CIS0830IS05

http://pegasus.javeriana.edu.co/~CIS0830IS05/

VMCREATIVE: Herramienta didáctica de apoyo al proceso de enseñanza y aprendizaje de los conceptos básicos de vectores y matrices.

Autor:

Oscar Andrés Montenegro Bocanegra

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
2009

CIS0830IS05

VMCREATIVE: Herramienta didáctica de apoyo al proceso de enseñanza y aprendizaje de los conceptos básicos de vectores y matrices.

Autor:

Oscar Andrés Montenegro Bocanegra

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TITULO DE INGENIERO DE SISTEMAS

Director

Ingeniero Luis Carlos Díaz Chaparro MSc.

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
Agosto, 2009

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS

Rector Magnífico

Joaquín Emilio Sánchez García S.J.

Decano Académico Facultad de Ingeniería

Ingeniero Francisco Javier Rebolledo Muñoz

Decano del Medio Universitario Facultad de Ingeniería

Padre Sergio Bernal Restrepo S.J.

Director de la Carrera de Ingeniería de Sistemas

Ingeniero Luis Carlos Díaz Chaparro

Director Departamento de Ingeniería de Sistemas

Ingeniero Germán Alberto Chavarro Flórez

Nota de Aceptación

_____

_____

_____

_____

Ing. Luis Carlos Díaz Cahparro

Director del Proyecto

_____

<>

Jurado

_____

<>

Jurado

Diciembre, 2009

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Agradezco a mi Director Ing. Luis Carlos Díaz Chaparro por su interés, apoyo y dedicación durante la realización con el Trabajo de Grado.

A mis padres y mi hermana por su amor, paciencia y por ser la voz de apoyo en los momentos oportunos.

A mis amigos por su agradable compañía, creer en mí y ser otro motivo para seguir adelante.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE ILUSTRACIONES

ABSTRACT

When any teacher is going to prepare his/her class, the principal motivation is that every student can understand all the presented content. But the main issues for obtaining this are the complexity of the ideas and the fact that every student has different models of thoughts, which means that any of these characters can process the ideas in different ways. For facing this problem, is more common the use of didactical materials which can generate a better interaction between communicated statements from the teacher and the intention that he/she has. In this Grade Work, the main objective is making a didactical tool that reforces the teachers' explanations in the topic of vectors and matrices making more efficient and effective his/her strategy of learning.

RESUMEN

La motivación de todo docente cuando va a preparar una cátedra es conseguir que el contenido emitido sea claramente comprendido por todos los estudiantes. Pero las dificultades para conseguir este propósito radican en la complejidad de los contenidos y en el hecho de que estos últimos tienen diferentes modelos de pensamiento causando que las ideas las procesen de diferente forma. Para poder enfrentarlo, se ha recurrido al uso de materiales didácticos lo cual puede generar una mejor interacción entre lo que se comunica y su intención. En este Trabajo de Grado, el objetivo es realizar una herramienta didáctica que refuerce las explicaciones de los docentes en el tema de vectores y matrices de manera que pueda hacer más eficaz y eficiente su estrategia de enseñanza.

RESUMEN EJECUTIVO

Este documento muestra los antecedentes, objetivos, metodología propuesta, procesos, resultados y conclusiones relacionados con la creación del prototipo de la herramienta didáctica VMCreative, cuyo objetivo principal consiste en apoyar el proceso de enseñanza y aprendizaje de los conceptos básicos en arreglos (vectores y matrices) haciendo uso de metáforas y representaciones visuales. Dentro de estos conceptos básicos, se tratarán temas como la definición, el contenido y la declaración de vectores y matrices además de una breve introducción sobre ordenamientos y búsquedas.

El prototipo en mención surge de la necesidad de reforzar el aprendizaje de conceptos en ciertos temas de fundamentos de programación ya que se hace evidente la importancia de contar con alternativas que favorezcan la claridad de los conceptos presentados en las clases asociadas y la retroalimentación que el estudiante realiza al momento de estudiar o leer algún texto que toque el tema anteriormente citado.

Este Trabajo de Grado forma parte del proyecto *Software Tangible* [10], el cual fue creado por el grupo de investigación *ISTAR* del Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana con la intención de proponer modelos didácticos basados en el uso de metáforas y micromundos virtuales para reforzar los conocimientos impartidos en clase por parte de los docentes además de brindar al estudiante alternativas que contribuyan a hacer una mejor retroalimentación de los temas de estudio relacionados.

Además de presentar estas memorias y el prototipo, se presentarán otros artefactos asociados con buenas prácticas de Ingeniería de Software para el desarrollo de la aplicación. Entre ellos, un guión que describe el funcionamiento de la aplicación, el documento de requerimientos de Software (SRS), un documento de arquitectura de Software (SAD) y un plan de pruebas con su respectiva retroalimentación.

1. INTRODUCCIÓN

La programación es una actividad que requiere de mucha organización mental para poderla plasmar sobre una aplicación. Para que esta actividad se ejecute apropiadamente, se necesita que los fundamentos básicos sean bien comprendidos. Sin embargo, este proceso no es tan sencillo ya que los estudiantes no tienen la misma forma de asimilar unos conceptos que, en cierto punto, pueden ser muy abstractos. Sumado a lo anterior, el hecho de no tener un hábito de lectura desarrollado por parte de los estudiantes también contribuye a la dificultad para aprender estos temas. El resultado de este panorama se puede observar en la mortalidad académica de las asignaturas pertenecientes al Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana en la cual se imparte la teoría relacionada.

Actualmente, los departamentos de Sistemas de varias universidades han considerado una variedad de propuestas para poder explicar sus conceptos saliendo de los recursos tradicionalmente utilizados (tablero, libros y demás), siendo uno de los más utilizados el computador, a través del cual se puede explicar de forma didáctica los conceptos que el docente desea impartir. Uno de los proyectos más conocidos es CUPI2 de la Universidad de los Andes, el cual está a cargo de los Doctores Jorge Villalobos y Ruby Casallas, que consiste en la enseñanza de los principios básicos de programación durante los primeros semestres de la carrera por medio de entrenadores que son aplicaciones que muestran, de una forma práctica, el funcionamiento de los temas dados[1].

De manera particular, en el Departamento de Sistemas de la Pontificia Universidad Javeriana, el grupo de investigación *ISTAR* ha desarrollado el pro*yecto Software Tangible* con la intención de analizar, identificar y formular metáforas, representaciones visuales y modelos que permitan hacer analogías con los principales conceptos de programación y comportamiento de las aplicaciones en tiempo de ejecución de manera que se puedan entender de una mejor forma y complementen los procesos de enseñanza para el docente y de aprendizaje para los estudiantes[2].

En este contexto, la herramienta de software que se presentará a continuación, VMCreative, es uno de los trabajos asociados al proyecto de *Software Tangible* cuya intención es contribuir al proceso de enseñanza y aprendizaje de los conceptos básicos, declaración, ordenamiento y búsquedas en vectores y matrices por medio de metáforas y representaciones visuales. Esto se consiguió, primero, definiendo las

[1] Tomado de Rubio [11]

[2] Tomado de *ISTAR* [10]

ya mencionadas metáforas y representaciones visuales para después desarrollar los prototipos funcionales y después hacer unas pruebas piloto de usabilidad. Vale la pena resaltar que esta aplicación es una herramienta más al alcance del docente para apoyarse en su proceso de impartir el conocimiento, y para el estudiante, para que pueda comprender mejor las ideas recibidas y hacer su aprendizaje más significativo.

La realización del prototipo de la aplicación cuenta con tres fases: la primera que consiste en la recolección de información, en donde se considera la bibliografía, proyectos relacionados con el tema de este trabajo y entrevistas con algunos docentes del departamento de Ingeniería de Sistemas que dictan la cátedra de Pensamiento Algorítmico. La segunda fase consiste en hacer el proceso de diseño en el cual se presentan un guión, para entender la estructura y el contenido que la herramienta debería tener, y un diseño en donde se establece la metáfora central del proyecto y las representaciones visuales para cada uno de los temas. Finalmente, la tercera fase está relacionada con el desarrollo de la herramienta para la cual se utiliza el ciclo de vida en espiral teniendo en cuenta elementos de la metodología LUCID y se contemplan, entre otros aspectos, las pruebas de usabilidad.

La estructura de este documento se divide en diferentes secciones, iniciando con una descripción general en donde se describe la problemática de forma más detallada junto con la justificación del proyecto y los objetivos que se pretenden cumplir. Posteriormente, se presenta el marco teórico que muestra los fundamentos que se consideraron para la realización del prototipo de la aplicación. Luego, se detalla el proceso el cual comienza con una caracterización que es la base para el desarrollo de la aplicación para después definir los requerimientos, desarrollo y pruebas. Finalmente, se da paso a los resultados obtenidos y las conclusiones del proyecto.

2. DESCRIPCION GENERAL DEL TRABAJO DE GRADO

2.1 **Oportunidad ó Problemática**

Si bien se requiere de cierto orden mental para que una idea que contenga la solución a un problema pase de manera exitosa a la ejecución por computador, también se necesita que los conceptos básicos de programación sean entendidos apropiadamente. Esto nos invita a pensar que la calidad del programa que se lanza refleja que las enseñanzas en este campo fueron efectivamente adquiridas.

Para el docente, la misión de enseñar estos conceptos resulta un verdadero desafío porque debe cumplir con los siguientes roles al mismo tiempo[3]:

- ***Estratega:*** El docente tiene que establecer una metodología que tenga en cuenta los diversos panoramas de los estudiantes cuando empiezan a ver la asignatura: en algunos casos se puede ver el total desconocimiento de los estudiantes y en otros el hecho de no tener la suficiente disciplina que el programador debe tener para entregar un producto más o menos entendible.
- ***Motivador:*** Se puede decir que el docente también debe ser una especie de motivador ya que debe buscar en el estudiante la forma de enfrentar dificultades como la apatía, desinterés o inclusive la frustración que están estrechamente ligadas con la no apropiación de los conceptos y los malos resultados en sus exámenes.
- ***Orientador:*** Un docente también debe ser orientador porque debe ser aquella persona capaz de transmitir sus conocimientos claramente haciendo un uso efectivo y eficiente de los recursos que tenga a su alcance.

Sin embargo, la intervención del docente no es suficiente para que un estudiante logre entender los conceptos que éste le emite ya que todos los estudiantes tienen diferentes formas de pensamiento; inclusive, hay veces que la misma bibliografía no alcanza porque sus contenidos pueden complicar la convergencia entre el estilo de enseñanza del profesor y el estilo de aprendizaje del estudiante[4]. Aparte de esto, la costumbre de leer no se encuentra establecida en los colombianos y, de hecho, su actitud tiende a ser negativa[5].

[3] Tomado de Oviedo Galdeano y Ortíz Uribe [9]

[4] Tomado de Suárez Valencia, García y López Trujillo [12]

[5] Tomado de Holmes, Tangney, Fitzgibbon, Savage, Mehan [24] y de Cañas, Novak y González [25].

De acuerdo con Salcedo [26], para un estudiante resulta importante la necesidad de tener contenidos que tengan práctica al momento de hacer su proceso de aprendizaje sobre los temas de las asignaturas. Sin embargo, debe existir una estrategia que no afecte la formación del estudiante ni la calidad de la enseñanza. Por tal motivo, resulta necesario pensar en herramientas de apoyo didáctico que complementen el proceso de enseñanza-aprendizaje brindando mayor probabilidad de comprensión de conceptos.

Dada esta situación, se han realizado diferentes esfuerzos como alternativas que puedan complementar el proceso de enseñanza-aprendizaje. Uno de los avances más conocidos en el país es el desarrollo del proyecto Cupi2 de la Universidad de los Andes, el cual está a cargo de los Doctores Jorge Villalobos y Ruby Casallas. Este proyecto consiste en la enseñanza de los principios básicos de programación durante los primeros semestres de la carrera para posteriormente entrar en detalles. Las herramientas que se utilizan como complemento a la instrucción de estos conceptos son unos "entrenadores" que no son más que aplicaciones que muestran, en la práctica, la forma en que se ven reflejados los conceptos emitidos[6].

En el caso de la Pontificia Universidad Javeriana, el nuevo Sistema Académico sugiere que el estudiante dedique fuera de clase el doble de horas semanales que la asignatura contiene, lo cual implica que el estudiante, en muchas ocasiones, dependa solamente de su responsabilidad y de su capacidad de lectura para poder comprender determinados temas[7]. Más específicamente, se trata el caso de la asignatura Pensamiento Algorítmico, en donde existen varios factores que generan dificultad para asimilar su contenido: la diversidad de intereses dada por la heterogeneidad de estudiantes en cuanto a los programas de pregrado que ven la asignatura, la falta de costumbre a la lectura, lo complicado que les suele ser acostumbrarse a la notación que tradicionalmente se utiliza y las carencias derivadas del sistema educativo básico que se reflejan en el desempeño inicial de la carrera.

Ante estos inconvenientes, el grupo de investigación *ISTAR*, el cual pertenece al Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana, ha desarrollado el proyecto *Software Tangible* con la idea de analizar, identificar y formular metáforas, representaciones visuales y modelos que nos permitan hacer analogías con los principales conceptos de programación y comportamiento de las aplicaciones en tiempo de ejecución de manera que se puedan entender de una mejor forma y complementen los procesos de enseñanza para el docente y de aprendizaje

[6] Tomado de Rubio [11]

[7] Tomado de Galvis [27]

para los estudiantes[8]. Se espera que se puedan integrar más materiales didácticos para formar un conjunto de herramientas más amplio. Entre estos materiales, son de especial valor los micromundos y ambientes virtuales que son fundamentales para la realización de este proyecto[9].

2.2 **Formulación**

Respecto a lo anterior, la pregunta puede ser: ¿Cómo utilizar las representaciones visuales o metáforas para poder representar con ella los conceptos básicos de vectores y matrices de manera que sea un complemento para el proceso de enseñanza que realiza el docente y de aprendizaje para el estudiante?

[8] Tomado de *ISTAR* [10]

[9] Tomado de Rubio [11]

3. DESCRIPCIÓN DEL PROYECTO

3.1 **Visión global**

Con este proyecto, se pretende realizar una herramienta didáctica que facilite el proceso de adquisición de los conocimientos fundamentales de vectores y matrices haciendo la función de complemento para obtener los siguientes objetivos:

- Captación del concepto por parte del estudiante mejorando la calidad de tiempo de estudio que le dedica fuera de las horas de clase.
- Apoyar la labor del docente en la explicación del contenido.

Para que la herramienta alcance este cometido, se debe tener en cuenta factores como la facilidad de interacción con el usuario y que la herramienta no sea monótona; esto se pretende cumplir representando los conceptos por medio del uso de representaciones visuales y micromundos virtuales.

3.2 **Justificación**

Como se mencionó anteriormente, la programación o desarrollo de aplicaciones es una actividad que requiere de orden y claridad mental a lo largo de toda su duración y para eso es necesario tener los fundamentos comprendidos. Por medio de este trabajo, que forma parte de uno de los tantos adelantos con el proyecto *Software Tangible*, y aprovechando que uno de los temas que más se dificulta a los estudiantes en la asignatura Pensamiento Algorítmico son vectores y matrices, se pretende comprobar si con la aplicación de metáforas, representaciones visuales y micromundos virtuales existe una alternativa para que los estudiantes se apropien de los conceptos y pueda ser un punto de partida para reducir la mortalidad académica en esta asignatura.

3.3 **Objetivo general**

Desarrollar una herramienta de software que contribuya al proceso de enseñanza y aprendizaje de los conceptos básicos, declaración, ordenamiento y búsquedas en vectores y matrices por medio de metáforas y representaciones visuales.

3.4 **Objetivos específicos**

- Definir metáforas o abstracciones visuales sobre los conceptos básicos de vectores y matrices y un conjunto de actividades significativas de aprendizaje relacionadas con el tema.

- Especificar los requerimientos que deberá tener la herramienta didáctica teniendo en cuenta la definición de la metáfora y las actividades significativas de aprendizaje.

- Desarrollar un prototipo funcional de la herramienta didáctica cumpliendo los requerimientos identificados y aplicando prácticas recomendadas de Ingeniería de Software.

- Realizar pruebas piloto de usabilidad de la herramienta didáctica sobre una muestra significativa de estudiantes que estén, o ya hayan cursado, la asignatura Pensamiento Algorítmico.

3.5 **Metodología propuesta**

La metodología a seguir para el desarrollo de este proyecto será haciendo la unión entre el ciclo en espiral y el modelo LUCID.

Básicamente, el modelo en espiral es un arquetipo de ciclo de vida de elaboración de software que consiste en la repetición de una secuencia de actividades: una vez que se termina con la última actividad de la secuencia implica que se ha terminado con una ronda del proceso y se vuelve a iniciar con la primera actividad. El número de rondas debe ser dispuesto por el equipo que desarrolla el proyecto. Ahora, en cada ronda, el modelo en espiral se centra mucho más hacia los riesgos que se puedan presentar en el proceso de desarrollo de software y trabaja en ellos según su prioridad.

Según Bruegge [2], Las fases que se realizan en el modelo en espiral son:

- Identificar objetivos y alternativas y definir restricciones del proyecto.
- Administrar (identificar y plan de mitigación) los riesgos que salgan según las restricciones dadas en la fase anterior.
- Desarrollo y validación de prototipo.
- Planeación de la próxima ronda según los resultados obtenidos con el prototipo.

Asimismo, Bruegge [2] explica que las actividades que se realizan a lo largo de estas fases son:

- Determinar objetivos del proyecto.
- Especificar restricciones.
- Generar alternativas de desarrollo del proyecto.
- Identificar y resolver riesgos
- Desarrollo y verificación del prototipo
- Planeación de la siguiente fase.

La siguiente gráfica explica la forma en que funciona el modelo de espiral:

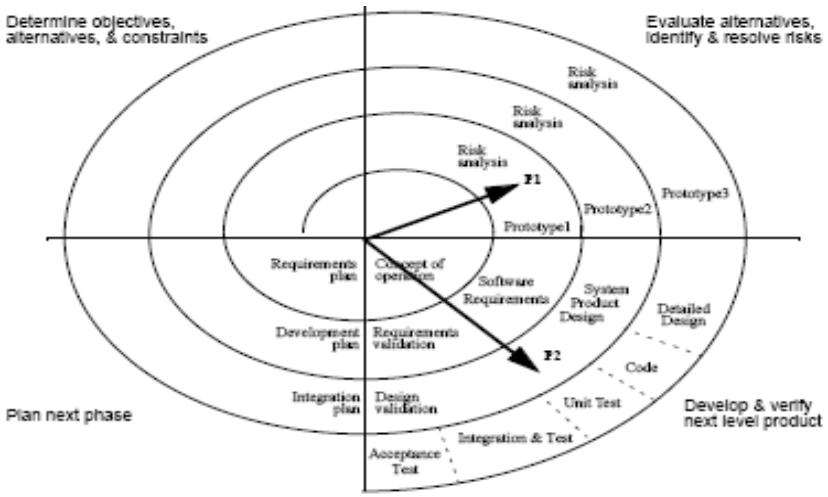


Ilustración 1. Funcionamiento ciclo de vida en espiral. Tomado de Bruegge [2]

Por otro lado, LUCID, que es un acrónimo de *Logical User-Centered Interactive Design*, es un modelo desarrollado por *Cognetics Corporation* que ha evolucionado hasta el punto de ser un framework concentrado en manejar el proceso de diseño de la interfaz de usuario de tal manera que puedan, al menos, promover la usabilidad del software en caso de que no pueda asegurarla[10].

Los objetivos que *Cognetics Corporation* persigue con su modelo son:

- Proporcionar a los diseñadores de interfaces de usuario un framework con el cual pueda aplicar buenas prácticas.
- Permitir una integración más sencilla de las actividades de diseño y usabilidad con las metodologías de desarrollo de software.
- Dar soporte para hacer mejores aproximaciones a los diseños de interfaz.
- Mejorar la usabilidad del software.

Los procesos que se siguen para obtener estos objetivos son:

[10] Tomado de Cognetics [3]

- Concepción: El objetivo es crear una visión concreta y general de la herramienta reconociendo el propósito principal con que se hace.

- Análisis: Los miembros del equipo de diseño se reúnen con el usuario o sus representantes para documentar los procesos que van a hacer parte del proyecto e identificar las necesidades específicas. Ésta información se utiliza para producir el análisis de requerimientos.

- Diseño: Se define el diseño básico del software, es decir, se define un mapa de navegación, el diseño visual y la cantidad de texto que se va a presentar. Esta etapa se caracteriza porque el proceso en general (la serie de tareas mencionadas anteriormente) se hace por iteraciones las cuales son revisadas hasta que el equipo vea que el diseño resultante cumple con los objetivos de usabilidad y sirve para completar las especificaciones de interfaz del usuario definidas con el cliente.

- Refinamiento: En esta etapa, el objetivo principal es completar el diseño básico del software. Esto se consigue arreglando los detalles de la especificación que hacen falta y añadiéndole las funciones que no habían sido implementadas antes pero que luego conviene para dar una herramienta más completa.

- Implementación: Ya con el diseño establecido y refinado, la herramienta es desarrollada en su funcionalidad e interfaz y se realiza la gestión en los cambios y problemas técnicos no esperados. También en esta etapa se crea la ayuda de la aplicación, la documentación de usuario y los tutoriales.

- Soporte: Es cuando la herramienta es presentada a los usuarios. Aunque en esta etapa generalmente no se planea mayor cosa, es una etapa clave para la aceptación del producto ya que la ayuda que se le puede presentar al usuario puede motivar a seguir utilizando el producto.

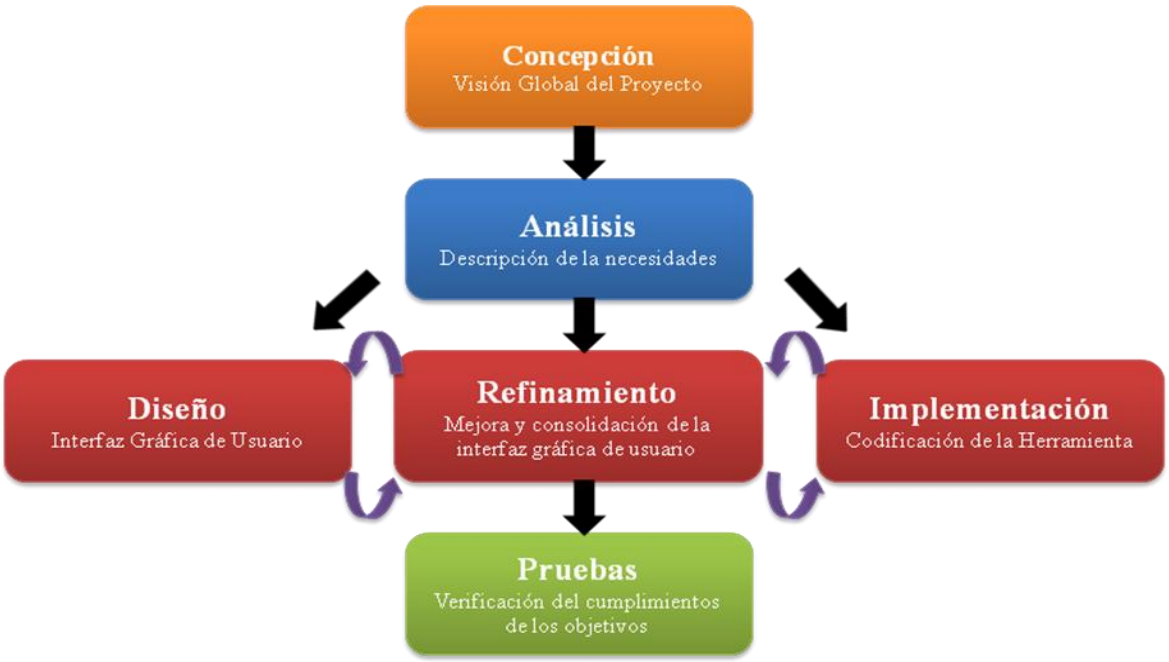


Ilustración 2. Diagrama metodología LUCID. Tomado de [47]

4. MARCO TEÓRICO

4.1 **Metáforas**

En ciertas ocasiones, la explicación del tema de una asignatura resulta ser un verdadero desafío para el docente ya que los contenidos no resultan tan fáciles de entender para el estudiante, el cual los ve por primera vez. La razón fundamental para justificar lo anterior está en el nivel de abstracción de las ideas, el cual cuanto más alto sea implica un mayor esfuerzo para el docente y lo lleva a replantear sus estrategias de enseñanza para conseguir un aprendizaje más significativo.

Una forma de salir de esta incómoda situación es por medio de las metáforas las cuales son elementos no muy nuevos en las estrategias de enseñanza, incluso, desde la edad antigua se venía utilizando. Aristóteles[11] lo definía como *"La aplicación a una cosa de un nombre que es propio de otra"*. Esta definición cobró mayor valor desde el punto de vista de la psicología educativa por la gran capacidad para poder enlazar elementos y facilitar el aprendizaje.

Actualmente, el uso de las metáforas en la educación se centra más en un mapeo entre dos conceptos con características similares (aunque puedan ser de dominios totalmente opuestos) para obtener una analogía en su funcionamiento. El uso de las metáforas es más efectivo si se utiliza como un organizador previo: crear un puente entre los contenidos previamente emitidos y las nuevas ideas como continuación a lo que anteriormente el estudiante había aprendido[12].

Ahora, Gayo [13] indica tres escenarios sobre los cuales se pueden plasmar las metáforas:

- Sistemas: Cuando el objeto de estudio se pretende explicar tiene varios componentes (mínimo dos, máximo siete). Para relacionarlo con una metáfora, se debe tener en cuenta una lista resumiendo las funciones principales de cada componente para después detallar una serie de elementos cotidianos que los puedan sustituir descartando aquellos que no se integren visiblemente al objeto de estudio. Se recomienda que si hay muchos componentes, se debe considerar una simplificación de la metáfora (es decir, dividirlo en partes) o construir una por componente.

[11] según lo enunciado por Gayo [13]

[12] Ibid.

- Conceptos tangibles: Cuando el objeto de estudio es lo suficientemente tangible como para señalar un vínculo directo con un objeto que funciona de manera análoga. Para relacionarlo con una metáfora, se toman una lista de sinónimos procedentes de ámbitos diferentes al del objeto a explicar y se van descartando los que menos relación tengan.

- Conceptos abstractos: Cuando el concepto es muy difícil de entender directamente. La estrategia más adecuada es la de enumerar el mayor número de características del concepto para poderlo representar y se ordenan en función de la relevancia de cara a su descripción. A partir de esta lista, se obtienen la lista de posibles elementos que pueden servir como metáfora y se aplican cuál de ellos puede ser más útil.

Para encontrar cuál de los términos cumple de mejor forma con el concepto a relacionar, Gentner [36] definió que las posibles candidatas deben cumplir con unos parámetros. Aquel término candidato que mejor reúna estos requisitos será la que se utilice como metáfora. Los requisitos o parámetros que deben reunir los términos son:

- Especificidad: Define qué tan inteligible es el término candidato a ser utilizado como metáfora.

- Riqueza: Precisión en la correspondencia entre el concepto a explicar y el término candidato a representar el concepto.

- Transparencia: Facilidad con la que el receptor percibe qué elementos de la metáfora aplican al concepto y cuáles no.

- Abstracción: Facilidad con la que el receptor puede relacionar el término candidato con el concepto a explicar.

- Sistematicidad: Precisión con la que cada característica del término candidato coincide con las características del concepto.

- Validez: Veracidad con la que el término candidato corresponde al concepto relacionado.

- Exhaustividad: Se mira la profundidad con la que cada característica del término candidato se relaciona con el concepto.

- Extensibilidad: Posibilidad de que el término candidato se le puedan añadir características y siga relacionándose con el concepto estudiado al momento que a este último se le sigan encontrando más cualidades.

4.2 **Representaciones visuales**

Se puede decir que una idea puede transmitirse bajo diferentes medios y que el responsable de que ésta se emita el sentido correcto es el interlocutor. Pero a veces las ideas son tan abstractas que utilizar un medio como la voz no resulta la mejor solución y puede terminar confundiendo al receptor.

Una alternativa para emitir estos mensajes son representaciones visuales en donde el Centro de Análisis y Visualización Nacional (National Visualization and Analytics Center) [32] las define como: "*La traducción de datos a una forma visible que señale características importantes, incluso las cosas en común y las anomalías.*" Con esto, se pretende aumentar la probabilidad de que una persona que recibe el conocimiento pueda percibir rápidamente las ideas que le están transmitiendo.

En el caso de un programador experto, e incluso, en el de un estudiante que está aprendiendo Fundamentos de Programación, una de las expectativas que tiene cuando está utilizando un lenguaje de programación consiste en que este último pueda hacer un seguimiento detallado de la aplicación que estamos implementando. Sin embargo, a veces estos seguimientos no son tan eficientes y no permiten que el usuario pueda entender las anomalías de su código. Con el uso de las representaciones visuales y basados en la definición que el Centro de Análisis y Visualización Nacional (National Visualization and Analytics Center) [32] ofrece, se podría aterrizar la definición dada anteriormente en el sentido de que el usuario puede percatarse de su falla, entender porqué ocurrió y de esta forma, aumentar su conocimiento.

En general, los usos que se le pueden dar a una representación visual, según Reiss [33], son:

- Ser una guía para comprender el funcionamiento de un sistema.
- Integrarlo a un debugger.
- Tener la posibilidad que con su uso se puedan hacer análisis y poder detectar cuellos de botella y anomalías.

Las representaciones visuales se pueden clasificar en las siguientes categorías (basado en lo que dice Reiss [33]):

- Representaciones concretas: En estas representaciones las líneas de código son resaltadas y señaladas mientras el código del programa se va ejecutando, y a la vez que esto ocurre, se muestran los resultados.

- Representaciones semi-abstractas: Es una representación jerárquica de la ejecución del programa. En primera instancia, se hace una visualización de los eventos y salidas del programa en alto nivel, y posteriormente, se representan de forma más detallada cada una de las acciones ocurridas.

- Representaciones abstractas: Son abstracciones que son resultado del desarrollo de los eventos originados durante la ejecución del programa.

Ahora, para visualizar las representaciones de la aplicación, Reiss [33] propone dos formas de visualización:

- Visualización *online*: Se presentan las representaciones visuales simultáneamente a los eventos que el código indica, es decir, que en el momento en que los eventos ocurren, las representaciones gráficas también suceden. Se aplican mucho en situaciones donde se necesita saber la ejecución del programa en un momento exacto (en otras palabras, es óptimo para utilizar con representaciones concretas) o cuando se hace un debug.

- Visualización *offline*: Esta visualización se da cuando el software captura los trazos del programa cuando se ejecuta y apenas este último termine, se pueden mostrar los resultados de los trazos. Esta visualización es útil cuando se quiere tener un panorama general de cómo ocurrió la ejecución del programa.

Para poder realizar una representación visual que se ajuste a la realidad que se necesita mostrar, el Centro de Análisis y Visualización Nacional (National Visualization and Analytics Center) [32] ofrece una metodología formulada en cuatro pasos:

- Entender cómo conectar los conocimientos que se tienen a la representación que hemos creado. Esto significa que de las representaciones que se han sugerido, aquella que es elegida debe cumplir con unas descripciones específicas. Para definir estas descripciones existen varios principios, pero los más importantes son los presentados por Norman [34] y Tversky [35]. Según Norman [34], los principios que rigen este entendimiento son:

    - Principio de apropiación: La representación que se tiene no debe dar ni menos ni más información que el conocimiento existente muestra.

- Principio de naturalidad: Los atributos de la representación visual deben coincidir con la información que se tiene.

- Principio de relación: La representación debe ejecutar fielmente las tareas que el usuario le plantee.

Y según Tversky [35], los principios para que una representación pueda representar el conocimiento dado son:

- Principio de congruencia: La estructura y contenido de la representación visual debe representar fielmente su dominio o tema de interés.

- Principio de aprehensión: La estructura y contenido de la representación visual debe ser tan clara que el usuario la pueda percibir y comprehender sin dificultades.

- Desarrollar unos diseños que definan formalmente las representaciones visuales y la modalidad de visualización que se pretende mostrar.

- Ejecutar la implementación de las representaciones visuales.

- Hacer las pruebas que muestran si la representación visual satisface las expectativas de presentar el conocimiento dado (código del programa) de manera fiel.

4.3 **Micromundos**

Según Galvis [6], “Un micromundo es un ambiente de trabajo reducido tan simple o complejo como amerite lo que se aprende, donde suceden o pueden suceder cosas relevantes a lo que se amerita aprender dependiendo de lo que el usuario realice”. El objetivo de crear un micromundo es identificar situaciones aptas para lograr aplicar los conceptos que se desean enseñar. Asimismo, resulta importante que el micromundo creado sea un agente asociativo entre el aprendiz, el conocimiento y el profesor para el reconocimiento de conceptos.

Existen básicamente cuatro tipos de micromundos:

- Descriptivo: Son relatos verbales que son el marco de aprendizaje que se le ofrece al estudiante. Aquí las variables de estado cambian según lo que se diga y lo que el usuario haga.

- Gráfico: Son casos vivenciales que son influidos por las decisiones que toma el usuario. Aquí, las variables de estado dependen exclusivamente del usuario. Por ejemplo, se puede tomar el caso de un simulador de vuelo al cual al con sólo mover las teclas de dirección (flechas) se le puede cambiar la posición al avión además de que cambian variables de estado como altura, temperatura, posición, combustible, etc.

- Numérico: Son las experiencias que se pueden tomar a partir de un medio como una hoja de cálculo en la cual se pueden manipular cantidades numéricas con funciones, relaciones, etc. Un caso de aplicación son las magnitudes físicas como la aceleración, altura, etc., en donde el cambio de algunas afectan a las demás.

- Sonoro: Son cuando se desarrollan ambientes generalmente "musicales" que generan alguna sensación en el usuario.

Vale la pena aclarar que un micromundo puede tener varias representaciones, pero hay que distinguirlos de los micromundos virtuales que son los que se van a utilizar en este trabajo. Según Galvis [5], un micromundo virtual es "donde se pueden vivir situaciones de las que se aprende a partir de experiencia directa (interacción del sujeto sobre el objeto de conocimiento), donde el usuario está en control del proceso (él decide qué hacer con base en el reto que se le ha propuesto, en el estado del sistema y tomando en cuenta las herramientas de que dispone), de modo que el micromundo se comporta de acuerdo con las iniciativas del aprendiz, dentro de las reglas de juego propias del mundo que se ha modelado".

Respecto a lo anterior, vale la pena decir que no se necesita de un computador para encontrar la existencia de un micromundo: el juego de roles entre humanos regido por unas reglas puede ser considerado como micromundo. Es más, el hecho de tener un computador junto a un software no garantiza ni siquiera que exista un micromundo virtual: en este caso, se tiene un ambiente interactivo.

4.4 **Estilos de aprendizaje**

A la hora de enseñar Fundamentos de Programación, un problema recurrente radica en la forma en que se transmiten los conocimientos ya que éstos suelen tener un grado de abstracción elevado y el riesgo que un docente corre de que no le comprendan es alto. Esto puede generar, además del ya mencionado desinterés, en una cadena de hechos que se resume en la debilidad de bases cognitivas que se extiende hacia más asignaturas y en una inseguridad en el estudiante.

Pero en este tipo de antecedentes se obvia un factor importante: los estudiantes tienen diferentes formas de aprender. De hecho, cada estudiante tiene su propia forma de aprender que es el resultado de haber crecido y desarrollado sobre un

contexto afectivo, cognoscitivo y psicológico [8]. Según Barros, Rojas y Sánchez [1], los principales estilos de aprendizaje son:

- Visual: Se basa en el uso de la vista para entender determinado contenido. Este tipo de personas, pueden aprender de dos tipos: visual verbal que es el que prefiere los textos; y el visual espacial que prefiere las gráficas.
- Auditivo***:*** Se basa en el uso del oído para entender determinado contenido. Este tipo de personas prefieren almacenar la información de manera secuencial y en estricto orden. Las actividades donde estas personas se sienten más cómodas son las conferencias, las exposiciones orales.
- Kinestésico***:*** Es aquel estilo de aprendizaje basado en las sensaciones y movimiento. Se caracteriza porque es un proceso algo lento pero supremamente efectivo para memorizar. Los laboratorios y proyectos son las actividades más comunes donde las personas con este tipo de conocimiento se logran apropiar de contenidos.
- Relacional***:*** Utiliza la reflexión e interacción con otros para aprender. Prefiere los trabajos en grupo, casos de estudio y simulaciones.
- Lógico matemático***:*** Utiliza el análisis, razonamiento, lógica y números para aprender. Prefiere el aprendizaje por problemas, conceptos abstractos.

Ahora, la forma de aprendizaje no es el único que se debe contemplar cuando se debe realizar una estrategia que dé como resultado una metodología que pueda facilitar la captación de contenidos: hay que tener en cuenta también el modelo de pensamiento integral que posee cada persona la cual está dividida en 4 categorías que hacen referencia al modelo de los colores establecido por Herrmann[13]:

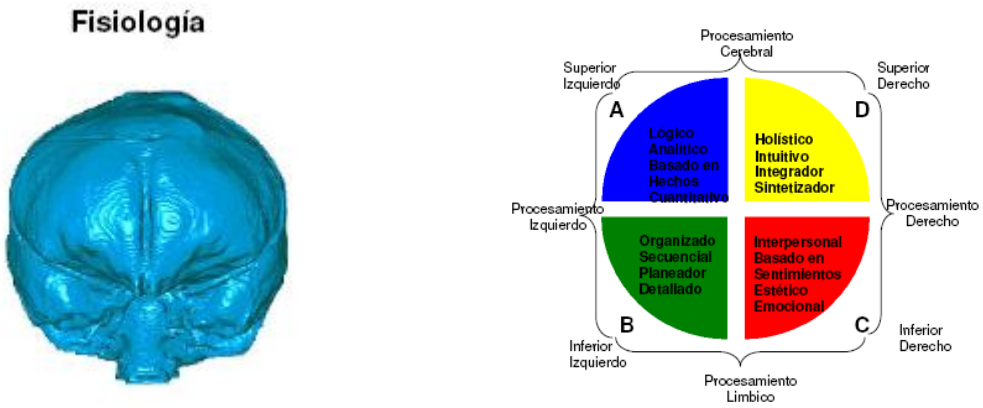


Ilustración 3. Comparación color del modelo y sus características respecto a su ubicación en el cerebro (vista desde atrás)

[13] Según lo enunciado por Barros, Rojas y Sánchez [1]

- Azul***:*** Aquella persona se caracteriza por ser analítica, sistemática y precisa. Él debe saber el origen de todos los componentes que forman un tema y cómo funcionan en conjunto: esto le permite memorizar cualquier material si encuentra la lógica a los principios y teorías que le apoyan. Tiende ser individualista y los ambientes competitivos son su predilección. Se identifica con los estilos de aprendizaje lógico-matemático, el visual-verbal y el auditivo.

- Verde***:*** Se caracteriza por ser conciso y no le interesa entrar en detalles innecesarios. La forma en que entiende las cosas es muy estructurada y requiere que el contenido en que se le presenta la información sea clara y ordenada. Los estilos de aprendizaje predominantes son el visual-verbal, relacional y el kinestésico.

- Rojo***:*** Están más orientados a aprender en ambientes que exijan e interacción con otras personas. Es un tipo de personaje al cual le gusta llegar al conocimiento dialogando y discutiendo de tal forma que se involucre emocionalmente con el material. Es especialmente bueno aprendiendo con ambientes multi-sensoriales. Los estilos de aprendizaje predominantes son el visual-espacial, el lógico-matemático y el kinestésico.

- Amarillo***:*** Es un personaje muy centrado en la asociación de pensamientos observando la forma en que los datos encajan con la visión global para entenderlo. Necesita estudio independiente y autónomo con fechas de entrega. Los estilos de aprendizaje predominantes son visual- espacial y lógico matemático.

Según Hundhausen [7], el conocimiento de un tema va a ser mejor comprendido si la estrategia o metodología de aprendizaje abarca la mayor cantidad de estilos de aprendizaje respectivos sin importar que tantas actividades sean necesarias aplicar (esto depende mucho de la cantidad de tiempo disponible que hay para explicar cada tema de la asignatura). Ahora, la capacidad para obtener un conocimiento concreto no para ahí, un refuerzo como las metáforas permite que la probabilidad de tener un aprendizaje significativo sea aún mayor.

4.5 **Didáctica**

4.5.1 ¿Qué es Didáctica?

Según Cabero [49], la didáctica se puede definir como la ciencia de la enseñanza y tiene como objetivo estudiar los procesos y elementos considerados para crear un proceso de aprendizaje completo. La didáctica tiene como componentes principales uno normativo y otro contextual que responde a la pregunta de ¿Cómo enseñar?, de manera que se cuente con los elementos suficientes para crear un proceso óptimo.

Este proceso se caracteriza por un flujo de comunicación bidireccional entre los personajes que intervienen, que según Rubio [11] son:

- El *profesor* o *docente*: se encarga de transmitir los conocimientos y dirigir y orientar el proceso de enseñanza-aprendizaje

- El *dicente* o *alumno*: recibe y adopta los conocimientos transmitidos durante el proceso

- El *contexto del aprendizaje*: representa el marco referencial dentro del cual está establecido el conocimiento a transmitir

- El *programa* o *curriculum*: sistema de vertebración de los procesos de enseñanza-aprendizaje y tiene fundamentalmente cuatro elementos constitutivos: objetivos, contenidos, metodología y evaluación

Ahora, la didáctica se puede dividir en tres categorías, que son[14]:

- Didáctica general, aplicable a cualquier individuo.

- Didáctica diferencial, que tiene en cuenta la evolución y características del individuo.

- Didáctica especial, que estudia los métodos específicos de cada materia.

4.5.2 Medios didácticos

Los medios didácticos son los elementos que normalmente utilizan los docentes para ejecutar su estrategia de enseñanza y aprendizaje. Es así como lo indica Cabero [49]: *"Son elementos curriculares, que por sus sistemas simbólicos y estrategias de utilización propician el desarrollo de habilidades cognitivas y valores en los sujetos en un contexto determinando, facilitando y estimulando la intervención medida sobre la realidad y la captación y comprensión de la información por el estudiante y la creación de entornos diferenciados que propicien los aprendizajes y el desarrollo de habilidades"*

[14] Según Rubio [11]

La idea del uso de estos medios didácticos es que sean adecuados para los estudiantes, es decir, que no sean muy sencillos (lo cual aportaría poco o nada al aprendizaje significativo de los estudiantes) y ni muy complejos como para que los estudiantes se confundan al momento de utilizar la herramienta.

Los medios educativos se dividen en las siguientes categorías (Según [50]):

- Reales: Se refiere a medios pertenecientes a un dominio cotidiano. Ejemplo: Una flor, una hoja, una guitarra.

- Escolares: Son materiales y espacios educativos. Ejemplo: Laboratorios, computador, cartulinas, tangrama.

- Simbólicos: Son los medios que se pueden percibir por medio de los sentidos. Ejemplo: libros, canciones, películas, etc.

4.6 **Software educativo**

El software educativo es una aplicación creada para ser el complemento didáctico para los docentes de manera que facilite los procesos de enseñanza y aprendizaje de algún área de conocimiento en particular.

Según Marqués [51], todo software educativo se caracteriza por:

- Su finalidad es didáctica

- Utilizan el computador como la forma para emitir un conocimiento.

- Son interactivos ya que permiten la construcción del conocimiento por parte del usuario por medio de su interacción con el computador y puede hacer una retroalimentación de su experiencia.

- Está hecho para que el usuario pueda trabajar bajo su propio ritmo.

Por el enfoque educativo que posee, Galvis [27] propone una clasificación del software educativo:

- Sistemas Tutoriales: Son sistemas que tienen la misión de guiar a los estudiantes a través de su contenido haciendo que éste personaje tome su tiempo para entender y retroalimentar todo lo que ha visto.

- Sistemas de ejercitación y práctica: Son sistemas que apoyan el proceso de enseñanza y aprendizaje de los estudiantes y muestran su contenido sabiendo de antemano que ellos ya tienen algún conocimiento. Otra cualidad que vale la pena remarcar es que estimula al usuario a continuar el uso de la herramienta por medio de juegos y retos.

- Simuladores y juegos educativos: Son sistemas que poseen las mismas cualidades que los sistemas de ejercitación y práctica pero se hacen sólo por medio de juegos educativos. Estos juegos, permiten al usuario descubrir las situaciones que se le proponen interactuando con el micromundo propuesto.

- Lenguajes sintónicos: Este sistema, al igual que los simuladores y los sistemas de ejercitación práctica, permite que el estudiante pueda resolver problemas que el micromundo le presenta pero de manera estratégica, es decir, dividiendo el problema principal en pequeños problemas en los cuales se debe enfocar.

- Sistemas expertos: Son sistemas capaces de razonar, representar y solucionar algún escenario que se les presenta como una forma para asesorar a un usuario que no es experto en el tema.

- Sistemas inteligentes para el aprendizaje apoyados por computador: Estos sistemas buscan crear funciones de aprendizaje de los estudiantes utilizando el computador como un medio educativo.

4.7 **Trabajos relacionados**

Como se mencionó en la sección 2.1, uno de los trabajos más importantes que se han hecho en cuanto a ambientes de aprendizaje de fundamentos de programación es el proyecto Cupi2 ya que cuenta con varias herramientas para que el estudiante asimile el conocimiento obtenido en clase: entrenadores, animaciones y videos hacen que las aplicaciones que fungen como ejemplos sean completas a la hora de tratar un tema. Otra característica de Cupi2 son los tutoriales enfocados en tópicos más complejos de programación y lenguajes enfocados en Internet: tal es el caso de RMI, Hibernate o PHP.

Asimismo, el proyecto Alice, que fue realizado por el Carnegie Mellon University, se ha convertido en otra alternativa bastante conocida ya que la herramienta de enseñanza es gratuita, está construida en 3D y le permite a los estudiantes tener un primer acercamiento a los conceptos básicos de Programación Orientada a Objetos por medios interactivos creando animaciones y videojuegos sencillos. Ahora, el

mismo proyecto no sólo cuenta con la herramienta, sino que también hay otro tipos de materiales como tutoriales en línea y textos guía[15].

Otro proyecto para tener en cuenta en estos ambientes de enseñanza es SCRATCH que consiste en un lenguaje que crea animaciones interactivas, juegos y reproduce música los cuales se pueden ser compartidos en la Web. Una característica importante de este proyecto es que la herramienta puede ser utilizada por adultos y niños en edades muy tempranas, más exactamente, desde los 8 años de edad. Básicamente lo que se propone para toda la población incluida es ayudarles a resolver problemas matemáticos y computacionales que los preparan para poder abstraer de manera más eficiente conceptos de programación dado el caso de que el usuario se decida por profundizar en áreas como la construcción de software. El proyecto SCRATCH fue desarrollado principalmente por el MIT Media Lab aunque cuenta con el apoyo de demás desarrolladores, diseñadores y personas ligadas al campo de las Ciencias de la Computación[16].

Un proyecto similar a Phrogram es Processing, que también es un lenguaje de programación que se encarga también de la enseñanza de fundamentos de programación pero su foco se centra más en la programación gráfica, es decir, manejo de imágenes y animaciones. Es una herramienta con licencia LPGL que trabaja para Windows, Mac y Linux y cuenta con proyectos hermanos para contribuir al crecimiento de este software creando librerías (Develop Processing) y aplicaciones móviles (mobile processing)[17].

Haciendo un enfoque más cercano al paradigma de programación estructurado, también se han realizado trabajos relacionándolos con metáforas para lograr un aprendizaje significativo a pesar de que la riqueza en contenido gráfico e interacción no sea su fuerte. Algunos son:

- Estudio experimental sobre la influencia de un organizador metafórico para la comprensión de vectores en Pascal: Trabajo de investigación que sugiere el uso de organizadores metafóricos como material de apoyo para tratar temas específicos. Uno de los casos de acción fueron los vectores y matrices en el cual se realizó una sesión de enseñanza que consistía en la enseñanza de un tutorial y la entrega de una aplicación a dos grupos de estudiantes diferentes: al primer grupo se le pasaba el organizador metafórico antes del en-

[15] Tomado de Carnegie Mellon University [14]

[16] Tomado de MIT [15]

[17] Tomado de The Phogram Company [29]

trenamiento y al otro grupo se le daba este mismo organizador después del entrenamiento. El resultado de la prueba mostró que el segundo grupo logró entender mejor la semántica de los vectores mejor que el primer grupo no lograron diferenciar apropiadamente ni la semántica relacionada ni las tareas que se debían hacer[18].

- Entorno de enseñanza basado en la comprensión de ejercicios en C: Es un tutor realizado por el Ingeniero y profesor de la Universidad de los Andes César Becerra en el cual se explica paso a paso el funcionamiento de temas básicos de programación como tipos de datos, variables, vectores y matrices, apuntadores, estructuras, manejo de archivos, entre otros[19].

Otro trabajo que vale la pena destacar es la "Construcción de un modelo de enseñanza de la programación desde su dimensión didáctica basado en Cupi2 para un adecuado desarrollo del pensamiento lógico, sistémico, formal y estructurado en el futuro ingeniero de sistemas de la Universidad Cooperativa de Colombia sede Villamaría, Caldas" ejecutada en octubre de 2006, en el cual se hace un diagnostico del potencial de enseñanza-aprendizaje que tienen los componentes de programación son utilizados por estudiantes y profesores y así construir un modelo didáctico basado en Cupi2 para enseñar conceptos básicos de programación en la Universidad Cooperativa de Colombia cuya sede se encuentra en Villamaría, Caldas[20].

En el caso de la Pontificia Universidad Javeriana, tal y como se dijo en la sección 2.1, el grupo de investigación *ISTAR*, el cual pertenece al Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana, ha desarrollado el proyecto *Software Tangible* con el fin de analizar, identificar y formular metáforas, representaciones visuales y modelos que nos permitan hacer analogías con los principales conceptos de programación y comportamiento de las aplicaciones en tiempo de ejecución de manera que se puedan entender de una mejor forma y complementen los procesos de enseñanza para el docente y de aprendizaje para los estudiantes[21].

Uno de los trabajos que forman parte de este proyecto es el realizado por Rubio [11], que es el prototipo de una herramienta didáctica para apoyar el proceso de

[18] Tomado de MacFarlane y Mynatt [16]

[19] Tomado de Becerra [17]

[20] Tomado de Suárez, García y López [12]

[21] Tomado de *ISTAR* [10]

enseñanza y aprendizaje de los conceptos fundamentales de análisis, diseño y programación Orientada a Objetos a partir del uso de metáforas, micromundos y representaciones visuales.

Otro trabajo importante que tiene que ver con el grupo de investigación *ISTAR* fue presentado por Barón y Ronderos [47], quienes desarrollaron la herramienta didáctica *TAVA,* en donde apoyaban el proceso de enseñanza y aprendizaje de variables, tipos de datos y apuntadores también bajo el uso de metáforas, micromundos y representaciones visuales.

Un último trabajo que también pertenece al proyecto *Software Tangible* merece consideración debido a que apoya el proceso de enseñanza y aprendizaje es *PsiCoder* desarrollado por Pérez y Vásquez [48], que sin utilizar micromundos ni metáforas, es una herramienta de software educativo creada para apoyar el proceso de enseñanza – aprendizaje en un curso básico de programación.

5. PROCESO

5.1 Caracterización[22] de VMCreative

En esta fase, el propósito principal es dar un acercamiento entre la teoría expuesta en el capítulo anterior y el objetivo general de la herramienta, lo cual se constituye en el punto de partida para el desarrollo del proyecto.

5.1.1 Estilos de aprendizaje a los cuales va orientada la herramienta:

De los cinco estilos de aprendizaje explicados en la sección 4.4 (visual, auditivo, kinestésico, relacional y lógico matemático), la metáfora desarrollada se dirige hacia personas en las que predominen los estilos:

- Visual: El hecho de utilizar información de manera gráfica hace que el estudiante pueda hacer mayor uso del hemisferio izquierdo del cerebro (que es en donde se aloja la parte "creativa y artística" de la persona) y tenga más probabilidades de memorizar y apropiarse de los conceptos que adquiere.

- Kinestésico: Por ser una aplicación guiada, en las personas que predomine este estilo de aprendizaje pueden apropiarse de los conceptos que se dan gracias a que ellos aprenden más de la interacción con el entorno en el cual se desenvuelven.

- Relacional: Por medio de esta herramienta, el estudiante puede reflexionar y obtener conclusiones significativas a partir de su experiencia con el objeto en cuestión.

5.1.2 Especificación del micromundo

5.1.2.1 *VMCreative como Software Educativo*

A partir de la definición y las características dadas en la sección 4.6, VMCreative es un software educativo ya que, con el hecho de interactuar con la herramienta, el estudiante puede generar un conocimiento más significativo enriqueciendo la experiencia del docente. Esto se logra cumpliendo las siguientes funciones:

[22] Según la Real Academia de la Lengua Española, caracterización es "determinar los atributos peculiares de alguien o de algo, de modo que claramente se distinga de los demás" [46]

- Informativa: Transmitir un conocimiento.

- Instructiva: Llevar un hilo conductor que orienta al estudiante en su proceso de adquisición del conocimiento.

- Motivadora: Incitar al estudiante a preguntarse qué hace la aplicación en determinada instancia facilitando el proceso de retroalimentación.

- Reflexiva: Dar el espacio al estudiante para que piense en cada una de las acciones o frases que observa y dar la oportunidad para medir estos conocimientos por medio de una retroalimentación.

- Lúdica: La aplicación emite el conocimiento desde un enfoque un poco más ligero involucrando imágenes y movimiento causando que el usuario se sienta más atraído por lo que ve y tenga más probabilidades de captar el concepto más eficientemente.

A partir de lo anterior, cabe decir que VMCreative también es interactivo ya que permite al usuario tener una mayor participación en el proceso de construcción del conocimiento, cosa que no se ve tan visible en otros medios educativos de mayor frecuencia de uso como los libros o los videos.

En cuanto al tipo de software educativo en el cual encajaría la herramienta VMCreative, se podría decir que encaja en la categoría de Sistemas de Ejercitación y Práctica ya que el estudiante posee un conocimiento previo del tema a tratar y lo que está haciendo es consolidarlo y poderlo aplicar no sólo ante los ejemplos y actividades que VMCreative ofrece, sino también ante los ejercicios que pueda encontrar en otras fuentes; además, si el estudiante ha seguido el hilo de la aplicación con ejemplos y actividades, resulta más probable que su retroalimentación sea más eficaz porque se podría dar cuenta de sus fallas más rápidamente y devolverse al tema para estudiarlo con más detalle.

Así mismo, VMCreative podría encajar también dentro de la categoría de sistema tutorial ya que posee los componentes para ejecutar su fase introductoria, orientación, aplicación y retroalimentación, pero el aprendizaje significativo que el estudiante puede obtener no sería tan provechoso en comparación a un conocimiento previo y un docente que con sus instrucciones podrían llevar a una mejor comprensión de los conceptos.

5.1.2.2 *Restricciones del micromundo*

Con la idea de que el prototipo de la aplicación VMCreative no pierda el sentido de Software Educativo explicado en el inciso anterior, se establecieron unas reglas o

restricciones que delimitan el micromundo en el que se desenvuelve la aplicación. Estas restricciones se obtuvieron principalmente de las entrevistas con algunos docentes del grupo de investigación *ISTAR* que dictan la asignatura de Pensamiento Algorítmico:

- El código que se genera para los ejemplos debe ser en C++ y no puede ser modificado.
- El contenido que se presenta debe mostrar la menor cantidad de texto posible.
- Todos los contenidos deben ser explicados a partir del uso de la representación visual.
- Ninguna sección de la aplicación debe apoyarse en el tema de posiciones de memoria para explicar su contenido.
- Para explicar el funcionamiento de ordenamientos y búsquedas, en términos del alcance del presente proyecto, es suficiente con mostrar una de las modalidades que cada uno maneja.
- Para el tema de vectores, bastará con utilizar una dimensión máxima de 3 columnas, mientras que en el caso de las matrices se considerará una dimensión de máximo 3 filas y 3 columnas.

5.1.2.3 *Actividades significativas de aprendizaje*

Las actividades significativas de aprendizaje que se pretenden alcanzar son el resultado de observar cuáles actividades podría hacer un docente con apoyo de la herramienta VMCreative para poder explicar el tema de una clase y cuáles actividades le servirían al estudiante para poder hacer una retroalimentación más eficiente respetando las restricciones del micromundo que se explicó en el inciso anterior. Las actividades significativas resultantes son:

ACTIVIDAD DE APRENDIZAJE SIGNIFICATIVO	EXPLICACIÓN
Identificar los elementos básicos para llamar un arreglo, sabiendo diferenciar entre los casos de vectores y matrices.	Definir qué es un arreglo, qué categorías hay (vectores y matrices) y que los caracteriza.
Diferenciar el significado de términos como posición, subíndice y valor e identificarlos al momento de ver un pedazo de código relacionado.	Señalar en qué parte de un ejemplo que involucre código se ubica el nombre, el subíndice y el valor y su equivalente en una animación que contenga la representación gráfica correspondiente.

Declarar vectores y matrices.	Mostrar a través de ejemplos que involucren código cómo se declara un vector y una matriz y mostrar su equivalente en una animación que contiene su representación gráfica.
Escribir valores sobre alguna posición del arreglo	Mostrar a través de ejemplos que involucren código cómo se asignaría un valor a un vector o a una matriz y mostrar su equivalente en una animación que contenga su representación gráfica.
Leer los valores que hay en una posición del arreglo.	Mostrar a través de ejemplos que involucren código cómo es el proceso para leer un valor de una posición de un vector o de una matriz y representarlo en una animación.
Inicializar arreglos	Explicar el proceso general de inicialización de un arreglo.
Distinguir las modalidades de la inicialización de arreglos.	Presentar cada uno mecanismos para inicializar un vector o una matriz con su representación en código y en una animación relacionada.
Explicar el funcionamiento del ordenamiento en modalidad burbuja.	Explicar el proceso de ordenamiento en burbuja utilizando simultáneamente un fragmento de código y una animación relacionada además de incluir una pequeña cantidad de texto como un complemento a la teoría animada.
Explicar el funcionamiento de la búsqueda en modalidad lineal.	Explicar el proceso de búsqueda en modo lineal utilizando un fragmento de código y una animación además de una pequeña porción de texto.

Tabla 1. Actividades significativas del prototipo de la aplicación VMCreative

5.1.3 Uso de las representaciones visuales

Teniendo en cuenta el micromundo con las restricciones anteriormente establecidas, las actividades de aprendizaje significativo y en el hecho de que los conceptos más importantes en los fundamentos de Programación son el reconocimiento de las variables, las funciones y las estructuras derivadas de las ya mencionadas variables, se sugirieron algunas representaciones visuales que expresarían mejor los conceptos de vectores y matrices en la aplicación VMCreative:

- Conjunto de cajas
- Caja de huevos y dentro de cada uno de ellos está un pollito.
- Cajas de hielo y dentro de cada uno de ellos está el valor a mostrar.
- Cuadrícula en la que se representa el estado de los bloques de memoria del computador cuando en una aplicación se trabaja con un arreglo.

Vale la pena aclarar que la última opción se tuvo en cuenta debido a que todavía hay profesores de la asignatura Pensamiento Algorítmico que todavía utilizan esta representación para explicar el tema.

El objetivo principal de tener varias opciones era el de encontrar aquella representación que fuera la más concreta posible y que, a su vez, permitiera una visualización "online" en el caso de que se quiera mostrar un ejemplo, obedeciendo a lo expresado por Reiss [33] en la sección 4.2:

El resultado del análisis de la elección de cada representación visual según Norman [34] y Tversky [35] se resume en la siguiente tabla:

REPRESENTACIÓN	RESULTADO SEGÚN NORMAN[34]			RESULTADO SEGÚN TVERSKY[35]	
	Apropiación	**Naturalidad**	**Relación**	**Congruencia**	**Aprehensión**
Cajas					
Huevos					
Cubitos de hielo					
Cuadrícula					

Tabla 2. Elección de la representación visual según Norman y Tversky

Como se pudo ver en la tabla 2, la caja “genérica”, la caja de huevos y los cubitos de hielo entraron en un “empate técnico”, lo que significa que cualquiera de las tres representaciones gráficas sirve para mostrar correctamente los mensajes que se muestran en la aplicación. Sin embargo, al momento de representar las posiciones de un arreglo, tanto la caja genérica como los cubitos de hielo muestran sólo la última fila del arreglo (que se ubicaría en la parte frontal de la representación gráfica). Además, sabiendo que también hay que mostrar el nombre del arreglo (que también se muestra sobre la parte frontal de la representación visual) la dificultad es aún mayor. Pero la caja de huevos puede compensar estas dos fallas utilizando como referencia el huevo para identificar las posiciones de memoria y el frente de la cubeta para ubicar el nombre del arreglo. Por tal motivo se utilizó la caja de huevos como la representación visual de la aplicación.

En la sección 5.2.2, “Implementación del proyecto”, se explicarán los detalles de la representación visual elegida para la aplicación.

5.2 Desarrollo del Proyecto

Dentro de la fase del proyecto, hay tres fases principales que son una creación de una metáfora a partir del reconocimiento de los requerimientos funcionales y no fun-

cionales, el diseño y construcción de la herramienta involucrando la metáfora validada y por último el desarrollo de unas pruebas de usabilidad.

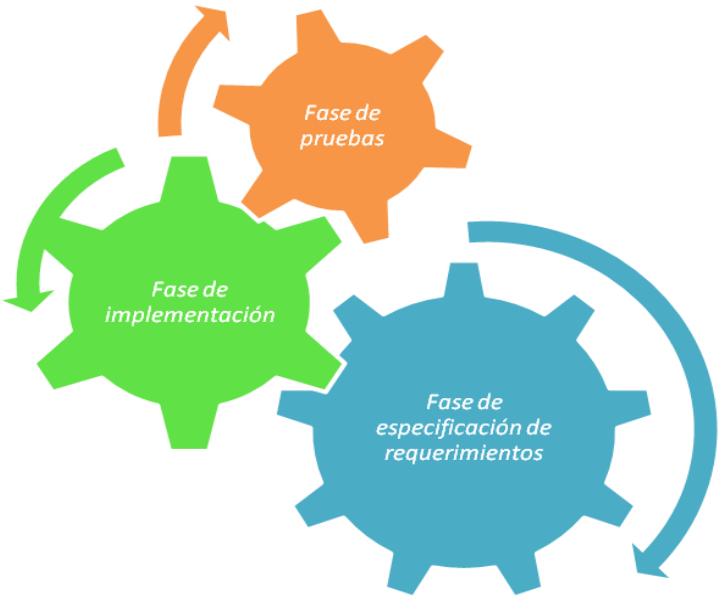


Ilustración 4. Fases de desarrollo de la herramienta VMCreative

5.2.1 Fase de especificación de requerimientos

Para la fase de especificación de requerimientos, básicamente se realizó en dos etapas: la primera consiste en la búsqueda de la bibliografía relevante al tema no sólo en micromundos virtuales, metáforas, software educativo y estilos de aprendizaje, sino también en fundamentos de programación (vectores y matrices). La segunda etapa fue realizar unas entrevistas a algunos profesores del grupo de investigación ISTAR para conocer las expectativas que les gustaría que cumpliera la aplicación.

Etapa 1: Búsqueda de bibliografía relevante

En esta etapa, aparte de la bibliografía para conocer todo lo concerniente a micromundos virtuales, metáforas, software educativo y estilos de aprendizaje, se realizó un estudio de la bibliografía que utilizan los docentes para explicar sus cátedras. A pesar de que el foco de estudio sean los estudiantes de la asignatura Pensamiento Algorítmico de la facultad de Ingeniería de la Pontificia Universidad Javeriana, se creyó conveniente reconocer la bibliografía que se utiliza en otras carreras de universidades que dictan la cátedra de Fundamentos de Programación para obtener un

panorama más concreto de los temas que se estudian. Es por este motivo que se obtuvo la bibliografía de las asignaturas Fundamentos de Programación de la Universidad Nacional [36] y Algoritmos y Programación I de la Universidad de los Andes [37]:

NOMBRE DE UNIVERSIDAD	NOMBRE DE ASIGNATURA	BIBLIOGRAFÍA
Universidad Javeriana	Pensamiento algorítmico	JOYANES AGUILAR, L. *Fundamentos de Programación*, Ed. McGraw Hill JOYANES AGUILAR L., *Problemas de Metodología de la Programación* , Mc Graw Hill BECERRA, C. *Algoritmos: Conceptos Básicos , 1995* SAVITCH, W., *Resolución de Problemas con C++*, Ed. Pearson DEITEL H.M., *Como Programar en C/C++,* Segunda Edición. Prentice Hall. 1995 JOYANES, L., *Programación en C++*, McGraw Hill. 2000 MATA TOLEDO, R., *Introducción a la programación*, McGraw Hill, 2001 WIRTH NIKLAUS., *Algoritmos y estructuras de datos*. Ed. Prentice Hall Hispanoamericana, 1987. DIKJSTRA, *a discipline of programming* FRIEDMAN FRANK L., *Problem solving, abstraction and design using c++*. 2da edición. Ed. Addison Wesley, 1997.
Universidad de los Andes	Algoritmos y Programación I	CASALLAS, R., VILLALOBOS, J.; *Fundamentos de programación: aprendizaje basado en casos.*
Universidad Nacional de	Fundamentos de Programa-	ANTONAKOS, J., *Programación Estructurada en C*,

Colombia	ción	Ed. Prentice Hall. BECERRA C. *Algoritmos Conceptos Básicos*. Ed. César Becerra, 1996. BRONSON, GARY. C++ para Ingeniería y Ciencias. THOMSON Editores. DEITEL Y DEITEL. *Introducción a la programación con C++*, 2ª ed., Ed. Prentice Hall. JOYANES, L., *Problemas de Metodología de la Programación*, Ed. McGraw Hill,1990. JOYANES, L., *Programación en C*, 1ª ed. Ed. McGraw Hill, 2001. MATA-TOLEDO, R. Y CUSHMAN P., *Introducción a la programación*, Serie Schaum, Ed. McGraw Hill 2001.

Tabla 3. Bibliografía asociada a los cursos de Fundamentos de Programación de las principales Universidades de Bogotá

De la tabla anterior, se puede concluir que los libros que más se utilizan para dictar las cátedras de Fundamentos de Programación en las correspondientes universidades son:

- JOYANES, L., *Problemas de Metodología de la Programación*, Ed. McGraw Hill,1990.
- JOYANES, L., *Programación en C*, 1ª ed. Ed. McGraw Hill, 2001.
- BECERRA C. *Algoritmos Conceptos Básicos*. Ed. César Becerra, 1996.
- MATA TOLEDO, R., *Introducción a la programación*, McGraw Hill, 2001
- DEITEL H.M., *Como Programar en C/C++,* Segunda Edición. Prentice Hall. 1995

Los libros anteriormente citados, serán el fundamento para construir el mapa de navegación que se muestran en los resultados de esta fase junto a los casos de uso

.

Etapa 2: Entrevista con docentes del grupo de investigación ISTAR

En esta etapa de la recolección de requerimientos, se tiene el objetivo principal de saber cuáles eran las principales expectativas de ellos al ver la aplicación y obtener algunos datos que puedan ayudar al proceso y a la entrega de un software que les sea útil para ellos al momento de conseguir material de apoyo para dar una cátedra y para los mismos estudiantes para que obtengan una retroalimentación más significativa de lo aprendido en clase.

Para esto, la entrevista se centró sobre cuatro ejes:

- Contexto: Se refiere a los problemas que se dan de forma más común sobre la cátedra de Fundamentos de Programación, en particular, se desea saber cuáles son los problemas de los estudiantes para entender los conceptos de vectores y matrices, cuáles son los temas más difíciles de tratar cuando se trata este tema y cuáles son las metodologías que han utilizado para explicar el tema en cuestión.

- Representaciones visuales: Sugerencias al momento de realizar la representación visual de la aplicación.

- Contenido de la herramienta: Se refiere a la obtención de un panorama que permita saber qué condiciones debe tener en cuenta la aplicación. Entre ellas: contenidos básicos que debe tener la herramienta, el lenguaje en el que deben estar los ejemplos, consideración de una sección de ejercicios y uso de texto.

- Pruebas: Aquí se considera la forma en que se deben realizar las pruebas, cuál es el objetivo que se pretende conseguir con ellas y mirar consideraciones de la población.

El resultado de estas entrevistas se pueden ver en el *Anexo 2*, en donde las personas entrevistadas dan su punto de vista sobre los preguntas de los ejes expuestos anteriormente.

Resultado de la fase de recolección de requerimientos:

Luego de la revisión de la bibliografía y de las entrevistas con algunos profesores del grupo de Investigación *ISTAR*, se obtuvo como resultado unas tareas que tiene

que cumplir la aplicación para satisfacer las actividades de aprendizaje significativo. Esto se resume en los casos de uso y mapa de navegación:

Casos de uso:

CASO DE USO #1	**Reproducir contenido**	
Objetivo en el contexto	Mostrar el contenido que tiene un tema del sistema.	
Precondición	Haber entrado al sistema.	
Condición de éxito	La animación haya terminado su ejecución.	
Condición de fallo.	La animación no se reprodujo; hay contenido en blanco	
Actores	Usuario, sistema	
Trigger	El usuario haya accedido a alguna sección del sistema.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El usuario entra a alguna sección del sistema
	2	El sistema carga el contenido de la animación y la presenta.
	3	El sistema se detiene pidiéndole al usuario si decide seguir con el contenido
	4	El usuario accede a la petición hecha por el sistema
	5	El sistema reproduce la animación hasta que finaliza.

Tabla 4. Caso de uso 1: Reproducir animación

CASO DE USO #2	**Detener animación de un ejemplo**	
Objetivo en el contexto	Permitirle al usuario detener la ejecución de un ejemplo en un punto determinado.	
Precondición	El usuario debió haber entrado en el sistema y debió haber entrado en una sección que contenga un ejemplo.	
Condición de éxito	El sistema logró detener la animación.	
Condición de fallo.	El sistema no detuvo la animación.	
Actores	Usuario, sistema.	
Trigger	El usuario ejecuta la acción de inicio de la animación del ejemplo.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El usuario le indica al sistema que puede iniciar la reproducción del ejemplo.
	2	El sistema inicia la reproducción del ejemplo
	3	El usuario pide al sistema que se detenga inmediatamente
	4	El sistema se detiene inmediatamente

Tabla 5.Caso de uso 2: Detener animación

CASO DE USO #3	**Retrasar animación del ejemplo hasta el inicio**	
Objetivo en el contexto	Retrasar la animación representativa de un ejemplo hasta su punto inicial.	
Precondición	El usuario debió haber entrado en el sistema y debió haber entrado en una sección que contenga un ejemplo.	
Condición de éxito	El sistema devuelve la animación hasta su punto inicial.	
Condición de fallo.	El sistema sigue reproduciendo la animación hasta el final.	
Actores	Usuario, sistema.	
Trigger	El usuario ejecuta la acción de inicio de la animación del ejemplo.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El usuario le indica al sistema que puede iniciar la reproducción del ejemplo.
	2	El sistema reproduce la animación del ejemplo.
	3	El usuario le pide al sistema que devuelva la animación del ejemplo hasta el inicio.
	4	El sistema se devuelve hasta el punto inicial de ejecución del ejemplo.

Tabla 6. Caso de uso 3: Retrasar animación hasta el inicio

CASO DE USO #4	**Adelantar animación hasta el final**	
Objetivo en el contexto	Adelantar la animación representativa de un ejemplo hasta su punto final.	
Precondición	El usuario debió haber entrado en el sistema y ubicarse en una sección que contiene un ejemplo.	
Condición de éxito	El sistema adelanta la animación hasta su punto final.	
Condición de fallo.	El sistema no llega hasta el punto final del ejemplo dada la indicación del usuario.	
Actores	Usuario, sistema.	
Trigger	El usuario ejecuta la acción de inicio de la animación del ejemplo.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El usuario le indica al sistema que puede iniciar la reproducción del ejemplo.
	2	El sistema reproduce la animación del ejemplo.
	3	El usuario le pide al sistema que adelante la animación del ejemplo hasta el final.
	4	El sistema se va hasta el punto final de ejecución del ejemplo.

Tabla 7. Caso de uso 4: Adelantar animación hasta el final

CASO DE USO #5	**Mostrar contenido del tema**	
Objetivo en el contexto	Presentar el contenido de una de las secciones de la aplicación.	
Precondición	Haber entrado en el sistema	
Condición de éxito	El sistema muestra el contenido de la sección solicitada por el usuario.	
Condición de fallo.	El sistema no muestra nada.	
Actores	Usuario, sistema	
Trigger	El usuario entra al sistema.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El sistema muestra la pantalla principal
	2	El usuario elige el capítulo y la sección que desea ver.
	3	El sistema carga el contenido y lo presenta al usuario.

Tabla 8. Caso de uso 5: Mostrar contenido del tema

CASO DE USO #6	**Seleccionar posición de un arreglo**	
Objetivo en el contexto	Permitir al usuario seleccionar determinada posición del arreglo cuando el sistema se lo permita.	
Precondición	Haber entrado en el sistema y haber ejecutado un ejemplo.	
Condición de éxito	El sistema deja indicada la posición del arreglo que desea ver.	
Condición de fallo.	El sistema no deja indicada la posición del arreglo que desea ver.	
Actores	Usuario, sistema.	
Trigger	El sistema termina la ejecución del ejemplo.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El sistema termina la ejecución del ejemplo y le pide al usuario que seleccione la posición del arreglo que desea ver
	2	El usuario selecciona la posición del arreglo que desea ver
	3	El sistema deja indicada la posición del arreglo que el usuario desea ver

Tabla 9. Caso de uso 6: Seleccionar posición de un arreglo

CASO DE USO #7	**Mostrar valor de un arreglo**	
Objetivo en el contexto	Permitir al usuario mostrar el contenido de determinada posición del arreglo cuando el sistema se lo permita.	
Precondición	Haber entrado en el sistema y haber ejecutado un ejemplo.	
Condición de éxito	El sistema muestra el contenido de la posición del arreglo que desea ver.	
Condición de fallo.	El sistema no muestra el contenido de la posición del arreglo que desea ver.	
Actores	Usuario, sistema.	
Trigger	El sistema termina la ejecución del ejemplo.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El sistema termina la ejecución del ejemplo y le pide al usuario que seleccione la posición del arreglo que desea ver
	2	El usuario selecciona la posición del arreglo que desea ver
	3	El sistema muestra el contenido de la posición del arreglo que el usuario desea ver

Tabla 10. Caso de uso 7: Mostrar valor de un arreglo

CASO DE USO #8	**Validar ejemplo**	
Objetivo en el contexto	Mostrar el resultado de la ejecución de un ejemplo conforme ha avanzado.	
Precondición	Entrar en el sistema y haber ejecutado el ejemplo.	
Condición de éxito	El sistema termina la ejecución del ejemplo indicando el resultado postrado por este último.	
Condición de fallo.	El sistema no ejecuta el ejemplo.	
Actores	Usuario, sistema.	
Trigger	El usuario ejecuta un ejemplo de una sección del sistema.	
DESCRIPCIÓN	**Pasos**	**Acción**
	1	El usuario pide al sistema que ejecute el ejemplo.
	2	El sistema ejecuta el ejemplo mostrando el lugar en donde se encuentra y los valores resultantes.
	3	El sistema finaliza la ejecución del ejemplo mostrando los valores finales citados en su ejecución.

Tabla 11. Caso de uso 8: Validar ejercicio

Mapa de navegación:

El mapa de navegación que se muestra a continuación fue construido a partir de los contenidos en común de los libros más utilizados en las Universidades que dictan la asignatura de Fundamentos de Programación, la cual está detallada en la fase 1 de esta misma sección (Búsqueda de bibliografía relevante), respetando las actividades significativas de aprendizaje que ya se habían establecido en la sección 5.1.2.3.

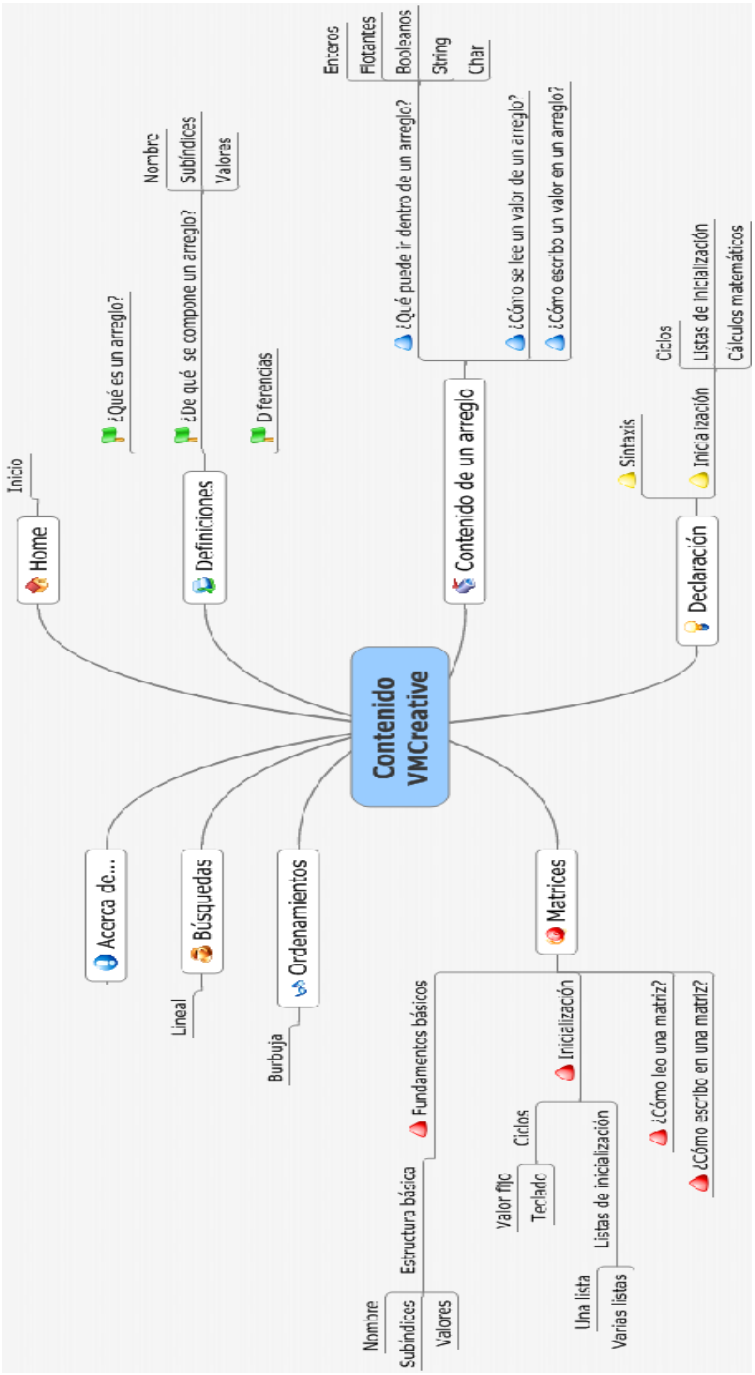


Ilustración 5. Mapa mental del mapa de navegación de VMCreative

5.2.2 Fase de implementación

Para la realización de la fase de implementación, se pudo observar que no sólo se debería tener en cuenta la programación de la aplicación, sino que también la forma en que debería presentarse la interfaz gráfica. Es por este motivo que para esta fase se decidió hacer uso de la metodología LUCID (Logical User-Centered Interface Design) [3] para complementar al ciclo de vida en espiral.

Etapas 1, 2 y 3: Concepción, análisis y diseño

A partir de lo realizado en la recolección de requerimientos (sección 5.2.1), se pudo cubrir las etapas de concepción, análisis y buena parte de diseño que propone la metodología mencionada anteriormente ya que con las entrevistas y la documentación se pudo tener una visión general de la herramienta y reconocer las principales necesidades que ésta debería satisfacer, aparte de que con el mapa de navegación ya se conocía de antemano qué contenidos deberían tratarse.

Con esto, ya se podía completar la etapa de diseño de la herramienta para lo cual se realizó un guión que servía como esqueleto para diseñar las animaciones en Adobe Flash de cada una de las secciones y añadir el texto que le acompañaría. El guión resultante se puede ver en el *anexo 3*.

Como siguiente paso, se empezó a definir la forma en que se iba a mostrar en la representación gráfica en donde se estableció que la caja de huevos debía tener una dimensión de 3 “huevos” de largo por 3 “huevos” de ancho. Cada posición tendrá un huevo de determinado color. Dentro de cada uno de ellos, se encontraría un valor representativo al tipo de dato que se quiere reflejar y para mostrarlo se debería romper el huevo. La siguiente gráfica dará una idea de lo expresado anteriormente:

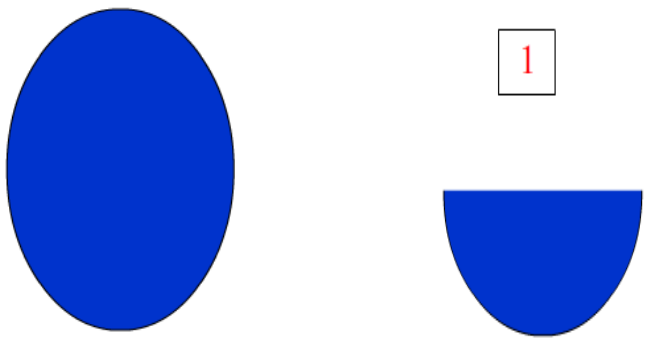


Ilustración 6. Primera versión de la representación gráfica a utilizar en la aplicación VMCreative

En la siguiente iteración de esta representación gráfica, se consideró encontrar una forma de mostrar una posición del arreglo. Para dar solución a esta situación, se propuso utilizar unos números al frente de cada huevo de manera que hagan la función de identificador de la posición del arreglo. A continuación, se muestra lo anteriormente expuesto:

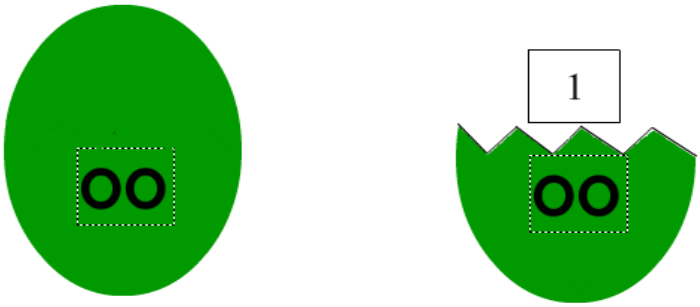


Ilustración 7. Segunda versión de la representación gráfica a utilizar en la aplicación VMCreative

La última fase de la evolución de esta representación visual consiste en terminar de refinar la apariencia del huevo, el identificador e incluir el pollito y un cartel simbolizando el valor que almacena cada posición del arreglo. Esta representación fue necesario realizarla debido a la ambientación que se tenía dispuesta para la aplicación la cual iba a recrear una granja como una forma de hacer más atractiva la aplicación. En la siguiente gráfica se muestra la forma final de la representación visual:

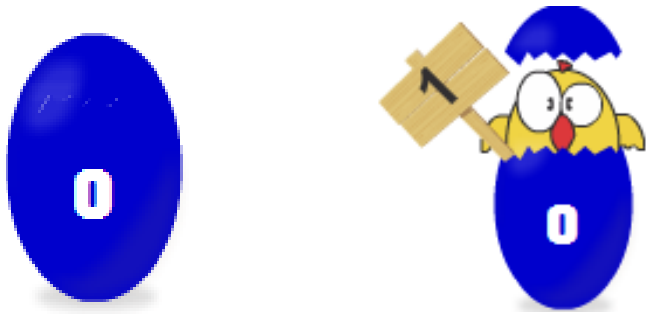


Ilustración 8. Tercera versión de la representación gráfica a utilizar en la aplicación VMCreative

Ahora, otra parte importante que se debe destacar de la representación visual son los colores de los tipos de dato que van a contener los arreglos los cuales se definieron manteniendo el estándar seguido por los trabajos anteriores del proyecto Software Tangible y que se muestran en la siguiente tabla:

COLOR DEL HUEVO	TIPO DE DATO REPRESENTADO
Verde	Real
Morado	Flotante
Rojo	Doble
Azul	Carácter
Amarillo	Booleano

Tabla 12. Tipo de dato y color a utilizar en la representación visual

Etapa 4: Refinamiento

Teniendo ya establecido el diseño de las representaciones visuales y de la estructura de la aplicación, se procede al refinamiento en donde se revisó y corrigió el contenido del guión, se verificó la navegación del mapa y se hicieron arreglos a la representación visual para que sea más congruente con el contenido de la aplicación y sea más sencilla su visualización.

Los resultados que se obtuvieron de esta etapa fueron:

- En caso de que se necesite ser señalado el huevo que representa una posición del arreglo sin la necesidad de mostrar su valor, se puede indicar con una tonalidad más fuerte o con otro color diferente a los que han sido mencionados en la tabla 11.

- La dimensión máxima de la caja de huevos, efectivamente debía ser de 3x3 ya que al considerar un área mayor, en la mayoría de las animaciones, la representación se veía mucho más apretada y no se podía distinguir bien a cuál posición del arreglo pertenecía un valor.

Etapa 5: Implementación

Como etapa final de la metodología LUCID, se dio paso a la implementación, para lo cual se utilizó el patrón Modelo Vista Controlador (MVC) [38] ya que permitía hacer avances en la lógica de la aplicación mientras que simultáneamente, se podían hacer avances sobre la apariencia gráfica. En este patrón, la lógica es el que hace las veces de *modelo* ya que se encarga de la parte de integración de los datos, mientras que la *vista* trabaja con la interfaz gráfica de usuario presentando los datos que ya fueron especificados en el modelo, y por último, el *controlador* es aquel que comunica los eventos de la interfaz de usuario a los componentes del *modelo*.

Para utilizar el modelo MVC en el prototipo de la aplicación VMCreative, se puede resumir en los siguientes paquetes:

- Vista: Es el paquete que muestra la interfaz gráfica de la aplicación basada en las acciones que recibe del modelo. Dentro de esta vista se encuentra el componente:

  Carga: La clase principal que muestra el contenido es *DemoShell-Content* del que junto a la librería Swing Suite crea el panel para acomodar el mapa de navegación y la animación que actualmente se quiere presentar.

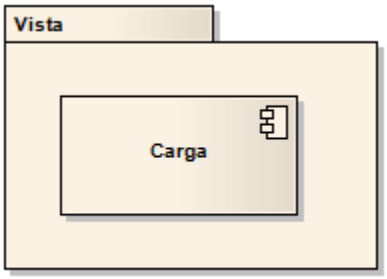


Ilustración 9. Componentes de la Vista

- Modelo: Es el paquete que representa la lógica de lo que el usuario ve a través de los componentes de la vista y asegura la integración de los datos de la aplicación. Los componentes HTML y FlashPlayer son los encargados de realizar esta labor cargando a las animaciones que el controlador registró. Esto lo hacen con las clases *JFlashPlayer* y *JHTMLEditor*.

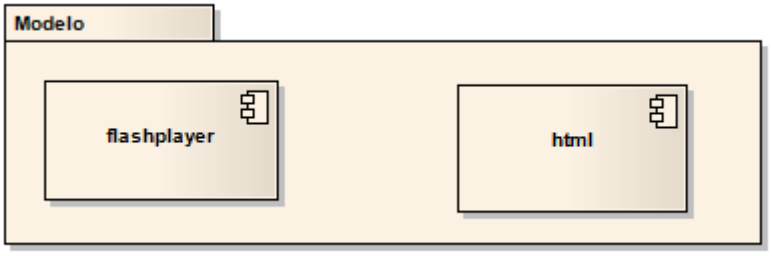


Ilustración 10. Componentes del Modelo

- Controlador: Es el paquete que se encarga de comunicar los eventos originados a los componentes del modelo. Dentro de él se encuentran los componentes:

    - Comunicación: Define las funcionalidades para hacer la comunicación con Flash, para lo cual hace uso de la librería *NativeSwing.*

    - Player: Contiene las clases que generan los llamados para cargar otras animaciones de Flash.

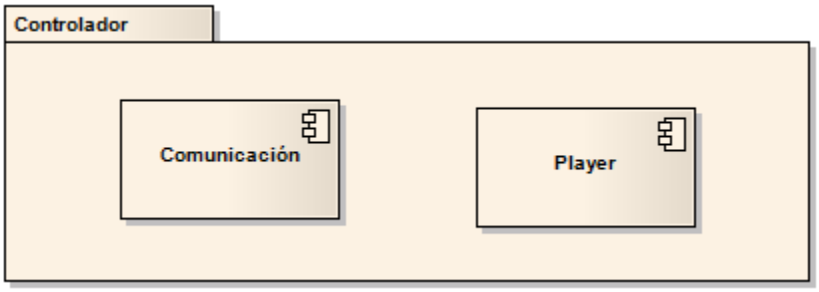


Ilustración 11. Componentes Controlador

Las librerías de apoyo para la implementación del modelo MVC fueron tomadas del proyecto DJProject desarrollado por Christopher Deckers [40] que además de dar soporte a la toma de eventos capturada, mejoran la interacción del usuario facilitando los mecanismos de comunicación con componentes de otra naturaleza.

Las características principales del proyecto DJProject son:

- El uso de la librería *NativeSwing* que hace las veces de puente de comunicación entre los componentes Swing y componentes nativos como Flash y Web haciendo el paso de mensajes.

- Posee una librería llamada Swing Suite que ofrece componentes para la creación de interfaces de usuario. Dentro de estos componentes incluyen campos de texto y de números, botones de combo y enlaces.

- Posee una librería llamada Sweet que contiene un navegador Web, un Flash Player y manejar sombreados de línea cuando se pretenda mostrar código.

Las ventajas del proyecto DJProject son:

- Posee utilidades para mejorar los primeros tales como la herramienta para solucionar problemas de hilos transparentemente.

- Reproduce animaciones en Flash

- Puede editar texto HTML en un contenedor Java.

- Puede reproducir elementos multimedia.

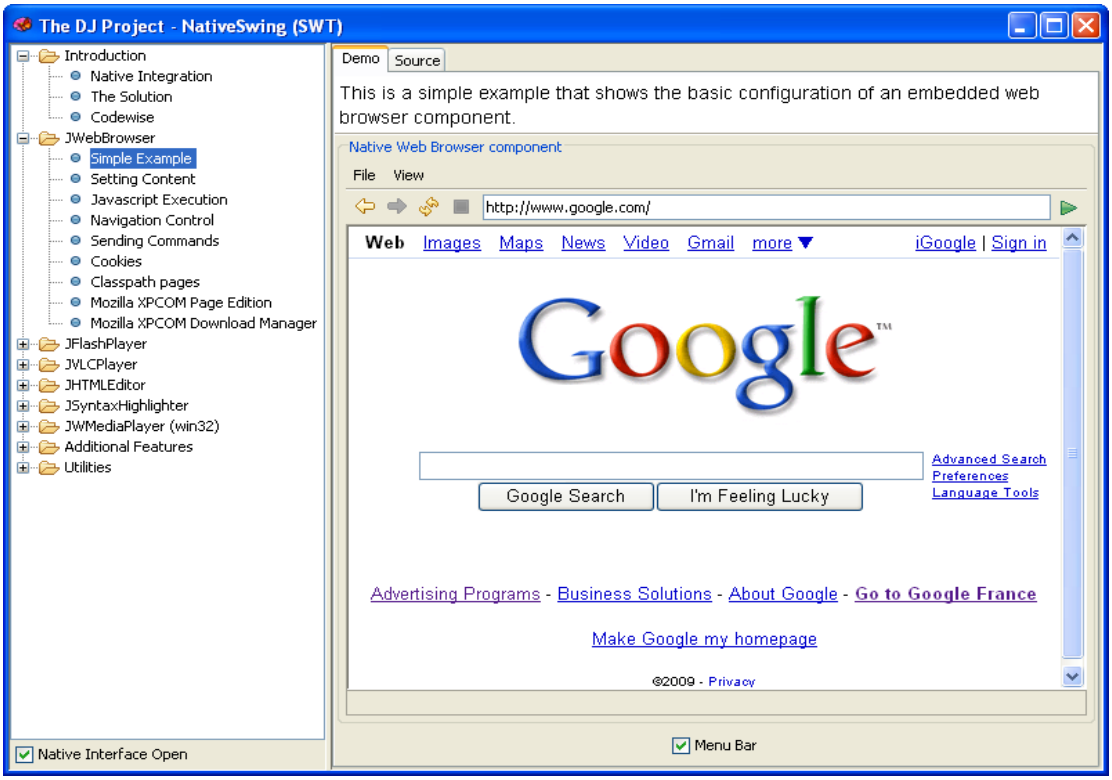


Ilustración 12. Uso de la librería NativeSwing de DJProject

En cuanto a las herramientas utilizadas para implementar VMCreative, el entorno de desarrollo utilizado fue NetBeans 6.7.1 [41], el entorno para crear las animaciones fue Adobe Flash CS3 [42] y el software para reproducir las animaciones fue Adobe Flash Player versión 10 [43]. Además, en buenas prácticas de programación, se utilizó como repositorio el CVS de *GoogleCode Project hosting* [45] y su conexión se pudo hacer con la herramienta Tortoise SVN 1.5.8 [44], con lo cual se crearon dos repositorios (ambos con licencia GPL):

- http://code.google.com/p/vmcreative/
- http://code.google.com/p/vmcreative-p2/

5.2.3 Fase de pruebas

Posterior a la segunda iteración de la implementación de VMCreative, se consideró realizar una prueba de usabilidad de la herramienta para lo cual se implemento un plan de pruebas. El objetivo principal de esta prueba era analizar la usabilidad de la herramienta según la opinión de la muestra a evaluar, comprobar si la aplicación es congruente con los conocimientos que se recibe en cátedra y saber cuáles eran las fortalezas y debilidades de la aplicación.

La estructura de la prueba se dividió en tres (3) partes:

- Usabilidad y apariencia: En esta parte de la prueba, se pretende conocer la facilidad de acceso a las secciones que en el cuestionario se piden además de observar si la apariencia de la interfaz gráfica de VMCreative es mucho, poco o no agradable para los encuestados.
- Teoría: Basados en el uso VMCreative, se quiere comprobar si las explicaciones y las animaciones dadas en la herramienta corresponden a la teoría dada por el docente.
- Complemento: En esta parte, se da la oportunidad al encuestado para que pueda dar sugerencias a la aplicación VMCreative y pueda expresar sus fortalezas y debilidades.

La muestra que se tomó fueron algunos estudiantes y profesores de la materia Pensamiento Algorítmico del departamento de Ingeniería de Sistemas de la Pontificia universidad Javeriana, que es la cátedra en donde se explica el tema de la aplicación VMCreative (vectores y matrices). Para considerar la cantidad de la muestra, se definió que ésta debía abarcar mínimo el 10% del total de estudiantes que estaban viendo la asignatura en el momento de realizase la prueba, lo cual dio un número

mínimo de veinticinco (25) alumnos considerando que el número de éstos iba a disminuir debido al tema de deserciones o retiro.

Antes de la realización de la prueba, se solicitó un permiso a algunos profesores que dictan la cátedra de Pensamiento Algorítmico para acordar la fecha y el lugar donde ellos podrían ceder un espacio de su clase para llevar a cabo la prueba. Después de solicitar estos permisos, se acordó que la prueba tendría lugar en las salas de computadores del Departamento de Ingeniería de Sistemas y que en cada una de las máquinas debía estar instalada una versión ejecutable del proyecto (VMCreative.exe)

Para la ejecución de la prueba, se brindó una inducción de ésta a los estudiantes indicándoles la finalidad con que se realizaba. El tiempo máximo de la prueba fue de treinta (30) minutos que corrían desde el momento en que se le entregaba la encuesta a los estudiantes. Vale la pena recalcar que el encuestador debía estar pendiente todo el tiempo en caso de que surgiera alguna duda por parte de los estudiantes.

Una vez terminada la prueba, se procedió a la recolección de los resultados tanto de estudiantes como de docentes. Los resultados de las pruebas se pueden ver en la sección *6.1.2 Resultados de la pruebas.*

5.3 **Reflexión Metodológica**

Se puede decir que la metodología para desarrollar la aplicación VMCreative se ejecutó exactamente a como fue propuesta y se debió a la delimitación de la herramienta en donde siempre se tuvo en cuenta que lo que se iba a construir era un software educativo y que se orientaron las actividades significativas para cumplir con los preceptos que éste maneja. Asimismo, otra razón por la cual el desarrollo se ejecutó exactamente respecto a lo que se tenía presupuestado es por el análisis hecho a cada una de las fases de cada una de las metodologías utilizadas (ciclo de vida en espiral y LUCID) lo cual permitió que se pudieran tomar los elementos que más convenían de uno (LUCID) para incorporarlo en el otro (ciclo de vida en espiral) y se tuviera una organización coherente.

A pesar de este análisis de las fases de cada metodología, el desarrollo del proyecto tuvo su demora debido al énfasis que se hizo sobre la fase de diseño para asegurar que la implementación fuera clara. Sobre lo que más se enfatizó en la fase anteriormente citada fue:

- Sobre el guión, en donde siempre hubo la intención de que el contenido de cada sección fuera explicada más por medio de las animaciones y reducien-

do al texto lo más posible dejándole la función de complemento, o bien, de inducción de la animación.

- La apariencia de la representación visual eligiendo cuál era el modelo que mejor se adaptaba a los contenidos que se iban a explicar para después, hacer las iteraciones de manera que pudieran representar fielmente los conceptos de posición, valor y subíndice y permitieran que la animación expresara exactamente lo que los ejemplos decían.

6. RESULTADOS Y RECOMENDACIONES

6.1 Resultados

6.1.1 Resultados de la aplicación

Dentro de los resultados más significativos de la herramienta VMCreative, se puede decir que uno de ellos es la forma en cómo se refleja cada concepto tomado del marco teórico sobre la aplicación. La siguiente tabla muestra lo mencionado anteriormente:

CONCEPTO DEL MARCO TEÓRICO	¿CÓMO SE REFLEJA EN VMCREATIVE?
Didáctica	Es un medio didáctico que representa de manera simbólica los conceptos que se emiten.
Software educativo interactivo	VMCreative es un software educativo ya que, por medio de la interacción del estudiante con la herramienta, éste puede generar un aprendizaje más significativo enriqueciendo la enseñanza del docente. Además, es interactivo ya que el usuario tiene activa participación al momento de construir el conocimiento ya que permite el avance de la aplicación en el momento en que él lo decida.
Micromundo	Se implementaron reglas para restringirlo: • Se utilizan cinco (5) tipos de dato y cada uno estará representado por un color diferente. • Se utilizarán dos (2) dimensiones. • Por cuestión de claridad de visualización de la metáfora respecto a la presentación de un pedazo de código de ejemplo, la dimensión máxima de la caja de huevos será, máximo, de tres por tres (3x3) • No se deben incluir explicaciones de posiciones de memoria.
Representación visual	La representación visual a utilizar fue la de la caja de huevos en donde cada uno de ellos representa una posición en un vector o una matriz.
Estilos de aprendizaje	Se utilizan el visual, kinestésico y relacional.

Tabla 13. Reflejo de la herramienta VMCreative sobre los conceptos del marco teórico

En cuanto a las actividades de aprendizaje relacionadas, que son las que apoyan el proceso de enseñanza del conocimiento, se pueden visualizar en los casos de uso de la sección 5.2.1 *Fase de especificación de requerimientos*.

Por otro lado, en cuanto al funcionamiento de la herramienta, VMCreative es una aplicación *StandAlone* que se ejecuta sobre el sistema operativo Windows. Tiene un instalador llamado VMCreativeInstaller.msi el cual permite que el usuario dejar el archivo ejecutable VMCreative.exe en el directorio de su preferencia.

En cuanto a la estructura que se puede visualizar en VMCreative, se puede decir que se divide en dos paneles:

- Navegación: Muestra un directorio en donde aparecen todos los capítulos que la aplicación contiene, y dentro de ellos, las correspondientes secciones. Para tener más información de los capítulos y secciones relacionadas con la aplicación, se puede remitir a la ilustración 4: Mapa mental del mapa de navegación de VMCreative que está ubicado en la sección 5.2.1.

- Animación: Es el espacio donde se recrean las explicaciones de las respectivas secciones con el texto y la animación respectiva.

Todos los contenidos que se ven en el panel de animaciones están hechos en *Adobe Flash CS3 Proffessional* porque permite hacer un manejo gráfico más claro y atractivo de las representaciones gráficas y de la forma en que se expresa el contenido. Estas animaciones son embebidas en la utilidad *NativeSwing* de *DJProject* para ser colocadas en el respectivo panel.

Ilustración 13. Paneles utilizados por VMCreative

Al iniciar la aplicación, la primera pantalla que se ve es la de inicio, la cual remite a los capítulos del contenido por medio de dos (2) opciones: por medio del panel de navegación y por medio de los botones que aparecen a la derecha de panel de animación. Además, muestra una explicación sobre el funcionamiento de la representación gráfica que se utiliza como analogía explicar los conceptos, además de mostrar los colores representativos a cada tipo de dato.

Ilustración 14. Pantalla principal VMCreative

Para el uso de conceptos importantes en la aplicación, se crearon pantallas emergentes que van a explicar los primeros. Para salir de esta ventana emergente, se utiliza el botón *Cerrar* el cual siempre se va a ubicar en la esquina superior derecha de la pantalla y que al activarse va a devolver la aplicación a la pantalla original. Por ejemplo, en la ilustración 15, que se puede ver a continuación, se van a explicar tres conceptos, el usuario puede elegir una de estas tres opciones. Supongamos que el usuario eligió la opción *Nombre* cuyo contenido es el que se muestra en la ilustración 16 y se presenta como una ventana emergente. Para salir de esta ventana, se ejecuta el botón *Cerrar* el cual hará que la aplicación se devuelva hasta la imagen mostrada en la ilustración 15.

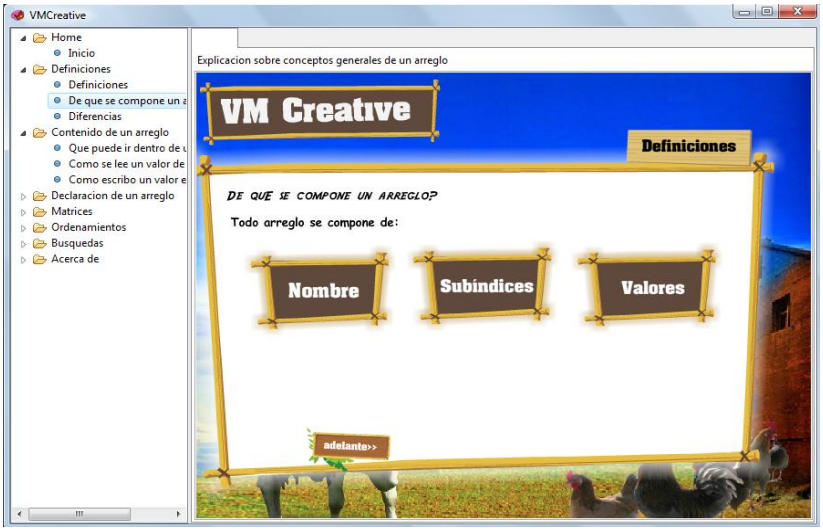


Ilustración 15. Concepto importante antes de ser accedido

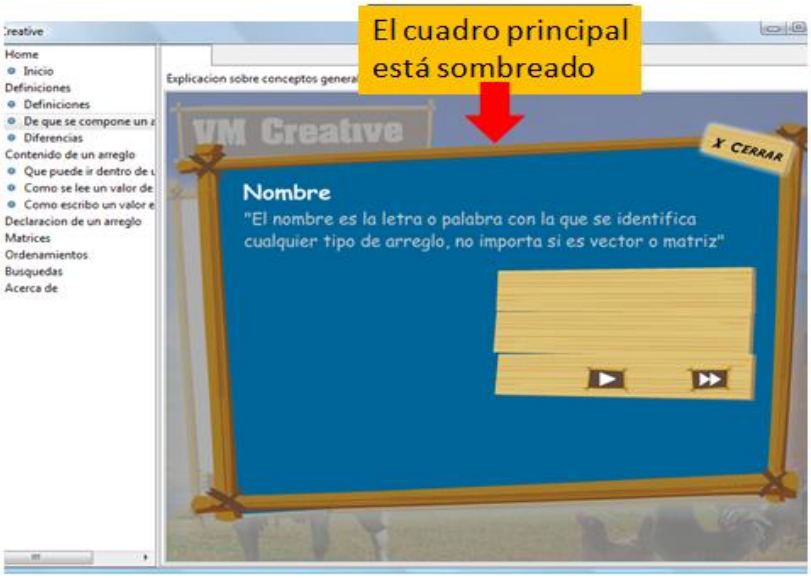


Ilustración 16. Concepto importante al ser accedido

Ahora, para explicar el comportamiento de una representación gráfica a partir de un pedazo de código que es mostrado, se hace uso de los botones de control de animación. Estos controles son utilizados para reproducir las animaciones alusivas a los ejemplos. Estos botones son:

- Retroceso completo: Devuelve la animación del ejemplo hasta el principio. El botón que lo representa es 

- *Play*: Ejecuta la animación alusiva al ejemplo. El botón que lo representa es 

- *Stop:* Detiene la animación alusiva al ejemplo. El botón que lo representa es 

- Avance completo: Avanza la animación hasta el final de su ejecución. El botón que lo representa es 

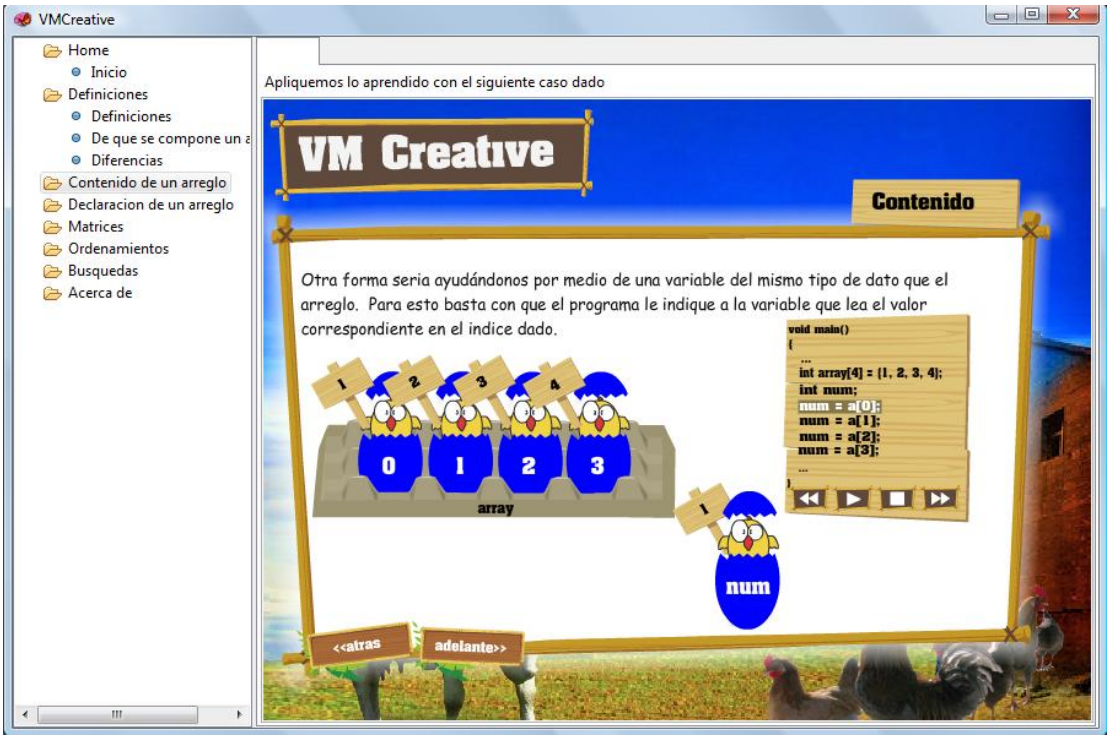


Ilustración 17. Uso de la representación gráfica en la ejecución de un ejemplo

Cuando se ejecuta un ejemplo, VMCreative sombrea la línea de código que se está ejecutando y hace su respectiva representación. Vale la pena recalcar que todos los ejemplos están en lenguaje C++ que es el utilizado en la cátedra de Pensamiento Algorítmico.

Cuando la línea de código no se ha señalado, la apariencia es la siguiente (Aclaración: el círculo rojo sólo se representa para mostrar la línea de código que aún no ha sido señalada):

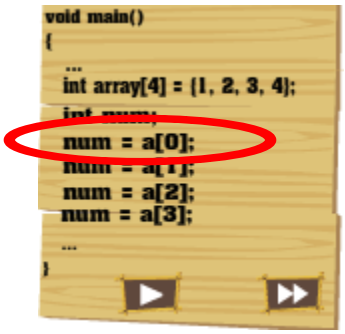


Ilustración 18. Representación línea de código cuando no se está ejecutando

Cuando se está ejecutando, la línea de código toma la siguiente apariencia (Aclaración: el círculo rojo sólo se representa para mostrar la línea de código que se ha señalado):

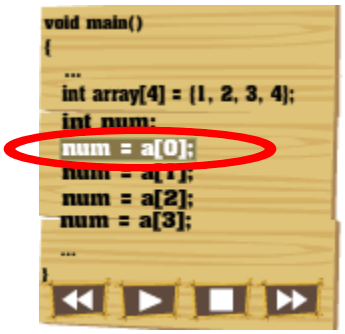


Ilustración 19. Representación línea de código cuando se está ejecutando

Y el resultado cuando la representación gráfica encarna la línea de código es:

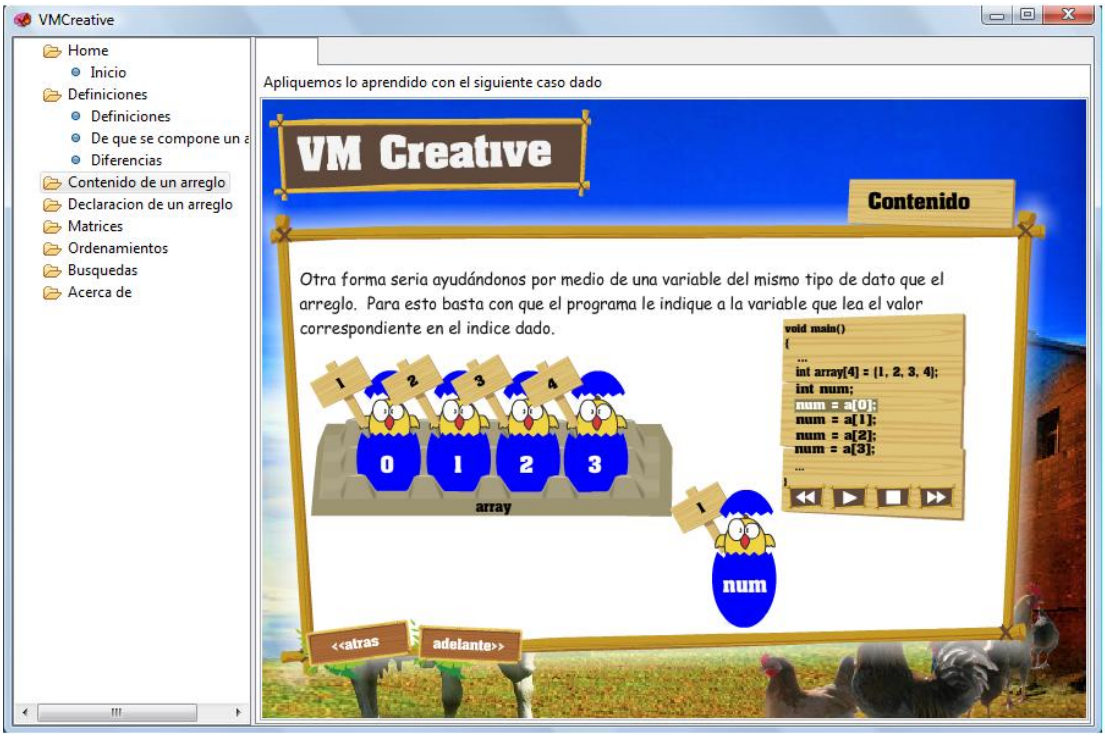


Ilustración 20. Personificación de la representación gráfica cuando se ejecuta una línea de código

En cuanto a la navegación de las secciones, se utilizan los botones *Atrás* y *Adelante* para retroceder o avanzar en la explicación de un contenido de determinada sección. La sección termina cuando no se encuentra el botón *Adelante*

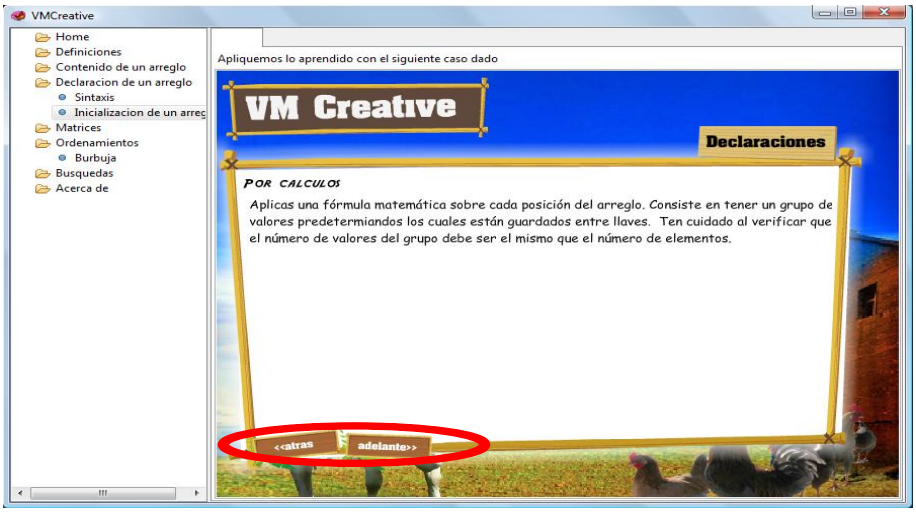


Ilustración 21. Botones de navegación dentro del contenido

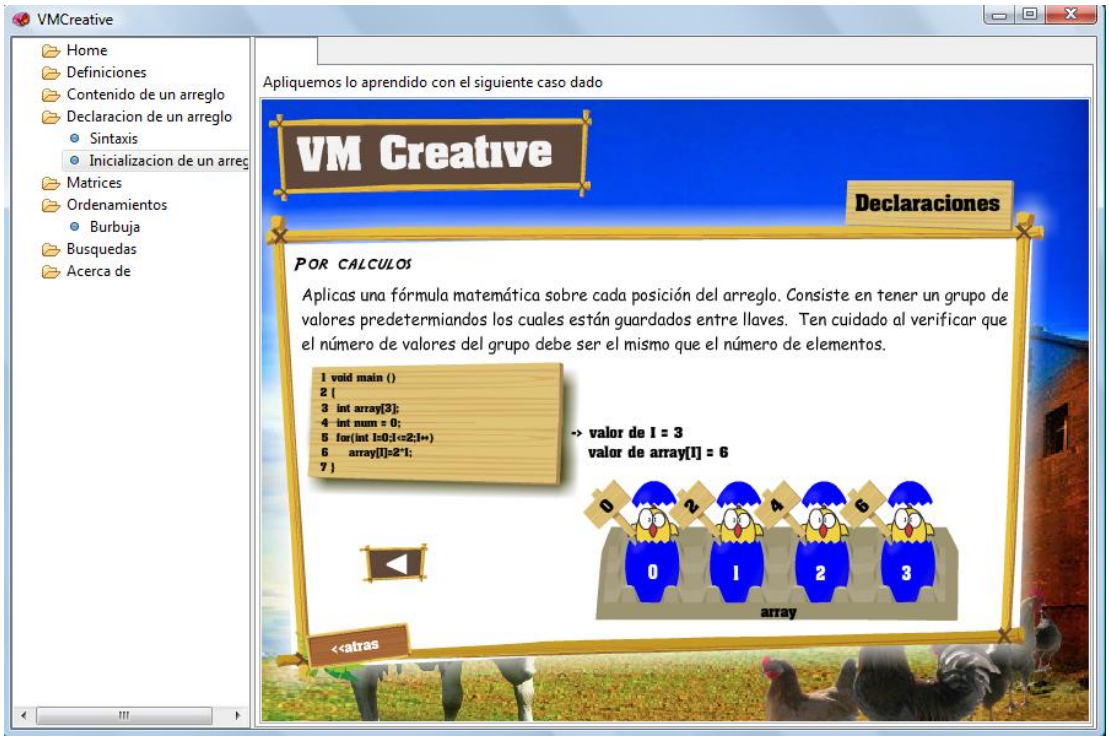


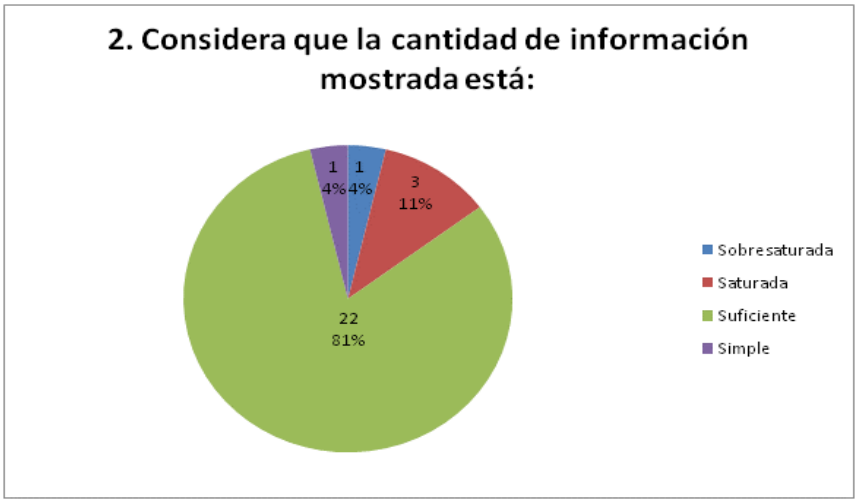
Ilustración 22. Ausencia del botón *Adelante*: indica fin de la explicación

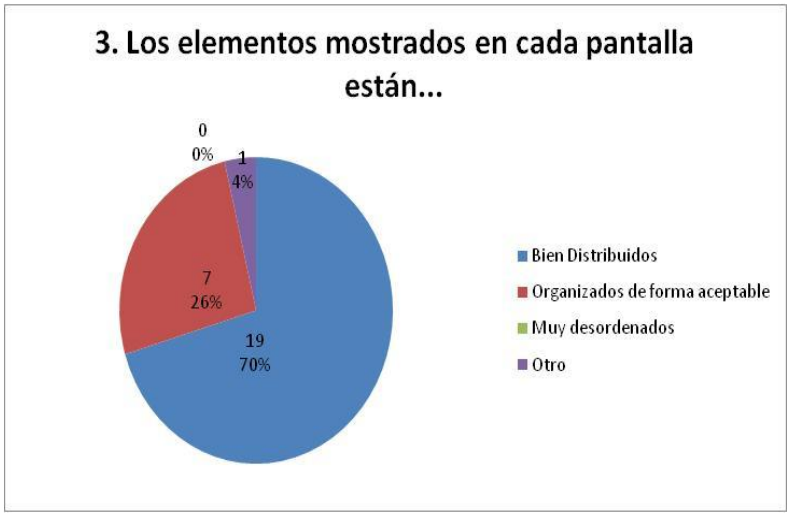
6.1.2 Resultados de las pruebas

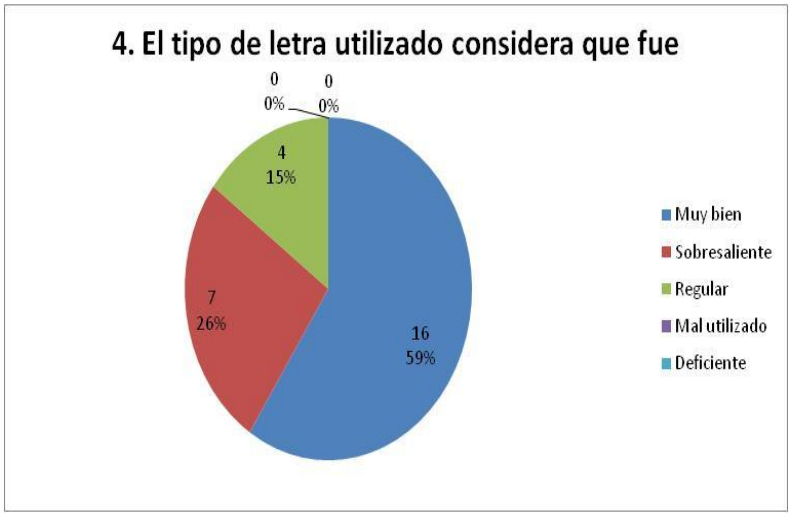
Tomando en cuenta que la población a analizar fueron los estudiantes de la cátedra de Pensamiento Algorítmico del departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana, se tomaron dos (2) cursos, de los cuales un total de veintisiete (27) alumnos se presentaron a sus respectivos grupos para realizar la prueba, superando la expectativa mínima (asumiendo que para tener una cantidad significativa de la prueba se necesitaba contar con el 10% de la población total de estudiantes que ven ésta cátedra) de veinticinco (25) que se estableció en la medición reseñada sección 5.2.3. Asimismo, dos (2) profesores tomaron también este ejercicio.

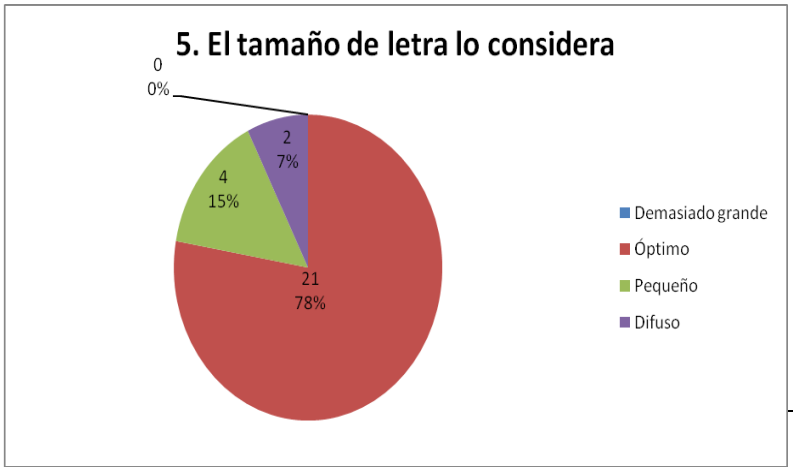
En la prueba se pidió a los encuestados que accedieran al capítulo *Matrices* de VMCreative y a partir de su interacción con la herramienta respondieran las preguntas. Ahora, en la sección anteriormente citada, se mencionó que la prueba se dividió en tres partes, las cuales analizaremos por separado:

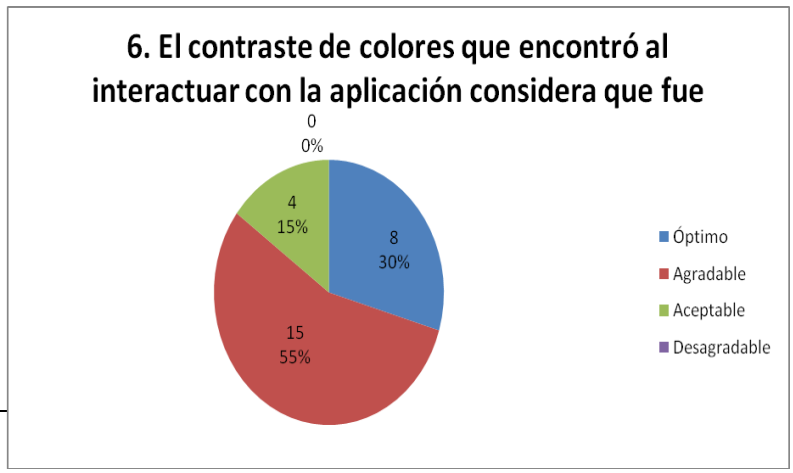
Parte 1: Usabilidad y apariencia

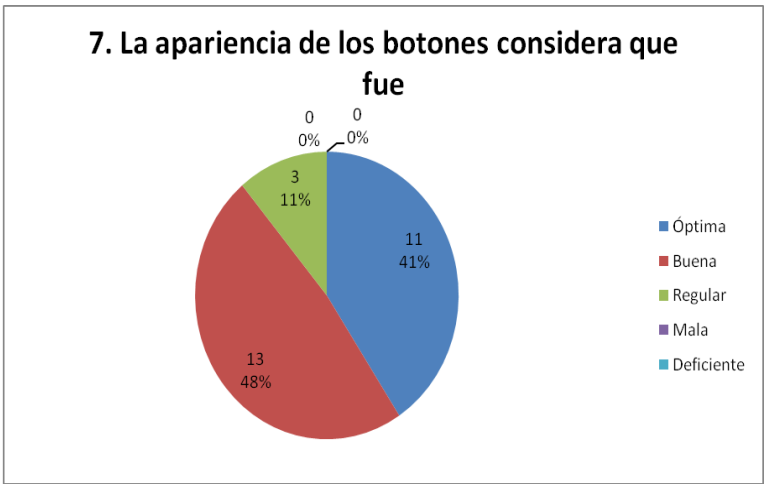


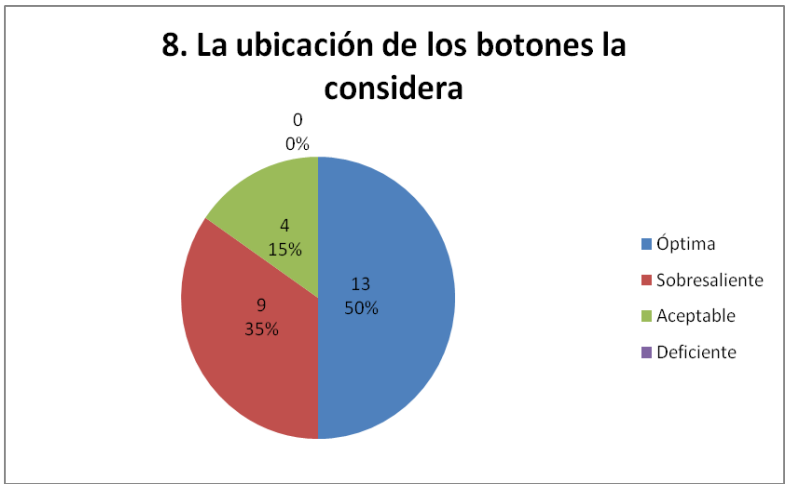


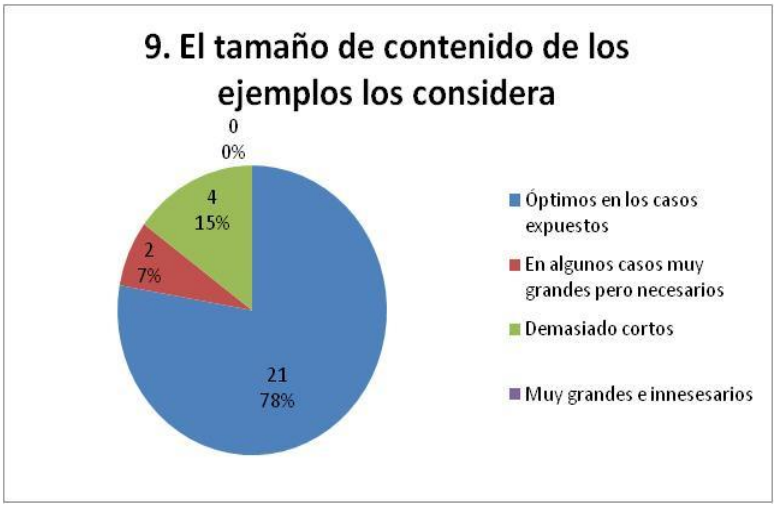


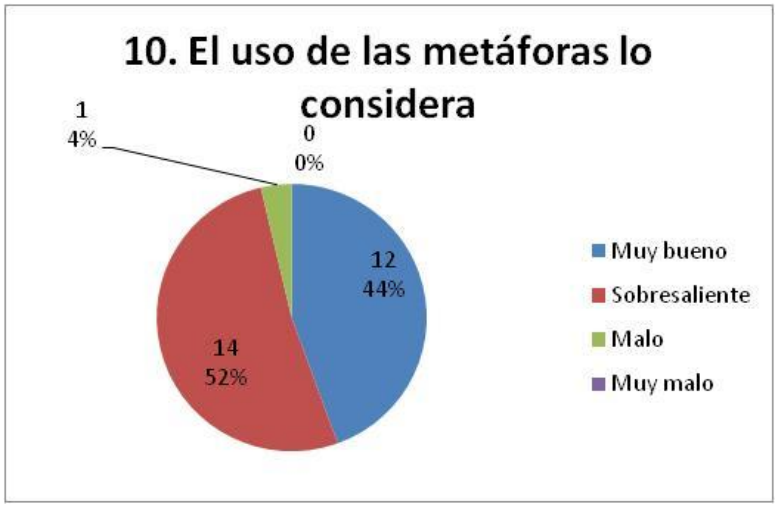


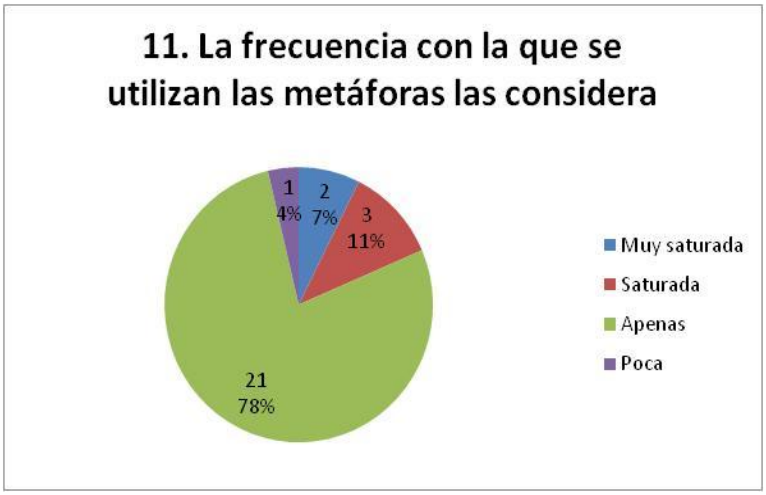


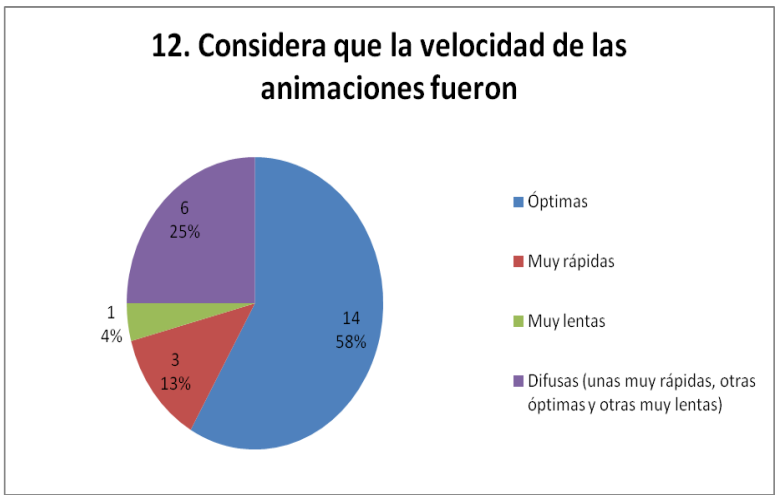

7. La apariencia de los botones considera que fue
0
0%
0
0%
3
11%
11
41%
13
48%
Óptima
Buena
Regular
Mala
Deficiente


8. La ubicación de los botones la considera
0
0%
4
15%
13
50%
9
35%
Óptima
Sobresaliente
Aceptable
Deficiente


9. El tamaño de contenido de los ejemplos los considera
0
0%
4
15%
2
7%
21
78%
Óptimos en los casos expuestos
En algunos casos muy grandes pero necesarios
Demasiado cortos
Muy grandes e innesesarios


10. El uso de las metáforas lo considera
1
4%
0
0%
12
44%
14
52%
Muy bueno
Sobresaliente
Malo
Muy malo

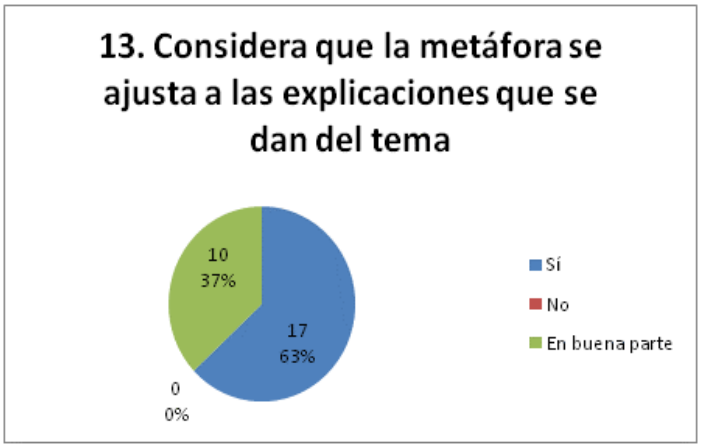

11. La frecuencia con la que se utilizan las metáforas las considera
1
4%
2
7%
3
11%
21
78%
Muy saturada
Saturada
Apenas
Poca

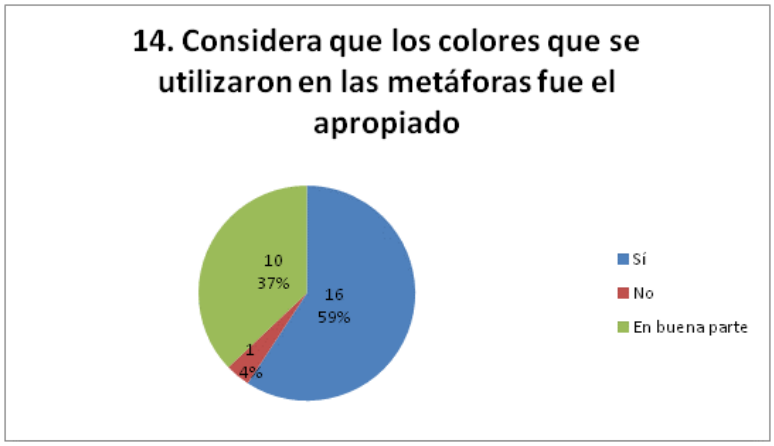

12. Considera que la velocidad de las animaciones fueron
6
25%
1
4%
14
58%
3
13%
Óptimas
Muy rápidas
Muy lentas
Difusas (unas muy rápidas, otras óptimas y otras muy lentas)

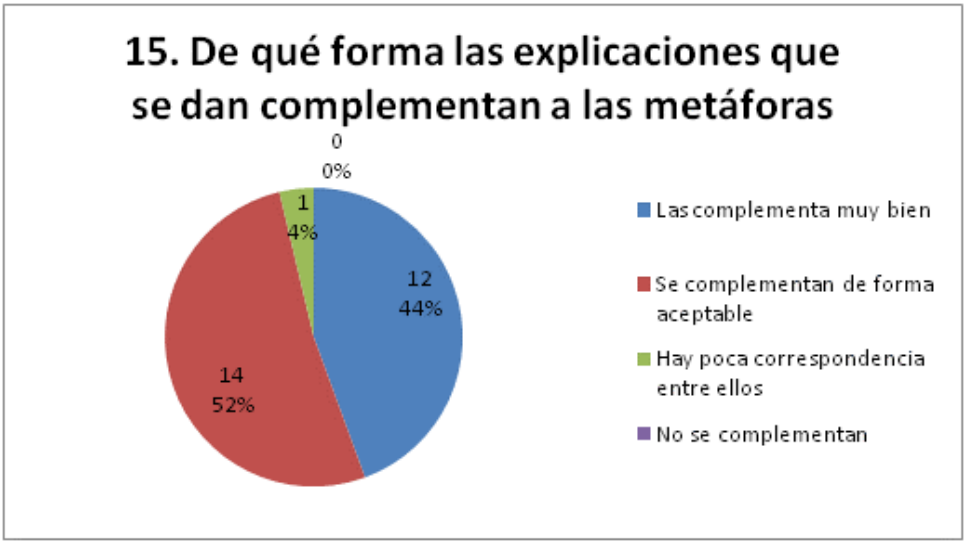
De esta parte se puede concluir que la herramienta es fácil de utilizar, se consiguió que fuese intuitiva y el uso de las representaciones visuales fue muy adecuado, pero algunos componentes fallaron en su función debido a que algunos de los encuestados se confundieron o se perdían con ellos. El caso concreto es el de la los botones del menú de Inicio los cuales no funcionaban causando la confusión. Pero con este caso, se comprobó que la herramienta fue intuitiva para los estudiantes ya que inmediatamente se fijaron en otra opción para seguir interactuando con la aplicación y encontraron el panel de navegación como salida.

Otra cosa a rescatar fue el uso de las letras ya que a pesar de que su tipografía era buena, el tamaño era difuso, es decir, a veces un tamaño normal y otras veces, muy grandes.

Parte 2: Teoría







En esta parte, el uso de las representaciones visuales tuvo un resultado favorable ya que, a consideración de la mayoría de personas que formaron la muestra, el uso de todos sus elementos aportó para que, asociándolos a lo que se hacía en cada sentencia, se pudiera comprender de mejor forma lo que se realizaba.

Parte 3: Complemento

En esta parte se estudió las opiniones de los encuestados para saber qué otras cosas, aparte de las que se evaluaron, fueron de su agrado, qué otras hay que mejorar además de dejar unas sugerencias.

En cuanto las fortalezas de VMCreative, los encuestados opinaron que el manejo de las ideas, el orden en el contenido y el diseño fueron las cosas que más gustaron de la herramienta. Incluso los profesores que tomaron la prueba, opinaron que el uso de las metáforas ayuda a comprender el tema y que la forma que se abordaron los ejemplos fue la apropiada ya que fueron concretos y se hizo uso del lenguaje C++. Algunas de las opiniones dadas fueron:

- "Buen contenido, con ideas claras."
- "Completo en los temas. Innovador."
- "La forma de hacer prueba de escritorio."
- "Las explicaciones nos sirven para estudiar una y otra vez hasta que nos quede claro."

Según los profesores:

- "La metáfora gráfica ayuda a comprender el tema."
- "Una guía paso a paso, fácil de usar, ejemplos concretos y el uso del lenguaje C++."

En cuanto a las debilidades de VMCreative, se señaló principalmente la poca cantidad de ejemplos y ejercicios para poder hacer una mejor retroalimentación de la teoría dada y que a veces la navegación era un poco confusa. Además, también se señaló la posibilidad de incluir ejercicios en donde se incluyeran niveles de dificultad o también la posibilidad de que los ejercicios fueran hechos con datos personalizados. Algunas de las opiniones fueron:

- “Deberían haber más ejemplos por tema.”

- “Claridad en el uso de botones.”

Según los profesores:

- “La navegación a veces no es clara.”

- “Faltan más actividades extras para interactuar con el estudiante y el aplicativo en cada tema.”

En cuanto a las sugerencias, se recomendó hacer un uso más parejo de la letra ya que a veces era muy grande, la velocidad de las animaciones que en ocasiones era muy rápida y la ausencia de más ejemplos. Algunas de las opiniones fueron:

- “La animación de las letras puede molestar algunas veces.”

- “En la parte inicial, enfatizar más ya que al hacer clic en los botones de la pantalla no se efectúa nada. Aclarar que es al lado izquierdo.”

Según los profesores:

- "Unificar la salida de menús, dejar sólo interacción gráfica sin el menú del lado izquierdo de la pantalla"

- "Una gráfica más estructurada de la caja de huevos, un manual de uso para el profesor, enfoque pedagógico."

Para ver los resultados con más detalle, se puede ver el anexo, *Tabulación resultados pruebas*.

7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 **Conclusiones**

Luego de haber hecho el desarrollo del prototipo de la aplicación VMCreative, se llegó a las siguientes conclusiones:

- Al concluir este trabajo, se realizó completamente la herramienta de software planteada logrando incluir todos los temas y fases de desarrollo que inicialmente se habían propuesto.

- Para que la realización de los temas y fases del proyecto fuera completa, fueron factores claves la selección de la metáfora y representaciones visuales y la metodología seguida.

- La realización conjunta de una investigación, entrevistas y caracterización de la herramienta facilitaron mucho el diseño y desarrollo de la aplicación ya que permitió tener una visión clara del alcance que ésta debería tener y de las expectativas de los usuarios finales.

- Durante el desarrollo del prototipo funcional, se pudo comprobar que aunque el hecho de combinar dos o más metodologías puede volver el desarrollo de una aplicación más compleja, pero es posible sacar provecho a cada una de ellas si se conocen bien las fortalezas y debilidades de cada una de ellas reconociendo que una puede ser el complemento de la otra.

- Asimismo, en la etapa de desarrollo, contar con una herramienta como DJProject facilitó mucho el desarrollo completo del prototipo de la aplicación VMCreative porque no es tan fácil conseguir que una herramienta pueda incluir de manera integrada los componentes claves que requirió el proyecto, especialmente los asociados con datos multimedia..

- El resultado de las pruebas de usabilidad de la herramienta mostró que se cumplieron las expectativas puestas sobre el prototipo de la aplicación VMCreative llegando incluso a ser propuesta como una opción rutinaria de apoyo a la cátedra tanto para profesores como para estudiantes.

- Cognoscitivamente, aunque hizo falta más variedad de ejemplos, la herramienta sí cumplió con las expectativas puestas en ella permitiendo al estu-

diante aclarar conceptos emitidos en la cátedra por medio asociaciones con la metáfora representativa en cada caso que se emitía.

- Por medio del prototipo de la herramienta VMCreative, se cumplió con la posibilidad de explicar contenidos de un tema abarcando más estilos de aprendizaje, lo cual permite dar alternativas para que un estudiante utilice más sus sentidos como una forma de apoyarse a favor de tener un aprendizaje más significativo.

7.2 **Trabajos Futuros**

Durante las fases de realización de la herramienta y en las pruebas, salieron propuestas interesantes que pueden ayudar a hacer de VMCreative una herramienta más completa. Algunas de ellas son:

- Tener la posibilidad que el programa tenga una opción que le permita elegir al usuario cuál representación visual quiere utilizar. Esto se debió a que cualquiera de las representaciones visuales evaluadas podría haber sido utilizada para representar los contenidos incluidos en la aplicación y de hecho, podría haber dado más variedad en el uso de colores y organización de espacios para colocar los componentes. Esto se puede lograr directamente sobre las animaciones hechas en Adobe Flash CS3.

- Poder realizar un compilador que permita al usuario insertar código directamente y que al momento de ejecutarse se pueda mostrar en un panel el resultado de la aplicación, y en otro panel, cómo funcionaría la representación visual.

- Permitir la oportunidad de incluir elementos multimedia de más complejidad que podrían aclarar conceptos. Por ejemplo, la inclusión de video o sonido. Esto se puede conseguir incluyendo la funcionalidad sobre la librería *NativeSwing* de DJProject.

- Dar la posibilidad al usuario de elegir en qué lenguaje quiere que los ejemplos sean representados. Esto podría ayudar a extender el uso de VMCreative a otras cátedras y posiblemente, hacer que funcione con otros paradigmas de programación, como por ejemplo, la orientada a objetos.

- Incluir ejercicios con niveles de dificultad.

8. REFERENCIAS Y BIBLIOGRAFÍA

8.1 **Referencias**

[1] Barros Barrios, R., Rojas Moreno, J., Sánchez, L.; *Diseño de instrumentos didácticos para aprendizaje activo basado en teoría de colores*; XXVII Reunión Nacional y VI Encuentro Iberoamericano de Instituciones de enseñanza de Ingeniería, octubre de 2007, Cartagena.

[2] Bruegge, B., Dutoit; *Object-oriented Software engineering;* Carnegie Mellon University; 1999; Pittsburgh, USA.

[3] Cognetics Corporation; *The LUCID design framework (Logical User Centered Interaction Design)*; Último acceso: Septiembre 2009; Disponible: http://www.cognetics.com

[4] Crook, Ch.; *Ordenadores y aprendizaje colaborativo*; Ediciones Morata; 1998; Madrid.

[5] Galvis Panqueva, A.; *Educación para el siglo XXI apoyada en ambientes interactivos, lúdicos, creativos y colaborativos, Universidad de los Andes;* 1998; Bogotá.

[6] Galvis Panqueva, A.; *Ingeniería de Software educativo*; Universidad de los Andes; 1992; Bogotá.

[7] Hundhausen, C., Lee Brown, J.; *An experimental study of the impact of visual semantic feedback on novice programming, Washington State University;* School of Electrical Engineering and Computer Science; Septiembre de 2006.

[8] Marrero Díaz, M.; *Estilos de aprendizaje y su impacto en el proceso enseñanza-aprendizaje en el curso TEOC 2007Aplicación de Terapia Ocupacional en disfunción*; Departamento de Terapia Ocupacional; 2007.

[9] Oviedo Galdeano, M., Ortíz Uribe, F.; *La enseñanza de la programación*; UPIICSA; Febrero 2002; México D.F.

[10] Grupo de investigación *ISTAR*; Proyecto Software Tangible; Pontificia Universidad Javeriana; Último acceso: Octubre 2009; Disponible: http://sophia.javeriana.edu.co/~lcdiaz/SoftwareTangible.html

[11] Rubio Sánchez, C.; *Prototipo de una herramienta didáctica para apoyar el proceso de enseñanza-aprendizaje de conceptos fundamentales de*

programación orientada a objetos; Pontificia Universidad Javeriana; Junio 2008; Bogotá.

[12] Suárez Valencia, F., García, R., López Trujillo, F.; *Modelo de Enseñanza de la Programación basado en MICEA y CUPI2*; Universidad Cooperativa de Colombia; octubre 2006; Manizales.

[13] Gayo Abello, D., Fernández Cuervo, H.; *Enseñar Informática es como…;* Universidad de Oviedo.

[14] Carnegie Mellon University; *Alice: An educational software that teaches students computer programming in a 3D environment;* Último acceso: Octubre 2009; Disponible: http://www.alice.org/index.php

[15] MIT; *Scratch: imagina, programa, comparte.* Último acceso: Octubre 2009; Disponible: http://scratch.mit.edu/

[16] MacFarlane, K., Mynatt, B.; *A study of an advance organizer as a technique for teaching computer programming concepts*; ACM SIGCSE Bulletin 20(1): 240-243; 1988.

[17] Becerra Santamaría, C.; *Turbo C tutor versión 2.0*. Derechos reservados; 1990.

[18] Red Hat Middleware; *Relational Persistence for Java and .NET;* Último acceso: Octubre 2009; Disponible: http://www.hibernate.org/

[19] Couloris, G.,Dolimore J., Kindberg T.; *Sistemas Distribuidos*; Tercera Edición; Addison Wesley; 2004.

[20] The PHP group*; PHP: Hypertext preprocessor* ; Disponible: http://www.php.net/

[21] IEEE computer society; *Software engineering body of knowledge (SWEBOK)*; 2004

[22] DUARTE, J.; *Ambientes de aprendizaje: Una aproximación conceptual*; En: Revista Iberoamericana de Educación.

[23] Burbano, A.; *Costos y presupuestos: Conceptos fundamentales para la gerencia*; Universidad de los Andes; 2006.

[24] Holmes, B., Tangney, B., Fitzgibbon, A., Savage, T., Mehan, S.; *Communal Constructivism: Students constructing learning for as well as with others*; Center for research in IT in Education; Trinity College; Dublin Ireland, 2001.

[25] Cañas, A.J., Novak, J.D., González, F.M.; *La teoría del aprendizaje significativo*, Centro de Educación a Distancia (C.E.A.D.); Pamplona, España, 2004

[26] Salcedo Lagos, P.; *Ingeniería de software educativo, teorías y metodologías que la sustentan;* Revista Informática Universidad de Concepción, 2002. Disponible: http://www.inf.udec.cl/revista/edicion6/psalcedo.htm

[27] Galvis Panqueva, A.; *Ambientes de enseñanza-aprendizaje enriquecidos con computador;* Universidad de los Andes; 2006; Bogotá.

[28] Villalobos, J., Casallas, R., Hernández, M., Vela, M., Jiménez, C.; *Cupi2: Buscando nuevas maneras de aprender a programar* ; Último acceso: Octubre 2009; Disponible: http://cupi2.uniandes.edu.co/inicio.php

[29] The Phrogram Company; *Phrogram: real programming, really fun;* Último acceso: Octubre 2009; Disponible: http://phrogram.com/

[30] *The Processing Project*. Disponible: http://processing.org/

[31] Qué es Logo; Último acceso: Octubre 2009; Disponible: http://neoparaiso.com/logo/lista-logo.html

[32] National Visualization and Analytics Center, *Visual representations and interaction technologies*; Último acceso: Noviembre 2009; Disponible: http://nvac.pnl.gov/

[33] Reiss, SP.; *Visual representations of executing programs*; Journal of Visual Languages and Computing; Último acceso: Noviembre 2007; Disponible: www.sciencedirect.com

[34] Norman, DA.; *Things that make us smart: Defending human attibutes in the age of the machine*; Perseus Books; 1993; New York.

[35] TVersky, B., Morrison, JB., Betrancourt, M.; *Animation: ¿Can it facilitate?*; Tomada de International Journal of human-computer studies páginas 247-262; Academic press Inc.; 2002; Duluth, Minnesota

[36] Programa Cátedra Programación de Computadores carrera de Ingeniería de Sistemas Universidad Nacional de Colombia; Último acceso: Noviembre 2009; Disponible en: http://www.unal.edu.co/dis/informacion/programas/pensum/SemestreI/programacion1.html

[37] Programa Cátedra Algorítmica y programación carrera de Ingeniería de Sistemas Universidad Nacional de Colombia; Último acceso: Noviembre 2009; Disponible en: http://cupi2.uniandes.edu.co/cursos/apo1/docs/2009-2-APO1-Programa.pdf

[38] Sun Microsystems; *"Java SE Application Design With MVC"*; Último acceso: Noviembre 2009; Disponible: http://java.sun.com/developer/technicalArticles/javase/mvc/

[39] Adobe. "Adobe Flash Player" Disponible en: http://www.adobe.com/es/products/flashplayer/

[40] Deckers, C.; *The DJProject; "Rediscover the Desktop"*; Último acceso: 2009; Disponible en: http://djproject.sourceforge.net/main/index.html

[41] *NetBeans IDE 6.7.1*; Disponible en: http://netbeans.org/

[42] Adobe; *Adobe Flash CS3*; Disponible en: http://www.adobe.com/es/products/flash/

[43] Adobe; *Adobe Flash Player 10*; Disponible en: http://www.adobe.com/es/products/flashplayer/

[44] CollabNet; *Tortoise SVN*; Disponible en: http://tortoisesvn.tigris.org/

[45] Google; *GoogleCode project hosting*; Disponible en: http://code.google.com/hosting/

[46] Diccionario de la Real Academia de la Lengua Española. Último acceso: Diciembre 2009. Disponible en: http://www.rae.es/rae.html

[47] Botero, LF., Ronderos MC.; *TAVA: Herramienta didáctica para apoyar el proceso de enseñanza-aprendizaje en los conceptos de variables, tipos de datos y apuntadores;* Pontificia Universidad Javeriana; Junio 2009; Bogotá.

[48] Pérez, LP., Vásquez, JH.; *PsiCoder: Software Educativo para Facilitar el Proceso de Enseñanza – Aprendizaje en un Curso Básico de Programación*; Pontificia Universidad Javeriana; Diciembre 2008; Bogotá.

[49] Cabero, J.; *Tecnología educativa: Diseño y utilización de medios en la enseñanza*; Ediciones Paidós Ibérica S.A.; 2001; Barcelona, España

[50] Rodríguez J., Pardo A.; *Didáctica General Bloque III Los Medios y Recursos Didácticos*; Departamento de Educación. Universidad de Huelva; España

[51] Marqués Graells, P; *El software educativo.* Obtenido de Departamento de Pedagogía Aplicada; 2003; Universidad Autónoma de Barcelona; Disponible: http://www.lmi.ub.es/te/any96/marques_software/#capitol2

9. ANEXOS

Anexo 1. Glosario

Ambiente de aprendizaje: dinámicas que constituyen los procesos educativos y que involucran acciones, experiencias y vivencias de cada uno de los participantes; actitudes, condiciones materiales y socioafectivas, múltiples relaciones con el entorno y la infraestructura necesaria para la concreción de los propósitos culturales que se hacen explícitos en toda propuesta educativa" [22]

Didáctica: Es el arte de enseñar en el cual el docente describe, explica y fundamenta los métodos más adecuados y eficaces para que el estudiante adquiera los conocimientos necesarios.[11]

Estilo de aprendizaje: "Forma como el estudiante se apropia del conocimiento, es decir, la estrategia de representación que utiliza en su interacción con el mundo" [1]. Dentro de estos estilos se destacan los estilos visual, auditivo, kinestésico, relacional y lógico-matemático.

Hibernate: Es un servicio de persistencia y consulta aplicado para Java y .NET. Este servicio se caracteriza porque permite desarrollar clases persistentes basado en una notación orientada a objetos y expresar queries en una extensión SQL portable llamada HQL [18].

LUCID: Logical User Centered Interaction Design o Diseño de Interacciones Lógicas Centradas en el Usuario. Es un framework para manejar el proceso de diseño de una interfaz de manera que se garantice o por lo menos se generar usabilidad en el software a implementar [3].

Metáfora: Es un mecanismo de comprensión de conceptos abstractos basado en la aplicación de una palabra u objeto como analogía para entender un concepto [13].

Micromundo: "Ambiente de trabajo reducido donde suceden o pueden suceder cosas relevantes respecto a los que interesa aprender, dependiendo de lo que el usuario realice. Suele incluir una situación y formas de incidir sobre ella" [6].

Modelo de ciclo de vida de software: Es la representación de todas las actividades y productos necesarios para desarrollar un software. Su objetivo consiste en que las personas que dirijan un proyecto de software traten de tener más control sobre la complejidad del proceso de desarrollo de software en todos los pasos que lleve el proceso hasta crear la herramienta requerida [2].

Modelo de pensamiento: Forma de pensamiento que posee una persona que se basa en las preferencias, tendencias, disposiciones y patrones de conducta al momento de aprender y realizar un trabajo. Los modelos se dividen en cuatro colores: azul, verde, amarillo y rojo [1].

Paradigma de programación: Enfoque o perspectiva sobre el cual se desarrolla una aplicación.

PHP: Lenguaje de programación de propósito general utilizado para desarrollo de páginas Web [20].

Prototipo funcional: Un prototipo funcional es una representación operativa o funcional de un sistema el cual ofrece una respuesta a las entradas que le proporciona el usuario [11].

Prueba de usabilidad: Prueba donde se evalúa que tan fácil es para los usuarios finales el uso y aprendizaje del software y la documentación relacionada [21].

SAD: Documento de descripción de arquitectura.

Software Tangible: "El proyecto Software Tangible se centra en el estudio, identificación y propuesta de un conjunto altamente relacionado de metáforas, abstracciones visuales y modelos, relacionados con los principales conceptos de programación, ejecución de algoritmos y construcción de software en general, que constituyan la base fundamental y faciliten el posterior desarrollo e integración de material didáctico, en especial micromundos y ambientes virtuales, que apoyen el proceso de enseñanza aprendizaje en estos temas" [10].

SRS: Documento de especificación de requerimientos.