**ESPECIFICACIÓN GLOBAL DE UN CURSO BASADO EN GRANDES IDEAS**

**HOJA DE TRABAJO No.1 (HT1)**

**MODELOS Y PARADIGMAS DE PROGRAMACIÓN**

Fecha del (re)diseño: 2017 semestre 2 Versión: 1.5

# Equipo de diseño del curso

**Experto en contenidos**: Juan Francisco Díaz Frías, profesor titular Universidad del Valle

**Asesor en metodología Grandes Ideas**: Álvaro H Galvis Panqueva, profesor titular Uniandes

# Etapa 1 de diseño a nivel de curso [Etapa C1]: ¿Qué se deberá aprender en el curso?

## 1.1 Acerca del Curso a rediseñar

**Nombre del curso**: Modelos y Paradigmas de Programación

**Código del curso**: 750039

**Nivel del curso**: \_X\_ postgrado

**Tipo de curso**: \_\_X\_\_ obligatorio

**Nombre del programa académico que ofrece el curso**: Maestría en Ingeniería con énfasis en Ingeniería de Sistemas y Computación, y Doctorado en Ingeniería, énfasis en Ciencias de la Computación.

**Número de créditos que tendrá el curso**: 4 créditos = 192 horas de trabajo académico por período

**Duración del período académico**: 16 semanas lectivas

**Intensidad horaria semanal**: 12 horas, a lo largo de 16 semanas

**Modalidad de enseñanza**: virtual

**Descripción del curso en el programa / Visión comprensiva del curso luego del rediseño:**

La programación es cada día un asunto de mayor abstracción y complejidad. Los avances en hardware e infraestructuras hacen que el desarrollo de lenguajes de programación que logren aprovechar esos avances sea un tema de investigación permanente. Lograr comprender los conceptos fundamentales subyacentes a los diferentes lenguajes de programación y cómo ellos dan origen a paradigmas de programación es una habilidad necesaria para los ingenieros de sistemas de hoy. Este curso explora precisamente diferentes conceptos y paradigmas de programación sobre un ambiente de programación multiparadigma especialmente adecuado para esta exploración.

**Elementos misionales o modelo educativo institucional que incide(n) en el curso**

Interesa dar flexibilidad a los estudiantes de posgrado para superar barreras espaciotemporales que puedan tener para participar en procesos de formación avanzada. Esto permite ampliar la cobertura.

**Cursos pre-requeridos:** ninguno

**Conceptos, habilidades o destrezas que se suponen dominados antes de tomar el curso**:

|  |  |
| --- | --- |
| Conceptos | Saber qué significa programar, correr un programa, ejecutarlo |
| Habilidades | Saber programar en algún lenguaje de programación |
| Destrezas | Analizar problemas y proponer maneras de solucionarlos |

**Curso(s) a los que sirve:** Cursos de programación del Doctorado en Ingeniería con énfasis en Ciencias de la Computación (DICC) y de la Maestría en Ingeniería con énfasis en Ingeniería de Sistemas y Computación (MIISC) de la Escuela de Ingeniería de Sistemas y Computación (EISC)

**Conceptos, habilidades o destrezas que desarrolla para curso(s) a los que sirve**:

|  |  |
| --- | --- |
| Conceptos | Los siete conceptos fundamentales de MPP sirven de base para los cursos de programación del DICC y de la MIISC |
| Habilidades | Análisis de lenguajes y programación haciendo uso de múltiples paradigmas |
| Destrezas | Programación haciendo uso de múltiples paradigmas usando el lenguaje MOzArt |

**Razones / motivadores para (re)diseñar el curso**:

Por ser MPP un curso obligatorio fundamental tanto del programa DICC como del programa MIISC y por la intención actual de la Universidad, la Facultad y la Escuela de ofrecer posgrados en las sedes de la Universidad del Valle, el estructurar uno de los cursos fundamentales de nuestros posgrados de forma virtual, es un primer paso para lograr ese propósito de extensión del posgrado más allá de la sede de Meléndez. Incluso, más allá del Valle del Cauca y más allá de Colombia, en países como Ecuador desde donde recibimos una constante demanda. El curso MPP en modalidad virtual garantiza coherencia de las versiones del curso donde se ofrezca, facilita el control y el desarrollo del curso, brinda acceso amplio para todos los estudiantes, asegura cubrimiento completo del tema, con dedicación presencial o asistida por parte de los profesores al aprendizaje de los estudiantes más que a la enseñanza como tal, entre otros.

## 1.2 Acerca de los destinatarios del curso

**Descripción**: Estudiantes-adultos de MIISC o de DICC distribuidos geográficamente, con acceso a tecnología digital y al menos alguna experiencia en programación

**Rangos de edades**: \_\_X\_\_ 20 a 25 años, \_\_X\_\_ 26 o más años

**Ocupación principal**: \_\_X\_ estudiar y trabajar

**Distribución geográfica**: región occidental del país, así como de Ecuador, Perú y centro américa

**Acceso a tecnologías digitales**: \_\_X\_ en la institución, \_\_X\_ en la casa, \_\_X\_ móvil

**Razones por las que tomarían el curso**: es obligatorio llevarlo como parte de MIISC o DICC

**Facilitadores / inhibidores de la participación**:

Los destinatarios del curso suelen ser, en maestría, personas que trabajan y, por tanto, pierden mucho tiempo en desplazamientos de su trabajo a las aulas presenciales, o muchas veces no pueden asistir por situaciones propias de sus trabajos. Tener el curso de manera virtual les permite resolver esos inconvenientes y no perder un tema o una clase o no perder tiempo por esas razones.

## 1.3 Necesidades que inciden en el diseño de este curso

HT1 - Tabla 01. Síntesis de necesidades que inciden en el curso que se rediseña

|  |  |
| --- | --- |
| **NECESIDADES EDUCATIVAS NORMATIVAS** | **NECESIDADES EDUCATIVAS SENTIDAS** |
| MPP apoya el logro de los siguientes propósitos del programa de Maestría en Ingeniería (MI), énfasis en Ingeniería de Sistemas y Computación (ISC):   * Formar científicos en ISC con una sólida capacidad de critica que permita la asimilación y transferencia de conocimiento necesario tanto para la posterior confrontación con pares nacionales como para la aplicación de ese conocimiento en la solución de problemas en áreas de interés para el desarrollo del país. * Formar docentes en ISC que sean capaces de contribuir a una docencia de excelencia en el área de Programación, a nivel de la educación superior. * Formar profesionales en ISC actualizados y capaces de actuar de manera productiva para satisfacer las necesidades económicas, culturales y sociales del país. * Formar profesionales que contribuyan a la creación y fortalecimiento de una industria de software nacional, así́ como a fortalecer la capacidad de definición de estrategias informáticas que impacten el desarrollo del país y de su sector productivo. | La aparición frecuente de nuevos lenguajes de programación (C, C++, Java, Python, javascript, Scala, Rubby, Erlang, ...) hace necesario tener la habilidad para apropiarlos rápidamente. Lograr estudiarlos en un contexto genérico que permita rápidamente formarse una opinión calificada sobre las ventajas y desventajas de un nuevo lenguaje es una necesidad de la mayoría de empresas y organizaciones en el mundo de hoy.  Es un curso muy bien visto por los estudiantes, y se pretende lograr ofrecerlo en las sedes sin necesidad de profesor presencial en ellas. Obviamente, se necesitará un número de profesores virtuales consecuente con el número de estudiantes inscritos cada que se abra. |
| **Otras necesidades educativas**  MPP apoya el logro de los siguientes propósitos de los egresados del programa de DICC--Doctorado en Ingeniería, énfasis en Ciencias de la Computación:   * Aplicar el método científico en su razonamiento e investigación a través de una actitud crítica frente a la apropiación del conocimiento, específicamente en lo referente a los lenguajes de programación. * Apropiar el estado del arte en el área de programación. * Mantener una postura crítica, desde la perspectiva de quien domina el por qué́ del conocimiento, frente a la apropiación de tecnología de pares pertenecientes a países con mayor nivel de desarrollo, específicamente en lo concerniente a tecnologías de programación. | |

## 1.4 Grandes ideas (GI) y entendimientos perdurables (EP) que focalizan el curso

HT1 - Tabla 02. Enunciado de grandes ideas y sus correspondientes entendimientos perdurables

| **GI** | **Enunciado de la GI** | **EP** | **Enunciado de los EP / GI** |
| --- | --- | --- | --- |
| **1** | Los lenguajes de programación se comprenden a través de los conceptos nucleares que ellos implementan. Estos conceptos definen los paradigmas de programación soportados. | 1.1 | Entender un Lenguaje de Programación, es entender los conceptos nucleares que él implementa. |
| 1.2 | La combinación coherente de conceptos de programación da lugar a Paradigmas de Programación. |
| **2** | El modelo de programación declarativa y sus conceptos básicos: lenguaje núcleo, Variables declarativas, Procedimientos y Registros como valores básicos, y técnicas de programación declarativa. | 2.1 | Una ​variable declarativa​ es un espacio de memoria donde se puede guardar un valor de cualquier tipo, que puede ser parcial, pero una vez está completo, no puede ser cambiado. Se llaman también variables de asignación única​. Son variables, en tanto pueden tomar cualquier valor durante una computación, sólo que una vez lo toman, ya no se puede cambiar. |
| 2.2 | Un ​procedimiento​ es un ​valor, ​así como lo son los enteros o los átomos. Como tal, se puede pasar como argumento a otro procedimiento, se puede devolver como resultado de un procedimiento y se puede almacenar en una variable declarativa. ​​Un registro ​es otro ​valor ​que permite modelar las estructuras arbitrariamente grandes como listas, colas, árboles, grafos, entre otras. Si se proveen en su máximo poder de expresión, permiten trabajar simbólicamente. |
| 2.3 | Los ​lenguajes de programación​, por más complejos y extensos que sean, siempre tienen un conjunto de instrucciones o ​declaraciones​, digamos ​nucleares​, sobre las cuales se construyen todas las demás abstracciones que provean. A ese conjunto de declaraciones se le conoce como el ​lenguaje núcleo. |
| 2.4 | La​ **iteración como mecanismo de solución de problemas**​ es un concepto independiente de la sintaxis que provee el lenguaje de programación para implementarlo. Es importante entonces saber cómo implementarlo en el modelo declarativo. Uno de los errores más comunes en los programadores es confundir el concepto de iteración con la sintaxis que provee el lenguaje de programación para implementarlo, en general, el while o el for |
|  | 2.5 | Complementario a la iteración, aparece​ **la recursión como mecanismo de solución de problemas**​, en el cual la solución a un problema (grande) se calcula mezclando las soluciones a problemas del mismo tipo pero más pequeños. |
|  | 2.6 | **La abstracción procedimental**​ es el mecanismo que permite encapsular un proceso dentro de un procedimiento, para ser ejecutado cuando se requiera y no al momento de la definición. |
|  | 2.7 | La ​**genericidad**​ es el mecanismo por el cual un procedimiento, en tanto que valor, puede ser pasado como argumento a otro procedimiento. |
|  | 2.8 | La ​**instanciación**​ es el mecanismo por el cual los procedimientos, en tanto que valores, pueden ser devueltos como resultado de un cálculo. |
|  | 2.9 | El ​**embebimiento**​ es el mecanismo por el cual los procedimientos, en tanto que valores, pueden ser almacenados en una variable declarativa. |
| **3** | El modelo de programación concurrente declarativa y sus conceptos básicos: Hilos, Flujos, Disparadores por necesidad, técnicas de programación dirigida por los datos, y dirigida por la demanda | 3.1 | Para lograr ejecuciones concurrentes, se agrega al modelo declarativo el ​**concepto de hilo**​, como el espacio computacional donde se ejecuta un programa. Ahora un programa puede tener varios hilos. Esto extiende el modelo de programación declarativa, el cual es secuencial: dos declaraciones independientes deben ser ejecutadas en algún orden preestablecido. |
| 3.2 | Un ​**flujo**​ es una lista cuya cola es una variable declarativa sin ligar. A esa lista se le llama un ​**flujo** porque por ella pueden “fluir” los datos. Gracias a los hilos, esa cola puede ser asignada por un hilo diferente a donde se declaró, logrando la comunicación entre hilos. ​**Programar con flujos**​ es propio de la programación concurrente declarativa. |
|  | 3.3 | La ​**limitación**​ más importante del modelo concurrente declarativo, es la ​**imposibilidad de modelar la comunicación cliente-servidor**​. |
|  | 3.4 | Un ​**disparador por necesidad**, ​es un concepto de programación que ​**asocia una variable declarativa a un hilo específico**​, el cual sólo se ejecutará si la variable se necesita. Sólo entonces, el hilo le calculará un valor a esa variable. |
|  | 3.5 | Los disparadores por necesidad permiten implementar un mecanismo de evaluación, denominado ​**evaluación perezosa**​, donde solamente se calcula el valor de una variable, si éste es utilizado en alguna parte del programa. Esto permite implementar, de forma finita. ​**estructuras de datos potencialmente infinitas**​. Este mecanismo contrasta con el mecanismo por defecto de los lenguajes de programación denominado evaluación ansiosa, es decir, todo se evalúa inmediatamente se procesa, aún si el resultado del proceso no lo utiliza ningún programa posteriormente. |
| **4** | El modelo de programación con estado y sus conceptos básicos: celdas y estado explícito. | 4.1 | Una de las limitaciones de la programación declarativa es la imposibilidad de guardar memoria de lo que pasa en una computación, para ser utilizada posteriormente. Esto debido a que las variables declarativas son de asignación única. Para resolver esa limitación se crean las ​celdas​, que funcionan análogamente a las variables de los lenguajes de programación convencionales, y se pueden asignar múltiples veces. |
|  | 4.2 | Con las celdas, es posible guardar el estado de una computación para ser usado posteriormente. A lo que se guarda se le conoce como el ​estado explícito​. |
|  | 4.3 | ¿Cuáles son las ventajas y desventajas del modelo de ​Programación declarativa vs el modelo con estado? La transparencia referencial es la propiedad más importante de la programación declarativa pero se pierde en el modelo con estado. Sin embargo, en éste se hace natural la modularidad y la reutilización, más difícil de lograr en el modelo declarativo. |
| **5** | El modelo de programación concurrente por paso de mensajes y sus conceptos básicos: puertos, programación de sistemas multiagentes. | 5.1 | La limitación principal de la programación concurrente declarativa es la imposibilidad de modelar una relación cliente servidor, muchos a uno. Con el ​concepto de puerto​ esta limitación se elimina y la comunicación muchos-a-uno es ahora posible. |
|  | 5.2 | Los puertos se utilizan para comunicar diversos procesos, cada uno con su propia programación. A estos procesos se les denomina ​objetos puerto​. |
|  | 5.3 | Con la posibilidad de construir objetos puerto, se pueden diseñar programas basados en la analogía de agentes inteligentes que actúan independientemente y comunicándose entre ellos. Para hacer bien esta programación, es importante dominar una metodología para la ​Programación de sistemas multiagentes​. |
| **6** | El modelo de programación orientada a objetos y sus conceptos básicos: objetos, clases y herencia | 6.1 | Un ​objeto​ se puede modelar como un procedimiento con estado encapsulado. No se necesitan conceptos de programación diferentes a los encontrados en el modelo con estado. |
|  | 6.2 | Una ​clase​ no es más que un procedimiento que genera instancias de objetos cada vez que se solicita un objeto nuevo. No se necesitan conceptos de programación diferentes a los encontrados en el modelo con estado. |
|  | 6.3 | La construcción de clases a partir de clases existentes es el valor agregado del modelo orientado objeto y se implementa con la ​herencia​. No se necesitan conceptos de programación diferentes a los encontrados en el modelo con estado para implementarla. |
|  | 6.4 | Cuando se combinan el modelo orientado a objetos con el modelo de concurrencia por paso de mensajes, se crean los ​objetos activos​. |
| **7** | El modelo de programación relacional y sus conceptos básicos: escogencia no determinística, espacio de computación | 7.1 | Para poder modelar el concepto de relación, se hace necesario que el modelo de computación pueda “escoger no determinísticamente” por dónde continuar una ejecución. Para incluir esta posibilidad en el modelo declarativo se agrega al modelo declarativo el ​concepto de escogencia no determinística. |
|  | 7.2 | Puesto que las máquinas reales no son “no determinísticas” se hace necesario simular este mecanismo en la máquina real. Para ello se agrega al modelo ​el concepto de espacio de computación​. |
|  | 7.3 | Con la posibilidad de escribir programas con escogencia no-determinística y espacios de computación, se pueden escribir programas que calculan relaciones. A esto se le denomina Programación relacional​. |

# Etapa 2 de diseño a nivel de curso [Etapa C2]: ¿Cómo demostrar que se aprende lo deseado en el curso?

## 2.1 Definiciones macro para el sistema de evaluación de los aprendizajes

HT1 - Tabla 03. Definiciones a nivel de curso del sistema de evaluación de aprendizajes y enfoque educativo

| **GI** | **EP** | **RO** | **Resultado observable (RO) / EP / GI**   * Conducta observable * Condiciones de ejecución * Criterios de aceptación | **Clasificación según Bloom**  CO CM AP  AN EV CR | **Valoración e información de retorno** | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pre/conceptos y conceptos errados** | **Evaluación formativa** | **Evaluación sumativa** |
| **1** | 1.1 | 1.1 | Comprender cuáles son los conceptos nucleares de programación, qué significan y cómo usarlos para analizar un lenguaje de programación. | **CO CM** AP  AN EV CR |  | Escoger un lenguaje de programación al principio del curso y solicitar un análisis al final del curso desde el punto de vista de los conceptos y los paradigmas que ese lenguaje provee. |  |
| 1.2 | 1.2 | Ser capaz de asociar conceptos nucleares de programación con los principales paradigmas de programación. | **CO CM**  AP  AN EV CR |  | 6 talleres,  GI 2 a 7 | 1 Evaluación final virtual indagando por apropiación de los conceptos tanto de los paradigmas fundamentales como de los paradigmas avanzados.  Habrá preguntas sobre todos los conceptos del curso |
| **2** | 2.1 | 2.1 | Ser capaz de definir variables declarativas, de operar con ellas y de entender cuándo se usan mal (y producen un error) | **CO CM** AP  AN EV CR |  | 1 Taller de  Programación declarativa.  Incluye todos  estos temas | 1 Quiz, conceptual, no de programación.  Las competencias de programación se evidencian en los talleres |
| 2.2 | 2.2 | Ser capaz de definir procedimientos con nombre y anónimos y de operar con ellos tanto en abstracto (papel) como en concreto (lenguaje de programación) y de entender cuándo se usan mal (y producen errores)  Ser capaz de modelar con registros estructuras de datos como listas y árboles, y de utilizar el poder de los átomos para trabajar simbólicamente. | **CO CM**  AP  AN EV CR |  |
| 2.3 | 2.3.1 | Ser capaz de usar el lenguaje núcleo de Oz, tanto sintáctica como semánticamente. | CO CM **AP**  AN EV CR |  |
| 2.3.2 | Ser capaz de describir la máquina abstracta de Oz. | **CO CM** AP  AN EV CR |  |
| 2.3.3 | Entender cuando hay errores sintácticos. | **CO CM** AP  AN EV CR |  |
| 2.3.4 | Entender la semántica de cada una de las declaraciones del lenguaje núcleo: skip, secuenciación, declaración con variables locales, asignación de valores a variables, asignación de variables a variables, condicional, invocación de procedimiento, reconocimiento de patrones. | **CO CM** AP  AN EV CR |  |
| 2.4 | 2.4.1 | Ser capaz de plantear soluciones iterativas a problemas. | CO **CM AP**  AN EV CR |  |
| 2.4.2 | Ser capaz de distinguir cuándo una solución es iterativa y cuándo es recursiva. | **CO CM** AP  AN EV CR |  |
| 2.5 | 2.5.1 | Ser capaz de plantear soluciones recursivas a problemas. | CO **CM AP**  AN EV CR |  |
| 2.5.2 | Ser capaz de distinguir cuándo una solución es iterativa y cuándo es recursiva. | **CO CM** AP  AN EV CR |  |
| 2.6 | 2.6 | Ser capaz de definir y operar abstracciones procedimentales. | CO **CM AP**  AN EV CR |  |
| 2.7 | 2.7 | Ser capaz de definir y operar procedimientos genéricos. | CO **CM AP**  AN EV CR |  |
| 2.8 | 2.8 | Ser capaz de definir y operar procedimientos que devuelven procedimientos. | CO **CM AP**  AN EV CR |  |
| 2.9 | 2.9 | Ser capaz de definir y operar estructuras de datos que albergan procedimientos. | CO **CM AP**  AN EV CR |  |
| **3** | 3.1 | 3.1.1 | Ser capaz de definir y operar con hilos programas concurrentes declarativos. | CO **CM AP**  AN EV CR |  | 1  Taller de  Programación  concurrente  declarativa.  Incluye todos  estos temas | 1 Quiz, conceptual, no de programación.  Las competencias de programación se evidencian en los talleres |
| 3.1.2 | Entender y usar las variables declarativas como flujos de datos. | **CO CM** AP  AN EV CR |  |
| 3.2 | 3.2.1 | Entender y usar el concepto de flujo. | **CO CM** AP  AN EV CR |  |
| 3.2.2 | Ser capaz de escribir programas en esquema productor-consumidor | CO CM  **AP**  AN EV CR |  |
| 3.3 | 3.3 | Entender la imposibilidad de modelar un proceso cliente-servidor con concurrencia declarativa | **CO CM** AP  AN EV CR |  |
| 3.4 | 3.4 | Ser capaz de definir y operar con disparadores por necesidad. | CO **CM AP**  AN EV CR |  |
| 3.5 | 3.5.1 | Comprender el modelo de evaluación perezosa y diferenciarlo de la evaluación ansiosa. | **CO CM** AP  AN EV CR |  |
| 3.5.2 | Ser capaz de definir y operar estructuras de datos de tamaño potencialmente infinito. | CO **CM AP**  AN EV CR |  |
| **4** | 4.1 | 4.1 | Ser capaz de definir y operar con celdas. | CO **CM AP**  AN EV CR |  | 1 Taller de  programación con estado | 1 Quiz, conceptual, no de programación.  Las competencias de programación se evidencian en los talleres |
| 4.2 | 4.2 | Ser capaz de definir y operar procedimientos con estado explicito. | CO **CM AP**  AN EV CR |  |
| 4.3 | 4.3 | Entender las limitaciones de la programación declarativa para modelar procesos donde la historia es relevante. | **CO CM** AP  AN EV CR |  |
| **5** | 5.1 | 5.1.1 | Ser capaz de definir y operar procedimientos con puertos | CO **CM AP**  AN EV CR |  | 1  Taller de  programación por paso  de mensajes | 1 Quiz, conceptual, no de programación.  Las competencias de programación se evidencian en los talleres |
| 5.1.2 | Ser capaz de simular celdas con puertos y puertos con celdas | CO CM **AP**  AN EV CR |  |
| 5.2 | 5.2 | Ser capaz de definir y operar procedimientos con objetos puerto | CO **CM AP**  AN EV CR |  |
| 5.3 | 5.3.1 | Conocer la metodología para programar sistemas multiagentes | **CO CM** AP  AN EV CR |  |
| 5.3.2 | Ser capaz de resolver un problema modelándolo y programándolo con multiagentes. | CO CM AP  **AN EV CR** |  |
| **6** | 6.1 | 6.1.1 | Entender que los objetos se pueden implementar con el concepto de estado y el concepto de procedimiento como valor. | **CO CM** AP  AN EV CR |  | 1  Taller de  programación OO | 1 Quiz, conceptual, no de programación.  Las competencias de programación se evidencian en los talleres |
|  | 6.1.2 | Ser capaz de implementar un objeto como un registro con procedimientos embebidos compartiendo estado explicito. | CO CM **AP**  AN EV CR |  |
| 6.2 | 6.2.1 | Ser capaz de definir y operar con clases. | **CO CM** AP  AN EV CR |  |
| 6.2.2 | Entender los conceptos de miembros de un objeto (atributos y métodos) y saber operar con ellos. | CO **CM AP**  AN EV CR |  |
| 6.3 | 6.3.1 | Entender el concepto de herencia. | **CO CM** AP  AN EV CR |  |
| 6.3.2 | Entender la necesidad del self y su implicación en la invocación de métodos | **CO CM** AP  AN EV CR |  |
| 6.3.3 | Ser capaz de definir y operar clases y objetos con herencia | CO **CM AP**  AN EV CR |  |
| 6.4 | 6.4 | Ser capaz de definir y operar objetos activos. | **CO CM** AP  AN EV CR |  |
| **7** | 7.1 | 7.1.1 | Entender el concepto de escogencia determinística | **CO**  CM AP  AN EV CR |  | 1  Taller de  programación relacional | 1 Quiz, conceptual, no de programación.  Las competencias de programación se evidencian en los talleres |
| 7.1.2 | Ser capaz de definir y operar relaciones | **CO CM** AP  AN EV CR |  |
| 7.2 | 7.2 | Ser capaz de definir y operar espacios de computación | **CO CM** AP  AN EV CR |  |
| 7.3 | 7.3 | Ser capaz de resolver problemas en el paradigma relacional | CO CM AP  **AN EV CR** |  |

## 2.2 Valor relativo de calificaciones y de tiempos para estudiar cada gran idea

HT1 - Tabla 04. Distribución de valores relativos del tiempo de estudio y evaluación por gran idea

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Valores relativos** | **Gran idea 1** | **Gran idea 2** | **Gran idea 3** | **Gran idea 4** | **Gran idea 5** | **Gran idea 6** | **Gran idea 7** | **Referencia o Evidencia** |
| Peso en tiempo de estudio | 18% | 17% | 13% | 13% | 13% | 13% | 13% | 100% de tiempo |
| Peso quizzes |  | 4% | 2% | 2% | 2% | 2% | 2% | 14% nota final |
| Peso talleres | 15% | 6% | 6% | 6% | 6% | 6% | 6% | 51% nota final |
| Peso evaluación final de conocimientos | 35% | | | | | | | 35% nota final |

# Etapa 3 de diseño a nivel de curso [Etapa C3]: ¿Cómo propiciar que se aprenda lo deseado en el curso?

## 3.1 Combinación de espacios y tiempos según modalidad de enseñanza

HT1 - Tabla 05. Escenarios mezcla de sesiones en AF y AV para curso según % de presencialidad deseado

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Duración sesión acompañada** | **Número sesiones / semana en Aula Física** | **Número sesiones / semana en Aula Virtual** | **Número de horas trabajo autónomo** | **Horas trabajo académico / semana** |
| 1 hora / sesión | 1 | 7 | 4 | 12 |

Nota:

Hay una sesión presencial cada dos semanas. No es clase como tal. Es un encuentro presencial para resolver dudas en la primera hora y hacer un quiz en la segunda hora. A veces este encuentro es sincrónico virtual y no presencial.

## 3.2 Recursos por entornos de aprendizaje

HT1 - Tabla 06. Recursos para promover aprendizaje usando ambientes híbridos en entornos formales o informales

| **Taxonomía de medios de Forté** (1998) | **Entorno formal de aprendizaje** (institución educativa) | | **Entorno autónomo de aprendizaje**  (hogar, trabajo, comunidad de práctica) | |
| --- | --- | --- | --- | --- |
| **Aula física o taller** | **Aula virtual** | **Entorno virtual** | **Entorno físico** |
| Interacción sincrónica | Interacción sincrónica o asincrónica | Interacción sincrónica o asincrónica | Interacción sincrónica |
| **Medios interactivos** | Foros virtuales por módulo |  | Evaluación de 6 talleres de compañeros (coevaluación) |  |
| **Medios activos** | Quizzes en línea, al final de las sesiones sincrónicas | Talleres de programación y de análisis de LP |  |  |
| **Medios expositivos** |  | Grabaciones en video con aclaraciones | Estudio con el libro de texto del curso |  |

## 3.3 Selección de estrategias pedagógicas para el cuso en (re)diseño

Hay una hipótesis básica: El dominio de los conceptos fundamentales de programación se obtiene entendiendo los conceptos y luego usándolos. Por ello, el curso usa como estrategia pedagógica de entendimiento de los conceptos la lectura del texto, el seguimiento de los videos y el desarrollo de quices conceptuales. Y como estrategia pedagógica para el uso de los conceptos, los talleres, 6 de programación y uno de análisis de un lenguaje de programación. Los talleres se coevalúan entre los estudiantes y el profesor, lo que permite a los estudiantes aprender de lo que hacen sus compañeros o sugerir a sus compañeros cómo mejorar sus productos. La coevaluación es entonces una estrategia pedagógica y de evaluación actual.

HT1 - Tabla 07. Selección de estrategias pedagógicas que tendría sentido usar en el curso rediseñado

|  |  |  |  |
| --- | --- | --- | --- |
| **Conceptos pedagógicos** | | **Técnicas didácticas** | |
| X | Aprendizaje activo |  | Aprendizaje adaptativo |
| X | Aprendizaje basado en retos |  | Aprendizaje basado en casos |
|  | Aprendizaje conectivista |  | Aprendizaje basado en equipos |
|  | Aprendizaje experiencial | X | Aprendizaje basado en problemas |
|  | Aprendizaje por indagación | X | Aprendizaje colaborativo |
|  | Aprendizaje / cognición situad@ |  | Aprendizaje cooperativo |
|  | Aprendizaje social |  | Aprendizaje orientado a proyectos |
|  | Comunidades de práctica | X | Aprendizaje invertido / Aula invertida |
|  |  |  | Gamificación |
|  |  |  | Storytelling |

## 3.4 Definición del modelo operativo para el curso

HT1 - Tabla 08. Modelo operativo expresado en horas de estudio por actividad por día por semana por tipo de entorno de aprendizaje

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tipo de Actividad** | **Entorno de aprendizaje** | | **Semana i-1** (x días, y horas/día) | | | | **Semana i** (x días, y horas/día) | | | | **Semana i+1** (x días, y horas/día) | | | |
| Sincrónica | Encuentro sincrónico resolviendo dudas del módulo en curso, del taller a entregar al final de la semana. | | 1 día, 1 hora ese día | | | | 0 | | | | 1 día, 1 hora ese día | | | |
| Asincrónica | Revisión de los videos de la semana y lecturas del libro texto | | 5 horas/semana | | | | 5 horas/semana | | | | 5 horas/semana | | | |
| Asincrónica | Desarrollo de un taller virtual (seis de programación, uno de análisis de un LP) | | 4 horas / semana | | | | 4 horas / semana | | | | 4 horas / semana | | | |
| Asincrónica | Evaluación de talleres de compañeros | |  | | | | 2 horas/semana | | | |  | | | |
| Sincrónica/  Asincrónica | Presentación de un quiz del módulo en curso. | | 1 hora/semana | | | |  | | | | 1 hora/semana | | | |
| Asincrónica | Foro Virtual | | Participación en foro virtual del módulo en curso.  No es obligatorio, es una herramienta más. | | | | Participación en foro virtual del módulo en curso.  No es obligatorio, es una herramienta más. | | | | Participación en foro virtual del módulo en curso.  No es obligatorio, es una herramienta más. | | | |
|  | |  | |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |

## 3.5 Secuencia de instrucción

El flujo de instrucción va de izquierda a derecha en el diagrama de bloques siguiente

3.6 Oportunidades para hacer reutilización de recursos producidos por otros

HT1 - Tabla 09. Oportunidades en recursos educativos relacionados con el curso que se (re)diseña

| **Nombre del Curso o recurso y su URL** | **Institución o grupo** | **Semejanzas y diferencias** | **Oportunidades** |
| --- | --- | --- | --- |
| Paradigms of Computer Programming – Fundamentals | LouvainX Cursos online gratuitos de Université catholique de Louvain | Cubre los paradigmas declarativo, concurrente declarativo y paso de mensajes esencialmente. Bastante menos que el curso.  Para la calificación automá-tica de los programas usa una aplicación: INGenious  No usa la evaluación por pares.  Usamos el mismo texto base y la misma metodología para comprender los paradigmas. | Buen material complementario al curso.  Está hecho para masificarse.  Está en una plataforma masiva: EdX |

## 3.7 Materiales existentes que calzan con el diseño macro del curso

HT1 - *Tabla 10. Materiales o recursos aplicables a parte o todo el curso que se rediseña*

| **Tipo de recurso** | **Referencia al recurso– Use formato APA** | **Útil en GI.EP** | **Uso deseable** |
| --- | --- | --- | --- |
| Expositivos:  Recursos que trasmiten a sus usuarios conocimientos en formato textual, visual, auditivo o multimedia | | | Obligatorio  Opcional  Profundización  Interesante |
| Libro | Concepts, techniques, and models of computer programming. Van Roy, Peter and Haridi, Seif (2004). The MIT Press USA ISBN: 978-0-262-22069-9.. Versión en español disponible | Todos | Obligatorio |
| Videoclips | Para cada uno de los conceptos se dispone de video grabado por el profesor donde se explica o ejemplifica el concepto respectivo | Todos | Obligatorio |
| Activos:  Recursos que tienen comportamiento orgánico modelado por el autor o por el usuario. Al ser estimulados por éste, ejecutan lo pedido y permiten aprender de los efectos. | | | Obligatorio  Opcional  Profundización  Interesante |
| Lenguaje de programación | MozArt http://mozart.github.io/ | Todos | Obligatorio para talleres |
| Interactivos:  Recursos que hacen posible diálogo entre humanos, mediados por tecnología y en forma asincrónica o sincrónica. Se aprende al interactuar digitalmente con otros. | | | Obligatorio  Opcional  Profundización  Interesante |
| Foro  Wiki  Blog  Red social | Moodle |  |  |
| Chat  eMail, VoiceIP  WA | Video conferencias en ZOOM |  |  |

## 3.8 Tecnologías digitales para el aprendizaje (TDA) según diseño macro

### 3.8.1 Requerimientos de inclusión o de puesta a punto de TDA para el curso

Acceso efectivo a tecnologías digitales con buen ancho de banda

### 3.8.2 Requerimientos tecnológicos a partir del diseño macro

Aula virtual del curso MPP para cada cohorte, N secciones