**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

CENTRO DE SOFTWARE LIBRE

DEPARTAMENTO DE SISTEMAS OPERATIVOS


# MANEJADOR DE COPIAS PARA NOVA 4.0

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERÍA EN CIENCIAS INFORMÁTICAS

**Autor:**

Andy Michel Cabriales Moreno

**Tutores:**

Profesor: MsC. Allan Pierra Fuentes

Profesor: Ing. Daniel Hernández Bahr

**Ciudad de La Habana, Cuba**

**2013**

**Declaración de autoría**

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

_____ _____

Firma del autor Firma del tutor

**Resumen**

Se propone aumentar el control sobre las copias de archivos en el orden de pausar, reanudar y de esa manera establecer prioridades entre dichas operaciones en la distribución Nova GNU/Linux. Para lograr dichos objetivos se construye una herramienta que se integre al manejador de archivos Nautilus con la finalidad de suplir estas deficiencias.

Dicha propuesta está fundamentada en los artefactos generados en el desarrollo del software y en el estudio inicial de las principales variantes de solución tales como tecnologías y otras aplicaciones similares.

Palabras clave: copia de archivos, manejador de archivos, Nautilus, Nova GNU/Linux

# Índice

**Introducción**

Los usuarios que utilizan software privativo están atados a los designios del propietario mediante las licencias. En cambio, los usuarios que recurren al software libre tienen la posibilidad de obtener y modificar dicho software por medio del código fuente, además de poder redistribuir sus cambios siempre que mantengan la libertad de su aporte. Además de dichas características, desde hace algunas décadas, el mundo de la informática ha visto cómo gira hacia esta manera de producir software una cantidad creciente de personas.

Los sistemas operativos no han quedado al margen de éste movimiento y es palpable la fuerza que hoy ostenta la familia GNU/Linux en pos de una soberanía auténtica.

Cuba, como otras tantas naciones, ha decidido migrar su tecnología de software, de privativa a libre debido a muchas razones y ya cuenta con una distribución propia: Nova GNU/Linux (en adelante Nova).

En toda distribución una función básica es la de Entrada/Salida (E/S), que permite la interacción de los usuarios con el ordenador. Entre las operaciones antes mencionadas se encuentra la de copiar archivos, ya sea para distintos dispositivos o dentro del mismo ordenador. Por tanto, es necesario que esta función se ejecute de la manera más transparente posible para el usuario, de modo que no tenga que recurrir a procesos más engorrosos como usar una aplicación adicional para ejecutar las copias.

Actualmente, el manejador de archivos de Nova: *Nautilus,* no cuenta con una forma de pausar las copias para así priorizar una(s) en detrimento de otra(s); tampoco existe una manera sencilla de integrarse con un copiador externo, lo cual es una limitación de peso en el citado manejador de archivos y, consecuentemente, en las distribuciones que lo utilizan.

Luego de analizar la situación existente relacionada con el manejo de archivos en la distribución cubana de GNU/Linux se tiene como **problema científico:**

¿Cómo controlar el proceso de copia de archivos en distribuciones GNU/Linux?

Para solucionar el problema anterior la presente investigación tiene como **objeto de estudio** el proceso de copia de archivos, acotando el **campo de acción** al proceso de copias de archivos en distribuciones GNU/Linux. Se designa pues, como **objetivo general:** desarrollar una herramienta que provea mayor control sobre el proceso de copia de archivos en Nova. Se definen como **objetivos específicos:**

- Estudiar el proceso de copia de archivos actual en Nautilus y otras herramientas libres para tomar buenas prácticas en caso de no poderse integrar las mismas.
- Identificar requisitos funcionales y no funcionales de la herramienta a desarrollar.
- Diseñar la nueva herramienta de copia de archivos.

- Implementar la solución diseñada.
- Validar mediante pruebas, el resultado.

Para cumplir los objetivos específicos se seguirán las siguientes **tareas de investigación:**

- Estudiar y comprender el proceso de copia actual en Nova así como otras herramientas para poder determinar qué se puede mejorar con una nueva aplicación si no pudiese integrarse una herramienta existente.
- Estudiar la metodología, lenguajes y tecnologías necesarias para el desarrollo de la herramienta con el objetivo de orientar el desarrollo siguiendo patrones ya establecidos que garanticen la calidad final del software en construcción a esquemas de probada funcionalidad anteriormente.
- Identificar requisitos para diseñar la solución al problema planteado para marcar una dirección en el desarrollo de la herramienta.
- Construir los diagramas de Casos de Uso del Sistema, de Secuencia y Clases del Diseño para que sean estos quienes guíen el desarrollo de la herramienta.
- Implementar la herramienta que maneje las copias.
- Probar la calidad y funcionamiento de la herramienta.

Se define como **idea a defender:**

El desarrollo y la integración al manejador de archivos de una herramienta que gestione las copias de archivos en Nova, de manera que permita pausar las mismas y así priorizar una(s) en detrimento de otra(s) pudiera significar un mayor control al usuario.

Se establecen como **métodos científicos** de la investigación:

**Método de investigación empírico:**

La **observación** servirá de ayuda en el estudio del estado del arte para identificar ventajas y desventajas de otras herramientas similares existentes, como parte de un análisis selectivo y objetivo de las mismas.

**Métodos de investigación teóricos:**

El **analítico-sintético** brindará la posibilidad de estudiar distintos copiadores de archivos para posteriormente sintetizar y aplicar el conocimiento adquirido en pos de una mayor contribución a la solución del problema planteado.

El **histórico lógico** permitirá estudiar el modo en que han evolucionado las herramientas de copia de archivos para diseñar una solución acorde a tendencias actuales conocida la historia.

El trabajo de diploma estará estructurado en tres capítulos como se describe a continuación:

**Capítulo 1. Fundamentación teórica del proceso de copia.** Se analizan los copiadores más populares y de mejor calidad según rendimiento, integración y facilidad de uso para los usuarios. Se describen los principales conceptos para la correcta comprensión del tema.

**Capítulo 2. Diseño de la propuesta de solución.** Se describe el diseño de solución para la posterior implementación de la herramienta que maneje las copias en Nova.

**Capítulo 3. Implementación y pruebas.** Se implementa la herramienta que solucione el problema de la investigación y se prueba la calidad mediante pruebas de Caja Negra, Caja Blanca y de Rendimiento. Además se modelan elementos físicos necesarios para el despliegue de la herramienta.

## Capítulo 1. Fundamentación teórica del proceso de copia

En el presente capítulo se analizan algunos de los copiadores más populares y de mejor calidad para de esta forma encontrar fortalezas que sean de ayuda al desarrollo de la herramienta en cuanto a estrategias y maneras de hacer. Se describen además los principales conceptos utilizados en un procesode copia.

### 1.1 Archivo/Fichero

Un archivo es un mecanismo de abstracción para representar datos de una manera simple usando básicamente su nombre y opcionalmente una extensión que indica el tipo de archivo. Por ejemplo el archivo *fichero.txt* indica que *fichero* es de tipo texto. Un *directorio* o *carpeta* es un archivo especial porque no tiene extensión y puede contener otros archivos, sea uno o varios ficheros simples o directorios [1].

Es una entidad lógica compuesta por una secuencia finita de bytes... El nombre forma la identificación única en relación a los otros archivos en el mismo directorio[2].

Los archivos son unidades lógicas de información creada por los procesos. En general, un disco contiene miles o incluso millones de archivos independientes. La información que se almacena en los archivos debe ser persistente, es decir, no debe ser afectada por la creación y terminación de los procesos. Un archivo debe desaparecer sólo cuando su propietario lo elimina de manera explícita. Aunque las operaciones para leer y escribir en archivos son las más comunes[3].

Concuerdo con las definiciones anteriores aunque considero la definición [2] parcialmente correcta porque un directorio es un archivo y no tiene extensión.

### 1.2 Proceso de copia de un fichero

Un proceso de copia es la acción de duplicar uno o varios archivos desde una dirección origen hacia una dirección destino. Copiar un fichero F desde un directorio A hacia un directorio B sería duplicar el fichero F en el directorio B[1].

### 1.3 Proceso de mover un fichero

Mover un fichero desde una dirección a otra es un proceso que puede ser divido en dos fases: (1) copiar el fichero desde el origen hacia el destino, (2) eliminar el fichero de la dirección origen. Por lo tanto, mover un fichero F desde un directorio A hacia un directorio B sería duplicar el fichero F en el directorio B y luego eliminar el fichero F del directorio A[1].

### 1.4 Copiadores de archivos existentes

Un copiador de archivos es un programa para realizar básicamente las operaciones de copiar o mover ficheros de una dirección a otra. En dichas operaciones es necesario el control por parte del usuario en cuanto a pausa y reanudación ya sea por decisión propia o por algún error que el copiador sea capaz de solventar.

#### 1.4.1 TeraCopy

TeraCopy es un copiador que permite pausar, reanudar y cancelar las operaciones sobre archivos. Comprueba la integridad final de la copia y procesa una cola de operaciones. Presenta mucha estabilidad en cuanto al manejo de ficheros de distintos tamaños. Posee una integración sin problemas que lo convierten en uno de los mejores copiadoras de archivos. Entre las desventajas está la velocidad de transferencia y el uso de recursos . Sólo está disponible para Windows[6].

#### 1.4.2 Super Copy

Es uno de los copiadores más populares debido a que provee una manera robusta en cuanto a la forma de ejecución de las operaciones sobre los archivos sin afectar el rendimiento del sistema operativo (SO). Resulta muy intuitivo y casi transparente después de la primera copia. Comprueba la veracidad final de la operación y reanuda la misma en caso de que ocurriesen problemas de electricidad o en la red. Es capaz de manejar grandes volúmenes de información mediante una cola de operaciones. La característica por la cual no se integra al Nautilus es que sólo está disponible para Windows[7].

#### 1.4.3 Copy Handler

Coy Handler es una herramienta gratis para copiar y mover archivos en sistemas operativos Windows. Es altamente personalizable y se integra completamente con el sistema operativo. Pausa y reanuda las operaciones sobre los archivos; dichas operaciones son gestionadas por el orden que ocupan en cola de tareas. Permite modificar la la velocidad de la operación y es muy similar al Super Copy en cuanto a recuperación de las operaciones con ficheros cuando las mismas han sido interrumpidas por fallas eléctricas o de red. Sólo está disponible para Windows y es por ello que es imposible la integración de esta herramienta[8].

### 1.4.4 Midnight Commander

Inicialmente pensado para sistemas tipo Unix, GNU Midnight Commander es un poderoso manejador de archivos que permite copiar, mover y eliminar ficheros. Una de sus fortalezas es que permite realizar las operaciones anteriores a través de los protocolos SSH, SAMBA, FTP y FISH. Es distribuido bajo la Licencia Pública General de GNU (GPL)[4]. A pesar de poseer las características antes mencionadas se decide no usarlo para integrarse a Nautilus porque es también un manejador de ficheros y no una herramienta de copia solamente. Aunque en sus últimas versiones tiene soporte para el ratón (mouse) su ejecución es en modo texto; teniendo así una curva de aprendizaje medianamente grande. Por las desventajas anteriores se decide no integrar dicha herramienta a Nautilus.

### 1.4.5 Dolphin

Es el manejador de archivos por defecto del entorno de escritorio KDE y actualmente está distribuido bajo la licencia GPL. Permite realizar operaciones sobre archivos tales como copiar, mover y eliminar localmente o usando protocolos de red. Las operaciones antes mencionadas se pueden pausar, reanudar o eliminar independientemente del destino o el origen de ellas[22]. El principal inconveniente para usarlo como posible elemento de integración a Nautilus es que más allá de las funcionalidades deseadas para el manejo de ficheros es otro gestor de archivos en sí y no una simple herramienta. Además está desarrollado en el lenguaje C++ usando las bibliotecas de componentes gráficos Qt, tecnologías estas muy distintas y con poca integración a las tencologías usadas para la creación de Nautilus; dichas tecnologías son: el lenguaje C y las bibliotecas de componentes gráficos GTK.

### 1.4.6 Ultracopier

Ultracopier es una herramienta de copia de archivos que posee una gran portabilidad como una de sus principales características puesto que se encuentra disponible para los sistemas operativos con mayor cantidad de usuarios (Windows, Mac y similares a UNIX). También permite gestionar una cola de operaciones (entiéndase copiar o mover) que el usuario puede ordenar según sus prioridades. Dichas operaciones pueden ser pausadas, reaunadadas o canceladas. Para todas las operaciones en conjunto se puede establecer una cota superior de velocidad siempre que estén disponiblen los recursos necesarios en el SO. Se distribuye bajo la licencia GPL[5]. Entre las razones que llevaron a no seleccionar esta herramienta para la integración con Nautilus se encuentra la imposibilidad de la misma para realizar operaciones mediante protocolos de comunicación en la red. Otro motivo es la diferencia de tecnologías ya que Ultracopier está escrito en C++ usando las bibliotecas de

componentes gráficos Qt y Nautilus está escrito en C y tiene como las bibliotecas de componentes gráficos a GTK, siendo tema determinante el rendimiento en las operaciones si se pudiesen salvar las distancias de tecnologías.

### 1.4.7 Rsync

Rsync es un copiador muy potente debido a que realiza las operaciones de copia de ficheros mediante protocolos de red o localmente a gran velocidad e integridad en los datos. Es capaz de mantener enlaces en los ficheros que copia y seguir expresiones regulares de inclusión o exclusión en las operaciones como algunas de sus principales caracterísitcas. Dichas operaciones pueden ser pausadas, reanudadas o canceladas. Tiene la capacidad de continuar una operación justo por donde iba el proceso si hubiese sido interrumpido por algún motivo. Permite la ejecución de copias usando túneles para encriptar y/o comprimir datos. Además es ampliamente usado para hacer copias de respaldo debido a que sólo copia los datos que no han sido modificados desde la última operación realizando una sincronización óptima de archivos. Es distribuído baja la licencia GPL[19]. Los motivos por los cuales se decide no usar Rsync es que no permite copiar desde un anfitrión (más conocido por el término inglés *host*) remoto a otro y es necesario que esté instalado en ambas computadoras (fuente y destino).

### 1.4.8 MMV

MMV es una aplicación que permite copiar, mover, agregar y hacer enlaces entre archivos. Su principal potencia es que se pueden realizar las operaciones antes mencionadas a través de patrones o expresiones regulares, de esta forma los ficheros que cumplan con el patrón definido serán los ficheros de origen en la operación. Otra de sus opciones es que comprueba la mayoría de posibles errores antes de realizar cada operación e informa al usuario de haber ocurrido alguno y espera la instrucción de este para ejecutar alguna acción; el usuario también puede escoger un comportamiento por defecto ante errores para no ser encuestado en caso de otro posible error[20]. Entre las desventajas que no hicieron posible la integración de esta aplicación a Nautilus está la imposibilidad de la misma de llevar a cabo operaciones mediante la red, además su interacción es mediante líneas de comandos y no permite mover o crear directorios.

### 1.4.9 Advanced command-line file copier

Como su nombre lo indica Advanced command-line file copier (GCP) es un copiador de ficheros usando líneas de comandos. Está escrito en Python y basado en el comando cp aunque agrega funcionalidades de más alto nivel. Es capaz de administrar una cola de operaciones y retomar las operaciones por donde estaba si ocurriese algún error externo. Además se recupera ante incompatibilidades de codificación y puede saltar operaciones con error y seguir con el próximo objetivo en la cola de operaciones. Mantiene al usuario actualizado en cuanto al avance de las operciones a través de una barra de progreso[21]. Se decidió no integrar esta herramienta a Nautilus porque no permite la realización de operaciones mediante la red y como anteriormente se especificó la intercción es mediante líneas de comandos. Otro inconveniente fue tecnológico ya que Python es un lenguaje interpretado, lo cual lacera la velocidad frente a grandes volúmenes de operaciones.

## 1.5 Herramientas a Utilizar

### 1.5.1 Herramientas CASE

CASE (Ingeniería de Software Asistida por Computadora, por sus siglas en Inglés). Una herramienta de tipo CASE está formada por programas y ayudas para la automatización de tareas en el desarrollo de un software, completamente o en alguna de sus fases.

#### 1.5.1.1 Visual Paradigm

Visual Paradigm es una herramienta CASE multiplataforma para el modelado de software. Ofrece calidad junto a simplicidad y ligereza en el desarrollo. Muestra además una gran integración con otras herramientas CASE y otros Entornos Integrados de Desarrollo. Dentro de sus mayores ventajas está el completo desarrollo en *modelo código despliegue* ya que permite generar código fuente a partir de modelada la solución deseada y posteriormente pasar al despliegue siguiendo las particularidades de los pasos anteriores (modelado y generado de código).[9]

### 1.5.2 Vala

Lenguaje de programación multiplataforma (está disponible para Windows, Mac y similares a UNIX), multiparadigma (se puede programar usando los paradigmas de programación: Orientado a Objetos o Estructurado). Creado con el objetivo de acercar las facilidades descriptivas y de abstracción de los lenguajes modernos como C# al lenguaje C. Su sintaxis se inspira en C#, con modificaciones para adaptarse mejor al sistema de objetos de Gobject y GLib pues está enfocado en dichas bibliotecas. No necesita requisitos adicionales en tiempo de ejecución ni gran cantidad de costos

computacionales, siendo más rápido que lenguajes como Python o Mono porque Vala traduce a lenguaje C que luego se compila para la arquitectura deseada. Ha sido desarrollado por Jürg Billeter y Raffaele Sandrini[10].

### 1.5.3 IDE

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en Inglés) es una herramienta para la creación y edición de código de programación. Generalmente está compuesto por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Puede ser una aplicación por sí sola o formar parte de otra.

#### 1.5.3.1 Anjuta

Es el IDE por excelencia del proyecto GNOME y aunque su lanzamiento es relativamente reciente (lanzado en 1999) en comparación a otros establecidos brinda muchas facilidades como la gestión de proyectos, varios asistentes, plantillas, resaltado de sintaxis y depuradores de código fuente. También tiene embebida una ayuda para desarrolladores *Devhelp.* Es capaz de generar clases y soporta varios lenguajes entre los que se encuentran Vala, C, C++, Java y Python. El control de versiones puede ser realizado mediante la herramienta Git que también está integrada el el mismo. Está diseñado para ser simple aunque potente. Está liberado bajo la licencia GPL[11].

### 1.5.4 GTK 3

Es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario. Licenciada bajo la Licencia Pública General de GNU. Permite programar usando funciones de lenguajes como Vala, C, C++, C#, Java, Ruby, Perl, PHP o Python mediante una arquitectura orientada a objetos basada en C que permite una máxima flexibilidad[12][13].

### 1.5.5 Metodología de desarrollo de software

Las metodologías en el desarrollo de software son excelentes prácticas y casi obligatorias si se desea lograr de antemano calidad en el producto final. Es un trabajo delicado la selección de una metodología para la creación de software porque no todas son buenas para cualquier ambiente, es por ello que se según ventajas y desventajas en distintas situaciones se escoge una u otra. Las metodologías, por estar probadas con anterioridad, guían a los desarrolladores en cuanto a organización y entendimiento de la implementación.

Existen dos ramas en las metodologías: (1) metodologías pesadas y (2) metodologías ligeras. Las primeras intentan conseguir el objetivo por medio de orden y documentación; en cambio las segundas se orientan a mejorar la calidad del proceso por medio de la comunicación directa e inmediata entre

las personas que que intervienen en el proyecto [15].

#### 1.5.5.1 OpenUp

OpenUp es una metodología ágil iterativa e incremental dentro de un ciclo de vida estructurado. Posee un conjunto mínimo de prácticas que ayudan a la efectividad en el desarrollo de software. OpenUp es un conjunto de herramientas neutrales que pueden extenderse para alcanzar una amplia variedad de tipos de proyectos. Surge a partir de la metodología RUP (Rational Unified Process) y está basada en la misma reduciéndose a una serie de prácticas. Consta de cuatro fases: Inicio, Elaboración, Construcción y Transición [16].

**Conclusiones del capítulo**

En este capítulo se realizó un estudio de los algunos de los copiadores de archivos existentes hoy día. Los copiadores privativos TeraCopy, Super Copy y Copy Handler no fueron escogidos por no estar disponible el código fuente, siendo así imposible un estudio de los mismos para una posible integración al manejador de archivos de la distribución Nova. Los manejadores de ficheros Midnight Commander y Dolphin, que tienen función de copia, no se integraron porque son manejadores de archivos como lo es Nautilus. Los copiadores Ultracopier y GCP fueron excluidos por incompatibilidad de tecnologías y al igual que MMV no son capaces de efectuar operaciones usando protocolos de red. Aunque Rsync si permite usar la red para realizar operaciones, tampoco fue escogido porque origen y destino no pueden ser remotos simultáneamente.

La situación del estado del arte antes planteada conlleva a que sea necesario el desarrollo de una nueva herramienta (Ncopier) capaz de integrarse a Nautilus; que permita la realización de operaciones mediante protocolos de red y también pausar, reanudar o cancelar a modo de asignación de prioridades. Para lograr este objetivo se escogió Visual Paradigm como herramienta para el modelado, Anjuta como IDE, Vala como lenguaje de programación y GTK como las bibliotecas de componentes gráficos.Todo el flujo de desarrollo estará guiado por la metodología OpenUp.

## Capítulo 2. Diseño de la propuesta de solución

En el capítulo actual se abordan los procesos concernientes al diseño de la herramienta. Se definen sus actores, principales funcionalidades, propiedades y restricciones para su uso. Se muestran los diagramas y elementos de diseño de arquitectura más importantes utilizados para la contrucción de la herramienta como posible solución al problema de la investigación.

### 2.1 Descripción de la propuesta de solución

Se pretende crear una herramienta que permita manejar distintos procesos de copia simultáneamente y que además brinde la posibilidad de pausar, reanudar y cancelar los mismos. Dichas operaciones podrán ser locales o remotas. La integración de la aplicación con el manejador de archivos Nautilus debe ser tan simple como se pueda de modo que el usuario no necesite grandes nuevos conocimientos para la realización de las operaciones. La herramienta deberá tener un rendimiento similar al proceso de copias actual de Nautilus y la usabilidad en la misma estará dada por el modo de establecer prioridades entre operaciones pausando o reanudando según el deseo del usuario.

### 2.2 Actores del Sistema

Un actor del sistema es una entidad que inicia una funcionalidad o brinda información al mismo.

| Nombre del Actor | Descripción |
|---|---|
| Usuario | El usuario o proceso es quien realiza las operaciones: copiar, mover. Dicho usuario o proceso puede cambiar el estado de las mismas; los estados pueden ser: pausar y reanudar la operación. |

### 2.3 Requistos

Un requerimiento o requisito puede ser una condición o capacidad que debe ser poseída por un sistema o componente del sistema para satisfacer un contrato, estándar, especificación, u otro documento formalmente impuesto.

Muchas veces se hace necesario el empleo de técnicas para establecer una buena comunicación con los interesados del producto y así lograr la satisfacción del cliente. Algunas son:

- Entrevista: es la más tradicional y consiste en reuniones analista-interesado en las cuales se suceden preguntas y respuestas para extraer el dominio de la aplicación.
- Juego de roles: variante en la que el desarrollador, el analista y cada uno de los miembros del equipo de desarrollo del software toman el lugar del interesado y ejecutan la actividad de trabajo que éste desempeña. Ellos experimentan las inexactitudes y problemas ligados con el

sistema que se está especificando. Se busca suministrarle al analista una perspectiva nueva del problema que le permita la obtención de los requisitos del sistema por construir.

### 2.3.1 Requisitos funcionales

Los requisitos funcionales representan características necesarias para la ejecución de la aplicación.

**RF1.** Copiar fichero[1].

**RF2.** Mover fichero.

**RF3.** Pausar operación[2].

**RF4.** Reanudar operación.

**RF5.** Cancelar operación.

### 2.3.2 Requisitos no funcionales

Los requisitos no funcionales definen propiedades y restricciones del sistema, que pueden estar relacionadas con el entorno, implementación, rendimiento, dependencia de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad.

**Interfaz**

**RNF1.** Sólo agregará al diseño actual del manejo de copias de Nautilus un botón con los posibles estados de las operaciones (pausar/reanudar) a la izquierda del botón cancelar. Cuando existan más de dos operaciones en ejecución (independientemente de que alguna operación esté pausada) sólo serán visibles dos en una única ventana, las demás podrán ser vistas a través una barra de desplazamiento a la derecha de la ventana.

**Usabilidad**

**RNF2.** El trabajo con la herramienta será intuitivo y no requerirá grandes conocimientos adicionales debido a los pequeños cambios a realizar (descritos en RNF1) sobre el diseño actual de el manejo de copias archivos de Nautilus.

**Rendimiento**

**RNF3.** La herramienta deberá realizar las operaciones de copia de archivos con una velocidad similar a la que posee Nautilus para dichas operaciones sin comprometer los datos con los cuales trabaja; de modo que velocidad y estabilidad no sean caracterísitcas excluyentes.

---

1 Un directorio es un fichero especial.

2 Una operación puede ser copiar o mover uno o varios ficheros.

**Software**

**RNF4.** La herramienta podrá ser utilizada en cualquier distribución similar a Nova 4.0. Tienen que estar presente además las bibliotecas de componentes gráficos GTK en su versión 3 y el compilador de Vala en la versión 0.16.1.

**Licencia**

**RNF5.** La herramienta será liberada bajo la licencia GPL por lo que no será necesario adquirir licencias o patentes para el uso de la misma.

### 2.4 Diagrama de Caso de Uso del Sistema

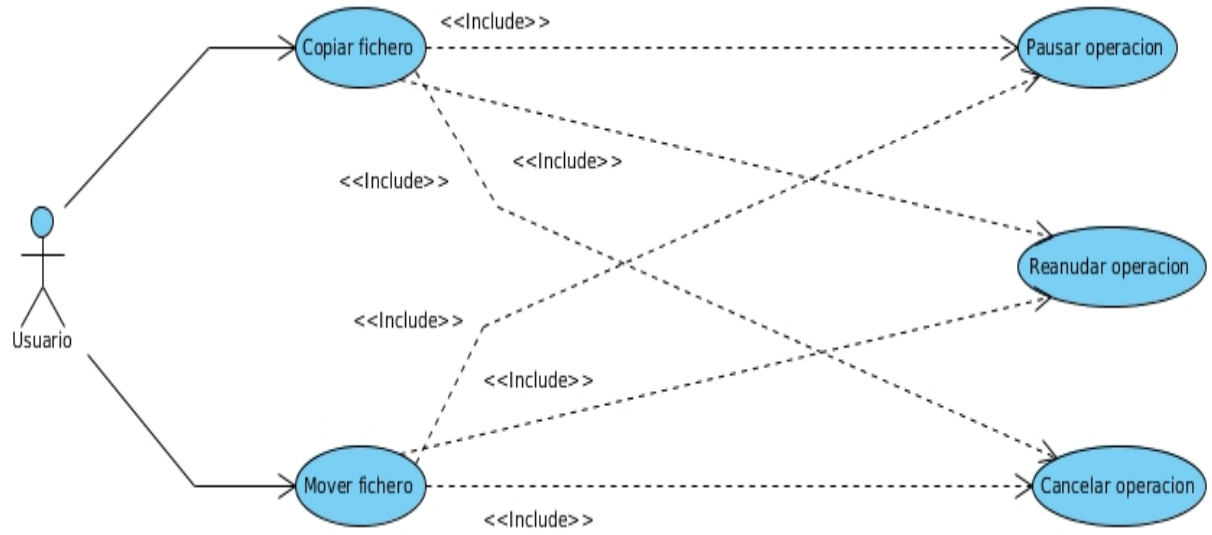


*Figura 1: Diagrama de Caso de Uso del Sistema*

#### 2.4.1 Descripción de los Casos de Uso del Sistema

Los requisitos funcionales y el Diagrama de Caso de Uso (CU) del Sistema anteceden a la descripción de las funcionalidades que puede ofrecer el sistema.

#### 2.4.1.1 CU Copiar fichero

| Caso de Uso: | Copiar fichero |
|---|---|
| Tipo: | Funcional |
| Actores: | Usuario |
| Resumen: | El CU inicia cuando el usuario copia y pega un directorio o fichero |
| Precondiciones: | |
| Referencia(s): | RF1 |
| Prioridad: | Alta |
| Complejidad: | Normal |
| Flujo normal de eventos | |
| Acción del actor | Respuesta del sistema |
| 1. El usuario copia y pega el fichero donde desea sea copiado el mismo | 2. El sistema muestra una interfaz donde se actualiza el proceso de copia mientras esta se realiza. Dicha operación puede ser pausada/reanudada o cancelada |

#### 2.4.1.2 CU Mover fichero

| Caso de Uso: | Mover fichero |
|---|---|
| Tipo: | Funcional |
| Actores: | Usuario |
| Resumen: | El CU inicia cuando el usuario corta y pega un directorio o fichero |
| Precondiciones: | |
| Referencia(s): | RF2 |
| Prioridad: | Alta |
| Complejidad: | Normal |
| Flujo normal de eventos | |
| Acción del actor | Respuesta del sistema |
| 1. El usuario corta y pega el fichero donde desea sea movido el mismo | 2. El sistema muestra una interfaz donde se actualiza el movimiento del fichero mientras este se realiza. Dicha operación puede ser pausada/reanudada o cancelada |

#### 2.4.1.3 CU Pausar operción

| Caso de Uso: | Pausar operación |
|---|---|
| Tipo: | Funcional |
| Actores: | Usuario |
| Resumen: | El CU inicia cuando el usuario pausa una operación |
| Precondiciones: | Tiene que haber una operación en ejecución |
| Referencia(s): | RF4 |
| Prioridad: | Alta |
| Complejidad: | Normal |
| Flujo normal de eventos | |
| Acción del actor | Respuesta del sistema |
| 1. El usuario pausa una operación | 2. El sistema muestra una interfaz donde la operación está pausada. Dicha operación puede ser reanudada o cancelada |

#### 2.4.1.4 CU Reanudar operación

| Caso de Uso: | Reanudar operación |
|---|---|
| Tipo: | Funcional |
| Actores: | Usuario |
| Resumen: | El CU inicia cuando el usuario reanuda una operación |
| Precondiciones: | Tiene que haber una operación pausada |
| Referencia(s): | RF5 |
| Prioridad: | Alta |
| Complejidad: | Normal |
| Flujo normal de eventos | |
| Acción del actor | Respuesta del sistema |
| 1. El usuario reanuda una operación | 2. El sistema muestra una interfaz donde la operación está en ejecución. Dicha operación puede ser reanudada o cancelada |

#### 2.4.1.5 CU Cancelar operación

| Caso de Uso: | Cancelar operación |
|---|---|
| Tipo: | Funcional |
| Actores: | Usuario |
| Resumen: | El CU inicia cuando el usuario cancela una operación |
| Precondiciones: | Tiene que haber una operación en ejecución |
| Referencia(s): | RF6 |
| Prioridad: | Alta |
| Complejidad: | Normal |
| Flujo normal de eventos | |
| Acción del actor | Respuesta del sistema |
| 1. El usuario cancela una operación | 2. El sistema elimina la interfaz que representaba la operación antes de ser cancelada la misma |

### 2.5 Diagrama de Clases del Sistema

Un diagrama de clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Este diagrama se compone de los nombres de las clases, sus atributos, funcionalidades y relaciones.(Ver anexo 1. Diagrama de Clases del Sistema)

### 2.6 Diagrama de Procesos del Negocio

Los diagramas de procesos del negocio sirven para modelar dicho negocio para localizar las actividades y su flujo lógico. Esta debe ser la forma de pensamiento lógico cuando se diseña un software.(Ver anexo 2. Diagramas de Procesos del Negocio)

### 2.7 Arquitectura del Sistema

La selección del patrón arquitectónico Modelo Vista Controlador (MVC) responde a que el mismo separa la lógica de negocio y la interfaz de usuario. Además facilita la funcionalidad y el mantenimiento del sistema proponiendo una división lógica entre componentes; la comunicación entre estos se realiza mediante el paso de mensajes según se necesite. El Modelo representa la lógica de negocio. Es el encargado de acceder de forma directa a los datos. En la herrmienta el modelo está compuesto por las clases Operation y AtomicOperation, las cuales representan y llevan a cabo la operación indicada por el usuario. La Vista es la encargada de mostrar la información al usuario de forma gráfica. Las vistas de la aplicación son representadas por las clases MainView y OperationView para informar al usuario el avance de la operación en todo momento. El Controlador es el intermediario entre la vista y el modelo; es quien controla las interacciones del usuario solicitando los

datos al modelo y entregándolos a la vista para que esta los presente al usuario. La clase Controller en la herramienta es la encargada de convertir la interacción del usuario a través de la vista en acciones sobre el modelo.

### 2.8 Diagrama de Componentes

Un componente es una parte o módulo del sistema, encapsula la implementación y un conjunto de interfaces proporcionando la realización del mismo. Generalmente contiene clases y puede ser agrupado en paquetes siguiendo un criterio lógico con vista a simplificar la implementación. El diagrama de componentes muestra un conjunto de componentes relacionados entre sí para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones de los elementos de implementación, modelando de esta manera la vista estática del sistema.(Ver anexo 3. Diagrama de Componentes)

### 2.9 Diagrama de Secuencia

La interacción de los objetos de software en una aplicación a través del tiempo es lo que representa el Diagrama de Secuencia. Éste tipo de diagrama posee especificidades de implementación del CU, incluye los objetos y clases que se usan para implementar el escenario del CU, así como los mensajes intercambiados entre los objetos. Se muestran los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.(Ver anexo 4. Diagramas de Secuencia)

### 2.10 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. En otras palabras, un patrón de diseño no es más que una solución estándar para un problema común de programación. Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios[17].

#### 2.10.1 Patrones GRASP

El grupo de Patrones Generales de Software para Asignar Responsabilidades (GRASP por sus siglas en Inglés) es un grupo de patrones de diseño que describe cómo asignar responsabilidades a objetos dentro del sistema.

En el diseño propuesto, pertenecientes al grupo GRASP, se encuentran los siguientes patrones:

- **Experto:** que tiene como objetivo asignar una determinada responsabilidad a la clase que posea la información necesaria para cumplirla. Se observa el uso del mismo en la clase AtomicOperation que es quien puede borrar directamente un fichero cuando sea necesario.

- **Creador:** para asignar a una clase la responsabilidad de crear una instancia de otra clase. Pueder ser visto en la clase Operation cuando crea objetos de tipo AtomicOperation para procesar las señales que esta última pueda emitir.
- **Controlador:** para asignar la responsabilidad del manejo de los eventos de un sistema a una clase. Puede observarse este patrón en la clase Controller, quien se encarga de ejecutar acciones enlos modelos luego de haber sido emitidas las señales por las clases de interfaz.
- **Bajo acoplamiento:** con el objetivo de asignar una responsabilidad para mantener bajo el acoplamiento[3] entre las clases.La utilización de este patrón puede ser observada en la clase Operation que solamente depende de la clase AtomicOperation para su funcionamiento.

[3] Es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras clases.

## Conclusiones del capítulo

Con el desarrollo de este capítulo quedaron definidos aspectos necesarios en el análisis y diseño de la aplicación. Se estableció al usuario como el actor del sistema. Además se fijaron los requisistos funcionales y no funcionales de la herramienta. Se construyeron los distintos diagramas que guiarán el proceso de implementación desde el estado inicial hasta la interacción de los objetos para responder a determinada operación. La implementación seguriá la arquitectura MVC y los patrones Experto, Creador, Controlador y Bajo Acoplamiento para la asignación de responsabilidades entre objetos.

## Capítulo 3. Implementación y Pruebas

En el presente capítulo se reflejan elementos primordiales en cuanto a implementación y pruebas luego de haber terminado el diseño del sistema. Se describe la implementación de la herramienta, así como sus principales métodos. Se incluye además, elementos referentes al resultado de un conjunto de pruebas de Caja Blanca y de Caja Negra realizadas al software para verificar su funcionamiento y respuesta a los requerimientos iniciales. Además se realizará una comparación con la herramienta actual de copias de Nautilus para analizar el rendimiento de la herramienta desarrollada. Quedará descrito cómo los elementos de diseño se implementan en términos de componentes y cómo los mismos se organizan en nodos específicos en el diagrama de despliegue.

### 3.1 Estándar de Codificación

Un estándar de codificación en informática son reglas que se rigen la escritura del código fuente; con el fin de homogeneizar la manera de codificar para que otros puedan entenderlo sin muchas dificultades. Parte esencial en cada estándar lo establecen las formas de escribir clases, métodos y variables.

#### 3.1.1 Estándar para la codificación en Vala

Los nombres de clases, interfaces, estructuras, espacios de nombres y enumeradores siguen el método Camel. Debe comenzar con letra inicial mayúscula y el resto minúscula; de existir la necesidad de un nombre compuesto cada letra con que inicie la palabra será mayúscula y el resto minúscula. Ejemplo: *AtomicOperation*. Los métodos, variables locales, campos, propiedades y señales se escribirán todos en minúsculas y en caso de ser más de una palabra, se separa con un guión bajo. Ejemplo: *lock_mutex*.

### 3.2 Diagrama de Despliegue

Lo s diagramas de despliegue visualizan la distribución de los componentes de software en los nodos físicos. En él se especifican tres elementos fundamentales: Nodos, Dispositivos y Conectores. Los Nodos describen los elementos de procesamiento con al menos un procesador, memoria y cualquier otro dispositivo. Los dispositivos: son nodos que no tienen capacidad de procesamiento cuando se modela. Los Conectores expresan el tipo de protocolo utilizado en el resto de los elementos del modelo. En el diagrama de despliegue del problema se aprecia como Nautilus utiliza la herramienta Ncopier para copiar o mover ficheros, ya sea localmente o a través de la red.(Ver anexo 5. Diagrama de Despliegue)

### 3.3 Pruebas

El nivel de utilización del software como un elemento casi indispensable en la mayoría de los sistemas en la actualidad reviste gran importancia en cuanto a costos por fallos del mismo; obligando dichas circunstancias a la creación de pruebas minuciosas y bien planificadas. Una prueba es una actividad en un sistema o componente ejecutado bajo condiciones específicas para ser observados y corregir futuros errores. Es la prueba de software un elemento crítico para la garantía de la calidad del software al convertirse esta en la revisión final de las especificaciones del diseño y de la codificación.

#### 3.3.1 Casos de prueba

Un caso de prueba es un conjunto de condiciones mediante las cuales se comprueba el correcto funcionamiento del software según los requisitos definidos para el mismo. El caso de prueba especifica la forma de probar un sistema incluyendo las entradas, salidas y resultados esperados, así como bajo qué condiciones debe probarse el software.

#### 3.3.2 Pruebas de Caja Negra

Las pruebas de Caja Negra o pruebas de comportamiento se centran en los requisitos funcionales del software e intentan encontrar errores tales como: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos, errores de rendimiento y errores de inicialización o terminación.

Entre los diversos métodos de prueba de Caja Negra uno de los más usados es la partición equivalente: este método divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar[18].

En los casos de prueba siguientes se vincula caso de prueba con su respectiva entrada, salida y condiciones de ejecución.

#### 3.3.2.1 Caso de prueba: Copiar fichero

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario selecciona, copia y pega el fichero que desea, donde desea. | El fichero en cuestión es copiado a donde el usuario eligió. | Es necesario tener permisos en el directorio donde se desea pegar el fichero inicialmente seleccionado. |

#### 3.3.2.2 Caso de prueba: Copiar directorio hacia otro sin sin espacio suficiente

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario selecciona, copia y pegael directorio que desea, en otro sin espacio libre. | Se muestra un diálogo que informa de la inexistencia de espacio libre en disco para efectuar la copia, en dicho diálogo se puede escoger entre cancelar, copiar de cualquier manera o reintentar la copia. | Es necesario tener permisos en el directorio donde se desea pegar el directorio inicialmente seleccionado. |

#### 3.3.2.3 Caso de prueba: Mover directorio hacia otro a través de SAMBA

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario selecciona y corta el directorio que desea y lo pega en el directorio compartido con SAMBA. | El directorio es movido a donde el usuario eligió. La interfaz que muestra los detalles de la operación no mostró correctamente el nombre del directorio destino. | Es necesario tener permisos en el directorio hacia donde se desea mover el directorio inicialmente seleccionado. La carpeta compartida mediante SAMBA tiene que estar montada[4] en el sistema de ficheros[5]. |

El error de la interfaz en esta prueba fue corregido en la segunda iteración de desarrollo.

[4] Anexada al sistema de fichero.

[5] Estructura lógica que representa la forma en que están almacenados los archivos en el disco.

#### 3.3.2.4 Caso de prueba: Copiar fichero desde un directorio a través de FTP

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario selecciona y copia el fichero que desea desde el FTP y lo pega un directorio destino. | El fichero en cuestión es copiado a donde el usuario eligió. | Es necesario tener permisos en el directorio a donde se desea copiar el fichero seleccionado.La carpeta compartida mediante FTP tiene que estar montada en el sistema de ficheros. |

#### 3.3.2.5 Caso de prueba: Pausar el movimiento de un fichero

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario hace click en el primer botón de izquierda a derecha que representa la opción de pausar el movimiento del fichero. | El proceso de mover el fichero se pausa, al igual que el desplazamiento de la barra de progreso. El botón previamente oprimido cambia el icono y muestra el icono que indica reanudar el movimiento. | Es necesario que esté en ejecución la operación de mover un fichero. |

#### 3.3.2.6 Caso de prueba: Reanudar el movimiento de un fichero

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario hace click en el primer botón de izquierda a derecha que representa la opción de reanudar el movimiento del fichero. | El proceso de mover el fichero se reanuda. La interfaz vuelve a actualizarse con los datos de la operación en ejecución. El botón previamente oprimido cambia el icono para indicar pausar el movimiento. | Es necesario que esté pausada la operación de mover un fichero. |

#### 3.3.2.7 Caso de prueba: Cancelar el movimiento de un fichero

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario hace click en el segundo botón de izquierda a derecha que representa la opción de cancelar el movimiento del fichero. | El proceso de mover el fichero se cancela y desaparece la interfaz que describía dicho proceso. El fichero final es eliminado y se mantiene el original. | Es necesario que esté en ejecución la operación de mover un fichero. |

#### 3.3.2.8 Caso de prueba: Cancelar la copia de un directorio

| Entrada | Resultados | Condiciones |
|---|---|---|
| El usuario hace click en el segundo botón de izquierda a derecha que representa la opción de cancelar la copia. | El proceso de copia se cancela y desaparece la interfaz que describía dicho proceso. El fichero final es eliminado y se mantienen los copiados hasta el momento. | Es necesario que esté en ejecución la operación de copiar un directorio. |

### 3.3.3 Pruebas de Caja Blanca

Las pruebas de Caja Blanca se realizan sobre funciones internas del software. Están encaminadas a probar el código del software. Algunas de sus técnicas son: la cobertura de caminos (hacen que se recorran todos los posibles caminos de ejecución), pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos y comprobación de bucles.

La siguiente figura representa este el funcionamiento de las pruebas de Caja Blanca:

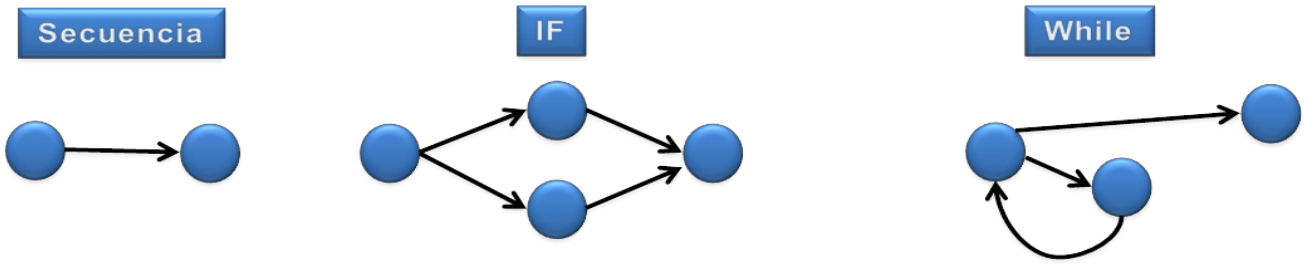


*Figura 2: Representación de algunas sentencias para pruebas de Caja Blanca*

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de Flujo en el cual: (1) Cada nodo del grafo corresponde a una o más sentencias de código fuente, (2) Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo y (3) Se calcula la complejidad ciclomática del grafo. En el grafo un nodo es un círculo que representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no

están asociados se utilizan al inicio y final del grafo. Las aristas son constituidas por las flechas en el grafo y signan el control del flujo del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no represente la sentencia de un procedimiento. Las regiones están constituidas por las áreas delimitadas por las aristas y nodos; además se incluye el área exterior del grafo como una región más.

Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

#### 3.3.3.1 Diseño del caso de prueba Copiar fichero

Esta funcionalidad copia el fichero origen en el fichero destino.

```
78      public int copy (int64 source_size, string source_f, string destination_f) {
79          int answer = 0;
80          uint writed = 0;
81          uint tmp = 0;
82          try {
83              source_file = File.new_for_uri (source_f);
84              destination_file = File.new_for_uri (destination_f);
85              DataInputStream data_input_stream = new DataInputStream (new BufferedInputStream
86                                                                       (source_file.read (null)));
87              DataOutputStream data_output_stream = new DataOutputStream (new BufferedOutputStream.sized
88                                      (destination_file.create ((FileCreateFlags.NONE), null), BUFFER_SIZE));
89              uint8[] buffer = new uint8[BUFFER_SIZE];
90              uint readed = (uint) data_input_stream.read (buffer, null);
91              while (!canceled_operation && (writed < source_size)) {
92                  while ((readed > 0) && (!canceled_operation) && (destination_file.query_exists ())) {
93                      mutex.lock ();
94                      tmp = (uint) data_output_stream.write (buffer[0:readed], null);
95                      writed += tmp;
96                      total_writed += tmp;
97                      readed = (uint) data_input_stream.read (buffer, null);
98                      mutex.unlock ();
99                      writed_signal (total_writed);
100                 }
101             }
102         } catch (IOError.IS_DIRECTORY e) {
103             stderr.printf ("The file '%s' is a directory.\n", source_file.get_basename ());
104         } catch (IOError.EXISTS e) {
105             answer = -1;
106             stderr.printf ("The file '%s' already exists.\n", destination_file.get_basename ());
107         } catch (Error e) {
108             answer = 1;
109             stderr.printf ("in catch error-source:%s-destination:%s-%s\n", source, destination, e.message);
110         }
111         return answer;
112     }
```

*Figura 3: Fragmento de código para realizar prueba de Caja Blanca*

Enumerando el código anterior para un posterior análisis se tiene que:

- El nodo 1 representa desde la línea 79 a la condición a la izquierda del símbolo && en la línea 91.
- El nodo 2 representa la condición que está a la derecha del símbolo && en la línea 91.
- El nodo 3 representa la condición que está a la izquierda del primer símbolo && en la línea 92.
- El nodo 4 representa la condición que está entre los símbolos && en la línea 92.
- El nodo 5 representa la condición que está a la derecha del segundo símbolo && en la línea 92.
- El nodo 6 representa desde la línea 93 a la 99.
- El nodo 7 representa la línea 100.
- El nodo 8 representa la línea 101.
- El nodo 9 representa desde la línea 102 a la 111.

A partir de dicha enumeración y teniendo en cuenta la notación para las instrucciones, se construye el grafo de flujo:

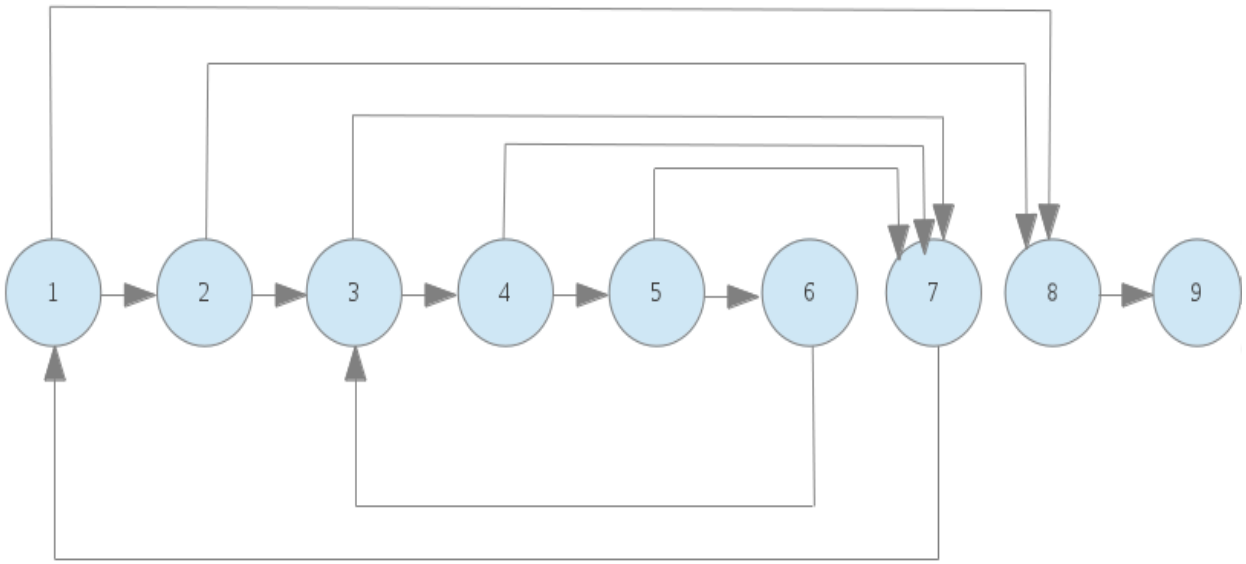


*Figura 4: Grafo asociado al fragmento de código para calcular la complejidad ciclomática*

Luego se calcula la complejidad ciclomática (V(G)) mediante las siguientes fórmulas:

V(G) = Aristas - Nodos + 2 = 13 - 9 + 2 = 6

V(G) = Nodos predicados + 1 = 5 + 1 = 6

V(G) = Número de regiones = 6

El cálculo anterior arroja el mismo valor para el resultado, afirmando que la complejidad ciclomática del código es 6, lo cual acota superiormente la cantidad de caminos[6] independientes que componen el conjunto básico y, consecuentemente, un valor límite superior para el número de pruebas que se deben diseñar y ejecutar para garantizar que se cubran todas las sentencias de los procedimientos.

| Camino | Secuencia de nodos |
|---|---|
| 1 | 1-8-9 |
| 2 | 1-2-8-9 |
| 3 | 1-2-3-7-1-8-9 |
| 4 | 1-2-3-4-7-1-8-9 |
| 5 | 1-2-3-4-5-7-8-9 |
| 6 | 1-2-3-4-5-6-3-7-1-8-9 |

#### 3.3.3.2 Caso de prueba para el camino 1

Descripción: En este caso se copia un archivo de tipo video el cual se espera sea copiado a un directorio, además se pasa como parámetro del método el tamaño de dicho archivo y se cancela la operación sin terminar.

Condición de ejecución: Debe cancelar la operación y eliminar dicho archivo por no haber llegado al fin de la copia correctamente.

Entrada: Fichero de tipo video, directorio hacia el cual copiar dicho archivo y el tamaño del archivo antes mencionado.

Resultados esperados: Se cancela la operación y se elimina dicho archivo.

#### 3.3.3.3 Caso de prueba para el camino 2

Descripción: En este caso se copia un archivo de tipo video el cual se espera sea copiado a un directorio, además se pasa como parámetro del método el tamaño de dicho archivo y se cancela la operación sin terminar.

Condición de ejecución: Debe cancelar la operación y eliminar dicho archivo por no haber llegado al fin de la copia correctamente.

Entrada: Fichero de tipo video, directorio hacia el cual copiar dicho archivo y el tamaño del archivo antes mencionado.

Resultados esperados: Se cancela la operación y se elimina dicho archivo.

[6] Un camino es la sucesión de nodos enlazados mediante aristas desde el primero hasta el último nodo.

Para este caso de prueba no existe forma de probar puesto que se puede variar los bytes escritos al fichero final manualmente ni se puede cambiar el tamaño del fichero inicial. Sólo se comprueba que funciona bien en la prueba del camino 6 cuando se copie el archivo correctamente.

#### 3.3.3.4 Caso de prueba para el camino 3

Descripción: En este caso se copia un archivo de tipo video el cual se espera sea copiado a un directorio, además se pasa como parámetro del método el tamaño de dicho archivo y se cancela la operación sin terminar.
Condición de ejecución: Debe cancelar la operación y eliminar dicho archivo por no haber llegado al fin de la copia correctamente.
Entrada: Fichero de tipo video, directorio hacia el cual copiar dicho archivo y el tamaño del archivo antes mencionado.
Resultados esperados: Se cancela la operación y se elimina dicho archivo.
Éste caso de prueba se ejecutó bien al saber que se comenzó al menos a copiar el archivo origen.

#### 3.3.3.5 Caso de prueba para el camino 4

Descripción: En este caso se copia un archivo de tipo video el cual se espera sea copiado a un directorio, además se pasa como parámetro del método el tamaño de dicho archivo y se cancela la operación sin terminar. La diferencia con respecto al caso 1 es que se espera a que comience la copia para cancelar la misma.
Condición de ejecución: Debe cancelar la operación y eliminar dicho archivo por no haber llegado al fin de la copia correctamente.
Entrada: Fichero de tipo video, directorio hacia el cual copiar dicho archivo y el tamaño del archivo antes mencionado.
Resultados esperados: Se cancela la operación y se elimina dicho archivo.

#### 3.3.3.6 Caso de prueba para el camino 5

Descripción: En este caso se copia un archivo de tipo video el cual se espera sea copiado a un directorio, además se pasa como parámetro del método el tamaño de dicho archivo y se cancela la operación sin terminar. La diferencia con respecto al caso 1 es que se espera a que comience la copia para cancelar la misma.
Condición de ejecución: Debe cancelar la operación al encontrar que el archivo a copiar ya existe.

Entrada: Fichero de tipo video, directorio hacia el cual copiar dicho archivo y el tamaño del archivo antes mencionado.
Resultados esperados: Se cancela la operación y se elimina dicho archivo.

#### 3.3.3.7 Caso de prueba para el camino 6

Descripción: En este caso se copia un archivo completo, debe reconocer que se llegó al fin del fichero.
Condición de ejecución: Debe cancelar la operación y eliminar dicho archivo por no haber llegado al fin de la copia correctamente.
Entrada: Fichero de tipo video, directorio hacia el cual copiar dicho archivo y el tamaño del archivo antes mencionado.
Resultados esperados: Se copia el archivo correctamente.

### 3.3.4 Pruebas de rendimiento

Para realizar las pruebasde velocidad se midieron los tiempos en realizar una copia en Nautilus y luego en la herramienta Ncopier. Los resultados son mostrados en la siguiente tabla.

| Operación | Herramienta | Cantidad de archivos | Tamaño total | Tiempo de duración | Media de uso de CPU | Media de memoria usada |
|---|---|---|---|---|---|---|
| 1 | Nautilus | 30 | 6.6 Gb | 06.44.06 | 20.4 | 20.4 |
| | Ncopier | | | 07.02.07 | 50 | 4.6 |
| 2 | Nautilus | 48 | 5.9 Gb | 05.49.50 | 10 | 20 |
| | Ncopier | | | 05.50.08 | 40 | 4.2 |
| 3 | Nautilus | 68 | 2.9 Gb | 03.14.05 | 10 | 19.3 |
| | Ncopier | | | 03.09.06 | 40 | 4.2 |
| 4 | Nautilus | 7 | 868.9 Mb | 00.50.09 | 6 | 16.9 |
| | Ncopier | | | 00.44.09 | 40 | 4 |

**Conclusiones del capítulo**

Durante el capítulo se describieron tareas enmarcadas en los flujos de trabajo: Implementación y Pruebas. Como parte del flujo Implementación se resolvió desarrollar un algoritmo propio que supliera la imposibilidad que presenta la biblioteca GIO (GNOME Input/Output por sus siglas en Inglés) para pausar/reanudar operaciones con ficheros (locales o remotos). Se describió el diagrama de despliegue mediante el modelo en aras de esclarecer la distribución de los nodos que intervienen en el mismo. Además se describió el estándar de codificación usado en el desarrollo de la herramienta. Esta fue probada por por medio de pruebas de Caja Negra, Caja Blanca y de Rendimiento. En las pruebas de Caja Negra fueron corregidos los errores pueños errores de interfaz. Las pruebas de Caja Blanca demostraron un correcto resultado y como resultados de las pruebas de Redimiento se logró ver cómo en operaciones con un tamaño total de hasta 5Gb y muchos ficheros Ncopier brinda una ligera rapidez sobre Nautilus; no así en operaciones con archivos de mayor tamaño. Se pudo constatar además que Ncopier usa mucha menos memoria RAM[7] sus grados de uso de CPU[8] son mucho mayores que Nautilus.

[7] RAM: Memoria de Acceso Aleatorio (por sus siglas en inglés)
[8] CPU: Unidad Central de Proceso (por sus siglas en inglés)

## Conclusiones

Ncopier es una herramienta desarrollada para resolver las problemáticas iniciales de la investigación referentes a las operaciones de mover y copiar ficheros en Nautilus. Dicha herramienta se integra al manejador de archivos deNova para brindaral usuario mayor control sobre las operaciones, ya sean locales o remotas. Ncopier implementa un algoritmo distinto al usado por la biblioteca GIO para poder pausar las operaciones y así brindarle al usuario mayor posibilidad de asignar prioridades a las mismas a través de las opciones: pausar, reanudar y cancelar.

Durante el desarrollo de la herramienta se ha cubierto completamente el ciclo de desarrollo de software de la metodología OpenUp a través de sus flujos de trabajos y se generaron varios artefactos definidos por la misma para el despliegue y entendimiento de la herramienta. La solución fue validada mediante pruebas de Caja Negra, Caja Blanca y de Rendimiento; demostrando así que la herramienta desarrollada cumple los objetivos planteados al inicio de la investigación.

**Recomendaciones**

1. Lograr que la herramienta sea capaz de gestionar una cola de operaciones.
2. Agregar la forma de poder ejecutar más de un hilo para cada copia siempre que sea posible.
3. Agregar una funcionalidad que permita continuar la operación luego de haber fallado la misma por motivos externos.

**Referencias Bibliográficas**


1. **Tanenbaum, Andrew S.; Woodhull, Albert S.** 2006. *Operating Systems Design and Implementation, Third Edition*
2. **Archivo o fichero informático.** [En línea] [Citado: 30 de Septiembre de 2012] http://www.ecured.cu/index.php/Archivo_%28Inform%C3%A1tica%29
3. **Tanenbaum, Andrew S.** 2009. *Sistemas Operativos Modernos, Tercera Edición.*
4. **Midnight Commander.** [En línea] [Citado: 30 de Septiembre de 2012] http://www.midnight-commander.org/
5. **Ultracopier.** [En línea] [Citado: 24 de Noviembre de 2012] http://ultracopier.first-world.info/
6. **Teracopy.** [En línea] [Citado: 12 de Enero de 2013] http://codesector.com/teracopy
7. **Supercopier.** [En línea] [Citado: 23 de Ocutbre de 2012] http://ultracopier.first-world.info/supercopier.html
8. **Copy Handler.** [En línea] [Citado: 13 de Enero de 2013] http://www.copyhandler.com/
9. **Visual Paradigm.** [En línea] [Citado: 01 de Febrero de 2013] http://www.visual-paradigm.com/
10. **Vala.** [En línea] [Citado: 25 de Septiembre de 2012] https://live.gnome.org/Vala
11. **Anjuta.** [En línea] [Citado: 26 de Septiembre de 2012] http://projects.gnome.org/anjuta/
12. **GTK.** [En línea] [Citado: 25 de Septiembre de 2012] http://www.gtk.org/
13. **Manual de referencia de Gtk3.** [En línea] [Citado: 27 de Septiembre de 2012] http://developer.gnome.org/gtk3/
14. **Jacobson, I.; Booch, G**. *2000. El proceso unificado de desarrollo de software*
15. **Metodologías de Desarrollo de Software.** [En línea] [Citado: 03 de Enero de 2013] http://www.ecured.cu/index.php/MetodologC3%ADas_de_desarrollo_de_software
16. **Gimson, Loraine.** *2012. Metodologías ágiles y desarrollo basado en conocimiento*
17. **Larman, C**. 2004. *Uml y patrones.*
18. **Pressman, Roger S**. 2001. *Ingeniería de Software. Un enfoque práctico.*
19. **Rsync.** [En línea] [Citado: 30 de Septiembre de 2012] http://rsync.samba.org/
20. **Manual de MMV.** [En línea] [Citado: 30 de Septiembre de 2012] http://manpages.ubuntu.com/manpages/precise/man1/mmv.1.html
21. **Manual de GCP.** [En línea] [Citado: 30 de Septiembre de 2012] http://dev.man-online.org/man1/gcp/
22. **Dolphin.** [En línea] [Citado: 30 de Septiembre de 2012] http://dolphin.kde.org/

**Bibliografía**


1. **Larsson, Alexander.** *Nautilus - Inside the Shell.*
2. **Tanenbaum, Andrew S.; Woodhull, Albert S.** *2006. Operating Systems Design and Implementation, Third Edition.*
3. **Tanenbaum, Andrew S.** 2009. *Sistemas Operativos Modernos, Tercera Edición.*
4. **Jacobson, Ivar; Rumbaugh, James; Booch, Grad.** *2009. El proceso unificado de desarrollo.*
5. **Vala.** https://live.gnome.org/Vala
6. **Ecured.** http://www.ecured.cu
7. **Anjuta.** http://projects.gnome.org/anjuta/
8. **Documentación oficial de Vala.** http://valadoc.org/
9. **Manual de referencia de Gtk3.** https://developer.gnome.org/gtk3/
10. **Proyecto GNOME.** https://www.gnome.org/
11. **Nautilus.** https://live.gnome.org/Nautilus
12. **Lista de correo de Nautilus.** https://mail.gnome.org/archives/nautilus-list/
13. **Comparación de ultracopier, supercopier, teracopy y copy handler.** http://ultracopier-es.first-world.info/articulos/ultracopier-supercopier-teracopy-copyhandler.html
14. **Pressman, Roger S**. 2001. *Ingeniería de Software. Un enfoque práctico.*

**Glosario de Términos**

**Software:** soporte lógico de un sistema informático comprende el conjunto de los componentes necesarios que hacen posible la realización de tareas específicas en computadoras.
**UNIX:** sistema operativo multitarea y multiusuario comenzado a desarrollar en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.

**GNU/Linux:** sistema operativo que resulta de la combinación entre un conjunto de herramientas GNU y el núcleo Linux.

**Distribución GNU/Linux:** distribución de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios**.**

**Manejador de Archivos:** aplicación informática que provee acceso a archivos y facilita el realizar operaciones con ellos, como copiar, mover o eliminar archivos.

**SAMBA:** implementación libre del protocolo de archivos compartidos de Microsoft Windows (antiguamente llamado SMB, renombrado recientemente a CIFS) para sistemas de tipo UNIX. De esta forma, es posible que ordenadores con GNU/Linux, Mac OS X o Unix en general se vean como servidores o actúen como clientes en redes de Windows.

**FTP:** Protocolo de Transferencia de Archivos. Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Protocolo de Control de Transmisión, por sus siglas en Inglés) independientemente del sistema operativo utilizado en cada equipo.

**GNU GPL:** licencia creada por la Fundación de Software Libre en 1989 (la primera versión), orientada principalmente a proteger la libre distribución, modificación y uso de software.

## Anexos

## Anexo 1. Diagrama de Clases del Sistema

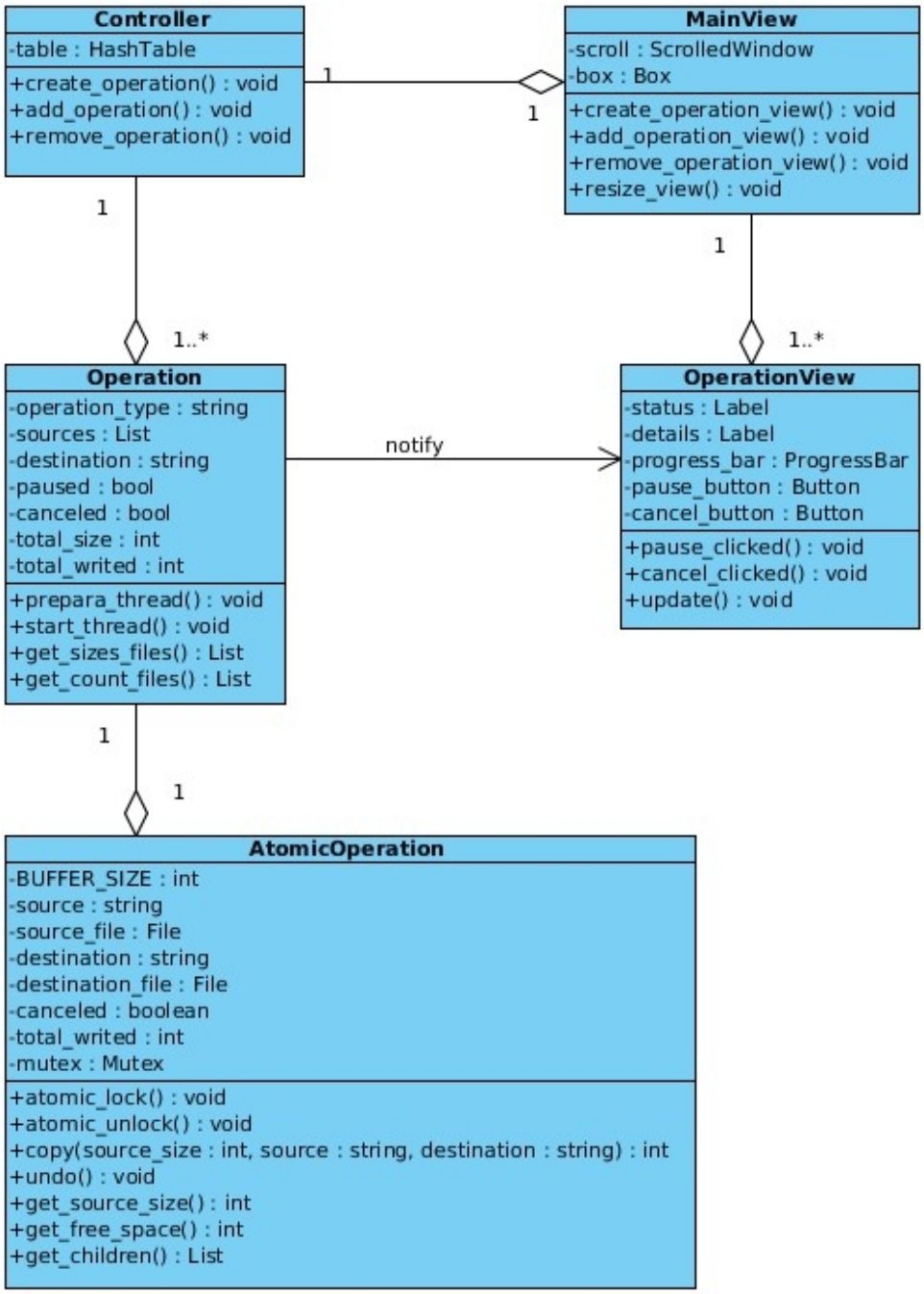


*Figura 5: Diagrama de Clases del Sistema*

## Anexo 2. Diagramas de Procesos del Negocio

## Anexo 2.1 Copiar Fichero

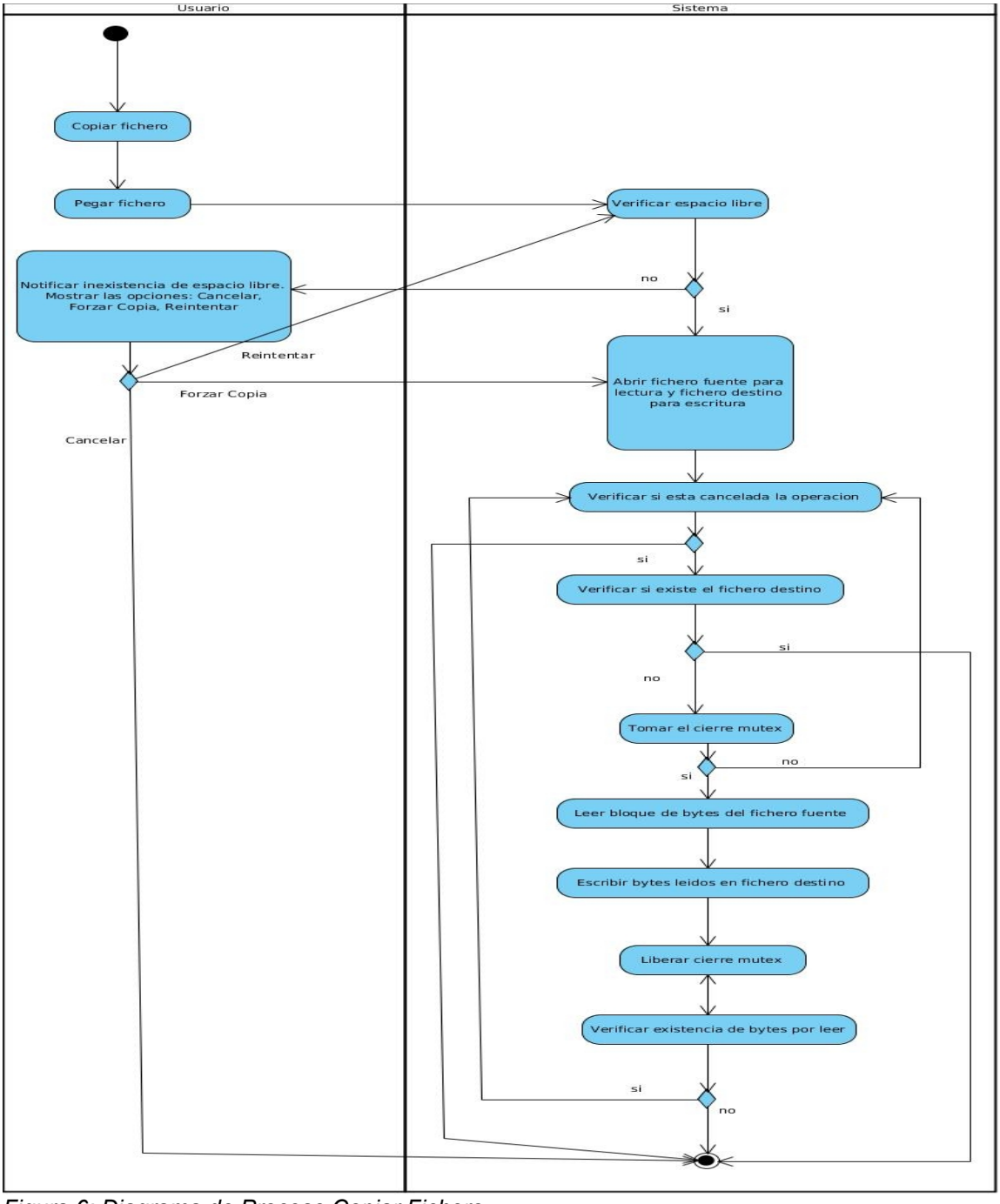


*Figura 6: Diagrama de Proceso Copiar Fichero*

## Anexo 2.2 Mover Fichero

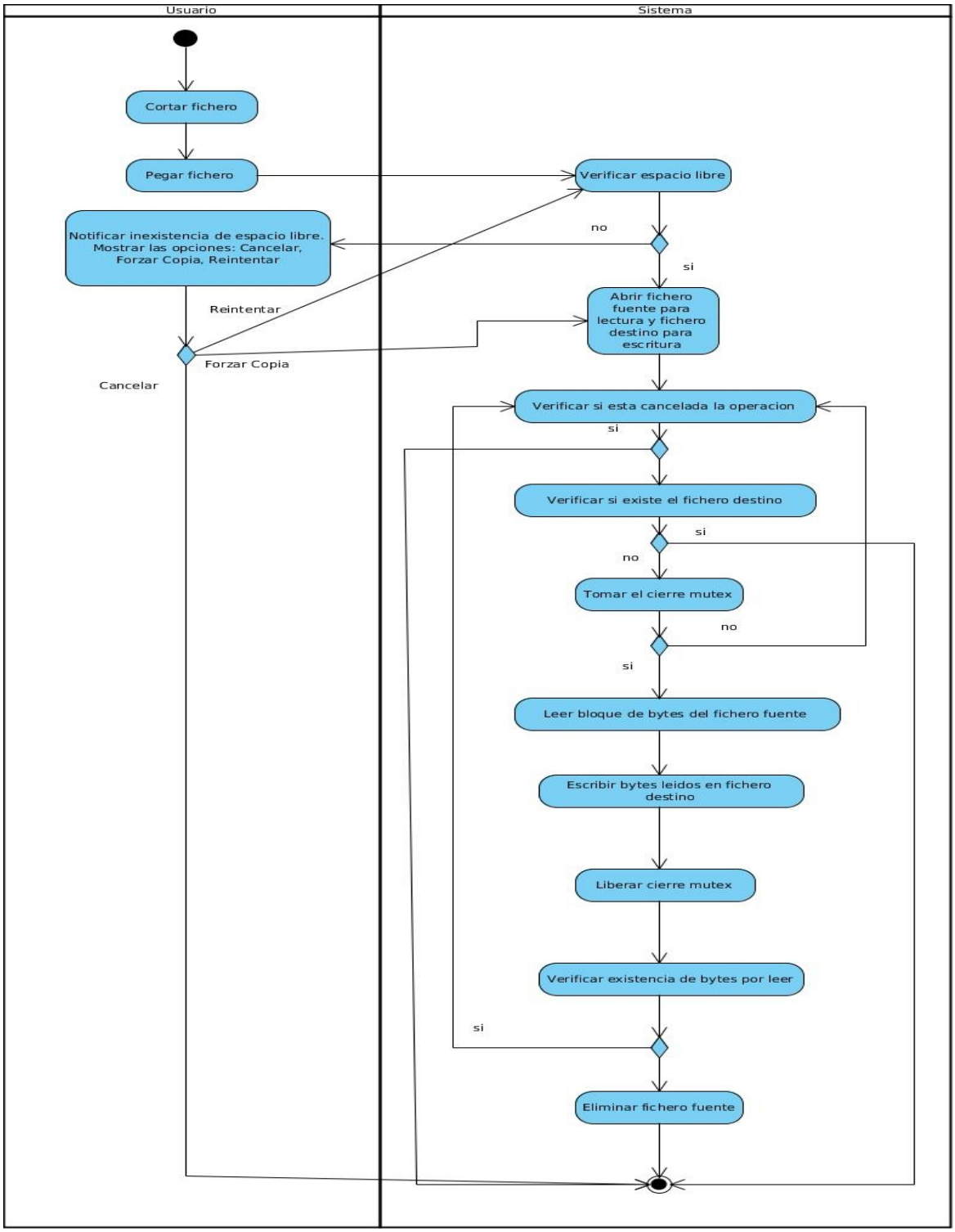


*Figura 7: Diagrama de Proceso Mover Fichero*

## Anexo 2.3 Pausar Operación

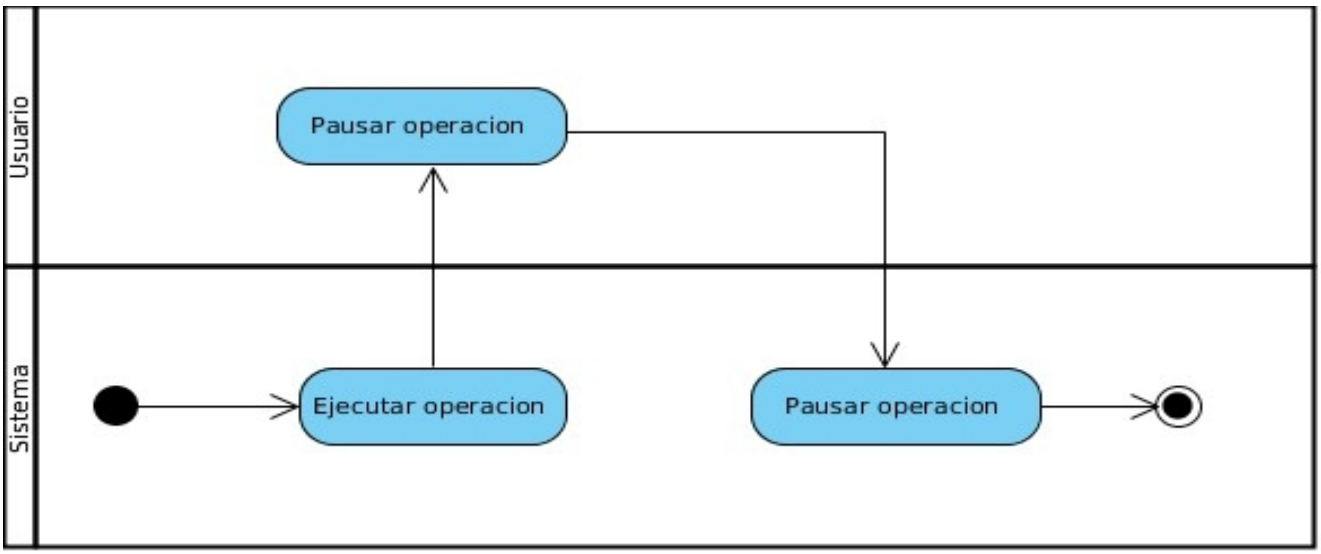


*Figura 8: Diagrama de Proceso Pausar Operación*

## Anexo 2.4 Reanudar Operación.

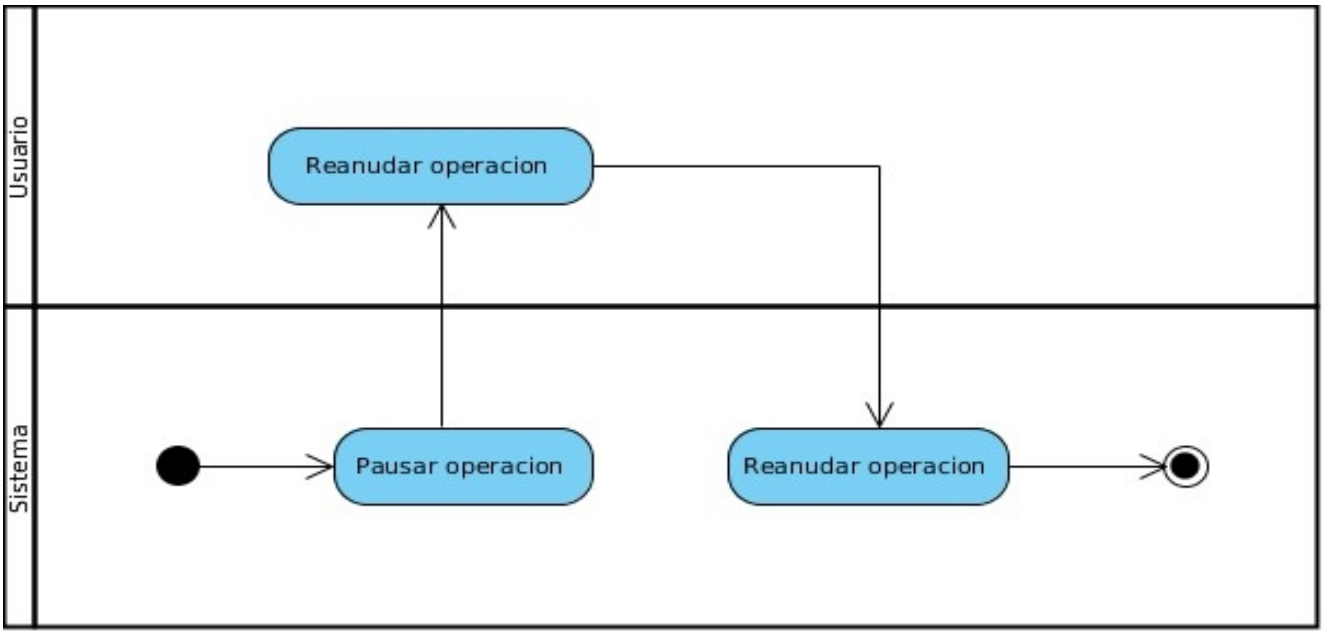


*Figura 9: Diagrama de Proceso Reanudar Operación*

## Anexo 2.5 Cancelar Operación

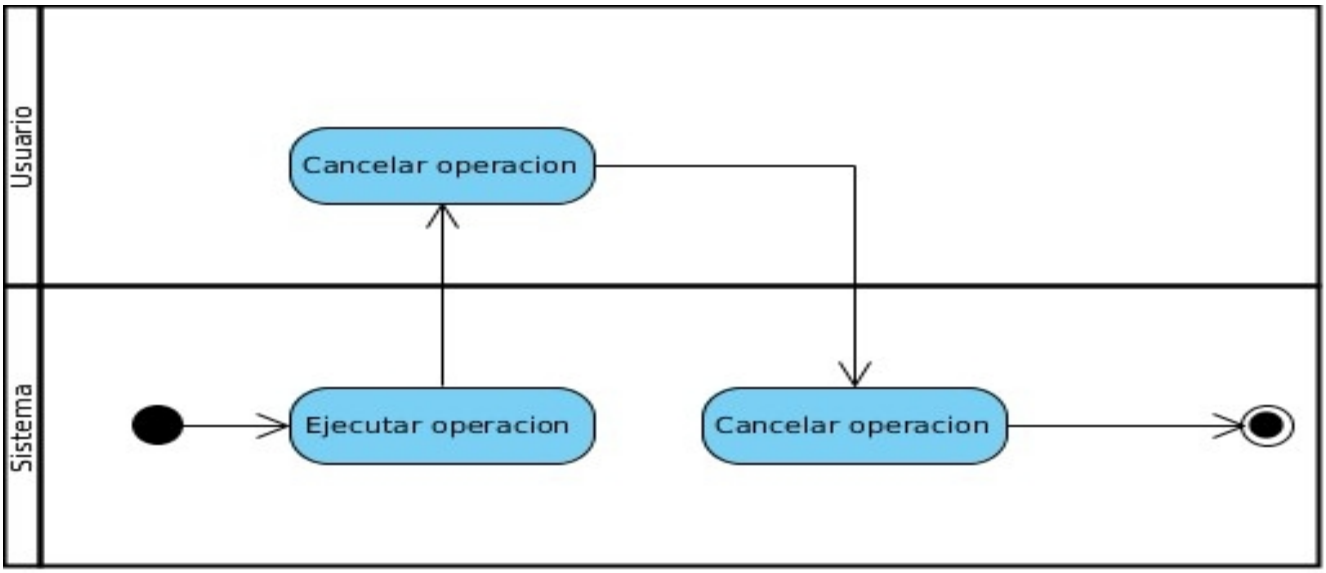


*Figura 10: Diagrama de Proceso Reanudar Operación*

## Anexo 3. Diagrama de Componentes

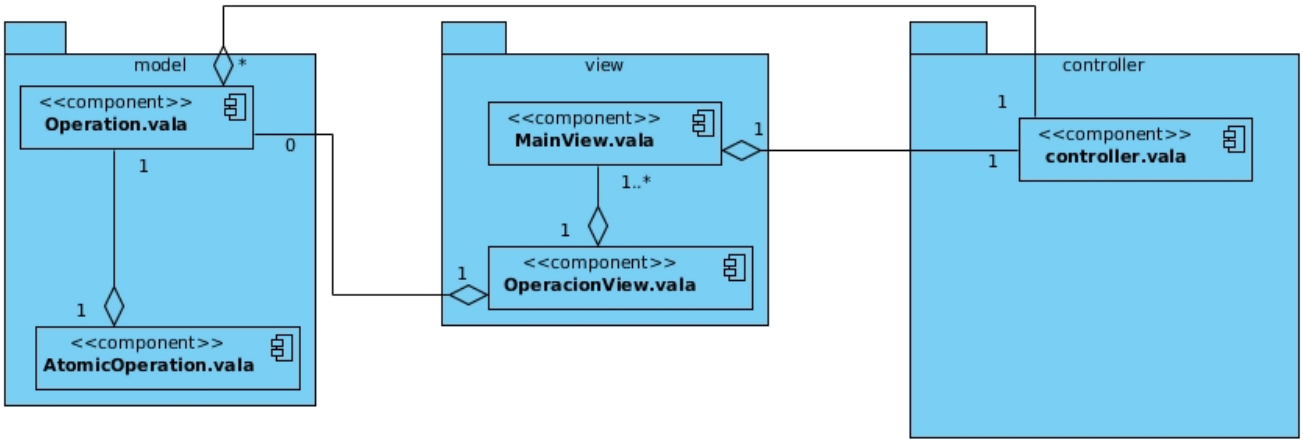


*Figura 11: Diagrama de Componentes*

## Anexo 4. Diagramas de Secuencia

## Anexo 4.1 Copiar Fichero

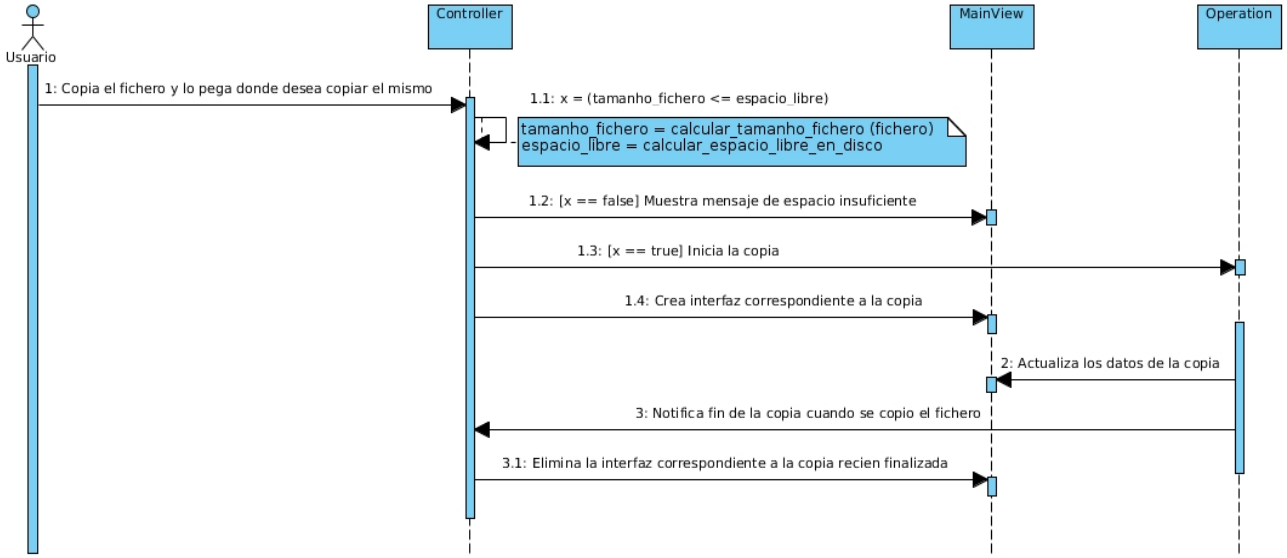


*Figura 12: Diagrama de Secuencia Copiar Fichero*

## Anexo 4.2 Mover Fichero

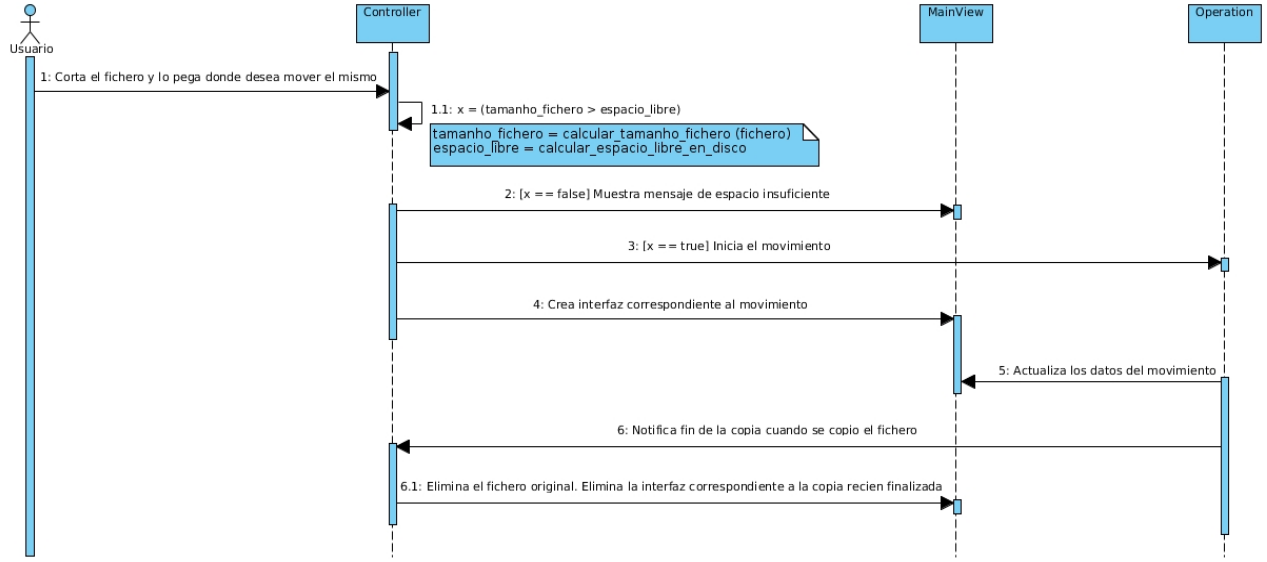


*Figura 13: Diagrama de Secuencia Mover Fichero*

## Anexo 5. Diagrama de Despliegue

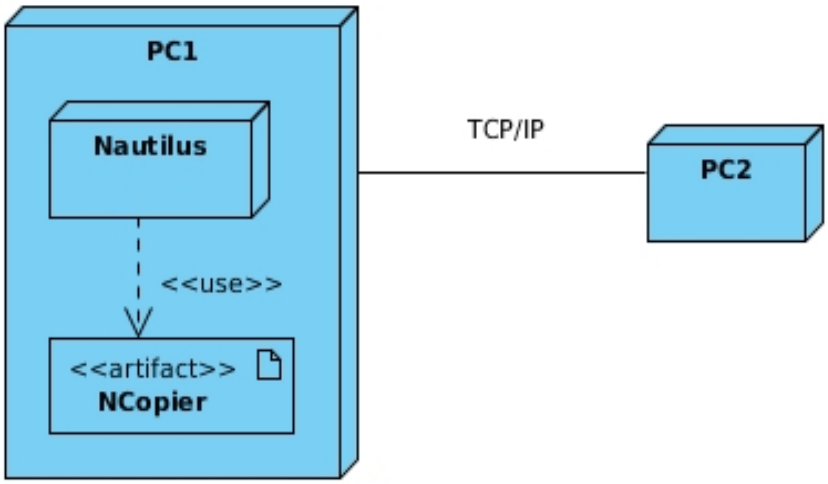


*Figura 14: Diagrama de Despliegue*