METODOLOGÍA DE DESARROLLO DE SOFTWARE PARA ESTUDIANTES EN ETAPAS INICIALES DE EDUCACIÓN UNIVERSITARIA

María E. Tirado Pérez.
mariatiperez@gmail.com
Jesús Lárez Mata
jjlarez@gmail.com
Universidad Católica Andrés Bello Extensión Guayana
Escuela de Ingeniería Informática

Fecha de recepción: 15 julio 2018
Fecha de aceptación: 30 agosto 2018

Resumen

A continuación se presenta la elaboración de una propuesta metodológica para desarrollar software con innovación, que permita a los estudiantes que se encuentran en etapas iníciales de educación universitaria adquirir competencias de un buen profesional de software. La metodología está sustentada sobre conceptos en el área de formación académica e ingeniería del software, los cuales fueron integrados haciendo uso de un proceso de análisis, selección y delimitación de sus herramientas y procesos. Además, está fundamentada en el Paradigma Pedagógico Ignaciano, el modelo de desarrollo en V y bajo los principios de programación ágil, en ella se definen una serie de principios, etapas, actividades y productos que conforman el proceso. Para la validación de la metodología se realizó un curso de programación con una muestra de estudiantes de Ingeniería Informática e Industrial (quienes habían cursado o estaban por cursar materias con contenido básico de programación) y se solicitó la validación por parte de expertos en desarrollo de software y en la propuesta pedagógica ignaciana, quienes revisaron y verificaron el uso correcto de conceptos en ambos ámbitos.

Palabras Clave: Desarrollo de software, formación educativa, modelo en V, paradigma pedagógico ignaciano, meta metodología.

Abstract

Below is presented the elaboration of a methodology for developing innovative software, which allows the students who are in the initial stages of university education to acquire the skills of a good software professional. The methodology is based on concepts in the area of education and software engineering, which were integrated using a process of analysis, selection and delimitation of its tools and processes. In addition, it is based on the Ignatian Pedagogical Paradigm, the development model in V and the principles of agile programming, if defines a series of principles, stages, activities and products that make up the process. For the validation of the methodology, a programming course was conducted with a sample of Computer and Industrial Engineering students (who had taken or were about to take subjects with basic programming content) and requested validation by experts in software development an in the Ignatian Pedagogical Proposal, those who reviewed and verified the correct use of concepts in both areas.

Keywords: Ssoftware development, university education, V model, Ignatian Pedagogical Paradigm, meta methodology.

INTRODUCCIÓN

En este artículo se describe la elaboración y resultados del trabajo de grado intitulado "Propuesta metodológica para el desarrollo de software con innovación para estudiantes en etapas iniciales de formación universitaria" **[1],** elaborado por María Tirado Pérez, bajo la tutoría del profesor Jesús Lárez Mata, para optar al grado de Ingeniero en Informática, de la Universidad Católica Andrés Bello. La metodología denominada Jade, es una propuesta metodológica destinada a estudiantes en etapas inicialesde su formación universitaria sobre cómo desarrollar software con innovación, la cual permitir enseñarles las buenas prácticas de programación y el buen uso de metodologías de desarrollo de software.

El artículo se encuentra estructurado en 8 apartados: primeramente se plantea el problema que dio origen a la investigación, los objetivos de la misma y sus trabajos relacionados. Seguidamente, se indica la motivación de la metodología y los modelos tomados en cuenta para su elaboración. Posteriormente, se describen los principios, actividades y fases de la metodología; luego se detallan las pruebas y validaciones realizadas a la misma. Finalmente, se describen las conclusiones y recomendaciones obtenidas al culminar el trabajo.

PLANTEAMIENTO DEL PROBLEMA

El software se ha convertido en un elemento con alta influencia en la sociedad actual, debido a que constituyen herramientas que facilitan la realización de procesos informáticos en organizaciones y en la vida cotidiana. En consecuencia, la demanda por desarrolladores de software ha ido en aumento, llegando a ser uno de los ámbitos de estudio con mayor demanda en universidades e institutos de formación alrededor del mundo.

Sin embargo, no solo se necesitan personas capaces de desarrollar software, sino que se necesitan personas que sean capaces de desarrollar software con calidad, quienes tengan competencias profesionales y éticas de un ingeniero de software, las cuales son cualidades y capacidades que se desarrollan a lo largo de su formación académica. Por lo cual, durante este tiempo se deben impartir conceptos de calidad e ingeniería del software, así como buenas prácticas de programación, siendo de suma importancia el nivel escogido para iniciar con la enseñanza de estos conceptos, puesto a que será el momento a partir del cual los estudiantes comenzarán a hacer uso de ellos. En este sentido, enseñar dichos conceptos desde las etapas iniciales de formación académica representa una ventaja en cuanto a etapas avanzadas, debido a que hay menor resistencia al cambio entre los estudiantes, además, cuentan con mayor cantidad de tiempo para practicar y aprender correctamente su uso antes de ingresar al campo laboral. A pesar de ello, las técnicas y herramientas de ingeniería del software, como las metodologías de desarrollo, están orientadas a personas con conocimientos más amplios en el tema, dificultando su enseñanza a personas que están iniciando sus

estudios.

En base a los motivos expuestos anteriormente, se definieron los objetivos de la investigación, para lograr que el profesional adquiera las competencias para el buen desarrollo de software desde el inicio de su formación.

OBJETIVOS DE LA INVESTIGACIÓN

Objetivo General

Elaborar una propuesta metodológica para el desarrollo de software con innovación para estudiantes en etapas iniciales de formación universitaria.

Objetivos Específicos

- Identificar los elementos involucrados en el proceso de formación académica en técnicas de desarrollo de software.
- Estudiar los beneficios de la innovación como parte del desarrollo de un software exitoso.
- Determinar los principales aspectos de la ingeniería del software que permitan la incorporación de un proceso de formación académica e innovación en el desarrollo de software.
- Diseñar la propuesta metodológica integrando los aspectos de formación académica e innovación definidos previamente.
- Validar la propuesta metodológica mediante un caso de estudio.

Trabajos relacionados

Enseñando a programar: un camino directo para desarrollar el pensamiento computacional [2]

Es una investigación sobre la enseñanza y aprendizaje del pensamiento computacional en materias de introducción a la programación, la cual relaciona aspectos a los cuales deben enfrentarse los profesores durante el proceso, como el desconocimiento en materia y la falta de un hábito adecuado de estudio por parte de los estudiantes, además de la dificultad presente en enseñar cómo abordar problemas desde el pensamiento computacional, donde las soluciones deben ser presentadas como instrucciones de computadora y algoritmos. Para afrontar este reto, los autores presentan las siguientes etapas que deberían atravesar los procesos de enseñanza y aprendizaje para que se obtengan resultados exitosos.

- Oír vs. Escuchar: clase magistral por parte del docente, donde explica la teoría del pensamiento computacional siendo fundamental que el estudiante esté motivado por el verdadero deseo de aprender, no solo para obtener una nota y un título.
- Ver vs. Observar: sistema de visualización de programas, se usa para que los estudiantes observen programas y asimile lo que están viendo, aprendiendo a través de simulaciones y abstracciones.
- Hacer equivale a practicar: los estudiantes deben poner manos a la obra sobre que se les ha enseñado, siendo más importante que ellos sean capaces de estructurar la solución de un problema por sí

mismos, en lugar de ver y aceptar la respuesta correcta del profesor, o sólo enfocarse en seguir perfectamente la sintaxis del lenguaje que estén usando.

- Infundiendo buenos hábitos: en programación escribir algoritmos que funcionen no es el único propósito, sino que también se deben cumplir ciertos parámetros, como el uso correcto de las estructuras, la indentación en código y la documentación.

Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming [3]

Es un artículo donde se discuten problemas fundamentales que se encuentran en los estudiantes de primer año en carreras de programación y propone un método de enseñanza para disminuir la deserción y promover el éxito en nuevos estudiantes de programación. Entre los problemas descritos por Miliszewska y Tan (2007) se refleja la falta de unidades de resolución de problemas lógicos dentro de la educación primaria y secundaria, siendo la universidad el lugar donde los estudiantes dan sus primeros pasos en el pensamiento computacional, aunado a que enseñar y aprender cómo resolver problemas desde una visión lógica tiene un grado medio-alto de dificultad, debido a que se debe comprender una gran cantidad de nuevos términos, los cuales resultan ser abstractos, no relacionados con su vida cotidiana. Por otro lado, cada estudiante tiene dificultades que van más allá de los mencionados anteriormente, afectando el rendimiento de su aprendizaje.

Los métodos propuestos para dar solución a dichos problemas abarcan desde cambiar el currículo del curso para que los conceptos más básicos de la programación fueran enseñados primeramente y luego pasar a conceptos abstractos como programación orientada a objetos, hasta enseñar mediante analogías, donde los conceptos de programación se enseñan utilizando ejemplos cotidianos que le permitieran al estudiante familiarizarse con el pensamiento computacional.

META METODOLOGÍA

La elaboración de la metodología estuvo basado en el trabajo de Abou-Zeid, incorporando información de innovación y formación académica que permitieran alimentar cada una de las etapas de la metodología resultante **[4]**.

Motivación

Tomando en cuenta la necesidad de desarrolladores de software con cualidades integrales que vayan más allá de habilidades técnicas sobre cómo desarrollar software, surge la metodología Jade, con la finalidad de establecer mecanismos que permitan a las instituciones forman a los estudiantes para que alcanzen una serie de competencias al momento de su egreso.

Con la elaboración de la metodología se deseó otorgar a estudiantes e instituciones una herramienta que permita enseñar a programar y a adquirir buenas prácticas de desarrollo de software desde los inicios en su formación como profesionales, desarrollando en ellos una serie de competencias profesionales y habilidades blandas a medida que avanzan en su carrera.

Así mismo, para cumplir con dicho propósito se recogió la información necesaria para brindar soporte al proceso a seguir posteriormente, para ello se consultó información sobre formación educativa y software. En cuanto a software, se consultó información acerca de ingeniería de software y principios a seguir en el proceso, al igual que diferentes modelos y metodologías ágiles y tradicionales utilizadas para desarrollar software **[5]**. En cuanto a formación educativa, se consultó información acerca de los etapas y pilares de la eduación **[2]**, incluyendo la perspectiva desde la enseñanza de la programación **[6]**. De igual manera, se incluyó información acerca de buenas prácticas **[7]** que deberían aplicar los estudiantes al momento de desarrollar software y que deberían aprender desde que inician su formación, así como conceptos a impartir en clases teóricas **[8][9]10]**.

Modelos

Luego de atravesar un proceso de análisis, evaluación y selección entre distintos modelos de desarrollo y aprendizaje, se seleccionaron el Paradigma Pedagógico Ignaciano y el Modelo en V como aquellos que brindaban mayor soporte a los objetivos del trabajo.

Paradigma Pedagógico Ignaciano

El Paradigma Pedagógico Ignaciano (PPI) es una propuesta educativa y pedagógica perteneciente a la Compañía de Jesús, es definido como un un proceso dinámico y sucesivo que se realiza a través de cinco etapas, donde el proceso educativo se refiere al aula y a la institución donde se realiza, siendo un paradigma con un camino enfocado al desarrollo de los estudiantes como personales integrales **[11]**.

1. Contexto: es el punto de partida para el proceso de educación, donde se debe contextualizar a los estudiantes con la realidad de aquello que se desea enseñar. La contextualización puede llevarse a cabo a la distancia o en el sitio, siendo importante en ambos casos que se cuente con el escenario adecuado para la vivencia de experiencias. De igual forma, el profesor debe contextualizarse dentro de la realidad de sus estudiantes, tomando en cuando lo que ellos conocen y lo que se desea que ellos aprendan.
2. Experiencia: se trata del momento en donde el individuo se abre a la realidad para tener un aprendizaje, cuando se produzca el entendimiento dejará de ser experiencia. Es una etapa donde el educador debe desarrollar en sus estudiantes la capacidad de atender para percibir la realidad y lo que ocurre en ella.

3. Reflexión: es la etapa donde el individuo relaciona la experiencia vivida con otros conocimientos. Las operaciones fundamentales de la reflexión permitirán al individuo interiorizar el significado de la experiencia y desarrollar una formación crítica.
4. Acción: la pedagogía ignaciana lleva al individuo a dar un paso más allá en el proceso de educación, a tomar una posición frente a la verdad recientemente descubierta y actuar en relación a ello con una nueva visión de la realidad.
5. Evaluación: en la etapa de evaluación se hace una revisión del proceso pedagógico y las etapas que se siguieron a lo largo del proceso, con el fin de determinar el grado en que los objetivos han sido alcanzados, e identificar los aciertos y dificultades del procedimiento.

Modelo en V

El modelo en V es un modelo de desarrollo basado en el modelo en cascada, añadiendo acciones de verificación y validación con el fin de asegurar la calidad del producto resultante **[12]**. En la figura 1 se puede apreciar una representación del proceso que sigue el modelo en V; en el lado izquierdo de la V se encuentran las actividades de definición, diseño de arquitectura y diseño detallado del sistema, una vez se complementen dichas etapas se procede a generar el código, luego se sube por el otro lado. En el lado derecho de la V se encuentran la serie de pruebas que validan los resultados del lado izquierdo.

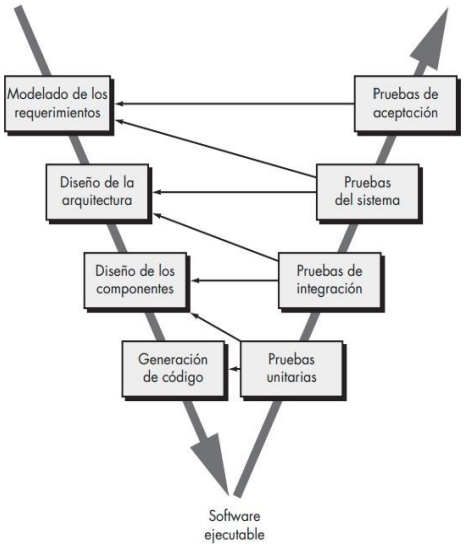


Figura 1. Representación gráfica de las etapas de desarrollo en el Modelo en V **[5]**.

- Nivel 1: es el nivel más cercano al cliente. En el modelado de requerimientos se recoge la información necesaria sobre las necesidades del cliente y sus usuarios con respecto al sistema, la cual se traduce en los requerimientos del proyecto. Por el otro lado, en las pruebas de aceptación se comprueba si los resultados obtenidos corresponden con los requerimientos y satisfacen lo solicitado por el cliente.
- Nivel 2: es el nivel general del sistema. En el diseño de la arquitectura se determinan los componentes y cómo van a interactuar entre sí, sin entrar en detalles técnicos de implementación de

cada uno. En las pruebas del sistema, se prueba que los componentes implementados se comunican e interactúan correctamente, las pruebas de integración también ayudan con este propósito.

- Nivel 3: es el nivel de especificación y datos técnicos. En el diseño de los componentes se describe cómo será su funcionamiento interno y la comunicación de cada uno de sus procesos internos. Las pruebas de integración y pruebas unitarias se encargan de probar que los componentes funcionen correctamente como unidades y sean capaces de brindar los resultados que se espera de ellos.
- Nivel 4: es el nivel de codificación, donde se implementan los diseños realizados y se genera el código de la aplicación. Durante este nivel también se realizan las pruebas unitarias a los elementos que se desarrollen. Al terminar con este nivel, se vuelve a subir por la V.

Metodología Jade

La metodología Jade está sustentada baja los enfoques del modelo en V y la Propuesta Pedagógica Ignaciana, tomando en cuenta algunos principios del Manifiesto Ágil; esta abarca dos procesos claves para poder cumplir con los objetivos planteados, la primera parte toma en cuenta la formación educativa que debe recibir un estudiante antes de iniciar el desarrollo de un proyecto y la segunda abarca el proceso de desarrollo de un proyecto tomando en cuenta los conceptos y prácticas que se impartieron durante la formación previa.

Principios

1. Todos los miembros del equipo son responsables del aprendizaje, participando activamente dentro del proceso, sin presentar obstáculos y ayudando a los compañeros a alcanzarlo.
2. La conversación cara a cara es primordial entre miembros del equipo.
3. La comunicación entre los miembros del equipo debe ser constante, responsable y respetuosa.
4. El equipo debe reflexionar continuamente sobre cómo ser más efectivo y perfeccionar su comportamiento.
5. Las entradas y salidas de cada fase son revisadas para asegurar que se obtuvo el producto correcto y se produjo un aprendizaje.

Ámbito

La metodología está orientada a equipos que están atravesando un proceso de iniciación en la programación, a los cuales se les debe enseñar cómo desarrollar un producto de software y fomentar en ellos una serie de competencias deseables en los profesionales de software.

Formación Educativa

La formación previa consiste en un proceso donde se imparte a los estudiantes los conceptos básicos de programación a través de clases teóricas, los cuales son usados por ellos mismos en las clases prácticas. En este apartado de la metodología se describen aspectos a considerar en las clases teóricas y prácticas de los cursos introductorios de programación.

Conceptos

Un curso introductorio de programación generalmente cuenta con un plan de clases donde se especifican los conceptos que serán abordados a lo largo del curso. Tomando en cuenta la información recopilada durante la investigación, tomada de diversas fuentes como **[6][8][9][10]**, se ofrece una lista de conceptos recomendados para ser impartidos en este tipo de cursos, sin embargo, no deben representar un límite para definir un plan de clases.

- Proceso de Resolución de Problemas: análisis, diseño, codificación y pruebas.
- Estructuras Algorítmicas Básicas: estructuras secuenciales, condicionales y repetitivas.
- Manejo de Datos: tipos y estructuras de datos.
- Paradigmas y Lenguajes de Programación: definir cuál o cuáles paradigmas y lenguajes serán enseñados y plantear otros que no serán abordados, así el estudiante podrá conocer de su existencia e iniciar un proceso de aprendizaje por su cuenta si lo desea.

Prácticas de Programación

Por otro lado, a los estudiantes se les debe enseñar cuáles son las buenas prácticas de programación y cómo debe ser su código, así inician a desarrollar su capacidad de escribir buen código desde sus inicios en la programación. A continuación se presentan una serie de buenas prácticas que deben ser impartidas a los estudiantes.

- Estilo: puntos que hacen el código más legible y mantenible en el tiempo.
- Pruebas: depuradores, herramientas para encontrar y solucionar errores en el programa, cómo validar que el programa cumple con su propósito.
- Control de Versiones: git, repositorios.

Es importante que cada concepto en teoría y práctica sea dado con ejemplos que ayude al estudiante a asimilar el conocimiento y aprender a ponerlo en práctica; dichos ejemplos pueden ir desde analogías con elementos de la vida cotidiana hasta un código donde sea puesto en práctica.

Sesiones de Teoría

Durante las sesiones de teoría se desea que el profesor aborde las etapas de contexto, experiencia y reflexión del proceso descrito en el PPI, etapas donde el profesor deberá enseñar a los estudiantes los conceptos teóricos según su plan de clases. A continuación se detalla la relación entre las clases y las etapas mencionadas.

1. Contexto: indicará el inicio de la sesión, en este momento se indica a los estudiantes el itinerario, el propósito y los resultados a obtener durante la sesión, tomando en cuenta lo que ellos conocen y lo que se desea que ellos aprendan. Es importante relacionar los conocimientos previos de los estudiantes con los

conocimientos a impartir, así el estudiante puede contextualizar la información que recibe en la clase y ubicarla dentro de un contexto que ya conoce. La contextualización debe incluir información acerca de la sesión teórica y su parte práctica.

2. Experiencia: se refiere al momento de la sesión donde el profesor transmite los conceptos a los estudiantes, apoyándose en ejemplos que relacionen el concepto con otros elementos conocidos por los estudiantes, se puede iniciar el curso con ejemplos de la vida real y a medida que se avance se usan ejemplos dentro del contexto de la programación. La etapa de experiencia también puede referirse a información que obtienen los estudiantes como resultado a otras actividades y su evaluación correspondiente.

3. Reflexión: es la etapa donde los estudiantes ubican la información recibida dentro de una realidad y relacionan los conocimientos recibidos con otros conocidos anteriormente; siendo capaces de tener opinión propia sobre ellos, sobre cómo, dónde y cuándo usarlos.

Sesiones de Práctica

Durante la práctica, se deben abordar las siguientes etapas del PPI, primeramente el estudiante tiene mayor protagonismo, luego el profesor ofrece retroalimentación al trabajo realizado.

4. Acción: durante esta etapa los estudiantes deben realizar tareas prácticas destinadas a generar destreza en la aplicación de los conceptos adquiridos durante la sesión teórica. La práctica puede ser realizada a través de pseudocódigo o haciendo uso de un lenguaje de programación. Es importante que las prácticas sean resueltas siguiendo un proceso de resolución de problemas igual o similar al descrito en las bases teóricas.

5. Evaluación: el profesor debe indicar a los estudiantes si su solución al problema planteado al inicio de la sesión práctica es correcta según los parámetros iniciales, también es recomendable que se les indique a los estudiantes cómo deberían haber resuelto el problema, puesto que ver código bien escrito ayuda a que ellos escriban un buen código.

DESARROLLO DE PROYECTOS

Equipo de Trabajo

- Profesor (Asesor, Cliente y Usuario): el profesor de la asignatura estará presente en el equipo de desarrollo en diferentes roles: como asesor, cliente y, en ocasiones, usuario. El asesor el miembro del equipo que cuenta con conocimientos sólidos sobre los temas a tratar en el desarrollo, es un rol que puede ser ocupado por el profesor de la cátedra para la cual está asignado el proyecto. El cliente es aquel que realiza la petición del producto de software, indicando sus necesidades y el problema que desea solucionar. Por último, el usuario la persona o grupo de personas hacia las cuales va dirigido el producto, aquellos que van a interactuar con el software. En algunos casos, dos o más de estos roles pueden ser ocupados por la

misma persona, siendo un ejemplo de esto cuando un profesor asigna el proyecto para la cátedra (cliente) y a su vez atiende las dudas de los equipos (asesor).

- Líder: es el responsable de guiar al equipo de trabajo para alcanzar los objetivos del proyecto, fomentar un ambiente sano de trabajo y hacer seguimiento continuo de los avances y entregas.
- Desarrolladores: son los miembros del equipo encargados de desarrollar el producto de software, cuentan con los conocimientos y habilidades necesarias para dar cumplimiento a la solicitud del cliente, apoyándose en el asesor del equipo de trabajo.

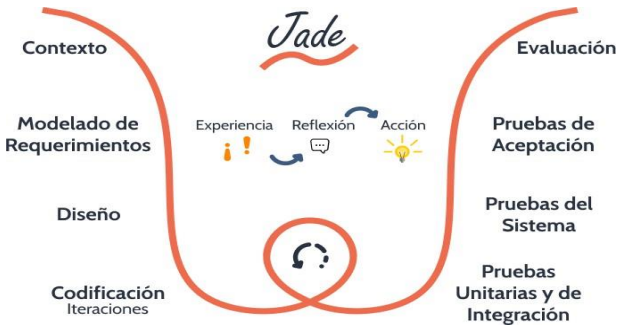
Actividades

- Reunión de Contexto: es una reunión donde participa todo el equipo, la cual tiene lugar al inicio de cada fases, en ella se discute el propósito de la etapa, cuáles son las entradas y cuáles son los resultados que se desean obtener.
- Reunión de Revisión: es una reunión que tiene lugar de manera intermedia entre las etapas de una fase, cuando se obtiene el producto de ella. Se desea que los miembros del equipo realicen una revisión de sus avances con el asesor del equipo, de esta forma se asegura que se cumpla el objetivo de la fase antes de avanzar a la siguiente. Si se determina que lo anterior no ha sido logrado, se reinicia el proceso de la fase.
- Reunión de Evaluación: es una reunión donde participa todo del equipo y tiene lugar al final de cada fase, en ella el equipo reflexiona sobre el proceso seguido, el aprendizaje obtenido y sobre cómo perfeccionar su comportamiento.

Fases

Las fases de la metodología permitirán que se pueda seguir un proceso de desarrollo de software siguiendo los principios establecidos anteriormente, las cuales son representadas en la figura 2. Cada fase inicia con una reunión de contexto, continúa con las etapas de experiencia, reflexión y acción para cada una, y finalmente se cierra con una reunión de evaluación. Adicionalmente, al obtener el producto de cada fase, se realiza una reunión de revisión junto con el asesor y el cliente del proyecto.

Figura 2. Etapas de desarrollo de proyectos.



1. Contexto: es la fase de inicial del proyecto, durante ella se busca contextualizar al equipo dentro de su nuevo papel en la realidad, son desarrolladores de software que van a realizar un producto de software. Para ello se hace uso de una clase magistral, donde el asesor les explica qué se desea que aprendan durante el proceso, las competencias a desarrollar, junto con los objetivos del proyecto y la dinámica a seguir, aunque sin entrar en detalles específicos de cada una.

2. Modelado de Requerimientos: una vez completada la contextualización, se procede a iniciar con la fase de reconocimiento del proyecto. Durante esta fase la experiencia será que el cliente transmita sus deseos y necesidades con respecto al software a desarrollar, para ello puede estar presente durante la reunión de contexto y tomar partido en ella, o tener una reunión aparte. Sea cual sea el caso, se recomienda que el cliente atienda a las consultas del equipo y aclare todas las dudas posibles. Así mismo, es tarea del equipo recoger la información necesaria por su cuenta, esto quiere decir, investigar acerca de las funcionalidades y características que poseen los sistemas similares para determinar si son elementos deseables en el software a desarrollar, lo que a su vez permite incorporar elementos innovadores al programa. En la reflexión se deben responder a los cuestionamientos generales propuestos, aunque el equipo de desarrollo también puede cuestionar otros elementos que les permiten tener juicio propio sobre la actividad. El resultado de esta fase es un documento con la lista de elementos con los que debe contar el software, la cual se alimenta con la información recogida a través del cliente y en la investigación realizada.

3. Diseño General: cuando los miembros del equipo tengan clara la idea del software a desarrollar, se procede a realizar los diseños del sistema. En el diseño general, se especifican los componentes y cómo será su interacción entre sí, entendiéndose como componente a un bloque de software, una parte modular, desplegable y sustituible de un sistema **[5]**. Esta fase se alimenta de la experiencia vivida durante la realización del listado de requerimientos, debido a que durante ese proceso los miembros del equipo se acercan a la realidad de lo solicitado por el cliente, transmitiendo sus deseos en palabras específicas con mayor sentido en el ámbito de desarrollo de software. El objetivo de la fase es generar un documento de diseño del sistema que cumpla con lo establecido en la lista de requerimientos y de respuesta a las necesidades del cliente. El mismo debe demostrar cómo se dará solución al problema, las herramientas a utilizar y la definición de componentes o módulos que pertenecerán al sistema.

4. Diseño de Componentes: el siguiente paso es realizar el diseño de componentes del software, basado en el diseño general y los componentes definidos en la fase anterior. El diseño de componentes otorga la especificación técnica de cada componente, con sus procesos internos, entradas y salidas. El documento a entregar debe incluir los diseños obtenidos y ser una visión detallada del diseño general, teniendo coherencia entre ellos mismos y con los comportamientos descritos durante la etapa anterior.

5. Codificación: La codificación será la puesta en práctica de los diseños realizados anteriormente, donde se genera el código fuente del software. Para ello se recomienda establecer pruebas unitarias y luego escribir el código para pasar dichas pruebas. Cuando un componente es finalizado, se pasa por las pruebas escritas anteriormente y se presentan avances durante reuniones de revisión. La fase es alimentada por los resultados obtenidos en las fases de diseño y el resultado a obtener consiste en el código del programa. La codificación debe seguir un proceso de refinamiento, donde el código escrito por los desarrolladores es refinado a través de iteraciones. Al inicio de cada iteración, los miembros del equipo escogen los requerimientos a desarrollar; posteriormente, cuando se tiene el código que da respuesta a dichos requerimientos, se tiene una reunión de revisión, donde se presentan los resultados y el asesor debe presentar una solución alternativa, así los estudiantes pueden realizar una reflexión entre lo que hicieron y lo que deberían haber hecho. El propósito de esta práctica es que los estudiantes pueden escribir código cada vez mejor y más eficiente, tomando en cuenta las enseñanzas del asesor. Durante esta fase se deben entregar documentos de avances a medida que se escribe el código del software, los cuales pueden incluir la lista de requerimientos establecidos al inicio del avance detallando aquellos que fuera cumplidos y aquellos que no, las pruebas realizadas y sus resultados, y los cambios realizados durante la iteración anterior en base a la solución que indicó el profesor (si aplica).

6. Pruebas: las pruebas son realizadas por dos propósitos primero, para descubrir defectos en el software y su comportamiento, y segundo, para demostrar a los desarrolladores y al cliente que el software satisface los requerimientos **[12]**. Dentro del marco de desarrollo de software, las pruebas estarán constituidas por 5 etapas, detalladas a continuación:

a. Pruebas Unitarias: son las pruebas que se realizan a los componentes del sistema y a sus funciones internas, donde se prueba que su funcionamiento es correcto y da las salidas esperadas dependiendo de las entradas que reciban.

b. Pruebas de Integración: son pruebas destinadas al funcionamiento de varios componentes y la interacción que se produce entre ellos. Sirven para validar a los componentes como individuales y a los componentes como miembros de un sistema.

c. Pruebas del Sistema: las pruebas del sistema son realizadas al sistema completo, siendo observado como una unidad completa, evaluando si posee el comportamiento deseado y cumple con los requerimientos iniciales.

d. Pruebas de Aceptación: por último, las pruebas de aceptación son realizadas con el cliente, pruebas realizadas al sistema para determinar si sus deseos fueron cumplidos y los objetivos del proyecto logrados.

La reunión de revisión de esta fase es el momento adecuado para que el profesor recoja la información necesaria para su evaluación docente sobre el proyecto, incluyendo la revisión del código fuente, la recepción de documentos y la discusión sobre el desarrollo del proyecto. El documento final del proyecto

está sujeto a criterios de evaluación del docente y la institución donde se lleve a cabo el curso, pero este debería contener: las pruebas realizadas al programa y sus resultados, cómo se dio solución a los requerimientos y por qué no se pudieron completar funcionalidades que estaban contempladas al inicio del proyecto (si aplica). Las revisiones de los proyectos pueden ser realizadas de manera abierta, donde estudiantes miembros de otros equipos también puedan observar la solución que dieron sus compañeros al problema presentado al inicio del proyecto e interactuar con ellos. Es una práctica que fomenta la transparencia de la revisión y ayuda a que los estudiantes desarrollen la comprensión de otros puntos de vista de un problema. De igual manera, revelar el código fuente de los proyectos a través de una presentación o publicación en un repositorio ayuda a que los estudiantes tengan una motivación adicional para prestar mayor atención a la calidad de su código. Por último, el profesor debe explicar a los estudiantes cómo le daría solución al problema, siendo ideal que se presente un programa funcional.

7. Evaluación: es la fase final del desarrollo del proyecto y representa un cierre del mismo, mas no es una evaluación académica, sino una evaluación personal. Durante esta fase solo se realiza una reunión donde participa todo el equipo o salón de clases y tiene como fin ofrecer un espacio de retroalimentación entre los miembros del equipo. El profesor y los estudiantes deben intercambiar opiniones y críticas constructivas que permitan a ambas partes reflexionar y ofrecer un mejor desempeño en futuros proyectos. Al finalizar esta reunión, se da por concluido el proyecto.

RESULTADOS

Caso de Estudio: Algoritmos y Programación 0

La propuesta metodológica fue aplicada en el curso Algoritmos y Programación 0 dictado en conjunto con la Escuela de Ingeniería Informática de la Universidad Católica Andrés Extensión Guayana a un grupo de estudiantes de primer y segundo semestre de dicha carrera junto con un grupo de estudiantes de cuarto semestre de Ingeniería Industrial. El caso de estudio constó de 4 sesiones teórico-prácticas donde se aplicaron los puntos de formación previa y desarrollo de proyectos pertenecientes a la metodología Jade.

El propósito del curso fue enseñar conceptos básicos de programación a aquellos estudiantes sin conocimiento previo, reforzar conceptos a aquellos que habían cursado materias similares y enseñar el uso de metodologías en desarrollo de proyectos.

Equipo de Trabajo

El grupo que conformó el caso de estudio estuvo formado por un asesor teórico, un asesor práctico y un grupo de estudiantes pertenecientes a las escuelas de Ingeniería Informática e Industrial de la universidad antes mencionada; además, se contó con la participación de una persona externa, quien participó en la dinámica de desarrollo de proyectos como cliente.

El grupo de estudiantes que asistió al curso estuvo conformado por 19 estudiantes de Ingeniería Informática, quienes ya habían cursado la materia Introducción a la Informática, y 3 estudiantes de Ingeniería Industrial, quienes no contaban con conocimientos previos sobre programación. Durante la formación previa no se realizó división entre los estudiantes, sin embargo, al momento de desarrollar el proyecto se les solicitó conformar equipos de 4 estudiantes, donde cada equipo tendría un líder y todos serían desarrolladores.

Formación Previa

Durante la primera etapa del curso se dictaron clases teórico-prácticas tomando en cuenta el programa de clases de las materias próximas a cursar por los estudiantes y aquellas que ya cursaron. El plan de clases para el curso fue organizado siguiendo el esquema propuesto en la metodología, el cual está basado en la Propuesta Pedagógica Ignaciana. Todas las sesiones de formación previa siguieron el esquema de teoría y práctica descrito en la metodología, donde la teoría abarca las etapas de contexto, experiencia y reflexión del PPI, mientras que la práctica aborda las etapas de acción y evaluación.

Al inicio de cada sesión se abría un espacio para conocer la realidad de los estudiantes, hablar sobre lo que conocían sobre el tema de programación, reforzar temas vistos en días anteriores y conversar sobre el plan para el día.

En la teoría de cada sesión se dictaron los contenidos del día a través de una clase magistral, las cuales estuvo acompañada de medios visuales; así mismo, en cada contenido se preguntó a los estudiantes que conocían acerca de ello, se ofrecieron ejemplos y se verificó que el tema fue comprendido antes de pasar al siguiente.

Posteriormente, se utilizó el lenguaje Python como lenguaje de programación en la práctica, debido a los beneficios del lenguaje en relación a la enseñanza de programación, detallado en las bases teóricas. Durante las prácticas se presentaron ejercicios propuestos a los estudiantes para encontrar su solución y al final de la práctica de ofrecía la solución a dichos ejercicios.

Para finalizar las sesiones, se hacía un repaso sobre lo visto durante el día y se ofrecían recomendaciones a los estudiantes sobre sus soluciones a los ejercicios. Así mismo, también se solicitó retroalimentación a los estudiantes.

1. Sesión 1: durante la primera sesión se dio inicio al curso, realizando la presentación introductoria correspondiente y dando la bienvenida a los estudiantes. Fue una sesión donde se vieron conceptos básicos ya conocidos por la mayoría de los estudiantes, teniendo especial atención en aquellos que no, para así poder llevar a los estudiantes a manejar los conceptos de igual forma. Se aplicó una prueba diagnóstica a los estudiantes al inicio de la sesión teórica; sin embargo, los estudiantes de Ingeniería Industrial no fueron

capaces de completar la prueba en dicho momento con el resto del grupo, debido a que no contaban con conocimientos previos en el área de programación.

El contenido de la primera sesión estuvo conformado con algunos temas del contenido que se ven en las asignaturas Introducción a la Informática y Fundamentos de Programación, además de temas relacionados a buenas prácticas de programación. Durante la teoría y práctica se prestó especial atención a los estudiantes de Ingeniería Industrial para que se pudieran cumplir los objetivos de la sesión y el curso al igual que el resto de los estudiantes. Con ello los estudiantes pudieron completar exitosamente los ejercicios de la prueba diagnóstica al finalizar la sesión.

2. Sesión 2: el contenido de la segunda sesión consistió en explicar los conceptos que los estudiantes deben conocer en la asignatura Algoritmos y Programación I, para que así pudieran conocer la base conceptual, necesaria para el desarrollo de un programa. La sesión fue llevada a cabo siguiendo la dinámica mencionada anteriormente.

3. Sesión 3: la tercera sesión estuvo destinada a enseñar a los estudiantes cómo podían hacer pruebas a sus programas de manera automatizada y como hacer sus programas más robustos, evitando que sean vulnerables por errores cometidos durante su construcción. El contenido estuvo basado en herramientas usadas comúnmente en la programación y que ayudan ampliamente en el desarrollo de software de calidad.

Desarrollo de Proyecto

En la última sesión del curso se llevó a cabo la dinámica de un proyecto, donde los estudiantes fueron considerados desarrolladores de software y quienes iban a realizar el programa solicitado por el cliente, haciendo uso de los conceptos impartidos durante las sesiones anteriores y siguiendo el apartado de desarrollo de proyectos de la metodología Jade. Al igual que las otras sesiones, la clase se llevó a cabo siguiendo el esquema de clases de la metodología, impartiendo en el contenido información sobre desarrollo de proyectos. Posteriormente, se dio inicio a la dinámica final del curso.

1. Contexto: en la fase de contexto, se indicó a los estudiantes cómo sería la dinámica y con qué fin sería realizada, especificando cada fase del proyecto y que se esperaba de ellos en cada una. Posteriormente, se realizó una reunión con el cliente, donde este último realizó la petición del sistema a los estudiantes. Previamente, se solicitó al cliente dar información general acerca del sistema a realizar, para que de esta manera los estudiantes pudieran experimentar, por su cuenta, la situación donde el cliente no tiene una idea concreta sobre lo que desea y el equipo de desarrollo debe recoger más información para dar forma a ideas imprecisas. En esta reunión se produjo una mayor interacción entre los estudiantes y el cliente, donde varios estudiantes formularon preguntas con el fin de recoger más información para definir correctamente su sistema.

2. Modelado de Requerimientos: la fase de modelado de requerimientos inició con una reunión de contexto, donde se les explicó a los estudiantes que debían hacer durante la fase, comenzando con realizar una lista con los requerimientos para su sistema. Se solicitó a cada grupo que escribiera sus requerimientos de forma separada, así cada uno podría definir su sistema de acuerdo a lo que sus integrantes comprendieron, diferenciándose entre sí y fomentando la innovación. Así mismo, se les indicó que los requerimientos de su sistema serían usados durante las pruebas de aceptación, que es una fase posterior del desarrollo. Cuando todos los grupos culminaron con el modelo de requerimientos y tenían una idea clara de cómo sería su sistema, una vez culminado, se realizó una reunión de evaluación, donde se revisaron los requerimientos de cada grupo para verificar que estuvieran en consonancia con lo solicitado por el cliente. Por último, se les brindó la lista de requerimientos oficial para el proyecto, permitiendo que los estudiantes pudieran realizar una comparación y mejoren la definición de su sistema.

3. Diseño: en la reunión de contexto se indicó a los estudiantes que debían establecer el diseño de su programa antes de comenzar a escribir el código, definir cómo iban a solucionar el problema, identificando sus entradas, procesos y salidas así como otros elementos que consideraran importantes en esta fase. Una vez los grupos realizaron el diseño de su sistema, se procedió a la reunión de evaluación, donde el asesor del curso procedió a evaluar los diseños presentados. Se encontró que la mayoría de los grupos harían uso de arreglos y funciones en su programa, también fue posible observar el diseño del menú de algunos grupos.

4. Codificación y Pruebas: seguidamente se dio inicio a la fase de codificación, en la reunión de contexto se dieron pautas a los estudiantes sobre su código y que debían lograr durante esta fase. A medida que cada grupo avanzó en el desarrollo de su sistema, donde se realizaron reuniones de revisión con cada uno, donde se verificó su avance y se atendieron las dudas de los estudiantes. La fase de pruebas fue de la mano con la fase de codificación, atendiendo al principio “probar a medida que se escribe el código”, por ello en la reunión de contexto anterior se solicitó a los estudiantes que realizaran pruebas a su sistema a medida que escribían bloques de código. Antes de finalizar el desarrollo del programa, se realizó otra ronda de reuniones de revisión, donde se apoyó a los estudiantes en las pruebas del sistema, haciendo un contraste entre el programa y el diseño que habían realizado anteriormente. Se encontró en los grupos que su sistema cumplió con el diseño que establecieron al inicio, por lo cual se procedió a realizar las pruebas de aceptación. Para las pruebas de aceptación, el usuario y el profesor del curso visitaron las estaciones de cada grupo, donde el usuario pudo interactuar con los programas desarrollados y ambos emitieron sus recomendaciones para futuros desarrollos. Durante las visitas, se pudo observar que todos los grupos cumplieron con las pautas establecidas para el proyecto, alcanzando los objetivos de la dinámica. Igualmente, algunos grupos fueron más allá de las especificaciones dadas, incorporando elementos extra a sus programas, como música que acompañó al juego y archivos para almacenar información del sistema.

5. Evaluación: Al finalizar la revisión de los resultados de cada equipo, se continuó con la fase de evaluación. Para ello, los instructores brindaron sus consejos y opiniones a los estudiantes sobre su desempeño en el curso y en la dinámica final. Igualmente, también se realizó una ronda de preguntas hacia los estudiantes para determinar si el contenido visto durante el curso había sido asimilado. Para recibir retroalimentación por parte de los estudiantes, se les solicitó su participación en una encuesta, donde podrían detallar su experiencia a lo largo del curso. De igual forma, algunos estudiantes contaron sus opiniones de forma presencial.

Encuestas a Participantes del Curso

Para finalizar el curso y recibir una retroalimentación más detallada, se solicitó a los estudiantes completar una encuesta de 11 preguntas a través de la herramienta Google Forms, donde hubo una participación del 85% de los estudiantes que asistieron al curso, obteniendo un total de 17 respuestas a la encuesta, siendo 2 de los estudiantes de Ingeniería Industrial y 15 de Ingeniería Informática. En los resultados obtenidos se destacó que:

- La mayoría de los contenidos fueron aprendidos.
- Hubo aceptación a la modalidad de clases teórico–prácticas.
- Prefieren realizar varias entregas y recibir orientación.
- Hubo alto nivel de satisfacción en cuanto al curso y a sus instructores.

Encuestas a Profesores y Tutores de Programación

Posteriormente, se realizó una encuesta a profesores y estudiantes tutores de cátedras relacionadas a programación con la finalidad de tener un punto de comparación acerca del rendimiento de los estudiantes en clases tradicionales, donde la teoría y la práctica son dictadas en clases separadas. De igual forma, se realizó a través de la herramienta Google Forms y en los resultados obtenidos se destacó que:

- En la clase práctica a menudo se deben volver a explicar conceptos tratados en la clase teórica.
- Los estudiantes demuestran mayor interés y mejor desempeño en la clase práctica.
- Consideran que mostrar los resultados de ejercicios podrían ser de mayor provecho para los estudiantes.
- Es necesario incluir dinámicas sobre el mundo real.

Validación por Expertos

La validación por expertos fue realizada por un experto de desarrollo de software y un experto en el enfoque ignaciano, quienes a su vez cuentan con experiencia en el área de formación educativa en una institución superior. Para ello, se les hizo entrega de un escrito con información de la metodología junto

con infografías que permiten visualizar el proceso y se realizaron entrevistas semiestructuradas para recibir su retroalimentación.

Experto en Desarrollo de Software

Para validar la propuesta metodológica desde el punto de vista de desarrollo de software se solicitó la opinión de un experto en el tema, quien brindó las siguientes opiniones y recomendaciones que pueden ayudar a los futuros grupos que hagan uso de la metodología.

- Puede ser usada en cursos institucionales de la escuela de Ingeniería Informática en la Universidad Católica Andrés Bello, así como en otras universidades que ofrecen carreras similares.
- Los estudiantes pueden realizar el mismo proyecto haciendo uso de diferentes lenguajes, así pueden ver soluciones haciendo uso de diferentes herramientas. En este sentido, indicó que sería conveniente entregar una lista de posibles lenguajes para que utilicen.
- La dinámica utilizada para el proyecto del caso de estudio sería de gran provecho en los cursos institucionales, agregando dinamismo a la clase y brindando experiencias parecidas al mundo real.
- El cliente debe estar más involucrado en el proceso, interactuando con el sistema en otras etapas.

Experto en Propuesta Pedagógica Ignaciana

Para validar la propuesta metodológica desde el punto de vista del enfoque ignaciano se solicitó la opinión de una persona con conocimientos amplios en el tema, el cual brindó las siguientes observaciones y recomendaciones para ayudar a refinar el proceso, algunas de las cuales coincidieron con las descritas por parte del experto en desarrollo de software.

- Incorporar preguntas a responder durante cada etapa, en la etapa de Evaluación es importante incluir aquellas relacionadas a solucionar alguna situación no provechosa durante el proceso, determinar si algo no funciona, por qué no funciona y cómo se puede arreglar para que no vuelva a pasar.
- Se debe llevar una bitácora del curso con las competencias que se trabajan a lo largo de las sesiones y el progreso de los estudiantes.
- Es importante revisar el progreso del proyecto en cada etapa, así se corrige si algún grupo está siguiendo una idea errónea y se mantiene la motivación en el curso.
- El cliente debe estar presente en las etapas de presentación de avances del programa, no solo durante la fase de Modelado de Requerimientos y Pruebas de Aceptación.
- La dinámica utilizada durante el caso de estudio puede ser utilizada en otros cursos, acercando al estudiante a experiencias reales del trabajo.
- La propuesta metodológica puede ser utilizado en otras carreras y universidades, adaptándola a otros contextos.

CONCLUSIONES

Las cualidades de un buen profesional en software son desarrolladas a lo largo de su formación académica, incorporar herramientas que permitan enseñar y promover dichas cualidades desde etapas tempranas genera productos de calidad e innovadores desde el inicio. Así mismo, fomentar valores y capacidades personales en los estudiantes ayuda a que egresen como profesionales integrales, independientes y sobresalientes. A continuación se presentan las conclusiones obtenidas luego de completar la investigación.

- Muchos de los temas que componen los Cursos de Ingeniería de Software ayudan a desarrollan un conjunto de competencias transversales, por lo que abordar aspectos relacionados con los mismos de forma temprana, siempre que fuese posible, en el proceso de enseñanza y aprendizaje va a redundar en un gran beneficio en la formación de los estudiantes.
- Incorporar la innovación como parte de un desarrollo de software promueve la curiosidad y originalidad en los estudiantes y fomenta a la mejora continua en sus programas.
- Los enfoques tradicionales ayudan a que los estudiantes puedan comprender y aprender las etapas básicas del proceso de desarrollo de software.
- La propuesta metodológica promueve el desarrollo de proyectos con calidad e innovación dentro de un ámbito universitario, tomando en cuenta el crecimiento académico y personal del estudiante.
- Incorporar el uso de la metodología en diferentes proyectos a lo largo de varios semestres puede ayudar en el crecimiento progresivo del estudiante.
- Un lenguaje con un alto nivel de abstracción ayuda a que los estudiantes se concentren en cómo resolver el problema, en lugar de preocuparse mayormente por la sintaxis del lenguaje.
- La motivación y capacidades como el trabajo en equipo son vitales para que un estudiante pueda llegar a ser un buen profesional de software, es por ello que la formación académica debe ir de la mano con la formación integral de la persona, guiando a los estudiantes en conocimientos técnicos y en su crecimiento personal.
- Indicar la solución correcta a un problema ayuda a que los estudiantes puedan identificar si tienen errores y a mejorar progresivamente sus programas.

Por otro lado, tomando en cuenta el trabajo que ha de ser realizado para continuar brindando validez a la metodología presentada en este trabajo, se ofrecen las siguientes recomendaciones a universidades y futuros investigadores que le darán uso.

- Incorporar elementos del Paradigma Pedagógico Ignaciano en clases dentro de las carreras ofrecidas.
- Aplicar la propuesta metodológica en un curso de mayor duración o en proyectos más extensos, donde las fases de desarrollo de proyectos puedan tomar mayor cantidad de tiempo.

- Incorporar temas más avanzados sobre programación en la propuesta metodológica, para que pueda ser aplicada a etapas posteriores de formación universitaria.
- Considerar el desarrollo de una metodología que incorpore un modelo ágil con los principios y etapas de la Propuesta Pedagógica Ignaciana, la cual puede ser utilizada en desarrollo de proyectos que requieran de un enfoque ágil.
- Evaluar otros esquemas que faciliten y mejoren el proceso de enseñanza y aprendizaje en cursos de formación académica.
- Considerar en desarrollo de una metodología destinada a enseñar conceptos básicos y desarrollo de proyectos en otros ámbitos.

REFERENCIAS

[1] Tirado, M. (2020). Propuesta Metodológica para el Desarrollo de Software con Innovación para Estudiantes en Etapas Iniciales de Educación Universitaria. Universidad Católica Andrés Bello, Venezuela.

[2] Compañ, P., Satorre, R., Llorens, F. y Molina R. (2015). Enseñando a Programar: Un Camino Directo para Desarrollar el Pensamiento Computacional. Revista de Edución a Distancia, número 46.

[3] Miliszewska, I. y Tan, G. (2007). Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming. Informing Science and Information Technology, volumen 4.

[4] Abou-Zeid, E. (2007). A Meta-Methodology for Knowledge Management Support Systems Development. ECIS 2007 Proceedigns, número 183.

[5] Pressman, R. (2010). Ingeniería del Software: Un Enfoque Práctico (7ma Edición). México: McGraw-Hill Interamericana Editores, S.A. de C.V.

[6] Wolovick, N. y Martínez, M. (2016). Enseñando a Programar y Programar para Aprender. Revistas de la Universidad Nacional de Córdoba, número 12.

[7] Kernighan, B. y Pike, R. (1999). The Practice of Programming. Estados Unidos: AddisonWesley Professional Computing Series.

[8] Rodríguez, J., Arana, L., Rabasa, A. y Martínez, O. (2003). Introducción a la Programación. Teoría y Práctica. España: Club Universitario.

[9] Ferris, R. y Albert, J., (s.f.) Tipos y Estructuras de Datos. España: Universidad de Valencia.

[10] Ferris, R. y Barber, D., (s.f.) Lenguajes de Programación. España: Universidad de Valencia.

[11] Granados, L. (2005). Paradigma Pedagógico Ignaciano. Reflexiones Pedagógicas con Inspiración Ignaciana.

[12] Sommerville, I. (2005). Ingeniería del Software (7ma Edición). Madrid: Pearson Educación, S.A.