
# Facultad de Ingenierías
# Escuela Profesional de Ciencia de la Computación
# Plan Curricular 2026
# https://cs.unam.edu.pe

*– Reporte Final–*

**Última modificación: Agosto de 2025**

Este documento está basado en la propuesta internacional de *ACM/IEEE-CS Computing Curricula*, que a su vez sirvió como base de la versión en Español creada por la Sociedad Peruana de Computación (SPC)

ISBN:

# Equipo de Trabajo

**Dr. Anibal F. Flores Garcia**
Presidente de la Comisión de Nuevos Programas DAISI
Director del Departamento Académico de Ingeniería de Sistemas e Informática
email: *afloresg@unam.edu.pe*
**M.Sc. Honorio Apaza Alanoca**
Miembro de la Comisión de Nuevos Programas DAISI
email: *hapazaa@unam.edu.pe*
**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*

# Resumen ejecutivo

Este documento representa el informe final de la nueva malla curricular 2026 del Escuela Profesional de Ciencia de la Computación de la Universidad Nacional de Moquegua (UNAM) (*https://www.unam.edu.pe*) en la ciudad de Moquegua-Perú.

Este documento presenta el diseño curricular de la carrera de Ciencia de la Computación bajo los estándares internacionales del *Computing Curricula* CC2020, marco de referencia desarrollado conjuntamente por la *Association for Computing Machinery* (ACM) y la *IEEE Computer Society* (IEEE-CS). El CC2020, disponible en http://www.acm.org/education, sintetiza tres décadas de evolución disciplinar (desde CS2001 hasta CC2020) y responde a las demandas actuales de la cuarta revolución industrial.

La propuesta adapta este referente global a nuestro contexto regional, incorporando:

- Requerimientos del mercado laboral local en tecnología
- Fortalezas institucionales en investigación aplicada
- Tendencias emergentes en educación computacional

La computación hoy en día presenta los siguientes perfiles de formación profesional para pregrado:

- Ingeniería de Computación (*Computer Engineering* – CE),
- Ciencia de la Computación (*Computer Science* – CS),
- Sistemas de Información (*Information Systems* – IS),
- Ingeniería de Software (*Software Engineering* – SE),
- Tecnología de la Información (*Information Technology* – IT),
- Ciberseguridad (*Cybersecurity* – CY) y
- Ciencia de Datos (*Data Science* – DS).

El modelo pedagógico se sustenta en tres ejes transformadores:

- **Excelencia técnica**: Dominio de fundamentos computacionales y especialización progresiva
- **Innovación disruptiva**: Metodologías ágiles, pensamiento computacional y creación de startups
- **Responsabilidad social**: Ética profesional, sostenibilidad tecnológica e impacto comunitario

Esta malla producirá profesionales con:

- Capacidad para resolver problemas complejos en entornos globales
- Competencias para liderar la transformación digital
- Sensibilidad hacia los desafíos éticos de la inteligencia artificial, privacidad y gobernanza tecnológica

Esta propuesta incorpora mecanismos de actualización continua, asegurando relevancia frente a la acelerada obsolescencia tecnológica y alineamiento con los estándares de acreditación internacional.

# Índice general

# Índice de figuras

# Índice de cuadros

# Agradecimientos

Además de los autores directos de este documento, también deseamos dejar manifiesto de nuestro agradecimiento a otros colegas de diversas universidades del país y del mundo que gentilmente han aportado parte de su tiempo a darnos sus sugerencias. Entre ellos debemos mencionar a:

- Todo de equipo de ACM/IEEE-CS Computing Curricula (CC2020, CS2023 y CS2013).
- Grace Alvarado por su valiosa contribución en la corrección de todos los detalles de cada curso en todo el currículo
- José Carreon (NCRPTD) por sus valiosas contribuciones en la línea de seguridad de la información
- Luis Diaz Basurco, Wilber Ramos por su valiosa ayuda en la línea de matemáticas
- Katia Cánepa (University of California, Davis, USA) por su contribución en general en todo el currículo
- Alex Cuadros Vargas (UCSP, Perú) por su contribución en la línea de Computación Gráfica y en todo el currículo en general
- Yessenia Yari (UCSP, Perú) por sus observaciones y ayuda en corregir diversos aspectos en toda la propuesta curricular
- Yván Túpac (UCSP, Perú) por sus aportes en la linea de bioinformática y computación molecular biológica
- Yamilet Serrano (UTEC, Perú) por su contribución en la línea de Ingeniería de Software y en todo el currículo en general
- Jesús Bellido (UTEC, Perú) por su contribución en la línea de Ingeniería de Software y en todo el currículo en general
- Juan Carlos Gutiérrez (UNSA, UCSP, Perú) por su contribución en la línea de Inteligencia Artificial
- Pedro Shiguihara (USIL, Perú) por su contribución en la línea de Inteligencia Artificial
- Javier Quinto Ancieta (Whitestack, USA) por su contribución en la línea de redes y comunicaciones
- Heider Sánchez y Teófilo Chambilla (UTEC, Perú) por su contribución en la línea de Base de Datos
- José Miguel Renom por su ayuda en los cursos de estadística
- Arturo Salazar por su ayuda en los cursos rellacionados a habilidades blandas.
- Carlos Arias por su contribución en la linea de Experiencia de Usuario (UX)
- Jaime Farfán por su contribución en la linea de *Cloud Computing*
- César Beltrán Castañón (PUCP Perú) por su invaluable ayuda en la elaboración de la malla propuesta
- Manuel Rodríguez Canales y Javier Rodríguez Canales por sus valiosos aportes en la línea ética
- Christian Jorge Delgado Polar (IME-USP-Brasil, UCSP-Perú)
- Ruben Acosta (INICTEL-UNI) por su valiosa contribución en el área de Internet de las cosas (*Internet of Things*)
- Enrique Solano (Kipu Quantum GmbH, Alemania) por sus valiosos aportes en cuanto a Computación Cuántica.
- Cuerpo docente de: CS-UCSP, CS-UNSA, CS-UTEC, CS-UNI.

Todo este equipo de trabajo asumió como premisa que el centro de nuestro esfuerzo, es la formación académica y humana de los estudiantes para formar agentes de cambio positivo y disruptivo en la sociedad.

A todos ellos deseamos agradecerles por su aporte que ha permitido generar este documento, único

en su género en nuestro país, que servirá para sentar las bases de una carrera más sólida en esta fantástica área que nos ha tocado estudiar y de la cual nos sentimos orgullosos de formar parte: **Computación**.

# Abreviaturas

**ACM** *Association for Computing Machinery*

**AIS** *Association for Information Systems*

**CC** Ciencia de la Computación

**CS** Ciencia de la Computación – *Computer Science*

**IEEE-CS** IEEE *Computer Society*

**KA** área de Conocimiento (*Knowledge Area*-KA)

.

# Capítulo 1

# Introducción

La computación ha sufrido un desarrollo impresionante en las últimas décadas, convirtiéndose en el motor del desarrollo científico, tecnológico, industrial, social, económico y cultural, transformando de manera significativa nuestro diario accionar.

El surgimiento del computador ha marcado una nueva era en la historia de la humanidad que era imposible de imaginar varias décadas atrás. La gran cantidad de aplicaciones que se han desarrollado en los últimos años están transformando el desarrollo de todas las disciplinas del saber, la comercialización en el ámbito globalizado en que vivimos, la manera en que nos comunicamos, los procesos de enseñanza-aprendizaje y hasta en la manera como nos entretenemos.

Para darnos una idea de la relevancia e importancia, que en nuestro país ha alcanzado esta disciplina, basta mencionar que actualmente se ofrecen aproximadamente más de 100 carreras de Computación a nivel nacional. Esto sin considerar los programas de nivel Técnico Superior No Universitario que se ofertan.

Todas estas carreras existentes tienen como centro de su estudio a la computación pero lo hacen con más de 50 nombres distintos como: Ingeniería de Sistemas, Ingeniería de Computación, Ingeniería de Computación y Sistemas, entre otros. A pesar de que todas ellas apuntan al mismo mercado de trabajo, resulta por lo menos sorprendente que no sea posible encontrar por lo menos dos que compartan la misma curricula aunque sea en el primer año.

Comúnmente, durante la década de los setenta, la Computación se desarrolló dentro de las Facultades de Ciencias en la mayoría de las universidades estadounidenses, británicas y de otros países. Durante la década de los ochenta, los grupos de computación en las universidades se esforzaron por lograr una legitimidad académica en su ámbito local. Frecuentemente, se transformaron en departamentos de Matemáticas y Computación, hasta finalmente dividirse en dos departamentos de Matemáticas y de Computación, en la década de los noventa. Es en esta década en que un número creciente de instituciones reconocieron la influencia penetrante de la Computación, creando unidades independientes como departamentos, escuelas o institutos dedicados a tal área de estudio, un cambio que ha demostrado tanto perspicacia como previsión.

En Perú, un número cada vez mayor de instituciones de educación superior han tratado de seguir el desarrollo de las universidades extranjeras (aunque no siempre en forma muy seria o exitosa), reconociendo a la Computación como un área de estudio en sí misma, así como su importancia estratégica en la educación, y creando departamentos, escuelas o institutos dedicados a su estudio. La Facultad de Ingenierías no puede ser la excepción a este cambio, en el que ya se tiene un retraso relativo con muchas de las instituciones educativas dentro y fuera de Perú.

## 1.1. Definiciones básicas

La referencia más sólida a nivel mundial en cuanto a la propuesta de carreras de computación para nivel de pregrado es la que fue propuesta en conjunto por la *Association for Computing Machinery* (ACM), IEEE *Computer Society* (IEEE-CS) y la *Association for Information Systems* (AIS). Estas tres organizaciones propusieron la Computing Curricula en el documento denominado: *Joint Task Force for Computing Curricula 2005, Computing Curricula 2005. Overview Report*[Shakelford et al., 2005] cuya version mas reciente es la denominada *Computing Curricula 2020*[ACM/IEEE-CS, 2020].

La Ciencia de la Computación es un término de origen estadounidense Ciencia de la Computación – *Computer Science* (CS). Este término es conocido también como informática en el ámbito europeo[1].

Según el diccionario de la Real Academia de la Lengua Española (http://www.rae.es) ambos términos también son sinónimos.

A nivel internacional, la computación presenta siete perfiles claramente definidos:

- Ciencia de la Computación (*Computer Science*) [ACM/IEEE-CS/AAAI Joint Task Force, 2023],
- Ingeniería de Computación (*Computer Engineering*) [Soldan et al., 2016],
- Sistemas de Información (*Information Systems*) [Force, 2020],
- Ingeniería de Software (*Software Engineering*) [Díaz-Herrera and Hilburn, 2004],
- Tecnología de la Información (*Information Technology*) [ACM and IEEE-CS, 2017],
- Ciencia de Datos (*Data Science*) [ACM Data Science Task Force, 2021],
- Ciberseguridad (*Cybersecurity*) [ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017].

La Figura 1.1 es tomada de la definición propuesta en la *Computing Curricula*[Shakelford et al., 2005] en el área de Ciencia de la Computación (CC) La CC cubre la mayor parte entre el extremo superior y el extremo inferior, porque el profesional en CC no trata "solamente con el hardware" que utiliza un software o de "solamente la organización" que hace uso de la información que la computación le puede proveer.

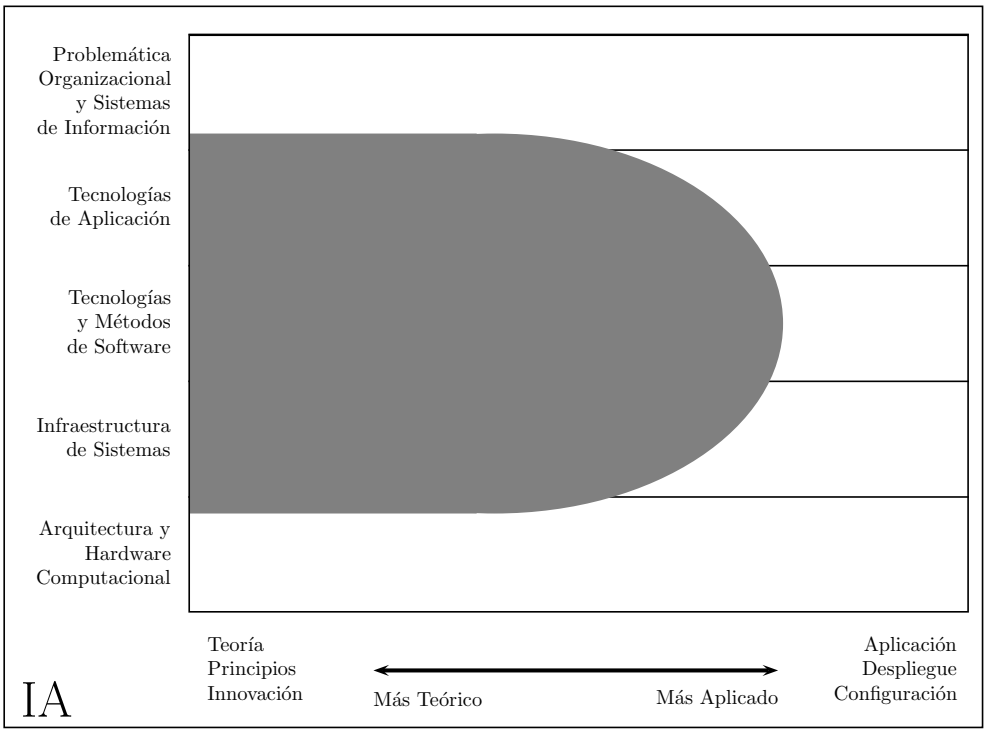



Figura 1.1: Campo acción de la Ciencia de la Computación

Ciencia de la Computación cubren un amplio rango, desde sus fundamentos teóricos y algorítmicos hasta los últimos desarrollos en robótica, visión por computadora, Inteligencia Artificial, bioinformática, y otras áreas emocionantes. Podemos pensar que el trabajo de un científico de la computación pertenece a las siguientes tres categorías:

- **Diseño e implementación de software**. Los científicos de computación se encargan de desafiantes labores de programación. También supervisan otros programadores, haciéndolos concientes de nuevas aproximaciones.
- **Instrumentación de nuevas formas para usar computadoras**. El progreso en las áreas de ciencias de la computación como redes, bases de datos, e interfaces humano-computadora permitieron el desarrollo de la www y actualmente se trabaja en el desarrollo de metasistemas Grid. Además, los investigadores trabajan ahora en hacer que los robots sean ayudantes prácticos y demuestren

[1] El término europeo es derivado del vocablo francés *Informatique*.

inteligencia, utilizan las bases de datos para crear nuevos conocimientos, y están utilizando computadoras para decifrar los secretos de nuestro ADN.
- **Desarrollo de formas efectivas de resolver problemas de computación.** Por ejemplo, los científicos de la computación desarrollan las mejores formas posibles de almacenar información en bases de datos, enviar datos a través de la red, y desplegar imágenes complejas. Sus bases teóricas les permiten determinar el mejor desempeño posible, y su estudio de algoritmos les ayuda a desarrollar nuevas aproximaciones para proveer un mejor desempeño.

La Ciencia de la Computación cubren todo el rango desde la teoría hasta la programación. Mientras otras disciplinas pueden producir titulados mejor preparados para trabajos específicos, las ciencias de la computación ofrecen un amplio fundamento que permite a sus titulados adaptarse a nuevas tecnologías y nuevas ideas.

El profesional en CC se preocupa por casi todo en medio de estas áreas. En dirección hacia el hardware, este profesional llega a desarrollar software que permite el funcionamiento de dispositivos *devices*. En dirección a aspectos organizacionales, el profesional de CC ayuda a que los sistemas de información operen correctamente en las organizaciones. Él genera la tecnología que permite que otras áreas como los sistemas de información se desarrollen adecuadamente.

El profesional en CC diseña y desarrolla todo tipo de software, desde infraestructura de plataformas (sistemas operativos, programas de comunicación, etc.) hasta aplicación de tecnologías (navegadores de Internet, bases de datos, motores de búsqueda, etc.). Este profesional crea estas capacidades, pero no está orientado al uso de las mismas. Por lo tanto, el área sombreada (fig. 1.1) para CC se estrecha y finaliza en la medida que nos movamos hacia la aplicación y configuración de productos.

## 1.2. Campo y mercado ocupacional

Nuestro egresado podrá prestar sus servicios profesionales en empresas e instituciones públicas y privadas que requieran sus capacidades en función del desarrollo que oferta, entre ellas:

- Empresas dedicadas a la producción de software con calidad internacional.
- Empresas, instituciones y organizaciones que requieran software de calidad para mejorar sus actividades y/o servicios ofertados.

Nuestro egresado puede desempeñarse en el mercado laboral sin ningún problema ya que, en general, la exigencia del mercado y campo ocupacional está mucho más orientada al uso de herramientas. Sin embargo, es poco común que los propios profesionales de esta carrera se pregunten: ¿qué tipo de formación debería tener si yo quisiera crear esas herramientas además de saber usarlas?. Ambos perfiles (usuario y creador) son bastante diferentes pues no sería posible usar algo que todavía no fue creado. En otras palabras, los creadores de tecnología son los que dan origen a nuevos puestos de trabajo y abren la posibilidad de que otros puedan usar esa tecnología.

Debido a la formación basada en la investigación, nuestro profesional debe siempre ser un innovador donde trabaje. Esta misma formación permite que el egresado piense también en crear su propia empresa de desarrollo de software. Considerando que países en vias de desarrollo tienen un costo de vida mucho menor que Norte América ó Europa, una posibilidad que se muestra interesante es la exportación de software pero eso requiere que la calidad del producto sea al mismo nivel de lo ofrecido a nivel internacional.

Este perfil profesional también posibilita que los egresados se queden en el país; producir software en nuestro país y venderlo fuera es más rentable que salir al extranjero y comercializarlo allá.

El campo ocupacional de un egresado es amplio y está en continua expansión y cambio. Prácticamente toda empresa u organización hace uso de servicios de computación de algún tipo, y la buena formación básica de nuestros egresados hace que puedan responder a los requerimientos de las mismas exitosamente. Este egresado, no sólo podrá dar soluciones a los problemas existentes sino que deberá proponer innovaciones tecnológicas que impulsen la empresa hacia un progreso constante.

A medida que la informatización básica de las empresas del país avanza, la necesidad de personas capacitadas para resolver los problemas de mayor complejidad aumenta y el plan de estudios que hemos desarrollado tiene como objetivo satisfacer esta demanda considerandola a mediano y largo plazo. El campo para las tareas de investigación y desarrollo de problemas complejos en computación es también muy amplio y está creciendo día a día a nivel mundial.

Debido a la capacidad innovadora de nuestro egresado, existe una mayor la probabilidad de registrar patentes con un alto nivel inventivo lo cual es especialmente importante en nuestros países.

## 1.3. Importancia de la carrera en la sociedad

Uno de los caminos que se espera que siga un profesional del área de Ciencia de la Computación es que el se dedique a producir software o que se integre a las empresas productoras de software. En el ámbito de la computación, es común observar que los países cuentas con Asociaciones de Productores de Software cuyas políticas están orientadas a la exportación. Siendo así, no tendría sentido preparar a nuestros alumnos sólo para el mercado local o nacional. Nuestros egresados deben estar preparados para desenvolverse en el mundo globalizado que nos ha tocado vivir.

Nuestros futuros profesionales deben estar orientados a crear nuevas empresas de base tecnológica que puedan incrementar las exportaciones de software peruano. Este nuevo perfil está orientado a generar industria innovadora. Si nosotros somos capaces de exportar software competitivo también estaremos en condiciones de atraer nuevas inversiones. Las nuevas inversiones generarían más puestos de empleo bien remunerados y con un costo bajo en relación a otros tipos de industria. Bajo esta perspectiva, podemos afirmar que esta carrera será un motor que impulsará al desarrollo del país de forma decisiva con una inversión muy baja en relación a otros campos.

Es necesario recordar que la mayor innovación de productos comerciales de versiones recientes utiliza tecnología que se conocía en el mundo académico hace varios años, décadas o más. Un ejemplo claro son las bases de datos que soportan datos y consultas espaciales desde hace muy pocos años. Sin embargo, utilizan estructuras de datos que ya existían hace algunas décadas. Es lógico pensar que la gente del área académica no se dedique a estudiar en profundidad la última versión de un determinado software cuando esa tecnología ya la conocían hace mucho tiempo. Por esa misma razón es raro en el mundo observar que una universidad tenga convenios con una transnacional de software para dictar solamente esa tecnología pues, nuestra función es generar esa tecnología **y no sólo saber usarla**.

Tampoco debemos olvidar que los alumnos que ingresan hoy saldrán al mercado dentro de 5 años aproximadamente y, en un mundo que cambia tan rápido, no podemos ni debemos enseñarles tomando en cuenta solamente el mercado actual. Nuestros profesionales deben estar preparados para resolver los problemas que habrá dentro de 10 o 15 años y eso sólo es posible a través de la investigación.

## 1.4. Información institutional

### 1.4.1. Misión

Por lo antes mencionado, nuestra misión es:

Contribuir al desarrollo científico, tecnológico y técnico del país formando profesionales competentes, orientados a la creación de nueva ciencia y tecnología computacional, como motor que impulse y consolide la industria del software en base a la investigación científica y tecnológica en áreas innovadoras formando, EN NUESTROS profesionales, un conjunto de habilidades y destrezas para la solución de problemas computacionales con un compromiso social.

### 1.4.2. Visión

- Queremos ser una carrera profesional acreditada con estándares internacionales que cuente con el reconocimiento en función de la calidad y competitividad de sus docentes y egresados.
- Queremos ser una carrera que trascienda por la relevancia y pertinencia de sus proyectos de investigación básica y aplicada.
- Buscamos ser una carrera que promueva el desarrollo de la industria del software relacionada a Ciencia de la Computación a nivel internacional, incorporando a sus egresados a la industria ya establecida o generando nuevas empresas en este mismo rubro.
- Queremos ser una carrera que comparta y difunda el conocimiento con todos los sectores de la población y contribuya a la solución de los problemas estratégicos de nuestra sociedad.

## 1.5. Información de acreditacion

### 1.5.1. Resultados de la carrera *Outcomes*

Al finalizar esta carrera, el(la) egresado(a), habrá logrado los siguientes resultados (*Outcomes*):

**1)** Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones.

**2)** Diseñar, implementar y evaluar una solución basada en la computación para satisfacer un conjunto dado de requisitos de computación en el contexto de la disciplina del programa.

**3)** Comunicarse efectivamente en diversos contextos profesionales.

**4)** Reconocer las responsabilidades profesionales y tomar decisiones informadas en la práctica de la computación basadas en principios legales y éticos.

**5)** Funcionar efectivamente como miembro o líder de un equipo involucrado en actividades apropiadas a la disciplina del programa.

**6)** Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación.

**7)** Desarrollar principios investigación en el área de computación con niveles de competividad internacional.

Las competencias y/o resultados propuestas en esta malla curricular corresponden a las competencias y/o resultados propuestas por ABET[2]. Por otro lado, las habilidades adicionales son aportes de este grupo de trabajo.

### 1.5.2. Objetivos educacionales

Después de cinco años de egresado de la carrera profesional de Ciencia de la Computación, nuestros profesionales deben ser capaces de:

1. Cumplir y superar las expectativas de trabajo definidas por el entorno laboral.

2. Desempeñarse como miembro, o líder, de un equipo de trabajo tanto especializado como multidisciplinario.

3. Proponer soluciones al contexto laboral, donde se desenvuelve, basadas en la implementación o mejora del estado del arte en Ciencia de la Computación.

4. Comunicar de forma efectiva propuestas tecnológicas a personas de distintos niveles de conocimiento y de diferentes ámbitos sociales.

5. Actualizarse y adaptarse a nuevos conocimientos en Ciencia de la Computación y a diferentes ámbitos laborales, de forma autónoma o mediante estudios complementarios.

6. Demostrar un claro entendimiento de las consecuencias que surgen a partir de creaciones tecnológicas relacionadas a la Ciencia de la Computación en aspectos tales como lo social, ético, humano, moral, legal, ambiental, económico, entre otros.

[2] http://www.abet.org

### 1.5.3. Objetivos de Aprendizaje (*Learning Outcomes*)

Cada KU dentro de un KA enumera tanto un conjunto de temas y los resultados de aprendizaje (*Learning Outcomes*) que los estudiantes deben alcanzar en lo que respecta a los temas especificados. Resultados de aprendizaje no son de igual tamaño y no tienen una asignación uniforme de horas curriculares; temas con el mismo número de horas pueden tener muy diferentes números de los resultados del aprendizaje asociados.

Cada resultado de aprendizaje tiene un nivel asociado de dominio. En la definición de los diferentes niveles que dibujamos de otros enfoques curriculares, especialmente la taxonomía de Bloom, que ha sido bien explorada dentro de la Ciencia de la Computación. En este documento no se aplicó directamente los niveles de Bloom en parte porque varios de ellos son impulsados por contexto pedagógico, que introduciría demasiada pluralidad en un documento de este tipo; en parte porque tenemos la intención de los niveles de dominio para ser indicativa y no imponer restricción teórica sobre los usuarios de este documento.

Nosotros usamos tres niveles de dominio esperados que son:

**Nivel 1 Familiarizarse(*Familiarity*)** : El estudiante **comprende** un concepto básicamente. Responde: **¿Qué sabe sobre esto?**

**Nivel 2 Usar(*Usage*)** : El estudiante **aplica** conceptos en situaciones prácticas (ej: programación). Responde: **¿Cómo lo haría?**

**Nivel 3 Evaluar(*Assessment*)** : El estudiante **evalúa y justifica** enfoques. Responde: **¿Por qué este método?**

Por ejemplo, para evaluar los niveles de dominio, consideremos la noción de iteración en el desarrollo de software (for, while e iteradores). En el plano de la "familiaridad", se espera que un estudiante tenga una definición del concepto de iteración en el desarrollo de software y saber por qué esta técnica es útil.

Con el fin de mostrar el dominio del nivel "Uso", el estudiante debe ser capaz de escribir un programa adecuadamente usando una forma de iteración.

En el nivel de "Evaluación", en la iteración se requeriría que un estudiante comprenda múltiples métodos de iteración y que sea capaz de seleccionar apropiadamente entre ellos para diferentes aplicaciones.

## 1.6. Perfiles

### 1.6.1. Perfil del estudiante

El aspirante a ingresar a la Escuela Profesional de Ciencia de la Computación de la Universidad Nacional de Moquegua (UNAM) debe tener:
Conocimientos de:

1. Conceptos básicos de operaciones algebraicas, geometría y precálculo.
2. Su entorno social en la actualidad.

Habilidades para:

1. Entender las relaciones entre los hechos y encontrar las causas que los produjeron, prever consecuencias y así poder resolver problemas de una manera coherente.
2. Diferenciar patrones, es decir, captar la diferencia entre la realidad observada y el modelo mental o idea preconcebida que se ha tenido.
3. Percibir las relaciones lógicas (de funcionamiento o de comportamiento) existentes entre las observaciones realizadas.
4. Expresarse de manera oral o escrita las posibles soluciones para un problema dado.
5. Concentrarse y apertura al esfuerzo.

6. Comprender, analizar y sintetizar.

7. Formar hábitos y métodos adecuados para el estudio.

Actitudes de:

1. Interés y gusto por el estudio de Ciencia de la Computación y matemáticas.

2. Disposición para el trabajo académico, en forma cooperativa y participativa, dentro y fuera del aula de clases.

3. Iniciativa y competencia en el desempeño escolar.

### 1.6.2. Perfil del graduado

El egresado del Programa Profesional de Ciencia de la Computación es capaz de:

1. Desarrollar tecnología computacional buscando el bien común de los individuos, la sociedad y las organizaciones.

2. Aportar con su formación humana y sus capacidades científicas y profesionales con la solución de los problemas sociales de nuestro entorno.

3. Transformar, acelerar y ampliar los límites de cualquier área del conocimiento a través de soluciones innovadoras basadas en el uso eficiente de tecnología computacional.

4. Adaptarse rápidamente a los cambios tecnológicos debido a su formación basada en la investigación constante.

5. Trabajar y liderar equipos multidisciplinarios que llevan a cabo proyectos de innovación tecnológica.

6. Incrementar las ventajas competitivas de cualquier organización a través del uso eficiente de tecnología computacional gracias a su alta capacidad de abstracción.

7. Crear empresas de base tecnológica.

8. Poder seguir estudios de postgrado con exigencia internacional en áreas relacionadas.

### 1.6.3. Perfil Profesional

El profesional en Ciencia de la Computación tiene las siguientes características:

- Desarrollar tecnología computacional buscando el bien común de los individuos, la sociedad y las organizaciones.
- Transformar, acelerar y ampliar los límites de cualquier área del conocimiento a través de soluciones innovadoras basadas en el uso eficiente de tecnología computacional.
- Adaptarse rápidamente a los cambios tecnológicos debido a su formación basada en la investigación constante.
- Trabajar y liderar equipos multidisciplinarios que llevan a cabo proyectos de innovación tecnológica.
- Incrementar las ventajas competitivas de cualquier organización a través del uso eficiente de tecnología computacional gracias a su alta capacidad de abstracción.
- Crear empresas de base tecnológica.
- Poder seguir estudios de postgrado con exigencia internacional en áreas relacionadas.

### 1.6.4. Perfil del docente

El docente de la carrera de Ciencia de la Computación es una persona que promueve la formación integral del estudiante a través de la enseñanza, la investigación y extensión, transmitiendo su sólidos conocimientos en los cursos que imparte fortaleciendo sus valores como persona y miembro de la sociedad.

Debe ser capaz de transmitir sus conocimientos de una forma simple y clara a los estudiantes fomentando en ellos la investigación y el aprendizaje continuo para que sean capaces de adaptarse fácilmente a los cambios propios del avance del área y promover en ellos la búsqueda del bien común de los individuos, la sociedad y las organizaciones.

Debe ser un profesional que busca con frecuencia capacitación y grados académicos avanzados **de calidad** y muy preferentemente con dedicación a tiempo completo con el objetivo de desplegarse mejor buscando el bien común.

El docente participa de la formación e integración del estudiante a una comunidad académica buscando siempre la exigencia continua.

## 1.7. Grados y Títulos

Cumplida la aprobación de los créditos académicos propuestos en la malla curricular y luego de haber cumplido con los requisitos estipulados en el reglamento de grados y títulos de la carrera, el egresado recibirá:

**Grado Académico:** Bachiller en Ciencia de la Computación y

**Titulo Profesional:** Licenciado en Ciencia de la Computación

## 1.8. Recursos para dictado de clases

Un profesional innovador debe estar al tanto de los últimos avances de su área siempre. Los últimos avances de esta área no son presentados en los libros necesariamente. Debemos utilizar publicaciones de revistas indexadas de circulación mundial.

Por esa razón, tomamos como base la suscripción institucional a la ACM y a la IEEE-CS. Es recomendado que el docente use este material para discutir en clase las tendencias en todas las áreas.

Los laboratorios de cómputo también deben ser renovados de acuerdo a la rapidez propia de esta área y esto implica renovaciones constantes para poder garantizar que los estudiantes estén actualizados.

El recurso físico en cuanto a laboratorios para cada curso está detallado en el Capítulo 7.

## 1.9. Organización del documento

El resto de este documento está organizado de la siguiente forma: el Capítulo 1, define y explica el campo de acción de la Ciencia de la Computación, además se hace una muy breve explicación de las distintas carreras relacionadas a esta disciplina. En el caso particular del area de computación, las propuestas de ACM/IEEE-CS *Computing Curricula* [ACM/IEEE-CS, 2020].

El Capítulo 2, muestra las áreas de Conocimiento de Ciencia de la Computación, indicando los tópicos y objetivos de aprendizaje de los temas, pertenecientes a estos grupos.

El Capítulo 3 contiene la distribución por semestres, por áreas, por niveles, visión gráfica de la malla curricular, comparación con las diversas propuestas internacionales, distribución de tópicos por curso así como la distribución de habilidades por materia.

El Capítulo 4 contiene información detallada para cada uno de los cursos incluyendo las habilidades con las cuales contribuye, bibliografía por cada unidad así como el número de horas mínimas por cada unidad.

En el Capítulo 5 podemos observar un listado de cursos por profesor y de profesor por curso.

En el Capítulo 6 se presentan las tablas de equivalencias con otros planes curriculares.

En el Capítulo 7 se presenta una sugerencia de los laboratorios requeridos para el dictado de clases las mismas que podrían variar de acuerdo al volumen de alumnos que se tenga.

# Capítulo 2

# Cuerpo del conocimiento de Ciencia de la Computación–*Computer Science*

Las 18 áreas de conocimiento en Ciencia de la Computación–*Computer Science* son:

## 2.1. Algoritmos y Complejidad (AL)

Los algoritmos son fundamentales para la ciencia de la computación y la ingeniería de software. En el mundo real el rendimiento de cualquier sistema de software depende de: (1) los algoritmos elegidos y (2) la idoneidad y la eficiencia de las diversas capas de aplicación. Un buen diseño de algoritmos es crucial para el rendimiento de todos los sistemas de software. Por otra parte, el estudio de algoritmos da una idea de la naturaleza intrínseca del problema y las posibles soluciones técnicas independientes del lenguaje de programación, paradigma de programación, hardware, o cualquier otro aspecto de implementación.

Una parte importante de la informática es la capacidad de seleccionar los algoritmos apropiados para determinados propósitos y aplicarlos, reconociendo la posibilidad de que no se encuentre un algoritmo adecuado. Este facilidad se basa en la comprensión de la variedad de algoritmos que abordan un importante conjunto de problemas bien determinadas, reconociendo sus fortalezas y debilidades, y su idoneidad en determinados contextos. La eficiencia es un tema omnipresente en toda esta área. Esta área de conocimiento se definen los conceptos centrales y los conocimientos necesarios para diseñar, implementar y analizar los algoritmos para resolver problemas. Los algoritmos son esenciales en todas las áreas avanzadas de la informática: inteligencia artificial, bases de datos, computación distribuida, gráficos, redes, sistemas operativos, lenguajes de programación, de seguridad, y así sucesivamente. Algoritmos que tienen utilidad específica en cada uno de éstos se enumeran en las áreas de conocimiento relevantes. Criptografía, por ejemplo, aparece en la nueva área de conocimiento en Aseguramiento y Seguridad de la Información (IAS), mientras que los algoritmos paralelos y distribuidos aparecen el Área de Conocimiento de Computación paralela y distribuída (PD).

Al igual que con todas las áreas de conocimiento, el orden de los temas y sus agrupaciones no ne-

cesariamente se correlaciona con un orden específico de presentación. Diferentes programas enseñarán los temas en diferentes cursos y deben hacerlo en el orden que ellos creen es el más apropiado para sus estudiantes.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.1.1 Análisis Básico (Pág. 13) | | | No |
| 2.1.2 Estrategias Algorítmicas (Pág. 14) | 1 | | No |
| 2.1.3 Algoritmos y Estructuras de Datos fundamentales (Pág. 15) | 3 | | No |
| 2.1.4 Computabilidad y complejidad básica de autómatas (Pág. 16) | 3 | | No |
| 2.1.5 Complejidad Computacional Avanzada (Pág. 16) | | | No |
| 2.1.6 Teoría y Computabilidad Avanzada de Autómatas (Pág. 17) | | | No |
| 2.1.7 Estructuras de Datos Avanzadas y Análisis de Algoritmos (Pág. 17) | | | No |

### 2.1.1. AL/Análisis Básico

**Temas:**
**Core Tier1**
- 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.

**Core Tier2**
- Uso de la notación Big O.
- Notación Little o, Big omega y Big theta.
- Relaciones recurrentes.
- Análisis de algoritmos iterativos y recursivos.
- Algunas versiones del Teorema Maestro.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Explique a que se refiere con “mejor”, “esperado” y “peor” caso de comportamiento de un algoritmo [Familiarizarse]
2. En el contexto de a algoritmos específicos, identifique las características de data y/o otras condiciones o suposiciones que lleven a diferentes comportamientos [Evaluar]
3. Determine informalmente el tiempo y el espacio de complejidad de simples algoritmos [Usar]
4. Indique la definición formal de Big O [Familiarizarse]
5. Lista y contraste de clases estándares de complejidad [Familiarizarse]
6. 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]
7. Da ejemplos que ilustran las compensaciones entre espacio y tiempo que se dan en los algoritmos [Familiarizarse]

**Core-Tier2:**

8. Use la notación formal de la Big O para dar límites superiores asintóticos en la complejidad de tiempo y espacio de los algoritmos [Usar]
9. Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Usar]

10. Explicar el uso de la notación theta grande, omega grande y o pequeña para describir la cantidad de trabajo hecho por un algoritmo [Familiarizarse]

11. Usar relaciones recurrentes para determinar el tiempo de complejidad de algoritmos recursivamente definidos [Usar]

12. Resuelve relaciones de recurrencia básicas, por ejemplo. usando alguna forma del Teorema Maestro [Usar]

### 2.1.2. AL/Estrategias Algorítmicas (1 horas Core-Tier1)

Un instructor puede optar por cubrir estas estrategias algorítmicas en el contexto de los algoritmos presentados en "Estructuras de Datos y Algoritmos Fundamentales". Mientras que el número total de horas para las dos unidades de conocimiento (18) se puede dividir de forma diferente entre ellos, nuestra sensación es que la proporción de 1: 2 es razonable.

**Temas:**

**Core Tier1**

- Algoritmos de fuerza bruta.
- Algoritmos voraces.
- Divide y vencerás.
- Bactraking recursivo.
- Programación Dinámica.

**Core Tier2**

- Ramificación y poda.
- Heurísticas.
- Reducción: Transformar y Conquistar.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. 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 [Familiarizarse]

2. Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima [Evaluar]

3. Usa un algoritmo de divide-y-vencerás para resolver un determinado problema [Usar]

4. Usa recursividad en reversa a fin de resover un problema como en el caso de recorrer un laberinto [Usar]

5. Usa programación dinámica para resolver un problema determinado [Usar]

6. Determina el enfoque algorítmico adecuado para un problema [Evaluar]

**Core-Tier2:**

7. Describe varios métodos basados en heurísticas para resolver problemas [Familiarizarse]

8. Usa en enfoque heurístico para resolver un problema determinado [Usar]

9. Describe las compensaciones que se dan entre usar estrategias de fuerza bruta y aquellas basadas en heurísticas [Evaluar]

10. Describe como un enfoque de ramificación y poda puede ser usado para mejorar el rendimiento de un método heurístico [Familiarizarse]

### 2.1.3. AL/Algoritmos y Estructuras de Datos fundamentales (3 horas Core-Tier1)

Esta unidad de conocimiento se ha hecho directamente con la base proporcionada por Fundamentos del desarrollo de software (SDF), en particular el material de Estructuras de Datos Fundamentales y Algoritmos y Diseño.

**Temas:**

**Core Tier1**

- 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: 1. Operaciones comunes en árboles de búsqueda binaria como seleccionar el mínimo, máximo, insertar, eliminar, recorrido en árboles.
- Grafos y algoritmos en grafos: 1. Representación de grafos (ej., lista de adyacencia, matriz de adyacencia) 2. Recorrido en profundidad y amplitud

**Core Tier2**

- Montículos (Heaps)
- Grafos y algoritmos en grafos: 1. Algoritmos de la ruta más corta (algoritmos de Dijkstra y Floyd) 2. 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Implementar algoritmos numéricos básicos [Usar]
2. Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar]
3. Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Usar]
4. Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse]
5. Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, busqueda y hashing [Familiarizarse]
6. 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]
7. Explicar como el balanceamiento del arbol afecta la eficiencia de varias operaciones de un arbol de búsqueda binaria [Familiarizarse]
8. Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Usar]
9. 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]

**Core-Tier2:**

10. Describir la propiedad del heap y el uso de heaps como una implementación de colas de prioridad [Familiarizarse]
11. 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]

12. Trazar y/o implementar un algoritmo de comparación de string [Usar]

### 2.1.4. AL/Computabilidad y complejidad básica de autómatas (3 horas Core-Tier1)

**Temas:**
**Core Tier1**
- Máquinas de estado finito.
- Expresiones regulares.
- Problema de la parada.

**Core Tier2**
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Discute el concepto de máquina de estado finito [Familiarizarse]
2. Diseñe una máquina de estado finito determinista para aceptar un determinado lenguaje [Usar]
3. Genere una expresión regular para representar un lenguaje específico [Usar]
4. Explique porque el problema de la parada no tiene solución algorítmica [Familiarizarse]

**Core-Tier2:**

5. Diseñe una gramática libre de contexto para representar un lenguaje especificado [Usar]
6. Define las clases P y NP [Familiarizarse]
7. Explique el significado de NP-Completitud [Familiarizarse]

### 2.1.5. AL/Complejidad Computacional Avanzada

**Temas:**
**Electivo**
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Define las clases P y NP (También aparece en AL / Automata Básico, Computalidad y Complejidad) [Familiarizarse]
2. Define la clase P-Space y su relación con la clase EXP [Familiarizarse]
3. Explique el significado de NP-Completo (También aparece en AL / Automata Básico, Computalidad y Complejidad) [Familiarizarse]
4. Muestre ejemplos de problemas clásicos en NP - Completo [Familiarizarse]
5. Pruebe que un problema es NP- Completo reduciendo un problema conocido como NP-Completo [Usar]

### 2.1.6. AL/Teoría y Computabilidad Avanzada de Autómatas

**Temas:**
**Electivo**

- Conjuntos y Lenguajes: 1. Lenguajes Regulares. 2. Revisión de autómatas finitos determinísticos (Deterministic Finite Automata DFAs) 3. Autómata finito no determinístico (Nondeterministic Finite Automata NFAs) 4. Equivalencia de DFAs y NFAs. 5. Revisión de expresiones regulares; su equivalencia con autómatas finitos. 6. Propiedades de cierre. 7. Probando no-regularidad de lenguajes, a través del lema de bombeo (Pumping Lemma) o medios alternativos.
- Lenguajes libres de contexto: 1. Autómatas de pila (Push-down automata (PDAs) 2. Relación entre PDA y gramáticas libres de contexto. 3. Propiedades de los lenguajes libres de contexto.
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Determina la ubicación de un lenguaje en la jerarquía de Chomsky (regular, libre de contexto, enumerable recursivamente) [Evaluar]
2. Convierte entre notaciones igualmente poderosas para un lenguaje, incluyendo entre estas AFDs, AFNDs, expresiones regulares, y entre AP y GLCs [Usar]
3. Explica la tesis de Church-Turing y su importancia [Familiarizarse]
4. Explica el teorema de Rice y su importancia [Familiarizarse]
5. Da ejemplos de funciones no computables [Familiarizarse]
6. Demuestra que un problema es no computable al reducir un problema clásico no computable en base a él [Usar]

### 2.1.7. AL/Estructuras de Datos Avanzadas y Análisis de Algoritmos

Muchos programas querrán que sus estudiantes tengan la exposición a los algoritmos más avanzados o los métodos de análisis. A continuación se presenta una selección de posibles temas avanzados que están al día y oportuna, pero no de una manera exhaustiva.
**Temas:**
**Electivo**

- Árboles balanceados (ej. árboles AVL, Arboles red-black, Arboles biselados (splay trees), Treaps)
- Grafos (ej. Ordenamiento Topológico, encontrando componentes puertemente conectados)
- Estructuras de Datos Avanzadas (ej. B-Trees, Fibonacci Heaps)
- Estructuras de Datos y Algoritmos basados en cadenas (ej. Arrays de sufijos, Arboles de sufijos, Arboles digitales (Tries)
- Redes de Flujo (ej. Flujo Máximo [Algoritmo de Ford-Fulkerson], Flujo Máximo - Mínimo Corte, Máxima Asignación Bipartita)
- Programación Lineal (Dualidad, Método Simplex, Algoritmos de punto interior)
- Algoritmos Teórico-Numéricos (Aritmética Modular, Prueba del Número Primo, Factorización Entera)
- Algoritmos Geométricos (Puntos, Segmentos de Línea, Polígonos [propiedades, intersecciones], encontrando el polígono convexo)
- Algoritmos aleatorios.
- Algortimos estocásticos.

- Algoritmos de aproximación.
- Análisis amortizado.
- Análisis Probabilístico.
- Algoritmos en línea y análisis competitivo.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Entender el mapeamento de problemas del mundo real a soluciones algorítmicas (ejemplo, problemas de grafos, programas lineares,etc) [Evaluar]
2. Seleccionar y aplicar técnicas de algoritmos avanzadas (ejemplo, randonmización, aproximación) para resolver problemas reales [Evaluar]
3. Seleccionar y aplicar técnicas avanzadas de análisis (ejemplo, amortizado, probabilistico,etc) para algoritmos [Evaluar]

## 2.2. Arquitectura y Organización (AR)

Profesionales de la informática no deben considerar a la computadora como un simple cuadro negro que ejecuta programas por arte de magia. El área de conocimiento Arquitectura y Organización se basa en Fundamentos de Sistemas (SF) para desarrollar una comprensión más profunda del entorno de hardware sobre la que se basa toda la computación, y la interfaz que proporciona a las capas superiores de software. Los estudiantes deben adquirir una comprensión y apreciación de los componentes de un sistema informático funcionales, sus características, el rendimiento y las interacciones, y, en particular, el reto de aprovechar el paralelismo para sostener mejoras de rendimiento, ahora y en el futuro. Los estudiantes necesitan entender la arquitectura de computadores para desarrollar programas que puede lograr un alto rendimiento a través de la conciencia de un programador de paralelismo y la latencia. En la selección de un sistema a utilizar, los estudiantes deben ser capaces de entender el equilibrio entre los diversos componentes, tales como la velocidad del reloj de la CPU, ciclos por instrucción, tamaño de la memoria, y el tiempo medio de acceso a memoria.

Los resultados del aprendizaje especificados para estos temas corresponden principalmente al núcleo y están destinados a apoyar los programas que eligen requerir sólo el mínimo de 16 horas de la arquitectura de computadores de sus estudiantes. Para los programas que quieren enseñar más que el mínimo, los mismos temas de AR pueden ser tratados en un nivel más avanzado en la implementación de una secuencia de dos platos. Para los programas que quieren cubrir los temas electivos, esos temas se pueden introducir dentro de una secuencia de dos cursos y / o ser tratados de una manera más completa en un tercer curso.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.2.1 Lógica digital y sistemas digitales (Pág. 18) | 3 | | No |
| 2.2.2 Representación de datos a nivel máquina (Pág. 19) | 3 | | No |
| 2.2.3 Organización de la Máquina a Nivel Ensamblador (Pág. 20) | 6 | | No |
| 2.2.4 Organización y Arquitectura del Sistema de Memoria (Pág. 20) | 3 | | No |
| 2.2.5 Interfaz y comunicación (Pág. 21) | 1 | | No |
| 2.2.6 Organización funcional (Pág. 21) | | | No |
| 2.2.7 Multiprocesamiento y arquitecturas alternativas (Pág. 22) | | | No |
| 2.2.8 Mejoras de rendimiento (Pág. 22) | | | No |

### 2.2.1. AR/Lógica digital y sistemas digitales (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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)
- Restricción física (Retrasos de Entrada, fan-in, fan-out, energia/poder)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. 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]
2. 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]
3. 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]
4. 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]
5. 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]
6. Usar herramientas CAD para capturar, sistetizar, y simular bloques de construcción (como ALUs, registros, movimiento entre registros) de un computador simple [Usar]
7. Evaluar el comportamiento de un diagrama de tiempos y funcional de un procesador simple implementado a nivel de circuitos lógicos [Evaluar]

### 2.2.2. AR/Representación de datos a nivel máquina (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Bits, Bytes y Words.
- Representación 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explicar porqué en computación todo es datos, inclusive las instrucciones [Familiarizarse]
2. Explicar las razones de usar formatos alternativos para representar datos numéricos [Familiarizarse]
3. Describir cómo los enteros negativos se almacenan con representaciones de bit de signo y complemento a 2 [Familiarizarse]
4. Explicar cómo las representaciones de tamaño fijo afectan en la exactitud y la precisión [Familiarizarse]
5. Describir la representación interna de datos no numéricos como caracteres, cadenas, registros y arreglos [Familiarizarse]
6. Convertir datos numéricos de un formato a otro [Usar]
7. Escribir programas simples a nivel de ensamblador o código máquina para procesamiento y manipulación de cadenas [Usar]

### 2.2.3. AR/Organización de la Máquina a Nivel Ensamblador (6 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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.
- Multiprocesadores de memoria compartida / organización multinúcleo.
- Introducción a SIMD vs. MIMD y Taxonomía de Flynn.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explicar la organización de la maquina clásica de von Neumann y sus principales unidades funcionales [Familiarizarse]
2. Describir cómo se ejecuta una instrucción en una máquina de von Neumann con extensión para hebras, sincronización multiproceso y ejecución SIMD (máquina vectorial) [Familiarizarse]
3. Describir el paralelismo a nivel de instrucciones y sus peligros, y cómo es esto tratado en pipelines de proceso típicos [Familiarizarse]
4. Resumir cómo se representan las instrucciones, tanto a nivel de máquina bajo el contexto de un ensamblador simbólico [Familiarizarse]
5. Demostrar cómo se mapean los patrones de lenguajes de alto nivel en notaciones en lenguaje ensamblador o en código máquina [Familiarizarse]
6. Explicar los diferentes formatos de instrucciones, así como el direccionamiento por instrucción, y comparar formatos de tamaño fijo y variable [Familiarizarse]
7. Explicar como las llamadas a subrutinas son manejadas a nivel de ensamblador [Familiarizarse]
8. Explicar los conceptos básicos de interrupciones y operaciones de entrada y salida (I/O) [Familiarizarse]
9. Escribir segmentos de programa simples en lenguaje ensamblador [Usar]
10. Ilustrar cómo los bloques constructores fundamentales en lenguajes de alto nivel son implementados a nivel de lenguaje máquina [Usar]

### 2.2.4. AR/Organización y Arquitectura del Sistema de Memoria (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Identifique las principales tecnologías de memoria (Por ejemplo: SRAM, DRAM, Flash,Disco Magnetico) y su relación costo beneficio [Familiarizarse]
2. Explique el efecto del retardo de la memoria en tiempo de ejecución [Familiarizarse]
3. Describa como el uso de jerarquía de memoria (caché, memoria virtual) es aplicado para reducir el retardo efectivo en la memoria [Familiarizarse]
4. Describa los principios de la administración de memoria [Familiarizarse]
5. Explique el funcionamiento de un sistema con gestión de memoria virtual [Familiarizarse]
6. 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 [Usar]

### 2.2.5. AR/Interfaz y comunicación (1 horas Core-Tier1)

Visión general de Sistemas Operativos Área de conocimiento para una discusión de la vision del sistema operativo de entrada / salida procesamiento y administración. Se enfoca en los mecanismos de hardware para soportar dispositivos interconectados y comunicaciones de procesador a procesador.
**Temas:**
**Core Tier2**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explicar como las interrupciones son aplicadas para implementar control de entrada-salida y transferencia de datos [Familiarizarse]
2. Identificar diversos tipos de buses en un sistema computacional [Familiarizarse]
3. Describir el acceso a datos desde una unidad de disco magnético [Familiarizarse]
4. Comparar organizaciones de red conocidas como organizaciones en bus/Ethernet, en anillo y organizaciones conmutadas versus ruteadas [Familiarizarse]
5. 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]
6. Describir las ventajas y limitaciones de las arquitecturas RAID [Familiarizarse]

### 2.2.6. AR/Organización funcional

Nota: Electiva para Ciencia de la Computación; Es fundamental para el currículo de Ingeniería de la Computación.
**Temas:**
**Electivo**

- 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.
- Instrucción (Pipelining)
- Introducción al paralelismo al nivel de instrucción (PNI)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Comparar implementaciones alternativas de ruta de datos [Familiarizarse]
2. Discutir el concepto de puntos de control y la generación de señales de control usando implementaciones a nivel de circuito o microprogramadas [Familiarizarse]
3. Explicar el paralelismo a nivel de instrucciones básicas usando pipelining y los mayores riesgos que pueden ocurrir [Familiarizarse]
4. Diseñar e implementar un procesador completo, incluyendo ruta de datos y control [Usar]
5. Calcular la cantidad promedio de ciclos por instrucción de una implementación con procesador y sistema de memoria determinados [Evaluar]

### 2.2.7. AR/Multiprocesamiento y arquitecturas alternativas

La vista aquí es sobre la implementación de hardware de SIMD y MIMD arquitecturas Arquitecturas paralelas.

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Discutir el concepto de procesamiento paralelo mas allá del clásico modelo de von Neumann [Familiarizarse]
2. Describir diferentes arquitecturas paralelas como SIMD y MIMD [Familiarizarse]
3. Explicar el concepto de redes de interconexión y mostrar diferentes enfoques [Familiarizarse]
4. Discutir los principales cuidados en los sistemas de multiprocesamiento presentes con respecto a la gestión de memoria y describir como son tratados [Familiarizarse]
5. 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 [Familiarizarse]

### 2.2.8. AR/Mejoras de rendimiento

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describir las arquitecturas superescalares y sus ventajas [Familiarizarse]
2. Explicar el concepto de predicción de bifurcaciones y su utilidad [Familiarizarse]
3. Caracterizar los costos y beneficios de la precarga prefetching [Familiarizarse]
4. Explicar la ejecución especulativa e identifique las condiciones que la justifican [Familiarizarse]
5. Discutir las ventajas de rendimiento ofrecida en una arquitectura de multihebras junto con los factores que hacen dificil dar el maximo beneficio de estas [Familiarizarse]
6. Describir la importancia de la escalabilidad en el rendimiento [Familiarizarse]

## 2.3. Ciencia Computacional (CN)

Ciencia Computacional es un campo de la Ciencia de la Computación aplicada, es decir, la aplicación de la Ciencia de la Computación para resolver problemas a través de una gama de disciplinas. En el libro Introducción a la Ciencia Computacional, los autores ofrecen la siguiente definición: ""El campo de la Ciencia computacional combina simulación por computador, visualización científica, modelos matemáticos, estructuras de programación y de datos para computadoras, Redes, Diseño de Base de Datos, cálculo simbólico y la computación de alto rendimiento con varias disciplinas"". La Ciencia de la Computación, que se centra en gran medida en la teoría, el diseño y implementación de algoritmos para la manipulación de datos e información, se puede remontar sus raíces a los primeros dispositivos utilizados para ayudar a la gente en computación hace más de cuatro mil años. Varios sistemas fueron creados y se utilizan para calcular las posiciones astronómicas. El logo de programación de Ada Lovelace fue pensado para calcular los números de Bernoulli. A finales del siglo XIX, las calculadoras mecánicas estuvieron disponibles y fueron utilizados inmediatamente por los científicos. Las necesidades de los Científicos e Ingenieros con relación a computación han impulsado mucho la investigación y la innovación en computación. A medida que las computadoras aumentan su poder en la solución de problemas, la ciencia computacional ha crecido tanto en amplitud e importancia. Es una disciplina por derecho propio y se considera que es una de las cinco con mayor crecimiento. Una increíble variedad de sub-campos han surgido bajo el paraguas de la Ciencia Computacional, incluyendo la biología computacional, química computacional, mecánica computacional, arqueología computacional, finanzas computacionales, sociología computacional y forenses computacionales.

Algunos conceptos fundamentales de la ciencia computacional son pertinentes a cada científico de la computación (por ejemplo, el modelado y la simulación), y temas de ciencias computacionales son componentes extremadamente valiosos de un programa de pregrado en ciencia de la computación. Esta área ofrece la exposición a muchas ideas y técnicas valiosas, incluyendo la precisión de la representación numérica, análisis de errores, técnicas numéricas, arquitecturas paralelas y algoritmos, modelado y simulación, visualización de la información, ingeniería de software y optimización. Temas de interés para la ciencia computacional incluyen conceptos fundamentales en la construcción de programas (Conceptos Fundamentales de Programación), el diseño de algoritmos (Algoritmos y Diseño), pruebas del programa (Métodos de Desarrollo), representaciones de datos (Representación de datos a nivel máquina), y arquitectura básica del computador (Organización y Arquitectura del Sistema de Memoria). Al mismo tiempo, los estudiantes que toman cursos en esta área tienen la oportunidad de aplicar estas técnicas en una amplia gama de áreas tales como la dinámica molecular y de fluidos, mecánica espacial, la economía, la biología, la geología, la medicina y análisis de redes sociales. Muchas de las técnicas que se utilizan en estas áreas requieren matemáticas avanzadas tales como cálculo, ecuaciones diferenciales y álgebra lineal. Las descripciones siguientes asumen que los estudiantes han adquirido los conocimientos matemáticos necesarios en otros lugares. En la comunidad de ciencia computacional, los términos de ejecución, modificación y creación a menudo se utilizan para describir los niveles de entendimiento. Este capítulo sigue las convenciones de otros capítulos de este volumen y usa los términos familiaridad, uso y evaluación.

| área de Conocimiento (*Knowledge Area*-KA) (KA) | Core Tier1 | Core Tier2 | Electivo |
|---|---|---|---|
| 2.3.1 Introducción al modelamiento y simulación (Pág. 24) | | | No |
| 2.3.2 Modelamiento y simulación (Pág. 24) | | | No |
| 2.3.3 Procesamiento (Pág. 25) | | | No |
| 2.3.4 Visualización interactiva (Pág. 26) | | | No |
| 2.3.5 Datos, información y conocimiento (Pág. 27) | | | No |
| 2.3.6 Análisis numérico (Pág. 28) | | | No |

### 2.3.1. CN/Introducción al modelamiento y simulación

La abstracción es un concepto fundamental en la Ciencia de la computación. Un enfoque principal de la computación es abstraer el mundo real, crear un modelo que puede ser simulado en una máquina. Las raíces de la Ciencia de la Computación se pueden remontar a este enfoque, las cosas de modelado como trayectorias de proyectiles de artillería y los protocolos criptográficos de modelado, los cuales impulsaron el desarrollo de los primeros sistemas computacionales a principios y mediados de los años 1940.

Modelado y simulación de sistemas del mundo real representan los conocimientos esenciales para los profesionales de Ciencia de la Computación y proporcionan una base para esta área. Cualquier introducción al modelamiento y la simulación sería incluir o presumir una introducción a la computación. Además, un conjunto general de modelado y simulación de las técnicas, métodos de visualización de datos y las pruebas de software y mecanismos de evaluación también son importantes.

**Temas:**

**Core Tier1**

- Modelos como abstracciones de situaciones.
- Simulaciones como modelamiento dinámico.
- Técnicas y herramientas de simulación, tales como simulaciones físicas, simulaciones humana-in-the-loop guiada, y la realidad virtual.
- Enfoques fundamentales para la validación de modelos (por ejemplo, comparando la salida de una simulación a los datos reales o la salida de otro modelo).
- Presentación de los resultados en un formulario de interés para el sistema que se está modelando.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Explicar el concepto de modelado y del uso de la abstracción que permita el uso de una máquina para resolver un problema [Familiarizarse]
2. Describir las relaciones entre modelado y simulación, por ejemplo, pensar en la simulación como un modelado dinámico [Familiarizarse]
3. Crear un modelo matemático formal, simple de una situación del mundo real y usa dicho modelo en una simulación [Usar]
4. Distinguir entre los distintos tipos de simulación, incluyendo simulaciones físicas, simulaciones guiadas por humanos, y realidad virtual [Familiarizarse]
5. Describir diversos enfoques para la validación de modelos [Familiarizarse]
6. Crear una visualización simple de los resultados de una simulación [Usar]

### 2.3.2. CN/Modelamiento y simulación

**Temas:**

**Electivo**

- Propósito de modelamiento y simulación incluyendo optimización; Soporte en la toma de decisiones, predicciones, consideraciones de seguridad; para entrenamiento y educación.
- Tradeoffs. Incluyendo rendimiento, precisión, validez y complejidad.

- Simulación de procesos; identificación de las principales características o comportamientos, simplificación de hipótesis; validación de resultados.
- Construcción del modelo: Uso de fórmulas matemáticas o ecuaciones, grafos, restricciones; metodologías y técnicas; uso del tiempo paso a paso (time-stepping) para sistemas dinámicos.
- Modelos formales y técnicas de modelamiento: descripciones matemáticas que implican simplificar hipótesis y eludir detalles. Ejemplos de técnicas incluidas: 1. Métodos de Monte Carlo. 2. Procesos estocásticos. 3. Teoría de colas. 4. Redes de Petri y redes de Petri coloreadas. 5. Estructuras de grafos tales como grafos dirigidos, árboles, redes. 6. Juegos, teoría de juegos, modelamiento de cosas usando teoría de juegos. 7. Programación lineal y sus extensiones . 8. Programación dinámica. 9. Ecuaciones diferenciales: EDO, EDP. 10. Técnicas no lineales. 11. Espacio de estados y transiciones.
- Valoración y evaluación de modelos y simulaciones en una variedad de contextos; verificación y validación de modelos y simulaciones.
- Áreas de aplicación importantes incluida la atención médica y el diagnóstico, la economía y las finanzas, la ciudad y el urbanismo, la ciencia y la ingeniería.
- Software en apoyo de la simulación y el modelado; paquetes, idiomas.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar y dar ejemplos de los beneficios de la simulación y el modelamiento en un rango de áreas de aplicación importantes [Familiarizarse]
2. Demostrar la habilidad para aplicar las técnicas de modelamiento y simulación a un rango de áreas problemáticas [Usar]
3. Explicar los constructores y conceptos de un enfoque de modelo en particular [Familiarizarse]
4. Explicar la diferencia entre verificación y validación de un modelo; demostrar la diferencia con ejemplos específicos [Evaluar]
5. Verificar y validar el resultado de una simulación [Evaluar]
6. Evaluar una simulación destacando sus beneficios y debilidades [Evaluar]
7. Escoger una propuesta de modelado apropiada para un determinado problema o situación [Evaluar]
8. Comparar resultados de diferentes simulaciones del mismo fenómeno y explicar cualquier diferencia [Evaluar]
9. Deducir el comportamiento de un sistema a partir de los resultados de simulación del sistema [Evaluar]
10. Extender o adaptar un modelo existente a nuevas situaciones [Evaluar]

### 2.3.3. CN/Procesamiento

El área temática de procesamiento incluye numerosos temas de otras áreas de conocimiento. En concreto, la cobertura del tratamiento debe incluir un análisis de las arquitecturas de hardware, incluyendo los sistemas paralelos, jerarquías de memoria, y las interconexiones entre procesadores. Estos se tratan en: Interfaz y comunicación (Pág 21), Multiprocesamiento y arquitecturas alternativas (Pág 22), Mejoras de rendimiento (Pág 22).

**Temas:**
**Electivo**

- Conceptos Fundamentales de Programación: 1. El concepto de un algoritmo que consiste en un número finito de pasos bien definidos, cada uno de los cuales se completa en una cantidad finita de tiempo, al igual que todo el proceso. 2. Ejemplos de algoritmos bien conocidos como ordenamiento y busqueda. 3. El concepto de análisis como entendimiento de lo que el problema realmente pregunta, cómo un problema puede ser abordado utilizando un algoritmo y cómo se representa la información de manera que una máquina pueda procesarla. 4. El desarrollo o la

identificación de un flujo de trabajo (*workflow*). 5. El proceso de convertir un algoritmo en código máquina ejecutable. 6. Procesos de Software incluyendo modelos de ciclo de vida, requerimientos, diseño, implementación, verificación y mantenimiento. 7. Representación máquina de datos de aritmética para computación.

- Métodos Numéricos 1. Algoritmos para encajar datos numéricamente (e.g. Método de Newton) 2. Algoritmos para computación numérica, incluyendo arquitecturas paralelas.
- Propiedades fundamentales de computación paralela y distribuida: 1. Ancho de banda. 2. Latencia. 3. Escalabilidad. 4. Granularidad. 5. Paralelismo, incluyendo parelelismo de tareas, datos y eventos. 6. Arquitecturas Paralelas incluyendo arquitecturas de procesador, memoria y caching. 7. Paradigmas de Programación Paralela incluyendo hilos, paso de mensajes, técnicas orientadas a eventos, Arquitecturas de Software Paralelo y *Map / Reduce*. 8. Computación en Grid. 9. El impacto de la arquitectura en tiempo computacional. 10. Tiempo total de la curva de la ciencia para el paralelismo: continuidad de las cosas.
- Costos de Cálculo, ej., el costo de recalcular un valor vs el costo de almacenar y buscar.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar las características y definir las propiedades de los algoritmos y como se relacionan con el procesamiento de la maquina [Familiarizarse]
2. Analizar declaraciones simples de problemas para identificar información relevante y seleccionar el procedimiento apropiado para resolver el problema [Evaluar]
3. Identificar o bosquejar un flujo de trabajo para un proceso computacional existente tal como la creación de un grafo basado en datos experimentales [Familiarizarse]
4. Describir el proceso de convertir un algoritmo a código maquina ejectuable [Familiarizarse]
5. Resumir las fases del desarrollo de software y comparar varios ciclos de vida en comín [Familiarizarse]
6. Explicar como los datos son representados en una máquina. Comparar representaciones de números enteros a numeros flotantes. Describir underflow, overflow, redondeo, y truncamiento de errores en la representación de datos [Familiarizarse]
7. Aplicar algoritmos numéricos estándar para resolver ecuaciones diferenciales ordinarias y parciales. Usar sistemas computaciones para resolver sistemas de ecuaciones [Usar]
8. Describir las propiedades básicas de ancho de banda, latencia, escalabilidad y granularidad [Familiarizarse]
9. Describir los niveles de paralelismo incluyendo tareas, datos, y eventos de paralelismo [Familiarizarse]
10. Comparar y contrastar paradigmas de programación paralela reconociendo las fortalezas y debilidades de cada una [Evaluar]
11. Identificar los problemas que afectan la corrección y eficiencia de un cálculo [Familiarizarse]
12. Diseñar, codificar, probar y depurar programas para un cálculo paralelo [Usar]

### 2.3.4. CN/Visualización interactiva

This sub-area is related to modeling and simulation. Most topics are discussed in detail in other knowledge areas in this document. There are many ways to present data and information, including immersion, realism, variable perspectives; haptics and heads-up displays, sonification, and gesture mapping.

Interactive visualization in general requires understanding of human perception (GV/Basics); graphics pipelines, geometric representationsand data structures (Conceptos Fundamentales); 2D and 3D rendering, surface and volume rendering (Rendering Básico, Modelado Geométrico, and Renderizado Avanzado); and the use of APIs for developing user interfaces using standard input components

such as menus, sliders, and buttons; and standard output components for data display, including charts, graphs, tables, and histograms.

**Temas:**

**Electivo**

- Principios de visualización de datos.
- Algoritmos de visualización y gráficos.
- Técnicas de procesamiento de imagenes.
- Preocupaciones de escalabilidad.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Comparar los mecanismos de interfaz de computadora comunes con respecto a la facilidad de uso, facilidad de aprendizaje, y costo [Evaluar]
2. Usar APIs estándar y herramientas para crear representaciones visuales de datos, incluyendo grafos, gráficos, tablas, e histogramas [Usar]
3. Describir varios enfoques para usar una computadora como un medio para interactuar y procesar datos [Familiarizarse]
4. Extraér información util de un conjunto de datos [Evaluar]
5. Analizar y seleccionar técnicas visuales para problemas específicos [Evaluar]
6. Describir problemas relacionados con la ampliación de análisis de datos, desde pequeñas a grandes conjuntos de datos [Familiarizarse]

### 2.3.5. CN/Datos, información y conocimiento

Muchos temas se discuten en detalle en otras áreas de conocimiento en este documento tales como: Conceptos de Gestión de la Información (Pág 53), Sistemas de Bases de Datos (Pág 54), Modelado de datos (Pág 55), Análisis Básico (Pág 13), Algoritmos y Estructuras de Datos fundamentales (Pág 15), Conceptos Fundamentales de Programación (Pág 98), Métodos de Desarrollo (Pág 99).

**Temas:**

**Electivo**

- Gestión de Contenido modelos, marcos de trabajo, sistemas, métodos de diseño (cómo GI: Gestión de Información).
- Representaciones digitales de contenido, incluyendo números, texto, imágenes (p.e. cuadricula y vector), video (p.e. Quicktime, MPEG2, MPEG4), audio (p.e. partituras escritas, MIDI, pista sonora digitalizada) y animaciones; objetos complejos/compuestos/agregados; registros bibliográficos.
- Creación/captura y preservación de contenido digital, incluyendo digitalización, muestreo, compresión, conversión, transformación/traducción, migración/emulación, rastreo, recolección.
- Estructura de contenido / administración, incluyendo librerias digitales y as pectos por estatica/dinamica/flujo: 1. Datos: Estructuras de datos, Bases de datos 2. Información: Colecciones de documentos, pools multimedia, hyperbases (hipertexto, hipermedios), catalogos, repositorios. 3. Conocimiento: ontologías, triple stores, redes semánticas, reglas
- Procesamiento y reconocimiento de patrones, incluyendo indexación, busquedas (incluyendo: consultas y lenguajes de consultas; centrales/federadas/P2P), recuperación, clusterisación, clasificación/categorización, análisis/minería/extracción, renderizado/reportes, operaciones de manipulación.
- Apoyo / sociedad de usuario para la presentación y la interacción, incluyendo exploración, buscar, filtrar, ruta, visualizar, compartir, colaborar, tasa, anotar, personalice, recomendar.
- Modelado, diseño, implementación lógica y física, el uso de sistemas pertinentes / software.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Identificar todos los datos, información y elementos de conocimiento y organizaciones afines, para una aplicación de ciencia computacional [Evaluar]

2. Describir cómo representar datos e información para su procesamiento [Familiarizarse]
3. Describir los requisitos típicos de usuario con respecto a los datos, información y conocimiento [Familiarizarse]
4. Seleccione una adecuada implementación del sistema o el software para gestionar los datos, información y conocimiento [Evaluar]
5. Listar y describir los informes, transacciones y otros procesos necesarios para una aplicación de la ciencia computacional [Familiarizarse]
6. Comparar y contrastar el manejo de la base de datos, recuperación de información, y sistemas de bibliotecas digitales en relación con el manejo de las aplicaciones típicas de la ciencia computacional [Evaluar]
7. Diseñar una biblioteca digital para algunos usuarios/sociaedades de la ciencia computacional, con contenido y servicios adecuados [Usar]

### 2.3.6. CN/Análisis numérico

AR/Representación de datos a nivel máquina.

**Temas:**

**Electivo**

- Error, estabilidad, convergencia, incluyendo truncado y redondeo.
- Aproximación de funciones, incluyendo series de Taylor, interpolación, extrapolación y regresión.
- Diferenciación e Integración numérica (Regla de Simpson, métodos explícitos e implícitos).
- Ecuaciones diferenciales (Método de Euler, diferencias finitas)

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Definir conceptos de error, estabilidad, precisión de máquina y la inexactitud de las aproximaciones computacionales [Familiarizarse]
2. Implementar series de Taylor, algoritmos de interpolación, extrapolación y regresión para aproximación de funciones [Usar]
3. Implementar algoritmos de diferenciación e integración [Usar]
4. Implementar algoritmos para solucionar ecuaciones diferenciales [Usar]

## 2.4. Estructuras Discretas (DS)

Estructuras discretas son el material fundamental para la informática. Por fundacional nos referimos a que son relativamente pocos los informáticos estarán trabajando principalmente en estructuras discretas, pero que muchas otras áreas de la informática requieren la capacidad de trabajar con los conceptos de estructuras discretas. Estructuras discretas incluyen material importante de áreas como la teoría de conjuntos, la lógica, la teoría de grafos y teoría de la probabilidad.

El material en estructuras discretas es un fenómeno generalizado en las áreas de estructuras de datos y algoritmos, sino que aparece en otras partes de la informática también. Por ejemplo, la capacidad de crear y entender una prueba-ya sea una prueba simbólica formal o una menos formal pero todavía matemáticamente rigurosa argumentación-es importante en prácticamente todas las áreas de ciencias de la computación, incluyendo (por nombrar sólo algunos) formal de especificación, verificación, bases de datos, y la criptografía. Conceptos teoría de grafos se utilizan en redes, sistemas operativos y compiladores. Conceptos teoría de conjuntos se utilizan en la ingeniería de software y bases de datos. Teoría de la probabilidad se utiliza en Inteligencia Artificial, redes y una serie de aplicaciones informáticas.

Dado que las estructuras discretas sirve como base para muchas otras áreas de la informática, vale la pena señalar que el límite entre estructuras discretas y otras áreas, particularmente Algoritmos y Complejidad, Software Fundamentos de desarrollo, lenguajes de programación e Inteligencia Artificial,

no siempre puede ser quebradizo . De hecho, diferentes instituciones pueden optar por organizar los cursos en los que cubren este material de muy diferentes maneras. Algunas instituciones pueden cubrir estos temas en uno o dos cursos enfocados con títulos como ""estructuras discretas"" o ""matemática discreta"", mientras que otros pueden integrar estos temas en cursos de programación, algoritmos, y / o la inteligencia artificial. Las combinaciones de estos enfoques también son frecuentes (por ejemplo, que cubre muchos de estos temas en un solo curso introductorio centrado y que cubre los temas restantes en los cursos de tópicos más avanzados).

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.4.1 Funciones, relaciones y conjuntos (Pág. 29) | | | No |
| 2.4.2 Lógica básica (Pág. 29) | | | No |
| 2.4.3 Técnicas de demostración (Pág. 30) | 1 | | No |
| 2.4.4 Fundamentos de conteo (Pág. 30) | | | No |
| 2.4.5 Árboles y Grafos (Pág. 31) | 1 | | No |
| 2.4.6 Probabilidad Discreta (Pág. 32) | 2 | | No |

### 2.4.1. DS/Funciones, relaciones y conjuntos

**Temas:**
**Core Tier1**

- Conjuntos: 1. Diagramas de Venn 2. Unión, intersección, complemento 3. Producto Cartesiano 4. Potencia de conjuntos 5. Cardinalidad de Conjuntos finitos
- Relaciones: 1. Reflexividad, simetria, transitividad 2. Relaciones equivalentes, ordenes parciales
- Funciones: 1. Suryecciones, inyecciones, biyecciones 2. Inversas 3. Composición

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Explicar con ejemplos la terminología básica de funciones, relaciones y conjuntos [Familiarizarse]
2. Realizar las operaciones asociadas con conjuntos, funciones y relaciones [Usar]
3. Relacionar ejemplos prácticos para conjuntos funciones o modelos de relación apropiados e interpretar la asociación de operaciones y terminología en contexto [Evaluar]

### 2.4.2. DS/Lógica básica

**Temas:**
**Core Tier1**

- 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: 1. Cuantificación universal y existencial
- Limitaciones de la lógica proposicional y de predicados (ej. problemas de expresividad)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Convertir declaraciones lógicas desde el lenguaje informal a expresiones de lógica proposicional y de predicados [Usar]
2. Aplicar métodos formales de simbolismo proposicional y lógica de predicados, como el cálculo de la validez de formulas y cálculo de formas normales [Usar]
3. Usar reglas de inferencia para construir demostraciones en lógica proposicional y de predicados [Usar]

4. 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 [Usar]

5. Aplicar demostraciones de lógica formal y/o informal, pero rigurosa, razonamiento lógico para problemas reales, como la predicción del comportamiento de software o solución de problemas tales como rompecabezas [Usar]

6. Describir las fortalezas y limitaciones de la lógica proposicional y de predicados [Familiarizarse]

### 2.4.3. DS/Técnicas de demostración (1 horas Core-Tier1)

**Temas:**
**Core Tier1**
- 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.

**Core Tier2**
- Conjuntos bien ordenados.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Identificar la técnica de demostración utilizada en una demostración dada [Familiarizarse]

2. Describir la estructura básica de cada técnica de demostración (demostración directa, demostración por contradicción e inducción) descritas en esta unidad [Usar]

3. Aplicar las técnicas de demostración (demostración directa, demostración por contradicción e inducción) correctamente en la construcción de un argumento solido [Usar]

4. Determine que tipo de demostración es la mejor para un problema dado [Evaluar]

5. Explicar el paralelismo entre ideas matemáticas y/o inducción estructural para la recursión y definir estructuras recursivamente [Evaluar]

6. Explicar la relación entre inducción fuerte y débil y dar ejemplos del apropiado uso de cada uno [Evaluar]

**Core-Tier2:**

7. Enunciar el principio del buen-orden y su relación con la inducción matemática [Familiarizarse]

### 2.4.4. DS/Fundamentos de conteo

**Temas:**
**Core Tier1**
- Técnicas de Conteo: 1. Conteo y cardinalidad de un conjunto 2. Regla de la suma y producto 3. Principio de inclusión-exclusión 4. Progresión geométrica y aritmética
- Principio de las casillas.
- Permutaciones y combinaciones: 1. Definiciones básicas 2. Identidad de Pascal 3. Teorema del binomio
- Resolviendo relaciones de recurrencia: 1. Un ejemplo de una relación de recurrencia simple, como los números de Fibonacci 2. Otras ejemplos, mostrando una variedad de soluciones

- Aritmetica modular basica

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Aplicar argumentos de conteo, incluyendo las reglas del producto y de la suma, principio de inclusión-exclusión y progresiones aritméticas/geométricas [Usar]

2. Aplicar el principio de las casillas en el contexto de una demostración formal [Usar]

3. Calcular permutaciones y combinaciones en un conjunto, e interpreta su significado en el contexto de una aplicación en particular [Usar]

4. 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) [Usar]

5. Resolver una variedad de relaciones de recurrencia básicas [Usar]

6. Analizar un problema para determinar las relaciones de recurrencia implícitas [Usar]

7. Realizar cálculos que involucran aritmética modular [Usar]

### 2.4.5. DS/Árboles y Grafos (1 horas Core-Tier1)

Ver también: Algoritmos y Estructuras de Datos fundamentales, especialmente con estratégias de recorrido en grafos.

**Temas:**
**Core Tier1**

- Árboles. 1. Propiedades 2. Estrategias de recorrido
- Grafos no dirigidos
- Grafos dirigidos
- Grafos ponderados

**Core Tier2**

- Arboles de expansion/bosques.
- Isomorfismo en grafos.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. 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]

2. Demostrar diversos métodos de recorrer árboles y grafos, incluyendo recorridos pre, post e inorden de árboles [Usar]

3. 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 [Usar]

4. Demuestrar como los conceptos de grafos y árboles aparecen en estructuras de datos, algoritmos, técnicas de prueba (inducción estructurada), y conteos [Usar]

**Core-Tier2:**

5. Explicar como construir un árbol de expansión de un grafo [Usar]

6. Determinar si dos grafos son isomorfos [Usar]

### 2.4.6. DS/Probabilidad Discreta (2 horas Core-Tier1)

**Temas:**
**Core Tier1**

- Espacio de probabilidad finita, eventos.
- Axiomas de Probabilidad y medidas de probabilidad.
- Probabilidad condicional, Teorema de Bayes.
- Independencia.
- Variables enteras aleatorias (Bernoulli, binomial).
- Esperado, Linearidad del esperado.

**Core Tier2**

- Varianza.
- Independencia Condicional.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Calcular las probablidades de eventos y el valor esperado de variables aleatorias para problemas elementales como en los juegos de azar [Usar]
2. Distinguir entre eventos dependientes e independientes [Usar]
3. Identificar un caso de la distribución binomial y calcula la probabilidad usando dicha distribución [Usar]
4. Aplicar el teorema de Bayes para determinar las probabilidades condicionales en un problema [Usar]
5. Aplicar herramientas de probabilidades para resolver problemas como el análisis de caso promedio en algoritmos o en el análisis de hash [Usar]

**Core-Tier2:**

6. Calcular la varianza para una distribución de probabilidad dada [Usar]
7. Explicar como los eventos que son independientes pueden ser condicionalmente dependientes (y vice versa) Identificar ejemplos del mundo real para estos casos [Usar]

## 2.5. Gráficos y Visualización (GV)

*Computer Graphics* es el término comúnmente utilizado para describir la generación y la manipulación de imagenes por medio de la computadora. Es la ciencia de la que permite la comunicación visual a través de la computación. Sus usos incluyen dibujos animados, efectos especiales de cine, videojuegos, imágenes médicas, de ingeniería, así como científicos, información y visualización de conocimiento. Tradicionalmente, los gráficos a nivel de pregrado se ha centrado en la representación, el álgebra lineal, y los enfoques fenomenológicos. Más recientemente, la atención se ha comenzado a incluir la física, integración numérica, escalabilidad y hardware de propósito específico. Para que los estudiantes se conviertan en expertos en el uso y generación de gráficos por ordenador, muchos temas específicos de la implementación deben ser abordados, tales como formatos de archivos, interfaces de hardware y las interfaces de programa de aplicación. Estos temas cambian rápidamente, y la descripción que sigue a los intentos para evitar que sean excesivamente prescriptivos sobre ellos. El área abarcada por Gráficos y visualización se divide en varios campos relacionados entre sí: 1. Fundamentos: La infografía depende de una comprensión de cómo los seres humanos utilizan la visión para percibir la información y cómo la información se puede representar en un dispositivo de visualización. Cada científico de la computación debe tener cierta comprensión de dónde y cómo los gráficos se pueden aplicarse según corresponda, así asthe procesos fundamentales involucrados en la prestación de visualización. 2. Modelado artículo: Información que se muestra debe ser codificada en la memoria del ordenador en alguna forma, a menudo en forma de una especificación matemática de la forma y la forma. 3. Representación: Representación es el proceso de visualización de la información contenida

en un modelo. 4. Animación: La animación es la prestación de una manera que hace que las imágenes parecen moverse y la síntesis o la adquisición de las variaciones en el tiempo de los modelos. 5. Visualización: El campo de la visualización busca determinar y presentar estructuras y relaciones en conjuntos de datos correlacionados subyacentes entre una amplia variedad de áreas de aplicación. El objetivo primordial de la presentación debe ser comunicar la información en un conjunto de datos con el fin de mejorar la comprensión 6. Geometría Computacional: Geometría Computacional es el estudio de los algoritmos que se indican en términos de geometría.

Gráficos y visualización está relacionado con la visión artificial y procesamiento de imágenes, que se encuentran en el área de Inteligencia Artificial (AI) y algoritmos tales como la geometría computacional, que se encuentran en el área Algoritmos y Complejidad (AL). Los temas de la realidad virtual son encontrados en el el área de conocimiento Interacción Humano-Computador (HCI).

Esta descripción asume que los estudiantes están familiarizados con los conceptos fundamentales de la representación de los datos, la abstracción y la implementación del programa.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.5.1 Conceptos Fundamentales (Pág. 33) | 1 | | No |
| 2.5.2 Rendering Básico (Pág. 34) | | | No |
| 2.5.3 Modelado Geométrico (Pág. 35) | | | No |
| 2.5.4 Renderizado Avanzado (Pág. 36) | | | No |
| 2.5.5 Animación por computadora (Pág. 37) | | | No |
| 2.5.6 Visualización (Pág. 37) | | | No |

### 2.5.1. GV/Conceptos Fundamentales (1 horas Core-Tier1)

Para casi todos los científicos de la computación y desarrolladores de software, una comprensión de cómo los seres humanos interactúan con las máquinas es esencial. Mientras que estos temas pueden ser cubiertos en un curso de pregrado de gráficos estándar, también pueden ser cubiertos en los cursos de ciencia de la computacióny programación introductorias. Parte de nuestra motivación para la inclusión de los modos inmediatos y retenidas es que estos modos son análogos a votación vs programación orientada a eventos. Esta es una cuestión fundamental en la ciencia de la computación: ¿ Hay un objeto botón o existe sólo la visualización de un botón en la pantalla? Tenga en cuenta que la mayoría de los resultados en esta sección se encuentran en el nivel de conocimiento y muchos de estos temas se revisan con mayor profundidad en las secciones posteriores.

**Temas:**

**Core Tier1**

- 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.

**Core Tier2**

- 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.

**Electivo**

- Almacenamiento doble.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Identificar usos comunes de presentaciones digitales de humanos (por ejemplo, computación gráfica,sonido) [Familiarizarse]

2. 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]
3. Explicar cómo las limitaciones en la percepción humana afectan la selección de la representación digital de señales analógicas [Usar]
4. Cronstruir una interfaz de usuario sencilla usando una API estándar [Familiarizarse]
5. 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]

**Core-Tier2:**

6. Describir modelos de color y su uso en los dispositivos de visualización de gráficos [Familiarizarse]
7. 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]

**Elective:**

8. 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]
9. Describir cómo el doble buffer puede eliminar el parpadeo de la animación [Familiarizarse]

### 2.5.2. GV/Rendering Básico

En esta sección se describe la prestación básica y las técnicas gráficas fundamentales que casi todos los cursos de pregrado en los gráficos se cubre y que son esenciales para el estudio adicional en gráficos. Muestreo y anti-aliasing están relacionados con el efecto de la digitalización y aparecen en otras áreas de la computación, por ejemplo, en el muestreo de audio.

**Temas:**

**Electivo**

- 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.
- Renderización con una API basada en shader.
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. 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]
2. Describir la tubería básica gráficos y cómo el factor de representación va hacia adelante y atrás en esta [Familiarizarse]
3. Crear un programa para visualizar modelos 3D de imagenes gráficas simples [Usar]

4. Derivar la perspectiva lineal de triángulos semejantes por conversión de puntos (x,y,z) a puntos (x/z, y/z, 1) [Usar]
5. Obtener puntos en 2-dimensiones y 3-dimensiones por aplicación de transformaciones afín [Usar]
6. 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]
7. Contrastar la renderización hacia adelanate *forward* y hacia atras *backward* [Evaluar]
8. Explicar el concepto y las aplicaciones de mapeo de texturas, muestreo y el *anti-aliasing* [Familiarizarse]
9. Explicar la dualidad de rastreo de rayos/rasterización para el problema de visibilidad [Familiarizarse]
10. Implementar simples procedimientos que realicen la transformación y las operaciones de recorte de imágenes simples en 2 dimensiones [Usar]
11. Implementar un sencillo renderizador en tiempo real utilizando una API de rasterización (por ejemplo, OpenGL) utilizando buffers de vértices y *shaders* [Usar]
12. Comparar y contrastar las diferentes técnicas de renderización [Evaluar]
13. Calcular las necesidades de espacio en base a la resolución y codificación de color [Evaluar]
14. Calcular los requisitos de tiempo sobre la base de las frecuencias de actualización, técnicas de rasterización [Evaluar]

### 2.5.3. GV/Modelado Geométrico

La visualización tiene fuertes lazos con el área de conocimiento de Interacción humano computador (HCI), así como Ciencias de la Computación (CN). Los lectores deben consultar la HCI y CN KAs para consultar otros temas relacionados con las evaluaciones de población y de interfaz de usuario.

**Temas:**

**Electivo**

- 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.
- Graftales, referenciados con Lenguajes de Programación (gramática a imágenes generadas)
- Modelos deformables de forma libre y elásticamente deformables.
- Subdivisión de superficies.
- Modelado multiresolución.
- Reconstrucción.
- Representación de Geometría Sólida Constructiva (GSC)

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Representar curvas y superficies utilizando formas tanto implícitas y paramétricas [Usar]
2. Crear modelos poliédrico simples por teselación de superficies [Usar]
3. Generar una representación de malla de una superficie implícita [Usar]

4. Generar un modelo fractal o terreno usando un método de procedimiento [Usar]

5. Generar una malla de un conjunto de puntos adquiridos por un scaner laser [Usar]

6. Construct modelos de geometría sólida constructiva a partir de simples primitivas, tales como cubos y superficies cuádricas [Usar]

7. Contrastar métodos de modelización con respecto a espacio y tiempo de complejidad y calidad de imagen [Evaluar]

### 2.5.4. GV/Renderizado Avanzado

**Temas:**
**Electivo**

- Soluciones y aproximaciones hacia la ecuación de renderizado, por ejemplo: 1. Distribución de *ray tracing* y *path tracing* 2. Mapeo de fotones 3. *Path tracing* bidireccional 4. *Reyes rendering* (micropoligono) 5. *Metropolis light transport*
- 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.
- Distribución función de dispersión bidireccional (BSDF) teoría y microfacets.
- Disperción de la Superficie.
- Recursos de luz de Área.
- Almacenamiento jerárquico por profundidad.
- El campo de luz, representación de imágenes.
- Renderizado no fotorealistico.
- Arquitectura del GPU.
- Sistemas visuales humanos incluida la adaptación a la luz, la sensibilidad al ruido, y la fusión de parpadeo.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Demostrar como un algoritmo calcula una solución a la ecuación de renderización [Evaluar]

2. Demostrar las propiedades de un algoritmo de renderización, por ejemplo, completo, consistente, e imparcial [Evaluar]

3. Analizar las demandas de ancho de banda y cálculo de un simple algoritmo [Evaluar]

4. 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]

5. Discutir como una técnica artística particular puede ser implementada en un renderizador [Familiarizarse]

6. Explicar como reconocer las técnicas gráficas usadas para crear una imagen en particular [Familiarizarse]

7. Implementar cualquiera de las técnicas gráficas especificadas utilizando un sistema gráfico primitivo a nivel de píxel individual [Usar]

8. Implementar un trazado de rayos(*ray tracer*) para escenas una simple Función de Distribución de Reflectancia Bidireccional(*BRDF*) por ejemplo Phong's, además de la reflexión y la refracción [Usar]

### 2.5.5. GV/Animación por computadora

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Calcular la localización y orientación de partes de un modelo usando un enfoque de cinemática hacia delante [Usar]
2. Calcular la orientación de partes articuladas de un modelo de una localización y orientación usando un enfoque de cinemática inversa [Usar]
3. Describir las ventajas y desventajas de diferentes representaciones de rotación [Evaluar]
4. Implementar el método de interpolación *spline* para producir las posiciones y orientaciones en medio [Usar]
5. 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]
6. 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]
7. Usar el software de animación común para construir formas orgánicas simples usando *metaball* y el esqueleto [Usar]

### 2.5.6. GV/Visualización

**Temas:**
**Electivo**

- Visualización de campos escalares de 2D/3D: mapeado de color, isosurfaces.
- Representación de datos Direct volume: ray-casting, funciones de transferencia, segmentación.
- Visualización de: 1. Campos de vector y flujo de datos 2. Datos que varian en el tiempo 3. High-dimensional data: reducción de dimensiones, coordenadas paralelas 4. Datos Non-espaciales: multi-variados, estructurados en árbol/grafo, texto
- Fundamentos perceptuales y cognitivos que conducen abstracciones visuales.
- Diseño de visualización.
- Evaluación de métodos de visualización.
- Aplicaciones de visualización.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describir los algitmos básicos para visualización escalar y vectorial [Familiarizarse]
2. Describir las ventajas y desventajas de los algoritmos de visualización en terminos de precisión y desempeño [Evaluar]

3. Proponer un diseño de visualización adecuado para una combinación particular de características de datos y tareas de la aplicación [Evaluar]

4. Analizar la eficacia de una visualización dada para una tarea en particular [Evaluar]

5. Diseñar un proceso para evaluar la utilidad de un algoritmo de visualización o del sistema [Evaluar]

6. Reconocer una variedad de aplicaciones de visualización incluyendo representaciones de datos científicos, médicos y matemática; visualización de flujo; y análisis espacial [Familiarizarse]

## 2.6. Interacción Humano-Computador (HCI)

La interacción humano-computador (Human-computer interaction HCI) tiene que ver con el diseño de las interacciones entre las actividades humanas y los sistemas computacionales que los apoyan y con la construcción de interfaces para permitirse esas interacciones.

La interacción entre los usuarios y artefactos computacionales se produce en una interfaz que incluye tanto de software como de hardware. Impactos de diseño de interfaz de esta manera el ciclo de vida del software, ya que debe darse al inicio; el diseño e implementación de la funcionalidad central pueden influir, para bien o para mal, en la interfaz de usuario.

Debido a que se trata de personas, así como los sistemas computacionales, como un área de conocimiento, HCI exige un análisis de las cuestiones culturales, sociales, organizativas, cognitivas y perceptivas. En consecuencia, se basa en una variedad de tradiciones disciplinarias, incluyendo la psicología, la ergonomía, la Ciencia de la Computación, el diseño gráfico y de producto, la antropología y la ingeniería.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.6.1 Fundamentos (Pág. 38) | | | No |
| 2.6.2 Diseño de Interacción (Pág. 39) | 4 | | No |
| 2.6.3 Programación de Sistemas Interactivos (Pág. 39) | | | No |
| 2.6.4 Diseño y Testing centrados en el usuario (Pág. 40) | | | No |
| 2.6.5 Nuevas Tecnologías Interactivas (Pág. 41) | | | No |
| 2.6.6 Colaboración y Comunicación (Pág. 41) | | | No |
| 2.6.7 Métodos estadísticos para HCI (Pág. 42) | | | No |
| 2.6.8 Factores Humanos y seguridad (Pág. 42) | | | No |
| 2.6.9 HCI orientada al diseño (Pág. 43) | | | No |
| 2.6.10 Realidad virtual y aumentada mezcladas (Pág. 43) | | | No |

### 2.6.1. HCI/Fundamentos

Motivación: Para los usuarios finales, la interfaz es el sistema. Así que el diseño en este ámbito debe ser centrado en la interacción y en el hombre. Los estudiantes necesitan un repertorio diferente de las técnicas para hacer frente a esto que está previsto en el plan de estudios en otros lugares.

**Temas:**

**Core Tier1**

- Contextos para IHC (cualquiera relacionado con una interfaz de usuario, p.e., página web, aplicaciones de negocios, aplicaciones móviles y juegos)
- Procesos para desarrollo centrado en usuarios, p.e., enfoque inicial en usuarios, pruebas empíricas, diseño iterativo.
- Diferentes medidas para evaluación, p.e., utilidad, eficiencia, facilidad de aprendizaje, satisfacción de usuario.
- Heurística de usabilidad y los principios de pruebas de usabilidad.
- Capacidades físicas que informan diseño de interacción, p.e. percepción del color, ergonomía.
- Modelos cognoscitivos que informan diseño de interacciones, p.e., atención, percepción y reconocimiento, movimiento, memoria, golfos de expectativa y ejecución.

- Modelos sociales que informan el diseño de interacción, e.g., cultura, comunicación, redes y organizaciones.
- Principios del buen diseño y buenos diseñadores; ventajas y desventajas de ingeniería.
- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Discutir por qué el desarrollo de software centrado en el hombre es importante [Familiarizarse]
2. Resumir los preceptos básicos de la interacción psicológica y social [Familiarizarse]
3. Desarrollar y usar un vocabulario conceptual para analizar la interación humana con el software: disponibilidad, modelo conceptual, retroalimentación, y demás [Usar]
4. 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 [Usar]
5. Crear y dirigir una simple pruebga de usabilidad para una aplicación existente de software [Evaluar]

### 2.6.2. HCI/Diseño de Interacción (4 horas Core-Tier1)

Motivación: Estudiantes de CS necesitan un conjunto mínimo de métodos y herramientas bien establecidas para llevar a la interfaz en la construcción.
**Temas:**
**Core Tier2**

- Principios de interfaces gráficas de usuario (GUIs)
- Elementos de diseño visual (disposición, color, fuentes, etiquetado)
- Análisis de tareas, incluidos los aspectos cualitativos de la generación de modelos de análisis de tareas.
- Prototipos de baja fidelidad (papel)
- Técnicas de evaluación cuantitativa ej. evaluación Keystroke-level.
- Ayuda y documentación.
- Manejo de fallas humanas/sistema.
- Estándares de interfaz de usuario.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Para un grupo de usuarios determinado, realizar y documentar un análisis de sus necesidades [Evaluar]
2. Crear una aplicación simple, junto con la ayuda y la documentación, que soporta una interfaz gráfica de usuario [Usar]
3. Llevar a cabo una evaluación cuantitativa y discutir / informar sobre los resultados [Usar]
4. Discutir al menos un standard nacional o internacional de diseño de interfaz de usuario [Familiarizarse]

### 2.6.3. HCI/Programación de Sistemas Interactivos

Motivación: Para tener una visión centrada en la experiencia de usuario de desarrollo de software y luego cubrir enfoques y tecnologías para hacer que eso suceda.
**Temas:**
**Electivo**

- Patrones de arquitectura de software. Ej Modelo Vista Controlador, Objetos de comando, online, offline.

- Patrones de diseño de Interacción: jerarquía visual, distancia navegacional.
- Manejo de eventos e interacción de usuario.
- Manejo de geometría.
- Elección de estilos de interacción y técnicas de interacción.
- Presentación de información: navegación, representación, manipulación.
- Técnicas de animación de interfaz (ej. grafo de escena)
- Clases Widget y bibliotecas.
- Bibliotecas modernas de GUI (ej. iOS, Android, JavaFX) constructores de GUI y entornos de programación UI.
- Especificación declarativa de Interfaz: Hojas de Estilo y DOMs.
- Aplicaciones dirigidas a datos ( Páginas web respaldadas por base de datos)
- Diseño multiplataforma.
- Diseño para dispositivos con restricción de recursos (ej. dispositivos pequeños, móviles)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar la importancia del controlador Modelo-Vista para la programación de la interfaz [Familiarizarse]
2. Crear una aplicación con una moderna interfaz gráfica de usuario [Usar]
3. Identificar puntos comunes y las diferencias en las *UIs* a través de diferentes plataformas [Familiarizarse]
4. Explicar y utilizar los conceptos de programación de GUI: la gestión de eventos, gestión de distribución basado en restricciones, etc [Familiarizarse]

### 2.6.4. HCI/Diseño y Testing centrados en el usuario

Motivación: Una exploración de técnicas para garantizar que los usuarios finales sean totalmente en cuenta en todas las etapas del proceso de diseño, desde su creación hasta su implementación.
**Temas:**
**Electivo**

- 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.
- Técnicas de creación de prototipos y herramientas, ej.bosquejos, *storyboards*, prototipos de baja fidelidad, esquemas de página.
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar cómo el diseño centrado en el usuario complementa a otros modelos de proceso software [Familiarizarse]
2. Utilizar *lo-fi* (baja fidelidad) técnicas de prototipado para recopilar y reportar, las respuestas del usuario [Usar]
3. Elegir los métodos adecuados para apoyar el desarrollo de una específica interfaz de usuario [Evaluar]

4. Utilizar una variedad de técnicas para evaluar una interfaz de usuario dada [Evaluar]

5. Comparar las limitaciones y beneficios de los diferentes métodos de evaluación [Evaluar]

### 2.6.5. HCI/Nuevas Tecnologías Interactivas

Motivación: Dado que las tecnologías evolucionan, se hacen posibles nuevos estilos de interacción. Esta unidad de conocimiento debe considerarse extensible, para realizar un seguimiento de las tecnologías emergentes.

**Temas:**

**Electivo**

- Elección de estilos de interacción y técnicas de interacción.
- Representación de información a usuarios; navegación, representación, manipulación.
- Enfoques para el diseño, implementación y evaluación de la interacción sin mouse 1. Interfaces táctiles y multitáctiles. 2. Interfaces compartidas, incorporadas y grandes 3. Nuevas modalidades de entrada (tales como datos de sensores y localización) 4. Nuevas ventanas, por ejemplo, iPhone, Android 5. Reconocimiento de voz y procesamiento del lenguaje natural 6. Interfaces utilizables y tangibles 7. Interacción persuasiva y emoción 8. Tecnologías de interacción ubicuas y contextuales (Ubicomp) 9. Inferencia bayesiana (por ejemplo, texto predictivo, orientación guiada) 10. Visualización e interacción de ambiente / periféricos

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Describe cuando son adecuadas las interfaces sin uso de ratón [Familiarizarse]

2. Comprende las posibilidades de interacción que van más allá de las interfaces de ratón y puntero [Familiarizarse]

3. Discute las ventajas (y desventajas) de las interfaces no basadas en ratón [Evaluar]

### 2.6.6. HCI/Colaboración y Comunicación

De motivación: No sólo los usuarios de interfaces de ordenador de apoyo en el logro de sus metas individuales, sino también en su interacción con los demás, ya sea centrada en la tarea (trabajo o juego) o en tareas fuera de foco (redes sociales).

**Temas:**

**Electivo**

- La comunicación asíncrona en grupo, por ejemplo, el correo electrónico, foros, redes sociales.
- Comunicación sincrónica en grupo, por ejemplo, las salas de chat, conferencias, juegos en línea.
- 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

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Describir la diferencia entre la comunicación sincrónica y asincrónica [Familiarizarse]

2. Comparar los problemas de IHC en la interacción individual con la interacción del grupo [Evaluar]

3. Discuta varias problemas de interés social planteados por el software colaborativo [Familiarizarse]

4. Discutir los problemas de IHC en software que personifica la intención humana [Familiarizarse]

### 2.6.7. HCI/Métodos estadísticos para HCI

Motivación: Mucho trabajo HCI depende del uso adecuado, la comprensión y aplicación de la estadística. Este conocimiento es a menudo ayuda en manos de los estudiantes que se incorporan al campo de la psicología, pero menos común en los estudiantes con un fondo CS.
**Temas:**
**Electivo**

- Pruebas T.
- Análisis de Varianza (ANOVA).
- La asignación al azar (no paramétrica) pruebas, dentro vs. entre sujetos diseño.
- Calculando el efecto producto del tamaño.
- Análisis exploratorio de datos.
- Presentación de datos estadísticos.
- Combinando resultados cuantitativos y cualitativos.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar los conceptos estadísticos básicos y sus áreas de aplicación [Familiarizarse]
2. Extraer y articular los argumentos estadísticos utilizados en papers que informan cuantitativamente los estudios de usuarios [Usar]
3. Diseñar un estudio de usuarios que va a dar resultados cuantitativos [Usar]
4. Llevar a cabo e informar sobre un estudio que utiliza evalución cualitativa y cuantitativa [Usar]

### 2.6.8. HCI/Factores Humanos y seguridad

Motivación: Diseño eficaz interfaz requiere un conocimiento básico de la psicología de la seguridad. Muchos de los ataques no tienen una base tecnológica, pero explotan propensiones humanas y vulnerabilidades. "Sólo los aficionados atacan máquinas; profesionales se dirigen las personas" (Bruce Schneier, https://www.schneier.com/blog/archives/2013/03/phishing_has_go.h.)
**Temas:**
**Electivo**

- Psicologia aplicada y politicas de seguridad
- Seguridad de economía.
- Entornos regulatorios, responsabilidad, riesgo y autodeterminación.
- Organización de vulnerabilidades y amenazas.
- Diseño de usabilidad y seguridad.
- Pretexto, suplantación y fraude, p.e., phishing y phishing focalizado.
- Confianza, privacidad y engaño.
- Autenticación biometrica.
- Manejo de Identidad.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar los conceptos de *phishing* y *spear phishing* y cómo reconocerlos [Familiarizarse]
2. Describir los problemas de confianza en el diseño de interfaz con un ejemplo de un sistema de confianza alto y bajo [Evaluar]
3. Diseñar una interfaz de usuario para un mecanismo de seguridad [Evaluar]
4. Explicar el concepto de gestión de la identidad y su importancia [Familiarizarse]
5. Analizar una política y/o procedimientos para demostrar si consideran o no se consideran, los factores humanos de la seguridad [Usar]

### 2.6.9. HCI/HCI orientada al diseño

Motivación: Algunos planes de estudio va a querer enfatizar la comprensión de las normas y valores de HCI trabajar como surge de y desplegado en contextos históricos, culturales y disciplinarias específicas.

**Temas:**

**Electivo**

- Estilos y perspectivas intelectuales para la tecnología y sus interfaces.
- Consideración de IHC como una disciplina de diseño: 1. Sketching 2. Diseño participativo
- Critically reflective HCI: 1. Práctica Crítica Técnica 2. Tecnologías para activismo político 3. Filosofía de la experiencia de usuario 4. Etnografía y etnometodología
- Dominios indicativos de aplicación: 1. Sostenibilidad 2. *Arts-informed computing*

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Explicar que significa que "La IHC es una disciplina orientada al diseño" [Familiarizarse]
2. Detallar los procesos de diseño apropiados para orientaciones específicas de diseño [Familiarizarse]
3. Aplicar una varidad de métodos de diseño para un problema determinado [Usar]

### 2.6.10. HCI/Realidad virtual y aumentada mezcladas

Motivación: Un examen detallado de los componentes de interfaz necesarios para la creación y el desarrollo de entornos inmersivos, especialmente juegos.

**Temas:**

**Electivo**

- Salida: 1. Sonido 2. Visualización estereoscópica 3. Forzar la simulación de retroalimentación, dispositivos hápticos
- Entrada del usuario: 1. Visor y seguimiento de objetos 2. Pose y gesto de reconocimiento 3. Los acelerómetros 4. marcadores de referencia 5. Los problemas de interfaz de usuario de artículo
- Modelamiento y representación física: 1. Simulación física: detección de colisiones y respuesta, animación 2. *Visibility computation* 3. Representación sensitiva al tiempo, múltiples niveles de detalle (LOD)
- Arquitectura de Sistemas: 1. Motores de Juego 2. Relidad Aumentada móvil 3. Simuladores de vuelo 4. CAVEs 5. Imágenes médicas
- Redes: 1. p2p, cliente-servidor, dead reckoning, encriptación, sincronización 2. Colaboración distribuida

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Describir el modelo óptico realizado por un sistema de gráficos por computadora para sintetizar una visión estereoscópica [Familiarizarse]
2. Describir los principios de las diferentes tecnologias de seguimiento de espectador [Familiarizarse]
3. Describir las diferencias entre geometria y realidad virtual basada en imágenes [Familiarizarse]
4. Describir los casos de la acción de sincronización de un usuario y la consistencia de los datos en un entorno de red [Familiarizarse]
5. Determinar los requerimientos básicos en interfaz, software, hardware, y cofiguraciones de software de un sistema VR para una aplicación específica [Usar]
6. Describir varios usos posibles para los motores de juegos, incluyendo su potencial y sus limitaciones [Familiarizarse]

## 2.7. Aseguramiento y Seguridad de la Información (IAS)

El área de Aseguramiento de la Información y de Seguridad se añade al cuerpo de conocimiento en reconocimiento a la dependencia del mundo en tecnología de la información y su papel fundamental en la educación de ciencia de la computación. Aseguramiento y seguridad de la información, como un dominio, es el conjunto de controles y procesos tanto técnicos como de política destinadas a proteger y defender los sistemas de información y de información garantizando su confidencialidad, integridad y disponibilidad, suministrándoles autenticación y el no rechazo. El concepto de garantía también lleva una certificación de que los procesos y los datos actuales y anteriores son válidos. Se necesitan tanto los conceptos de aseguramiento y de seguridad para garantizar una perspectiva completa. Aseguramiento de la información y la educación de seguridad y, a continuación, incluye todos los esfuerzos para preparar un grupo de trabajo con los conocimientos necesarios, habilidades y capacidades para proteger nuestros sistemas de información y dar fe de la seguridad del pasado y actual estado de los procesos y datos. La importancia de los conceptos de seguridad y los temas se ha convertido en un requisito básico de la disciplina Ciencia de la Computación, al igual que la importancia de los conceptos de rendimiento ha sido durante muchos años.

El área de conocimiento de Aseguramiento y Seguridad de la Información es único entre el conjunto de áreas que aquí se presenta, dada la manera en que los temas son omnipresentes en toda otras áreas de conocimiento. Los temas que constarán únicamente en el área de IAS se presentan en esta sección; otros temas se observan y tienen referencias cruzadas. En esta área muchos temas son representados con sólo 9 horas de cobertura Core-Tier1 y Tier2. Esto se equilibra con el nivel de dominio principalmente a nivel familiaridad y la cobertura más profundizada distribuido en la áreas de conocimiento donde estas se aplican. La amplia aplicación de los conceptos de la esta área (63,5 horas) en todas las demás áreas ofrece la profundidad y cobertura de dominio necesario para un estudiante de ciencia de la computación de pregrado.

El área de IAS se muestra en dos grupos: (1) conceptos donde la profundidad es exclusiva de esta área y (2) los temas de IAS que se integran en otras áreas que reflejan temas naturalmente implícitos o expresadas con un papel importante en los conceptos y temas de seguridad.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.7.1 Fundamentos y Conceptos en Seguridad (Pág. 44) | 2 | | No |
| 2.7.2 Principios de Diseño Seguro (Pág. 45) | 2 | | No |
| 2.7.3 Programación Defensiva (Pág. 46) | 2 | | No |
| 2.7.4 Ataques y Amenazas (Pág. 47) | 1 | | No |
| 2.7.5 Seguridad de Red (Pág. 47) | 2 | | No |
| 2.7.6 Criptografía (Pág. 48) | 1 | | No |
| 2.7.7 Seguridad en la Web (Pág. 49) | | | No |
| 2.7.8 Seguridad de plataformas (Pág. 50) | | | No |
| 2.7.9 Política de Seguridad y Gobernabilidad (Pág. 50) | | | No |
| 2.7.10 Investigación digital (Digital Forensics) (Pág. 51) | | | No |
| 2.7.11 Seguridad en Ingeniería de Software (Pág. 52) | | | No |

### 2.7.1. IAS/Fundamentos y Conceptos en Seguridad (2 horas Core-Tier1)

**Temas:**
**Core Tier1**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Analizar las ventajas y desventajas de equilibrar las propiedades clave de seguridad(Confidenciabilidad, Integridad, Disponibilidad) [Usar]

2. 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]
3. Explicar los conceptos de autentificación, autorización, control de acceso [Familiarizarse]
4. Explicar el concepto de confianza y confiabilidad [Familiarizarse]
5. 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]

### 2.7.2. IAS/Principios de Diseño Seguro (2 horas Core-Tier1)

**Temas:**
**Core Tier1**
- 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.

**Core Tier2**
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Describir el principio de privilegios mínimos y el aislamiento que se aplican al diseño del sistema [Familiarizarse]
2. Resumir el principio de prueba de fallos y negar por defecto [Familiarizarse]
3. Discutir las implicaciones de depender de diseño abierto o secreto de diseño para la seguridad [Familiarizarse]
4. Explicar los objetivos de seguridad de datos de extremo a extremo [Familiarizarse]
5. Discutir los beneficios de tener múltiples capas de defensas [Familiarizarse]
6. Por cada etapa en el ciclo de vida de un producto, describir que consideraciones de seguridad deberian ser evaluadas [Familiarizarse]
7. Describir el costo y ventajas y desventajas asociadas con el diseño de seguridad de un producto. [Familiarizarse]

**Core-Tier2:**

8. Describir el concepto de mediación y el principio de mediación completa [Familiarizarse]
9. Conocer los componentes estándar para las operaciones de seguridad, en lugar de reinventar las operaciones fundamentales [Familiarizarse]
10. 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]
11. Discutir la importancia de la usabilidad en el diseño de mecanismos de seguridad [Familiarizarse]

12. Describir problemas de seguridad que surgen en los límites entre varios componentes [Familiarizarse]

13. Identificar los diferentes roles de mecanismos de prevención y mecanismos de eliminación/disuación [Familiarizarse]

### 2.7.3. IAS/Programación Defensiva (2 horas Core-Tier1)

Temas en la programación defensiva en general no se piensa en el aislamiento, pero aplicados a otros temas en particular en SDF, SE y PD Áreas de Conocimiento.

**Temas:**

**Core Tier1**

- 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. 1. Desbordamiento de búfer 2. Errores enteros 3. Inyección SQL 4. Vulnerabilidad XSS
- Las condiciones de carrera.
- Manejo correcto de las excepciones y comportamientos inesperados.

**Core Tier2**

- Uso correcto de los componentes de terceros.
- Desplegar eficazmente las actualizaciones de seguridad.

**Electivo**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Explicar por que la validación de entrada y desinfección de datos es necesario en el frente del control contencioso del canal de entrada [Familiarizarse]

2. 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++ [Familiarizarse]

3. Clasificar los errores de validación de entrada común, y escribir correctamente el código de validación de entrada [Usar]

4. 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]

5. Demostrar la identificación y el manejo elegante de las condiciones de error [Usar]

**Core-Tier2:**

6. Explique los riesgos de mal uso de las interfaces con código de terceros y cómo utilizar correctamente el código de terceros [Familiarizarse]

7. Discutir la necesidad de actualizar el software para corregir las vulnerabilidades de seguridad y la gestión del ciclo de vida de la corrección [Familiarizarse]

**Elective:**

8. Listar ejemplos de flujos de información directa e indirecta [Familiarizarse]

9. Explicar la función de números aleatorios en la seguridad, más allá de la criptografía (por ejemplo, la generación de contraseñas, algoritmos aleatorios para evitar la negación algorítmica de los ataques del servicio) [Familiarizarse]
10. Explicar los diferentes tipos de mecanismos para detectar y mitigar los errores de desinfección de datos [Familiarizarse]
11. Demostrar cómo se prueban los programas para el manejo de errores de entrada [Usar]
12. Usar herramientas estáticas y dinámicas para identificar errores de programación [Usar]
13. Describir cómo se utiliza la arquitectura de memoria para proteger de ataques en tiempo de ejecución [Familiarizarse]

### 2.7.4. IAS/Ataques y Amenazas (1 horas Core-Tier1)

**Temas:**
**Core Tier2**
- 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)

**Electivo**
- Los ataques a la privacidad y el anonimato .
- El malware / comunicaciones no deseadas, tales como canales encubiertos y esteganografía.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir tipos de ataques similares en contra de un sistema en particular [Familiarizarse]
2. Discutir los limitantes de las medidas en contra del malware (ejm. detección basada en firmas, detección de comportamiento) [Familiarizarse]
3. Identificar las instancias de los ataques de ingeniería social y de los ataques de negación de servicios [Familiarizarse]
4. Discutir como los ataques de negación de servicos puede ser identificados y reducido [Familiarizarse]

**Elective:**

5. Describir los riesgos de la privacidad y del anonimato en aplicaciones comunmente usadas [Familiarizarse]
6. Discutir los conceptos de conversión de canales y otros procedimientos de filtrado de datos [Familiarizarse]

### 2.7.5. IAS/Seguridad de Red (2 horas Core-Tier1)

La discusión de la seguridad de red se basa en la comprensión previa sobre conceptos fundamentales de la creación de redes, incluyendo protocolos, como TCP / IP y la arquitectura de red / organización Aplicaciones en red.

**Temas:**
**Core Tier2**
- 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)

**Electivo**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir las diferentes categorías de amenazas y ataques en redes [Familiarizarse]
2. Describir las arquitecturas de criptografía de clave pública y privada y cómo las ICP brindan apoyo a la seguridad en redes [Familiarizarse]
3. Describir ventajas y limitaciones de las tecnologías de seguridad en cada capa de una torre de red [Familiarizarse]
4. Identificar los adecuados mecanismos de defensa y sus limitaciones dada una amenaza de red [Familiarizarse]

**Elective:**

5. Discutir las propiedades de la seguridad y sus limitaciones de redes no cableadas [Familiarizarse]
6. Identificar amenazas adicionales que enfrentan las redes no cableadas [Familiarizarse]
7. Describir amenazas que pueden y no pueden ser protegidas en contra del uso de canales de comunicación seguros [Familiarizarse]
8. Resumir las defensas en contra de la censura en redes [Familiarizarse]
9. Diagramar una red de seguridad [Familiarizarse]

### 2.7.6. IAS/Criptografía (1 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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.

**Electivo**

- Preliminares matemáticos esenciales para la criptografía, incluyendo temas de álgebra lineal, teoría de números, teoría de la probabilidad y la estadística.
- Primitivas criptográficas: 1. generadores pseudo-aleatorios y cifrados de flujo 2. cifrados de bloque (permutaciones pseudo-aleatorios), por ejemplo, AES 3. funciones de pseudo-aleatorios 4. funciones de hash, por ejemplo, SHA2, resistencia colisión 5. códigos de autenticación de mensaje 6. funciones derivaciones clave
- Criptografía de clave simétrica: 1. El secreto perfecto y el cojín de una sola vez 2. Modos de funcionamiento para la seguridad semántica y encriptación autenticada (por ejemplo, cifrar-entonces-MAC, OCB, GCM) 3. Integridad de los mensajes (por ejemplo, CMAC, HMAC)
- La criptografía de clave pública: 1. Permutación de trampilla, por ejemplo, RSA 2. Cifrado de clave pública, por ejemplo, el cifrado RSA, cifrado El Gamal 3. Las firmas digitales 4. Infraestructura de clave pública (PKI) y certificados 5. Supuestos de dureza, por ejemplo, Diffie-Hellman, factoring entero
- Protocolos de intercambio de claves autenticadas, por ejemplo, TLS .

- Los protocolos criptográficos: autenticación desafío-respuesta, protocolos de conocimiento cero, el compromiso, la transferencia inconsciente, seguro 2-partido o multipartidista computación, compartición de secretos y aplicaciones .
- Motivar a los conceptos que utilizan las aplicaciones del mundo real, por ejemplo, dinero electrónico, canales seguros entre clientes y servidores, correo electrónico seguro, autenticación de la entidad, el emparejamiento de dispositivos, sistemas de votación.
- Definiciones de seguridad y los ataques a las primitivas criptográficas: 1. Objetivos: indistinción, unforgeability, colisión-resistencia 2. Capacidades atacante: ataque-mensaje elegido (para firmas), ataques de cumpleaños, ataques de canal lateral, ataques de inyección de fallos.
- Estándares criptográficos y referencias de implementaciones.
- La criptografía cuántica.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir el propósito de la Criptografía y listar formas en las cuales es usada en comunicación de datos [Familiarizarse]
2. 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]
3. Discutir la importancia de los números primos en criptografía y explicar su uso en algoritmos criptográficos [Familiarizarse]
4. Explicar como una infraestructura de Clave Pública soporta firmas digitales y encriptación y discutir sus limitaciones/vulnerabilidades [Familiarizarse]

**Elective:**

5. Usar primitivas criptográficas y sus propiedades básicas [Usar]
6. Ilustrar como medir la entropía y como generar aleatoriedad criptográfica [Usar]
7. Usa primitivas de clave pública y sus aplicaciones [Usar]
8. Explicar como los protocolos de intercambio de claves trabajan y como es que pueden fallar [Familiarizarse]
9. Discutir protocolos criptográficos y sus propiedades [Familiarizarse]
10. Describir aplicaciones del mundo real de primitivas criptográficas y sus protocolos [Familiarizarse]
11. Resumir definiciones precisas de seguridad, capacidades de ataque y sus metas [Familiarizarse]
12. Aplicar técnicas conocidas y apropiadas de criptografía para un escenario determinado [Usar]
13. Apreciar los peligros de inventarse cada uno sus propios métodos criptográficos [Familiarizarse]
14. Describir la criptografía cuántica y el impacto de la computación cuántica en algoritmos criptográficos [Familiarizarse]

### 2.7.7. IAS/Seguridad en la Web

**Temas:**
**Electivo**

- Modelo de seguridad Web 1. Modelo de seguridad del navegador incluida la política de mismo origen 2. Los límites de confianza de cliente-servidor, por ejemplo, no pueden depender de la ejecución segura en el cliente
- Gestión de sesiones, la autenticación: 1. Single Sign-On 2. HTTPS y certificados
- Vulnerabilidades de las aplicaciones y defensas : 1. Inyección SQL 2. XSS 3. CSRF

- Seguridad del lado del cliente : 1. Política de seguridad Cookies 2. Extensiones de seguridad HTTP, por ejemplo HSTS 3. Plugins, extensiones y aplicaciones web 4. Seguimiento de los usuarios Web
- Herramientas de seguridad del lado del servidor, por ejemplo, los cortafuegos de aplicación Web (WAFS) y fuzzers

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describe el modelo de seguridad de los navegadores incluyendo las políticas del mismo origen y modelos de amenazas en seguridad web [Familiarizarse]
2. 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]
3. Investigar los tipos comunes de vulnerabilidades y ataques en las aplicaciones web, y defensas contra ellos [Familiarizarse]
4. Utilice las funciones de seguridad del lado del cliente [Usar]

### 2.7.8. IAS/Seguridad de plataformas

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explica el concepto de integridad de código y firma de códigos, así como el alcance al cual se aplica [Familiarizarse]
2. Discute los conceptos del origen de la confidencialidad y el de los procesos de arranque y carga segura [Familiarizarse]
3. Describe los mecanismos de arresto remoto de la integridad de un sistema [Familiarizarse]
4. Resume las metas y las primitivas claves de los modelos de plataforma confiable (TPM) [Familiarizarse]
5. Identifica las amenazas de conectar periféricos en un dispositivo [Familiarizarse]
6. Identifica ataques físicos y sus medidas de control [Familiarizarse]
7. Identifica ataques en plataformas con hardware que no son del tipo PC [Familiarizarse]
8. Discute los conceptos y la importancia de ruta confiable [Familiarizarse]

### 2.7.9. IAS/Política de Seguridad y Gobernabilidad

Véase en general Políticas de seguridad, Leyes y crímenes computacionales
**Temas:**
**Electivo**

- Política de privacidad.

- Controles de inferencia / limitación divulgación estadística .
- Política de copia de seguridad, política de contraseñas de actualización.
- Incumplimiento política de divulgación.
- La recolección de datos y políticas de retención
- Política de la cadena de suministro.
- Puntos de equilibrio de la seguridad en la nube.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describe el concepto de privacidad incluyendo información privada personal, violaciones potenciales de privacidad debido a mecanismos de seguridad, y describe como los mecanismos de protección de la privacidad se ejecutan en conflicto con los mecanismos de seguridad [Familiarizarse]
2. Describe como un atacante puede descubrir un secreto al interactuar con una base de datos [Familiarizarse]
3. Explica como configurar una política de respaldo de datos o política de actualización de claves [Familiarizarse]
4. Discute como configurar una política en caso de revelación de la información [Familiarizarse]
5. Describe las consecuencias del políticas de retención de datos [Familiarizarse]
6. Identifica los riesgos de basarse en productos externos [Familiarizarse]
7. Identifica los riesgos y beneficios de emplear outsourcing en la nube [Familiarizarse]

### 2.7.10. IAS/Investigación digital (Digital Forensics)

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describe qué es una investigación digital, las fuentes de evidencia digital, y los límites de técnicas forenses [Familiarizarse]
2. Explica como diseñar software de apoyo a técnicas forenses [Familiarizarse]
3. Describe los requisitos legales para usar datos recuperados [Familiarizarse]
4. Describe el proceso de recolección de evidencia desde el tiempo en que se identifico el requisito hasta la colocación de los datos [Familiarizarse]

5. Describe como se realiza la recolección de datos y el adecuado almacenamiento de los datos originales y de la copia forense [Familiarizarse]

6. Realiza recolección de datos en un disco duro [Usar]

7. Describe la responsabilidad y obligación de una persona mientras testifica como un examinador forense [Familiarizarse]

8. Recupera datos basados en un determinado término de búsqueda en una imagen del sistema [Usar]

9. Reconstruye el historial de una aplicación a partir de los artefactos de la aplicación [Usar]

10. Reconstruye el historial de navegación web de los artefactos web [Usar]

11. Captura e interpreta el tráfico de red [Usar]

12. Discute los retos asociados con técnicas forenses de dispositivos móviles [Familiarizarse]

13. Inspecciona un sistema (red, computadora, o aplicación) para determinar la presencia de malware o de actividad maliciosa [Usar]

14. Aplica herramientas forenses a fin de investigar fugas en la seguridad [Usar]

15. Identifica métodos anti-forenses [Familiarizarse]

### 2.7.11. IAS/Seguridad en Ingeniería de Software

Fundamentos de las prácticas de codificación segura cubiertos en otras áreas de conocimiento, incluyendo SDF y SE Construcción de Software, Verificación y Validación de Software.
**Temas:**
**Electivo**

- 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)
- Aseguramiento de la calidad del software y las mediciones de benchmarking.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describir los requisitos para la integración de la seguridad en el SDL [Familiarizarse]

2. 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 [Usar]

3. Desarrollar especificaciones para un esfuerzo de desarrollo de software que especifica completamente los requisitos funcionales y se identifican las rutas de ejecución esperadas [Usar]

4. Describir las mejores prácticas de desarrollo de software para minimizar las vulnerabilidades en el código de programación [Familiarizarse]

5. Llevar a cabo una verificación de seguridad y la evaluación (estático y dinámico) de una aplicación de software [Usar]

## 2.8. Gestión de la información (IM)

Gestión de la información se refiere principalmente a la captura, digitalización, representación, organización, transformación y presentación de la información; algoritmos para el acceso y la actualización de la información almacenada de forma eficiente y eficaz; modelado de datos y abstracción; y técnicas de almacenamiento de archivos físicos. El estudiante tiene que ser capaz de desarrollar modelos de datos conceptuales y físicos, determinar qué métodos y técnicas de mensajería instantánea son apropiadas para un problema dado y ser capaz de seleccionar e implementar una solución de mensajería instantánea apropiada que aborde las preocupaciones de diseño relevantes, incluyendo escalabilidad, accesibilidad y usabilidad.

También observamos que el área de IM está relacionada con conceptos fundamentales de seguridad de información que se describen en el tópico IAS/FundamentalConcepts.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.8.1 Conceptos de Gestión de la Información (Pág. 53) | 2 | | No |
| 2.8.2 Sistemas de Bases de Datos (Pág. 54) | 3 | | No |
| 2.8.3 Modelado de datos (Pág. 55) | 4 | | No |
| 2.8.4 Indexación (Pág. 55) | | | No |
| 2.8.5 Bases de Datos Relacionales (Pág. 56) | | | No |
| 2.8.6 Lenguajes de Consulta (Pág. 57) | | | No |
| 2.8.7 Procesamiento de Transacciones (Pág. 57) | | | No |
| 2.8.8 Bases de Datos Distribuidas (Pág. 58) | | | No |
| 2.8.9 Diseño Físico de Bases de Datos (Pág. 58) | | | No |
| 2.8.10 Minería de Datos (Pág. 59) | | | No |
| 2.8.11 Almacenamiento y Recuperación de Información (Pág. 59) | | | No |
| 2.8.12 Sistemas Multimedia (Pág. 60) | | | No |

### 2.8.1. IM/Conceptos de Gestión de la Información (2 horas Core-Tier1)

**Temas:**
**Core Tier1**

- Sistemas de Información como Sistemas Socio Técnicos.
- Conceptos Almacenamiento y Recuperación de Información Básica (IS&R)
- Representación y Captura de Información.
- Apoyo a las necesidades humanas:búsqueda, recuperación, enlace, navegación.

**Core Tier2**

- Aplicaciones de Administración de la Información.
- Consultas navegacionales y declarativas, uso de enlaces.
- Análisis e Indexación.
- Temas de calidad:confiabilidad, escalabilidad, eficiencia y efectividad.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Describir cómo los seres humanos obtienen acceso a información y datos para apoyar sus necesidades [Familiarizarse]

2. Describir las ventajas y desventajas del control organizacional central sobre los datos [Evaluar]

3. Identificar las carreras/roles asociados con la gestión de la información (por ejemplo, el administrador de base de datos, modelador de datos, desarrollador de la aplicación, el usuario final) [Familiarizarse]

4. Comparar y contrastar información con los datos y el conocimiento [Evaluar]

5. Demostrar usos del almacenamiento explicito de metadata/esquemas asociados con los datos [Usar]

6. Identificar problemas de la persistencia de datos para una organización [Familiarizarse]

**Core-Tier2:**

7. Critique una solicitud de información con respecto a satisfacer las necesidades de información del usuario [Evaluar]
8. Explicar los usos de las consultas declarativas [Familiarizarse]
9. Dar una versión declarativa para una consulta de navegación [Familiarizarse]
10. Describir varias soluciones técnicas a los problemas relacionados con la privacidad de la información, integridad, seguridad y preservación [Familiarizarse]
11. Explicar medidas de eficiencia (rendimiento, tiempo de respuesta) y efectividad (llamada, precisión) [Familiarizarse]
12. Describir métodos para aumentar la escala de los sistemas de información [Familiarizarse]
13. Identificar las vulnerabilidades y escenarios de fallo en las formas comunes de los sistemas de información [Usar]

### 2.8.2. IM/Sistemas de Bases de Datos (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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.

**Electivo**

- Enfoques para la gestión de grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce).

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explica las características que distinguen un esquema de base de datos de aquellos basados en la programación de archivos de datos [Familiarizarse]
2. Describe los diseños más comunes para los componentes base de sistemas de bases de datos incluyendo el optimizador de consultas, ejecutor de consultas, administrador de almacenamiento, métodos de acceso y procesador de transacciones [Familiarizarse]
3. Cita las metas básicas, funciones y modelos de un sistema de bases de datos [Familiarizarse]
4. Describe los componentes de un sistema de bases datos y da ejemplos de su uso [Familiarizarse]
5. Identifica las funciones principales de un SGBD y describe sus roles en un sistema de bases de datos [Familiarizarse]
6. Explica los conceptos de independencia de datos y su importancia en un sistema de bases de datos [Familiarizarse]
7. Usa un lenguaje de consulta declarativo para recoger información de una base de datos [Usar]
8. Describe las capacidades que las bases de datos brindan al apoyar estructuras y/o la secuencia de flujo de datos, ejm. texto [Familiarizarse]

**Elective:**

9. Describe los enfoques principales para almacenar y procesar larges volúmenes de datos [Familiarizarse]

### 2.8.3. IM/Modelado de datos (4 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Compare y contrasta modelos apropiados de datos, incluyendo estructuras sus estructuras internas, para diversos tipos de datos [Evaluar]
2. Describe los conceptos en notación de modelos (ejm. Diagramas Entidad-Relación o UML) y cómo deben de ser usados [Familiarizarse]
3. Define la terminología fundamental a ser usada en un modelo relacional de datos [Familiarizarse]
4. Describe los principios básicos del modelo relacional de datos [Familiarizarse]
5. Aplica los conceptos de modelado y la notación de un modelo relacional de datos [Usar]
6. Describe los conceptos principales del modelado OO como son identidad de objetos, constructores de tipos, encapsulación, herencia, polimorfismo, y versiones [Familiarizarse]
7. Describe las diferencias entre modelos de datos relacionales y semi-estructurados [Evaluar]
8. Da una semi estructura equivalente (ejm. en DTD o Esquema XML) para un esquema relacional dado [Usar]

### 2.8.4. IM/Indexación

**Temas:**
**Electivo**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Generar un archivo índice para una colección de recursos [Usar]
2. Explicar la función de un índice invertido en la localización de un documento en una colección [Familiarizarse]
3. Explicar cómo rechazar y detener palabras que afectan a la indexación [Familiarizarse]
4. Identificar los índices adecuados para determinado el esquema relacional y el conjunto de consultas [Usar]
5. Estimar el tiempo para recuperar información, cuando son usados los índices comparado con cuando no son usados [Usar]
6. Describir los desafíos claves en el rastreo web, por ejemplo, la detección de documentos duplicados, la determinación de la frontera de rastreo [Familiarizarse]

### 2.8.5. IM/Bases de Datos Relacionales

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Prepara un esquema relacional de un modelo conceptual desarrollado usando el modelo entidad-relación [Usar]

2. Explica y demuestra los conceptos de restricciones de integridad de la entidad e integridad referencial (incluyendo la definición del concepto de clave foránea) [Usar]

3. 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]

4. Escribe consultas en álgebra relacional [Usar]

5. Escribe consultas en cálculo relacional de tuplas [Usar]

6. Determina la dependencia funcional entre dos o más atributos que son subconjunto de una relación [Evaluar]

7. Conecta restricciones expresadas como clave primaria y foránea, con dependencias funcionales [Usar]

8. Calcula la cerradura de un conjunto de atributos dado dependencias funcionales [Usar]

9. Determina si un conjunto de atributos forma una superclave y/o una clave candidata de una relación dada dependencias funcionales [Evaluar]

10. Evalua una descomposición propuesta, a fin de determinar si tiene una unión sin pérdidas o preservación de dependencias [Evaluar]

11. Describe las propiedades de la FNBC, FNUP (forma normal unión de proyecto), 5FN [Familiarizarse]

12. Explica el impacto de la normalización en la eficacia de las operaciones de una base de datos especialmente en la optimización de consultas [Familiarizarse]

13. Describe que es una dependencia de multi valor y cual es el tipo de restricciones que especifica [Familiarizarse]

### 2.8.6. IM/Lenguajes de Consulta

**Temas:**
**Electivo**

- Visión general de lenguajes de base de datos.
- SQL (definición de datos, formulación 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Crear un esquema relacional de bases de datos en SQL que incorpora restricciones clave y restricciones de integridad de entidad e integridad referencial [Usar]
2. Usar SQL para crear tablas y devuelve (SELECT) la información de una base de datos [Usar]
3. Evaluar un conjunto de estrategias de procesamiento de consultas y selecciona la estrategia óptima [Evaluar]
4. Crear una consulta no-procedimental al llenar plantillas de relacines para construir un ejemplo del resultado de una consulta requerida [Usar]
5. Adicionar consultas orientadas a objetos en un lenguaje stand-alone como C++ o Java (ejm. SELECT ColMethod() FROM Objeto) [Usar]
6. Escribe un procedimiento almacenado que trata con parámetros y con algo de flujo de control de tal forma que tenga funcionalidad [Usar]

### 2.8.7. IM/Procesamiento de Transacciones

**Temas:**
**Electivo**

- Transacciones.
- Fallo y recuperación.
- Control concurente.
- Interacción de gestión de transacciones con el almacenamiento, especialmente en almacenamiento.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Crear una transacción mediante la incorporación de SQL en un programa de aplicación [Usar]
2. Explicar el concepto de confimaciones implicitas [Familiarizarse]
3. Describir los problemas especificos para la ejecución de una transacción eficiente [Familiarizarse]
4. Explicar cuando y porqué se necesita un *rollback*, y cómo registrar todo asegura un *rollback* adecuado [Evaluar]
5. Explicar el efecto de diferentes niveles de aislamiento sobre los mecanismos de control de concurrencia [Evaluar]

6. Elejir el nivel de aislamiento adecuado para la aplicación de un protocolo de transacción especificado [Evaluar]

7. Identificar los límites apropiados de la transacción en programas de aplicación [Evaluar]

### 2.8.8. IM/Bases de Datos Distribuidas

**Temas:**
**Electivo**

- DBMS Distribuidas 1. Almacenamiento de datos distribuido 2. Procesamiento de consultas distribuido 3. Modelo de transacciones distribuidas 4. Soluciones homogéneas y heterogéneas 5. Bases de datos distribuidas cliente-servidor
- Parallel DBMS 1. Arquitecturas paralelas DBMS: memoria compartida, disco compartido, nada compratido; 2. Aceleración y ampliación, por ejemplo, el uso del modelo de procesamiento MapReduce 3. Replicación de información y modelos de consistencia debil

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar las técnicas usadas para la fragmentación de datos, replicación, y la asignación durante el proceso de diseño de base de datos distribuida [Familiarizarse]

2. Evaluar estrategias simples para la ejecución de una consulta distribuida para seleccionar una estrategia que minimise la cantidad de transferencia de datos [Evaluar]

3. 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 [Familiarizarse]

4. Describir el control concurrente distribuido basados en técnicas de copia distinguidos y el método de votación. [Familiarizarse]

5. Describir los tres niveles del software en el modelo cliente servidor [Familiarizarse]

### 2.8.9. IM/Diseño Físico de Bases de Datos

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explica los conceptos de registro, tipos de registro, y archivos, así como las diversas técnicas para colocar registros de archivos en un disco [Familiarizarse]

2. Da ejemplos de la aplicación de índices primario, secundario y de agrupamiento [Familiarizarse]

3. Distingue entre un índice no denso y uno denso [Evaluar]

4. Implementa índices de multinivel dinámicos usando árboles-B [Usar]

5. Explica la teoría y la aplicación de técnicas de hash internas y externas [Familiarizarse]

6. Usa técnicas de hasp para facilitar la expansión de archivos dinámicos [Usar]

7. Describe las relaciones entre hashing, compresión, y búsquedas eficientes en bases de datos [Familiarizarse]
8. Evalúa el costo y beneficio de diversos esquemas de hashing [Evaluar]
9. Explica como el diseño físico de una base de datos afecta la eficiencia de las transacciones en ésta [Familiarizarse]

### 2.8.10. IM/Minería de Datos

**Temas:**
**Electivo**

- Uso de minería de datos.
- Algoritmos de minería de datos.
- Patrón asociativo y secuencial.
- Agrupación de datos.
- Análisis de la cesta de mercado.
- Limpieza de datos.
- Visualización de datos.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Compara y contrasta los diversos usos de la minería de datos como se evidencia en campos tanto de investigación como de aplicación [Evaluar]
2. Explica el valor de encontrar associaciones de datos en mercados financieros [Familiarizarse]
3. Caracteriza los tipos de patrones que pueden ser descubiertos por minería de reglas de asociación [Evaluar]
4. Describe como expandir un sistema relacional para encontrar patrones usando reglas de asociación [Familiarizarse]
5. Evalúa diversas metodologías para la aplicación efectiva de minería de datos [Evaluar]
6. Identifica y caracteriza fuentes de ruido, redundancia y valores atípicos en los datos presentes [Evaluar]
7. Identifica mecanismos (agregación en línea, comportamiento cualquiera, visualización interactiva) a fin de cerrar el ciclo en un proceso de minería de datos [Familiarizarse]
8. Describe porqué los diversos procesos de cierre de ciclo mejoran la efectividad de la minería de datos [Familiarizarse]

### 2.8.11. IM/Almacenamiento y Recuperación de Información

**Temas:**
**Electivo**

- 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).

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explica los conceptos básicos de almacenamiento y recuperación de la información [Familiarizarse]
2. Describe que temas son específicos para una recuperación de la información eficiente [Familiarizarse]
3. Da aplicaciones de estrategias alternativas de búsqueda y explica porqué una estrategia en particular es apropiada para una aplicación [Evaluar]
4. Diseña e implementa un sistema de almacenamiento y recuperación de la información o librería digital de tamaño pequeño a mediano [Usar]
5. Describe algunas de las soluciones técnicas a los problemas relacionados al archivamiento y preservación de la información en una librería digital [Familiarizarse]

### 2.8.12. IM/Sistemas Multimedia

**Temas:**
**Electivo**

- Dispositivos de entrada/salida, controladores de dispositivo, señales de control y protocolos, DSPs
- Estándares (Ej. audio, gráfico, video)
- Aplicaciones, editores de media, sistemas de autoría , y autoría
- Flujos/estructuras, captura/ representación/transformarción, espacios/dominios, compresión/codificación.
- Análisis basado en contenido, indexación y recuperación de audio, imágenes, animación y video.
- Presentación, renderización, sincronización, integración multimodal , interfaces.
- Entrega en tiempo real, calidad de servicio (incluye rendimiento), capacidad de planeamiento, conferencia audio/video, video en demanda.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describe los medios y dispositivos de soporte más comunmente asociados a los sistemas de información multimedia [Familiarizarse]
2. Demostrar el uso de contenidos basados en análisis de información en un sistema de información multimedia [Usar]
3. Evaluar presentaciones multimedia en terminos de su uso apropiado de audio, video, gráficos, color y otros conceptos de presentación de información [Evaluar]
4. Implementar una aplicación multimedia usando un sistema de autoráa [Usar]
5. Para cada uno de los muchos estándares de medios de comunicación o multimedia, describe en lenguaje no técnico para lo que el estandar llama y explica como aspectos de la percepción humana deberian ser suceptibles a las limitaciones del estandar [Familiarizarse]

6. Describir las caracteristicas de un sistema de computador (incluyendo identificación de herramientas de soporte y estándares apropiados) que tiene que organizar de uno de un rango de aplicaciones multimedia posibles [Familiarizarse]

## 2.9. Inteligencia Artificial (AI)

La inteligencia artificial (AI) es el estudio de soluciones para los problemas que son difíciles o muy difícil de resolver con los métodos tradicionales. Se utiliza de forma transversal en apoyo de aplicaciones de uso diario, tales como el correo electrónico, procesamiento de textos y la búsqueda, así como en el diseño y análisis de agentes autónomos que perciben su entorno y racionalmente interactúan con el medio ambiente.

Las soluciones se basan en un amplio conjunto de esquemas de representación de conocimientos generales y especializados, mecanismos de resolución de problemas y técnicas de aprendizaje. Se ocupan de la detección (por ejemplo, el reconocimiento del habla, la comprensión del lenguaje natural, la visión por computador), la resolución de problemas (por ejemplo, la búsqueda, planificación), y actuar (por ejemplo, la robótica) y las arquitecturas necesarias para apoyarlos (por ejemplo, agentes, multi-agentes). El estudio de la Inteligencia Artificial prepara al estudiante para determinar cuándo un enfoque AI es apropiada para un problema dado, identificar la representación adecuada y mecanismo de razonamiento para ponerla en práctica y evaluarla.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.9.1 Cuestiones fundamentales (Pág. 61) | 1 | | No |
| 2.9.2 Estrategias de búsquedas básicas (Pág. 62) | 4 | | No |
| 2.9.3 Raciocinio y representación básica de conocimiento (Pág. 62) | 3 | | No |
| 2.9.4 Aprendizaje Automático Básico (Pág. 63) | 2 | | No |
| 2.9.5 Búsqueda Avanzada (Pág. 63) | | | No |
| 2.9.6 Representación Avanzada y Razonamiento (Pág. 64) | | | No |
| 2.9.7 Razonamiento Bajo Incertidumbre (Pág. 64) | | | No |
| 2.9.8 Agentes (Pág. 65) | | | No |
| 2.9.9 Procesamiento del Lenguaje Natural (Pág. 66) | | | No |
| 2.9.10 Aprendizaje de máquina avanzado (Pág. 66) | | | No |
| 2.9.11 Robótica (Pág. 67) | | | No |
| 2.9.12 Visión y percepción por computador (Pág. 68) | | | No |

### 2.9.1. AI/Cuestiones fundamentales (1 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Descripción general de los problemas de Inteligencia Artificial, ejemplos recientes de aplicaciones de Inteligencia artificial.
- ¿Qué es comportamiento inteligente? 1. El Test de Turing 2. Razonamiento Racional versus No Racional
- Características del Problema: 1. Observable completamente versus observable parcialmente 2. Individual versus multi-agente 3. Deterministico versus estocástico 4. Estático versus dinámico 5. Discreto versus continuo
- Naturaleza de agentes: 1. Autónomo versus semi-autónomo 2. Reflexivo, basado en objetivos, y basado en utilidad 3. La importancia en percepción e interacciones con el entorno

**Electivo**

- Cuestiones filosóficas y éticas.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir el test de Turing y el experimento pensado cuarto chino" (*Chinese Room*) [Familiarizarse]

2. Diferenciar [Familiarizarse]
3. Determinando las caracteristicas de un problema dado que la Inteligencia Artificial deberian resolver [Evaluar]

### 2.9.2. AI/Estrategias de búsquedas básicas (4 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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).

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. 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]
2. Describe el rol de las heurísticas y describe los intercambios entre completitud, óptimo, complejidad de tiempo, y complejidad de espacio [Familiarizarse]
3. Describe el problema de la explosión combinatoria del espacio de búsqueda y sus consecuencias [Familiarizarse]
4. Selecciona e implementa un apropiado algoritmo de búsqueda no informado para un problema, y describe sus complejidades de tiempo y espacio [Usar]
5. Selecciona e implementa un apropiado algoritmo de búsqueda informado para un problema al definir la función heurística de evaluación necesaria [Usar]
6. Evalúa si una heurística dada para un determinado problema es admisible/puede garantizar una solución óptima [Evaluar]
7. Formula un problema en particular en lenguaje natural (ejm. Inglés) como un problema de satisfacción de restricciones y lo implementa usando un algoritmo de retroceso cronológico o una búsqueda estocástica local [Usar]
8. Compara y contrasta tópicos de búsqueda básica con temas jugabilidad de juegos [Familiarizarse]

### 2.9.3. AI/Raciocinio y representación básica de conocimiento (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Revisión de la lógica proposicional y de predicados
- Resolución y demostración de teoremas (sólo la lógica proposicional).
- Encadenamiento hacia adelante, encadenamiento hacia atrás.
- Examen de razonamiento probabilístico, el teorema de Bayes.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Traducir una sentencia en lenguaje natural (Por ejemplo español) en una declaración lógica de predicados [Usar]
2. Convertir una declaración lógica en forma de cláusula [Usar]
3. Aplicar resolución a un conjunto de declaraciones lógicas para responder una consulta [Usar]

4. Hacer una inferencia probabilística para un problema real usando el teorema de Bayes para determinar la probabilidad que se cumpla una hipótesis [Usar]

### 2.9.4. AI/Aprendizaje Automático Básico (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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.
- Medición clasificada con exactitud.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Listar las diferencias entre los tres principales tipos de aprendizaje: supervisado, no supervisado y por refuerzo [Familiarizarse]
2. Identificar ejemplos de tareas de clasificación, considerando las características de entrada disponibles y las salidas a ser predecidas [Familiarizarse]
3. Explicar la diferencia entre aprendizaje inductivo y deductivo [Familiarizarse]
4. Describir el sobre ajuste (*overfitting*) en el contexto de un problema [Familiarizarse]
5. Aplicar un algoritmo de aprendizaje estadístico simple como el Clasificador Naive Bayesiano e un problema de clasificación y medirla precisión del clasificador [Usar]

### 2.9.5. AI/Búsqueda Avanzada

Tenga en cuenta que los temas generales de la rama-y-atado y dinámico Programación, se enumeran en AL / algorítmico Estrategias.
**Temas:**
**Electivo**

- Construcción de árboles de búsqueda, espacio de búsqueda dinámico, explosión combinatoria del espacio de búsqueda.
- Búsqueda estocástica: 1. Simulated annealing 2. Algoritmos genéticos 3. Búsqueda de árbol Monte-Carlo
- Implementación de búsqueda A *, búsqueda en haz.
- Búsqueda Minimax, poda alfa-beta.
- Búsqueda Expectimax (MDP-Solving) y los nodos de azar.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Diseñar e implementar una solución a un problema con algoritmo genético [Usar]
2. Diseñar e implementar un esquema de recocido simulado (*simulated annealing*) para evitar mínimos locales en un problema [Usar]
3. Diseñar e implementar una búsqueda A* y búsqueda en haz (*beam search*) para solucionar un problema [Usar]
4. Aplicar búsqueda minimax con poda alfa-beta para simplifiar el espacio de búsqueda en un juego con dos jugadores [Usar]
5. Comparar y contrastar los algoritmos genéticos con técnicas clásicas de búsqueda [Evaluar]
6. Comparar y contrastar la aplicabilidad de varias heurísticas de búsqueda, para un determinado problema [Evaluar]

### 2.9.6. AI/Representación Avanzada y Razonamiento

**Temas:**
**Electivo**

- Problemas de Representación del Conocimiento: 1. Lógica de Descripción 2. Ingeniería de Ontología
- Razonamiento no monotónico (p.e., lógica no clásica, razonamiento por defecto)
- Argumentación
- El razonamiento sobre la acción y el cambio (por ejemplo, la situación y cálculo de eventos).
- Razonamiento temporal y espacial.
- Sistemas Expertos basados en reglas.
- Redes semánticas.
- Razonamiento basado en modelos y razonamiento basado en casos.
- Planeamiento: 1. Planeamiento Parcial y Totalmente ordenado 2. *Plan graphs* 3. Planeamiento Jerárquico 4. Planeamiento y Ejecución incluyendo planeamiento condicional y planeamiento continuo 5. Planeamiento Agente móvil/Multiagente

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Comparar y contrastar los modelos más usados para la representación del conocimiento estructurado, destacando sus puntos fuertes y débiles [Evaluar]
2. Identificar los componentes de razonamiento no monótono y su utilidad como mecanismo de representación de los sistemas de confianza [Familiarizarse]
3. Comparar y contrastas las técnicas básicas para la representación de la incertidumbre [Evaluar]
4. Comparar y contrastar las técnicas básicas para la representación cualitativa [Evaluar]
5. Aplicar cálculo de situaciones y eventos a problemas de acción y cambios [Usar]
6. Explicar la diferencia entre razonamiento temporal y espacial, y cómo se relacionan entre sí. [Familiarizarse]
7. Explicar la diferencia entre técnicas de razonamiento basado en modelos, basado en casos y basados en reglas [Familiarizarse]
8. Definir el concepto de un sistema planificación y cómo se diferencia de las técnicas de búsqueda clásicas [Familiarizarse]
9. Describir las diferencias entre la planificación como búsqueda, la planificación basada en el operador, y la planificación proposicional, dando ejemplos de ámbitos en los que cada una es más aplicable [Familiarizarse]
10. Explique la diferencia entre la inferencia monótona y no monótona [Familiarizarse]

### 2.9.7. AI/Razonamiento Bajo Incertidumbre

**Temas:**
**Electivo**

- Revisión de Probabilidad Básica
- Variables aleatorias y distribuciones de probabilidad: 1. Axiomas de probabilidad 2. Inferencia probabilística 3. Regla de Bayes
- Independecia Condicional
- Representaciones del conocimiento: 1. Redes bayesianas *a*) Inferencia exacta y su complejidad *b*) Métodos de Muestreo aleatorio (Monte Carlo) (p.e. Muestreo de Gibbs) 2. Redes Markov 3. Modelos de probabilidad relacional 4. Modelos ocultos de Markov
- Teoria de decisiones: 1. Preferencias y funciones de utilidad 2. Maximizando la utilidad esperada

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Aplicar la regla de Bayes para determinar el cumplimiento de una hipótesis [Usar]
2. Explicar cómo al tener independencia condicional permite una gran eficiencia en sistemas probabilísticos [Evaluar]
3. Identificar ejemplos de representación de conocimiento para razonamiento bajo incertidumbre [Familiarizarse]
4. Indicar la complejidad de la inferencia exacta. Identificar métodos para inferencia aproximada [Familiarizarse]
5. Diseñar e implementar, al menos una representación de conocimiento para razonamiento bajo incertidumbre [Usar]
6. Describir la complejidad del razonamiento probabilístico en el tiempo [Familiarizarse]
7. Diseñar e implementar un Modelo Oculto de Markov (HMM) como un ejemplo de sistema probabilístico en el tiempo [Usar]
8. Describir las relaciones entre preferencias y funciones de utilidad [Familiarizarse]
9. Explicar como funciones de utilidad y razonamiento probabilístico puede ser combinado para tomar decisiones razonables [Evaluar]

### 2.9.8. AI/Agentes

**Temas:**
**Electivo**

- Definición de Agentes
- Arquitectura de agentes (Ej. reactivo, en capa, cognitivo)
- Teoría de agentes
- Racionalidad, teoría de juegos: 1. Agentes de decisión teórica 2. Procesos de decisión de Markov (MDP)
- Agentes de Software, asistentes personales, y acceso a información: 1. Agentes colaborativos 2. Agentes de recolección de información 3. Agentes creíbles (carácter sintético, modelamiento de emociones en agentes)
- Agentes de aprendizaje
- Sistemas Multi-agente 1. Agentes Colaborativos 2. Equipos de Agentes 3. Agentes Competitivos (ej., subastas, votaciones) 4. Sistemas de enjambre y modelos biológicamente inspirados

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Lista las características que definen un agente inteligente [Familiarizarse]
2. Describe y contrasta las arquitecturas de agente estándares [Evaluar]
3. Describe las aplicaciones de teoría de agentes para dominios como agentes de software, asistentes personales, y agentes creibles [Familiarizarse]
4. Describe los paradigmas primarios usados por agentes de aprendizaje [Familiarizarse]
5. Demuestra mediante ejemplos adecuados como los sistemas multi-agente soportan interacción entre agentes [Usar]

### 2.9.9. AI/Procesamiento del Lenguaje Natural

**Temas:**
**Electivo**

- Gramaticas determinísticas y estocásticas
- Algoritmos de parseo 1. Gramáticas libres de contexto (CFGs) y cuadros de parseo (e.g. Cocke-Younger-Kasami CYK) 2. CFGs probabilísticos y ponderados CYK
- Representación del significado / Semántica 1. Representación de conocimiento basado en lógica 2. Roles semánticos 3. Representaciones temporales 4. 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: 1. Modelo de espacio vectorial *a*) TF & IDF 2. Precision y cobertura
- Extracción de información
- Traducción de lenguaje
- Clasificación y categorización de texto: 1. Modelo de bolsa de palabras

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Define y contrasta gramáticas de tipo estocásticas y determinísticas, dando ejemplos y demostrando como adecuar cada una de ellas [Evaluar]

2. Simula, aplica, o implementa algoritmos clásicos y estocásticos para el parseo de un lenguaje natural [Usar]

3. Identifica los retos de la representación del significado [Familiarizarse]

4. Lista las ventajas de usar corpus estándares. Identifica ejemplos de corpus actuales para una variedad de tareas de PLN [Familiarizarse]

5. Identifica técnicas para la recuperación de la información, traducción de lenguajes, y clasificación de textos [Familiarizarse]

### 2.9.10. AI/Aprendizaje de máquina avanzado

**Temas:**
**Electivo**

- 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 1. Aprendizaje basado en árboles de decisión 2. Aprendizaje basado en redes neuronales 3. Aprendizaje basado en máquinas de soporte vectorial (*Support vector machines SVMs*)
- Comité de Máquinas (*Emsembles methods*)
- Algoritmos del vecino mas próximo
- Aprendizaje y *clustering* no supervisado 1. EM 2. K-means 3. Mapas auto-organizados
- Aprendizaje semi-supervisado.
- Aprendizaje de modelos gráficos
- Evaluación del desempeño (tal como cross-validation, area bajo la curva ROC)
- Teoria del aprendizaje.
- El problema del sobreajuste, la maldición de la dimensionalidad.
- Aprendizaje por refuerzo 1. Equilibrio Exploración vs. explotación 2. Procesos de decisión de Markov 3. Iteración por valor e iteración por politica
- Aplicación de algoritmos Machine Learning para Minería de datos.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explica las diferencias entre los tres estilos de aprendizaje: supervisado, por refuerzo y no supervisado [Familiarizarse]
2. Implementa algoritmos simples para el aprendizaje supervisado, aprendizaje por refuerzo, y aprendizaje no supervisado [Usar]
3. Determina cuál de los tres estilos de aprendizaje es el apropiado para el dominio de un problema en particular [Usar]
4. 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 [Evaluar]
5. Evalúa el rendimiento de un sistema de aprendizaje simple en un conjunto de datos reales [Evaluar]
6. Describe el estado del arte en la teoría del aprendizaje, incluyendo sus logros y limitantes [Familiarizarse]
7. Explica el problema del sobreajuste, conjuntamente con técnicas para determinar y manejar el problema [Usar]

### 2.9.11. AI/Robótica

**Temas:**
**Electivo**

- Vision general: problemas y progreso 1. Estado del arte de los sistemas robóticos, incluyendo sus sensores y una visión general de su procesamiento 2. Arquitecturas de control robótico, ejem., deliverado vs. control reactivo y vehiculos Braitenberg 3. Modelando el mundo y modelos de mundo 4. Incertidumbre inherente en detección y control
- Configuración de espacio y mapas de entorno.
- Interpretando datos del sensor con incertidumbre.
- Localización y mapeo.
- Navegación y control.
- Planeando el movimiento.
- Coordinación multi-robots.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Listar capacidades y limitaciones de sistemas del estado del arte en robótica de hoy , incluyendo sus sensores y el procesamiento del sensor crucial que informa a esos sistemas [Familiarizarse]
2. Integrar sensores, actuadores y software en un robot diseñado para emprender alguna tarea [Usar]
3. Programar un robot para llevar a cabo tareas simples usando arquitecturas de control deliverativo, reactivo y/o híbrido [Usar]
4. Implementar algoritmos de planificación de movimientos fundamentales dentro del espacio de configuración de un robot [Usar]
5. Caracterizar las incertidumbres asociadas con sensores y actuadores de robot comunes; articular estrategias para mitigar esas incertidumbres. [Familiarizarse]
6. Listar las diferencias entre representaciones de los robot de su enterno externo, incluyendo sus fortalezas y defectos [Familiarizarse]
7. Comparar y contrastar al menos tres estrategias para la navegación de robots dentro de entornos conocidos y/o no conocidos, incluyendo sus fortalezas y defectos [Evaluar]
8. Describir al menos una aproximación para la coordinación de acciones y detección de varios robots para realizar una simple tarea [Familiarizarse]

### 2.9.12. AI/Visión y percepción por computador

**Temas:**
**Electivo**

- Visión Computacional 1. Adquisición de imágenes, representación, procesamiento y propiedades 2. Representación de formas, reconocimiento y segmentación de objetos 3. Análisis de movimiento
- Audio y reconocimiento de dictado.
- Modularidad en reconocimiento.
- Enfoques de reconocimiento de patrones 1. Algoritmos de clasificación y medidas de calidad de la clasificación. 2. Técnicas estadísticas.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Resumir la importancia del reconocimiento de imagenes y objetos en Inteligencia Artificial (AI) e indicar varias aplicaciones significativas de esta tecnologia [Familiarizarse]
2. 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 [Familiarizarse]
3. Implementar reconocimiento de objetos en 2d basados en la representación del contorno y/o regiones basadas en formas [Usar]
4. Destinguir las metas de reconocimiento de sonido, palabras y del habla e identificar como la señal de audio bruto sera manejada diferentemente en cada uno de esos casos. [Familiarizarse]
5. 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 [Familiarizarse]
6. 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]
7. Implementar un algoritmo que combina características en percepciones de más alto nivel, p.e., un contorno o poligono a partir de primitivas visuales o fonemas de una señal de audio [Usar]
8. Implementar un algoritmo de clasificación que segmenta percepciones de entrada en categorias de salida y evalua cuantitativamente la clasificación resultante [Usar]
9. 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 [Evaluar]
10. Describir por lo menos tres enfoques de clasificación, sus pre requisitos para aplicabilidad, fortalezas y deficiencias [Familiarizarse]

## 2.10. Redes y comunicaciones (NC)

Las redes de Internet y de la computadora ahora son ubicuos y un número creciente de actividades informáticas dependen en gran medida de la correcta operación de la red subyacente. Redes, tanto fijos como móviles, son una parte clave del entorno informático de hoy y de mañana. Muchas de las aplicaciones informáticas que se utilizan hoy en día no sería posible sin las redes. Esta dependencia de la red subyacente es probable que aumente en el futuro.

El objetivo de aprendizaje de alto nivel de este módulo se puede resumir como sigue: 1. Pensar en un mundo en red. El mundo es cada vez más interconectados y el uso de redes seguirá aumentando. Los estudiantes deben entender cómo se comportan las redes y los principios clave detrás de la organización y el funcionamiento de las redes. 2. Continúa estudio. El dominio de red está evolucionando rápidamente y un primer curso de redes debe ser un punto de partida para otros cursos

más avanzados en el diseño de redes, gestión de redes, redes de sensores, etc. 3. Principios y práctica interactúan. Networking es real y muchas de las decisiones de diseño que involucran redes también dependen de las limitaciones prácticas. Los estudiantes deben ser expuestos a estas limitaciones prácticas por experimentar con la creación de redes, el uso de herramientas, y la escritura de software en red.

Hay diferentes maneras de organizar un curso de creación de redes. Algunos educadores prefieren un enfoque de arriba hacia abajo, es decir, el curso se inicia desde las aplicaciones y luego explica confiable entrega, enrutamiento y reenvío. Otros educadores prefieren un enfoque de abajo hacia arriba, donde los estudiantes comienzan con las capas inferiores y construir su comprensión de las capas de red, transporte y aplicación posterior.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.10.1 Introducción a redes (Pág. 69) | | | No |
| 2.10.2 Aplicaciones en red (Pág. 69) | | | No |
| 2.10.3 Entrega confiable de datos (Pág. 70) | 2 | | No |
| 2.10.4 Ruteo y reenvío (Pág. 70) | 1.5 | | No |
| 2.10.5 Redes de área local (Pág. 70) | 1.5 | | No |
| 2.10.6 Asignación de recursos (Pág. 71) | 1 | | No |
| 2.10.7 Celulares (Pág. 71) | 1 | | No |
| 2.10.8 Redes sociales (Pág. 71) | | | No |

## 2.10.1. NC/Introducción a redes

IAS / Seguridad de la red, que analiza la seguridad de red y de sus aplicaciones.

**Temas:**

**Core Tier2**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Articular la organización de la Internet [Familiarizarse]
2. Listar y definir la terminología de red apropiada [Familiarizarse]
3. Describir la estructura en capas de una arquitectura típica en red [Familiarizarse]
4. Identificar los diferentes tipos de complejidad en una red (bordes, núcleo, etc.) [Familiarizarse]

## 2.10.2. NC/Aplicaciones en red

**Temas:**

**Core Tier1**

- 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

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Listar las diferencias y las relaciones entre los nombres y direcciones en una red [Familiarizarse]
2. Definir los principios detrás de esquemas de denominación y ubicación del recurso [Familiarizarse]
3. Implementar una aplicación simple cliente-servidor basada en *sockets* [Usar]

### 2.10.3. NC/Entrega confiable de datos (2 horas Core-Tier1)

**Temas:**
**Core Tier2**
- Control de errores (técnicas de retransmisión, temporizadores)
- El control de flujo (agradecimientos, ventana deslizante)
- Problemas de rendimiento (pipelining)
- TCP

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir el funcionamiento de los protocolos de entrega fiables [Familiarizarse]
2. Listar los factores que afectan al rendimiento de los protocolos de entrega fiables [Familiarizarse]
3. Diseñar e implementar un protocolo confiable simple [Usar]

### 2.10.4. NC/Ruteo y reenvío (1.5 horas Core-Tier1)

**Temas:**
**Core Tier2**
- Enrutamiento vs reenvío .
- Enrutamiento estático .
- Protocolo de Internet (IP)
- Problemas de escalabilidad (direccionamiento jerárquico)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir la organización de la capa de red [Familiarizarse]
2. Describir cómo los paquetes se envían en una red IP [Familiarizarse]
3. Listar las ventajas de escalabilidad de direccionamiento jerárquico [Familiarizarse]

### 2.10.5. NC/Redes de área local (1.5 horas Core-Tier1)

**Temas:**
**Core Tier2**
- Problemas de Acceso Múltiple.
- Enfoques comunes a Acceso múltiple (exponencial backoff, multiplexación por división de tiempo, etc)
- Redes de área local .
- Ethernet .
- Switching .

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir como los paquetes son enviados en una red Ethernet [Familiarizarse]
2. Describir las diferencias entre IP y Ethernet [Familiarizarse]
3. Describir las relaciones entre IP y Ethernet [Familiarizarse]
4. Describir las etapas usadas en un enfoque común para el problema de múltiples accesos [Familiarizarse]

### 2.10.6. NC/Asignación de recursos (1 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir como los recursos pueden ser almacenados en la red [Familiarizarse]
2. Describir los problemas de congestión en una red grande [Familiarizarse]
3. Comparar y contrastar las técnicas de almacenamiento estático y dinámico [Evaluar]
4. Comparar y contrastar los enfoques actuales de la congestión [Evaluar]

### 2.10.7. NC/Celulares (1 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Principios de redes celulares.
- Redes 802.11
- Problemas en el apoyo a los nodos móviles (agente local)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir la organización de una red inalambrica [Familiarizarse]
2. Describir como las redes inalámbricas soportan usuarios móviles [Familiarizarse]

### 2.10.8. NC/Redes sociales

**Temas:**
**Electivo**

- Panorama de las redes sociales.
- Ejemplo plataformas de redes sociales.
- Estructura de los grafos de redes sociales.
- Análisis de redes sociales.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Discutir los principios fundamentales(como pertenencia, confianza) de una red social [Familiarizarse]
2. Describir como redes sociales existentes operan [Familiarizarse]
3. Construir un grafo de una red social a partir de datos de la red [Usar]
4. Analizar una red social para determinar quienes son las personas importantes [Usar]
5. Evaluar una determinada interpretación de una pregunta de red social con los datos asociados [Evaluar]

## 2.11. Sistemas Operativos (OS)

Un sistema operativo define una abstracción del hardware y gestiona el intercambio de recursos entre los usuarios de la computadora. Los temas de esta área explican los conocimientos más básicos de los sistemas operativos en el sentido de interfaz de un sistema operativo de redes, la enseñanza de la diferencia entre los modos del núcleo y de los usuarios, y el desarrollo de enfoques clave para el diseño del sistema operativo y la aplicación. Esta área de conocimiento está estructurado para ser complementarios a los Fundamentos de Sistemas (SF), Redes y Comunicación (NC), Seguridad de la Información y de Seguridad (IAS), y las áreas de Informática (PD) de conocimiento paralela y distribuida. Los Sistemas Fundamentales y Aseguramiento de la Información y de Seguridad áreas de conocimiento son las nuevas que para incluir temas contemporáneos. Por ejemplo, Sistemas Fundamentales incluye temas tales como el rendimiento, la virtualización y el aislamiento, y la asignación de recursos y la programación; Sistemas Paralelos y Distribuidos incluye fundamentos de paralelismo; y Aseguramiento de la Información y de Seguridad incluye los forenses y los problemas de seguridad en profundidad. Muchos cursos de Sistemas Operativos dibujarán material a través de estas áreas de conocimiento.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.11.1 Visión general de Sistemas Operativos (Pág. 72) | | | No |
| 2.11.2 Principios de Sistemas Operativos (Pág. 73) | | | No |
| 2.11.3 Concurrencia (Pág. 73) | 3 | | No |
| 2.11.4 Planificación y despacho (Pág. 74) | 3 | | No |
| 2.11.5 Manejo de memoria (Pág. 74) | 3 | | No |
| 2.11.6 Seguridad y protección (Pág. 75) | 2 | | No |
| 2.11.7 Máquinas virtuales (Pág. 75) | | | No |
| 2.11.8 Manejo de dispositivos (Pág. 75) | | | No |
| 2.11.9 Sistema de archivos (Pág. 76) | | | No |
| 2.11.10 Sistemas empotrados y de tiempo real (Pág. 76) | | | No |
| 2.11.11 Tolerancia a fallas (Pág. 77) | | | No |
| 2.11.12 Evaluación del desempeño de sistemas (Pág. 77) | | | No |

### 2.11.1. OS/Visión general de Sistemas Operativos

**Temas:**
**Core Tier1**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Explicar los objetivos y funciones de un sistema operativo moderno [Familiarizarse]
2. Analizar las ventajas y desventajas inherentes en el diseño de un sistema operativo [Usar]
3. Describir las funciones de un sistema operativo contemporaneo respecto a conveniencia, eficiencia, y su habilidad para evolucionar [Familiarizarse]
4. Discutir acerca de sistemas operativos cliente-servidor, en red, distribuidos y cómo se diferencian de los sistemas operativos de un solo usuario [Familiarizarse]
5. Identificar amenazas potenciales a sistemas operativos y las características del diseño de seguridad para protegerse de ellos [Familiarizarse]

### 2.11.2. OS/Principios de Sistemas Operativos

**Temas:**
**Core Tier1**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Explicar el concepto de una capa lógica [Familiarizarse]
2. Explicar los beneficios de construir capas abstractas en forma jerárquica [Familiarizarse]
3. Describir el valor de la API y *middleware* [Evaluar]
4. Describir como los recursos computacionales son usados por aplicaciones de software y administradas por el software del sistema [Familiarizarse]
5. Contrastar el modo *kernel* y modo usuario en un sistema operativo [Usar]
6. Discutir las ventajas y desventajas del uso de procesamiento interrumpido [Familiarizarse]
7. Explicar el uso de una lista de dispositivos y el controlador de colas de entrada y salida [Familiarizarse]

### 2.11.3. OS/Concurrencia (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir la necesidad de concurrencia en el marco de un sistema operativo [Familiarizarse]
2. Demostrar los potenciales problemas de tiempo de ejecución derivados de la operación simultánea de muchas tareas diferentes [Usar]
3. 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]
4. Explicar los diferentes estados por los que una tarea debe pasar y las estructuras de datos necesarias para el manejo de varias tareas [Familiarizarse]
5. Resumir las técnicas para lograr sicronización en un sistema operativo(por ejemplo, describir como implementar semáforos usando primitivas del sistema operativo.) [Familiarizarse]
6. Describir las razones para usar interruptores, despacho, y cambio de contexto para soportar concurrencia en un sistema operativo [Familiarizarse]
7. Crear diagramas de estado y transición para los dominios de problemas simples [Usar]

### 2.11.4. OS/Planificación y despacho (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Planificación preventiva y no preferente.
- Planificadores y políticas.
- Procesos y subprocesos.
- Plazos y cuestiones en tiempo real.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. 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 [Usar]

2. Describir las relaciones entre los algoritmos de planificación y dominios de aplicación [Familiarizarse]

3. Discutir los tipos de planeamiento de procesos *scheduling* de corto, a mediano, a largo plazo y I/O [Familiarizarse]

4. Describir las diferencias entre procesos y hebras [Usar]

5. Comparar y contrastar enfoques estáticos y dinámicos para *scheduling* en tiempo real [Usar]

6. Hablar sobre la necesidad de tiempos límites de *scheduling* [Familiarizarse]

7. 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 [Usar]

### 2.11.5. OS/Manejo de memoria (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Revisión de la memoria física y hardware de gestión de memoria.
- Conjuntos de trabajo y thrashing.
- El almacenamiento en caché

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explicar la jerarquía de la memoria y costo-rendimiento de intercambio [Familiarizarse]

2. Resumir los principios de memoria virtual tal como se aplica para el almacenamiento en cache y paginación [Familiarizarse]

3. Evaluar las ventajas y desventajas en términos del tamaño de memoria (memoria principal, memoria caché, memoria axiliar) y la velocidad del procesador [Evaluar]

4. Defiende las diferentes formas de asignar memoria a las tareas, citando las ventajas relativas de cada uno [Evaluar]

5. 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]

6. 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]

### 2.11.6. OS/Seguridad y protección (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Articular la necesidad para la protección y seguridad en un sistema operativo [Evaluar]
2. Resumir las caracteristicas y limitaciones de un sistema operativo usado para proporcionar protección y seguridad [Familiarizarse]
3. Explicar el mecanismo disponible en un OS para controlar los accesos a los recursos [Familiarizarse]
4. 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 [Usar]

### 2.11.7. OS/Máquinas virtuales

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar el concepto de memoria virtual y la forma cómo se realiza en hadware y software [Familiarizarse]
2. Diferenciar emulación y el aislamiento [Familiarizarse]
3. Evaluar virtualización de compensaciones [Evaluar]
4. Discutir sobre hipervisores y la necesidad para ellos en conjunto con diferentes tipos de hipervisores [Usar]

### 2.11.8. OS/Manejo de dispositivos

**Temas:**
**Electivo**

- Características de los dispositivos serie y paralelo.
- Haciendo de abstracción de dispositivos.
- Estrategias de buffering.
- Acceso directo a memoria.
- La recuperación de fallos.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explique la diferencia clave entre dispositivos seriales y paralelos e identificar las condiciones en las cuales cada uno es apropiado [Familiarizarse]
2. Identificar la relación entre el hardware físico y los dispositivos virtuales mantenidos por el sistema operativo [Usar]
3. Explique *buffering* y describir las estrategias para su aplicación [Familiarizarse]
4. 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 [Usar]
5. Describir las ventajas y desventajas de acceso directo a memoria y discutir las cirscunstancias en cuales se justifica su uso [Usar]
6. Identificar los requerimientos para recuperación de errores [Familiarizarse]
7. Implementar un controlador de dispositivo simple para una gama de posibles equipos [Usar]

### 2.11.9. OS/Sistema de archivos

**Temas:**
**Electivo**

- 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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describir las decisiones que deben tomarse en el diseño de sistemas de archivos [Familiarizarse]
2. Comparar y contrastar los diferentes enfoques para la organización de archivos, el reconocimiento de las fortalezas y debilidades de cada uno. [Usar]
3. 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]
4. Resumir el uso de diarios y como los sistemas de archivos de registro estructurado mejora la tolerancia a fallos [Familiarizarse]

### 2.11.10. OS/Sistemas empotrados y de tiempo real

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describir que hace a un sistema un sistema en tiempo real [Familiarizarse]
2. Explicar la presencia y describir las características de latencia en sistemas de tiempo real [Familiarizarse]

3. Resumir los problemas especiales que los sistemas en tiempo real presentan, incluyendo el riesgo, y cómo se tratan estos problemas [Familiarizarse]

### 2.11.11. OS/Tolerancia a fallas

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar la importancia de los términos tolerancia a fallos, fiabilidad y disponibilidad [Familiarizarse]
2. Explicar en términos generales la gama de métodos para implementar la tolerancia a fallos en un sistema operativo [Familiarizarse]
3. Explicar cómo un sistema operativo puede continar funcionando después de que ocurra una falla [Familiarizarse]

### 2.11.12. OS/Evaluación del desempeño de sistemas

**Temas:**
**Electivo**

- ¿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)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describir las medidas de rendimiento utilizados para determinar cómo el sistema funciona [Familiarizarse]
2. Explicar los principales modelos de evaluación utilizados para evaluar un sistema [Familiarizarse]

## 2.12. Desarrollo basados en plataforma (PBD)

Desarrollo basado en la plataforma tiene que ver con el diseño y desarrollo de aplicaciones de software que residen en plataformas de software específicos. En contraste con la programación de propósito general, el desarrollo basado en la plataforma tiene en cuenta las limitaciones específicas de la plataforma. Por ejemplo la programación web, desarrollo multimedia, informática móvil, desarrollo de aplicaciones, y la robótica son ejemplos de plataformas relevantes que proporcionan servicios específicos / API / hardware que limitan el desarrollo. Estas plataformas se caracterizan por el uso de APIs especializadas, mecanismos de entrega / de actualización distintos, y se resumieron lejos del nivel de la máquina. Desarrollo basado en la plataforma se puede aplicar sobre una amplia extensión de los ecosistemas.

Si bien reconocemos que algunas plataformas (por ejemplo, desarrollo web) son prominentes, también somos conscientes del hecho de que ninguna plataforma en particular se debe especificar como requisito en los lineamientos curriculares. En consecuencia, esta área de conocimiento destaca muchas de las plataformas thathave hecho popular, sin incluir dicha plataforma en el plan de estudios básico.

Tomamos nota de que la habilidad general de desarrollo con respecto a una API o un entorno restringido está cubierta en otras áreas de conocimiento, tales como Fundamentos de Desarrollo de Software (SDF). Desarrollo basado en la Plataforma enfatiza aún más esas habilidades generales en el contexto de las plataformas particulares.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.12.1 Introducción (Pág. 78) | | | No |
| 2.12.2 Plataformas web (Pág. 78) | | | No |
| 2.12.3 Plataformas móviles (Pág. 79) | | | No |
| 2.12.4 Plataformas industriales (Pág. 79) | | | No |
| 2.12.5 Plataformas para video juegos (Pág. 79) | | | No |

### 2.12.1. PBD/Introducción

Esta unidad de conocimiento describe las diferencias fundamentales que la plataforma basada en el Desarrollo tiene sobre el desarrollo de software tradicional.

**Temas:**

**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Describir cómo el desarrollo basado en plataforma difiere de la programación de proposito general [Familiarizarse]
2. Listar las características de lenguajes de plataforma [Familiarizarse]
3. Escribir y ejecutar un programa simple basado en plataforma [Usar]
4. Listar las ventajas y desventajas de la programación con restricciones de plataforma [Familiarizarse]

### 2.12.2. PBD/Plataformas web

**Temas:**

**Electivo**

- Lenguajes de programación web (e.g., HTML5, Javascript, PHP, CSS)
- Restricción de plataformas web.
- Software como servicio.
- Estándares web.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Diseñar e implementar una aplicación web sencilla [Usar]
2. Describir las limitaciones que la web pone a los desarrolladores [Familiarizarse]
3. Comparar y contrastar la programación web con la programación de proposito general [Evaluar]
4. Describir las diferencias entre software como un servicio y productos de software tradicionales [Familiarizarse]
5. Discutir cómo los estándares de web impactan el desarrollo de software [Familiarizarse]
6. Revise una aplicación web existente con un estándar web actual [Evaluar]

### 2.12.3. PBD/Plataformas móviles

**Temas:**
**Electivo**

- Lenguajes de Programación para Móviles.
- Desafíos con mobilidad y comunicación inalámbrica.
- Aplicaciones Location-aware.
- Rendimiento / Compensación de Potencia.
- Restricciones de las Plataformas Móviles.
- Tecnologías Emergentes.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Diseñar e implementar una aplicación móvil para una plataforma móvil dada [Usar]
2. Discutir las limitaciones que las plataformas móviles ponen a los desarrolladores [Familiarizarse]
3. Discutir el rendimiento vs perdida de potencia [Familiarizarse]
4. Compare y contraste la programación móvil con la programación de proposito general [Evaluar]

### 2.12.4. PBD/Plataformas industriales

**Temas:**
**Electivo**

- Tipos de Plataformas Industriales (Matemática, Robótica, Control Industrial)
- Software para Robótica y su Arquitectura.
- Lenguajes de Dominio Específico.
- Restricciones de las Plataformas Industiales.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Diseñar e implementar una aplicación industrial en una plataforma dada (por ejemplo, usando *Lego Mindstorms* o *Matlab*) [Usar]
2. Comparar y contrastar lenguajes específicos de dominio con los lenguajes de programación de porposito general [Evaluar]
3. Discutir las limitaciones que una dada plataforma industrial impone a los desarrolladores [Familiarizarse]

### 2.12.5. PBD/Plataformas para video juegos

**Temas:**
**Electivo**

- Tipos de Plataformas de Juego (ej. XBox, Wii, PlayStation)
- Lenguajes de Plataformas de Juego (ej. C++, Java, Lua, Python)
- Restricciones de las Plataformas de Juegos.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Diseñar e implementar una aplicación simple en una plataforma de juego. [Usar]
2. Describir las limitaciones que las plataformas de juego imponen en los desarrolladores [Familiarizarse]
3. Comparar y contrastar la programación de juegos con la programación de proposito general [Evaluar]

## 2.13. Computación paralela y distribuída (PD)

La última década ha traído un crecimiento explosivo en multiprocesador computación, incluyendo los procesadores de varios núcleos y centros de datos distribuidos. Como resultado, la computación paralela y distribuida se ha movido de un tema ampliamente electiva a ser más de un componente central de los planes de estudios de computación de pregrado. Tanto la computación paralela y distribuida implica la ejecución lógicamente simultánea de múltiples procesos, cuyas operaciones tienen el potencial para intercalar de manera compleja. La computación paralela y distribuida construye sobre cimientos en muchas áreas, incluyendo la comprensión de los conceptos fundamentales de los sistemas, tales como la concurrencia y la ejecución en paralelo, la consistencia en el estado / manipulación de la memoria, y la latencia. La comunicación y la coordinación entre los procesos tiene sus raíces en el paso de mensajes y modelos de memoria compartida de la computación y conceptos algorítmicos como atomicidad, el consenso y espera condicional. El logro de aceleración en la práctica requiere una comprensión de algoritmos paralelos, estrategias para la descomposición problema, arquitectura de sistemas, estrategias de implementación detallados y análisis de rendimiento y el ajuste. Los sistemas distribuidos destacan los problemas de la seguridad y tolerancia a fallos, hacen hincapié en el mantenimiento del estado replicado, e introducen problemas adicionales que el puente a las redes de computadoras.

Debido paralelismo interactúa con tantas áreas de la computación, incluyendo al menos algoritmos, lenguajes, sistemas, redes y hardware, muchos planes de estudio pondrán diferentes partes del área de conocimiento en diferentes cursos, en lugar de en un curso dedicado. Si bien reconocemos que la informática se está moviendo en esa dirección y puede llegar a ese punto, en 2013 este proceso se encuentra aún en proceso de cambio y creemos que ofrece una guía más útil para diseñadores curriculares para agregar los temas fundamentales de paralelismo en un solo lugar. Tenga en cuenta, sin embargo, que los fundamentos de la concurrencia y la exclusión mutua aparecen en el área de conocimiento Fundamentos de Sistemas (SF). Muchos planes de estudios pueden optar por introducir el paralelismo y la concurrencia en el mismo curso (ver más abajo para la distinción prevista por estos términos). Además, observamos que los temas y resultados de aprendizaje que figuran a continuación incluyen sólo breves menciones de cobertura puramente electiva. En la actualidad, hay demasiada diversidad en temas que comparten poco en común (incluyendo por ejemplo, la computación científica paralela, cálculos de procesos y estructuras de datos no-bloqueo) para recomendar determinados temas se tratarán en cursos electivos.

Debido a la terminología de la computación paralela y distribuida varía entre las comunidades, ofrecemos aquí una breve descripción de los sentidos previstos de algunos términos. Esta lista no es exhaustiva ni definitiva, pero se proporciona para mayor claridad.

1. Paralelismo: El uso de los recursos computacionales adicionales simultáneamente, por lo general durante la aceleración. 2. Concurrencia: gestionar de manera eficiente y correcta el acceso simultáneo a los recursos. 3. Actividad: Un cálculo que pueden ejecutarse simultáneamente con los demás; por ejemplo, un programa, proceso, hilo, o activa componente de hardware paralelo. 4. Atomicity: Normas y propiedades de legalidad de una acción es observacional indivisible; por ejemplo, el establecimiento de todos los bits en una palabra, la transmisión de un único paquete, o completar una transacción. 5. Consenso artículo: Acuerdo entre dos o más actividades alrededor de un predicado dado; por ejemplo, el valor de un contador, el propietario de una cerradura, o la terminación de un hilo. 6. Consistencia: Las reglas y propiedades que rigen el acuerdo sobre los valores de las variables escritos o mensajes producidos, por algunas actividades y usados por otros (así posiblemente exhibiendo una raza de datos); por ejemplo, la consistencia secuencial, indicando que los valores de todas las variables de un programa paralelo de memoria compartida son equivalentes a la de un único programa de la realización de algún intercalado de la memoria de accesos de estas actividades.

7. Multicast: Un mensaje enviado a varios destinatarios, posiblemente, en general sin restricciones acerca de si algunos destinatarios reciban el mensaje antes que otros. Un evento es un mensaje de multidifusión enviado a un conjunto designado de oyentes o suscriptores.

Como multi-procesador de computación continúa creciendo en los próximos años, también lo hará el papel de la computación paralela y distribuida en los programas informáticos de pregrado. Además de las directrices que se presentan aquí, también nos dirigimos al lector interesado al documento titulado “NSF/TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduates””, disponible en el sitio web: http://www.cs.gsu.edu/ tcpp/curriculum/.

Nota general de referencias cruzadas: Fundamentos de Sistemas también contiene una introducción al paralelismo (Paradigmas computacionales, Paralelismo).

La introducción al paralelismo en SF complementa el uno aquí y no hay restricción de pedido entre ellos. En San Francisco, la idea es proporcionar una visión unificada del apoyo del sistema para su ejecución simultánea en varios niveles de abstracción (el paralelismo es inherente a las puertas, procesadores, sistemas operativos y servidores), mientras que aquí la atención se centra en una comprensión preliminar de paralelismo como la computación primitivo y las complicaciones que surgen en paralelo y la programación concurrente. Teniendo en cuenta estas diferentes perspectivas, las horas asignadas a cada uno no son redundantes: ver los sistemas de capas y los conceptos computacionales de alto nivel se contabilizan por separado en termsof las horas centrales.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.13.1 Fundamentos de paralelismo (Pág. 81) | | | No |
| 2.13.2 Descomposición en paralelo (Pág. 81) | 2 | | No |
| 2.13.3 Comunicación y coordinación (Pág. 82) | 3 | | No |
| 2.13.4 Análisis y programación de algoritmos paralelos (Pág. 83) | 3 | | No |
| 2.13.5 Arquitecturas paralelas (Pág. 84) | 2 | | No |
| 2.13.6 Desempeño en paralelo (Pág. 85) | | | No |
| 2.13.7 Sistemas distribuídos (Pág. 85) | | | No |
| 2.13.8 Cloud Computing (Pág. 86) | | | No |
| 2.13.9 Modelos y semántica formal (Pág. 86) | | | No |

### 2.13.1. PD/Fundamentos de paralelismo

Hacer que los estudiantes estén familiarizados con las nociones básicas de una ejecución paralela, un concepto de observado en Fundamentos de Sistemas, para resolver problemas de compilación que derivan de estas nociones, al igual que las condiciones de carrera (*Race Conditions*) y vida Paradigmas computacionales,Paralelismo.

**Temas:**

**Core Tier1**

- Procesamiento Simultáneo Múltiple.
- Metas del Paralelismo (ej. rendimineto) frente a Concurrencia (ej. control de acceso a recursos compartidos)
- Paralelismo, comunicación, y coordinación: 1. Paralelismo, comunicación, y coordinación 2. Necedidad de Sincronización
- Errores de Programación ausentes en programación secuencial: 1. Tipos de Datos ( lectura/escritura simultánea o escritura/escritura compartida) 2. Tipos de Nivél más alto (interleavings violating program intention, no determinismo no deseado) 3. Falta de vida/progreso (deadlock, starvation)

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Distinguir el uso de recursos computacionales para una respuesta mas rápida para administrar el acceso eficiente a un recurso compartido [Familiarizarse]

2. Distinguir múltiples estructuras de programación suficientes para la sincronización que pueden ser inter-implementables pero tienen ventajas complementarias [Familiarizarse]

3. Distinguir datos de carrera (*data races*) a partir de carreras de mas alto nivel [Familiarizarse]

### 2.13.2. PD/Descomposición en paralelo (2 horas Core-Tier1)

**Temas:**

**Core Tier1**

- Necesidad de Comunicación y coordinación/sincronización.
- Independencia y Particionamiento.

**Core Tier2**

- Conocimiento Básico del Concepto de Descomposición Paralela.
- Decomposición basada en tareas: 1. Implementación de estrategias como hebras
- Descomposición de Información Paralela 1. Estrategias como SIMD y MapReduce
- Actores y Procesos Reactivos (solicitud de gestores)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Explicar por qué la sincronización es necesaria en un programa paralelo especifico [Usar]
2. Identificar oportunidades para particionar un programa serial en módulos paralelos independientes [Familiarizarse]

**Core-Tier2:**

3. Escribir un algoritmo paralelo correcto y escalable [Usar]
4. Paralelizar un algoritmo mediante la aplicación de descomposición basada en tareas [Usar]
5. Paralelizar un algoritmo mediante la aplicación de descomposición datos en paralelo [Usar]
6. Escribir un programa usando actores y/o procesos reactivos [Usar]

### 2.13.3. PD/Comunicación y coordinación (3 horas Core-Tier1)

**Temas:**
**Core Tier1**

- Memoria Compartida.
- La consistencia, y su papel en los lenguaje de programación garantias para los programas de carrera libre.

**Core Tier2**

- Pasos de Mensaje: 1. Mensajes Punto a Punto versus multicast (o basados en eventos) 2. Estilos para enviar y recibir mensajes Blocking vs non-blocking 3. Buffering de mensajes
- Atomicidad: 1. Especificar y probar atomicidad y requerimientos de seguridad 2. Granularidad de accesos atómicos y actualizaciones, y uso de estructuras como secciones críticas o transacciones para describirlas 3. Exclusión mutua usando bloques, semáforos, monitores o estructuras relacionadas *a*) Potencial para fallas y bloqueos (*deadlock*) (causas, condiciones, prevención) 4. Composición *a*) Componiendo acciones atómicas granulares más grandes usando sincronización *b*) Transacciones, incluyendo enfoques optimistas y conservadores

**Electivo**

- Consensos: 1. (Ciclicos) barerras, contadores y estructuras relacionadas
- Acciones condicionales: 1. Espera condicional (p.e., empleando variables de condición)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Usar exclusión mútua para evitar una condición de carrera [Usar]
2. 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]

**Core-Tier2:**

3. Dar un ejemplo de un escenario en el que el bloqueo de mensajes enviados pueden dar *deadlock* [Usar]
4. Explicar cuándo y por qué mensajes de multidifusión (*multicast*) o basado en eventos puede ser preferible a otras alternativas [Familiarizarse]
5. Escribir un programa que termine correctamente cuando todo el conjunto de procesos concurrentes hayan sido completados [Usar]

6. Usar una apropiada cola de sincronización para almacenar temporalmente datos pasados entre actividades [Usar]

7. Explicar por qué verificaciones para precondiciones, y acciones basados en estas verificaciones, deben compartir la misma unidad de atomicidad para ser efectivo [Familiarizarse]

8. Escribir un programa de prueba que pueda revelar un error de programación concurrente; por ejemplo, falta una actualización cuando dos actividades intentan incrementar una variable [Usar]

9. Describir al menos una técnica de diseño para evitar [Familiarizarse]

10. Describir las ventajas relativas del control de concurrencia optimista y conservadora bajo diferentes tipos de carrera entre actualizaciones [Familiarizarse]

11. Dar un ejemplo de un escenario en el que un intento optimista de actualización puede nunca completarse [Familiarizarse]

**Elective:**

12. Usar semaforos o variables de condición para bloquear hebras hasta una necesaria precondición de mantenga [Usar]

### 2.13.4. PD/Análisis y programación de algoritmos paralelos (3 horas Core-Tier1)

**Temas:**
**Core Tier2**
- 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) 1. Algortimos específicos (p.e., MergeSort paralelo)

**Electivo**
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Definir: camino crítico, trabajo y *span* [Familiarizarse]

2. Calcular el trabajo y el *span* y determinar el camino crítico con respecto a un diagrama de ejecución paralela. [Usar]

3. Definir *speed-up* y explicar la noción de escalabilidad de un algoritmo en este sentido [Familiarizarse]

4. Identificar tareas independientes en un programa que debe ser paralelizado [Usar]

5. Representar características de una carga de trabajo que permita o evite que sea naturalmente paralelizable [Familiarizarse]

6. Implementar un algoritmo dividir y conquistar paralelo (y/o algoritmo de un grafo) y medir empiricamente su desempeño relativo a su analogo secuencial [Usar]

7. Descomponer un problema (por ejemplo, contar el número de ocurrencias de una palabra en un documento) via operaciones *map* y *reduce* [Usar]

**Elective:**

8. Proporcionar un ejemplo de un problema que se corresponda con el paradigma productor-consumidor [Familiarizarse]

9. Dar ejemplos de problemas donde el uso de *pipelining* sería un medio eficaz para la paralelización [Familiarizarse]

10. Implementar un algoritmo de matriz paralela [Usar]

11. Identificar los problemas que surgen en los algoritmos del tipo productor-consumidor y los mecanismos que pueden utilizarse para superar dichos problemas [Familiarizarse]

### 2.13.5. PD/Arquitecturas paralelas (2 horas Core-Tier1)

Los temas que se muestran aquí están relacionados con las unidades de conocimiento en la Arquitectura y Organización (AR) área de conocimiento (AR / ensamblaje nivel de máquina y AR / multiprocesamiento y Alternativa Architectures). Aquí, nos centramos en la arquitectura paralela desde el punto de vista de las aplicaciones, mientras que el área de conocimiento Arquitectura y Organización presenta el tema desde la perspectiva del hardware.

**Temas:**

**Core Tier1**

- Procesadores mutlinúcleo.
- Memoria compartida vs memoria distribuida.

**Core Tier2**

- Multiprocesamiento simétrico.
- SIMD, procesamiento de vectores.

**Electivo**

- GPU, coprocesamiento.
- Taxonomia de Flynn.
- Soporte a nivel de instrucciones para programación paralela. 1. Instrucciones atómicas como Compare/Set (Comparar / Establecer)
- Problemas de Memoria: 1. Caches multiprocesador y coherencia de cache 2. Acceso a Memoria no uniforme (NUMA)
- Topologías. 1. Interconecciones 2. Clusters 3. Compartir recursos (p.e., buses e interconexiones)

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Explicar las diferencias entre memoria distribuida y memoria compartida [Familiarizarse]

**Core-Tier2:**

2. Describir la arquitectura SMP y observar sus principales caracteristicas [Familiarizarse]

3. Distinguir los tipos de tareas que son adecuadas para máquinas SIMD [Familiarizarse]

**Elective:**

4. Describir las ventajas y limitaciones de GPUs vs CPUs [Familiarizarse]

5. Explicar las caracteristicas de cada clasificación en la taxonomía de Flynn [Familiarizarse]

6. Describir el soporte a de nivel de lenguaje ensamblador para las operaciones atómicas [Familiarizarse]

7. Describir los desafíos para mantener la coherencia de la caché [Familiarizarse]

8. Describir los desafíos clave del desempeño en diferentes memorias y topologías de sistemas distribuidos [Familiarizarse]

### 2.13.6. PD/Desempeño en paralelo

**Temas:**
**Electivo**

- Equilibrio de carga.
- La medición del desempeño.
- Programación y contención.
- Evaluación de la comunicación de arriba.
- Gestión de datos: 1. Costos de comunicación no uniforme debidos a proximidad 2. Efectos de Cache (p.e., false sharing) 3. Manteniendo localidad espacial
- Consumo de energía y gestión.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Detectar y corregir un desbalanceo de carga [Usar]
2. Calcular las implicaciones de la ley de Amdahl para un algoritmo paralelo particular [Usar]
3. Describir como la distribuición/disposición de datos puede afectar a los costos de comunicación de un algoritmo [Familiarizarse]
4. Detectar y corregir una instancia de uso compartido falso (*false sharing*) [Usar]
5. Explicar el impacto de la planificación en el desempeño paralelo [Familiarizarse]
6. Explicar el impacto en el desempeño de la localidad de datos [Familiarizarse]
7. Explicar el impacto y los puntos de equilibrio relacionados al uso de energía en el desempeño paralelo [Familiarizarse]

### 2.13.7. PD/Sistemas distribuídos

**Temas:**
**Electivo**

- Fallos: 1. Fallos basados en red (incluyendo particiones) y fallos basados en nodos 2. Impacto en garantías a nivel de sistema (p.e., disponibilidad)
- Envío de mensajes distribuido: 1. Conversión y transmisión de datos 2. Sockets 3. Secuenciamiento de mensajes 4. Almacenando *Buffering*, renviando y desechando mensajes
- Compensaciones de diseño para Sistemas Distribuidos: 1. Latencia versus rendimiento 2. Consistencia, disponibilidad, tolerancia de particiones
- Diseño de Servicio Distribuido: 1. Protocolos y servicios Stateful versus stateless 2. Diseños de Sesión (basados en la conexión) 3. Diseños reactivos (provocados por E/S) y diseños de múltiples hilos
- Algoritmos de Distribución de Núcleos: 1. Elección, descubrimiento

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Distinguir las fallas de red de otros tipos de fallas [Familiarizarse]
2. Explicar por qué estructuras de sincronización como cerraduras simples (*locks*) no son útiles en la presencia de fallas distribuidas [Familiarizarse]
3. Escribir un programa que realiza cualquier proceso de *marshalling* requerido y la conversión en unidades de mensajes, tales como paquetes, para comunicar datos importantes entre dos *hosts* [Usar]
4. Medir el rendimiento observado y la latencia de la respuesta a través de los *hosts* en una red dada [Usar]
5. Explicar por qué un sistema distribuido no puede ser simultaneamente Consistente (*Consistent*), Disponible (*Available*) y Tolerante a fallas (*Partition tolerant*). [Familiarizarse]

6. Implementar un servidor sencillo - por ejemplo, un servicio de corrección ortográfica [Usar]
7. Explicar las ventajas y desventajas entre: *overhead*, escalabilidad y tolerancia a fallas entre escojer un diseño sin estado (*stateless*) y un diseño con estado (*stateful*) para un determinado servicio [Familiarizarse]
8. Describir los desafios en la escalabilidad, asociados con un servicio cresciente para soportar muchos clientes, así como los asociados con un servicio que tendrá transitoriamente muchos clientes [Familiarizarse]
9. Dar ejemplos de problemas donde algoritmos de consenso son requeridos, por ejemplo, la elección de líder [Usar]

### 2.13.8. PD/Cloud Computing

**Temas:**
**Electivo**

- Computación a Escala de Internet: 1. Particionamiento de Tareas 2. Acceso a datos 3. Clusters, grids y mallas
- Servicios en la nube. 1. Infraestructura como servicio *a*) Elasticidad de recursos *b*) APIs de la Platforma 2. Software como servicio 3. Securidad 4. Administración del Costo
- Virtualización. 1. Gestión de recursos compartidos 2. Migración de procesos
- Almacenamiento de datos en la nube: 1. Acceso compartido a data stores de consistencia débil 2. Sincronización de datos 3. Particionamiento de datos 4. Sistemas de Archivos Distribuidos 5. Replicación

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Discutir la importancia de la elasticidad y administración de recursos en la Computación en la Nube (*Cloud Computing*) [Familiarizarse]
2. Explicar las estrategias para sincronizar una vista comun de datos compartidos a través de una colección de dispositivos [Familiarizarse]
3. Explicar las ventajas y desventajas de usar una infraestructura vistualizada [Familiarizarse]
4. Desplegar una aplicación que usa una infraestructura en la nube para computación y/o recursos de datos [Usar]
5. Apropiadamente particionar una aplicación entre un cliente y los recursos [Usar]

### 2.13.9. PD/Modelos y semántica formal

**Temas:**
**Electivo**

- Modelos formales de procesos y paso de mensajes, incluyendo algebras como Procesos Secuenciales de Comunicación (CSP) y Pi-Calculus
- Modelos formales de computación paralela, incluyendo la Máquina de Acesso Aleatorio Paralelo (PRAM) y alternativas como Bulk Synchronous Parallel (BSP)
- Modelos formales de dependencias computacionales.
- Modelos de consistencia (relajado) de memoria compartida y su relación con las especificaciones del lenguaje de programación.
- Criterios de corrección de algoritmos incluyendo (linearizability).
- Modelos de progreso algorítmic, incluyendo garantias de no bloqueo y equidad.
- Técnicas para especificar y comprobar las propiedades de corrección tales como atomicidad y la libertad de las carreras de datos.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Modelar un proceso concurrente usando un modelo formal, por ejemplo, cálculo pi [Usar]

2. Explicar las caracteristicas de un particular modelo paralelo formal [Familiarizarse]
3. Formalmente modelar un sistema de memoria compartida para mostrar y éste es consistente [Usar]
4. Usar un modelo para mostrar las garantias de progreso en un algoritmo paralelo [Usar]
5. Usar técnicas formales para mostrar que un algoritmo paralelo es correcto con respecto a la seguridad o la propiedad *liveness* [Usar]
6. Decidir si una ejecución específica es linealizable o no [Usar]

## 2.14. Lenguajes de programación (PL)

Los lenguajes de programación son el medio a través del cual los programadores describen con precisión los conceptos, formulan algoritmos, y la representan sus soluciones. Un científico de la computación con diferentes lenguajes, por separado o en conjunto. Los científicos de la computación deben entender los modelos de programación de los diferentes lenguajes y tomar decisiones de diseño basados en el lenguaje de programación y conceptos complementarios. El profesional a menudo necesitará aprender nuevos lenguajes y construcciones de programación y debe entender los fundamentos de como las características del lenguaje de programación estan definidas, compuestas, y implementadas. El uso eficaz de los lenguajes de programación, y la apreciación de sus limitaciones, también requiere un conocimiento básico de traducción de lenguajes de programación y su análisis de ambientes estáticos y dinámicos, así como los componentes de tiempo de ejecución tales como la gestión de memoria, entre otros detalles de relevancia.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.14.1 Programación orientada a objetos (Pág. 87) | 6 | | No |
| 2.14.2 Programación funcional (Pág. 88) | 4 | | No |
| 2.14.3 Programación reactiva y dirigida por eventos (Pág. 89) | 2 | | No |
| 2.14.4 Sistemas de tipos básicos (Pág. 89) | 4 | | No |
| 2.14.5 Representación de programas (Pág. 90) | 1 | | No |
| 2.14.6 Traducción y ejecución de lenguajes (Pág. 91) | 3 | | No |
| 2.14.7 Análisis de sintaxis (Pág. 91) | | | No |
| 2.14.8 Análisis semántico de compiladores (Pág. 92) | | | No |
| 2.14.9 Generación de código (Pág. 92) | | | No |
| 2.14.10 Sistemas de tiempo de ejecución (Pág. 92) | | | No |
| 2.14.11 Análisis estático (Pág. 93) | | | No |
| 2.14.12 Construcciones de programación avanzados (Pág. 93) | | | No |
| 2.14.13 Concurrencia y Paralelismo (Pág. 94) | | | No |
| 2.14.14 Sistemas de tipos (Pág. 94) | | | No |
| 2.14.15 Semántica formal (Pág. 95) | | | No |
| 2.14.16 Pragmática de lenguajes (Pág. 95) | | | No |
| 2.14.17 Programación lógica (Pág. 96) | | | No |

### 2.14.1. PL/Programación orientada a objetos (6 horas Core-Tier1)

**Temas:**
**Core Tier1**

- Diseño orientado a objetos: 1. Descomposición en objetos que almacenan estados y poseen comportamiento 2. 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.

**Core Tier2**

- Subtipificación: 1. Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos. 2. Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos. 3. Relación entre subtipos y la herencia.
- Lenguajes orientados a objetos para la encapsulación: 1. privacidad y la visibilidad de miembros de la clase 2. Interfaces revelan único método de firmas 3. clases base abstractas
- Uso de colección de clases, iteradores, y otros componentes de la libreria estandar.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Diseñar e implementar una clase [Usar]
2. Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar]
3. Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar]
4. 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]

**Core-Tier2:**

5. 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]
6. Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar]
7. 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]

### 2.14.2. PL/Programación funcional (4 horas Core-Tier1)

**Temas:**
**Core Tier1**

- El efecto de la programación libre: 1. Llamadas a función que no tiene efecto secundarios, para facilitar el razonamiento composicional 2. Variables inmutables, prevención de cambios no esperados en los datos del programa por otro código. 3. Datos que pueden ser subnombrados o copiados libremente sin introducir efectos no deseados del cambio
- Procesamiento de estructuras de datos (p.e. arboles) a través de fuciones con casos para cada variación de los datos. 1. Constructores asociados al lenguaje tales como uniones discriminadas y reconocimiento de patrones sobre ellos. 2. Funciones definidas sobre datos compuestos en términos de funciones aplicadas a las piezas constituidas.
- Funciones de primera clase (obtener, retornar y funciones de almacenamiento)

**Core Tier2**

- Cierres de función (funciones que usan variables en entornos léxicos cerrados) 1. Significado y definición básicos - creación de cierres en tiempo de ejecución mediante la captura del entorno. 2. Idiomas canónicos: llamadas de retorno, argumentos de iteradores, código reusable mediante argumentos de función 3. Uso del cierre para encapsular datos en su entorno 4. Evaluación y aplicación parcial
- Definición de las operaciones de orden superior en los agregados, especialmente en mapa, reducir / doblar, y el filtro.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Escribir algoritmos básicos que eviten asignación a un estado mutable o considerar igualdad de referencia [Usar]

2. Escribir funciones útiles que puedan tomar y retornar otras funciones [Usar]

3. 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]

**Core-Tier2:**

4. Razonar correctamente sobre variables y el ámbito léxico en un programa usando funciones de cierre (*function closures*) [Usar]

5. Usar mecanismos de encapsulamiento funcional, tal como *closures* e interfaces modulares [Usar]

6. Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, seleccionar la forma mas natural por cada lenguaje [Usar]

### 2.14.3. PL/Programación reactiva y dirigida por eventos (2 horas Core-Tier1)

Este material puede ser independiente o estar integrado con otras unidades de conocimiento sobre la concurrencia, la asincronía, y roscado para permitir contrastar hechos con hilos.
**Temas:**
**Core Tier2**

- Eventos y controladores de eventos.
- Usos canónicos como interfaces gráficas de usuario, dispositivos móviles, robots, servidores.
- Uso de frameworks reactivos. 1. Definición de controladores/oyentes (handles/listeners) de eventos. 2. Bucle principal de enventos no controlado po el escritor controlador de eventos (event-handler-writer)
- Eventos y eventos del programa generados externamente generada.
- La separación de modelo, vista y controlador.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Escribir manejadores de eventos para su uso en sistemas reactivos tales como GUIs [Usar]

2. Explicar porque el estilo de programación manejada por eventos es natural en dominios donde el programa reacciona a eventos externos [Familiarizarse]

3. Describir un sistema interactivo en términos de un modelo, una vista y un controlador [Familiarizarse]

### 2.14.4. PL/Sistemas de tipos básicos (4 horas Core-Tier1)

Las horas-core tier2 serían bien gastadas tanto en los temas estratégicos Core-Tier2 y en un tratamiento menos superficial de los resultados Core-Tier1 aprendizaje topicsand.
**Temas:**
**Core Tier1**

- Tipos como conjunto de valores junto con un conjunto de operaciones. 1. Tipos primitivos (p.e. números, booleanos) 2. 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.

- Metas y limitaciones de tipos estáticos 1. Eliminación de algunas clases de errores sin ejecutar el programa 2. Indecisión significa que un análisis estatico puede aproximar el comportamiento de un programa

**Core Tier2**

- Tipos genéricos (polimorfismo paramétrico) 1. Definición 2. Uso de librerías genéricas tales como colecciones. 3. Comparación con polimorfismo ad-hoc y polimorfismo de subtipos
- Beneficios complementarios de tipos estáticos y dinámicos: 1. Errores tempranos vs. errores tardíos/evitados. 2. Refuerzo invariante durante el desarrollo y mantenimiento del código vs. decisiones pospuestas de tipos durante la la creación de prototipos y permitir convenientemente la codificación flexible de patrones tales como colecciones heterogéneas. 3. Evitar el mal uso del código vs. permitir más reuso de código. 4. Detectar programas incompletos vs. permitir que programas incompletos se ejecuten

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse]
2. 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]
3. Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse]
4. Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar]
5. Dar un ejemplo de un programa que no verifique tipos en un lenguaje particular y sin embargo no tenga error cuando es ejecutado [Familiarizarse]
6. Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar]

**Core-Tier2:**

7. Explicar como las reglas de tipificación definen el conjunto de operaciones que legales para un tipo [Familiarizarse]
8. Escribir las reglas de tipo que rigen el uso de un particular tipo compuesto [Usar]
9. Explicar por qué indecidibilidad requiere sistemas de tipo para conservadoramente aproximar el comportamiento de un programa [Familiarizarse]
10. Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar]
11. Discutir las diferencias entre, genéricos (*generics*), subtipo y sobrecarga [Familiarizarse]
12. Explicar múltiples beneficios y limitaciones de tipificación estática en escritura, mantenimiento y depuración de un software [Familiarizarse]

### 2.14.5. PL/Representación de programas (1 horas Core-Tier1)

**Temas:**
**Core Tier1**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explicar como programas que procesan otros programas tratan a los otros programas como su entrada de datos [Familiarizarse]

2. Describir un árbol de sintaxis abstracto para un lenguaje pequeño [Usar]
3. Describir los beneficios de tener representaciones de programas que no sean cadenas de código fuente [Familiarizarse]
4. 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]

### 2.14.6. PL/Traducción y ejecución de lenguajes (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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: 1. Ejecución como código nativo o con una máquina virtual 2. 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: 1. Implementación de bucles, recursividad y llamadas de cola
- Gestión de memoria: 1. Gestión manual de memoria: asignación, limpieza y reuso de la pila de memoria 2. Gestión automática de memoria: recolección de datos no utilizados (*garbage colletion*) como una técnica automática usando la noción de accesibilidad

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. 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) [Familiarizarse]
2. Distinguir sintaxis y parseo de la semantica y la evaluación [Familiarizarse]
3. Bosqueje una representación de bajo nivel de tiempo de ejecución de construcciones del lenguaje base, tales como objetos o cierres (*closures*) [Familiarizarse]
4. 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 [Familiarizarse]
5. Identificar y corregir las pérdidas de memoria y punteros desreferenciados [Usar]
6. Discutir los beneficios y limitaciones de la recolección de basura (*garbage collection*), incluyendo la noción de accesibilidad [Familiarizarse]

### 2.14.7. PL/Análisis de sintaxis

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Usar gramáticas formales para especificar la sintaxis de los lenguajes [Usar]
2. Usar herramientas declarativas para generar parseadores y escáneres [Usar]

3. Identificar las características clave en las definiciones de sintaxis: ambiguedad, asociatividad, precedencia [Familiarizarse]

### 2.14.8. PL/Análisis semántico de compiladores

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. 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 [Usar]

2. Describir analizadores semanticos usando una gramatica con atributos [Usar]

### 2.14.9. PL/Generación de código

**Temas:**
**Electivo**

- Llamadas a procedimientos y métodos en envío.
- Compilación separada; vinculación.
- Selección de instrucciones.
- Calendarización de instrucciones.
- Asignación de registros.
- Optimización por rendija (peephole)

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Identificar todos los pasos esenciales para convertir automáticamente código fuente en código emsamblador o otros lenguajes de bajo nivel [Familiarizarse]

2. Generar código de bajo nivel para llamadas a funciones en lenguajes modernos [Usar]

3. Discutir por qué la compilación separada requiere convenciones de llamadas uniformes [Familiarizarse]

4. Discutir por qué la compilación separada limita la optimización debido a efectos de llamadas desconocidas [Familiarizarse]

5. 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*) [Familiarizarse]

### 2.14.10. PL/Sistemas de tiempo de ejecución

**Temas:**
**Electivo**

- Gestión dinámica de memoria, aproximaciones y técnicas: malloc/free, garbage collection (mark-sweep. copia, referencia), regiones (también conocidas como arenas o zonas)
- Disposición de datos para objetos y activación de registro.
- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Comparar los beneficios de diferentes esquemas de administración de memoria, usando conceptos tales como, fragmentación, localidad, y sobrecarga de memoria [Familiarizarse]
2. Discutir beneficios y limitaciones de la gestión automática de la memoria [Familiarizarse]
3. 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]
4. Discutir las ventajas, desventajas y dificultades del término (*just-in-time*) y recompilación automática [Familiarizarse]
5. Identificar los servicios proporcionados por los sistemas de tiempo de ejecución en lenguajes modernos [Familiarizarse]

### 2.14.11. PL/Análisis estático

**Temas:**
**Electivo**

- Representaciones relevantes de programas, tales como bloques básicos, grafos de control de flujos, cadenas de definiciones y asignación estática simple.
- Indecisión y consecuencias en el analisis de programas.
- Análisis y minúsculas de flujo, tales como la comprobación de tipos y puntero escalable y análisis de alias.
- Análisis sensibles al flujo, como hacia delante y hacia atrás de flujo de datos de análisis.
- Análisis sensibles de camino, como modelo de software de cheques.
- Herramientas y frameworks para definir análisis.
- Rol del análisis estático en la optimización de programas.
- Rol del análisis estático en la verificación parcial y busqueda de errores.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Definir análisis estáticos útiles en términos de un marco conceptual, como el análisis de flujo de datos [Usar]
2. Explicar por qué los análisis estáticos de tipos no triviales (*non-trivial sound static analyses*) deben ser aproximados [Familiarizarse]
3. Comunicar por qué un análisis es correcto (*sound and terminating*) [Usar]
4. Distinguir análisis de tipo: “puede” y “debe” [Familiarizarse]
5. Explicar por qué el *aliasing* potencial limita el análisis de tipos en los programas y como el análisis de alias puede ayudar [Familiarizarse]
6. Usar los resultados de un análisis estático para una optimización de un programa y/o la correctitud parcial de dicho programa [Usar]

### 2.14.12. PL/Construcciones de programación avanzados

**Temas:**
**Electivo**

- Evaluación perezosa y corrientes infinitas.
- Abstracciones de control: Control de excepciones, continuaciones, mónadas.
- Abstracciones orientadas a objetos: La herencia múltiple, mixins, Rasgos, métodos múltiples.
- Metaprogramación: Macros, programación generativa, el desarrollo basado en modelos.
- Sistemas modulares.

- Manipulación de cadenas a través de expresiones regulares.
- Evaluación dinámica de código (eval)
- Soporte de los lenguajes para verificación de (assert), invariantes, pre y post condiciones.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Usar diversas construcciones de programación avanzada correctamente [Usar]
2. Discutir cómo diversas construcciones de programación avanzada tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Familiarizarse]
3. Discutir cómo diversas construcciones de programación avanzada interactúan con la definición e implementación de otras características del lenguaje [Familiarizarse]

### 2.14.13. PL/Concurrencia y Paralelismo

Apoyo a la concurrencia es una cuestión de programación-idiomas fundamental con material rico en diseño de lenguajes de programación, implementación del lenguaje y la teoría del lenguaje. Debido a la cobertura en otras áreas de conocimiento, esta unidad de conocimiento electiva tiene como único objetivo complementar el material incluido en el Cuerpo de Conocimientos de otros lugares. Cursos sobre lenguajes de programación son un excelente lugar para incluir un tratamiento general de concurrencia incluyendo este otro material PDParallelandDistributedComputing, SFParallelism

**Temas:**
**Electivo**

- Construye para las variables de rosca-compartida y la sincronización de memoria compartida.
- Modelos de Actor.
- Futuros.
- Soporte de lenguaje para paralelismo de datos.
- Modelos para pasar mensajes entre procesos secuenciales.
- Efecto de modelos de consistenicia de memoria en la semántica del lenguaje y la correcta generación de código.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Escribir programas concurrentes correctos usando múltiples modelos de programación, tales como memoria compartida, actores, *futures* y primitivas de paralelismo de datos [Usar]
2. Usar un modelo de paso de mensajes para analizar un protocolo de comunicación [Usar]
3. Explicar porqué los lenguajes de programación no garantizan consistencia secuencial en la presencia de la carrera de datos (*data race*) y qué es lo que los programadores deben hacer como resultado de esto [Familiarizarse]

### 2.14.14. PL/Sistemas de tipos

**Temas:**
**Electivo**

- Constructores de tipo composicional, como tipos de producto (para agregados), tipos de suma (para uniones), tipos de función, tipos cuantificados y tipos recursivos.
- Comprobación de tipos.
- Seguridad de tipos como preservación más progreso.
- Inferencia de tipos.
- Sobrecarga estática.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Definir un sistema de tipo de forma precisa y en su composición [Usar]

2. Para varias construcciones de tipo fundamental, identificar los valores que describen y las invariantes que hacen que se cumplan [Familiarizarse]
3. Precisar las invariantes preservadas por un sistema de tipos seguro (*sound type system*) [Familiarizarse]
4. Demostrar la seguridad de tipos para un lenguaje simple en términos de conservación y progreso teoremas [Usar]
5. Implementar un algoritmo de inferencia de tipos basado en la unificación para un lenguaje básico [Usar]
6. Explicar cómo la sobrecarga estática y algoritmos de resolución asociados influyen el comportamiento dinámico de los programas [Familiarizarse]

### 2.14.15. PL/Semántica formal

**Temas:**
**Electivo**

- Sintáxis vs. semántica.
- Cálculo Lambda.
- Enfóques a semántica:Operacional, Denotativo, Axiomático.
- Demostración por inducción sobre lenguajes semánticos.
- Definición formal y demostración para sistemas de tipo.
- Parametricidad.
- Uso de semántica formal para modelamiento de sistemas.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Define una semántica formal para un lenguaje pequeño [Usar]
2. Escribe un programa en cálculo lambda y muestra su evaluación hacia un forma normal [Usar]
3. Discute los diversos enfoques de semánticas operacionales, de notación y axiomáticas [Familiarizarse]
4. Usa la inducción para demostrar las propiedades de todos los programas de un lenguaje [Usar]
5. Usa inducción para demostrar las propiedades de todos los programas de un lenguaje que es bien tipado de acuerdo a un sistema formalmente definido de tipos [Usar]
6. Usa parametricidad para establecer el comportamiento de un código dado solamente su tipo [Usar]
7. Usa semánticas formales para construir un modelo formal de un sistema de software en vez de un lenguaje de programación [Usar]

### 2.14.16. PL/Pragmática de lenguajes

**Temas:**
**Electivo**

- Principios de diseño de lenguaje tales como la ortogonalidad.
- Orden de evaluación, precedencia y asociatividad.
- Evaluación tardía vs. evaluación temprana.
- Definiendo controles y constructos de iteración.
- Llamadas externas y sistema de librerías.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Discute el rol de conceptos como ortogonalidad y el buen criterio de selección en el diseño de lenguajes [Familiarizarse]

2. Utiliza criterios objetivos y nítidos para evaluar las decisiones en el diseño de un lenguaje [Usar]

3. Da un ejemplo de un programa cuyo resultado puede diferir dado diversas reglas de orden de evaluación, precedencia, o asociatividad [Usar]

4. Muestra el uso de evaluación con retraso, como en el caso de abstracciones definidas y controladas por el usuario [Familiarizarse]

5. Discute la necesidad de permitir llamadas a librerias externas y del sistema y las consecuencias de su implementación en un lenguaje [Familiarizarse]

### 2.14.17. PL/Programación lógica

**Temas:**
**Electivo**
- Representación causal de estructura de datos y algoritmos.
- Unificación.
- Bactracking y busqueda.
- Cuts.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Usa un lenguaje lógico para implementar un algoritmo convencional [Usar]

2. Usa un lenguaje lógico para implementar un algoritmo empleando búsqueda implícita usando claúsulas, relaciones, y cortes [Usar]

## 2.15. Fundamentos del desarrollo de software (SDF)

La fluidez en el proceso de desarrollo de software es un requisito previo para el estudio de la mayor parte de la ciencia de la computación. Para utilizar las computadoras para resolver problemas de manera eficaz, los estudiantes deben ser competentes en los programas de lectura y escritura en múltiples lenguajes de programación. Más allá de conocimientos de programación, sin embargo, debe ser capaz de diseñar y analizar algoritmos, seleccionar paradigmas apropiados y utilizar el desarrollo moderno y herramientas de prueba. Esta área de conocimiento reúne a aquellos conceptos y habilidades fundamentales relacionadas con el proceso de desarrollo de software. Como tal, proporciona una base para otras áreas de conocimiento orientadas a software, sobre todo Lenguajes de Programación, Algoritmos y Complejidad, e Ingeniería de Software.

Es importante señalar que esta área de conocimiento es distinta de la antigua área conocimiento de de CC2001: Fundamentos de Programación . Considerando que dicha área de conocimiento se centró exclusivamente en los conocimientos de programación necesarios en un curso introductoria de ciencia de la computación, esta nueva área del conocimiento tiene la intención de llenar un propósito mucho más amplio. Se centra en todo el proceso de desarrollo de software, la identificación de los conceptos y habilidades que debe dominar en el primer año de un programa de ciencia de la computación. Esto incluye el diseño y análisis de algoritmos simples, conceptos fundamentales de programación, estructuras de datos, métodos y herramientas de desarrollo de software básicos. Como resultado de su propósito más amplio, esta área de conocimiento incluye conceptos y habilidades fundamentales que naturalmente podrían ser listados en otras áreas de conocimiento de software orientado a (por ejemplo, construcciones de programación de lenguajes de programación, análisis simple de Algoritmos y Complejidad, de desarrollo simple metodologías de Ingeniería de Software). Del mismo modo, cada una de estas áreas de conocimiento contendrá material más avanzado que se basa en los conceptos y habilidades fundamentales enumerados aquí.

Aunque más amplio que el área antigua de Fundamentos de Programación, esta área de conocimiento todavía permite una gran flexibilidad en el diseño de planes de estudios de primer año. Por ejemplo, la unidad Conceptos Fundamentales de Programación identifica sólo aquellos conceptos que son comunes a todos los paradigmas de programación. Se espera que un instructor seleccione uno o más paradigmas de programación (por ejemplo, la programación orientada a objetos, programación

funcional, scripting) para ilustrar estos conceptos de programación, y se retiraría contenido paradigma específico del área de conocimiento Lenguajes de Programación para completar un curso. Del mismo modo, un instructor puede optar por enfatizar el análisis formal (por ejemplo, Big-O, computabilidad) o metodologías de diseño (por ejemplo, proyectos de equipo, el ciclo de vida del software), integrando así las horas de áreas de conocimiento tales como Lenguajes de Programación, Algoritmos y Complejidad, y/o Software Ingeniería. Por lo tanto, las 43 horas de material en esta área de conocimiento típicamente serán aumentadas con material de núcleo de una o más de estas otras áreas de conocimiento para formar una experiencia completa y coherente de primer año.

Al considerar las horas asignadas a cada unidad de conocimiento, cabe señalar que estas horas reflejan la cantidad mínima de cobertura de clase necesaria para introducir el material. Muchos de los temas de desarrollo de software volverán a aparecer y ser completados por temas posteriores (por ejemplo, aplicando estructuras iterativas al procesar listas). Además, el dominio de los conceptos y habilidades de esta área de conocimiento requiere una cantidad significativa de experiencia en desarrollo de software fuera de la clase

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.15.1 Algoritmos y Diseño (Pág. 97) | | | No |
| 2.15.2 Conceptos Fundamentales de Programación (Pág. 98) | | | No |
| 2.15.3 Estructuras de Datos Fundamentales (Pág. 98) | | | No |
| 2.15.4 Métodos de Desarrollo (Pág. 99) | | | No |

### 2.15.1. SDF/Algoritmos y Diseño

Esta unidad construye las bases para los conceptos fundamentales del área de conocimiento de Algoritmos y Complejidad, de forma más clara en las unidades de Análisis Básico y Estrategias Algorítmicas.

**Temas:**

**Core Tier1**

- Conceptos y propiedades de los algoritmos 1. 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 1. Funciones matemáticas iterativas y recursivas 2. Recorrido iterativo y recursivo en estructura de datos 3. Estrategias Divide y Conquistar
- Conceptos y principios fundamentales de diseño 1. Abstracción 2. Descomposición de Program 3. Encapsulamiento y camuflaje de información 4. Separación de comportamiento y aplicación

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Discute la importancia de los algoritmos en el proceso de solución de un problema [Familiarizarse]
2. Discute como un problema puede ser resuelto por múltiples algoritmos, cada uno con propiedades diferentes [Familiarizarse]
3. Crea algoritmos para resolver problemas simples [Usar]
4. Usa un lenguaje de programación para implementar, probar, y depurar algoritmos para resolver problemas simples [Usar]
5. Implementa, prueba, y depura funciones recursivas simples y sus procedimientos [Usar]
6. Determina si una solución iterativa o recursiva es la más apropiada para un problema [Evaluar]
7. Implementa un algoritmo de divide y vencerás para resolver un problema [Usar]
8. Aplica técnicas de descomposición para dividir un programa en partes más pequeñas [Usar]
9. Identifica los componentes de datos y el comportamiento de mútiples tipos de datos abstractos [Usar]

10. Implementa un tipo de dato abstracto coherente, con la menor pérdida de acoplamiento entre componentes y comportamientos [Usar]

11. Identifica las fortalezas y las debilidades relativas entre múltiples diseños e implementaciones de un problema [Evaluar]

### 2.15.2. SDF/Conceptos Fundamentales de Programación

Esta unidad de conocimiento sienta las bases para los conceptos básicos en el área de conocimiento Lenguajes de programación, sobre todo en las unidades de paradigma específico: Programación orientada a objetos, programación funcional, programación reactiva y Event-Driven.

**Temas:**

**Core Tier1**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. 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]

2. Identifica y describe el uso de tipos de datos primitivos [Familiarizarse]

3. Escribe programas que usan tipos de datos primitivos [Usar]

4. Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar]

5. 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]

6. Escribe un programa que usa E/S de archivos para brindar persistencia a través de ejecuciones múltiples [Usar]

7. Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Evaluar]

8. Describe el concepto de recursividad y da ejemplos de su uso [Familiarizarse]

9. Identifica el caso base y el caso general de un problema basado en recursividad [Evaluar]

### 2.15.3. SDF/Estructuras de Datos Fundamentales

Esta unidad desarrolla las bases de los conceptos básicos en los Algoritmos y Area del conocimiento Complejo, sobre todo en las Estructuras de Datos y Algoritmos fundamentales y básicos computabilidad y unidades de conocimiento complejo.

**Temas:**

**Core Tier1**

- Arreglos
- Registros/estructuras (datos heterogéneos)
- Cadenas y procesamiento de cadenas.
- Tipos Abstractos de datos y sus implementaciones: 1. Pilas 2. Colas 3. Colas de prioridad 4. Conjuntos 5. Mapas

- Referencias y aliasing.
- Listas enlazadas
- Estrategias para escoger la estructura de datos apropiada.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Discute el uso apropiado de estructuras de datos incorporadas [Familiarizarse]
2. Describe aplicaciones comunes para cada de las siguientes estructuras de datos: pila, cola, cola de prioridad, conjunto y mapa [Familiarizarse]
3. Escribe programas que usen cada una de las siguientes estructuras de datos: arreglos, registros/estructuras, cadenas, listas enlazadas, pilas, colas, conjuntos, y mapas [Usar]
4. Compara implementaciones alternas de estructuras de datos con respecto a su rendimiento [Evaluar]
5. Describe cómo las referencias permiten que los objetos sean accesibles de diversas formas [Familiarizarse]
6. Compara y contrasta el costo y beneficio de implementar estructuras de datos dinámicas y estáticas [Evaluar]
7. Escoje la estructura de dato apropiada para modelar un problema determinado [Evaluar]

### 2.15.4. SDF/Métodos de Desarrollo

Esta unidad desarrolla las bases de los conceptos básicos en el área de conocimiento de Ingeniería de Software, más notablemente en los Procesos de Software, Diseño de software y unidades de conocimiento Software Evolution.

**Temas:**
**Core Tier1**

- Comprensión de programas.
- Exactitud del programa. 1. Tipos de error (sintaxis, logica, tiempo de ejecucion) 2. El concepto de la especificacion 3. Programación defensiva (ejem., codificación segura, manejo de excepciones) 4. Revision de codigo 5. Fundamentos de *testing* y generación de casos de prueba 6. El rol y el uso de contratos, incluyendo pre- y post- condiciones 7. Pruebas unitarias
- Refactorización simple.
- Entornos modernos de programación: 1. Búsqueda de código. 2. Programación usando libreria de componentes y sus APIs.
- Estrategias de depuración.
- Documentación y estilo de programas.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Traza la ejecución de una variedad de segmentos de código y escribir resúmenes de sus cálculos [Evaluar]
2. Explique por qué la creación de componentes de programa correctos es importante en la producción de software de alta calidad [Familiarizarse]
3. Identificar los errores de codificación comunes que conducen a programas inseguros (por ejemplo, desbordamientos de buffer, pérdidas de memoria, código malicioso) y aplicar estrategias para evitar este tipo de errores [Usar]
4. Llevar a cabo una revisión de código personal (centrado en los errores comunes de codificación) en un componente del programa utilizando una lista de verificación [Usar]
5. Contribuir en un pequeño grupo para una revisión de código centrada en la corrección de componentes [Usar]

6. Describir cómo un contrato se puede utilizar para especificar el comportamiento de un componente del programa [Familiarizarse]

7. Refactorizar un programa mediante la identificación de oportunidades para aplicar la abstracción del procesamiento [Usar]

8. Aplicar una variedad de estrategias para la prueba y depuración de programas sencillos [Usar]

9. Contruir, ejecutar y depurar programas usando un IDE moderno y herramientas asociadas tales como herramientas de pruebas de unidad y depuradores visuales [Usar]

10. Construir y depurar programas que utilizan las bibliotecas estándar disponibles con un lenguaje de programación elegido [Usar]

11. Analizar el grado en que el código de otro programador cumple con los estándares de documentación y de estilo de programación [Evaluar]

12. Aplicar los estandares de documentación y estilo de programación que contribuyan a la legibilidad y mantenimiento de software [Usar]

## 2.16. Ingeniería de Software (SE)

En cada dominio de la aplicación computación, la profesionalidad, la calidad, el cronograma y el costo son fundamentales para la producción de sistemas de software. Debido a esto, los elementos de la ingeniería de software se aplican al desarrollo de software en todas las áreas de la computación. Una amplia variedad de prácticas de ingeniería de software se han desarrollado y utilizado desde que la necesidad de la disciplina de la ingeniería de software fue reconocida por primera vez. También se han identificado muchos puntos de equilibrio entre estas diferentes prácticas. La práctica de la ingeniería de software tiene que seleccionar y aplicar las técnicas y prácticas adecuadas para un esfuerzo de desarrollo determinado con el fin de maximizar el valor. Para aprender a hacerlo así, ellos estudian los elementos de ingeniería de software.

La ingeniería de software es la disciplina que estudia la aplicación de la teoría, el conocimiento y la práctica de construir con eficacia y eficiencia los sistemas de software confiables que satisfacen los requisitos de los clientes y usuarios. Esta disciplina es aplicable a sistemas de gran escala pequeña, mediana . Abarca todas las fases del ciclo de vida de un sistema de software, incluyendo la obtención de requisitos, análisis y especificación; diseño; construcción; verificación y validación; despliegue; así como la operación y mantenimiento. Ya sea pequeña o grande, siguiendo un proceso tradicional plan de desarrollo dirigido, un enfoque ágil, o algún otro método, la ingeniería de software se refiere a la mejor manera de construir buenos sistemas de software.

Ingeniería del software utiliza métodos de ingeniería, procesos, técnicas y mediciones. Se beneficia de la utilización de herramientas para la gestión de desarrollo de software; análisis y modelado de artefactos de software ; evaluación y control de calidad; y para garantizar un enfoque disciplinado y controlado para la evolución del software y su reutilización. La caja de herramientas de ingeniería de software ha evolucionado a lo largo de los años. Por ejemplo, el uso de los contratos, con cláusulas que requiere y garantiza invariantes de clase, es una buena práctica que se ha vuelto más común. El desarrollo de software, que puede implicar un desarrollador individual o un equipo o equipos de desarrolladores, requiere la elección de las herramientas más apropiadas, métodos y enfoques para un entorno de desarrollo determinado.

Los estudiantes y los instructores tienen que entender los impactos de la especialización en los enfoques de ingeniería de software. Por ejemplo, los sistemas especializados incluyen:

1. Sistemas de tiempo real 2. Sistemas cliente-servidor 3. Sistemas distribuidos 4. Sistemas paralelas 5. sistemas basados en Web 6. Sistemas de alta integridad 7. Juegos 8. Computación móvil 9. Software de dominio específico (por ejemplo, la computación científica o aplicaciones de negocio)

Las cuestiones planteadas por cada uno de estos sistemas especializados exigen tratamientos específicos en cada fase de la ingeniería de software. Los estudiantes deben ser conscientes de las diferencias entre las técnicas y los principios de la ingeniería de software en general y las técnicas y los principios necesarios para abordar los problemas específicos de los sistemas especializados.

Un efecto importante de la especialización es que se puede necesitar tomar decisiones diferentes de material en la enseñanza de las aplicaciones de la ingeniería de software, como entre los diferentes modelos de procesos, diferentes enfoques para los sistemas de modelamiento, o diferentes opciones de técnicas para llevar a cabo cualquiera de las actividades clave. Esto se refleja en la asignación de núcleo y el material electivo, con los temas básicos y los resultados de aprendizaje centrados en los principios que subyacen a las distintas opciones, y los detalles de las diversas alternativas entre las que hay que decidir el material electivo .

Otra división de las prácticas de ingeniería de software es entre los interesados en la necesidad fundamental de desarrollar sistemas que implementan correctamente la funcionalidad que se requiere, los relacionados con otras cualidades de los sistemas y las compensaciones necesarias para equilibrar estas cualidades. Esta división también se refleja en la asignación de núcleo y el material electivo, por lo que los temas y resultados de aprendizaje relacionados con los métodos básicos para el desarrollo de tal sistema se asignan al núcleo y aquellos que se ocupan de otras cualidades y compensaciones entre ellos están asignados al material electivo.

En general, los estudiantes pueden aprender mejor a aplicar gran parte del material definido en el Ingeniería Sofware, al participar en un proyecto. Tales proyectos deben exigir a los estudiantes a trabajar en equipo para desarrollar un sistema de software a través de la mayor cantidad de su ciclo de vida como sea posible. Gran parte de la ingeniería de software se dedica a la comunicación efectiva entre los miembros del equipo y las partes interesadas. La utilización de los equipos de proyectos, los proyectos pueden ser lo suficientemente desafiantes para requerir a los estudiantes a utilizar técnicas de ingeniería de software efectivas para desarrollar y practicar sus habilidades de comunicación. Si bien la organización y realización de proyectos eficaces en el marco académico puede ser un reto, la mejor manera de aprender a aplicar la teoría de la ingeniería del software y el conocimiento está en el entorno práctico de un proyecto. Las horas mínimas especificadas para algunas unidades de conocimiento en este documento pueden aparecer insuficientes para lograr los resultados del aprendizaje a nivel de la aplicación asociada. Se debe entender que estos resultados deben ser alcanzados a través de la experiencia del proyecto que incluso puede ocurrir más adelante en el plan de estudios que cuando se introducen los temas dentro de la unidad de conocimiento.

Además, cada vez hay más evidencia de que los estudiantes aprenden a aplicar los principios de ingeniería de software con mayor eficacia a través de un enfoque iterativo, donde los estudiantes tienen la oportunidad de trabajar a través de un ciclo de desarrollo, evaluar su trabajo, y luego aplicar los conocimientos adquiridos a través de su evaluación a otro ciclo de desarrollo. Modelos de ciclo de vida ágiles e iterativos inherentemente ofrecen tales oportunidades.

La terminología del ciclo de vida del software en este documento se basa en la utilizada en las fuentes anteriores, como el Cuerpo de Conocimiento de la Software Ingeniería (SWEBOK) y las Directrices Curriculares ACM/IEEE-CS Software Engineering 2004 (SE2004). Mientras que algunos términos se definieron originalmente en el contexto de los procesos de desarrollo del plan impulsado, son tratados aquí como genérico y por lo tanto igualmente aplicable a procesos ágiles.

Nota: La unidad de conocimiento Métodos de Desarrollo incluye 9 horas Core Tier1-que constituyen una introducción a ciertos aspectos de la ingeniería de software. Las unidades de conocimiento, los temas y las especificaciones básicas horas de esta área de Ingeniería de Software deben ser entendidos como asumiendo una exposición previa al material descrito en el Métodos de Desarrollo.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.16.1 Procesos de Software (Pág. 102) | 1 | | No |
| 2.16.2 Gestión de Proyectos de Software (Pág. 103) | 2 | | No |
| 2.16.3 Herramientas y Entornos (Pág. 104) | 2 | | No |
| 2.16.4 Ingeniería de Requisitos (Pág. 105) | 3 | | No |
| 2.16.5 Diseño de Software (Pág. 106) | 5 | | No |
| 2.16.6 Construcción de Software (Pág. 108) | 2 | | No |
| 2.16.7 Verificación y Validación de Software (Pág. 109) | 3 | | No |
| 2.16.8 Evolución de Software (Pág. 110) | 2 | | No |
| 2.16.9 Fiabilidad de Software (Pág. 110) | 1 | | No |
| 2.16.10 Métodos Formales (Pág. 111) | | | No |

### 2.16.1. SE/Procesos de Software (1 horas Core-Tier1)

**Temas:**
**Core Tier1**
- 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): 1. Actividades con ciclos de vida de software.
- Programación a gran escala versus programación individual.

**Core Tier2**
- Evaluación de modelos de proceso de software.

**Electivo**
- Conceptos de calidad de software.
- Mejoramiento de procesos.
- Modelos de madurez de procesos de software.
- Mediciones del proceso de software.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Describa cómo el software puede interactuar y participar en varios sistemas, incluyendo la gestión de información, integración, control de procesos y sistemas de comunicaciones [Familiarizarse]
2. Describir las ventajas y desventajas relativas entre varios modelos importantes de procesos (por ejemplo, la cascada, iterativo y ágil) [Familiarizarse]
3. Describir las diferentes prácticas que son componentes clave de los diversos modelos de procesos [Familiarizarse]
4. Diferenciar entre las fases de desarrollo de software [Familiarizarse]
5. 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]

**Core-Tier2:**

6. Explicar el concepto de ciclo de vida del software y proporcionar un ejemplo que ilustra sus fases incluyendo los entregables que se producen [Familiarizarse]
7. 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]

**Elective:**

8. Definir la calidad del software y describir el papel de las actividades de aseguramiento de la calidad en el proceso de software [Familiarizarse]
9. Describir el objetivo y similitudes fundamentales entre los enfoques de mejora de procesos [Familiarizarse]
10. Comparar varios modelos de mejora de procesos, tales como CMM, CMMI, CQI, *Plan-Do-Check-Act*, o ISO9000 [Evaluar]
11. 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]
12. Explicar el papel de los modelos de madurez de procesos en la mejora de procesos [Familiarizarse]
13. Describir varias métricas de procesos para la evaluación y el control de un proyecto [Familiarizarse]
14. Usar las medidas en proyecto para describir el estado actual de un proyecto [Usar]

### 2.16.2. SE/Gestión de Proyectos de Software (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- La participación del equipo: 1. Procesos elemento del equipo, incluyendo responsabilidades de tarea, la estructura de reuniones y horario de trabajo 2. Roles y responsabilidades en un equipo de software 3. Equipo de resolución de conflictos 4. Los riesgos asociados con los equipos virtuales (comunicación, la percepción, la estructura)
- Estimación de esfuerzo (a nivel personal)
- Riesgo. 1. El papel del riesgo en el ciclo de vida 2. Categorías elemento de riesgo, incluyendo la seguridad, la seguridad, mercado, finanzas, tecnología, las personas, la calidad, la estructura y el proceso de

**Electivo**

- Gestión de equipos: 1. Organización de equipo y la toma de decisiones 2. Roles de identificación y asignación 3. Individual y el desempeño del equipo de evaluación
- Gestión de proyectos: 1. Programación y seguimiento de elementos 2. Herramientas de gestión de proyectos 3. Análisis de Costo/Beneficio
- Software de medición y técnicas de estimación.
- Aseguramiento de la calidad del software y el rol de las mediciones.
- Riesgo. 1. Identificación de riesgos y gestión. 2. Análisis riesgo y evaluación. 3. La tolerancia al riesgo (por ejemplo, riesgo adverso, riesgo neutral, la búsqueda de riesgo) 4. Planificación de Riesgo
- En todo el sistema de aproximación al riesgo, incluyendo riesgos asociados con herramientas.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Discutir los comportamientos comunes que contribuyen al buen funcionamiento de un equipo [Familiarizarse]
2. Crear y seguir un programa para una reunión del equipo [Usar]
3. Identificar y justificar las funciones necesarias en un equipo de desarrollo de software [Usar]
4. Entender las fuentes, obstáculos y beneficios potenciales de un conflicto de equipo [Usar]
5. Aplicar una estrategia de resolución de conflictos en un ambiente de equipo [Usar]
6. Utilizar un método ad hoc para estimar el esfuerzo de desarrollo del software (ejemplo, tiempo) y comparar con el esfuerzo actual requerido [Usar]
7. Listar varios ejemplos de los riesgos del software [Familiarizarse]
8. Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Familiarizarse]
9. Describir las diferentes categorías de riesgo en los sistemas de software [Familiarizarse]

**Elective:**

10. Demostrar a través de la colaboración de proyectos de equipo los elementos centrales de la contrucción de equipos y gestión de equipos [Usar]
11. Describir como la elección de modelos de procesos afectan la estructura organizacional de equipos y procesos de toma de decisiones [Familiarizarse]
12. Crear un equipo mediante la identificación de los roles apropiados y la asignación de funciones a los miembros del equipo [Usar]
13. Evaluar y retroalimentar a los equipos e individuos sobre su desempeño en un ambiente de equipo [Usar]

14. 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]
15. Realizar el seguimiento del progreso de alguna etapa de un proyecto que utiliza métricas de proyectos apropiados [Usar]
16. Comparar las técnicas simples de tamaño de software y estimación de costos [Usar]
17. Usar una herramienta de gestión de proyectos para ayudar en la asignación y rastreo de tareas en un proyecto de desarrollo de software [Usar]
18. Describir el impacto de la tolerancia de riesgos en el proceso de desarrollo de software [Evaluar]
19. Identificar riesgos y describir enfoques para manejar riesgos (evitar, aceptar, tranferir, mitigar) y caracterizar fortalezas y defectos para cada uno [Familiarizarse]
20. Explicar cómo el riesgo afecta las decisiones en el proceso de desarrollo de software [Usar]
21. Identificar los riesgos de seguridad para un sistema de software [Usar]
22. Demostrar un enfoque sistemático para la tarea de identificar los peligros y riesgos en una situación particular [Usar]
23. Aplicar los principios básicos del manejo de riesgos en una variedad de escenarios simples incluyendo una situación de seguridad [Usar]
24. Dirigir un análisis de costo/beneficio para el enfoque de mitigación de riesgos [Usar]
25. Identificar y analizar alguno de los riesgos para un sistema entero que surgen de aspectos distintos del software [Usar]

### 2.16.3. SE/Herramientas y Entornos (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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) 1. Integración continua.
- Mecanismos y conceptos de herramientas de integración.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir la diferencia entre el manejo de configuración de software centralizado y distribuido [Familiarizarse]
2. Describir como el control de versión puede ser usado para ayudar a manejar la administración de versiones del software [Familiarizarse]
3. Identificar elementos de configuración y usar herramientas de control de código fuente en un equipo de un proyecto pequeño [Usar]
4. Describir cómo la disponibilidad de herramientas de pruebas estática y dinámica se puede integrar en el entorno de desarrollo de software [Familiarizarse]
5. Describir los aspectos que son importantes en la selección de un conjunto de herramientas para el desarrollo de un sistema de software en particular, incluyendo herramientas para el seguimiento de los requisitos, modelado de diseño, implementación, construcción automática y pruebas [Familiarizarse]

6. Demostrar la capacidad de utilizar herramientas de software para apoyar el desarrollo de un producto de software de tamaño medio [Usar]

### 2.16.4. SE/Ingeniería de Requisitos (3 horas Core-Tier1)

El propósito de la ingeniería de requisitos es desarrollar un entendimiento común de las necesidades, prioridades y restricciones pertinentes a un sistema de software. Muchos fallos de software surgen de una comprensión incompleta de los requisitos para el software a desarrollar o manejo inadecuado de esos requisitos.

Especificaciones de los requisitos varían en trámite desde completamente informal (por ejemplo, habla) a rigor matemático (por ejemplo, escrito en un lenguaje de especificación formal, como Z o lógica de primer orden). En la práctica, los esfuerzos de ingeniería de software exitosos usan especificaciones de requisitos para reducir la ambigüedad y mejorar la coherencia y la integridad de la comprensión del equipo de desarrollo de la visión de los programas informáticos destinados. Enfoques del plan impulsado tienden a producir documentos formales con los requisitos numerados. Enfoques ágiles tienden a favorecer a las especificaciones menos formales que incluyen historias de usuario, casos de uso y casos de prueba.

**Temas:**

**Core Tier1**

- 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.

**Core Tier2**

- 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.

**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Enumerar los componentes clave de un caso de uso o una descripción similar de algún comportamiento que es requerido para un sistema [Familiarizarse]

2. Describir cómo el proceso de ingeniería de requisitos apoya la obtención y validación de los requisitos de comportamiento [Familiarizarse]

3. Interpretar un modelo de requisitos dada por un sistema de software simple [Familiarizarse]

**Core-Tier2:**

4. Describir los retos fundamentales y técnicas comunes que se utilizan para la obtención de requisitos [Familiarizarse]

5. Enumerar los componentes clave de un modelo de datos (por ejemplo, diagramas de clases o diagramas ER) [Familiarizarse]

6. Identificar los requisitos funcionales y no funcionales en una especificación de requisitos dada por un sistema de software [Usar]

7. 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 [Usar]

**Elective:**

8. 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 [Usar]

9. 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 [Familiarizarse]

10. Usar un método común, no formal para modelar y especificar los requisitos para un sistema de software de tamaño medio [Usar]

11. 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 [Usar]

12. Crear un prototipo de un sistema de software para reducir el riesgo en los requisitos [Usar]

13. Diferenciar entre el rastreo (*tracing*) hacia adelante y hacia atrás y explicar su papel en el proceso de validación de requisitos [Familiarizarse]

### 2.16.5. SE/Diseño de Software (5 horas Core-Tier1)

**Temas:**
**Core Tier1**

- 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.

**Core Tier2**

- 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: selección de componentes,diseño,adaptación y componentes de ensamblaje, componentes y patrones, componentes y objetos(por ejemplo,construir una GUI usando un standar widget set)
- Diseños de refactorización utilizando patrones de diseño

**Electivo**

- 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 orientación 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). 1. Principio de privilegios mínimos 2. Principio de falla segura por defecto 3. Principio de aceptabilidad psicológica

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Formular los principios de diseño, incluyendo la separación de problemas, ocultación de información, acoplamiento y cohesión, y la encapsulación [Familiarizarse]

2. 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]

3. Construir modelos del diseño de un sistema de software simple los cuales son apropiado para el paradigma utilizado para diseñarlo [Usar]

4. 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]

**Core-Tier2:**

5. Para un sistema simple adecuado para una situación dada, discutir y seleccionar un paradigma de diseño apropiado [Usar]

6. Crear modelos apropiados para la estructura y el comportamiento de los productos de software desde la especificaciones de requisitos [Usar]

7. Explicar las relaciones entre los requisitos para un producto de software y su diseño, utilizando los modelos apropiados [Evaluar]

8. 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]

9. 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]

10. Investigar el impacto de la selección arquitecturas de software en el diseño de un sistema simple [Evaluar]

11. Aplicar ejemplos simples de patrones en un diseño de software [Usar]

12. Describir una manera de refactorar y discutir cuando esto debe ser aplicado [Familiarizarse]

13. Seleccionar componentes adecuados para el uso en un diseño de un producto de software [Usar]

14. Explicar cómo los componentes deben ser adaptados para ser usados en el diseño de un producto de software [Familiarizarse]

15. Diseñar un contrato para un típico componente de software pequeño para el uso de un dado sistema [Usar]

**Elective:**

16. Discutir y seleccionar la arquitectura de software adecuada para un sistema de software simple para un dado escenario [Usar]

17. Aplicar modelos de cualidades internas y externas en el diseño de componentes de software para lograr un equilibrio aceptable entre los aspectos de calidad en conflictos [Usar]

18. Analizar un diseño de software desde la perspectiva de un atributo significativo de la calidad interna [Evaluar]

19. Analizar un diseño de software desde la perspectiva de un atributo significativo de calidad externa [Evaluar]

20. Explicar el papel de los objetos en los sistemas de middleware y la relación con los componentes [Familiarizarse]

21. Aplicar métodos orientado a componentes para el diseño de una amplia gama de software, tales como el uso de componentes para la concurrencia y transacciones, para los servicios de comunicación confiables, para la interacción con la base de datos que incluye los servicios de consulta remota y gestión de bases de datos, o para la comunicación segura y el acceso [Usar]

22. Refactorizar una implementación de software existente para mejorar algún aspecto de su diseño [Usar]

23. Determinar y aplicar los principios de mínimo privilegio y defectos-a prueba de errores [Familiarizarse]

### 2.16.6. SE/Construcción de Software (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Prácticas de codificación: técnicas, idiomas/patrones, mecanismos para construcción de programas de calidad: 1. Prácticas de codificación defensive 2. Prácticas de codificación segura 3. 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 : 1. Análisis de cambio impacto 2. Cambio de actualización

**Electivo**

- Los problemas de seguridad potenciales en los programas : 1. Buffer y otros tipos de desbordamientos 2. Condiciones elemento Race 3. Inicialización incorrecta, incluyendo la elección de los privilegios 4. Entrada Comprobación 5. Suponiendo éxito y corrección 6. La validación de las hipótesis

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Describir técnicas, lenguajes de codificación y mecanismos de implementación para conseguir las propiedades deseadas, tales como la confiabilidad, la eficiencia y la robustez [Familiarizarse]

2. Construir código robusto utilizando los mecanismos de manejo de excepciones [Usar]

3. Describir la codificación segura y prácticas de codificación de defensa [Familiarizarse]

4. Seleccionar y utilizar un estándar de codificación definido en un pequeño proyecto de software [Usar]

5. Comparar y contrastar las estrategias de integración incluyendo: de arriba hacia abajo (*top-down*), de abajo hacia arriba (*bottom-up*), y la integración Sándwich [Familiarizarse]

6. Describir el proceso de analizar e implementar los cambios a la base de código desarrollado para un proyecto específico [Familiarizarse]

7. Describir el proceso de analizar e implementar los cambios a una gran base de código existente [Familiarizarse]

**Elective:**

8. Reescribir un programa sencillo para eliminar vulnerabilidades comunes, tales como desbordamientos de búffer, desbordamientos de enteros y condiciones de carrera [Usar]

9. Escribir un componente de software que realiza alguna tarea no trivial y es resistente a errores en la entrada y en tiempo de ejecución [Usar]

### 2.16.7. SE/Verificación y Validación de Software (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- 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: 1. Pruebas de Unit, integración, validación y de Sistema 2. Creación de plan de pruebas y generación de casos de test 3. Técnicas de test de caja negra y caja blanca 4. 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.

**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Distinguir entre la validación y verificación del programa [Familiarizarse]
2. Describir el papel que las herramientas pueden desempeñar en la validación de software [Familiarizarse]
3. Realizar, como parte de una actividad de equipo, una inspección de un segmento de código de tamaño medio [Usar]
4. Describir y distinguir entre diferentes tipos y niveles de pruebas (unitaria, integracion, sistemas y aceptacion) [Familiarizarse]
5. Describir tecnicas para identificar casos de prueba representativos para integracion, regresion y pruebas del sistema [Familiarizarse]
6. Crear y documentar un conjunto de pruebas para un segmento de código de mediano tamaño [Usar]
7. Describir cómo seleccionar buenas pruebas de regresión y automatizarlas [Familiarizarse]
8. Utilizar una herramienta de seguimiento de defectos para manejar defectos de software en un pequeño proyecto de software [Usar]
9. Discutir las limitaciones de las pruebas en un dominio particular [Familiarizarse]

**Elective:**

10. Evaluar un banco de pruebas (*a test suite*) para un segmento de código de tamaño medio [Usar]
11. Comparar los enfoques estáticos y dinámicos para la verificación [Familiarizarse]
12. Identificar los principios fundamentales de los métodos de desarrollo basado en pruebas y explicar el papel de las pruebas automatizadas en estos métodos [Familiarizarse]
13. Discutir los temas relacionados con las pruebas de software orientado a objetos [Usar]

14. Describir las técnicas para la verificación y validación de los artefactos de no código [Familiarizarse]

15. Describir los enfoques para la estimación de fallos [Familiarizarse]

16. Estimar el número de fallos en una pequeña aplicación de software basada en la densidad de defectos y siembra de errores [Usar]

17. Realizar una inspección o revisión del de código fuente de un software para un proyecto de software de tamaño pequeño o mediano [Usar]

### 2.16.8. SE/Evolución de Software (2 horas Core-Tier1)

**Temas:**
**Core Tier2**
- Desarrollo de Software en el contexto de código grande pre existente 1. Cambios de software 2. Preocupaciones y ubicación de preocupaciones 3. *Refactoring*
- Evolución de Software.
- Características de Software mantenible.
- Sistemas de Reingeniería.
- Reuso de Software. 1. Segmentos de código 2. Bibliotecas y *frameworks* 3. Componentes 4. Líneas de Producto

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Identificar los problemas principales asociados con la evolución del software y explicar su impacto en el ciclo de vida del software [Familiarizarse]

2. Estimar el impacto del cambio de requerimientos en productos existentes de tamaño medio [Usar]

3. Usar refactorización en el proceso de modificación de un componente de sosftware [Usar]

4. Estudiar los desafios de mejorar sistemas en un entorno cambiante [Familiarizarse]

5. Perfilar los procesos de pruebas de regresión y su rol en el manejo de versiones [Familiarizarse]

6. Estudiar las ventajas y desventajas de diferentes tipos de niveles de confiabilidad [Familiarizarse]

### 2.16.9. SE/Fiabilidad de Software (1 horas Core-Tier1)

**Temas:**
**Core Tier2**
- Conceptos ingenieriles de confiabilidad de software.
- Confiabilidad de software, de sistemas y comportamiento de fallas.
- Conceptos y técnicas del ciclo de vida de fallas.

**Electivo**
- Modelos de confiabilidad de Software.
- Técnicas y modelos de software tolerante a fallas.
- Prácticas ingenieriles de confiabilidad de software.
- Análisis de confiabilidad de software basado en mediciones.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Estudiar los problemas que existe en lograr niveles muy altos de confiabilidad [Familiarizarse]

2. Describir como el software confiable contribuye a la confiabilidad del sistema [Familiarizarse]

3. Listar los enfoques para minimizar fallas que pueden ser aplicadas en cada etapa de el ciclo de vida del software [Familiarizarse]

**Elective:**

4. Comparar las caracteristicas de los tres diferentes métodos de modelización confiable [Familiarizarse]
5. Demostrar la capacidad de aplicar multiples métodos para desarrollar estimaciones confiables para un sistema de software [Usar]
6. Identificar métodos que conduzcan a la realización de una arquitectura de software que permita alcanzar un nivel especifico de fiabilidad [Usar]
7. Identificar formas de aplicar la redundancia para lograr tolerancia a fallos para una aplicación de tamaño medio [Usar]

### 2.16.10. SE/Métodos Formales

**Temas:**
**Electivo**

- Los tópicos listed a continuación tienen una fuerte dependencia con el material del Área de Conocimiento de Estructuras Discretas (DS).
- El rol de la especificación formal y técnicas de análisis en el ciclo de desarrollo de software
- Lenguajes controlar (assert) en un programa y abordajes de análisis (incluyendo lenguajes para escribir y analizar pre y post condiciones tales como OCL, JML)
- Abordajes formales para modelamiento y análisis de software. 1. Verificadores de modelos *Model checkers* 2. Buscadores de Modelo *Model finders*
- Herramientas para el soporte de métodos formales.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Describir la especificación formal del rol y técnicas de análisis que pueden jugar en el desarrollo de software complejo y comparar su uso como técnicas de validación y verificación con pruebas [Familiarizarse]
2. Aplicar especificación formal y técnicas de análisis para diseños de software y programas con baja complejidad [Usar]
3. Explicar los beneficios potenciales y desventajas de usar lenguajes de especificación formal [Familiarizarse]
4. Crear y evaluar validaciones de programa para una variedad de comportamientos que van desde lo simple hasta lo complejo [Usar]
5. Usando un lenguaje de especificación formal, formular la especificación de un sistema de software simple y derivar ejemplos de casos de prueba a partir de la especificación [Usar]

## 2.17. Fundamentos de Sistemas (SF)

La infraestructura subyacente de hardware y software sobre la que se construyen las aplicaciones se describe colectivamente por el término ”sistemas de computadores”. Los sistemas de computadores en general abarcan las subdisciplinas de sistemas operativos, sistemas paralelos y distribuidos, redes de comunicaciones y la arquitectura de computadores. Tradicionalmente, estas áreas se imparten en forma no integrada a través de cursos independientes. Sin embargo, estas subdisciplinas comparten cada vez más importantes conceptos fundamentales comunes dentro de sus respectivos núcleos. Estos conceptos incluyen paradigmas computacionales, paralelismo, comunicaciones entre capas, el estado y transición de estado, la asignación de recursos y la programación y así sucesivamente. El Área de Conocimiento de Fundamentos de Sistemas está diseñada para presentar una visión integradora de estos conceptos fundamentales en una unificada aunque de manera simplificada proporcionando una base común para los diferentes mecanismos especializados y políticas apropiadas para un área de dominio particular.

| área de Conocimiento (*Knowledge Area*-KA) (KA) | Core Tier1 | Core Tier2 | Electivo |
|---|---|---|---|
| 2.17.1 Paradigmas computacionales (Pág. 112) | | | No |
| 2.17.2 Comunicación a través de múltiples capas (Pág. 113) | | | No |
| 2.17.3 Estados y máquinas de estados (Pág. 113) | | | No |
| 2.17.4 Paralelismo (Pág. 114) | | | No |
| 2.17.5 Evaluación (Pág. 114) | | | No |
| 2.17.6 Asignación de recursos y planeamiento (Pág. 115) | 2 | | No |
| 2.17.7 Proximidad (Pág. 115) | 3 | | No |
| 2.17.8 Virtualización y aislamiento (Pág. 115) | 2 | | No |
| 2.17.9 Confiabilidad a través de redundancia (Pág. 116) | 2 | | No |
| 2.17.10 Evaluación cuantitativa (Pág. 116) | | | No |

### 2.17.1. SF/Paradigmas computacionales

El punto de vista que aquí se presenta es las múltiples representaciones de un sistema a través de capas, desde bloques de construcción de hardware hasta los componentes de aplicaciones y el paralelismo disponible en cada representación.

**Temas:**

**Core Tier1**

- Bloques básicos de construcción de un computador (compuertas, flip-flops, registros, interconexiones; Caminos de datos (Datapath) + Control + Memoria)
- Hardware como un paradigma computacional: bloques de construcción lógicos fundamentales; expresiones lógicas, minimización, formas de suma de productos.
- Procesamiento secuencial a nivel de aplicacion: single thread.
- Procesamiento paralelo simple a nivel de aplicación: niveles de pedidos (web-services, cliente-servidor, distribuído), una hebra simple por servidor, múltiples hebras con múltiples servidores.
- Concepto básico de la secuencia de ejecución (pipelining), etapas de superposición de procesamiento.
- Conceptos básicos de escalabilidad: ir más rápido vs manejar problemas grandes.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Enumerar los patrones comúnmente encontrados de cómo los cálculos son organizados [Familiarizarse]
2. Describir los bloques de construcción básicos de las computadoras y su rol en desarrollo histórico de la arquitectura del computador. [Familiarizarse]
3. Articular las diferencias entre un solo thread contra múltiples threads, un solo servidor contra modelos de servidores múltiples, motivados por ejemplos del mundo real (recetas de cocina, lineas de múltiples cajeros y parejas) [Familiarizarse]
4. Articular el concepto de escalabilidad fuerte y débil, es decir, cómo el rendimiento se ve afectado por la escala del problema contra escala de los recursos para resolver el problema. Esto puede ser motivado por lo simple, ejemplos en el mundo real [Familiarizarse]
5. Diseñar un circuito lógico simple usando los bloques de construcción fundamental del diseño lógico [Usar]
6. Usar herramientas para la captura, síntesis, y la simulación para evaluar un diseño lógico [Usar]
7. Escribir problema secuencial simple y una versión paralela simple de un mismo programa [Usar]
8. Evaluar el desempeño de las versiones simples secuenciales y paralelas de un programa con diferentes tamaños de problemas, y ser capaz de describir los planos de velocidad obtenidos [Evaluar]

### 2.17.2. SF/Comunicación a través de múltiples capas

**Temas:**
**Core Tier1**

- Abstracciones de programación, interfaces, uso de librerías.
- Distinción entre Aplicaciones y Servicios del SO, Llamadas a procedimientos remotos.
- Interacción entre la aplicación y la máquina virtual.
- Confiabilidad.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Describir cómo los sistemas computacionales están construídos con capas sobre capas basada en la separación de los problemas, con interfaces bien definidas, ocultando los detalles de las capas bajas a las capas superiores [Familiarizarse]

2. Describir que el hardware, VM, OS, aplicación son capas adicional interpretación/procesamiento [Familiarizarse]

3. Describir los mecanismos de como los errores se detectan, nos notificados hacia atrás y se manejan a través de capas [Familiarizarse]

4. Construir un programa simple utilizando métodos de capas, la detección y recuperación de errores, y reflexión de la condición de error a través de capas [Usar]

5. Encontrar *bugs* en un programa multi capas a través de la utilización de programas de rastreo, paso a paso y depuración *debugging* [Usar]

### 2.17.3. SF/Estados y máquinas de estados

**Temas:**
**Core Tier1**

- Sistemas Digitales vs Analógicos and Discretos vs Continuos.
- Compuertas lógicas simples, expresiones lógicas, simplificación lógica booleana.
- Relojes, estado, secuenciamiento.
- Lógica combinacional, lógica secuencial, registros, memorias.
- Computadoras y protocolos de red como ejemplos de estado de máquinas.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Describir los cálculos como un sistema que se caracteriza por un conjunto conocido de configuraciones con las transiciones de una configuración (estado) a otra (estado) [Familiarizarse]

2. Describir la distinción entre sistemas cuya salida es sólo una función de su entrada (combinacional) y los que tienen memoria/historia (secuencial) [Familiarizarse]

3. Describir una computadora como una máquina de estados que interpreta las instrucciones de la máquina [Familiarizarse]

4. Explicar cómo un programa o protocolo de red también se pueden expresar como una máquina de estados y que pueden existir representaciones alternativas para el mismo cálculo [Familiarizarse]

5. Desarrollar descripciones de máquinas de estado para soluciones de problemas de planteamiento simple (por ejemplo, la secuencia del semáforo, reconocedores de patrones) [Usar]

6. Deducir el comportamiento de series de tiempo de una máquina de estado a partir de su representación de estados de máquina [Evaluar]

### 2.17.4. SF/Paralelismo

**Temas:**
**Core Tier1**

- Procesamiento secuencial vs paralelo.
- Programación paralela vs concurrente.
- Paralelismo de solicitudes vs Paralelismo de tareas.
- Cliente-Servidor/Web Services, Hilos (Fork-join), Pipelining.
- Arquitecturas multinúcleo y soporte de hardware para sincronización.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Dado un programa, distinguir entre su ejecución secuencial y paralela y las implicaciones en el desempeño de los mismos [Familiarizarse]

2. Demostrar sobre una linea de tiempo de ejecución que los eventos y operaciones paralelas pueden tomar lugar simultaneamente (al mismo tiempo). Explicar como la carga de trabajo puede ser realizada en menos tiempo si se explora el paralelismo [Familiarizarse]

3. Explicar otros usos del paralelismo, tales como la confiabilidad/redundancia de la ejecución [Familiarizarse]

4. Definir las diferencias entre los conceptos de Paralelismo a nivel de Instrucción, Paralelismo a nivel de datos, Paralelimos/Multitareas a nivel de hebras, Paralelimos a nivel de Procesos/Peticiones [Familiarizarse]

5. Escribir mas de un programa en paralelo (por ejemplo, un programa paralelo en mas de un paradigma de programación paralela; un programa paralelo que administre recursos compartidos a través de primitivas de sincronización; un programa paralelo que realiza operaciones simultaneas sobre datos particionados a través de paralelización de tareas/procesos (por ejemplo, busqueda de términos en paralelo; un programa que realiza paso a paso un procesamiento *pipeline* a través de paso de mensajes)) [Usar]

6. Usar herramientas de desempeño para medir el *speed-up* alcanzado por un programa paralelo en términos de tamaño de los datos y número de recursos [Evaluar]

### 2.17.5. SF/Evaluación

**Temas:**
**Core Tier1**

- Cifras de desempeño de mérito.
- Las cargas de trabajo y puntos de referencia representativos y los métodos de recolección y análisis de las cifras de desempeño.
- La ecuación CPI (*Cycles per Instruction*) como una herramienta para entender puntos de equilibrio en el diseño de un conjunto de instrucciones, secuencia de ejecución en un procesador y organización de un sistema de memoria.
- La ley de Amdahl: la parte de la computación que no se puede acelerar limita el efecto de las partes que si pueden.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Explique cómo los componentes de la arquitectura del sistema contribuyen a la mejora del rendimiento [Familiarizarse]

2. Describir la ley de Amdahl y discutir sus limitaciones [Familiarizarse]

3. Diseñar e implementar un experimento orientado al desempeño [Usar]

4. Usar herramientas de software para perfilar y medir el desempeño de un programa [Evaluar]

### 2.17.6. SF/Asignación de recursos y planeamiento (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Clases de recursos (p.e., unidad de procesador, memoria, disco, ancho de banda neto)
- Clases de programación (p.e., FIFO, prioridad)
- Ventajas de la planificación equitativa, la programación preventiva.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Definir como recursos computacionales limitados (por ejemplo, participación de procesador, memoria, almacenamiento y ancho de banda) son gestionadas por una cuidadosa asignación para entidades existentes [Familiarizarse]
2. Describir los algoritmos de planificación mediante el cual, recursos son asignados a entidades competentes, y los factores de calidad por el cual se evaluan estos algoritmos, tal como equidad (*fairness*) [Familiarizarse]
3. Implementar un algoritmo básico de planificación [Usar]
4. Usar factores de calidad de implementaciones de planificadores alternativos [Evaluar]

### 2.17.7. SF/Proximidad (3 horas Core-Tier1)

**Temas:**
**Core Tier2**

- La velocidad de la luz y los computadores (un pie por nano segundo vs reloj de 1 GHz)
- Latencia en sistemas de computadores: memoria vs latencia de disco vs memoria a lo largo de la red.
- Memoria *cache* y el efecto de localidad espacial y temporal en el desempeño en el procesador y sistemas
- Memoria *cache* y coherencia de memoria *cache* en bases de datos, sistemas operativos, sistemas distribuídos y arquitectura de computadores.
- Introducción a la jerarquía de memorias de procesador y la fórmula para el promedio de acceso a memoria.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explicar la importancia de localidad en la determinación de desempeño [Familiarizarse]
2. Describir por qué las cosas que están cerca en espacio toman menos tiempo para su acceso [Familiarizarse]
3. Calcular el tiempo promedio de acceso a la memoria y describir los tradeoffs en el desempeño jerarquico de la memoria en términos de capacidad, tasa de fallos/éxitos y el tiempo de acceso [Evaluar]

### 2.17.8. SF/Virtualización y aislamiento (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Justificación de la protección y el rendimiento predecible.
- Los niveles de indirección, ilustrados por la memoria virtual para la gestión de los recursos de memoria física.
- Los métodos para la implementación de la memoria virtual y las máquinas virtuales.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explicar por qué es importante isolar y proteger la ejecución de programas individuales y ambientes que comparten recursos subyacentes comunes [Familiarizarse]

2. Describa cómo el concepto de indirección puede crear la ilusión de un equipo y recursos dedicados, incluso cuando físicamente esté compartida entre varios programas y entornos [Familiarizarse]

3. Medir el desempeño de dos instancias de aplicaciones que se ejecutan en máquinas virtuales independientes y determinar el efecto de aislamiento rendimiento [Evaluar]

### 2.17.9. SF/Confiabilidad a través de redundancia (2 horas Core-Tier1)

**Temas:**
**Core Tier2**

- Distinción entre bugs y fallas.
- Redundancia a través de chequear y reintentar
- Redundancia a través de código redundante (código de corrección de errores, CRC, FEC)
- Duplicación/espejos/réplicas.
- Otros abordajes para tolerancia a fallas y disponibilidad.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier2:**

1. Explique la diferencia entre los errores del programa, errores del sistema y fallos de hardware (por ejemplo, mala memoria) y excepciones (por ejemplo, intento de dividir por cero) [Familiarizarse]

2. Articular la distinción entre la detección, manejo y recuperación de fallas y los métodos para su aplicación. [Familiarizarse]

3. Describir el papel de los códigos de corrección de errores en la prestación de técnicas de control y corrección de errores en la memoria, almacenamiento y redes. [Familiarizarse]

4. Aplicar algoritmos simples para la explotación de la información redundante para los propósitos de corrección de datos. [Usar]

5. Comparar los diferentes métodos de detección de errores y de corrección para sus sobre gastos de datos, complejidad de implementación y el tiempo de ejecución para la codificación relativa, detección y corrección de errores [Evaluar]

### 2.17.10. SF/Evaluación cuantitativa

**Temas:**
**Electivo**

- Herramientas analíticas para guiar evaluación cuantitativa.
- Análisis de orden de magnitud (Notación Big O)
- Análisis del camino corto y rápido de un sistema.
- Eventos sobre su efecto en el rendimiento (por ejemplo, lugar de instrucción, fallos de caché, los errores de página)
- La comprensión de los sistemas de capas, cargas de trabajo y plataformas, sus implicaciones para el rendimiento, y los retos que representan para la evaluación
- Trampas de micro puntos de referencia.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Explicar las circunstancias en las que una figura de una métrica del rendimiento del sistema es útil [Familiarizarse]

2. Explicar las insuficiencias de los puntos de referencia como medida de rendimiento del sistema. [Familiarizarse]

3. Utilice estudios limitados o cálculos sencillos para producir estimaciones de orden de magnitud para una métrica de rendimiento en un contexto dado. [Usar]

4. Llevar a cabo un experimento de rendimiento en un sistema de capas para determinar el efecto de un parámetro del sistema en la figura del rendimiento del sistema [Evaluar]

## 2.18. Asuntos sociales y práctica profesional (SP)

Si bien las cuestiones técnicas son fundamentales para el plan de estudios de computación, no constituyen un programa educativo completo en el campo. Los estudiantes también deben estar expuestos al contexto social más amplio de la computación para desarrollar una comprensión de las cuestiones sociales, éticas, legales y profesionales pertinentes. Esta necesidad de incorporar el estudio de estos problemas no técnicos en el plan de estudios ACM fue reconocido formalmente en el año 1991, como puede verse en el siguiente extracto:

> Estudiantes universitarios también deben comprender los aspectos culturales, sociales, legales y éticos básicos inherentes a la disciplina de la computación. Ellos deben entender donde la disciplina ha estado, dónde está y hacia dónde se dirige. También deben entender su papel individual en este proceso, así como apreciar las cuestiones filosóficas, problemas técnicos y los valores estéticos que desempeñan un papel importante en el desarrollo de la disciplina.

Los estudiantes también tienen que desarrollar la capacidad de formular preguntas serias sobre el impacto social de la computación y de evaluar las respuestas propuestas a esas preguntas. Los futuros profesionales deben ser capaces de anticipar el impacto de la introducción de un producto dado en un entorno determinado. ¿Será ese producto mejorar o degradar la calidad de vida? ¿Cuál será el impacto sobre los individuos, los grupos y las instituciones?

Por último, los estudiantes deben ser conscientes de los derechos legales básicos de software y hardware y los usuarios así como y también tienen que apreciar los valores éticos que son la base de estos derechos. Los futuros profesionales deben entender la responsabilidad que van a llevar y las posibles consecuencias del fracaso. Ellos deben comprender sus propias limitaciones, así como las limitaciones de sus herramientas. Todos los profesionales deben hacer un compromiso a largo plazo para permanecer al día en sus especialidades elegidas y en la disciplina de la computación en su conjunto.

Como los avances tecnológicos siguen afectando de manera significativa la forma en que vivimos y el trabajo, la importancia crítica de asuntos sociales y práctica profesional continua aumentando; nuevos productos basados en computadores y lugares plantean problemas cada vez más difíciles cada año. Son nuestros estudiantes que deben entrar en el mundo laboral y el mundo académico con el objetivo de identificar y resolver estos problemas.

Los educadores de ciencia de la computación pueden optar por enseñar este núcleo y el material electivo en cursos independientes, integrados en cursos técnicos y teóricos tradicionales, o como unidades especiales en cursos de fina de carrera y cursos de práctica profesional. El material en esta área está mejor cubierto a través de una combinación de un curso obligartorio junto con módulos breves en otros cursos. Por un lado, algunas unidades que figuran como Core Tier-1 (en particular, el contexto social, herramientas de análisis, la ética profesional y la propiedad intelectual) no se prestan fácilmente a ser cubiertos en otros cursos tradicionales. Sin un curso independiente, es difícil cubrir estos temas adecuadamente. Por otro lado, si las consideraciones éticas y sociales son cubiertas solamente en el curso independiente y no en contexto, esto reforzará la falsa idea de que los procesos técnicos no necesitan de estas otras cuestiones pertinentes.

Debido a esta amplia relevancia, es importante que varios cursos tradicionales incluyan módulos con estudios de casos que analizan los aspectos éticos, legales, sociales y profesionales en el contexto de la materia técnica del curso. Los cursos en áreas como la ingeniería de software, bases de datos, redes de computadores, y seguridad aseguramiento de la información y la introducción a la computación proporcionan un contexto obvio para el análisis de cuestiones éticas. Sin embargo, un módulo relacionado con la ética podría desarrollarse para casi cualquier curso en el currículo. Sería explícitamente en contra del espíritu de la recomendaciones tener sólo un curso independiente. A lo largo de todos los asuntos en esta área existe la necesidad de hablar con el profesional para que siempre enfrente estos problemas desde la perspectiva moral y técnica. Las cuestiones éticas discutidas en cualquier clase deben surgen naturalmente de la materia objeto de esa clase. Los ejemplos incluyen una discusión en el curso de base de datos de la agregación de datos o minería de datos, o una discusión en el curso de ingeniería de software de los posibles conflictos entre las obligaciones para con el cliente y las obligaciones para el usuario y otras personas afectadas por su trabajo. Asignaciones de programación construidas alrededor de aplicaciones tales controlar el movimiento de un láser durante la cirugía de

los ojos puede ayudar a hacer frente a los impactos profesionales, éticos y sociales de la computación. El cuerpo docente que no esté familiarizados con el contenido y/o la pedagogía de la ética aplicada debe aprovechar los recursos considerables de ACM, IEEE-CS, SIGCAS (grupo de interés especial en las computadoras y la sociedad), y otras organizaciones.

Cabe señalar que la aplicación de un análisis ético subyace en cada sección de esta área de conocimiento. El Código ACM de Ética y Conducta Profesional (http://www.acm.org/about/code-of-ethics) proporciona directrices que sirven como base para la realización de nuestro trabajo profesional. Los imperativos morales generales proporcionan un mejor entendimiento de nuestro compromiso con la responsabilidad personal, la conducta profesional y nuestros roles de liderazgo.

| **área de Conocimiento (*Knowledge Area*-KA) (KA)** | **Core Tier1** | **Core Tier2** | **Electivo** |
|---|---|---|---|
| 2.18.1 Contexto Social (Pág. 118) | 2 | | No |
| 2.18.2 Herramientas de Análisis (Pág. 119) | | | No |
| 2.18.3 Ética Profesional (Pág. 119) | 2 | | No |
| 2.18.4 Propiedad Intelectual (Pág. 121) | | | No |
| 2.18.5 Privacidad y Libertades Civiles (Pág. 121) | | | No |
| 2.18.6 Comunicación profesional (Pág. 122) | | | No |
| 2.18.7 Sostenibilidad (Pág. 123) | 1 | | No |
| 2.18.8 Historia (Pág. 124) | | | No |
| 2.18.9 Economía de la Computación (Pág. 124) | | | No |
| 2.18.10 Políticas de seguridad, Leyes y crímenes computacionales (Pág. 125) | | | No |

### 2.18.1. SP/Contexto Social (2 horas Core-Tier1)

Las computadoras y el Internet, quizás más que cualquier otra tecnología, han transformado la sociedad durante los últimos 75 años, con un aumento espectacular en la productividad humana; una explosión de opciones para las noticias, el entretenimiento y la comunicación; y los avances fundamentales en casi todas las ramas de la ciencia e ingeniería. Social Contexto proporciona la base para todas las demás unidades de conocimiento de esta área (SP), especialmente la ética profesional.

**Temas:**

**Core Tier1**

- Implicancias sociales de la computación en un mundo conectado en red.
- Impacto de los medios sociales en el individualismo, colectivismo y en la cultura.

**Core Tier2**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. 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]

2. 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 [Familiarizarse]

3. Interpretar el contexto social de un determinado diseño y su aplicación [Familiarizarse]

4. Evaluar la eficacia de un diseño y aplicación dada a partir de datos empíricos [Evaluar]

5. Resumir las implicaciones de los medios sociales en el individualismo frente al colectivismo y la cultura [Usar]

**Core-Tier2:**

6. 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]
7. 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) [Evaluar]
8. 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) [Familiarizarse]
9. Explicar las consecuencias de la sensibilidad al contexto en los sistemas de computación ubicua [Familiarizarse]

### 2.18.2. SP/Herramientas de Análisis

Teorías y principios éticos son los fundamentos del análisis ético, ya que son los puntos de vista desde el que se pueden obtener directrices a lo largo de la vía a una decisión. Cada teoría hace hincapié en diferentes puntos, como la predicción del resultado y siguiendo los propios deberes con los demás con el fin de llegar a una decisión éticamente guiada. Sin embargo, para que una teoría ética sea útil, la teoría debe ser dirigida hacia un conjunto común de objetivos. Los principios éticos son los objetivos comunes que cada teoría intenta alcanzar con el fin de tener éxito. Estos objetivos incluyen la beneficencia, menor daño, el respeto por la autonomía y la justicia.

**Temas:**

**Core Tier1**

- Argumentación ética.
- Teorías éticas y toma de decisiones.
- Suposiciones morales y valores.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Evaluar las posiciones de las partes interesadas en una situación dada [Evaluar]
2. Analizar errores lógicos básicos en una discusión [Evaluar]
3. Analizar un argumento para identificar premisas y la conclusión [Evaluar]
4. Ilustrar el uso de ejemplo y analogía en el argumento ético [Usar]
5. Evaluar compensaciones éticos / sociales en las decisiones técnicas [Evaluar]

### 2.18.3. SP/Ética Profesional (2 horas Core-Tier1)

La ética de la computadora es una rama de la filosofía práctica que se ocupa de cómo los profesionales en computación deben tomar decisiones relativas a la conducta profesional y social. Hay tres influencias principales: 1) propio código personal de un individuo; 2) cualquier código informal de comportamiento ético existente en el lugar de trabajo; y 3) la exposición a los códigos formales de ética. Ver *Information Assurance and Security* Aseguramiento y Seguridad de la Información (IAS)

**Temas:**

**Core Tier1**

- 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)

**Core Tier2**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Identificar los problemas éticos que se plantean en el desarrollo de software y determinar cómo abordarlos técnica y éticamente [Familiarizarse]
2. Explicar la responsabilidad ética de velar por la corrección de software, confiabilidad y seguridad [Familiarizarse]
3. Describir los mecanismos que normalmente existen para que profesional se mantenga al día [Familiarizarse]
4. Describir las fortalezas y debilidades de códigos profesionales relevantes como expresiones de profesionalismo y guías para la toma de decisiones [Familiarizarse]
5. Analizar un problema mundial de computación, observando el papel de los profesionales y funcionarios del gobierno en el manejo de este problema [Evaluar]
6. Evaluar los códigos de ética profesional de la ACM, la Sociedad de Computación de la IEEE, y otras organizaciones [Evaluar]

**Core-Tier2:**

7. Describir las formas en que los profesionales pueden contribuir a las políticas públicas [Familiarizarse]
8. Describir las consecuencias de la conducta profesional inadecuada [Familiarizarse]
9. Identificar las etapas progresivas en un incidente de denuncia de irregularidades [Familiarizarse]
10. Identificar ejemplos de cómo interactúa la cultura regional con dilemas éticos [Familiarizarse]
11. Investigar las formas de acoso, discriminación y formas de ayuda [Usar]
12. Examine las diversas formas de acreditación de profesionales [Usar]
13. Explicar la relación entre la ergonomía en los ambientes y la salud de las personas de computación [Familiarizarse]
14. Desarrollar un uso del computador/política de uso aceptable con medidas coercitivas [Evaluar]
15. 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 [Familiarizarse]

### 2.18.4. SP/Propiedad Intelectual

La propiedad intelectual se refiere a una serie de derechos intangibles de propiedad de un activo tal como un programa de software. Cada ”derecho” de propiedad intelectual es en sí mismo un activo. La ley establece diferentes métodos para la protección de estos derechos de propiedad en función de su tipo. Hay básicamente cuatro tipos de derechos de propiedad intelectual relacionados con el software: patentes, derechos de autor, secretos comerciales y marcas registradas. Cada una proporciona un tipo diferente de protección legal. Ver Gestión de la información (IM).

**Temas:**

**Core Tier1**

- 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.

**Electivo**

- Fundamentos del movimiento Open Source.
- Piratería de Software.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Discute las bases filosóficas de la propiedad intelectual [Familiarizarse]
2. Discute la racionalidad de la protección legal de la propiedad intelectual [Familiarizarse]
3. Decribe la legislación orientada a los delitos de derechos de autor digitales [Familiarizarse]
4. Critica la legislación orientada a los delitos digitales de derechos de autor [Evaluar]
5. Identifica ejemplos contemporáneos de propiedad intelectual digital intangible [Familiarizarse]
6. Justifica el uso de material con derechos de autor [Evaluar]
7. Evalúa los asuntos éticos inherentes a diversos mecanismos de detección de plagio [Evaluar]
8. Interpreta el intento y la implementación de licencias de software [Familiarizarse]
9. Discute asuntos que involucran la seguridad de patentes en software [Familiarizarse]
10. Caracteriza y contrasta los conceptos de derechos de autor, patentes y de marcas comerciales [Evaluar]

**Elective:**

11. Identifica los objetivos del movimiento de software libre [Familiarizarse]
12. Identifica la naturaleza global de la piratería de software [Familiarizarse]

### 2.18.5. SP/Privacidad y Libertades Civiles

El intercambio de información electrónica destaca la necesidad de equilibrar la protección de la privacidad con acceso a la información. La facilidad de acceso digital a muchos tipos de datos hace que los derechos de privacidad y las libertades civiles sean mas complejas, siendo difierentes entre la variedad de culturas en todo el mundo.

**Temas:**

**Core Tier1**

- 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.

**Electivo**

- Legislación de privacidad en áreas de práctica.
- Libertades civiles y diferencias culturales.
- Libertad de expresión y sus limitaciones.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Discute las bases filosóficas para la protección legal de la privacidad personal [Familiarizarse]
2. Evalúa soluciones para amenazas a la privacidad en bases de datos transaccionales y almacenes de datos [Evaluar]
3. 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]
4. Describe las ramificaciones de la privacidad diferenciada [Familiarizarse]
5. Investica el impacto de soluciones tecnológicas a los problemas de privacidad [Usar]

**Elective:**

6. Critica la intención, el valor potencial y la implementación de las diversas formas de legislación en privacidad [Evaluar]
7. Identifica estrategias que permitan la apropiada libertad de expresión [Familiarizarse]

### 2.18.6. SP/Comunicación profesional

La comunicación profesional transmite información técnica a los diversos públicos que pueden tener diferentes objetivos y necesidades de esa información. Comunicación profesional efectiva de información técnica es rara vez un don heredado, sino que necesita ser enseñado en su contexto a través del currículo de pregrado.

**Temas:**
**Core Tier1**

- Leer, comprender y resumir el material técnico, incluyendo el código fuente y la documentación.
- Escritura de documentación y material técnico eficaz.
- Dinámica de comunicación oral, escrita, electrónica de equipos y en grupos.
- Comunicarse profesionalmente con los accionistas y otros involucrados en una empresa.
- Utilizando herramientas colaborativas.

**Electivo**

- Tratar con ambientes interculturales.
- Puntos de equilibrio de riesgos en proyectos de software, en términos de la tecnología, la estructura/proceso, la calidad, la gente, el mercado y financiero.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Core-Tier1:**

1. Escribe documentos técnicos claros, concisos y exactos siguiendo estándares bien definidos para su formato y para la inclusión de tablas, figuras y referencias adecuadas [Usar]
2. Evalúa documentación técnica escrita para detectar problemas de diversos tipos [Evaluar]
3. Desarrolla y brinda una presentación formal de buena calidad [Evaluar]

4. Interacciones del plan (por ejemplo, cara a cara, documentos compartidos virtuales) con otros en los que son capaces de obtener su punto de vista, escuchar con atención y apreciar los puntos de los demás, incluso cuando no están de acuerdo y son capaces de transmitir a los demás que han oído [Usar]

5. Describir las fortalezas y debilidades de las diversas formas de comunicación (por ejemplo, cara a cara, documentos compartidos virtuales) [Familiarizarse]

6. Examinar las medidas adecuadas que se utilizan para comunicarse con las partes involucrados en un proyecto [Usar]

7. Comparar y contrastar diversas herramientas de colaboración [Evaluar]

**Elective:**

8. Discutir formas de influir en el rendimiento y los resultados en equipos multi-culturales [Familiarizarse]

9. Examine los puntos de equilibrio y las fuentes comunes de riesgo en proyectos relacionados con el software de tecnología, estructura/proceso, la calidad, la gente, el mercado y la parte financiera [Usar]

10. Evaluar las fortalezas y debilidades personales para trabajar de forma remota como parte de un equipo multinacional [Evaluar]

### 2.18.7. SP/Sostenibilidad (1 horas Core-Tier1)

La sostenibilidad, de acuerdo a las Naciones Unidas es "el desarrollo que satisface las necesidades del presente sin comprometer la capacidad de generaciones futuras para satisfacer sus propias necesidades." La sostenibilidad se introdujo por primera vez en las directrices curriculares CS2008. Los temas en esta área emergente se pueden integrar de forma natural en otras áreas y unidades que presenten familiaridad, como la interacción humano-computador y la evolución del software. Ver Interacción Humano-Computador (HCI), Ingeniería de Software (SE).

**Temas:**

**Core Tier1**

- Comenzando una partición sustentable para tener en consideración cultural y impacto ambiental de desicione sd eimplementación.
- Explorar la sociedad global y impacto ambiental del uso de computadoras y disposición (e-waste)

**Core Tier2**

- Impactos ambientales de las elecciones de diseño en areas específicas tal como los algoritmos de los sistemas operativos, internet, bases de datos o interacción humano computador.

**Electivo**

- Lineamientos para estandares de diseño sostenible.
- Efectos sistémicos de fenómenos complejos mediados por computador (e.g. teleconmutación o compras web)
- Computación penetrante; procesamiento integrado de información dentro de objetos cotidianos y actividades, tal como sistemas de inteligentes de energía, redes sociales y sistemas retroalimentados para promover comportamiento sostenible, transporte, monitoreo del medio ambiente, ciencias ciudadana y activismo.
- Investigación de aplicaciones de la computación a problemas ambientales, como la energía, contaminación, uso de recursos, reciclaje y reutilización, administración de alimentos, agricultura y otros.
- La interdependencia de la sostenibilidad de los sistemas de software con sistemas sociales, incluyendo el conocimiento y las habilidades de sus usuarios, procesos organizacionales y políticas y su contexto social (ejem. fuerzas del mercado, políticas gubernamentales).

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Core-Tier1:**

1. Identificar maneras para ser un practicante sostenible [Familiarizarse]
2. Ilustrar los impactos sociales y ambientales globales de uso y la eliminación de computadores (e-waste) [Usar]

**Core-Tier2:**

3. Describir los impactos ambientales de las opciones de diseño en el campo de la informática que se relacionan con el diseño de algoritmo, el diseño del sistema operativo, diseño de redes, diseño de bases de datos, etc [Familiarizarse]
4. Investigar los impactos sociales y ambientales de los nuevos diseños de sistemas a través de proyectos [Usar]

**Elective:**

5. Identificar los lineamientos para el diseño o despliegue de TI sostenible [Familiarizarse]
6. Enumerar los efectos sostenibles de teletrabajo o de compras web [Familiarizarse]
7. Investigar la computación ubicua en áreas como los sistemas inteligentes de energía, redes sociales, el transporte, la agricultura, los sistemas de la cadena de suministro, monitoreo ambiental y el activismo ciudadano [Usar]
8. Desarrollar aplicaciones de computación y evaluar a través de areas de investigación relacionadas a cuestiones ambientales (ejemplo energia, contaminación, uso de recursos, reciclaje y reutilización, gestión de alimentos y agricultura) [Evaluar]

### 2.18.8. SP/Historia

La historia de la computación nos provee de un camino de como los cambios rápidos en los impactos computacionales en la sociedad y en escala global. esto amenudo se sabe en los conceptos fundamentales al igual que los Fundamentos de Sistemas y los Fundamentos de desarrollo de software.

**Temas:**

**Electivo**

- Pre-historia – El mundo antes de 1946.
- Historia del hardware, software, redes.
- Pioneros de la Computación.
- Historia de Internet.

**Objetivos de Aprendizaje (*Learning Outcomes*):**

**Elective:**

1. Identificar importantes tendencias en la historia del campo de la computación [Familiarizarse]
2. Identificar las contribuciones de varios pioneros en el campo de la computación [Familiarizarse]
3. Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse]
4. Comparar la vida diaria antes y después de la llegada de los ordenadores personales y el Internet [Evaluar]

### 2.18.9. SP/Economía de la Computación

Las métricas y mejores prácticas para la gestión del personal y gestión financiera en sistemas de información

**Temas:**

**Electivo**

- 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 relación 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Resumir los fundamentos para los esfuerzos antimonopolio [Familiarizarse]
2. Identificar diversas maneras en que la industria de la tecnología de la información está afectada por la escasez de la oferta de trabajo [Familiarizarse]
3. Identificar la evolución de la estrategía de precios para el cálculo de los bienes y servicios [Familiarizarse]
4. Discutir los beneficios, los inconvenientes y las implicaciones de *off-shoring* y *outsourcing* [Familiarizarse]
5. Investigar y defender maneras de tratar las limitaciones en el acceso a la computación. [Usar]
6. Describir los beneficios económicos de efectos de la red [Familiarizarse]

### 2.18.10. SP/Políticas de seguridad, Leyes y crímenes computacionales

Si bien las políticas de seguridad, las leyes y los delitos informáticos son temas importantes, es esencial que se vean con la base de otras unidades de conocimiento relacionadas al área social y profesional, tales como la propiedad intelectual, de privacidad y las libertades civiles, Contexto Social y Ética Profesional. Las computadoras y la Internet, quizás más que cualquier otra tecnología, han transformado la sociedad en los últimos 75 años. Al mismo tiempo, han contribuído a amenazas sin precedentes a la vida privada; nuevas categorías de delitos y conductas antisociales; grandes trastornos a las organizaciones; y la concentración a gran escala de los riesgos en los sistemas de información. Ver las áreas Interacción Humano-Computador (HCI) y Aseguramiento y Seguridad de la Información (IAS).

**Temas:**
**Electivo**

- 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.

**Objetivos de Aprendizaje (*Learning Outcomes*):**
**Elective:**

1. Listar ejemplos clásicos de delitos informáticos y incidentes de ingeniería social con impacto social [Familiarizarse]
2. Indentificar leyes que se aplican a delitos informáticos [Familiarizarse]
3. Describir la motivación y ramificaciones de cyberterrorismo y hackeo criminal [Familiarizarse]

4. Examinar los problemas éticos y legales relacionados con el mal uso de accesos y diversas violaciones en la seguridad [Usar]
5. Discutir el rol del profesional en seguridad y los problemas que están envueltos [Familiarizarse]
6. 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 [Usar]
7. Escribir una politica de seguridad de una empresa, la cual incluye procedimientos para administrar contraseñas y monitorizar a los empleados [Usar]

# Capítulo 3

# Plan de estudios

## 3.1. Codificación de los cursos

Los cursos son clasificados en 4 niveles:

- **Introductorios** Código 1.. ejemplo: CS100
- **Intermedios** Código 2.. ejemplo: MA200
- **Avanzados** Código 3.. y ejemplo: IS370
- **Trabajo de final de carrera** Código 4.. ejemplo: CS403

Los cursos se encuentran codificados bajo el esquema que se muestra en la Figura 3.1.

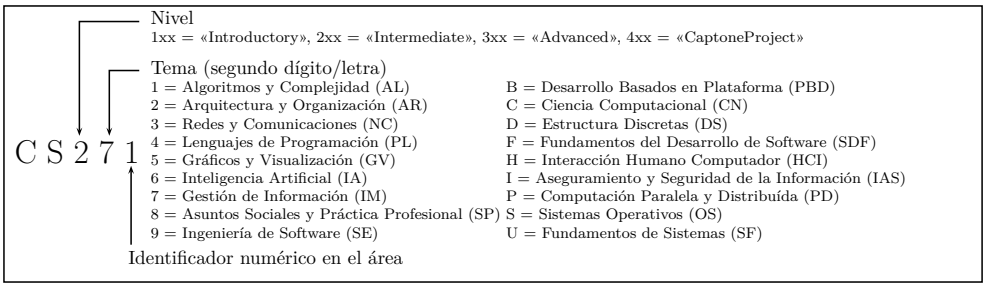



Figura 3.1: Esquema de codificación para los cursos.

El tipo de curso esta determinado por las letras iniciales. Los posibles códigos para estos tipos son:

1. **CS** Área de Ciencia de la Computación (*Computer Science* – CS);
2. **AI** Área de Inteligencia Artificial (*Artificial Intelligence* – AI);
3. **MA** Matemáticas (MA);
4. **ST** Estadística (ST);
5. **CB** Área de Ciencias Básicas (Física, Biología);
6. **FI** Área de Física;
7. **CH** Ciencias Químicas (CH);
8. **BI** Área de Ciencias Básicas (Física, Biología)
9. **FG** Área de Formación General;
10. **EX** Actividades extracurriculares
11. **ID** Área de Idiomas.

Al mismo tiempo, los cursos están clasificados en:

- Estudios generales
- Específicos
- Especialización

## 3.2. Estructura Curricular

La relación de cursos se muestra a continuación:

**Primer Semestre**

| Código | Curso | Área | HT | HP | CR | TIPO | Requisitos |
|---|---|---|---|---|---|---|---|
| line:30 CS111 | line:30 Introducción a la Programación (Pág line:30 **??**) | Específicos | 2 | 4 | 4 | O | |
| line:30 BMA101 | line:30 Algebra Linear (Pág line:30 **??**) | Estudios generales | 3 | 2 | 4 | O | |
| line:30 BMA102 | line:30 Cálculo Diferencial (Pág line:30 **??**) | Estudios generales | 4 | 2 | 5 | O | |
| line:30 BCH101 | line:30 Química I (Pág line:30 **??**) | Estudios generales | 4 | 2 | 5 | O | |
| line:30 FG001 | line:30 Electivo Formación General (Pág line:30 **??**) | Estudios generales | 2 | 2 | 3 | O | |
| line:30 BEI101 | line:30 Inglés I (Pág line:30 **??**) | Estudios generales | | 4 | 2 | O | |
| | | | | | 23 | | |

**Segundo Semestre**

| Código | Curso | Área | HT | HP | CR | TIPO | Requisitos |
|---|---|---|---|---|---|---|---|
| line:65 CS100 | line:65 Introducción a la Ciencia de la Computación (Pág line:65 ??) | Específicos | 2 | 2 | 3 | O | |
| line:65 CS112 | line:65 Programación Orientada a Objetos I (Pág line:65 ??) | Específicos | 2 | 4 | 4 | O | line:65 CS111 ($1^{er}$ Sem) |
| line:65 CS1D1 | line:65 Estructuras Discretas (Pág line:65 ??) | Específicos | 2 | 4 | 4 | O | |
| line:65 BMA103 | line:65 Cálculo Integral (Pág line:65 ??) | Estudios generales | 4 | 2 | 5 | O | line:65 BMA102 ($1^{er}$ Sem) |
| line:65 BFI101 | line:65 Física I (Pág line:65 ??) | Estudios generales | 4 | 4 | 5 | O | |
| line:65 BEI102 | line:65 Inglés II (Pág line:65 ??) | Estudios generales | | 4 | 2 | O | line:65 BEI101 ($1^{er}$ Sem) |
| | | | | | 23 | | |

**Tercer Semestre**

| Código | Curso | Área | HT | HP | CR | TIPO | Requisitos |
|---|---|---|---|---|---|---|---|
| line:103 CS113 | line:103 Programación Orientada a Objetos II (Pág line:103 **??**) | Específicos | 2 | 4 | 4 | O | line:103 CS112 ($2^{do}$ Sem) |
| line:103 CS2B1 | line:103 Desarrollo Basado en Plataformas (Pág line:103 **??**) | Específicos | 1 | 4 | 3 | O | line:103 CS112 ($2^{do}$ Sem) |
| line:103 AI161 | line:103 IA Aplicada (Pág line:103 **??**) | Específicos | 2 | 4 | 4 | O | line:103 CS111 ($1^{er}$ Sem) |
| line:103 BMA104 | line:103 Cálculo Diferencial e Integral Avanzado (Pág line:103 **??**) | Estudios generales | 4 | 2 | 5 | O | line:103 BMA102 ($1^{er}$ Sem) |
| line:103 ST251 | line:103 Estadística y Probabilidades (Pág line:103 **??**) | Estudios generales | 2 | 2 | 3 | O | line:103 BMA102 ($1^{er}$ Sem) |
| line:103 BEI201 | line:103 Inglés III (Pág line:103 **??**) | Estudios generales | | 4 | 2 | O | line:103 BEI102 ($2^{do}$ Sem) |
| | | | | | 21 | | |

| Cuarto Semestre | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Código** | **Curso** | **Área** | **HT** | **HP** | **CR** | **TIPO** | **Requisitos** |
| line:147 CS210 | line:147 Algoritmos y Estructuras de Datos (Pág line:147 ??) | Específicos | 2 | 4 | 4 | O | line:147 CS113 ($3^{er}$ Sem) |
| line:147 CS211 | line:147 Teoría de la Computación (Pág line:147 ??) | Específicos | 2 | 4 | 4 | O | line:147 CS1D1 ($2^{do}$ Sem) |
| line:147 CS221 | line:147 Arquitectura de Computadores (Pág line:147 ??) | Específicos | 2 | 2 | 3 | O | line:147 CS1D1 ($2^{do}$ Sem) |
| line:147 CS271 | line:147 Bases de Datos I (Pág line:147 ??) | Específicos | 2 | 4 | 4 | O | line:147 CS112 ($2^{do}$ Sem), line:147 CS1D1 ($2^{do}$ Sem) |
| line:147 CS401 | line:147 Metodología de la Investigación (Pág line:147 ??) | Especialización | 1 | 2 | 2 | O | line:147 CS100 ($2^{do}$ Sem) |
| line:147 MA202 | line:147 Métodos Numéricos (Pág line:147 ??) | Estudios generales | 2 | 2 | 3 | O | line:147 BMA103 ($2^{do}$ Sem) |
| Elective line:147 BEI202 | line:147 Inglés IV (Pág line:147 ??) | Estudios generales | | 4 | 2 | E | line:147 BEI201 ($3^{er}$ Sem) |
| | | | | | 22 | | |

| Quinto Semestre | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Código** | **Curso** | **Área** | **HT** | **HP** | **CR** | **TIPO** | **Requisitos** |
| line:187 CS212 | line:187 Análisis y Diseño de Algoritmos (Pág line:187 ??) | Específicos | 2 | 4 | 4 | O | line:187 CS210 ($4^{to}$ Sem), line:187 CS211 ($4^{to}$ Sem) |
| line:187 CS261 | line:187 Inteligencia Artificial (Pág line:187 ??) | Específicos | 2 | 4 | 4 | O | line:187 ST251 ($3^{er}$ Sem) |
| line:187 CS272 | line:187 Bases de Datos II (Pág line:187 ??) | Específicos | 1 | 4 | 3 | O | line:187 CS271 ($4^{to}$ Sem) |
| line:187 CS291 | line:187 Ingeniería de Software I (Pág line:187 ??) | Específicos | 2 | 4 | 4 | O | line:187 CS113 ($3^{er}$ Sem), line:187 CS271 ($4^{to}$ Sem) |
| line:187 CS2S1 | line:187 Sistemas Operativos (Pág line:187 ??) | Específicos | 2 | 4 | 4 | O | line:187 CS221 ($4^{to}$ Sem) |
| Elective | line:187 BEI203 line:187 Inglés V (Pág line:187 ??) | Estudios generales | | 4 | 2 | E | line:187 BEI202 ($4^{to}$ Sem) |
| | | | | | 21 | | |

**Sexto Semestre**

| Código | Curso | Área | HT | HP | CR | TIPO | Requisitos |
|---|---|---|---|---|---|---|---|
| line:226 CS231 | line:226 Redes y Comunicación (Pág line:226 ??) | Específicos | 1 | 4 | 3 | O | line:226 CS2S1 ($5^{to}$ Sem) |
| line:226 CS311 | line:226 Programación Competitiva (Pág line:226 ??) | Específicos | 2 | 2 | 3 | O | line:226 CS212 ($5^{to}$ Sem) |
| line:226 CS312 | line:226 Estructuras de Datos Avanzadas (Pág line:226 ??) | Específicos | 2 | 4 | 4 | O | line:226 CS212 ($5^{to}$ Sem), line:226 BEI202 ($4^{to}$ Sem) |
| line:226 CS342 | line:226 Compiladores (Pág line:226 ??) | Específicos | 2 | 4 | 4 | O | line:226 CS211 ($4^{to}$ Sem) |
| line:226 AI263 | line:226 Introducción al Aprendizaje de Máquina (Pág line:226 ??) | Específicos | 2 | 4 | 4 | O | line:226 CS261 ($5^{to}$ Sem) |
| line:226 FI201 | line:226 Física Computacional (Pág line:226 ??) | Estudios generales | 2 | 2 | 3 | O | line:226 BFI101 ($2^{do}$ Sem) |
| | | | | | 21 | | |

**Séptimo Semestre**

| Código | Curso | Área | HT | HP | CR | TIPO | Requisitos |
|---|---|---|---|---|---|---|---|
| line:262 CS251 | line:262 Computación Gráfica (Pág line:262 ??) | Especialización | 2 | 4 | 4 | O | line:262 MA202 ($4^{to}$ Sem) |
| line:262 CS292 | line:262 Ingeniería de Software II (Pág line:262 ??) | Específicos | 2 | 4 | 4 | O | line:262 CS291 ($5^{to}$ Sem) |
| line:262 CS2H1 | line:262 Experiencia de Usuario (UX) (Pág line:262 ??) | Específicos | 2 | 4 | 4 | O | line:262 CS291 ($5^{to}$ Sem) |
| line:262 AI264 | line:262 Aprendizaje Profundo (Pág line:262 ??) | Específicos | 2 | 4 | 4 | O | line:262 AI263 ($6^{to}$ Sem) |
| line:262 BBI101 | line:262 Biología (Pág line:262 ??) | Estudios generales | 2 | 4 | 4 | O | |
| line:262 FG211 | line:262 Ética Profesional (Pág line:262 ??) | Específicos | 1 | 2 | 2 | O | |
| | | | | | 22 | | |

| Octavo Semestre | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Código** | **Curso** | **Área** | **HT** | **HP** | **CR** | **TIPO** | **Requisitos** |
| line:313 CS281 | line:313 Computación en la Sociedad (Pág line:313 ??) | Específicos | 2 | | 2 | O | |
| line:313 CS3I1 | line:313 Seguridad en Computación (Pág line:313 ??) | Específicos | 1 | 4 | 3 | O | line:313 CS231 ($6^{to}$ Sem) |
| line:313 CS3P1 | line:313 Computación Paralela y Distribuída (Pág line:313 ??) | Específicos | 2 | 4 | 4 | O | line:313 CS212 ($5^{to}$ Sem), line:313 CS231 ($6^{to}$ Sem) |
| line:313 CS402 | line:313 Proyecto de tesis 1 (Pág line:313 ??) | Especialización | 1 | 4 | 3 | O | line:313 CS401 ($4^{to}$ Sem) |
| line:313 FG106 | line:313 Teatro (Pág line:313 ??) | Estudios generales | 1 | 2 | 2 | O | |
| line:313 EX301 | line:313 Actividades Extracurriculares (Pág line:313 ??) | Estudios generales | 1 | 2 | 2 | O | |
| Elective | | | | | | | |
| line:313 AI268 | line:313 Visión Computacional (Pág line:313 ??) | Especialización | 2 | 2 | 3 | E | line:313 AI264 ($7^{mo}$ Sem) |
| line:313 CS391 | line:313 Ingeniería de Software III (Pág line:313 ??) | Específicos | 2 | 2 | 3 | E | line:313 CS292 ($7^{mo}$ Sem) |
| line:313 CS393 | line:313 Sistemas de Infomación | Específicos | 2 | 2 | 3 | E | line:313 CS291 ($5^{to}$ Sem) |

| Noveno Semestre | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Código** | **Curso** | **Área** | **HT** | **HP** | **CR** | **TIPO** | **Requisitos** |
| line:370 CS370 | line:370 Big Data (Pág line:370 ??) | Especialización | 1 | 4 | 3 | O | line:370 CS3P1 ($8^{vo}$ Sem) |
| line:370 CS400 | line:370 Prácticas Pre-profesionales (Pág line:370 ??) | Especialización | 1 | 2 | 2 | O | |
| line:370 CS403 | line:370 Proyecto de tesis 2 (Pág line:370 ??) | Especialización | 1 | 4 | 3 | O | line:370 CS402 ($8^{vo}$ Sem) |
| line:370 AI365 | line:370 Modelos Generativos Avanzados en IA (Pág line:370 ??) | Específicos | 2 | 4 | 4 | O | line:370 AI264 ($7^{mo}$ Sem) |
| line:370 CB309 | line:370 Bioinformática (Pág line:370 ??) | Especialización | 2 | 4 | 4 | O | line:370 BBI101 ($7^{mo}$ Sem) |
| Elective | | | | | | | |
| line:370 CS351 | line:370 Tópicos en Computación Gráfica (Pág line:370 ??) | Especialización | 2 | 2 | 3 | E | line:370 CS251 ($7^{mo}$ Sem) |
| line:370 CS353 | line:370 Computación Cuántica (Pág line:370 ??) | Especialización | 2 | 2 | 3 | E | line:370 CS3I1 ($8^{vo}$ Sem) |
| line:370 AI369 | line:370 Robótica (Pág line:370 ??) | Especialización | 2 | 2 | 3 | E | line:370 AI268 ($8^{vo}$ Sem) |
| line:370 CS392 | line:370 Tópicos en Ingeniería de Software (Pág line:370 ??) | Especialización | 2 | 2 | 3 | E | line:370 CS391 ($8^{vo}$ Sem) |

**Décimo Semestre**

| Código | Curso | Área | HT | HP | CR | TIPO | Requisitos |
|---|---|---|---|---|---|---|---|
| line:403 CS3P2 | line:403 Cloud Computing (Pág line:403 ??) | Específicos | 1 | 4 | 3 | O | line:403 CS370 ($9^{no}$ Sem) |
| line:403 CS404 | line:403 Taller de Investigación (Pág line:403 ??) | Especialización | 1 | 4 | 3 | O | line:403 CS403 ($9^{no}$ Sem) |
| line:403 AI367 | line:403 Tópicos en Inteligencia Artificial (Pág line:403 ??) | Especialización | 2 | 4 | 4 | O | line:403 AI264 ($7^{mo}$ Sem) |
| line:403 AI368 | line:403 Computación Evolutiva (Pág line:403 ??) | Específicos | 2 | 4 | 4 | O | line:403 AI264 ($7^{mo}$ Sem) |
| line:403 FG350 | line:403 Liderazgo y Desempeño (Pág line:403 ??) | Estudios generales | 2 | | 2 | O | line:403 FG106 ($8^{vo}$ Sem) |
| | | | | | 16 | | |

**Total de créditos de la carrera:** 207 .

## 3.3. Tópicos distribuídos por curso

| | | Primer Sem | | | | | | . |
|---|---|---|---|---|---|---|---|---|
| | **Unidades de Conocimiento** | CS111 line:83 | BMA101 line:83 | BMA102 line:83 | BCH101 line:83 | FG001 line:83 | BEI101 line:83 | Total |
| | AL/Análisis Básico | line:83 2 | | | | | | 2 |
| ★ | AL/Algoritmos y Estructuras de Datos fundamentales (3 horas Core-Tier1) | line:83 8 | | | | | | 8 |
| ★ | PL/Programación orientada a objetos (6 horas Core-Tier1) | line:83 4 | | | | | | 4 |
| ★ | PL/Sistemas de tipos básicos (4 horas Core-Tier1) | line:83 2 | | | | | | 2 |
| | SDF/Algoritmos y Diseño | line:83 9 | | | | | | 9 |
| | SDF/Conceptos Fundamentales de Programación | line:83 9 | | | | | | 9 |
| | SDF/Métodos de Desarrollo | line:83 1 | | | | | | 1 |
| | SP/Historia | line:83 5 | | | | | | 5 |
| | Total | 40 | 0 | 0 | 0 | 0 | 0 | |

Cuadro 3.1: Tópicos por curso del $1^{er}$ al $1^{er}$ Semestre

| Segundo Sem | | CS100 | CS112 | CS1D1 | BMA103 | BFI101 | BEI102 | Total |
|---|---|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | | | | | | | | |
| AL/Análisis Básico | | line:226 4 | | | | | | 4 |
| ★ AL/Algoritmos y Estructuras de Datos fundamentales (3 horas Core-Tier1) | | line:226 8 | | | | | | 8 |
| ★ AR/Lógica digital y sistemas digitales (3 horas Core-Tier1) | | line:226 4 | | | | | | 4 |
| ★ AR/Organización y Arquitectura del Sistema de Memoria (3 horas Core-Tier1) | | line:226 4 | | | | | | 4 |
| DS/Funciones, relaciones y conjuntos | | | | line:226 22 | | | | 22 |
| DS/Lógica básica | | | | line:226 14 | | | | 14 |
| ★ DS/Técnicas de demostración (1 horas Core-Tier1) | | | | line:226 14 | | | | 14 |
| DS/Fundamentos de conteo | | | | line:226 10 | | | | 10 |
| ★ DS/Árboles y Grafos (1 horas Core-Tier1) | | | | line:226 10 | | | | 10 |
| ★ DS/Probabilidad Discreta (2 horas Core-Tier1) | | | | line:226 10 | | | | 10 |
| ★ GV/Conceptos Fundamentales (1 horas Core-Tier1) | | line:226 2 | | | | | | 2 |
| GV/Rendering Básico | | line:226 2 | | | | | | 2 |
| ★ IAS/Criptografía (1 horas Core-Tier1) | | line:226 2 | | | | | | 2 |
| ★ IM/Sistemas de Bases de Datos (3 horas Core-Tier1) | | line:226 4 | | | | | | 4 |
| ★ AI/Cuestiones fundamentales (1 horas Core-Tier1) | | line:226 2 | | | | | | 2 |
| ★ AI/Estrategias de búsquedas básicas (4 horas Core-Tier1) | | line:226 1 | | | | | | 1 |
| ★ AI/Aprendizaje Automático Básico (2 horas Core-Tier1) | | line:226 1 | | | | | | 1 |
| NC/Introducción a redes | | line:226 4 | | | | | | 4 |
| ★ NC/Entrega confiable de datos (2 horas Core-Tier1) | | line:226 4 | | | | | | 4 |
| OS/Visión general de Sistemas Operativos | | line:226 4 | | | | | | 4 |

| Segundo Sem | . | Total |
|---|---|---|
| **Unidades de Conocimiento** | | |
| Total | | |

Cuadro 3.3: Tópicos por curso del $2^{do}$ al $2^{do}$ Semestre

| Tercer Sem | . | Total |
|---|---|---|
| **Unidades de Conocimiento** | | |
| PBD/Introducción | | |
| PBD/Plataformas web | | |
| PBD/Plataformas móviles | | |
| Total | | |

Cuadro 3.4: Tópicos por curso del $3^{er}$ al $3^{er}$ Semestre

| Cuarto Sem | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | CS210 | CS211 | CS221 | CS271 | CS401 | MA202 | BEI202 | Total | |
| ★ AR/Lógica digital y sistemas digitales (3 horas Core-Tier1) | | | line:117 18 | | | | | 18 | |
| ★ AR/Organización de la Máquina a Nivel Ensamblador (6 horas Core-Tier1) | | | line:117 8 | | | | | 8 | |
| ★ AR/Organización y Arquitectura del Sistema de Memoria (3 horas Core-Tier1) | | | line:117 8 | | | | | 8 | |
| ★ AR/Interfaz y comunicación (1 horas Core-Tier1) | | | line:117 8 | | | | | 8 | |
| AR/Organización funcional | | | line:117 8 | | | | | 8 | |
| AR/Multiprocesamiento y arquitecturas alternativas | | | line:117 8 | | | | | 8 | |
| AR/Mejoras de rendimiento | | | line:117 8 | | | | | 8 | |
| ★ IM/Sistemas de Bases de Datos (3 horas Core-Tier1) | | | | line:117 14 | | | | 14 | |
| ★ IM/Modelado de datos (4 horas Core-Tier1) | | | | line:117 14 | | | | 14 | |
| IM/Indexación | | | | line:117 4 | | | | 4 | |
| IM/Bases de Datos Relacionales | | | | line:117 14 | | | | 14 | |
| IM/Lenguajes de Consulta | | | | line:117 12 | | | | 12 | |
| Total | 0 | 0 | 66 | 58 | 0 | 0 | 0 | | |

Cuadro 3.5: Tópicos por curso del $4^{to}$ al $4^{to}$ Semestre

| Quinto Sem | | CS212 | CS261 | CS272 | CS291 | CS251 | BEI203 | Total |
|---|---|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | | line:226 | line:226 | line:226 | line:226 | line:226 | line:226 | . |
| AL/Análisis Básico | | 10 | | | | | | 10 |
| AL/Estrategias Algorítmicas (1 horas Core-Tier1) | ★ | 30 | | | | | | 30 |
| AL/Algoritmos y Estructuras de Datos fundamentales (3 horas Core-Tier1) | ★ | 10 | | | | | | 10 |
| AL/Computabilidad y complejidad básica de autómatas (3 horas Core-Tier1) | ★ | 2 | | | | | | 2 |
| AL/Estructuras de Datos Avanzadas y Análisis de Algoritmos | | 8 | | | | | | 8 |
| IAS/Fundamentos y Conceptos en Seguridad (2 horas Core-Tier1) | ★ | | | | 6 | | | 6 |
| IAS/Principios de Diseño Seguro (2 horas Core-Tier1) | ★ | | | | 4 | | | 4 |
| IM/Procesamiento de Transacciones | ★ | | | 12 | | | | 12 |
| IM/Bases de Datos Distribuidas | | | | 36 | | | | 36 |
| IM/Diseño Físico de Bases de Datos | | | | 10 | | | | 10 |
| IM/Almacenamiento y Recuperación de Información | | | | 10 | | | | 10 |
| AI/Cuestiones fundamentales (1 horas Core-Tier1) | ★ | | 2 | | | | | 2 |
| AI/Estrategias de búsquedas básicas (4 horas Core-Tier1) | ★ | | 2 | | | | | 2 |
| AI/Aprendizaje Automático Básico (2 horas Core-Tier1) | ★ | | 4 | | | | | 4 |
| AI/Búsqueda Avanzada | | | 18 | | | | | 18 |
| AI/Razonamiento Bajo Incertidumbre | | | 18 | | | | | 18 |
| AI/Agentes | | | 2 | | | | | 2 |
| AI/Procesamiento del Lenguaje Natural | | | 12 | | | | | 12 |
| AI/Aprendizaje de máquina avanzado | | | 20 | | | | | 20 |

| Quinto Sem | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | CS212 | CS261 | CS272 | CS291 | CS251 | BEI203 | Total | . |
| | line:361 | line:361 | line:361 | line:361 | line:361 | line:361 | | |
| OS/Máquinas virtuales | | | | | 6 | | 6 | |
| OS/Manejo de dispositivos | | | | | 6 | | 6 | |
| OS/Sistema de archivos | | | | | 6 | | 6 | |
| OS/Sistemas empotrados y de tiempo real | | | | | 6 | | 6 | |
| OS/Tolerancia a fallas | | | | | 3 | | 3 | |
| OS/Evaluación del desempeño de sistemas | | | | | 3 | | 3 | |
| SDF/Algoritmos y Diseño | | | | 6 | | | 6 | |
| SDF/Métodos de Desarrollo | | | | 4 | | | 4 | |
| ★ SE/Procesos de Software (1 horas Core-Tier1) | | | | 4 | | | 4 | |
| ★ SE/Gestión de Proyectos de Software (2 horas Core-Tier1) | | | | 5 | | | 5 | |
| ★ SE/Ingeniería de Requisitos (3 horas Core-Tier1) | | | | 6 | | | 6 | |
| ★ SE/Diseño de Software (5 horas Core-Tier1) | | | | 16 | | | 16 | |
| ★ SE/Verificación y Validación de Software (3 horas Core-Tier1) | | | | 10 | | | 10 | |
| ★ SE/Evolución de Software (2 horas Core-Tier1) | | | | 3 | | | 3 | |
| Total | 60 | 90 | 68 | 64 | 66 | 0 | | |

Cuadro 3.7: Tópicos por curso del $5^{to}$ al $5^{to}$ Semestre

| Sexto Sem | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | CS231 | CS311 | CS312 | CS342 | AI263 | FI201 | Total |
| NC/Introducción a redes | line:123 5 | | | | | | 5 |
| NC/Aplicaciones en red | line:123 5 | | | | | | 5 |
| ★ NC/Entrega confiable de datos (2 horas Core-Tier1) | line:123 10 | | | | | | 10 |
| ★ NC/Ruteo y reenvío (1.5 horas Core-Tier1) | line:123 12 | | | | | | 12 |
| ★ NC/Redes de área local (1.5 horas Core-Tier1) | line:123 10 | | | | | | 10 |
| ★ NC/Asignación de recursos (1 horas Core-Tier1) | line:123 12 | | | | | | 12 |
| ★ NC/Celulares (1 horas Core-Tier1) | line:123 5 | | | | | | 5 |
| NC/Redes sociales | line:123 5 | | | | | | 5 |
| ★ PL/Representación de programas (1 horas Core-Tier1) | | | line:123 5 | | | | 5 |
| ★ PL/Traducción y ejecución de lenguajes (3 horas Core-Tier1) | | | line:123 10 | | | | 10 |
| PL/Análisis de sintaxis | | | line:123 10 | | | | 10 |
| PL/Análisis semántico de compiladores | | | line:123 15 | | | | 15 |
| PL/Generación de código | | | line:123 20 | | | | 20 |
| Total | 64 | 0 | 0 | 60 | 0 | 0 | |

Cuadro 3.8: Tópicos por curso del $6^{to}$ al $6^{to}$ Semestre

| | | Séptimo Sem | | | | | | . |
|---|---|---|---|---|---|---|---|---|
| | **Unidades de Conocimiento** | CS251 line:155 | CS292 line:155 | CS2H1 line:155 | AI264 line:155 | BBI101 line:155 | FG211 line:155 | Total |
| ★ | GV/Conceptos Fundamentales (1 horas Core-Tier1) | line:155 6 | | | | | | 6 |
| | GV/Rendering Básico | line:155 12 | | | | | | 12 |
| | GV/Modelado Geométrico | line:155 15 | | | | | | 15 |
| | GV/Renderizado Avanzado | line:155 6 | | | | | | 6 |
| | GV/Animación por computadora | line:155 4 | | | | | | 4 |
| | HCI/Fundamentos | | | line:155 8 | | | | 8 |
| ★ | HCI/Diseño de Interacción (4 horas Core-Tier1) | | | line:155 8 | | | | 8 |
| | HCI/Programación de Sistemas Interactivos | line:155 2 | | | | | | 2 |
| | HCI/Diseño y Testing centrados en el usuario | | | line:155 16 | | | | 16 |
| | HCI/Nuevas Tecnologías Interactivas | | | line:155 8 | | | | 8 |
| | HCI/Colaboración y Comunicación | | | line:155 8 | | | | 8 |
| ★ | IAS/Principios de Diseño Seguro (2 horas Core-Tier1) | | line:155 8 | | | | | 8 |
| | NC/Introducción a redes | | line:155 12 | | | | | 12 |
| | OS/Principios de Sistemas Operativos | | line:155 8 | | | | | 8 |
| | PBD/Introducción | | line:155 10 | | | | | 10 |
| | SDF/Conceptos Fundamentales de Programación | | line:155 10 | | | | | 10 |
| | SDF/Métodos de Desarrollo | | line:155 16 | | | | | 16 |
| | Total | 45 | 64 | 48 | 0 | 0 | 0 | |

Cuadro 3.9: Tópicos por curso del $7^{mo}$ al $7^{mo}$ Semestre

Octavo Sem

| | Unidades de Conocimiento | CS281 | CS311 | CS3P1 | CS402 | FG106 | EX301 | AI268 | CS391 | CS393 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | HCI/Programación de Sistemas Interactivos | | | | | | | | line:232 12 | | 12 |
| | HCI/Diseño y Testing centrados en el usuario | | | | | | | | line:232 12 | | 12 |
| ★ | IAS/Fundamentos y Conceptos en Seguridad (2 horas Core-Tier1) | | line:232 5 | | | | | | line:232 8 | | 13 |
| ★ | IAS/Principios de Diseño Seguro (2 horas Core-Tier1) | | line:232 5 | | | | | | | | 5 |
| ★ | IAS/Programación Defensiva (2 horas Core-Tier1) | | line:232 5 | | | | | | | | 5 |
| ★ | IAS/Ataques y Amenazas (1 horas Core-Tier1) | | line:232 5 | | | | | | | | 5 |
| ★ | IAS/Seguridad de Red (2 horas Core-Tier1) | | line:232 4 | | | | | | | | 4 |
| ★ | IAS/Criptografía (1 horas Core-Tier1) | | line:232 5 | | | | | | | | 5 |
| | IAS/Seguridad en la Web | | line:232 5 | | | | | | | | 5 |
| | IAS/Seguridad de plataformas | | line:232 5 | | | | | | | | 5 |
| | IAS/Investigación digital (Digital Forensics) | | line:232 5 | | | | | | | | 5 |
| | IAS/Seguridad en Ingeniería de Software | | line:232 4 | | | | | | | | 4 |
| ★ | AI/Cuestiones fundamentales (1 horas Core-Tier1) | | | | | | | | line:232 18 | | 18 |
| | PD/Fundamentos de paralelismo | | | line:232 18 | | | | | | | 18 |
| ★ | PD/Descomposición en paralelo (2 horas Core-Tier1) | | | line:232 18 | | | | | | | 18 |
| ★ | PD/Comunicación y coordinación (3 horas Core-Tier1) | | | line:232 18 | | | | | | | 18 |
| ★ | PD/Análisis y programación de algorit- | | | line:232 18 | | | | | | | 18 |

| Octavo Sem | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | CS281 | CS311 | CS3P1 | CS402 | FG106 | EX301 | AI268 | CS391 | CS393 | . | Total |
| SP/Economía de la Computación | line:277 2 | | | | | | | | | | 2 |
| SP/Políticas de seguridad, Leyes y crímenes computacionales | line:277 2 | | | | | | | | | | 2 |
| Total | 24 | 48 | 102 | 0 | 0 | 0 | 0 | 64 | 0 | | |

Cuadro 3.11: Tópicos por curso del $8^{vo}$ al $8^{vo}$ Semestre

| Noveno Sem | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | CS370 | CS400 | CS403 | AI365 | CB309 | CS351 | CS353 | AI369 | CS392 | CS3P3 | Total | . |
| PD/Fundamentos de paralelismo | | | | | | | | | | line:91 18 | 18 | |
| ★ PD/Descomposición en paralelo (2 horas Core-Tier1) | | | | | | | | | | line:91 18 | 18 | |
| ★ PD/Comunicación y coordinación (3 horas Core-Tier1) | | | | | | | | | | line:91 18 | 18 | |
| ★ PD/Análisis y programación de algoritmos paralelos (3 horas Core-Tier1) | | | | | | | | | | line:91 18 | 18 | |
| ★ PD/Arquitecturas paralelas (2 horas Core-Tier1) | | | | | | | | | | line:91 12 | 12 | |
| PD/Desempeño en paralelo | | | | | | | | | | line:91 18 | 18 | |
| ★ SE/Gestión de Proyectos de Software (2 horas Core-Tier1) | | | | | | | | | line:91 14 | | 14 | |
| ★ SE/Diseño de Software (5 horas Core-Tier1) | | | | | | | | | line:91 18 | | 18 | |
| Total | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 32 | 102 | | |

Cuadro 3.12: Tópicos por curso del $9^{no}$ al $9^{no}$ Semestre

| | Décimo Sem | | | | | . |
|---|---|---|---|---|---|---|
| **Unidades de Conocimiento** | CS3P2 line:25 | CS404 line:25 | AI367 line:25 | AI368 line:25 | FG350 line:25 | Total |
| PD/Sistemas distribuídos | line:25 15 | | | | | 15 |
| Total | 15 | 0 | 0 | 0 | 0 | |

Cuadro 3.13: Tópicos por curso del $10^{mo}$ al $10^{mo}$ Semestre

## 3.4. Resultados esperados distribuídos por curso

Las siquientes tablas nos muestras una visión global de los resultados que se esperan lograr en cada curso de la presente malla curricular. La lista completa de resultados esperados se encuentra en la Sección: 1.5.1.

Cuadro 3.14: Resultados esperados por curso $1^{er}$ al $1^{er}$ Semestre

Cuadro 3.15: Resultados esperados por curso $2^{do}$ al $2^{do}$ Semestre

Cuadro 3.16: Resultados esperados por curso $3^{er}$ al $3^{er}$ Semestre

Cuarto Sem

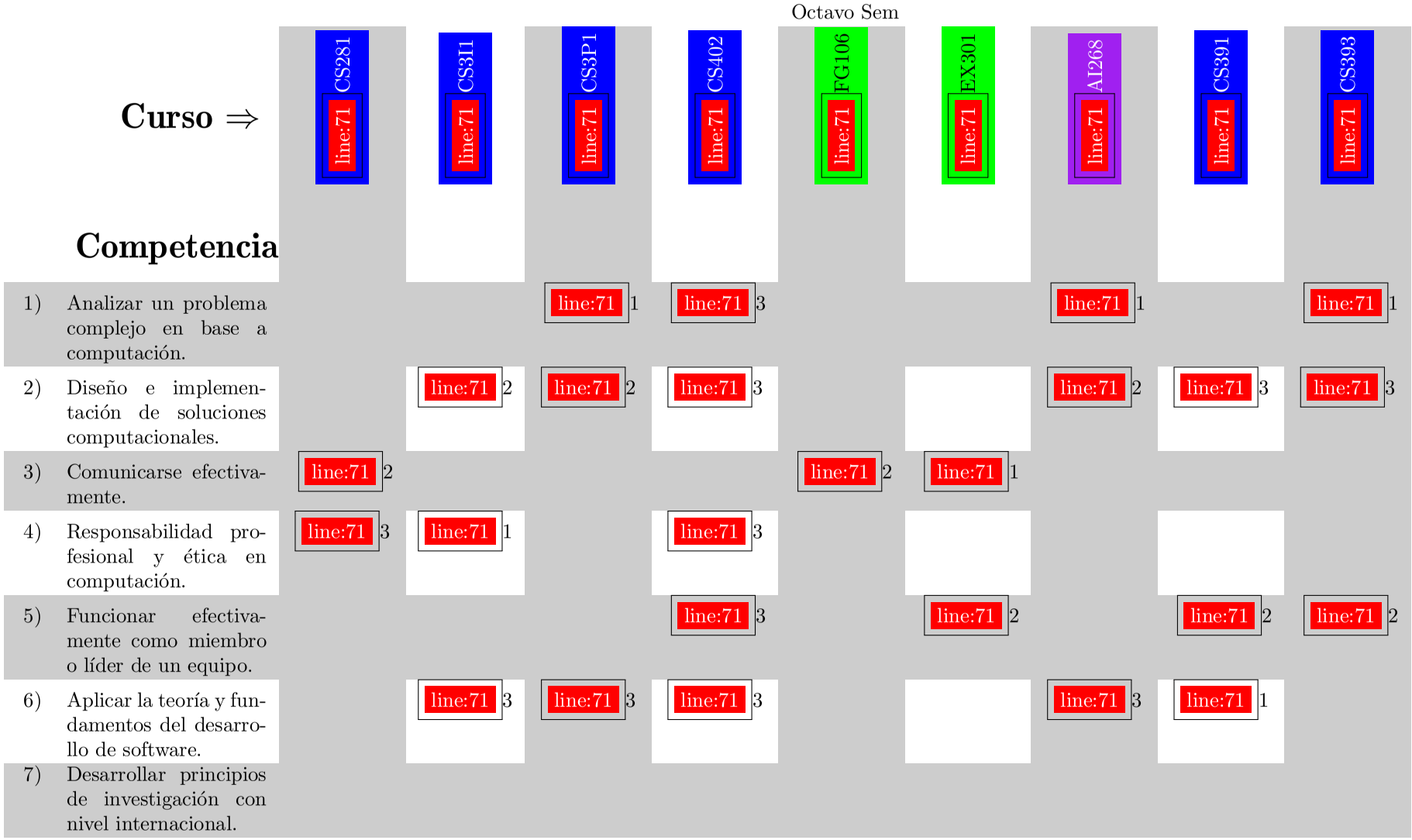
| **Competencia ⇓** / **Curso ⇒** | CS210 | CS211 | CS221 | CS271 | CS401 | MA202 | BEI202 |
|---|---|---|---|---|---|---|---|
| 1) Analizar un problema complejo en base a computación. | 3 | 2 | 2 | | 2 | 3 | |
| 2) Diseño e implementación de soluciones computacionales. | 1 | | | 2 | | | |
| 3) Comunicarse efectivamente. | | | | | 2 | | 3 |
| 4) Responsabilidad profesional y ética en computación. | | | | | 2 | | |
| 5) Funcionar efectivamente como miembro o líder de un equipo. | | | | | | | 2 |
| 6) Aplicar la teoría y fundamentos del desarrollo de software. | 2 | 3 | 3 | 3 | | 3 | 2 |
| 7) Desarrollar principios de investigación con nivel internacional. | | | | | 3 | | |

Cuadro 3.17: Resultados esperados por curso $4^{to}$ al $4^{to}$ Semestre

Cuadro 3.18: Resultados esperados por curso $5^{to}$ al $5^{to}$ Semestre

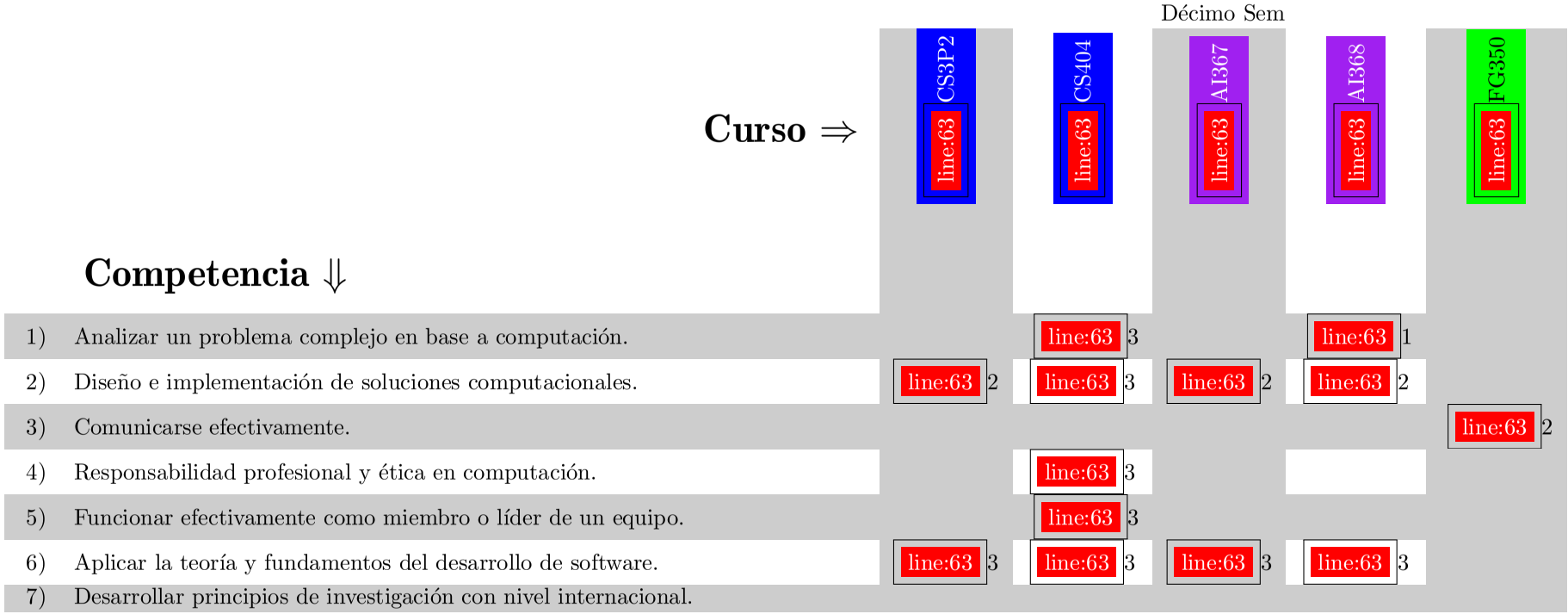
Cuadro 3.19: Resultados esperados por curso $6^{to}$ al $6^{to}$ Semestre

Cuadro 3.20: Resultados esperados por curso $7^{mo}$ al $7^{mo}$ Semestre




Cuadro 3.21: Resultados esperados por curso $8^{vo}$ al $8^{vo}$ Semestre

Noveno Sem
Curso =
Comp
CS370
CS400
CS403
AI365
CB309
CS351
CS353
AI369
CS392
CS3P3
line:73
1) Analizar un problema complejo en base a computación.
2) Diseño e implementación de soluciones computacionales.
3) Comunicarse efectivamente.
4) Responsabilidad profesional y ética en computación.
5) Funcionar efectivamente como miembro o líder de un equipo.
6) Aplicar la teoría y fundamentos




Cuadro 3.23: Resultados esperados por curso $10^{mo}$ al $10^{mo}$ Semestre

## 3.5. Resultados esperados distribuídos por curso

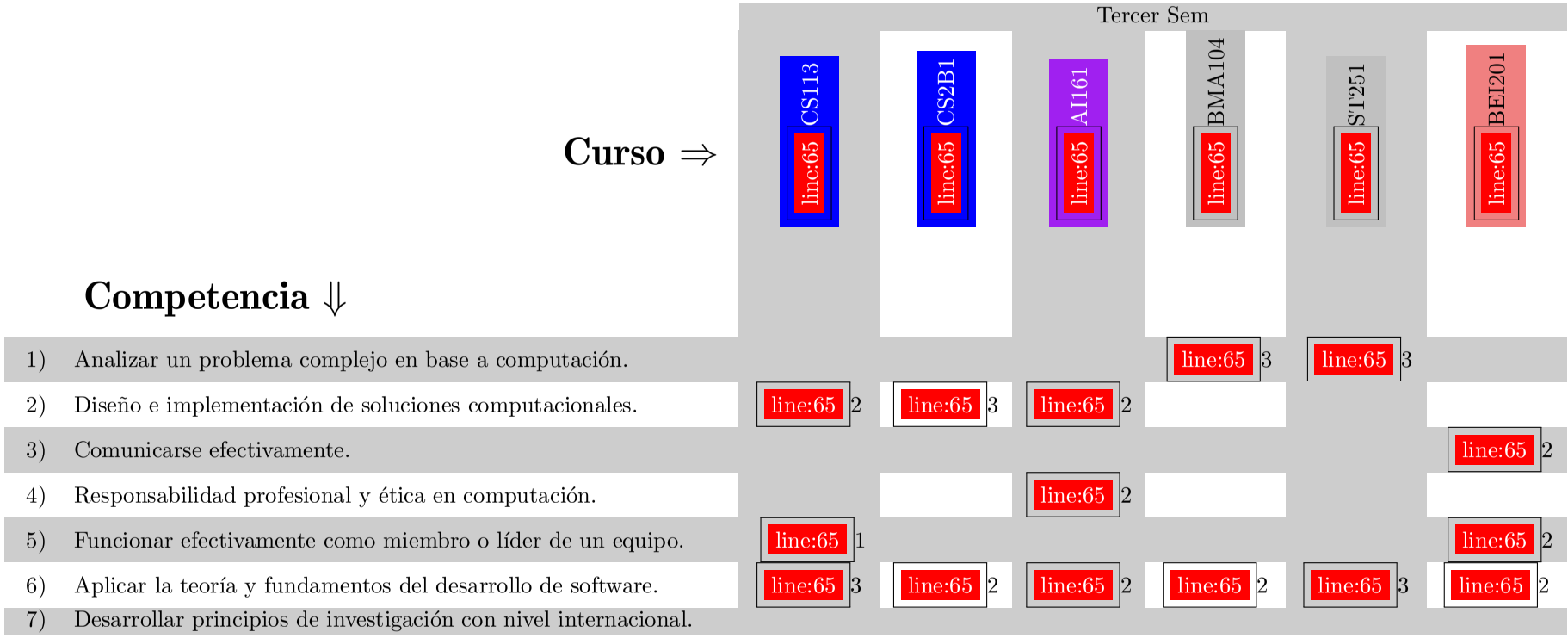
Las siquientes tablas nos muestras una visión global de los resultados que se esperan lograr en cada curso de la presente malla curricular. La lista completa de resultados esperados se encuentra en la Sección: 1.5.1.

| Primer Sem | | | | | | |
|---|---|---|---|---|---|---|
| **Competencia ⇓** / **Curso ⇑** | CS111 | BMA101 | BMA102 | BCH101 | FG001 | BEI101 |
| 1) Analizar un problema complejo en base a computación. | line:65 2 | line:65 3 | line:65 3 | line:65 2 | | |
| 2) Diseño e implementación de soluciones computacionales. | | | | | line:65 1 line:65 2 | line:65 3 |
| 3) Comunicarse efectivamente. | | | | | | |
| 4) Responsabilidad profesional y ética en computación. | | | | | | |
| 5) Funcionar efectivamente como miembro o líder de un equipo. | line:65 3 | line:65 2 | line:65 2 | line:65 1 | | line:65 1 |
| 6) Aplicar la teoría y fundamentos del desarrollo de software. | | | | | | |
| 7) Desarrollar principios de investigación con nivel internacional. | | | | | | |

Cuadro 3.24: Resultados esperados por curso $1^{er}$ al $1^{er}$ Semestre

| Competencia ⇓ / Curso ⇒ | CS100 | CS112 | CS1D1 | BMA103 | BFI101 | BEI102 | Segundo Sem |
|---|---|---|---|---|---|---|---|
| 1) Analizar un problema complejo en base a computación. | line:65 2 | | line:65 2 | line:65 2 | line:65 3 | | |
| 2) Diseño e implementación de soluciones computacionales. | | line:65 2 | | | | line:65 2 | |
| 3) Comunicarse efectivamente. | | | | | | | |
| 4) Responsabilidad profesional y ética en computación. | | | | | | | |
| 5) Funcionar efectivamente como miembro o líder de un equipo. | | | | | | line:65 2 | |
| 6) Aplicar la teoría y fundamentos del desarrollo de software. | line:65 3 | line:65 3 | line:65 2 | line:65 1 | line:65 2 | line:65 2 | |
| 7) Desarrollar principios de investigación con nivel internacional. | | | | | | | |

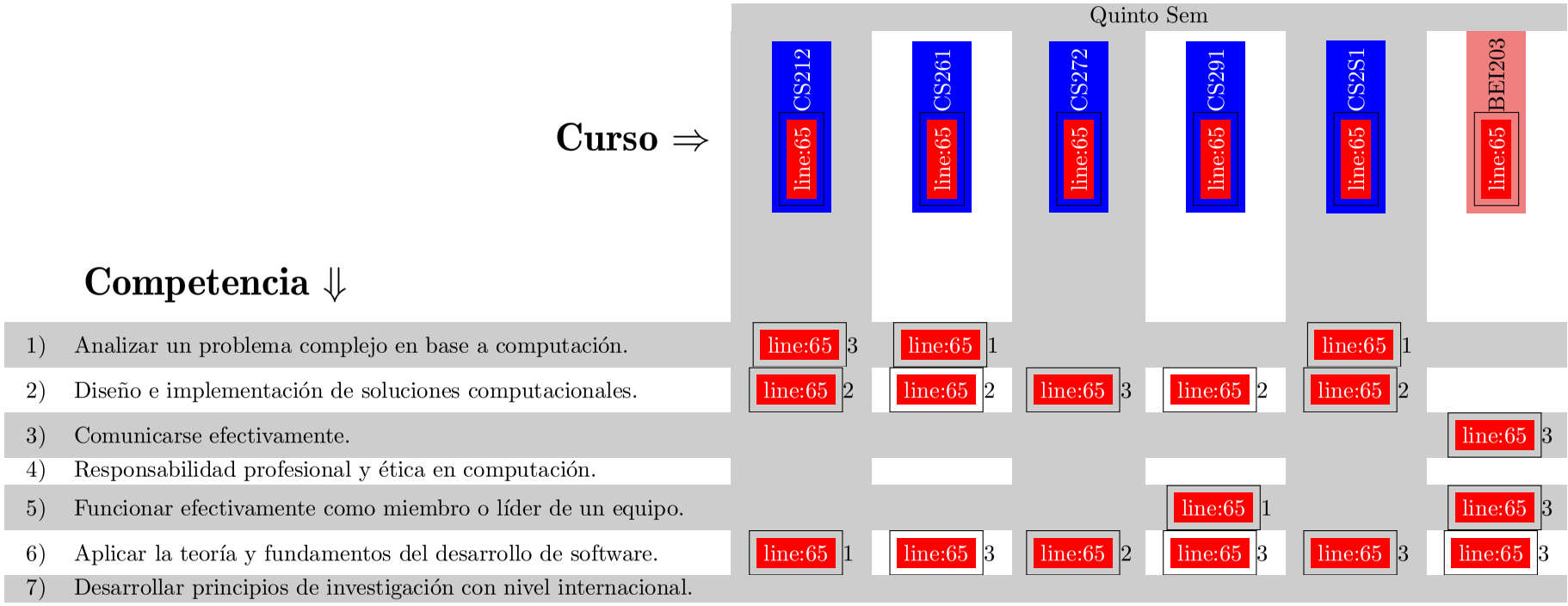
Cuadro 3.25: Resultados esperados por curso $2^{do}$ al $2^{do}$ Semestre




Cuadro 3.26: Resultados esperados por curso $3^{er}$ al $3^{er}$ Semestre

| Cuarto Sem | Competencia ⇓ / Curso ⇒ | CS210 | CS211 | CS221 | CS271 | CS401 | MA202 | BEI202 |
|---|---|---|---|---|---|---|---|---|
| | 1) Analizar un problema complejo en base a computación. | line:67 3 | line:67 2 | line:67 2 | | line:67 2 | line:67 3 | |
| | 2) Diseño e implementación de soluciones computacionales. | line:67 1 | | | line:67 2 | | | |
| | 3) Comunicarse efectivamente. | | | | | line:67 2 | | line:67 3 |
| | 4) Responsabilidad profesional y ética en computación. | | | | | line:67 2 | | |
| | 5) Funcionar efectivamente como miembro o líder de un equipo. | | | line:67 3 | line:67 3 | | | line:67 2 |
| | 6) Aplicar la teoría y fundamentos del desarrollo de software. | line:67 2 | line:67 3 | | | | line:67 3 | line:67 2 |
| | 7) Desarrollar principios de investigación con nivel internacional. | | | | | line:67 3 | | |

Cuadro 3.27: Resultados esperados por curso $4^{to}$ al $4^{to}$ Semestre




Cuadro 3.28: Resultados esperados por curso $5^{to}$ al $5^{to}$ Semestre

Sexto Sem

| Competencia ⇓ / Curso ⇑ | CS231 | CS311 | CS312 | CS342 | AI263 | FI201 |
|---|---|---|---|---|---|---|
| 1) Analizar un problema complejo en base a computación. | line:65 1 | line:65 3 | line:65 2 | line:65 1 | | line:65 3 |
| 2) Diseño e implementación de soluciones computacionales. | line:65 2 | line:65 1 | line:65 1 | line:65 2 | line:65 3 | |
| 3) Comunicarse efectivamente. | | | | | | |
| 4) Responsabilidad profesional y ética en computación. | | | | | line:65 2 | |
| 5) Funcionar efectivamente como miembro o líder de un equipo. | | | | | | |
| 6) Aplicar la teoría y fundamentos del desarrollo de software. | line:65 3 | line:65 2 | line:65 3 | line:65 3 | line:65 3 | line:65 2 |
| 7) Desarrollar principios de investigación con nivel internacional. | | | | | | |

Cuadro 3.29: Resultados esperados por curso $6^{to}$ al $6^{to}$ Semestre

Cuadro 3.30: Resultados esperados por curso $7^{mo}$ al $7^{mo}$ Semestre

Octavo Sem

| Competencia | Curso ⇑ | CS281 | CS31I | CS3P1 | CS402 | FG106 | EX301 | AI268 | CS391 | CS393 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1) | Analizar un problema complejo en base a computación. | | | line:71 1 | line:71 3 | | | line:71 1 | | line:71 1 |
| 2) | Diseño e implementación de soluciones computacionales. | | line:71 2 | line:71 2 | line:71 3 | | | line:71 2 | line:71 3 | line:71 3 |
| 3) | Comunicarse efectivamente. | line:71 2 | | | | line:71 2 | line:71 1 | | | |
| 4) | Responsabilidad profesional y ética en computación. | line:71 3 | line:71 1 | | line:71 3 | | | | | |
| 5) | Funcionar efectivamente como miembro o líder de un equipo. | | | | line:71 3 | | line:71 2 | | line:71 2 | line:71 2 |
| 6) | Aplicar la teoría y fundamentos del desarrollo de software. | | line:71 3 | line:71 3 | line:71 3 | | | line:71 3 | line:71 1 | |
| 7) | Desarrollar principios de investigación con nivel internacional. | | | | | | | | | |

Cuadro 3.31: Resultados esperados por curso $8^{vo}$ al $8^{vo}$ Semestre

Noveno Sem

**Curso =**

CS370 CS400 CS403 AI365 CB309 CS351 CS353 AI369 CS392 CS3P3

**Comp**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1) Analizar un problema complejo en base a computación. | line:73 2 | line:73 3 | | | | line:73 2 | | | | |
| 2) Diseño e implementación de soluciones computacionales. | line:73 2 | line:73 3 | line:73 3 | line:73 3 | line:73 2 | line:73 2 | line:73 2 | line:73 2 | | line:73 2 |
| 3) Comunicarse efectivamente. | line:73 3 | | | | | | | | | |
| 4) Responsabilidad profesional y ética en computación. | line:73 3 | line:73 3 | line:73 2 | line:73 2 | line:73 2 | | line:73 1 | | | |
| 5) Funcionar efectivamente como miembro o líder de un equipo. | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 | | line:73 3 | line:73 3 | line:73 3 |
| 6) Aplicar la teoría y fundamentos del desa- | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 | line:73 3 |

Cuadro 3.33: Resultados esperados por curso $10^{mo}$ al $10^{mo}$ Semestre

## 3.6. Distribución de cursos en la carrera

Esta propuesta puede ser analizada por el número de créditos dedicados a cada área, por niveles de cursos (Introductorios, Intermedios, Avanzados y Proyectos).

| | **Estudios generales** | **Específicos** | **Especialización** | |
|---|---|---|---|---|
| Primer Sem | 19 | 4 | | 23 |
| Segundo Sem | 12 | 11 | | 23 |
| Tercer Sem | 10 | 11 | | 21 |
| Cuarto Sem | 5 | 15 | 2 | 22 |
| Quinto Sem | 2 | 19 | | 21 |
| Sexto Sem | 3 | 18 | | 21 |
| Séptimo Sem | 4 | 14 | 4 | 22 |
| Octavo Sem | 4 | 12 | 3 | 19 |
| Noveno Sem | | 7 | 12 | 19 |
| Décimo Sem | 2 | 7 | 7 | 16 |
| **Total** | 61 | 118 | 28 | 207 |
| | 29.4 % | 57 % | 13.5 % | |

Cuadro 3.34: Distribución de cursos por áreas

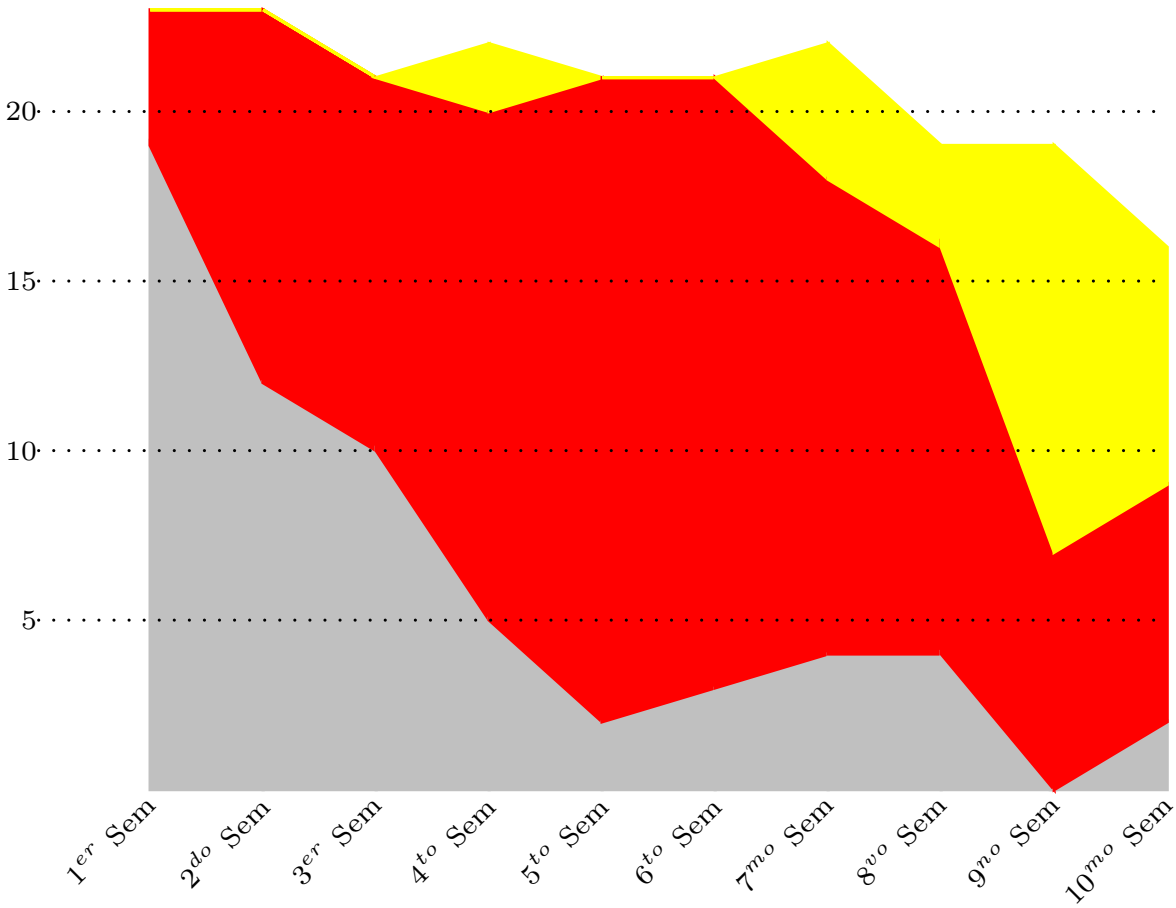



Figura 3.2: Créditos por área por semestre

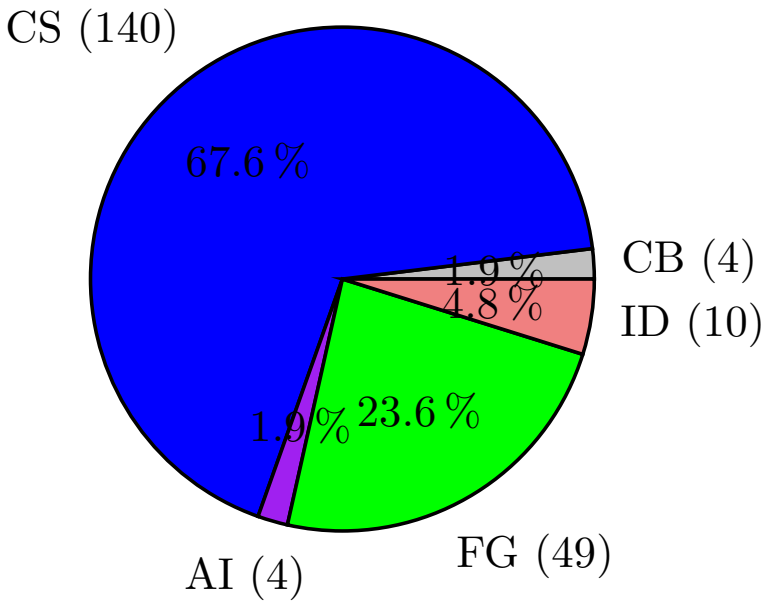



Figura 3.3: Distribución de cursos por áreas considerando creditaje.

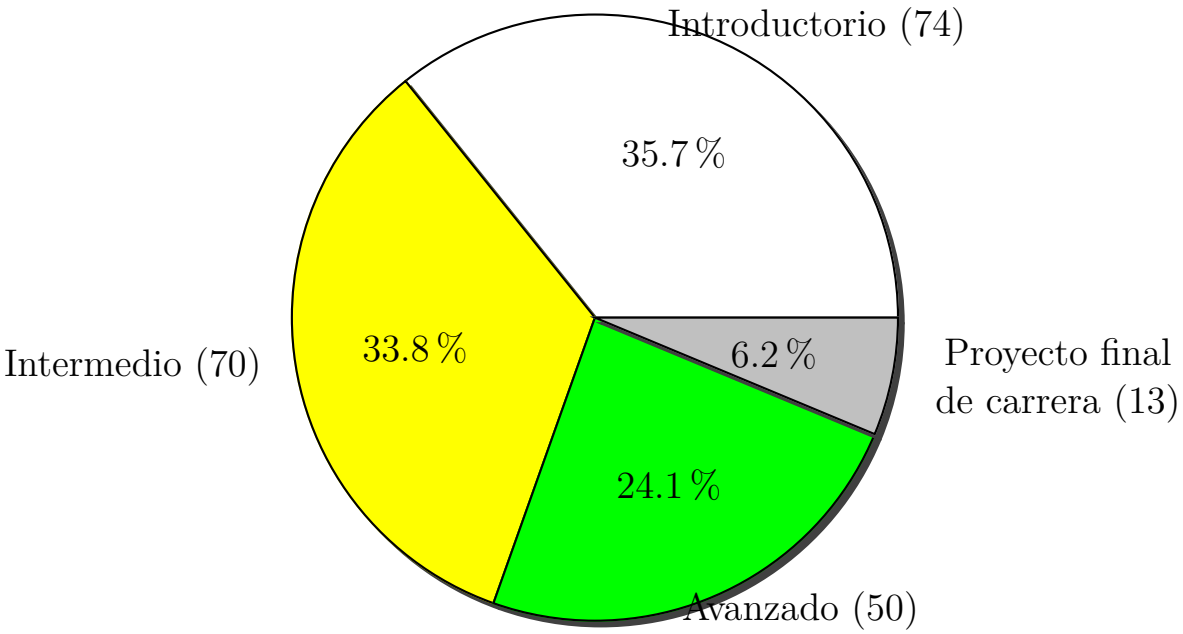



Figura 3.4: Distribución de créditos por niveles de cursos.

## 3.7. Compatibilidad de la carrera con relación a estandares internacionales

En esta sección presentamos la distribución de cursos por áreas de concentración en contraste con las propuestas internacionales de las carreras de la *Computing Curricula* de IEEE-CS/ACM.

Es necesario notar que en algunos casos las materias podrían aparecen en más de un eje pues tienen contenido de más de una área. Por ejemplo, la materia de sistemas operativos contiene unidades de aplicación que pueden ser clasificadas en Tecnología de Información pero al mismo tiempo contiene fundamentos de como está estructurado un Sistema Operativo que es del eje de Ciencia de la

Computación. En estos casos el creditaje ha sido divido entre los ejes correspondientes.

1. Hardware y Arquitectura (7.5 Créditos)
   - line:4 CS221. Arquitectura de Computadores ($4^{to}$ Sem-Pág line:4 ??)
   - line:5 AI369. Robótica ($9^{no}$ Sem-Pág line:5 ??)
2. Ciencia de la Computación (206.66 Créditos)
   - line:9 CS111. Introducción a la Programación ($1^{er}$ Sem-Pág line:9 ??)
   - line:10 CS100. Introducción a la Ciencia de la Computación ($2^{do}$ Sem-Pág line:10 ??)
   - line:11 CS112. Programación Orientada a Objetos I ($2^{do}$ Sem-Pág line:11 ??)
   - line:12 CS113. Programación Orientada a Objetos II ($3^{er}$ Sem-Pág line:12 ??)
   - line:13 AI161. IA Aplicada ($3^{er}$ Sem-Pág line:13 ??)
   - line:14 CS210. Algoritmos y Estructuras de Datos ($4^{to}$ Sem-Pág line:14 ??)
   - line:15 CS211. Teoría de la Computación ($4^{to}$ Sem-Pág line:15 ??)
   - line:16 CS271. Bases de Datos I ($4^{to}$ Sem-Pág line:16 ??)
   - line:17 CS401. Metodología de la Investigación ($4^{to}$ Sem-Pág line:17 ??)
   - line:18 CS212. Análisis y Diseño de Algoritmos ($5^{to}$ Sem-Pág line:18 ??)
   - line:19 CS261. Inteligencia Artificial ($5^{to}$ Sem-Pág line:19 ??)
   - line:20 CS272. Bases de Datos II ($5^{to}$ Sem-Pág line:20 ??)
   - line:21 CS2S1. Sistemas Operativos ($5^{to}$ Sem-Pág line:21 ??)
   - line:22 CS231. Redes y Comunicación ($6^{to}$ Sem-Pág line:22 ??)
   - line:23 CS312. Estructuras de Datos Avanzadas ($6^{to}$ Sem-Pág line:23 ??)
   - line:24 CS342. Compiladores ($6^{to}$ Sem-Pág line:24 ??)
   - line:25 AI263. Introducción al Aprendizaje de Máquina ($6^{to}$ Sem-Pág line:25 ??)
   - line:26 CS251. Computación Gráfica ($7^{mo}$ Sem-Pág line:26 ??)
   - line:27 CS2H1. Experiencia de Usuario (UX) ($7^{mo}$ Sem-Pág line:27 ??)
   - line:28 AI264. Aprendizaje Profundo ($7^{mo}$ Sem-Pág line:28 ??)
   - line:29 CS281. Computación en la Sociedad ($8^{vo}$ Sem-Pág line:29 ??)
   - line:30 CS3P1. Computación Paralela y Distribuída ($8^{vo}$ Sem-Pág line:30 ??)
   - line:31 CS402. Proyecto de tesis 1 ($8^{vo}$ Sem-Pág line:31 ??)
   - line:32 AI268. Visión Computacional ($8^{vo}$ Sem-Pág line:32 ??)
   - line:33 CS370. Big Data ($9^{no}$ Sem-Pág line:33 ??)
   - line:34 CS403. Proyecto de tesis 2 ($9^{no}$ Sem-Pág line:34 ??)
   - line:35 AI365. Modelos Generativos Avanzados en IA ($9^{no}$ Sem-Pág line:35 ??)
   - line:36 CB309. Bioinformática ($9^{no}$ Sem-Pág line:36 ??)

- line:37 CS351. Tópicos en Computación Gráfica ($9^{no}$ Sem-Pág line:37 ??)
- line:38 CS353. Computación Cuántica ($9^{no}$ Sem-Pág line:38 ??)
- line:39 AI369. Robótica ($9^{no}$ Sem-Pág line:39 ??)
- line:40 CS3P3. Internet de las Cosas ($9^{no}$ Sem-Pág line:40 ??)
- line:41 CS3P2. Cloud Computing ($10^{mo}$ Sem-Pág line:41 ??)
- line:42 CS404. Taller de Investigación ($10^{mo}$ Sem-Pág line:42 ??)
- line:43 AI367. Tópicos en Inteligencia Artificial ($10^{mo}$ Sem-Pág line:43 ??)
- line:44 AI368. Computación Evolutiva ($10^{mo}$ Sem-Pág line:44 ??)

3. Sistemas de Información
   - Ninguno
4. Tecnología de Información (32.16 Créditos)
   - line:52 CS2B1. Desarrollo Basado en Plataformas ($3^{er}$ Sem-Pág line:52 ??)
   - line:53 CS271. Bases de Datos I ($4^{to}$ Sem-Pág line:53 ??)
   - line:54 CS2S1. Sistemas Operativos ($5^{to}$ Sem-Pág line:54 ??)
   - line:55 CS231. Redes y Comunicación ($6^{to}$ Sem-Pág line:55 ??)
   - line:56 CS2H1. Experiencia de Usuario (UX) ($7^{mo}$ Sem-Pág line:56 ??)
   - line:57 CS281. Computación en la Sociedad ($8^{vo}$ Sem-Pág line:57 ??)
   - line:58 CS3I1. Seguridad en Computación ($8^{vo}$ Sem-Pág line:58 ??)
   - line:59 CS3P3. Internet de las Cosas ($9^{no}$ Sem-Pág line:59 ??)
   - line:60 CS3P2. Cloud Computing ($10^{mo}$ Sem-Pág line:60 ??)
5. Ingeniería de Software (35.66 Créditos)
   - line:64 CS112. Programación Orientada a Objetos I ($2^{do}$ Sem-Pág line:64 ??)
   - line:65 CS113. Programación Orientada a Objetos II ($3^{er}$ Sem-Pág line:65 ??)
   - line:66 CS291. Ingeniería de Software I ($5^{to}$ Sem-Pág line:66 ??)
   - line:67 CS292. Ingeniería de Software II ($7^{mo}$ Sem-Pág line:67 ??)
   - line:68 CS2H1. Experiencia de Usuario (UX) ($7^{mo}$ Sem-Pág line:68 ??)
   - line:69 CS391. Ingeniería de Software III ($8^{vo}$ Sem-Pág line:69 ??)
   - line:70 CS393. Sistemas de Infomación ($8^{vo}$ Sem-Pág line:70 ??)
   - line:71 CS392. Tópicos en Ingeniería de Software ($9^{no}$ Sem-Pág line:71 ??)
6. Matemática para Computación (14 Créditos)
   - line:75 CS1D1. Estructuras Discretas ($2^{do}$ Sem-Pág line:75 ??)
   - line:76 CS311. Programación Competitiva ($6^{to}$ Sem-Pág line:76 ??)
7. Ciencias Básicas (84 Créditos)

- line:80 BMA101. Algebra Linear ($1^{er}$ Sem-Pág line:80 **??**)
- line:81 BMA102. Cálculo Diferencial ($1^{er}$ Sem-Pág line:81 **??**)
- line:82 BCH101. Química I ($1^{er}$ Sem-Pág line:82 **??**)
- line:83 BMA103. Cálculo Integral ($2^{do}$ Sem-Pág line:83 **??**)
- line:84 BFI101. Física I ($2^{do}$ Sem-Pág line:84 **??**)
- line:85 BMA104. Cálculo Diferencial e Integral Avanzado ($3^{er}$ Sem-Pág line:85 **??**)
- line:86 ST251. Estadística y Probabilidades ($3^{er}$ Sem-Pág line:86 **??**)
- line:87 MA202. Métodos Numéricos ($4^{to}$ Sem-Pág line:87 **??**)
- line:88 FI201. Física Computacional ($6^{to}$ Sem-Pág line:88 **??**)
- line:89 BBI101. Biología ($7^{mo}$ Sem-Pág line:89 **??**)

8. Contenido Empresarial

- Ninguno

9. Formación General (38 Créditos)

- line:97 FG001. Electivo Formación General ($1^{er}$ Sem-Pág line:97 **??**)
- line:98 BEI101. Inglés I ($1^{er}$ Sem-Pág line:98 **??**)
- line:99 BEI102. Inglés II ($2^{do}$ Sem-Pág line:99 **??**)
- line:100 BEI201. Inglés III ($3^{er}$ Sem-Pág line:100 **??**)
- line:101 BEI202. Inglés IV ($4^{to}$ Sem-Pág line:101 **??**)
- line:102 BEI203. Inglés V ($5^{to}$ Sem-Pág line:102 **??**)
- line:103 FG211. Ética Profesional ($7^{mo}$ Sem-Pág line:103 **??**)
- line:104 FG106. Teatro ($8^{vo}$ Sem-Pág line:104 **??**)
- line:105 EX301. Actividades Extracurriculares ($8^{vo}$ Sem-Pág line:105 **??**)
- line:106 CS400. Prácticas Pre-profesionales ($9^{no}$ Sem-Pág line:106 **??**)

Considerando esta distribución, las próximas figuras nos brindan una visión gráfica en comparación con los estandares internacionales presentados en presentadas por IEEE-CS/ACM en la *Computing Curricula* [ACM/IEEE-CS, 2020].

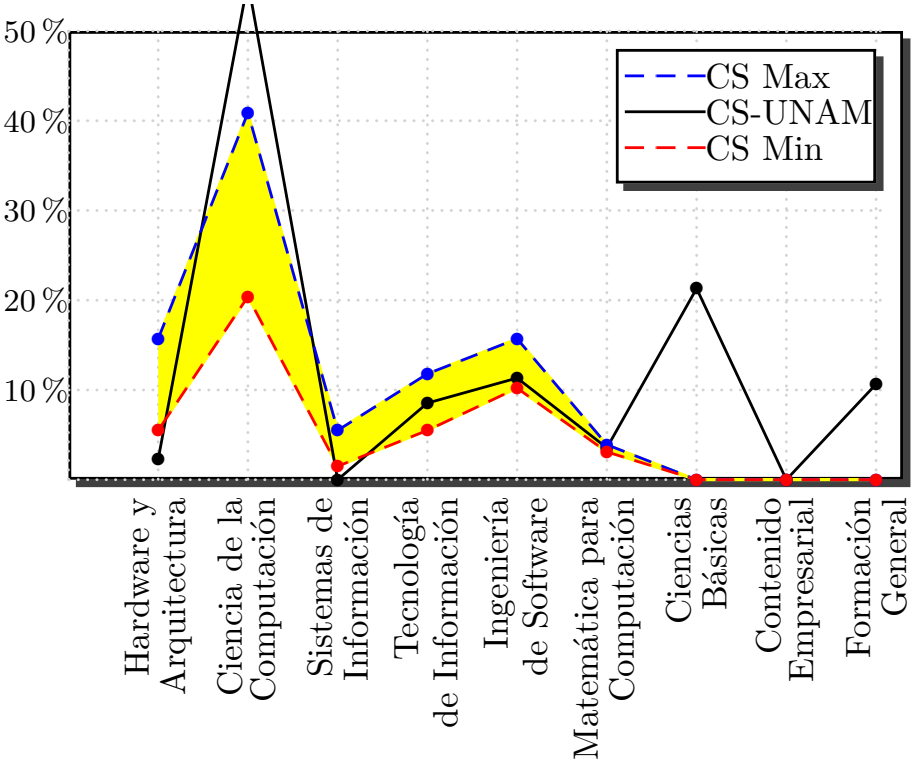



Figura 3.5: Comparación de CS-UNAM con CS de ACM/IEEE-CS.

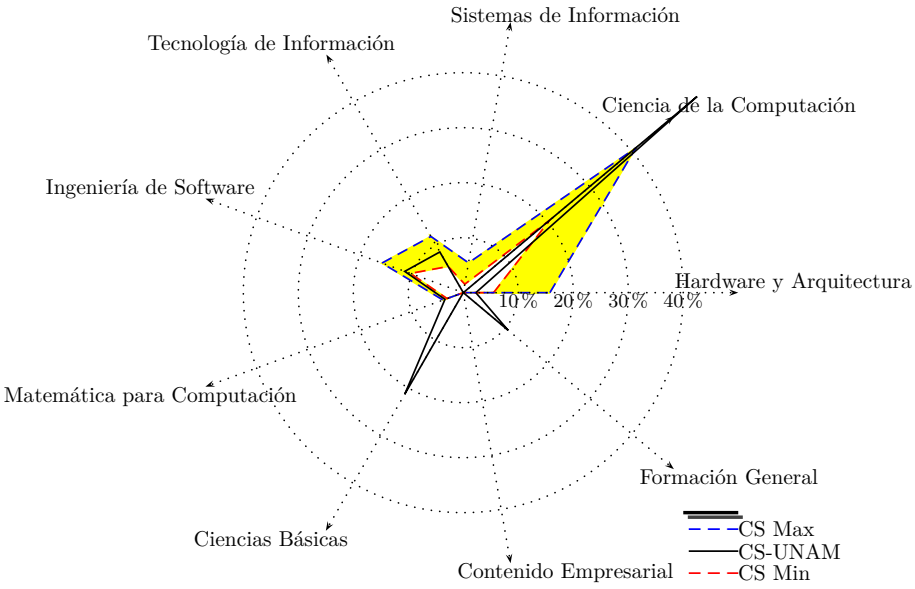



Figura 3.6: Comparación de CS-UNAM con CS de ACM/IEEE-CS.

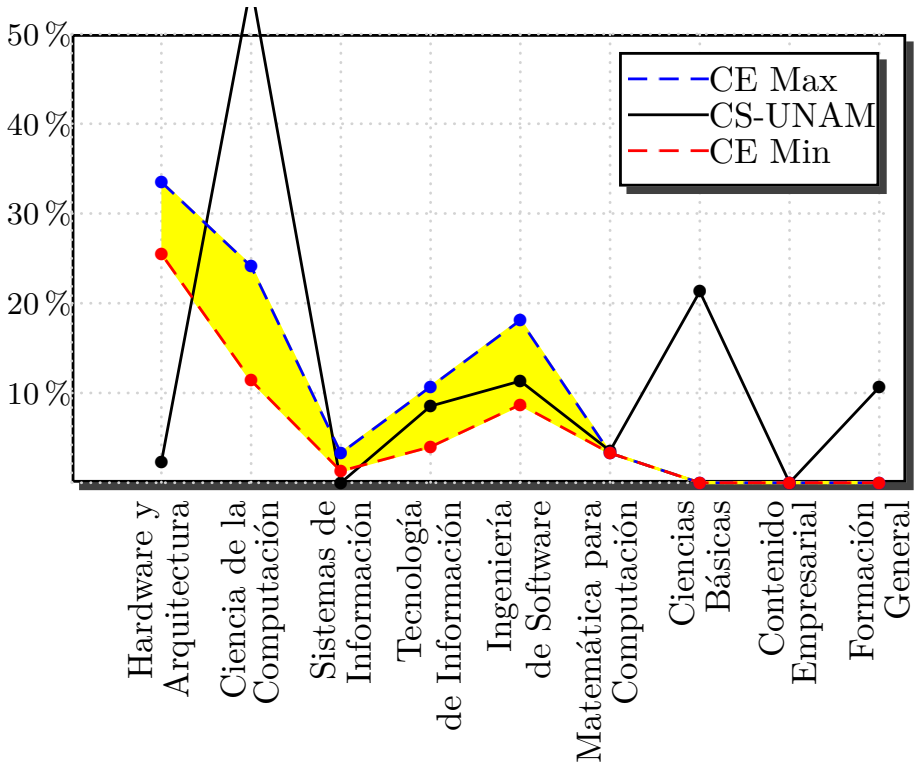



Figura 3.7: Comparación de CS-UNAM con CE de ACM/IEEE-CS.

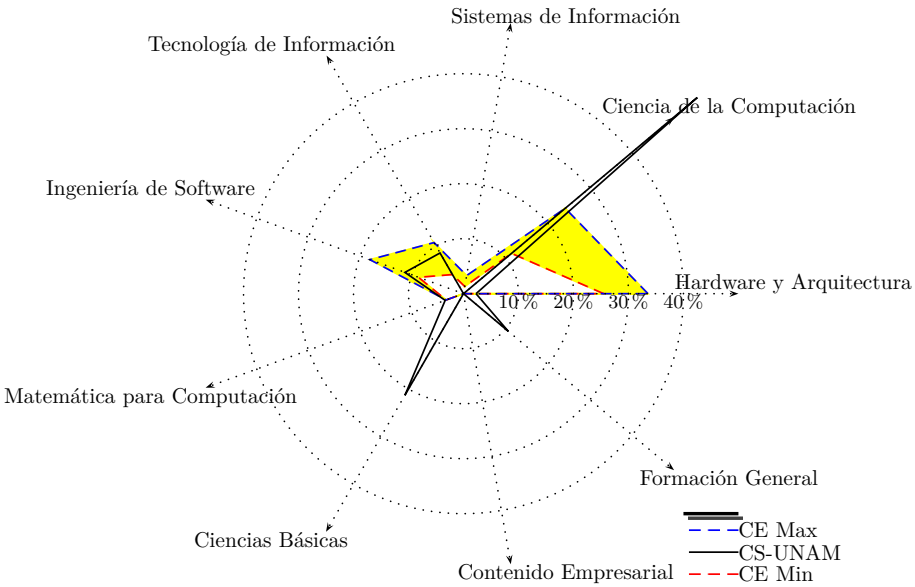



Figura 3.8: Comparación de CS-UNAM con CE de ACM/IEEE-CS.

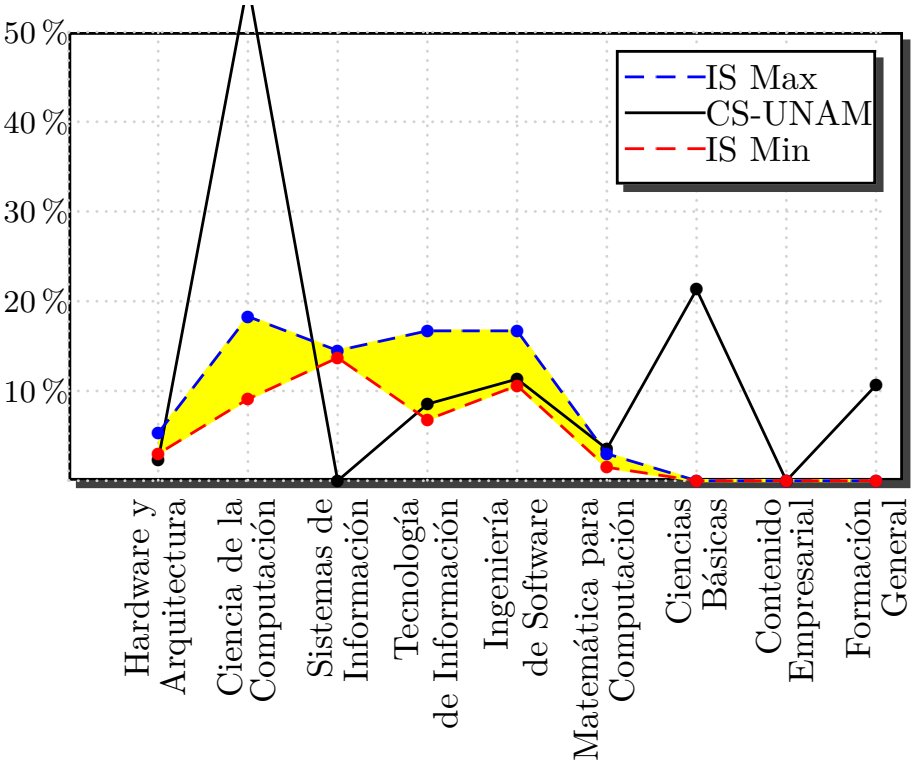



Figura 3.9: Comparación de CS-UNAM con IS de ACM/IEEE-CS.

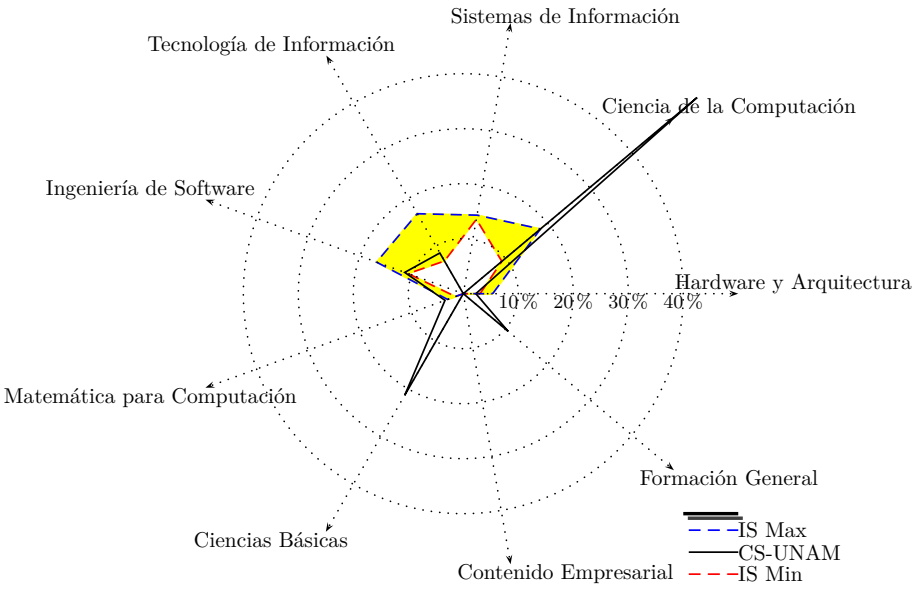



Figura 3.10: Comparación de CS-UNAM con IS de ACM/IEEE-CS.

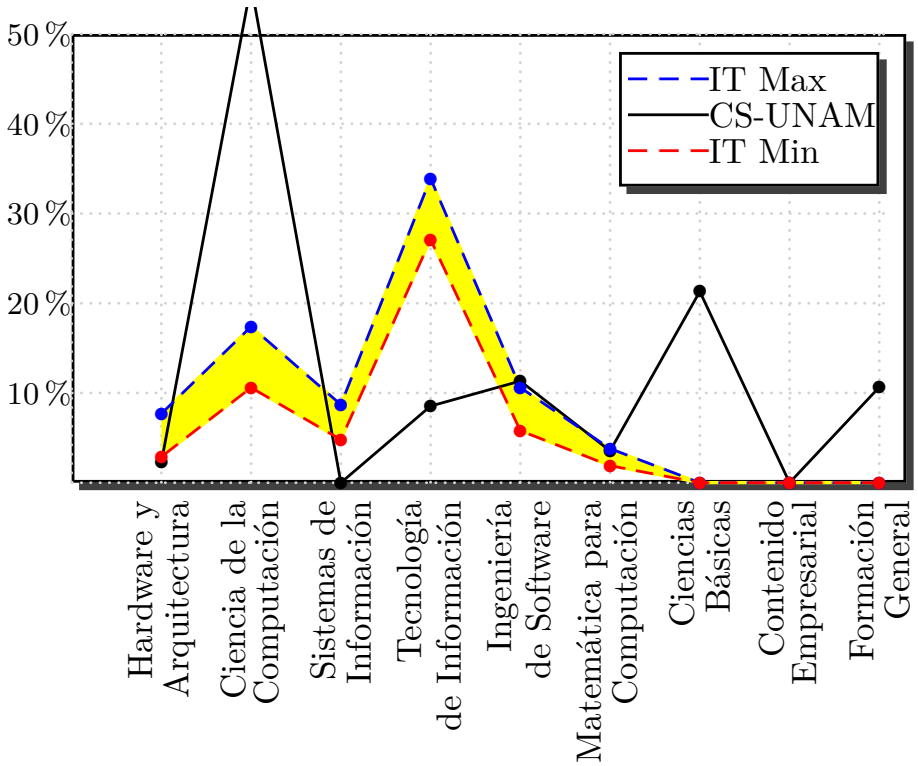



Figura 3.11: Comparación de CS-UNAM con IT de ACM/IEEE-CS.

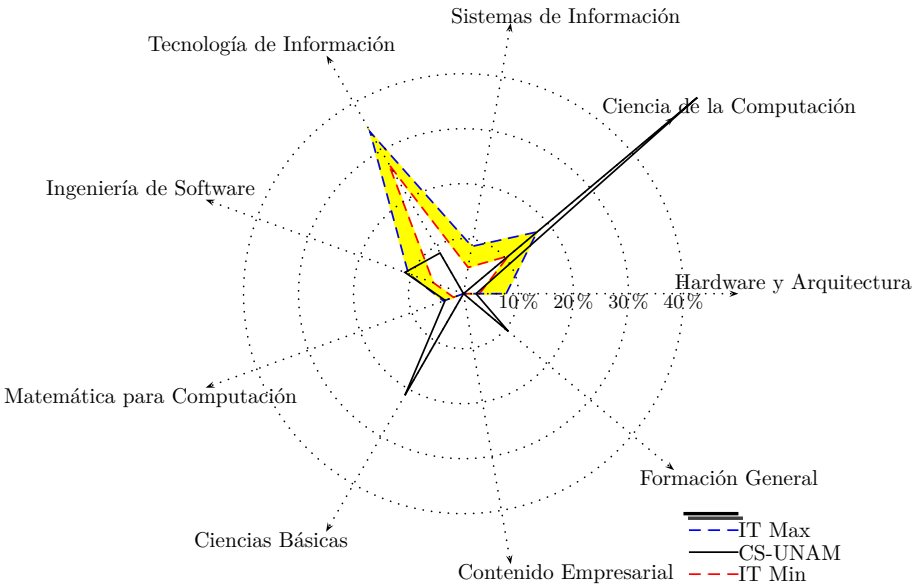



Figura 3.12: Comparación de CS-UNAM con IT de ACM/IEEE-CS.

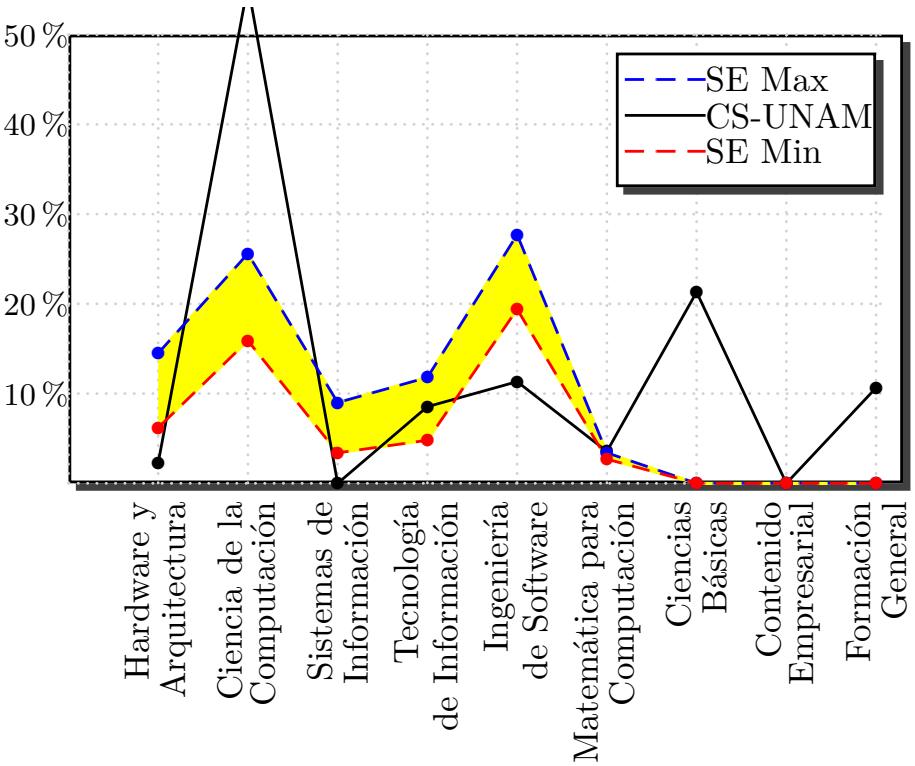



Figura 3.13: Comparación de CS-UNAM con SE de ACM/IEEE-CS.

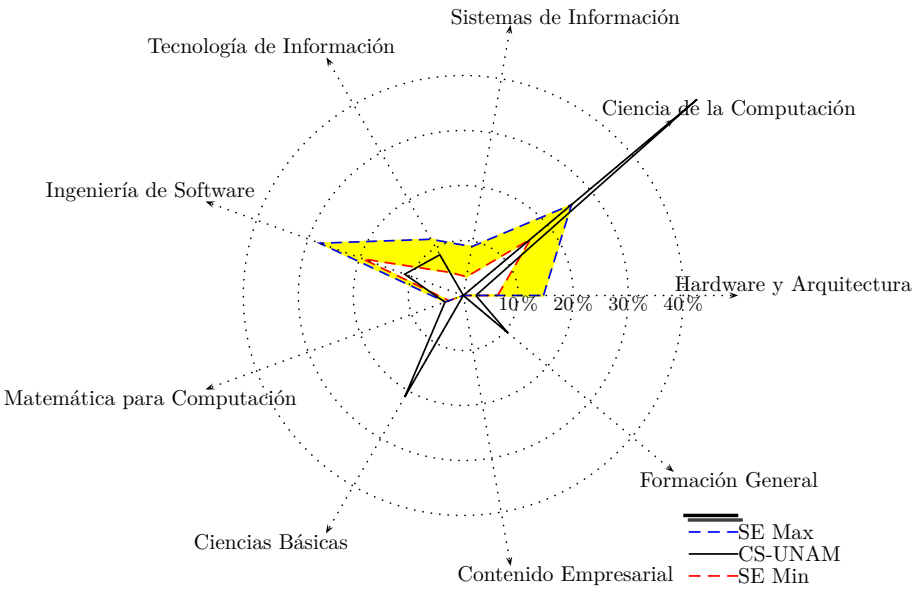



Figura 3.14: Comparación de CS-UNAM con SE de ACM/IEEE-CS.

## Referencias Bibliográficas

[ACM and IEEE-CS, 2017] ACM and IEEE-CS (2017). Information technology curricula 2017. Technical report, ACM and IEEE-CS.

[ACM Data Science Task Force, 2021] ACM Data Science Task Force (2021). Computing competencies for undergraduate data science curricula. Technical report, Association for Computing Machinery (ACM).

[ACM/IEEE-CS, 2020] ACM/IEEE-CS (2020). Computing curricula 2020. Technical report, ACM Press and IEEE Computer Society Press.

[ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017] ACM/IEEE-CS Joint Task Force on Cybersecurity Education (2017). Cybersecurity curricula 2017. Technical report, ACM Press and IEEE Computer Society Press and Association for Information Systems Special Interest Group on Information Security and Privacy (AIS SIGSEC) and International Federation for Information Processing Technical Committee on Information Security Education (IFIP WG 11.8).

[ACM/IEEE-CS/AAAI Joint Task Force, 2023] ACM/IEEE-CS/AAAI Joint Task Force (2023). Cs2023: Acm/ieee-cs/aaai computer science curricula. Technical report, ACM Press and IEEE Computer Society Press and AAAI Press.

[Díaz-Herrera and Hilburn, 2004] Díaz-Herrera, J. L. and Hilburn, T. B. (2004). Software engineering: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical report, ACM, IEEE.

[Force, 2020] Force, T. J. A. I. T. (2020). A competency model for undergraduate programs in information systems. Technical report, ACM Press and AIS Press.

[Shakelford et al., 2005] Shakelford, R., Cross, J. H., Davies, G., Impagliazzo, J., Kamali, R., LeBlanc, R., Lunt, B., McGettrick, A., Sloan, R., and Topi, H. (2005). Computing curricula 2005. Technical report, ACM/IEEE, http://www.acm.org/education.

[Soldan et al., 2016] Soldan, D., Aylor, J., Clements, A., Engel, G., Hoelzeman, R., Hughes, E. A., Hughes, J. L., Impagliazzo, J., Jaeger, R. C., Klenke, R., Lyon, D. A., McGettrick, A., Nelson, V. P., Neebel, D. J., Page, I., Peterson, G. D., Ranganathan, N., Sloan, R., Srimani, P. K., Theys, M. D., Wolf, W., and Varanasi, M. (2016). Computer engineering curricula 2016. Technical report, ACM and IEEE-CS.

# Capítulo 4

# Contenido detallado por curso

# Capítulo 5

# Profesores & Cursos

## 5.1. Cursos asignados por profesor

## 5.2. Profesor asignado por curso

- Primer Semestre
  - line:6 CS111. Introducción a la Programación
  - line:11 BMA101. Algebra Linear
  - line:16 BMA102. Cálculo Diferencial
  - line:21 BCH101. Química I
  - line:26 FG001. Electivo Formación General
  - line:31 BEI101. Inglés I
- Segundo Semestre
  - line:40 CS100. Introducción a la Ciencia de la Computación
  - line:45 CS112. Programación Orientada a Objetos I
  - line:50 CS1D1. Estructuras Discretas
  - line:55 BMA103. Cálculo Integral
  - line:60 BFI101. Física I
  - line:65 BEI102. Inglés II
- Tercer Semestre
  - line:74 CS113. Programación Orientada a Objetos II
  - line:79 CS2B1. Desarrollo Basado en Plataformas
  - line:84 AI161. IA Aplicada

- line:89 BMA104. Cálculo Diferencial e Integral Avanzado
  - –
- line:94 ST251. Estadística y Probabilidades
  - –
- line:99 BEI201. Inglés III
  - –

- Cuarto Semestre
  - line:108 CS210. Algoritmos y Estructuras de Datos
    - –
  - line:113 CS211. Teoría de la Computación
    - –
  - line:118 CS221. Arquitectura de Computadores
    - –
  - line:123 CS271. Bases de Datos I
    - –
  - line:128 CS401. Metodología de la Investigación
    - –
  - line:133 MA202. Métodos Numéricos
    - –
  - line:138 BEI202. Inglés IV
    - –
- Quinto Semestre
  - line:147 CS212. Análisis y Diseño de Algoritmos
    - –
  - line:152 CS261. Inteligencia Artificial
    - –
  - line:157 CS272. Bases de Datos II
    - –
  - line:162 CS291. Ingeniería de Software I
    - –
  - line:167 CS2S1. Sistemas Operativos
    - –
  - line:172 BEI203. Inglés V
    - –
- Sexto Semestre
  - line:181 CS231. Redes y Comunicación
    - –
  - line:186 CS311. Programación Competitiva
    - –
  - line:191 CS312. Estructuras de Datos Avanzadas
    - –
  - line:196 CS342. Compiladores
    - –
  - line:201 AI263. Introducción al Aprendizaje de Máquina
    - –
  - line:206 FI201. Física Computacional
    - –
- Séptimo Semestre
  - line:215 CS251. Computación Gráfica
    - –
  - line:220 CS292. Ingeniería de Software II

- line:225 CS2H1. Experiencia de Usuario (UX)
- line:230 AI264. Aprendizaje Profundo
- line:235 BBI101. Biología
- line:240 FG211. Ética Profesional

- Octavo Semestre
  - line:249 CS281. Computación en la Sociedad
  - line:254 CS3I1. Seguridad en Computación
  - line:259 CS3P1. Computación Paralela y Distribuída
  - line:264 CS402. Proyecto de tesis 1
  - line:269 FG106. Teatro
  - line:274 EX301. Actividades Extracurriculares
  - line:279 AI268. Visión Computacional
  - line:284 CS391. Ingeniería de Software III
  - line:289 CS393. Sistemas de Infomación
- Noveno Semestre
  - line:298 CS370. Big Data
  - line:303 CS400. Prácticas Pre-profesionales
  - line:308 CS403. Proyecto de tesis 2
  - line:313 AI365. Modelos Generativos Avanzados en IA
  - line:318 CB309. Bioinformática
  - line:323 CS351. Tópicos en Computación Gráfica
  - line:328 CS353. Computación Cuántica
  - line:333 AI369. Robótica
  - line:338 CS392. Tópicos en Ingeniería de Software
  - line:343 CS3P3. Internet de las Cosas
- Décimo Semestre
  - line:352 CS3P2. Cloud Computing

- line:357 CS404. Taller de Investigación
- line:362 AI367. Tópicos en Inteligencia Artificial
- line:367 AI368. Computación Evolutiva
- line:372 FG350. Liderazgo y Desempeño

# Capítulo 6

# Equivalencias con otros planes curriculares

A continuación se pueden observar las equivalencias de la presente malla con el(los) Plan(es) curricular(es) anterior(ES). Para ver mayores detalles de la malla propuesta observar la sección 3.2 Pág. 129.

## 6.1. Equivalencia del Plan IS2021 al Plan2026

| **Primer Semestre – Plan IS2021** | | | **Plan2026** | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-121 | Fundamentos de Programación | 4 | line:10 | line:10 Introducción a la Prog |
| IS-127 | Biología y Medio Ambiente | 3 | line:10 | line:10 Biología |
| IS-122 | Matemática I | 4 | line:10 | line:10 Cálculo Diferencial |

| **Segundo Semestre – Plan IS2021** | | | **Plan2026** | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-224 | Matemáticas Discretas I | 3 | line:25 | line:25 Estructuras Discretas |
| IS-226 | Estadística Básica | 3 | line:25 | line:25 Cálculo Diferencial e In do |
| IS-222 | Programación Orientada a Objetos I | 3 | line:25 | line:25 Programación Orientac |
| IS-225 | Matemática II | 4 | line:25 | line:25 Cálculo Integral |
| IS-223 | Algebra Lineal | 4 | line:25 | line:25 Algebra Linear |

| Tercer Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-324 | Matemática III | 3 | line:43 B | line:43 Cálculo Diferencial e Integ<br>do |
| | | | line:43 A | line:43 IA Aplicada |
| IS-325 | Matemáticas Discretas II | | | |
| IS-326 | Probabilidades | 3 | line:43 S | line:43 Estadística y Probabilidad |
| IS-322 | Programación Orientada a Objetos II | 3 | line:43 C | line:43 Programación Orientada a |
| IS-321 | Análisis y Diseño de Algoritmos | 3 | line:43 C | line:43 Análisis y Diseño de Algor |
| IS-323 | Fundamentos de Sistemas de Información | 3 | line:43 C | line:43 Sistemas de Infomación |

| Cuarto Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-424 | Sistemas Operativos | 3 | line:52 C | line:52 Sistemas Operativos |
| IS-423 | Base de Datos I | 3 | line:52 C | line:52 Bases de Datos I |

| Quinto Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-523 | Base de Datos II | 3 | line:67 C | line:67 Bases de Datos II |
| IS-521 | Sistemas Distribuidos | 3 | line:67 C | line:67 Computación Paralela y D |
| IS-527 | Investigación Operativa II | 3 | line:67 C | line:67 Estructuras de Datos Ava |
| IS-524 | Aplicaciones Web I | 3 | line:67 C | line:67 Programación Competitiva |
| IS-526 | Sistemas Digitales | 3 | line:67 B | line:67 Inglés V |

| Sexto Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-623 | Programación de Dispositivos Móviles I | 4 | line:80 C | line:80 Compiladores |
| IS-621 | Ingeniería de Software | 4 | line:80 C | line:80 Ingeniería de Software I |
| IS-624 | Aplicaciones Web II | 4 | line:80 A | line:80 Introducción al Aprendiza<br>ña |
| IS-626 | Arquitectura de Computadoras | 3 | line:80 F | line:80 Física Computacional |

| Séptimo Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-722 | Calidad de Software | 3 | line:91 | line:91 Ingeniería de Software |
| | | 4 | line:91 | line:91 Computación Gráfica |
| IS-725 | Redes I | 4 | line:91 | line:91 Redes y Comunicación |

| Octavo Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-823 | Proyecto de Investigacion I | 3 | line:106 | line:106 Proyecto de tesis 1 |
| IS-821 | Cloud Computing | 3 | line:106 | line:106 Cloud Computing |
| | | | line:106 | line:106 Teatro |
| IS-826 | Redes II | 4 | line:106 | line:106 Redes y Comunicació |
| IS-822 | Procesamiento de Imágenes y Video | 4 | line:106 | line:106 Visión Computacional |

| Noveno Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-926 | Robótica II | 3 | line:125 | line:125 Proyecto de tesis 2 |
| IS-922 | Seguridad Informática | 4 | line:125 | line:125 Big Data |
| IS-927 | Electivo I | 3 | line:125 | line:125 Robótica |
| IS-923 | Proyecto de Investigación II | 3 | line:125 | line:125 Internet de las Cosas |
| IS-921 | Inteligencia Artificial I | 3 | line:125 | line:125 Inteligencia Artificial |
| IS-924 | Formación de Empresas con Base Tecnológica | 3 | line:125 | line:125 Tópicos en Computac |
| IS-925 | Proyectos Informáticos I | 3 | line:125 | line:125 Tópicos en Ingeniería |

| Décimo Semestre – Plan IS2021 | | | Plan2026 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| IS-1023 | Seguridad de la Información | 4 | line:141 | line:141 Tópicos en Inteligenci |
| IS-1024 | Seminario de Tesis | 4 | line:141 | line:141 Liderazgo y Desempe |
| IS-1026 | Proyectos Informáticos II | | | |
| IS-1025 | Electivo II | 3 | line:141 | line:141 Proyecto de tesis 2 |
| IS-1021 | Inteligencia Artificial II | 3 | line:141 | line:141 Taller de Investigació |
| IS-1022 | Auditoría de Sistemas de Información | 4 | line:141 | line:141 Computación Evoluti |

## 6.2. Equivalencia del Plan Plan2026 al IS2021

| Primer Semestre – Plan2026 | | | Plan IS2021 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:10 CS line:10 | Introducción a la Programación | 4 | IS-121 | Fundamentos de Programación |
| line:10 BM line:10 | Algebra Linear | 4 | IS-223 | Algebra Lineal |
| line:10 BM line:10 | Cálculo Diferencial | 5 | IS-122 | Matemática I |
| **Segundo Semestre – Plan2026** | | | **Plan IS2021** | |
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:21 CS line:21 | Programación Orientada a Objetos I | 4 | IS-222 | Programación Orientada a Objeto |
| line:21 CS line:21 | Estructuras Discretas | 4 | IS-224 | Matemáticas Discretas I |
| line:21 BM line:21 | Cálculo Integral | 5 | IS-225 | Matemática II |
| **Tercer Semestre – Plan2026** | | | **Plan IS2021** | |
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:36 CS line:36 | Programación Orientada a Objetos II | 4 | IS-322 | Programación Orientada a Objeto |
| line:36 AI line:36 | IA Aplicada | 4 | | |
| line:36 BM line:36 | Cálculo Diferencial e Integral Avanzado | 5 | IS-324 | Matemática III |
| line:36 BM line:36 | Cálculo Diferencial e Integral Avanzado | 5 | IS-226 | Estadística Básica |
| line:36 ST line:36 | Estadística y Probabilidades | 3 | IS-326 | Probabilidades |
| **Cuarto Semestre – Plan2026** | | | **Plan IS2021** | |
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:43 CS line:43 | Bases de Datos I | 4 | IS-423 | Base de Datos I |
| **Quinto Semestre – Plan2026** | | | **Plan IS2021** | |
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:62 CS line:62 | Análisis y Diseño de Algoritmos | 4 | IS-321 | Análisis y Diseño de Algoritmos |
| line:62 CS line:62 | Inteligencia Artificial | 4 | IS-1021 | Inteligencia Artificial II |
| line:62 CS line:62 | Inteligencia Artificial | 4 | IS-921 | Inteligencia Artificial I |
| line:62 CS line:62 | Bases de Datos II | 3 | IS-523 | Base de Datos II |
| line:62 CS line:62 | Ingeniería de Software I | 4 | IS-621 | Ingeniería de Software |
| line:62 CS line:62 | Sistemas Operativos | 4 | IS-424 | Sistemas Operativos |
| line:62 BE line:62 | Inglés V | 2 | IS-526 | Sistemas Digitales |

| Sexto Semestre – Plan2026 | | | Plan IS20 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:81 CS2 | line:81 Redes y Comunicación | 3 | IS-826 | Redes II |
| line:81 CS2 | line:81 Redes y Comunicación | 3 | IS-725 | Redes I |
| line:81 CS2 | line:81 Programación Competitiva | 3 | IS-524 | Aplicaciones Web I |
| line:81 CS2 | line:81 Estructuras de Datos Avanzadas | 4 | IS-527 | Investigación Operativa II |
| line:81 CS2 | line:81 Compiladores | 4 | IS-623 | Programación de Dispositivos |
| line:81 AI2 | line:81 Introducción al Aprendizaje de Máquina | 4 | IS-624 | Aplicaciones Web II |
| line:81 FI2 | line:81 Física Computacional | 3 | IS-626 | Arquitectura de Computadoras |

| Séptimo Semestre – Plan2026 | | | Plan IS20 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:92 CS2 | line:92 Computación Gráfica | 4 | | |
| line:92 CS2 | line:92 Ingeniería de Software II | 4 | IS-722 | Calidad de Software |
| line:92 BB | line:92 Biología | 4 | IS-127 | Biología y Medio Ambiente |

| Octavo Semestre – Plan2026 | | | Plan IS20 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:113 CS | line:113 Computación en la Sociedad | 2 | | |
| line:113 CS | line:113 Seguridad en Computación | 3 | IS-1023 | Seguridad de la Información |
| line:113 CS | line:113 Computación Paralela y Distribuída | 4 | IS-521 | Sistemas Distribuidos |
| line:113 CS | line:113 Proyecto de tesis 1 | 3 | IS-823 | Proyecto de Investigacion I |
| line:113 FC | line:113 Teatro | 2 | | |
| line:113 AI | line:113 Visión Computacional | 3 | IS-822 | Procesamiento de Imágenes y |
| line:113 CS | line:113 Ingeniería de Software III | 3 | | |
| line:113 CS | line:113 Sistemas de Infomación | 3 | IS-323 | Fundamentos de Sistemas de I |

| Noveno Semestre – Plan2026 | | | Plan IS2021 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:132 CS line:132 | Big Data | 3 | IS-922 | Seguridad Informática |
| line:132 CS line:132 | Proyecto de tesis 2 | 3 | IS-1025 | Electivo II |
| line:132 CS line:132 | Proyecto de tesis 2 | 3 | IS-926 | Robótica II |
| line:132 CS line:132 | Tópicos en Computación Gráfica | 3 | IS-924 | Formación de Empresas con Base |
| line:132 AI line:132 | Robótica | 3 | IS-927 | Electivo I |
| line:132 CS line:132 | Tópicos en Ingeniería de Software | 3 | IS-925 | Proyectos Informáticos I |
| line:132 CS line:132 | Internet de las Cosas | 3 | IS-923 | Proyecto de Investigación II |

| Décimo Semestre – Plan2026 | | | Plan IS2021 | |
|---|---|---|---|---|
| **Código** | **Curso** | **Cr** | **Código** | **Curso** |
| line:147 CS line:147 | Cloud Computing | 3 | IS-821 | Cloud Computing |
| line:147 CS line:147 | Taller de Investigación | 3 | IS-1021 | Inteligencia Artificial II |
| line:147 AI line:147 | Tópicos en Inteligencia Artificial | 4 | IS-1023 | Seguridad de la Información |
| line:147 AI line:147 | Computación Evolutiva | 4 | IS-1022 | Auditoría de Sistemas de Informa |
| line:147 FG line:147 | Liderazgo y Desempeño | 2 | IS-1024 | Seminario de Tesis |

# Capítulo 7

# Laboratorios

## CS111. Introducción a la Programación (Obligatorio) 1er Sem, Lab: 4 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## BCH101. Química I (Obligatorio) 1er Sem, Lab: 2 hr(s)

**Laboratorio de Química**:

- **Equipos**: Balanzas analíticas, microscopios, espectrofotómetros, kits de vidriería.
- **Seguridad**: Campanas extractoras, duchas de emergencia, kits para derrames.
- **Reactivos**: Ácidos/bases, soluciones tampón, solventes orgánicos, indicadores de pH.

## CS112. Programación Orientada a Objetos I (Obligatorio) 2do Sem, Lab: 4 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## BFI101. Física I (Obligatorio) 2do Sem, Lab: 4 hr(s)

**Laboratorio de Física**:

- **Equipos**: Bancos ópticos, mesas de fuerzas, péndulos, carros dinámicos, espectrómetros.
- **Herramientas**: Sensores PASCO, registradores de datos, MATLAB para análisis.
- **Experimentos**: Mecánica, termodinámica, electromagnetismo.

## CS113. Programación Orientada a Objetos II (Obligatorio) 3er Sem, Lab: 2 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## CS2B1. Desarrollo Basado en Plataformas (Obligatorio) 3er Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## AI161. IA Aplicada (Obligatorio) 3er Sem, Lab: 4 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## CS210. Algoritmos y Estructuras de Datos (Obligatorio) 4to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS211. Teoría de la Computación (Obligatorio) 4to Sem, Lab: 2 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.

- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## CS221. Arquitectura de Computadores (Obligatorio) 4to Sem, Lab: 2 hr(s)

**Laboratorio de Redes y Ciberseguridad**:
- **Hardware**: 30 PCs + racks de red (switches/routers Cisco, clusters Raspberry Pi).
- **Sistemas Operativos**: Linux (Kali, CentOS) / Windows Server.
- **Software**: Wireshark, GNS3, VMware ESXi, emuladores 8086.
- **Enfoque**: Simulación de redes, pruebas de penetración, protocolos IoT.

## CS271. Bases de Datos I (Obligatorio) 4to Sem, Lab: 4 hr(s)

**Laboratorio Avanzado de Computación**:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS212. Análisis y Diseño de Algoritmos (Obligatorio) 5to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS261. Inteligencia Artificial (Obligatorio) 5to Sem, Lab: 2 hr(s)

**Laboratorio Básico de Computación**:
- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## CS272. Bases de Datos II (Obligatorio) 5to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).

- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS291. Ingeniería de Software I (Obligatorio) 5to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS2S1. Sistemas Operativos (Obligatorio) 5to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS231. Redes y Comunicación (Obligatorio) 6to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS311. Programación Competitiva (Obligatorio) 6to Sem, Lab: 2 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## CS312. Estructuras de Datos Avanzadas (Obligatorio) 6to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS342. Compiladores (Obligatorio) 6to Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## AI263. Introducción al Aprendizaje de Máquina (Obligatorio) 6to Sem, Lab: 4 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## FI201. Física Computacional (Obligatorio) 6to Sem, Lab: 2 hr(s)

**Laboratorio de Física**:

- **Equipos**: Bancos ópticos, mesas de fuerzas, péndulos, carros dinámicos, espectrómetros.
- **Herramientas**: Sensores PASCO, registradores de datos, MATLAB para análisis.
- **Experimentos**: Mecánica, termodinámica, electromagnetismo.

## CS251. Computación Gráfica (Obligatorio) 7mo Sem, Lab: 4 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS292. Ingeniería de Software II (Obligatorio) 7mo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS2H1. Experiencia de Usuario (UX) (Obligatorio) 7mo Sem, Lab: 4 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## AI264. Aprendizaje Profundo (Obligatorio) 7mo Sem, Lab: 4 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## BBI101. Biología (Obligatorio) 7mo Sem, Lab: 2 hr(s)

**Laboratorio de Biología/Bioinformática**:

- **Equipos**: Microscopios (compuestos/digitales), centrifugadoras, máquinas PCR.
- **Bioinformática**: 10 PCs con PyMol, BLAST, R/Bioconductor.
- **Muestras**: Portaobjetos preparados (microbiología, histología), kits de extracción de ADN.

## CS3I1. Seguridad en Computación (Obligatorio) 8vo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS3P1. Computación Paralela y Distribuída (Obligatorio) 8vo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS402. Proyecto de tesis 1 (Obligatorio) 8vo Sem, Lab: 4 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## AI268. Visión Computacional (Electivo) 8vo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS393. Sistemas de Infomación (Electivo) 8vo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS370. Big Data (Obligatorio) 9no Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS403. Proyecto de tesis 2 (Obligatorio) 9no Sem, Lab: 4 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## AI365. Modelos Generativos Avanzados en IA (Obligatorio) 9no Sem, Lab: 4 hr(s)

**Laboratorio Básico de Computación**:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

## CS351. Tópicos en Computación Gráfica (Electivo) 9no Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS353. Computación Cuántica (Electivo) 9no Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.

- *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
- *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## AI369. Robótica (Electivo) 9no Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS392. Tópicos en Ingeniería de Software (Electivo) 9no Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS3P3. Internet de las Cosas (Electivo) 9no Sem, Lab: 4 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS3P2. Cloud Computing (Obligatorio) 10mo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## CS404. Taller de Investigación (Obligatorio) 10mo Sem, Lab: 4 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## AI367. Tópicos en Inteligencia Artificial (Obligatorio) 10mo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

## AI368. Computación Evolutiva (Obligatorio) 10mo Sem, Lab: 2 hr(s)

**Laboratorio Avanzado de Computación**:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.