UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

TULIO ENRIQUE LEON ALCALA

Barquisimeto, 2006

UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGIA
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN

DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

Trabajo presentado para optar al grado de
Magíster Scientiarum

Por: TULIO ENRIQUE LEON ALCALA

Barquisimeto, 2006

DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

Por: TULIO ENRIQUE LEON ALCALA

Trabajo de grado aprobado

_____	_____
Ismael Alvaro Muñoz Peralta Tutor	Luis Adelmo Alvarez Díaz

_____
Edgar Ramón González Muñoz

Barquisimeto, ___ de _____ de 2006

DEDICATORIA

A mi esposa Reyna, y a mis hijos Miguel y Oriana, por haberles “robado” gran parte de mi atención...

AGRADECIMIENTO

A Dios, creador del Universo, quien ilumina mis pasos y me impulsa a tratar de ser una mejor persona.

A mi Tutor, profesor Alvaro Muñoz, por su valiosa ayuda en la conducción de esta investigación.

Al profesor Luis Alvarez, por sus enseñanzas, por haberme ayudado cuando tuve problemas durante el Postgrado y por su guía en la revisión del Proyecto de Grado.

Al profesor Edgar González, por sus enseñanzas, y por su ayuda en las ideas y en la revisión del modelado a través de UML.

A los profesores Rómulo Bervins, Leonardo Ponte, Hugo Lara, Nancy Zambrano, Rodolfo Canelón, Maritza Bracho, Daniel Rojas, Ana de González y Ledis Chirinos, por sus enseñanzas durante el Postgrado.

A mi esposa Reyna, por su gran ayuda en el inicio del Proyecto de Grado y durante el desarrollo de este Trabajo.

A mi hijo Miguel, por su ayuda en el desarrollo del Prototipo del Software Educativo.

A mi hija Oriana, por su ayuda en la elaboración de la Presentación para la Defensa.

Al profesor Julio Véliz, Giovanni Torrealba y Glennys Clemant, por el suministro de las Notas de los estudiantes y a la profesora Olga Palma por su ayuda en la Presentación para la Defensa.

A la profesora Eunice Ugel, por su ayuda en el Marco Metodológico.

A mis alumnos de las Clases Complementarias de Programación I, por someterse a las encuestas y evaluaciones.

INDICE GENERAL

Página

INDICE DE CUADROS

Página

INDICE DE GRAFICOS

Página

INDICE DE FIGURAS

Página

INDICE DE TABLAS

Página

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN

DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

Autor: Tulio Enrique León Alcalá

Tutor: Ismael Alvaro Muñoz Peralta

RESUMEN

Las Tecnologías de Información y Comunicación han penetrado profundamente en el campo de la Pedagogía. Una innovadora estrategia pedagógica es el uso de software educativo para ayudar a los estudiantes en temas donde se requiere comprender conceptos abstractos como es el tema de las estructuras de datos en programación. Es por ello que este trabajo de investigación tuvo como objetivo diseñar un modelo de software educativo para la enseñanza de estructuras de datos en programación, haciendo uso de la simulación como estrategia en el proceso de enseñanza - aprendizaje. Para desarrollar el diseño del modelo, en una primera fase diagnóstica, se recolectó información a través de la investigación documental, cuestionario impreso y aplicación de instrumentos de evaluación, cuyo propósito fundamental fue precisar las debilidades conceptuales de los estudiantes acerca de las estructuras de datos y determinar las funcionalidades requeridas para el software educativo. Una vez determinada la factibilidad en una segunda fase, se procedió a desarrollar el modelo apoyándose en una metodología adaptada para el desarrollo de software educativo a partir de la metodología RUP y del Modelo Sistémico de Calidad (MOSCA). Luego, se creó un prototipo de una parte del modelo y se evaluó, concluyendo que con su uso se logra afianzar más los conocimientos sobre el tema tratado. Por lo tanto, finalmente, se recomienda desarrollar e implantar el software educativo propuesto con la finalidad de poner al alcance del docente, del estudiante y del proceso de enseñanza - aprendizaje de la programación, un nuevo elemento que minimice las debilidades y problemas que se presentan a los estudiantes en el logro de sus objetivos instruccionales y en el desarrollo de habilidades y destrezas que se requieren para el logro de los mismos.

Palabras Claves: Modelo de Software, Software Educativo, Estructuras de Datos, Programación, Simulación.

INTRODUCCION

Las Estructuras de Datos se pueden definir como la organización de la información que permite un determinado lenguaje de programación. Son herramientas muy útiles para el manejo de información y pueden enfocarse desde el punto de vista de abstracción de datos. Cada estructura posee sus propias características de almacenamiento y recuperación de los datos. Definir y manipular las estructuras de datos adecuadas a un problema dado y hacer un manejo eficiente de las mismas es uno de los objetivos instruccionales que persigue la asignatura de Programación I en la carrera Ingeniería en Informática de la UCLA (ver Anexo B).

Para Salamó y otros (2001), la motivación de los alumnos para el aprendizaje de la asignatura Programación, es baja debido a la dificultad de los mismos en aprender conceptos abstractos y a la falta de capacidad de trabajo continuo. Pero existen varias formas de enfrentarse desde diferentes puntos de vista al problema de la falta de motivación de los alumnos. A tal efecto, se pudieran describir tres modalidades para evitar todos estos problemas. En primer lugar, se plantea un entorno de comunicación vía Web entre alumnos y profesores. En segundo lugar, un plan de trabajo continuo que, a la vez, permite conocer la evolución de los alumnos. Y finalmente, el empleo de una aplicación de entorno educativo para introducir a los alumnos en el concepto de Estructuras de Datos.

En el caso de la UCLA, existe un entorno de comunicación vía Web entre alumnos y profesores a través de la herramienta SABER del Proyecto Universidad Virtual. (UCLA 2003). Por otro lado, existe un plan de trabajo continuo por medio de las Prácticas Complementarias de Programación I dictadas por el Auxiliar Docente de la asignatura.

Dado lo anterior, la presente investigación tiene como objetivo principal, el diseño de un modelo de software educativo para la enseñanza de estructuras de datos en el área de programación del Decanato de Ciencias y Tecnología de la Universidad Centroccidental “Lisandro Alvarado”.

Capítulo 1, El Problema. Inicia con el planteamiento de la situación que genera el trabajo investigativo, de igual manera se describen los objetivos del mismo, la justificación e importancia, alcances y limitaciones, aspectos importantes que sirven de base y guía en toda investigación.

Capítulo 2, Marco Teórico. Se inicia con los antecedentes y las bases teóricas que apoyan el conocimiento del tema en estudio, se describen conceptos como: Modelo, Construcción de Modelos, Software, Modelo de Software, Software Educativo, Simulación y sus tipos, Estructuras de Datos, etc.

Capítulo 3, Marco Metodológico. Se describen los tópicos metodológicos que condujeron la realización del estudio diagnóstico, se presenta el análisis estadístico de la información recopilada a través de los instrumentos de medición. Seguidamente se presenta un estudio factible de la alternativa de solución que permita el diseño del modelo del software educativo para la enseñanza de estructuras de datos.

Capítulo 4, Propuesta del Estudio. Presenta el diseño del modelo del software educativo en base al estudio diagnóstico y de factibilidad del capítulo anterior, presentando los principales artefactos UML que describen las diversas vistas del sistema en estudio.

Finalmente, el Capítulo 5, Conclusiones y Recomendaciones. Presenta un conjunto de conclusiones obtenidas de la investigación y algunas recomendaciones donde se sugiere el desarrollo del Software Educativo para su posterior implantación en el Decanato de Ciencias y Tecnología de la UCLA.

CAPITULO I

EL PROBLEMA

Planteamiento del problema.

Para Salamó y otros (2001), la asignatura Programación constituye una base fundamental para las diversas carreras de ingeniería. Los posibles enfoques que se pueden utilizar, tanto de contenidos como de método docente, son muy diversos. Las estrategias instruccionales para el aprendizaje de esta asignatura deben tener como objetivo dar una herramienta de ayuda para que el alumno se sienta motivado con la asignatura y amplíe sus horas de estudio, con la finalidad de afianzar los conocimientos, buscando nuevos enfoques de la asignatura. El objetivo prioritario debe ser motivar al alumno a conseguir el máximo rendimiento en la asignatura y a afianzar gradualmente los conocimientos sin depender de la proximidad del examen. El mayor problema para el aprendizaje significativo de la asignatura ha sido encontrar fuentes de motivación para los alumnos. Se observa que la mayoría de los alumnos no tienen un hábito de trabajo constante. Además, debe añadirse el stress que supone la distribución de los exámenes a lo largo del curso. Este problema lleva a nuestros alumnos a abandonar la asignatura prematuramente al considerarla demasiado difícil debido a su falta de hábito en asimilar conceptos abstractos y en su posterior uso en la resolución de problemas. La motivación es un factor muy importante para que los alumnos se preocupen más en adquirir conocimientos y no se desanimen hasta abandonar la asignatura. Como plantea Alvarez (1999), "El curso debe ser agradable y hasta donde se pueda divertido. Esto se aplica a cualquier curso debido a que

mientras mas agradable y divertido lo sea, los estudiantes usarán mas de su tiempo en él y como consecuencia trabajarán mas eficientemente".

Para Vargas y Gutierrez (2004), un aspecto que dificulta el aprendizaje de la Programación es que esta envuelve una serie de conceptos que los alumnos deben dominar aparte de un Lenguaje de Programación en sí. Conceptos tales como: Algoritmos, Estructuras de Control, Estructuras de Datos, etc. (ver Figura 01).

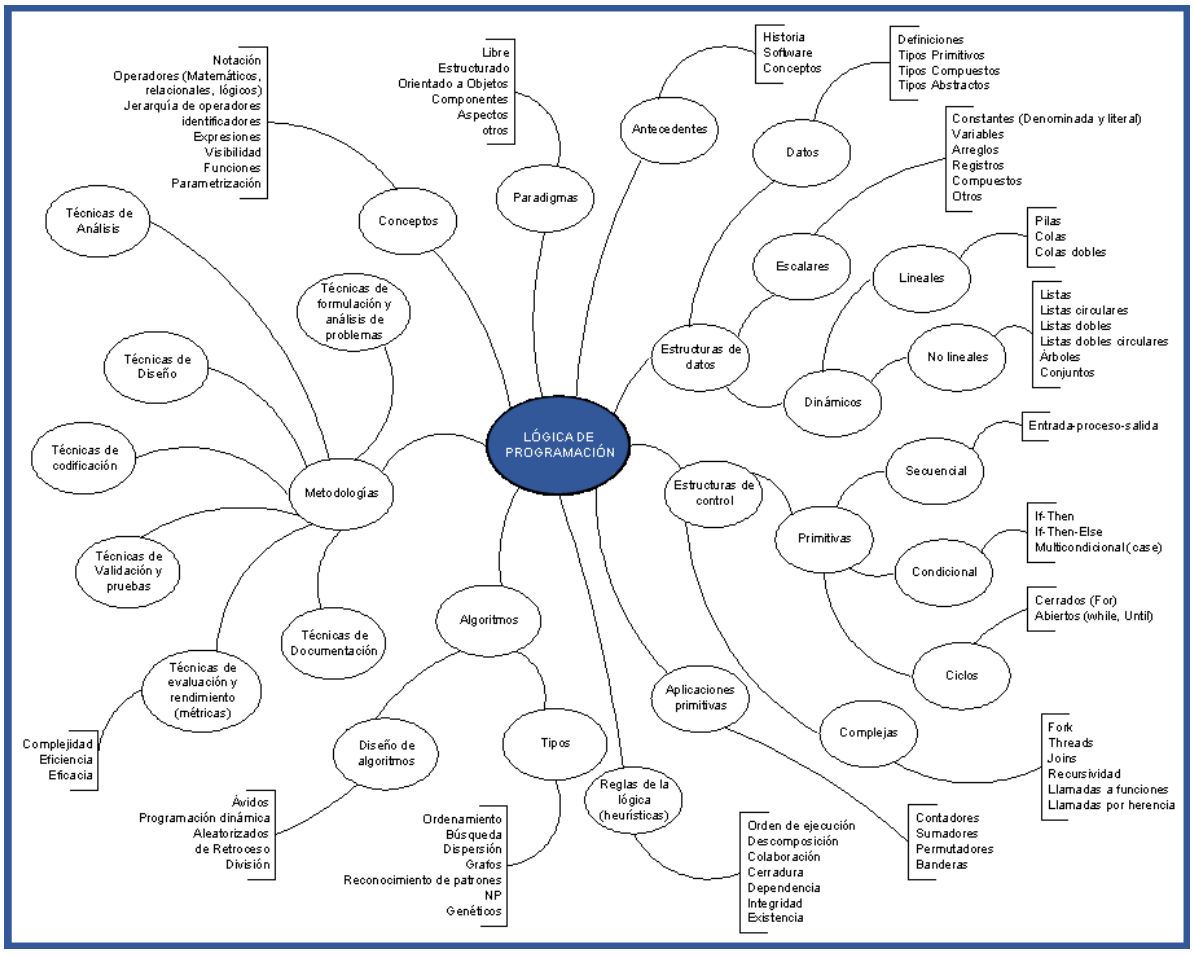


Figura 01: Diagrama Mental de la Lógica de Programación.
Fuente: Universidad del Valle de México
Autor: Vargas y Gutierrez (2004)

Según Boada y otros (2005), se han realizado distintos estudios con el fin de detectar cuales pueden ser las causas del elevado índice de fracaso asociado a las asignaturas de programación. La mayoría de ellos coinciden en que para poder programar deben combinarse una serie de conocimientos y habilidades que la mayoría de los alumnos no han aplicado anteriormente (capacidad de abstracción, cierto nivel matemático, aprendizaje de un nuevo lenguaje, unas nuevas normas de escritura, etc.). Ante la carencia de estos conocimientos el alumno se siente desmotivado, se desanima y va perdiendo el interés por la asignatura. A todos estos factores debemos añadirle el hecho de que generalmente estas asignaturas de introducción a la programación suelen impartirse en los primeros semestres universitarios, los cuales se caracterizan por tener un elevado número de alumnos que dificulta la interacción profesor-alumno vital en este tipo de asignaturas.

Carreón (2004), plantea la siguiente interrogante: ¿Cómo mejorar la enseñanza y el aprendizaje de la programación?, y expresa, existen recomendaciones generales para mejorar la enseñanza y el aprendizaje de la programación, como las de que: éstas sólo pueden apoyarse en el interés y el gusto personal con base en metodologías, como la de aprender haciendo mediante esfuerzos deliberados, tareas bien definidas, niveles apropiados de dificultad según las posibilidades de cada individuo, retroalimentación, y oportunidades de repetición y corrección de errores.

Según Van (2003), autor de "The e-Learning Fieldbook", los alumnos retienen el 90% de lo que hacen, el 70% de lo que dicen o escriben, el 50% de lo que ven y un 10% de lo que leen. Plantea Van que el e-Learning y el computador personal, permiten colocar la simulación como herramienta al alcance de una gran mayoría de alumnos. Actualmente es posible simular el conocimiento potenciando enormemente la retención y comprensión de lo que se aprende.

Aldrich (2004), en su libro “Simulations and the Future of Learning”, un innovador y quizás revolucionario acercamiento al e-Learning, parte de la idea de que

los jóvenes de hoy están aprendiendo a aprender de otro modo, a través de la complejidad, la interactividad y el entretenimiento que ofrecen los videojuegos. Sin embargo, las prácticas educativas siguen siendo las mismas de siempre, con las clases magistrales como principal ejemplo. El autor se pregunta cuál será la mejor manera de educar a estos jóvenes que en poco tiempo formarán parte de las filas de las empresas privadas, el gobierno, las fuerzas armadas, las ONG. La respuesta, claramente, es hacerlo a través de simulaciones, de entornos similares a los de los videojuegos. Como primer paso para llevar a la práctica esta idea es saber cómo es realmente un videojuego. No saberlo nos hace correr el riesgo de apostar a muy poco, o esperar imposibles. Aldrich asegura que de poco sirve leer al respecto, o ver cómo juega otra persona. Por eso, su primera recomendación es sentarse y jugar, para saber qué se siente al hacerlo.

Según Delgado (2004), se llama modelo a la imagen o representación de un sistema, generalmente simplificada e incompleta. Y se llama simulación a la experimentación con un modelo para extraer conclusiones o realizar predicciones. La simulación como método de formación consiste en situar al alumno en un contexto que imite algún aspecto de la realidad (modelo), y en establecer en ese ambiente situaciones similares a las que él deberá enfrentar en su vida profesional, de manera que pueda "experimentar" sin riesgo y extraer conclusiones. Por lo tanto, para reforzar la comprensión de las Estructuras de Datos, los simuladores se deben considerar como herramientas imprescindibles para la facilitación del proceso de enseñanza – aprendizaje.

En la Universidad Centroccidental "Lisandro Alvarado", Decanato de Ciencias y Tecnología, se imparte la asignatura programación I, es común a las diversas carreras: Ingeniería en Informática, Análisis de Sistemas, y Licenciatura en Matemáticas. El objetivo general del programa instruccional está diseñado para suministrar al alumno las herramientas tecnológicas necesarias para el diseño e implementación de programas eficientes y de mediana complejidad, cuyas aplicaciones estén dirigidas a

diversas áreas del ámbito comercial, financiero, científico o doméstico. El objetivo Terminal de la Unidad III contempla "Definir y manipular las estructuras de datos adecuadas a un problema dado y hacer un manejo eficiente de las mismas", mientras que el de la Unidad IV contempla "Adquirir destreza en la manipulación de los algoritmos de clasificación y búsqueda, con aplicación en arreglos" (ver Anexo B).

Por lo tanto, se plantean en esta investigación las siguientes interrogantes:

1. ¿Cuál es el grado de satisfacción de los estudiantes con las enseñanzas recibidas acerca de las estructuras de datos en programación?
2. ¿Existen en la UCLA problemas en el proceso de enseñanza - aprendizaje para estructuras de datos en programación que inciden en el rendimiento de los estudiantes?
3. ¿Cuál será el grado de satisfacción de los estudiantes con el uso de un software educativo para la enseñanza de estructuras de datos en programación?
4. ¿El uso de un software educativo para la enseñanza de estructuras de datos en programación logrará afianzar más estos conocimientos en los estudiantes?

Con el objeto de responder a estas interrogantes, en esta investigación se propone diseñar un modelo de software educativo para la enseñanza de estructuras de datos en programación, haciendo uso de la simulación como estrategia que permita facilitar la comprensión de las mismas. Esta investigación pretende contribuir con los objetivos planteados por la UCLA en uno de sus proyectos como lo es el proyecto "Universidad Virtual", y donde se señala:

> "Facilitar el proceso de enseñanza-aprendizaje a través del uso de nuevas tecnologías de la informática y las telecomunicaciones, sin limitaciones de espacios y horarios". UCLA, (2001).

Objetivos

Objetivo General.

Diseñar un modelo de software educativo para la enseñanza de estructuras de datos en el área de programación del Decanato de Ciencias y Tecnología de la Universidad Centroccidental “Lisandro Alvarado”.

Objetivos específicos.

- Identificar los objetivos instruccionales para el aprendizaje de las estructuras de datos en la unidad curricular de Programación I.
- Diagnosticar los problemas en el proceso enseñanza - aprendizaje sobre estructuras de datos en programación y su incidencia en el rendimiento de los estudiantes.
- Proponer el modelo de software educativo para la enseñanza de estructuras de datos en programación.
- Construir un prototipo del software educativo que permita al estudiante comprender las estructuras de datos a través del ordenamiento de elementos dentro de arreglos presentados como problemas.
- Determinar el grado de satisfacción de los estudiantes en el uso del prototipo de software educativo.
- Verificar si el uso del software educativo permite al estudiante afianzar los conocimientos sobre el algoritmo de ordenamiento de elementos dentro de un arreglo.

Justificación e Importancia

Esta investigación servirá como base para el posterior desarrollo de un Software Educativo que permita ayudar en la comprensión de las Estructuras de Datos en el área de programación a los alumnos de las carreras de Ingeniería en Informática, Análisis de Sistemas y Licenciatura en Matemáticas. De esta manera, se pretenderá bajar el alto índice de aplazados en dicha área, así como mejorar el índice académico y potenciar el conocimiento sobre las Estructuras de Datos, tema común a muchas unidades curriculares dentro de las carreras de Ingeniería en Informática, Análisis de Sistemas y Licenciatura en Matemáticas.

Como establece la misión de la Universidad Virtual, UCLA (2003), se tiene lo siguiente:

> "La necesidad de modernizar y activar el proceso de enseñanza aprendizaje tradicional ha dado origen a una constante búsqueda de herramientas educativas que permitan hacer uso de tecnologías de apoyo a la enseñanza con el fin de optimizar el proceso de aprendizaje. Es así como surge en la Universidad Centroccidental Lisandro Alvarado, UCLA, la propuesta de un nuevo escenario educativo denominado Proyecto de Educación Virtual, donde la combinación de la informática y las telecomunicaciones se presenta como un medio de transmisión poderoso y efectivo para impartir educación a distancia".

Alcance y Limitaciones

El alcance de la presente investigación está circunscrito al diseño del modelo de software para la unidad curricular de Programación I de las carreras Ingeniería en Informática y Análisis de Sistemas en el Decanato de Ciencias y Tecnología de la Universidad Centroccidental "Lisandro Alvarado", existiendo la posibilidad de ser aplicado a otras unidades curriculares que contemplen en su contenido las estructuras de datos. Dado los diferentes tópicos que comprenden las estructuras de datos, y en particular a los arreglos como estructuras de datos, se ha creído conveniente focalizar dicho modelo en el ordenamiento de los elementos de un arreglo a través del algoritmo Burbuja, de tal manera que en un futuro pueda ser extendido a otros tópicos relacionados con arreglos y estructuras de datos en general.

CAPITULO II

MARCO TEORICO

Antecedentes

Morello (1996), elaboró un trabajo con el propósito de implementar las estructuras de datos no primitivas utilizando un lenguaje de alto nivel como es el Lenguaje de Programación Pascal. Este trabajo está adaptado al programa de estudio en la asignatura Programación No Numérica l para la carrera de Ingeniería en Informática y la asignatura Estructuras de Datos para la carrera de Análisis de Sistemas, carreras que se administran en el Decanato de Ciencias y Tecnología de la Universidad Centroccidental "Lisandro Alvarado", por lo tanto es de gran utilidad para estudiantes de las mencionadas carreras, como también para estudiantes de carreras afines.

Como plantea Suárez (1997), al resolver un problema a través del computador se necesita definir la estructura de datos apropiada. Para ello, presentó unas notas donde se da la explicación necesaria para: primero conocer ¿Qué es una estructura de datos?, ¿Cuáles son las estructuras de datos? y ¿Cuándo se utiliza cada una de ellas? También se explica la forma de simular estructuras de datos no implementadas en un lenguaje de programación, usando como base las estructuras de datos existentes (primitivas). Todos los programas mostrados fueron realizados en Turbo Pascal.

León (1998), desarrolló un programa en Pascal para hallar la Solución de Sistemas de Ecuaciones Lineales usando el Método de Gauss y donde plantea:

> "Una de las características más importantes del programa es que permite el Procesamiento paso a paso, de manera que el usuario puede ver como se va ejecutando el Algoritmo paso por paso. Por lo tanto, de una manera didáctica, el programa le ayuda al usuario a comprender mejor el Algoritmo de Eliminación Gaussiana con Sustitución hacia atrás. Con esto, este programa puede ser incluido en el grupo de programas que van a permitir lo que se conoce como Universidad Virtual".

Salamó y otros (2001), diseñaron y desarrollaron una aplicación destinada a reforzar la comprensión de las estructuras lineales de datos, que se consideran como temario imprescindible de la asignatura programación. Esta aplicación, llamada SimEd (Simulador de Estructuras de Datos), permite hacer uso de las estructuras estudiadas, así como experimentar como van evolucionando éstas según se incorpora o se elimina información. Con la finalidad de favorecer su uso, SimEd es accesible desde Internet. Desde un principio, se ha concebido de forma integrada en la plataforma WEB de la asignatura, como una herramienta más que permite reforzar los conocimientos de los alumnos. Como consecuencia inmediata al criterio anterior, para permitir su ejecución en cualquier sistema, se ha desarrollado en lenguaje Java. Otro objetivo importante es la posibilidad de poder hacer un seguimiento del uso que hacen de ella los alumnos. A medida que trabajan con SimEd se va registrando su uso en una base de datos que, a posteriori, permite sacar alguna conclusión estadística sobre la utilidad de la herramienta según el grado de uso que se haga de ella. El propósito inicial del diseño de la aplicación fue trabajar con las estructuras básicas impartidas en la asignatura de programación. No obstante, puede ser ampliada con otras estructuras mucho más complejas tales como multilistas, grafos o árboles. Esto ayudaría a la comprensión de las mismas a alumnos de asignaturas tales como estructuras de datos o programación II.

Martín (2003), diseñó un modelo de programa para el aprendizaje del concepto de Herencia en POO. Básicamente realizó una aplicación en JAVA, que realiza las siguientes tareas: Presentación y breve explicación del significado de la Herencia,

demostración con dos o más ejemplos de herencia. El demo, lleva una explicación paso a paso de todo lo que implica que una clase herede de otra clase. Permite que el usuario controle la demostración (avanzar y retroceder cuando lo desee). El módulo permite la realización de ejercicios, normalmente tipo examen y permite corregir de manera automática las respuestas del alumno, indicándole los aciertos y fallos, y mensajes de explicación. Por otra parte, permite la realización de una pequeña práctica o parte de un programa, en la que el usuario demuestre que ha comprendido y sabe aplicar la herencia en JAVA.

Losada (2003), diseñó e implementó una herramienta educativa que mostraba como se va ocupando y como cambia la memoria con la parte de declaración de variables, con el uso de objetos dinámicos (punteros) y con las instrucciones de asignación. También, Lázaro (2003), desarrolló una herramienta educativa para facilitar el aprendizaje de Java desde el punto de vista del diseño. Esta herramienta ayuda a identificar los conceptos básicos de la programación orientada a objetos (clase, objeto, método, atributo, etc.) a partir del enunciado de los problemas. Para ello se establecen plantillas adecuadas que permiten a los docentes incorporar los mensajes de control que establezcan oportunos en función de las decisiones de los alumnos.

Díaz (2003), propuso una metodología de desarrollo de software educativo bajo un enfoque de calidad sistémica. A partir de una metodología de desarrollo de software del área de la ingeniería, como lo es Rational Unified Process (RUP), se realiza una adaptación y extensión para la construcción de software educativo, a través de un proceso bien definido, en donde se incorporan las mejores prácticas de diseño instruccional y de la ingeniería de software. Esta propuesta analiza y describe las fases para el desarrollo de software educativo a fines de producir un producto educativo de calidad, apoyada en el Modelo Sistémico de Calidad (MOSCA) propuesto por el Laboratorio de Información y Sistemas (LISI), Universidad Simón Bolívar, ampliado y enriquecido con los parámetros educativos propuestos por

profesionales del área de la educación, del gobierno y de la empresa privada, seleccionados para este estudio. MOSCA integra el modelo de calidad del producto y el modelo de calidad del proceso de desarrollo y soporta los conceptos de calidad sistémica. El uso de esta metodología asegura que se produzca desde sus primeras fases de desarrollo, un producto de calidad que cumpla con las características de funcionalidad, usabilidad y fiabilidad, características éstas deseables y necesarias para un material educativo multimedia interactivo. Se elabora además un prototipo de software educativo para niños de 8 a 10 años, para ser usado en Internet, que incorpora la metodología planteada, dentro de un proyecto pedagógico de aula llamado "Conservemos nuestra fauna", conteniendo textos y ejercicios sobre el tema de los animales en peligro de extinción. Este trabajo colabora con el uso de las tecnologías en la educación, donde el estudiante aprende conceptos, practica compresión lectora, busca información y trabaja en equipo. La metodología de desarrollo de software implicó el estudio de varios aspectos, entre los cuales están el diseño instruccional, el diseño técnico y la evaluación de software. Se toma un enfoque ecléctico sobre el uso de las metodologías establecidas por cada teoría de aprendizaje y desarrollo instruccional estudiada en el desarrollo del producto final. El diseño técnico se apoya en los estudios realizados sobre las más recientes investigaciones sobre el uso del color, el texto, la imagen, el sonido y el video. MOSCA consta de 4 niveles (ver Figura 02), los cuales son:

Nivel 0: Dimensiones. Eficiencia del proceso, Efectividad del proceso, Eficiencia del producto y Efectividad del producto son las cuatro dimensiones propuestas en el prototipo de modelo. Sólo un balance y una buena interrelación entre ellas permiten garantizar la calidad Sistémica global de una organización.

Nivel 1: Categorías. Se contemplan 11 categorías: 6 pertenecientes al producto y las otras 5 al proceso de desarrollo.

- Producto: Funcionalidad (FUN), Fiabilidad (FIA), Usabilidad (USA), Eficiencia (EFI), Mantenibilidad (MAB) y Portabilidad (POR).

- Proceso: Cliente-Proveedor (CUS), Ingeniería (ENG), Soporte (SUP), Gestión (MAN) y Organizacional (ORG).

Nivel 2: Características. Cada categoría tiene asociado un conjunto de características (56 asociadas al producto y 27 al proceso de desarrollo), las cuales definen las áreas claves a satisfacer para lograr, asegurar y controlar la calidad tanto en el producto como en el proceso. Entre las características asociadas a cada categoría del producto, se proponen en el modelo MOSCA, una serie de características del proceso. Esto se debe a que algunas características de la calidad del proceso, impactan directamente en las categorías del producto, al igual que ciertas características de la calidad del producto definen categorías del proceso.

Nivel 3: Métricas. La cantidad de métricas asociadas a cada una de las características que conforman MOSCA es de 587 en total.

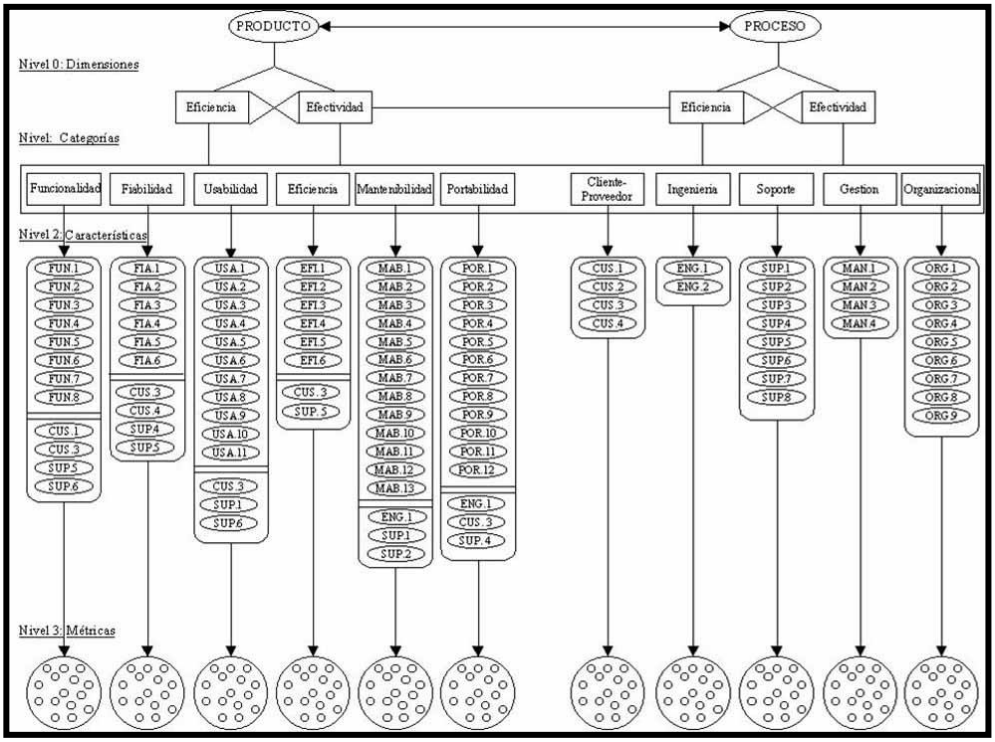


Figura 02: Estructura de MOSCA.
Fuente: Universidad Simón Bolívar Venezuela
Autor: DIAZ, A. y otros (2003)

Adecuación de MOSCA para software educativo, según Díaz (2003):

Dado que MOSCA es un modelo de especificación de la calidad de los sistemas de software, que además permite su medición, se hace una adecuación del mismo para software educativo. Como un primer alcance, se decidió utilizar sólo la Perspectiva Producto, y de esta perspectiva sólo la Dimensión de la Efectividad del Producto; en virtud de que la necesidad más inmediata es dar a las escuelas venezolanas una orientación para su proceso de selección del software educativo a adquirir. La propuesta del modelo de evaluación de calidad de software educativo consiste, entonces, en un conjunto de categorías, características, sub-características y las métricas asociadas (ver Figura 03 y Tabla 01). La estructura del modelo consta de cuatro niveles que se explican brevemente a continuación:

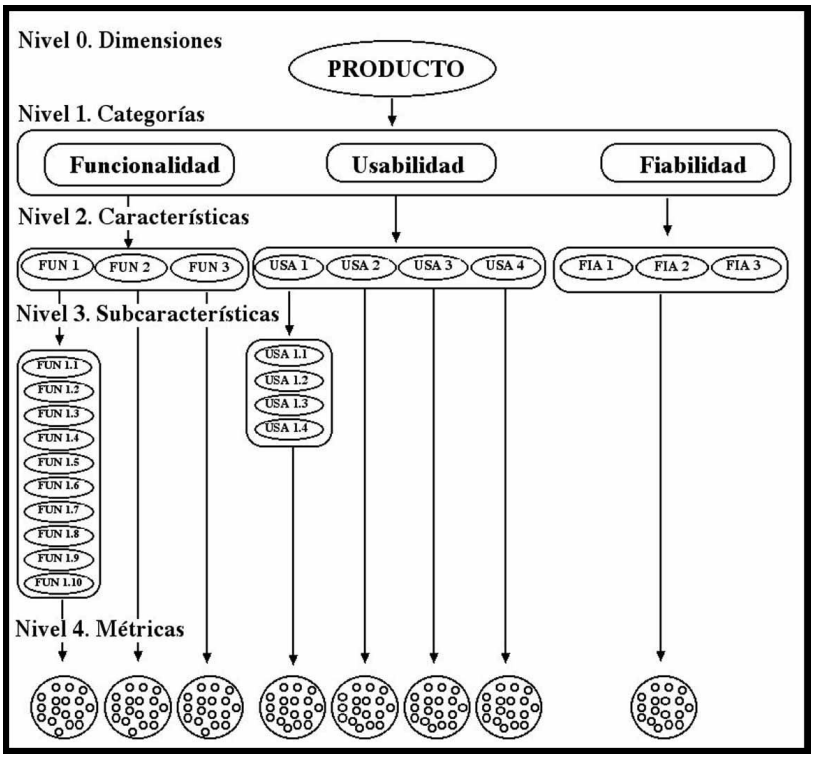


Figura 03: Propuesta del modelo de evaluación de software educativo adaptado de MOSCA
Fuente: Universidad Simón Bolívar Venezuela
Autor: DIAZ, A. y otros (2003)

Perspectiva: Producto

Nivel 0: Dimensiones. Efectividad Producto.

Nivel 1: Categorías. Se contemplan tres categorías:

- Funcionalidad (FUN): Es la capacidad del producto de software para proveer funciones que cumplan con necesidades específicas o implícitas, cuando el software es utilizado bajo ciertas condiciones.

- Usabilidad (USA): Esta categoría se refiere a la capacidad del producto de software para ser atractivo, entendido, aprendido y utilizado por el usuario bajo condiciones específicas.

- Fiabilidad (FIA): La fiabilidad es la capacidad del producto de software para mantener un nivel especificado de rendimiento cuando es utilizado bajo condiciones especificadas.

Como se indicó en la sección anterior, MOSCA consta de seis categorías, de las cuales sólo se deben utilizar tres para la evaluación de software educativo. Debido a que la categoría de Funcionalidad siempre debe estar presente, en esta actividad se seleccionan dos categorías de las cinco restantes del modelo del producto (Usabilidad, Fiabilidad, Eficiencia, Mantenibilidad y Portabilidad). Se seleccionaron Usabilidad y Fiabilidad. La Usabilidad es seleccionada debido a que para que un software educativo motive al aprendizaje, es fundamental que el material educativo sea atractivo y de fácil manejo, debe generar actividades interactivas que motiven y mantengan la atención, actividades que deben ser variadas y que respondan a los diversos estilos de aprendizaje. Se seleccionó Fiabilidad debido a que es importante

que el producto funcione bajo las condiciones establecidas y mantenga un nivel específico de rendimiento para garantizar un ambiente de aprendizaje adecuado.

Nivel 2: Características. Cada categoría tiene asociado un conjunto de características (10 en total). Una vez seleccionadas las categorías que están relacionadas con la evaluación de software educativo (Funcionalidad, Usabilidad y Fiabilidad), se seleccionan las características asociadas a estas categorías en MOSCA, que están relacionadas con el área educativa. Se decidió seleccionar ciertas características asociadas a la efectividad del producto y no a la eficiencia del producto, debido a que, al adquirir un software comercial, no se tiene acceso a los documentos que permiten aplicar las métricas correspondientes a esta dimensión, por lo que no es posible evaluarla.

Nivel 3: Sub-características. Para algunas de las características se asocian un conjunto de sub-características. Para algunas características, tales como 'Ajuste a los propósitos' y 'Facilidad de comprensión del software', se agregó un conjunto de sub-características (14 en total) que añadieron el componente educativo a MOSCA.

Nivel 4: Métricas. Para cada característica se propone una serie de métricas utilizadas para medir la calidad sistémica. Es necesaria una selección de métricas adicionales relacionadas con Funcionalidad, Usabilidad y Fiabilidad, que permitan adaptar MOSCA en el área de software educativo.

En resumen, la propuesta del modelo de evaluación de software educativo consta de un total de 3 categorías, 10 características, 14 sub-características y 276 métricas (ver Tabla 01).

CATEGORÍA	CARACTERÍSTICAS	SUBCARACTERÍSTICAS
FUNCIONALIDAD (FUN)	FUN.1 Ajuste a los propósitos (118)	FUN.1.1 General (6) FUN.1.2 Objetivos de aprendizaje (10) FUN.1.3 Contenidos de aprendizaje (24) FUN.1.4 Actividades de aprendizaje (17) FUN.1.5 Ejemplos (5) FUN.1.6 Motivación (17) FUN.1.7 Retroalimentación (11) FUN 1.8 Ayudas (5) FUN.1.9 Evaluación y registro de datos (11) FUN.1.10 Metodología de enseñanza (12)
	FUN.2 Precisión (4) FUN.3 Seguridad (4)	
USABILIDAD (USA)	USA.1 Facilidad de comprensión (91)	USA.1.1 General (13) USA.1.2 Interactividad (21) USA.1.3 Diseño de la interfaz (34) USA.1.4 Guías didácticas (23)
	USA.2 Capacidad de uso(11) USA.3 Interfaz Gráfica (14) USA.4 Operabilidad (15)	
FIABILIDAD (FIA)	FIA.1 Madurez (11) FIA.2 Recuperación (4) FIA.3 Tolerancia a fallas (4)	
Total de métricas: 276		

Tabla 01: Propuesta de categorías, características, sub-características y número de métricas, para el modelo propuesto basado en MOSCA
Fuente: Universidad Simón Bolívar Venezuela
Autor: DIAZ y otros (2003)

Por otro lado, Morrión (2004), expuso algunas sugerencias para la presentación de materiales educativos en formatos informatizados multimediales. Este diseño didáctico de los materiales educativos debe nutrirse de los avances científicos que se producen en las diferentes disciplinas vinculadas (psicología del aprendizaje, ciencias

de la comunicación, tecnología educativa apropiada, etc.) y, a su vez, responder a las demandas del contexto en que serán utilizados. Ya que la irrupción de las nuevas tecnologías, y muy especialmente las hipertextuales e hipermediales, nos obligan a poner en juego todos nuestros conocimientos y a reestructurar nuestro saber para poder aceptar nuevos desafíos: la generación de cursos que utilicen estos elementos. Más específicamente los cursos denominados a distancia. El trabajo con cursos en hipermedia posibilita la mediatización de la relación docente-alumno. Los estudiantes interactúan con los contenidos de la disciplina a través del curso en hipermedia. Entonces, leen, acceden a las animaciones, simulaciones, videos, sonidos, realizan ejercicios, etc., según el formato decidido para el diseño. Este diálogo se enriquece por el acceso a la información a través de diferentes caminos. Estos cursos pueden brindar la oportunidad de reflexionar y reconocer los propios procesos de aprendizaje (si el curso ofrece la posibilidad de hacerlo). Las limitaciones que suponen estos métodos, no son inherentes a los atributos del medio, sino al diseño pedagógico/didáctico que se elija para el curso.

González (2004), presentó un enfoque novedoso para la evaluación de software educativo: se evalúa para orientar el uso de este tipo de programas por parte de los docentes; el resultado de la evaluación se traduce en una Guía de Uso, en que se consignan los juicios evaluativos, descriptivamente, sin llegar a prescribir formas de uso concretas, sino posibilidades de integración del Software con sentido pedagógico en un currículo o proyecto pedagógico real. Se da énfasis, por tanto, más a los aspectos culturales, ideológicos y valorativos del contenido, que a sus aspectos informáticos o técnicos; se exploran ampliamente las potencialidades pedagógicas, de estructura y metodológicas. Por último, se incorpora un mecanismo de enriquecimiento sucesivo del uso del programa, incorporando a la guía para su difusión los casos más ricos pedagógicamente, descritos por estudiantes y profesores. A continuación se presenta que es lo que se evalúa según este enfoque:

El Programa como Objeto Material:

A. Equipo requerido. Descripción de los requerimientos de equipo mínimos que exige el programa para funcionar; esta información suele aparecer en los folletos que acompañan al CD, medio de almacenamiento ya usual. En la guía de uso se completan los aspectos que hacen referencia a las condiciones de instalación de las instituciones participantes en el proyecto.

B. Usabilidad. Medida en que el sistema es fácil de aprender y fácil de utilizar. Se examinan los siguientes aspectos de usabilidad:

    1. Facilidad de aprendizaje. Medida en que el usuario novel comprende cómo utilizar inicialmente el sistema y cómo a partir de esta utilización llegar a un máximo nivel de conocimiento y uso del sistema. Sus indicadores son: a) Predictivo, los conocimientos adquiridos por el usuario son suficientes para poder determinar los resultados de sus futuras interacciones; b) Sintetizable, habilidad del usuario para evaluar los efectos de las operaciones anteriores al estado actual (capacidad de captar los cambios de estado que produce cada operación); c) Familiar, correlación entre el conocimiento que tiene el usuario y el conocimiento que necesita para una interacción efectiva; d) Consistente, medida en que todos los mecanismos son usados siempre de la misma manera.

    2. Flexibilidad. Multiplicidad de formas en las que el usuario y el sistema intercambian información. Sus indicadores son: a) Iniciativa de diálogo, quien tiene la iniciativa en la conducción del diálogo, hay o no libertad para iniciar cualquier acción en el sistema; b) Diálogo multi-hilo, un hilo de un diálogo es un subconjunto coherente del mismo, si el sistema soporta diálogos multi-hilo al mismo tiempo; c) Migración de tareas, transferencia del control, del sistema al usuario y viceversa, para la ejecución de tareas, medida en que se puede

pasar de una tarea a otra, pasar una a segundo plano o repartirse entre ambas; d) Adaptabilidad, si el sistema puede adaptarse a distintos usuarios.

3. Solidez. Características de la interacción que permiten lograr los objetivos, y su asesoramiento. Sus indicadores son: a) Recuperabilidad, posibilidad del usuario para corregir una acción una vez ha reconocido un error; b) Tiempos de respuesta, tiempo que necesita el sistema para expresar los cambios al usuario; c) Adecuación a las tareas, en qué grado los servicios del sistema soportan todas las tareas que el usuario quiere hacer y la manera en que el usuario las comprende.

4. Mecanismos de soporte. Recursos de ayuda y forma en que el usuario puede utilizarlos. Sus indicadores son: a) Disponibilidad, posibilidad de consultar la ayuda en cualquier momento, sin tener que salir de la aplicación; b) Precisión y detalle, medida en que la ayuda cubre todo el sistema, con concisión; c) Consistencia, en términos de contenidos, terminología y estilo; d) Robustez, que soporte más que el sistema, en términos de funcionamiento; e) Flexibilidad, en que medida permite interactuar de manera adecuada a las necesidades del usuario; f) No obstructiva, que no impida el uso normal de la aplicación; g) Organización del texto de ayuda, lenguaje, longitud de frase y párrafo; cantidad de texto; espacios en blanco; gráficos e iconos.

El Programa como Objeto Pedagógico:

A. Contenido. Se examinan los siguientes aspectos de contenido:

1. Contenido Científico. Se trata de evaluar la calidad y cantidad de la información ofrecida. Sus indicadores son: a) Exactitud y actualidad, fechas de edición, referencias o fuentes citadas, términos técnicos, datos estadísticos,

visión de ciencia, visión de tecnología; b) Adecuación, valor absoluto que se refiere a la significatividad de los contenidos en sí mismos, y valor relativo que se refiere a la adecuación en nivel de tratamiento a la situación pedagógica dada.

2. Contenido socio-cultural e ideológico. Qué representación de la sociedad encierra el programa, cómo representa otras sociedades. Sus indicadores son: a) Visión sociocultural, a qué grupos sociales (o culturales) se refieren los ejemplos, los personajes, los problemas planteados. Qué muestran las ilustraciones, representación racial, género, referencias geográficas, etc.; b) Personajes, reales, imaginarios, sexo, edad, raza, nacionalidad, condición o estado, patronos, obreros, campesinos, militares; c) Marcos espacio-temporales, contexto geográfico (urbano, rural, mar, montaña), medio de referencia (flora, fauna, estaciones), épocas de referencia, medio tecnológico y objetos de la vida cotidiana; d) Contexto social, representación del trabajo, categorías socio-profesionales representadas, familia (composición), habitación (casa, cabaña, finca, conjunto urbano, etc.); e) Situaciones y temas, vida cotidiana (en la casa en la escuela, en el trabajo), situaciones excepcionales (crisis, héroes); f) Ideología implícita, justicia y autenticidad (presentación de los hechos sin distorsión y en perspectiva); g) Valores, contribución a la paz, a la tolerancia, a la formación de actitudes culturales y ecológicas.

3. Contenido pedagógico. Se trata de determinar la adecuación pedagógica de los objetivos y contenidos, frente a los usuarios, su nivel y el programa que están desarrollando. Sus indicadores son: a) Intenciones formativas, lo que pretende el programa, los objetivos de aprendizaje que persigue, explícita o implícitamente; b) Conocimientos previos, si los usuarios dominan los conocimientos previos, en caso que el programa los requiera; c) Niveles de aprendizaje, qué niveles de aprendizaje (hechos, conceptos, principios,

habilidades valores) pretende desarrollar el programa; d) Organización, la progresión del aprendizaje responde a qué tipo de secuencia pedagógica, rígida, espiral o controla por el usuario. En este caso, ¿son necesarias instrucciones o de progreso o es preferible que el usuario encuentre sus propias secuencias?; e) Adecuación curricular, los objetivos y contenidos del programa se pueden integrar con facilidad al curriculum vigente; f) Organizadores y auto evaluación, contiene síntesis (resúmenes), ejercicios (con o sin respuesta), complementos informativos. Contiene evaluaciones, auto-evaluaciones, respuestas razonadas, refuerzo, sistema de seguimiento de logros, evaluación sumativa.

B. Comunicación. Se trata de evaluar la forma del mensaje (significante), es decir el conjunto de recursos que permiten transmitir un mensaje de un emisor a un receptor. Se examinan los siguientes aspectos de comunicación:

1. Sentido de la comunicación. Dirección y control de la interacción programa-usuario Unidireccional, bidireccional, control del usuario sobre la secuencia, mutitareas, multivías.

2. Formas del mensaje. Los aspectos formales de los códigos elegidos (texto, audio, fotos, animación, gráficos, colores) se justifican en sí y frente a la función que se espera de ellos. Sus indicadores son: a) Estética, las formas elegidas son visualmente agradables, manteniendo su sentido comunicativo (no están ahí sólo llenando bellamente espacio); b) Integración, están integrados entre sí los lenguajes verbales y figurativos; c) Innovación, en qué medida son innovadoras las formas de presentación; d) Adecuación, los códigos verbales y figurativos son descifrables por los usuarios, facilitan la comprensión; e) Densidad, la densidad de la información ofrecida (en cada pantalla) es excesiva, adecuada, escasa.

C. Método. Qué metodología, implícita o explícita, contiene el software para la exposición de las ideas, la organización del trabajo, las formas de uso que determina. Se examinan los siguientes aspectos de método:

1. Organización: estructura del manual, forma de exposición y organización de las secuencias. Sus indicadores son: a) Secuencias, se componen de una serie de partes que están presentes regularmente; b) Estructura, el programa es un elemento de enseñanza, de aprendizaje o de enseñanza-aprendizaje; c) Guías o manuales, el programa viene acompañado de un manual para el maestro, el alumno, el usuario en general; d) Elementos de organización interna, el programa incluye instrucciones de empleo, índices, objetivos, léxico, preguntas/ejercicios/, respuestas razonadas, recapitulaciones, evaluaciones; e) Facilitadotes, modo de empleo, índice de materias, lista de objetivos, léxico, referencias, fuentes, plan de capítulos, resúmenes, preguntas, ejercicios, tareas, correcciones control de logro, llamadas; f) Papel del maestro, se limita a dar instrucciones de uso, es necesario para complementar, aclarar o integrar la información; es hacer un seguimiento del uso y de los logros del estudiante; g) Exigencias de aprendizaje, el programa exige principalmente (con mayor frecuencia, como acciones centrales) al estudiante acciones y habilidades para memorizar información, construir conceptos, seguir instrucciones, construir secuencias aprendizaje propias, hacer preguntas, construir respuestas originales, relacionar lo aprendido con otros conocimientos, colaborar con compañeros; h) Distribución de tiempos, un estudiante típico, en una sesión de trabajo normal con el programa, distribuye su tiempo en (% aprox.) a aprender a navegar y buscar información desplazándose por el programa, leer texto, escuchar narración, plantear preguntas al programa, responder preguntas, realizar tareas o ejercicios.

2. Adaptabilidad. En qué medida el software impone obligaciones para su uso: materiales; metodológicas (maestro), pedagógicas (alumno), o es

metodológicamente abierto. Sus indicadores son: a) Materiales, medida en que el software exige el uso de materiales y equipos determinados, implicaciones para la organización del ambiente de aprendizaje; b) Limitaciones metodológicas, el programa impone un método al docente, o éste tiene opción de escoger objetivos, ritmos de trabajo, secuencias; c) Limitaciones para el alumno, El programa ofrece diferentes maneras de entrada, ofrece ejercicios diferentes y graduados según el nivel de los alumnos, posibilidades diferentes de utilización de acuerdo con las necesidades e intenciones del usuario.

Todas las investigaciones referenciadas anteriormente expresan la importancia de disponer de otras herramientas como estrategias del proceso de enseñanza - aprendizaje, utilizando particularmente en algunas de ellas, la simulación. Es por estas razones, que surge esta propuesta del diseño de un Modelo de Software Educativo para la enseñanza de estructuras de datos en programación, haciendo uso de la simulación como estrategia instruccional y que pueda ser aplicada en el área de Programación del Decanato de Ciencias y Tecnología de la UCLA.

Bases Teóricas

Modelo

Según Delgado (2004), se llama modelo a la imagen o representación de un sistema, generalmente simplificada e incompleta. Para Gomes (2002), la representación de un sistema en cualquier otra forma distinta a la propia entidad es llamada modelo. Los modelos pueden asumir formas diversas, desde los modelos físicos y los diagramas, hasta los modelos conceptuales, de los cuales los modelos matemáticos (o cuantitativos) son la expresión más útil para el científico. El concepto de modelo es común a toda la metodología científica. De hecho, en cualquier enfoque aplicado, es a través de modelos que la ciencia se ha expresado para entender la naturaleza de los fenómenos. Los modelos matemáticos, por lo tanto, han satisfecho la función de universalizar el conocimiento, de forma inequívoca. No es por coincidencia que el enfoque sistémico se ha apoyado principalmente en estos tipos de modelos. La complejidad de los sistemas se simplifica en los modelos que los representan, como forma de facilitar la comprensión de su funcionamiento. Aunque se puede elaborar los modelos sin el marco conceptual sistémico, los modelos de sistemas son los más eficaces para aumentar la comprensión de los fenómenos. Así, el enfoque sistémico se puede aplicar en la metodología de la investigación de muchas formas.

Duarte (1997), plantea que el modelo de un sistema usado en una simulación de computador es un programa. Para sistemas complejos, este programa puede ser constituido por muchas subrutinas y tablas interconectadas, y puede no ser posible resumirlo analíticamente como un modelo matemático. Por lo tanto, se debe usar el término *modelo de software* para tales descripciones computarizadas. Estos modelos han tomado un papel cada vez más importante en la toma de decisiones para sistemas complicados.

Construcción de modelos

Para Duarte (1997), básicamente, un modelo debe construirse a partir de datos observados. Los modelos gráficos se construyen a partir de ciertas mediciones. Los modelos matemáticos pueden ser desarrollados siguiendo dos caminos (o una combinación de ellos): Un camino consiste en descomponer el sistema, hablando en figurado, en subsistemas cuyas propiedades se entiendan bien gracias a la experiencia previa. Este camino se conoce como modelamiento y no necesariamente involucra una experimentación sobre el sistema actual. El proceso de modelamiento depende fuertemente de la aplicación y a menudo tiene sus raíces en la tradición y en técnicas específicas del área de aplicación en cuestión. Las técnicas básicas típicamente involucran una estructuración del proceso en diagramas de bloques, cuyos bloques son elementos simples. La reconstrucción del sistema a partir de esos bloques simples se hace cada vez más con el computador, resultando así un modelo de software en lugar de un modelo matemático.

Software

Una de las definiciones formales de Software es dada por el estándar IEEE 729: la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo (IEEE, 1993).

Para Wikipedia (2005), el software se clasifica en dos categorías:

- Software de base o de sistema: Consistente en todo aquel software cuyo propósito es facilitar la ejecución de otro software. Entran en esta categoría:
    - Sistemas operativos.
    - Compiladores.

- Sistemas gestores de bases de datos.
- Etc.

- Software de aplicación: Consistente en aquel software que automatiza un sistema de información, es decir, con relevancia para un fin concreto. Entran en esta categoría:
  - Procesadores de texto.
  - Hojas de cálculo.
  - Etc.

Modelo de Software

Un Modelo de Software es una Simplificación de la Realidad. Los modelos bien elegidos reflejan los aspectos relevantes del sistema y permiten una implementación adecuada. El modelo conceptual se constituye de un conjunto de abstracciones (conceptos) relacionadas con el dominio del problema. Ayuda el ENTENDER el problema y no su resolución. (Universidad de Valparaíso, 2005).

Software Educativo

Para la Revista de divulgación y educación científica AlephZero (2000), el Software Educativo en general se trata de cualquier pieza de software que tiene como objetivo final el de agregar conocimientos a cierto grupo de individuos o a uno en particular. En este sentido se sobreentiende que dicho software educativo debe estar adecuado en cuanto a su estructura, contenido y presentación sobre todo al sector social al que se pretenda llegar. Lo anterior implica que debe tenerse cuidado al diseñar los contenidos, la presentación y muy especialmente, se debe poner atención en el modo de interacción que este software tendrá con el usuario final.

Según Marqués (1996), las expresiones software educativo, programas educativos y programas didácticos se utilizan como sinónimos para designar genéricamente los programas para computadoras creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje. Esta definición engloba todos los programas que han sido elaborados con fin didáctico, desde los tradicionales programas basados en los modelos conductistas de la enseñanza, los programas de Enseñanza Asistida por Computador (EAC), hasta los aun programas experimentales de Enseñanza Inteligente Asistida por Computador (EIAC), que, utilizando técnicas propias del campo de los Sistemas Expertos y de la Inteligencia Artificial en general, pretenden imitar la labor tutorial personalizada que realizan los profesores y presentan modelos de representación del conocimiento en consonancia con los procesos cognitivos que desarrollan los alumnos. No obstante según esta definición, más basada en un criterio de finalidad que de funcionalidad, se excluyen del software educativo todos los programas de uso general en el mundo empresarial que también se utilizan en los centros educativos con funciones didácticas o instrumentales como por ejemplo: procesadores de textos, gestores de bases de datos, hojas de cálculo, editores gráficos. Estos programas, aunque puedan desarrollar una función didáctica, no han estado elaborados específicamente con esta finalidad.

Funciones del Software Educativo:

Marqués (1996), señala que los programas didácticos, cuando se aplican a la realidad educativa, realizan las funciones básicas propias de los medios didácticos en general y además, en algunos casos, según la forma de uso que determina el profesor, pueden proporcionar funcionalidades específicas. Por otra parte, como ocurre con otros productos de la actual tecnología educativa, no se puede afirmar que el software educativo por sí mismo sea bueno o malo, todo dependerá del uso que de él se haga, de la manera cómo se utilice en cada situación concreta. En última instancia su

funcionalidad y las ventajas e inconvenientes que pueda comportar su uso serán el resultado de las características del material, de su adecuación al contexto educativo al que se aplica y de la manera en que el profesor organice su utilización. Las Funciones que pueden realizar los programas son los siguientes:

- Función informativa. La mayoría de los programas a través de sus actividades presentan unos contenidos que proporcionan una información estructuradora de la realidad a los estudiantes. Como todos los medios didácticos, estos materiales representan la realidad y la ordenan. Los programas tutoriales, los simuladores y, especialmente, las bases de datos, son los programas que realizan más marcadamente una función informativa.

- Función instructiva. Todos los programas educativos orientan y regulan el aprendizaje de los estudiantes ya que, explícita o implícitamente, promueven determinadas actuaciones de los mismos encaminadas a facilitar el logro de unos objetivos educativos específicos. Además condicionan el tipo de aprendizaje que se realiza pues, por ejemplo, pueden disponer un tratamiento global de la información (propio de los medios audiovisuales) o a un tratamiento secuencial (propio de los textos escritos). Con todo, si bien el computador actúa en general como mediador en la construcción del conocimiento y el metaconocimiento de los estudiantes, son los programas tutoriales los que realizan de manera más explícita esta función instructiva, ya que dirigen las actividades de los estudiantes en función de sus respuestas y progresos.

- Función motivadora. Generalmente los estudiantes se sienten atraídos e interesados por todo el software educativo, ya que los programas suelen incluir elementos para captar la atención de los alumnos, mantener su interés y, cuando sea necesario, focalizarlo hacia los aspectos más importantes de las actividades. Por lo tanto la función motivadora es una de las características

más resaltantes de este tipo de materiales didácticos, y resulta extremadamente útil para los profesores.

- Función evaluadora. La interactividad propia de estos materiales, que les permite responder inmediatamente a las respuestas y acciones de los estudiantes, les hace especialmente adecuados para evaluar el trabajo que se va realizando con ellos. Esta evaluación puede ser de dos tipos:

  - Implícita, cuando el estudiante detecta sus errores, se evalúa, a partir de las respuestas que le da la computadora.

  - Explícita, cuando el programa presenta informes valorando la actuación del alumno. Este tipo de evaluación sólo la realizan los programas que disponen de módulos específicos de evaluación.

- Función investigadora. Los programas no directivos, especialmente las bases de datos, simuladores y programas constructores, ofrecen a los estudiantes interesantes entornos donde investigar: buscar determinadas informaciones, cambiar los valores de las variables de un sistema, etc. Además, tanto estos programas como los programas herramienta, pueden proporcionar a los profesores y estudiantes instrumentos de gran utilidad para el desarrollo de trabajos de investigación que se realicen básicamente al margen de las computadoras.

- Función expresiva. Dado que las computadoras son unas máquinas capaces de procesar los símbolos mediante los cuales las personas representamos nuestros conocimientos y nos comunicamos, sus posibilidades como instrumento expresivo son muy amplias. Desde el ámbito de la informática que estamos tratando el software educativo, los estudiantes se expresan y se comunican con el computador y con otros compañeros a través de las

actividades de los programas y, especialmente, cuando utilizan lenguajes de programación, procesadores de textos, editores de gráficos, etc. Otro aspecto a considerar al respecto es que las computadoras no suelen admitir la ambigüedad en sus "diálogos" con los estudiantes, de manera que los alumnos se ven obligados a cuidar más la precisión de sus mensajes.

- Función metalingüística. Mediante el uso de los sistemas operativos y los lenguajes de programación, los estudiantes pueden aprender los lenguajes propios de la informática.

- Función lúdica. Trabajar con las computadoras realizando actividades educativas es una labor que a menudo tiene unas connotaciones lúdicas y festivas para los estudiantes. Además, algunos programas refuerzan su atractivo mediante la inclusión de determinados elementos lúdicos, con lo que potencian aún más esta función.

- Función innovadora. Aunque no siempre sus planteamientos pedagógicos resulten innovadores, los programas educativos se pueden considerar materiales didácticos con esta función ya que utilizan una tecnología recientemente incorporada a los centros educativos y, en general, suelen permitir muy diversas formas de uso. Esta versatilidad abre amplias posibilidades de experimentación didáctica e innovación educativa en el aula.

Simulación

Según Salas (1995), el empleo de la simulación permite acelerar el proceso de aprendizaje y contribuye a elevar su calidad. No puede constituir un elemento aislado del proceso docente, sin un factor integrador, sistémico y ordenado de dicho proceso. Su utilización debe tener una concatenación lógica dentro del plan calendario de la

asignatura que se corresponda con las necesidades y requerimientos del plan de estudio y de los programas analíticos de las diferentes asignaturas. En el empleo de la simulación se requieren determinados requisitos, entre los cuales tenemos: a) Elaboración de guías orientadoras para los educandos y guías metodológicas para los profesores de cada tipo de simulación (y simulador) que se empleen, que contenga una definición clara de los objetivos a lograr. b) Demostración práctica inicial a los educandos por parte del profesor, que contenga su introducción teórica, donde se puedan emplear otros medios de enseñanza de forma combinada. c) Ejercitación del educando de forma independiente. d) Evaluación por el profesor de los resultados alcanzados por cada estudiante de forma individual.

Para Delgado (2004), se llama simulación a la experimentación con un modelo para extraer conclusiones o realizar predicciones. La simulación como método de formación consiste en situar al alumno en un contexto que imite algún aspecto de la realidad (modelo), y en establecer en ese ambiente situaciones similares a las que él deberá enfrentar en su vida profesional, de manera que pueda "experimentar" sin riesgo y extraer conclusiones. Se pueden distinguir dos enfoques de concepción en este tipo de simuladores utilizados en aula:

- Por un lado, los simuladores basados en PC y en aplicaciones informáticas en las que se introducen modelos que incorporan un amplio conjunto de reglas que tratan de reflejar la realidad empresarial. Guardan similitudes con los sistemas tipo "caja negra": Los participantes toman una serie de decisiones (Planificación, costes, estilo de liderazgo con el equipo de proyecto, subcontratación), las introducen en el simulador, y se originan unos resultados en el cuadro de mando del proyecto. Habitualmente, el trabajo de los participantes es tratar de entender qué errores han cometido en sus decisiones teniendo en cuenta los resultados obtenidos; no obstante, no hay un mecanismo directo que permita ligar los resultados pobres a las decisiones incorrectas. Lo que se busca no es que identifiquen las reglas que implementa

el simulador, sino que construyan su propio "modelo mental" para entender la lógica de los proyectos.

- Por otro, los simuladores basados en apoyos visuales (tableros, fichas, paneles, cuadros de control, etc.) donde las reglas de simulación son relativamente sencillas. Se usan para representar la lógica de un proyecto o de una función concreta de su gestión. Este tipo de simuladores presenta ventajas y desventajas con respecto a las simulaciones basadas en PC. El disponer de un panel que incorpora los elementos más relevantes en la realidad del proyecto simplifica el proceso de "arranque" y de familiarización con la simulación, y por otro lado, se vincula más fácilmente la simulación a la realidad del propio proyecto. El "feedback" necesario para aprender de la simulación es mucho más directo y objetivo con este tipo de simuladores. Efectivamente, al no existir modelos de simulación complejos (que requerirían del uso de un PC), las reglas de la simulación son claras y permiten relacionar los resultados que se obtienen con las decisiones tomadas. El trabajo y la interacción del grupo en este tipo de simulaciones son más fáciles; la discusión del grupo o equipo sobre un panel donde se está reflejando la realidad de su proyecto es más atractiva que sobre una pantalla de computador, cuyas limitaciones son evidentes. Por otro lado, en la medida en que se introduzcan dinámicas de liderazgo en el aula, se podrá explotar mejor la riqueza en la reflexión y en las conclusiones extraídas.

Plantea Delgado (2004), que la simulación tiene dos grandes aplicaciones en el proceso formativo:

- Durante la enseñanza-aprendizaje: los diversos tipos de simulación disponibles pueden utilizarse no sólo para la mejora de las técnicas de evaluación y planificación, sino también para analizar el impacto en el

rendimiento del estudiante de la toma de decisiones relacionada con el desarrollo del proceso de aprendizaje, su motivación, ó el tratamiento de la resolución de conflictos en las relaciones humanas, donde en ocasiones pueden ser más eficaces que muchos métodos tradicionales.

- En la evaluación: además de mejorar la percepción global del asistente debido a la amenidad de la metodología, permite objetivar en qué medida ha sido capaz de extraer conclusiones que le permitan aplicar, de manera práctica, los conceptos tratados en el modelo.

Estructuras de Datos

Según Microsoft MSDN (2004), Las Estructuras de Datos se pueden definir como la organización de la información que permite un determinado lenguaje de programación. Cada estructura posee sus propias características de almacenamiento y recuperación de los datos. Para Wikipedia (2005), en programación, una Estructura de Datos es una forma de organizar un conjunto de datos elementales (un dato elemental es la mínima información que se tiene en el sistema) con el objetivo de facilitar la manipulación de estos datos como un todo y/o individualmente. Una estructura de datos define la organización e interrelación de estos, y un conjunto de operaciones que se pueden realizar sobre él. Las operaciones básicas son: Agregar un nuevo valor a la estructura, eliminar un valor de la estructura y encontrar un determinado valor en la estructura para realizar una operación con este valor, en forma SECUENCIAL o BINARIO (siempre y cuando los datos estén ordenados), esto se denomina búsqueda. Otras operaciones que se pueden realizar son: Ordenamiento, de los elementos pertenecientes a la estructura y apareo, dadas dos estructuras originar una nueva ordenada y que contenga a las apareadas. Algunas estructuras de datos utilizadas en programación son:

- Arrays (Arreglos)
    - Vectores
    - Matrices

- Listas Enlazadas
    - Listas Simples
    - Listas Dobles
    - Listas Circulares
    - Listas por saltos (Skip lists)

- Pilas (stack)

- Colas (queue)
    - Colas de Prioridad

- Árboles

En Wikipedia (2005), un arreglo (más correctamente denominado vector), es un conjunto de variables o registros del mismo tipo que pueden estar almacenados en memoria principal o en memoria auxiliar. Los array de 1 dimensión se denominan vectores, los de 2 o más dimensiones se denominan matrices. La forma de acceder a los elementos del array es directa; esto es, el elemento deseado es obtenido a partir de su índice. Pueden tener tantas dimensiones como se deseen, aunque los anteriormente expuestos son los más comunes. En lenguajes compilados y en la mayoría de máquinas virtuales, la representación interna de un array suele ser en una dimensión: es decir, un conjunto consecutivo de celdas de memoria, independientemente de la dimensión. La representación de un elemento en un vector se suele hacer mediante el identificador del vector seguido del índice entre paréntesis, corchetes o llaves; vector [*índice*], vector (*índice*), o bien vector {*índice*}. Por otro lado, la página de Internet MailxMail (2005), refiere que las Estructuras de Datos de acuerdo a como ocupan la

memoria pueden ser divididas en estáticas y dinámicas. Las Estructuras de Datos estáticas son aquellas en las que el espacio ocupado en memoria se define en tiempo de compilación y no puede ser modificado durante la ejecución del programa. Corresponden a este tipo los arrays y registros. Las Estructuras de Datos Dinámicas son aquellas en las que el espacio ocupado en memoria puede ser modificado en tiempo de ejecución. Corresponden a este tipo las listas, árboles y grafos. Estas estructuras no son soportadas en todos los lenguajes. La elección de la estructura de datos idónea dependerá de la naturaleza del problema a resolver y, en menor medida, del lenguaje.

Ingeniería de Software

Para Cota (1994), según la definición del IEEE, citada por Lewis (1994) "software es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo". Según el mismo autor, "un producto de software es un producto diseñado para un usuario". En este contexto, la Ingeniería de Software (SE del inglés Software Engineering) es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software", que en palabras más llanas, se considera que "la Ingeniería de Software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software", es decir, "permite elaborar consistentemente productos correctos, utilizables y costo-efectivos".

Jacobson (1998), plantea que el proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad". El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en

diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo".

Según Zavala (2000), el proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición. La concepción define el alcance del proyecto y desarrolla un caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

RUP

El Proceso Unificado "es un proceso de desarrollo de software configurable que se adapta a través de los proyectos variados en tamaños y complejidad. Se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos en el desarrollo de software de misión crítica en una variedad de industrias por la compañía Rational", donde confluyen 'los tres amigos' como se llaman a sí mismos o los tres grandes OO: Grady Booch, James Rumbaugh e Ivar Jacobson. (M&R, 1998). Para Jacobson (1998), el Proceso Unificado es un proceso porque "define quién está haciendo qué, cuándo lo hace y cómo alcanzar cierto objetivo, en este caso el desarrollo de software". Y para Booch (1998), los conceptos clave del Proceso Unificado son:

Fase e iteraciones	¿Cuándo se hace?
Flujos de trabajo de procesos (actividades y pasos)	¿Qué se está haciendo?
Artefactos (modelos, reportes, documentos)	¿Qué se produjo?
Trabajador: un arquitecto	¿Quién lo hace?

Para Zavala (2000), el Proceso Unificado guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica lo más pronto, para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura. El proceso describe qué entregables producir, cómo desarrollarlos y también provee patrones. El proceso unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

UML

Jacobson (1998), expresa que, el Lenguaje de Modelado Unificado UML es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software. El desarrollo de sistemas con UML siguiendo el proceso unificado incluye actividades específicas, cada una de ellas a su vez contienen otras sub-actividades las cuales sirven como una guía de cómo deben ser las actividades desarrolladas y secuenciadas con el fin de obtener sistemas exitosos; consecuentemente el desarrollo de los sistemas puede variar de desarrollador en desarrollador, de proyecto en proyecto, de empresa en empresa adoptando siempre un Proceso de Desarrollo.

Operacionalización de las Variables

Variables Conceptuales

- Grado de satisfacción de enseñanzas recibidas: Es el grado de satisfacción de los estudiantes con las enseñanzas recibidas acerca de las estructuras de datos en programación.

- Prototipo Simulador: Es un software que permite a los estudiantes interactuar a través de la simulación para ejercitar con las Estructuras de Datos.

Variables Operacionales

Para efectos de la operacionalización de las variables conceptuales se procedió de la siguiente manera:

La variable -Grado de satisfacción de enseñanzas recibidas- se operacionalizó considerando dos dimensiones, la dimensión Estructuras de Datos y la dimensión Arreglos.

La dimensión Estructuras de Datos está constituida por la base teórica sobre las estructuras de datos, considerando en este caso, sólo el concepto. La dimensión Arreglos está constituida por todos los elementos que constituyen la base teórica sobre Arreglos: concepto, dimensión, declaración, tipo de dato del subíndice, tipo de dato del arreglo, acceso a los elementos, determinación del número de elementos.

La variable -Prototipo Simulador- se operacionalizó considerando dos dimensiones, la dimensión Apreciación y la dimensión Proceso de Enseñanza – Aprendizaje en algoritmo Burbuja.

La dimensión Apreciación está constituida por los elementos del prototipo que pueden ser apreciados en el proceso de enseñanza - aprendizaje, estos son: aprendizaje, motivación, ayuda, uso, interface. La dimensión Proceso de Enseñanza – Aprendizaje en Algoritmo Burbuja está constituida por todos los elementos que permiten determinar el grado de fortalecimiento del conocimiento sobre el algoritmo: definición del ciclo externo, definición del ciclo interno, comparación de los elementos, intercambio de los elementos, proceso iterativo de ordenamiento.

En el Anexo D se presenta la operacionalización de las variables en estudio.

CAPITULO III

MARCO METODOLOGICO

En toda investigación científica, es necesario que los eventos estudiados, así como los vínculos que se establecen entre estos, los resultados obtenidos y las evidencias significativas encontradas en relación con el problema investigado, además de los nuevos conocimientos que es posible aportar, reúnan las condiciones de fiabilidad, objetividad y validez interna; para lo cual se requiere delimitar los procedimientos de orden metodológico, a través de los cuales se intenta dar respuestas a las interrogantes objeto de investigación.

Por consiguiente, el marco metodológico de la presente investigación donde se propuso estudiar el diseño de un modelo de software educativo para la enseñanza de estructuras de datos en programación; es la instancia que alude al momento tecno-operacional presente en todo proceso de investigación; donde se sitúa en detalle, el conjunto de métodos, técnicas y protocolos instrumentales que se emplearon en el proceso de recolección de los datos requeridos en la investigación propuesta.

En este sentido, este capítulo desarrolla aspectos metodológicos relativos al tipo de estudio y su diseño de investigación, incorporados en relación a los objetivos establecidos, el universo o población estudiada, así como el número total de sujetos que la integran; la muestra que se utilizó y como fue seleccionada; las técnicas e instrumentos empleados en la recolección de los datos incluyendo sus características; las formas de codificación, presentación de los datos; y el análisis e interpretación de los resultados que permitieron las conclusiones y el diseño del modelo del software educativo para la enseñanza de estructuras de datos en programación.

Naturaleza del Estudio

El trabajo que se presenta es de tipo Proyectos especiales**,** en esta modalidad se encuentran aquellos proyectos conducentes a creaciones tangibles, tales como el desarrollo de software, prototipos o productos tecnológicos en general. Se espera que el tipo de trabajo propuesto corresponda con el perfil profesional del futuro egresado. (Manual para la Presentación del Trabajo conducente al Grado Académico de: Especialización – Maestría – Doctorado de la Universidad Centroccidental Lisandro Alvarado, 2002). En estos casos, el estudiante debe demostrar la necesidad de la investigación, además del aporte que con su producto hace al desarrollo científico - tecnológico del área de desarrollo le la aplicación. Este tipo de investigación se identifica, entre otros, como el desarrollo de Sistemas Multimedios y/o Expertos que sirven de apoyo al material instruccional de la Universidad Centroccidental "Lisandro Alvarado", para aquellas asignaturas que por su complejidad, son de difícil comprensión por el estudiante.

Fases del Estudio

En atención a esta modalidad de investigación, se ejecutaron dos grandes fases en el estudio, a fin de cumplir con los requisitos involucrados en un Proyecto Factible. En la primera de ellas, se desarrolló el conocimiento de la situación existente en la realidad objeto de estudio, a fin de describir la situación respecto al grado de satisfacción de los estudiantes acerca de las enseñanzas recibidas sobre las estructuras de datos y de los conocimientos que poseen sobre las mismas, y por otro lado, las funcionalidades requeridas por el software educativo para la enseñanza de estructuras de datos en programación. En la segunda fase de la investigación y atendiendo a los resultados de la fase diagnóstica, se formuló el modelo operativo

propuesto, para dar respuestas a los problemas planteados inicialmente en la investigación.

Fase Diagnóstica

Tuvo como objetivo fundamental precisar las debilidades conceptuales de los estudiantes en el proceso de enseñanza - aprendizaje de estructuras de datos (una vez visto el tema en clase), esto permitió enfatizar en algunos aspectos del contenido del programa a los cuales se les brindó mayor interés en el momento de diseñar el modelo. Este diagnóstico a su vez facilitó el desarrollo de las fases siguientes, es decir el estudio de factibilidad y el diseño de la propuesta del modelo para la solución del problema planteado.

Para completar esta fase, fueron elaborados cuatro instrumentos de medición que permitieron la recolección de los datos, los cuales fueron sometidos a evaluación por un juicio de expertos a fin de confirmar la confiabilidad y validez del mismo. Luego, se procedió a la interpretación de los resultados a través de un Análisis Estadístico: medidas de tendencia central, distribución de frecuencias y promedios.

Complementariamente para mejorar la fiabilidad en el diagnóstico y la necesidad de justificar la propuesta del modelo, se revisaron los resultados de las calificaciones obtenidas por los alumnos en los últimos cinco años en la asignatura Programación I, tanto en Ingeniería en Informática como en Análisis de Sistemas; se tomó previsión de exceptuar las notas de aquellos alumnos que cancelaron el semestre. Estos datos fueron revisados estadísticamente con el programa estadístico SPSS para Windows y fueron presentados en tablas de distribución de frecuencias, promedios y porcentajes.

En base a estos análisis se procedió a presentar una propuesta del modelo de software educativo para la enseñanza de estructuras de datos en programación.

Universo y Muestra

El Universo y Muestra, según Arnau (1980), "Se refiere a un conjunto de elementos, seres o eventos concordantes entre sí en cuanto a una serie de características, de las cuales se desea obtener alguna información.". Para Hurtado (2000), se le denomina también a las características compartidas por los integrantes de la población o "criterio de inclusión", por lo tanto, "La población de una investigación está constituida por el conjunto de seres en los cuales se va a estudiar la variable o evento, y que además comparten, como características comunes, los criterios de inclusión". Y también, se refiere al contexto, ser o entidad poseedoras de las características, evento, cualidad o variable, que se desea estudiar.

Para el desarrollo de esta investigación se utilizó como población los alumnos inscritos en las Prácticas Complementarias de la asignatura Programación I, tanto de la carrera Ingeniería en Informática como de Análisis de Sistemas del Decanato de Ciencias y Tecnología de la UCLA (138 alumnos). La muestra se obtuvo por muestreo probabilístico determinada estadísticamente (ver Anexo E), estando representada por 85 alumnos con un intervalo de confianza del 95% y además fue dividida intencionalmente en dos grupos: un Grupo Estudio (45 alumnos) y un Grupo Control (40 alumnos). Tanto al Grupo Estudio como al Grupo Control le fueron aplicados el Cuestionario 01 y la Evaluación 01, luego se les suministró la información con respecto al algoritmo de la burbuja mediante una clase tradicional. Posterior a la clase, al Grupo Control le fue aplicada la Evaluación 02. Al Grupo Estudio, además de la clase tradicional, se les dio la oportunidad de ejercitar durante 15 minutos con el prototipo del software simulador de ordenamiento de elementos

dentro de un arreglo (ver Anexo S), luego le fueron aplicados el Cuestionario 02 y la Evaluación 02 (ver Tabla 02).

		GRUPO	
Pasos	**Descripción**	**ESTUDIO**	**CONTROL**
1	Aplicación del Cuestionario 01	√	√
2	Aplicación de la Evaluación 01	√	√
3	Aplicación de la Clase Tradicional	√	√
4	Ejercitación con el Prototipo	√	
5	Aplicación del Cuestionario 02	√	
6	Aplicación de la Evaluación 02	√	√

Tabla 02: Aplicación de los instrumentos de medición según Grupo
Fuente: Autor de la Investigación

Procedimiento

Se llevó a cabo el diseño y desarrollo de un modelo destinado a ser utilizado como estrategia para el aprendizaje de estructuras de datos en programación. Fue necesario primeramente recopilar cierta información que sirvió de base para la elaboración del modelo. Se utilizó como método de recolección la revisión bibliográfica, lo cual permitió reconocer conceptos, características y especificaciones del software educativo y las características de calidad requeridas. Por otro lado, se emplearon instrumentos de medición como cuestionarios y evaluaciones para indagar sobre el nivel de conocimientos que tienen los estudiantes sobre estructuras de datos en programación.

En el siguiente aparte se listan los pasos de la presente investigación:

1. Elaborar los instrumentos de medición.
2. Determinar la validez de los instrumentos.
3. Aplicar los instrumentos de medición.
4. Determinar la confiabilidad de los instrumentos.
5. Analizar los datos recabados por los instrumentos.
6. Presentar las conclusiones del diagnóstico.
7. Efectuar el análisis de factibilidad.
8. Elaborar la propuesta.

Técnicas e Instrumentos de Recolección de Datos

El plan de recolección de datos se llevó a cabo en las siguientes etapas:

Primera Etapa. Consistió en el diseño de cuatro instrumentos de medición, dichos instrumentos se describen a continuación:

El primer instrumento, el cual se muestra en el Anexo F, es el Cuestionario 01, contentivo de ocho (8) ítem con escalamiento de tipo Lickert, valoradas según una escala de cinco (5) puntos de acuerdo a las categorías: Muy Bueno (1), Bueno (2), A Medias (3), Un Poco (4), Nada (5). Su objetivo fue determinar los aspectos académicos relacionados con el logro de objetivos para la enseñanza de estructuras de datos en programación, para ello se recogió información sobre el grado de satisfacción de los estudiantes acerca de las enseñanzas recibidas en el área de estructuras de datos, particularmente con respecto a los arreglos.

El segundo instrumento, el cual se muestra en el Anexo G, es la Evaluación 01, prueba estructurada mixta con ítems de selección, identificación y respuestas breves, contentiva de ocho (8) preguntas, cuatro (4) de ellas de tipo Verdadero – Falso, tres (3) de tipo Respuesta y una (1) de tipo identificación. Su objetivo fue determinar las funcionalidades asociadas con un software educativo para la enseñanza de estructuras de datos en programación, para ello se determinó el grado de conocimientos de los estudiantes en el área de estructuras de datos.

El tercer instrumento, el cual se muestra en el Anexo H, es el Cuestionario 02, contentivo de ocho (8) ítem con escalamiento de tipo Lickert, valoradas según una escala de cinco (5) puntos de acuerdo a las categorías: Totalmente de Acuerdo (1), De Acuerdo (2), Indeciso (3), En Desacuerdo (4), Totalmente en Desacuerdo (5). Su objetivo fue determinar el grado de satisfacción que encuentran los estudiantes en el prototipo del software simulador de ordenamiento de elementos dentro de un arreglo (ver Anexo S).

El cuarto instrumento, el cual se muestra en el Anexo I, es la Evaluación 02, contentiva de dos (2) partes de tipo solución de problemas con respuesta abierta, en la primera parte los estudiantes debían completar el procedimiento de ordenamiento Burbuja con el código requerido, en la segunda parte los estudiantes debían completar el orden de los elementos del arreglo según las iteraciones del proceso. Su objetivo fue evaluar el conocimiento de los alumnos tanto del Grupo Estudio, que ejercitaron con el software simulador de ordenamiento de elementos dentro de un arreglo, y del Grupo Control que no ejercitaron, y de esta forma poder relacionar los resultados de la evaluación en ambos grupos.

Segunda Etapa. Validez de los instrumentos. La validez de los instrumentos se hizo a través del juicio expertos, quienes pudieron determinar la relación que existe entre los objetivos del estudio, las variables, dimensiones e indicadores que lo conforman y los

ítems que contienen los instrumentos. A cada uno de los expertos se les entregó una carpeta la cual contenía: a) una hoja de solicitud de validación (ver anexo J), b) un ejemplar preliminar de los 2 cuestionarios y las 2 evaluaciones (ver Anexos F, G, H, I), conjuntamente con el plan de operacionalización de las variables en estudio (ver Anexo D), c) un formato para la evaluación de la pertinencia, claridad y congruencia de cada ítem en relación a las dimensiones e indicadores definidos. Para opinar sobre dichos aspectos los expertos debían seleccionar en los recuadros según la dicotomía Si / No, además de escribir sus observaciones, opiniones y sugerencias para cada ítem y para los instrumentos en general, (ver Anexo K).

Los expertos (intrajueces) estuvieron integrados por Alvaro Muñoz, Doctor en Educación a Distancia y Tecnología Instruccional, profesor adscrito al Decanato de Ciencias y Tecnología, Reyna Figueras, MSc. en Educación, profesora adscrita al Decanato de Medicina, y Eunice Ugel, profesora en el área de Metodología de la Investigación, adscrita al Decanato de Medicina.

Tercera Etapa. Análisis de los resultados de la validación. Se analizaron los resultados de la validación de los instrumentos, para los dos cuestionarios se procedió a realizar las modificaciones de forma y fondo que permitieron mejorar la calidad y el logro de los objetivos percibidos por dicho instrumento, dichos cambios se realizaron de acuerdo a las recomendaciones de los expertos. Para las evaluaciones, se sugirió por parte de los expertos, reformular las preguntas para que los estudiantes fueran más concisos en sus respuestas.

Cuarta Etapa. Aplicación de los instrumentos. En la aplicación de los instrumentos se realizaron las siguientes actividades: 1) Se aplicó el Cuestionario 01 y la Evaluación 01 a los estudiantes asistentes a las Prácticas Complementarias de Programación I, tanto del Grupo Estudio como al Grupo Control. 2) Una vez

desarrollado y utilizado el prototipo del software simulador (ver Anexo S), se aplicó el Cuestionario 02 al Grupo Estudio. 3) Se aplicó la Evaluación 02 tanto al Grupo Estudio como al Grupo Control.

Quinta Etapa. Análisis de los datos. Se recolectaron y se transcribieron los datos al programa estadístico SPSS para Windows y se generaron los resultados. Estos resultados generados por el programa, fueron analizados según la Operacionalización de Variables, lo que permitió obtener información adecuada como ayuda para describir las funcionalidades asociadas con un modelo de software educativo para la enseñanza de estructuras de datos en programación.

A continuación se muestran los resultados obtenidos durante la investigación.

Resultados

Los resultados de esta investigación son presentados en dos fases: una primera fase donde se muestran los resultados de la fase diagnóstica, los cuales nos demuestran las debilidades de la enseñanza tradicional de las estructuras de datos, y nos permitieron el abordaje de la información necesaria para la elaboración del modelo de software educativo, el cual se presenta en la segunda fase. Ambas fases están relacionadas en la evaluación del proyecto, puesto que una vez realizado el modelo, se desarrolló un prototipo del software simulador de ordenamiento de elementos dentro de un arreglo (ver Anexo S), que fue aplicado y evaluado como parte del procedimiento.

Cuadro 01: Distribución de las notas de los alumnos según Lapso de Estudio

Lapsos	Aprobados		Aplazados		T O T A L	
	N°	%	N°	%	N°	%
2000-1	142	39.01	222	60.99	364	100.0
2000-2	187	49.87	188	50.13	375	100.0
2001-1	166	45.11	202	54.89	368	100.0
2001-2	150	36.50	261	**63.50**	411	100.0
2002-2	157	**61.33**	99	38.67	**256**	100.0
2003-1	181	41.90	251	58.10	**432**	100.0
2003-2	174	40.56	255	59.44	429	100.0
2004-1	144	40.91	208	59.09	352	100.0
2004-2	162	46.15	189	53.85	351	100.0
2005-1	157	39.95	236	60.05	393	100.0
TOTAL	1620	43.42	2111	**56.58**	3.731	100.0

Se revisaron las calificaciones finales de la asignatura programación I en 5 años, lo cual representó 10 lapsos, para un total de 3.731 veces que fue cursada la asignatura por estudiantes, de las cuales el 56.58 % fueron aplazados. Se observa que, a excepción del lapso 2002-2, el porcentaje de aplazados fue siempre mayor que el porcentaje de aprobados, siendo el porcentaje mayor de aplazados el del lapso 2001-2 (63.50%). Luego en el siguiente lapso (2002-2) se evidencia un cambio donde el porcentaje de aprobados (61.33%) superó el porcentaje de aplazados, es de hacer notar que en este lapso se registra el menor número de estudiantes inscritos (256), mientras que en el siguiente lapso 2003-1 se registra el mayor número de alumnos inscritos (432).

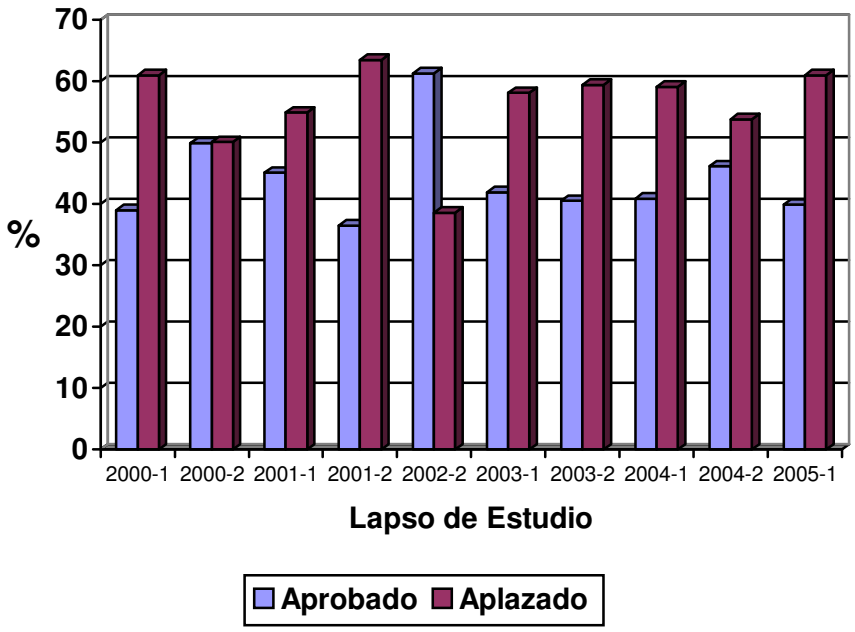


Gráfico 01: Distribución de las notas de los alumnos según Lapso de Estudio

En casi la totalidad de los lapsos estudiados prevalece un mayor porcentaje de estudiantes aplazados contra los estudiantes aprobados.

Cuadro 02: Distribución de las notas de los alumnos según su condición

Condición	Aprobados		Aplazados		T O T A L	
	N°	%	N°	%	N°	%
0	801	40.70	**1176**	**59.30**	1977	100.0
1	485	47.40	**537**	**52.60**	1022	100.0
2	264	**57.20**	197	42.80	461	100.0
>=3	70	25.80	**201**	**74.20**	271	100.0
TOTAL	1620	43.42	2111	56.58	3.731	100.0

El mayor porcentaje de alumnos aplazados se observa en aquellos que cursan la asignatura en condición 3 o mayor (74.20 %) seguidos de aquellos que la cursan en condición 0 (59.30 %) y aquellos de condición 1 (52.60 %). Sólo en el grupo de condición 2, es decir, los que cursan por tercera vez, se aprecia que el porcentaje de aprobados (57.20 %) supera a los aplazados.

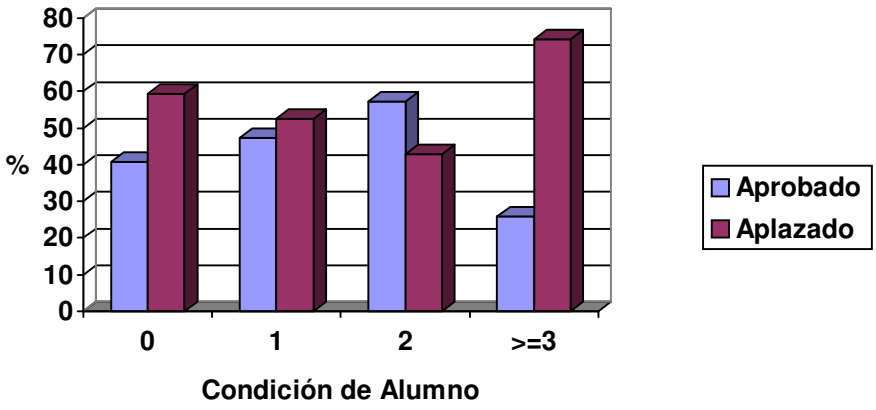


Gráfico 02: Distribución de las notas de los alumnos según su condición

Un porcentaje muy elevado de estudiantes de condición 3 o mayor, son aplazados. Solo en los de condición 2 los aprobados superan a los aplazados.

Cuadro 03: Promedio de notas por lapso de estudio en las 2 carreras

Lapso	Ing. en Informática		Análisis de Sistemas		t	p
	Prom.	DE	Prom.	DE		
2000-1	36.03	21.80	27.77	25.35	3.31	0.001*
2000-2	40.82	25.80	33.39	26.10	2.71	0007*
2001-1	**42.62**	22.56	**23.06**	25.98	7.63	**0.0001***
2001-2	34.91	22.70	28.20	26.50	2.74	0.006*
2002-2	**51.37**	16.90	**35.74**	23.90	6.21	**0.0001***
2003-1	38.21	25.90	31.74	23.30	2.71	0.007*
2003-2	37.35	25.90	30.23	24.80	2.85	0.009*
2004-1	**39.68**	26.60	**24.95**	24.30	5.36	**0.0001***
2004-2	43.55	29.20	30.71	25.50	4.31	0.001*
2005-1	**39.02**	28.30	**27.31**	28.14	4.11	**0.0001***

*** Diferencias estadísticamente significativas**

Existen diferencias estadísticamente significativas en los promedios de notas de la misma asignatura entre las carreras de Ingeniería en Informática y Análisis de sistemas, con una p (0.0001), intervalo de confianza muy cercano al 100%, en los lapsos 2001-1, 2002-2, 2004-1 y 2005-1 el promedio de notas en Ingeniería Informática fue significativamente mayor que el de Análisis de Sistemas (42.62, 51.37, 39.68, y 39.02 respectivamente). Se vuelve a observar que en el lapso 2000-2, se registra el mayor promedio de notas en ambas carreras: 51.37 para Ingeniería en Informática y 35.74 para Análisis de Sistemas, apreciándose que sólo el promedio de notas en Ingeniería en Informática supera la barrera de 48 puntos para aprobar la asignatura

Cuadro 04: Promedio de notas por sección en las 2 carreras del lapso 2000-1

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	**29.66**	19.34	29.79	27.71
2	37.76	24.53	**20.97**	21.92
3	37.05	18.82	**32.83**	25.18
4	33.43	21.16	26.73	25.37
5	34.50	21.50	-	-
6	**43.05**	24.10	-	-

En el lapso 2000-1, el mayor promedio de notas en Ingeniería en Informática fue en la sección 6 (43.05) y el menor promedio en la sección 1 (29.66). En Análisis de Sistemas el mayor promedio fue en la sección 3 (32.83) y el menor promedio en la sección 2 (20.97)

Cuadro 05: Promedio de notas por sección en las 2 carreras del lapso 2000-2

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	**18.92**	23.02	**45.04**	29.57
2	38.84	20.89	**13.54**	17.19
3	41.00	21.89	35.97	25.34
4	25.47	26.68	29.87	25.44
5	48.46	23.23	40.29	23.03
6	**59.44**	20.89	-	-

En el lapso 2000-2, el mayor promedio de notas en Ingeniería en Informática fue en la sección 6 (59.44) y el menor promedio en la sección 1 (18.92). En Análisis de Sistemas el mayor promedio fue en la sección 1 (45.04) y el menor promedio en la sección 2 (13.54).

Cuadro 06: Promedio de notas por sección en las 2 carreras del lapso 2001-1

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	**25.53**	29.65	**28.62**	28.27
2	42.13	20.56	**17.95**	23.63
3	32.12	17.80	22.93	23.90
4	37.69	17.25	23.23	28.53
5	**63.43**	16.47	-	-
6	39.08	18.00	-	-
7	45.94	25.63	-	-

En el lapso 2001-1, el mayor promedio de notas en Ingeniería en Informática fue en la sección 5 (63.43) y el menor promedio en la sección 1 (25.53). En Análisis de Sistemas el mayor promedio fue en la sección 1 (28.62) y el menor promedio en la sección 2 (17.95).

Cuadro 07: Promedio de notas por sección en las 2 carreras del lapso 2001-2

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	**26.57**	24.80	**44.88**	22.76
2	29.00	20.39	24.56	23.52
3	31.94	19.56	**10.43**	19.44
4	27.42	20.02	30.98	24.68
5	**52.73**	20.71	-	-
6	34.14	16.43	-	-
7	39.37	26.27	-	-

En el lapso 2001-2, el mayor promedio de notas en Ingeniería en Informática fue en la sección 5 (52.73) y el menor promedio en la sección 1 (26.57). En Análisis de Sistemas el mayor promedio fue en la sección 1 (44.88) y el menor promedio en la sección 3 (10.43).

Cuadro 08: Promedio de notas por sección en las 2 carreras del lapso 2002-2

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	-	-	-	-
2	43.29	14.01	**27.41**	22.90
3	45.61	13.16	**55.11**	28.11
4	**25.00**	25.69	33.65	20.95
5	**63.61**	15.22	39.79	21.59
6	56.46	11.86	-	-
7	49.00	12.82	-	-

En el lapso 2002-2, el mayor promedio de notas en Ingeniería en Informática fue en la sección 5 (63.61) y el menor promedio en la sección 4 (25.00). En Análisis de Sistemas el mayor promedio fue en la sección 3 (55.11) y el menor promedio en la sección 2 (27.41).

Cuadro 09: Promedio de notas por sección en las 2 carreras del lapso 2003-1

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	34.15	25.95	46.65	19.98
2	48.46	21.66	**55.23**	15.32
3	**25.36**	21.52	26.85	19.60
4	25.97	26.32	27.31	20.04
5	**53.91**	27.13	32.74	25.23
6	42.91	28.16	**17.42**	17.81
7	31.50	14.73	19.69	19.62

En el lapso 2003-1, el mayor promedio de notas en Ingeniería en Informática fue en la sección 5 (53.91) y el menor promedio en la sección 3 (25.36). En Análisis de Sistemas el mayor promedio fue en la sección 2 (55.23) y el menor promedio en la sección 6 (17.42).

Cuadro 10: Promedio de notas por sección en las 2 carreras del lapso 2003-2

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	**48.31**	28.84	**37.74**	27.17
2	33.76	21.19	**18.50**	27.02
3	**23.14**	21.65	33.07	20.10
4	35.32	30.84	33.11	21.41
5	46.78	25.56	21.41	22.76
6	33.49	22.70	31.68	23.69
7	37.07	21.22	34.07	24.40

En el lapso 2003-2, el mayor promedio de notas en Ingeniería en Informática fue en la sección 1 (48.31) y el menor promedio en la sección 3 (23.14). En Análisis de Sistemas el mayor promedio fue en la sección 1 (37.74) y el menor promedio en la sección 2 (18.50).

Cuadro 11: Promedio de notas por sección en las 2 carreras del lapso 2004-1

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	49.38	28.88	25.52	25.84
2	41.96	22.06	**20.89**	29.81
3	31.29	18.87	**33.17**	24.55
4	**29.39**	27.15	22.32	23.77
5	**51.52**	29.75	23.96	21.26
6	34.00	25.48	21.97	22.19
7	32.64	22.36	-	-

En el lapso 2004-1, el mayor promedio de notas en Ingeniería en Informática fue en la sección 5 (51.52) y el menor promedio en la sección 4 (29.39). En Análisis de Sistemas el mayor promedio fue en la sección 3 (33.17) y el menor promedio en la sección 2 (20.89).

Cuadro 12: Promedio de notas por sección en las 2 carreras del lapso 2004-2

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	49.17	20.03	33.96	25.59
2	38.61	32.57	39.91	28.71
3	41.83	26.44	21.78	20.54
4	**31.30**	28.72	29.87	23.25
5	**66.12**	28.81	26.88	25.37
6	32.62	35.14	**47.24**	20.16
7	36.79	20.27	**13.83**	21.24
8	50.11	24.19	-	-

En el lapso 2004-2, el mayor promedio de notas en Ingeniería en Informática fue en la sección 5 (66.12) y el menor promedio en la sección 4 (31.30). En Análisis de Sistemas el mayor promedio fue en la sección 6 (47.24) y el menor promedio en la sección 7 (13.83).

Cuadro 13: Promedio de notas por sección en las 2 carreras del lapso 2005-1

Sección	Ing. en Informática		Análisis de Sistemas	
	Prom.	DE	Prom.	DE
1	**52.90**	18.57	35.84	28.06
2	27.04	26.57	36.77	31.63
3	39.38	22.91	20.66	23.74
4	41.07	27.96	**37.29**	30.40
5	39.59	31.88	24.90	22.18
6	30.18	34.02	**6.21**	6.17
7	**23.09**	25.23	16.64	28.51
8	49.00	26.52	-	-

En el lapso 2005-1, el mayor promedio de notas en Ingeniería en Informática fue en la sección 1 (52.90) y el menor promedio en la sección 7 (23.09). En Análisis de Sistemas el mayor promedio fue en la sección 4 (37.29) y el menor promedio en la sección 6 (6.21).

Cuadro 14: Secciones con mayor y menor Promedio de notas por Lapso en ambas carreras

	Ingeniería en Informática					Análisis de Sistemas				
	Mayor Promedio		Menor Promedio		Dif.	Mayor Promedio		Menor Promedio		Dif.
Lapso	Sec.	Prom.	Sec.	Prom.	Prom.	Sec.	Prom.	Sec.	Prom.	Prom.
2000-1	6	43.05	1	29.66	13.39	3	32.83	**2**	20.97	11.86
2000-2	6	59.44	1	**18.92**	**40.52**	1	45.04	**2**	13.54	31.50
2001-1	**5**	63.43	1	25.53	37.90	1	28.62	**2**	17.95	10.67
2001-2	**5**	52.73	1	26.57	26.16	1	44.88	3	10.43	34.45
2002-2*	**5**	63.61	4	25.00	38.61	3	55.11	**2**	27.41	27.70
2003-1	**5**	53.91	3	25.36	28.55	2	**55.23**	6	17.42	**37.81**
2003-2	1	48.31	3	23.14	25.17	1	37.74	**2**	18.50	19.24
2004-1	**5**	51.52	4	29.39	22.13	3	33.17	**2**	20.89	12.28
2004-2	**5**	**66.12**	4	31.30	34.82	6	47.24	7	13.83	33.41
2005-1	1	52.90	7	23.09	29.81	4	37.29	6	**06.21**	31.08

* No se abrió la sección 1 para ninguna de las 2 carreras

Para Ingeniería en Informática el mayor promedio (66.12) fue en la sección 5 del lapso 2004-2, el menor promedio (18.92) fue en la sección 1 del lapso 2000-2, en este lapso también hubo la mayor diferencia de promedio (40.52) entre dos secciones. Para Análisis de Sistemas el mayor promedio (55.23) fue en la sección 2 del lapso 2003-1, en este lapso también hubo la mayor diferencia de promedio (37.81) entre dos secciones, el menor promedio (06.21) fue en la sección 4 del lapso 2005-1.

Por otro lado, también se observa que en Ingeniería en Informática la sección 5 tiene el mayor número de veces (6) donde se registra el mayor promedio, mientras que en Análisis de Sistemas la sección 2 tiene el mayor número de veces (6) donde se registra el menor promedio.

Cuadro 15: Opinión del Grupo Estudio sobre el grado de satisfacción de las enseñanzas recibidas sobre Estructuras de datos. Cuestionario 01

	Muy Bien		Bien		A medias		Un Poco		Nada		TOTAL	
	N°	%	N°	%	N°	%	N°	%	N°	%	N°	%
Concepto de Estructuras de Datos	2	4.4	12	26.7	25	**55.6**	4	8.9	2	4.4	45	100
Concepto de arreglos	**4**	**8.9**	26	57.8	14	31.1	1	2.2	-	-	45	100.0
Dimensión de un Arreglo	5	11.1	21	**46.7**	16	35.6	2	4.4	1	2.2	45	100.0
Cómo se declara un tipo de Arreglo	22	**48.9**	18	40.0	4	8.9	1	2.2	-	-	45	100.0
Que tipo debe ser el Subíndice de un Arreglo	13	28.9	13	28.9	14	**31.1**	2	4.4	3	6.7	45	100.0
Tipo de elementos que puede contener un Arreglo	10	22.2	14	31.1	15	**33.3**	5	11.1	1	2.2	45	100.0
Cómo se puede acceder un elemento dentro de una variable de tipo Arreglo	8	17.8	11	24.2	18	**40.0**	4	8.9	4	8.9	45	100.0
Cómo determinar el número máximo de elementos que puede contener una variable de tipo Arreglo	7	15.6	19	**42.2**	13	28.9	4	8.9	2	4.4	45	100.0

El 55.6% de los estudiantes está satisfecho a medias de las enseñanzas recibidas sobre el concepto de Estructuras de Datos, sólo 4 de los alumnos (8.9%) manifestó estar satisfecho muy bien sobre el concepto de arreglos. El 46.7% está satisfecho bien sobre que es la dimensión de un arreglo y 48.9% está satisfecho muy bien sobre como se declara un tipo de Arreglo. El 31.1%, 33.3% y 40% considera estar satisfecho solo a medias sobre que tipo debe ser el subíndice de un arreglo, el tipo de elementos que puede contener un arreglo y como se puede acceder a un elemento dentro de una variable de tipo arreglo, respectivamente. Sólo el 42.2% de los estudiantes estuvo satisfecho bien sobre como determinar el número de elementos que puede contener una variable de tipo arreglo.

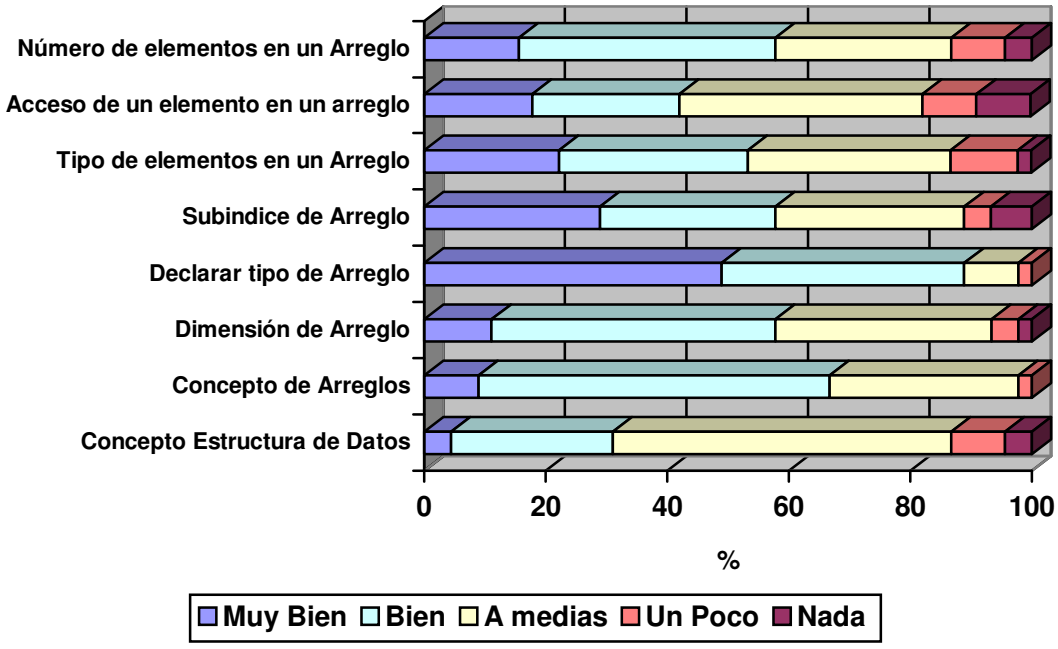


Gráfico 03: Opinión del Grupo Estudio sobre el grado de satisfacción de las enseñanzas recibidas sobre Estructuras de datos. Cuestionario 01

Sólo en el ítem relacionado a declarar tipo de arreglo, los estudiantes manifestaron estar satisfecho muy bien y bien. En General, en los demás Items, cerca de la mitad del grupo se ubica de a medias hacia muy bien, y la otra mitad de a medias hacia nada.

Cuadro 16: Opinión del Grupo Control sobre el grado de satisfacción de las enseñanzas recibidas sobre Estructuras de datos. Cuestionario 01

	Muy Bien		Bien		A medias		Un Poco		Nada		TOTAL	
	N°	%	N°	%	N°	%	N°	%	N°	%	N°	%
Concepto de Estructuras de Datos	3	7.5	10	25.0	14	**35.0**	10	25.0	3	7.5	40	100.0
Concepto de arreglos	7	17.5	23	**57.5**	8	20.0	2	5.0	-	-	40	100.0
Dimensión de un Arreglo	3	7.5	21	**52.5**	9	22.5	6	15.0	1	2.5	40	100.0
Cómo se declara un tipo de Arreglo	13	32.5	15	**37.5**	11	27.5	1	2.5	-	-	40	100.0
Que tipo debe ser el Subíndice de un Arreglo	10	25.0	15	**37.5**	11	27.5	3	7.5	1	2.5	40	100.0
Tipo de elementos que puede contener un Arreglo	5	12.5	19	**47.5**	10	25.0	6	15.0	-	-	40	100.0
Cómo se puede acceder un elemento dentro de una variable de tipo Arreglo	6	15.0	9	22.5	14	**35.0**	4	10.0	7	17.5	40	100.0
Cómo determinar el número máximo de elementos que puede contener una variable de tipo Arreglo	4	10.0	15	**37.5**	15	**37.5**	4	10.0	2	5.0	40	100.0

El 35% de los estudiantes considera estar satisfecho a medias sobre el concepto de Estructuras de Datos y el 57.5% de los alumnos manifestó estar satisfecho bien sobre el concepto de arreglos. El 52.5% considera estar satisfecho bien sobre que es la dimensión de un arreglo y 37.5% considera estar satisfecho bien sobre como se declara un tipo de Arreglo y el tipo que debe ser el subíndice de un arreglo. El 47.5% manifiesta estar satisfecho bien sobre el tipo de elementos que puede contener un arreglo y 35.0 % considera estar satisfecho solo a medias sobre como se puede acceder un elemento dentro de una variable de tipo arreglo. El 75% de los estudiantes manifiesta estar satisfecho muy bien y bien sobre como determinar el número de elementos que puede contener una variable de tipo arreglo.

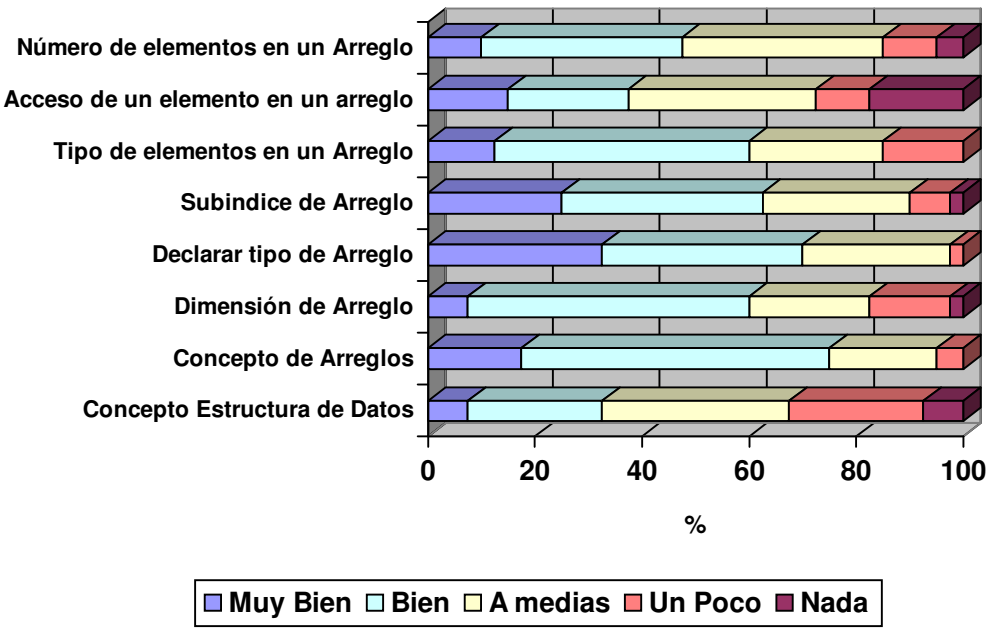


Gráfico 04: Opinión del Grupo Control sobre el grado de satisfacción de las enseñanzas recibidas sobre Estructuras de datos. Cuestionario 01

En los ítems relacionados con: declarar tipo de arreglo, concepto de arreglos, tipos de elementos, subíndice y dimensión del arreglo, cerca de la mitad de los estudiantes manifestaron estar satisfecho muy bien y bien. En los restantes ítems, manifestaron que el nivel de satisfacción se ubicaba de a medias hacia nada.

Cuadro 17: Evaluación de los alumnos Grupo Estudio sobre los conocimientos que poseen sobre Estructuras de datos. Evaluación 01

	Correctas		Incorrectas		TOTAL	
	N°	%	N°	%	N°	%
Sobre el concepto de Estructuras de Datos	33	**73.3**	12	26.7	45	100.0
Sobre el concepto de Arreglos	40	**88.9**	5	11.1	45	100.0
Sobre la dimensión de un Arreglo	23	**51.1**	22	**48.9**	45	100.0
Sobre la declaración del sub-índice de un tipo Arreglo (a)	22	**48.9**	23	**51.1**	45	100.0
Sobre el chequeo de rango del sub-índice de una variable Arreglo (b)	24	**53.3**	21	**46.7**	45	100.0
Sobre el chequeo del tipo de dato asignado a una variable Arreglo (c)	7	15.6	38	**84.4**	45	100.0
Sobre el tipo de dato que debe ser el sub-índice de un Arreglo	17	38.7	28	**61.3**	45	100.0
Sobre el tipo de elementos que puede contener un Arreglo	18	40.0	27	**60.0**	45	100.0
Sobre cómo acceder a un elemento dentro de una variable de tipo Arreglo	7	15.6	38	**84.4**	45	100.0
Sobre cómo determinar el número máximo de elementos que puede contener una variable de tipo Arreglo	17	37.8	28	**62.2**	45	100.0

El 73.3% de los estudiantes del Grupo Estudio respondió correctamente la pregunta relacionada al concepto de estructuras de datos, igualmente hay un alto porcentaje de 88.9% de respuestas correctas referentes al concepto de arreglos. En las preguntas relacionadas a la dimensión de un arreglo y declaración y chequeo de rango del sub-índice de un arreglo, no hay mucha diferencia en los porcentajes de las respuestas correctas e incorrectas. En las preguntas relacionadas al chequeo del tipo de dato asignado a una variable arreglo y a como acceder a un elemento dentro de la variable arreglo, se observa un porcentaje alto de 84.4% de respuestas incorrectas; lo mismo sucede pero en un porcentaje menor de alrededor del 61% en las preguntas sobre el tipo de dato que debe ser el sub-índice de un arreglo, el tipo de elementos que puede contener un arreglo y sobre cómo determinar el número máximo de elementos que puede contener una variable de tipo arreglo.

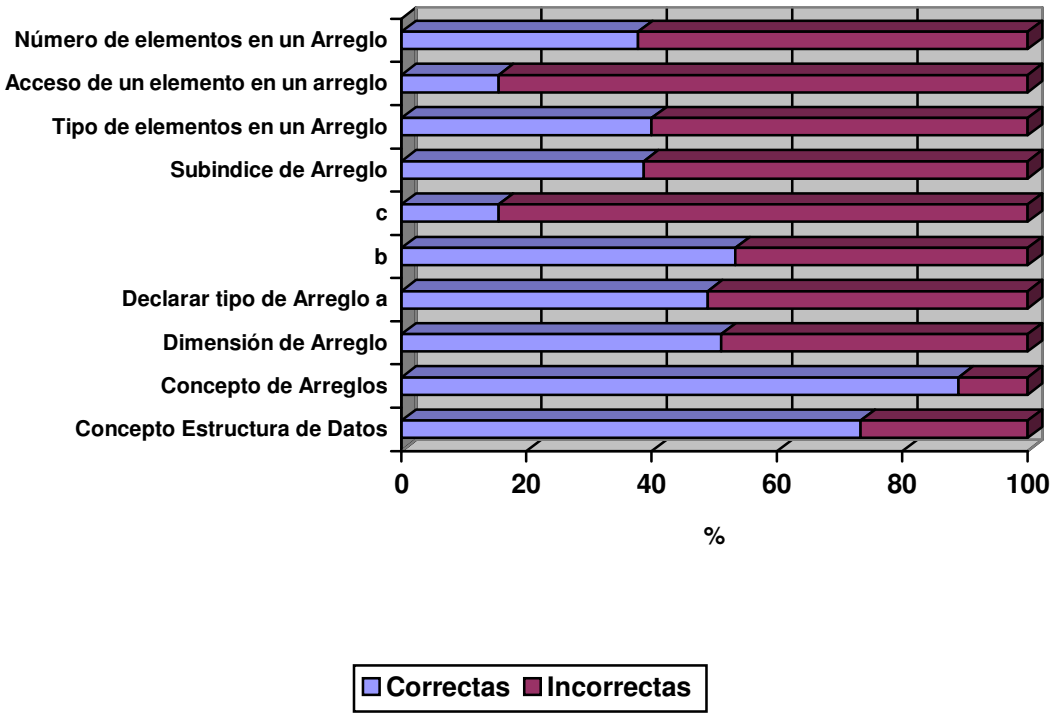


Gráfico 05: Evaluación de los alumnos Grupo Estudio sobre los conocimientos que poseen sobre Estructuras de datos. Evaluación 01

Para el Grupo Estudio, sólo en las preguntas relacionadas con el concepto de estructuras de datos y arreglos se observa un porcentaje alto de respuestas correctas, mientras que para las demás preguntas las respuestas incorrectas van desde porcentajes medios a más altos.

Cuadro 18: Evaluación de los alumnos Grupo Control sobre los conocimientos que poseen sobre Estructuras de datos. Evaluación 01

	Correctas		Incorrectas		TOTAL	
	N°	%	N°	%	N°	%
Sobre el concepto de Estructuras de Datos	39	**97.5**	1	2.5	40	100.0
Sobre el concepto de Arreglos	37	**92.5**	3	7.5	40	100.0
Sobre la dimensión de un Arreglo	21	**52.5**	19	**47.5**	40	100.0
Sobre la declaración del sub-índice de un tipo Arreglo (a)	17	**42.5**	23	**57.5**	40	100.0
Sobre el chequeo de rango del sub-índice de una variable Arreglo (b)	25	**62.5**	15	37.5	40	100.0
Sobre el chequeo del tipo de dato asignado a una variable Arreglo (c)	9	22.5	31	**77.5**	40	100.0
Sobre el tipo de dato que debe ser el sub-índice de un Arreglo	11	27.5	29	**72.5**	40	100.0
Sobre el tipo de elementos que puede contener un Arreglo	19	**47.5**	21	**52.5**	40	100.0
Sobre cómo acceder a un elemento dentro de una variable de tipo Arreglo	6	15.0	34	**85.0**	40	100.0
Sobre cómo determinar el número máximo de elementos que puede contener una variable de tipo Arreglo	17	42.5	23	**57.5**	40	100.0

El 97.5% de los estudiantes del Grupo Control respondió correctamente la pregunta relacionada al concepto de estructuras de datos, igualmente hay un alto porcentaje de 92.5% de respuestas correctas referentes al concepto de arreglos. Se observa un porcentaje de 62.5% de respuestas correctas referentes al chequeo de rango del sub-índice de una variable arreglo. En las preguntas relacionadas a la dimensión de un arreglo, declaración del sub-índice de un arreglo, sobre el tipo de elementos que puede contener un arreglo y sobre cómo determinar el número máximo de elementos que puede contener una variable de tipo arreglo, no hay mucha diferencia en los porcentajes de las respuestas correctas e incorrectas. En las preguntas relacionadas al chequeo del tipo de dato asignado a una variable arreglo, el tipo de dato que debe ser el sub-índice de un arreglo y a como acceder a un elemento

dentro de la variable arreglo, se observan porcentajes altos de 77.5, 72.5 y 85.0% respectivamente de respuestas incorrectas.

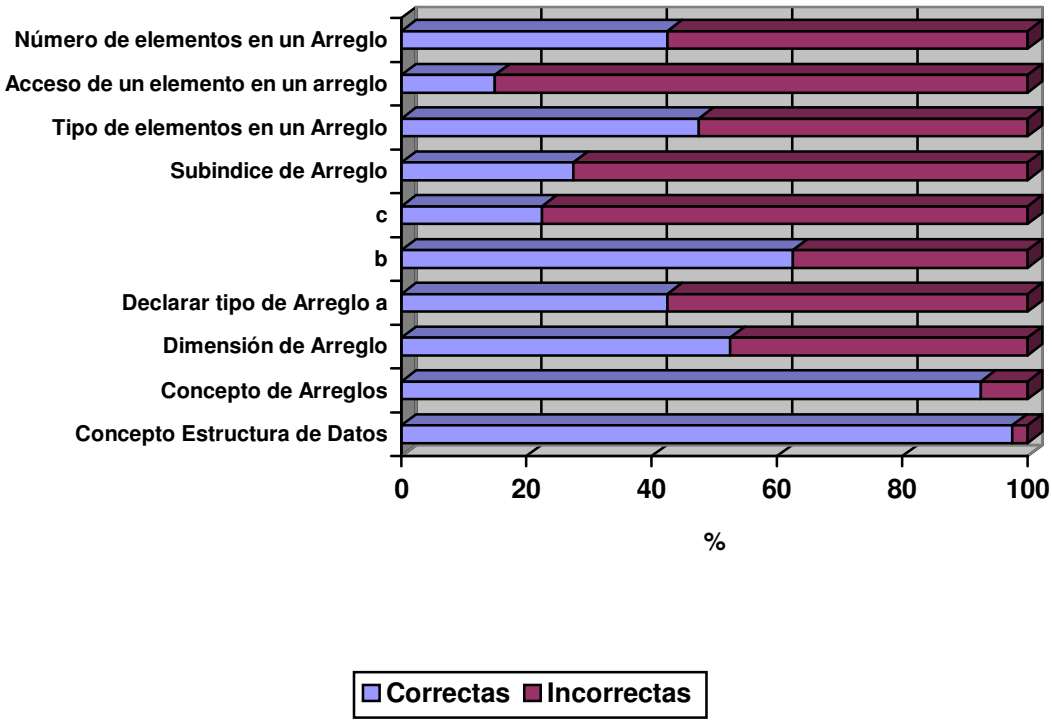


Gráfico 06: Evaluación de los alumnos Grupo Control sobre los conocimientos que poseen sobre Estructuras de datos. Evaluación 01

Para el Grupo Control, sólo en las preguntas relacionadas con el concepto de estructuras de datos y arreglos, y al chequeo de rango del sub-índice de una variable arreglo se observa un porcentaje alto de respuestas correctas, mientras que para las demás preguntas las respuestas incorrectas van desde porcentajes medios a más altos.

Cuadro 19: Evaluación de los alumnos ambos Grupos sobre los conocimientos que poseen sobre Estructuras de datos. Evaluación 01

	Correctas		Incorrectas		TOTAL	
	N°	%	N°	%	N°	%
Sobre el concepto de Estructuras de Datos	72	**84.7**	13	15.3	85	100.0
Sobre el concepto de Arreglos	77	**90.6**	8	9.4	85	100.0
Sobre la dimensión de un Arreglo	44	**51.8**	41	**48.2**	85	100.0
Sobre la declaración del sub-índice de un tipo Arreglo (a)	39	**45.9**	46	**54.1**	85	100.0
Sobre el chequeo de rango del sub-índice de una variable Arreglo (b)	49	**57.6**	36	**42.4**	85	100.0
Sobre el chequeo del tipo de dato asignado a una variable Arreglo (c)	16	18.8	69	**81.2**	85	100.0
Sobre el tipo de dato que debe ser el sub-índice de un Arreglo	28	32.9	57	**67.1**	85	100.0
Sobre el tipo de elementos que puede contener un Arreglo	37	**43.5**	48	**56.5**	85	100.0
Sobre cómo acceder a un elemento dentro de una variable de tipo Arreglo	13	15.3	72	**84.7**	85	100.0
Sobre cómo determinar el número máximo de elementos que puede contener una variable de tipo Arreglo	34	40.0	51	**60.0**	85	100.0

El 84.7% de los estudiantes de ambos Grupos respondió correctamente la pregunta relacionada al concepto de estructuras de datos, igualmente hay un alto porcentaje de 90.6% de respuestas correctas referentes al concepto de arreglos. En las preguntas relacionadas a la dimensión de un arreglo, declaración del sub-índice de un arreglo, chequeo de rango del sub-índice de una variable arreglo y sobre el tipo de elementos que puede contener un arreglo, no hay mucha diferencia en los porcentajes de las respuestas correctas e incorrectas. En las preguntas relacionadas al chequeo del tipo de dato asignado a una variable arreglo, el tipo de dato que debe ser el sub-índice de un arreglo, como acceder a un elemento dentro de la variable arreglo y sobre cómo determinar el número máximo de elementos que puede contener una variable de tipo arreglo, se observan porcentajes altos de 81.2, 67.1, 84.7 y 60.0% respectivamente de respuestas incorrectas.

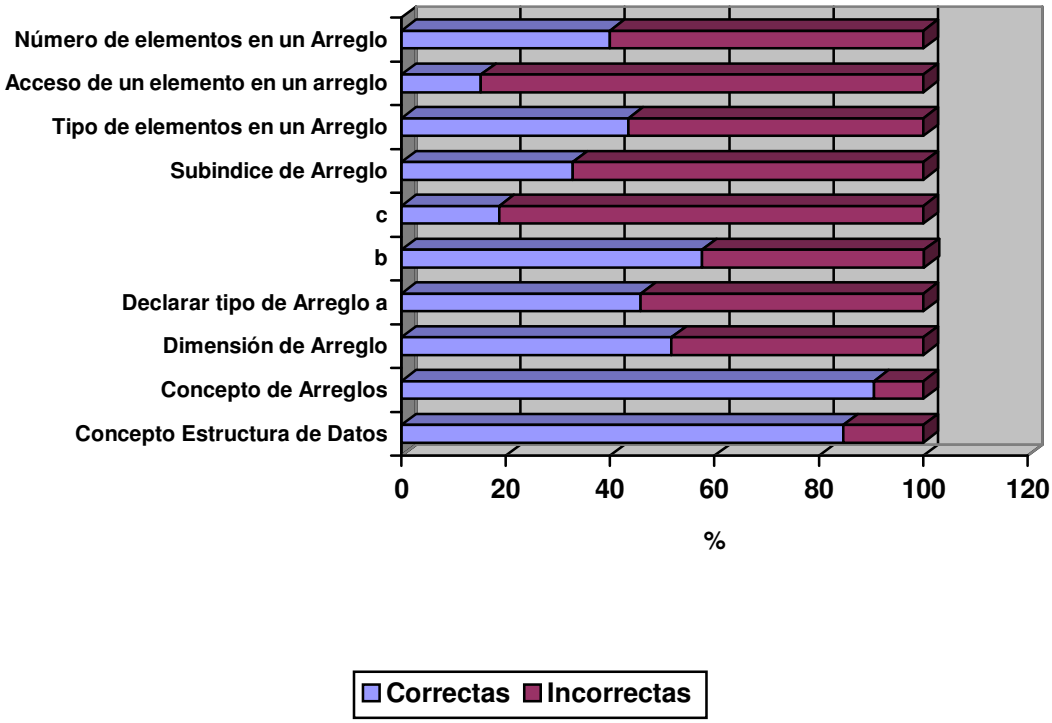


Gráfico 07: Evaluación de los alumnos ambos Grupos sobre los conocimientos que poseen sobre Estructuras de datos. Evaluación 01

Para ambos Grupos, sólo en las preguntas relacionadas con el concepto de estructuras de datos y arreglos, y al chequeo de rango del sub-índice de una variable arreglo se observa un porcentaje alto de respuestas correctas, mientras que para las demás preguntas las respuestas incorrectas van desde porcentajes medios a más altos.

Cuadro 20: Nivel de Conocimiento de los estudiantes sobre Estructuras de Datos. Grupo Estudio y Grupo Control. Evaluación 01

Nivel de Conocimiento	Grupo Estudio		Grupo Control	
	N°	%	N°	%
Deficiente	23	**51.1**	16	**40.0**
Regular	17	37.8	20	50.0
Bueno	5	**11.1**	4	**10.0**
TOTAL	45	100.0	40	100.0

El 51.1% de los estudiantes del Grupo Estudio mostró un nivel de conocimiento deficiente y apenas un 11.1% bueno. En el Grupo Control se observan resultados similares, 40.0% deficiente y apenas un 10.0% bueno. Por lo tanto, ambos grupos se consideran equilibrados respecto al nivel de conocimientos sobre Estructuras de Datos.

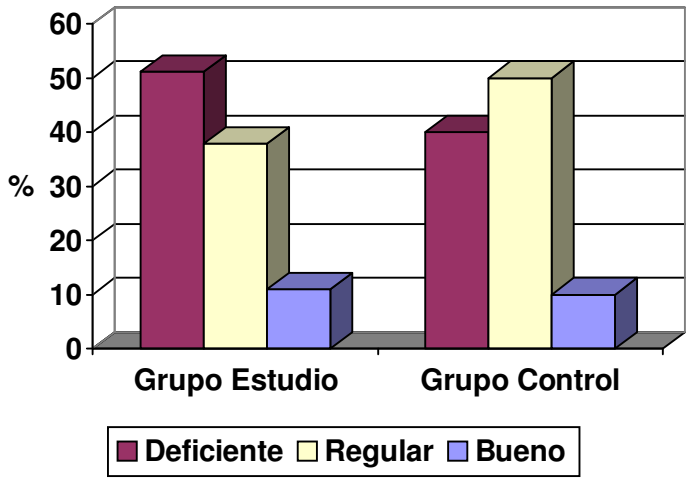


Gráfico 08: Nivel de Conocimiento de los estudiantes sobre Estructuras de Datos. Grupo Estudio y Grupo Control. Evaluación 01

En ambos grupos, alrededor del 90 % de los estudiantes se ubica en un nivel de conocimiento entre deficiente y regular.

Cuadro 21: Opinión de los estudiantes del Grupo Estudio sobre el software simulador de ordenamiento de elementos en un Arreglo

	Totalmente de acuerdo		De acuerdo		Indeciso		En desacuerdo		Totalmente en desacuerdo		Total	
	N°	%	N°	%	N°	%	N°	%	N°	%	N°	%
Puedo lograr aprender de una manera más fácil el algoritmo de ordenamiento de datos en un arreglo	25	**55.6**	19	**42.2**	0	0	0	0	1	2.2	45	100
Con el software me siento motivado a aprender sobre el algoritmo de ordenamiento de arreglos	20	**44.4**	23	**51.1**	2	4.4	0	0	0	0	45	100.0
El software brinda suficiente ayuda en el proceso de ordenamiento de un arreglo	24	**53.3**	16	**35.6**	5	11.1	0	0	0	0	45	100.0
El software es fácil de comprender y utilizar	29	**64.4**	15	**33.3**	1	2.2	0	0	0	0	45	100.0
El software presenta una pantalla donde puedo captar la esencia del algoritmo	25	**55.6**	18	**40.0**	2	4.4	0	0	0	0	45	100.0
El software presenta suficientes elementos gráficos	20	**44.4**	15	**33.3**	7	15.6	3	6.7	0	0	45	100.0
El software presenta suficiente colores para distinguir los elementos	22	**48.9**	20	**44.4**	1	2.2	2	4.4	0	0	45	100.0
El software debería utilizar sonido y video	22	**48.9**	14	**31.1**	5	11.1	4	8.9	0	0	45	100.0

Un alto porcentaje de alumnos, en todos los ítems se encuentra de acuerdo y completamente de acuerdo en que se puede lograr el aprendizaje de una manera más fácil y con mayor motivación el algoritmo de ordenamiento de datos, igualmente, en que brinda suficiente ayuda y es fácil de comprender y utilizar, que presenta una

pantalla donde se puede captar la esencia del algoritmo y que presenta suficientes elementos gráficos y colores. Finalmente, que debería utilizar sonido y video.

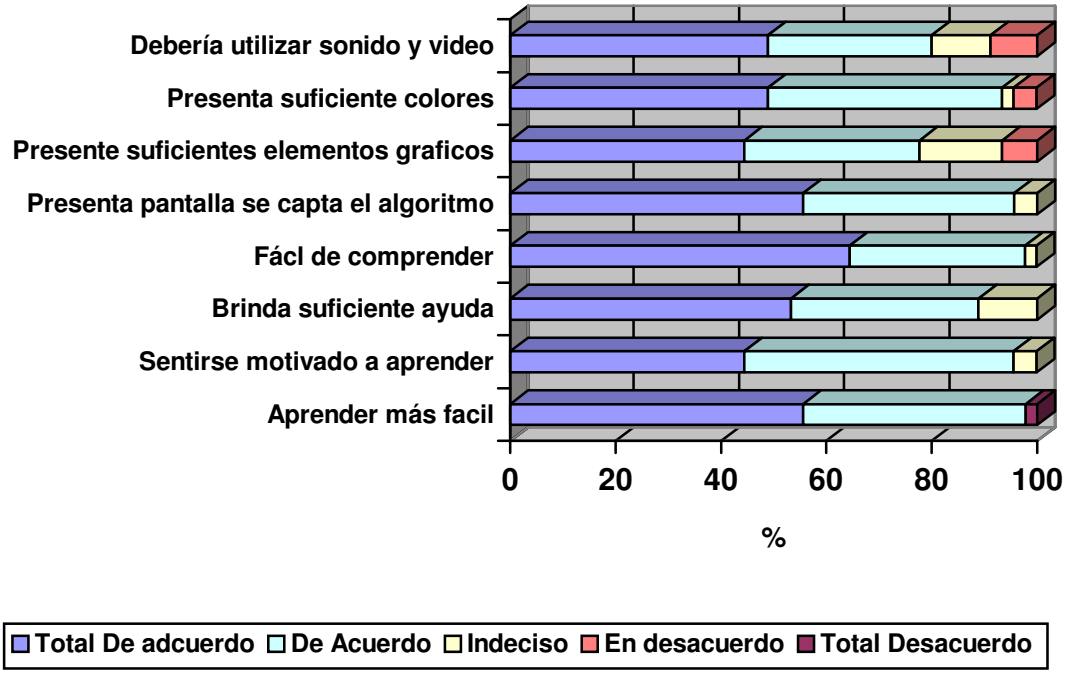


Gráfico 09: Opinión de los estudiantes del Grupo Estudio sobre el software simulador de ordenamiento de elementos en un Arreglo

La gran mayoría de los estudiantes se encuentra de acuerdo y completamente de acuerdo con todos los ítems referentes al software simulador de ordenamiento de elementos en un Arreglo.

Cuadro 22: Nivel de conocimiento de los estudiantes de ambos grupos en el aprendizaje referente al código del algoritmo. Evaluación 02

Nivel de Conocimiento	Grupo Estudio		Grupo Control	
	N°	%	N°	%
Deficiente	4	8.9	22	**55.0**
Bueno	41	**91.1**	18	45.0
TOTAL	45	100.0	40	100.0

Chi²: 19.09 p: 0.00001

El 91.1% de los estudiantes del Grupo Estudio se ubicó en un nivel de conocimiento bueno, en contraste con el 45.0% de los estudiantes del Grupo Control. Se hace notar que el 55.0% de los estudiantes del Grupo Control se ubicaron en el nivel de conocimiento deficiente. Con una p: 0.00001 se puede asegurar que existen diferencias estadísticamente significativas entre el Grupo Estudio y el Grupo Control.

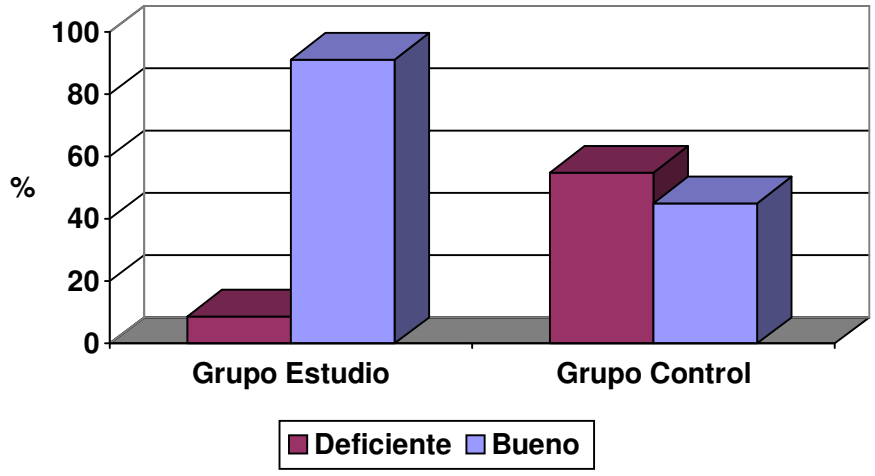


Gráfico 10: Nivel de conocimiento de los estudiantes de ambos grupos en el aprendizaje referente al código del algoritmo. Evaluación 02

En lo referente al código del algoritmo, los estudiantes del Grupo Estudio se ubicaron en un nivel de conocimiento bueno, en tanto que los del Grupo Control se distribuyeron casi uniformemente entre deficiente y bueno.

Cuadro 23: Nivel de conocimiento de los estudiantes de ambos grupos en el aprendizaje referente a iteraciones del algoritmo. Evaluación 02

Nivel de Conocimiento	Grupo Estudio		Grupo Control	
	N°	%	N°	%
Deficiente	3	6.7	15	**37.5**
Bueno	42	**93.3**	25	**62.5**
TOTAL	45	100.0	40	100.0

Chi²: 10.28 p: 0.0001

El 93.3% de los estudiantes del Grupo Estudio se ubicó en un nivel de conocimiento bueno, en contraste con el 62.5% de los estudiantes del Grupo Control. A pesar de que en ambos grupos se obtuvieron porcentajes altos en relación al nivel de conocimiento bueno, se puede asegurar, con una p: 0.00001 que existen diferencias estadísticamente significativas entre el Grupo Estudio y el Grupo Control. Se hace notar que el 37.5% de los estudiantes del Grupo Control se ubicaron en el nivel de conocimiento deficiente en contraste con un 6.7% del Grupo Estudio.

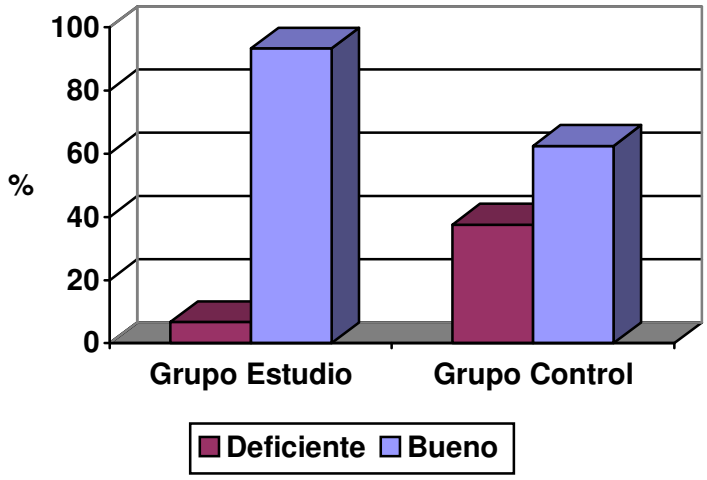


Gráfico 11: Nivel de conocimiento de los estudiantes de ambos grupos en el aprendizaje referente a iteraciones del algoritmo. Evaluación 02

En lo referente al aprendizaje de las iteraciones del proceso del Algoritmo Burbuja, la mayoría de los estudiantes de ambos grupos se ubicaron en un nivel de conocimiento bueno, no obstante, este porcentaje fue mayor en el Grupo Estudio.

Cuadro 24: Nivel de conocimiento de los estudiantes de ambos grupos en el aprendizaje referente al algoritmo Burbuja. Evaluación 02

Nivel de Conocimiento	Grupo Estudio		Grupo Control	
	N°	%	N°	%
Deficiente	1	**2.2**	16	**40.0**
Regular	4	8.9	10	25.0
Bueno	40	**88.9**	14	**35.0**
TOTAL	45	100.0	40	100.0

El 88.9% de los estudiantes del Grupo Estudio se ubicó en un nivel de conocimiento bueno, en contraste con el 35.0% de los estudiantes del Grupo Control. Se hace notar que apenas un 2.2% del Grupo Estudio se ubicó en un nivel de conocimiento deficiente en contraste con el 40.0% de los estudiantes del Grupo Control.

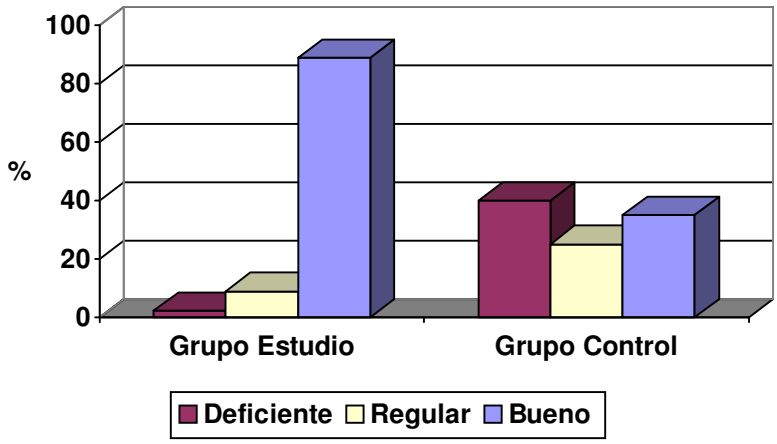


Gráfico 12: Nivel de conocimiento de los estudiantes de ambos grupos en el aprendizaje referente al algoritmo Burbuja. Evaluación 02

Se resalta el alto porcentaje de estudiantes del Grupo Estudio con un buen nivel de conocimientos adquiridos en relación al Grupo Control donde el nivel de conocimiento se reparte casi uniformemente entre deficiente, regular y bueno.

Fase de Factibilidad

Social

La factibilidad social se fundamentó en determinar las necesidades de los estudiantes a fin de describir las funcionalidades asociadas con un software educativo para la enseñanza de estructuras de datos en programación. En base a los resultados obtenidos en las evaluaciones, se aprecia una gran deficiencia en los conocimientos básicos relacionados con los arreglos. Por lo tanto, se considera que los estudiantes valorarán y apreciarán el diseño de un modelo de software educativo para la enseñanza de estructuras de datos en programación, razón por la cual se considera una alternativa viable que permitirá satisfacer las exigencias de un cuerpo social en la búsqueda de métodos alternativos que permitan afianzar los conocimientos en el proceso de enseñanza - aprendizaje.

Operativa

Respecto a la factibilidad operativa desde la perspectiva del cliente, se tiene que esta queda garantizada en virtud de que el manejo de aplicaciones de Internet es conocido por los miembros del Decanato de Ciencias y Tecnología. Por otra parte, desde la perspectiva del servidor, un software educativo para la enseñanza de estructuras de datos en programación necesita de un administrador de la página donde se coloque el software. En este caso, se cuenta con el Proyecto Universidad Virtual que permite que los profesores puedan administrar de una manera sencilla los recursos requeridos. En consecuencia, la operatividad queda garantizada por la estructura organizativa asociada al Proyecto Universidad Virtual.

Legal

La presente investigación se basa en fundamentos legales que establece la Universidad Centroccidental "Lisandro Alvarado" en su Reglamento General de Postgrado. En particular el artículo 50 relata el aporte humanístico, científico o técnico de un problema referente a la realidad nacional, ajustándose la presente investigación en tales lineamientos a fin de obtener una alternativa al problema del bajo rendimiento de los estudiantes de Programación del Decanato de Ciencias y Tecnología.

Por otro lado, la colocación de un software educativo a disposición de los estudiantes, debería apegarse a la normativa dictada por la Universidad Centroccidental "Lisandro Alvarado" referente al Desarrollo Experimental de Cursos en Línea.

Institucional

En correspondencia al aspecto institucional, el diseño de un modelo de software educativo para la enseñanza de estructuras de datos en programación, posibilitaría a la institución la puesta en marcha de una opción viable que permita mejorar la efectividad del proceso de enseñanza-aprendizaje, que se lleva a cabo en el aula. En este sentido es importante destacar que la UCLA, a través del Proyecto Universidad Virtual, cuenta con el personal requerido para administrar un software basado en el modelo de la presente investigación.

Tecnológica

El desarrollo del modelo del software educativo propuesto requiere de recursos de Hardware y Software. Debido a la iniciativa del gobierno nacional en promover el

uso del software libre en los organismos gubernamentales, se dará prioridad a la utilización de herramientas de licencia libre para el diseño del modelo de software educativo para la enseñanza de estructuras de datos en programación.

El diseño de un modelo de software educativo para la enseñanza de estructuras de datos en programación, requiere de herramientas de hardware y software para su documentación. El producto sobre el cual se basó el diseño es Visual Paradigm (versión 5.2), herramienta visual que permite representar las diversas especificaciones del software educativo basado en el lenguaje de modelado unificado UML. Para el uso de este producto no se requiere licencia de adquisición. Por otra parte, la herramienta necesita para su instalación el ambiente de ejecución Java, requiriéndose el uso de un computador con capacidades para soportar un sistema operativo de licencia libre, por lo tanto, se utilizó Linux en su versión Red Hat 9.0. En este caso, la UCLA cuenta con el equipo necesario para permitir el desarrollo del diseño que se propone en esta investigación.

Con respecto al Hardware necesario, no es el alcance de esta investigación desarrollar e implantar el modelo propuesto, sin embargo, la UCLA cuenta con toda una gama de recursos de Hardware que se consideraron a futuro.

Económica

Como se expuso anteriormente, no es el alcance de esta investigación el desarrollo e implantación del modelo propuesto, sin embargo, la factibilidad económica esta garantizada ya que la UCLA no necesita hacer inversiones adicionales para la puesta en marcha de un software educativo basado en el modelo propuesto.

CAPITULO IV

PROPUESTA DEL ESTUDIO

Introducción

Luego de haber examinado los antecedentes y el marco teórico relacionados con esta investigación, asimismo de hacer una evaluación y análisis de los resultados obtenidos en el marco metodológico, se procede a proponer un modelo que es el resultado del análisis e interpretación de los resultados obtenidos durante este estudio, lo cual permitirá a futuro el desarrollo de un Software Educativo para la Enseñanza de Estructuras de Datos en Programación, denominado de ahora en adelante como SEDPRO.

El modelo del sistema SEDPRO se apoyó en la metodología planteada por Díaz (2003), donde a partir de la metodología RUP, hizo una adaptación y extensión para la construcción de software educativo apoyándose en el Modelo Sistémico de Calidad (MOSCA), propuesto por el Laboratorio de Información y Sistemas (LISI) de la Universidad Simón Bolívar, asegurándose que se produzca desde las primeras fases de desarrollo, un producto educativo de calidad que cumpla con las características de funcionalidad, usabilidad y fiabilidad (ver Tabla 01). El modelo también se apoya en muchos de los aspectos considerados por González (2004) para la evaluación de software educativo, ya que pretende orientar el uso de este tipo de programas por parte de los docentes.

Una vez diseñado el modelo, se creará un prototipo programado en Delphi, en el cual sólo se contemplará el proceso de ordenamiento de elementos en un arreglo a

través del algoritmo “Burbuja”. Se escoge el lenguaje de Programación Delphi ya que este es Orientado a Objeto y muchas de sus instrucciones son iguales al Lenguaje de Programación Pascal, el cual es utilizado por los alumnos de la asignatura programación I.

Definición del Sistema SEDPRO

Un sistema para la enseñanza de estructuras de datos en programación, SEDPRO, es una solución de software que hace uso de la simulación para representar conceptos y procesos relacionados a la programación. Este enfoque permite a los estudiantes afianzar sus conocimientos sobre las Estructuras de Datos.

Propósitos

- Establecer los lineamientos que permitan desarrollar un sistema que haga uso de la simulación y que facilite el aprendizaje de las estructuras de datos en programación.

Objetivos

- Precisar una solución de software educativo que pueda ser fácilmente convertida en código fuente.

- Plantear una arquitectura simple y extensible que permita conceptuar mediante clases y objetos las funcionalidades asociadas con un sistema de software educativo.

Descripción de la propuesta

La propuesta se caracteriza por los siguientes aspectos:

- Explica las funcionalidades asociadas con un software educativo que hace uso de la simulación para la enseñanza de estructuras de datos en programación.

- Aporta un patrón detallado para el futuro desarrollo de un software educativo para la Universidad Centroccidental "Lisandro Alvarado".

- Considera el uso de tecnologías abiertas, posibilitando su implementación sobre cualquier tipo de plataforma.

- Es flexible respecto a su tiempo y espacio.

Estructura del Modelo Propuesto

El modelo propuesto se fundamenta de acuerdo la metodología planteada por Díaz (2003), la cual es una adaptación de la metodología RUP, apoyada en el Modelo Sistémico de Calidad (MOSCA). Entre las fases contempladas por el modelo están:

Fase de Inicio.

Componentes del Proceso:

1. Funcionalidades del Sistema.
2. Análisis de Requerimientos.

Fase de Elaboración.

Componentes del Proceso:

3. Análisis del Sistema.
4. Diseño del Sistema.

En la fase de diseño la mayor parte de la información generada e expresada por símbolos gráficos y conexiones, para ello, se utilizó la herramienta Visual Paradigm 5.2 Edición Comunidad.

Características del grupo al que va dirigido:

Está dirigido a Analistas de Sistemas, Ingenieros en Informática, Computación y afines, interesados en el desarrollo de un software educativo para la enseñanza de estructuras de datos en programación, y muy especialmente al personal que integra el Proyecto Universidad Virtual de la Universidad Centroccidental "Lisandro Alvarado".

Fase de Inicio

1. Funcionalidades del Sistema

Una especificación textual (ver Anexo M) para la primera versión del sistema SEDPRO, considerando todas las funcionalidades y problemas obtenidos mediante las encuestas y evaluaciones a los estudiantes de las Prácticas Complementarias de programación, podría ser la siguiente:

Se plantea diseñar un software educativo que emplee la simulación como estrategia en el proceso de enseñanza – aprendizaje, permitiendo a los estudiantes de programación ejercitar con estructuras de datos para afianzar el conocimiento sobre las mismas. Para ello cada estudiante (persona) para poder utilizar el sistema, debe ser un usuario registrado, por lo tanto, el sistema debe contemplar un proceso de validar usuario para que este pueda tener acceso al mismo. El sistema debe permitir al usuario ejercitar con arreglo y ejercitar con registro; para la ejercitación con arreglo se debe contemplar ejercitar ordenamiento burbuja, ejercitar búsqueda lineal y ejercitar búsqueda binaria. Cada vez que un usuario tenga acceso al sistema, este debe registrar el uso a través de manejador de BD que, a posteriori, permita a un profesor consultar estadísticas sobre el uso de la herramienta por parte de sus estudiantes. Para un mejor desenvolvimiento del sistema debe existir un operador del sistema que se encargue de actualizar usuarios registrados y hacer mantenimiento al sistema de modo que se pueda configurar el ambiente.

A continuación se presentan algunas de las características que debe tener el modelo, el cual podría conllevar a desarrollar e implantar una herramienta que permita:

- ✓ Ser accesible para los estudiantes sin las limitaciones del aula, por lo que la herramienta debería enfocarse hacia el e-Learning y para ello se plantea un acceso vía Internet usando la RedUCLA.

- ✓ Cumplir con la Funcionalidad, ayudando a que el estudiante aprenda haciendo de manera que el porcentaje de retención del conocimiento sea lo más alto posible: aprendizaje significativo.

- ✓ Cumplir con la Usabilidad, por lo tanto, debe ayudar a que el estudiante se sienta motivado y logre afianzar los conocimientos acerca de las estructuras de datos en programación, por lo tanto debe generar actividades interactivas que motiven y mantengan la atención, actividades que deben ser variadas y que respondan a los diversos estilos de aprendizaje. Ahora para que el software educativo motive el aprendizaje, es fundamental que el material educativo sea atractivo y fácil de aprender. Como indica González (2004), la facilidad de aprendizaje es la medida en que el usuario novel comprende cómo utilizar inicialmente el sistema y cómo a partir de esta utilización llegar a un máximo nivel de conocimiento y uso del sistema.

- ✓ Cumplir con la Fiabilidad, debido a que es importante que el producto funcione bajo las condiciones establecidas y mantenga un nivel específico de rendimiento para garantizar un ambiente de aprendizaje adecuado.

- ✓ Ser sintetizable, de manera que el estudiante pueda evaluar los efectos de las operaciones anteriores al estado actual (capacidad de captar los cambios de estado que produce cada operación), por lo tanto, el software debe ejecutar el procesamiento paso a paso permitiendo al estudiante controlar la simulación (avanzar y retroceder cuando lo desee) de tal forma que pueda comprender mejor los tópicos estudiados.

- ✓ Ser familiar, de manera que el estudiante pueda tener una interacción efectiva. Por lo tanto, el software debe presentar una interface sencilla mostrando elementos conocidos por los estudiantes.

- ✓ Ser consistente, de manera que el estudiante pueda usar los mecanismos siempre de la misma manera. Por lo tanto, el software debe presentar similitudes de operación en cada uno de los tópicos cubiertos.

- ✓ Ser recuperable, brindándole posibilidad al estudiante de corregir una acción una vez reconocido un error. Por lo tanto, el software debe presentar alternativas que permitan corregir los errores de los estudiantes.

- ✓ Brindar mecanismos de soporte, los cuales son recursos de ayuda y forma en que el estudiante puede utilizarlos. Estos mecanismos de soporte deben tener: a) Disponibilidad, posibilidad de consultar la ayuda en cualquier momento, sin tener que salir de la aplicación; b) Precisión y detalle, medida en que la ayuda cubre todo el sistema, con concisión; c) Consistencia, en términos de contenidos, terminología y estilo; d) Robustez, que soporte más que el sistema, en términos de funcionamiento; e) Flexibilidad, en que medida permite interactuar de manera adecuada a las necesidades del usuario; f) No obstructiva, que no impida el uso normal de la aplicación; g) Organización del texto de ayuda, lenguaje, longitud de frase y párrafo, cantidad de texto, espacios en blanco, gráficos e iconos.

- ✓ Ser comunicativo, es decir, el mensaje del software (contenido) debe ser significante para el estudiante. Por lo tanto, debe tomarse en cuenta: a) Estética, las formas elegidas son visualmente agradables, manteniendo su sentido comunicativo; b) Integración, están integrados entre sí los lenguajes verbales y figurativos; c) Innovación, en qué medida son innovadoras las formas de presentación; d) Adecuación, los códigos verbales y figurativos son

descifrables por los usuarios, facilitan la comprensión; e) Densidad, la densidad de la información ofrecida (en cada pantalla) es excesiva, adecuada, escasa.

En relación a estas especificaciones es muy importante considerar los siguientes aspectos:

- **Estructuras de Datos**: Es un tipo de dato construido a partir de otros tipos de datos, los cuales pueden ser homogéneos (del mismo tipo) o heterogéneos (de diferentes tipos).

- **Arreglo**: Es una estructura de datos en la que se almacena en posiciones de memoria continua una colección de elementos del mismo tipo, donde el acceso a cada uno de ellos se hace a través del nombre (único) de la variable del arreglo seguido de un índice o subíndice.

- **Registro**: Es una estructura de datos en la que se almacena una colección de elementos que pueden ser del mismo tipo o de tipo diferentes, donde el acceso a cada uno de ellos (denominados campos) se hace a través del nombre (único) de la variable del registro seguido de un identificador del campo.

- **Ordenamiento Burbuja**: Es un algoritmo no recursivo que permite ordenar los elementos dentro de un arreglo. Básicamente recorre el arreglo intercambiando los elementos adyacentes que estén desordenados. Recorre el arreglo tantas veces hasta que ya no haya cambios. Prácticamente lo que hace es tomar el elemento mayor y lo coloca en las últimas posiciones o tomar el menor y colocarlo en las primeras posiciones.

- **Búsqueda Lineal**: Es un algoritmo que permite buscar un elemento dentro de un arreglo. Consiste en recorrer el arreglo elemento a elemento e ir comparando con el valor buscado (clave). Se empieza con la primera posición del arreglo y se observa una posición tras otra hasta que se encuentra el elemento buscado o se han visto todas las posiciones.

- **Búsqueda Binaria**: Es un algoritmo que permite buscar un elemento dentro de un arreglo ordenado. El proceso comienza comparando el elemento central del arreglo con el valor buscado. Si ambos coinciden finaliza la búsqueda. Si no ocurre así, el elemento buscado será mayor o menor en sentido estricto que el central del arreglo. Si el elemento buscado es mayor se procede a hacer búsqueda binaria en el sub-arreglo superior, si el elemento buscado es menor que el contenido de la casilla central, se debe cambiar el segmento a considerar al segmento que está a la izquierda de tal sitio central.

- **Estudiante**: Persona inscrita en una asignatura de un pensum de estudios de un programa del Decanato de Ciencias y Tecnología, y que cumple con los deberes y derechos que le confiere el reglamento interno de la Universidad.

- **Profesor**: Persona perteneciente a varias áreas de conocimiento de un departamento y encargado de administrar una o más secciones de una o más asignaturas.

- **Usuario Registrado**: Es todo aquel miembro registrado en el sistema. El sistema debe permitir almacenar los datos de los usuarios en forma persistente, entre estos datos están: Login, clave del usuario, cédula, nombres y apellidos.

2. Análisis de Requerimientos

El análisis de requerimientos consiste en definir los casos de uso para el sistema, los cuales describen lo que el sistema SEDPRO proporcionará en términos de funcionalidad. El desarrollo de los casos de uso proviene de leer y analizar las especificaciones detalladas en la sección anterior, las cuales se relacionan con las funcionalidades asociadas al software educativo.

Diccionario de Actores

Los actores del sistema SEDPRO fueron identificados como:

- **Persona**: Es un supertipo del cual todos los actores humanos heredan.

- **Usuario Registrado**: Es la persona que posee un registro de identificación en el sistema, el cual le permite interactuar con él.

- **Profesor**: Es la persona que tiene la oportunidad de consultar las estadísticas sobre el uso de la herramienta por parte de sus estudiantes

- **Operador del Sistema**: Es la persona que tiene la responsabilidad de hacerle mantenimiento al sistema actualizando los usuarios registrados y configurando el ambiente.

Diccionario de Casos de Uso

Basados en los actores y las necesidades planteadas en los requerimientos del sistema SEDPRO, fueron identificados los siguientes casos de uso:

- **Validar Usuario**: Este Caso de Uso es iniciado por un usuario registrado. Permite verificar el usuario y darle o no acceso al sistema.

- **Ejercitar con Estructuras de Datos**: Este Caso de Uso es iniciado por un usuario registrado. Ofrece la capacidad de ejercitar con las estructuras de datos arreglos y registros.

- **Ejercitar con Arreglo**: Este Caso de Uso es iniciado por un usuario registrado. Ofrece la capacidad de ejercitar la estructura de datos arreglo, mediante los procesos ejercitar ordenamiento burbuja, ejercitar búsqueda lineal y ejercitar búsqueda binaria.

- **Ejercitar Ordenamiento Burbuja**: Este Caso de Uso es iniciado por un usuario registrado. Ofrece la capacidad de ejercitar a través de la simulación con el Algoritmo Burbuja de ordenamiento de Arreglos.

- **Ejercitar Búsqueda Lineal**: Este Caso de Uso es iniciado por un usuario registrado. Ofrece la capacidad de ejercitar a través de la simulación con el Algoritmo de Búsqueda Lineal.

- **Ejercitar Búsqueda Binaria**: Este Caso de Uso es iniciado por un usuario registrado. Ofrece la capacidad de ejercitar a través de la simulación con el Algoritmo de Búsqueda Binaria.

- **Ejercitar con Registro**: Este Caso de Uso es iniciado por un usuario registrado. Ofrece la capacidad de ejercitar la estructura de datos registro.

- **Registrar el Uso**: Este Caso de Uso es iniciado por el Caso de Uso Ejercitar con Arreglo o Ejercitar con Registro. Permite hacer un registro del uso del sistema.

- **Consultar Estadísticas**: Este Caso de Uso es iniciado por el Profesor. Ofrece la capacidad de consultar las estadísticas sobre el uso de la herramienta por parte de los estudiantes.

- **Hacer Mantenimiento al Sistema**: Este Caso de Uso es iniciado por el Operador del Sistema. Ofrece la capacidad de actualizar usuarios y modificar las configuraciones del sistema.

- **Actualizar Usuarios Registrados**: Este Caso de Uso es iniciado por el Operador del Sistema. Permite crear, modificar, eliminar y consultar los datos de los usuarios registrados.

- **Configurar el Ambiente**: Este Caso de Uso es iniciado por el Operador del Sistema. Ofrece la capacidad de actualizar las configuraciones del sistema.

Complementariamente, en el Análisis de Requerimientos un artefacto imprescindible que permite la documentación del sistema es el Diagrama de los Casos de Uso. Para permitir el entendimiento de los requerimientos del sistema, a continuación se presenta la Figura 04 que representa el Diagrama de Casos de Uso del sistema SEDPRO:

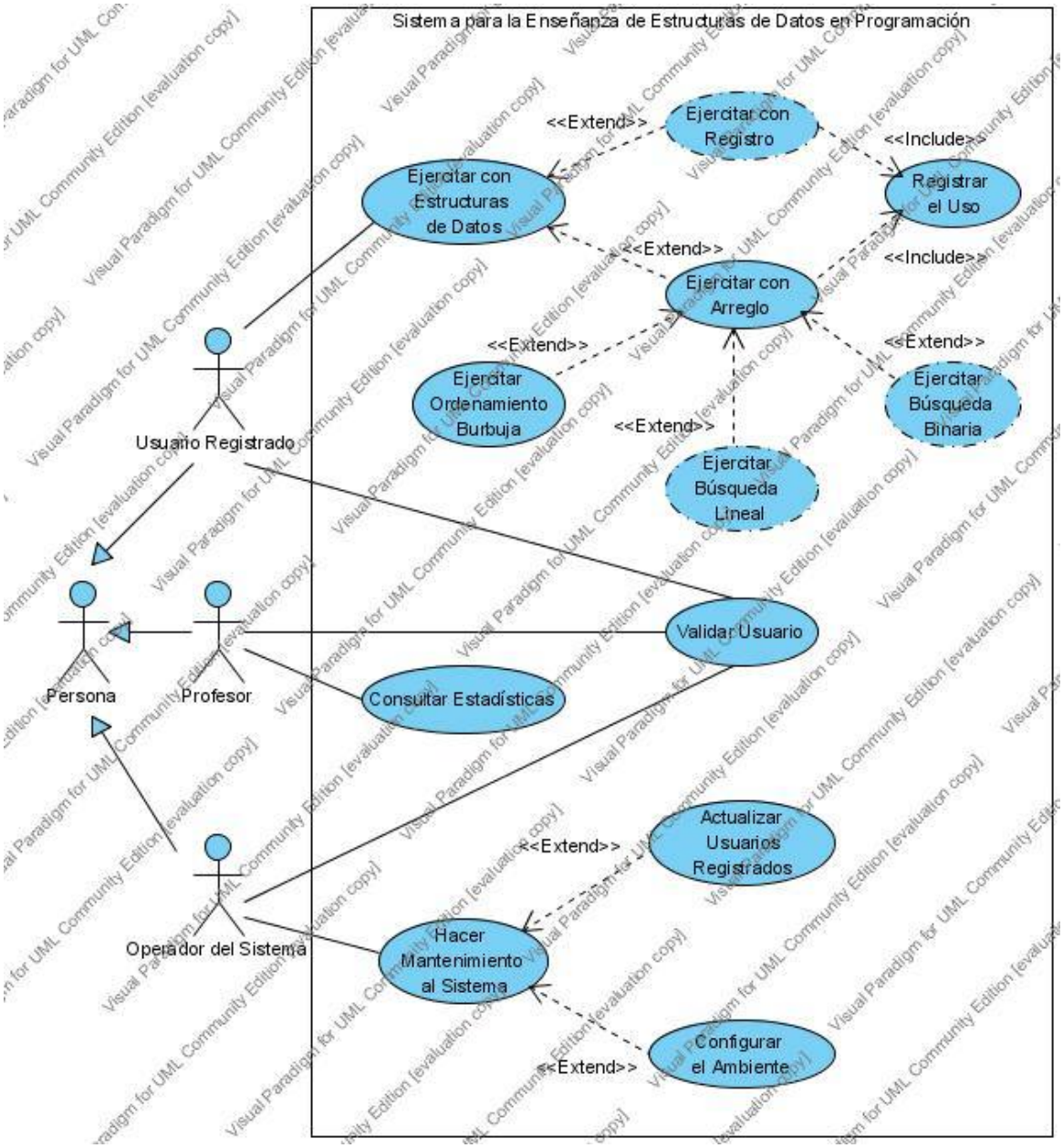


Figura 04: Diagrama de Casos de Uso del sistema SEDPRO.
Fuente: El autor de la investigación.

Fase de Elaboración.

3. Análisis del Sistema

El objetivo del Análisis del Sistema es capturar todos los requerimientos del sistema y elaborar un modelo que defina las clases del dominio del sistema. Para lograr este objetivo se tomaron las especificaciones del sistema y los Casos de Uso creados en la fase de requerimientos a fin de encontrar que "Conceptos" deben ser utilizados por el sistema y en base a esto se definieron las clases de tipo entidad, interfaz y control.

Clases Entidad

Las clases de tipo entidad en el sistema SEDPRO, son definidas con el estereotipo << **Entity** >>; esto significa que los objetos de la clase son parte del dominio del problema y deben ser almacenados persistentemente en el sistema.

Las clases de tipo entidad identificadas junto con su descripción se muestran a continuación:

- **Estructura:** Representa la información general de una Estructura de Datos en programación.
- **Arreglo**: Representa la información de la estructura de datos Arreglo.
- **Registro**: Representa la información de la estructura de datos Registro.
- **Persona**: Es un supertipo que define los atributos de una persona de la cual heredan los usuarios registrados en el sistema.
- **Usuario**: Representa a una persona que tiene los privilegios para acceder y usar el sistema.
- **Uso**: Representa la información sobre el uso del sistema SEDPRO.

Clases Control

Con respecto a este tipo de clases, se definieron para controlar el flujo de eventos del sistema. Las clases de tipo control en el sistema son definidas con el estereotipo **<< Control >>** y representan la lógica de secuenciación de los casos de uso creados en el análisis de los requerimientos.

Las clases de tipo control identificadas junto con su descripción se muestran a continuación:

- **Acceso**: Esta clase proporciona la lógica de secuenciación para validar el acceso del usuario al sistema, como se representa en el caso de uso –Validar Usuario-.
- **ILogout**: Esta clase proporciona la lógica de secuenciación para cerrar la sesión de un usuario.
- **ListadoUso**: Representa la clase que proporciona la lógica de secuenciación para visualizar las estadísticas de uso de los usuarios.
- **ManejadorBD**: Esta clase proporciona la lógica de secuenciación para acceder a los servicios ofrecidos por un manejador de base de datos que mantiene toda la información referente a los usuarios registrados.

Clases Interfaz

Con respecto a este tipo de clases, se definieron para definir ventanas y cuadros de diálogo como interfaz a los actores del sistema. Las clases de tipo Interfaz en el sistema son definidas con el estereotipo **<< Boundary >>**.

Las clases de tipo Interfaz identificadas junto con su descripción se muestran a continuación:

- **Browser**: Representa la interfaz que permite indicar la dirección URL del servidor de aplicaciones donde se encuentra alojado el sistema.
- **IAcceso**: Representa la interfaz de captura de los datos de entrada al sistema.
- **IConfigurar**: Representa la interfaz que permite configurar los datos del ambiente.
- **IEdicionUsuario**: Representa la interfaz que permite efectuar las operaciones de actualización de los usuarios que utilizan el sistema.
- **IListadoUso**: Representa la interfaz que permite consultar las estadísticas de uso de los usuarios.
- **IMantenimiento**: Representa la interfaz que observa el operador del sistema cuando ingresa al mismo.
- **IBurbuja**: Representa la interfaz que observa el usuario cuando ejercita con el ordenamiento de arreglo.

Una vez definidas las diversas clases del sistema, en esta fase se tomaron en cuenta los siguientes elementos requeridos para elaborar la documentación tanto en la fase de análisis como en la fase de diseño.

Paquetes

Para separar las clases de tipo Entidad, Interfaz y Control, se debe considerar el uso de los paquetes definidos por el lenguaje UML, por lo tanto, en esta fase la figura 05 que a continuación se presenta, permite expresar la especificación de los paquetes del sistema SEDPRO.

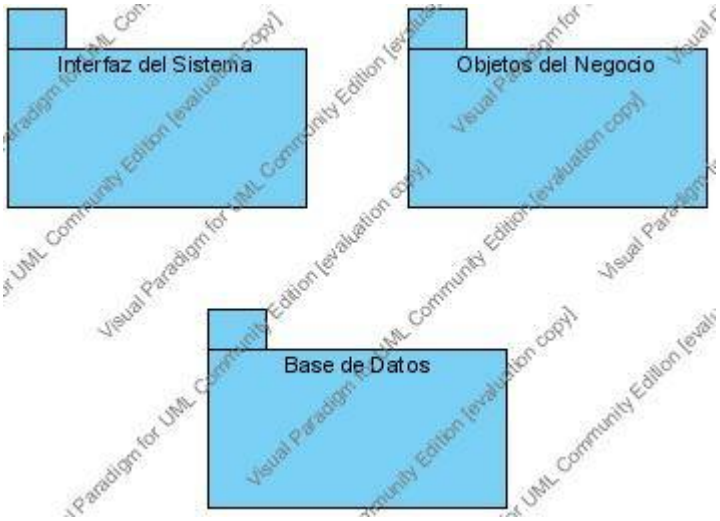


Figura 05: Especificación de los paquetes del sistema SEDPRO en la fase de análisis
Fuente: El autor de la investigación.

Diagramas de Secuencia

Aunque se consideró describir el comportamiento dinámico del sistema SEDPRO, objeto de esta investigación, esta fase de análisis no presenta los artefactos necesarios para llevar a cabo las especificaciones desde esta perspectiva dinámica del sistema. No obstante, la fase de diseño describe la vista dinámica del sistema. Consecutivamente, se presentan en el Anexo R los Diagramas de Secuencia del sistema SEDPRO.

Relaciones entre Clases

Las relaciones entre las clases son mostradas en el Diagrama de Clases, el cual considera los requerimientos del sistema. Entre las relaciones consideradas tenemos:

asociación binaria, herencia, dependencia y agregación, (ver Figura 06) donde se presenta una parte del diagrama de clases en el que se detallan las relaciones entre clases. Consecutivamente, para observar todas las relaciones entre clases ver el Anexo P.

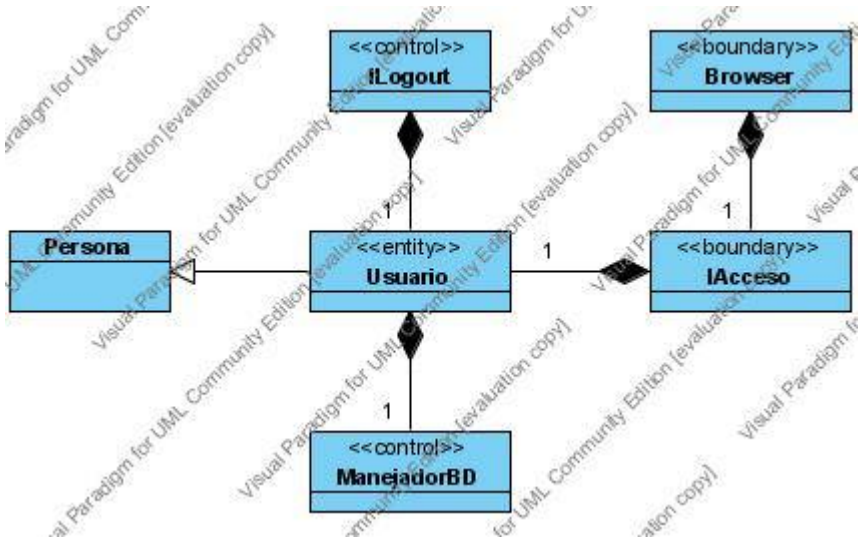


Figura 06: Relaciones entre clases contempladas por el sistema SEDPRO.
Fuente: El autor de la investigación.

4. Diseño del Sistema

La fase de diseño del sistema expande y detalla las especificaciones consideradas en la fase de análisis, tomando en cuenta todas las implicaciones y restricciones técnicas obtenidas en la investigación. A continuación se presenta la Figura 07, donde se muestra la relación que existe entre los paquetes del sistema SEDPRO.

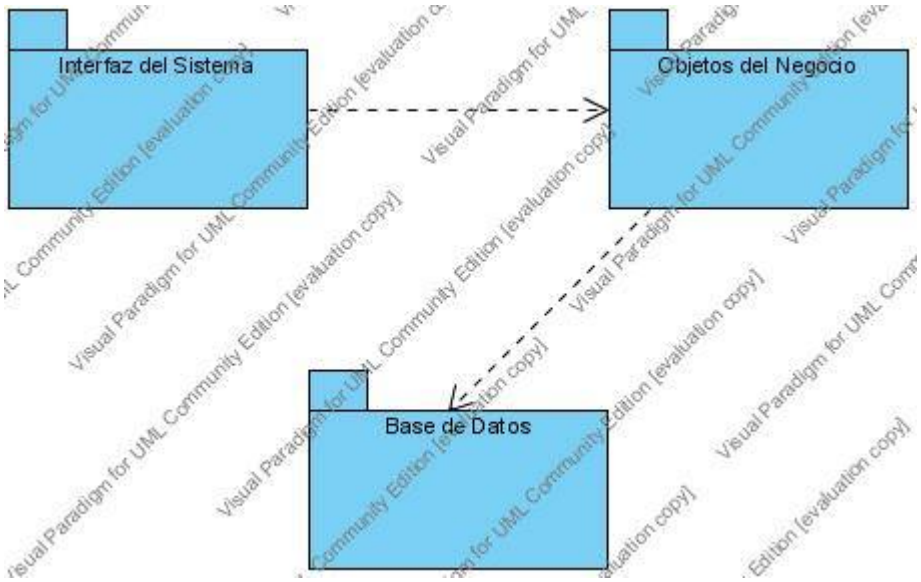


Figura 07: Especificación de los paquetes del sistema SEDPRO en la fase de diseño
Fuente: El autor de la investigación.

Durante esta etapa se expanden los paquetes del sistema, incluyendo sus dependencias y mecanismos de comunicación. Estos paquetes se detallan de tal forma que las clases sean definidas de manera suficiente para dar especificaciones precisas y claras al programador que las codifique. Los paquetes son definidos tomando en cuenta la separación entre áreas funcionales y áreas técnicas, así mismo, se establecen las reglas de dependencia entre los paquetes, de tal forma que se eviten las

dependencias bidireccionales entre ellas y de identificar la necesidad de librerías estándar que puedan ser usadas y ayuden a simplificar el trabajo. A continuación se explican brevemente estos paquetes, así mismo se muestra la clase para cada uno de ellos.

Paquete de Base de Datos

Debido a que el sistema debe almacenar sus objetos persistentes, este paquete incluye la clase denominada ManejadorBD, la cual se encarga de llevar a cabo la interacción con el manejador de base de datos a través de la ejecución de las cuatro operaciones básicas como lo son: insertar, modificar, consultar y eliminar. La figura 08 detalla las especificaciones de la clase ManejadorBD.

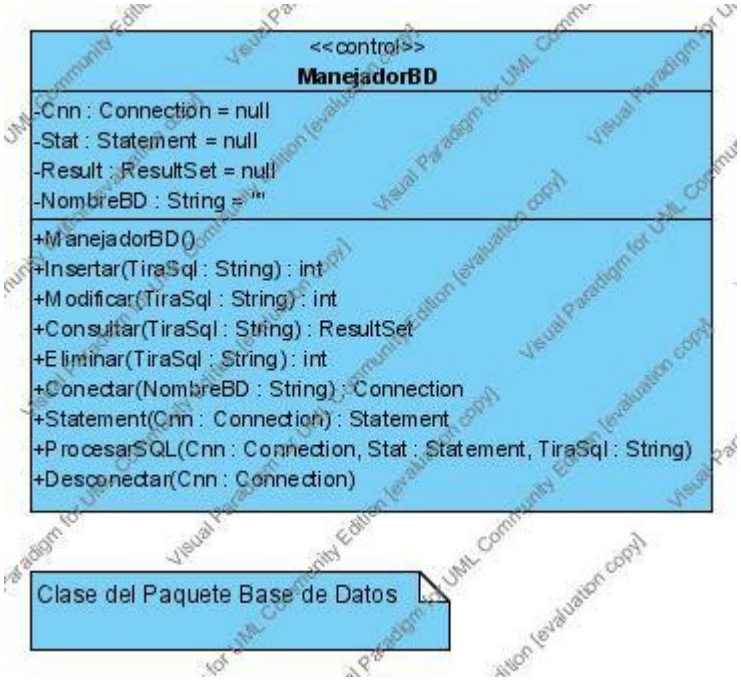


Figura 08: Especificación del paquete Base de Datos del sistema SEDPRO.
Fuente: El autor de la investigación.

Paquete de Interfaz del Sistema

Se basa en las capacidades proporcionadas por los navegadores de Internet, ya que la definición de sus clases presenta los servicios y la información del sistema requerida por sus actores. La figura 09 muestra las especificaciones de la clase interfaz "IBurbuja" considerada de vital importancia en la ejercitación de ordenamiento Burbuja.

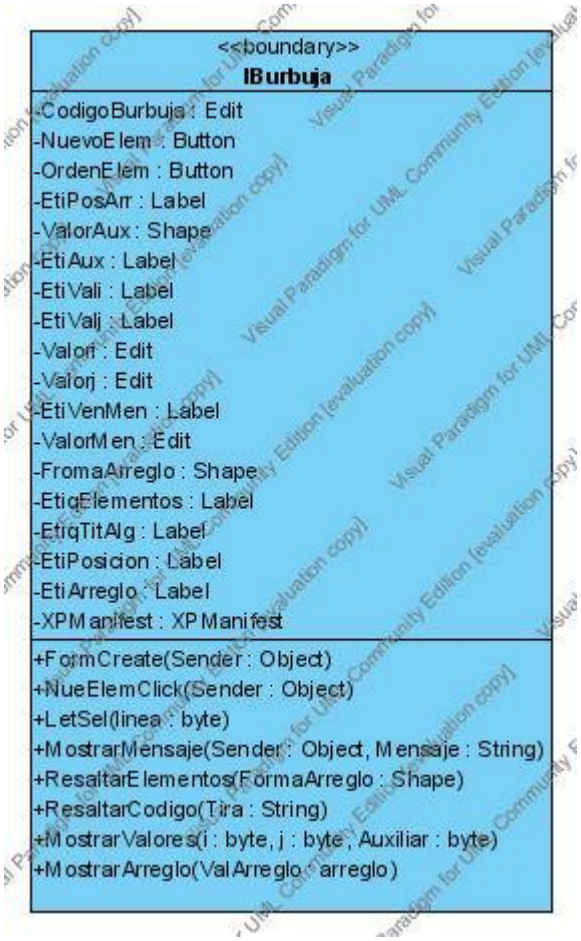


Figura 09: Especificación de la clase IBurbuja del sistema SEDPRO.
Fuente: El autor de la investigación.

Paquete de Objetos del Negocio

Este paquete contempla las clases de tipo entidad y control definidas durante el análisis. Para la fase de diseño estas clases son detalladas, relacionadas y expandidas. Se considera que este paquete representa la descripción detallada del sistema ya que especifica las clases incluyendo sus relaciones y comportamientos. La figura 10 muestra las especificaciones de la clase entidad "Arreglo", la cual es considerada de vital importancia para el sistema SEDPRO.

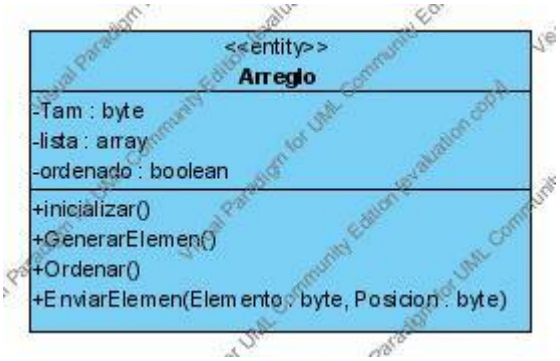


Figura 10: Especificación de la clase Arreglo del sistema SEDPRO.
Fuente: El autor de la investigación.

Luego de observar las clases más importantes contenidas en cada uno de los paquetes del sistema SEDPRO y teniendo en cuenta que el alcance de esta investigación no contempla la fase de implementación, se presenta de manera adicional los aspectos tales como: la interfaz del usuario y el diagrama de despliegue del sistema.

Interfaz del Usuario

Una de las tareas que se debe ejecutar durante la fase de diseño es la creación de la interfaz del usuario. Aunque el alcance de este estudio no considera el desarrollo del sistema, se diseño una interfaz de usuario para el Caso de Uso Ejercitar Ordenamiento Burbuja, la cual se utilizó para la creación del prototipo, generando como resultado la interfaz IBurbuja que se muestra en la figura 11:

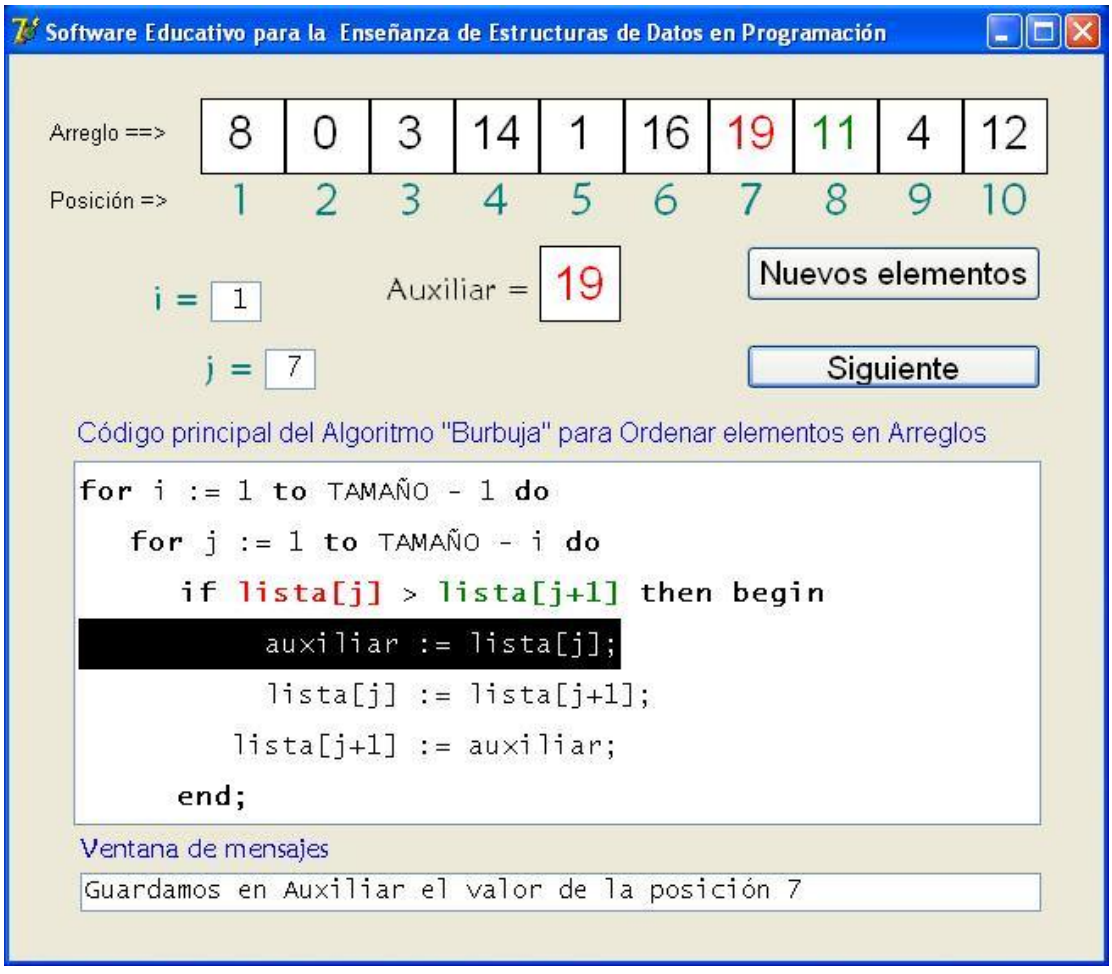


Figura 11: Interfaz IBurbuja del sistema SEDPRO.
Fuente: El autor de la investigación.

Diagrama de despliegue

El objetivo de mostrar un diagrama de despliegue es describir la arquitectura física para el sistema SEDPRO. La figura 12 presenta este diagrama en el que se describen los elementos tales como servidores, componentes instalados, conexiones y protocolos de red utilizados por el sistema.

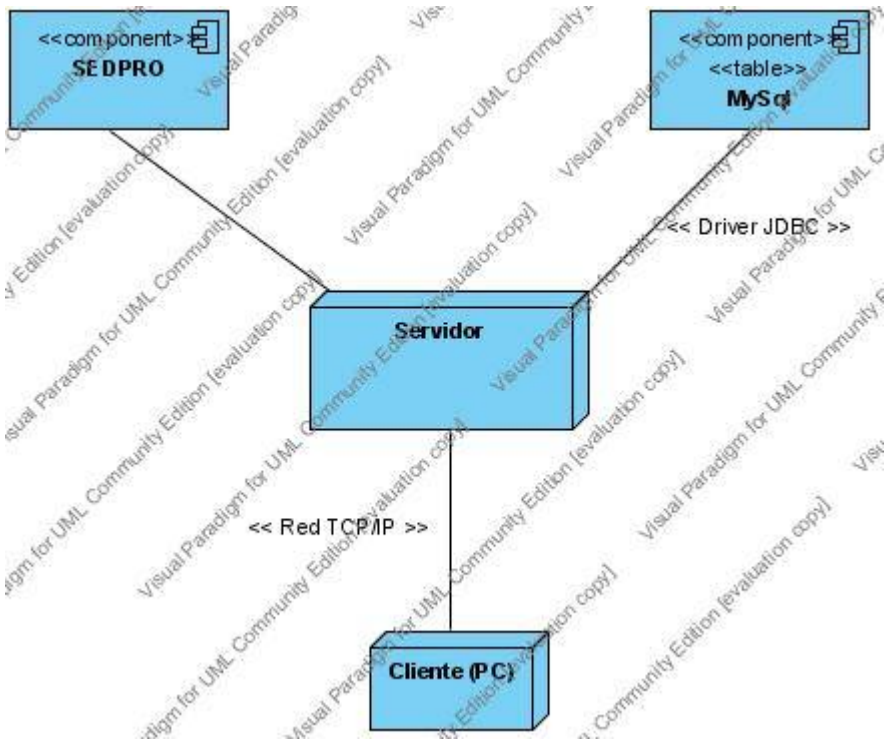


Figura 12: Diagrama de Despliegue del sistema SEDPRO.
Fuente: El autor de la investigación.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Luego de realizar el diseño de un modelo de software educativo para la enseñanza de estructuras de datos en programación, el autor de esta investigación llegó a las siguientes conclusiones que dan respuestas a las interrogantes planteadas:

1. Un alto porcentaje de estudiantes no se sienten satisfechos con las enseñanzas recibidas sobre los conceptos básicos y operaciones imprescindibles para definir y manipular las estructuras de datos, en especial los arreglos. Este grado de insatisfacción puede ser causante de una baja motivación en el estudiante para conseguir un mayor rendimiento en la asignatura.

2. Quedó evidenciado a través de las evaluaciones realizadas en este estudio, que los estudiantes conocen el concepto sobre estructuras de datos y arreglos, pero sobre la declaración, operaciones y tratamiento de los mismos, el nivel de conocimientos está entre deficiente y regular, conocimientos estos imprescindibles para resolver problemas más complejos como los planteados en las evaluaciones de la unidad curricular de Programación I, lo que trae como consecuencia el bajo rendimiento en la asignatura.

3. La mayoría de los estudiantes estuvieron de acuerdo en que el software simulador de ordenamiento de arreglos es fácil de comprender y utilizar, y que al utilizarlo se puede aprender de una manera más fácil el algoritmo de ordenamiento Burbuja, por lo que el uso de un software educativo que haga uso de la simulación como el planteado en esta investigación, podría aumentar el grado de satisfacción y conseguir una mayor motivación de los estudiantes hacia la asignatura.

4. Los resultados estadísticos obtenidos en la evaluación sobre el algoritmo Burbuja, avalan que los estudiantes que ejercitaron con el software simulador de ordenamiento de arreglos consiguieron una mejor asimilación sobre el código y proceso del algoritmo, lo que permite inferir que el modelo de software educativo propuesto podría ayudar a mejorar el rendimiento académico en el tema de las estructuras de datos.

Complementariamente a estas conclusiones, se presentan un conjunto de hallazgos que guardan relación con aspectos relacionados al estudio. A continuación se presentas estas conclusiones en base a los hallazgos encontrados:

1. Durante los últimos 5 años, en casi la totalidad de los lapsos académicos, ha habido un mayor porcentaje de estudiantes aplazados que estudiantes aprobados en la asignatura Programación I tanto en la carrera Ingeniería en Informática como en Análisis de Sistemas, mientras que los promedios por lapso en ambas carreras no muestran una tendencia sino que bajan y suben erráticamente.

2. Sólo en los estudiantes de condición 2, es decir, los que repiten por tercera vez, el porcentaje de aprobados supera a los aplazados, esto probablemente debido a que cursan menos asignaturas y a un mayor esfuerzo por los estudiantes por el hecho

de estar en condición para Régimen de Repitencia, lo cual implicaría ser retirado de la Universidad por espacio de un (1) período lectivo.

3. En los estudiantes de condición 3 o mayor, que reingresaron a la UCLA luego de habérseles aplicado el Régimen de Repitencia, se observó que un alto porcentaje son de nuevo aplazados, posiblemente a una baja motivación por el hecho de haber perdido 1 o más periodos lectivos.

4. En los últimos 10 lapsos, el promedio de notas por lapso de los estudiantes de Programación I de Ingeniería en Informática ha sido mayor que los de Análisis de Sistemas, registrándose diferencias estadísticamente significativas en todos los lapsos. Estas diferencias pueden ser debidas al perfil del estudiante de Ingeniería en Informática que dedica más de su tiempo a los estudios en contraste con el perfil del estudiante de Análisis de Sistemas que comparte su tiempo con un trabajo. Por otro lado, hay que destacar que los estudiantes de Ingeniería en Informática reciben 5 horas de clases de la asignatura programación mientras que los de Análisis de Sistemas sólo reciben 4 horas.

5. Se encontraron altas diferencias en los promedios de notas de diferentes secciones de un mismo lapso, tanto para Ingeniería en Informática como para Análisis de Sistemas. Asimismo se observó que una misma sección de Ingeniería en Informática tiene el mayor número de veces con el promedio más alto por lapso, y en el caso de Análisis de Sistemas, una misma sección tiene el mayor número de veces con el promedio más bajo por lapso. Estas diferencias podrían ser originadas en diferentes estrategias de enseñanza – aprendizaje por parte de los profesores.

6. La metodología utilizada para el desarrollo de software educativo propuesta por Díaz (2003), permitió crear el modelo de forma lógica y ordenada al igual que utilizar RUP, pero en este caso se consigue producir un producto educativo de

calidad, ampliado y enriquecido con los parámetros educativos propuestos por profesionales del área de la educación.

7. El uso de muchos de los aspectos considerados por González (2004) para la evaluación de software educativo, conllevó a ser más preciso en los detalles requeridos para lograr un producto educativo de calidad.

8. Con la herramienta de diseño Visual Paradigm para UML 5.2 Edición Comunidad, se pudo diseñar el modelo de una manera fácil y sencilla generando los artefactos pertinentes en cada fase de la metodología. Sin embargo, para poder imprimir los Casos de Uso y algunos otros reportes, se requiere adquirir la Edición Estándar a través de una licencia.

Recomendaciones

Tomando en cuenta las conclusiones y los hallazgos planteados en este estudio, el autor recomienda:

1. Desarrollar e implantar de manera progresiva el modelo de software educativo propuesto, ya que a través del uso del prototipo desarrollado se demostró que los estudiantes de programación pueden lograr afianzar sus conocimientos sobre estructuras de datos.

2. Aplicar estudios a otros tópicos dentro de la asignatura Programación I, e incluso a otros tópicos dentro de otras asignaturas, con la finalidad de Desarrollar Modelos de Software Educativos que hagan uso de la Simulación para facilitar la comprensión de los temas relacionados.

3. Establecer una comisión para estudiar y desarrollar estándares propios para el desarrollo de Software Educativo que haga uso de la simulación.

4. Promover a través de diferentes medios el desarrollo de Software Educativo como el modelo propuesto, para las diversas asignaturas de las diferentes carreras que se imparten en la misma.

5. Facilitar a Profesores y Estudiantes, el acceso a determinado espacio de almacenamiento en las computadoras que se utilizan como Servidores en la UCLA, con el objetivo de que se puedan realizar las pruebas necesarias para desarrollar Software Educativo con uso de la simulación para las diversas asignaturas de las diferentes carreras que se imparten en la misma.

6. Crear una comisión para evaluar el Software Educativo con Simulación que desarrollen tanto Profesores como Estudiantes, con la finalidad de certificar la calidad del mismo antes de ser publicado para su uso.

7. Hacer un estudio en la unidad curricular de Programación I respecto a las evaluaciones, donde primero se plantee aplicar evaluaciones sencillas sobre la parte básica teórica, declaración, operaciones y tratamiento de los arreglos antes de plantear problemas más complejos con los mismos.

8. Realizar nuevas investigaciones para identificar otras variables que incidan en el bajo rendimiento académico y para determinar si los problemas de conocimientos encontrados en los estudiantes tienen su origen en la base que traen desde la asignatura Introducción a la Computación

REFERENCIAS BIBLIOGRAFICAS

ALAS, A. y otros, 2002. Las Tecnologías de la Información y de la Comunicación en la Escuela. Editorial Graó.

ALDRICH, C. 2004. Simulations and the Future of Learning. Published by Pfeiffer.

ALEPHZERO, 2000. Revista de Divulgación y Educación Científica. Número 18, Año 5. Junio-Julio 2000. URL: http://aleph.cs.buap.mx/az18/ (Consulta: Noviembre 04, 2005)

ALVAREZ, L. 1999. Modelaje de Sistemas de Software usando el UML. Trabajo de Ascenso. Universidad Centroccidental "Lisandro Alvarado".

ARNAU, C. 1980. Criterios Metodológicos de la investigación. Resumen del libro de metodología de la investigación holística de Jacqueline Hurtado de Barrera.

BOADA y otros 2005. Entorno Virtual de Ayuda a la Docencia de un Curso de Programación Básica. Departamento de Informática y Matemática Aplicada. Universidad de Girona. URL: http://acme.udg.es/cidui1.pdf
(Consulta: Noviembre 11, 2005)

BOOCH, G. 1998. Software Architecture and the UML. URL: http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html
(Consulta: Noviembre 17, 2005)

CARREON, G 2004. Aprendizaje Significativo y Efectivo de Diseño de Programas. Simposio La Investigación en la Facultad de Ingeniería 2004.
URL: http://www.finam.mx/simposio_investigacion2dic04/aprendizaje_extenso.html
(Consulta: Noviembre 08, 2005)

COTA, A. 1994. "Ingeniería de Software". Soluciones Avanzadas. Julio de 1994. pp. 5-13.
URL: http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html#Cota1994
(Consulta: Noviembre 11, 2005)

DELGADO, C. 2004. La simulación Aplicada a la enseñanza de la Dirección de Proyectos. URL: http://www.degerencia.com/articulos.php?artid=636
(Consulta: Noviembre 13, 2005)

DIAZ, A. y otros 2003. Propuesta de una metodología de desarrollo de Software educativo bajo un enfoque de calidad sistémica (MOSCA). Presentado en el IV Congreso Multimedia Educativo, Universidad de Barcelona, España.
URL: http://www.academia-interactiva.com/ise.pdf (Consulta: Agosto 01, 2005)

DUARTE, V. 1997. Anotaciones al concepto de Modelo, URL: http://www.virtual.unal.edu.co/cursos/ingenieria/2001619/lecciones/ljung/node1.html (Consulta: Septiembre 15, 2005)

ESTRATEGIAS DE GOBIERNO ELECTRONICO EN VENEZUELA. URL: http://portal.cnti.ve/cnti_docmgr/sharedfiles/gobiernoelectronico.pdf (Consulta: Enero 06, 2006)

GOMEZ, A. 2002. Revista espacios. Vol. 23 año 2002. URL: http://www.revistaespacios.com/a02v23n02/02230212.html#inicio (Consulta: Noviembre 10, 2005)

GONZALEZ, M. 2004. Evaluación de Software Educativo, URL: http://www.conexiones.eafit.edu.co/sobreConexiones/publicaciones/libroPdfs/capitulo14.pdf (Consulta: Agosto 06, 2005)

IEEE Std 729 1993. IEEE Software Engineering Standard 729-1993: Glossary of Software Engineering Terminology. IEEE Computer Society Press, 1993.
URL: http://es.wikipedia.org/wiki/IEEE (Consulta: Noviembre 18, 2005)

JACOBSON, I. 1998. "Applying UML in The Unified Process" Presentation Rational Software. URL: http://www.rational.com/uml como UMLconf.zip (Consulta: Noviembre 17, 2005)

KOBE, D. 2004. Aprender Haciendo.
URL: http://www.supportforfamilies.org/ pdf%20files/NL-2005-Spring-Spanish.pdf (Consulta: Agosto 20, 2005)

LAZARO, C. 2003. URL: http://vido.escet.urjc.es/docencia.html#PFCs (Consulta: Septiembre 01, 2005)

LEON, T. 1998. Solución de Sistemas de Ecuaciones Lineales Usando el Método de Gauss. Programación Numérica. Decanato de Ciencias y Tecnología. UCLA

LEWIS G. 1994. "What is Software Engineering?" DataPro (4015). Feb 1994. pp. 1-10.
URL: http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html#Lewis1994 (Consulta: Noviembre 17, 2005)

LOSADA, R. 2003. URL: http://vido.escet.urjc.es/docencia.html#PFCs (Consulta: Octubre 12, 2005)

MAILXMAIL 2005. Aprende a Programar. URL: http://www.mailxmail.com/curso/informatica/programacionestructurada/capitulo18.htm (Consulta: Noviembre 14, 2005)

MARQUES, P. 1996. El Software Educativo. URL: http://www.doe.d5.ub.es/te/any96/marques_software/marques_96.htm (Consulta: Agosto 18, 2005)

MARTIN, L. 2003. URL: http://vido.escet.urjc.es/docencia.html#PFCs (Consulta: Noviembre 01, 2005)

MICROSOFT MSDN 2004. Estructuras de Datos y Algoritmos, URL: http://www.microsoft.com/spanish/MSDN/estudiantes/algoritmica/estructuras/default.asp (Consulta: Noviembre 30, 2005)

MICROSOFT y RATIONAL 1998. A White Paper on the Benefits of Integrating Microsoft Solutions Framework and The Rational Process. Rational Software Corporation y Microsoft Corporation. Documento msfratprocs.doc
URL: http://www.rational.com/uml/papers (Consulta: Diciembre 01, 2005)

MORELLO, V. 1996. Notas de estructuras de datos. Decanato de ciencia y tecnología de la UCLA. 1996. Trabajo de ascenso.

MORRION, A. 2004. Diseño de Modelos para Educación a Distancia. Buenos Aires. Argentina. URL: http://www.fvet.uba.ar/invet/sup137.pdf (Consulta: Agosto 04, 2005)

PHILLIPS, E. 2001. Cómo Obtener un Doctorado. Manual para Estudiantes y Tutores. Editorial Gedisa. España.

SALAMO y otros 2001. Iniciativas para motivar a los alumnos de Programación. Universidad Ramón Llull. Barcelona. España. URL: http://bioinfo.uib.es/~joemiro/semdoc/sacom83.pdf (Consulta: Noviembre 05, 2005)

SALAS, R. 1995. Centro Nacional de Perfeccionamiento Médico y Medios de Enseñanza. Calle Línea e I, Vedado, Ciudad de La Habana, Cuba.
URL: http://www.bvs.sld.cu/revistas/ems/vol9_1_95/ems03195.htm
(Consulta: Octubre 16, 2005)

SENGE, P. 1994. La Quinta Disciplina.
URL:http://www.monografias.com/trabajos27/quinta_disciplina/quintadisciplina.shtml (Consulta: Noviembre 20, 2005)

SUAREZ, P. 1997. Estructuras de datos en pascal. 1997. trabajo de ascenso. UCLA.

UCLA 2001. Normativa para el desarrollo experimental del programa de cursos en línea. UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO". Barquisimeto.

UCLA 2002. Manual para la Presentación del Trabajo Conducente al Grado Académico de: Especialización, Maestría, Doctorado. UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO". Barquisimeto.

UCLA 2003. Proyecto Universidad Virtual. UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO". Barquisimeto.
URL: http://virtual.ucla.edu.ve (Consulta: Julio 30, 2005)

UNIVERSIDAD DE VALPARAISO 2005. Modelos de Software [Internet] URL: http://www.decom-uv.cl/~INF203/docs/07%20-%20TGS%20-%20CRC-OO.ppt (Consulta: Agosto 25, 2005)

Van, N. 2003. The E-Learning Fieldbook. Editorial McGraw-Hill Trade.

VARGAS R. y GUTIERREZ A., 2004. Visión retrospectiva de los principios de la programación y su impacto en la formación de ingenieros y en la calidad de software. Universidad del Valle de México. Dirección General Académica. Revista Episteme No.2. URL:http://www.uvmnet.edu/investigacion/episteme/numero1-05/enfoque/a_vision.asp (Consulta: Noviembre 06, 2005)

Visual Paradigm 2006. URL: http://www.visual-paradigm.com. (Consulta: Marzo 07, 2006)

WIKIPEDIA 2005. La Enciclopedia Libre. URL: http://es.wikipedia.org/wiki/Estructura_de_datos (Consulta: Septiembre 29, 2005)

ZAVALA, R. 2000. Diseño de un Sistema de Información Geográfica sobre Internet. Tesis de Maestría en Ciencias de la Computación. Universidad Autónoma Metropolitana-Azcapotzalco. México, D.F. En prensa.
URL: http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html#Zavala2000 (Consulta: Diciembre 04, 2005)

ANEXOS

ANEXO A

CURRICULUM VITAE

DATOS PERSONALES			
Nombre	Tulio Enrique León Alcalá	**Sexo**	Masculino
Cédula de Identidad	V - 9.272.490	**Pasaporte N°**	2076966
Lugar de Nacimiento	Cumaná – Estado Sucre	**Fecha de Nacimiento**	07-05-1964
Nacionalidad	Venezolano	**Estado Civil**	Casado
Dirección	Urb. Villa Roca III – Casa 4-07 - Cabudare. Estado Lara - Venezuela		
Teléfonos	0251-261.62.05 / 0416-353.28.77		
E-mail	teleon@ucla.edu.ve	tulio.leon@cantv.net	

INFORMACIÓN ACADÉMICA

Nivel	Instituto	Título o Grado
Universitaria	Universidad Centroccidental "Lisandro Alvarado". Barquisimeto, Edo. Lara. 1996 – 2000. 9no lugar en promoción.	Ingeniero en Informática
	Instituto Universitario de Tecnología "Antonio José de Sucre". Barquisimeto, Edo. Lara. 1988 - 1990. 1er lugar en promoción.	T.S.U. en Informática

CURSOS, TALLERES, SEMINARIOS Y CONGRESOS

Evento	Instituto	Fecha
Sist. Inf. en toma de decisiones	U.C.L.A. Bqto. - Lara	09/98–01/99
Sist. Inf. en la Economía Digital	U.C.L.A. Bqto. - Lara	Abril 1998
P.O.O. en lenguaje C++	U.C.L.A. Bqto. - Lara	Junio 1997
Congreso "Internet hoy"	Corporación INFORSOFT, C.A. Bqto.	Octubre 1996
Locución	Centro de Comunicación Audiovisual.	Sept. 1995
Análisis y cambios Procesos Org.	U.C.L.A. Bqto. - Lara	05/95–07/95
Reingeniería	U.C.L.A. Bqto. - Lara	01/95–05/95
Protección Jurídica del Software	U.C.L.A. Bqto. - Lara	Febrero 1995
Jornadas Sectoriales de Planificación	U.C.L.A. Bqto. - Lara	Dic. 1994
Sistemas de Transmisión	U.C.L.A. Bqto. - Lara	10/94–12/94
Sistemas Operativos	U.C.L.A. Bqto. - Lara	10/94–12/94
Sistemas de Información I	U.C.L.A. Bqto. - Lara	10/94–12/94
Control y Administración de Inventarios	Col. Contadores Públicos. Bqto.	Junio 1992
Sistema Operativo Xenix / Unix	I.U.T.A.J.S. Bqto. - Lara	Abril 1991
Programación en Assembler	MicroCenter. Bqto. - Lara	Marzo 1990
Programación en Clipper	MicroCenter. Bqto. - Lara	Dic. 1989
Computación y Sistemas	ILCA. Bqto. - Lara	05/85-10/88
Primeros Auxilios	Cruz Roja Bqto. - Lara	Sept. 1989
Electrónica Básica, Radio y TV	Instituto "Los Andes". Bqto. - Lara	05/85-12/85

EXPERIENCIAS Y CARGOS DESEMPEÑADOS

Cargos	Empresa	Fecha
Auxiliar Docente IV	Universidad Centroccidental "Lisandro Alvarado"	04/94-Actual
Analista - Programador	Constructora Montalca, C.A. Barquisimeto - Lara	07/97–12/98
Analista - Programador	Distribuidora CAPRENCA, C.A. Barquisimeto - Lara	04/93–11/96
Analista - Programador	Hermanos "BENEDETTO", C.A. Barquisimeto - Lara	11/92–08/95
Gerente de Informática	Líder ArrendaCars C.A. Barquisimeto - Lara	04/92-12/93
Gerente de Operaciones	TecnoMicro C.A. Barquisimeto - Lara	03/91-04/92
Analista - Programador	TecnoMicro C.A. Barquisimeto - Lara	10/89-02/91
Instructor	MicroCenter C.A. Barquisimeto - Lara	03/91-11/91
Preparador en P.E.D.	I.U.T.A.J.S. Barquisimeto - Lara	09/89-02/90
Preparador Tecn. Programación	I.U.T.A.J.S. Barquisimeto - Lara	09/88-08/89

Universidad Centroccidental "Lisandro Alvarado"

Decanato de Ciencias y Tecnología

Departamento de Sistemas

PROGRAMA INSTRUCCIONAL

PROGRAMA: INGENIERIA EN INFORMATICA	**DEPARTAMANTO:** SISTEMAS

ASIGNATURA: PROGRAMACION I.	**AREA COORDINACIÓN:** PROGRAMACIÓN
AREA CURRICULAR: CONOCIMIENTOS	**EJE CURRICULAR:** COMPUTACION

CODIGO: 3154 **SEMESTRE:** III	**CARÁCTER: OBLIGATORIA**
HORAS TEORICAS: 3 **HORAS PRACTICAS:** 2 **CREDITOS:** 4	**PRE-REQUISITOS:** 2154

PROFESORES: NORMA CASTILLO EDUARDO VARGAS ALIRIO GONZALEZ CAROL VIVAS ROSA DIAZ ERYS PIÑERO **AUXILIAR DOCENTE:** ÓSCAR GARCÍA	**COORDINADOR**: GIOVANNI TORREALBA

FECHA ELABORACION: 06- 2001	**FECHA ULTIMA REVISION**: 01- 2003	**LAPSO ACADEMICO:** 2002 – 1

FUNDAMENTACIÓN

Los progresos alcanzados en materia tecnológica durante las últimas décadas en lo que se refiere al manejo computarizado de la información, han marcado una evolución muy superior a la de cualquier otro campo de actividad. Esta evolución pasó a ser revolución con la comercialización masiva de productos tecnológicos orientados a facilitar procesos y comunicaciones de diversa índole. La aparición de nuevas herramientas de programación diseñadas en función de nuevos enfoques para el análisis y solución de problemas, la estandarización de procesos y las tendencias hacia el uso de ciertas herramientas tecnológicas, obligan a una constante revisión y estudio en esta materia. Desde esta perspectiva, es indispensable que el estudiante de ingeniería informática, a través de la secuencia de las materias de programación, desarrolle habilidades para diseñar e implementar programas eficientes, mediante el uso de las más recientes tecnologías. Estas habilidades son las que le permitirán asimilar oportunamente el vertiginoso avance tecnológico y lo motivarán a la búsqueda constante del conocimiento bajo la clara conciencia de que el uso del computador y sus tecnologías asociadas implica la ineludible tarea de la actualización.

OBJETIVO GENERAL

Suministrar al estudiante las herramientas tecnológicas necesarias para el diseño e implementación de programas eficientes y de mediana complejidad, cuyas aplicaciones estén dirigidas a diversas áreas del ámbito comercial, financiero, científico o doméstico.

UNIDAD I: MODULARIDAD	**OBJETIVO TERMINAL:** Diseñar, codificar y probar programas estrictamente modulares, de mediana complejidad. Conocer y utilizar estándares de programación.	
Duración: 10 horas	**Ponderación:** 15.15%	
OBJETIVOS ESPECÍFICOS	**CONTENIDO**	**ESTRATEGIAS DE ENSEÑANZA- APRENDIZAJE**
Aplicar las características básicas que deben estar siempre presentes en todo programa a desarrollar. Adquirir destrezas en un enfoque metodológico que permitan expresar un buen estilo de programación. Utilizar la lista de parámetros como medio de comunicación entre los diferentes subprogramas desarrollados y su importancia en el diseño de módulos reusables. Desarrollar mediante un sencillo esquema descriptivo la implementación de librerías personales (unidades). Promover pruebas efectivas para la producción de programas confiables y de mejor calidad.	1. Características de un Programa 1.1 Claro 1.2 Correcto 1.3 Completo 1.4 Modular 2. Estándares de Programación 2.1 Identificadores. 3. Subprogramas: Funciones, Procedimientos 3.1 Justificación. 3.2 Definición. 3.3 Activación 3.4 Parámetros 3.4.1 Formales y Actuales 3.4.2 De valor y de Referencia 4. Compilación Separada: Unidades. 4.1 Estructura de una Unidad 5. Alcance de constantes, tipos, variables, procedimientos, funciones y unidades 6. Prueba de Programas	- Exposición por parte del Docente. - Discusión Grupal. - Uso de multimedia - Ejercicios Prácticos.

UNIDAD II: VALIDACION	**OBJETIVO TERMINAL:** Usar y diseñar herramientas de validación en procesos interactivos y en procesos en lotes.	
Duración: 8	**Ponderación:** 12.12%	
OBJETIVOS ESPECÍFICOS	**CONTENIDO**	**ESTRATEGIAS DE ENSEÑANZA-APRENDIZAJE**
1. Definir validación: interactiva, en lote, de errores y de consistencia para valores numéricos, letras y cadena de caracteres. 2. Usar herramientas de validación existentes. 3. Definir y diseñar herramientas de validación reusables.	1. Definición de validación. 2. Enfoques de validación 2.1 Interactiva y en lotes 2.2 De rango 2.3 De consistencia 2.4 De longitud 3. Herramientas básicas del lenguaje 3.1 Lectura de caracteres 3.2 Conversiones de cadena de caracteres a número y viceversa 4. Herramientas de validación existentes 5. Elaborar herramientas de validación propias	- Exposición por parte del Docente. - Discusión Grupal. - Uso de multimedia - Ejercicios Prácticos.

UNIDAD III: ESTRUCTURA DE DATOS	**OBJETIVO TERMINAL:** Definir y manipular las estructuras de datos adecuadas a un problema dado y hacer un manejo eficiente de las mismas. Usar los conocimientos elementales referentes a la manipulación de archivos de textos y de registros.	
Duración: 28 horas	**Ponderación:** 42.42%	
OBJETIVOS ESPECÍFICOS	**CONTENIDO**	**ESTRATEGIAS DE ENSEÑANZA-APRENDIZAJE**
7. Describir el alcance y magnitud de las operaciones a efectuarse sobre los datos de tipo simples. 8. Manipular los módulos de operaciones que pueden ser aplicadas exclusivamente a las cadenas de caracteres. 9. Manipular otros tipos de datos definidos por el usuario 10. Desarrollar aplicaciones que manipulen datos de tipos homogéneos. 11. Implementar operaciones que manipulen datos de tipos heterogéneos. 12. Diseñar programas que hagan uso y/o bien generen archivos que residan en dispositivos externos.	1. Datos Simples 1.1 Char 1.2 Integer 1.3 Real 1.4 Boolean 2. Datos de tipo cadena de caracteres 2.1 Funciones 2.2 Procedimientos 3. Datos definidos por el usuario 3.1 Enumerado: Subrango 3.2 Conjunto 4. Datos estructurados homogéneos 4.1 Arreglos Unidimensionales 4.2 Arreglos Multidimensionales 5. Datos estructurados heterogéneos 5.1 Registros 5.1.1. Definición y Manipulación 5.1.2 Arreglos de Registros 7. Archivos 12.1. Texto 12.2. De Registro	- Exposición por parte del Docente. - Discusión Grupal. - Uso de multimedia - Ejercicios Prácticos.

UNIDAD IV: ALGORITMOS DE CLASIFICACION Y BÚSQUEDA	**OBJETIVO TERMINAL:** Adquirir destreza en la manipulación de los algoritmos de clasificación y búsqueda, con aplicación en arreglos.	
Duración: 5 horas	**Ponderación:** 7.58%	
OBJETIVOS ESPECÍFICOS	**CONTENIDO**	**ESTRATEGIAS DE ENSEÑANZA-APRENDIZAJE**
1. Manejar algoritmos elementales de clasificación u ordenamiento. 2. Manejar algoritmos elementales de búsqueda	1. Ordenamiento 1.1 Concepto 1.2 Algoritmo de burbuja mejorado 2. Búsqueda 2.1 Concepto 2.2 Algoritmo de búsqueda lineal o secuencial 2.3 Algoritmo de búsqueda binaria 2.4 Ventajas y desventajas entre búsqueda secuencial y binaria	- Exposición por parte del Docente. - Discusión Grupal. - Uso de multimedia - Ejercicios Prácticos.

UNIDAD V: INTRODUCCIÓN A POO	**OBJETIVO TERMINAL:** Manejar los conceptos básicos de programación orientada a objetos, diseñar básicamente los objetos en aplicaciones y estar en capacidad de realizar pequeños programas con POO.	
Duración: 15 horas	**Ponderación:** 22.73%	
OBJETIVOS ESPECÍFICOS	**CONTENIDO**	**ESTRATEGIAS DE ENSEÑANZA-APRENDIZAJE**
1. Manejar los conceptos básicos relacionados con la Programación Orientada a Objetos (POO). 2. Conocer las propiedades de la POO. 3. Diseñar objetos en aplicaciones elementales. 4. Programar ejemplos simples con una aplicación que incluya POO.	1. Conceptos Básicos de POO 1.1 Programación orientada a objetos 1.2 Clases 1.3 Instancias 1.4 Objetos 1.5 Atributos 1.6 Métodos 1.7 Mensajes 1.8 Diseño orientado a objetos 2. Propiedades de POO 2.1 Herencia 2.2 Encapsulamiento de información 2.3 Polimorfismo 2.4 Ocultamiento de la información 2.5 Abstracción de datos 2.6 Reusabilidad 2.7 Ejemplos y Ejercicios 3. Diseño e implementación de los objetos que intervienen en los ejercicios planteados. 3.1 Atributos y métodos 3.2 Programación de métodos 3.3 Corrida y prueba de programas 4. Asignación práctica	- Exposición por parte del Docente - Discusión Grupal - Uso de multimedia - Ejercicios de Implementación

PLAN DE EVALUACIÓN

			ESTRATEGIAS DE EVALUACIÓN				
	UNIDAD	**OBJETIVO**	**TECNICAS**	**INSTRUMENTOS**	**ACTIVIDADES**	**TIPO DE EVALUACIÓN**	**PONDERACIÓN CORTE**
CORTE I	I, II, III	I : 1, 2, 3, 4, 5 II : 1, 2, 3 III : 1, 3	Primera Prueba	Prueba Escrita	Aplicación de Prueba	Sumativa	15 a 25%
	I, II, III	I : 1, 2, 3, 4, 5 II : 1, 2, 3 III : 1, 3	Trabajo Práctico/ Evaluación Continua	Computador/ ejercicios	Defensa del Trabajo Ejercicios elaborados en clase	Sumativa	3 a 5%

			ESTRATEGIAS DE EVALUACIÓN				
	UNIDAD	**OBJETIVO**	**TECNICAS**	**INSTRUMENTOS**	**ACTIVIDADES**	**TIPO DE EVALUACIÓN**	**PONDERACIÓN CORTE**
CORTE II	III IV	III : 2, 4 IV : 1, 2	Segunda Prueba	Prueba Escrita	Aplicación de Prueba	Sumativa	20 a 30%
	III IV	III : 2, 4, IV : 1, 2	Trabajo Práctico/ Evaluación Continua	Computador/ ejercicios	Defensa del Trabajo Ejercicios elaborados en clase	Sumativa	3 a 5%

			ESTRATEGIAS DE EVALUACIÓN				
	UNIDAD	**OBJETIVO**	**TÉCNICAS**	**INSTRUMENTOS**	**ACTIVIDADES**	**TIPO DE EVALUACIÓN**	**PONDERACIÓN CORTE**
CORTE III	III IV	III : 5, 6 IV : 1, 2	Tercera Prueba	Prueba Escrita	Aplicación de Prueba	Sumativa	10 a 15%
	V	V: 1, 2, 3	Cuarta Prueba	Prueba Escrita	Aplicación de Prueba	Sumativa	10 a 15%
	V	V: 4	Trabajo Práctico/ Evaluación Continua	Computador/ ejercicios	Defensa del Trabajo Ejercicios elaborados en clase	Sumativa	3 a 5 %

Observaciones:

1. Los trabajos prácticos por su naturaleza pueden cubrir cualquier, varios o todos las unidades de la asignatura.
2. Por naturaleza de la asignatura Teórico-Práctico, se considera que las pruebas escritas deben cubrir todos los objetivos de la asignatura y los trabajos prácticos son para afianzar los conocimientos teóricos.
3. La prueba sustitutiva versará sobre toda la evaluación escrita, realizada en el corte correspondiente.
4. Las cuatro pruebas escritas se realizarán en las siguientes semanas: 1era. Prueba **la semana 6**, 2da. Prueba la **semana 11,** la 3era. Prueba la **semana 14** y la 4ta. Prueba en la **semana 16.**

BIBLIOGRAFIA.

1. JOYANES Luis. (1997) Programación en Turbo Pascal Versiones 5.5 6.0 y 7.0. Mc.Graw Hill: Madrid.
2. CRAWLEY & McARTHUR. Pascal Programación Estructurada. Prentice may
3. JAMSA Y NAMEROF. Turbo Pascal. Biblioteca de Programas. Mc.Graw Hill: Madrid.
4. YESTER. Using Turbo Pascal. Que Corporation.
5. BORLAND. Turbo Pascal Reference & Programmer´s Guides.
6. MITCHELL, Edward. Borland Pascal Developer´s Guides. Que Corporation.
7. KONVALINE & WILEMAN. Programming Tools in Pascal. Mc.Graw Hill: Madrid.
8. ALONSO Y MORALES. Técnicas de Programación. Ed. Paraninfo
9. ALONSO Y MORALES. Problemas de Programación. Ed. Paraninfo
10. COÑAT Y NÚÑEZ. Turbo Pascal Práctico y Programas. Ed. Paraninfo

ANEXO C

Barquisimeto, 7 de Diciembre del 2005

Prof. Edgar Rodríguez
Jefe de Registro Académico
Decanato de Ciencias y Tecnología

Ante todo un respetuoso saludo, Yo, Tulio E. León A., titular de la C.I. Nº 9.272.490. En mi carácter de participante del Programa Maestría en Ciencias de la Computación, por medio de la presente solicito a Ustedes favor suministrarme en formato digital los archivos históricos con las notas de los últimos 5 años de los alumnos de Introducción a la Computación (2154, C1) y Programación I (3154, C4) de las carreras de Ingeniería en Informática y Análisis de Sistemas. El uso que se le dará a estos datos es meramente con carácter de investigación para ser utilizados en mí Trabajo de Grado, Titulado: DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION.

En espera de una respuesta satisfactoria, quedo de Ustedes

Atentamente,

_____

Ing. Tulio E. León A.
C.I. 9.272.490

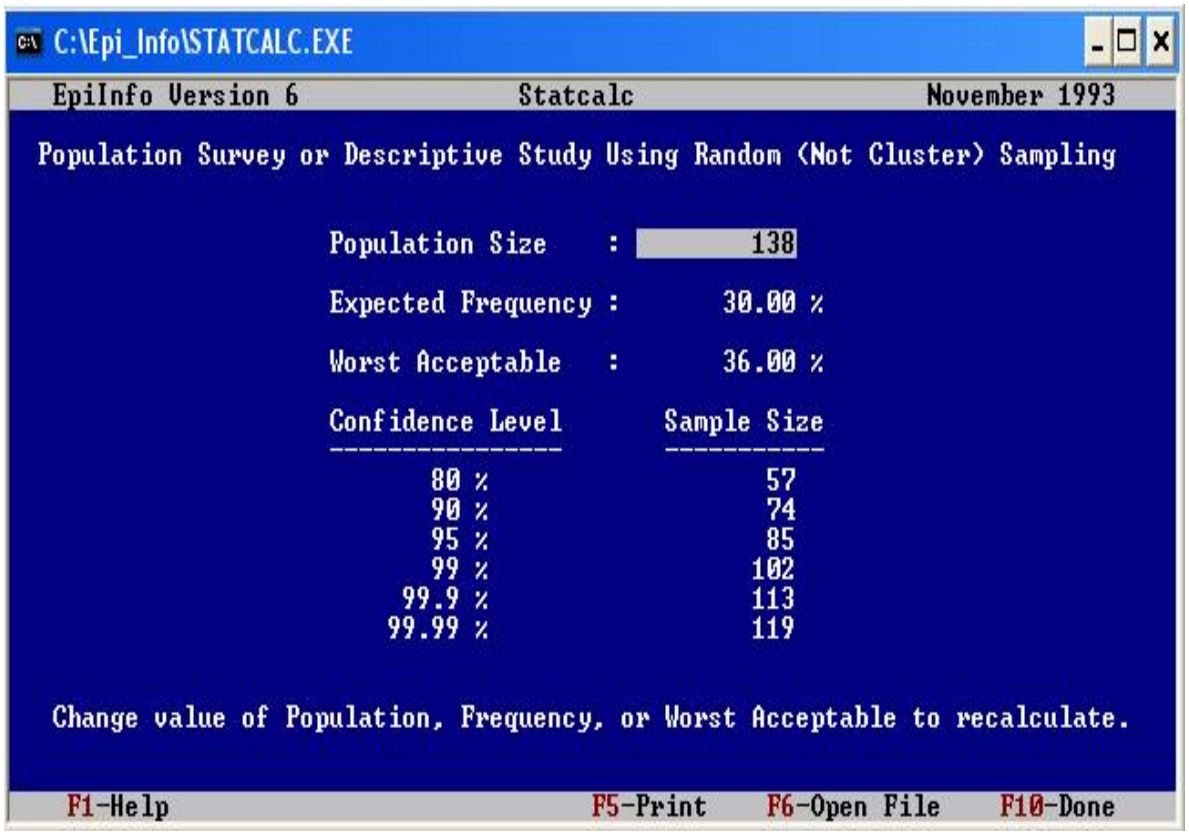
Tabla de operacionalización de las variables estudiadas

	Variable	Dimensiones	Indicadores	Items Cuestionario y Evaluación 01	Items Cuestionario 02	Items Evaluación 02
Propósito Determinar las funcionalidades asociadas con un sistema basado en tecnología Web para la enseñanza de estructuras de datos en programación **Definición** Un sistema para la enseñanza de estructuras de datos en programación es un software educativo que permite al usuario interactuar a través de la simulación para ejercitar con las Estructuras de Datos	Grado de satisfacción de enseñanzas recibidas	Estructuras de Datos	Concepto	1		
		Arreglos	Concepto	2		
			Dimensión	3		
			Declaración	4		
			Tipo de dato del Subíndice	5		
			Tipo de dato del Arreglo	6		
			Acceso a los elementos	7		
			Determinación del # de elementos	8		
	Prototipo Simulador	Apreciación	Aprendizaje		1	
			Motivación		2	
			Ayuda		3	
			Uso		4	
			Interface		5, 6, 7, 8	
		Proceso de Enseñanza – Aprendizaje en Algoritmo Burbuja	Definición del Ciclo Externo			1
			Definición del Ciclo Interno			2
			Comparación de los elementos			3
			Intercambio de los elementos			4, 5, 6
			Proceso Iterativo de ordenamiento			7, 8, 9, 10

Fuente: El autor de la investigación

ANEXO E

Tabla de muestra probabilística con intervalos de confianza



DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

CUESTIONARIO 01

Grupo: _____ **Fecha: _____**

El Instrumento que a continuación se presenta pretende recoger información sobre el grado de satisfacción acerca de los conocimientos recibidos en el área de Estructuras de Datos. Está diseñado como una escala valorativa que se limita a medir aspectos académicos sobre el logro de objetivos verificables con respecto al tema de Arreglos. En vista de que este Cuestionario es de Respuesta Subjetiva, se agradece contestarlo con la mayor sinceridad posible.

Según los siguientes criterios (1 al 5), marque con una equis para cada ítem en el cuadro según el Grado de Satisfacción de los conocimientos recibidos que más se adecue a Usted:

ITEMS	MUY BUENO 1	BUENO 2	A MEDIAS 3	UN POCO 4	NADA 5
1. Acerca del concepto de Estructuras de Datos					
2. Acerca del concepto de Arreglo					
3. Acerca de la Dimensión de un Arreglo					
4. Acerca de como se declara un tipo Arreglo					
5. Acerca de que tipo de dato debe ser el Subíndice de un Arreglo					
6. Acerca del tipo de elementos que puede contener un Arreglo					
7. Acerca de como se puede acceder a un elemento dentro de una variable de tipo Arreglo					
8. Acerca de como determinar el número máximo de elementos que puede contener una variable de tipo Arreglo					

DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

EVALUACION 01

Grupo: _____ **Fecha: _____**

El Instrumento que a continuación se presenta pretende recoger información sobre el grado de conocimientos en el área de Estructuras de Datos. Está diseñado para medir aspectos académicos sobre el logro de objetivos verificables con respecto al tema de Arreglos.

Responda las siguientes preguntas (1 al 8), según el enunciado:

1) Las Estructuras de Datos pueden contener más de un elemento

Verdadero ___ Falso ___ Explique: _____

_____

2) En un Arreglo todos los elementos deben ser del mismo tipo

Verdadero ___ Falso ___ Explique: _____

_____

3) Cual es la dimensión del tipo de Arreglo definido a continuación:

```
TYPE
   tiarreglo = ARRAY[ 1..3 , 1..5 ] OF integer;
```

Respuesta: _____

4) Dado el siguiente fragmento de programa:

```
TYPE
   tilista = ARRAY(  -10..10 ) OF integer;
VAR
   x : tilista;
   j : integer;

BEGIN
   FOR j := 1 TO 11 DO
      x[ j ] := j / 2;
END.
```

Identifique los errores, si es que existen, márquelos y descríbalos sobre el mismo código.

5) El Subíndice de un Arreglo podría ser de tipo Char

Verdadero ___ Falso ___ Explique: _____

_____

6) Un Arreglo puede contener a su vez elementos de tipo arreglo

Verdadero ___ Falso ___ Explique: _____

_____

7) De la pregunta (4), indique como puede asignar el valor 0 (cero) al primer elemento de la variable arreglo x. Respuesta: _____

8) De la pregunta (4), cuantos elementos como máximo puede contener la variable arreglo x.

Respuesta: _____

ANEXO H

DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

CUESTIONARIO 02

Grupo: ___Estudio___ **Fecha:** _____

El presente Instrumento tiene como finalidad determinar el grado de satisfacción que encuentran los estudiantes en el Prototipo del Software Simulador de Ordenamiento de Arreglos.

Según los siguientes criterios (1 al 5), marque con una equis para cada pregunta en el cuadro según el Grado de Satisfacción que Usted encontró en el Software presentado:

Criterios:

1. **TOTALMENTE DE ACUERDO**
2. **DE ACUERDO**
3. **INDECISO**
4. **EN DESACUERDO**
5. **TOTALMENTE EN DESACUERDO**

ITEMS **Respecto al Software presentado**	**TOTALMENTE DE ACUERDO** **1**	**DE ACUERDO** **2**	**INDECISO** **3**	**EN DESACUERDO** **4**	**TOTALMENTE EN DESACUERDO** **5**
1. Puedo lograr aprender de una manera más fácil, el algoritmo de ordenamiento de datos en un Arreglo					
2. Con el Software me siento motivado a aprender sobre el algoritmo de ordenamiento de Arreglos					
3. El Software brinda suficiente ayuda en el proceso de ordenamiento de un Arreglo					
4. El Software es fácil de comprender y utilizar					
5. El Software presenta una pantalla donde puedo captar la esencia del algoritmo					
6. El Software presenta suficientes elementos gráficos					
7. El Software presenta suficientes colores para distinguir los elementos					
8. El Software debería utilizar sonido y vídeo					

DISEÑO DE UN MODELO DE SOFTWARE EDUCATIVO PARA LA ENSEÑANZA DE ESTRUCTURAS DE DATOS EN PROGRAMACION

EVALUACION 02

Grupo: _____ **Fecha: _____**

1) Dado el siguiente fragmento de programa, complete el Procedimiento Ordenar y muestre como se ordenan los datos en cada iteración:

```
PROGRAM burbuja;
USES crt;
CONST
  tamano = 4;
TYPE
  tilista = ARRAY[ 1..tamano ] OF integer;
VAR
  lista : tilista;

PROCEDURE ordenar( VAR lista : tilista );
VAR
  i, j, auxiliar : integer;
BEGIN              { COMPLETE ESTE PROCEDIMIENTO }




END;

BEGIN
  lista[ 1 ] := 6;
  lista[ 2 ] := 3;
  lista[ 3 ] := 8;
  lista[ 4 ] := 1;
  ordenar( lista );
END.
```

Iteración 0 →	6	3	8	1
Iteración 1 →				
Iteración 2 →				
Iteración 3 →				
Iteración 4 →				
Iteración 5 →				

ANEXO J
Carta de solicitud de validación a los expertos

UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACION

Barquisimeto, Abril de 2006

Ciudadano (a): _____

Ante todo le saludo muy respetuosamente. Por medio de la presente solicito a Usted su colaboración en calidad de experto para determinar la validez de contenido de los Cuestionarios y Evaluaciones con los cuales se pretende recabar información necesaria para el Trabajo de Grado titulado "Diseño de un Modelo de Software Educativo para la Enseñanza de Estructuras de Datos en Programación", el cual será aplicado a los estudiantes de las Prácticas Complementarias de la asignatura Programación I de las carreras de Ingeniería en Informática y Análisis de Sistemas del Decanato de Ciencias y Tecnología de la UCLA.

Para tal propósito se anexa a la presente: 1) Matriz de Operacionalización de las Variables, 2) Formatos de Validación con sus respectivas instrucciones, 3) Cuestionarios y Evaluaciones que se aplicarán a la muestra.

Su opinión al respecto representa un gran aval para esta investigación, dada la excelente labor docente, de investigación y extensión que Usted ha realizado en pro de la formación profesional de la Universidad.

Sin otro particular al cual hacer referencia y agradeciendo de antemano su colaboración, quedo de Usted

Atentamente,

_____
Prof. Tulio E. León A.
C.I.: 9.272.490

UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
MAESTRIA EN CIENCIAS DE LA COMPUTACION

Barquisimeto, Abril de 2006

Trabajo de Grado: Diseño de un Modelo de Software Educativo para la Enseñanza de Estructuras de Datos en Programación.

Formato de Validación del instrumento: _____

Ciudadano (a): _____

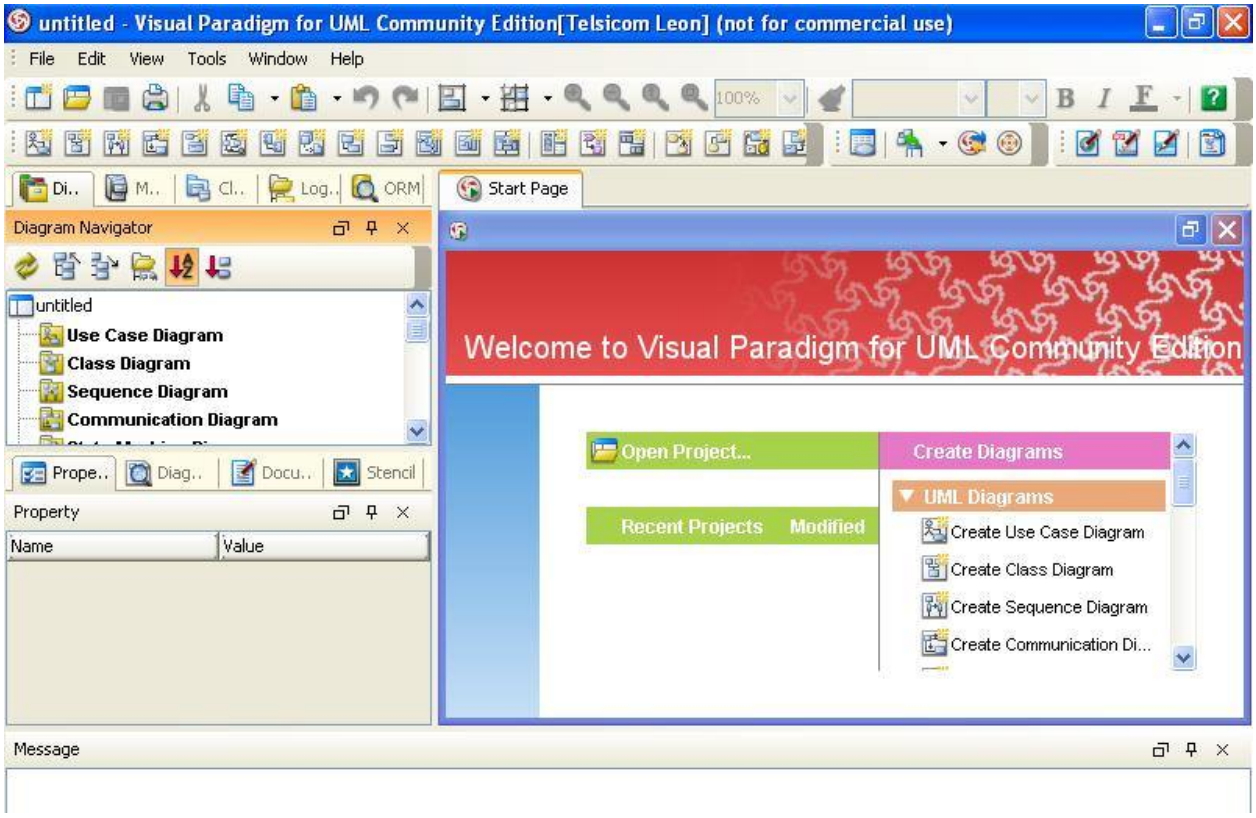
Para efectos de la evaluación correspondiente a los ítems planteados se determinará la validez de cada instrumento en los siguientes términos:

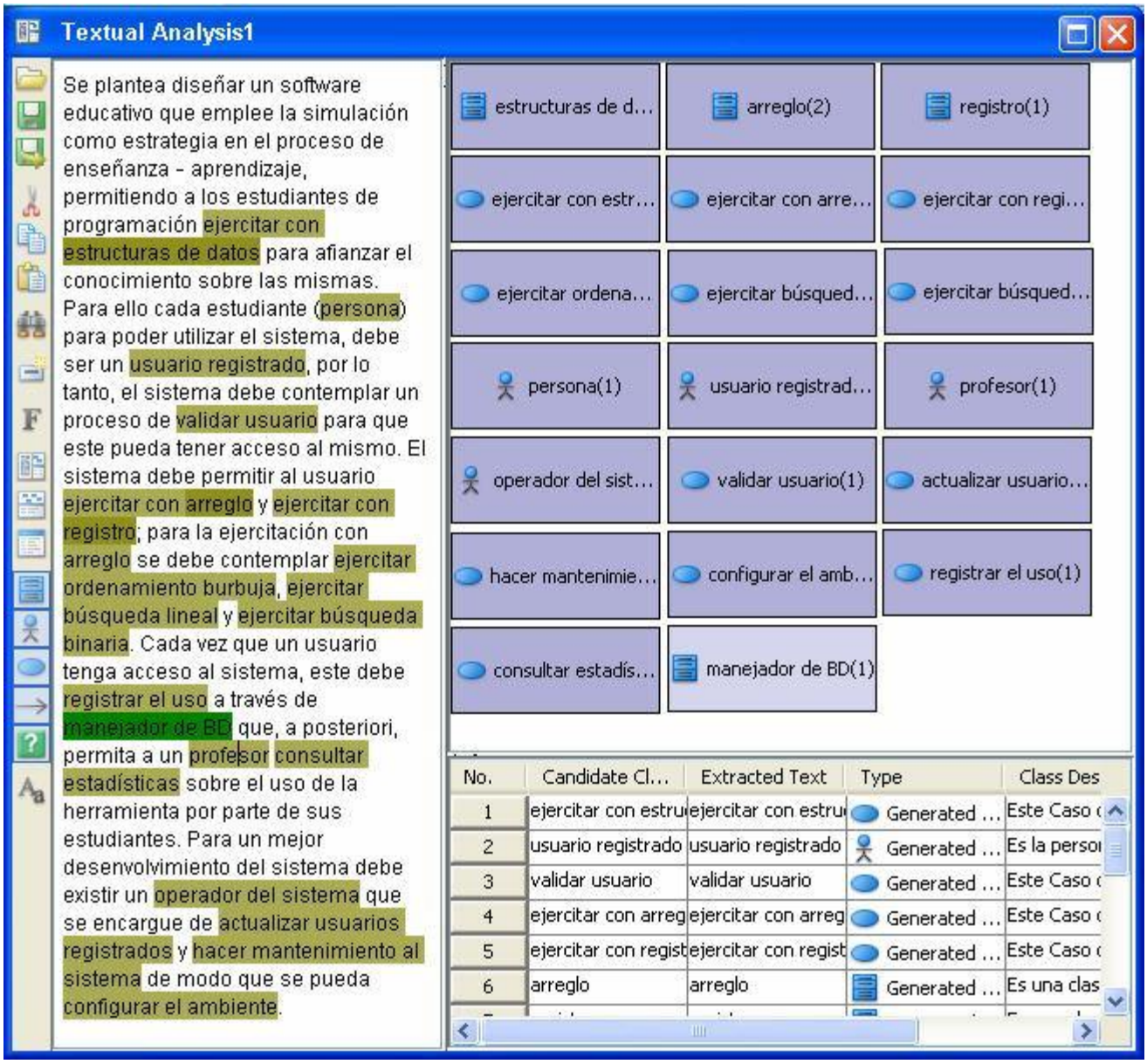
Se tomarán en cuenta los siguientes aspectos: a) **Pertinencia**: Es la correspondencia del ítem con el aspecto; b) **Claridad**: Se refiere a la redacción precisa y sencilla del ítem; y c) **Congruencia**: entendida como la lógica interna del ítem.

Se le agradece seleccionar una de las 2 posibles opciones (Si/No) para cada ítem con el objetivo de señalar el grado de pertinencia, claridad y congruencia de los ítems.

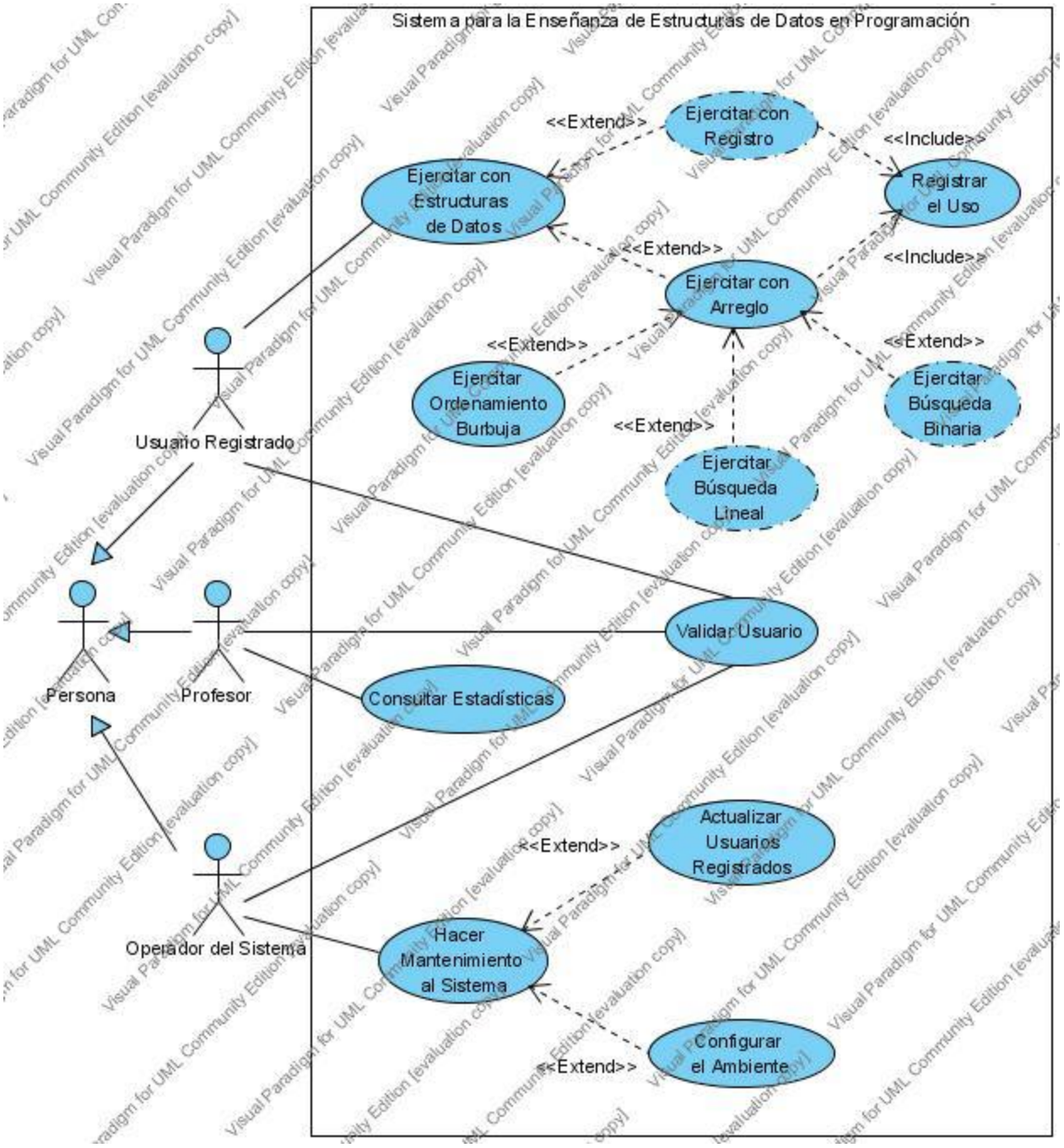
	Pertinencia		**Claridad**		**Congruencia**		
Item	SI	NO	SI	NO	SI	NO	Observaciones
1							
2							
3							
4							
5							
6							
7							
8							

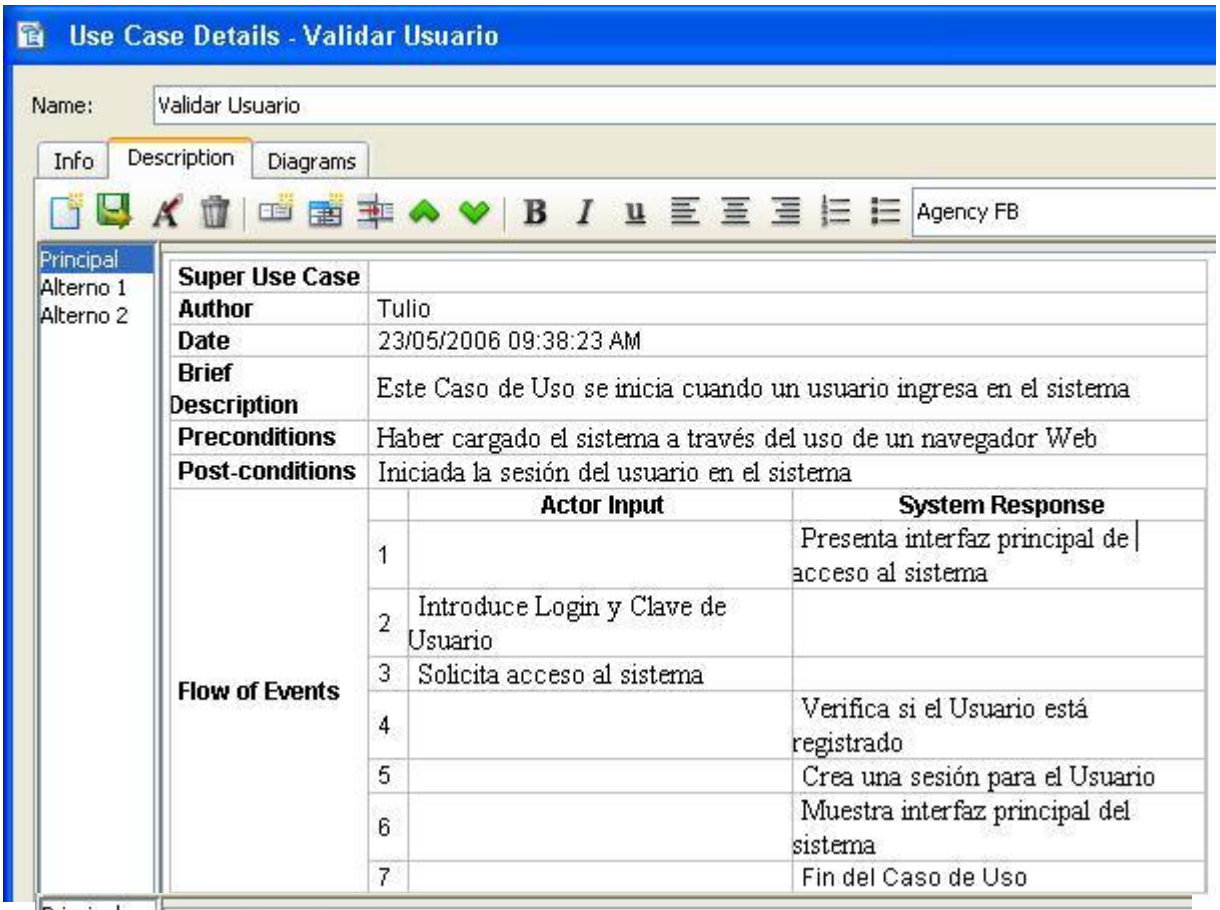
ANEXO L
Entorno de desarrollo de Visual Paradigm 5.2




Textual Analysis1
Se plantea diseñar un software educativo que emplee la simulación como estrategia en el proceso de enseñanza - aprendizaje, permitiendo a los estudiantes de programación ejercitar con estructuras de datos para afianzar el conocimiento sobre las mismas. Para ello cada estudiante (persona) para poder utilizar el sistema, debe ser un usuario registrado, por lo tanto, el sistema debe contemplar un proceso de validar usuario para que este pueda tener acceso al mismo. El sistema debe permitir al usuario ejercitar con arreglo y ejercitar con registro; para la ejercitación con arreglo se debe contemplar ejercitar ordenamiento burbuja, ejercitar búsqueda lineal y ejercitar búsqueda binaria. Cada vez que un usuario tenga acceso al sistema, este debe registrar el uso a través de manejador de BD que, a posteriori, permita a un profesor consultar estadísticas sobre el uso de la herramienta por parte de sus estudiantes. Para un mejor desenvolvimiento del sistema debe existir un operador del sistema que se encargue de actualizar usuarios registrados y hacer mantenimiento al sistema de modo que se pueda configurar el ambiente.
estructuras de d...
arreglo(2)
registro(1)
ejercitar con estr...
ejercitar con arre...
ejercitar con regi...
ejercitar ordena...
ejercitar búsqued...
ejercitar búsqued...
persona(1)
usuario registrad...
profesor(1)
operador del sist...
validar usuario(1)
actualizar usuario...
hacer mantenimie...
configurar el amb...
registrar el uso(1)
consultar estadís...
manejador de BD(1)
No.
Candidate Cl...
Extracted Text
Type
Class Des
1 ejercitar con estru ejercitar con estru Generated ... Este Caso
2 usuario registrado usuario registrado Generated ... Es la perso
3 validar usuario validar usuario Generated ... Este Caso
4 ejercitar con arreg ejercitar con arreg Generated ... Este Caso
5 ejercitar con regist ejercitar con regist Generated ... Este Caso
6 arreglo arreglo Generated ... Es una clas

ANEXO N
Diagrama de Casos de Uso del sistema SEDPRO





Super Use Case	
Author	Tulio
Date	23/05/2006 09:38:23 AM
Brief Description	Este Caso de Uso se inicia cuando un usuario ingresa en el sistema
Preconditions	Haber cargado el sistema a través del uso de un navegador Web
Post-conditions	Iniciada la sesión del usuario en el sistema

Flow of Events

	Actor Input	System Response
1		Presenta interfaz principal de acceso al sistema
2	Introduce Login y Clave de Usuario	
3	Solicita acceso al sistema	
4		Verifica si el Usuario está registrado
5		Crea una sesión para el Usuario
6		Muestra interfaz principal del sistema
7		Fin del Caso de Uso

Principal
Alterno 1
Alterno 2

Super Use Case	
Author	Tulio
Date	23/05/2006 09:45:39 AM
Brief Description	
Preconditions	
Post-conditions	

Flow of Events

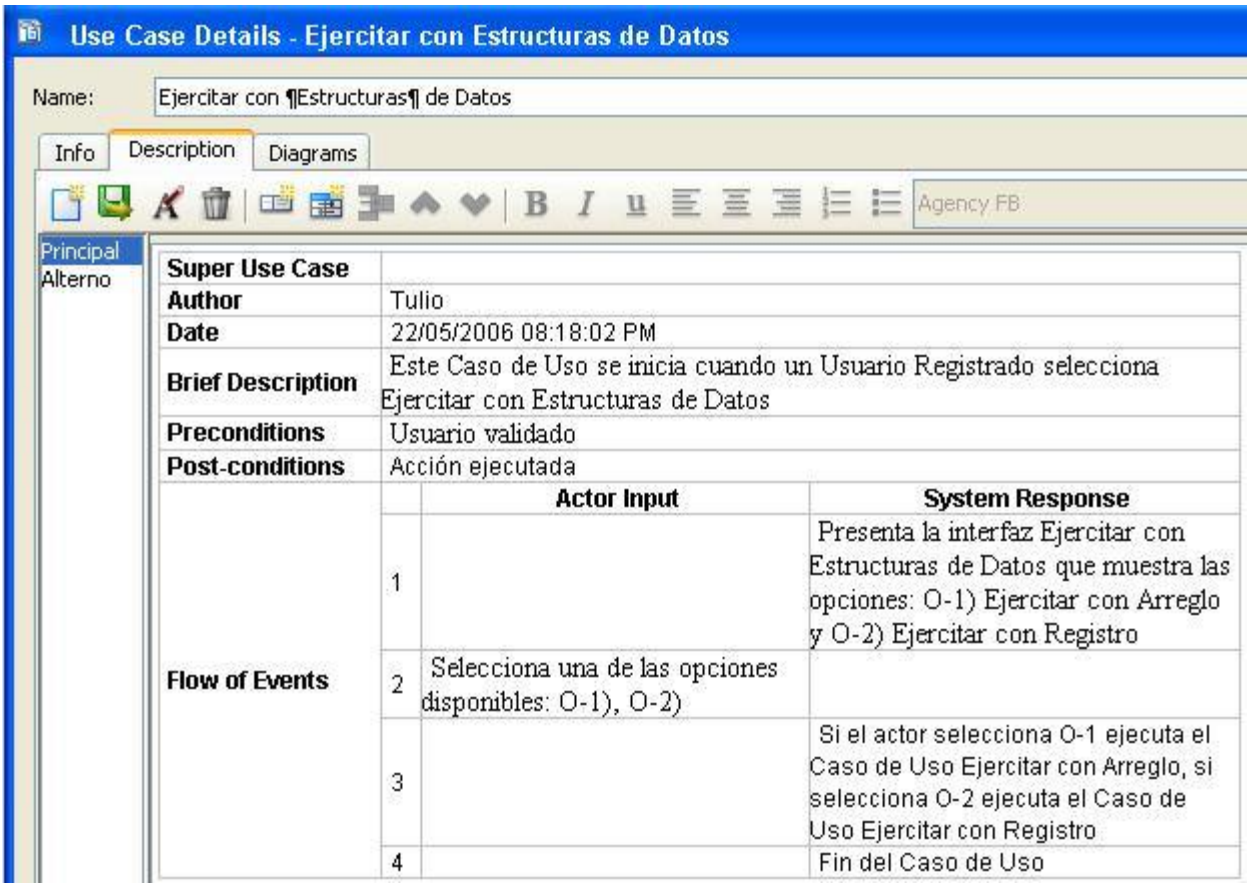
	Actor Input	System Response
1	El Usuario desea salir del Caso de Uso	
2		Ir al paso 7 del escenario Principal

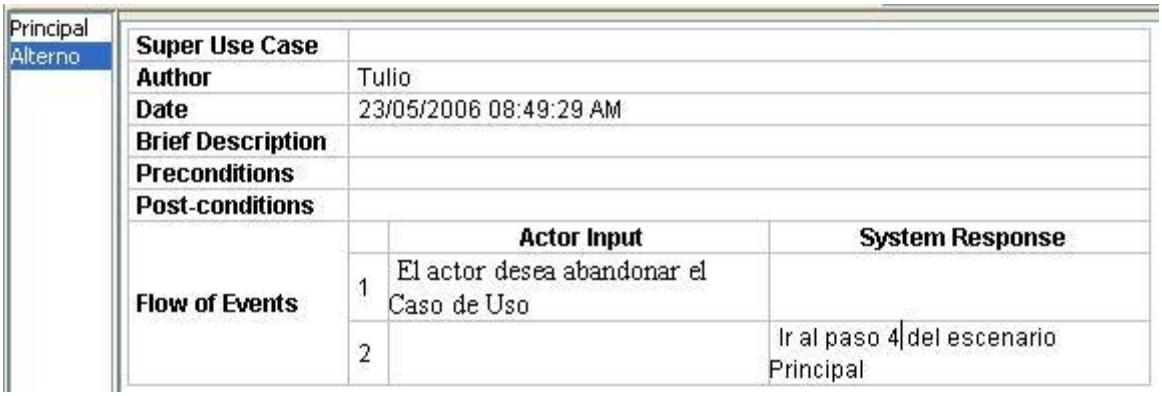
Principal
Alterno 1
Alterno 2

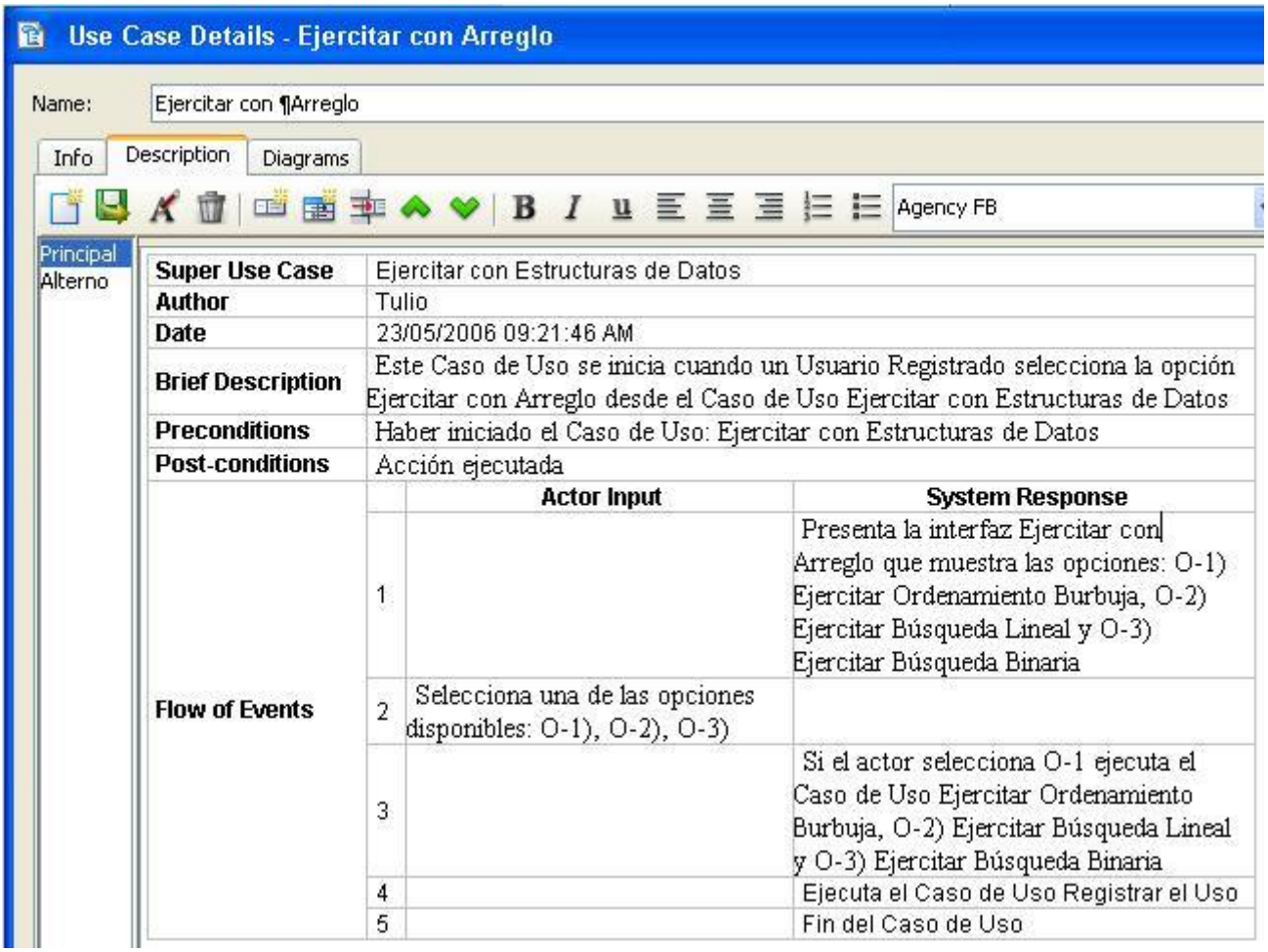
Super Use Case	
Author	Tulio
Date	23/05/2006 09:46:48 AM
Brief Description	
Preconditions	
Post-conditions	

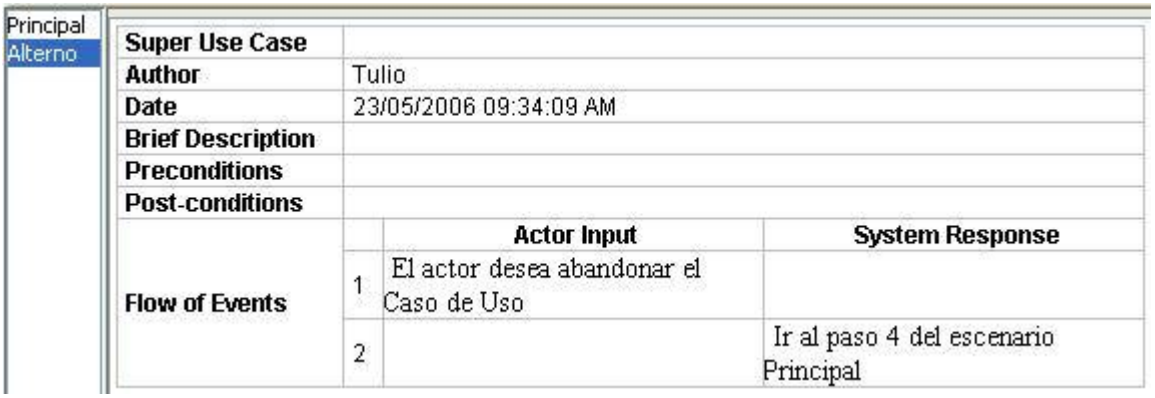
Flow of Events

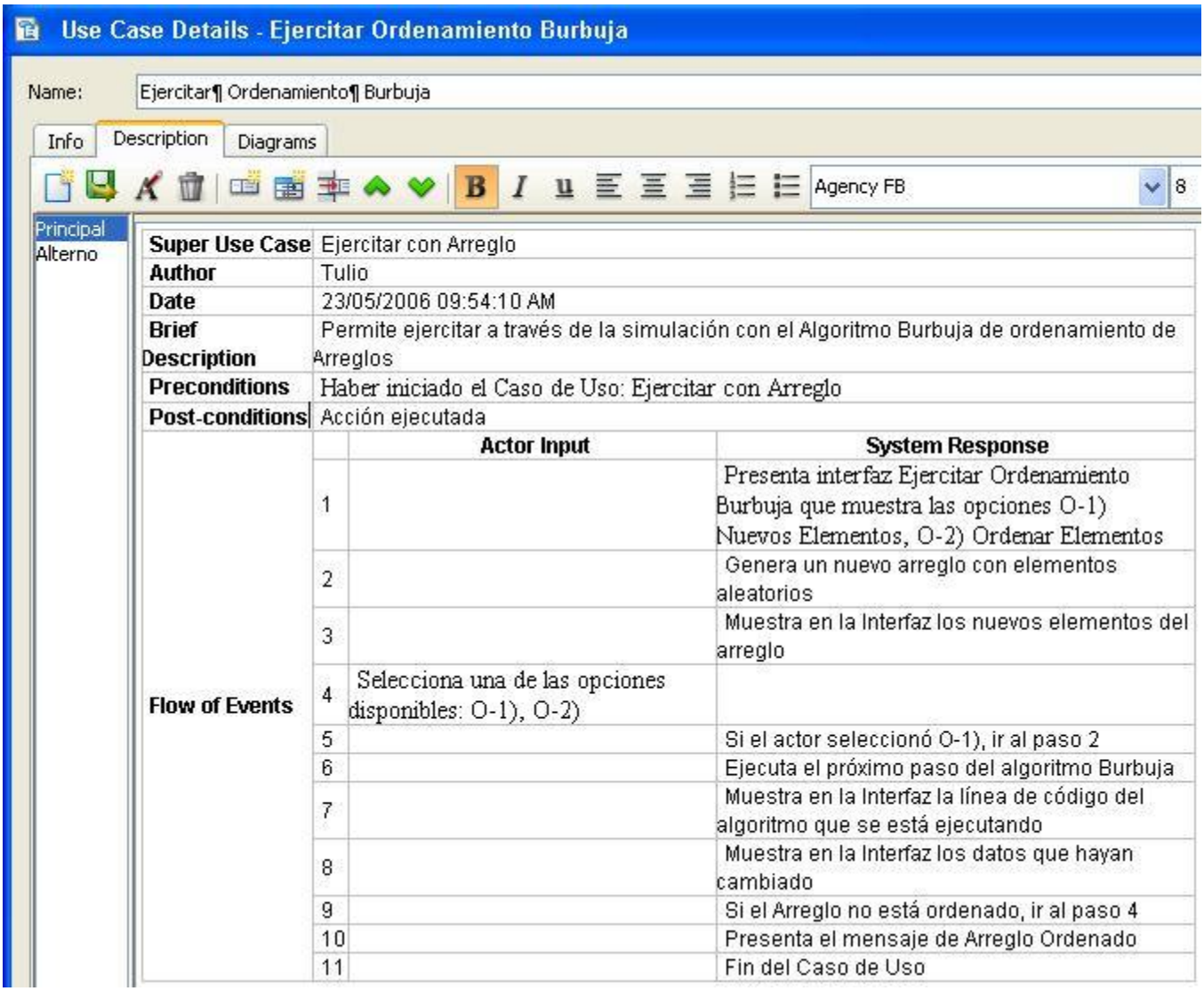
	Actor Input	System Response
1	El Login y/o Clave introducido por el Usuario no están registrados	
2		Notificarle al Usuario
3		Ir al paso 2 del escenario Principal


Use Case Details - Ejercitar con Estructuras de Datos
Name:
Ejercitar con ¶Estructuras¶ de Datos
Info
Description
Diagrams
Agency FB
Principal
Alterno
Super Use Case
Author
Tulio
Date
22/05/2006 08:18:02 PM
Brief Description
Este Caso de Uso se inicia cuando un Usuario Registrado selecciona Ejercitar con Estructuras de Datos
Preconditions
Usuario validado
Post-conditions
Acción ejecutada
Flow of Events
Actor Input
System Response
1
Presenta la interfaz Ejercitar con Estructuras de Datos que muestra las opciones: O-1) Ejercitar con Arreglo y O-2) Ejercitar con Registro
2
Selecciona una de las opciones disponibles: O-1), O-2)
3
Si el actor selecciona O-1 ejecuta el Caso de Uso Ejercitar con Arreglo, si selecciona O-2 ejecuta el Caso de Uso Ejercitar con Registro
4
Fin del Caso de Uso

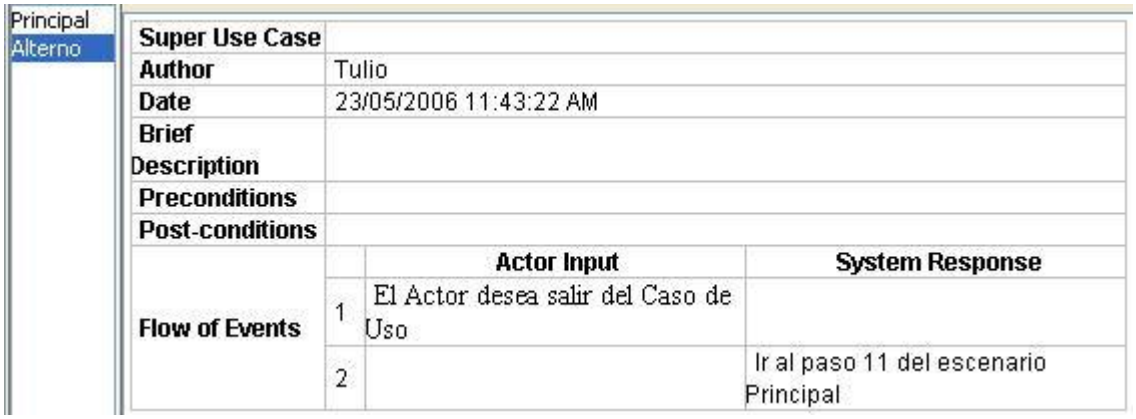

Principal
Alterno
Super Use Case
Author
Tulio
Date
23/05/2006 08:49:29 AM
Brief Description
Preconditions
Post-conditions
Flow of Events
Actor Input
System Response
1
El actor desea abandonar el Caso de Uso
2
Ir al paso 4 del escenario Principal


Use Case Details - Ejercitar con Arreglo
Name:
Ejercitar con ¶Arreglo
Info
Description
Diagrams
Agency FB
Principal
Alterno
Super Use Case
Ejercitar con Estructuras de Datos
Author
Tulio
Date
23/05/2006 09:21:46 AM
Brief Description
Este Caso de Uso se inicia cuando un Usuario Registrado selecciona la opción Ejercitar con Arreglo desde el Caso de Uso Ejercitar con Estructuras de Datos
Preconditions
Haber iniciado el Caso de Uso: Ejercitar con Estructuras de Datos
Post-conditions
Acción ejecutada
Flow of Events
Actor Input
System Response
1
Presenta la interfaz Ejercitar con Arreglo que muestra las opciones: O-1) Ejercitar Ordenamiento Burbuja, O-2) Ejercitar Búsqueda Lineal y O-3) Ejercitar Búsqueda Binaria
2
Selecciona una de las opciones disponibles: O-1), O-2), O-3)
3
Si el actor selecciona O-1 ejecuta el Caso de Uso Ejercitar Ordenamiento Burbuja, O-2) Ejercitar Búsqueda Lineal y O-3) Ejercitar Búsqueda Binaria
4
Ejecuta el Caso de Uso Registrar el Uso
5
Fin del Caso de Uso

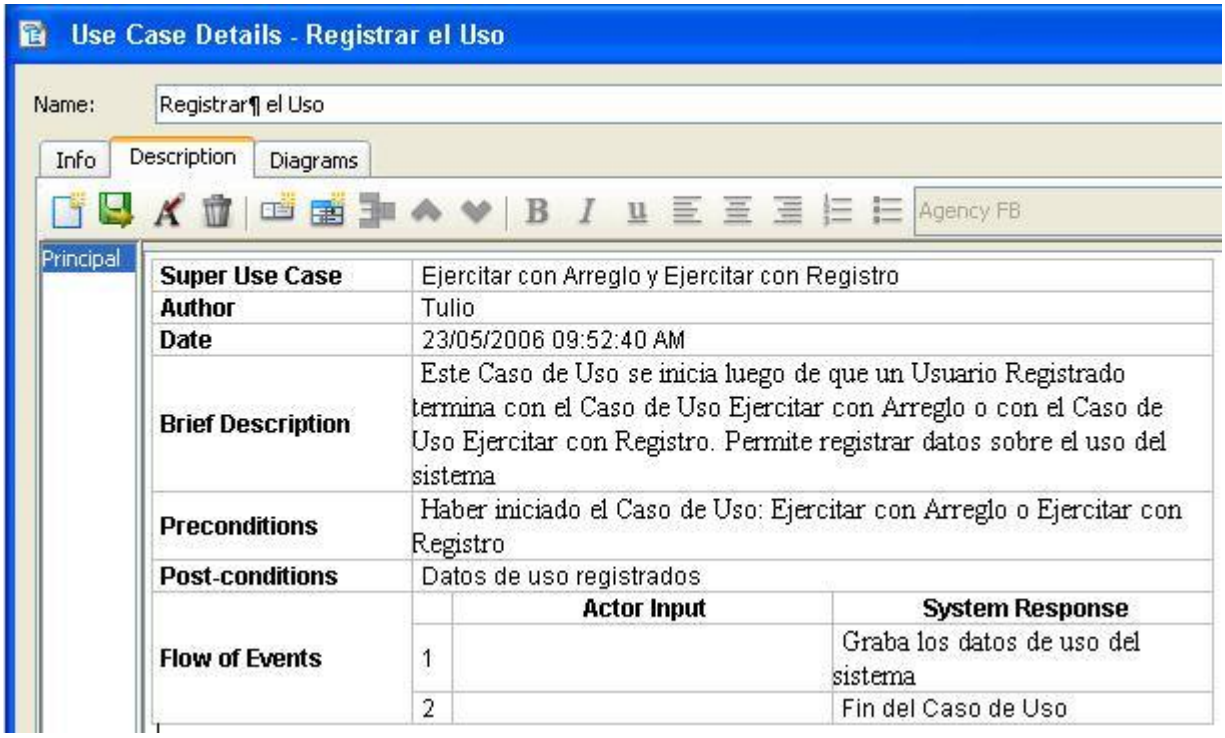

Principal
Alterno
Super Use Case
Author
Tulio
Date
23/05/2006 09:34:09 AM
Brief Description
Preconditions
Post-conditions
Flow of Events
Actor Input
System Response
1
El actor desea abandonar el Caso de Uso
2
Ir al paso 4 del escenario Principal



Super Use Case	Ejercitar con Arreglo		
Author	Tulio		
Date	23/05/2006 09:54:10 AM		
Brief Description	Permite ejercitar a través de la simulación con el Algoritmo Burbuja de ordenamiento de Arreglos		
Preconditions	Haber iniciado el Caso de Uso: Ejercitar con Arreglo		
Post-conditions	Acción ejecutada		
Flow of Events		**Actor Input**	**System Response**
	1		Presenta interfaz Ejercitar Ordenamiento Burbuja que muestra las opciones O-1) Nuevos Elementos, O-2) Ordenar Elementos
	2		Genera un nuevo arreglo con elementos aleatorios
	3		Muestra en la Interfaz los nuevos elementos del arreglo
	4	Selecciona una de las opciones disponibles: O-1), O-2)	
	5		Si el actor seleccionó O-1), ir al paso 2
	6		Ejecuta el próximo paso del algoritmo Burbuja
	7		Muestra en la Interfaz la línea de código del algoritmo que se está ejecutando
	8		Muestra en la Interfaz los datos que hayan cambiado
	9		Si el Arreglo no está ordenado, ir al paso 4
	10		Presenta el mensaje de Arreglo Ordenado
	11		Fin del Caso de Uso



Super Use Case			
Author	Tulio		
Date	23/05/2006 11:43:22 AM		
Brief Description			
Preconditions			
Post-conditions			
Flow of Events		**Actor Input**	**System Response**
	1	El Actor desea salir del Caso de Uso	
	2		Ir al paso 11 del escenario Principal


Use Case Details - Registrar el Uso
Name:
Registrar¶ el Uso
Info
Description
Diagrams
Agency FB
Principal
Super Use Case
Ejercitar con Arreglo y Ejercitar con Registro
Author
Tulio
Date
23/05/2006 09:52:40 AM
Brief Description
Este Caso de Uso se inicia luego de que un Usuario Registrado termina con el Caso de Uso Ejercitar con Arreglo o con el Caso de Uso Ejercitar con Registro. Permite registrar datos sobre el uso del sistema
Preconditions
Haber iniciado el Caso de Uso: Ejercitar con Arreglo o Ejercitar con Registro
Post-conditions
Datos de uso registrados
Flow of Events
Actor Input
System Response
1
Graba los datos de uso del sistema
2
Fin del Caso de Uso

Use Case Details - Consultar Estadísticas

Name: Consultar Estadísticas

Info | Description | Diagrams

Agency FB 8

Principal
Alterno 1
Alterno 2

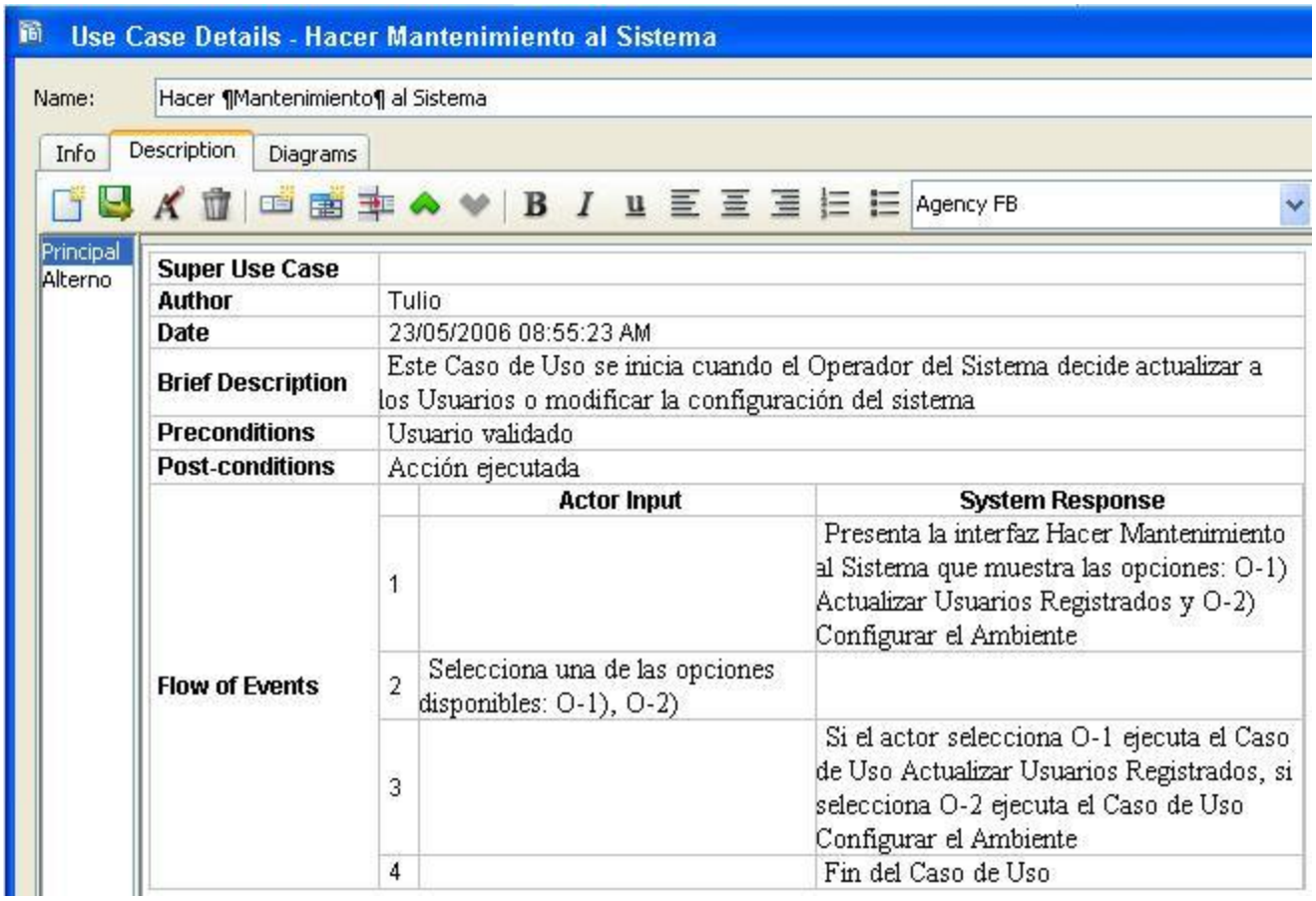
Super Use Case			
Author	Tulio		
Date	23/05/2006 10:37:04 AM		
Brief Description	Este Caso de Uso se inicia cuando un Profesor selecciona Consultar Estadísticas		
Preconditions	Usuario validado		
Post-conditions	Acción ejecutada		
Flow of Events		**Actor Input**	**System Response**
	1		Presenta interfaz Consultar Estadísticas que muestra las opciones O-1) Por Estudiante, O-2) General
	2	Selecciona una de las opciones disponibles: O-1), O-2)	
	3		Si el Profesor seleccionó O-2), ir al paso 8
	4		Presenta la Interfaz Consulta Estadística por Estudiante
	5	Introduce cédula del estudiante	
	6		Busca las estadísticas del Estudiante
	7		Muestra las estadísticas del estudiante, ir al paso 11
	8		Presenta la Interfaz Consulta Estadística General
	9		Busca las estdísticas Generales
	10		Presenta las estadisticas generales
	11		Fin del Caso de Uso

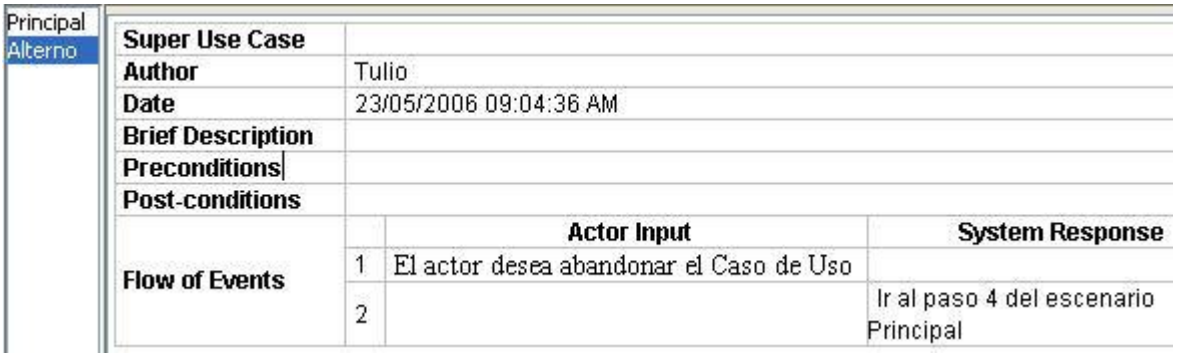
Principal
Alterno 1
Alterno 2

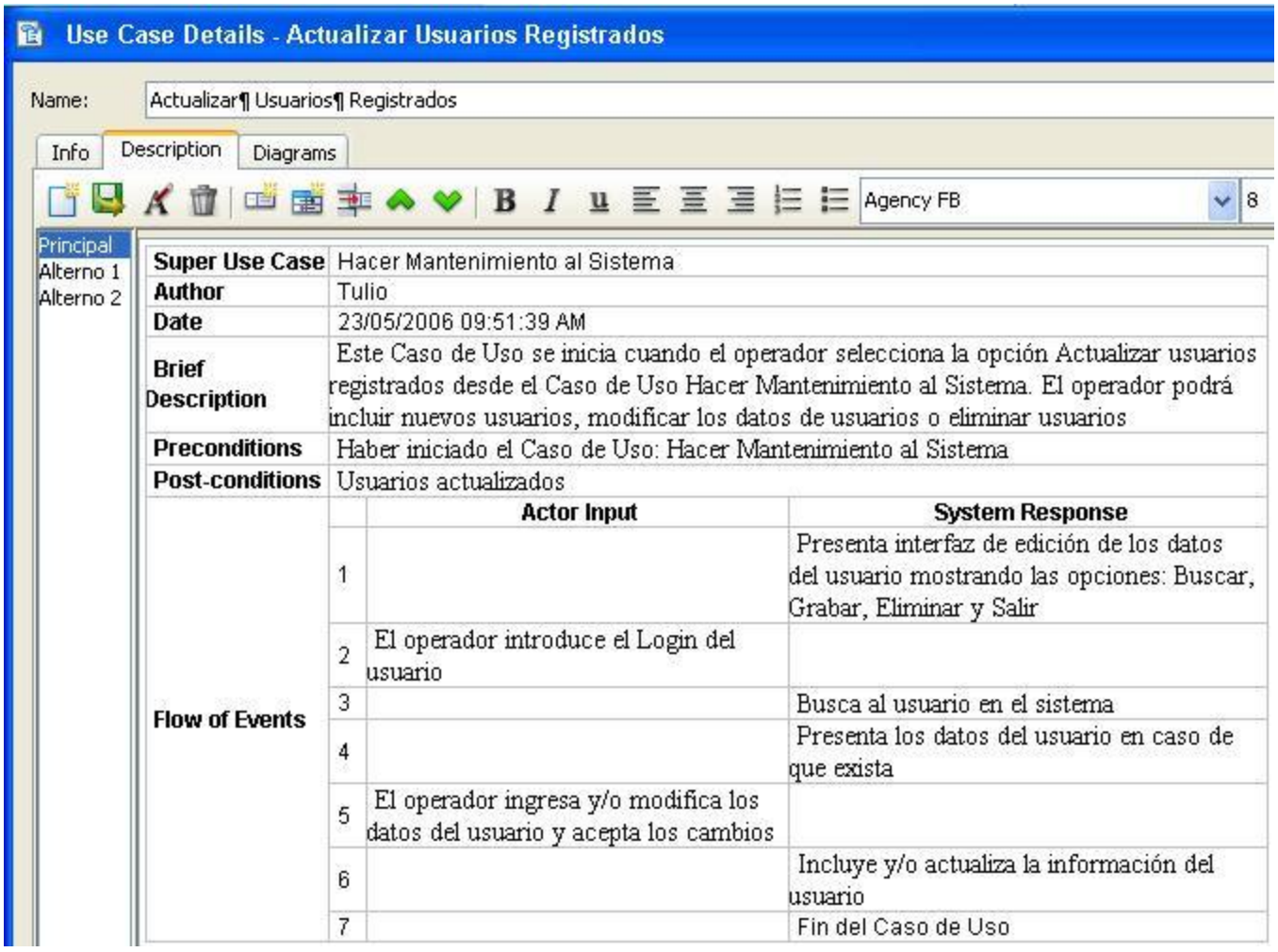
Super Use Case			
Author	Tulio		
Date	23/05/2006 10:56:33 AM		
Brief Description			
Preconditions			
Post-conditions			
Flow of Events		**Actor Input**	**System Response**
	1	El Profesor desea salir del Caso de Uso	
	2		Ir al paso 11 del escenario Principal

Principal
Alterno 1
Alterno 2

Super Use Case			
Author	Tulio		
Date	23/05/2006 10:58:15 AM		
Brief Description			
Preconditions			
Post-conditions			
Flow of Events		**Actor Input**	**System Response**
	1	La cédula del estudiante no está registrada	
	2		Notificarle al Profesor
	3		Ir al paso 5 del escenario Principal


Use Case Details - Hacer Mantenimiento al Sistema
Name:
Hacer ¶Mantenimiento¶ al Sistema
Info
Description
Diagrams
Agency FB
Principal
Alterno
Super Use Case
Author
Tulio
Date
23/05/2006 08:55:23 AM
Brief Description
Este Caso de Uso se inicia cuando el Operador del Sistema decide actualizar a los Usuarios o modificar la configuración del sistema
Preconditions
Usuario validado
Post-conditions
Acción ejecutada
Flow of Events
Actor Input
System Response
1
Presenta la interfaz Hacer Mantenimiento al Sistema que muestra las opciones: O-1) Actualizar Usuarios Registrados y O-2) Configurar el Ambiente
2
Selecciona una de las opciones disponibles: O-1), O-2)
3
Si el actor selecciona O-1 ejecuta el Caso de Uso Actualizar Usuarios Registrados, si selecciona O-2 ejecuta el Caso de Uso Configurar el Ambiente
4
Fin del Caso de Uso


Principal
Alterno
Super Use Case
Author
Tulio
Date
23/05/2006 09:04:36 AM
Brief Description
Preconditions
Post-conditions
Flow of Events
Actor Input
System Response
1
El actor desea abandonar el Caso de Uso
2
Ir al paso 4 del escenario Principal



Use Case Details - Actualizar Usuarios Registrados

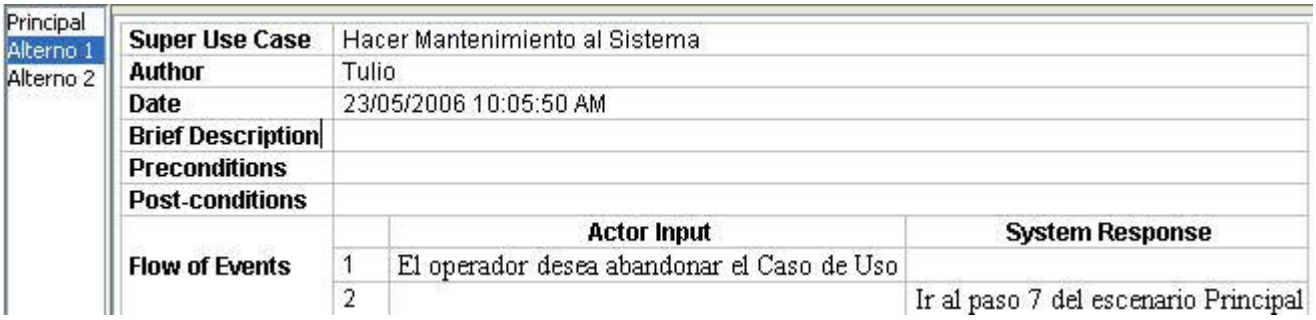
Name: Actualizar¶ Usuarios¶ Registrados

Info | Description | Diagrams

Agency FB 8

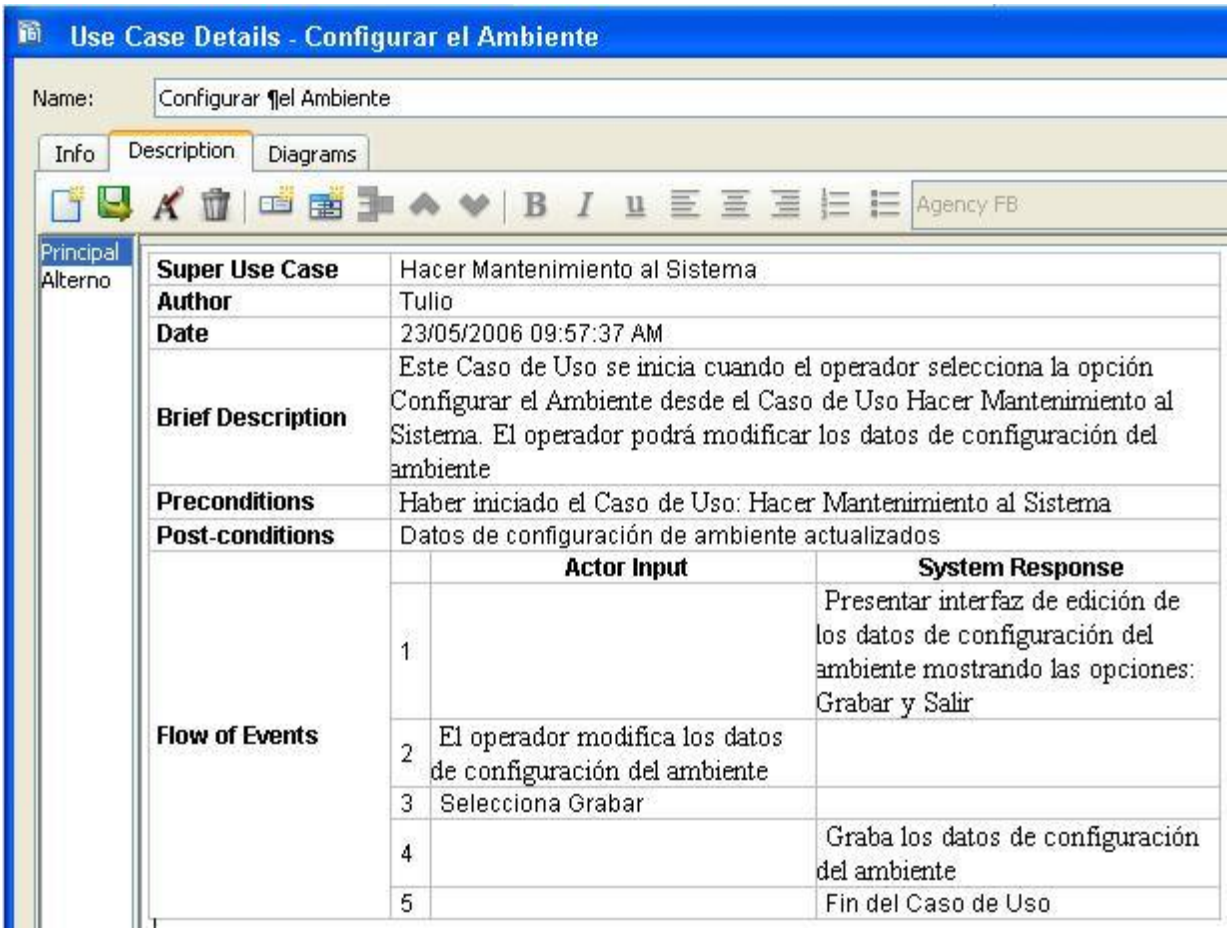
Principal
Alterno 1
Alterno 2

Super Use Case	Hacer Mantenimiento al Sistema		
Author	Tulio		
Date	23/05/2006 09:51:39 AM		
Brief Description	Este Caso de Uso se inicia cuando el operador selecciona la opción Actualizar usuarios registrados desde el Caso de Uso Hacer Mantenimiento al Sistema. El operador podrá incluir nuevos usuarios, modificar los datos de usuarios o eliminar usuarios		
Preconditions	Haber iniciado el Caso de Uso: Hacer Mantenimiento al Sistema		
Post-conditions	Usuarios actualizados		
Flow of Events		**Actor Input**	**System Response**
	1		Presenta interfaz de edición de los datos del usuario mostrando las opciones: Buscar, Grabar, Eliminar y Salir
	2	El operador introduce el Login del usuario	
	3		Busca al usuario en el sistema
	4		Presenta los datos del usuario en caso de que exista
	5	El operador ingresa y/o modifica los datos del usuario y acepta los cambios	
	6		Incluye y/o actualiza la información del usuario
	7		Fin del Caso de Uso



Principal
Alterno 1
Alterno 2

Super Use Case	Hacer Mantenimiento al Sistema		
Author	Tulio		
Date	23/05/2006 10:05:50 AM		
Brief Description			
Preconditions			
Post-conditions			
Flow of Events		**Actor Input**	**System Response**
	1	El operador desea abandonar el Caso de Uso	
	2		Ir al paso 7 del escenario Principal



Use Case Details - Configurar el Ambiente

Name: Configurar ¶el Ambiente

Info | Description | Diagrams

Agency FB

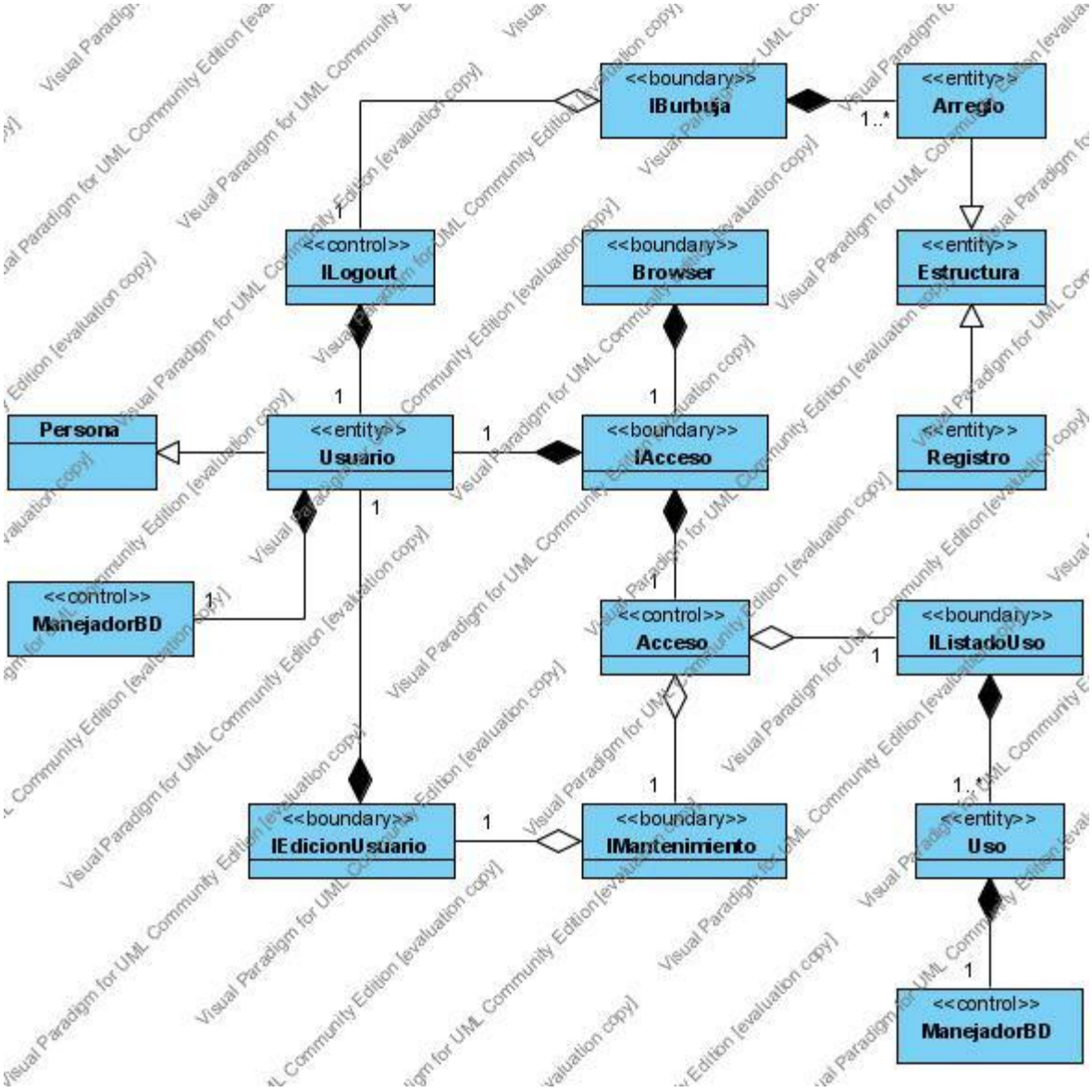
Principal
Alterno

Super Use Case	Hacer Mantenimiento al Sistema		
Author	Tulio		
Date	23/05/2006 09:57:37 AM		
Brief Description	Este Caso de Uso se inicia cuando el operador selecciona la opción Configurar el Ambiente desde el Caso de Uso Hacer Mantenimiento al Sistema. El operador podrá modificar los datos de configuración del ambiente		
Preconditions	Haber iniciado el Caso de Uso: Hacer Mantenimiento al Sistema		
Post-conditions	Datos de configuración de ambiente actualizados		
Flow of Events		**Actor Input**	**System Response**
	1		Presentar interfaz de edición de los datos de configuración del ambiente mostrando las opciones: Grabar y Salir
	2	El operador modifica los datos de configuración del ambiente	
	3	Selecciona Grabar	
	4		Graba los datos de configuración del ambiente
	5		Fin del Caso de Uso

Principal
Alterno

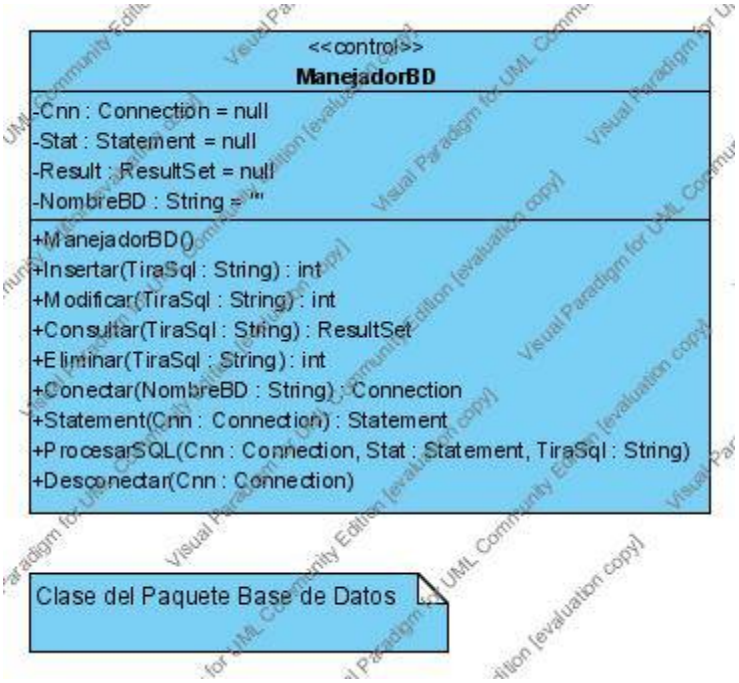
Super Use Case			
Author	Tulio		
Date	23/05/2006 10:18:41 AM		
Brief Description			
Preconditions			
Post-conditions			
Flow of Events		**Actor Input**	**System Response**
	1	El operador desea abandonar el Caso de Uso	
	2		Ir al paso 5 del escenario Principal

ANEXO P
Diagrama de Clases del sistema SEDPRO

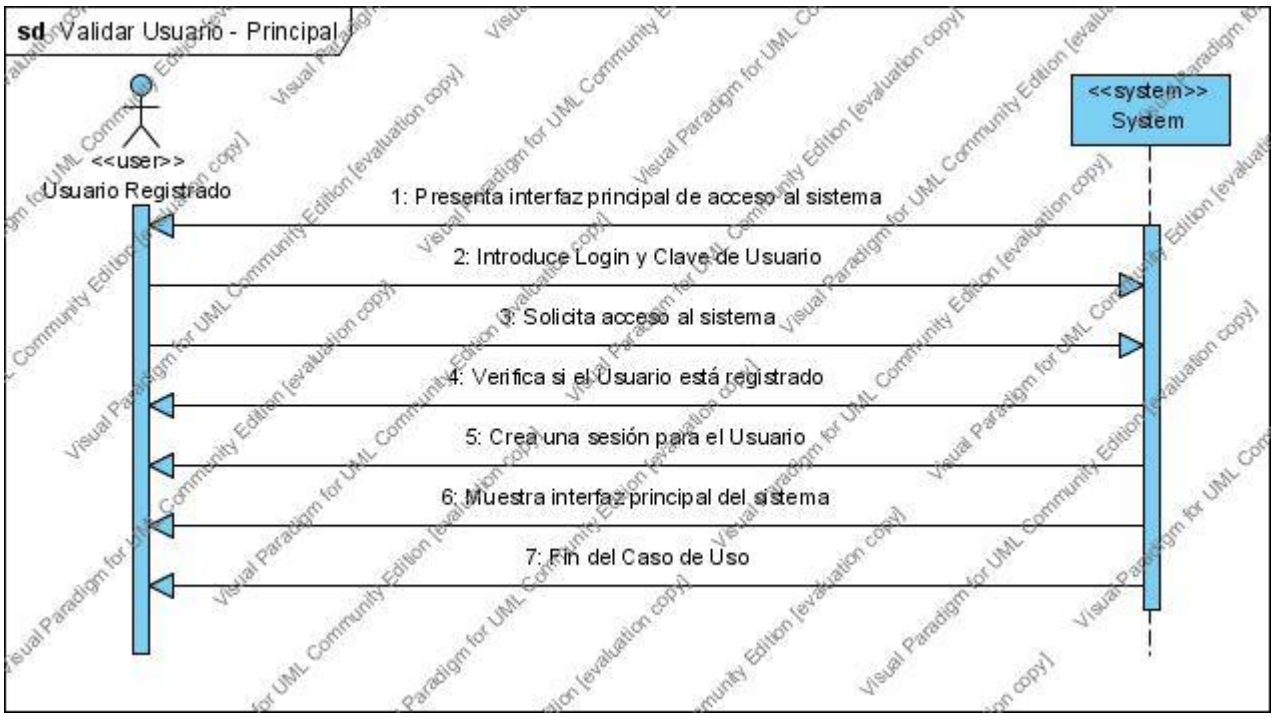


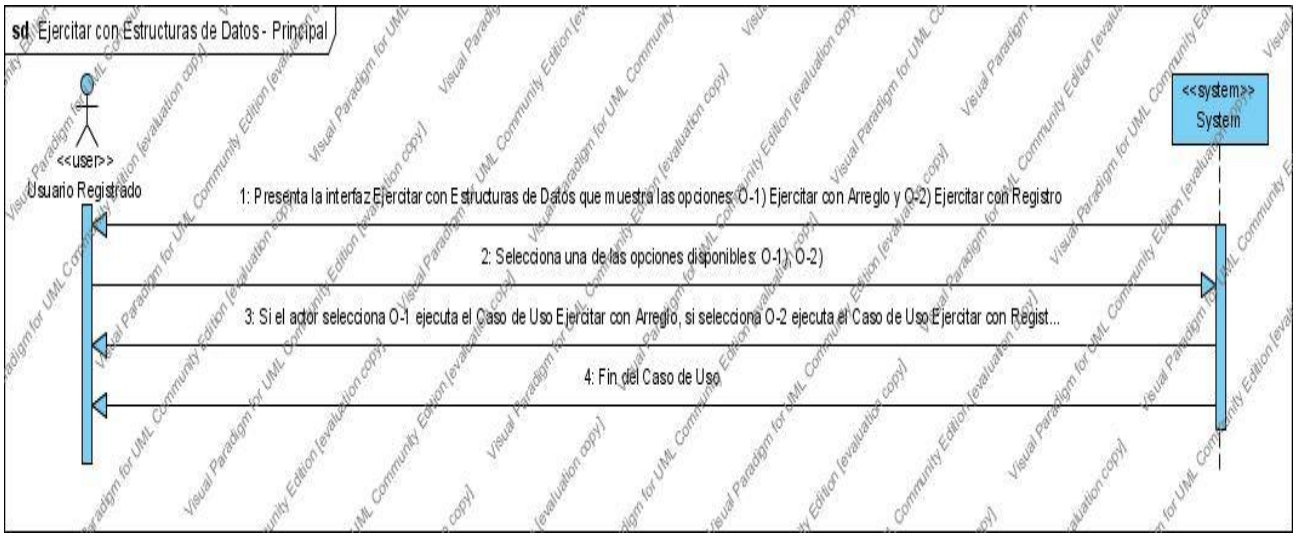
ANEXO Q

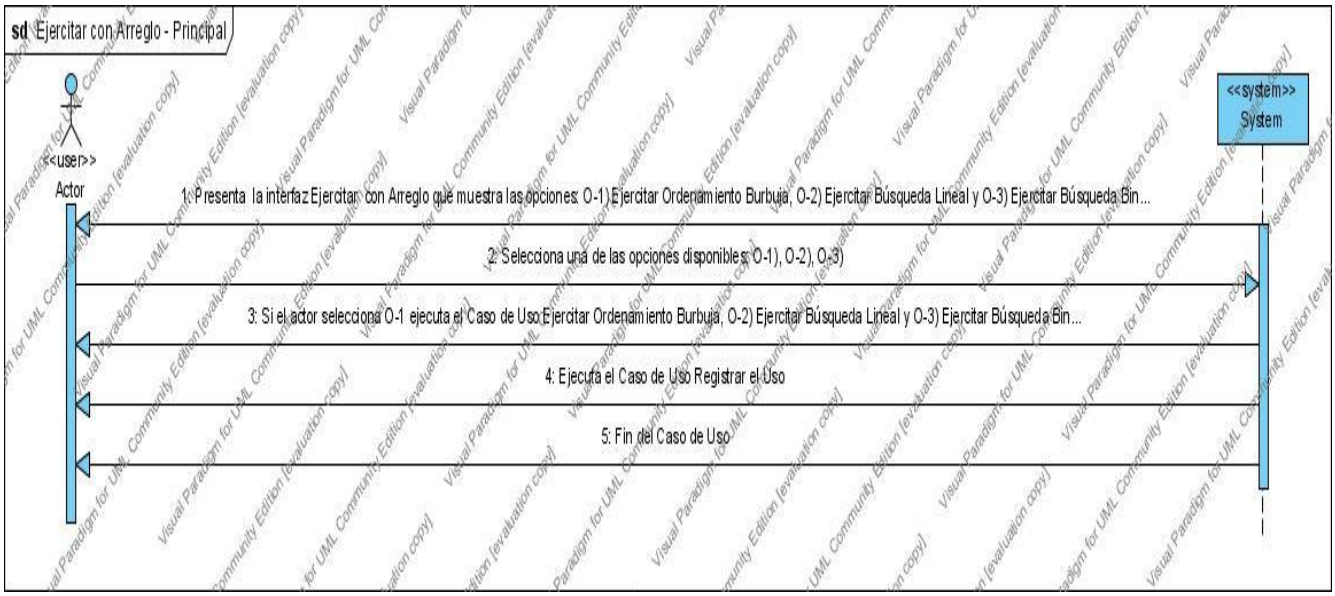
Especificación del paquete Base de Datos del sistema SEDPRO.

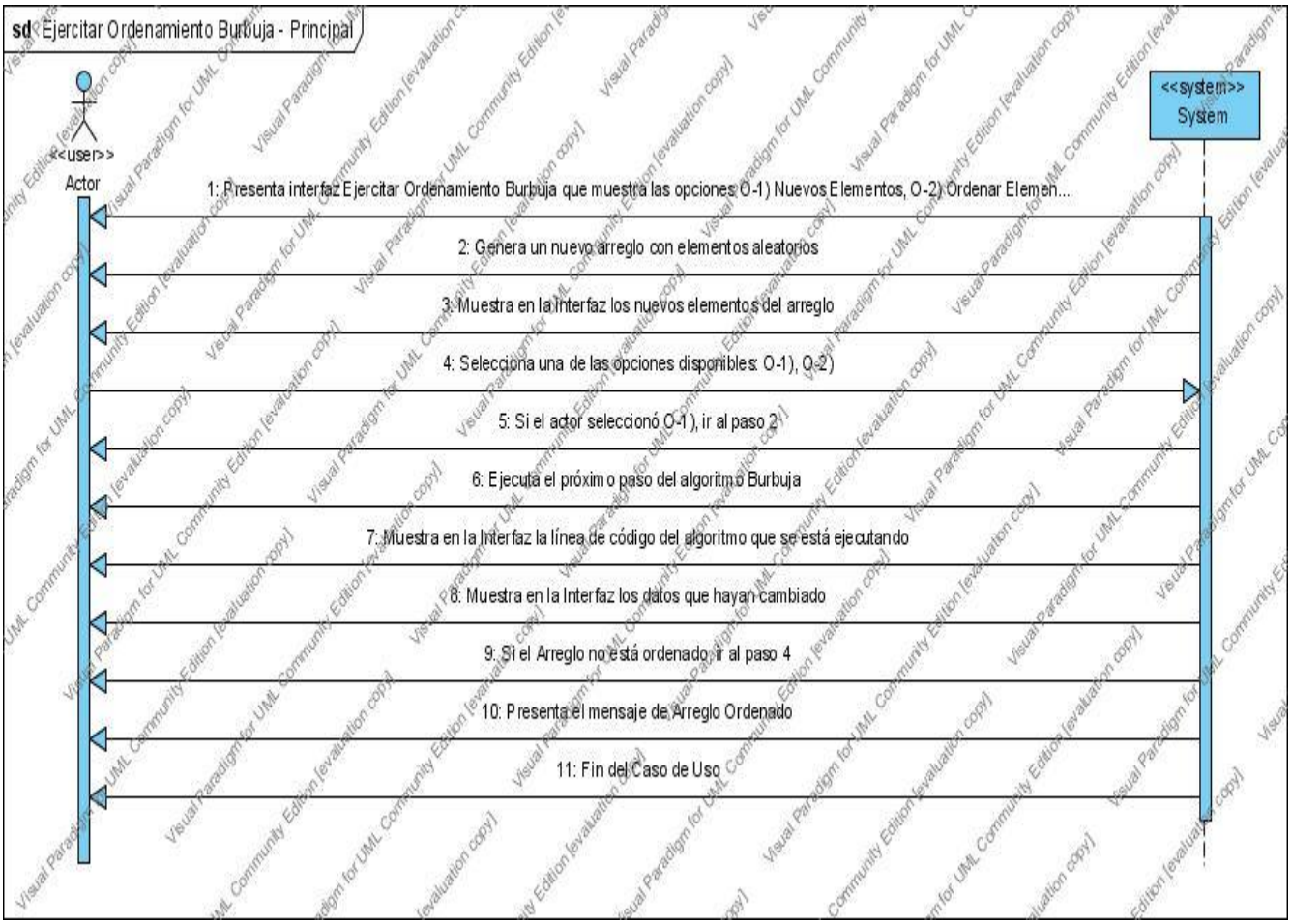


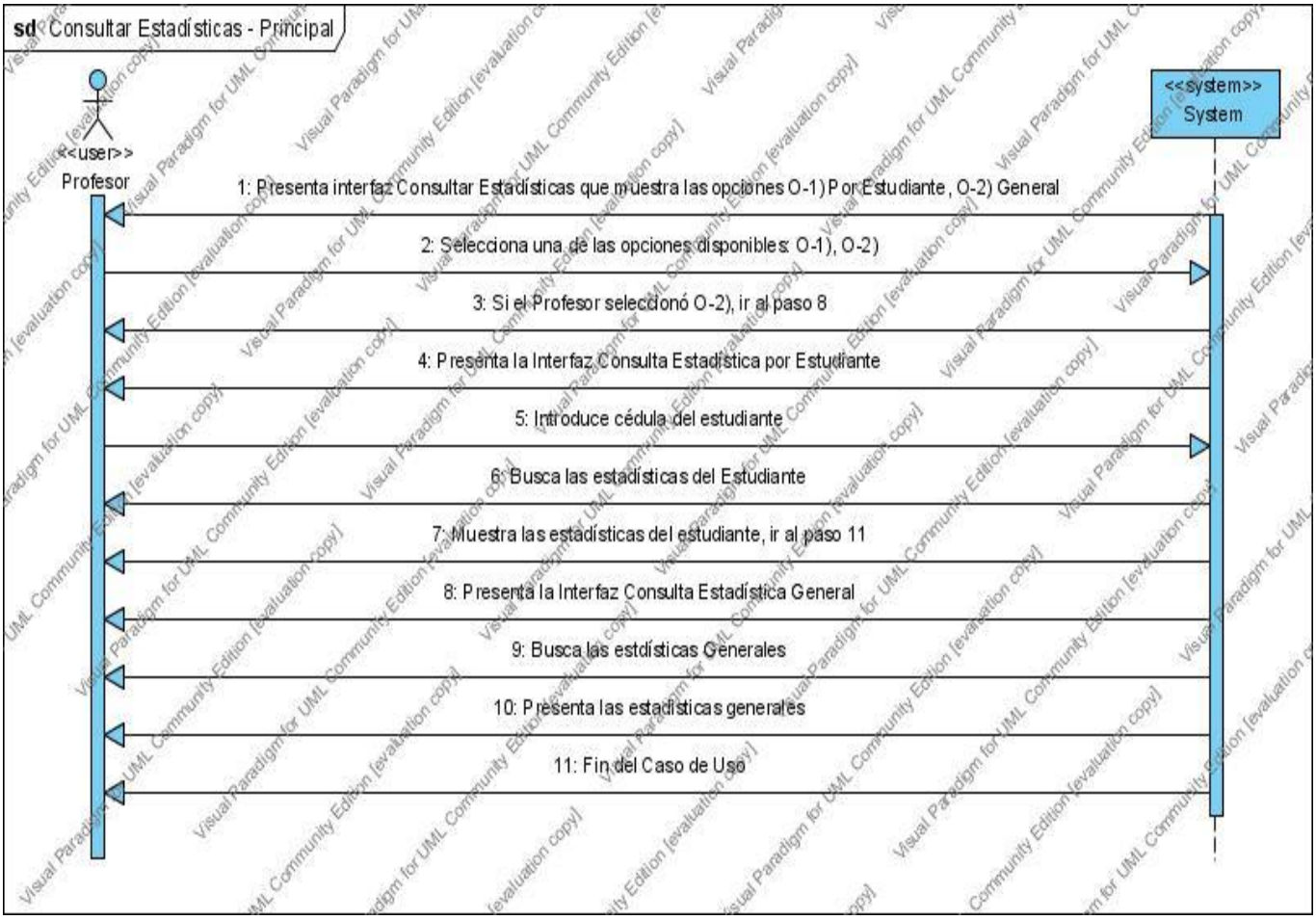
ANEXO R
Diagramas de Secuencia del sistema SEDPRO

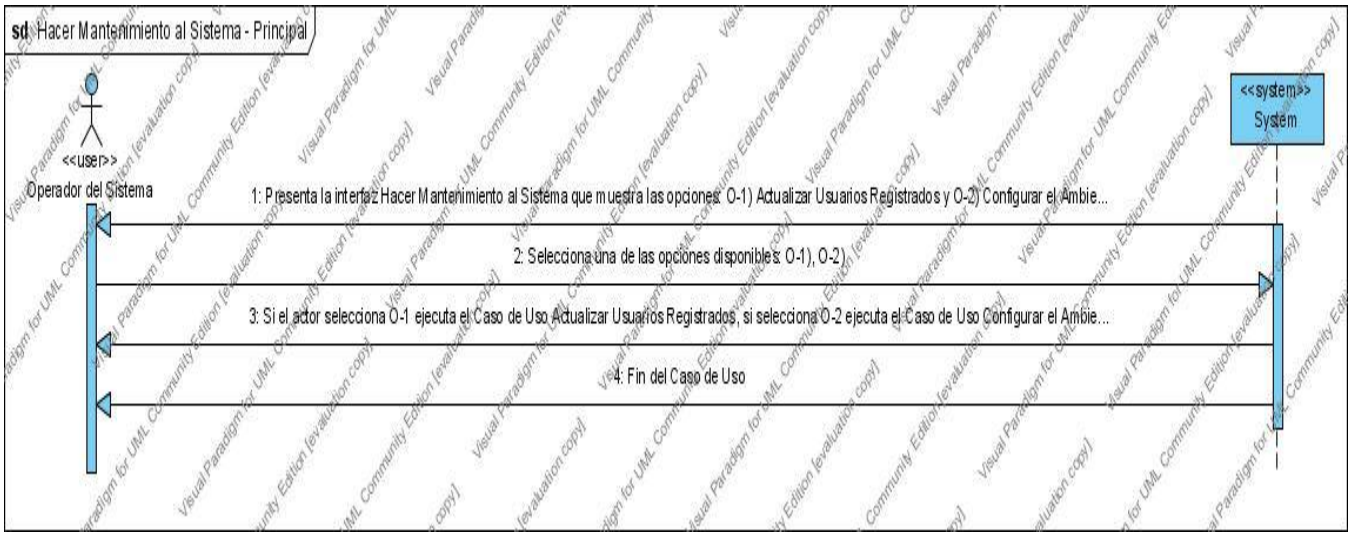


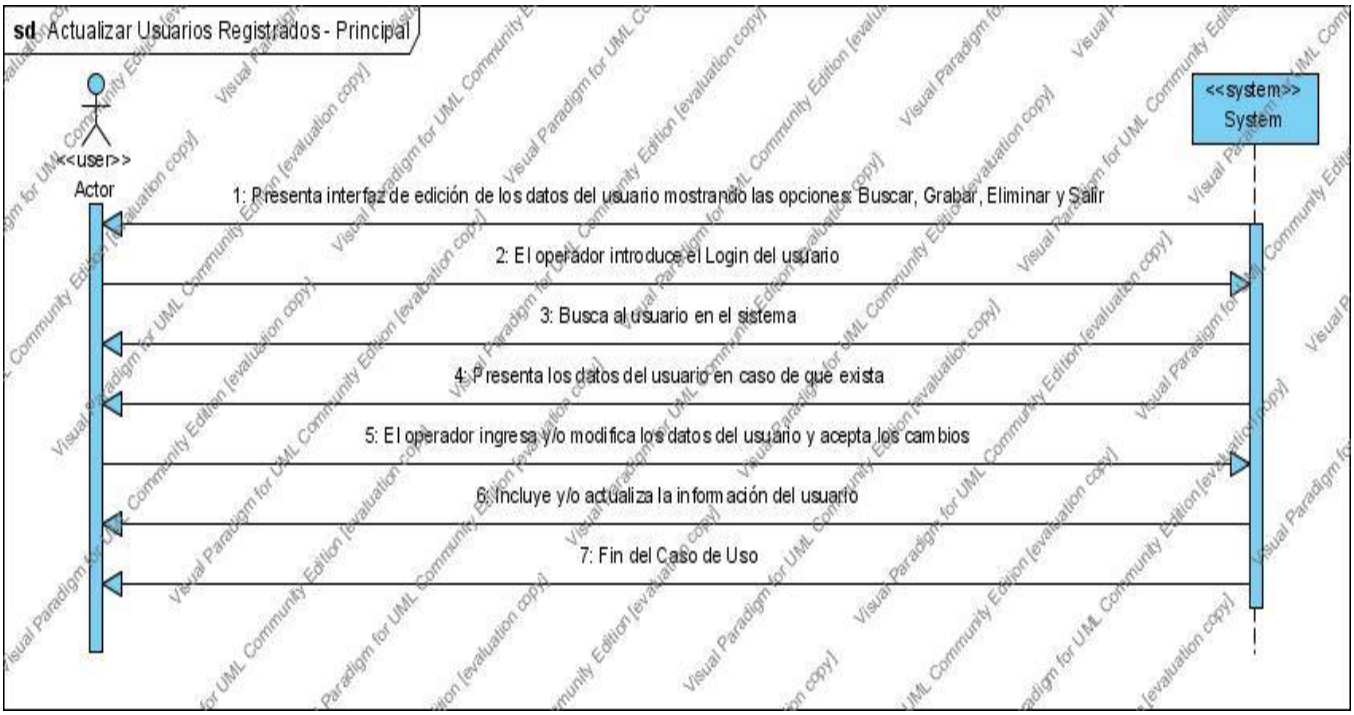


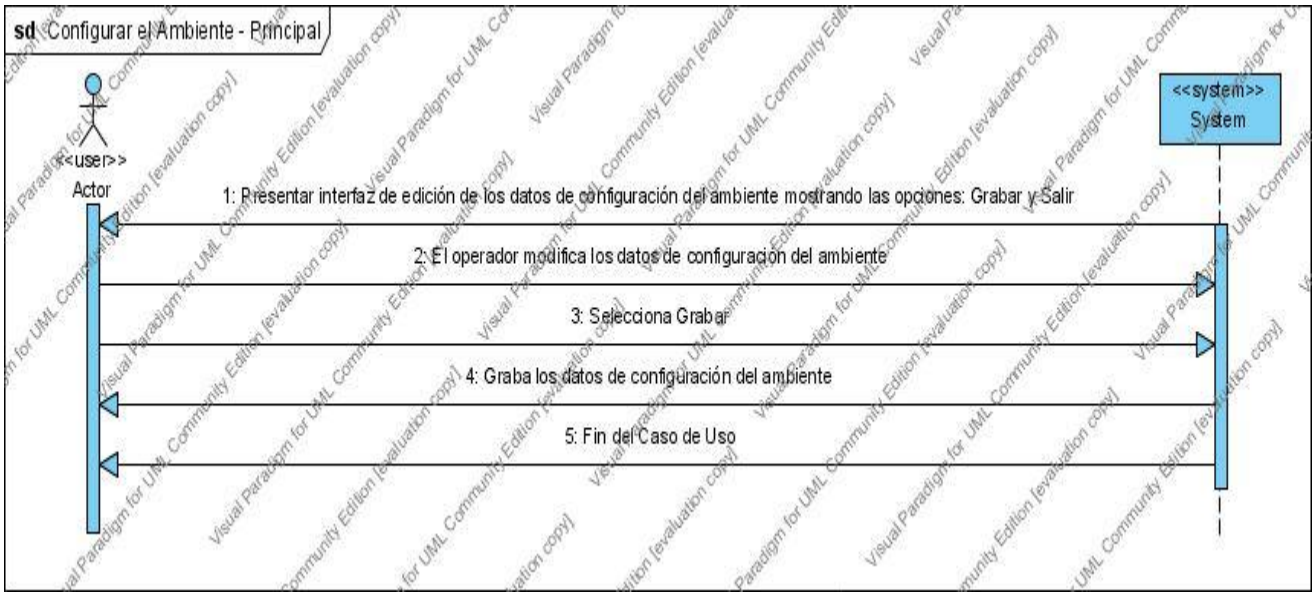

sd Ejercitar con Arreglo - Principal
<<user>>
Actor
<<system>>
System
1: Presenta la interfaz Ejercitar con Arreglo que muestra las opciones: O-1) Ejercitar Ordenamiento Burbuja, O-2) Ejercitar Búsqueda Lineal y O-3) Ejercitar Búsqueda Bin...
2: Selecciona una de las opciones disponibles: O-1), O-2), O-3)
3: Si el actor selecciona O-1 ejecuta el Caso de Uso Ejercitar Ordenamiento Burbuja, O-2) Ejercitar Búsqueda Lineal y O-3) Ejercitar Búsqueda Bin...
4: Ejecuta el Caso de Uso Registrar el Uso
5: Fin del Caso de Uso

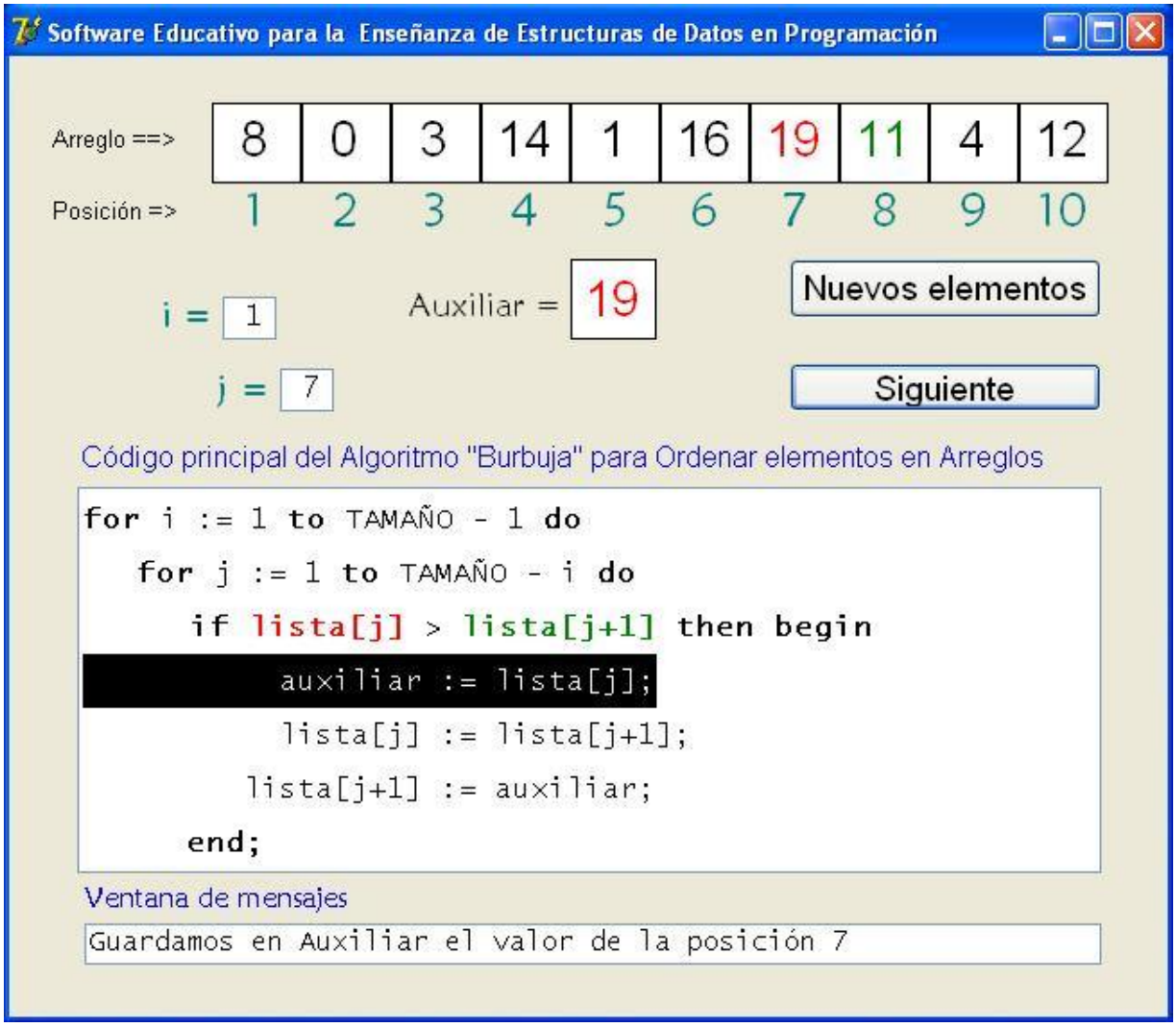

sd Ejercitar Ordenamiento Burbuja - Principal
<<user>>
Actor
<<system>>
System
1: Presenta interfaz Ejercitar Ordenamiento Burbuja que muestra las opciones: O-1) Nuevos Elementos, O-2) Ordenar Elemen...
2: Genera un nuevo arreglo con elementos aleatorios
3: Muestra en la Interfaz los nuevos elementos del arreglo
4: Selecciona una de las opciones disponibles: O-1), O-2)
5: Si el actor seleccionó O-1), ir al paso 2
6: Ejecuta el próximo paso del algoritmo Burbuja
7: Muestra en la Interfaz la línea de código del algoritmo que se está ejecutando
8: Muestra en la Interfaz los datos que hayan cambiado
9: Si el Arreglo no está ordenado, ir al paso 4
10: Presenta el mensaje de Arreglo Ordenado
11: Fin del Caso de Uso


sd Consultar Estadísticas - Principal
<<user>>
Profesor
<<system>>
System
1: Presenta interfaz Consultar Estadísticas que muestra las opciones O-1) Por Estudiante , O-2) General
2: Selecciona una de las opciones disponibles: O-1), O-2)
3: Si el Profesor seleccionó O-2), ir al paso 8
4: Presenta la Interfaz Consulta Estadística por Estudiante
5: Introduce cédula del estudiante
6: Busca las estadísticas del Estudiante
7: Muestra las estadísticas del estudiante, ir al paso 11
8: Presenta la Interfaz Consulta Estadística General
9: Busca las estdísticas Generales
10: Presenta las estadísticas generales
11: Fin del Caso de Uso


sd Hacer Mantenimiento al Sistema - Principal
<<user>>
Operador del Sistema
<<system>>
System
1: Presenta la interfaz Hacer Mantenimiento al Sistema que muestra las opciones: O-1) Actualizar Usuarios Registrados y O-2) Configurar el Ambie...
2: Selecciona una de las opciones disponibles: O-1), O-2)
3: Si el actor selecciona O-1 ejecuta el Caso de Uso Actualizar Usuarios Registrados, si selecciona O-2 ejecuta el Caso de Uso Configurar el Ambie...
4: Fin del Caso de Uso


sd Actualizar Usuarios Registrados - Principal
<<user>>
Actor
<<system>>
System
1: Presenta interfaz de edición de los datos del usuario mostrando las opciones: Buscar, Grabar, Eliminar y Salir
2: El operador introduce el Login del usuario
3: Busca al usuario en el sistema
4: Presenta los datos del usuario en caso de que exista
5: El operador ingresa y/o modifica los datos del usuario y acepta los cambios
6: Incluye y/o actualiza la información del usuario
7: Fin del Caso de Uso


sd Configurar el Ambiente - Principal
<<user>>
Actor
<<system>>
System
1: Presentar interfaz de edición de los datos de configuración del ambiente mostrando las opciones: Grabar y Salir
2: El operador modifica los datos de configuración del ambiente
3: Selecciona Grabar
4: Graba los datos de configuración del ambiente
5: Fin del Caso de Uso


Software Educativo para la Enseñanza de Estructuras de Datos en Programación
Arreglo ==>
8 0 3 14 1 16 19 11 4 12
Posición =>
1 2 3 4 5 6 7 8 9 10
i = 1
Auxiliar = 19
Nuevos elementos
j = 7
Siguiente
Código principal del Algoritmo "Burbuja" para Ordenar elementos en Arreglos
for i := 1 to TAMAÑO - 1 do
for j := 1 to TAMAÑO - i do
if lista[j] > lista[j+1] then begin
auxiliar := lista[j];
lista[j] := lista[j+1];
lista[j+1] := auxiliar;
end;
Ventana de mensajes
Guardamos en Auxiliar el valor de la posición 7