vividos día a día. [Usar] • Presentar a los alumnos el aporte de la Doctrina Social de la Iglesia en el quehacer profesional. [Usar]
Readings : [Com92], [Sch95], [Loz00], [Arg06]	

Unit 2: (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• La responsabilidad individual del trabajador en la empresa. • Liderazgo y ética profesional en el entorno laboral. • Principios generales sobre la colaboración en hechos inmorales. • El profesional frente al soborno: ¿víctima o colaboración?	• Presentar al alumno el rol de la responsabilidad social individual y del liderazgo en la empresa. [Familiarizarse] • Conocer el juicio de la ética frente a la corrupción y sobornos como forma de relación laboral. [Familiarizarse] • Presentar la profesión como una forma de realización personal, y como consecuencia. []
Readings : [Com92], [Man07], [Sch95], [Pér98], [Nie03]	

Unit 3: (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• La ética profesional frente a la ética general. • Trabajo y profesión en los tiempos actuales. • Ética, ciencia y tecnología. • Valores éticos en organizaciones relacionadas con el uso de la información. • Valores éticos en la era de la Sociedad de la Información.	• Presentar al alumno las interrelaciones entre ética y las disciplinas de la última era tecnológica.[Familiarizarse]
Readings : [Com92], [IEE04], [Her06]	

Unit 4: (12)	
Competences Expected:	
Topics	**Learning Outcomes**
• Ética informática. – Ética y software. – El software libre. • Regulación y ética de telecomunicaciones. – Ética en Internet. • Derechos de autor y patentes. • Ética en los servicios de consultoría. • Ética en los procesos de innovación tecnológica. • Ética en la gestión tecnológica y en empresas de base tecnológica.	• Presentar al alumno algunos aspectos que confrontan la ética con el quehacer de las disciplinas emergentes en la sociedad de la información.[Familiarizarse]
Readings : [Com02], [Her06], [Com92]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[Com92] Association for Computing Machinery (ACM). "ACM Code of Ethics and Professional Conduct". In: (1992). URL: `http://www.acm.org/about/code-of-ethics`.

[Sch95] E. Schmidt. *Ética y Negocios para América Latina*. Universidad del Pacífico, 1995.

[Pér98] J. A. Pérez López. *Liderazgo y Ética en la Dirección de Empresas*. Bilbao, 1998.

[Loz00] C Loza. "El aporte de la Doctrina Social de la Iglesia a la Toma de Decisiones Empresariales". In: *Separata ofrecida por el profesor* (2000).

[Com02] Pontificio Consejo para las Comunicaciones Sociales. *Ética en Internet*. 2002.

[Nie03] R. Nieburh. *El Yo Responsable. Ensayo de Filosofía Moral Cristiana*. Bilbao, 2003.

[IEE04] IEEE. "IEEE Code of Ethics". In: *IEE* (2004). URL: `http://www.ieee.org/about/corporate/governance/p7-8.html`.

[Arg06] Argandoña. "La identidad Cristiana del Directivo de Empresa". In: *IESE* (2006).

[Her06] A. Hernández. *Ética Actual y Profesional. Lecturas para la Convivencia Global en el Siglo XXI*. Ed. Thomson, 2006.

[Man07] G. Manzone. *La Responsabilidad de la Empresa, Business Ethics y Doctrina Social de la Iglesia en Diálogo*. Universidad Católica San Pablo, 2007.