UNIVERSIDAD AUTONOMA METROPOLITANA

Casa abierta al tiempo Azcapotzalco

*Universidad Autónoma Metropolitana*
*Unidad Azcapotzalco*

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto Terminal:

# *EDAPIX*

*Una aplicación gráfica para asistir al proceso de enseñanza - aprendizaje de las Estructuras de Datos*


Alumno:
Torres Moro Abraham
204205638

Asesora del proyecto:
Dra. Beatriz Adriana González Beltrán
30246


**[EDAPIX ...]**

... proporciona al usuario una amplia interacción con las estructuras de datos incorporando características como: visualización de animaciones y de la traza de los algoritmos, modificación de operaciones en las estructuras de datos, comparación de la eficiencia entre dos estructuras y capacidad de almacenar y recuperar el estado de una estructura de datos.

## Contenido

## Resumen

El estudio de las estructuras de datos es pieza clave en los planes de estudios de licenciaturas en el área de la computación y se vuelve primordial que los estudiantes comprendan ampliamente los principios teóricos y prácticos de las estructuras de datos para que puedan aplicarlas adecuadamente en el diseño de algoritmos computacionales.

Con el fin de asistir el proceso de enseñanza-aprendizaje de este tema tan importante se pueden utilizar herramientas tecnológicas basadas en la visualización de animaciones que ilustren de manera gráfica el comportamiento dinámico de las estructuras de datos.

El objetivo de este proyecto terminal es la realización de una aplicación gráfica para asistir al proceso de enseñanza-aprendizaje de las estructuras de datos. Esta aplicación permite al usuario interactuar directamente con las estructuras de datos incorporando las siguientes características:

- Manipulación gráfica de las estructuras de datos.
- Visualización de animaciones y de la traza de los algoritmos asociados con las estructuras de datos.
- Modificación de algunas operaciones dentro de los algoritmos de cada estructura de datos.
- Comparación de la eficiencia entre dos estructuras de datos al realizar inserciones y extracciones de datos.
- Capacidad de almacenar y recuperar el estado de una estructura de datos previamente definida.

## Introducción

El estudio de las estructuras de datos es primordial en los planes de estudios de licenciaturas en el área de la computación. La aplicación de las estructuras de datos representa la base para realizar cualquier tipo de software y por ello resulta fundamental que los estudiantes de estas licenciaturas asimilen desde el inicio de su formación, los principios teóricos y prácticos de las estructuras de datos de modo que las apliquen de manera efectiva y eficiente en el diseño de algoritmos computacionales.

Durante este proceso de enseñanza-aprendizaje se pueden utilizar herramientas tecnológicas basadas en la visualización de animaciones, para ilustrar de manera gráfica el comportamiento dinámico de las estructuras de datos. En este sentido, existe una gran cantidad de animaciones que han sido desarrolladas utilizando los applets de Java [6] y que se pueden encontrar en Internet ([1], [2], [3], [4], [5]).

Cabe mencionar que existe un sitio Web titulado *Estructuras de datos y Algoritmos usando Programación Orientada a Objetos* [7] que contiene una colección importante de applets que ilustran el funcionamiento de algunas de las estructuras de datos más representativas.

Otro trabajo que permite la enseñanza de las estructuras de datos es la aplicación "EDApplets[1]: Una Herramienta Web para la Enseñanza de Estructuras de datos y Técnicas Algorítmicas". EDApplets ([8], [9]) está orientada a la animación y visualización mediante trazas de algoritmos y estructuras de datos. Esta herramienta también utiliza los applets de Java.

La aplicación DSTool[2] [10] es una biblioteca de clases que también implementa diversas estructuras de datos. Estas clases proporcionan la opción de depurar gráficamente las estructuras de datos mediante la aparición, en tiempo de ejecución, de una ventana que muestra una representación gráfica actualizada de la estructura de datos. DSTool es una aplicación que se ejecuta de manera local.

En este proyecto terminal se conservan las siguientes características que poseen las aplicaciones DSTool y EDApplets:

- Orientación a la animación y visualización de las estructuras de datos y sus algoritmos asociados.
- Integración de los algoritmos y las estructuras de datos en un único entorno.
- Módulo para comparar la eficiencia de las estructuras de datos.
- Portabilidad (multiplataforma).

Sin embargo, DSTool y EDApplets carecen de la manipulación gráfica de las estructuras de datos, lo que limita en gran medida la interacción del usuario con estos sistemas. Tampoco exploran la posibilidad de mostrar al usuario mediante ejemplos y animaciones lo que puede ocasionar la modificación de las operaciones de una estructura de datos. No incluyen la característica de almacenamiento y recuperación del trabajo; esto representa un problema cuando se trata de una estructura de datos que presentó diversas modificaciones mientras se trabajaba con ella o cuando se invirtió demasiado tiempo y esfuerzo en definir su estado actual. El hecho de que el usuario no pueda guardar su trabajo con el fin de continuarlo en otro momento puede generar rechazo o frustración hacia la aplicación. En este proyecto terminal se proponen soluciones para manipular de manera gráfica las estructuras de datos y almacenar y recuperar las mismas.

[1] EDApplets fue desarrollada en 2003 por Almeida F., Blanco V. y Moreno L. M., miembros del Departamento de Estadística, Investigación de Operativa y Computación de la Universidad de La Laguna, España.

[2] En el desarrollo de DSTool participaron María del Puerto Paule Ruiz, Juan Ramón Pérez Pérez, Sergio Sama Villanueva, Bernardo Martín González Rodríguez, miembros del Departamento de Informática, de la Universidad de Oviedo, España.

### *Objetivos*

El objetivo general de EDAPIX es apoyar la enseñanza y el aprendizaje de las estructuras de datos, mediante la representación y la manipulación de las mismas en un entorno gráfico.

EDAPIX tiene como objetivos particulares:

- Manipular de manera gráfica una estructura de datos utilizando un área de dibujo.
- Modificar las operaciones de las estructuras de datos para experimentar los efectos de implementarlas de manera diferente
- Guardar y recuperar el estado de las estructuras de datos a petición del usuario.
- Combinar en un mismo entorno las principales ventajas que poseen entre sí las aplicaciones DSTool y EDApplets.

## Desarrollo

A continuación se describirá la arquitectura de EDAPIX y el proceso general de desarrollo de software utilizado en su construcción.

Se eligió al Proceso Unificado (UP) como proceso de desarrollo de software ya que éste combina las prácticas comúnmente aceptadas como "buenas prácticas", tales como ciclo de vida iterativo y desarrollo dirigido por la arquitectura. El UP describe actividades de trabajo en **disciplinas**. A su vez, una disciplina es un conjunto de actividades (y artefactos relacionados) en un área determinada, como las actividades en el análisis de requisitos. En el UP, un artefacto es el término general para cualquier producto del trabajo: código, gráficos Web, esquema de base de datos, documentos de texto, diagramas, modelos, etc.

Es importante señalar que en el desarrollo de este proyecto terminal (aplicación EDAPIX) se aplicaron las disciplinas de análisis, diseño e implementación del UP.

## *Arquitectura de EDAPIX*

La parte central de EDAPIX es la representación y manipulación gráfica de las estructuras de datos, las cuales se basan en la representación lógica de las mismas. La operación de esta parte de EDAPIX se puede expresar por medio de cuatro módulos principales (ver figura 1).

La representación lógica de una estructura de datos se puede ver de forma estática y dinámica. La forma estática se refiere a los elementos que permiten el funcionamiento de la estructura de datos, es decir, los datos y la definición de las operaciones permitidas sobre los datos; a tal definición se le denomina implementación de la estructura de datos. Por otro lado, la forma dinámica se refiere a la ejecución de la implementación de la estructura de datos, dicha ejecución dará como resultado la representación de los datos en la memoria principal, los cuales serán modificados (en tiempo de ejecución) por las operaciones definidas en la implementación.

La representación gráfica de una estructura de datos debe estar directamente relacionada con su representación lógica, de otro modo, carecería de sentido la representación gráfica de una estructura que no simbolice su funcionamiento interno. Dicha relación la proporciona el módulo intérprete, el cual, se basa en un algoritmo que recibe de entrada la operación aplicada en la estructura de datos, evalúa el efecto que tuvo tal operación y determina la representación gráfica correspondiente (si fuera el caso).


Figura 1. Bloques principales para la representación y manipulación gráfica en EDAPIX

El motor gráfico es el módulo encargado de generar la representación gráfica de la estructura de datos. En base a la salida del módulo intérprete, el módulo motor gráfico muestra los elementos gráficos, que ilustran la estructura de manera dinámica. Los elementos gráficos son los nodos de las estructuras y su relación con otros nodos. De manera interna, un nodo contiene el dato y una o más ligas que lo relacionan con otros nodos. Gráficamente, el nodo se representa como una pequeña caja u óvalo con la información que representa al dato. Las ligas se representan como líneas o flechas.

Dependiendo de las estructuras, los nodos y las ligas tienen formas y colocaciones diferentes. El módulo motor gráfico también es responsable de determinar tales características. La representación gráfica es enriquecida con animaciones con el fin de atraer el interés del usuario (ver figura 2).

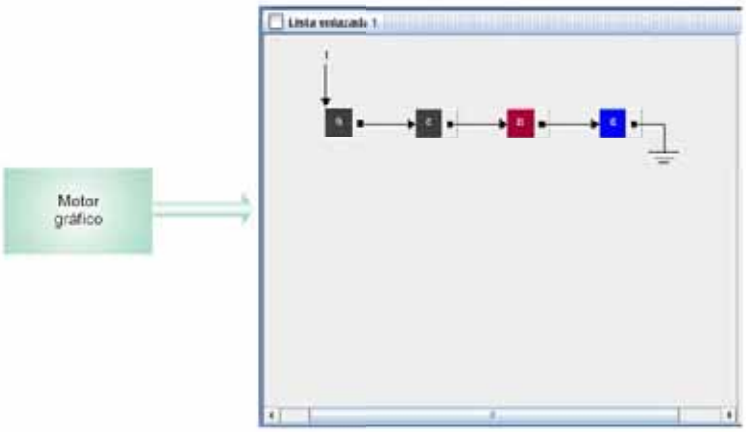


Figura 2. El motor gráfico y la representación gráfica de las estructuras.

Si un usuario manipula gráficamente una estructura, la acción se debe reflejar en su representación lógica. Por ello, nuevamente se requiere la intervención del módulo intérprete para determinar el efecto que tiene la manipulación gráfica de una estructura de datos sobre su representación lógica (ver figura 3).

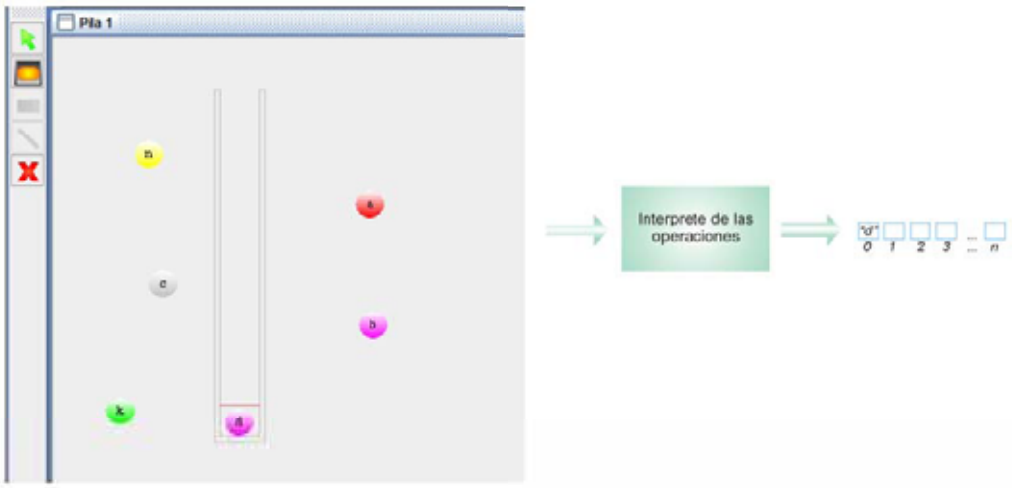


Figura 3. Manipulación gráfica de una pila la interpretación de su estado para su representación lógica contenida en un arreglo.

La figura 4 muestra la arquitectura global de EDAPIX.

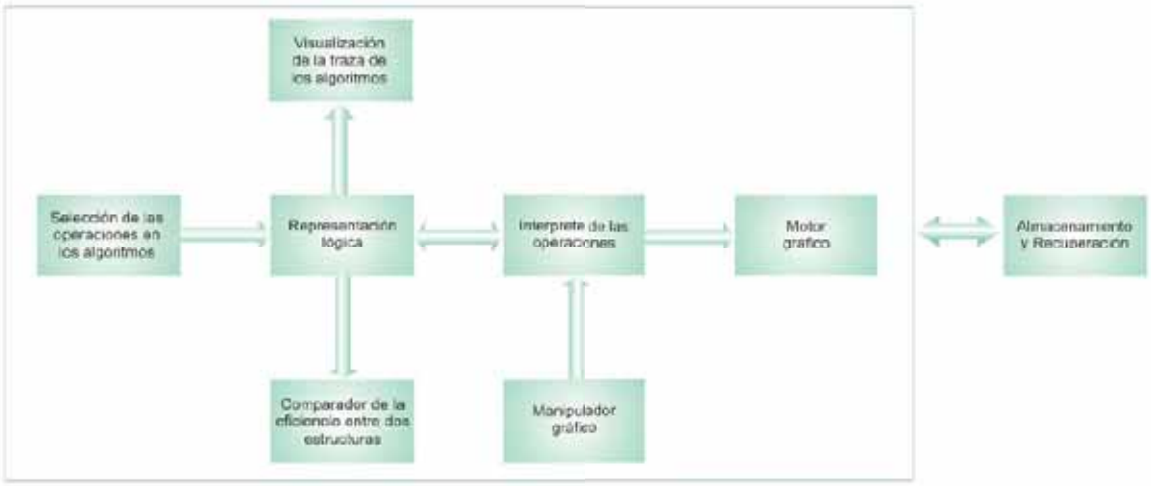


Figura 4. Arquitectura de EDAPIX

EDAPIX ejecuta el módulo de visualización de la traza de algoritmos (ver figura 4). Esta actividad permite visualizar la traza de los algoritmos de acuerdo con su representación lógica. Así, en la visualización se resalta la línea del algoritmo que se está procesando en cada instante y paralelamente se muestra la evolución de la representación gráfica que se genera por medio del intérprete y el motor gráfico (ver figura 5).

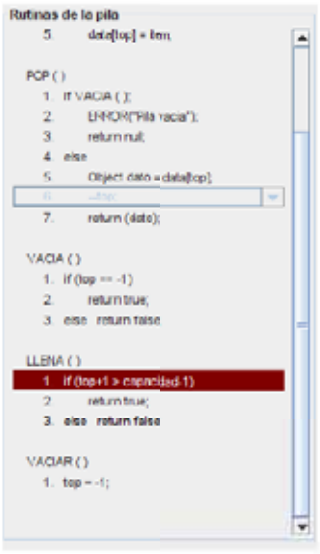


Figura 5. Ejemplo de la visualización de la traza (en un instante de tiempo) de los algoritmos asociados a la estructura pila, al hacer una extracción.

EDAPIX utiliza el módulo selección de operaciones (ver figura 4) que se comunica con la representación lógica para que las modificaciones en la implementación (introducidas intencionalmente) se traduzcan en la representación gráfica a través del módulo intérprete y del motor gráfico. Esto permite seleccionar y modificar algunas operaciones en los algoritmos de las estructuras de datos (ver figura 6).

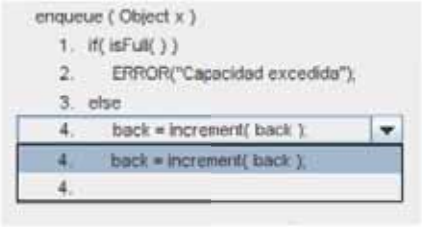


Figura 6. Modificación de las operaciones mediante cajas de selección colocadas entre las líneas de los algoritmos

El módulo que se encarga de la comparación de la eficiencia entre estructuras de datos se relaciona directamente con la representación lógica porque ejecuta los algoritmos correspondientes a las dos estructuras de datos que se estén comparando (ver figura 4).

El módulo responsable de guardar el estado de las estructuras de datos que el usuario haya definido se relaciona directamente con los demás módulos, ya que cada uno de éstos representa información importante sobre el estado actual del trabajo con una estructura de datos (ver figura 4).

## *Análisis y diseño de EDAPIX*

La figura 7 presenta el diagrama de casos de usos para EDAPIX. Cada uno de los casos uso representados por óvalos son los objetivos de alto nivel para el usuario. En este diagrama podemos observar que el único caso de uso que se especializa en las estructuras de datos disponibles en EDAPIX es la "creación de una estructura de datos". Debido a que la recomendación es definir diagramas de casos con el menor grado de complejidad posible se asume que al estar en otro caso posterior a la creación este se referirá a la estructura de datos actual.

También podemos apreciar relaciones *include* utilizadas para especificar que un caso de uso incluye a otro, por ejemplo, el caso de uso "abrir" requiere incluir el caso de uso "crear" ya que durante el proceso de recuperación de una estructura de datos se debe crear una estructura de datos vacía que posteriormente adquiera las características de la estructura de datos almacenada.

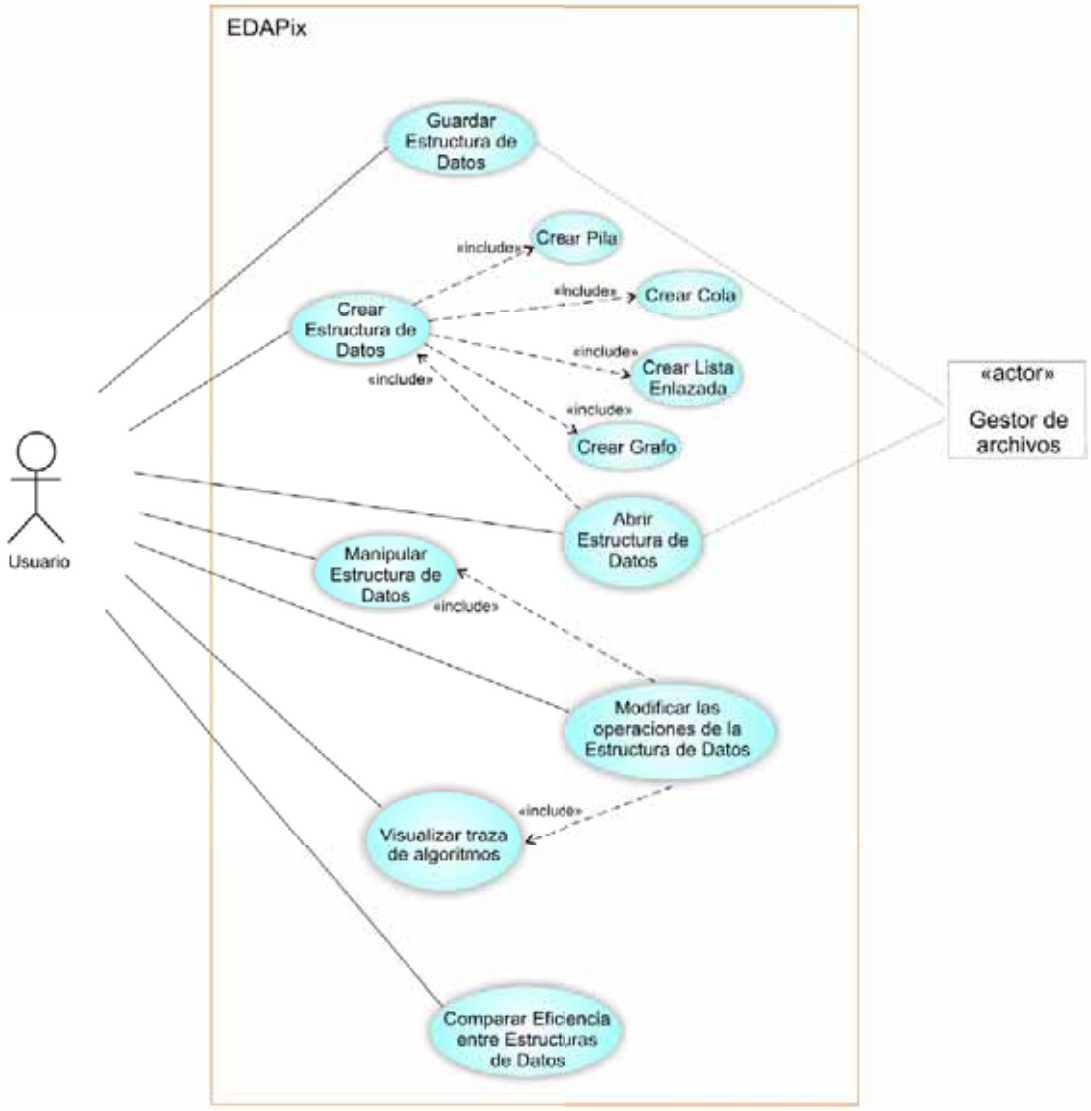


Figura 7. Diagrama de casos de uso para EDAPIX

A continuación se presenta un caso de uso de texto elaborado durante el análisis del sistema.

## Caso de uso UC3: Crear una Estructura de Datos Pila (EDP).

**Actor principal:** Usuario (profesor o alumno).
**Personal involucrado e intereses:**

- Usuario: crear una EDP con la finalidad de llevar a cabo de manera gráfica las operaciones relacionadas con esta estructura, observar su comportamiento dinámico y asimilar las ventajas o desventajas de su aplicación en la resolución de problemas específicos.

**Precondiciones:** Puesta en marcha del sistema.

**Garantías de éxito:** tras ordenar la creación de una EDP, se mostrará en pantalla la representación gráfica de una EDP vacía así como los controles necesarios para poder aplicar las operaciones de una EDP.

**Escenario principal de éxito:**

1. El usuario selecciona la opción "crear" y posteriormente la opción "pila"
2. El sistema le solicita al usuario que especifique la capacidad requerida para la EDP.
3. El usuario determina la capacidad para la EDP y acepta la creación de la estructura.
4. El sistema muestra la representación grafica de una EDP vacía y habilita los controles necesarios para manipular la estructura y poder trabajar con ella.

**Extensiones:**

3a. El usuario acepta la creación de la EDP sin especificar su capacidad.
   1. El sistema le asigna una capacidad predeterminada y prosigue con la creación de la estructura.

3b El usuario proporciona una entrada inválida para la capacidad.
   1. El sistema no permite la creación de la estructura hasta que el usuario proporcione una entrada válida para la capacidad.

3c. El usuario cancela la creación de la estructura.
   1. El sistema regresa al estado anterior en el que se solicito la creación de una EDP.

Como se puede apreciar, este caso de uso se refiere a la creación de una estructura de datos pila. El actor principal es el que recurre a los servicios del sistema para cumplir un objetivo. La sección *personal involucrado e intereses"* detalla los objetivos de usuario. En la caso de EDAPIX sólo está definido hasta el momento un solo rol de usuario que al mismo tiempo representa al *profesor* y al *alumno* como únicos usuarios del sistema.

Las precondiciones establecen lo que siempre debe cumplirse antes de comenzar un escenario en el caso de uso. En el ejemplo se toma como precondición la *puesta en marcha del sistema* y se asume que esto siempre es verdad antes de crear una estructura de datos.

Las garantías de éxito establecen qué debe cumplirse cuando el caso de uso se completa con éxito. En el caso de uso "crear una EDP" la garantía de éxito satisface las necesidades de todo el personal involucrado.

En el escenario principal de éxito se describe el camino de éxito típico que satisface los intereses del personal involucrado. Las extensiones indican todos los otros escenarios o bifurcaciones, tanto de éxito como de fracaso.

## Diagramas de secuencia del sistema

Un diagrama de secuencia del sistema (DSS) es un dibujo que muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y eventos entre los sistemas. En este caso tomaremos como ejemplo el DSS utilizado para representar el escenario de la manipulación grafica de una pila al realizar una operación de inserción (ver figura 8).

En este DSS es posible apreciar el orden básico en el que se da el escenario de la manipulación grafica de una pila al realizar una operación push. Los mensajes con parámetros que se dirigen desde el usuario hacia el sistema son una abstracción que representa eventos entrada para el sistema. Los mensajes que se dirigen desde el sistema hacia el usuario y que se marcan con líneas discontinuas son valores de retorno asociados con algún mensaje anterior. Las cajas sirven para encerrar la iteraciones que pueden presentarse y entre corchetes se coloca la clausula de iteración.

## Modelo de dominio

Un modelo de dominio (ver figura 9) se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos software. El modelo de dominio que se presenta muestra las clases conceptuales significativas en el dominio del problema. En este modelo de dominio también se muestran las asociaciones entre las clases conceptuales y los atributos de algunas de ellas.

Cabe mencionar la existencia de clases sumamente significativas en un modelo de dominio; a estas clases se les denomina de especificación o descripción. En el modelo de dominio usado para EDAPIX existe la clase conceptual "Estructura de Datos" que es una clase de especificación ya que de esta se derivan las clases conceptuales encargadas de representar a cada estructura de datos.

Los documentos completos de análisis y diseño fueron entregados al asesor.

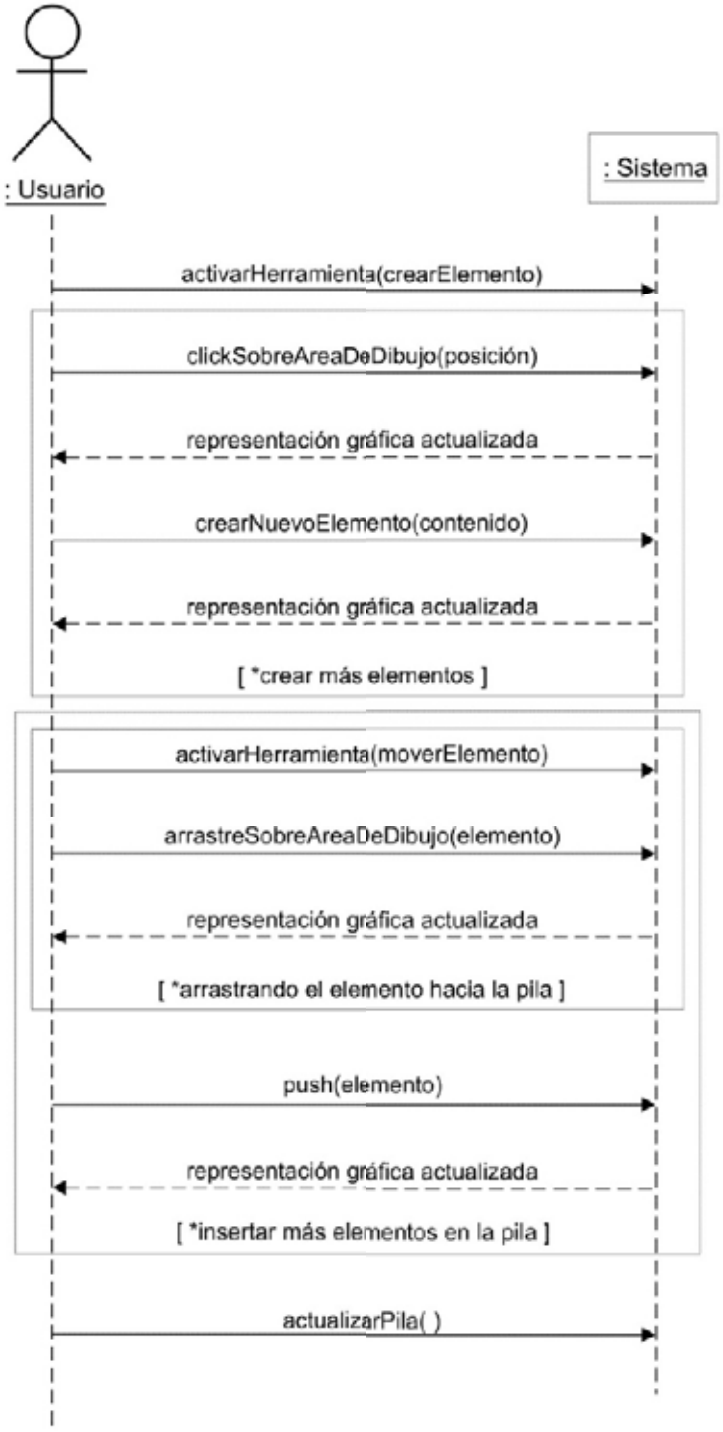


Figura 8. DSS para un escenario de *Manipulación gráfica con una operación push.*

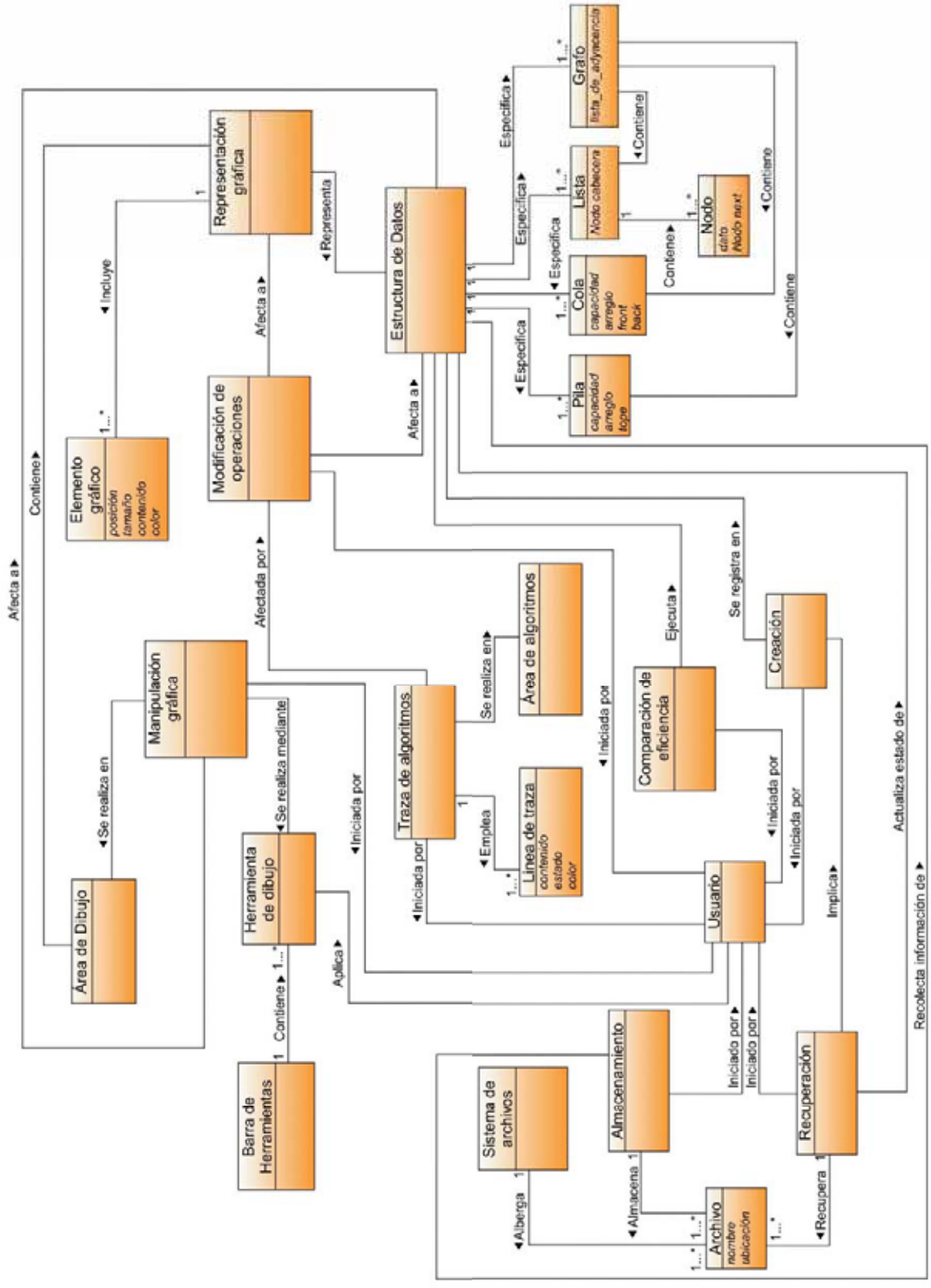


Figura 9. Modelo del dominio de EDAPIX

### Tecnología utilizada

Se usa Windows como plataforma de trabajo y el ambiente de desarrollo es Eclipse SDK 3.1.1 [11] para Windows. Sin embargo, esto no significa que la aplicación pierda la característica de ser multiplataforma. Por esta razón el lenguaje de programación a utilizar será Java [6].

En el desarrollo de la interfaz gráfica de usuario se usan los paquetes *java.awt* [14] y *javax.swing* [12]. El módulo responsable de la manipulación gráfica de las estructuras de datos se basará en la clase *Canvas* [13] que pertenece al paquete *java.awt*. *Canvas* representa un área de la pantalla rectangular y en blanco en la que se puede dibujar. Un objeto de la clase *Canvas* también puede recibir datos de entrada del usuario en forma de eventos producidos por el ratón o el teclado.

## Conclusión

El grado de manipulación gráfica sobre las estructuras de datos que se adquiere con EDAPIX es muy cercano al que se esperaba conseguir; esta característica puede mejorar si se agregan otras funciones como la edición de contenido de los elementos o el cambio de color y tamaño. Las aplicaciones de dibujo actuales hacen uso de otras herramientas para mejorar la usabilidad. Entre estas herramientas destaca el uso de menús contextuales sobre el área de dibujo, la selección de distintos elementos al mismo tiempo y la posibilidad de acercar o alejar el área de dibujo. Con el fin dotar a la aplicación de una mayor capacidad de dibujo de estructuras de datos, se puede explorar la inclusión de dichas herramientas.

Los resultados obtenidos en el desarrollo del modulo encargado del almacenamiento y la recuperación son muy significativos. Para simplificar el proceso de recuperación se uso XML. Esta característica enriquece las posibilidades del usuario que tiene la capacidad de guardar su trabajo en el momento que lo desee y recuperarlo cuando así lo necesite.

La modificación de operaciones se logró acoplar muy bien con la visualización de la traza de algoritmos mediante la inclusión de cajas de selección en el lugar correspondiente a la línea del algoritmo que se podía modificar.

Se entregó al asesor del proyecto un CD que contiene los manuales de usuario y del programador, así como el código fuente de toda la aplicación, una guía de instalación del sistema y los documentos de análisis, diseño e implementación, tal como se describe en la propuesta de este proyecto terminal.

Estos documentos forman parte de un proyecto de investigación en desarrollo y no pueden ser proporcionados a la comunidad.

**Referencias**


1. *AVL tree applet.*
http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm
Consultada el 12 de abril de 2009.

2. *Implementing Doubly-Linked Lists.*
http://remote.science.uva.nl/~heck/JAVAcourse/ch4/sss1_2_3.html
Consultada el 12 de abril de 2009.

3. *Queue Applet.*
http://courses.cs.vt.edu/csonline/DataStructures/Lessons/QueuesImplementationView/applet.html. Consultada el 12 de abril de 2009.

4. *A stack applet !!!*
http://acc6.its.brooklyn.cuny.edu/~cis22/animations/tsang/html/STACK/stack640.html.
Consultada el 12 de abril de 2009.

5. *Java Applet Demos of Dijkstra's Algorithm.* http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/DijkstraApp.shtml?demo1. Consultada el 12 de abril de 2009.

6. *Sun Microsystems Java Technology.* http://java.sun.com/. Consultada el 12 de abril de 2009.

7. *Estructuras de datos y Algoritmos usando Programación Orientada a Objetos.* http://personales.unican.es/corcuerp/EDAOO%5CEDAOO_index.html. Consultada el 13 de abril de 2009.

8. *EDApplest. Applets de estructuras de datos y algoritmos.*
http://www.pcg.ull.es/edapplets/. Consultada el 13 de abril de 2009.

9. Almeida F., Blanco V. y Moreno L. M. *EDApplets: Una Herramienta Web para la Enseñanza de Estructuras de datos y Técnicas Algorítmicas*
http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2004/ponencias/ponencia48.pdf.
Consultado el 13 de abril de 2009.

10. María del P. Paule R., Juan R. Pérez P., Sergio Sama V., Bernardo M. González R. *DSTool: Prototipo para la enseñanza, evaluación y depuración de estructuras de datos basado en mecanismos de reflexión estructural.* http://lsm.dei.uc.pt/ribie/docfiles/txt2003729183325paper-056.pdf. Consultado el 13 de abril de 2009.

11. Ambiente de desarrollo *Eclipse*. http://www.eclipse.org/. Consultada el 14 de abril de 2009.

12. *Package javax.swing.* http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html Consultada el 14 de abril de 2009

13. *Class Canvas.* http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Canvas.html Consultada el 14 de abril de 2009

14. *Package java.awt.* http://java.sun.com/j2se/1.4.2/docs/api/java/awt/package-summary.html. Consultada el 14 de abril de 2009.