ICD-CHAPIN: INTERPRETE DE COMANDOS PARA DIAGRAMAS NASSI-SHNEIDERMAN (N-S).

JHON JAIRO BASTIDAS ESTUPIÑAN
EDINSON ARIEL CEBALLOS HERNANDEZ
PAOLA GRISEL ZAMORA MANTILLA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2008

ICD-CHAPIN: INTERPRETE DE COMANDOS PARA DIAGRAMAS NASSI-SHNEIDERMAN (N-S).

JHON JAIRO BASTIDAS ESTUPIÑAN
EDINSON ARIEL CEBALLOS HERNANDEZ
PAOLA GRISEL ZAMORA MANTILLA

Presentado como requisito parcial para
optar al título de Ingeniero de Sistemas.

DIRECTOR
ANIVAR NESTOR CHAVES TORRES.
Ingeniero de sistemas.

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2008

Nota de aceptación

_____
_____
_____
_____
_____
_____

_____
Firma del Jurado

_____
Firma del Jurado

_____
Firma del director del proyecto

San Juan de Pasto, 4de marzo de 2008

“Las ideas y conclusiones aportadas en la tesis de grado son responsabilidad exclusiva de los autores”

Artículo 1. Del acuerdo No. 324 del 11 de Octubre de 1966, emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

AGRADECIMIENTOS

Agradecemos a Dios por habernos dado toda la ayuda y las ganas de salir adelante en nuestro proyecto, a quien entregamos nuestra profesión. También agradecemos a nuestros padres y familiares por el gran apoyo brindado. Finalmente, agradecemos a nuestros docentes y compañeros, en especial a nuestro asesor Ing. Anivar Chaves Torres.

CONTENIDO

LISTA DE ANEXOS

pág.

GLOSARIO

ALGORITMO: Se sabe que la palabra algoritmo se dio en honor del matemático persa del siglo IX, *Khowârizmî*. Con éste término se hace referencia a un conjunto de reglas, ordenadas de forma lógica, para desarrollar un cálculo o para solucionar un problema, ya sea de forma manual o utilizando una máquina.

DECISIÓN: en ocasiones es necesario ejecutar una acción dependiendo de una condición.

DEMO: Demostración didáctica del funcionamiento del programa.

DEPURAR: detectar, localizar y corregir errores en tiempo de interpretación de un diagrama N-S.

DFD: Diagrama de Flujo de Datos.

GPL: *General Public License*, es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

ICD-CHAPIN: Intérprete de comandos de diagramas de Chapin o conocidos como N-S.

INTÉRPRETE: Reconoce las diferentes estructuras de los diagramas N-S para su posterior evaluación.

ITERACIÓN: si necesita repetir una o varias operaciones dependiendo de una condición, se debe realizar una iteración.

JavaCC (*Java Compiler Compiler*): es un metacompilador o generador de parsers es una herramienta que, a partir de la especificación de un lenguaje, construye un programa o analizador que es capaz de reconocer secuencias o elementos de dicho lenguaje.

JEP: *Java Math Expression Parser* es un paquete de Java para analizar y evaluar expresiones matemáticas.

JVM: *Java Virtual Machine*. Máquina virtual de Java.

LIENZO: lugar donde se dibujan los diagramas N-S

LL (k): Tipo de gramática que utiliza el parser JEP.

LINUX: Sistema Operativo de distribución libre.

NASSI - SHNEIDERMAN: Son los apellidos de los creadores de esta forma de especificación de algoritmos, sus nombres completos son: Isaac Nassi y Ben Shneiderman. Los diagramas que ellos propusieron se reconocen con sus apellidos o con sus iniciales N – S, o también como diagramas de Chapin.

N-S: Nassi Shneiderman.

OPERACIÓN: un algoritmo o diagrama está compuesto de muchas operaciones, todos los comandos (Leer, escribir, asignación) y las estructuras de control (decisiones, iteraciones) son operaciones.

PARSER: Compilador o analizador.

PSEUDO-CÓDIGO: Representación de algoritmos en lenguaje natural.

SPLASH: ventana de presentación inicial del programa.

UBUNTU: Sistema operativo basado en el núcleo de Linux.

UML: Lenguaje de Modelado Unificado. Utilizado en el análisis y modelación de un sistema a desarrollar.

UNIX: Sistema operativo.

UTF8: (8-bit Unicote Transformation Format) es una norma de transmisión de longitud variable para caracteres codificados utilizando Unicode, creada por Rob Pike y Ken Thompson. UTF-8 usa grupos de bytes para representar el estándar de Unicode para los alfabetos de muchos de los lenguajes del mundo. Es especialmente útil para la transmisión sobre sistemas de correo de 8 bits.

XML: (*Extended Markup Language*), lenguajes de marcado de hipertexto.

RESUMEN

El diseño de algoritmos es fundamental para crear soluciones a los distintos problemas que el medio presenta, por esta razón es de suma importancia crear herramientas que faciliten una correcta evaluación e interpretación de estos.

El software ICD-CHAPIN se presenta como una herramienta intérprete de comandos para diagramas Nassi- Shneiderman, con el fin de promover el uso de dichos diagramas, que se muestran como una alternativa con ventajas significativas frente a las otras técnicas.

El uso de la plataforma Java junto con la metodología aplicada para el desarrollo, y UML como lenguaje de modelado facilitan que esta herramienta sea fiable por su portabilidad, facilidad de mantenimiento y evolución.

ABSTRACT

The design of algorithms is fundamental for create solutions to the several problems that the mid to present, by this reason is of much important to create tools that to facilitate a correct evaluation and interpretation of these.

The software ICD-CHAPIN is presents like a tool interpreter of commands for Nassi- Shneiderman diagrams, with the purpose of to promote the use of them diagrams, that is shows with an alternative with significant advantages in front of others techniques.

The use of the Java platform next to with the methodology applied for the development and UML like language of modelling to facilitate that this tool is reliable for its portability, facility of maintenance and evolution.

INTRODUCCIÓN

Teniendo en cuenta la importancia del desarrollo de algoritmos para la solución de problemas informáticos, es adecuada la aplicación de los distintos métodos de representación y verificación de estos, como también contar con herramientas informáticas apropiadas para este fin. Para la representación de algoritmos se tienen varias técnicas como son: pseudocódigo, diagrama de flujo y diagramas de Nassi-Shneiderman (N-S). En el contexto local se observa una inclinación por la utilización del pseudocódigo y al interior de la universidad de Nariño, también es común, la representación mediante diagramas de flujo, pero se ha dado poca importancia a los diagramas N-S, quizá por carecer de herramientas para su diseño y verificación.

Considerando lo anterior, el proyecto está enfocado a crear un software para construcción e interpretación de diagramas N-S, con el fin de promover la utilización de está metodología de diseño de algoritmos, que tiene ventajas significativas con respecto a las otras dos metodologías mencionadas.

Aplicando la metodología orientada a objetos, junto con Java y el lenguaje de modelado UML, se ha buscado que el software sea fiable, en cuanto a su capacidad de evolución, mantenimiento, y portabilidad.

1. TEMA

1.1 TITULO

ICD-CHAPIN: Intérprete de Comandos para Diagramas de Chapin (N-S).

1.2 LÍNEA DE INVESTIGACIÓN

El proyecto se orienta a la construcción de un editor e interprete de comandos para diagramas N-S o Chapin, para lo cual se apoya en temáticas como: Fundamentos de programación, Lógica matemática, Teoría de lenguajes y Estructuras de información. En consecuencia se clasificó el desarrollo de este proyecto en la línea de investigación **Sistemas Computacionales**.

1.3 ALCANCE Y DELIMITACIÓN

El software implementa las estructuras básicas de programación como son: estructuras lineales, selectivas e iterativas, lo que permite tener un software completo de apoyo a la construcción e interpretación de diagramas N-S.

En la construcción de algoritmos se utilizan estructuras selectivas e iterativas anidadas y este anidamiento no puede hacerse de forma ilimitada y teniendo en cuenta, además, la dificultad que implica mostrar en pantalla las estructuras anidadas se propone implementar un máximo de 10 niveles de anidamiento.

El desarrollo de este tipo de herramienta tiene como beneficiarios a los estudiantes, profesores y personas que requieran la aplicación de dichos diagramas, los cuáles están de alguna u otra forma relacionados con esta rama y además vean la necesidad de corregir errores y observar el correcto funcionamiento de los pasos realizados para la solución de un determinado problema

2. DESCRIPCIÓN DEL PROBLEMA

2.1 PLANTEAMIENTO DEL PROBLEMA

La investigación preliminar con estudiantes de los diferentes semestres de Ingeniería de Sistemas (ver Anexo B) reveló que los diagramas N-S son conocidos por el 55% de los encuestados, sin embargo solo el 15% los utiliza para la representación de soluciones algorítmicas. Se indagó sobre la existencia de alguna herramienta para el diseño e interpretación de este tipo de diagramas y la totalidad de los estudiantes coincidieron en que no se conoce ninguna, a la vez, se conoció que el 70% de los estudiantes consideran necesario la implementación de una herramienta para trabajar con estos diagramas.

También se encontró que en la enseñanza de la asignatura fundamentos de programación, la cual está enfocada al diseño de algoritmos, sólo se utiliza Diagrama de flujo y pseudocódigo, dejando a un lado los diagramas N-S (técnica que combina la representación textual del pseudocódigo, con la representación gráfica del diagrama de flujo), la causa principal de que no se utilice este tipo de notación es que no existe una herramienta para su diseño y ejecución, mientras que ésta si existe para los diagramas de flujo: el DFD.

Al no existir una herramienta de software que le permita diseñar e interpretar los diagramas N-S, los estudiantes no pueden realizar prácticas en las que verifiquen el funcionamiento de sus algoritmos o identificar los errores, por lo cual prefieren utilizar otras técnicas de representación. Esto hace que se presente un desconocimiento generalizado de los diagramas N-S.

2.2 FORMULACIÓN DEL PROBLEMA

¿Cómo solucionar la falta de una herramienta software que interprete diagramas Nassi-Shneiderman (N-S), para que de esta manera se pueda tener otra alternativa diferente de representar algoritmos?

2.3 SISTEMATIZACIÓN DEL PROBLEMA

- ¿Cómo manejar las diferentes estructuras de programación en diagramas N-S?

- ¿Cómo implementar una interfaz de usuario para la construcción de diagramas N-S?

- ¿Cómo interpretar las instrucciones algorítmicas y ejecutarlas?

- ¿Cómo almacenar los diagramas que genera el usuario?

- ¿Cómo validar los símbolos y las instrucciones utilizados en los diagramas N-S?

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Desarrollar una herramienta software para la construcción e interpretación de diagramas Nassi-Shneiderman o de Chapin.

3.2 OBJETIVOS ESPECÍFICOS

- Implementar las diferentes estructuras básicas de programación: secuenciales, de decisión e iterativas.

- Construir una interfaz gráfica que permita el diseño y edición de diagramas N-S.

- Construir un módulo de interpretación y ejecución de las diversas instrucciones a implementar.

- Construir una gramática propia para los símbolos e instrucciones de los diagramas N-S.

- Definir el formato de archivo para almacenar y recuperar los diagramas.

4. MARCO TEÓRICO

4.1 ALGORITMO

Se sabe que la palabra algoritmo se dio en honor del matemático persa del siglo IX, Khowârizmî. Con éste término se hace referencia a un conjunto de reglas, ordenadas de forma lógica, para desarrollar un cálculo o para solucionar un problema, ya sea de forma manual o utilizando una máquina. Los algoritmos están, con mayor o menor complejidad, en todas las actividades desarrolladas por el hombre y han sido utilizados por todos, infinidad de veces.

Desde los primeros años de escolaridad se trabaja con algoritmos, en especial en el campo de las matemáticas. Los métodos utilizados para sumar, restar, multiplicar y dividir son algoritmos que cumplen perfectamente las características de precisión, finitud, definición y eficiencia. Para que la solución de un problema sea llevada hasta un lenguaje de programación, los pasos expresados en el algoritmo deben ser lo más detallados posible, de manera que cada uno de ellos implique una operación trivial; es decir, que los pasos no impliquen procesos que requieran de una solución algorítmica. Si el problema que se desea solucionar es muy grande o complejo, es recomendable dividirlo en tareas que se puedan abordar independientemente y que resulten más sencillas de solucionar. A esto se le llama diseño por módulos.

4.1.1 Características de un Algoritmo

Un algoritmo debe tener al menos las siguientes características:

*** Ser preciso:** esto significa que las operaciones o pasos del algoritmo deben desarrollarse en un orden estricto, ya que el desarrollo de cada paso debe obedecer a un orden lógico.

*** Ser definido:** Ya que en el área de programación, el algoritmo se desarrolla como paso fundamental para desarrollar un programa, el algoritmo debe estar plenamente definido; esto es que el resultado depende estrictamente de los datos suministrados. Si se ejecuta con un mismo conjunto de datos de entrada, el resultado será siempre el mismo.

*** Ser finito:** esta característica implica que el número de pasos de un algoritmo, por grande y complicado que sea el problema que soluciona, debe ser limitado, debe llegar a un final. Para hacer evidente esta característica, en la representación de un algoritmo siempre se incluyen los pasos inicio y fin.

*** Presentación formal:** para que el algoritmo sea entendido por cualquier persona interesada es necesario que se exprese en alguna de las formas comúnmente aceptadas; por el contrario puede no ser muy útil ya que solo lo entenderá quien lo diseñó. Las formas de presentación de algoritmos son: el pseudocódigo, diagrama de flujo y diagramas de Nassi-Shneiderman, entre otras.

*** Corrección:** el algoritmo debe ser correcto, es decir debe satisfacer la necesidad o solucionar el problema para el cual fue diseñado. Para esto es necesario ponerlo a prueba; a esto se le llama prueba de escritorio.

*** Eficiencia:** hablar de eficiencia o complejidad de un algoritmo es evaluar los recursos de cómputo que requiere para almacenar datos y para ejecutar operaciones frente al beneficio que ofrece. En cuanto menos recursos se requieran el algoritmo será más eficiente. [1]

El primer paso en el diseño de un algoritmo es conocer la temática a tratar (Definición del problema), el segundo será pensar en las actividades a realizar (Análisis del problema: Datos de entrada, proceso y datos de salida) y el orden en que deben ejecutarse para lograr el objetivo (Diseño de la solución), el tercero y no menos importante es la presentación formal (Prueba y documentación).

4.2 DIAGRAMAS DE NASSI – SHNEIDERMAN

El diagrama N-S o también conocido como diagrama de Chapin, son tambien llamados Estructogramas o diagramas de cajas, fueron desarrollados en 1972 por Isaac Nassi y Ben Shneiderman, mientras participaban en una conferencia de la ACM sobre programación. Nassi y Shneiderman propusieron su trabajo denominado diagramas de flujo estructurado *(structured flowcharts)* para ser publicado en la revista *Communications of the ACM.* Nassi y Shneiderman decidieron reenviar su artículo a la ACM, ahora *ACM SIGPLAN Notices*, donde fue

[1] CHAVES TORRES, Anivar.Algoritmos Pseudocodigos, Diagramas de Flujo y Diagramas N-S. San Juan de Pasto, Colombia. Multigráfico impresores.2004.297p.

publicado en 1973, bajo el título *"Flowchart Techniques for Structured Programming"* (SIGPLAN Notices 8, 8 Agosto de 1973).

Los diagramas Nassi-Shneiderman tuvieron bastante acogida en Alemania, donde se les conoce como *Struktogramme*, y estandarizados en 1985 mediante la norma DIN 66261.

Su principal ventaja es que adopta la programación estructurada y utiliza un número limitado de símbolos, el primer símbolo es un cuadro, que se usa para representar cualquier proceso en el programa, el segundo símbolo es un triángulo divisor de columnas, que representa una decisión, la forma más básica de una decisión, "cierto" o "falso", y el tercer símbolo es el de un cuadro dentro de otro cuadro, usado para mostrar que se realiza una iteración, de tal forma que el diagrama de flujo ocupa menos espacio y puede leerse con mayor facilidad. [1]

Estos diagramas son una técnica de especificación de algoritmos que combina la descripción textual, propia del pseudocódigo, con la representación gráfica del diagrama de flujo. El diagrama N-S cuenta con un conjunto limitado de símbolos para representar los pasos del algoritmo, por ello se apoya en expresiones del lenguaje natural; sin embargo, dado que el lenguaje natural es muy extenso y se presta para la ambigüedad, solo se utiliza un conjunto de palabras, a las que se denomina palabras reservadas.

Las palabras reservadas más utilizadas son:
Inicio, Fin, Leer, Escribir
Mientras, Repita Hasta, Para
Incrementar, Decrementar, Hacer, Función
Entero, Real, Caracter, Cadena
Lógico, Retornar

Los símbolos utilizados en el diagrama de Chapin corresponden a cada tipo de estructura. Dado que se tienen tres tipos de estructuras, se utilizan tres símbolos. Esto hace que los procesos del algoritmo sean más fáciles de representar y de interpretar (Ver Anexo A.)

[1] JOYANES AGUILAR, Luís. Fundamentos de Programación, Algoritmos y estructuras de datos. Santa Fe de Bogotá D.C. McGraw-Hill. 1998. 714 p.

En la actualidad hay escasas herramientas para trabajar diagramas N-S, pero existen herramientas CASE (Ingeniería del software asistida por computador), que necesitan de los diagramas N-S para su funcionamiento, entre ellas se destacan XperCASE y EasyCode.

Investigación de desarrollo de software para trabajar los diagramas N-S existen muy pocos y poca información en la red mundial, pero en Enero del 2007, se publicó un trabajo de investigación de la universidad de Valparaíso en Chile, donde fue dirigido por Ph.D. Franco Guidi-Polanco y por M.Sc. Matilde Castillo.

4.3 UML (LENGUAJE DE MODELADO UNIFICADO)

El Lenguaje de Modelado Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software.[1] UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reutilizables.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COADYOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

UML es ahora un estándar, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro ramo.

[1] BOOCH, Grady & RUMBAUGH, James & JACOBSON, Ivar. El lenguaje unificado de modelado UML. Madrid, España. Addison Wesley Iberoamericana.1999. 464 p.

4.3.1 Artefactos de UML: en todos los ámbitos de la ingeniería se construyen modelos, en realidad, simplificaciones de la realidad, para comprender mejor el sistema que se va a desarrollar: los arquitectos utilizan y construyen planos (modelos) de los edificios, los grandes diseñadores de coches preparan modelos en sistemas CAD/CAM con todos los detalles y los ingenieros de software deberían igualmente construir modelos de los sistemas software.

Para la construcción de modelos, hay que centrarse en los detalles relevantes mientras se ignoran los demás, por lo cual con un único modelo no se tiene bastante. Varios modelos aportan diferentes vistas de un sistema los cuales ayudan a comprenderlo desde varios frentes. Así, UML recomienda la utilización de varios diagramas para representar las distintas vistas de un sistema.

Uno de los artefactos mas utilizados para el desarrollo del sistema es:

- **Casos de Uso.** Un caso de uso es un documento narrativo que describe la secuencia de eventos cuando uno o más actores utilizan un sistema para llevar a cabo un proceso. Un caso de uso cuenta una historia del uso del sistema. Para el desarrollo del sistema propuesto se utiliza la siguiente plantilla para la captura de los respectivos requerimientos funcionales:

Caso de uso	
Actor principal	
Precondiciones	✍
Poscondiciones	✍
Flujo básico	
1.	
Flujos alternativos	
Observaciones	

- **Actores:** es un usuario del sistema que tiene un rol particular (en realidad un actor puede ser un sistema externo que es como un usuario desde el punto de vista de un sistema: el punto crucial es que se está diseñando alguien o algo externo a él, que interactúa con el sistema y que puede hacerle peticiones).[1]

[1] PERDITA STEVENS, Rob Pooley. Utilización de UML en Ingeniería del software con objetos y componentes. Madrid, España. Addison Wesley. 2002. 300 p

- **Diagramas de Casos de Uso:** un caso de uso es una descripción de las acciones del sistema desde el punto de vista de las acciones de un usuario. Para los desarrolladores del sistema, esta es una herramienta valiosa ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista del usuario. Esto es importante si la finalidad es crear un sistema que pueda ser utilizado por la gente en general, (no solo por expertos en computación).

- **Diagramas de Secuencia:** el Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista '*business*' del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.[1]

Típicamente uno examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se tiene modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede detallar esos pasos para descubrir qué objetos son necesarios para que se pueda seguir los pasos.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria.

[1] BOOCH, Grady & RUMBAUGH, James & JACOBSON, Ivar. El lenguaje unificado de modelado UML. Madrid, España. Addison Wesley Iberoamericana.1999. 464 p

5. JUSTIFICACIÓN

Teniendo en cuenta que toda solución de un problema implica desarrollar un algoritmo eficiente que cumpla con los objetivos planteados, es de suma importancia aprender a manejar las herramientas de solución para dichos algoritmos.

El desarrollo de este proyecto presenta beneficios académicos, puesto que ayuda a la representación de un algoritmo en un diagrama N-S por medio de un software, ya que actualmente esto se realiza en papel. Al desarrollar una herramienta se pretende mejorar la representación de una solución a un determinado problema y además con esto se genera un importante y novedoso recurso educativo.

Al realizar una observación en el entorno y plantear el problema, se observó la verdadera importancia que representa para la sociedad, en especial para los estudiantes de primer semestre de Ingeniería de Sistemas, quienes requieren tener las bases fundamentales sobre la construcción de algoritmos en sus diferentes técnicas de representación, logrando generar en ellos mayor interés, por el dinamismo en el diseño de los algoritmos que cuando se realizan simplemente en papel.

En cuanto al valor teórico y metodológico que este proyecto aporta, este trabajo de investigación puede dar las pautas para la realización de trabajos futuros en la misma línea de investigación en donde se puedan basar, o posiblemente conocer la herramienta para poder ampliar sus funciones (desarrollar nuevas versiones).

Al terminar del proyecto ICD-Chapin se pone a disposición de la sociedad el nuevo producto de manera gratuita y con código abierto, lo cual redunda en reconocimiento para la Universidad de Nariño en general y de la facultad y el programa en particular.

6. ANTECEDENTES

Esta investigación toma como referencia el proyecto desarrollado por Fabián Cárdenas Varela, Nelson Castillo Izquierdo y Eduardo Daza Castillo, estudiantes de la Universidad del Magdalena que obtuvo el primer puesto con el trabajo de investigación y desarrollo "Editor e Intérprete de Algoritmos Representados en Diagramas de Flujo" en el Segundo Evento de Investigación y Divulgación Tecnológica en el Área de Sistemas y Computación, realizado a nivel nacional por la universidad Cooperativa de Colombia (Santa Marta, octubre de 1997). Este software tiene como características principales: su interfaz gráfica facilita en gran medida la creación de diagramas de flujo para la representación de algoritmos que solucionan problemas por computadora. Dichos diagramas pueden ser guardados en disco, recuperados de disco y pueden ser impresos en diferentes tamaños sin importar el tipo de impresora. En un disco de 3 1/2'' de alta densidad se pueden almacenar más de 360 algoritmos de mediana complejidad como archivos de extensión *.dfd.

El programa Dfd en su totalidad, fue realizado en lenguaje C++ utilizando programación orientada a objetos y corre bajo Windows a 32 bits. El software Dfd incluye un archivo de ayuda para Windows, que provee un fácil acceso a información necesaria para trabajar con Dfd, documentación sobre los símbolos u objetos que conforman los diagramas, las funciones, los operadores, los mensajes de error y demás aspectos relacionados con Diagramas de flujo de datos.

Otra referencia de esta investigación, encontrada en enero del año 2007, es el software Nessi que es un Editor de diagramas Nassi-Shneiderman , version: 1.0.6.6.21, de código abierto bajo la licencia GNU(*General Public License),* proyecto de tésis de André Michalland, Escuela de Ingeniería Industrial (2006), guíado por el Dr. Franco Guidi y M.Sc. Matilde Castillo, Escuela de Ingeniería Industrial, Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile.

El Nessi permite dibujar diagramas de Nassi Shneiderman, también ofrece la posibilidad de interpretar y hacer seguimiento de los diagramas.

7. METODOLOGÍA

La metodología de investigación que se utilizó para dar solución al problema planteado fue el modelo Orientado a Objetos y conjuntamente la Programación Orientada a Objetos, que permite el envío de mensajes entre objetos, de tal forma que se entiende fácilmente la interacción entre ellos, aplicando el desarrollo por módulos y la reutilización de código, además de la abstracción (Representar la realidad como un objeto), encapsulamiento (Proteger la información de un objeto), herencia (Acceso a la información de otras clases) y polimorfismo (Varias formas de un mismo método), que son características propias del modelo Orientado a Objetos.

Por otro lado, se utilizó como herramienta de modelado, la herramienta denominada UML (Lenguaje Modelado Unificado), que permite generar diseños de sistemas capturando las ideas en forma aceptada y fácil de comprender, de manera que puedan ser comunicadas a otras personas sin ambigüedad. Se utilizó UML con el fin de tener una idea general del sistema y un correcto análisis para su posterior implementación, en otras palabras tener un lenguaje común y organizado para el desarrollo del proyecto.

Los pasos que se realizaron en cuestión del proyecto son los siguientes:

En primera instancia se realizó el análisis de requerimientos que son la base para la construcción del sistema, es decir, determinar las necesidades más importantes del usuario en el contexto donde se aplica el software.

Luego de identificar los requerimientos esenciales, se tiene la etapa de análisis, en la que se logró mayor claridad sobre lo que desea el usuario y se construyó un modelo de análisis que sirvió para comunicar y orientar el desarrollo del proyecto, estableciendo las funciones del sistema, con ello se dio paso a la realización de la descripción de los casos de uso y el correspondiente Modelo de casos de Uso.

El siguiente paso consistió en identificar los objetos y la relación entre ellos, para luego obtener un diagrama de objetos de manera más detallada que se da en el sistema a desarrollar y que es el que generalmente tiene sentido para el usuario.

Teniendo los objetos, estos se clasificaron para formar las correspondientes clases (Se identificó los sustantivos del enunciado del problema y determinó si son clases del modelo del mundo, se identificó clases desde el punto de vista de: la información, funcional (casos de uso) y de sus estados) y se realizó finalmente el diagrama completo de clases con sus respectivas relaciones o interacción existente.

En la etapa de diseño preeliminar se creó paquetes (Se combinó clases fuertemente relacionadas en un paquete y clases que tienen que ver con los mismos casos de uso en un sólo).

Se pasó luego a realizar los diagramas de interacción por medio de los diagramas de secuencias (que muestran el orden de los mensajes en el tiempo) tanto del sistema como también de forma detallada de los procesos más importantes, de igual forma se realizó los diagramas de estado y se empezó a diseñar las interfaces del software a desarrollar.

Por consiguiente se tiene la realización de un diseño detallado, donde se especifican las diferentes clases y objetos que se tuvieron en cuenta anteriormente. Una ayuda fundamental en esta etapa fueron los patrones de diseño y/o componentes, los cuales se tuvieron en cuenta para el desarrollo del sistema.

Como último ítem en la metodología orientada a objetos, se realizó la implementación y las pruebas para verificar el correcto funcionamiento, para que posteriormente pueda ser utilizado.

8. REQUERIMIENTOS DEL SISTEMA

8.1 REQUERIMIENTOS FUNCIONALES

- Mostrar un mensaje de entrada o de bienvenida que indique el logotipo, el nombre del sistema e información adicional y general del sistema.

- Leer un algoritmo N-S desde un archivo que se encuentra almacenado en disco o en una unidad de almacenamiento.

- Permitir editar y diseñar algoritmos de acuerdo a la herramienta de diagramación N-S.

- Teniendo en cuenta el diagrama realizado, poder generar el código para ser ejecutado en lenguaje C y Java.

- Permitir la ejecución de dichos algoritmos N-S.

- Depurar algoritmos diseñados en la herramienta, indicando el tipo de error y ubicación del mismo.

- Guardar los algoritmos de acuerdo a la extensión: *.dns.

- Implementar las estructuras básicas que se encuentran en el programa de la materia fundamentos de programación, entre las cuáles están: estructuras lineales, de decisión (simples y anidados), iterativas (para, mientras, entre otras).

8.2 REQUERIMIENTOS NO FUNCIONALES

- Dirigido a estudiantes y personas que lo requieran en el campo de algoritmos de acuerdo a la herramienta N-S.

- El sistema deberá funcionar con la máquina virtual de Java (VJM).

- No deberá presentar demoras en cuanto al proceso de ejecución, ya que se contempla la depuración o en otras palabras la revisión general del algoritmo, de igual manera en cuanto a la forma de guardar o abrir archivos.(2 segundos)

- Ejecutar totalmente un algoritmo.

- De igual manera, ejecutar paso a paso el algoritmo diseñado para mayor comprensión del usuario.

- Por otra parte, permitir marcar la instrucción y ejecutar hasta ese determinado punto, con el fin de analizar por partes el diagrama.

- El sistema a desarrollar tiene la característica de portabilidad, es decir que es fácil de llevarse a diferentes equipos, dependiendo de la ubicación del usuario, sólo tendría que contar con la Máquina Virtual Java.

- Implementar ejemplos reales dentro de los cuales se representa la sintaxis y la forma de edición de la herramienta.

- La interfaz en general tendrá un menú principal, menús desplegables, menús contextuales, además de una barra de herramientas que contenga las principales representaciones de un diagrama N-S, y herramientas adicionales para que el usuario pueda visualizar mejor los diagramas (zoom).

- Las diferentes estructuras de los diagramas N-S, deben ser gráficas, de tal manera que el usuario pueda dibujar el objeto y ubicarla de acuerdo a su criterio, además de digitar la operación que se espera que realice.

9. MODELOS DE DOMINIO

9.1 DESCRIPCIÓN DE CASOS DE USO

Ide	CU1
Caso de uso	IniciarPrograma.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El programa debe estar instalado.
Post-condiciones	✍ El programa queda listo para ser usado por el usuario.
Flujo básico	
1. El usuario debe iniciar el programa. 2. El sistema presenta un mensaje de bienvenida y luego carga la interfaz de trabajo para el usuario. 3. El usuario ejecuta el caso de uso **DiseñarDiagrama.**	
Flujos alternativos	

Ide	CU2
Caso de uso	DiseñarDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El programa debe estar listo para usarse.
Post-condiciones	✍ Obtener un diagrama de un determinado algoritmo.
Flujo básico	
1. El usuario escoge la opción Nuevo diagrama. 2. El sistema presenta un área de trabajo en blanco. 3. El sistema carga las herramientas necesarias para el diseño del diagrama. 4. El usuario interactúa con el sistema para plasmar un algoritmo.	
Flujos alternativos	

Ide	CU3
Caso de uso	ModificarDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El programa debe estar listo para usarse.

Post-condiciones	✍ Diagrama modificado.
Flujo básico	
1. El usuario debe elegir la opción Abrir Diagrama. 2. El sistema presenta una pantalla para elegir el archivo en una ruta específica. 3. El usuario elige el archivo y presiona Abrir. 4. El sistema verifica la extensión del archivo y carga el diagrama elegido, con las herramientas necesarias para poder modificar el diagrama. 5. El usuario inicia caso de uso **EditarEstructura**.	
Flujos alternativos	

Ide	CU4
Caso de uso	EjecutarPasoaPaso
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ En el programa debe haber un diagrama abierto.
Post-condiciones	✍ El diagrama es interpretado paso a paso por el sistema.
Flujo básico	
1. El usuario elige la opción Ejecutar Diagrama Paso a Paso. 2. El sistema verifica cada una de las estructuras del diagrama y su gramática. 3. El usuario repite el paso 1. 4. El sistema presenta el resultado de la ejecución.	
Flujos alternativos	
3a. Si el usuario decide detener la ejecución paso a paso 1. El sistema presenta un mensaje confirmando la acción. 3b. Si el sistema llega al final de la ejecución 1. El sistema presenta los resultados del diagrama.	

Ide	CU5
Caso de uso	EjecutarTodo
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ En el programa debe haber un diagrama.
Post-condiciones	✍ Todo el diagrama es interpretado por el sistema.
Flujo básico	
1. El usuario elige la opción Ejecutar Todo. 2. El sistema verifica en todo el diagrama la gramática y las estructuras. 3. El sistema presenta el resultado de la ejecución.	

Flujos alternativos
3a. Si el sistema llega al final de la ejecución 1. El sistema presenta los resultados del diagrama.

Ide	CU6
Caso de uso	GuardarDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ En el programa debe haber un diagrama diseñado.
Post-condiciones	✍ El diagrama es guardado en un archivo.
Flujo básico	
1. El usuario elige la opción Guardar diagrama. 2. El sistema presenta el formulario guardar como para elegir la ruta y el nombre del archivo. 3. El usuario escoge la ruta del archivo y le da un nombre al archivo. 4. El usuario confirma la acción. 5. El sistema crea el archivo plano en la ruta especificada con el formato *.dns	
Flujos alternativos	
1a. Si el archivo ya ha sido guardado antes 1. El sistema guarda los cambios realizados. 3a. Si el nombre del archivo es incorrecto 1. El sistema le asigna un nombre consecutivo al archivo.	

Ide	CU7
Caso de uso	ImprimirDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ En el programa debe haber un diagrama diseñado.
Post-condiciones	✍ El diagrama está impreso.
Flujo básico	
1. El usuario elige la opción Imprimir diagrama. 2. El sistema presenta una ventana para configurar la forma de impresión. 3. El usuario confirma que desea Imprimir el diagrama. 4. El sistema realiza la impresión del diagrama, presentando el resultado en un formato impreso.	
Flujos alternativos	
4a. Si la impresora presenta fallos 1. El sistema informa del error existente a la hora de imprimir el diagrama.	

Ide	CU8
Caso de uso	CortarDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El programa por lo menos debe tener un diagrama abierto y haber seleccionado una o varias partes de ese diagrama.
Post-condiciones	✍ El software crea una copia de la estructura seleccionada actualmente y cuando pega la estructura, borra la anterior.
Flujo básico	
1. El usuario selecciona la opción cortar del menú. 2. El sistema realiza una copia en memoria de la estructura previamente seleccionada, para luego borrarla.	
Flujos alternativos	
1a. Si el usuario no ha seleccionado una estructura 1. El sistema informa que se debe seleccionar como mínimo una estructura del diagrama.	

Ide	CU9
Caso de uso	BorrarDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El programa por lo menos debe tener un diagrama realizado y haber seleccionado una o varias partes de ese diagrama.
Post-condiciones	✍ El software quita la estructura seleccionada.
Flujo básico	
1. El usuario selecciona la opción borrar. 2. El sistema quita la estructura previamente seleccionada del diagrama.	
Flujos alternativos	
1a. Si el usuario no ha seleccionado una estructura 1. El sistema informa que se debe seleccionar como mínimo una estructura del diagrama.	

Ide	CU10
Caso de uso	PegarDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El usuario por lo menos debe haber cortado una de las estructuras del diagrama.
Post-condiciones	✍ El usuario pega la estructura del diagrama en la

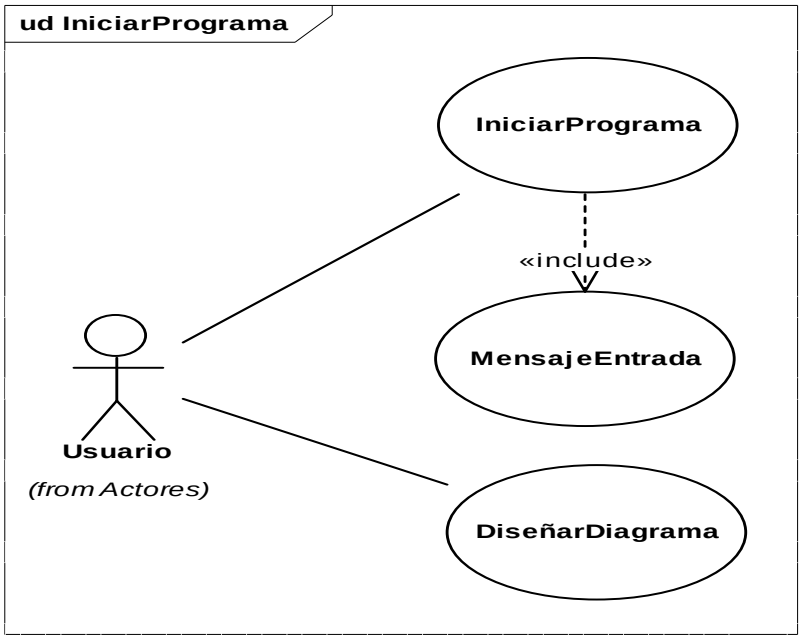
	ubicación indicada por el usuario.
Flujo básico	
1. El usuario selecciona la opción pegar. 2. El sistema recupera de la memoria la estructura previamente seleccionada y la ubica en la posición deseada.	
Flujos alternativos	

Ide	CU11
Caso de uso	EditarEstructura.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El software debe tener un diagrama como mínimo.
Post-condiciones	✍ Se edita las estructuras del usuario de acuerdo a las necesidades que tenga.
Flujo básico	
1. El usuario selecciona la opción editar. 2. El usuario selecciona alguna de las estructuras del diagrama y cambia sus valores. 3. El sistema dibuja los nuevos cambios realizados.	
Flujos alternativos	

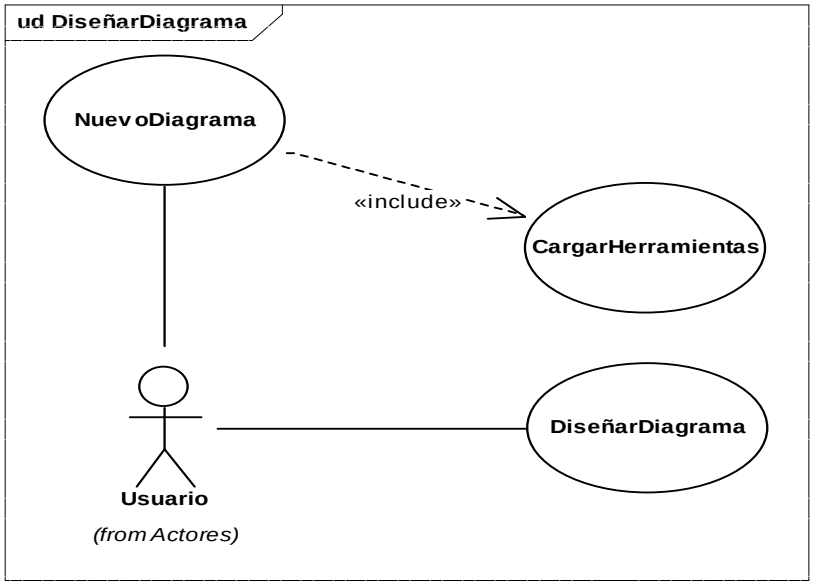
Ide	CU12
Caso de uso	ZoomDiagrama.
Actor principal	Usuario.
Otros actores	Docente, estudiante.
Precondiciones	✍ El software debe tener un diagrama como mínimo.
Post-condiciones	✍ Se cambia de tamaño el diagrama en pantalla.
Flujo básico	
1. El usuario selecciona la opción Aumentar o Disminuir tamaño. 2. El sistema cambia la escala del diagrama.	
Flujos alternativos	

9.2 MODELO DE CASOS DE USO

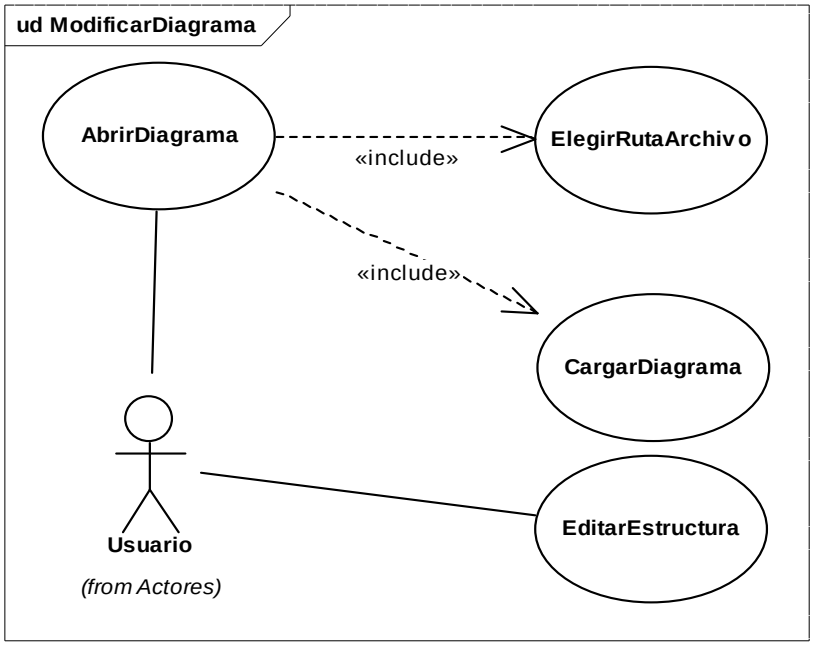
9.2.1 Diagrama IniciarPrograma



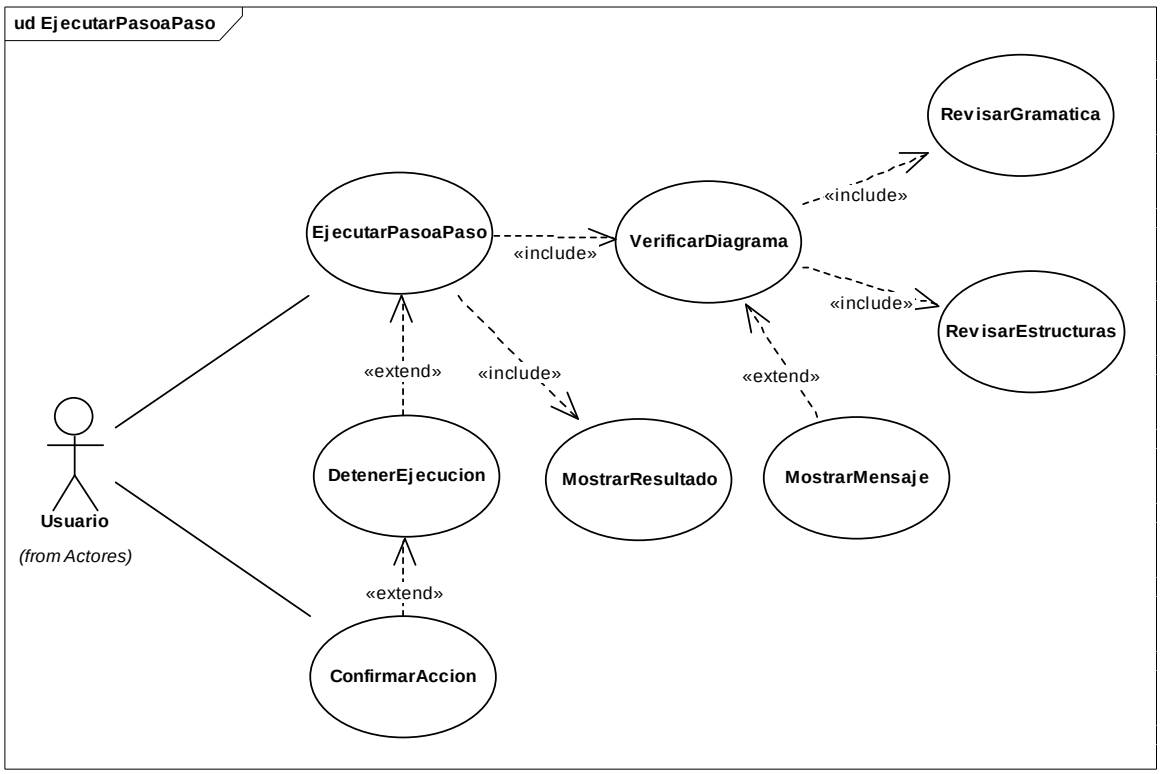
9.2.2 Diagrama DiseñarDiagrama



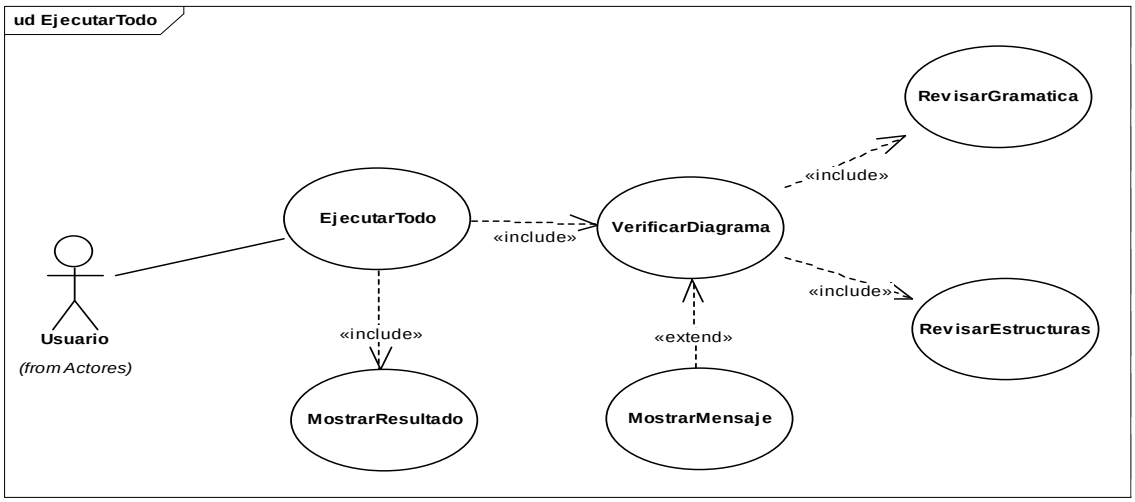
9.2.3 Diagrama ModificarDiagrama



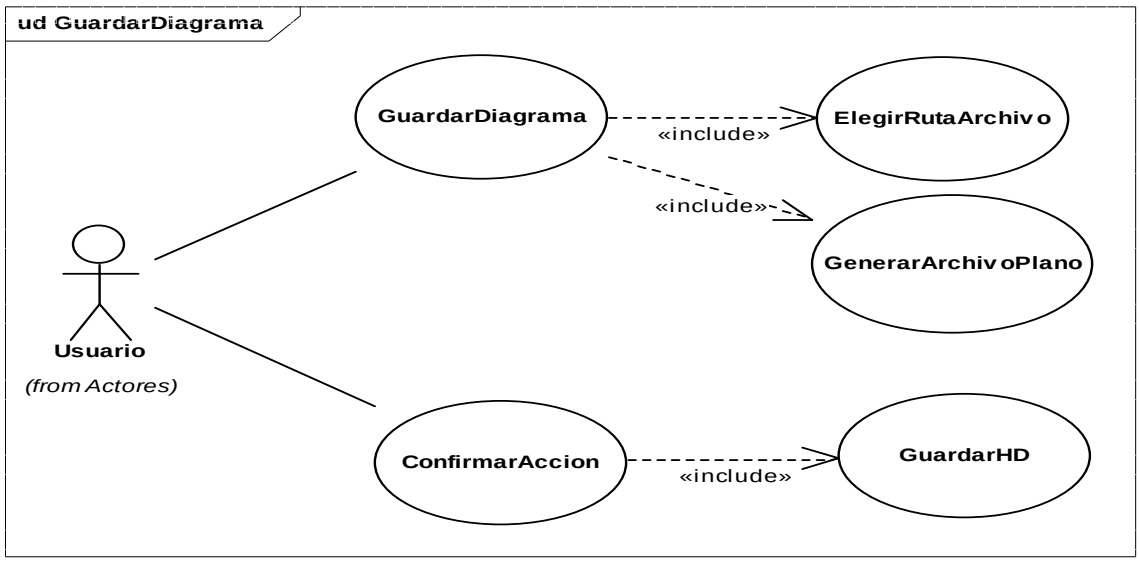
9.2.4 Diagrama EjecutarPasoaPaso



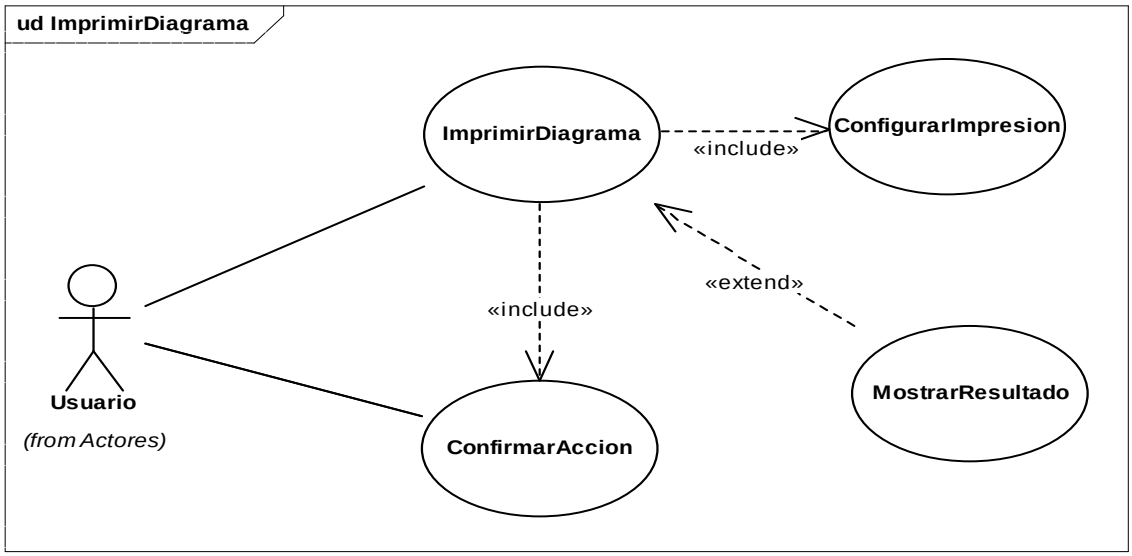
9.2.5 Diagrama EjecutarTodo



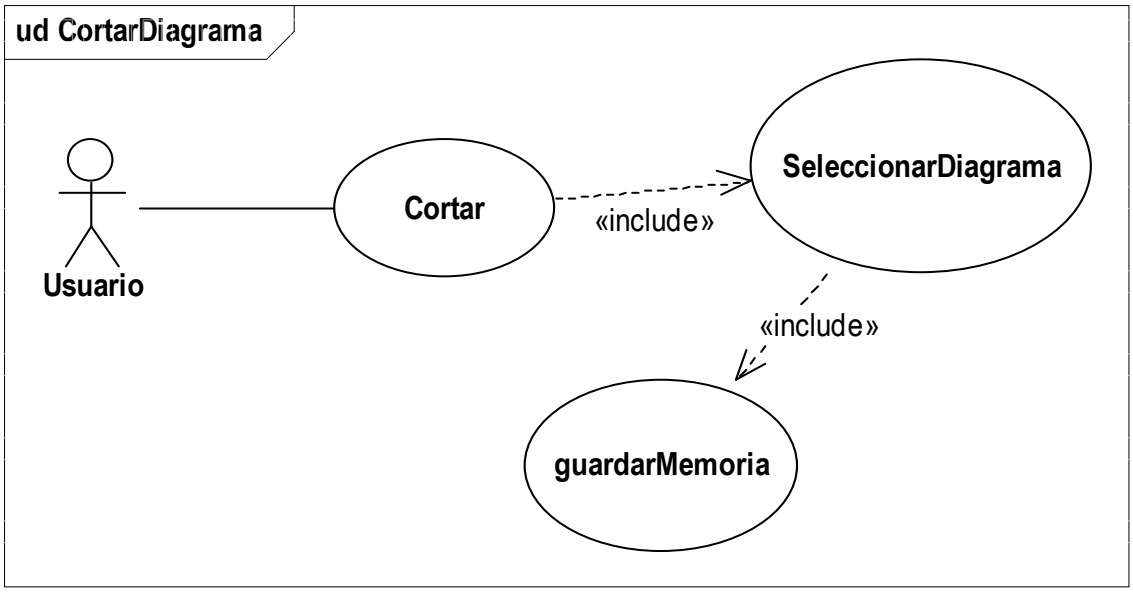
9.2.6 Diagrama GuardarDiagrama



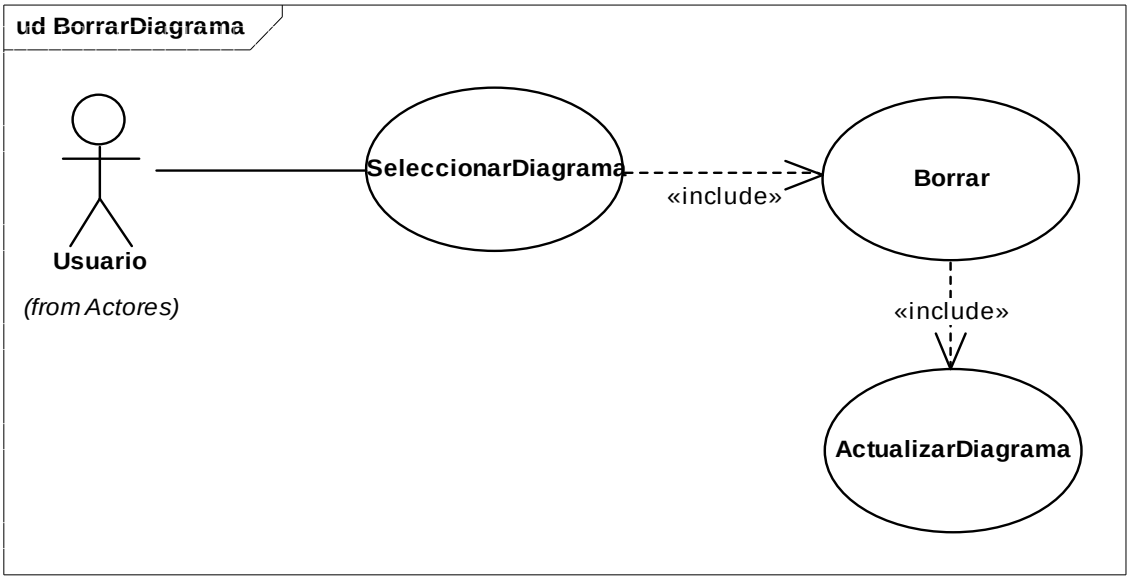
9.2.7 Diagrama ImprimirDiagrama



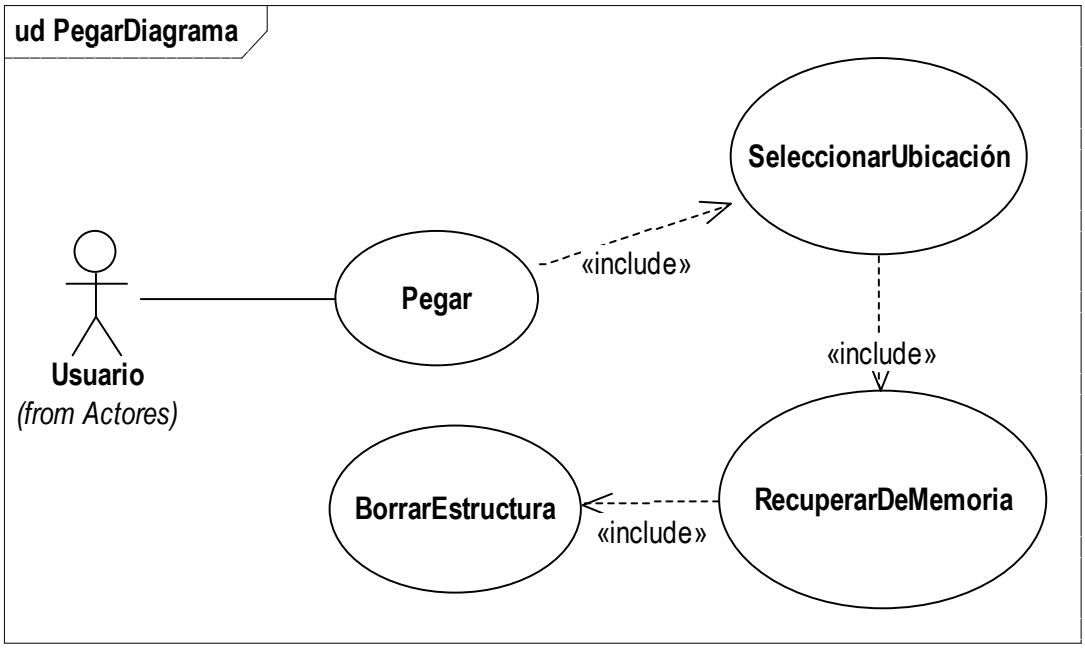
9.2.8 Diagrama CortarDiagrama



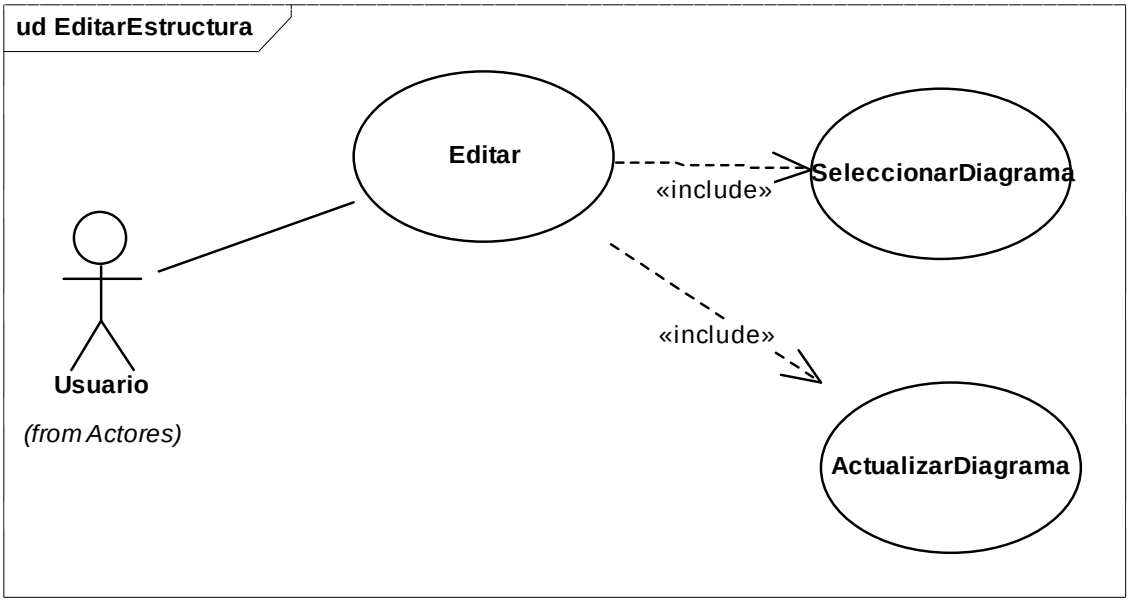
9.2.9 Diagrama BorrarDiagrama



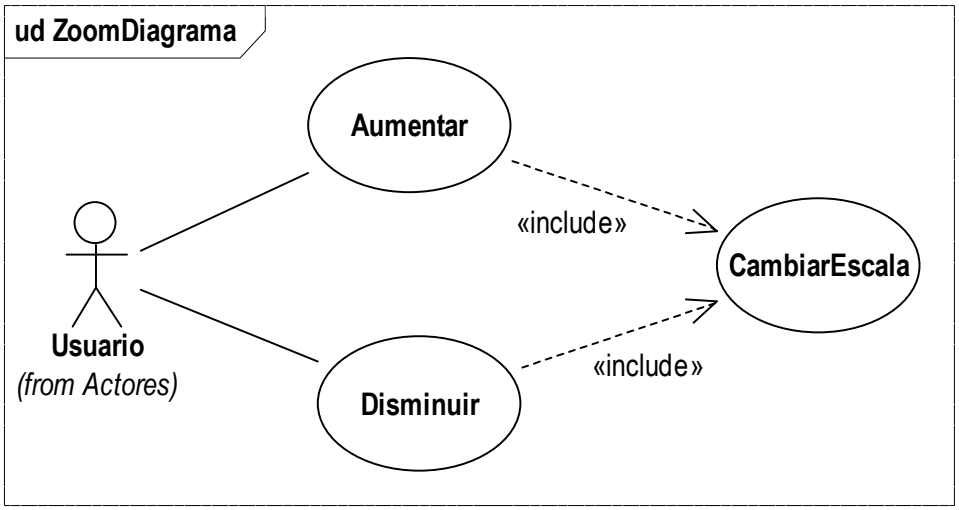
9.2.10 Diagrama PegarDiagrama



9.2.11 Diagrama EditarEstructura



9.2.12 Diagrama ZoomDiagrama



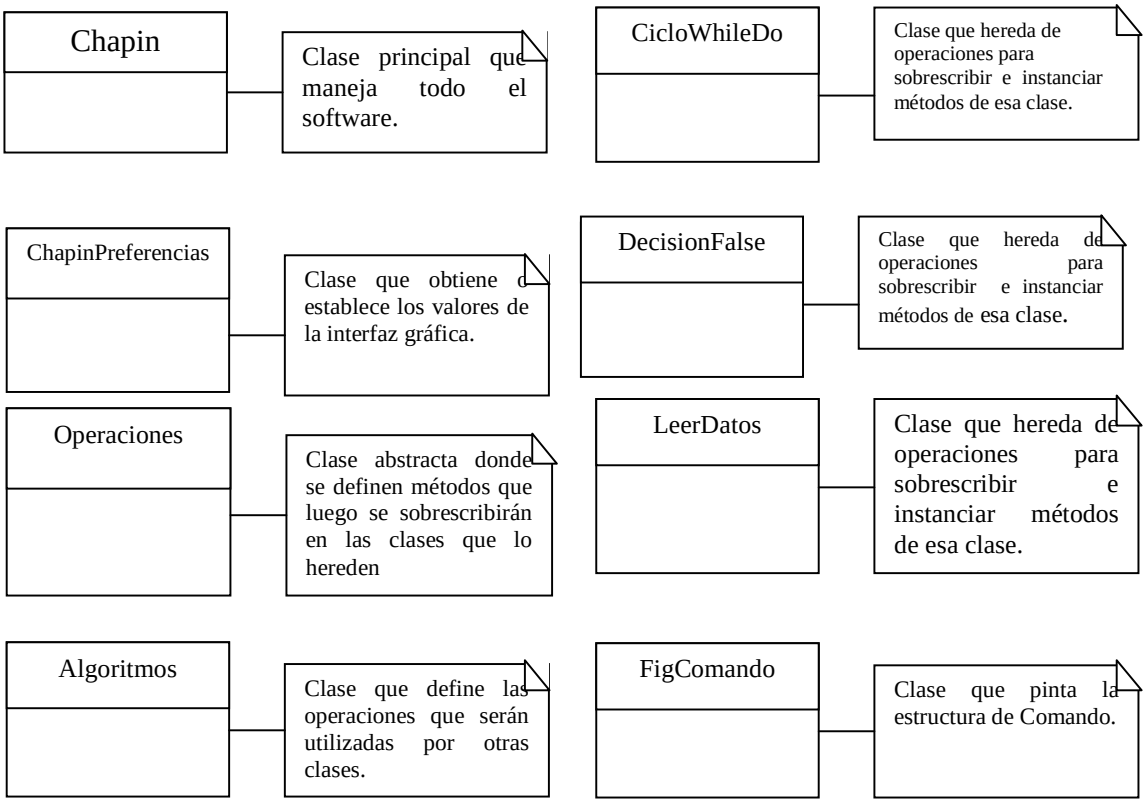
9.3 MODELO DE OBJETOS

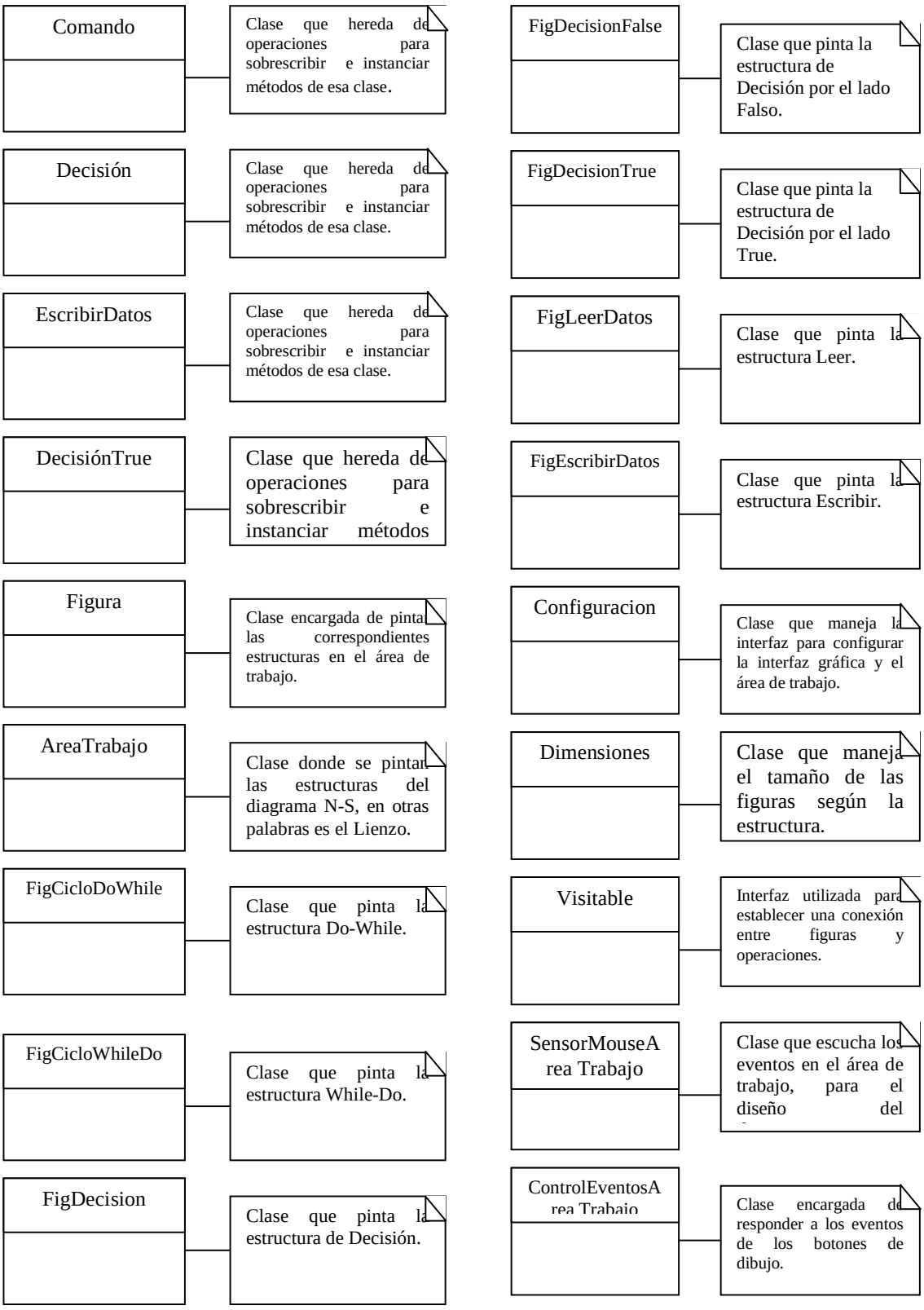
- **Objetos Entidad:** Diagrama, Estructura, Expresión, Variable.

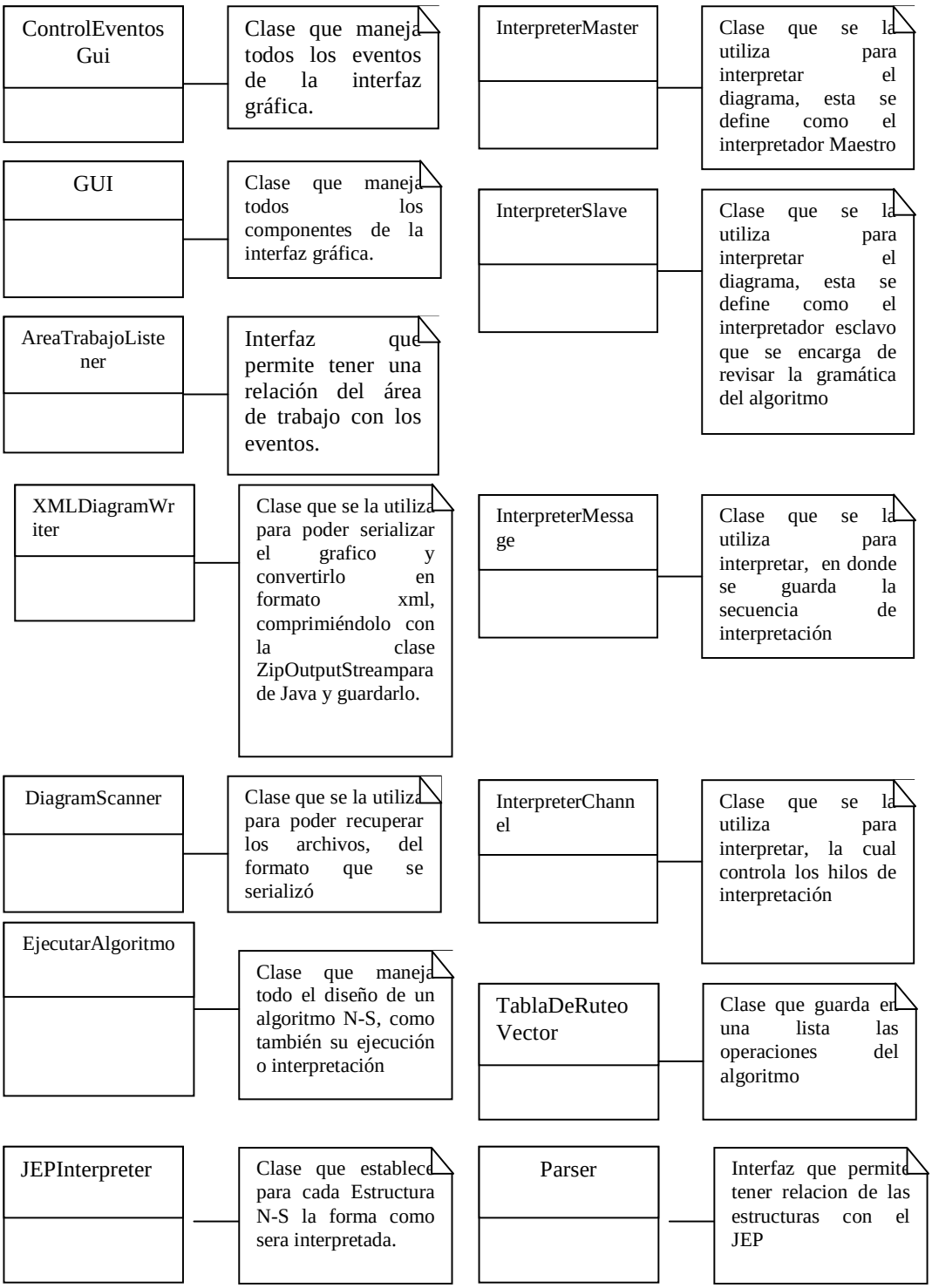
- **Objetos de Interfaz:** Barra de Tareas, Barra de Herramientas, Mensajes, Ventanas de configuración.

- **Objetos de Control:** Impresión de Diagrama, Editar Estructura, Validar variable, Validar expresión.

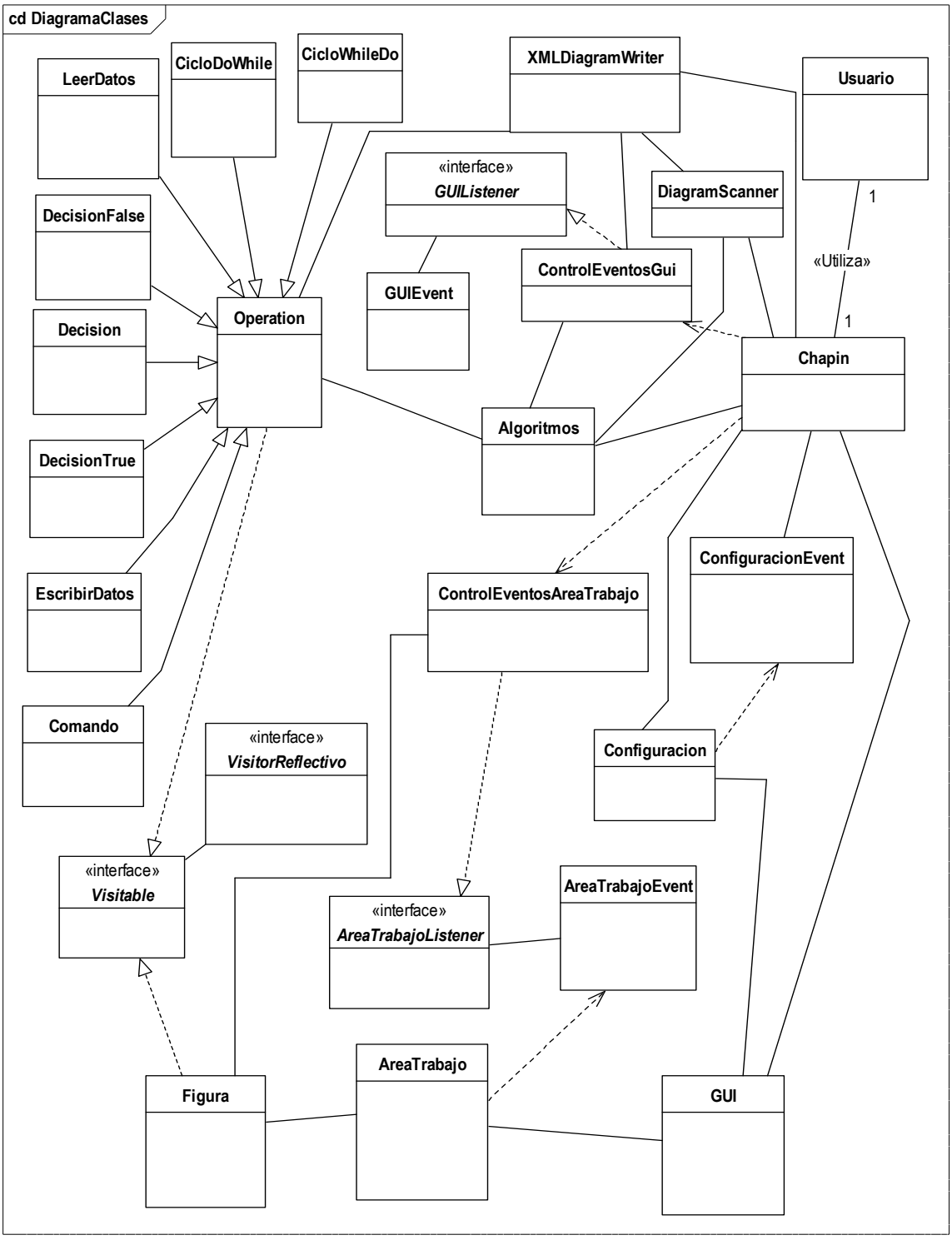
9.3.1 Diccionario de datos. Este es una breve descripción de los objetos y clases principales que se utilizan en el modelo de análisis y más exactamente en el diagrama de clases.

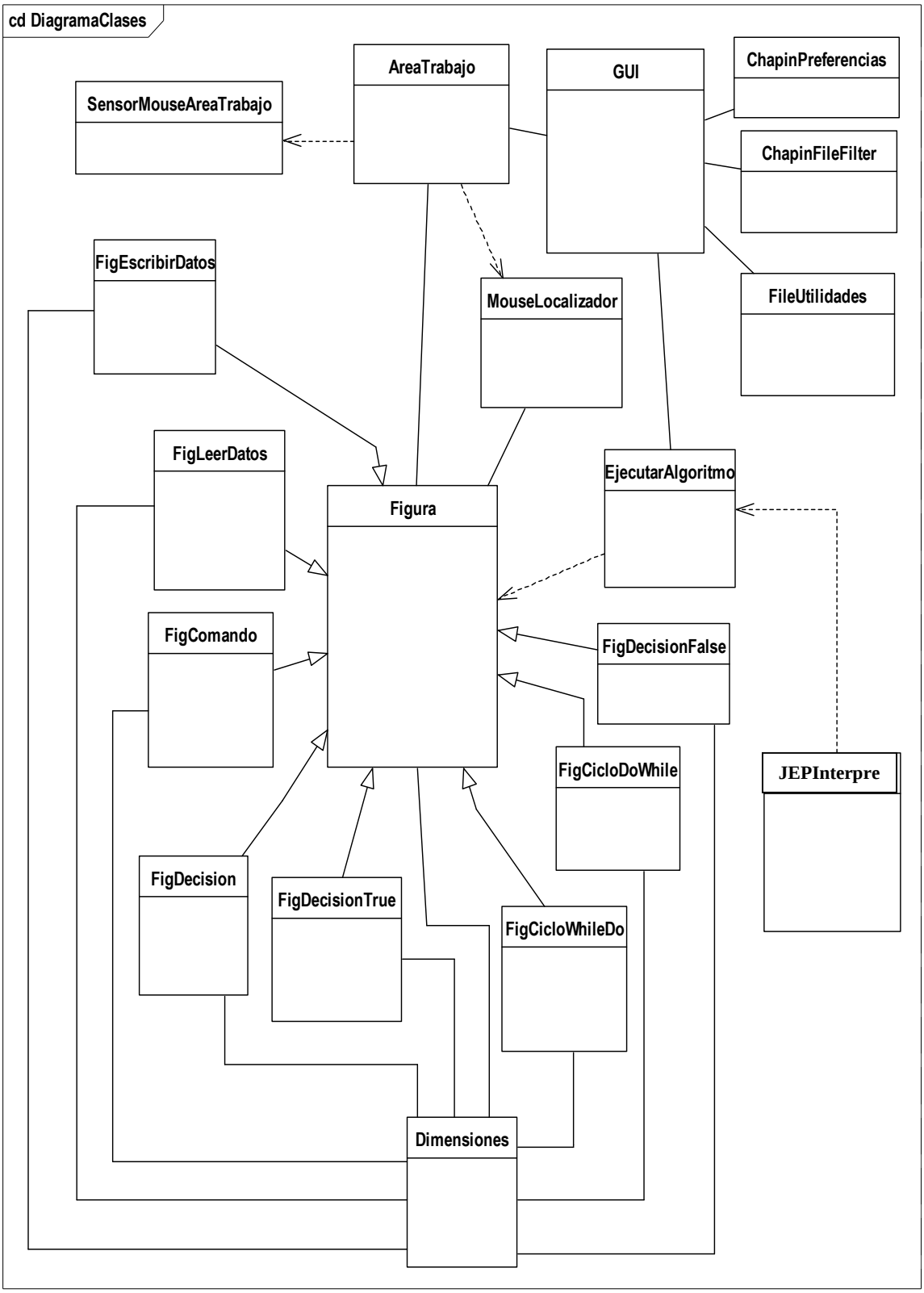

Chapin
Clase principal que maneja todo el software.
CicloWhileDo
Clase que hereda de operaciones para sobrescribir e instanciar métodos de esa clase.
ChapinPreferencias
Clase que obtiene o establece los valores de la interfaz gráfica.
DecisionFalse
Clase que hereda de operaciones para sobrescribir e instanciar métodos de esa clase.
Operaciones
Clase abstracta donde se definen métodos que luego se sobrescribirán en las clases que lo hereden
LeerDatos
Clase que hereda de operaciones para sobrescribir e instanciar métodos de esa clase.
Algoritmos
Clase que define las operaciones que serán utilizadas por otras clases.
FigComando
Clase que pinta la estructura de Comando.


Comando
Clase que hereda de operaciones para sobrescribir e instanciar métodos de esa clase.
Decisión
Clase que hereda de operaciones para sobrescribir e instanciar métodos de esa clase.
EscribirDatos
Clase que hereda de operaciones para sobrescribir e instanciar métodos de esa clase.
DecisiónTrue
Clase que hereda de operaciones para sobrescribir e instanciar métodos
Figura
Clase encargada de pintar las correspondientes estructuras en el área de trabajo.
AreaTrabajo
Clase donde se pintan las estructuras del diagrama N-S, en otras palabras es el Lienzo.
FigCicloDoWhile
Clase que pinta la estructura Do-While.
FigCicloWhileDo
Clase que pinta la estructura While-Do.
FigDecision
Clase que pinta la estructura de Decisión.
FigDecisionFalse
Clase que pinta la estructura de Decisión por el lado Falso.
FigDecisionTrue
Clase que pinta la estructura de Decisión por el lado True.
FigLeerDatos
Clase que pinta la estructura Leer.
FigEscribirDatos
Clase que pinta la estructura Escribir.
Configuracion
Clase que maneja la interfaz para configurar la interfaz gráfica y el área de trabajo.
Dimensiones
Clase que maneja el tamaño de las figuras según la estructura.
Visitable
Interfaz utilizada para establecer una conexión entre figuras y operaciones.
SensorMouseA rea Trabajo
Clase que escucha los eventos en el área de trabajo, para el diseño del
ControlEventosA rea Trabajo
Clase encargada de responder a los eventos de los botones de dibujo.


ControlEventos Gui
Clase que maneja todos los eventos de la interfaz gráfica.
InterpreterMaster
Clase que se la utiliza para interpretar el diagrama, esta se define como el interpretador Maestro
GUI
Clase que maneja todos los componentes de la interfaz gráfica.
InterpreterSlave
Clase que se la utiliza para interpretar el diagrama, esta se define como el interpretador esclavo que se encarga de revisar la gramática del algoritmo
AreaTrabajoListener
Interfaz que permite tener una relación del área de trabajo con los eventos.
XMLDiagramWriter
Clase que se la utiliza para poder serializar el grafico y convertirlo en formato xml, comprimiéndolo con la clase ZipOutputStreampara de Java y guardarlo.
InterpreterMessage
Clase que se la utiliza para interpretar, en donde se guarda la secuencia de interpretación
DiagramScanner
Clase que se la utiliza para poder recuperar los archivos, del formato que se serializó
InterpreterChannel
Clase que se la utiliza para interpretar, la cual controla los hilos de interpretación
EjecutarAlgoritmo
Clase que maneja todo el diseño de un algoritmo N-S, como también su ejecución o interpretación
TablaDeRuteo Vector
Clase que guarda en una lista las operaciones del algoritmo
JEPInterpreter
Clase que establece para cada Estructura N-S la forma como sera interpretada.
Parser
Interfaz que permite tener relacion de las estructuras con el JEP

9.3.2 Diagrama de Clases.

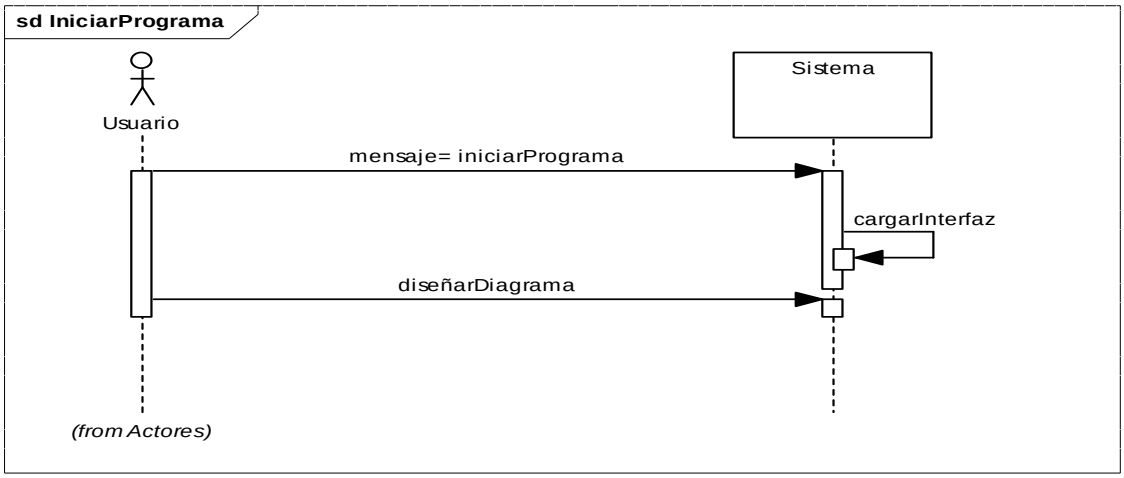



cd DiagramaClases
AreaTrabajo
GUI
ChapinPreferencias
SensorMouseAreaTrabajo
ChapinFileFilter
FigEscribirDatos
MouseLocalizador
FileUtilidades
FigLeerDatos
EjecutarAlgoritmo
Figura
FigComando
FigDecisionFalse
FigCicloDoWhile
JEPInterpre
FigDecision
FigDecisionTrue
FigCicloWhileDo
Dimensiones

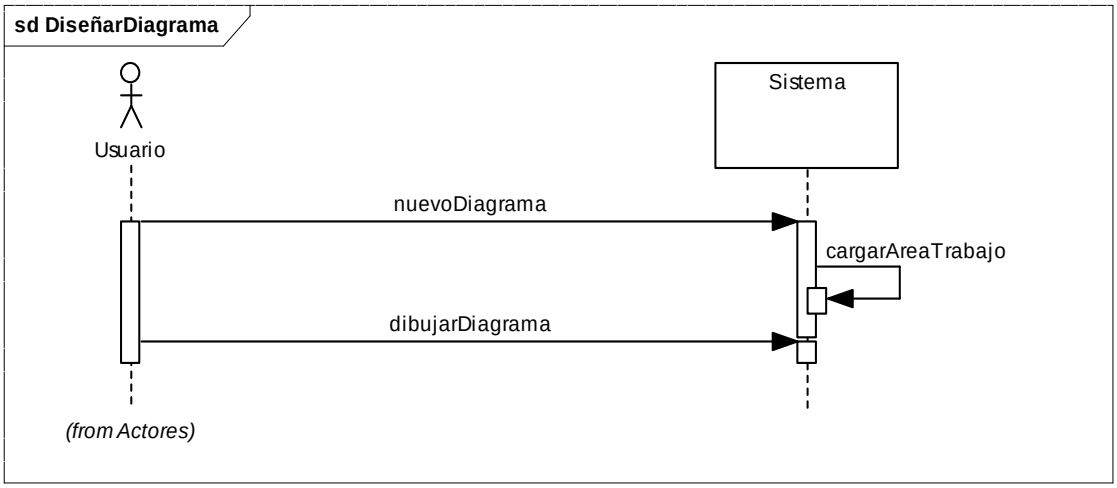
9.4 MODELOS DINÁMICOS

9.4.1 Diagramas de Secuencia del Sistema.

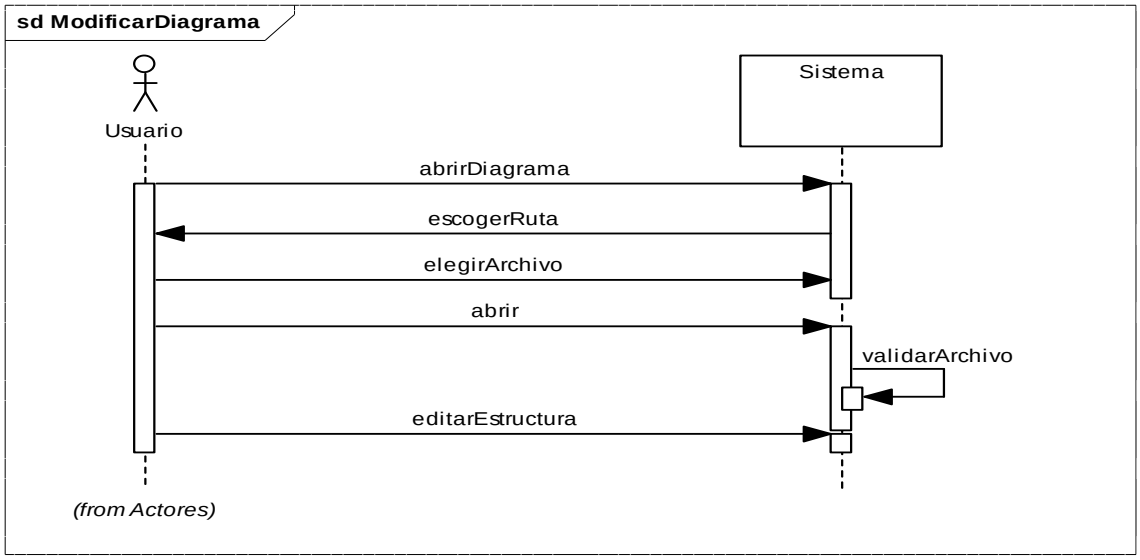
- **Diagrama de Secuencia de IniciarPrograma.** Secuencia de operaciones para iniciar el programa



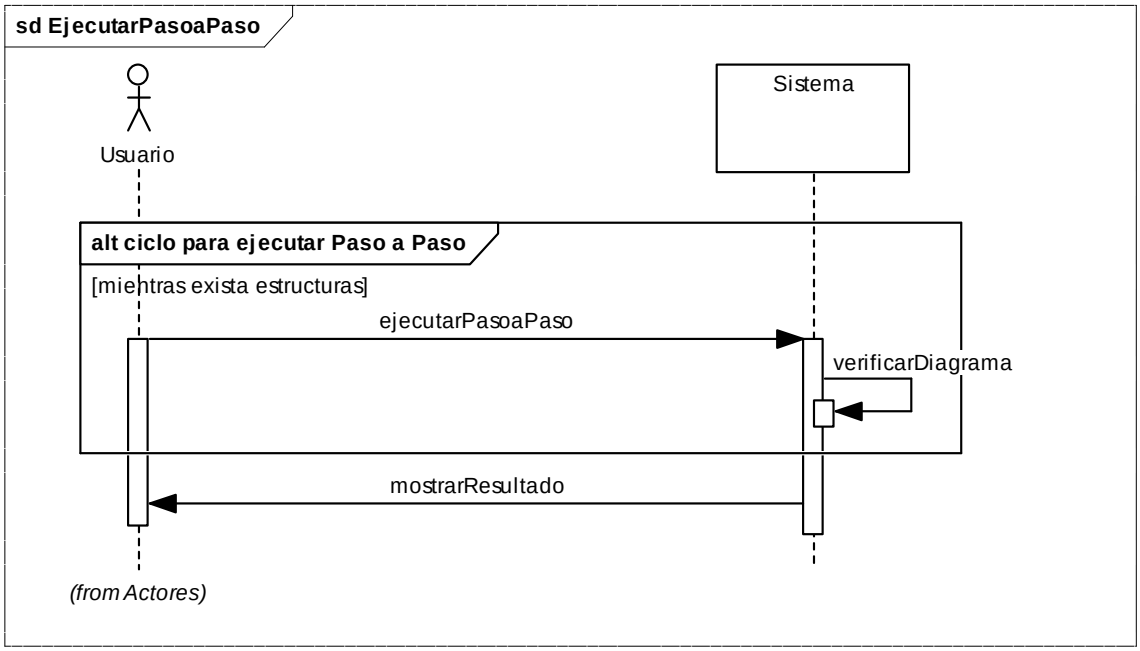
- **Diagrama de Secuencia de DiseñarDiagrama.** Secuencia de mensajes necesarios para diseñar algoritmo N-S



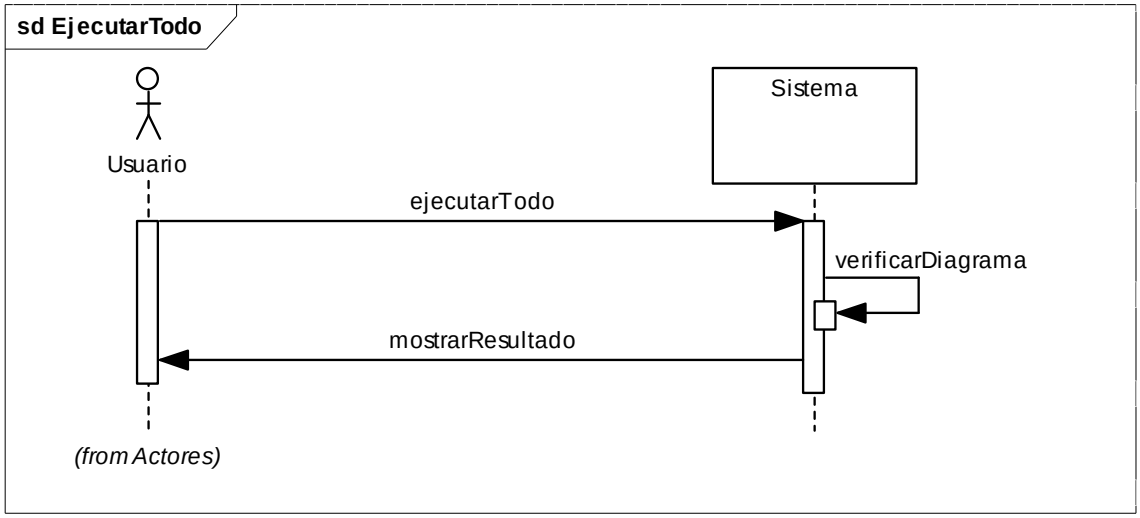
- **Diagrama de Secuencia de ModificarDiagrama.** Secuencia de mensajes necesarios para modificar una estructura de un algoritmo



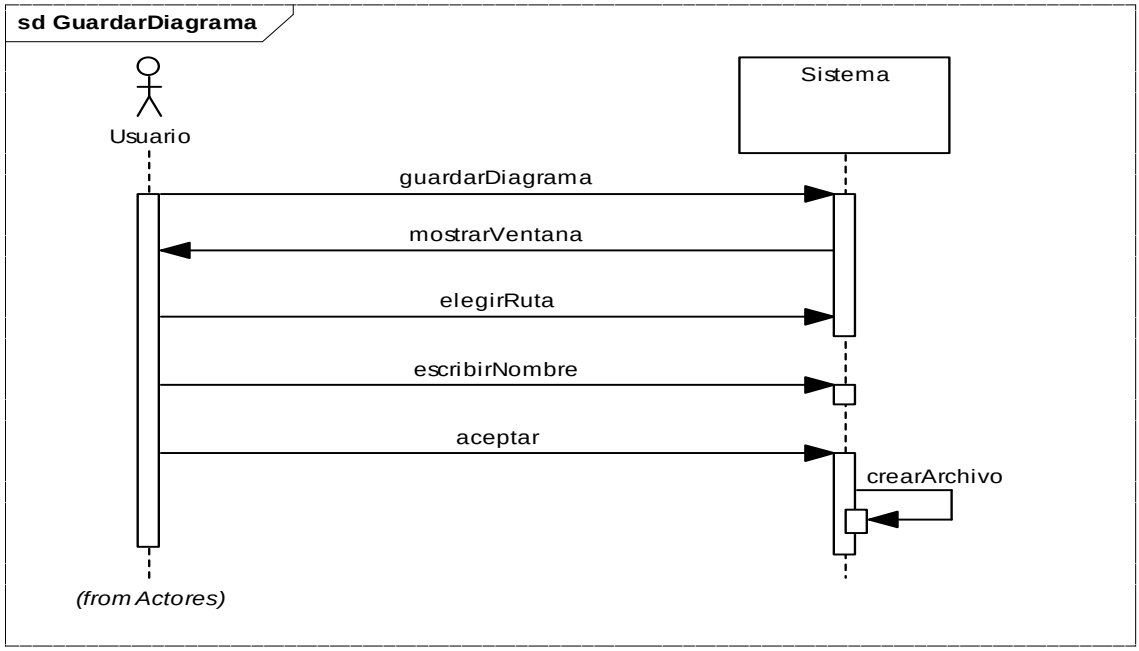
- **Diagrama de Secuencia de EjecutarPasoaPaso.** Secuencia de mensajes necesarios para ejecutar un diagrama paso a paso.



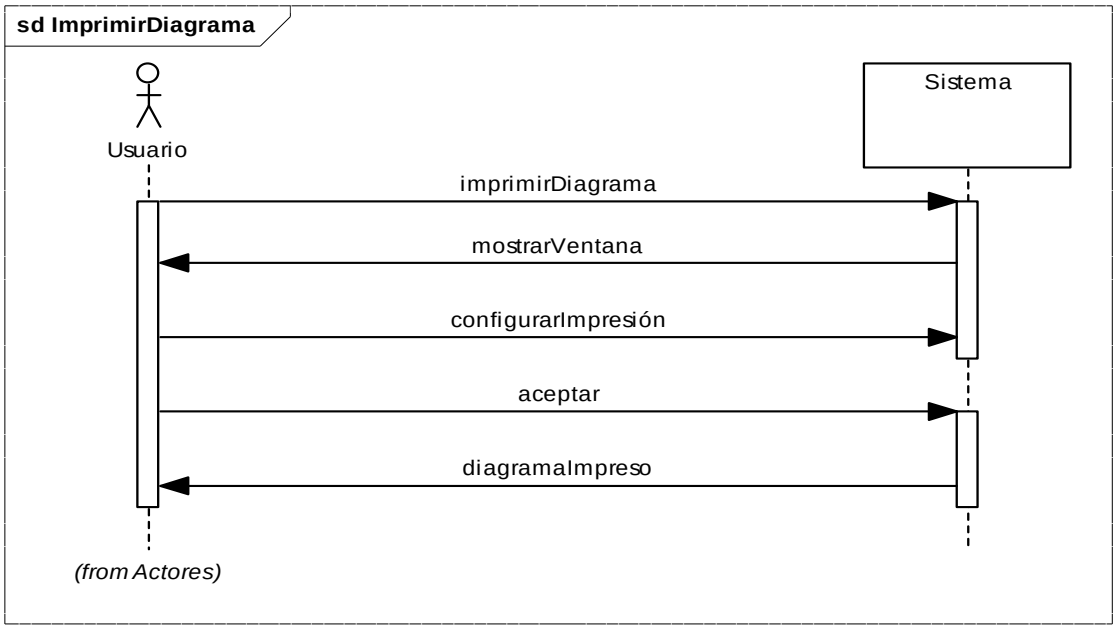
- **Diagrama de Secuencia de EjecutarTodo.** Secuencia de mensajes necesarios para ejecutar todo el diagrama



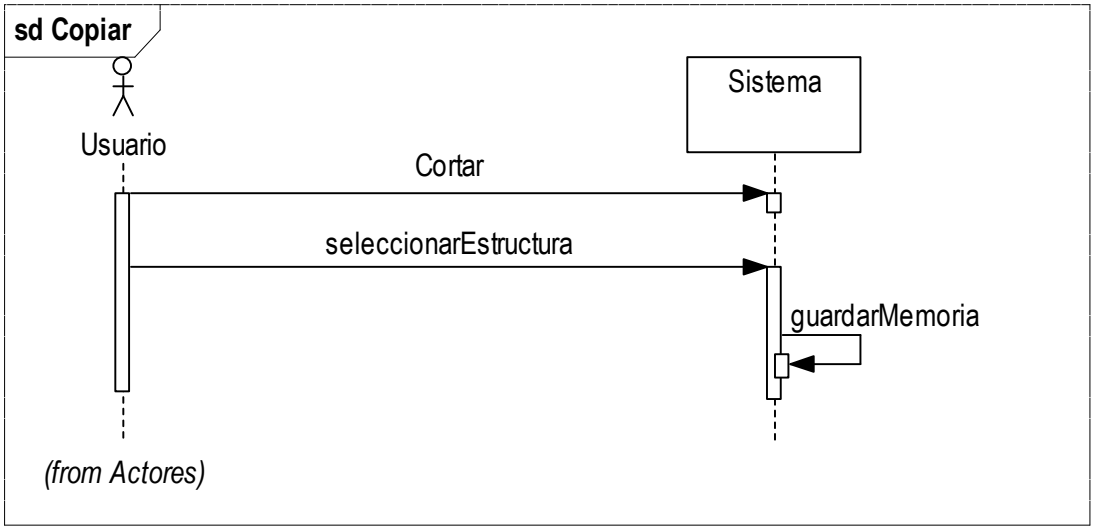
- **Diagrama de Secuencia de GuardarDiagrama.** secuencia de mensajes para que el usuario pueda guardar un diagrama



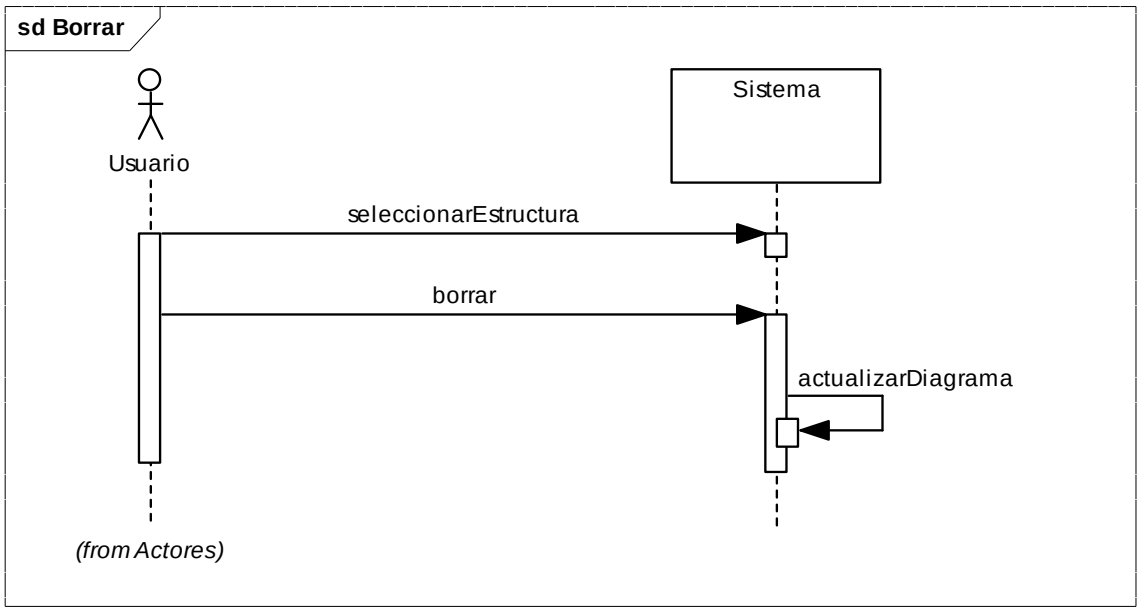
- **Diagrama de Secuencia de ImprimirDiagrama**. Secuencia de mensajes para obtener un diagrama en formato impreso



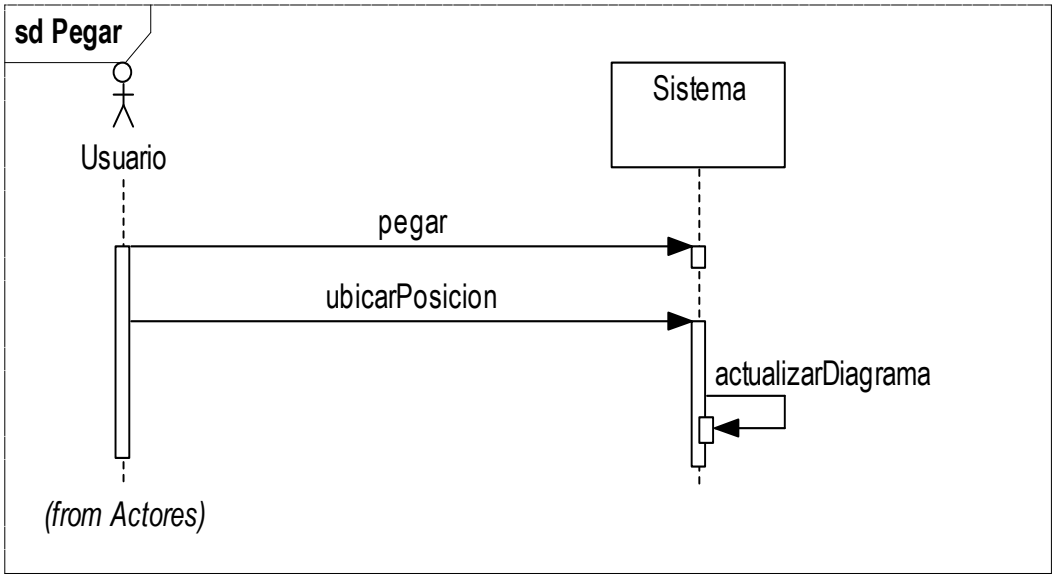
- **Diagrama de Secuencia de CortarDiagrama.** Secuencia de mensajes para cortar una estructura de un diagrama



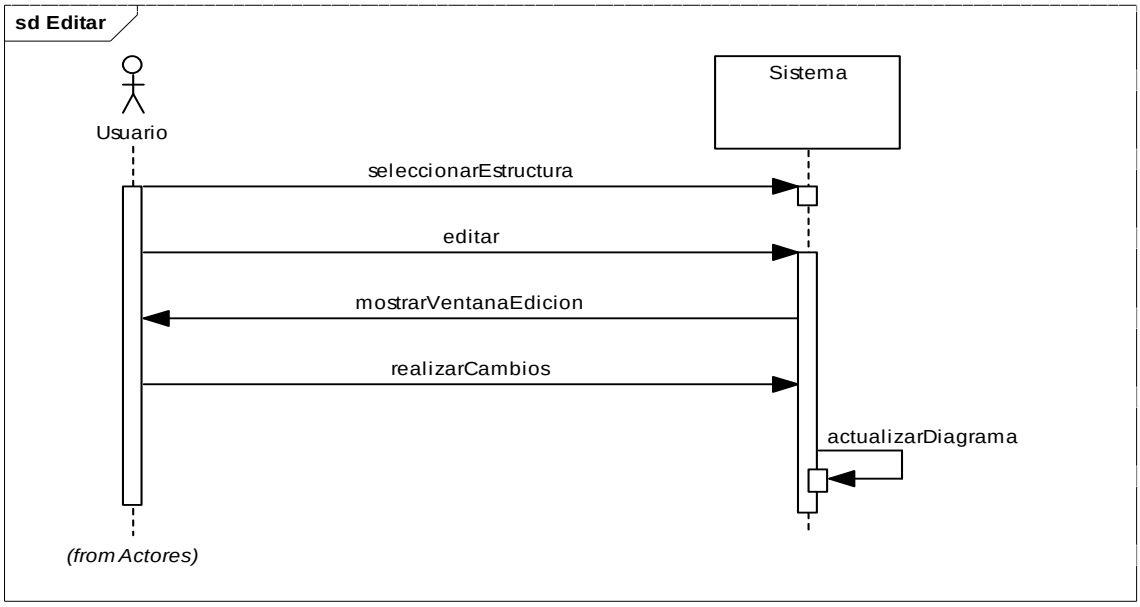
- **Diagrama de Secuencia de BorrarDiagrama.** Secuencia de mensajes para borra una estructura de un diagrama



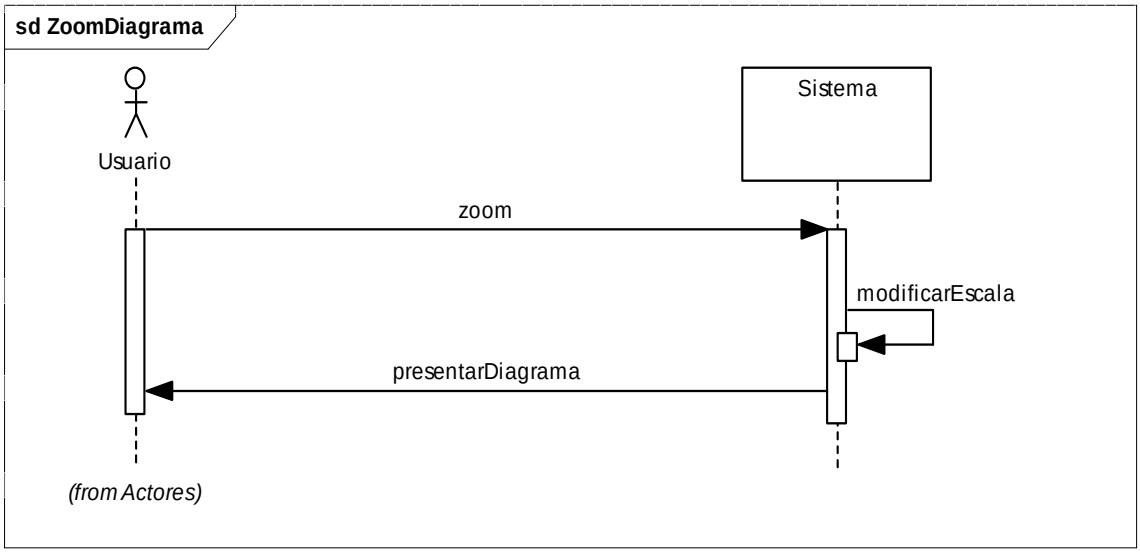
- **Diagrama de Secuencia de PegarDiagrama.** Secuencia de mensajes para recupera de memoria una estructura



- **EditarEstructura:** Secuencia de mensajes necesarios para editar una estructura de un diagrama

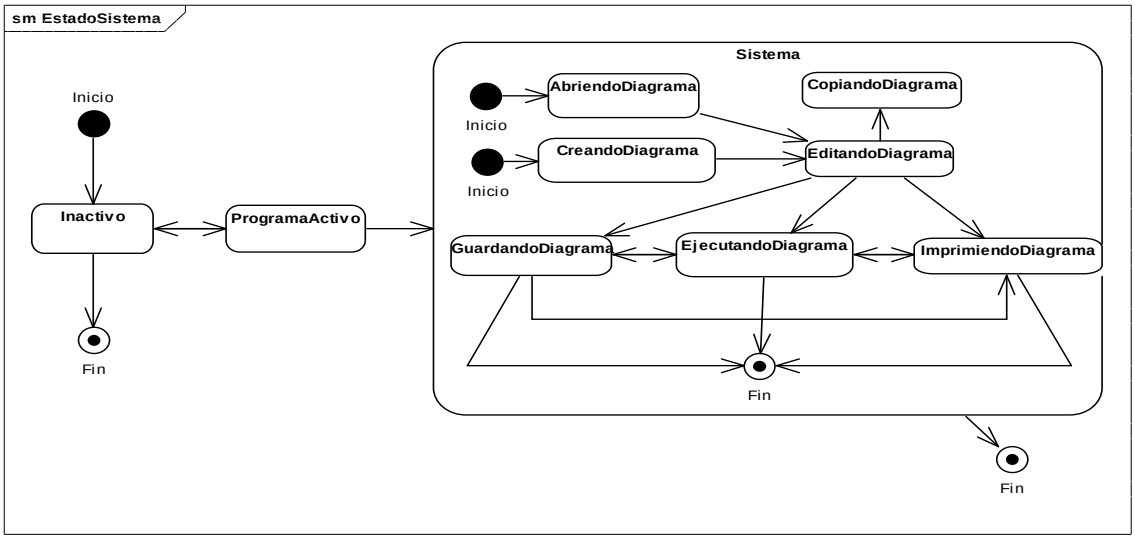


- **Diagrama de Secuencia de ZoomDiagrama.** Secuencia de mensajes necesarios para cambiar de escala a un diagrama

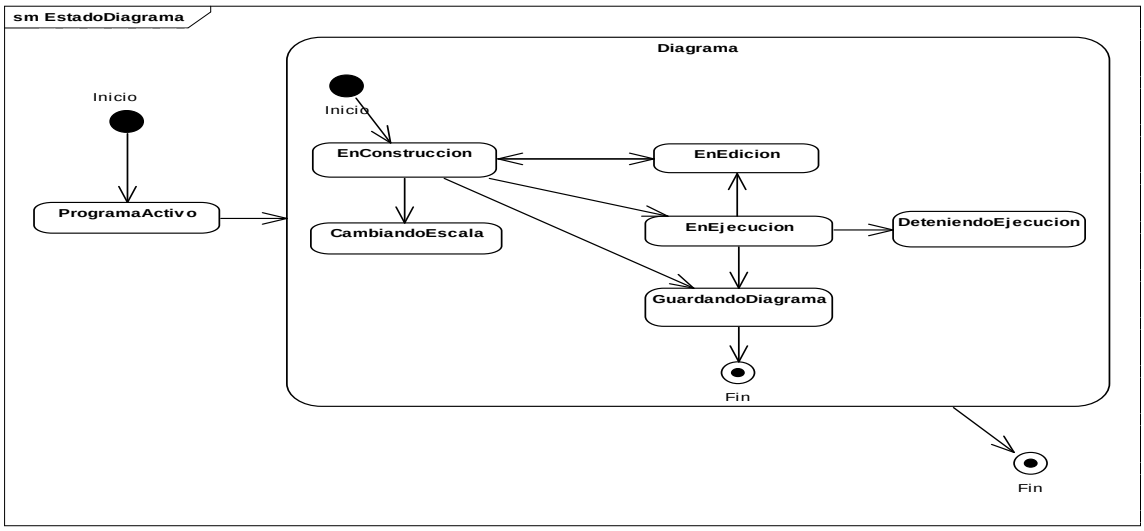


9.4.2 Diagramas de Estado. Son los diferentes estados en que se encuentra un objeto en algún momento de su ciclo de vida, entendiendo como estado a: una condición, en espera de una orden o realizar una actividad.

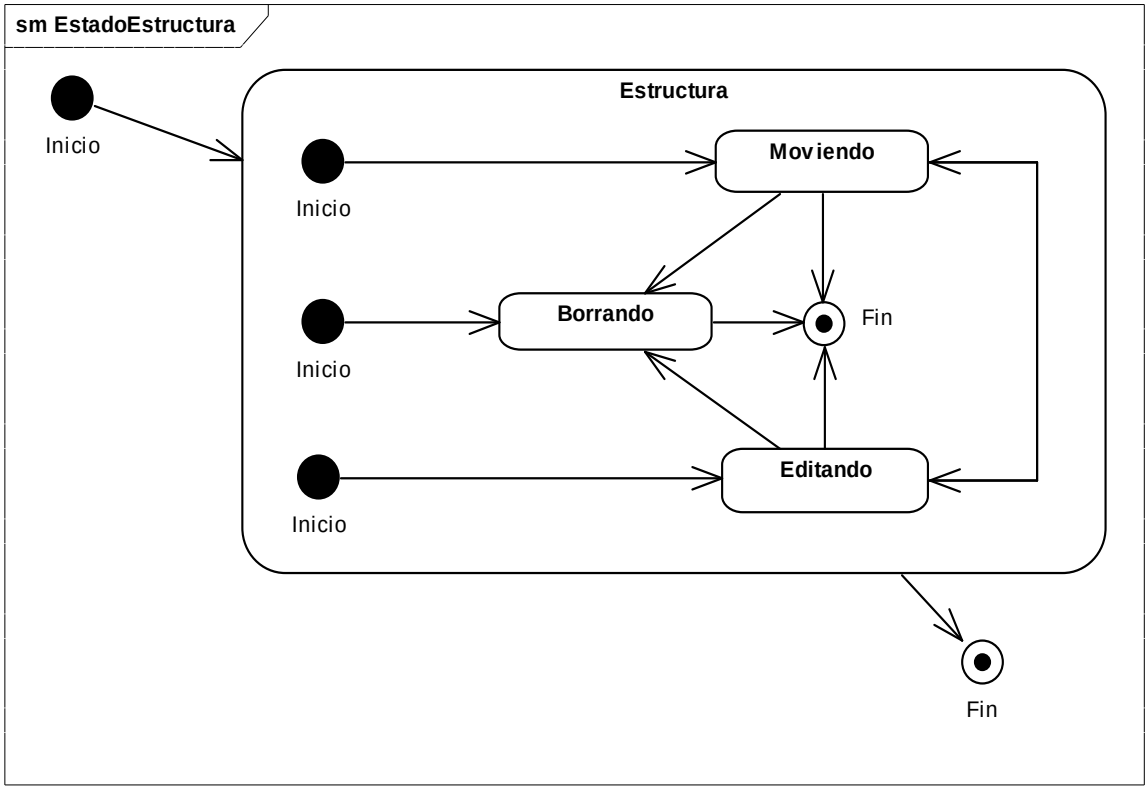
- **Diagrama EstadoSistema:** Representa las posibles situaciones en las que puede estar el sistema en general.



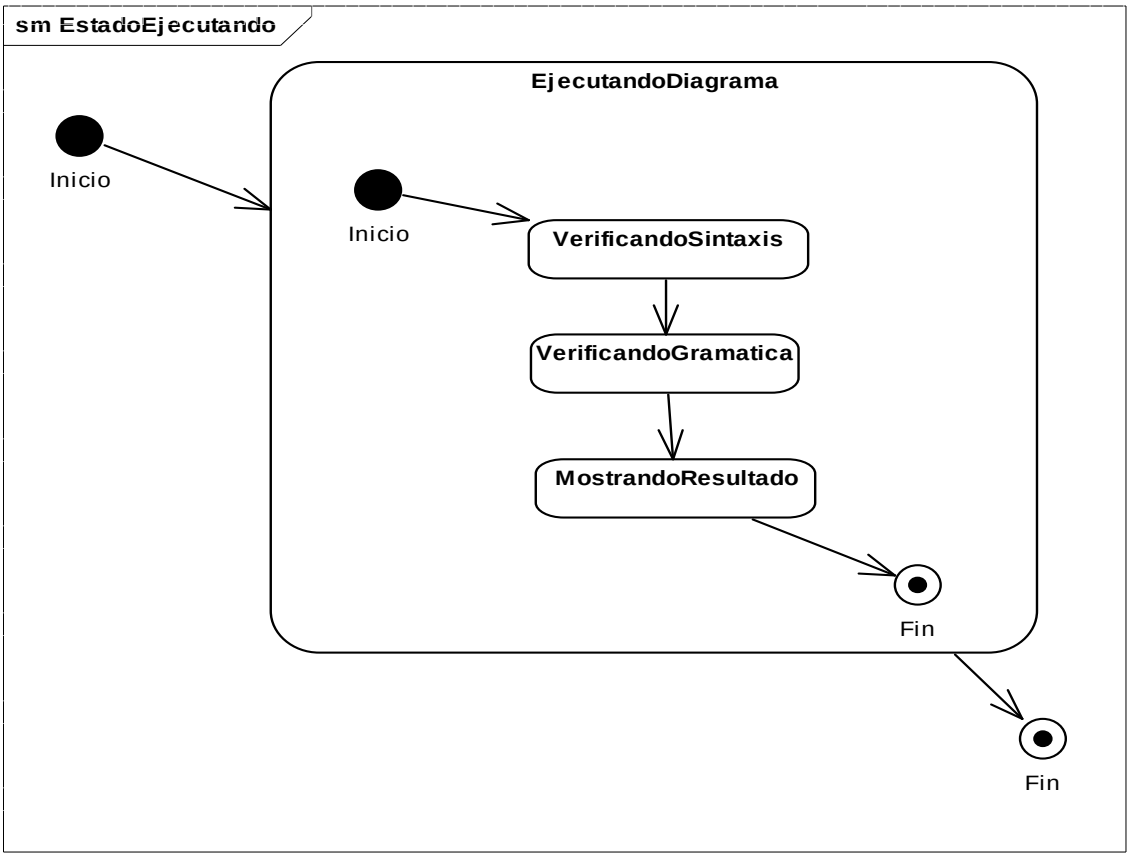
- **Diagrama EstadoDiagrama:** Representa las posibles situaciones en las que un diagrama, como objeto particular



- **Diagrama EstadoEstructura:** Representa los estados en los que puede estar una de las diferentes estructuras del diagrama en cuestión.

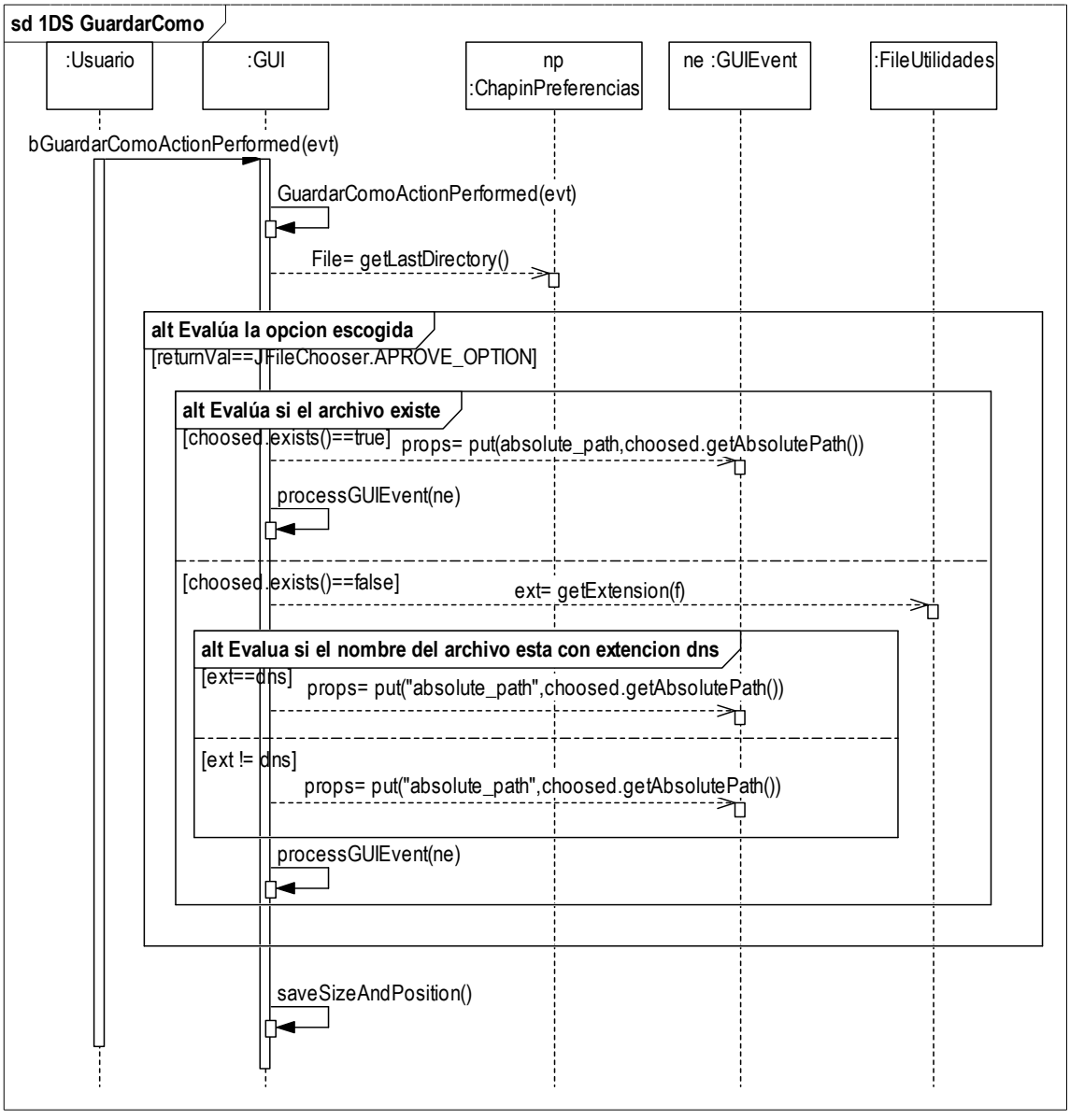


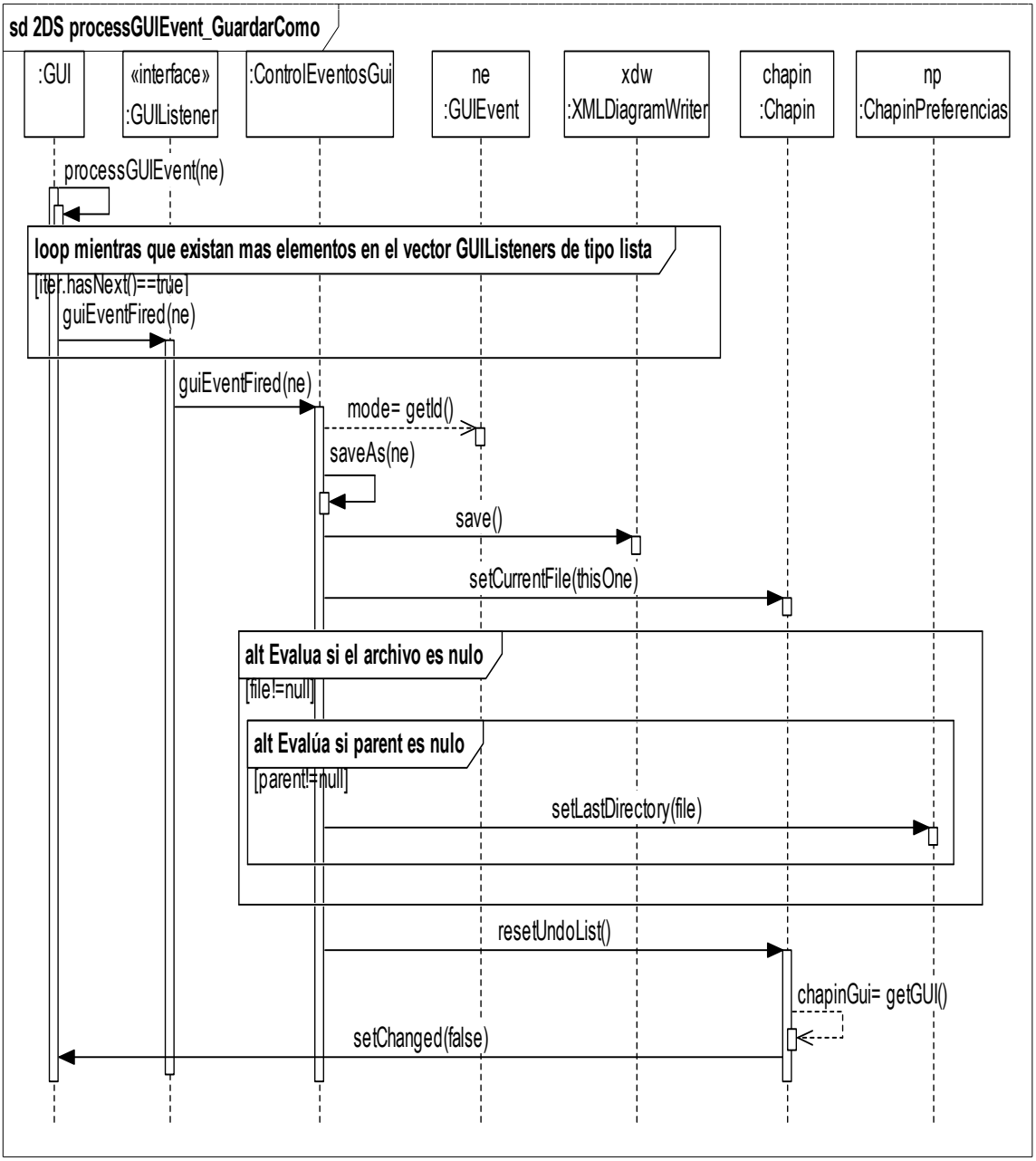
- **DiagramaEstadoEjecutando:** Representa los diferentes estados por los que pasa un diagrama al momento de ejecutar un diagrama.

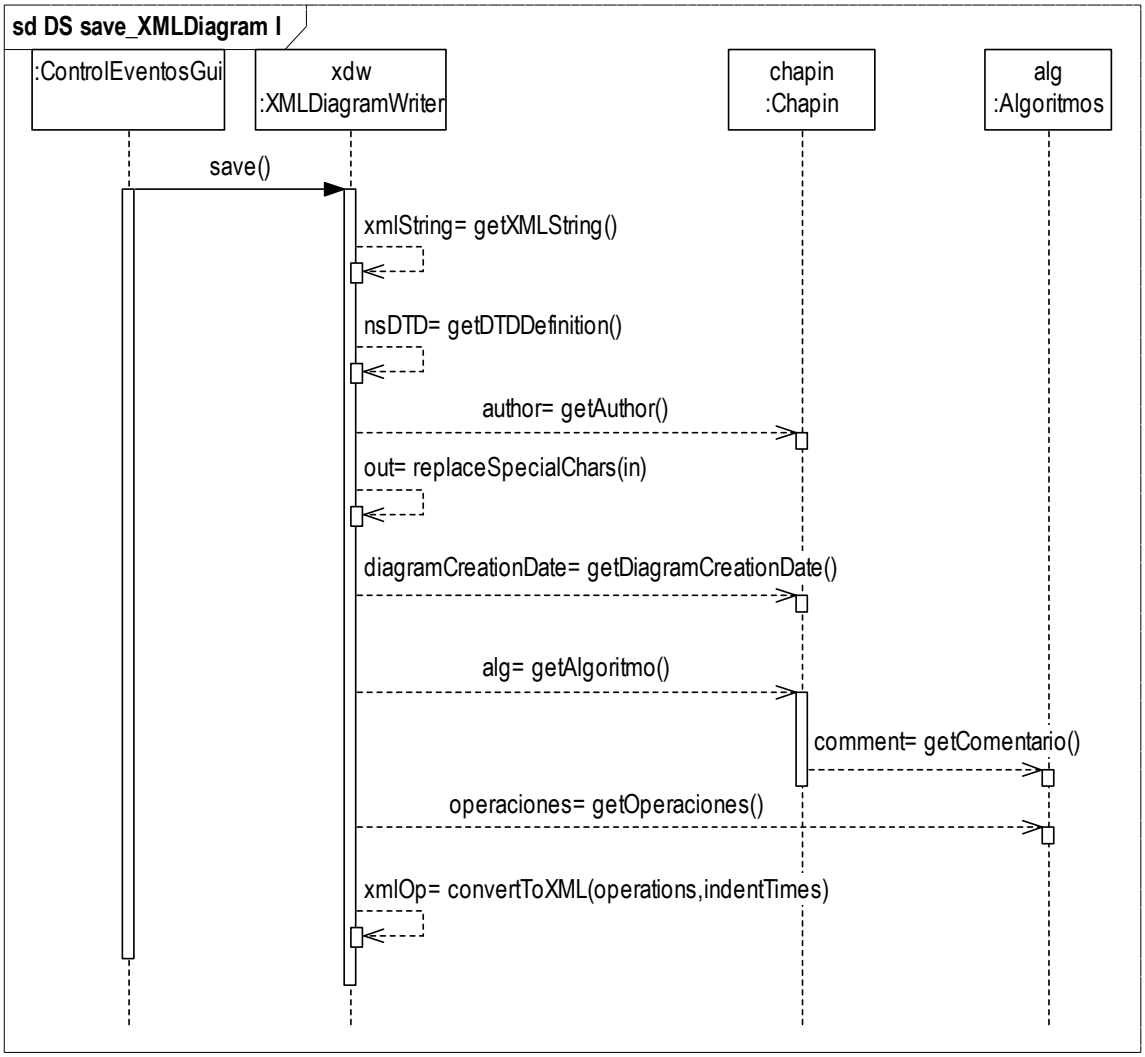


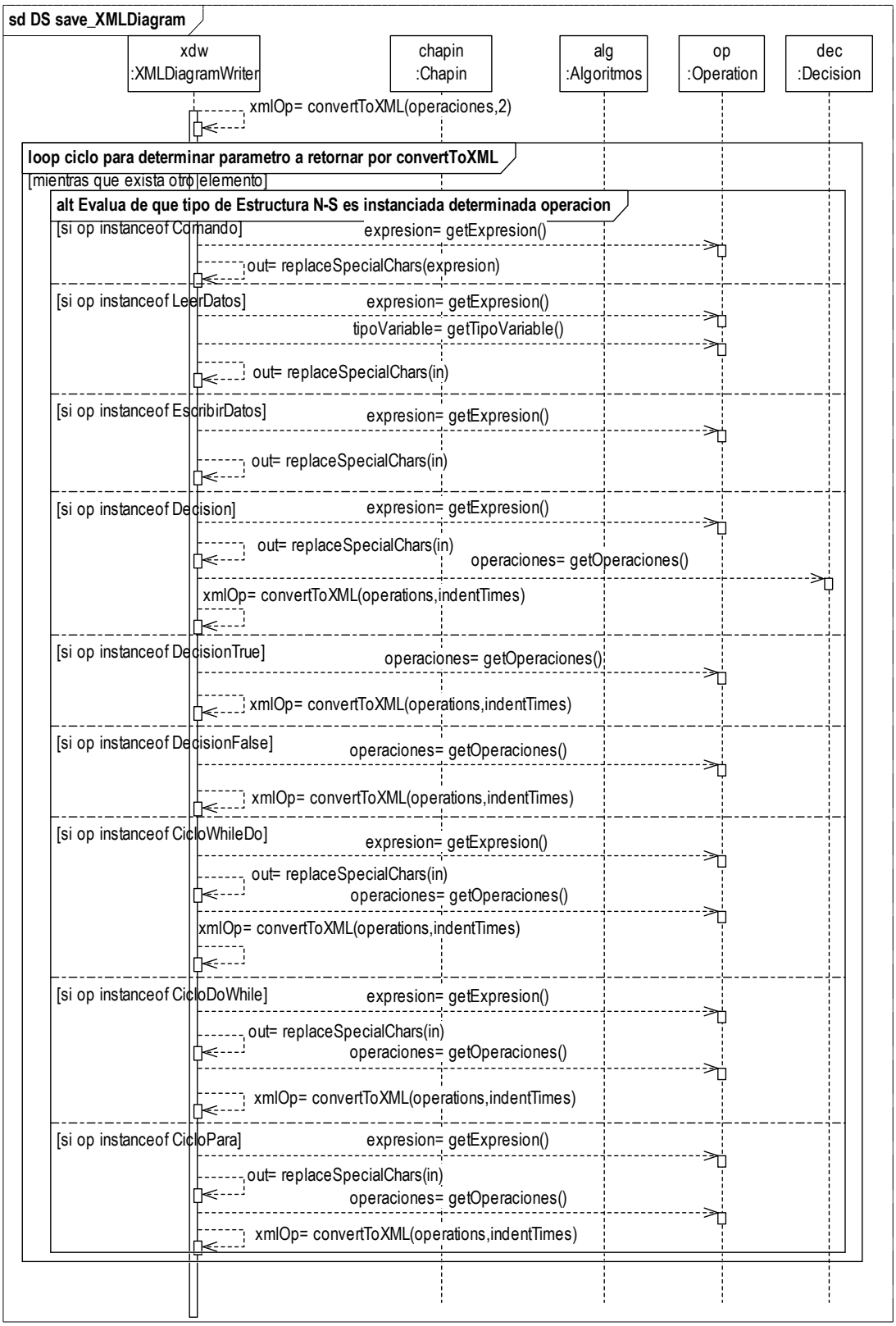
10. MODELO DE DISEÑO: DIAGRAMA DE SECUENCIA DETALLADO

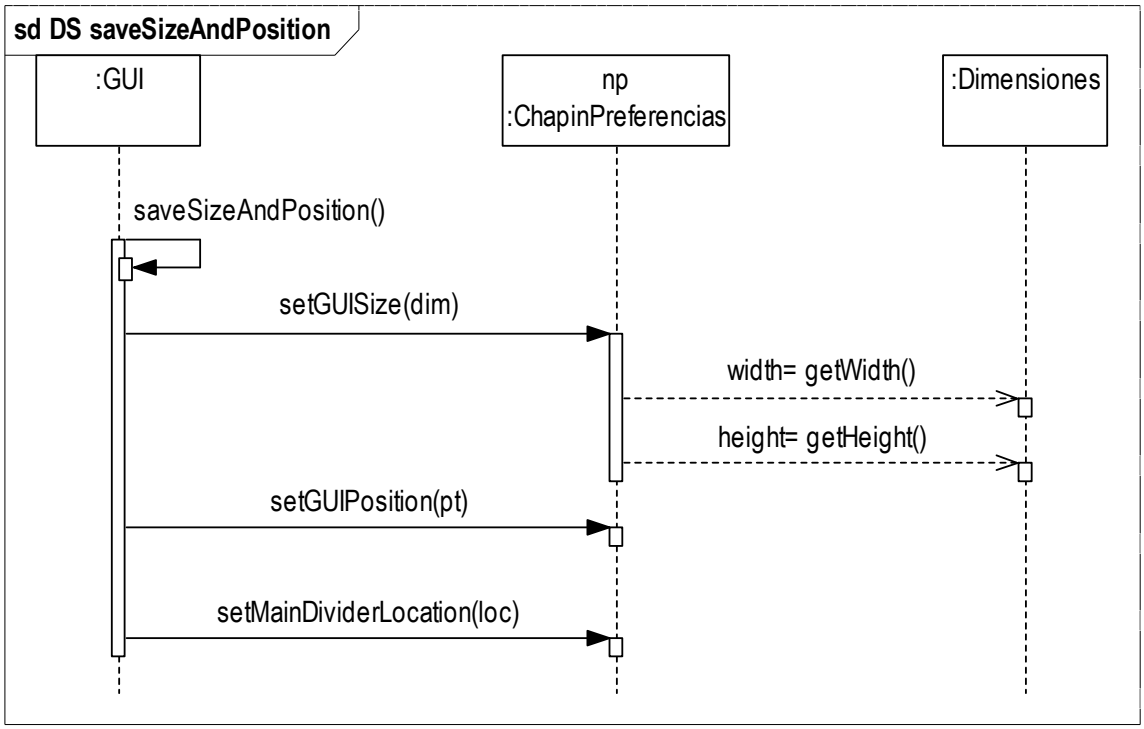
10.1 DIAGRAMA DE SECUENCIA DE GUARDAR COMO



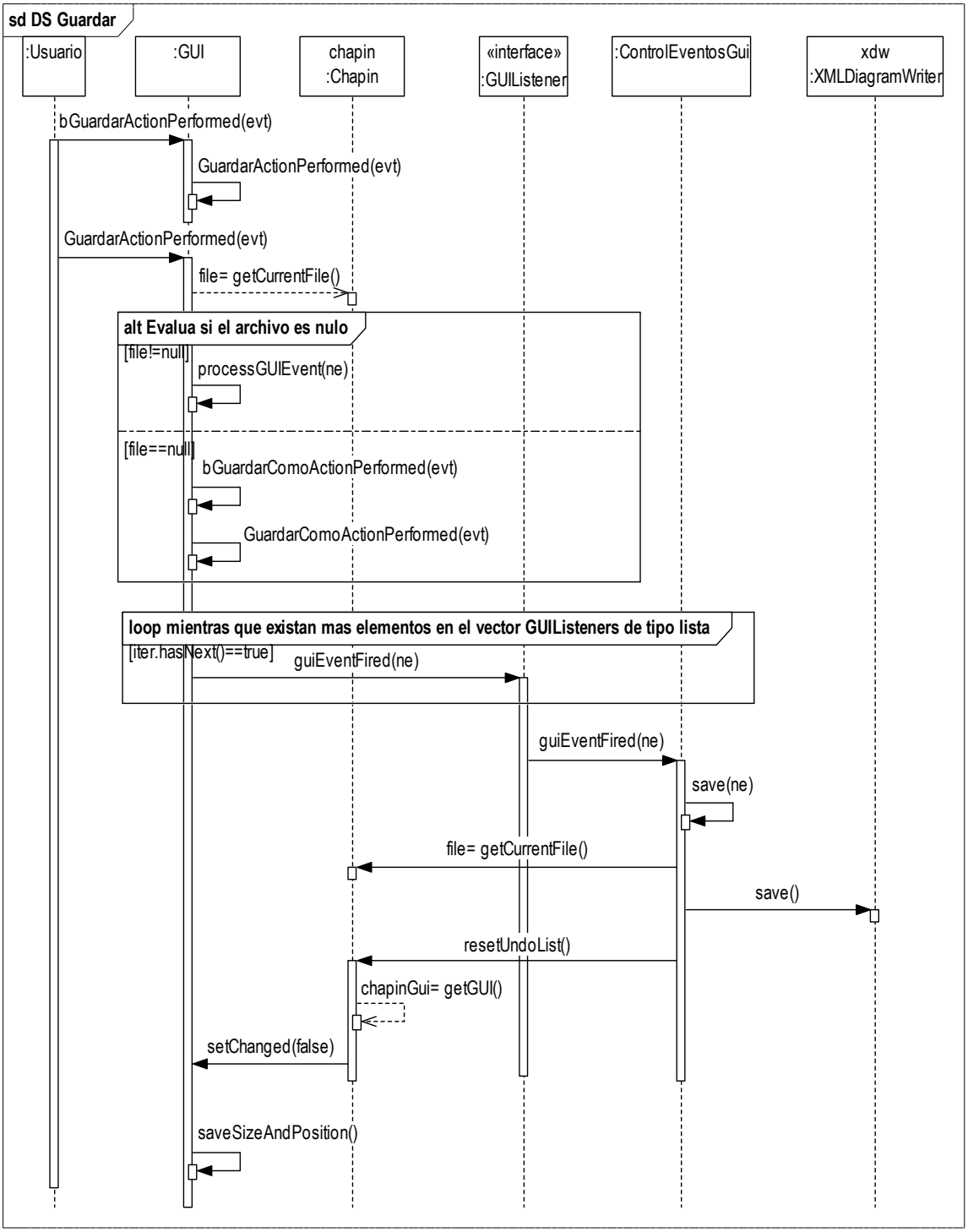

sd 2DS processGUIEvent_GuardarComo
:GUI
«interface»
:GUIListener
:ControlEventosGui
ne
:GUIEvent
xdw
:XMLDiagramWriter
chapin
:Chapin
np
:ChapinPreferencias
processGUIEvent(ne)
loop mientras que existan mas elementos en el vector GUIListeners de tipo lista
[iter.hasNext()==true]
guiEventFired(ne)
guiEventFired(ne)
mode= getId()
saveAs(ne)
save()
setCurrentFile(thisOne)
alt Evalua si el archivo es nulo
[file!=null]
alt Evalúa si parent es nulo
[parent!=null]
setLastDirectory(file)
resetUndoList()
chapinGui= getGUI()
setChanged(false)

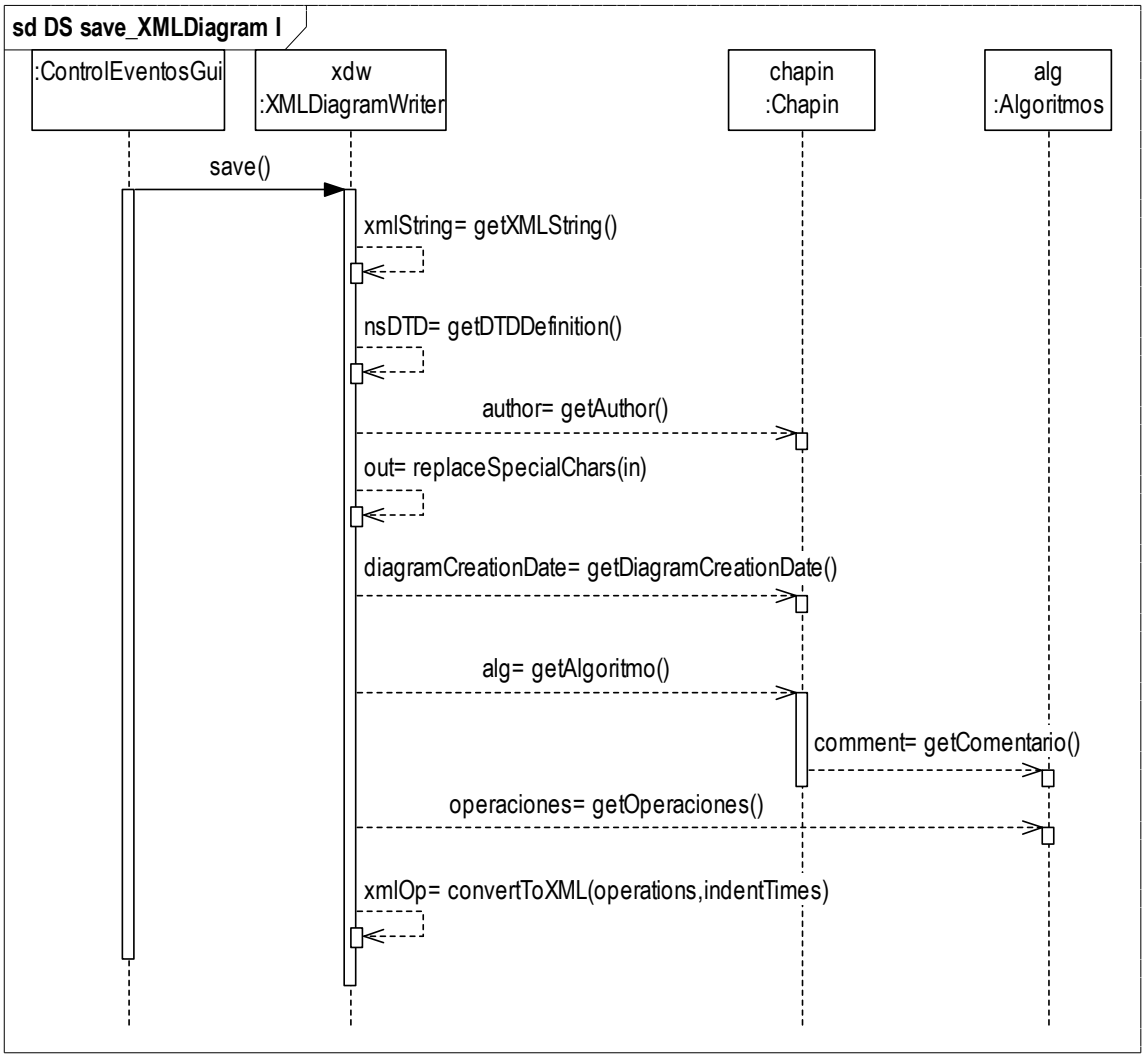

sd DS save_XMLDiagram I
:ControlEventosGui
xdw
:XMLDiagramWriter
chapin
:Chapin
alg
:Algoritmos
save()
xmlString= getXMLString()
nsDTD= getDTDDefinition()
author= getAuthor()
out= replaceSpecialChars(in)
diagramCreationDate= getDiagramCreationDate()
alg= getAlgoritmo()
comment= getComentario()
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)

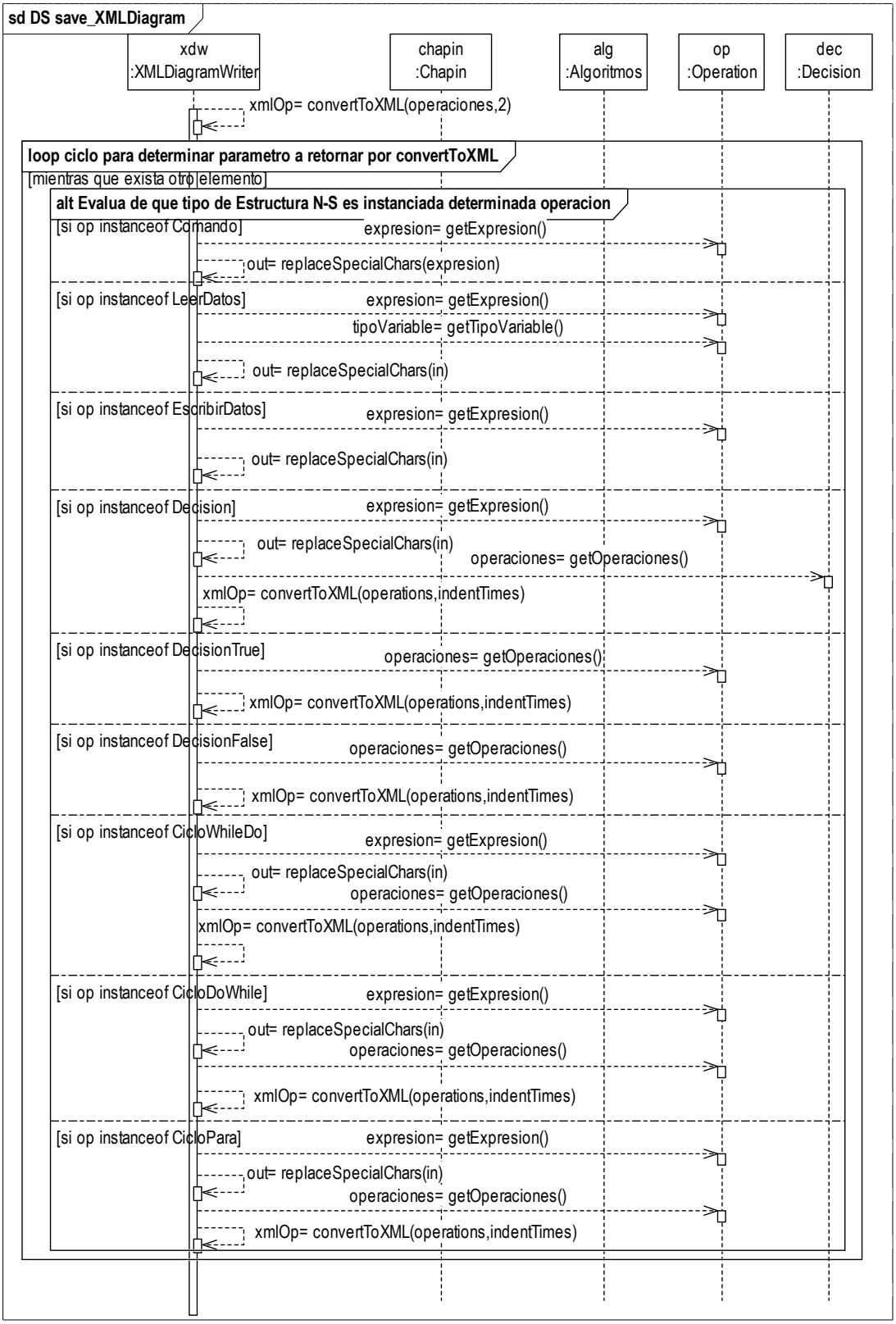

sd DS save_XMLDiagram
xdw :XMLDiagramWriter
chapin :Chapin
alg :Algoritmos
op :Operation
dec :Decision
xmlOp= convertToXML(operaciones,2)
loop ciclo para determinar parametro a retornar por convertToXML
[mientras que exista otro elemento]
alt Evalua de que tipo de Estructura N-S es instanciada determinada operacion
[si op instanceof Comando]
expresion= getExpresion()
out= replaceSpecialChars(expresion)
[si op instanceof LeerDatos]
expresion= getExpresion()
tipoVariable= getTipoVariable()
out= replaceSpecialChars(in)
[si op instanceof EscribirDatos]
expresion= getExpresion()
out= replaceSpecialChars(in)
[si op instanceof Decision]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof DecisionTrue]
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof DecisionFalse]
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof CicloWhileDo]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof CicloDoWhile]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof CicloPara]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)

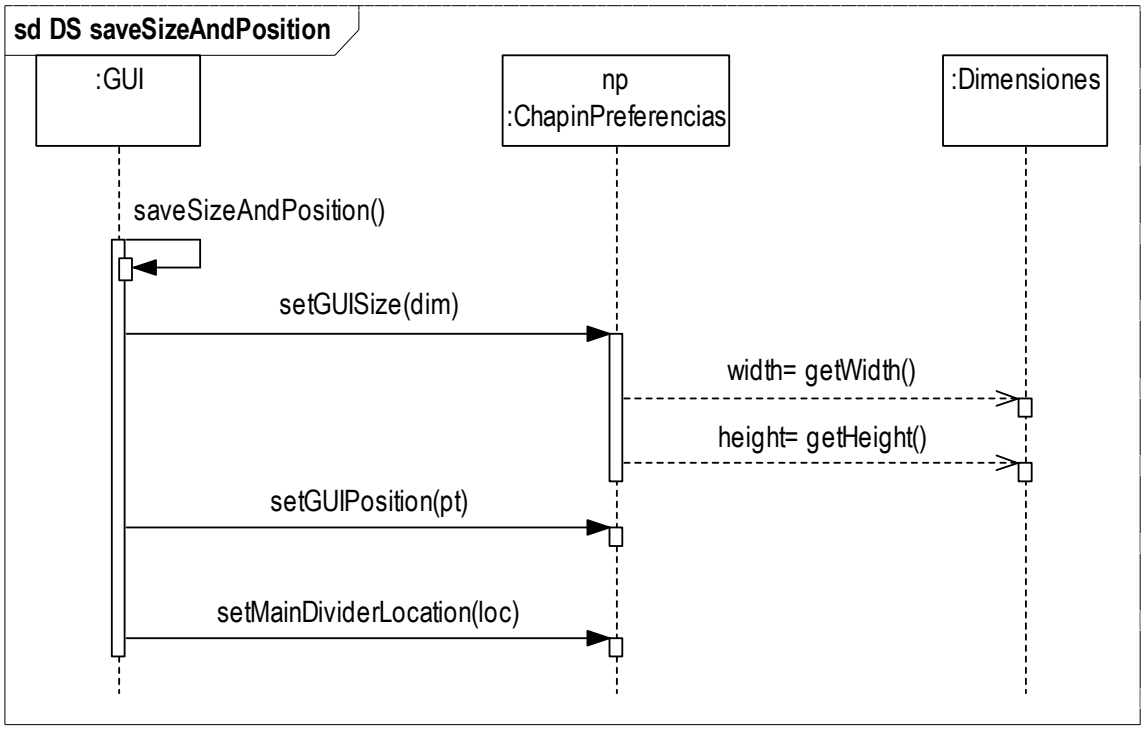

sd DS saveSizeAndPosition
:GUI
np
:ChapinPreferencias
:Dimensiones
saveSizeAndPosition()
setGUISize(dim)
width= getWidth()
height= getHeight()
setGUIPosition(pt)
setMainDividerLocation(loc)

10.2 DIAGRAMA DE SECUENCIA DE GUARDAR

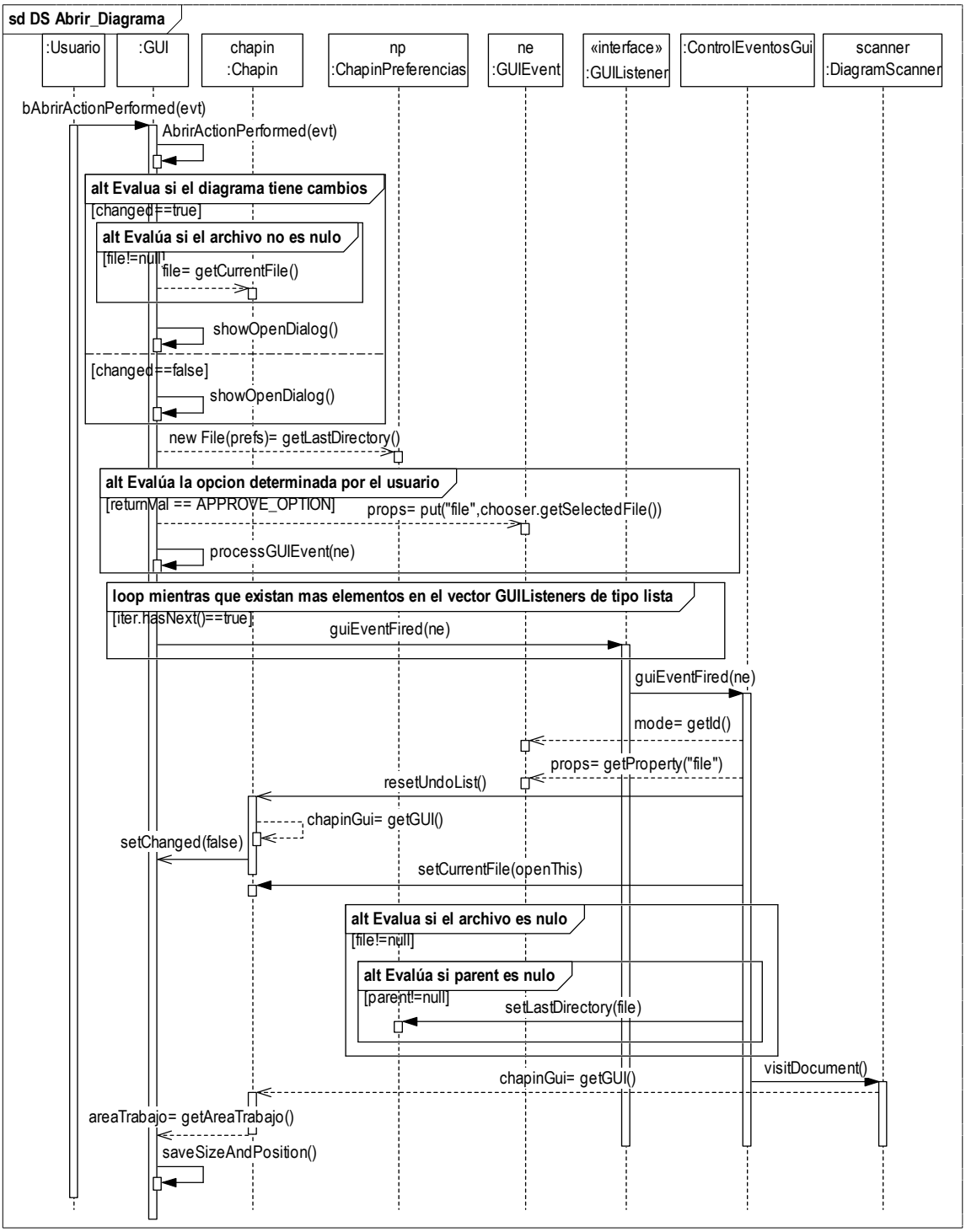



sd DS save_XMLDiagram I
:ControlEventosGui
xdw
:XMLDiagramWriter
chapin
:Chapin
alg
:Algoritmos
save()
xmlString= getXMLString()
nsDTD= getDTDDefinition()
author= getAuthor()
out= replaceSpecialChars(in)
diagramCreationDate= getDiagramCreationDate()
alg= getAlgoritmo()
comment= getComentario()
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)

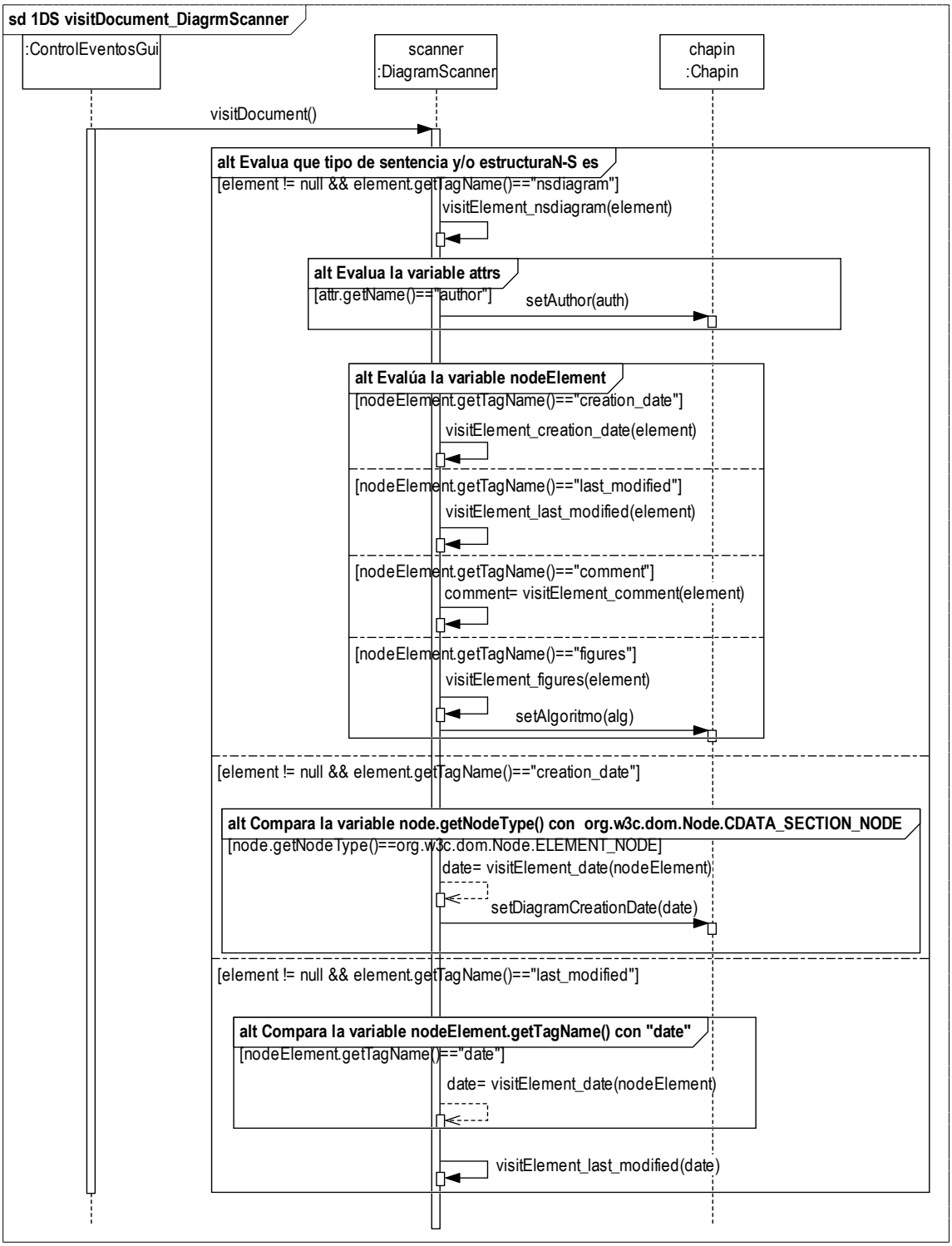

sd DS save_XMLDiagram
xdw :XMLDiagramWriter
chapin :Chapin
alg :Algoritmos
op :Operation
dec :Decision
xmlOp= convertToXML(operaciones,2)
loop ciclo para determinar parametro a retornar por convertToXML
[mientras que exista otro elemento]
alt Evalua de que tipo de Estructura N-S es instanciada determinada operacion
[si op instanceof Comando]
expresion= getExpresion()
out= replaceSpecialChars(expresion)
[si op instanceof LeerDatos]
expresion= getExpresion()
tipoVariable= getTipoVariable()
out= replaceSpecialChars(in)
[si op instanceof EscribirDatos]
expresion= getExpresion()
out= replaceSpecialChars(in)
[si op instanceof Decision]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof DecisionTrue]
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof DecisionFalse]
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof CicloWhileDo]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof CicloDoWhile]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)
[si op instanceof CicloPara]
expresion= getExpresion()
out= replaceSpecialChars(in)
operaciones= getOperaciones()
xmlOp= convertToXML(operations,indentTimes)

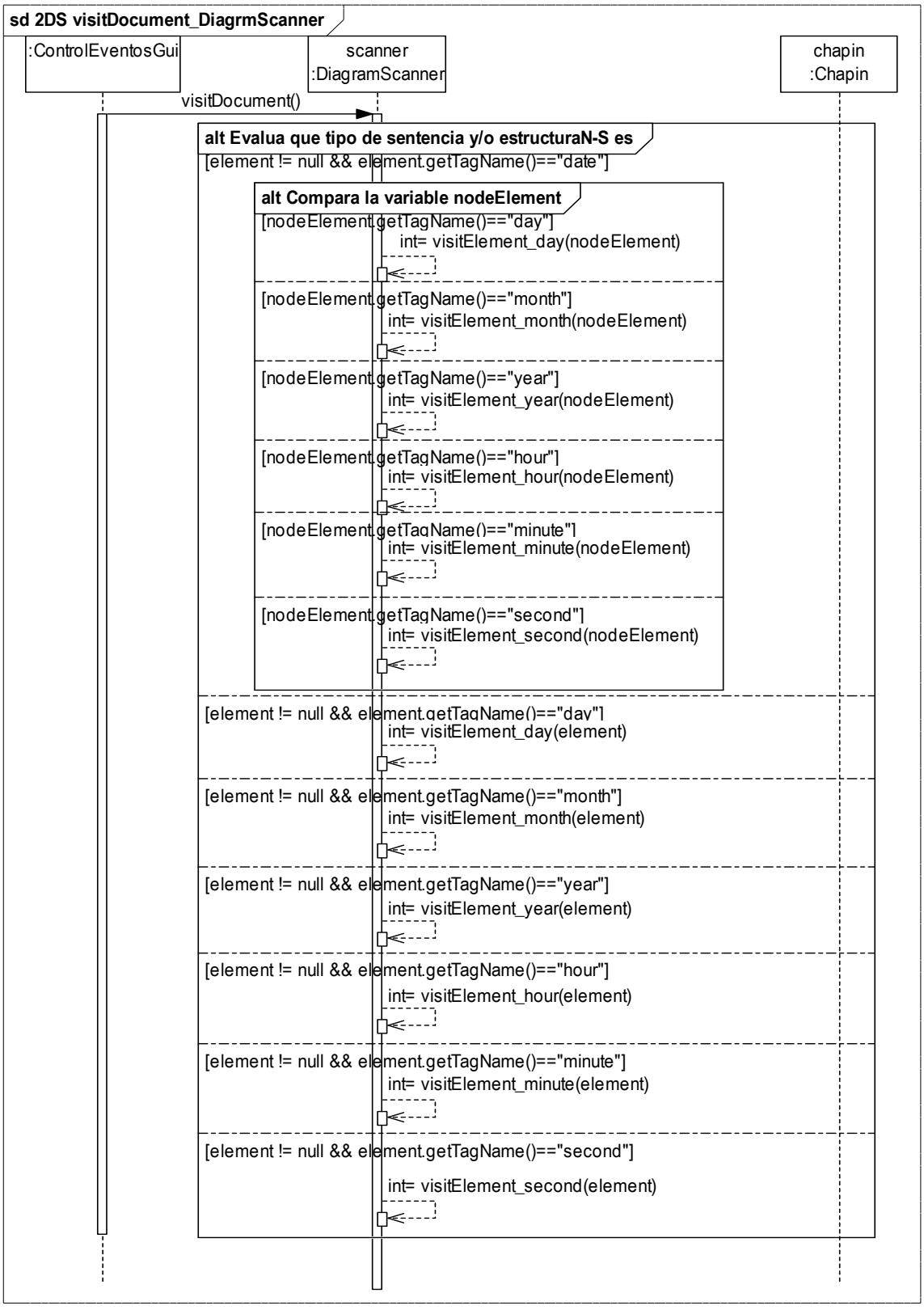


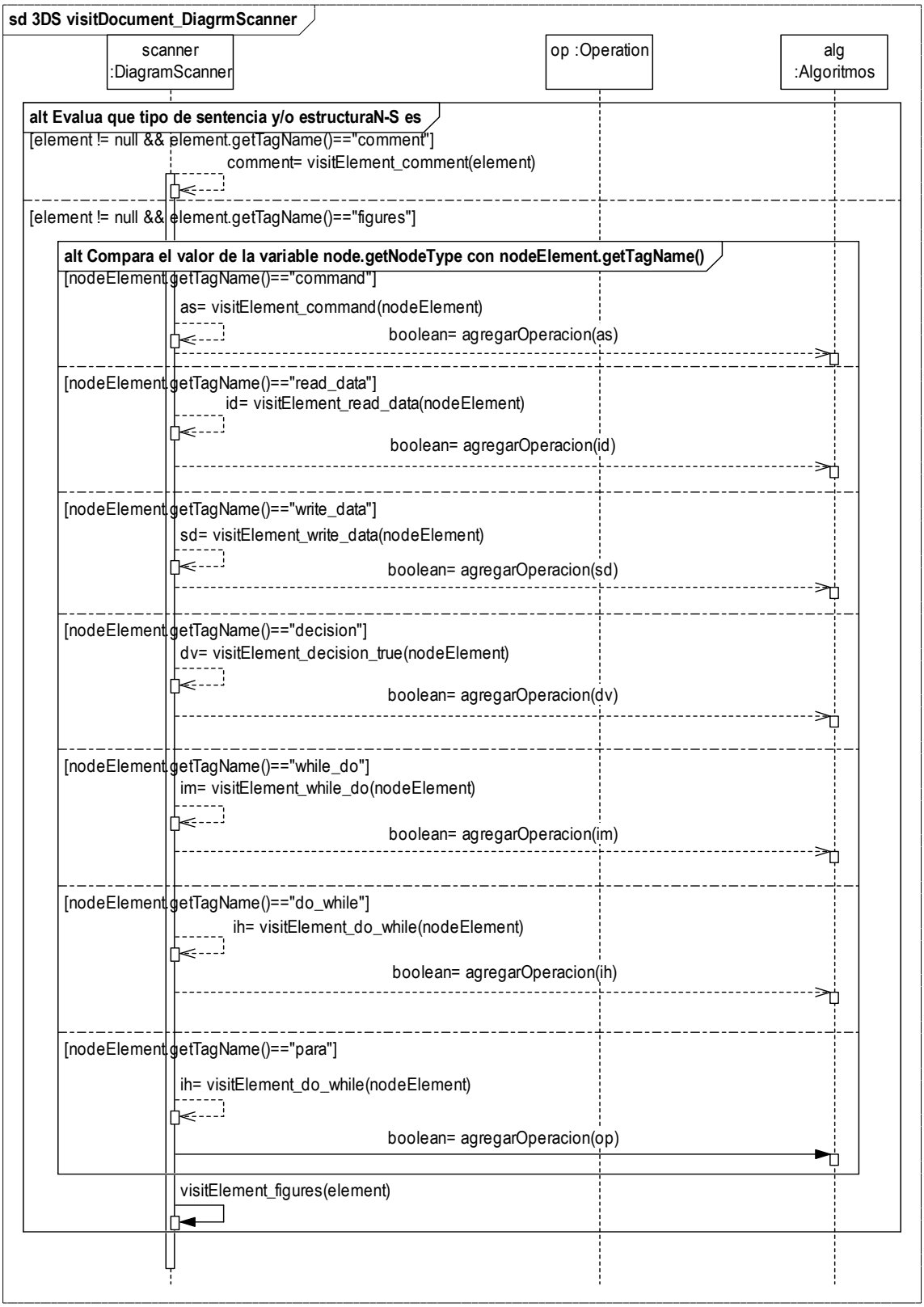
10.3 DIAGRAMA DE SECUENCIA DE ABRIR

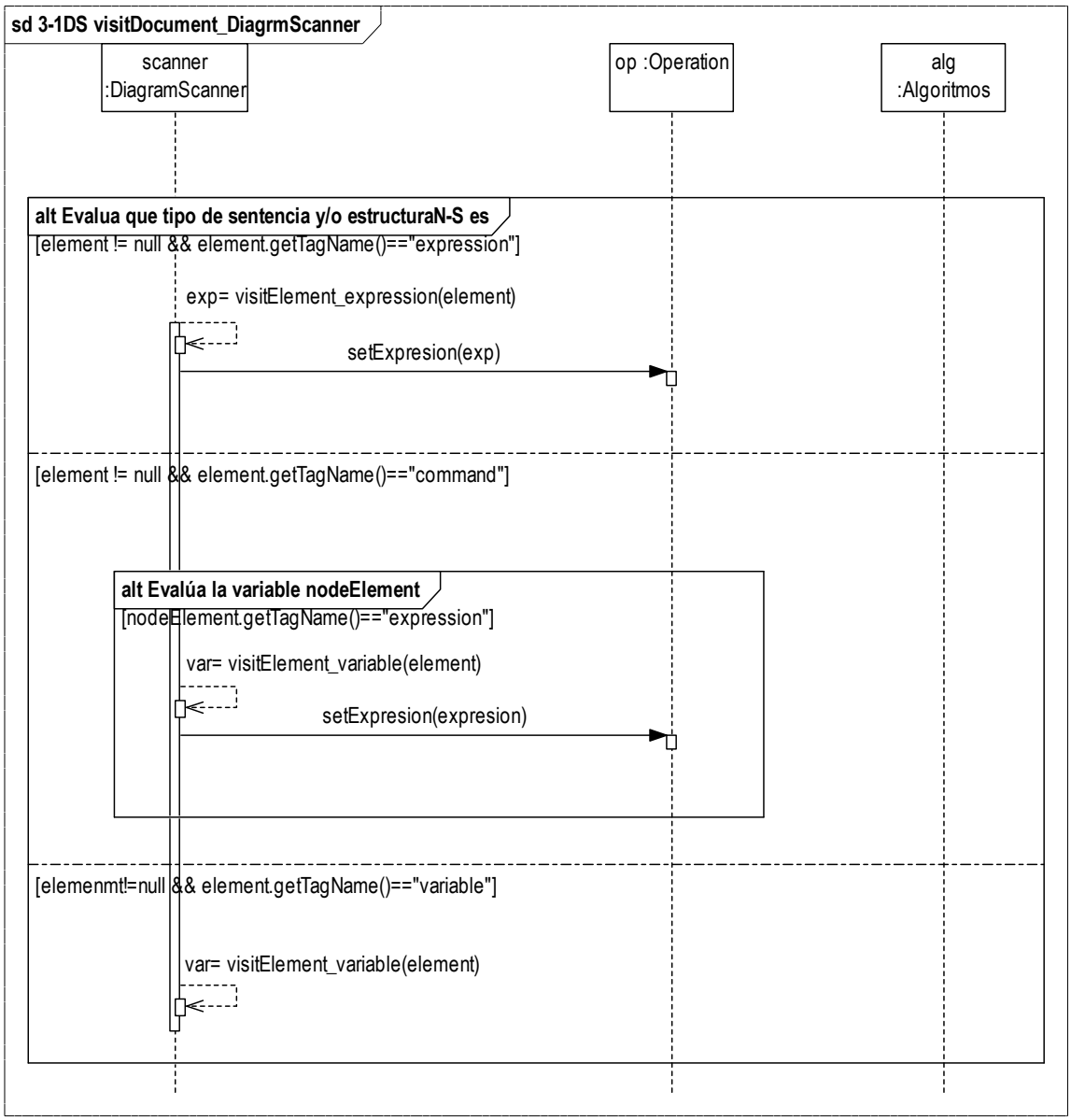


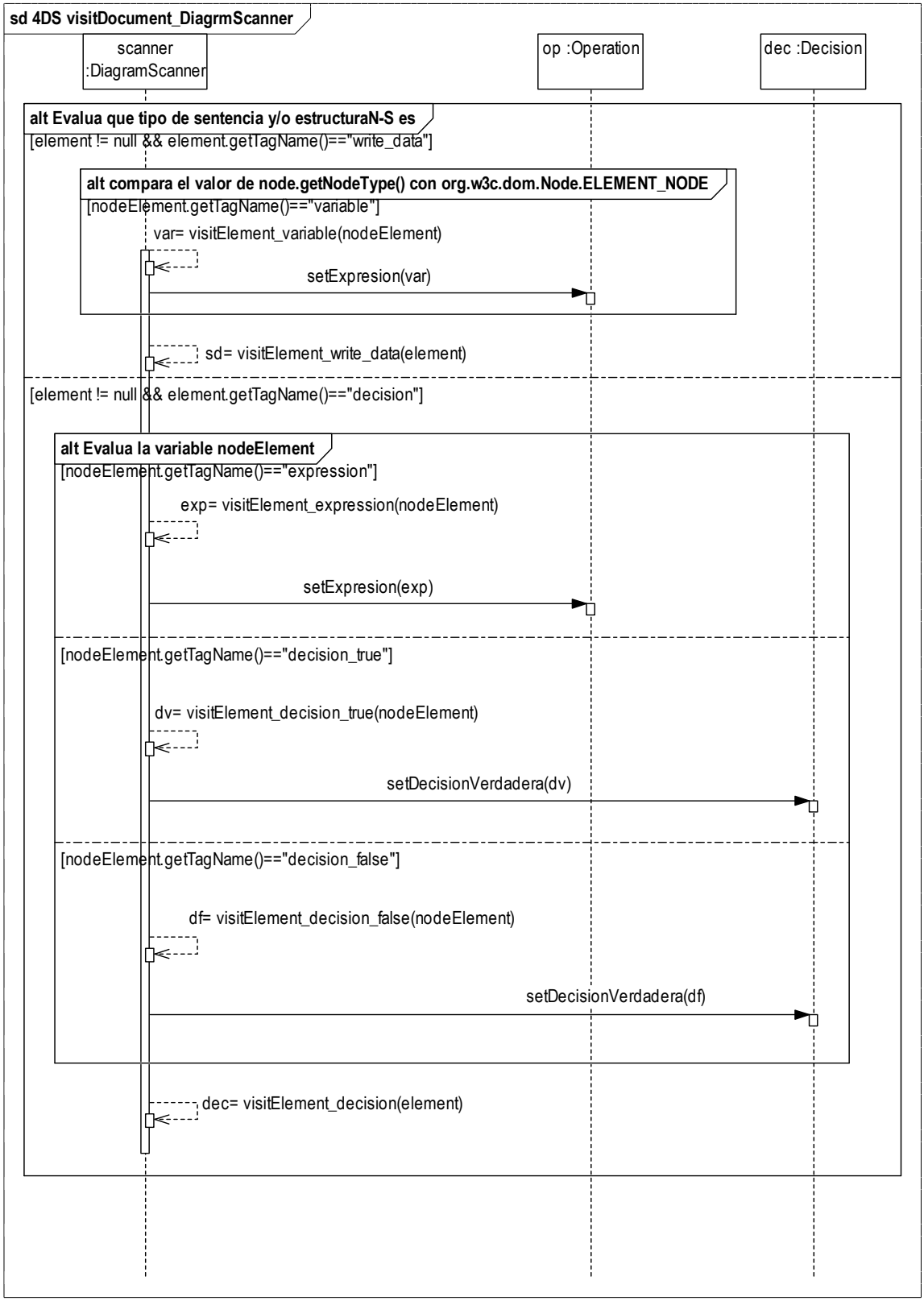
10.3.1 Proceso visitDocument () de DiagramScanner

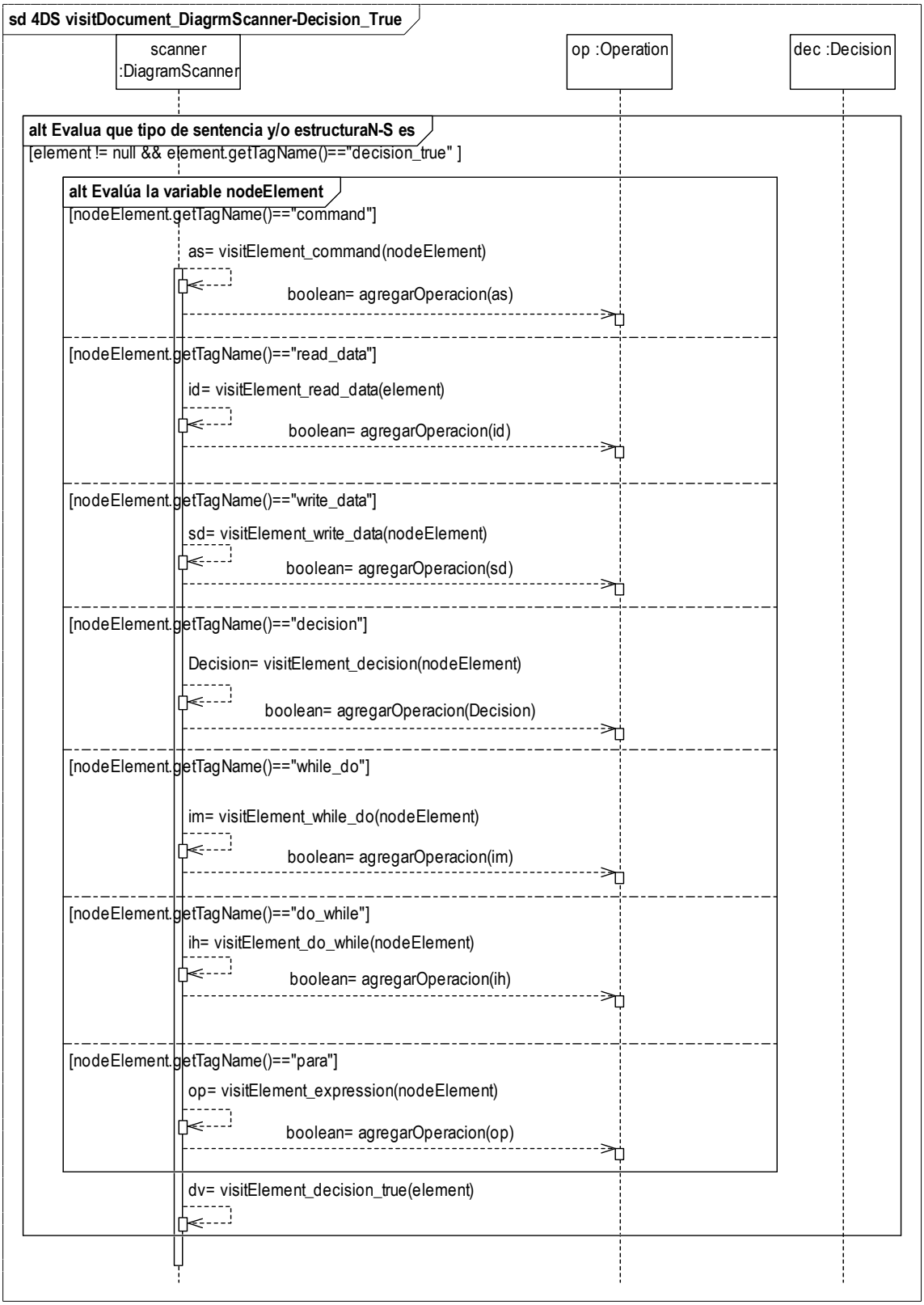

sd 1DS visitDocument_DiagrmScanner
:ControlEventosGui
scanner
:DiagramScanner
chapin
:Chapin
visitDocument()
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="nsdiagram"]
visitElement_nsdiagram(element)
alt Evalua la variable attrs
[attr.getName()=="author"]
setAuthor(auth)
alt Evalúa la variable nodeElement
[nodeElement.getTagName()=="creation_date"]
visitElement_creation_date(element)
[nodeElement.getTagName()=="last_modified"]
visitElement_last_modified(element)
[nodeElement.getTagName()=="comment"]
comment= visitElement_comment(element)
[nodeElement.getTagName()=="figures"]
visitElement_figures(element)
setAlgoritmo(alg)
[element != null && element.getTagName()=="creation_date"]
alt Compara la variable node.getNodeType() con org.w3c.dom.Node.CDATA_SECTION_NODE
[node.getNodeType()==org.w3c.dom.Node.ELEMENT_NODE]
date= visitElement_date(nodeElement)
setDiagramCreationDate(date)
[element != null && element.getTagName()=="last_modified"]
alt Compara la variable nodeElement.getTagName() con "date"
[nodeElement.getTagName()=="date"]
date= visitElement_date(nodeElement)
visitElement_last_modified(date)

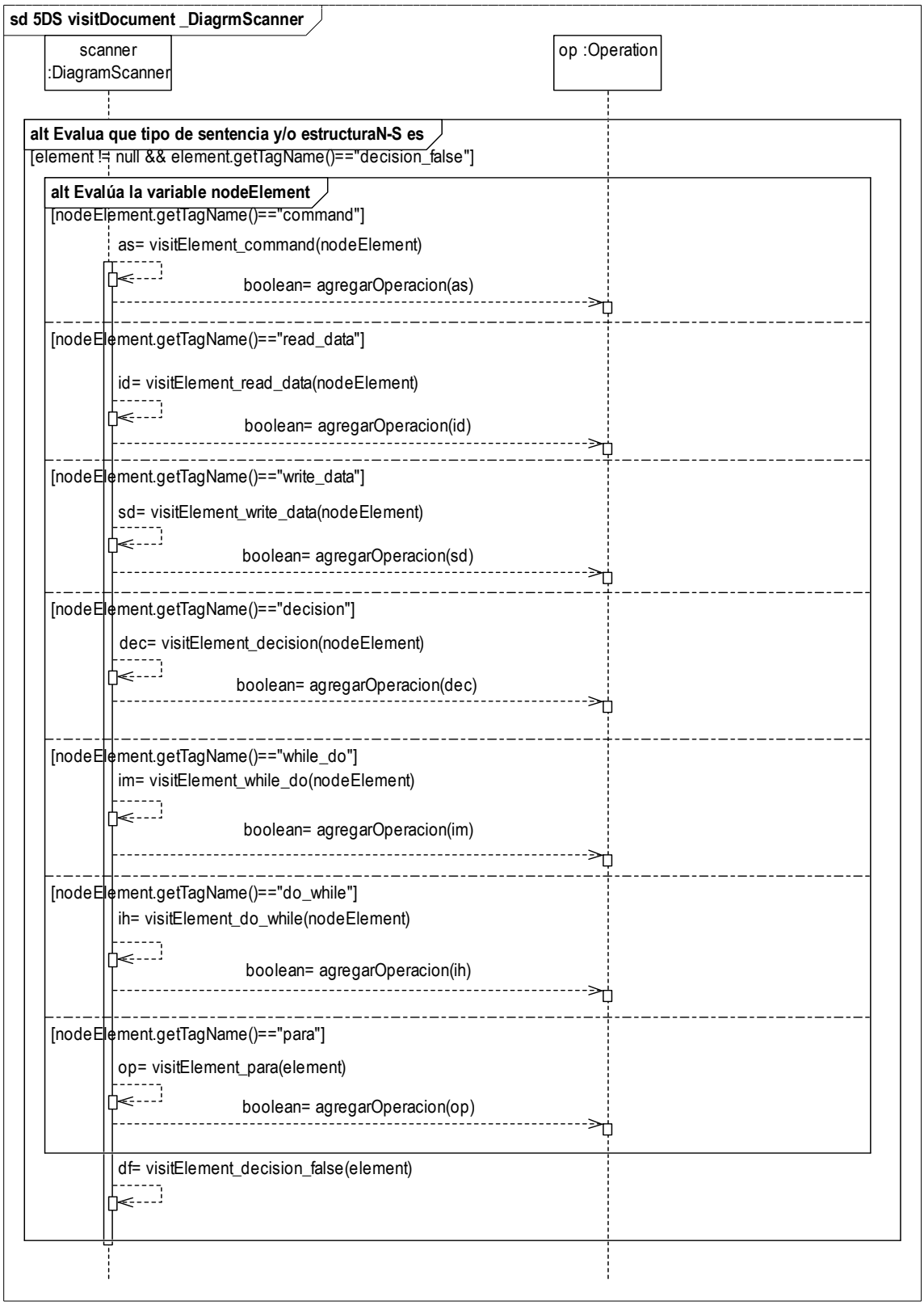

sd 2DS visitDocument_DiagrmScanner
:ControlEventosGui
scanner
:DiagramScanner
chapin
:Chapin
visitDocument()
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="date"]
alt Compara la variable nodeElement
[nodeElement.getTagName()=="day"]
int= visitElement_day(nodeElement)
[nodeElement.getTagName()=="month"]
int= visitElement_month(nodeElement)
[nodeElement.getTagName()=="year"]
int= visitElement_year(nodeElement)
[nodeElement.getTagName()=="hour"]
int= visitElement_hour(nodeElement)
[nodeElement.getTagName()=="minute"]
int= visitElement_minute(nodeElement)
[nodeElement.getTagName()=="second"]
int= visitElement_second(nodeElement)
[element != null && element.getTagName()=="day"]
int= visitElement_day(element)
[element != null && element.getTagName()=="month"]
int= visitElement_month(element)
[element != null && element.getTagName()=="year"]
int= visitElement_year(element)
[element != null && element.getTagName()=="hour"]
int= visitElement_hour(element)
[element != null && element.getTagName()=="minute"]
int= visitElement_minute(element)
[element != null && element.getTagName()=="second"]
int= visitElement_second(element)


sd 3DS visitDocument_DiagrmScanner
scanner :DiagramScanner
op :Operation
alg :Algoritmos
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="comment"]
comment= visitElement_comment(element)
[element != null && element.getTagName()=="figures"]
alt Compara el valor de la variable node.getNodeType con nodeElement.getTagName()
[nodeElement.getTagName()=="command"]
as= visitElement_command(nodeElement)
boolean= agregarOperacion(as)
[nodeElement.getTagName()=="read_data"]
id= visitElement_read_data(nodeElement)
boolean= agregarOperacion(id)
[nodeElement.getTagName()=="write_data"]
sd= visitElement_write_data(nodeElement)
boolean= agregarOperacion(sd)
[nodeElement.getTagName()=="decision"]
dv= visitElement_decision_true(nodeElement)
boolean= agregarOperacion(dv)
[nodeElement.getTagName()=="while_do"]
im= visitElement_while_do(nodeElement)
boolean= agregarOperacion(im)
[nodeElement.getTagName()=="do_while"]
ih= visitElement_do_while(nodeElement)
boolean= agregarOperacion(ih)
[nodeElement.getTagName()=="para"]
ih= visitElement_do_while(nodeElement)
boolean= agregarOperacion(op)
visitElement_figures(element)

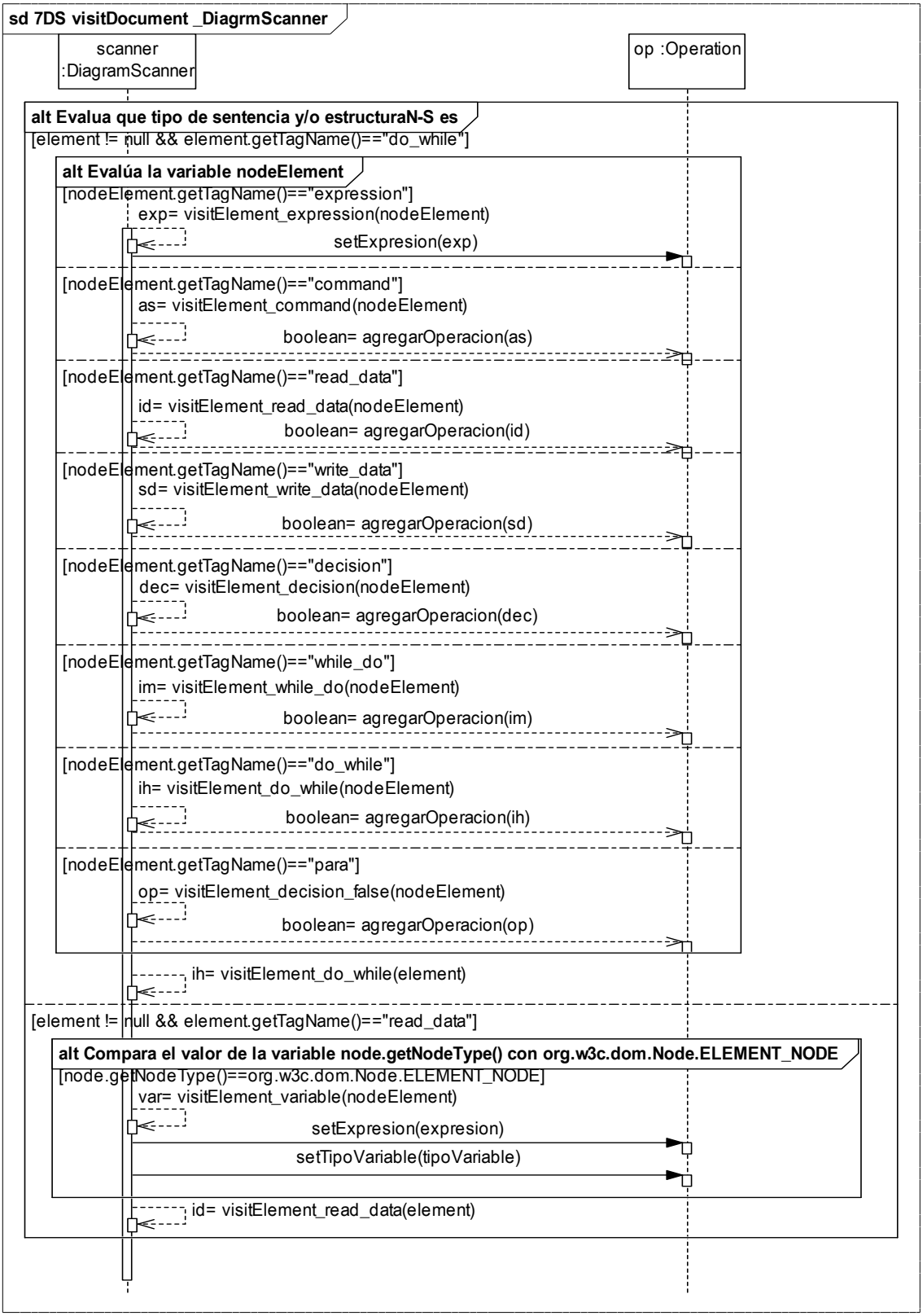

sd 3-1DS visitDocument_DiagrmScanner
scanner
:DiagramScanner
op :Operation
alg
:Algoritmos
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="expression"]
exp= visitElement_expression(element)
setExpresion(exp)
[element != null && element.getTagName()=="command"]
alt Evalúa la variable nodeElement
[nodeElement.getTagName()=="expression"]
var= visitElement_variable(element)
setExpresion(expresion)
[elemenmt!=null && element.getTagName()=="variable"]
var= visitElement_variable(element)

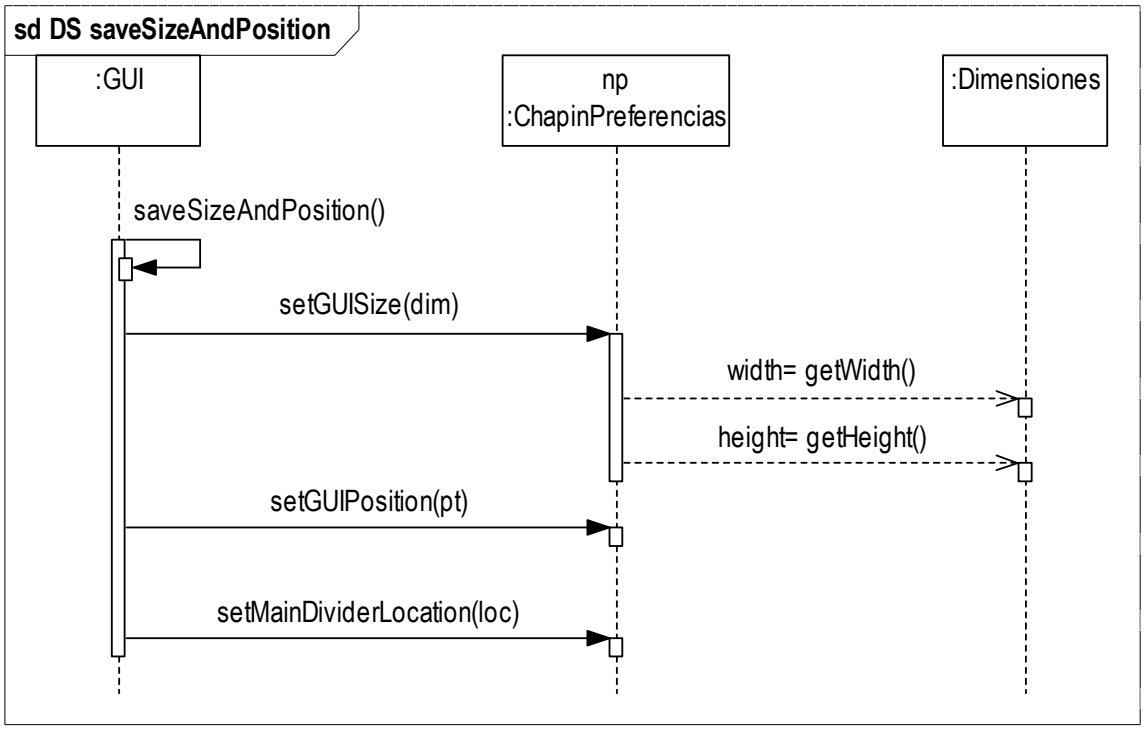

sd 4DS visitDocument_DiagrmScanner
scanner :DiagramScanner
op :Operation
dec :Decision
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="write_data"]
alt compara el valor de node.getNodeType() con org.w3c.dom.Node.ELEMENT_NODE
[nodeElement.getTagName()=="variable"]
var= visitElement_variable(nodeElement)
setExpresion(var)
sd= visitElement_write_data(element)
[element != null && element.getTagName()=="decision"]
alt Evalua la variable nodeElement
[nodeElement.getTagName()=="expression"]
exp= visitElement_expression(nodeElement)
setExpresion(exp)
[nodeElement.getTagName()=="decision_true"]
dv= visitElement_decision_true(nodeElement)
setDecisionVerdadera(dv)
[nodeElement.getTagName()=="decision_false"]
df= visitElement_decision_false(nodeElement)
setDecisionVerdadera(df)
dec= visitElement_decision(element)


sd 4DS visitDocument_DiagrmScanner-Decision_True
scanner :DiagramScanner
op :Operation
dec :Decision
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="decision_true"]
alt Evalúa la variable nodeElement
[nodeElement.getTagName()=="command"]
as= visitElement_command(nodeElement)
boolean= agregarOperacion(as)
[nodeElement.getTagName()=="read_data"]
id= visitElement_read_data(element)
boolean= agregarOperacion(id)
[nodeElement.getTagName()=="write_data"]
sd= visitElement_write_data(nodeElement)
boolean= agregarOperacion(sd)
[nodeElement.getTagName()=="decision"]
Decision= visitElement_decision(nodeElement)
boolean= agregarOperacion(Decision)
[nodeElement.getTagName()=="while_do"]
im= visitElement_while_do(nodeElement)
boolean= agregarOperacion(im)
[nodeElement.getTagName()=="do_while"]
ih= visitElement_do_while(nodeElement)
boolean= agregarOperacion(ih)
[nodeElement.getTagName()=="para"]
op= visitElement_expression(nodeElement)
boolean= agregarOperacion(op)
dv= visitElement_decision_true(element)

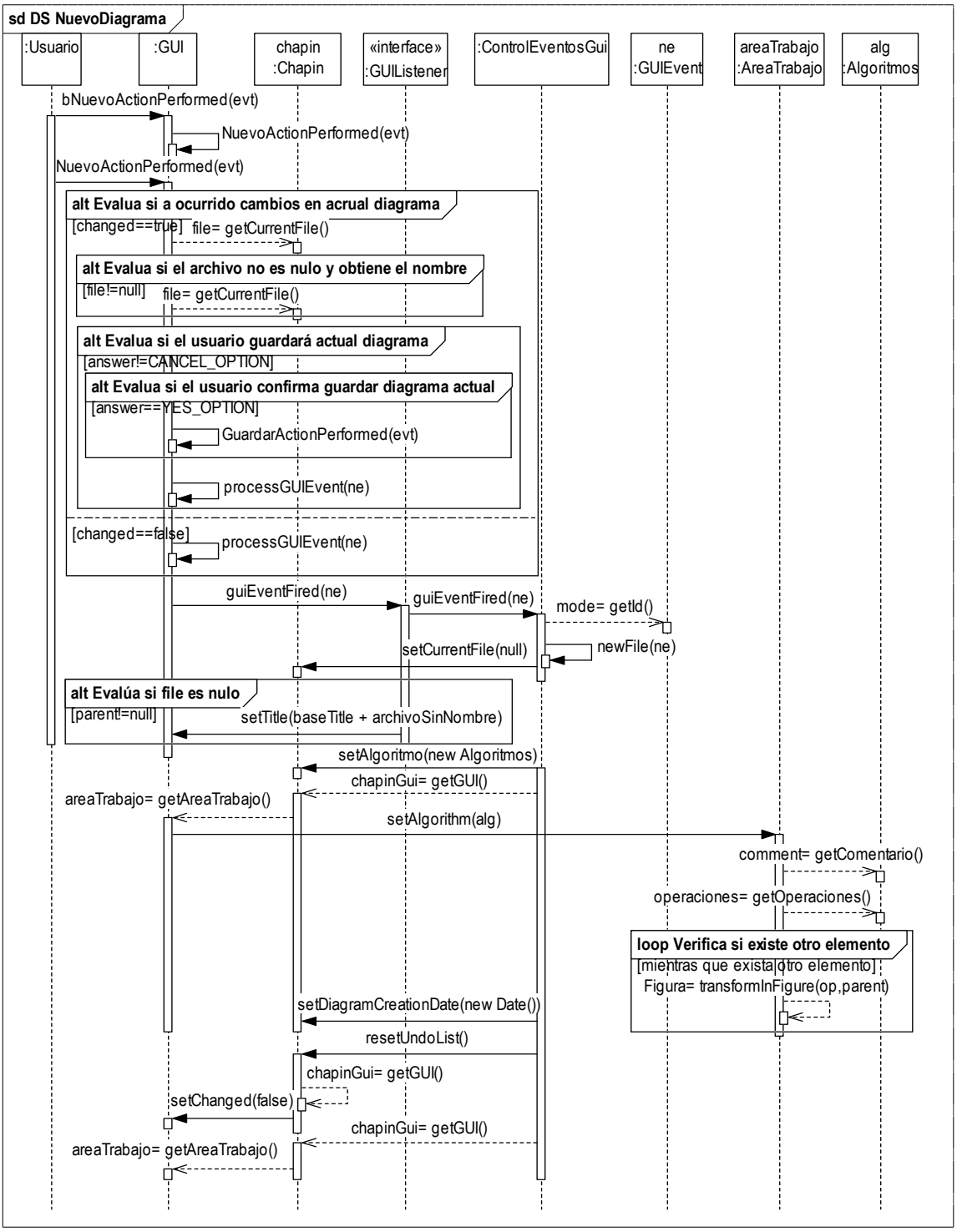

sd 5DS visitDocument _DiagrmScanner
scanner :DiagramScanner
op :Operation
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="decision_false"]
alt Evalúa la variable nodeElement
[nodeElement.getTagName()=="command"]
as= visitElement_command(nodeElement)
boolean= agregarOperacion(as)
[nodeElement.getTagName()=="read_data"]
id= visitElement_read_data(nodeElement)
boolean= agregarOperacion(id)
[nodeElement.getTagName()=="write_data"]
sd= visitElement_write_data(nodeElement)
boolean= agregarOperacion(sd)
[nodeElement.getTagName()=="decision"]
dec= visitElement_decision(nodeElement)
boolean= agregarOperacion(dec)
[nodeElement.getTagName()=="while_do"]
im= visitElement_while_do(nodeElement)
boolean= agregarOperacion(im)
[nodeElement.getTagName()=="do_while"]
ih= visitElement_do_while(nodeElement)
boolean= agregarOperacion(ih)
[nodeElement.getTagName()=="para"]
op= visitElement_para(element)
boolean= agregarOperacion(op)
df= visitElement_decision_false(element)


sd 6DS visitDocument_DiagrmScanner
scanner :DiagramScanner
op :Operation
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="while_do"]
alt Evalúa la variable nodeElement
[nodeElement.getTagName()=="expression"]
exp= visitElement_expression(nodeElement)
setExpresion(exp)
[nodeElement.getTagName()=="command"]
as= visitElement_command(nodeElement)
boolean= agregarOperacion(as)
[nodeElement.getTagName()=="read_data"]
id= visitElement_read_data(nodeElement)
boolean= agregarOperacion(id)
[nodeElement.getTagName()=="write_data"]
sd= visitElement_write_data(nodeElement)
boolean= agregarOperacion(sd)
[nodeElement.getTagName()=="decision"]
dec= visitElement_decision(nodeElement)
boolean= agregarOperacion(dec)
[nodeElement.getTagName()=="while_do"]
im= visitElement_while_do(nodeElement)
boolean= agregarOperacion(im)
[nodeElement.getTagName()=="do_while"]
ih= visitElement_do_while(nodeElement)
boolean= agregarOperacion(ih)
[nodeElement.getTagName()=="para"]
op= visitElement_nsdiagram(nodeElement)
boolean= agregarOperacion(op)
im= visitElement_while_do(element)

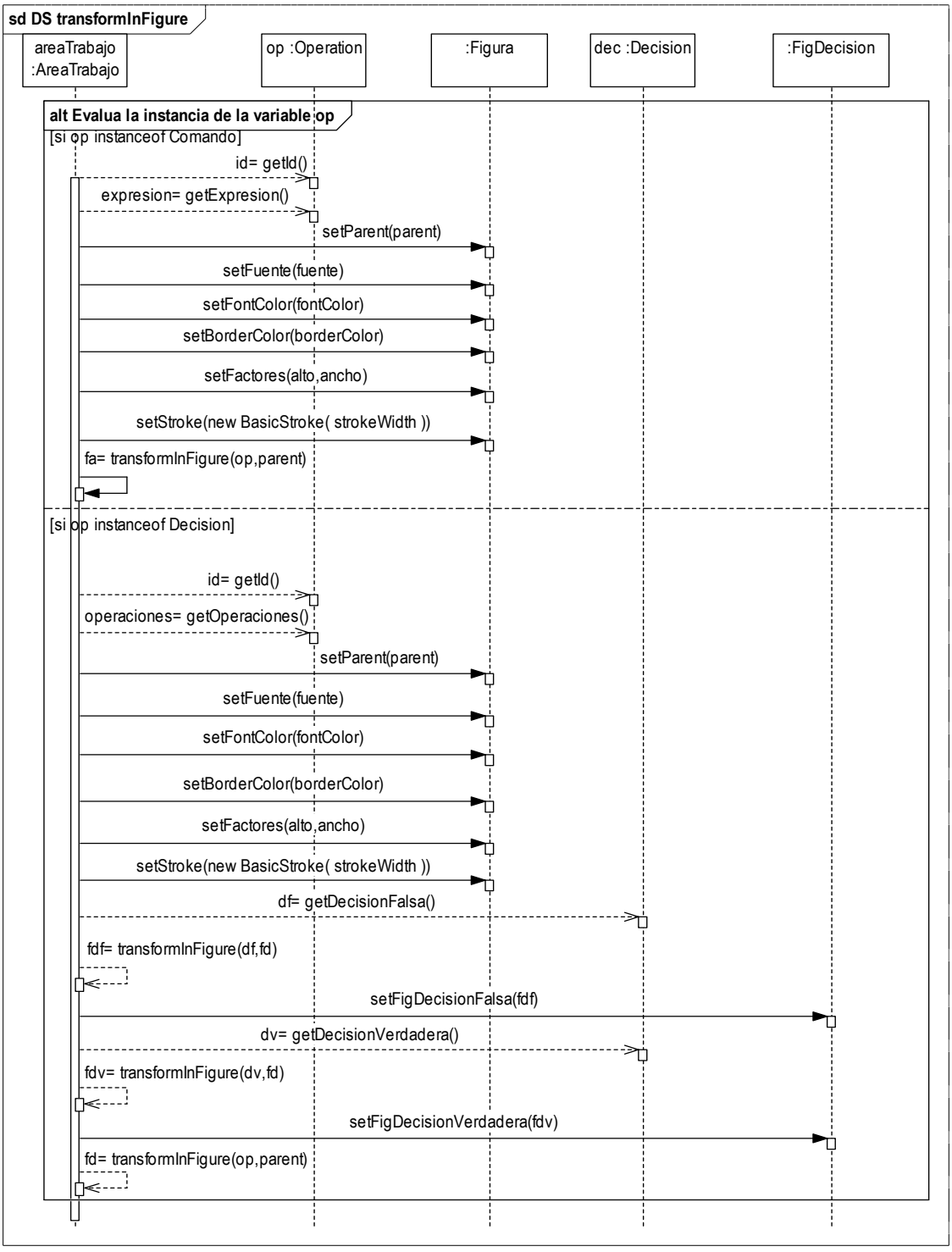

sd 7DS visitDocument _DiagrmScanner
scanner :DiagramScanner
op :Operation
alt Evalua que tipo de sentencia y/o estructuraN-S es
[element != null && element.getTagName()=="do_while"]
alt Evalúa la variable nodeElement
[nodeElement.getTagName()=="expression"]
exp= visitElement_expression(nodeElement)
setExpresion(exp)
[nodeElement.getTagName()=="command"]
as= visitElement_command(nodeElement)
boolean= agregarOperacion(as)
[nodeElement.getTagName()=="read_data"]
id= visitElement_read_data(nodeElement)
boolean= agregarOperacion(id)
[nodeElement.getTagName()=="write_data"]
sd= visitElement_write_data(nodeElement)
boolean= agregarOperacion(sd)
[nodeElement.getTagName()=="decision"]
dec= visitElement_decision(nodeElement)
boolean= agregarOperacion(dec)
[nodeElement.getTagName()=="while_do"]
im= visitElement_while_do(nodeElement)
boolean= agregarOperacion(im)
[nodeElement.getTagName()=="do_while"]
ih= visitElement_do_while(nodeElement)
boolean= agregarOperacion(ih)
[nodeElement.getTagName()=="para"]
op= visitElement_decision_false(nodeElement)
boolean= agregarOperacion(op)
ih= visitElement_do_while(element)
[element != null && element.getTagName()=="read_data"]
alt Compara el valor de la variable node.getNodeType() con org.w3c.dom.Node.ELEMENT_NODE
[node.getNodeType()==org.w3c.dom.Node.ELEMENT_NODE]
var= visitElement_variable(nodeElement)
setExpresion(expresion)
setTipoVariable(tipoVariable)
id= visitElement_read_data(element)

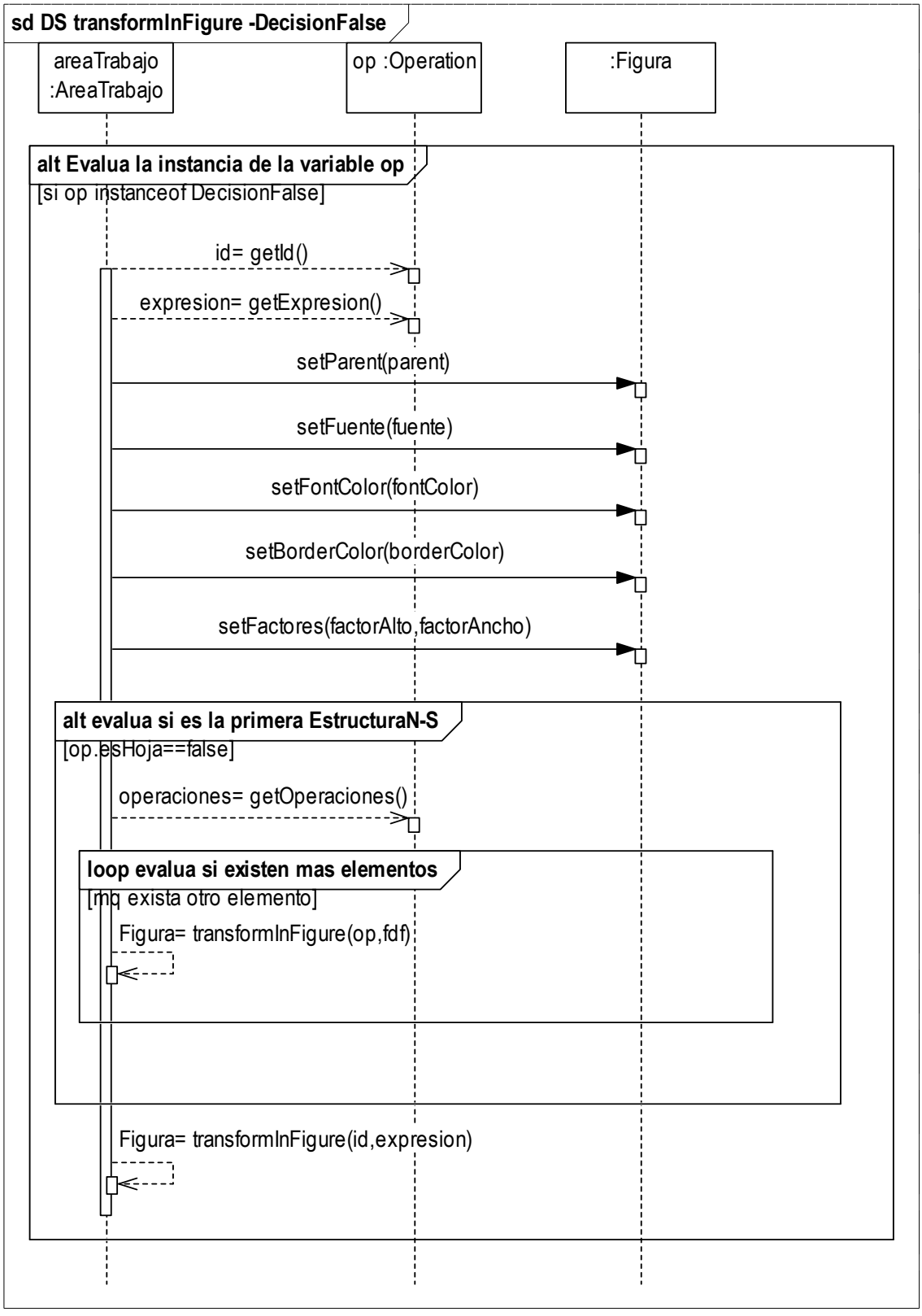


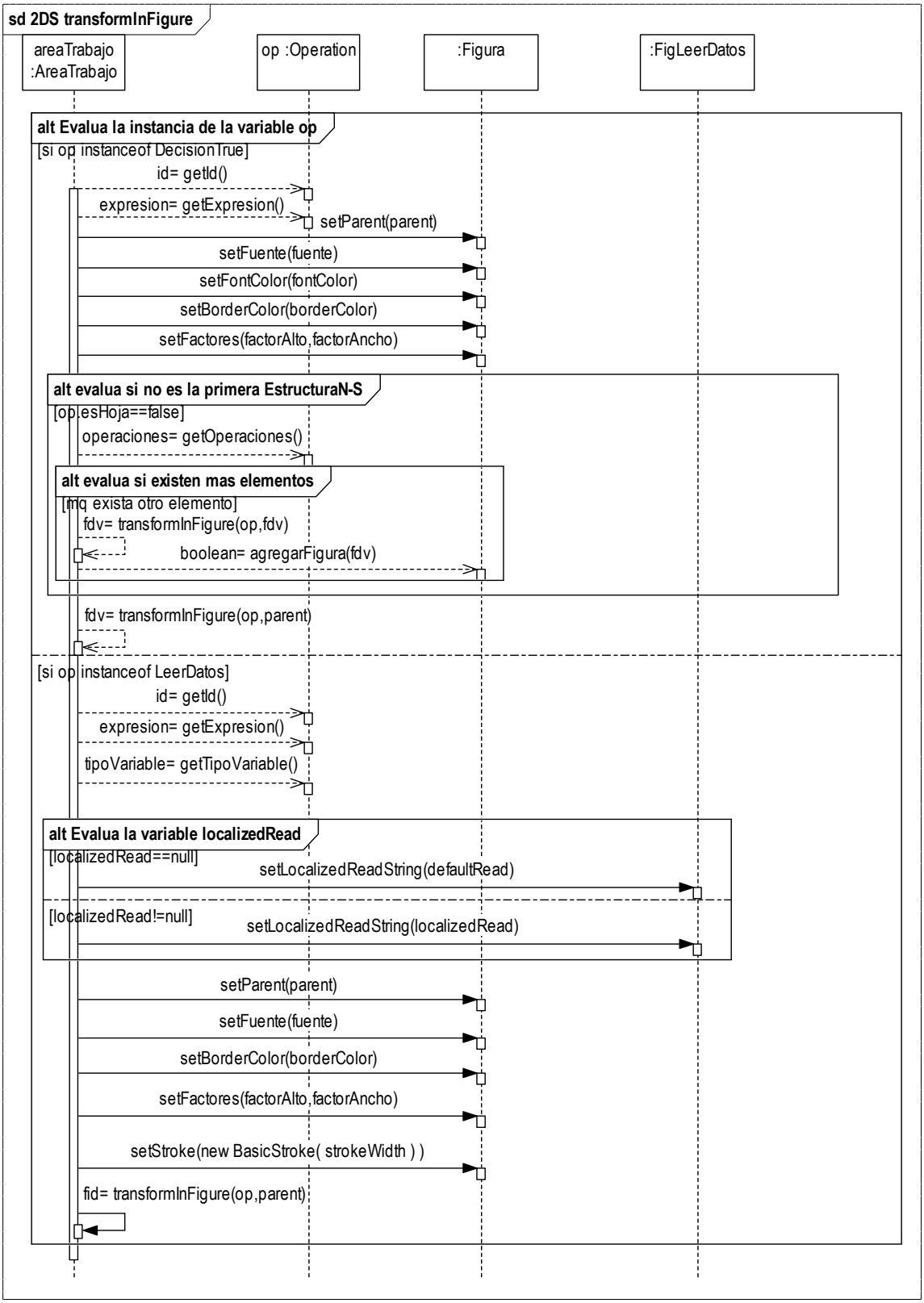
10.4 DIAGRAMA DE SECUENCIA DE NUEVO DIAGRAMA

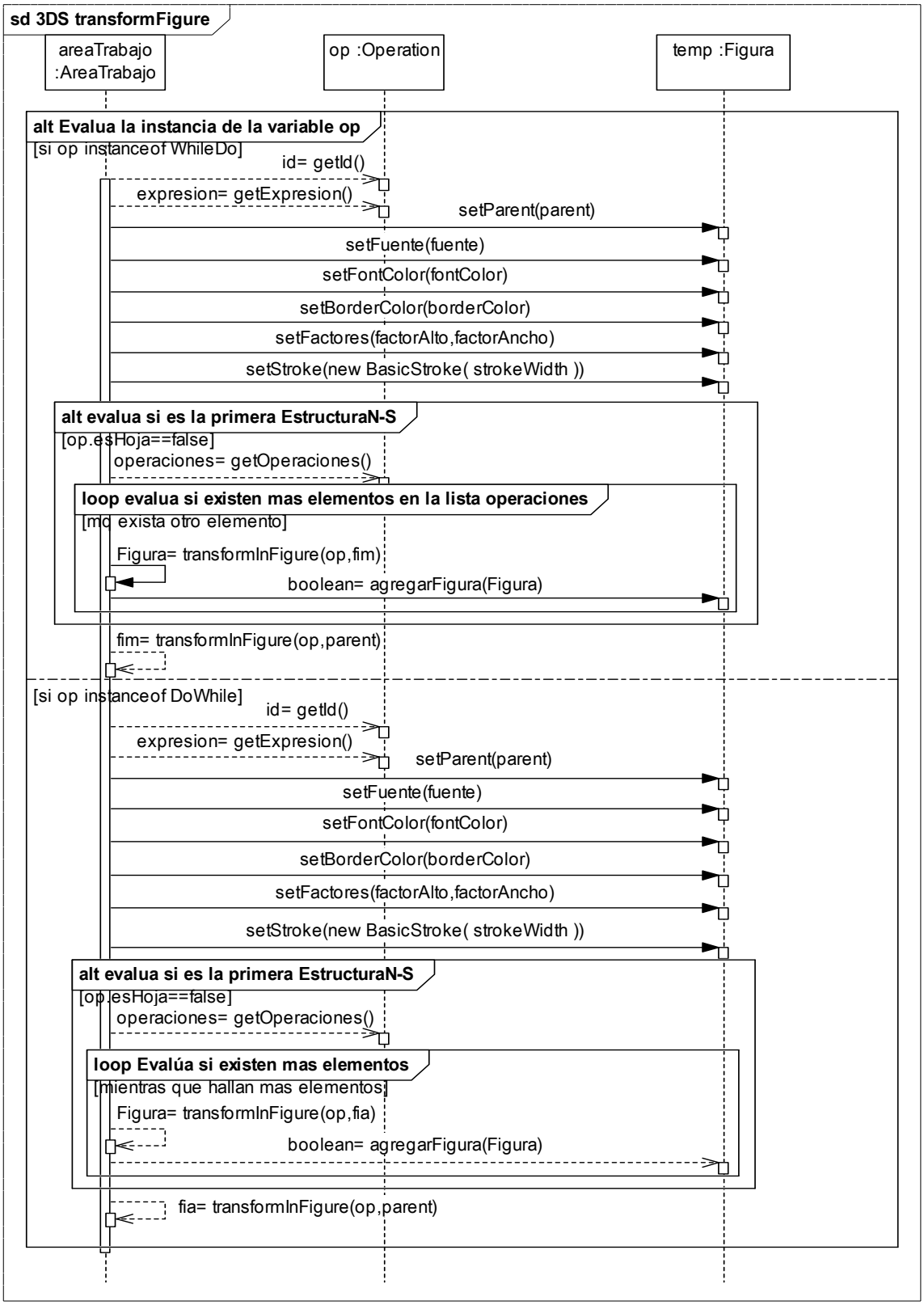


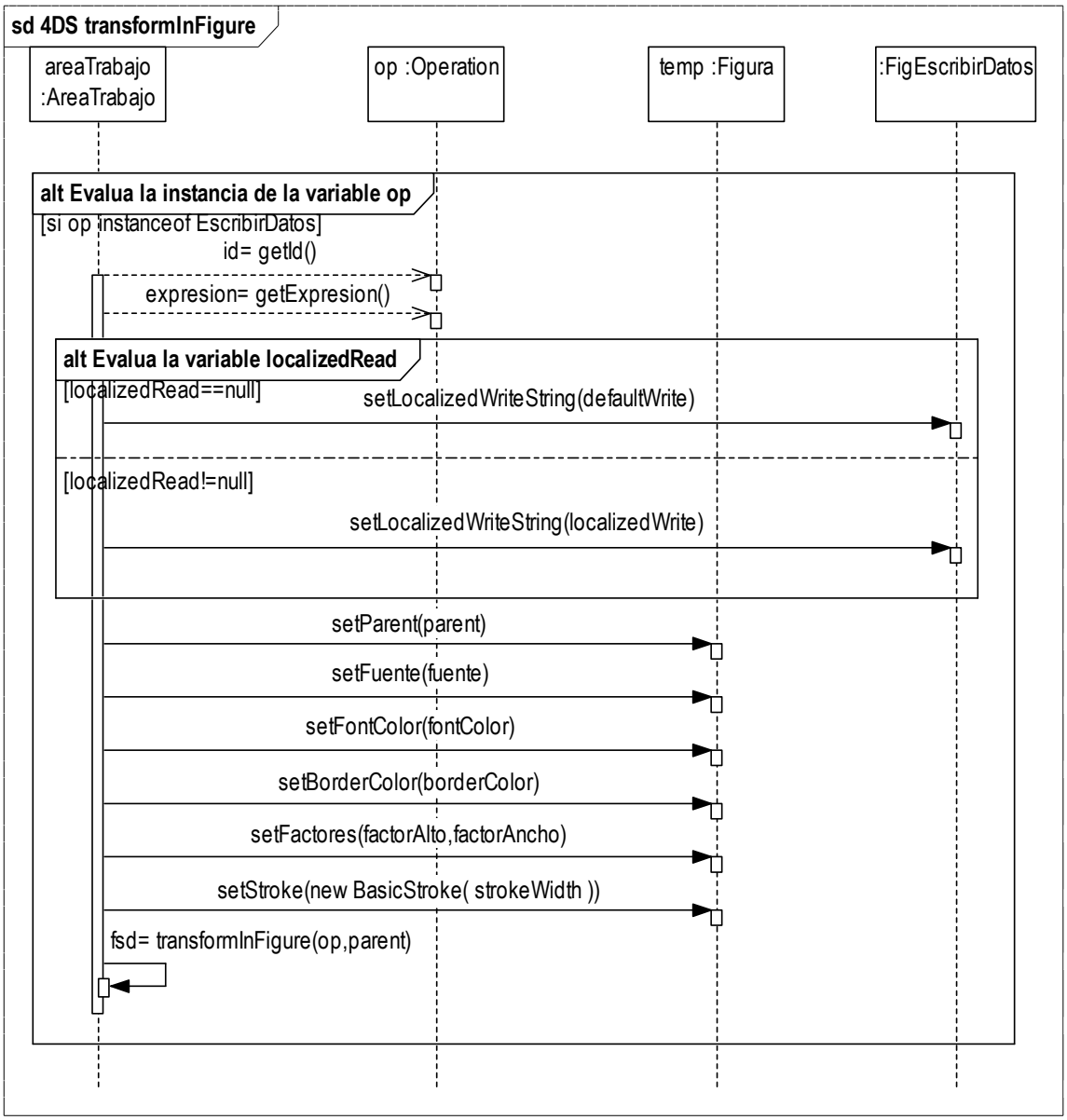
10.4.1 Proceso transformInFigure() de AreaTrabajo

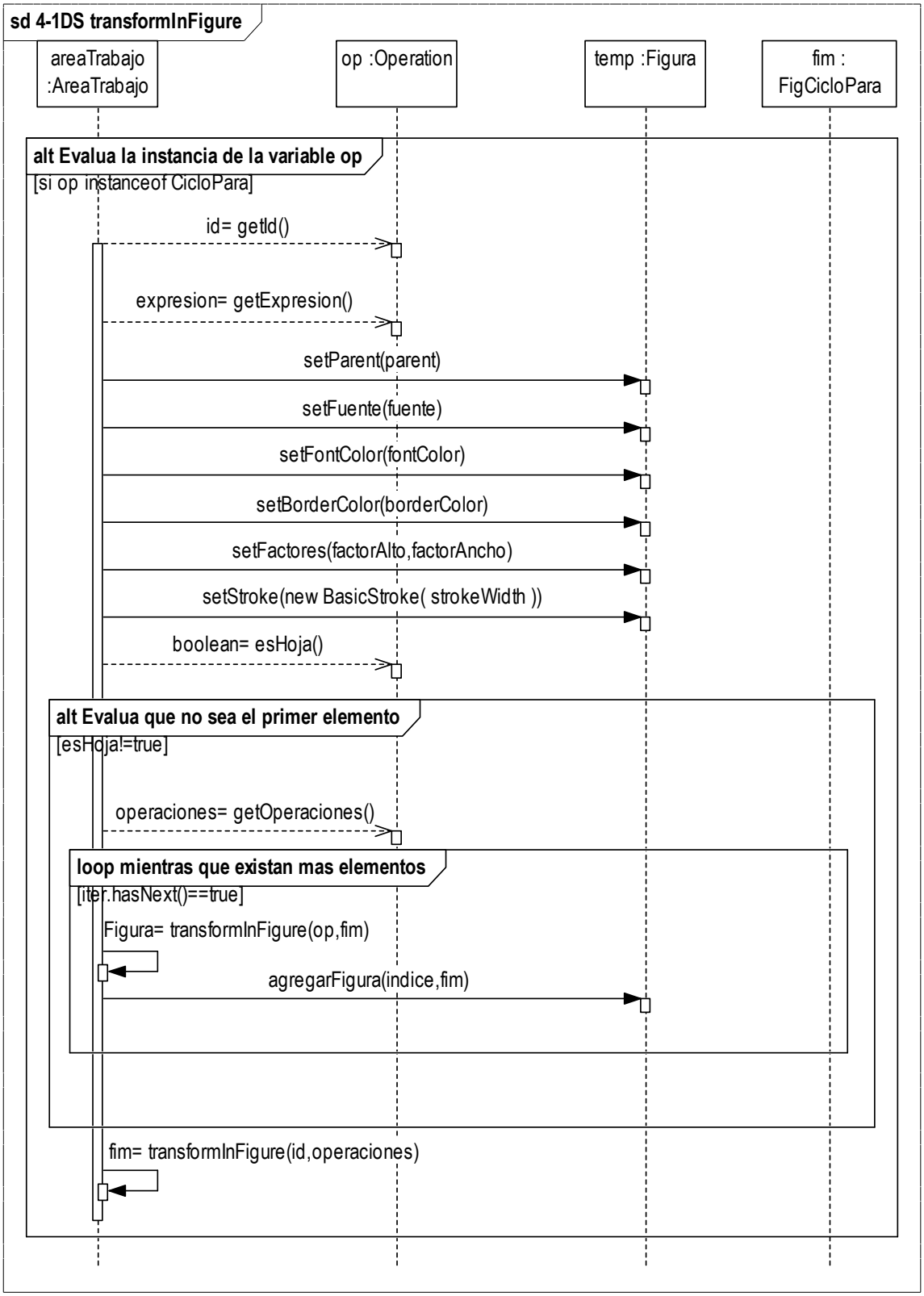

sd DS transformInFigure
areaTrabajo :AreaTrabajo
op :Operation
:Figura
dec :Decision
:FigDecision
alt Evalua la instancia de la variable op
[si op instanceof Comando]
id= getId()
expresion= getExpresion()
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(alto,ancho)
setStroke(new BasicStroke(strokeWidth))
fa= transformInFigure(op,parent)
[si op instanceof Decision]
id= getId()
operaciones= getOperaciones()
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(alto,ancho)
setStroke(new BasicStroke(strokeWidth))
df= getDecisionFalsa()
fdf= transformInFigure(df,fd)
setFigDecisionFalsa(fdf)
dv= getDecisionVerdadera()
fdv= transformInFigure(dv,fd)
setFigDecisionVerdadera(fdv)
fd= transformInFigure(op,parent)


sd DS transformInFigure -DecisionFalse
areaTrabajo
:AreaTrabajo
op :Operation
:Figura
alt Evalua la instancia de la variable op
[si op instanceof DecisionFalse]
id= getId()
expresion= getExpresion()
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(factorAlto,factorAncho)
alt evalua si es la primera EstructuraN-S
[op.esHoja==false]
operaciones= getOperaciones()
loop evalua si existen mas elementos
[mq exista otro elemento]
Figura= transformInFigure(op,fdf)
Figura= transformInFigure(id,expresion)

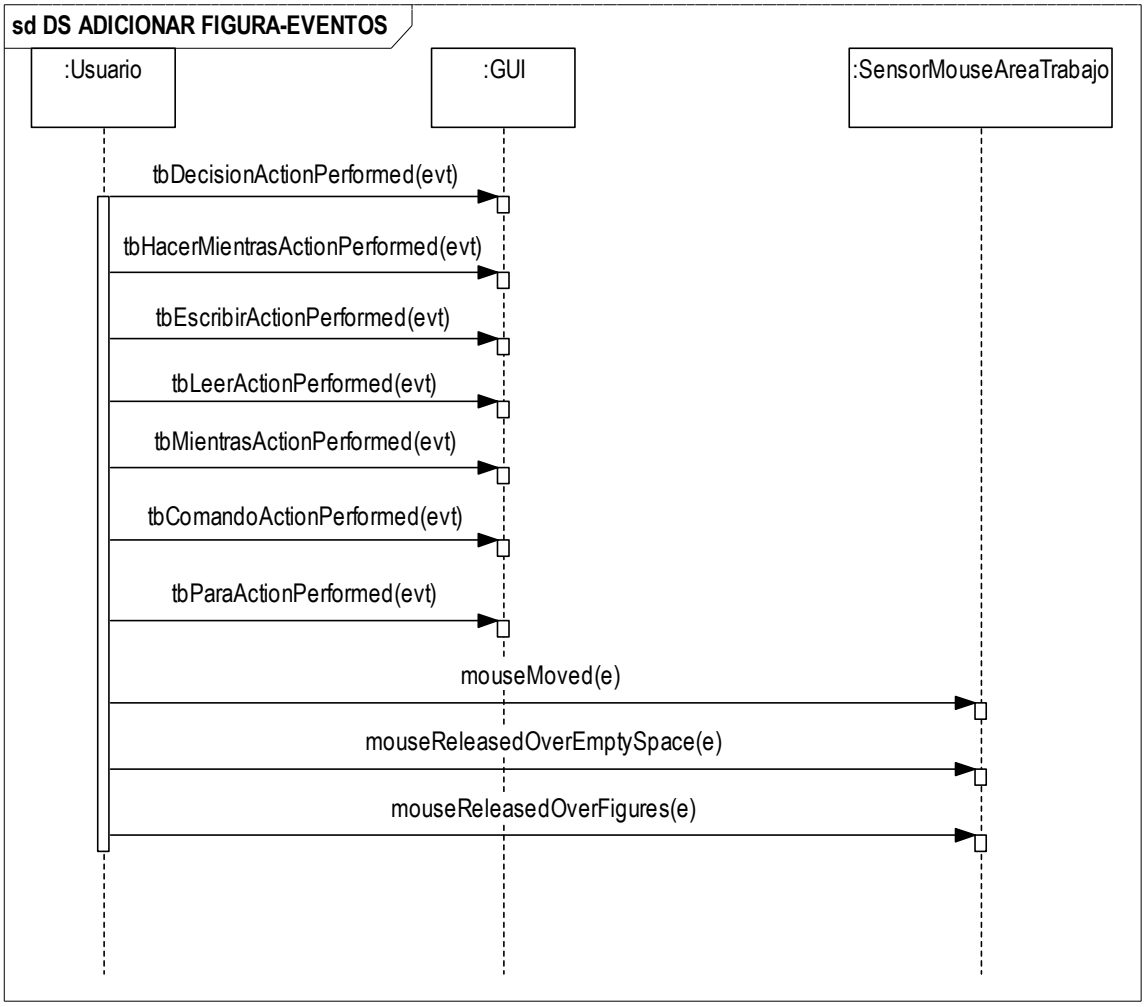

sd 2DS transformInFigure
areaTrabajo :AreaTrabajo
op :Operation
:Figura
:FigLeerDatos
alt Evalua la instancia de la variable op
[si op instanceof DecisionTrue]
id= getId()
expresion= getExpresion()
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(factorAlto,factorAncho)
alt evalua si no es la primera EstructuraN-S
[op.esHoja==false]
operaciones= getOperaciones()
alt evalua si existen mas elementos
[mq exista otro elemento]
fdv= transformInFigure(op,fdv)
boolean= agregarFigura(fdv)
fdv= transformInFigure(op,parent)
[si op instanceof LeerDatos]
id= getId()
expresion= getExpresion()
tipoVariable= getTipoVariable()
alt Evalua la variable localizedRead
[localizedRead==null]
setLocalizedReadString(defaultRead)
[localizedRead!=null]
setLocalizedReadString(localizedRead)
setParent(parent)
setFuente(fuente)
setBorderColor(borderColor)
setFactores(factorAlto,factorAncho)
setStroke(new BasicStroke(strokeWidth))
fid= transformInFigure(op,parent)

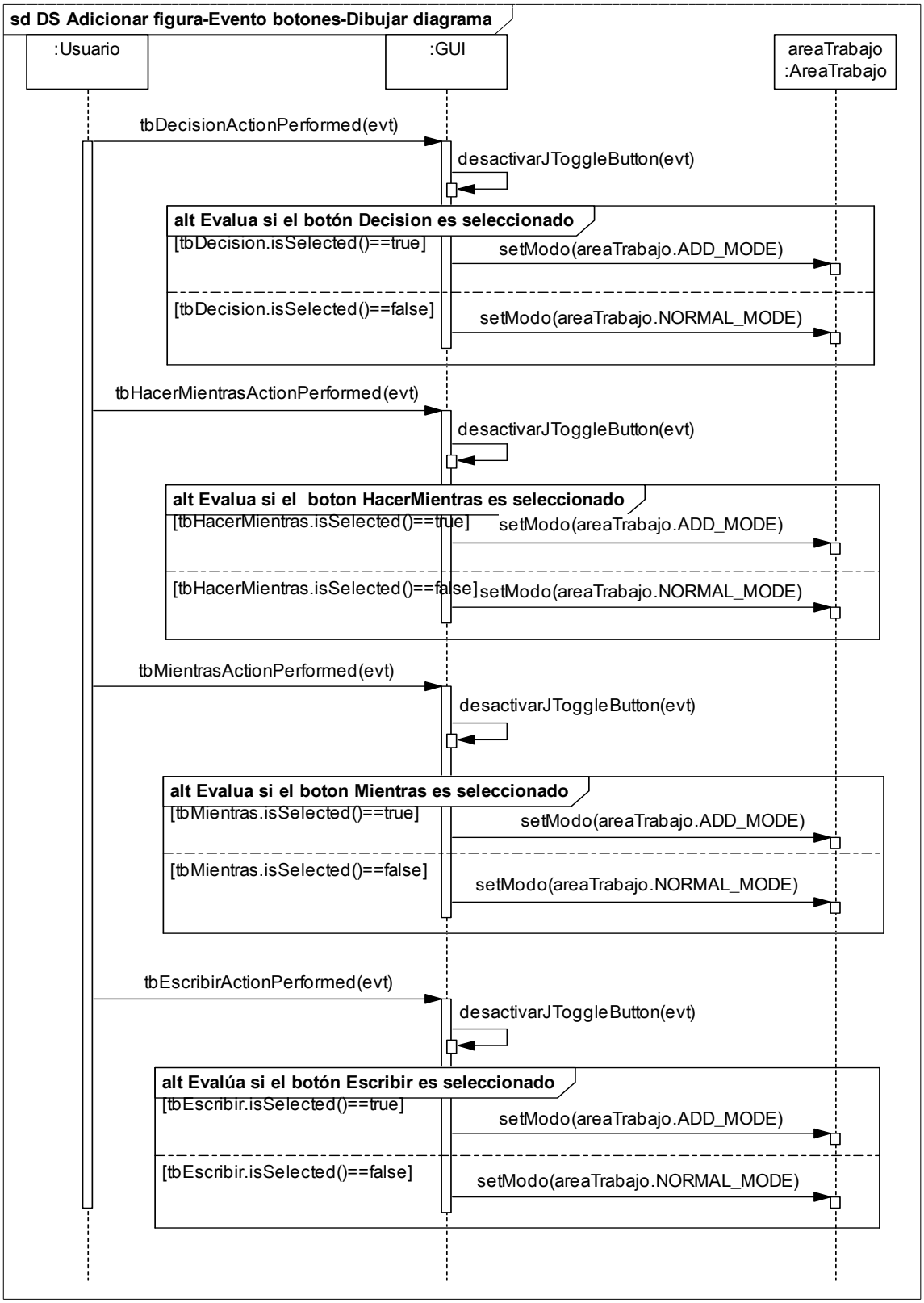

sd 3DS transformFigure
areaTrabajo :AreaTrabajo
op :Operation
temp :Figura
alt Evalua la instancia de la variable op
[si op instanceof WhileDo]
id= getId()
expresion= getExpresion()
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(factorAlto,factorAncho)
setStroke(new BasicStroke(strokeWidth))
alt evalua si es la primera EstructuraN-S
[op.esHoja==false]
operaciones= getOperaciones()
loop evalua si existen mas elementos en la lista operaciones
[mq exista otro elemento]
Figura= transformInFigure(op,fim)
boolean= agregarFigura(Figura)
fim= transformInFigure(op,parent)
[si op instanceof DoWhile]
id= getId()
expresion= getExpresion()
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(factorAlto,factorAncho)
setStroke(new BasicStroke(strokeWidth))
alt evalua si es la primera EstructuraN-S
[op.esHoja==false]
operaciones= getOperaciones()
loop Evalúa si existen mas elementos
[mientras que hallan mas elementos]
Figura= transformInFigure(op,fia)
boolean= agregarFigura(Figura)
fia= transformInFigure(op,parent)

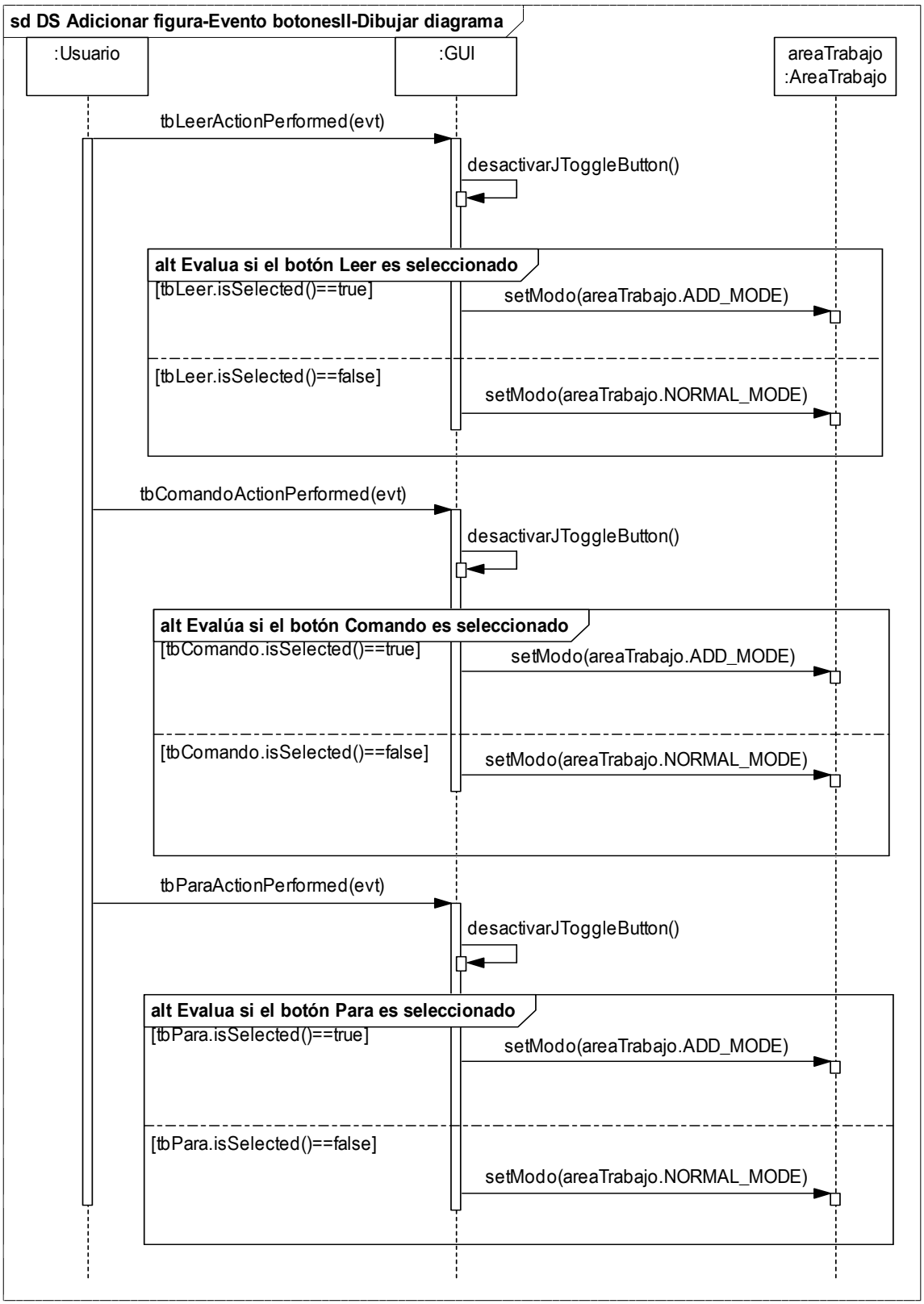

sd 4DS transformInFigure
areaTrabajo :AreaTrabajo
op :Operation
temp :Figura
:FigEscribirDatos
alt Evalua la instancia de la variable op
[si op instanceof EscribirDatos]
id= getId()
expresion= getExpresion()
alt Evalua la variable localizedRead
[localizedRead==null]
setLocalizedWriteString(defaultWrite)
[localizedRead!=null]
setLocalizedWriteString(localizedWrite)
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(factorAlto,factorAncho)
setStroke(new BasicStroke(strokeWidth))
fsd= transformInFigure(op,parent)


sd 4-1DS transformInFigure
areaTrabajo :AreaTrabajo
op :Operation
temp :Figura
fim : FigCicloPara
alt Evalua la instancia de la variable op
[si op instanceof CicloPara]
id= getId()
expresion= getExpresion()
setParent(parent)
setFuente(fuente)
setFontColor(fontColor)
setBorderColor(borderColor)
setFactores(factorAlto,factorAncho)
setStroke(new BasicStroke(strokeWidth))
boolean= esHoja()
alt Evalua que no sea el primer elemento
[esHoja!=true]
operaciones= getOperaciones()
loop mientras que existan mas elementos
[iter.hasNext()==true]
Figura= transformInFigure(op,fim)
agregarFigura(indice,fim)
fim= transformInFigure(id,operaciones)

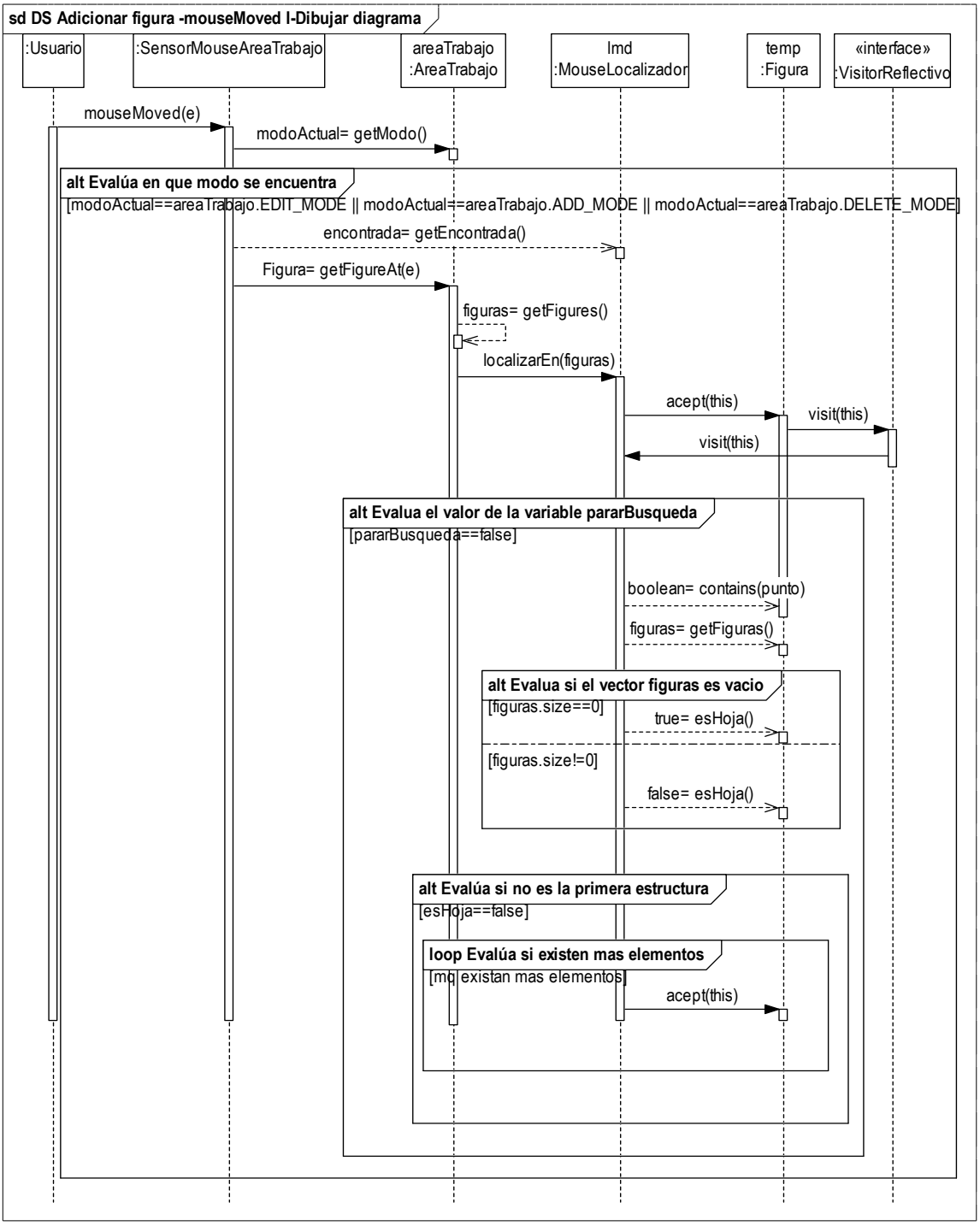
10.5 DIAGRAMAS DE SECUENCIA DE ADICIONAR FIGURA (DIBUJAR DIAGRAMA)

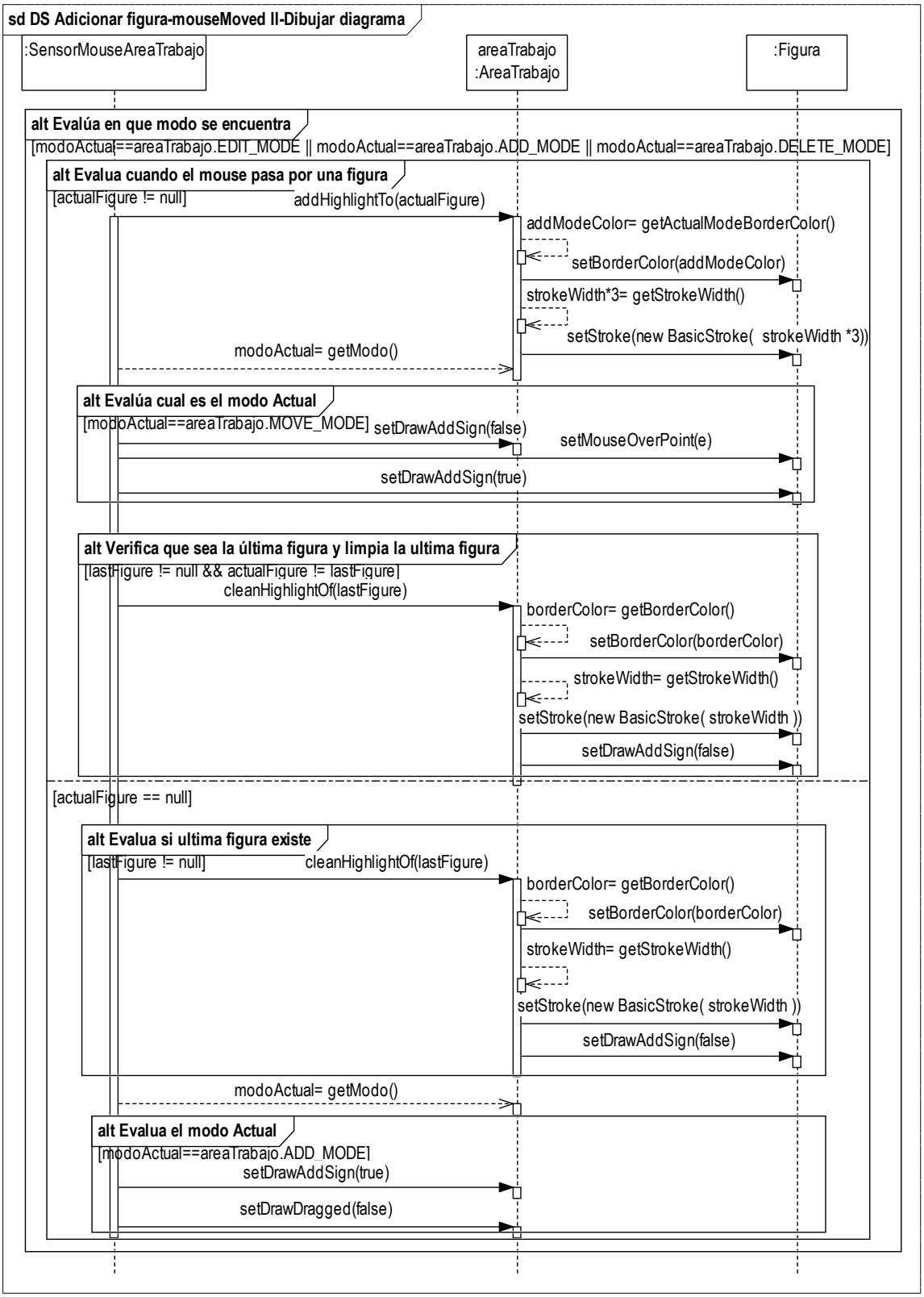


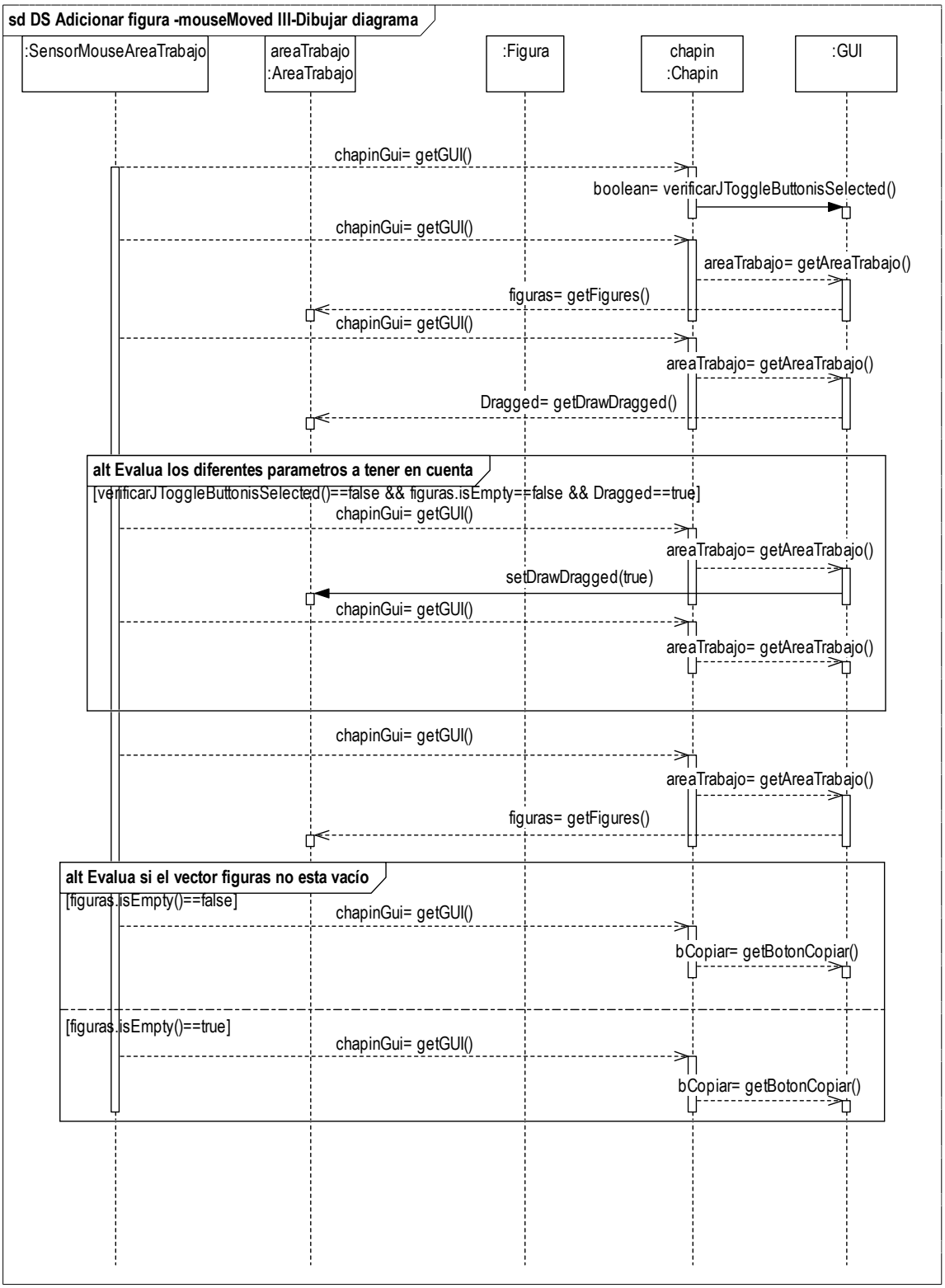

sd DS Adicionar figura-Evento botones-Dibujar diagrama
:Usuario
:GUI
areaTrabajo
:AreaTrabajo
tbDecisionActionPerformed(evt)
desactivarJToggleButton(evt)
alt Evalua si el botón Decision es seleccionado
[tbDecision.isSelected()==true]
setModo(areaTrabajo.ADD_MODE)
[tbDecision.isSelected()==false]
setModo(areaTrabajo.NORMAL_MODE)
tbHacerMientrasActionPerformed(evt)
desactivarJToggleButton(evt)
alt Evalua si el boton HacerMientras es seleccionado
[tbHacerMientras.isSelected()==true]
setModo(areaTrabajo.ADD_MODE)
[tbHacerMientras.isSelected()==false]
setModo(areaTrabajo.NORMAL_MODE)
tbMientrasActionPerformed(evt)
desactivarJToggleButton(evt)
alt Evalua si el boton Mientras es seleccionado
[tbMientras.isSelected()==true]
setModo(areaTrabajo.ADD_MODE)
[tbMientras.isSelected()==false]
setModo(areaTrabajo.NORMAL_MODE)
tbEscribirActionPerformed(evt)
desactivarJToggleButton(evt)
alt Evalúa si el botón Escribir es seleccionado
[tbEscribir.isSelected()==true]
setModo(areaTrabajo.ADD_MODE)
[tbEscribir.isSelected()==false]
setModo(areaTrabajo.NORMAL_MODE)


sd DS Adicionar figura-Evento botonesII-Dibujar diagrama
:Usuario
:GUI
areaTrabajo
:AreaTrabajo
tbLeerActionPerformed(evt)
desactivarJToggleButton()
alt Evalua si el botón Leer es seleccionado
[tbLeer.isSelected()==true]
setModo(areaTrabajo.ADD_MODE)
[tbLeer.isSelected()==false]
setModo(areaTrabajo.NORMAL_MODE)
tbComandoActionPerformed(evt)
desactivarJToggleButton()
alt Evalúa si el botón Comando es seleccionado
[tbComando.isSelected()==true]
setModo(areaTrabajo.ADD_MODE)
[tbComando.isSelected()==false]
setModo(areaTrabajo.NORMAL_MODE)
tbParaActionPerformed(evt)
desactivarJToggleButton()
alt Evalua si el botón Para es seleccionado
[tbPara.isSelected()==true]
setModo(areaTrabajo.ADD_MODE)
[tbPara.isSelected()==false]
setModo(areaTrabajo.NORMAL_MODE)

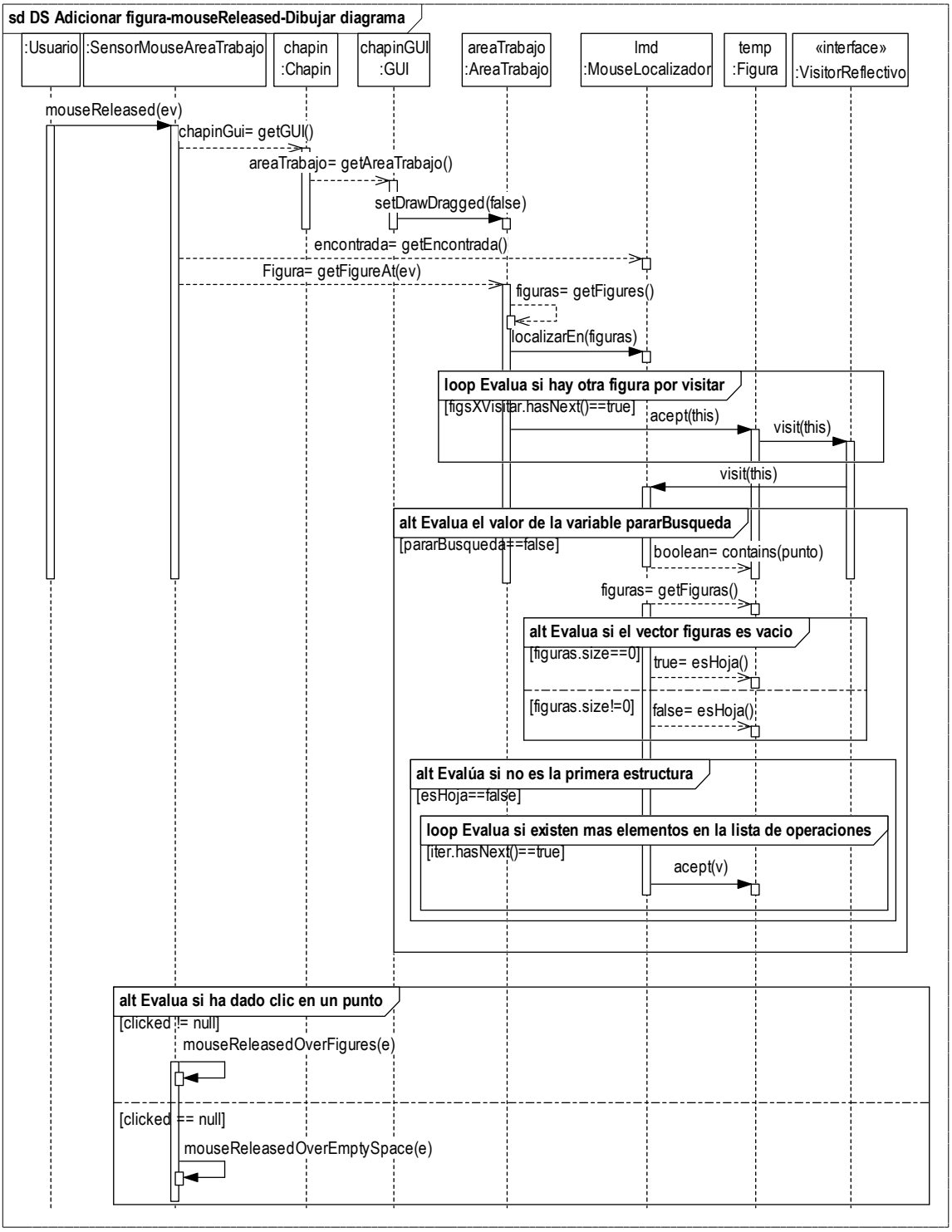
10.5.1 Operación mouseMoved ()



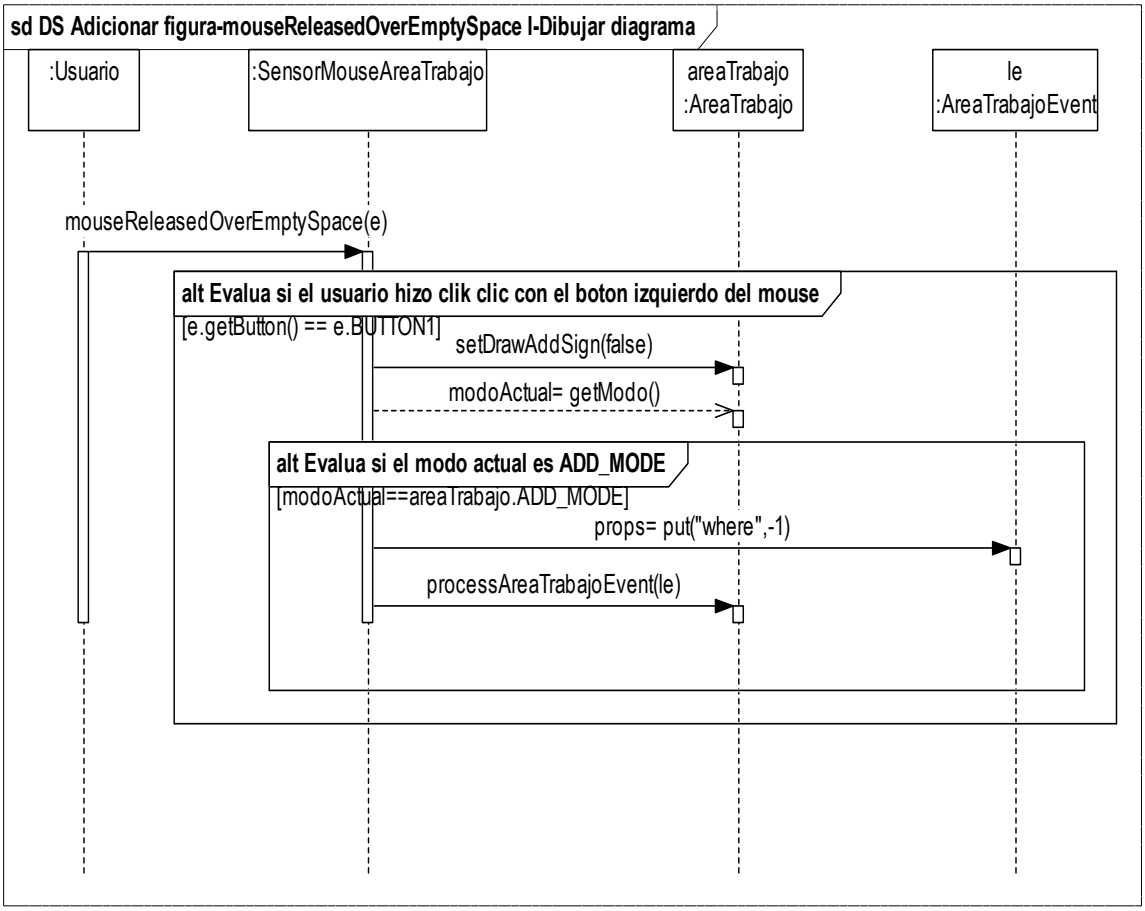

sd DS Adicionar figura-mouseMoved II-Dibujar diagrama
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
:Figura
alt Evalúa en que modo se encuentra
[modoActual==areaTrabajo.EDIT_MODE || modoActual==areaTrabajo.ADD_MODE || modoActual==areaTrabajo.DELETE_MODE]
alt Evalua cuando el mouse pasa por una figura
[actualFigure != null]
addHighlightTo(actualFigure)
addModeColor= getActualModeBorderColor()
setBorderColor(addModeColor)
strokeWidth*3= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth *3))
modoActual= getModo()
alt Evalúa cual es el modo Actual
[modoActual==areaTrabajo.MOVE_MODE]
setDrawAddSign(false)
setMouseOverPoint(e)
setDrawAddSign(true)
alt Verifica que sea la última figura y limpia la ultima figura
[lastFigure != null && actualFigure != lastFigure]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
[actualFigure == null]
alt Evalua si ultima figura existe
[lastFigure != null]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
modoActual= getModo()
alt Evalua el modo Actual
[modoActual==areaTrabajo.ADD_MODE]
setDrawAddSign(true)
setDrawDragged(false)

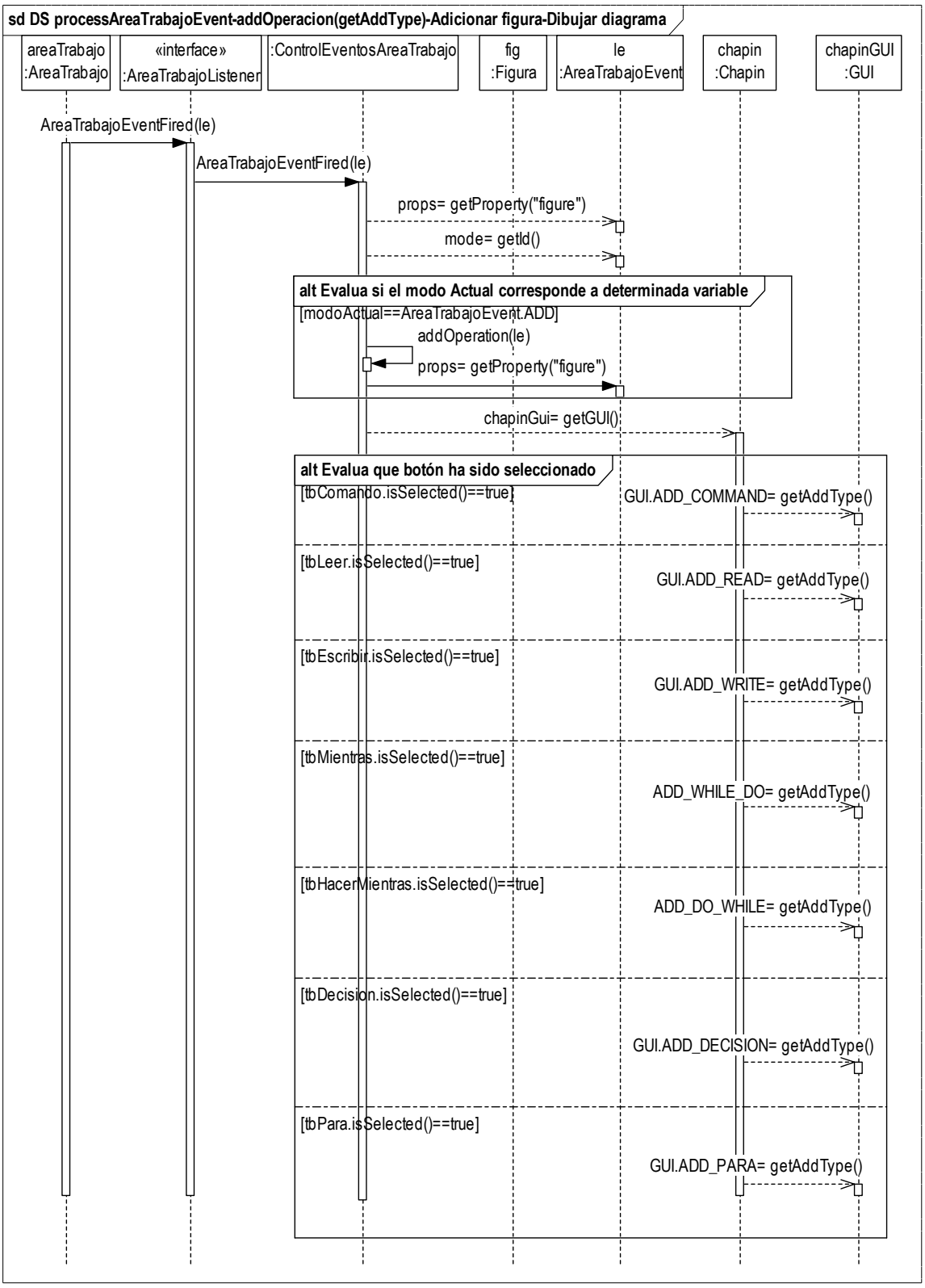

sd DS Adicionar figura -mouseMoved III-Dibujar diagrama
:SensorMouseAreaTrabajo
areaTrabajo :AreaTrabajo
:Figura
chapin :Chapin
:GUI
chapinGui= getGUI()
boolean= verificarJToggleButtonisSelected()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
Dragged= getDrawDragged()
alt Evalua los diferentes parametros a tener en cuenta
[verificarJToggleButtonisSelected()==false && figuras.isEmpty==false && Dragged==true]
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setDrawDragged(true)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
alt Evalua si el vector figuras no esta vacío
[figuras.isEmpty()==false]
chapinGui= getGUI()
bCopiar= getBotonCopiar()
[figuras.isEmpty()==true]
chapinGui= getGUI()
bCopiar= getBotonCopiar()

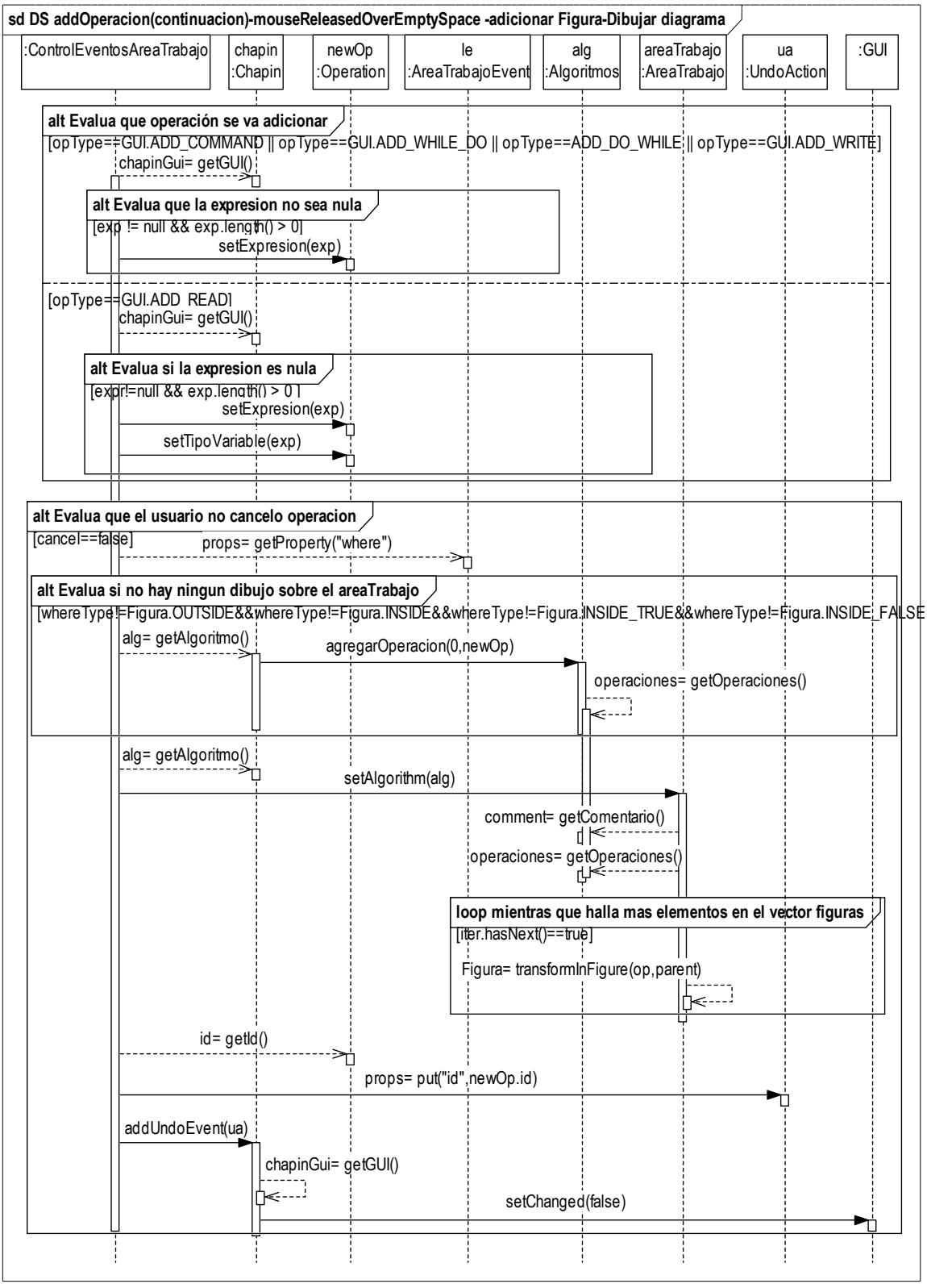
10.5.2 Operación mouseReleased ()



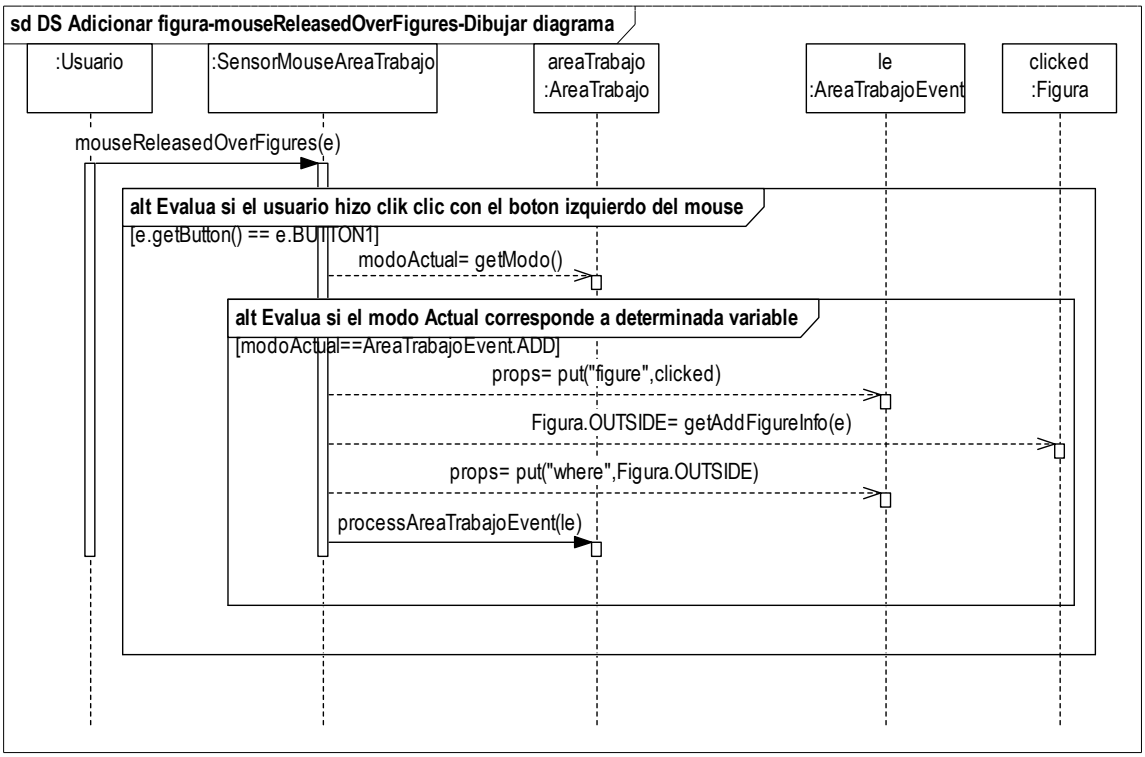
10.5.3 Operación mouseReleasedOverEmptySpace ()

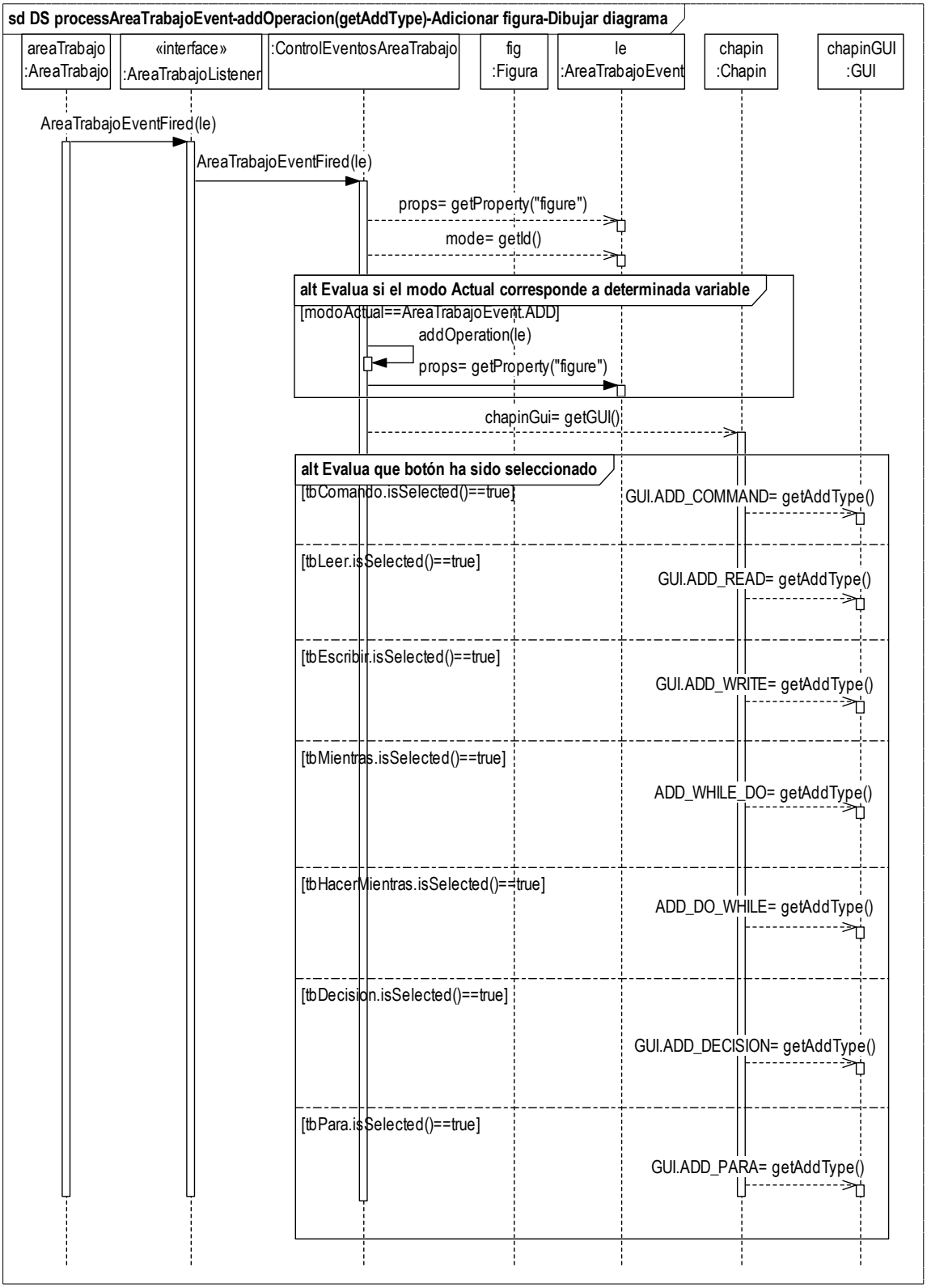


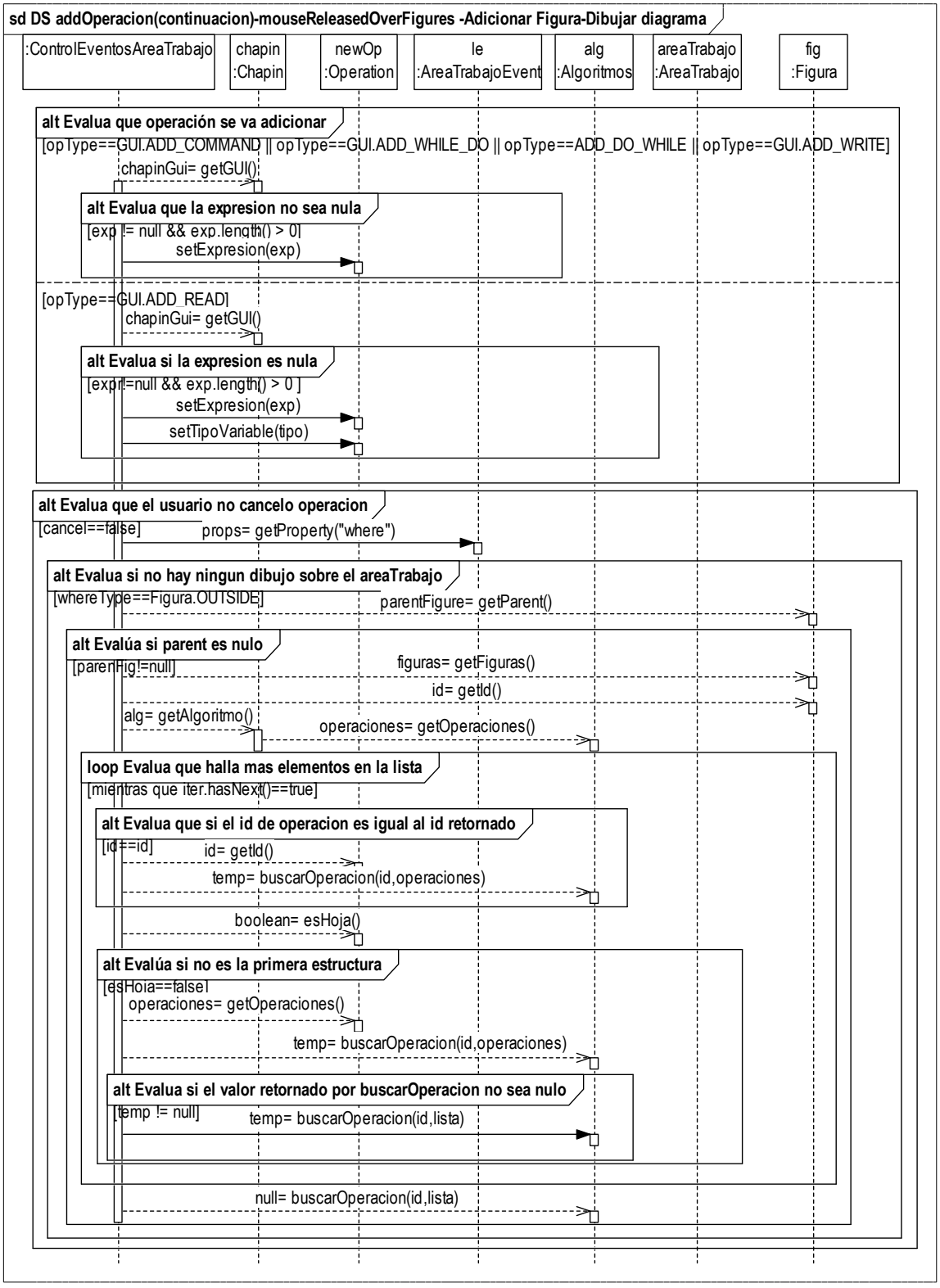

sd DS processAreaTrabajoEvent-addOperacion(getAddType)-Adicionar figura-Dibujar diagrama
areaTrabajo :AreaTrabajo
«interface» :AreaTrabajoListener
:ControlEventosAreaTrabajo
fig :Figura
le :AreaTrabajoEvent
chapin :Chapin
chapinGUI :GUI
AreaTrabajoEventFired(le)
AreaTrabajoEventFired(le)
props= getProperty("figure")
mode= getId()
alt Evalua si el modo Actual corresponde a determinada variable
[modoActual==AreaTrabajoEvent.ADD]
addOperation(le)
props= getProperty("figure")
chapinGui= getGUI()
alt Evalua que botón ha sido seleccionado
[tbComando.isSelected()==true]
GUI.ADD_COMMAND= getAddType()
[tbLeer.isSelected()==true]
GUI.ADD_READ= getAddType()
[tbEscribir.isSelected()==true]
GUI.ADD_WRITE= getAddType()
[tbMientras.isSelected()==true]
ADD_WHILE_DO= getAddType()
[tbHacerMientras.isSelected()==true]
ADD_DO_WHILE= getAddType()
[tbDecision.isSelected()==true]
GUI.ADD_DECISION= getAddType()
[tbPara.isSelected()==true]
GUI.ADD_PARA= getAddType()


sd DS addOperacion(continuacion)-mouseReleasedOverEmptySpace -adicionar Figura-Dibujar diagrama
:ControlEventosAreaTrabajo
chapin :Chapin
newOp :Operation
le :AreaTrabajoEvent
alg :Algoritmos
areaTrabajo :AreaTrabajo
ua :UndoAction
:GUI
alt Evalua que operación se va adicionar
[opType==GUI.ADD_COMMAND || opType==GUI.ADD_WHILE_DO || opType==ADD_DO_WHILE || opType==GUI.ADD_WRITE]
chapinGui= getGUI()
alt Evalua que la expresion no sea nula
[exp != null && exp.length() > 0]
setExpresion(exp)
[opType==GUI.ADD_READ]
chapinGui= getGUI()
alt Evalua si la expresion es nula
[expr!=null && exp.length() > 0]
setExpresion(exp)
setTipoVariable(exp)
alt Evalua que el usuario no cancelo operacion
[cancel==false]
props= getProperty("where")
alt Evalua si no hay ningun dibujo sobre el areaTrabajo
[whereType!=Figura.OUTSIDE&&whereType!=Figura.INSIDE&&whereType!=Figura.INSIDE_TRUE&&whereType!=Figura.INSIDE_FALSE
alg= getAlgoritmo()
agregarOperacion(0,newOp)
operaciones= getOperaciones()
alg= getAlgoritmo()
setAlgorithm(alg)
comment= getComentario()
operaciones= getOperaciones()
loop mientras que halla mas elementos en el vector figuras
[iter.hasNext()==true]
Figura= transformInFigure(op,parent)
id= getId()
props= put("id",newOp.id)
addUndoEvent(ua)
chapinGui= getGUI()
setChanged(false)

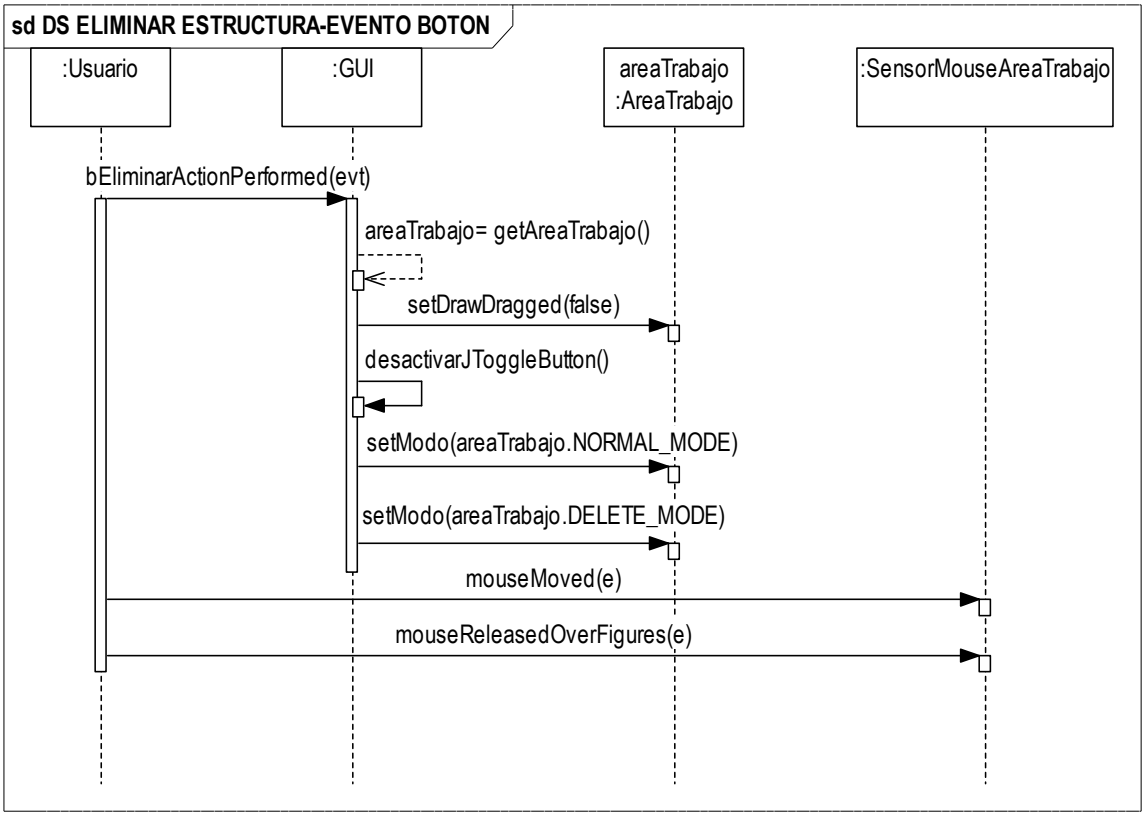
10.5.4 Operación mouseReleasedOverFigures ()



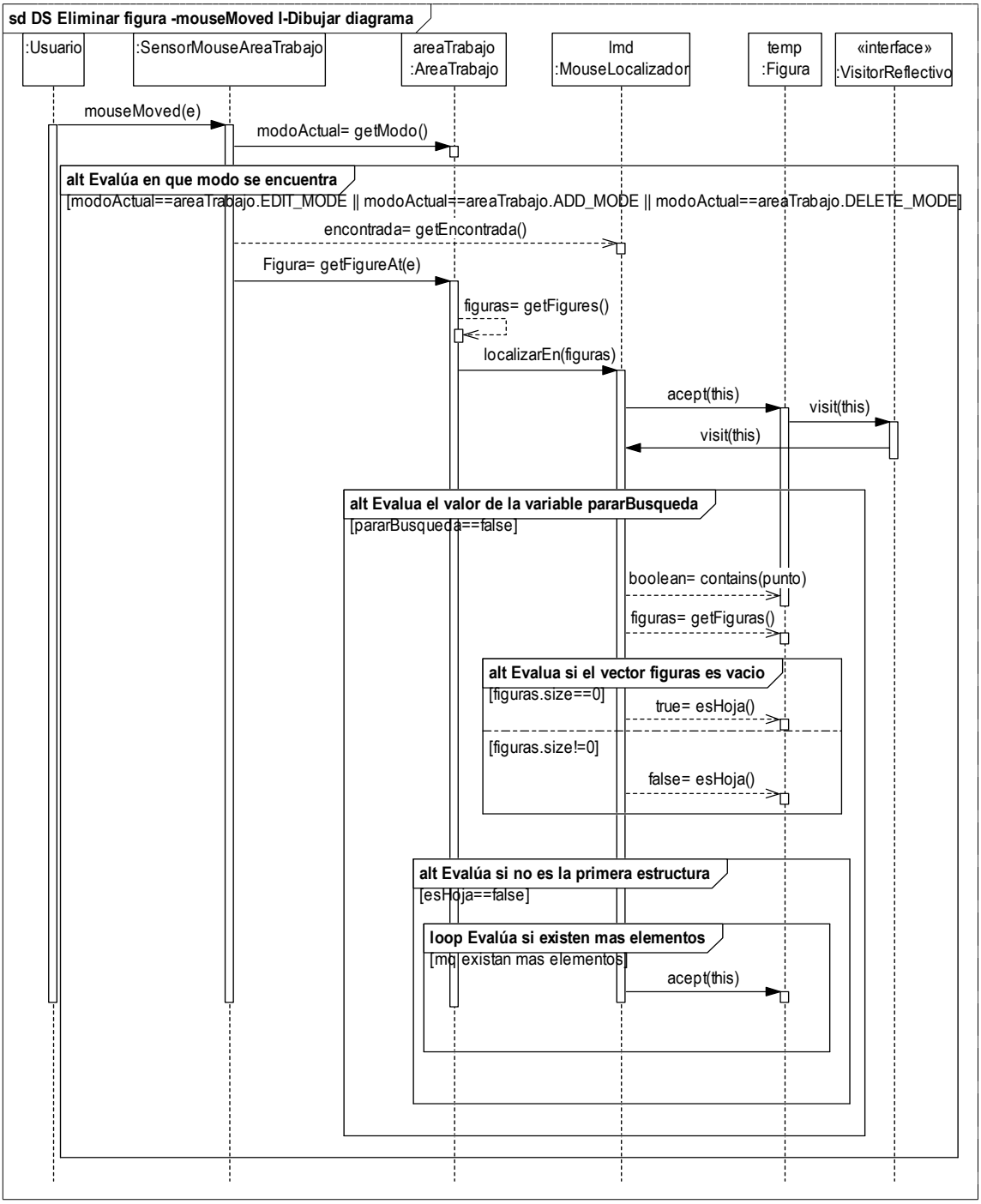

sd DS processAreaTrabajoEvent-addOperacion(getAddType)-Adicionar figura-Dibujar diagrama
areaTrabajo :AreaTrabajo
«interface» :AreaTrabajoListener
:ControlEventosAreaTrabajo
fig :Figura
le :AreaTrabajoEvent
chapin :Chapin
chapinGUI :GUI
AreaTrabajoEventFired(le)
AreaTrabajoEventFired(le)
props= getProperty("figure")
mode= getId()
alt Evalua si el modo Actual corresponde a determinada variable
[modoActual==AreaTrabajoEvent.ADD]
addOperation(le)
props= getProperty("figure")
chapinGui= getGUI()
alt Evalua que botón ha sido seleccionado
[tbComando.isSelected()==true]
GUI.ADD_COMMAND= getAddType()
[tbLeer.isSelected()==true]
GUI.ADD_READ= getAddType()
[tbEscribir.isSelected()==true]
GUI.ADD_WRITE= getAddType()
[tbMientras.isSelected()==true]
ADD_WHILE_DO= getAddType()
[tbHacerMientras.isSelected()==true]
ADD_DO_WHILE= getAddType()
[tbDecision.isSelected()==true]
GUI.ADD_DECISION= getAddType()
[tbPara.isSelected()==true]
GUI.ADD_PARA= getAddType()

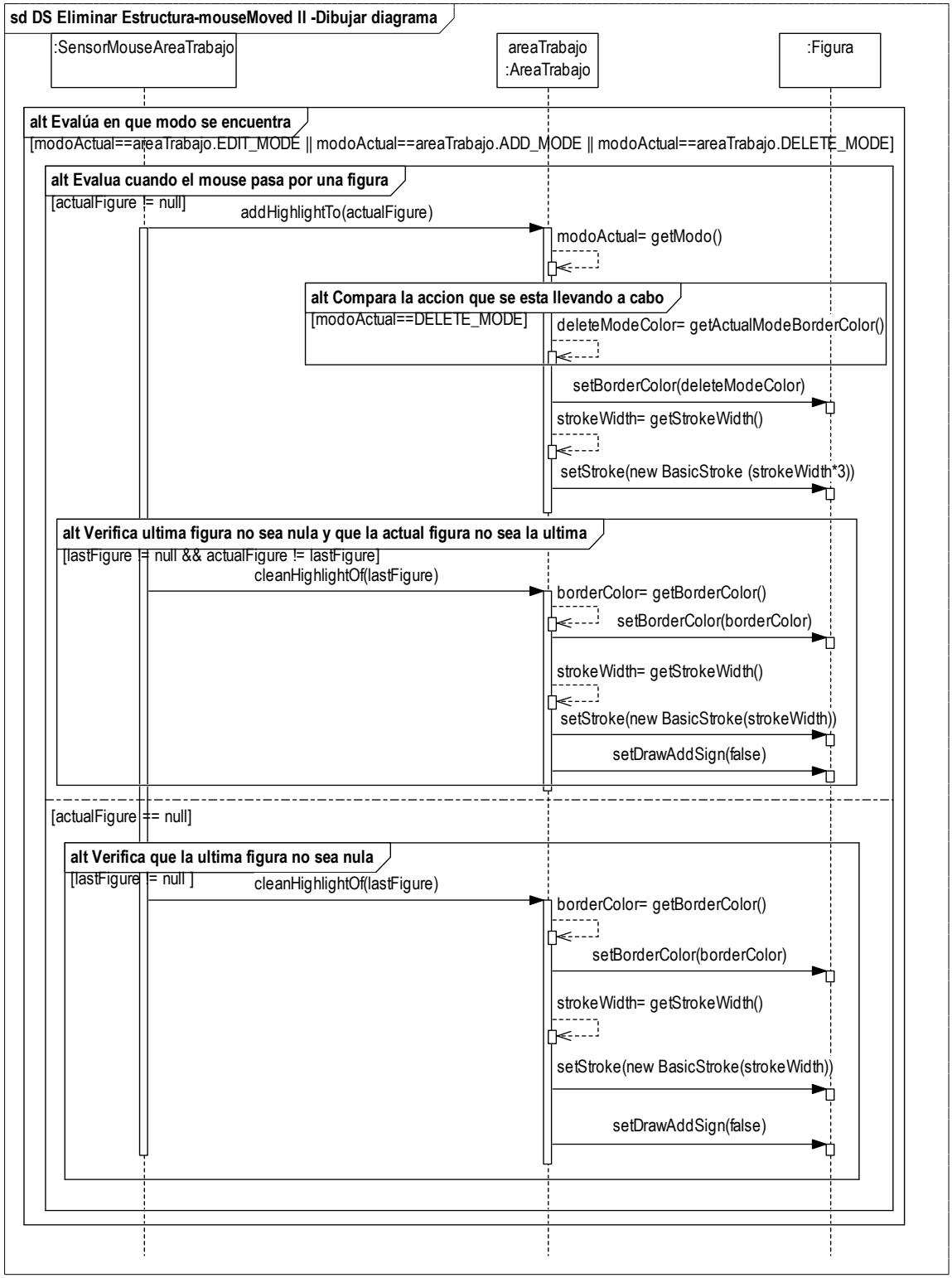

sd DS addOperacion(continuacion)-mouseReleasedOverFigures -Adicionar Figura-Dibujar diagrama
:ControlEventosAreaTrabajo
chapin :Chapin
newOp :Operation
le :AreaTrabajoEvent
alg :Algoritmos
areaTrabajo :AreaTrabajo
fig :Figura
alt Evalua que operación se va adicionar
[opType==GUI.ADD_COMMAND || opType==GUI.ADD_WHILE_DO || opType==ADD_DO_WHILE || opType==GUI.ADD_WRITE]
chapinGui= getGUI()
alt Evalua que la expresion no sea nula
[exp != null && exp.length() > 0]
setExpresion(exp)
[opType==GUI.ADD_READ]
chapinGui= getGUI()
alt Evalua si la expresion es nula
[expr!=null && exp.length() > 0]
setExpresion(exp)
setTipoVariable(tipo)
alt Evalua que el usuario no cancelo operacion
[cancel==false]
props= getProperty("where")
alt Evalua si no hay ningun dibujo sobre el areaTrabajo
[whereType==Figura.OUTSIDE]
parentFigure= getParent()
alt Evalúa si parent es nulo
[parentFig!=null]
figuras= getFiguras()
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop Evalua que halla mas elementos en la lista
[mientras que iter.hasNext()==true]
alt Evalua que si el id de operacion es igual al id retornado
[id==id]
id= getId()
temp= buscarOperacion(id,operaciones)
boolean= esHoja()
alt Evalúa si no es la primera estructura
[esHoja==false]
operaciones= getOperaciones()
temp= buscarOperacion(id,operaciones)
alt Evalua si el valor retornado por buscarOperacion no sea nulo
[temp != null]
temp= buscarOperacion(id,lista)
null= buscarOperacion(id,lista)

10.6 DIAGRAMA DE SECUENCIA DE ELIMINAR FIGURA

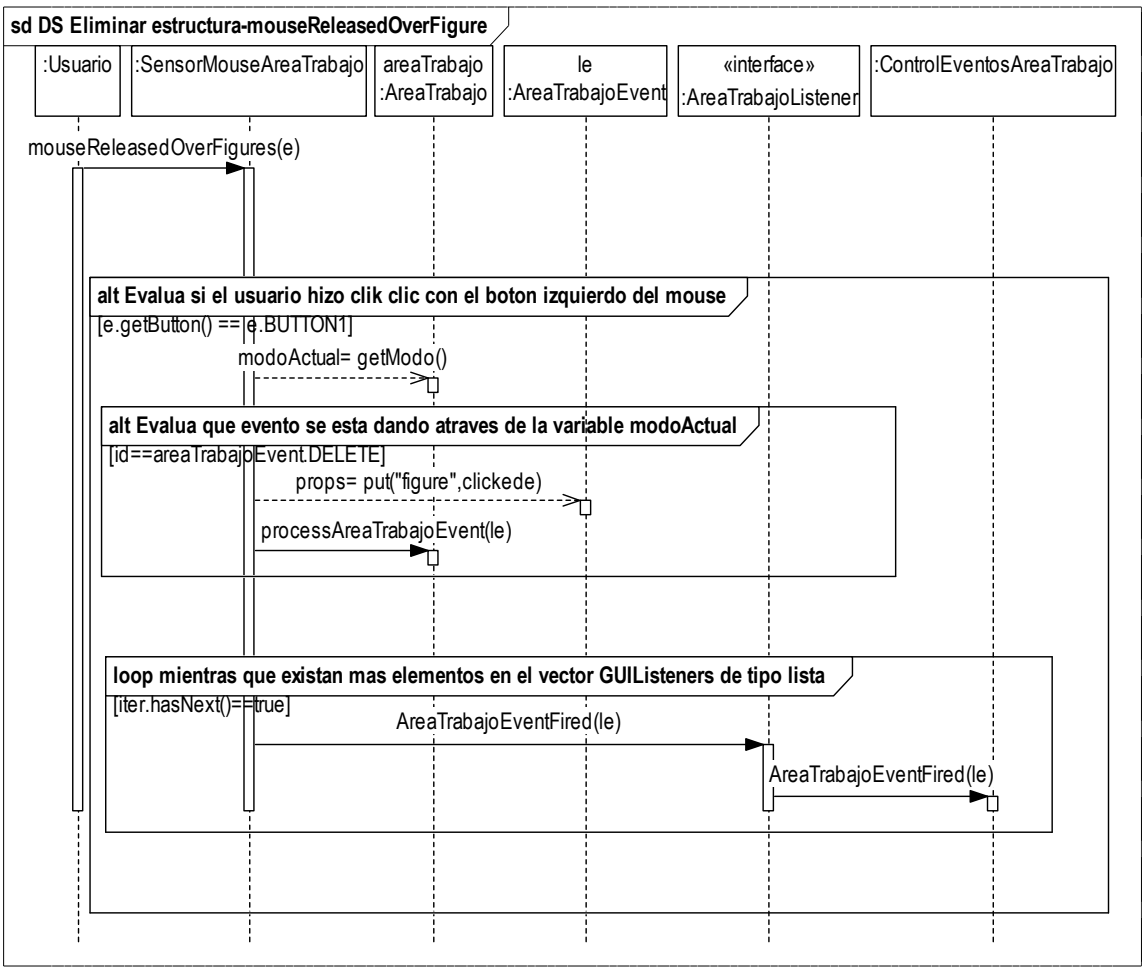


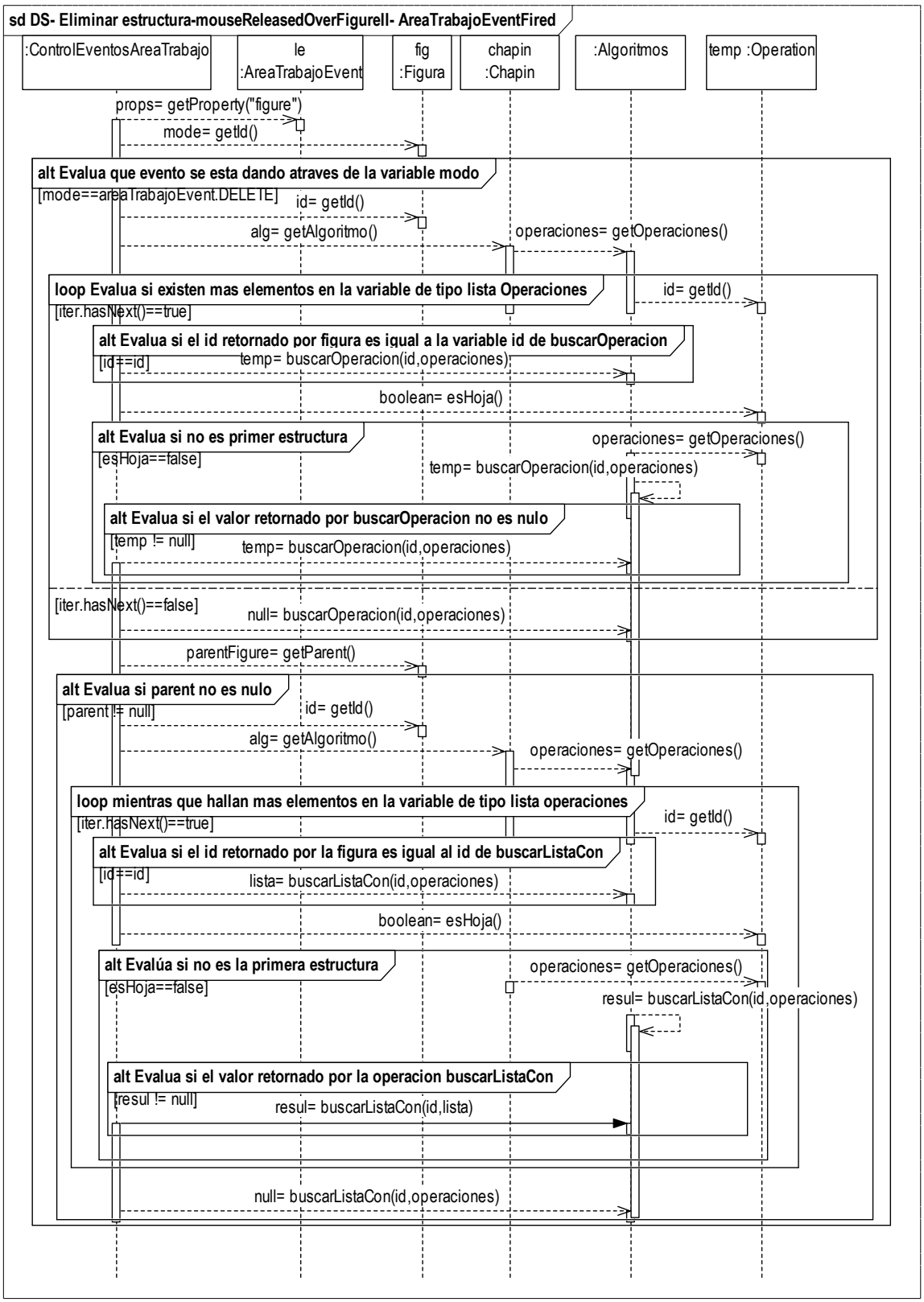
10.6.1 Operación mouseMoved ()

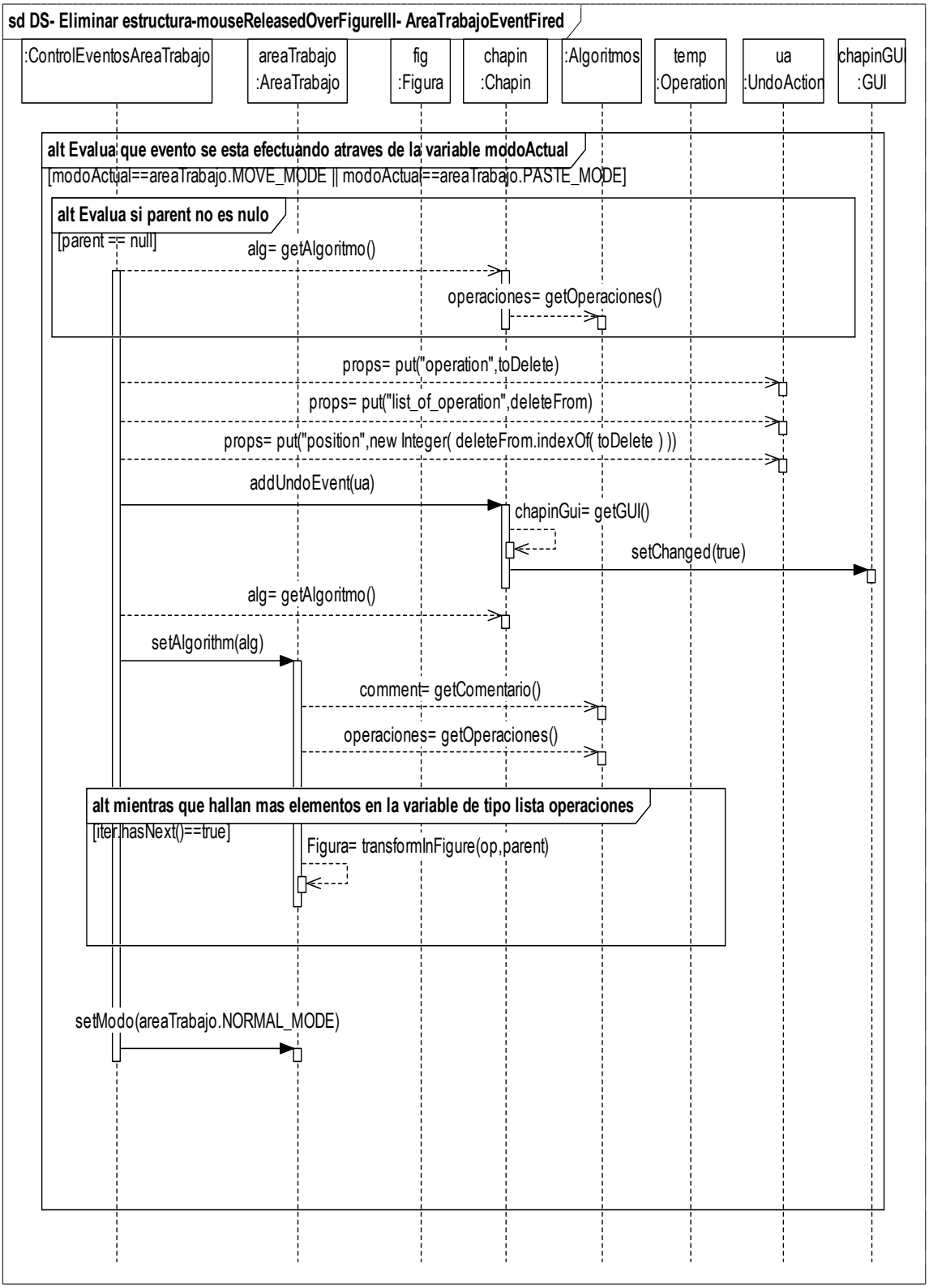



sd DS Eliminar Estructura-mouseMoved II -Dibujar diagrama
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
:Figura
alt Evalúa en que modo se encuentra
[modoActual==areaTrabajo.EDIT_MODE || modoActual==areaTrabajo.ADD_MODE || modoActual==areaTrabajo.DELETE_MODE]
alt Evalua cuando el mouse pasa por una figura
[actualFigure != null]
addHighlightTo(actualFigure)
modoActual= getModo()
alt Compara la accion que se esta llevando a cabo
[modoActual==DELETE_MODE]
deleteModeColor= getActualModeBorderColor()
setBorderColor(deleteModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke (strokeWidth*3))
alt Verifica ultima figura no sea nula y que la actual figura no sea la ultima
[lastFigure != null && actualFigure != lastFigure]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
[actualFigure == null]
alt Verifica que la ultima figura no sea nula
[lastFigure != null]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)

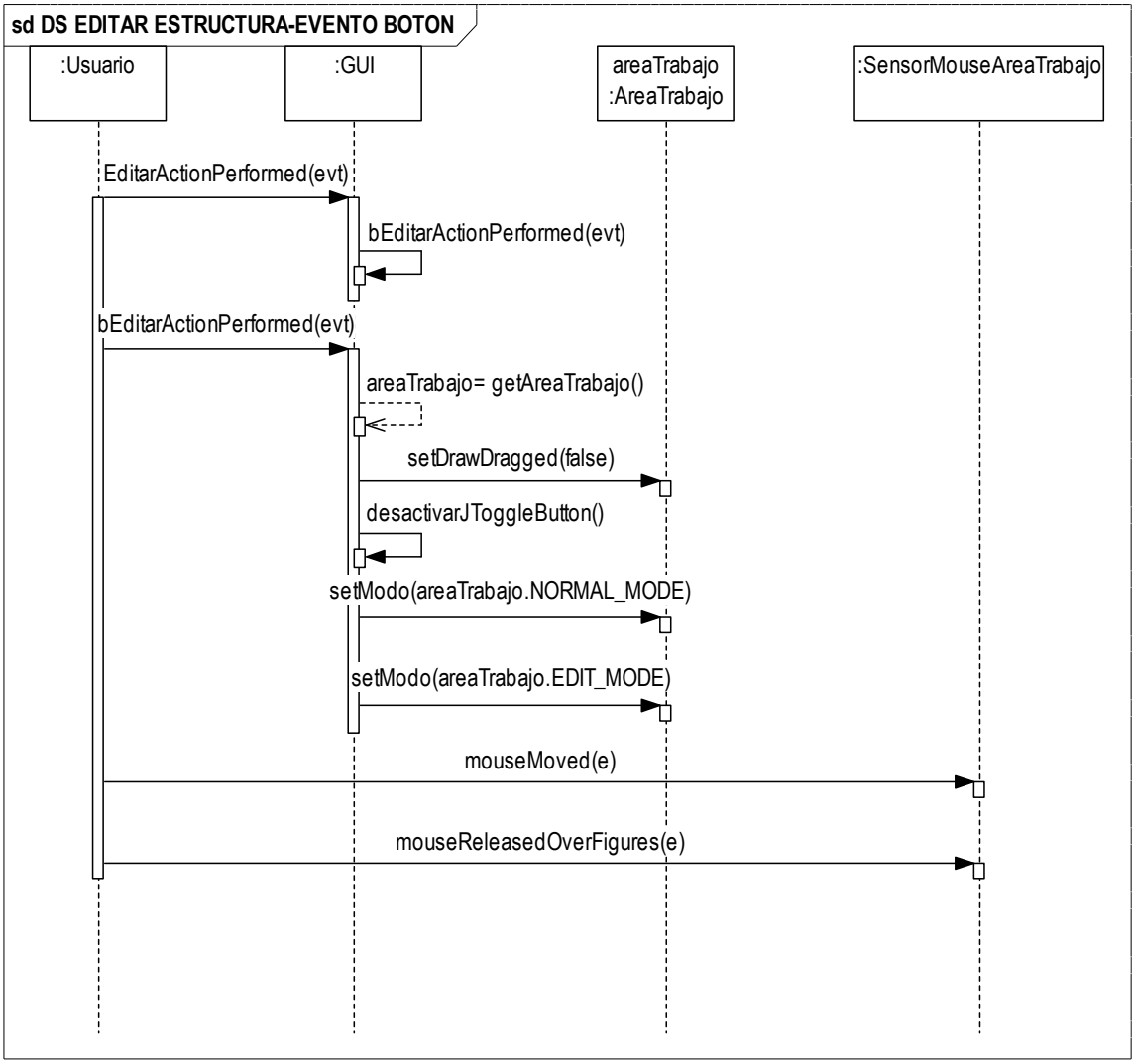
10.6.2 Operación mouseReleasedOverFigures ()

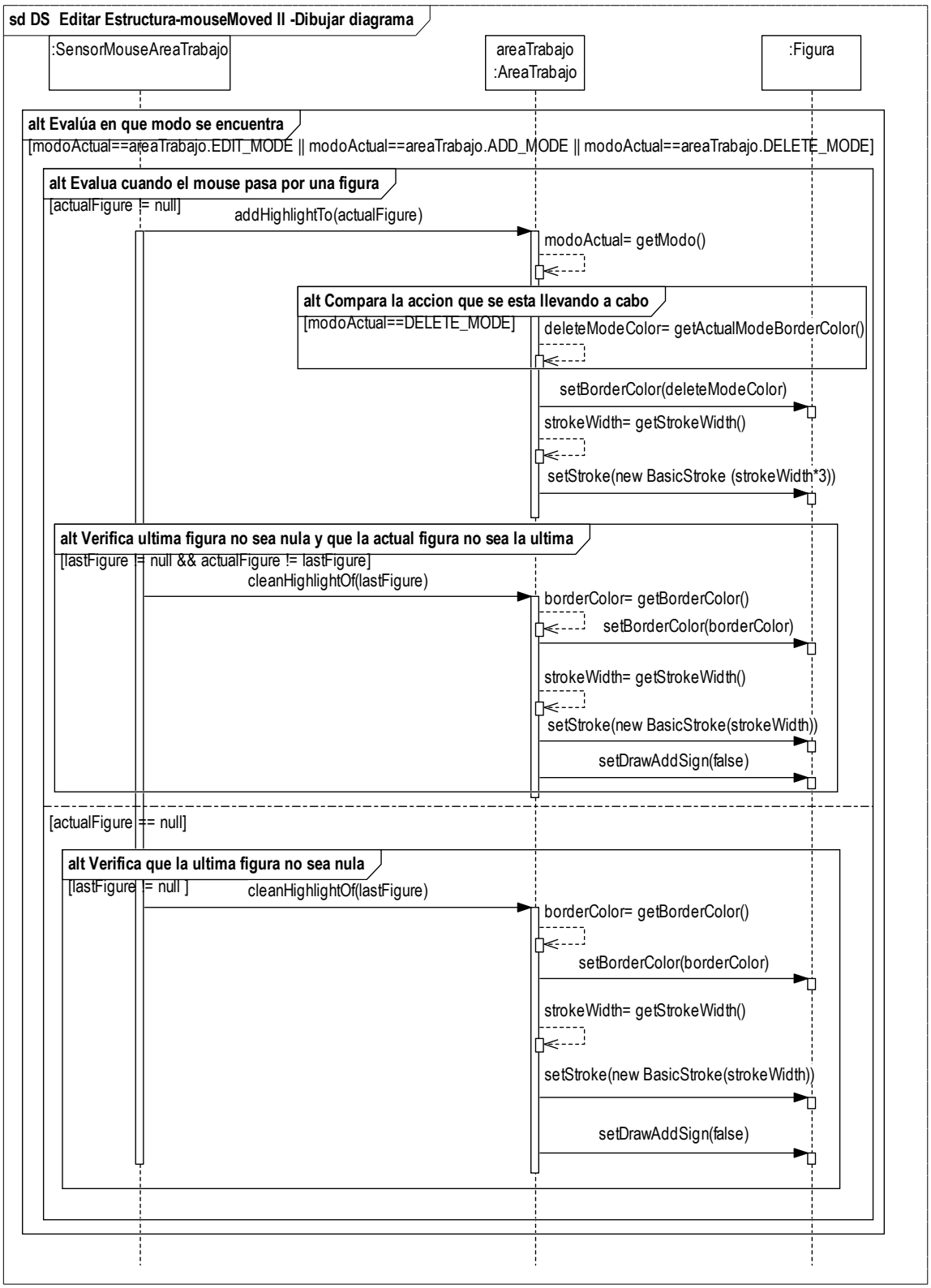


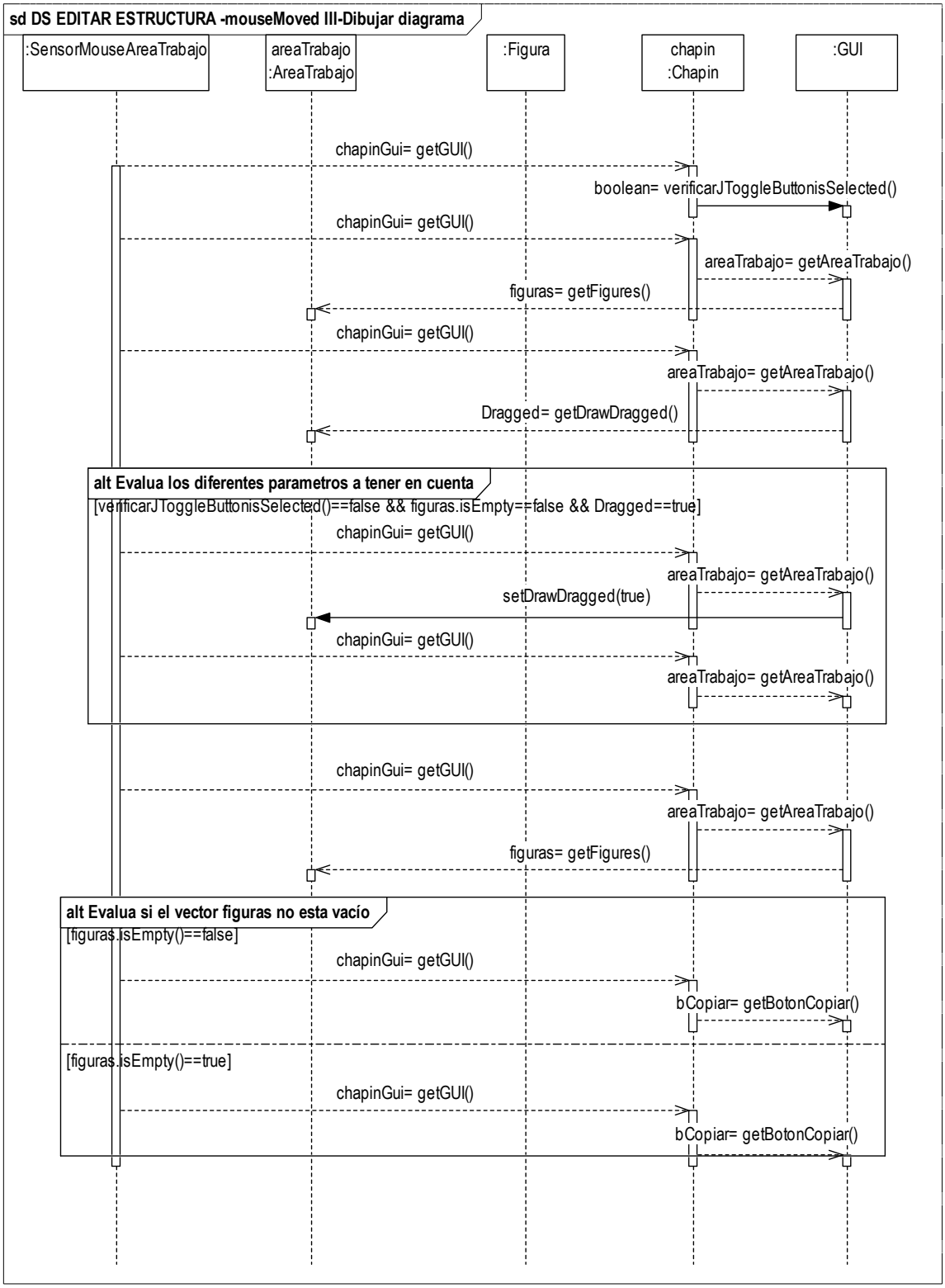

sd DS- Eliminar estructura-mouseReleasedOverFigureII- AreaTrabajoEventFired
:ControlEventosAreaTrabajo
le
:AreaTrabajoEvent
fig
:Figura
chapin
:Chapin
:Algoritmos
temp :Operation
props= getProperty("figure")
mode= getId()
alt Evalua que evento se esta dando atraves de la variable modo
[mode==areaTrabajoEvent.DELETE]
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop Evalua si existen mas elementos en la variable de tipo lista Operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua si el id retornado por figura es igual a la variable id de buscarOperacion
[id==id]
temp= buscarOperacion(id,operaciones)
boolean= esHoja()
alt Evalua si no es primer estructura
[esHoja==false]
operaciones= getOperaciones()
temp= buscarOperacion(id,operaciones)
alt Evalua si el valor retornado por buscarOperacion no es nulo
[temp != null]
temp= buscarOperacion(id,operaciones)
[iter.hasNext()==false]
null= buscarOperacion(id,operaciones)
parentFigure= getParent()
alt Evalua si parent no es nulo
[parent != null]
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua si el id retornado por la figura es igual al id de buscarListaCon
[id==id]
lista= buscarListaCon(id,operaciones)
boolean= esHoja()
alt Evalúa si no es la primera estructura
[esHoja==false]
operaciones= getOperaciones()
resul= buscarListaCon(id,operaciones)
alt Evalua si el valor retornado por la operacion buscarListaCon
[resul != null]
resul= buscarListaCon(id,lista)
null= buscarListaCon(id,operaciones)


sd DS- Eliminar estructura-mouseReleasedOverFigureIII- AreaTrabajoEventFired
:ControlEventosAreaTrabajo
areaTrabajo :AreaTrabajo
fig :Figura
chapin :Chapin
:Algoritmos
temp :Operation
ua :UndoAction
chapinGUI :GUI
alt Evalua que evento se esta efectuando atraves de la variable modoActual
[modoActual==areaTrabajo.MOVE_MODE || modoActual==areaTrabajo.PASTE_MODE]
alt Evalua si parent no es nulo
[parent == null]
alg= getAlgoritmo()
operaciones= getOperaciones()
props= put("operation",toDelete)
props= put("list_of_operation",deleteFrom)
props= put("position",new Integer(deleteFrom.indexOf(toDelete)))
addUndoEvent(ua)
chapinGui= getGUI()
setChanged(true)
alg= getAlgoritmo()
setAlgorithm(alg)
comment= getComentario()
operaciones= getOperaciones()
alt mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
Figura= transformInFigure(op,parent)
setModo(areaTrabajo.NORMAL_MODE)

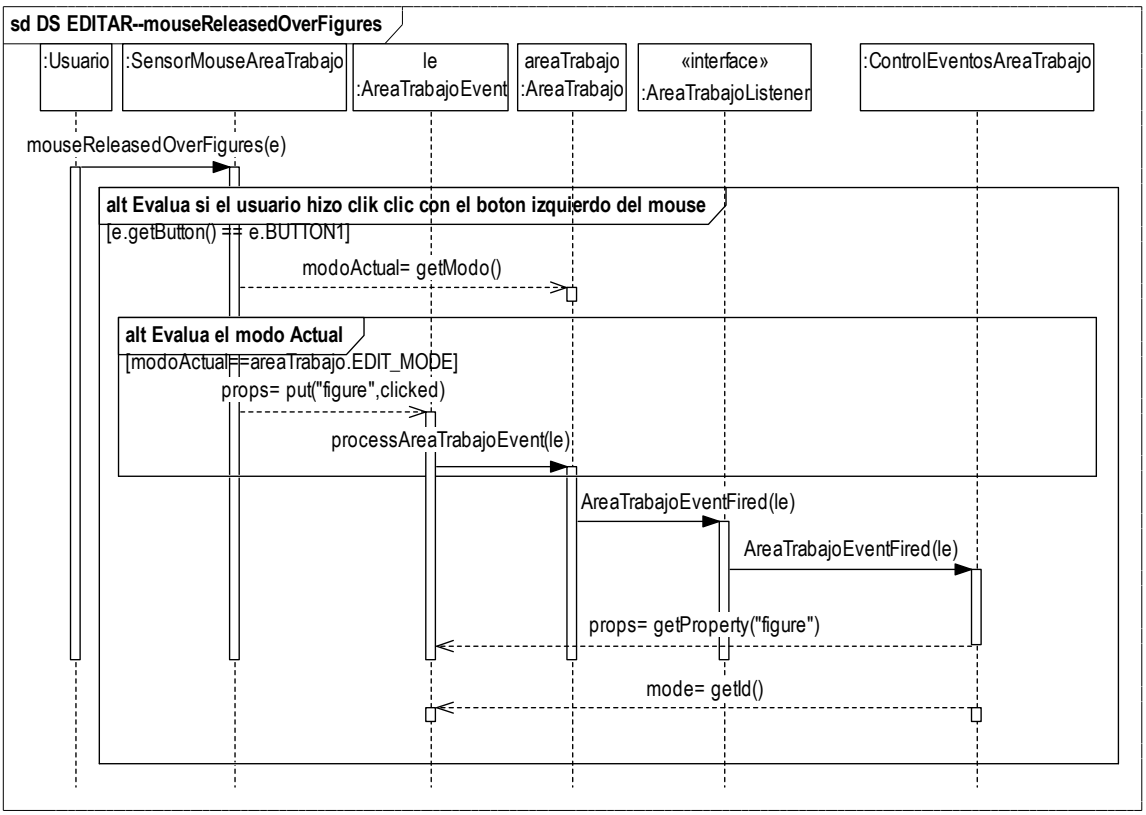
10.7 DIAGRAMA DE SECUENCIA DE EDITAR ESTRUCTURA

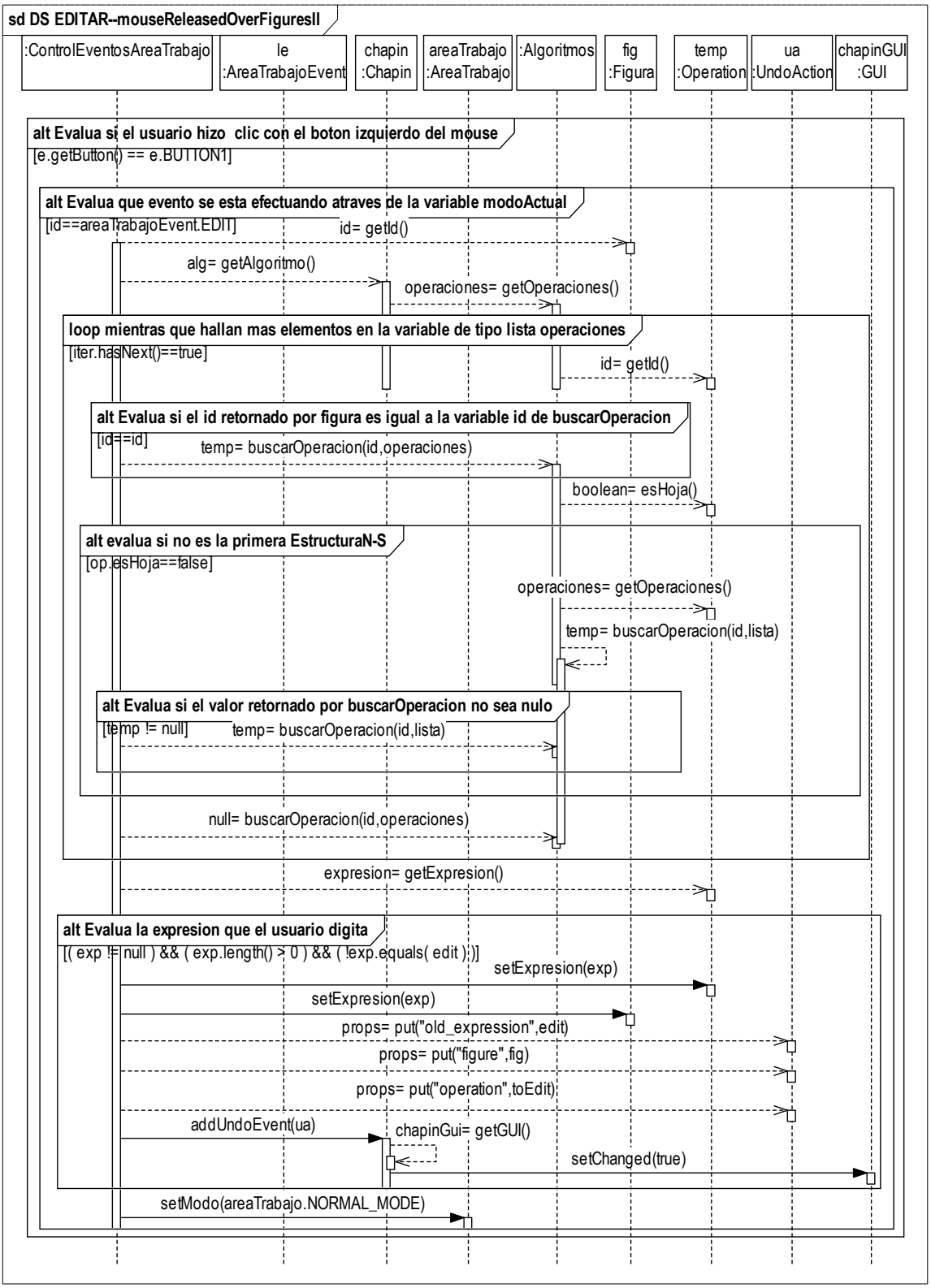



sd DS Editar Estructura-mouseMoved II -Dibujar diagrama
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
:Figura
alt Evalúa en que modo se encuentra
[modoActual==areaTrabajo.EDIT_MODE || modoActual==areaTrabajo.ADD_MODE || modoActual==areaTrabajo.DELETE_MODE]
alt Evalua cuando el mouse pasa por una figura
[actualFigure != null]
addHighlightTo(actualFigure)
modoActual= getModo()
alt Compara la accion que se esta llevando a cabo
[modoActual==DELETE_MODE]
deleteModeColor= getActualModeBorderColor()
setBorderColor(deleteModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke (strokeWidth*3))
alt Verifica ultima figura no sea nula y que la actual figura no sea la ultima
[lastFigure != null && actualFigure != lastFigure]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
[actualFigure == null]
alt Verifica que la ultima figura no sea nula
[lastFigure != null]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)

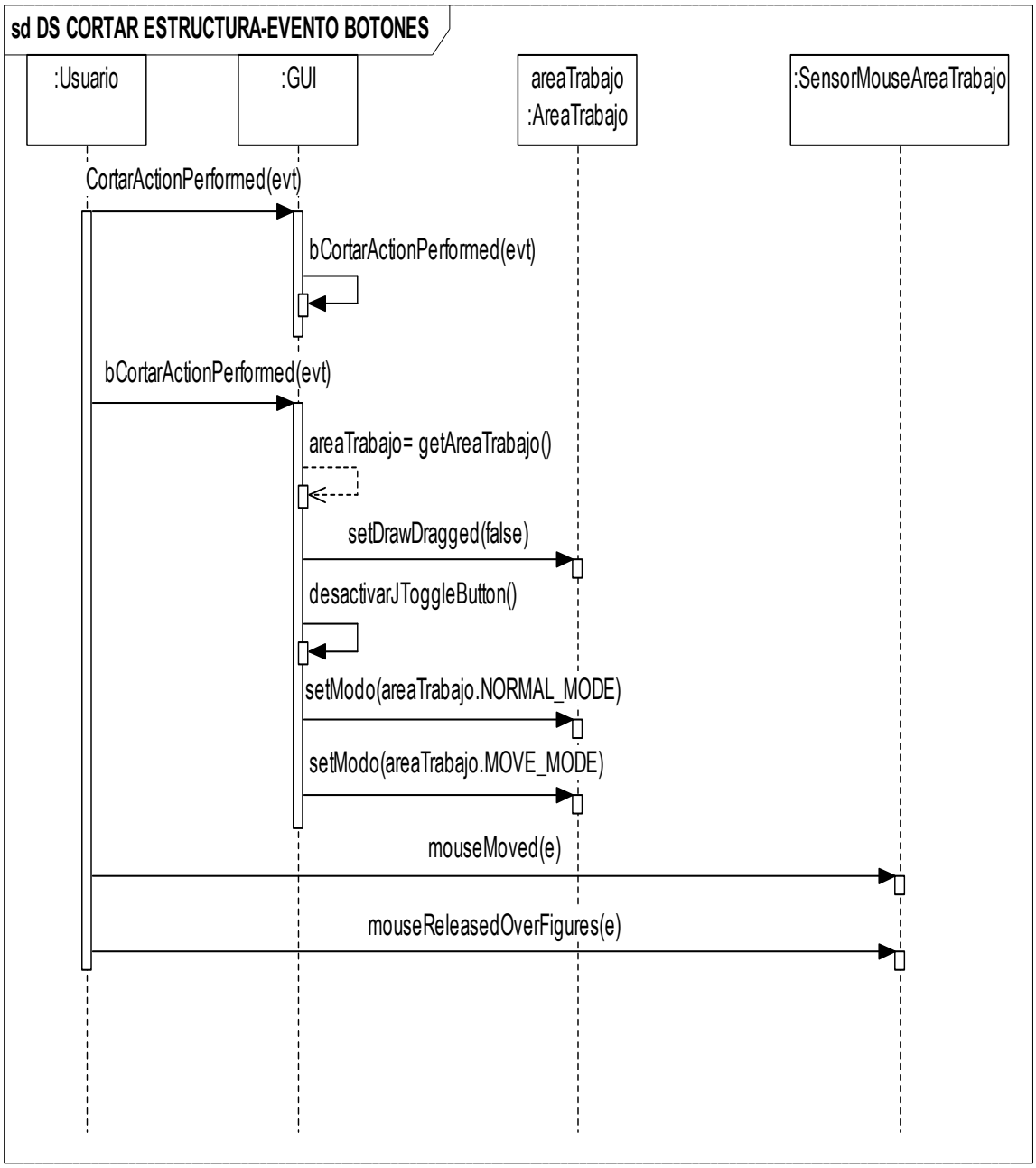

sd DS EDITAR ESTRUCTURA -mouseMoved III-Dibujar diagrama
:SensorMouseAreaTrabajo
areaTrabajo :AreaTrabajo
:Figura
chapin :Chapin
:GUI
chapinGui= getGUI()
boolean= verificarJToggleButtonisSelected()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
Dragged= getDrawDragged()
alt Evalua los diferentes parametros a tener en cuenta
[verificarJToggleButtonisSelected()==false && figuras.isEmpty==false && Dragged==true]
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setDrawDragged(true)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
alt Evalua si el vector figuras no esta vacío
[figuras.isEmpty()==false]
chapinGui= getGUI()
bCopiar= getBotonCopiar()
[figuras.isEmpty()==true]
chapinGui= getGUI()
bCopiar= getBotonCopiar()

10.7.1 Operación mouseReleasedOverFigures ()

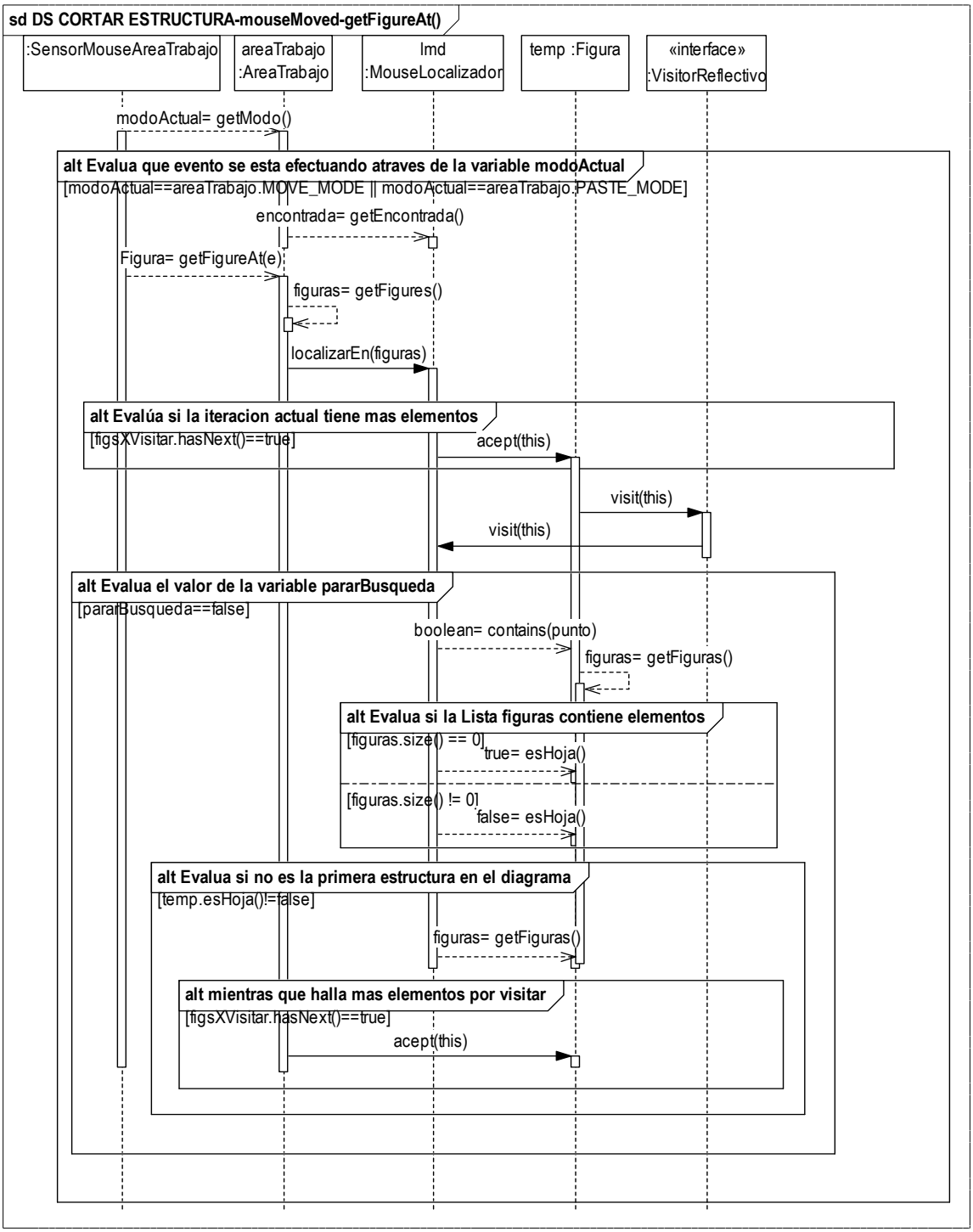


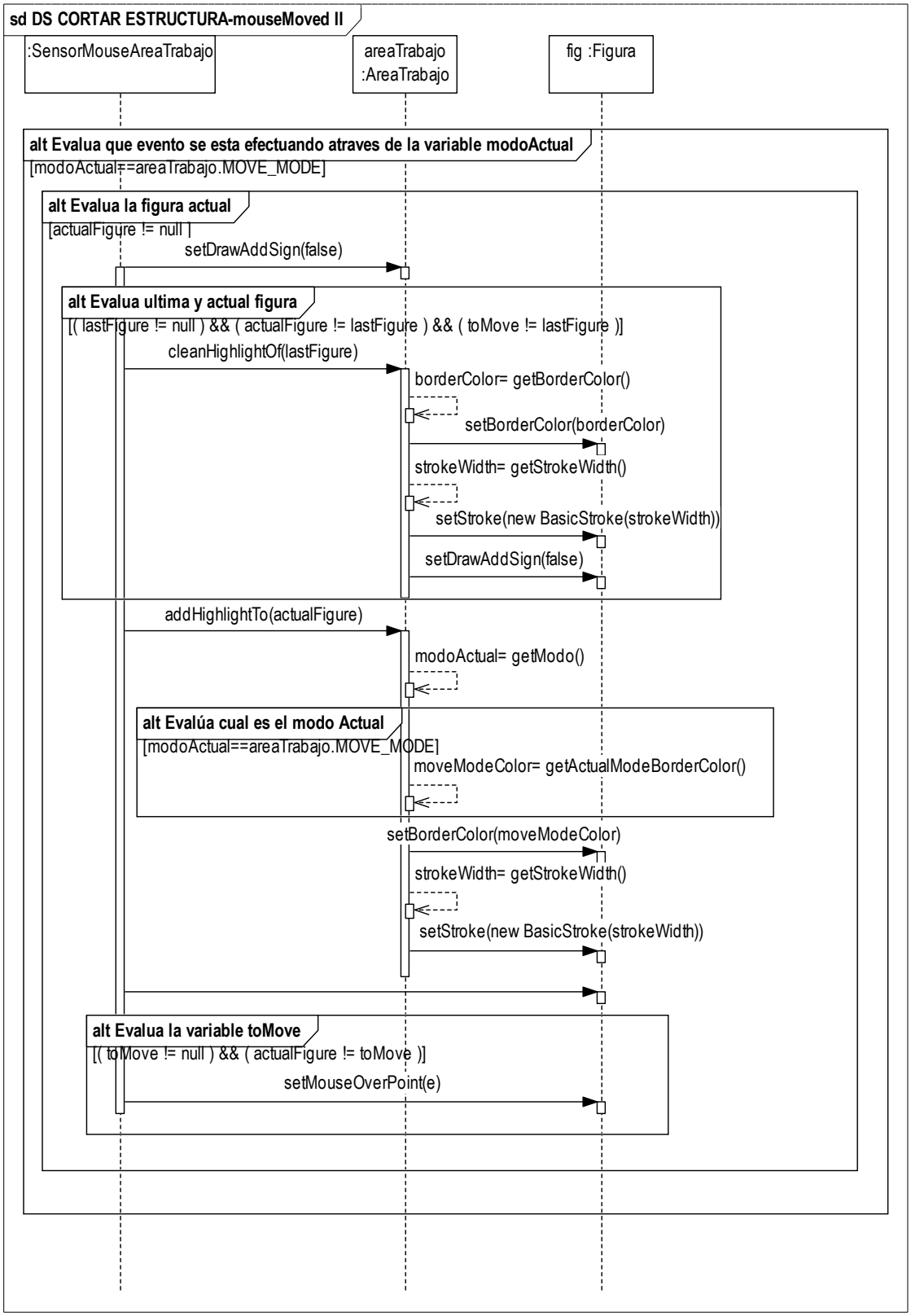

sd DS EDITAR--mouseReleasedOverFiguresII
:ControlEventosAreaTrabajo
le
:AreaTrabajoEvent
chapin
:Chapin
areaTrabajo
:AreaTrabajo
:Algoritmos
fig
:Figura
temp
:Operation
ua
:UndoAction
chapinGUI
:GUI
alt Evalua si el usuario hizo clic con el boton izquierdo del mouse
[e.getButton() == e.BUTTON1]
alt Evalua que evento se esta efectuando atraves de la variable modoActual
[id==areaTrabajoEvent.EDIT]
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua si el id retornado por figura es igual a la variable id de buscarOperacion
[id==id]
temp= buscarOperacion(id,operaciones)
boolean= esHoja()
alt evalua si no es la primera EstructuraN-S
[op.esHoja==false]
operaciones= getOperaciones()
temp= buscarOperacion(id,lista)
alt Evalua si el valor retornado por buscarOperacion no sea nulo
[temp != null]
temp= buscarOperacion(id,lista)
null= buscarOperacion(id,operaciones)
expresion= getExpresion()
alt Evalua la expresion que el usuario digita
[(exp != null) && (exp.length() > 0) && (!exp.equals(edit))]
setExpresion(exp)
setExpresion(exp)
props= put("old_expression",edit)
props= put("figure",fig)
props= put("operation",toEdit)
addUndoEvent(ua)
chapinGui= getGUI()
setChanged(true)
setModo(areaTrabajo.NORMAL_MODE)

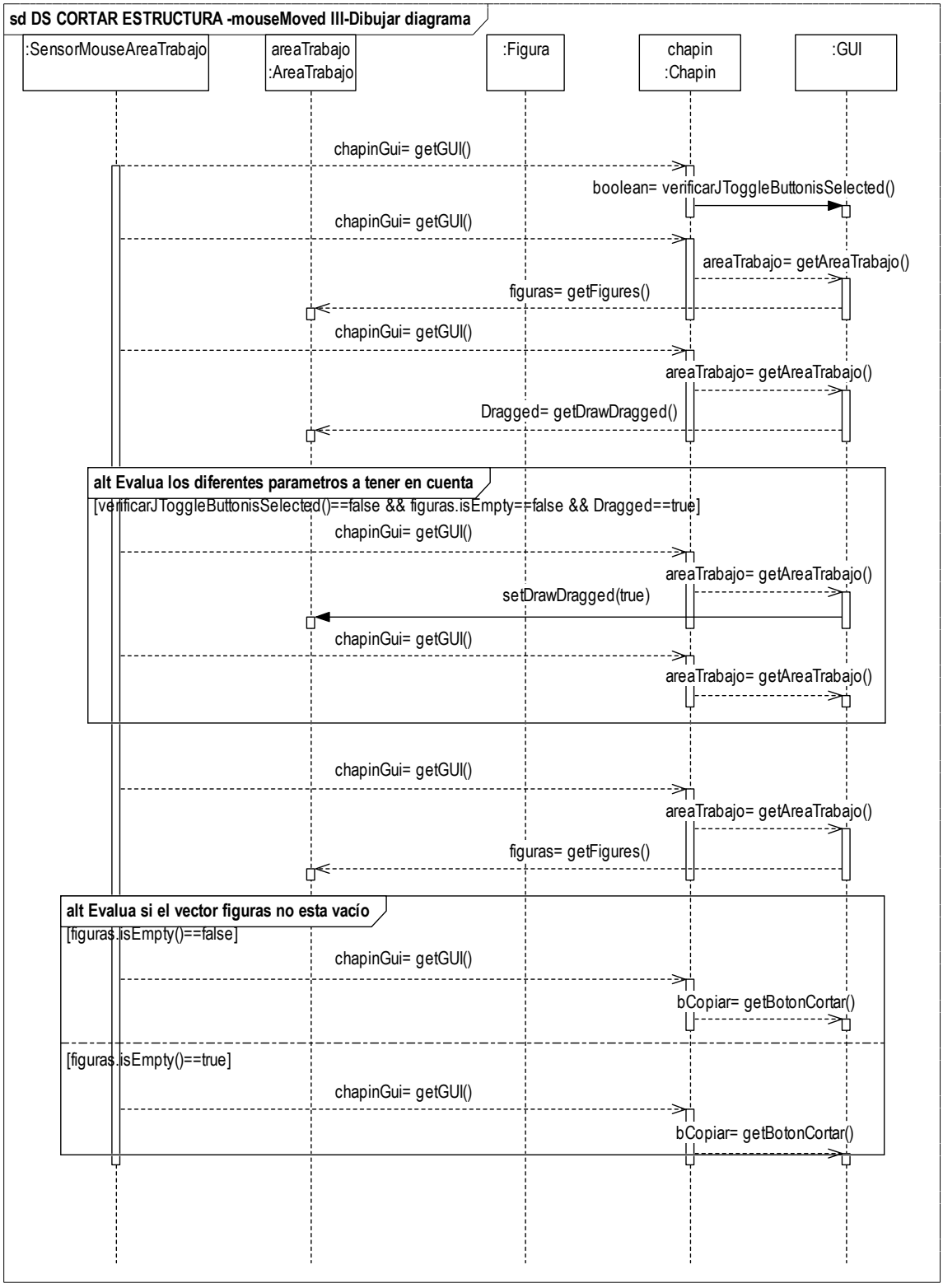
10.8 DIAGRAMA DE SECUENCIA DE CORTAR ESTRUCTURA

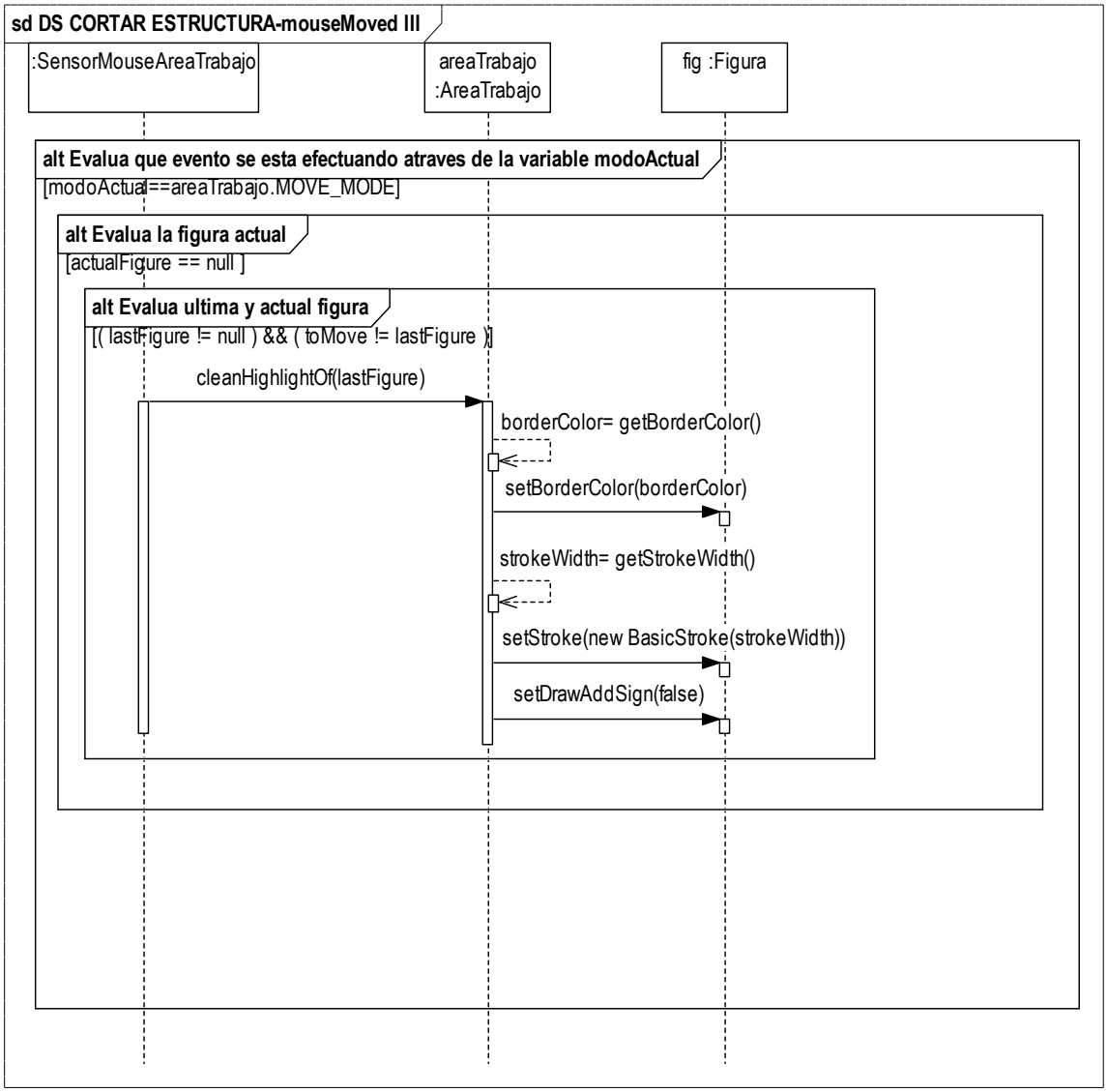


10.8.1 Operación mouseMoved ()

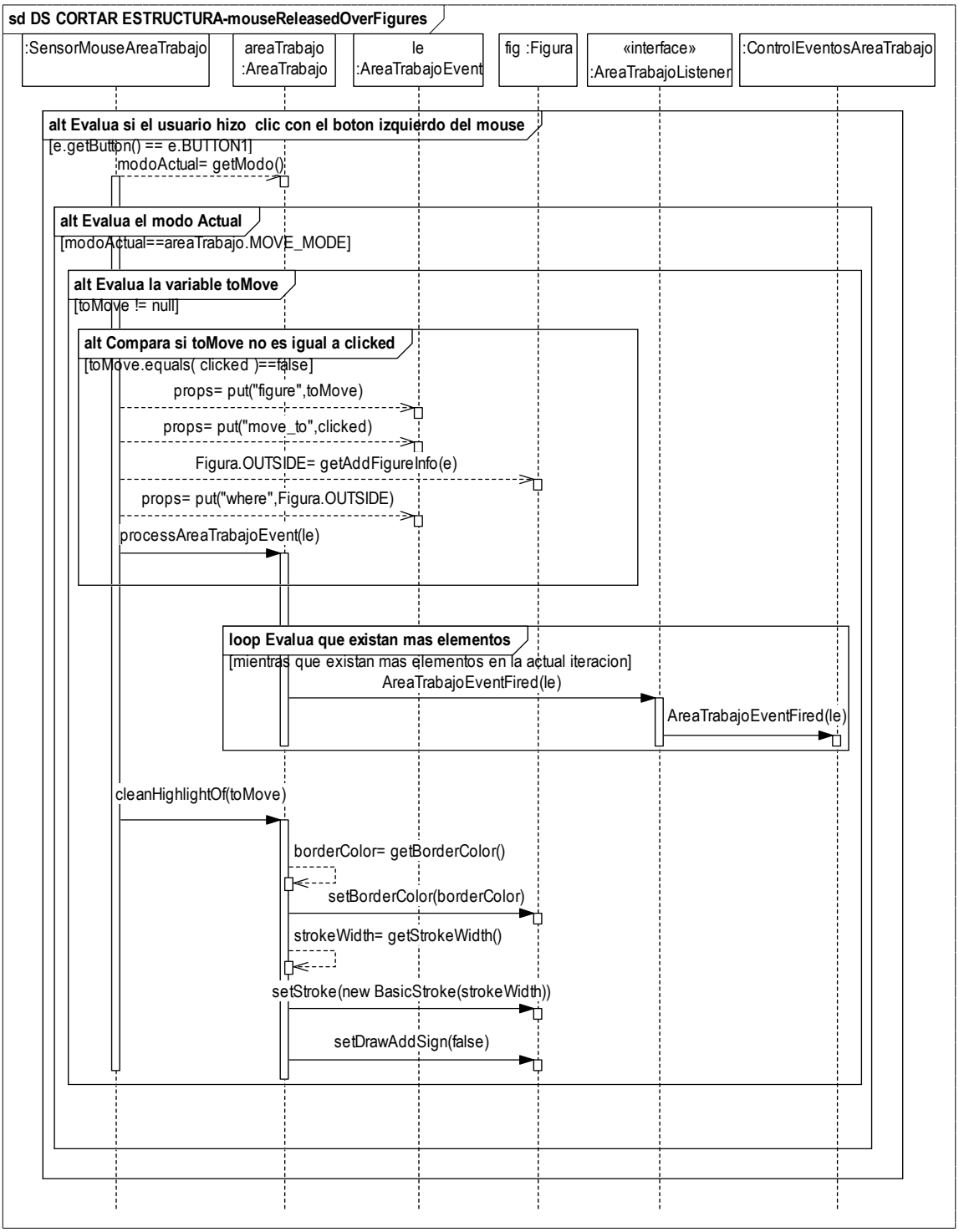


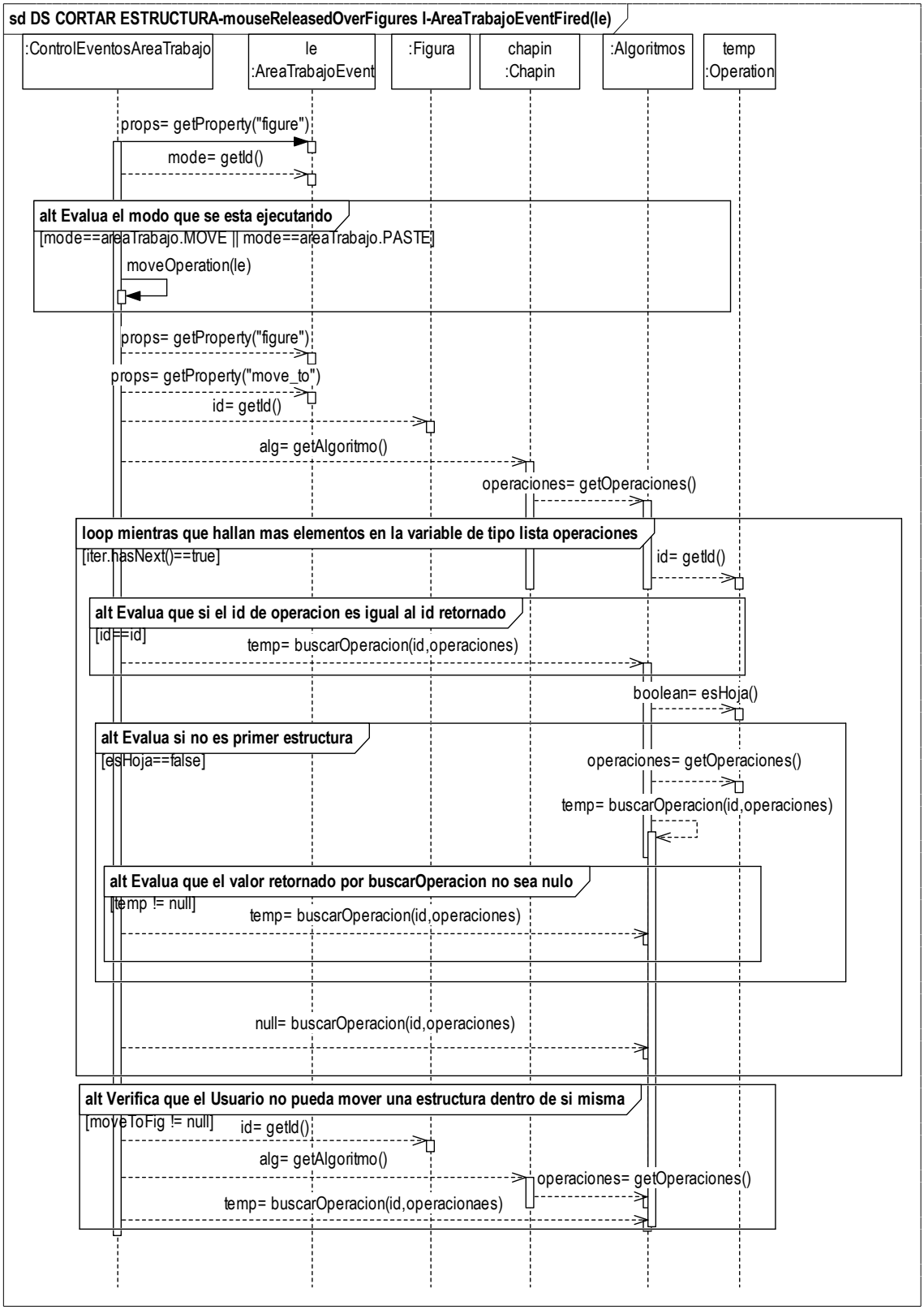

sd DS CORTAR ESTRUCTURA-mouseMoved II
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
fig :Figura
alt Evalua que evento se esta efectuando atraves de la variable modoActual
[modoActual==areaTrabajo.MOVE_MODE]
alt Evalua la figura actual
[actualFigure != null]
setDrawAddSign(false)
alt Evalua ultima y actual figura
[(lastFigure != null) && (actualFigure != lastFigure) && (toMove != lastFigure)]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
addHighlightTo(actualFigure)
modoActual= getModo()
alt Evalúa cual es el modo Actual
[modoActual==areaTrabajo.MOVE_MODE]
moveModeColor= getActualModeBorderColor()
setBorderColor(moveModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
alt Evalua la variable toMove
[(toMove != null) && (actualFigure != toMove)]
setMouseOverPoint(e)

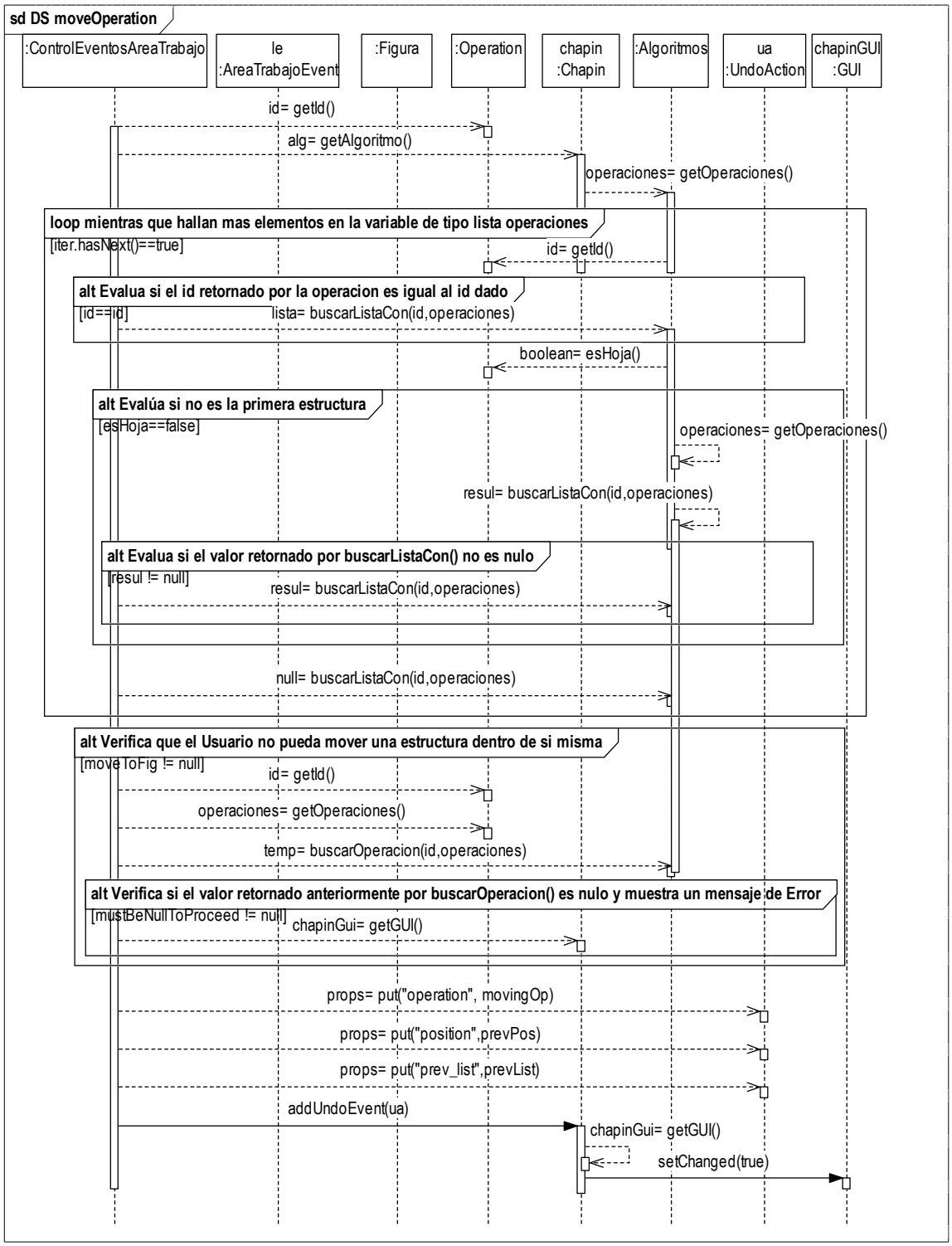

sd DS CORTAR ESTRUCTURA -mouseMoved III-Dibujar diagrama
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
:Figura
chapin
:Chapin
:GUI
chapinGui= getGUI()
boolean= verificarJToggleButtonisSelected()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
Dragged= getDrawDragged()
alt Evalua los diferentes parametros a tener en cuenta
[verificarJToggleButtonisSelected()==false && figuras.isEmpty==false && Dragged==true]
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setDrawDragged(true)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
alt Evalua si el vector figuras no esta vacío
[figuras.isEmpty()==false]
chapinGui= getGUI()
bCopiar= getBotonCortar()
[figuras.isEmpty()==true]
chapinGui= getGUI()
bCopiar= getBotonCortar()

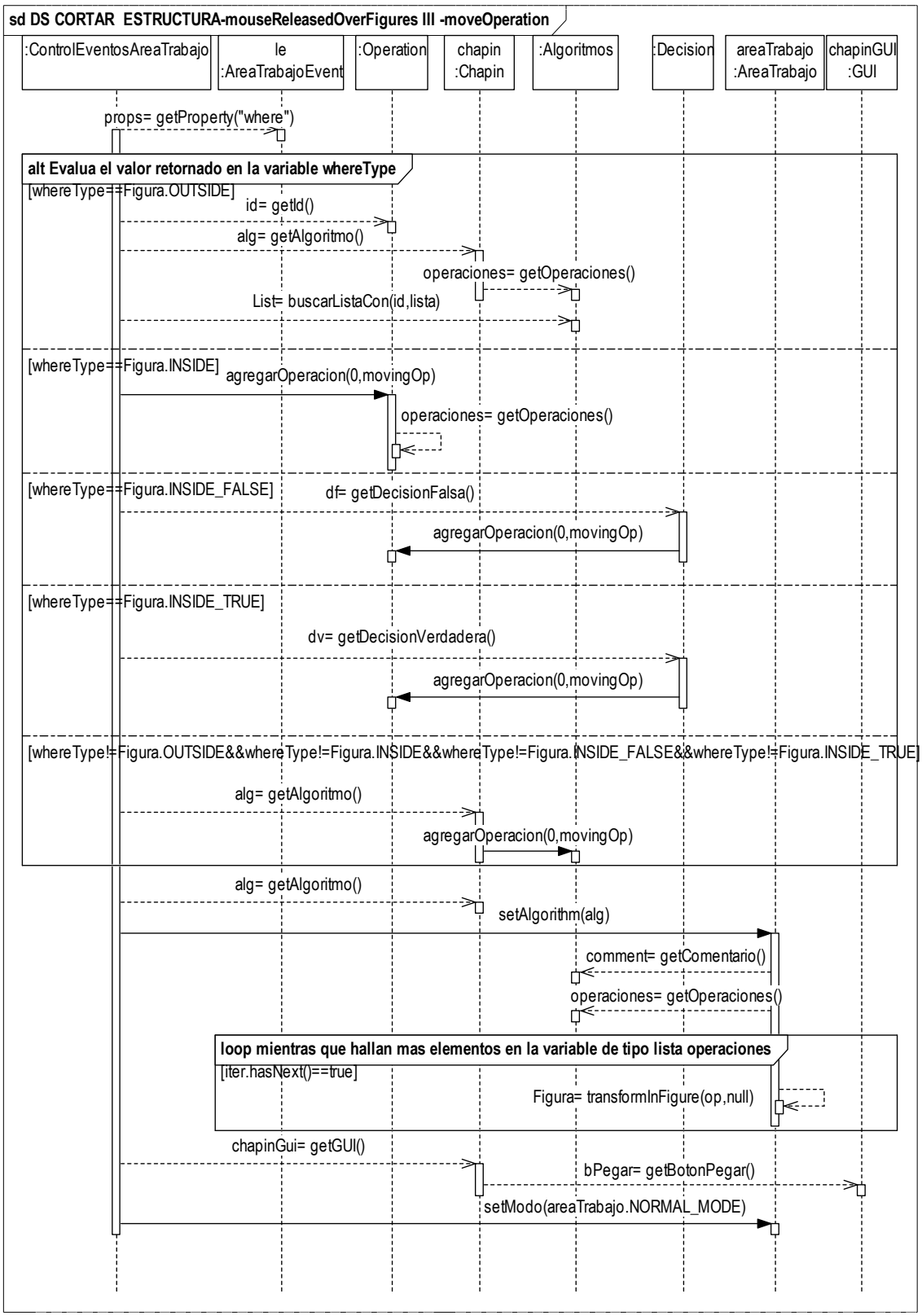

sd DS CORTAR ESTRUCTURA-mouseMoved III
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
fig :Figura
alt Evalua que evento se esta efectuando atraves de la variable modoActual
[modoActual==areaTrabajo.MOVE_MODE]
alt Evalua la figura actual
[actualFigure == null]
alt Evalua ultima y actual figura
[(lastFigure != null) && (toMove != lastFigure)]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)

10.8.2 Operación mouseReleasedOverFigures

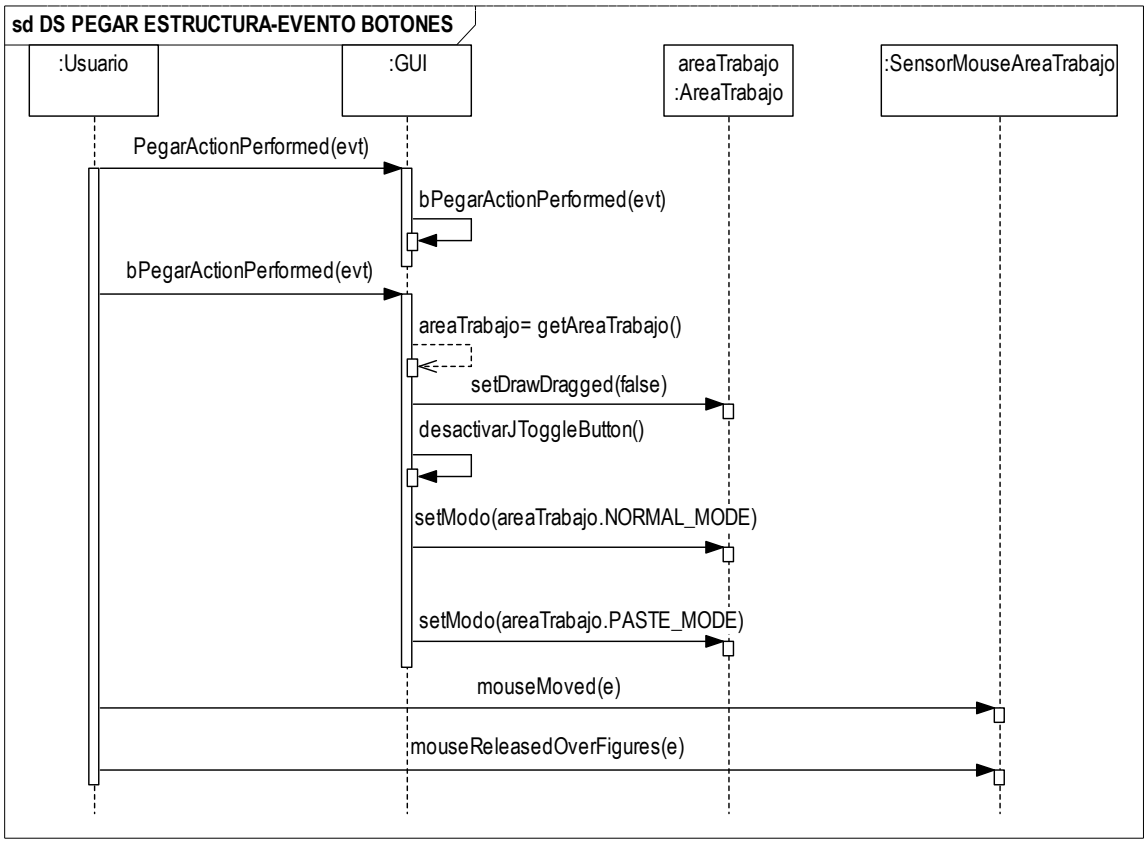



sd DS CORTAR ESTRUCTURA-mouseReleasedOverFigures I-AreaTrabajoEventFired(Ie)
:ControlEventosAreaTrabajo
Ie
:AreaTrabajoEvent
:Figura
chapin
:Chapin
:Algoritmos
temp
:Operation
props= getProperty("figure")
mode= getId()
alt Evalua el modo que se esta ejecutando
[mode==areaTrabajo.MOVE || mode==areaTrabajo.PASTE]
moveOperation(Ie)
props= getProperty("figure")
props= getProperty("move_to")
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua que si el id de operacion es igual al id retornado
[id==id]
temp= buscarOperacion(id,operaciones)
boolean= esHoja()
alt Evalua si no es primer estructura
[esHoja==false]
operaciones= getOperaciones()
temp= buscarOperacion(id,operaciones)
alt Evalua que el valor retornado por buscarOperacion no sea nulo
[temp != null]
temp= buscarOperacion(id,operaciones)
null= buscarOperacion(id,operaciones)
alt Verifica que el Usuario no pueda mover una estructura dentro de si misma
[moveToFig != null]
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
temp= buscarOperacion(id,operacionaes)

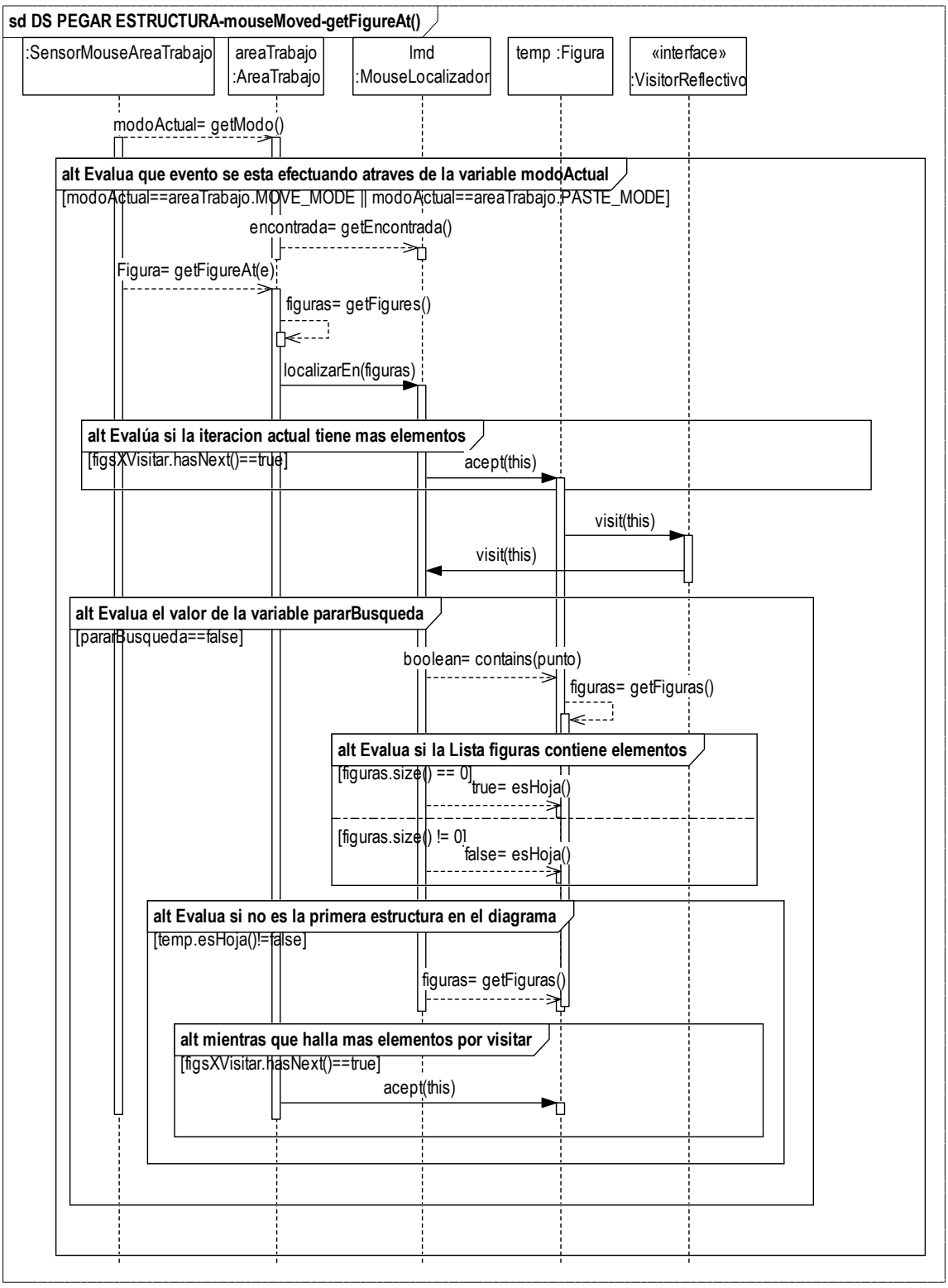

sd DS moveOperation
:ControlEventosAreaTrabajo
le
:AreaTrabajoEvent
:Figura
:Operation
chapin
:Chapin
:Algoritmos
ua
:UndoAction
chapinGUI
:GUI
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua si el id retornado por la operacion es igual al id dado
[id==id]
lista= buscarListaCon(id,operaciones)
boolean= esHoja()
alt Evalúa si no es la primera estructura
[esHoja==false]
operaciones= getOperaciones()
resul= buscarListaCon(id,operaciones)
alt Evalua si el valor retornado por buscarListaCon() no es nulo
[resul != null]
resul= buscarListaCon(id,operaciones)
null= buscarListaCon(id,operaciones)
alt Verifica que el Usuario no pueda mover una estructura dentro de si misma
[moveToFig != null]
id= getId()
operaciones= getOperaciones()
temp= buscarOperacion(id,operaciones)
alt Verifica si el valor retornado anteriormente por buscarOperacion() es nulo y muestra un mensaje de Error
[mustBeNullToProceed != null]
chapinGui= getGUI()
props= put("operation", movingOp)
props= put("position",prevPos)
props= put("prev_list",prevList)
addUndoEvent(ua)
chapinGui= getGUI()
setChanged(true)

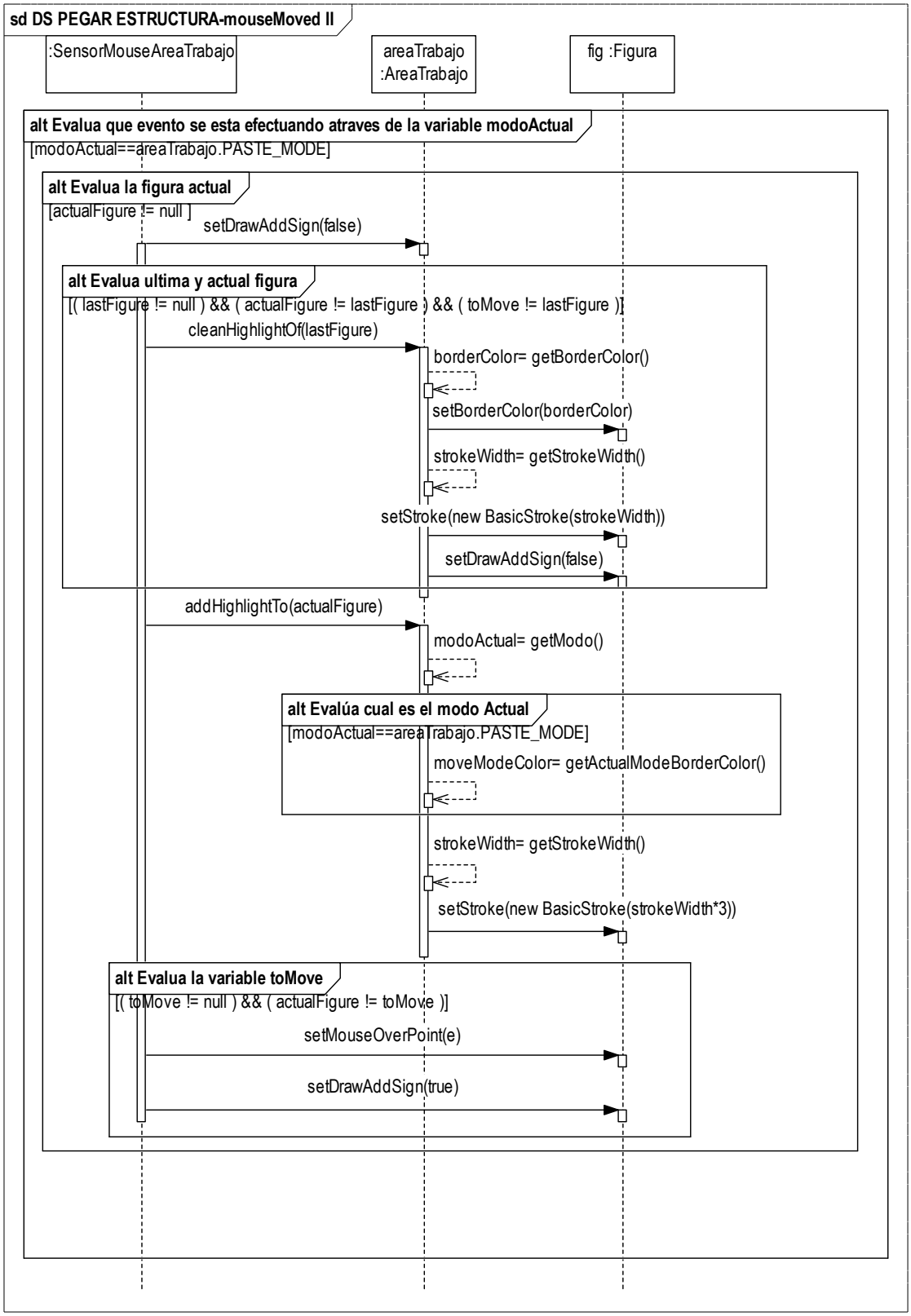

sd DS CORTAR ESTRUCTURA-mouseReleasedOverFigures III -moveOperation
:ControlEventosAreaTrabajo
le
:AreaTrabajoEvent
:Operation
chapin
:Chapin
:Algoritmos
:Decision
areaTrabajo
:AreaTrabajo
chapinGUI
:GUI
props= getProperty("where")
alt Evalua el valor retornado en la variable whereType
[whereType==Figura.OUTSIDE]
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
List= buscarListaCon(id,lista)
[whereType==Figura.INSIDE]
agregarOperacion(0,movingOp)
operaciones= getOperaciones()
[whereType==Figura.INSIDE_FALSE]
df= getDecisionFalsa()
agregarOperacion(0,movingOp)
[whereType==Figura.INSIDE_TRUE]
dv= getDecisionVerdadera()
agregarOperacion(0,movingOp)
[whereType!=Figura.OUTSIDE&&whereType!=Figura.INSIDE&&whereType!=Figura.INSIDE_FALSE&&whereType!=Figura.INSIDE_TRUE]
alg= getAlgoritmo()
agregarOperacion(0,movingOp)
alg= getAlgoritmo()
setAlgorithm(alg)
comment= getComentario()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
Figura= transformInFigure(op,null)
chapinGui= getGUI()
bPegar= getBotonPegar()
setModo(areaTrabajo.NORMAL_MODE)

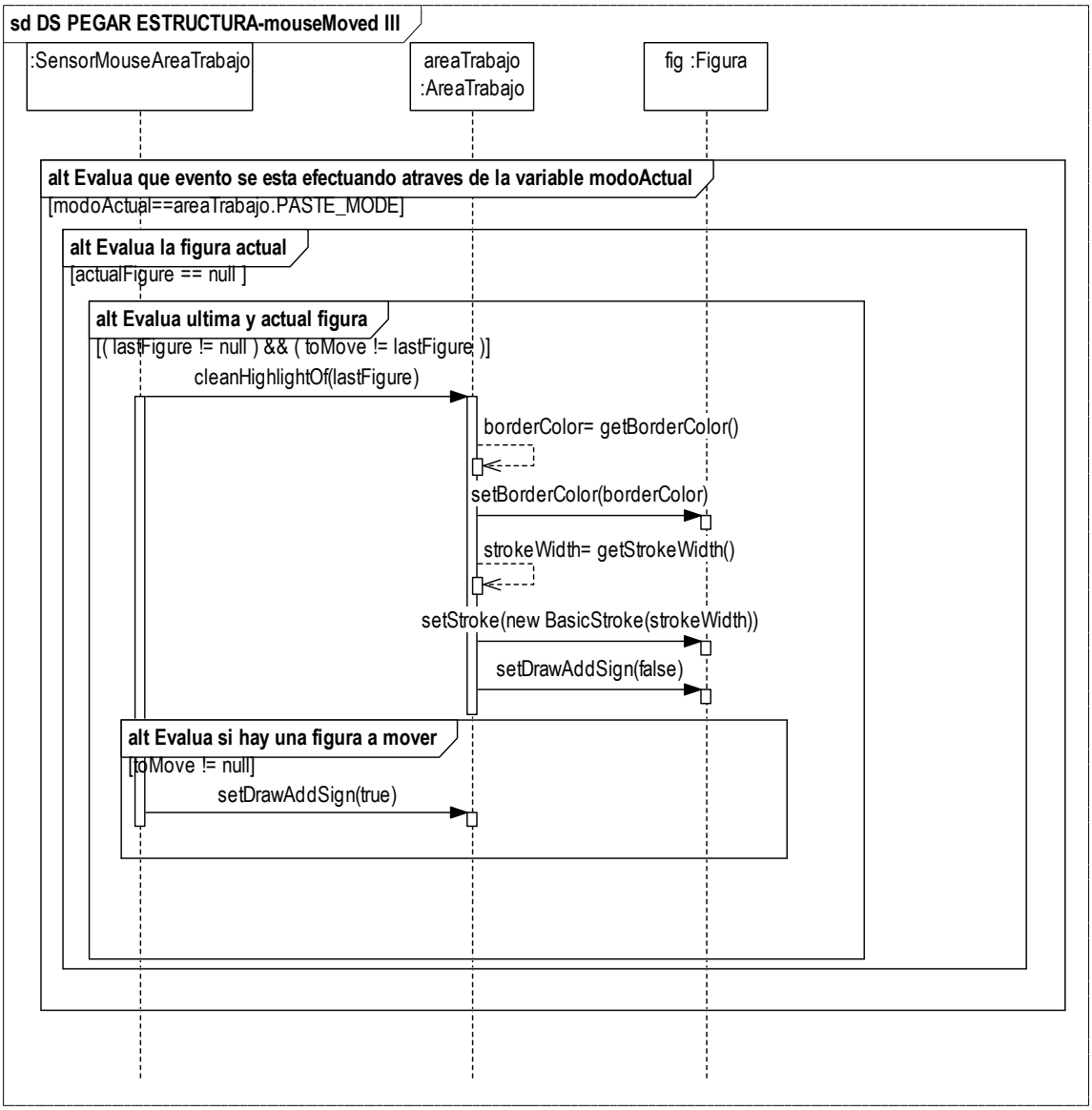
10.9 DIAGRAMA DE SECUENCIA DE PEGAR ESTRUCTURA

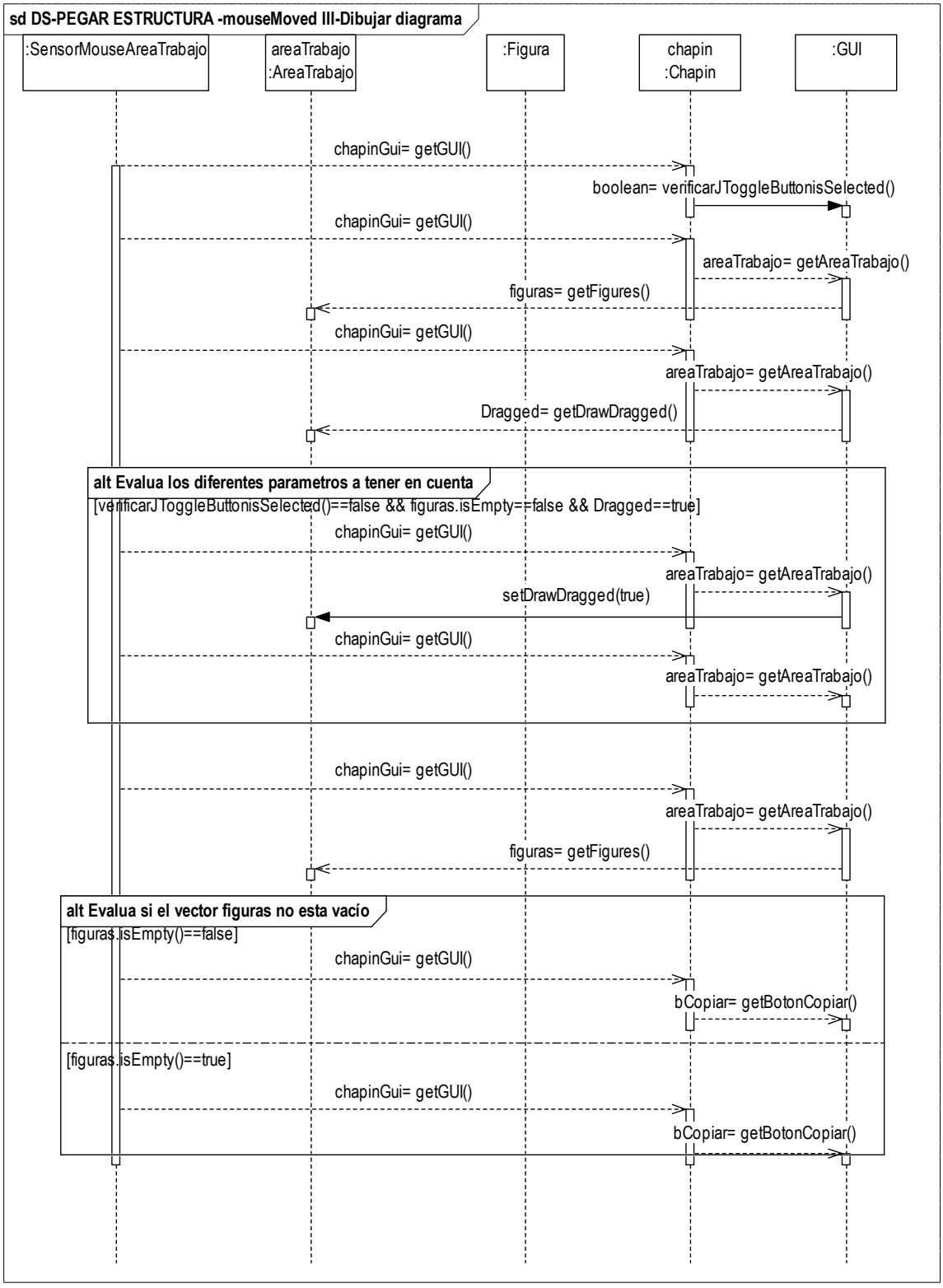


10.9.1 Operación mouseMoved ()

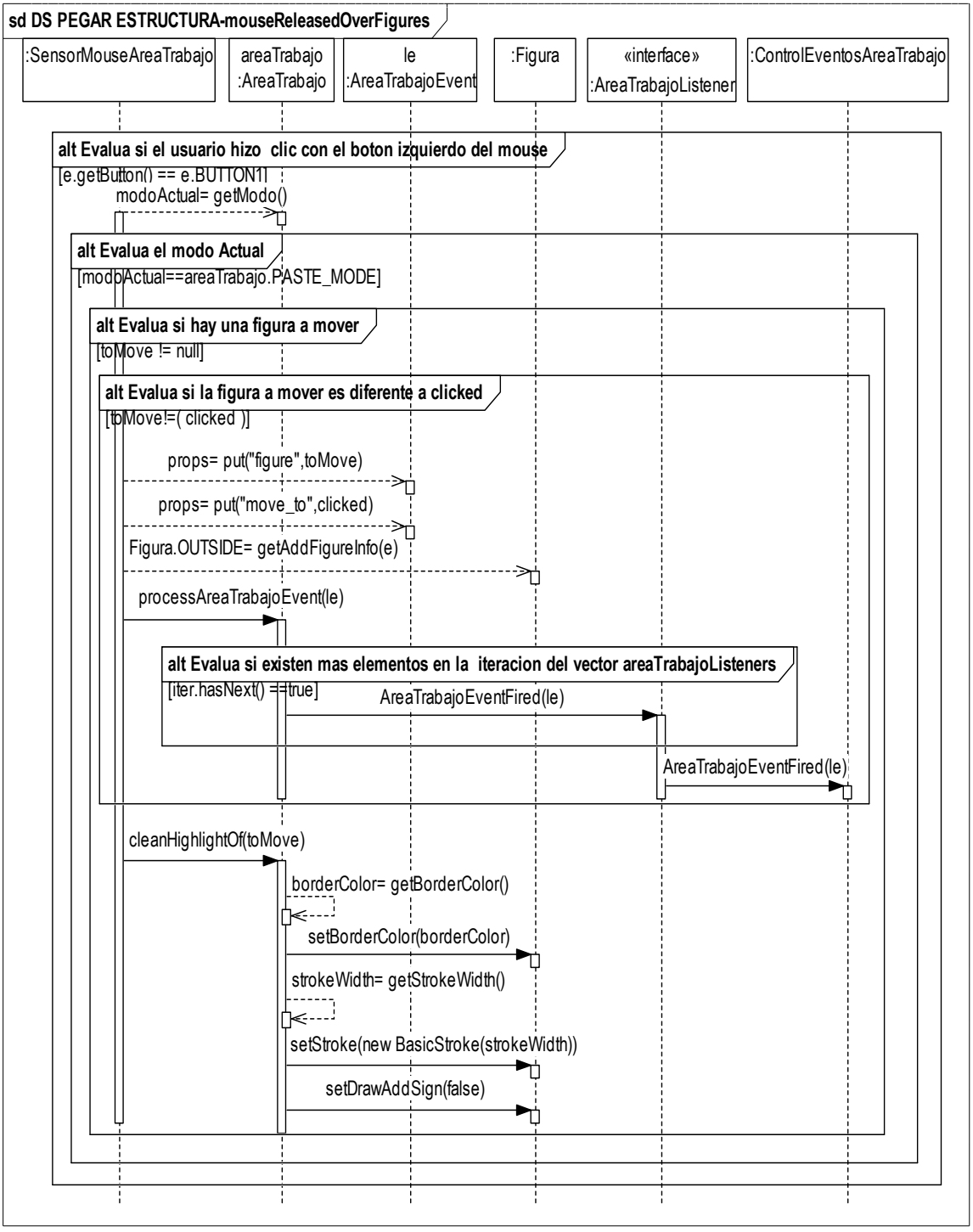


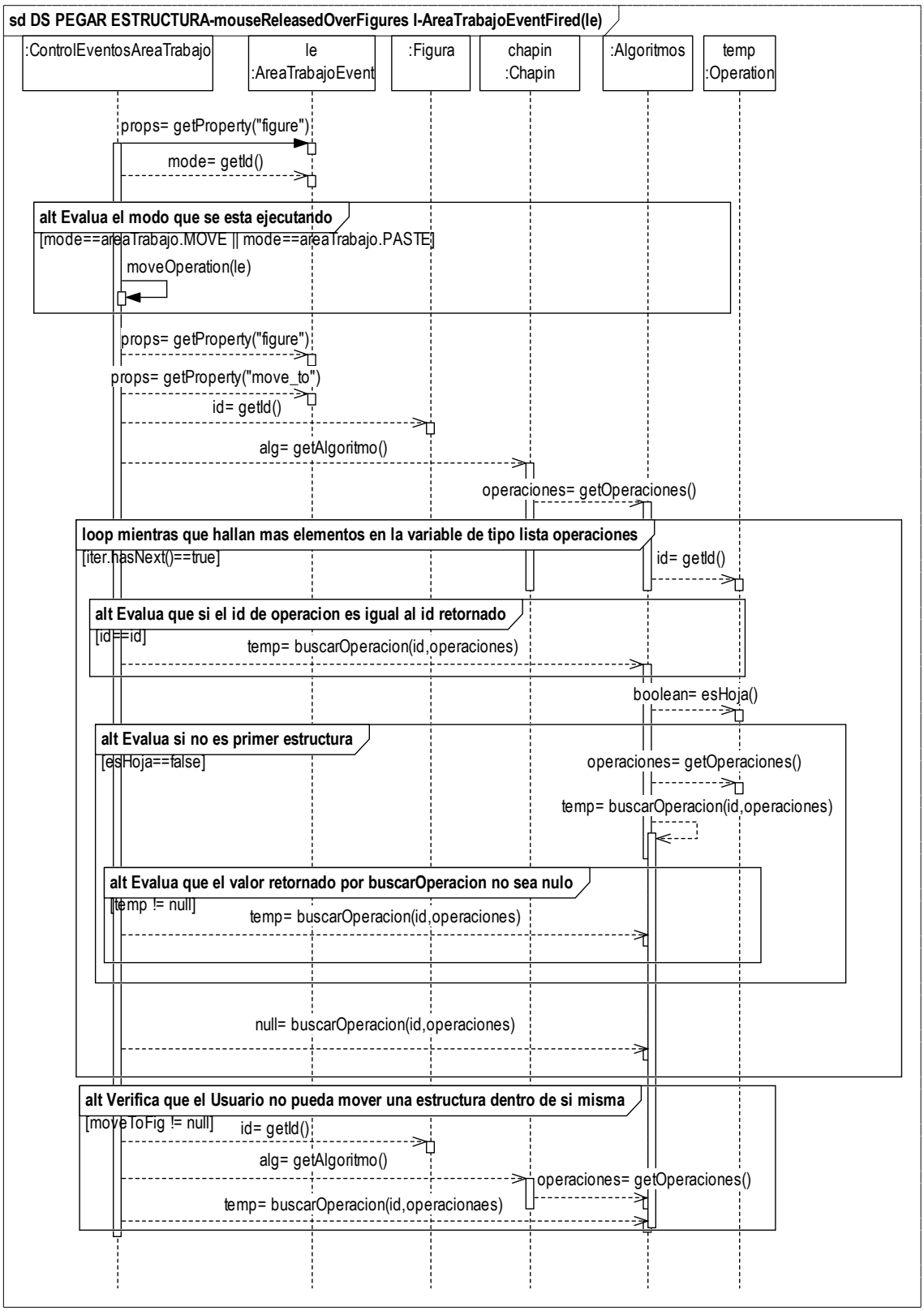

sd DS PEGAR ESTRUCTURA-mouseMoved II
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
fig :Figura
alt Evalua que evento se esta efectuando atraves de la variable modoActual
[modoActual==areaTrabajo.PASTE_MODE]
alt Evalua la figura actual
[actualFigure != null]
setDrawAddSign(false)
alt Evalua ultima y actual figura
[(lastFigure != null) && (actualFigure != lastFigure) && (toMove != lastFigure)]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
addHighlightTo(actualFigure)
modoActual= getModo()
alt Evalúa cual es el modo Actual
[modoActual==areaTrabajo.PASTE_MODE]
moveModeColor= getActualModeBorderColor()
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth*3))
alt Evalua la variable toMove
[(toMove != null) && (actualFigure != toMove)]
setMouseOverPoint(e)
setDrawAddSign(true)

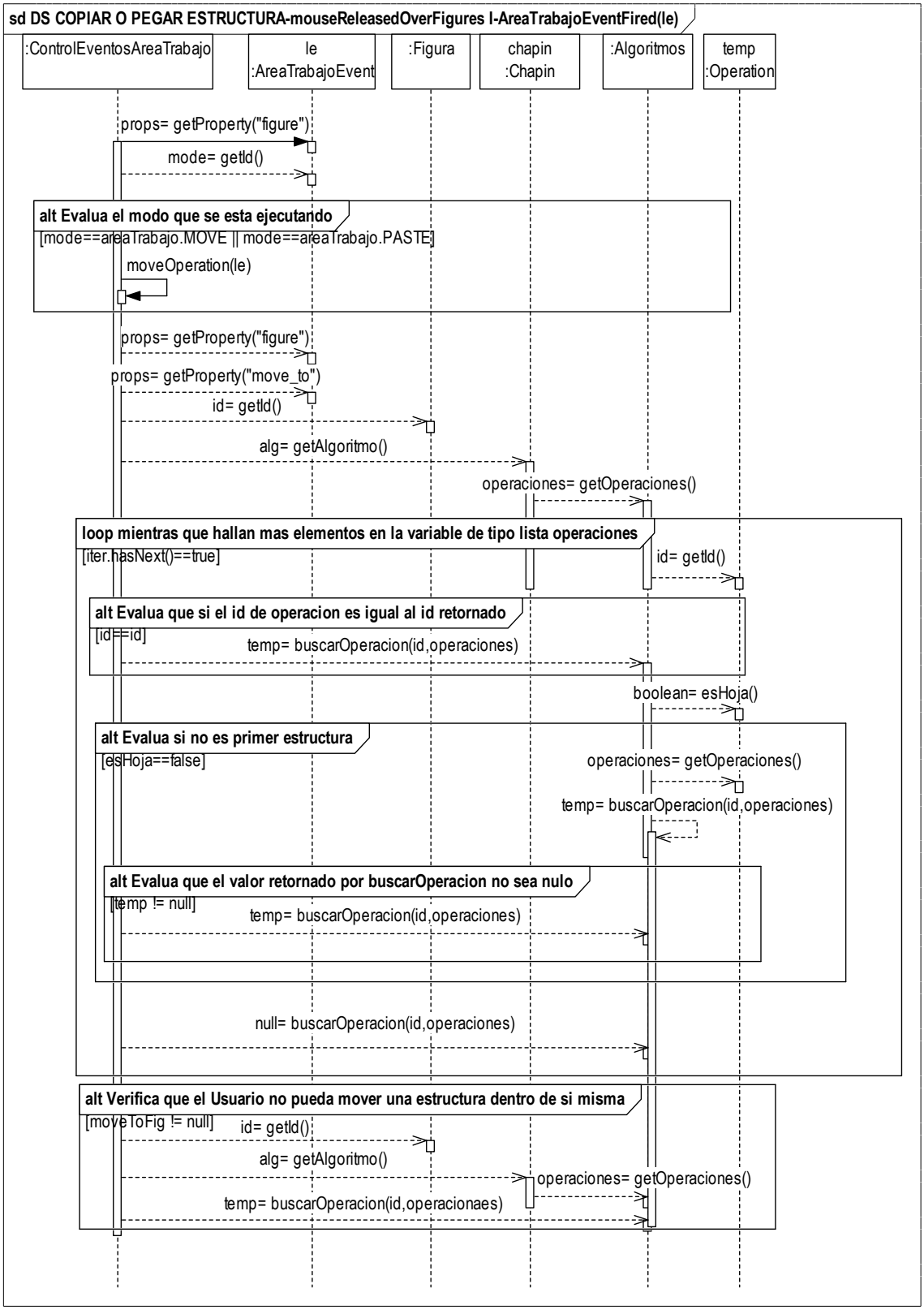

sd DS PEGAR ESTRUCTURA-mouseMoved III
:SensorMouseAreaTrabajo
areaTrabajo
:AreaTrabajo
fig :Figura
alt Evalua que evento se esta efectuando atraves de la variable modoActual
[modoActual==areaTrabajo.PASTE_MODE]
alt Evalua la figura actual
[actualFigure == null]
alt Evalua ultima y actual figura
[(lastFigure != null) && (toMove != lastFigure)]
cleanHighlightOf(lastFigure)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
alt Evalua si hay una figura a mover
[toMove != null]
setDrawAddSign(true)

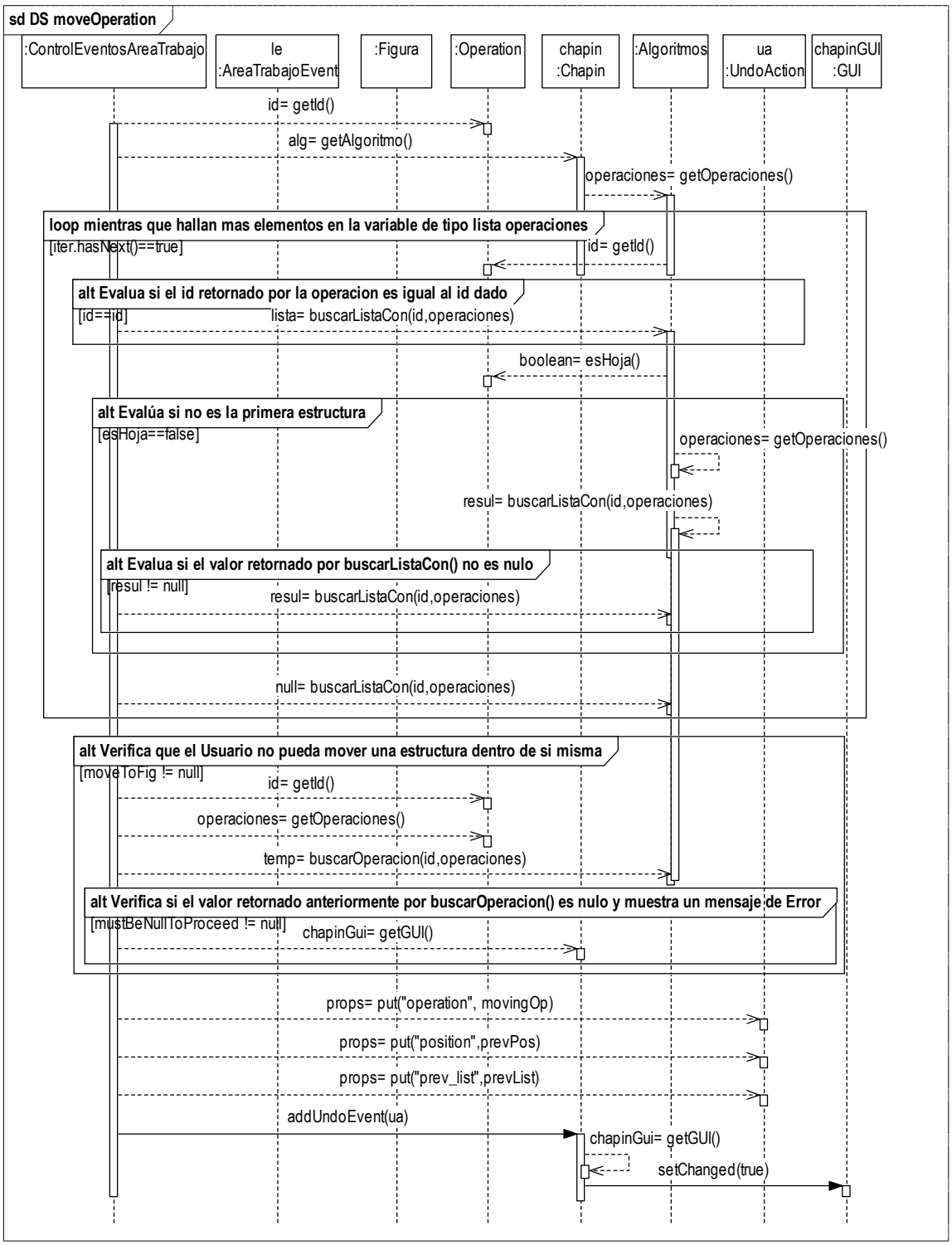

sd DS-PEGAR ESTRUCTURA -mouseMoved III-Dibujar diagrama
:SensorMouseAreaTrabajo
areaTrabajo :AreaTrabajo
:Figura
chapin :Chapin
:GUI
chapinGui= getGUI()
boolean= verificarJToggleButtonisSelected()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
Dragged= getDrawDragged()
alt Evalua los diferentes parametros a tener en cuenta
[verificarJToggleButtonisSelected()==false && figuras.isEmpty==false && Dragged==true]
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setDrawDragged(true)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
alt Evalua si el vector figuras no esta vacío
[figuras.isEmpty()==false]
chapinGui= getGUI()
bCopiar= getBotonCopiar()
[figuras.isEmpty()==true]
chapinGui= getGUI()
bCopiar= getBotonCopiar()

10.9.2 Operación mouseReleasedOverFigures

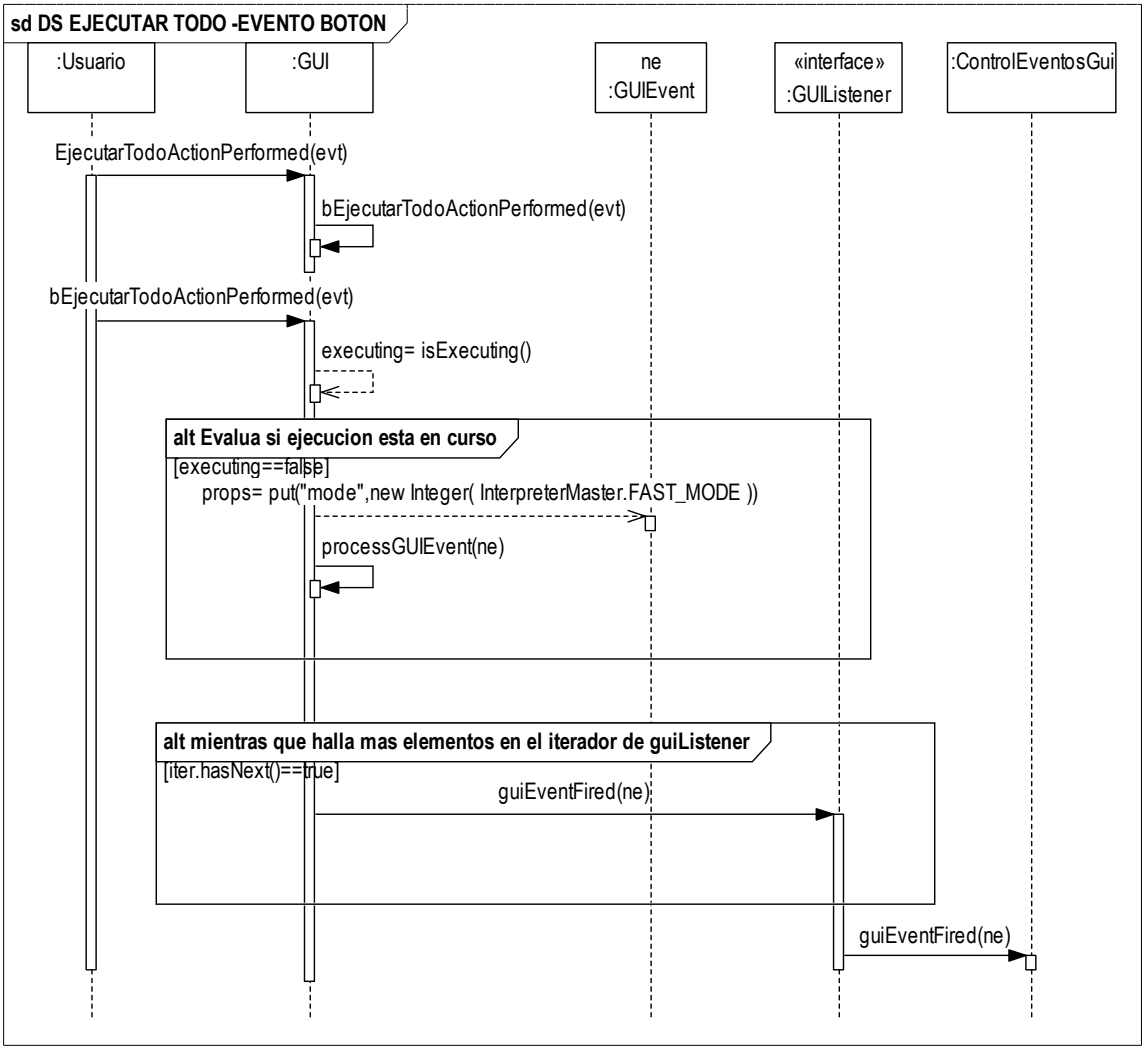


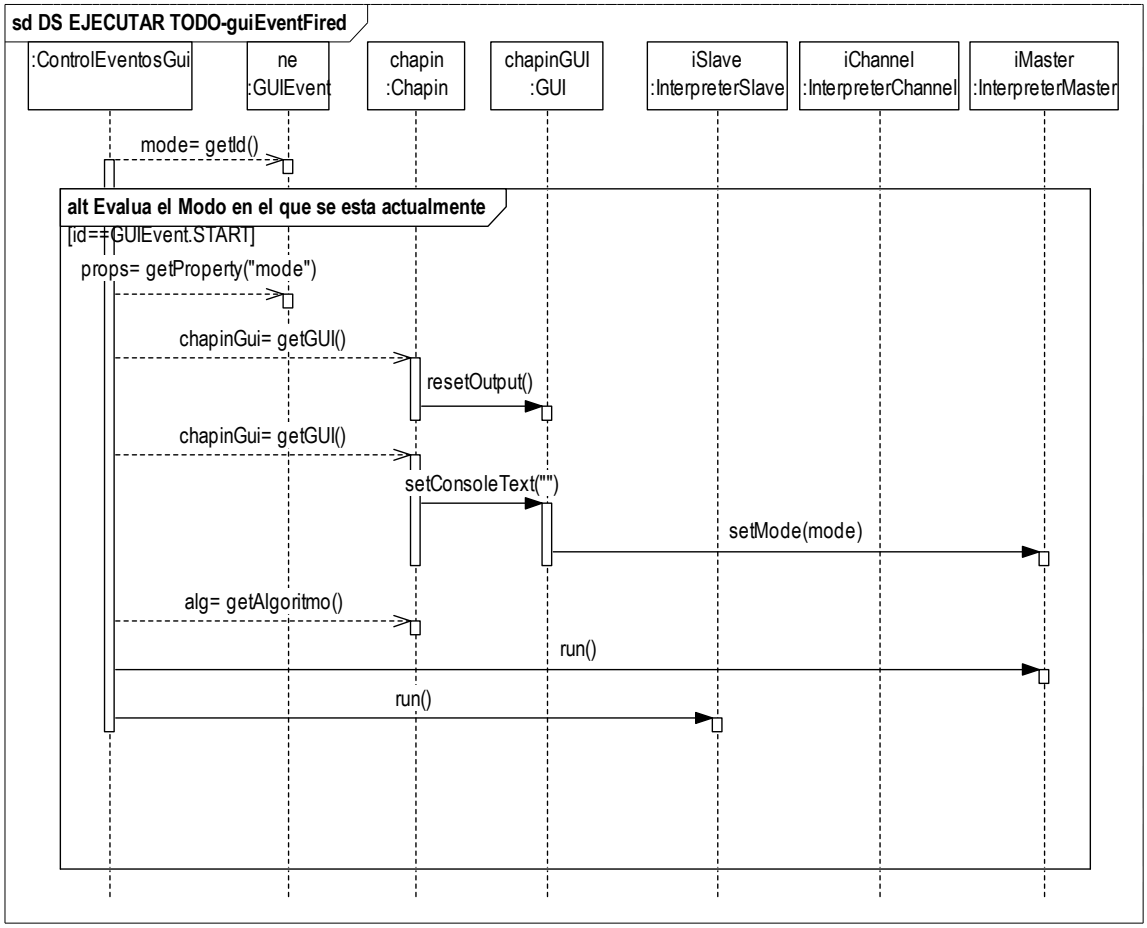

sd DS PEGAR ESTRUCTURA-mouseReleasedOverFigures l-AreaTrabajoEventFired(le)
:ControlEventosAreaTrabajo
le
:AreaTrabajoEvent
:Figura
chapin
:Chapin
:Algoritmos
temp
:Operation
props= getProperty("figure")
mode= getId()
alt Evalua el modo que se esta ejecutando
[mode==areaTrabajo.MOVE || mode==areaTrabajo.PASTE]
moveOperation(le)
props= getProperty("figure")
props= getProperty("move_to")
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua que si el id de operacion es igual al id retornado
[id==id]
temp= buscarOperacion(id,operaciones)
boolean= esHoja()
alt Evalua si no es primer estructura
[esHoja==false]
operaciones= getOperaciones()
temp= buscarOperacion(id,operaciones)
alt Evalua que el valor retornado por buscarOperacion no sea nulo
[temp != null]
temp= buscarOperacion(id,operaciones)
null= buscarOperacion(id,operaciones)
alt Verifica que el Usuario no pueda mover una estructura dentro de si misma
[moveToFig != null]
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
temp= buscarOperacion(id,operacionaes)


sd DS COPIAR O PEGAR ESTRUCTURA-mouseReleasedOverFigures l-AreaTrabajoEventFired(le)
:ControlEventosAreaTrabajo
le
:AreaTrabajoEvent
:Figura
chapin
:Chapin
:Algoritmos
temp
:Operation
props= getProperty("figure")
mode= getId()
alt Evalua el modo que se esta ejecutando
[mode==areaTrabajo.MOVE || mode==areaTrabajo.PASTE]
moveOperation(le)
props= getProperty("figure")
props= getProperty("move_to")
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua que si el id de operacion es igual al id retornado
[id==id]
temp= buscarOperacion(id,operaciones)
boolean= esHoja()
alt Evalua si no es primer estructura
[esHoja==false]
operaciones= getOperaciones()
temp= buscarOperacion(id,operaciones)
alt Evalua que el valor retornado por buscarOperacion no sea nulo
[temp != null]
temp= buscarOperacion(id,operaciones)
null= buscarOperacion(id,operaciones)
alt Verifica que el Usuario no pueda mover una estructura dentro de si misma
[moveToFig != null]
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
temp= buscarOperacion(id,operacionaes)

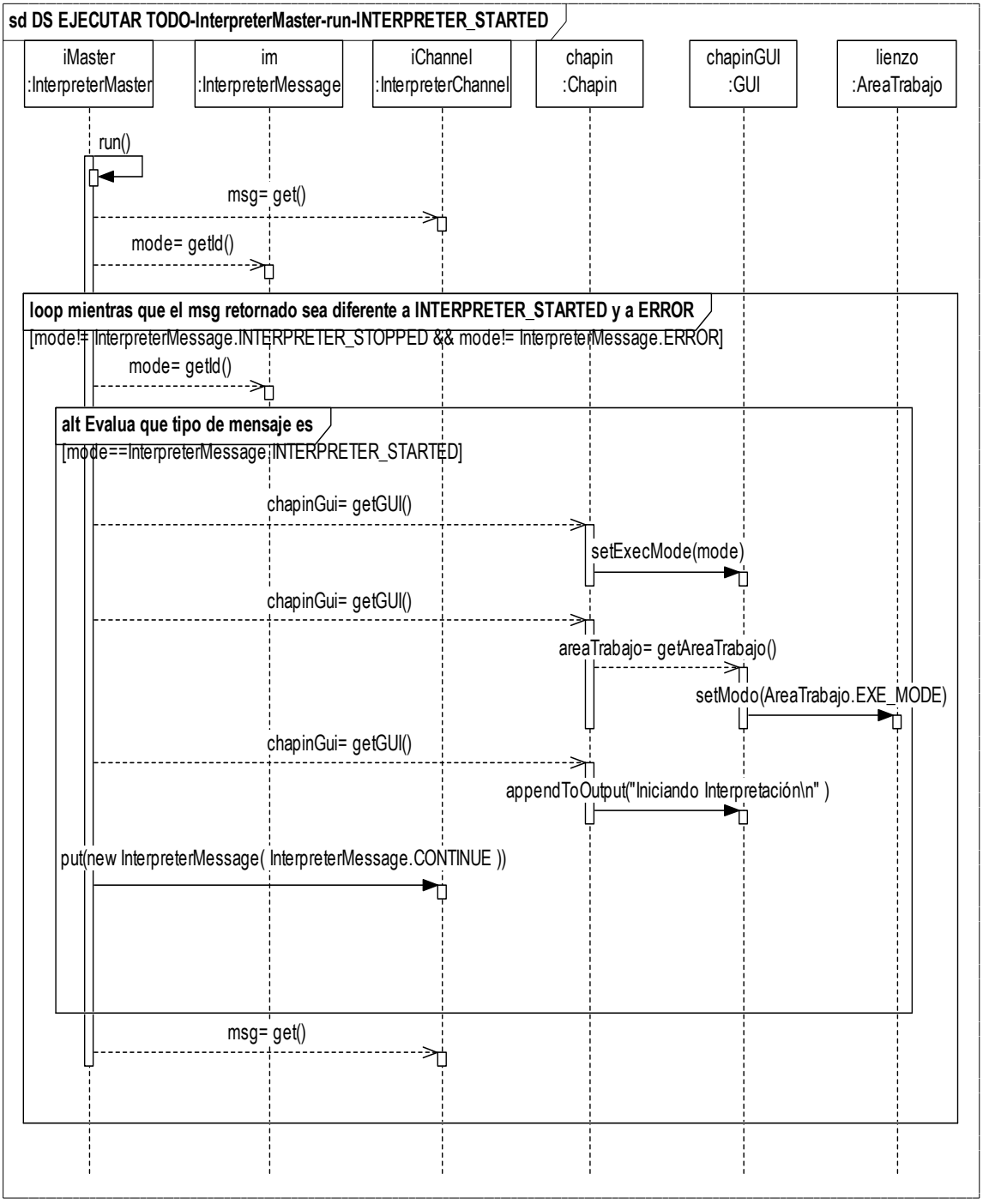

sd DS moveOperation
:ControlEventosAreaTrabajo
le
:AreaTrabajoEvent
:Figura
:Operation
chapin
:Chapin
:Algoritmos
ua
:UndoAction
chapinGUI
:GUI
id= getId()
alg= getAlgoritmo()
operaciones= getOperaciones()
loop mientras que hallan mas elementos en la variable de tipo lista operaciones
[iter.hasNext()==true]
id= getId()
alt Evalua si el id retornado por la operacion es igual al id dado
[id==id]
lista= buscarListaCon(id,operaciones)
boolean= esHoja()
alt Evalúa si no es la primera estructura
[esHoja==false]
operaciones= getOperaciones()
resul= buscarListaCon(id,operaciones)
alt Evalua si el valor retornado por buscarListaCon() no es nulo
[resul != null]
resul= buscarListaCon(id,operaciones)
null= buscarListaCon(id,operaciones)
alt Verifica que el Usuario no pueda mover una estructura dentro de si misma
[moveToFig != null]
id= getId()
operaciones= getOperaciones()
temp= buscarOperacion(id,operaciones)
alt Verifica si el valor retornado anteriormente por buscarOperacion() es nulo y muestra un mensaje de Error
[mustBeNullToProceed != null]
chapinGui= getGUI()
props= put("operation", movingOp)
props= put("position",prevPos)
props= put("prev_list",prevList)
addUndoEvent(ua)
chapinGui= getGUI()
setChanged(true)

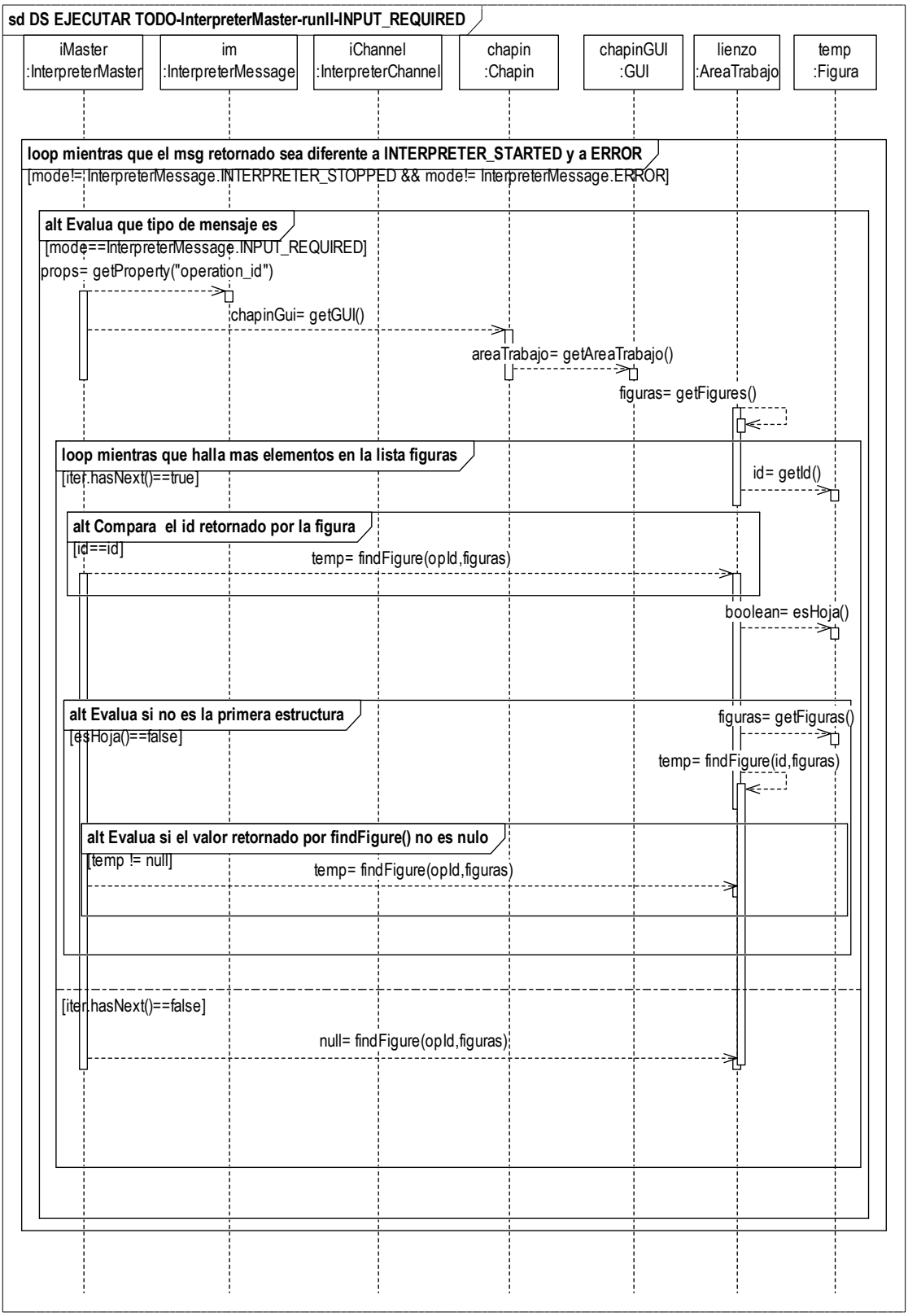
10.10 DIAGRAMA DE SECUENCIA DE EJECUTAR TODO (EJECUTAR DIAGRAMA)

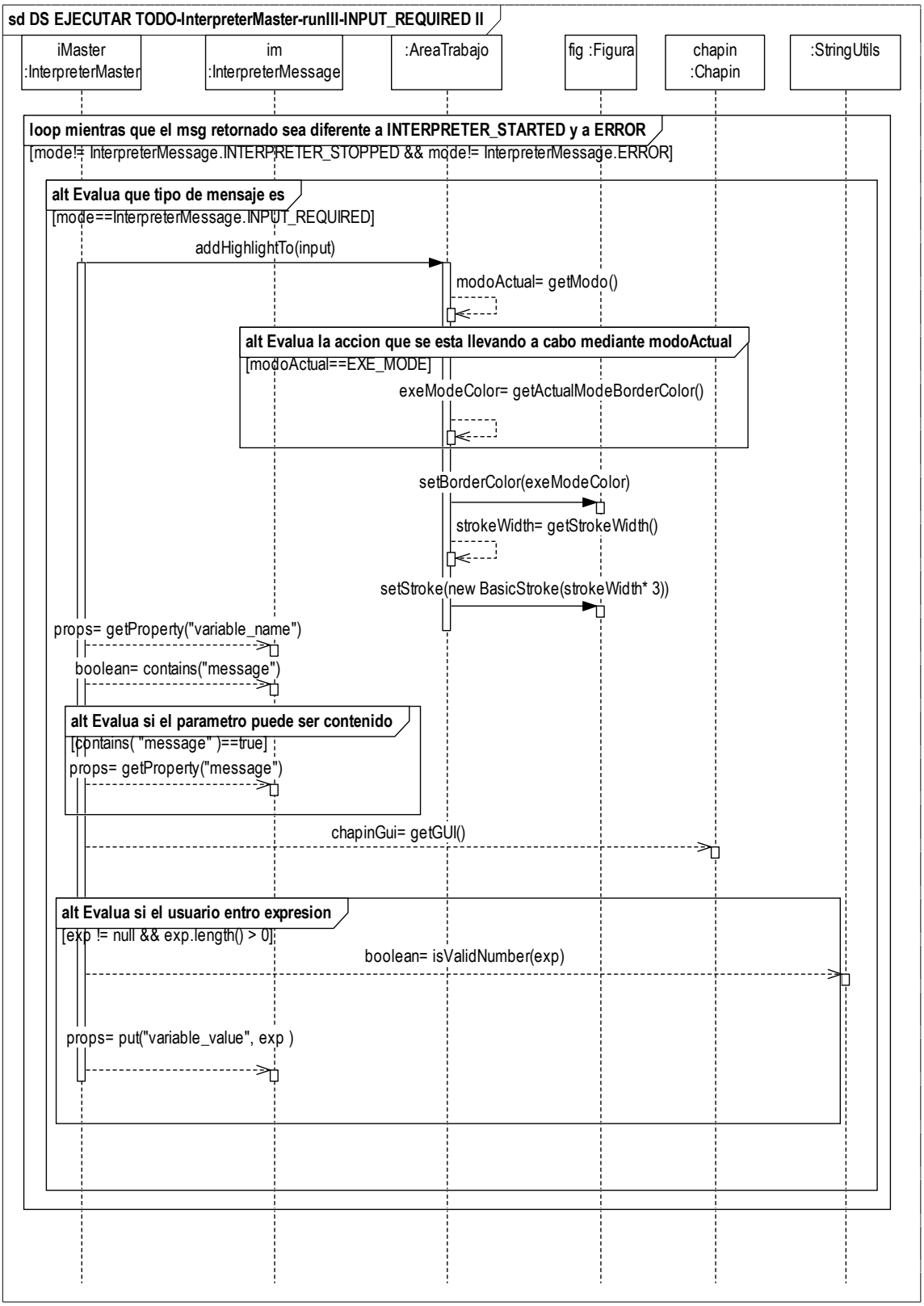


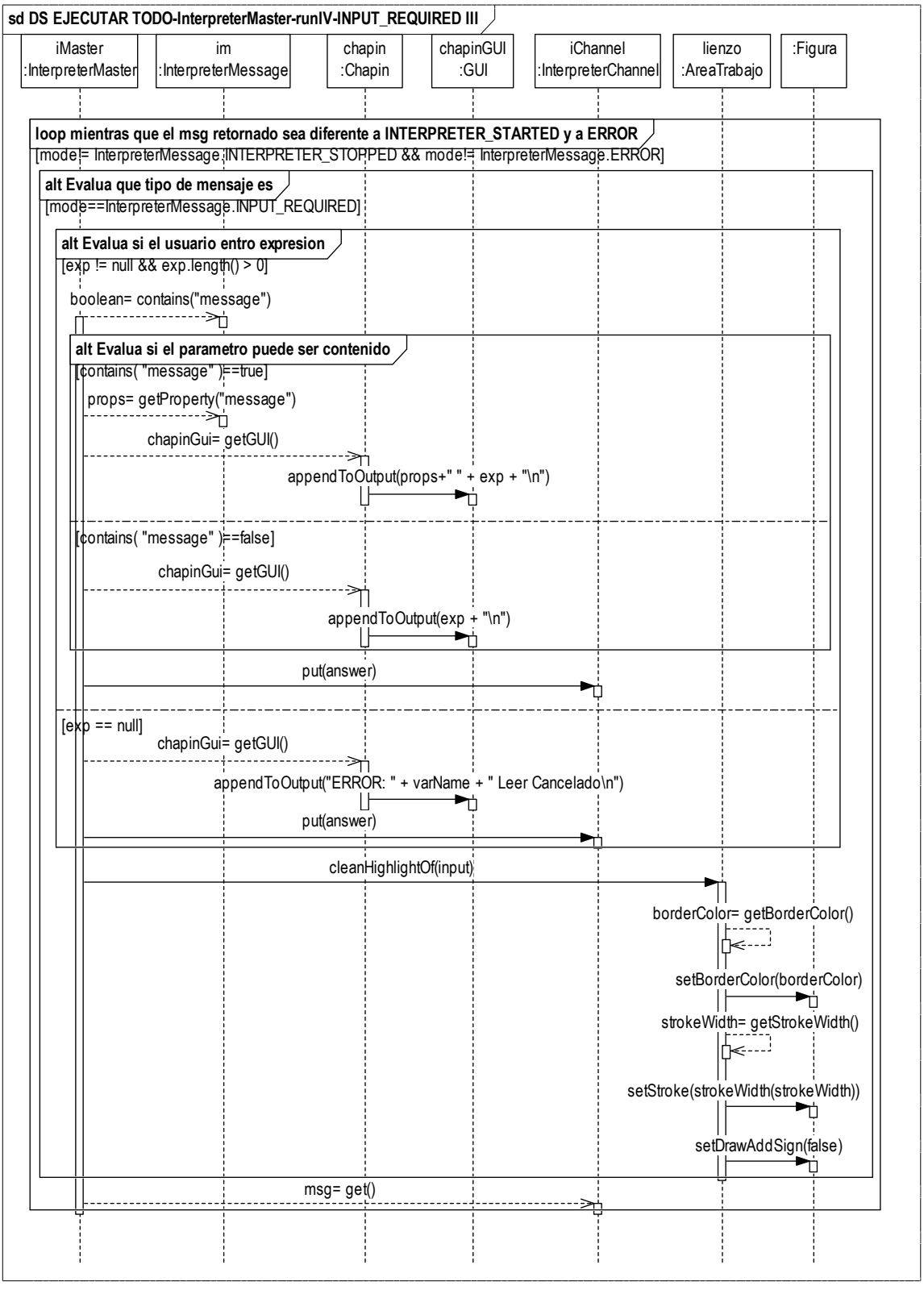

sd DS EJECUTAR TODO-guiEventFired
:ControlEventosGui
ne
:GUIEvent
chapin
:Chapin
chapinGUI
:GUI
iSlave
:InterpreterSlave
iChannel
:InterpreterChannel
iMaster
:InterpreterMaster
mode= getId()
alt Evalua el Modo en el que se esta actualmente
[id==GUIEvent.START]
props= getProperty("mode")
chapinGui= getGUI()
resetOutput()
chapinGui= getGUI()
setConsoleText("")
setMode(mode)
alg= getAlgoritmo()
run()
run()

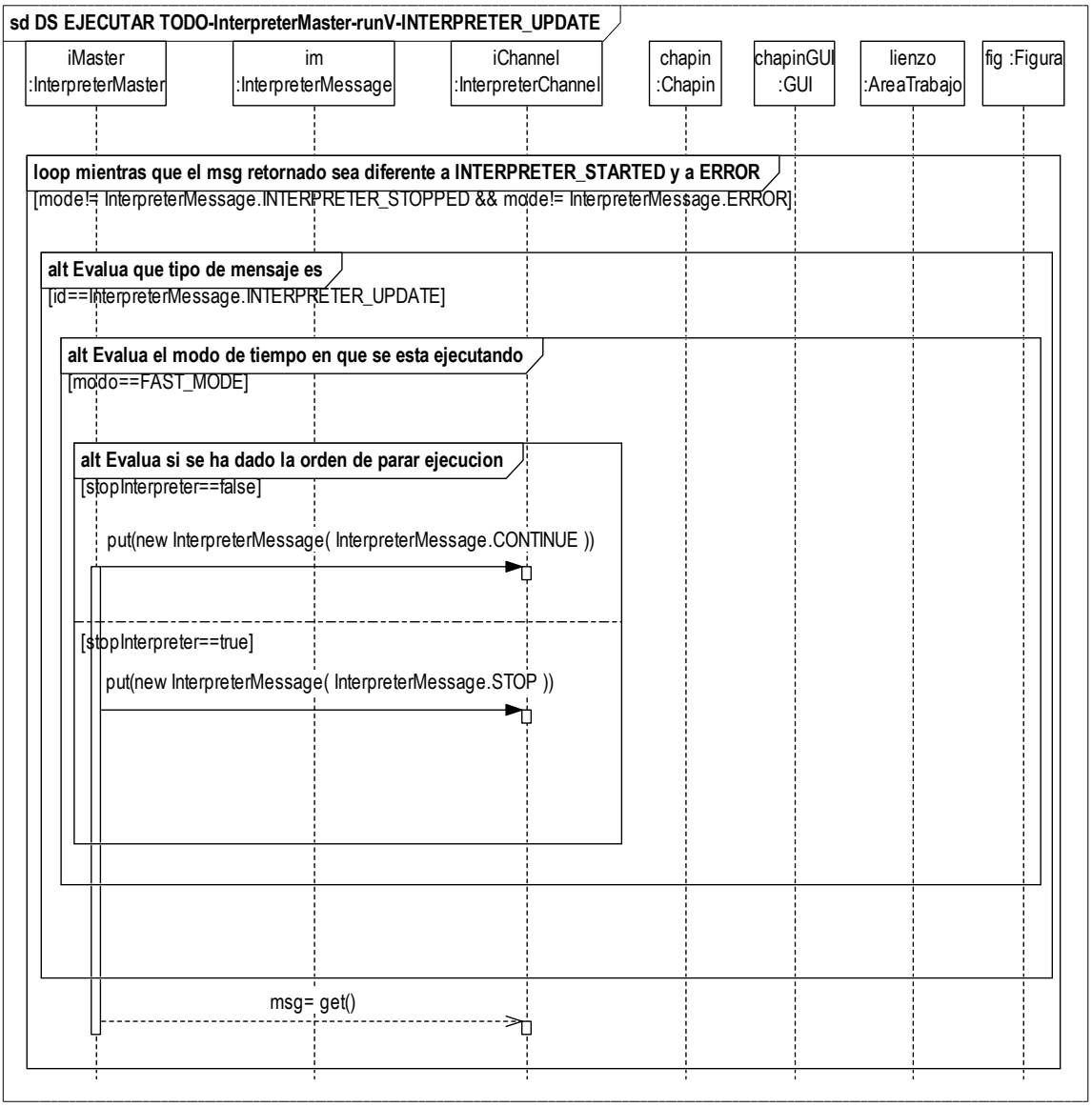
10.10.1 Operación run() de InterpreterMaster()

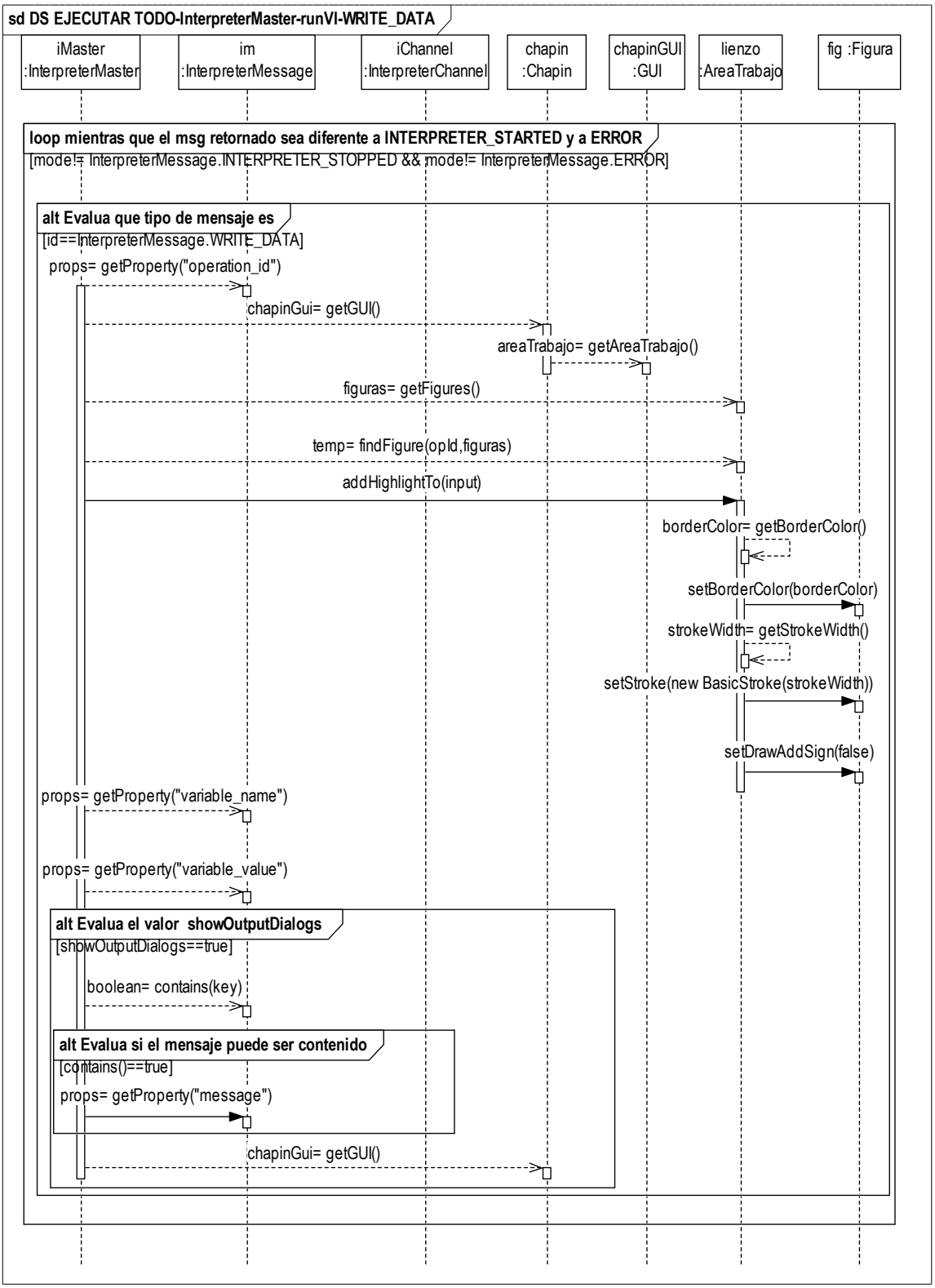


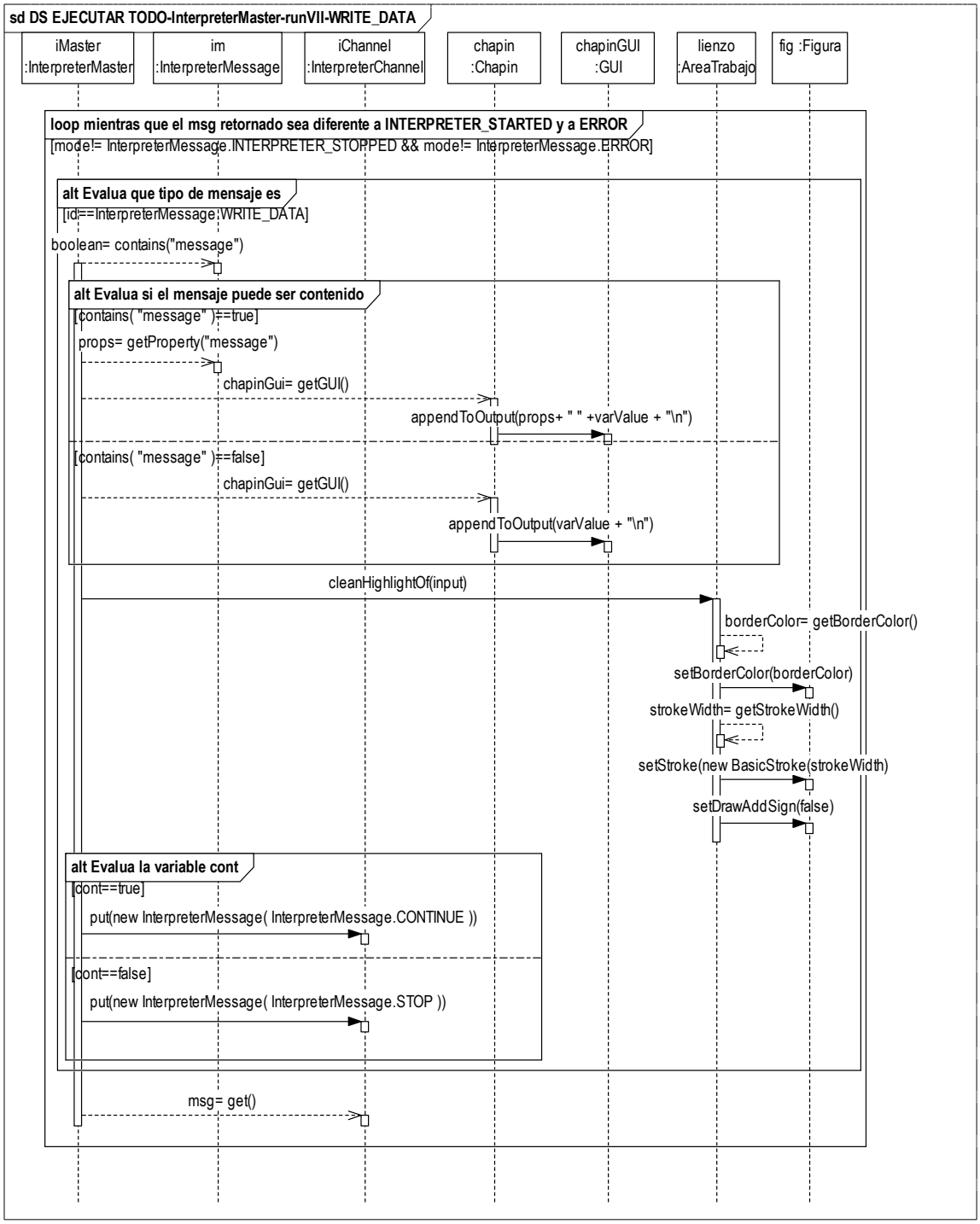

sd DS EJECUTAR TODO-InterpreterMaster-runII-INPUT_REQUIRED
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
temp :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
loop mientras que halla mas elementos en la lista figuras
[iter.hasNext()==true]
id= getId()
alt Compara el id retornado por la figura
[id==id]
temp= findFigure(opId,figuras)
boolean= esHoja()
alt Evalua si no es la primera estructura
[esHoja()==false]
figuras= getFiguras()
temp= findFigure(id,figuras)
alt Evalua si el valor retornado por findFigure() no es nulo
[temp != null]
temp= findFigure(opId,figuras)
[iter.hasNext()==false]
null= findFigure(opId,figuras);

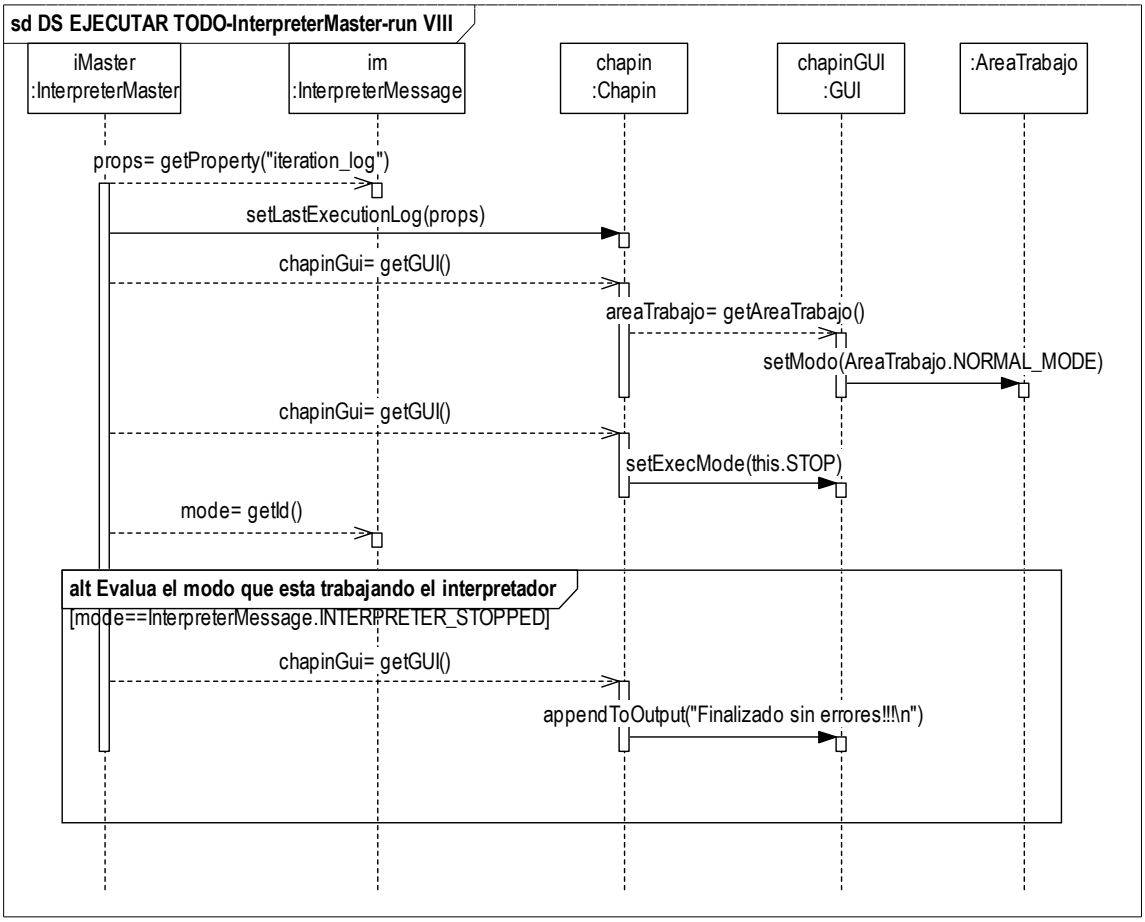

sd DS EJECUTAR TODO-InterpreterMaster-runIII-INPUT_REQUIRED II
iMaster :InterpreterMaster
im :InterpreterMessage
:AreaTrabajo
fig :Figura
chapin :Chapin
:StringUtils
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
addHighlightTo(input)
modoActual= getModo()
alt Evalua la accion que se esta llevando a cabo mediante modoActual
[modoActual==EXE_MODE]
exeModeColor= getActualModeBorderColor()
setBorderColor(exeModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth* 3))
props= getProperty("variable_name")
boolean= contains("message")
alt Evalua si el parametro puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
alt Evalua si el usuario entro expresion
[exp != null && exp.length() > 0]
boolean= isValidNumber(exp)
props= put("variable_value", exp)


sd DS EJECUTAR TODO-InterpreterMaster-runIV-INPUT_REQUIRED III
iMaster :InterpreterMaster
im :InterpreterMessage
chapin :Chapin
chapinGUI :GUI
iChannel :InterpreterChannel
lienzo :AreaTrabajo
:Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
alt Evalua si el usuario entro expresion
[exp != null && exp.length() > 0]
boolean= contains("message")
alt Evalua si el parametro puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
appendToOutput(props+" " + exp + "\n")
[contains("message")==false]
chapinGui= getGUI()
appendToOutput(exp + "\n")
put(answer)
[exp == null]
chapinGui= getGUI()
appendToOutput("ERROR: " + varName + " Leer Cancelado\n")
put(answer)
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(strokeWidth(strokeWidth))
setDrawAddSign(false)
msg= get()

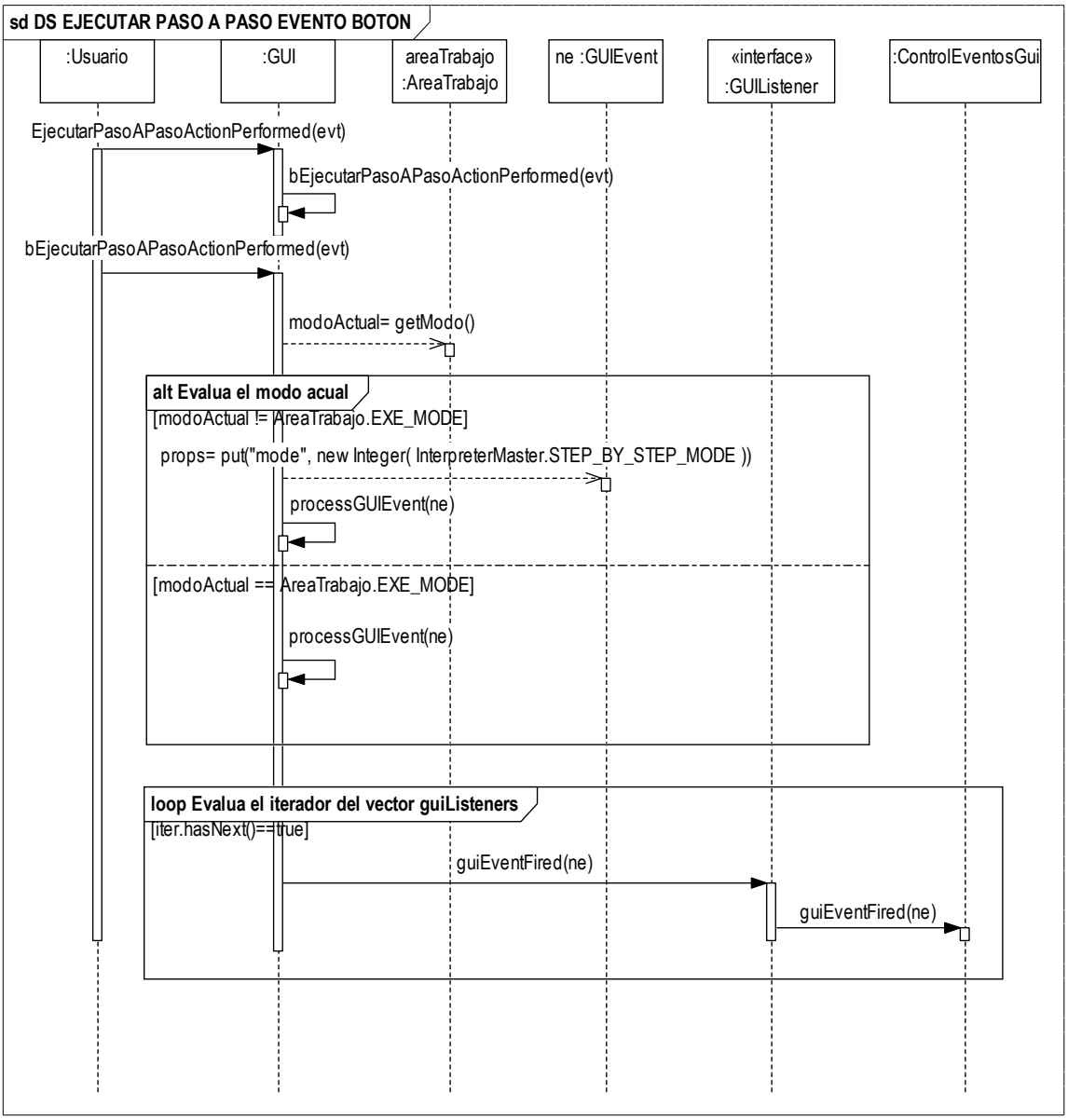

sd DS EJECUTAR TODO-InterpreterMaster-runV-INTERPRETER_UPDATE
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.INTERPRETER_UPDATE]
alt Evalua el modo de tiempo en que se esta ejecutando
[modo==FAST_MODE]
alt Evalua si se ha dado la orden de parar ejecucion
[stopInterpreter==false]
put(new InterpreterMessage(InterpreterMessage.CONTINUE))
[stopInterpreter==true]
put(new InterpreterMessage(InterpreterMessage.STOP))
msg= get()

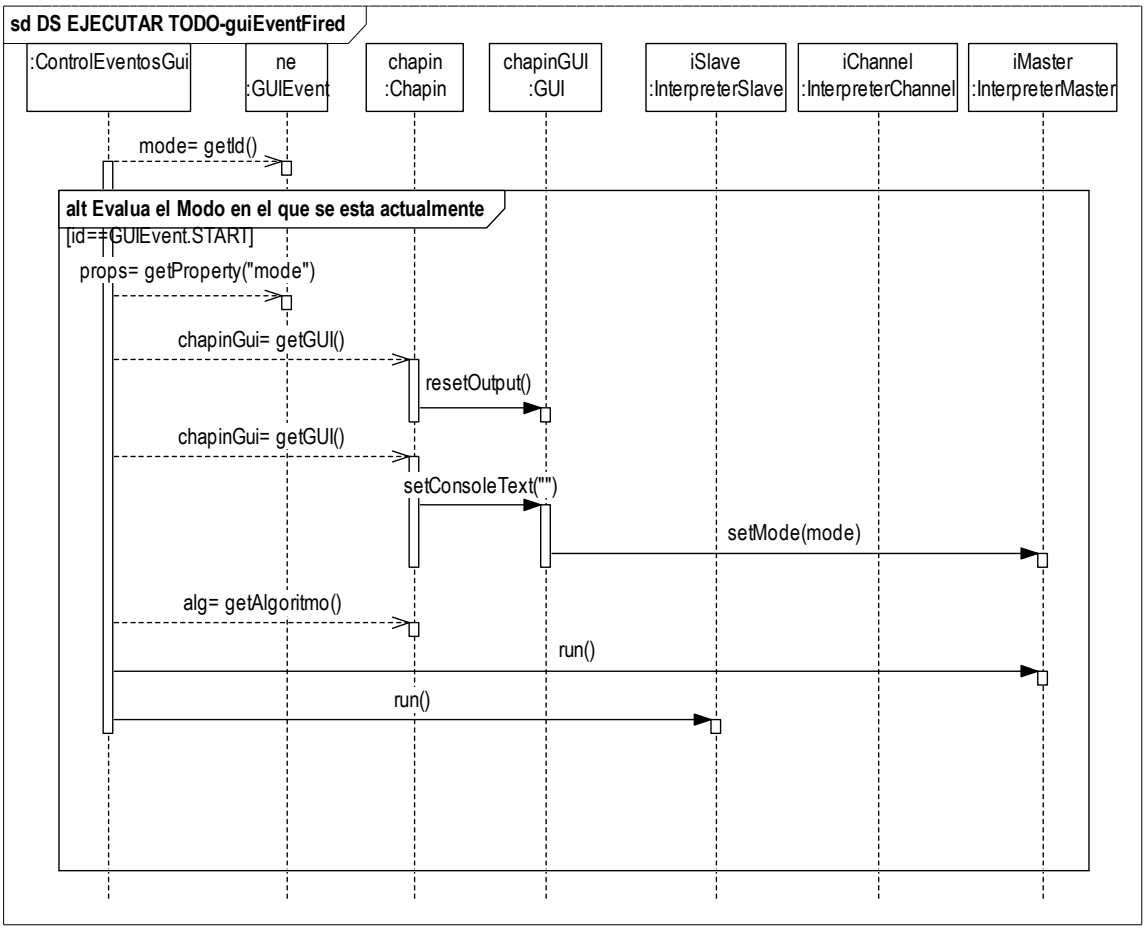

sd DS EJECUTAR TODO-InterpreterMaster-runVI-WRITE_DATA
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.WRITE_DATA]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
temp= findFigure(opId,figuras)
addHighlightTo(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
props= getProperty("variable_name")
props= getProperty("variable_value")
alt Evalua el valor showOutputDialogs
[showOutputDialogs==true]
boolean= contains(key)
alt Evalua si el mensaje puede ser contenido
[contains()==true]
props= getProperty("message")
chapinGui= getGUI()

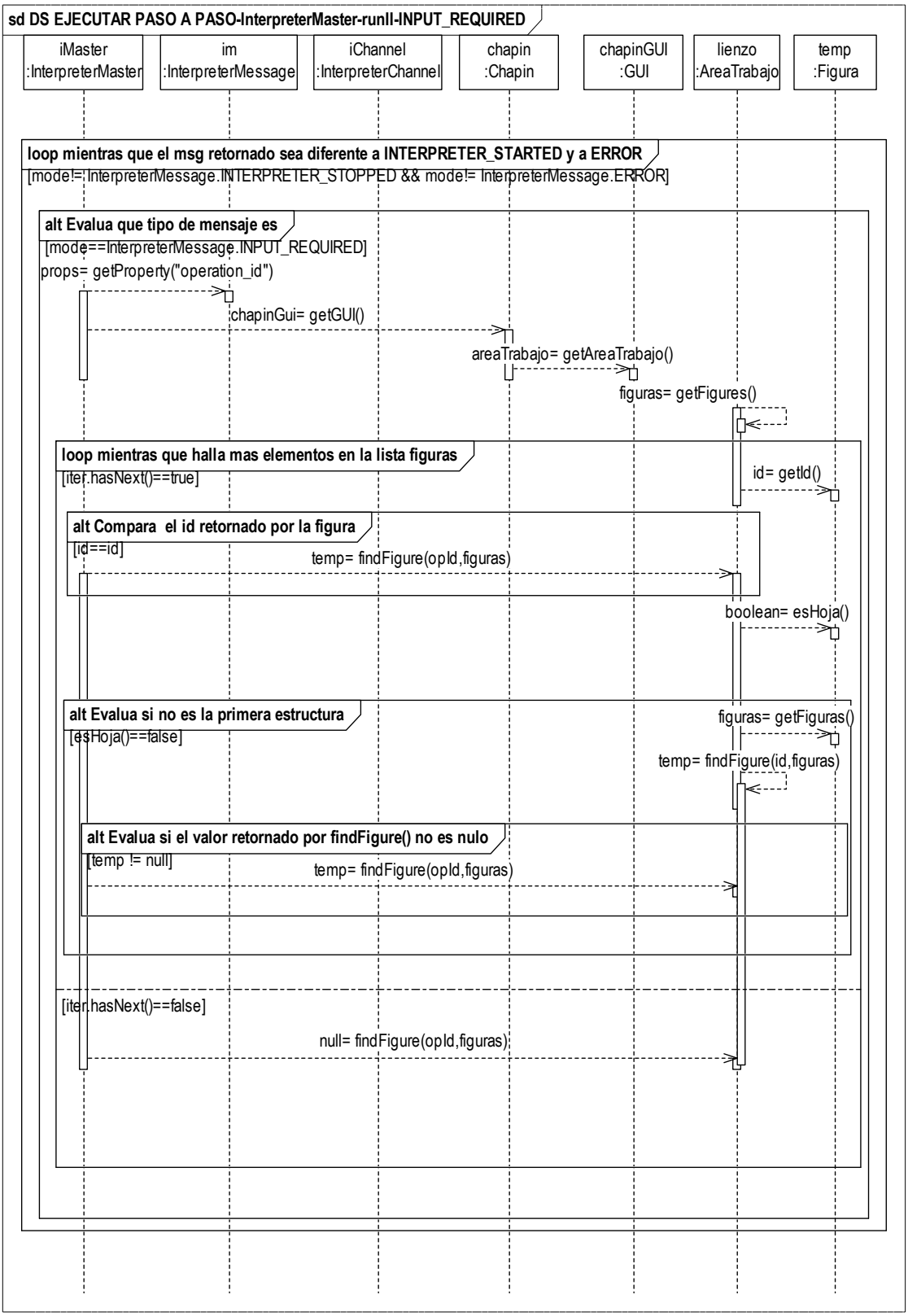

sd DS EJECUTAR TODO-InterpreterMaster-runVII-WRITE_DATA
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.WRITE_DATA]
boolean= contains("message")
alt Evalua si el mensaje puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
appendToOutput(props+ " " +varValue + "\n")
[contains("message")==false]
chapinGui= getGUI()
appendToOutput(varValue + "\n")
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth)
setDrawAddSign(false)
alt Evalua la variable cont
[cont==true]
put(new InterpreterMessage(InterpreterMessage.CONTINUE))
[cont==false]
put(new InterpreterMessage(InterpreterMessage.STOP))
msg= get()

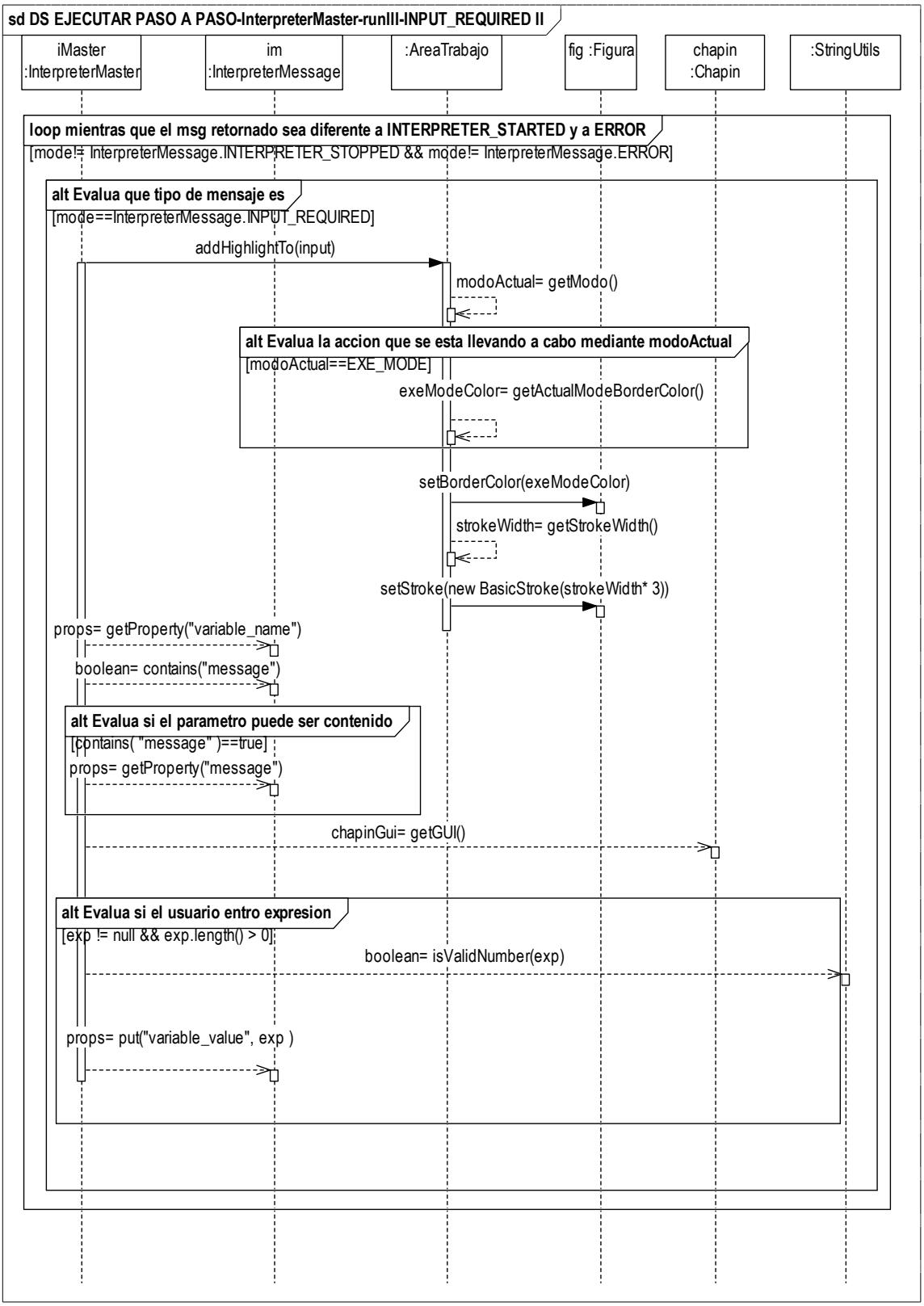

sd DS EJECUTAR TODO-InterpreterMaster-run VIII
iMaster
:InterpreterMaster
im
:InterpreterMessage
chapin
:Chapin
chapinGUI
:GUI
:AreaTrabajo
props= getProperty("iteration_log")
setLastExecutionLog(props)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setModo(AreaTrabajo.NORMAL_MODE)
chapinGui= getGUI()
setExecMode(this.STOP)
mode= getId()
alt Evalua el modo que esta trabajando el interpretador
[mode==InterpreterMessage.INTERPRETER_STOPPED]
chapinGui= getGUI()
appendToOutput("Finalizado sin errores!!!\n")

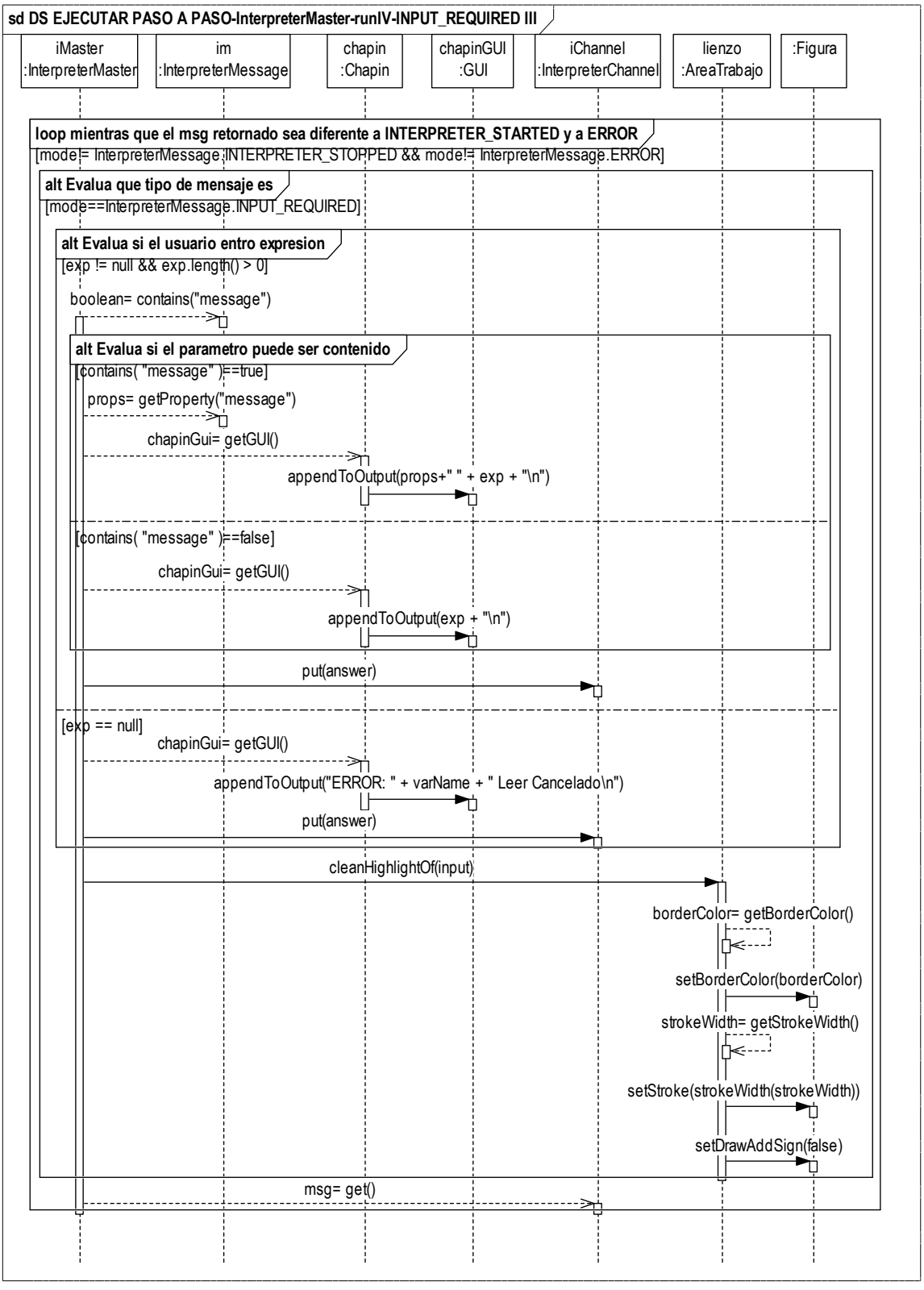
10.11 DIAGRAMA DE SECUENCIA DE EJECUTAR PASO A PASO

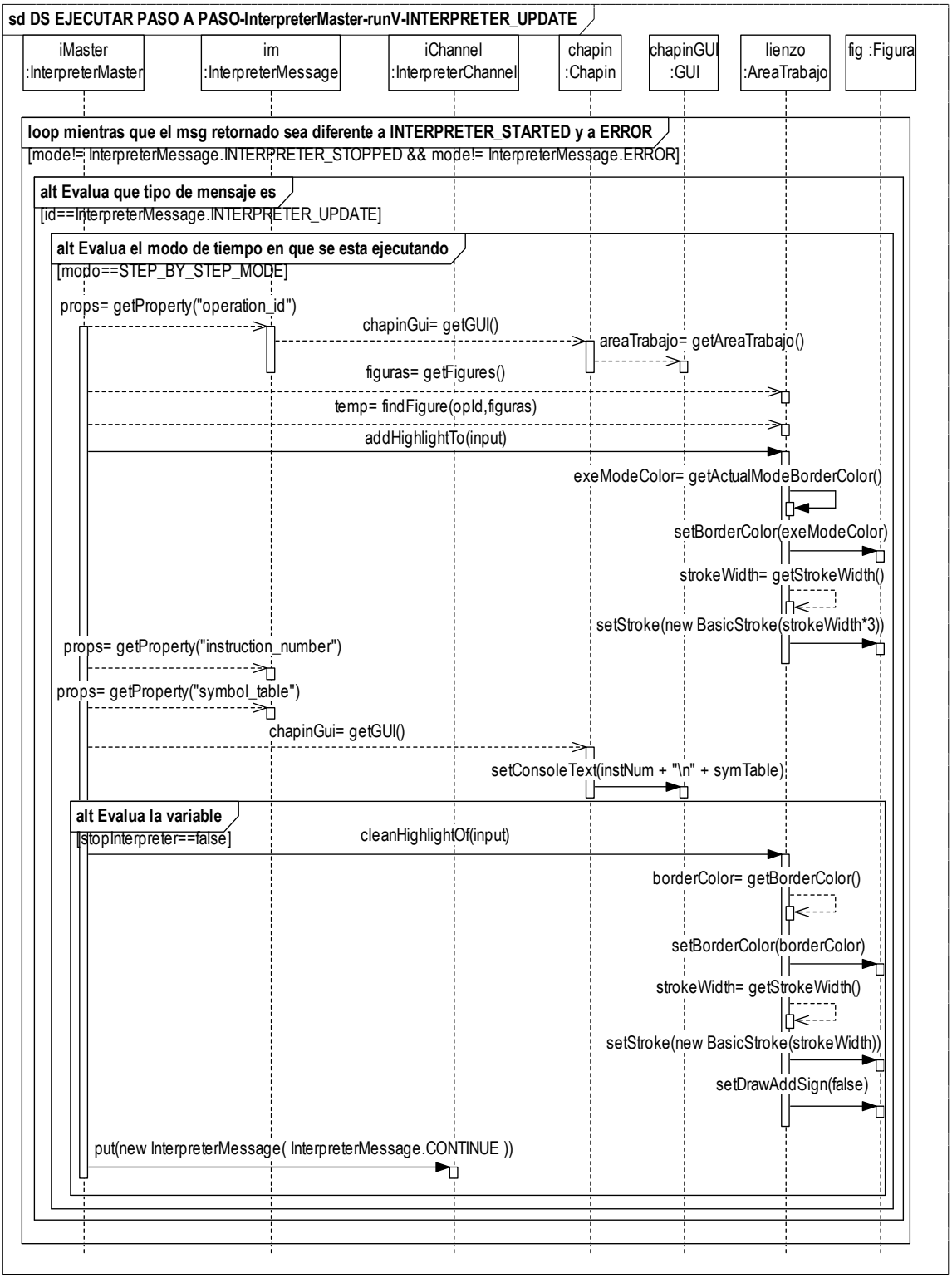


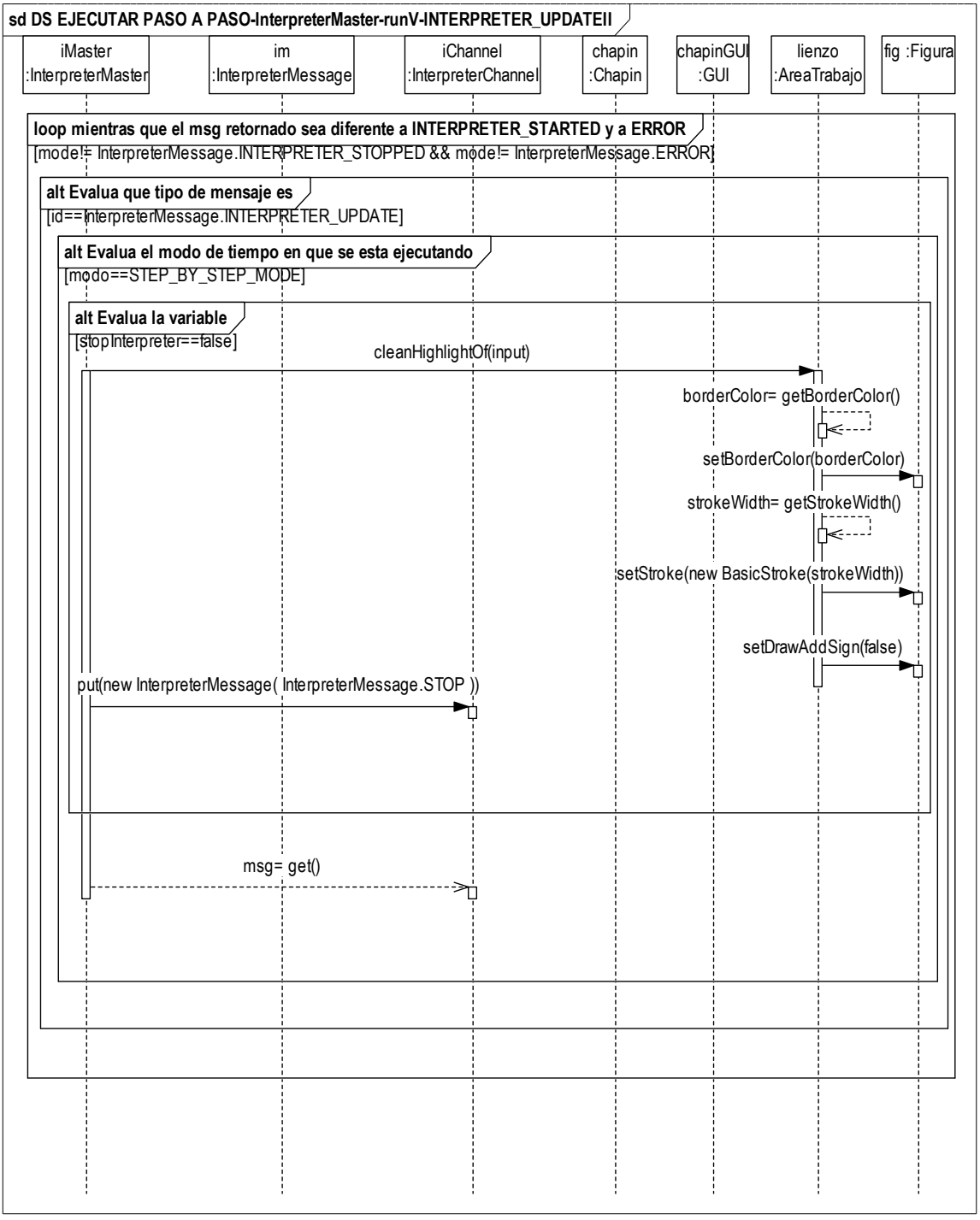

sd DS EJECUTAR TODO-guiEventFired
:ControlEventosGui
ne
:GUIEvent
chapin
:Chapin
chapinGUI
:GUI
iSlave
:InterpreterSlave
iChannel
:InterpreterChannel
iMaster
:InterpreterMaster
mode= getId()
alt Evalua el Modo en el que se esta actualmente
[id==GUIEvent.START]
props= getProperty("mode")
chapinGui= getGUI()
resetOutput()
chapinGui= getGUI()
setConsoleText("")
setMode(mode)
alg= getAlgoritmo()
run()
run()

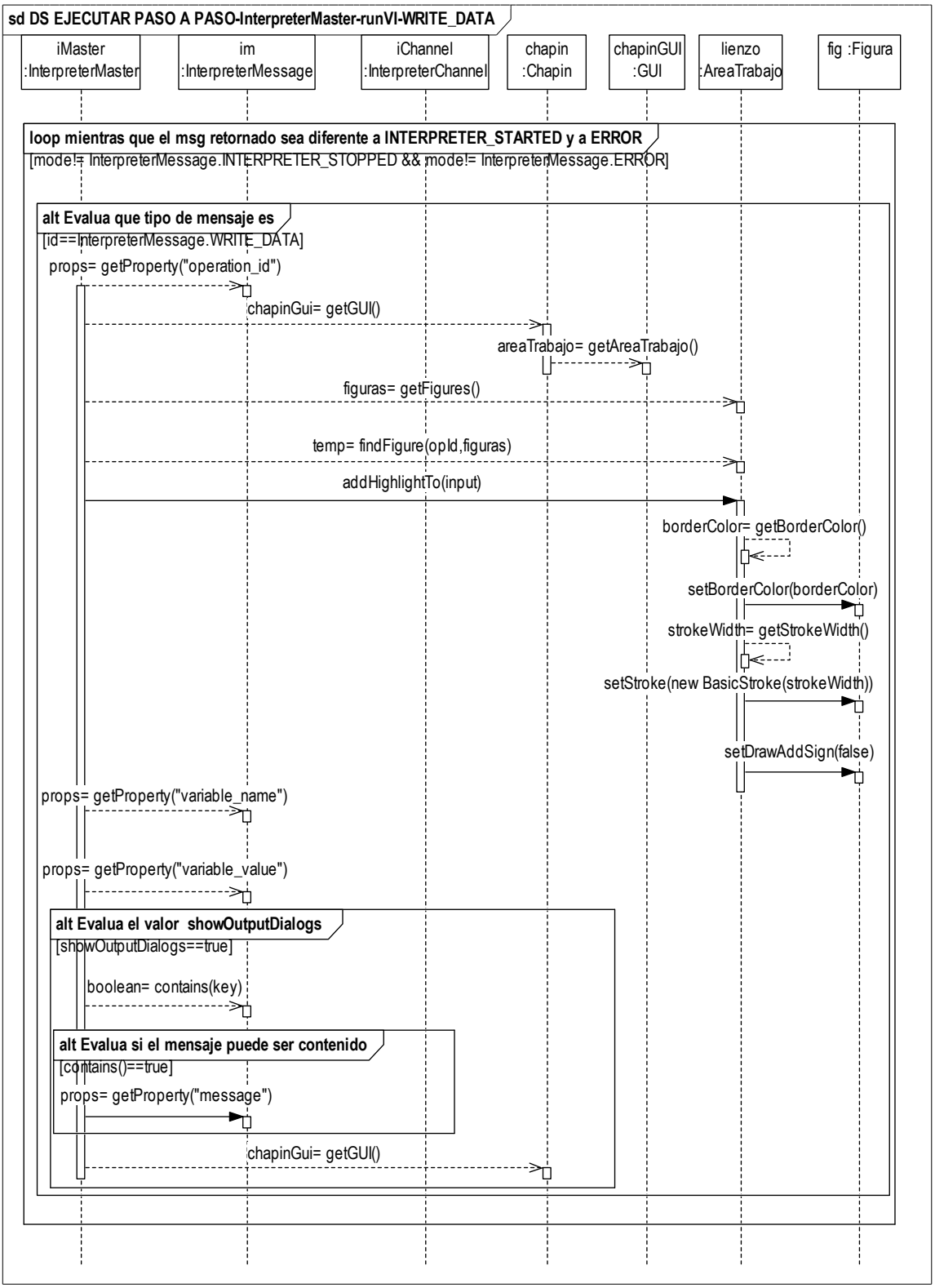

sd DS EJECUTAR PASO A PASO-InterpreterMaster-runII-INPUT_REQUIRED
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
temp :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
loop mientras que halla mas elementos en la lista figuras
[iter.hasNext()==true]
id= getId()
alt Compara el id retornado por la figura
[id==id]
temp= findFigure(opId,figuras)
boolean= esHoja()
alt Evalua si no es la primera estructura
[esHoja()==false]
figuras= getFiguras()
temp= findFigure(id,figuras)
alt Evalua si el valor retornado por findFigure() no es nulo
[temp != null]
temp= findFigure(opId,figuras)
[iter.hasNext()==false]
null= findFigure(opId,figuras);

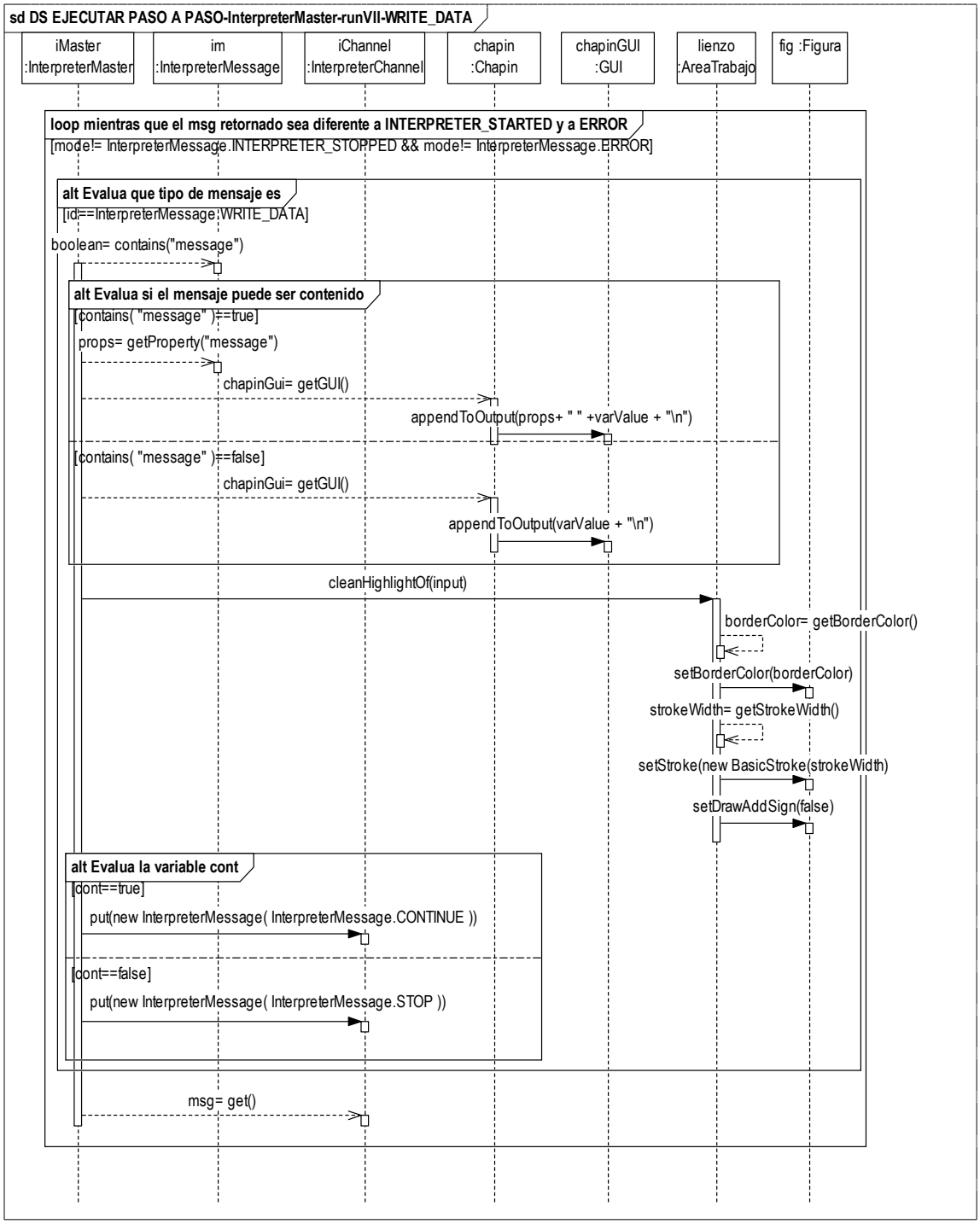

sd DS EJECUTAR PASO A PASO-InterpreterMaster-runIII-INPUT_REQUIRED II
iMaster :InterpreterMaster
im :InterpreterMessage
:AreaTrabajo
fig :Figura
chapin :Chapin
:StringUtils
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
addHighlightTo(input)
modoActual= getModo()
alt Evalua la accion que se esta llevando a cabo mediante modoActual
[modoActual==EXE_MODE]
exeModeColor= getActualModeBorderColor()
setBorderColor(exeModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth* 3))
props= getProperty("variable_name")
boolean= contains("message")
alt Evalua si el parametro puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
alt Evalua si el usuario entro expresion
[exp != null && exp.length() > 0]
boolean= isValidNumber(exp)
props= put("variable_value", exp)

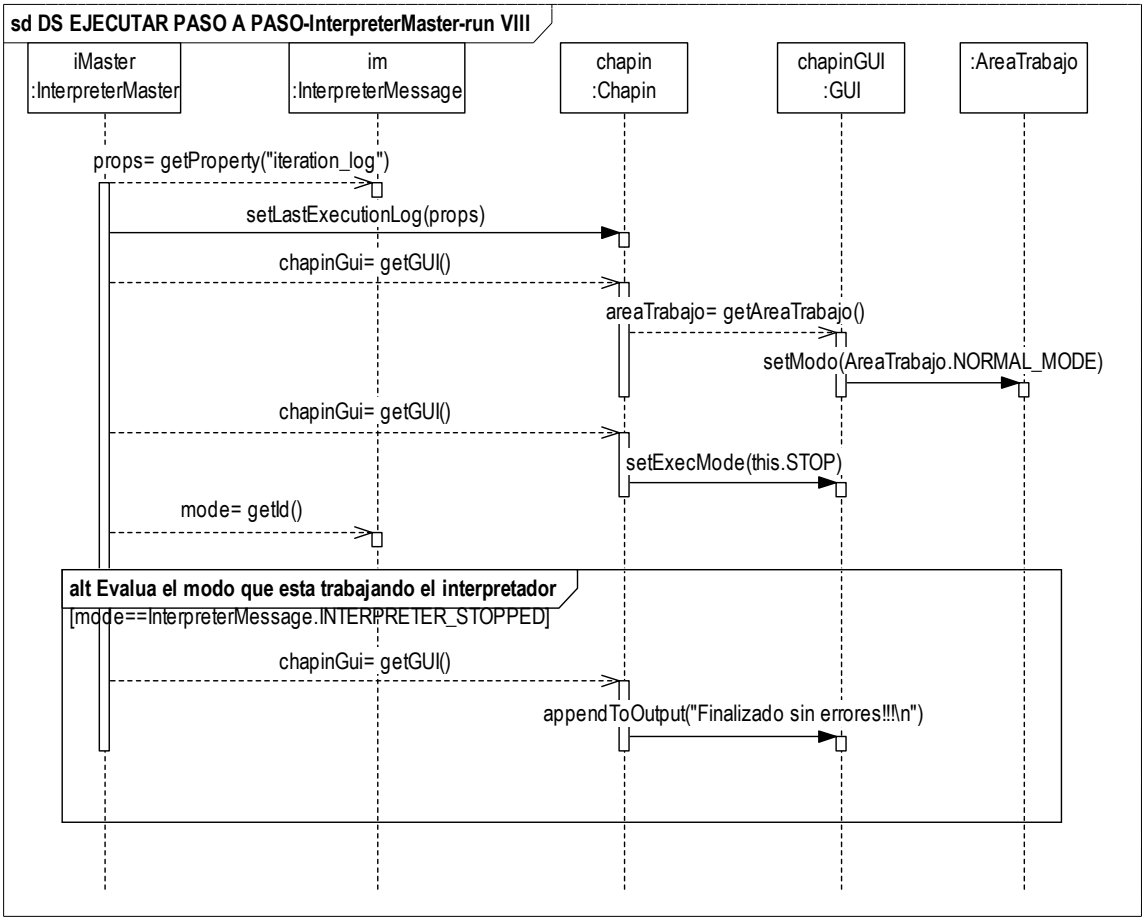

sd DS EJECUTAR PASO A PASO-InterpreterMaster-runIV-INPUT_REQUIRED III
iMaster :InterpreterMaster
im :InterpreterMessage
chapin :Chapin
chapinGUI :GUI
iChannel :InterpreterChannel
lienzo :AreaTrabajo
:Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
alt Evalua si el usuario entro expresion
[exp != null && exp.length() > 0]
boolean= contains("message")
alt Evalua si el parametro puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
appendToOutput(props+" " + exp + "\n")
[contains("message")==false]
chapinGui= getGUI()
appendToOutput(exp + "\n")
put(answer)
[exp == null]
chapinGui= getGUI()
appendToOutput("ERROR: " + varName + " Leer Cancelado\n")
put(answer)
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(strokeWidth(strokeWidth))
setDrawAddSign(false)
msg= get()


sd DS EJECUTAR PASO A PASO-InterpreterMaster-runV-INTERPRETER_UPDATE
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.INTERPRETER_UPDATE]
alt Evalua el modo de tiempo en que se esta ejecutando
[modo==STEP_BY_STEP_MODE]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
temp= findFigure(opId,figuras)
addHighlightTo(input)
exeModeColor= getActualModeBorderColor()
setBorderColor(exeModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth*3))
props= getProperty("instruction_number")
props= getProperty("symbol_table")
chapinGui= getGUI()
setConsoleText(instNum + "\n" + symTable)
alt Evalua la variable
[stopInterpreter==false]
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
put(new InterpreterMessage(InterpreterMessage.CONTINUE))

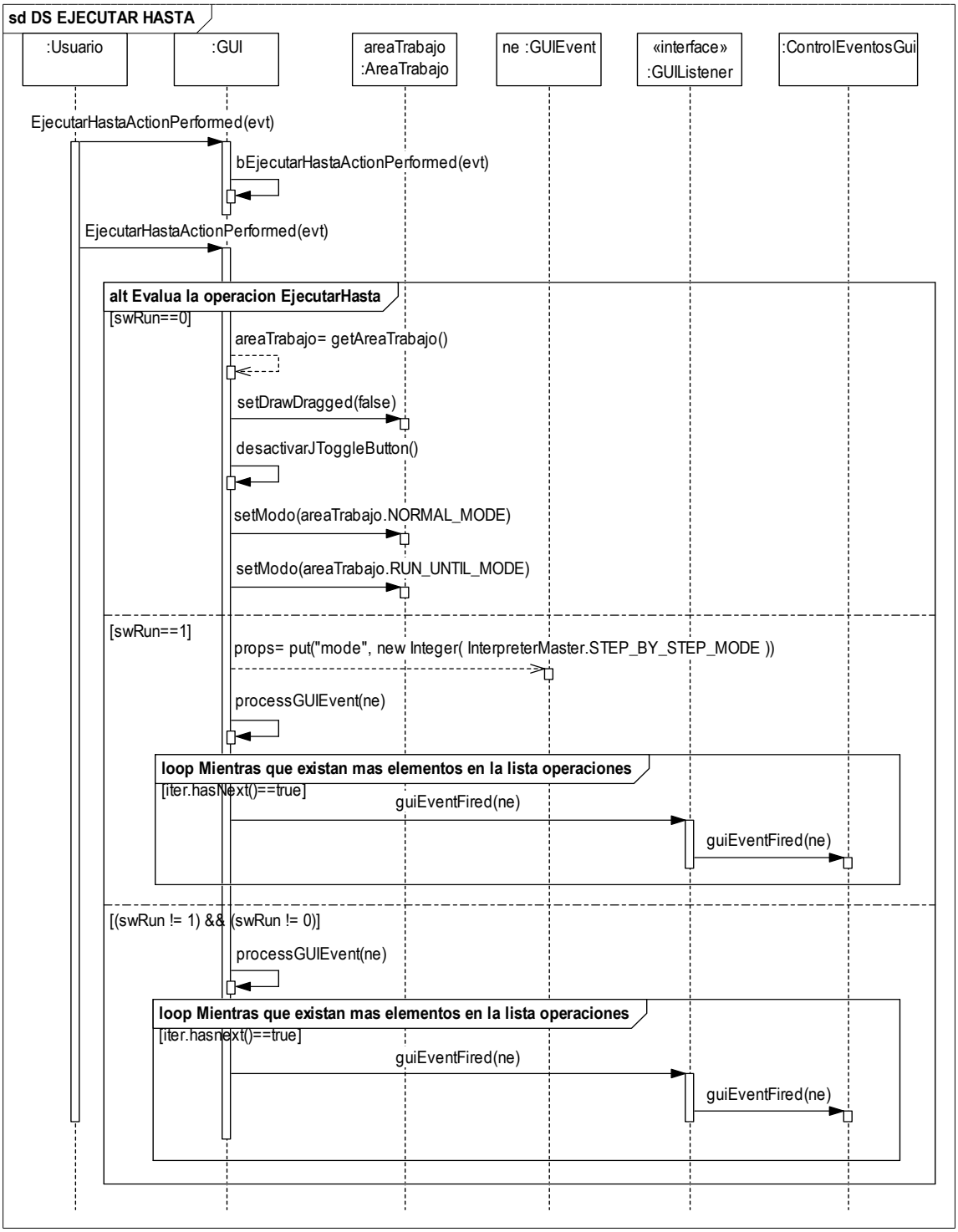

sd DS EJECUTAR PASO A PASO-InterpreterMaster-runV-INTERPRETER_UPDATEII
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.INTERPRETER_UPDATE]
alt Evalua el modo de tiempo en que se esta ejecutando
[modo==STEP_BY_STEP_MODE]
alt Evalua la variable
[stopInterpreter==false]
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
put(new InterpreterMessage(InterpreterMessage.STOP))
msg= get()

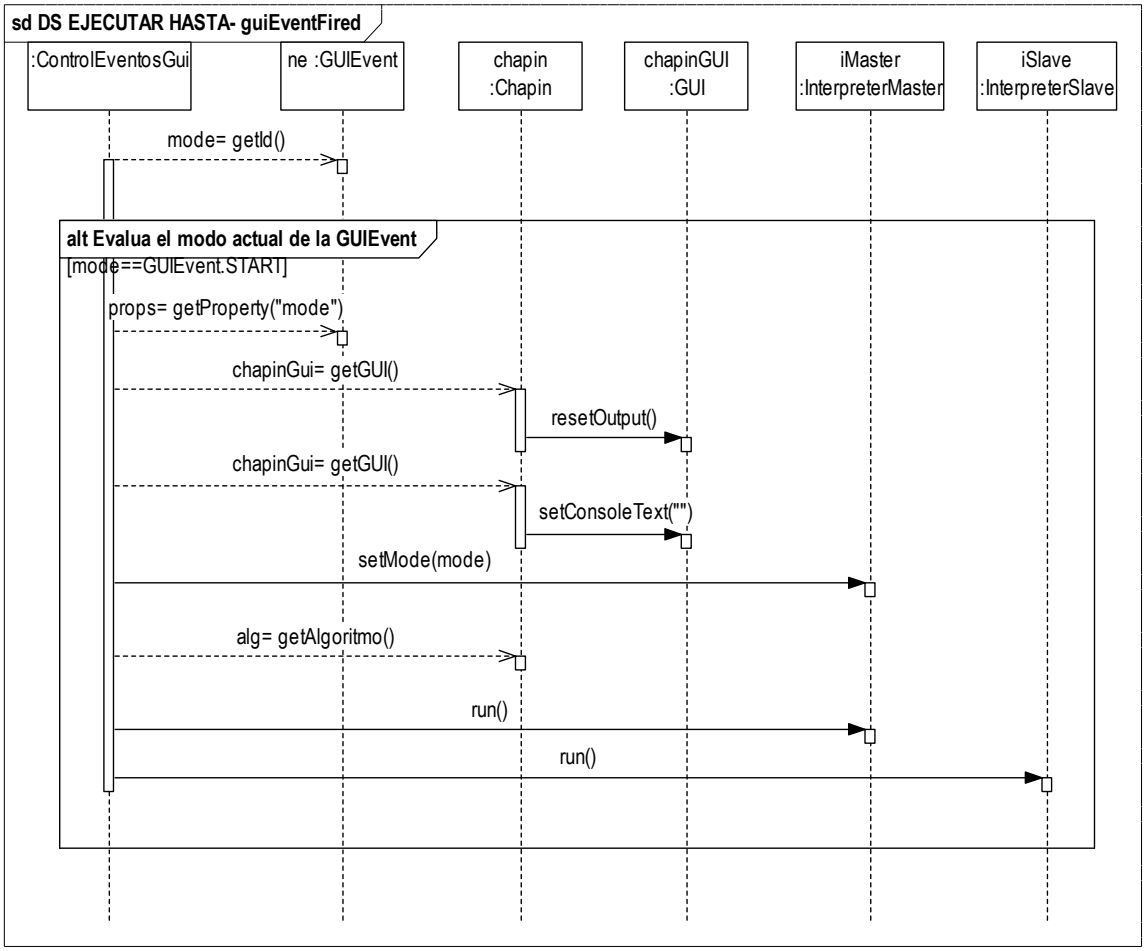

sd DS EJECUTAR PASO A PASO-InterpreterMaster-runVI-WRITE_DATA
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.WRITE_DATA]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
temp= findFigure(opId,figuras)
addHighlightTo(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
props= getProperty("variable_name")
props= getProperty("variable_value")
alt Evalua el valor showOutputDialogs
[showOutputDialogs==true]
boolean= contains(key)
alt Evalua si el mensaje puede ser contenido
[contains()==true]
props= getProperty("message")
chapinGui= getGUI()

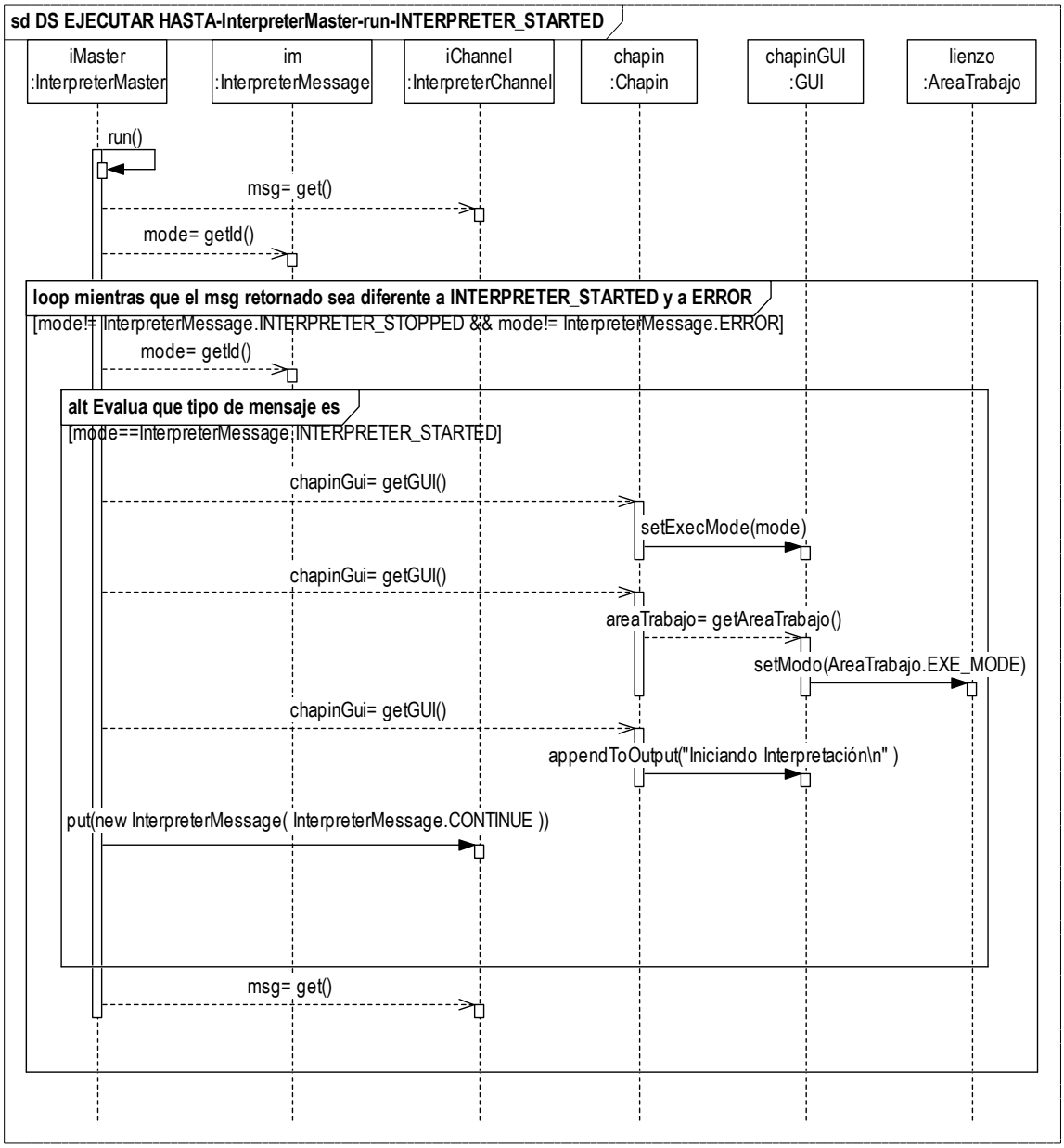

sd DS EJECUTAR PASO A PASO-InterpreterMaster-runVII-WRITE_DATA
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.WRITE_DATA]
boolean= contains("message")
alt Evalua si el mensaje puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
appendToOutput(props+ " " +varValue + "\n")
[contains("message")==false]
chapinGui= getGUI()
appendToOutput(varValue + "\n")
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth)
setDrawAddSign(false)
alt Evalua la variable cont
[cont==true]
put(new InterpreterMessage(InterpreterMessage.CONTINUE))
[cont==false]
put(new InterpreterMessage(InterpreterMessage.STOP))
msg= get()

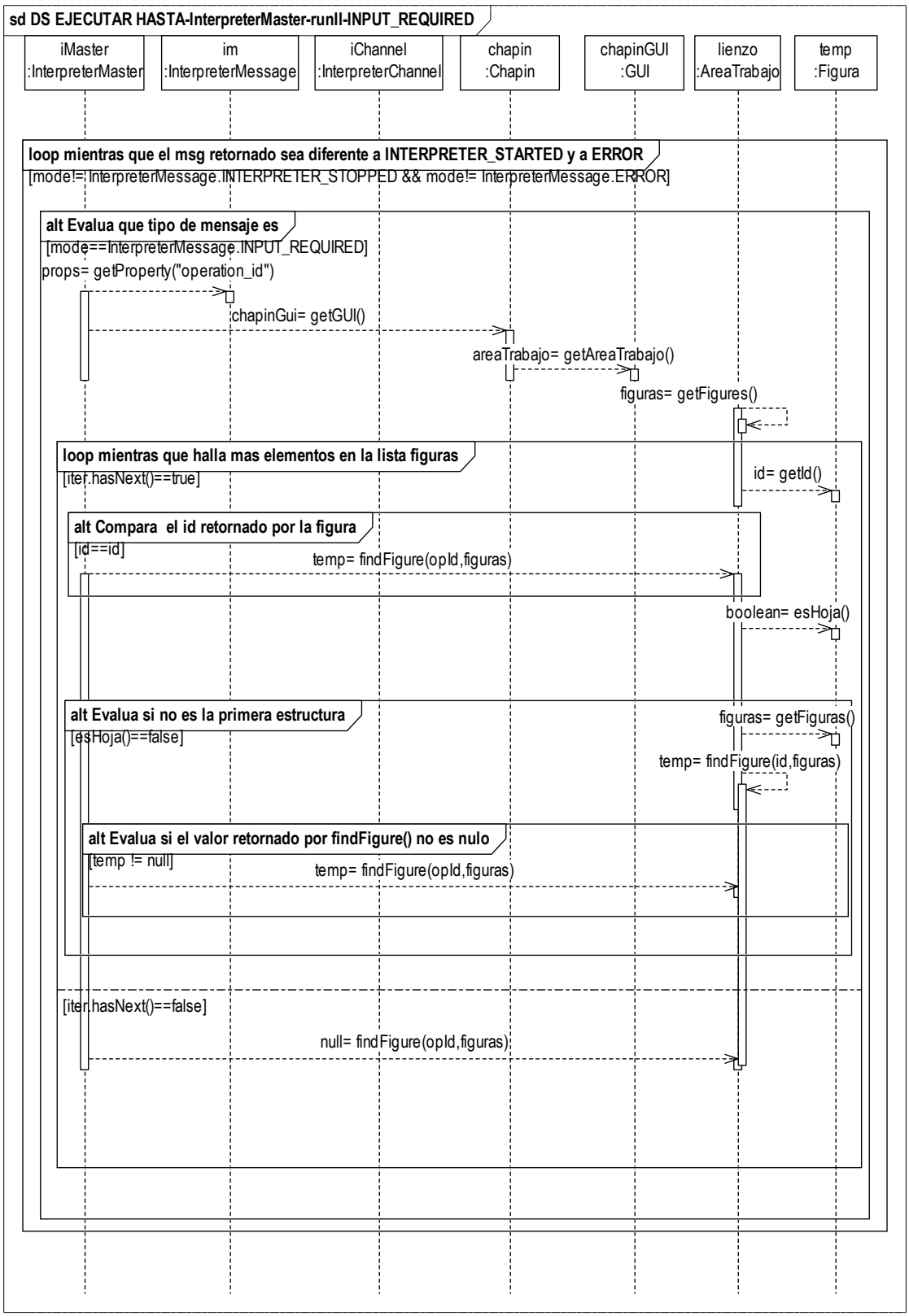

sd DS EJECUTAR PASO A PASO-InterpreterMaster-run VIII
iMaster
:InterpreterMaster
im
:InterpreterMessage
chapin
:Chapin
chapinGUI
:GUI
:AreaTrabajo
props= getProperty("iteration_log")
setLastExecutionLog(props)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setModo(AreaTrabajo.NORMAL_MODE)
chapinGui= getGUI()
setExecMode(this.STOP)
mode= getId()
alt Evalua el modo que esta trabajando el interpretador
[mode==InterpreterMessage.INTERPRETER_STOPPED]
chapinGui= getGUI()
appendToOutput("Finalizado sin errores!!!\n")

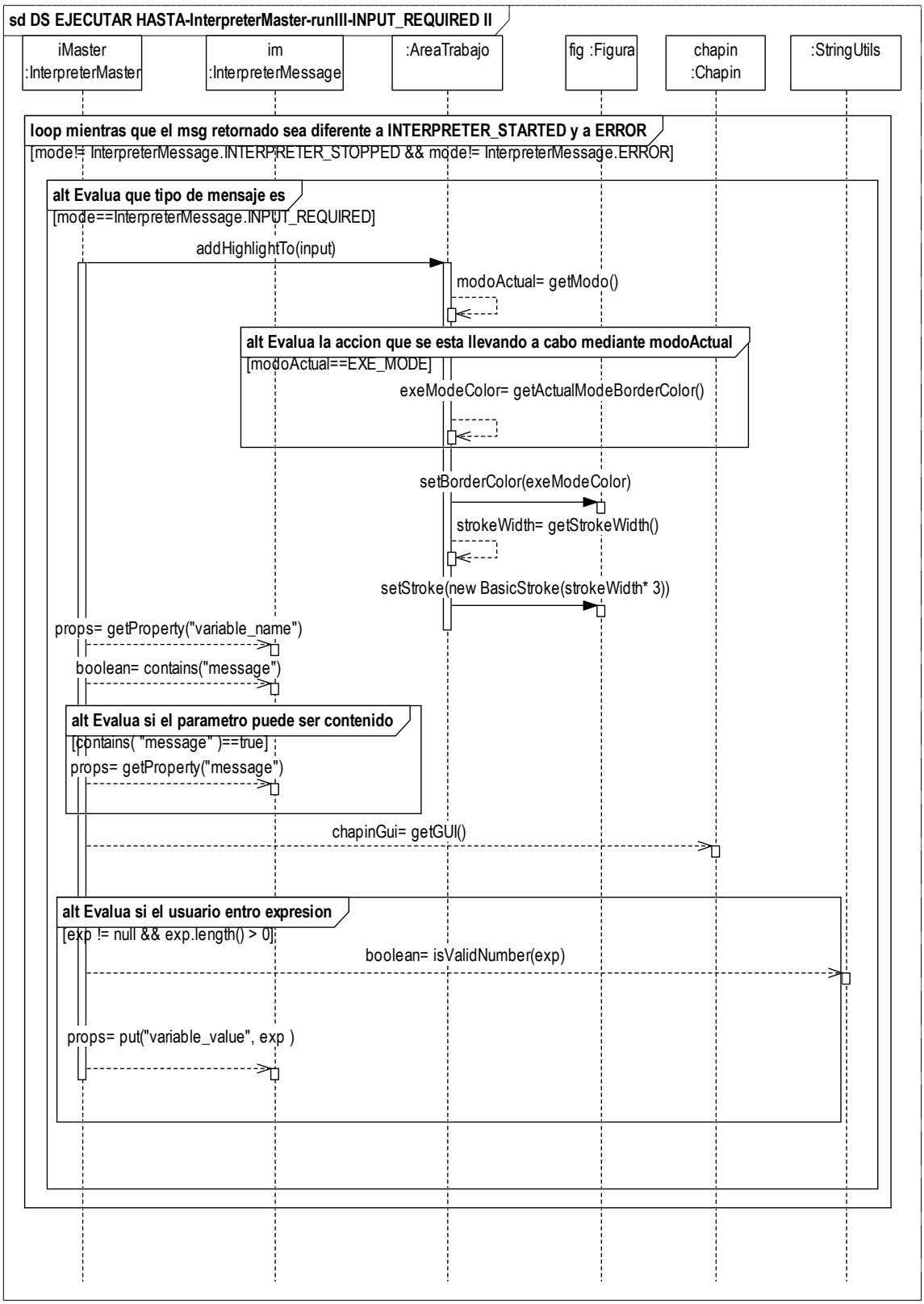
10.12 DIAGRAMAS DE SECUENCIA DE EJECUTAR HASTA

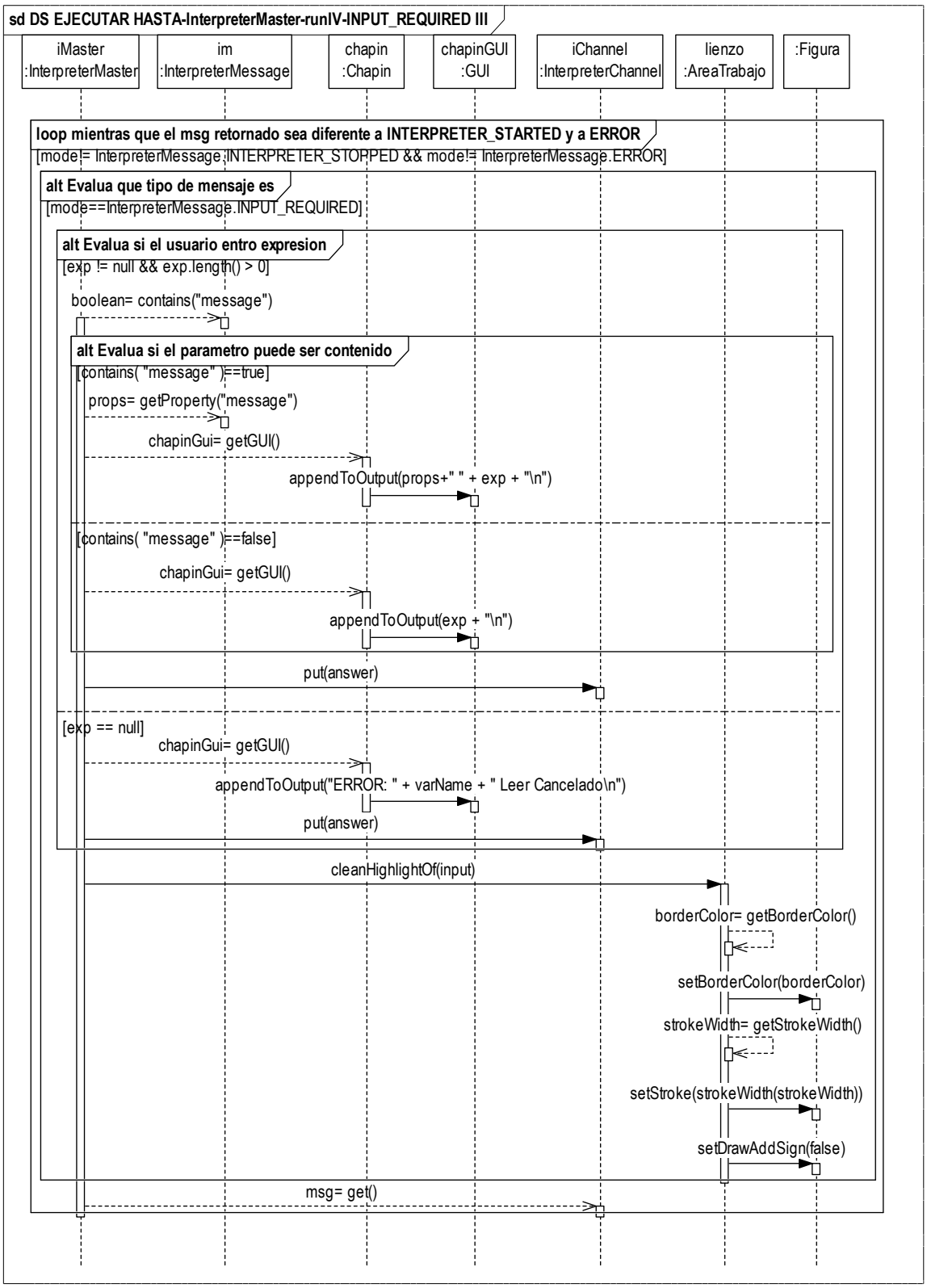


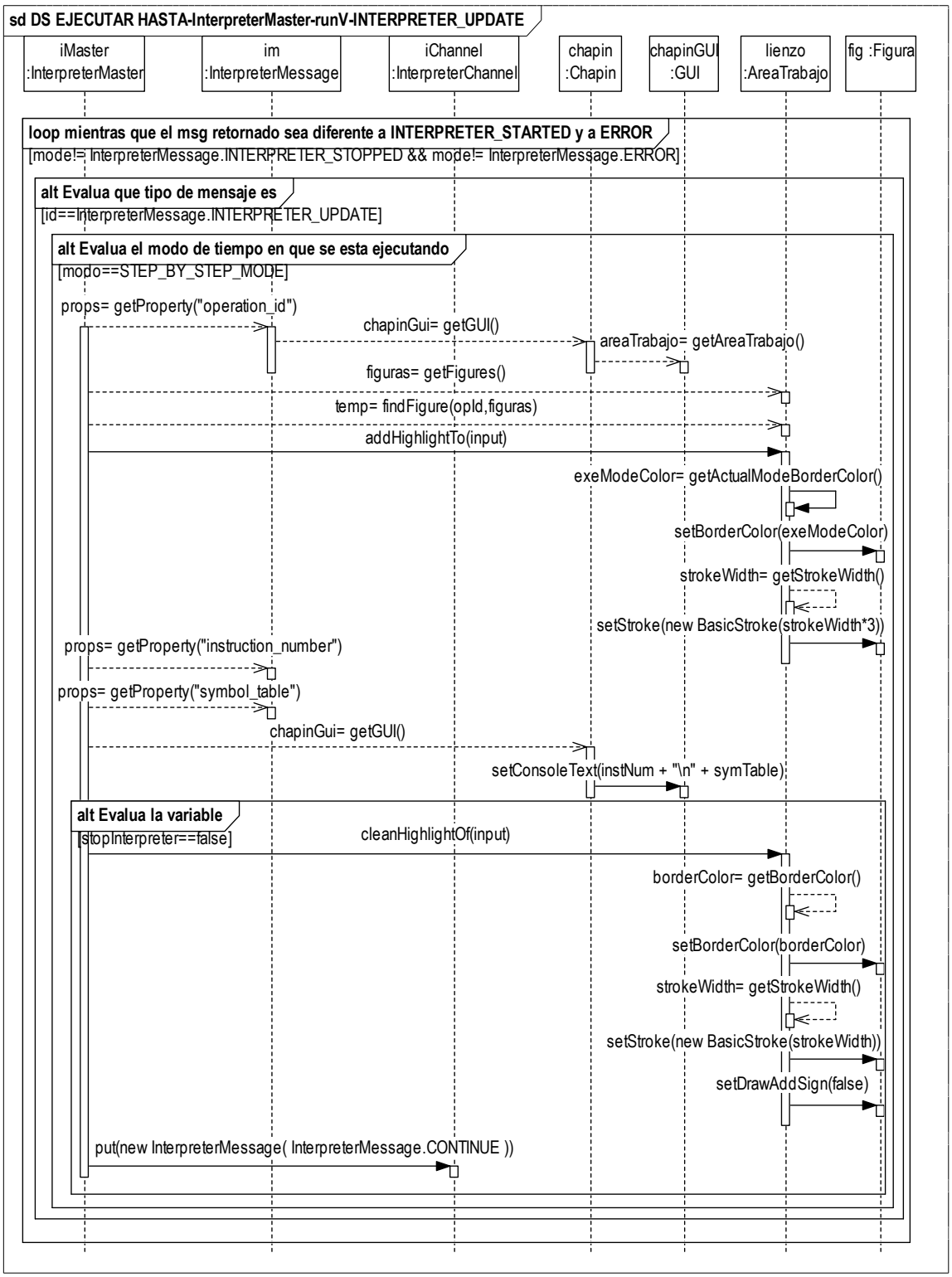

sd DS EJECUTAR HASTA- guiEventFired
:ControlEventosGui
ne :GUIEvent
chapin
:Chapin
chapinGUI
:GUI
iMaster
:InterpreterMaster
iSlave
:InterpreterSlave
mode= getId()
alt Evalua el modo actual de la GUIEvent
[mode==GUIEvent.START]
props= getProperty("mode")
chapinGui= getGUI()
resetOutput()
chapinGui= getGUI()
setConsoleText("")
setMode(mode)
alg= getAlgoritmo()
run()
run()

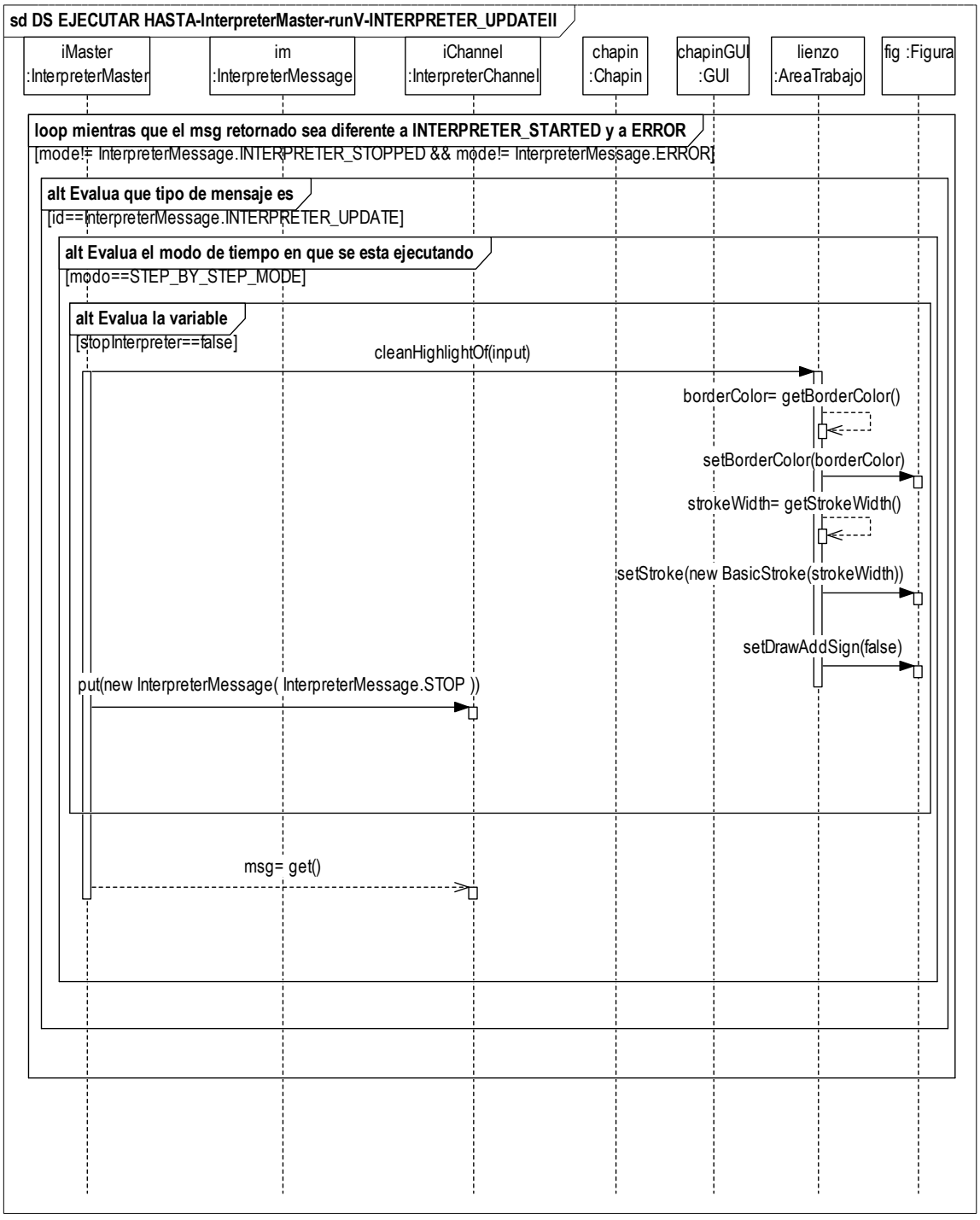

sd DS EJECUTAR HASTA-InterpreterMaster-run-INTERPRETER_STARTED
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
run()
msg= get()
mode= getId()
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
mode= getId()
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INTERPRETER_STARTED]
chapinGui= getGUI()
setExecMode(mode)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setModo(AreaTrabajo.EXE_MODE)
chapinGui= getGUI()
appendToOutput("Iniciando Interpretación\n")
put(new InterpreterMessage(InterpreterMessage.CONTINUE))
msg= get()

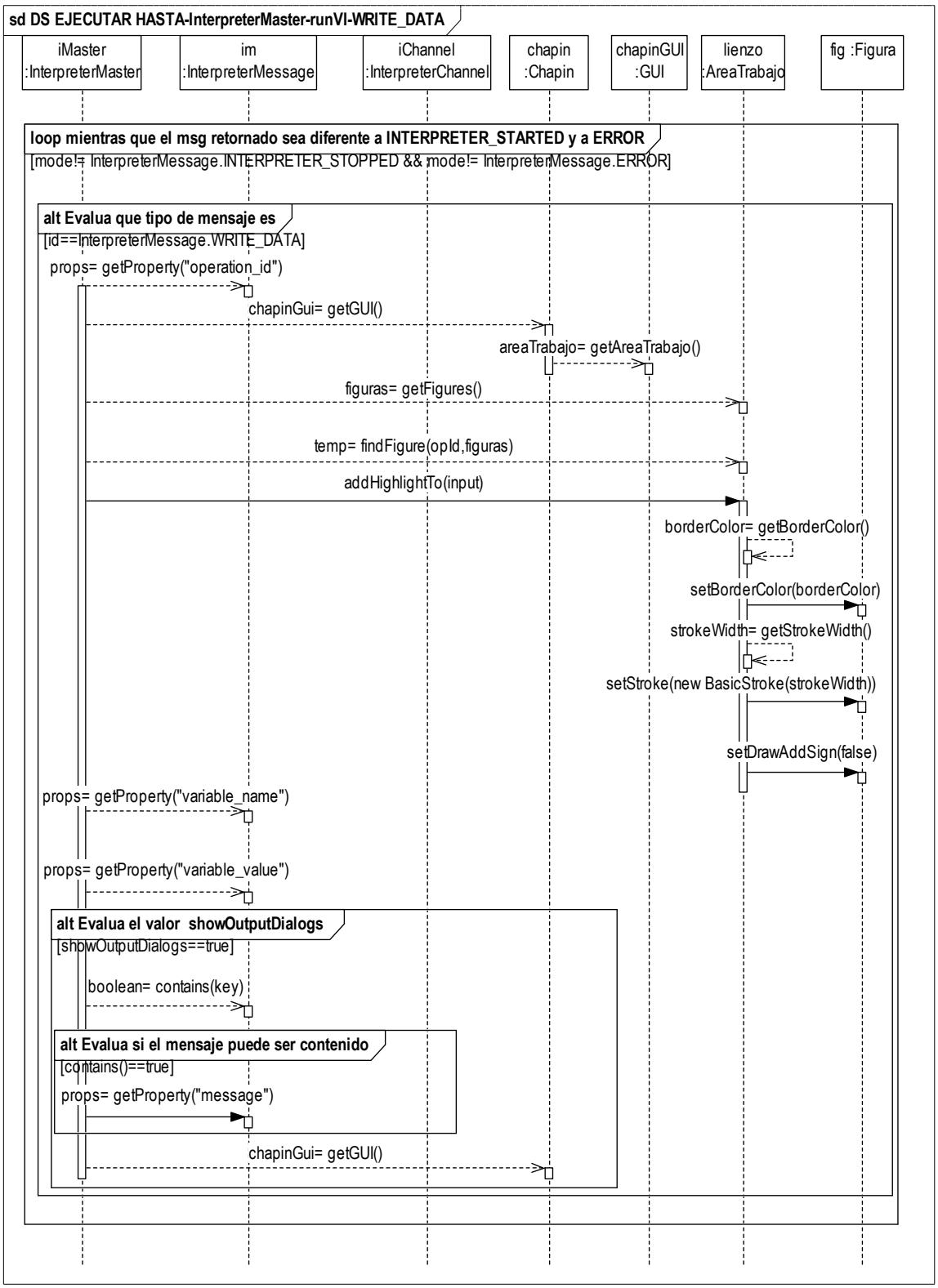

sd DS EJECUTAR HASTA-InterpreterMaster-runII-INPUT_REQUIRED
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
temp :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
loop mientras que halla mas elementos en la lista figuras
[iter.hasNext()==true]
id= getId()
alt Compara el id retornado por la figura
[id==id]
temp= findFigure(opId,figuras)
boolean= esHoja()
alt Evalua si no es la primera estructura
[esHoja()==false]
figuras= getFiguras()
temp= findFigure(id,figuras)
alt Evalua si el valor retornado por findFigure() no es nulo
[temp != null]
temp= findFigure(opId,figuras)
[iter.hasNext()==false]
null= findFigure(opId,figuras);

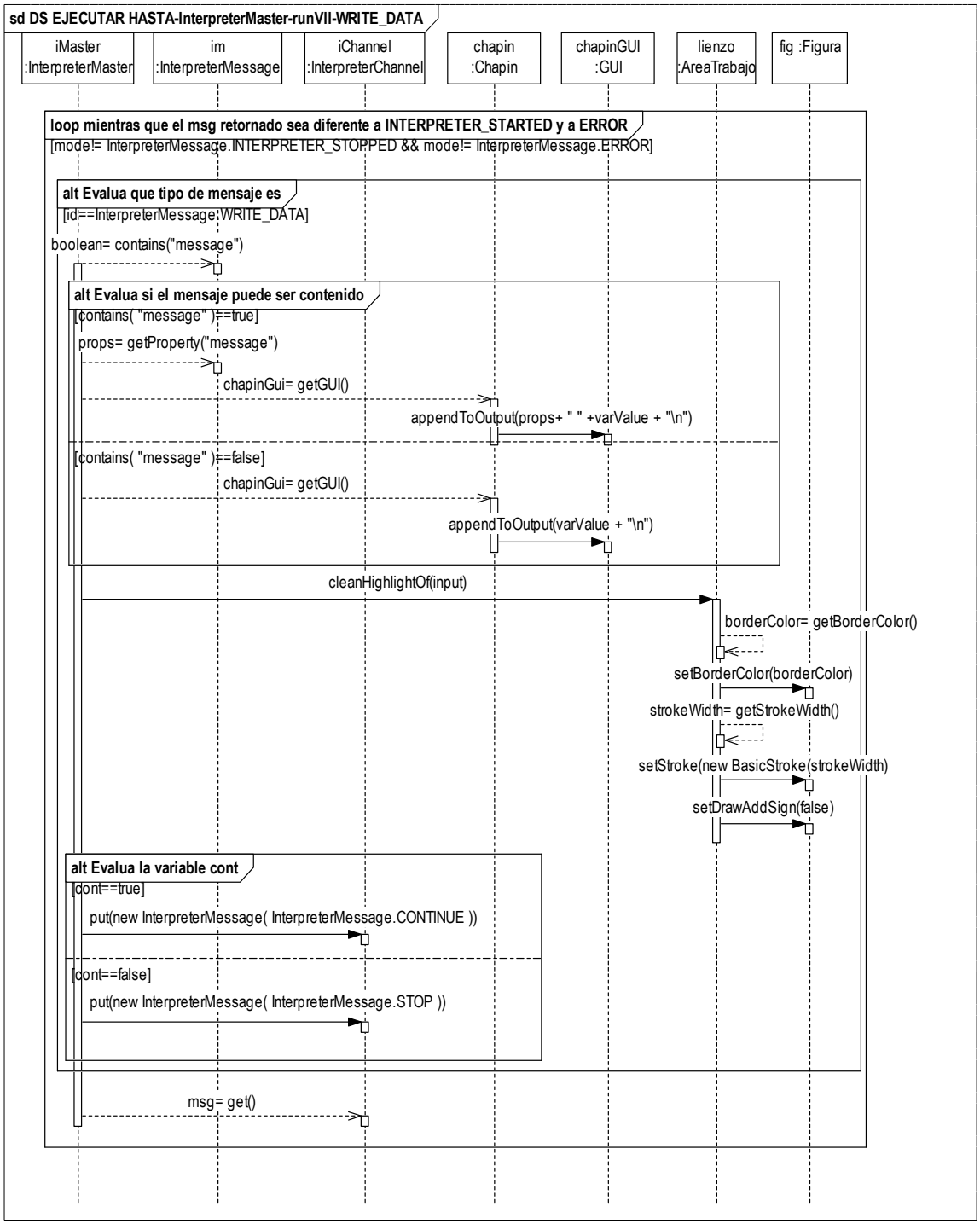

sd DS EJECUTAR HASTA-InterpreterMaster-runIII-INPUT_REQUIRED II
iMaster
:InterpreterMaster
im
:InterpreterMessage
:AreaTrabajo
fig :Figura
chapin
:Chapin
:StringUtils
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
addHighlightTo(input)
modoActual= getModo()
alt Evalua la accion que se esta llevando a cabo mediante modoActual
[modoActual==EXE_MODE]
exeModeColor= getActualModeBorderColor()
setBorderColor(exeModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth* 3))
props= getProperty("variable_name")
boolean= contains("message")
alt Evalua si el parametro puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
alt Evalua si el usuario entro expresion
[exp != null && exp.length() > 0]
boolean= isValidNumber(exp)
props= put("variable_value", exp)

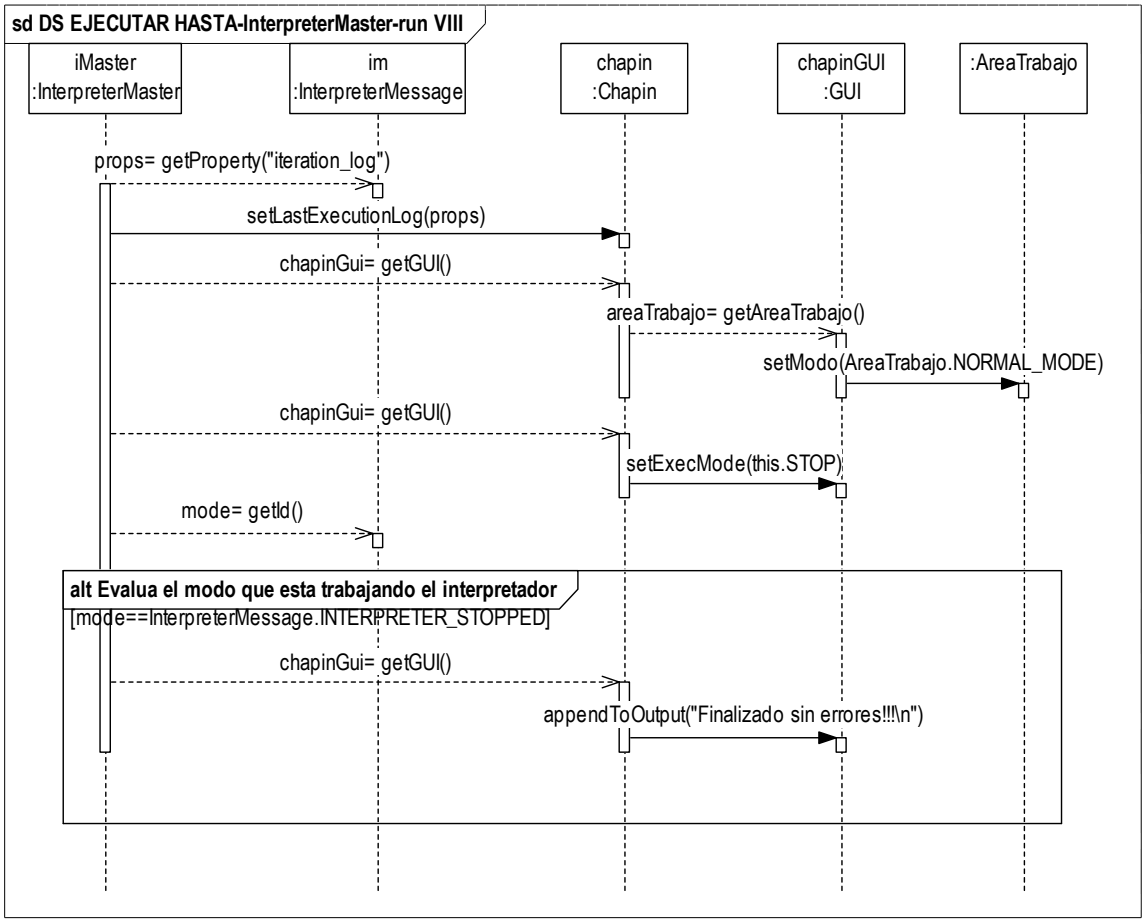

sd DS EJECUTAR HASTA-InterpreterMaster-runIV-INPUT_REQUIRED III
iMaster :InterpreterMaster
im :InterpreterMessage
chapin :Chapin
chapinGUI :GUI
iChannel :InterpreterChannel
lienzo :AreaTrabajo
:Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[mode==InterpreterMessage.INPUT_REQUIRED]
alt Evalua si el usuario entro expresion
[exp != null && exp.length() > 0]
boolean= contains("message")
alt Evalua si el parametro puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
appendToOutput(props+" " + exp + "\n")
[contains("message")==false]
chapinGui= getGUI()
appendToOutput(exp + "\n")
put(answer)
[exp == null]
chapinGui= getGUI()
appendToOutput("ERROR: " + varName + " Leer Cancelado\n")
put(answer)
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(strokeWidth(strokeWidth))
setDrawAddSign(false)
msg= get()

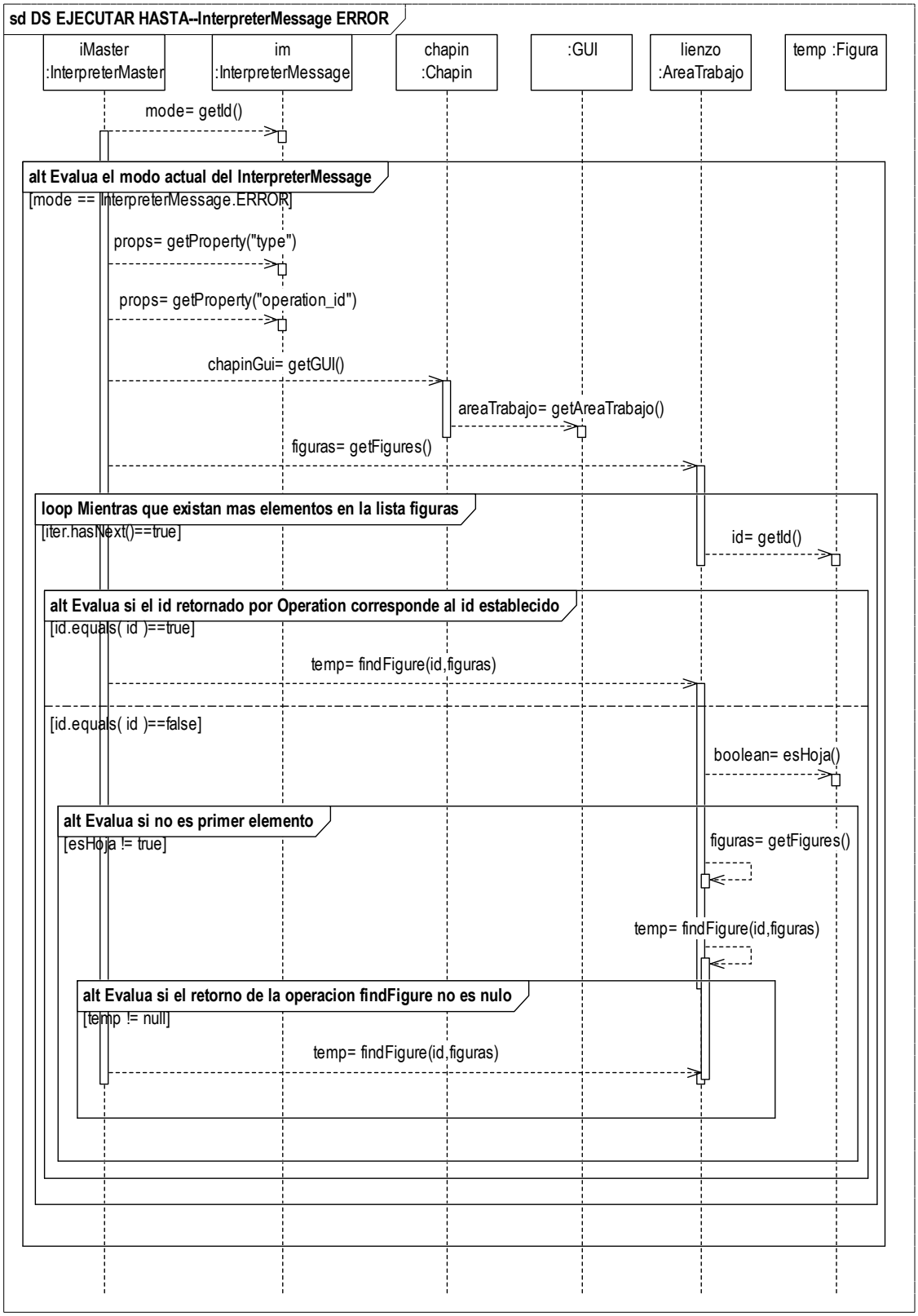

sd DS EJECUTAR HASTA-InterpreterMaster-runV-INTERPRETER_UPDATE
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.INTERPRETER_UPDATE]
alt Evalua el modo de tiempo en que se esta ejecutando
[modo==STEP_BY_STEP_MODE]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
temp= findFigure(opId,figuras)
addHighlightTo(input)
exeModeColor= getActualModeBorderColor()
setBorderColor(exeModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth*3))
props= getProperty("instruction_number")
props= getProperty("symbol_table")
chapinGui= getGUI()
setConsoleText(instNum + "\n" + symTable)
alt Evalua la variable
[stopInterpreter==false]
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
put(new InterpreterMessage(InterpreterMessage.CONTINUE))

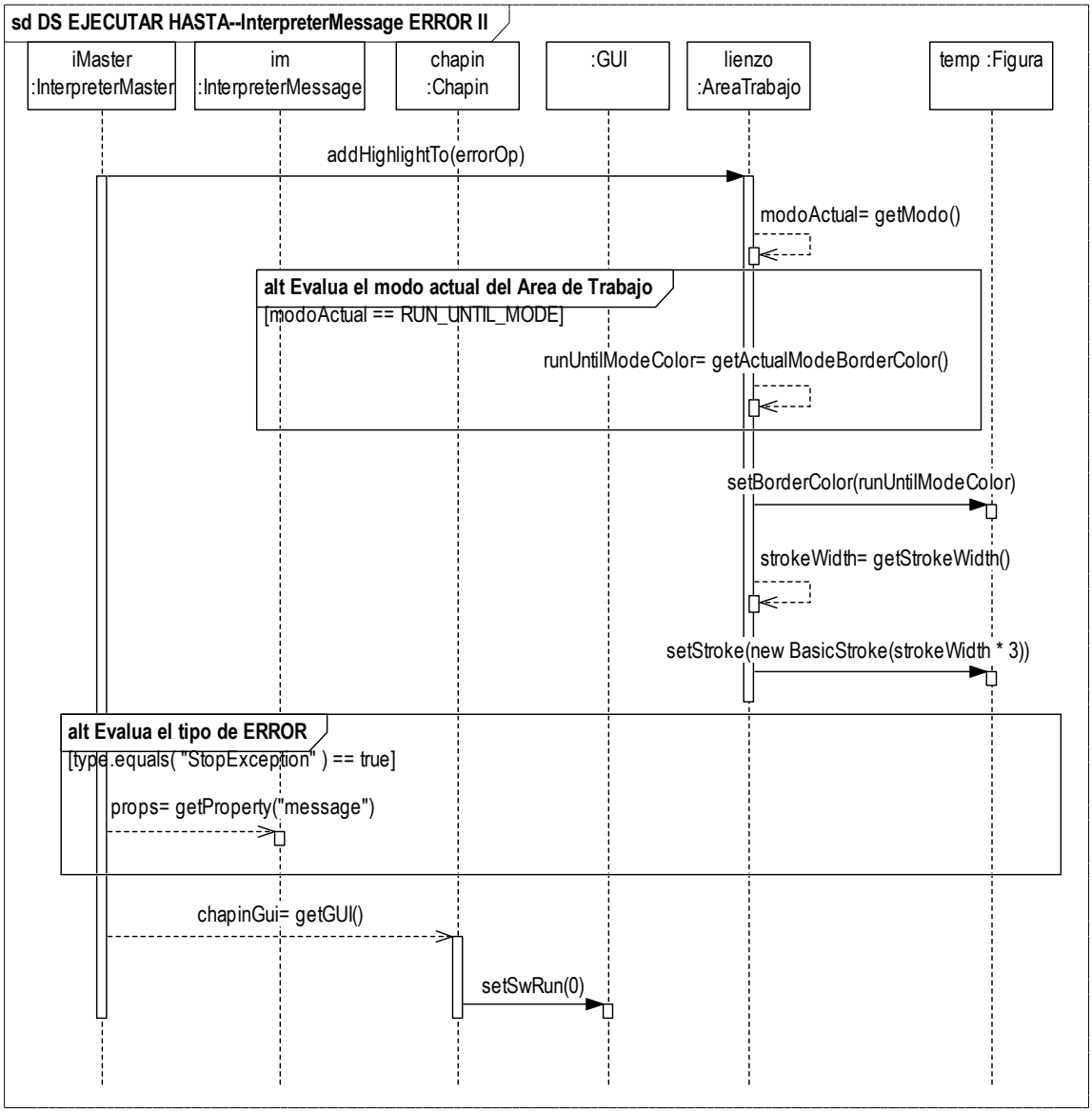

sd DS EJECUTAR HASTA-InterpreterMaster-runV-INTERPRETER_UPDATEII
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.INTERPRETER_UPDATE]
alt Evalua el modo de tiempo en que se esta ejecutando
[modo==STEP_BY_STEP_MODE]
alt Evalua la variable
[stopInterpreter==false]
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
put(new InterpreterMessage(InterpreterMessage.STOP))
msg= get()


sd DS EJECUTAR HASTA-InterpreterMaster-runVI-WRITE_DATA
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.WRITE_DATA]
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
temp= findFigure(opId,figuras)
addHighlightTo(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth))
setDrawAddSign(false)
props= getProperty("variable_name")
props= getProperty("variable_value")
alt Evalua el valor showOutputDialogs
[showOutputDialogs==true]
boolean= contains(key)
alt Evalua si el mensaje puede ser contenido
[contains()==true]
props= getProperty("message")
chapinGui= getGUI()

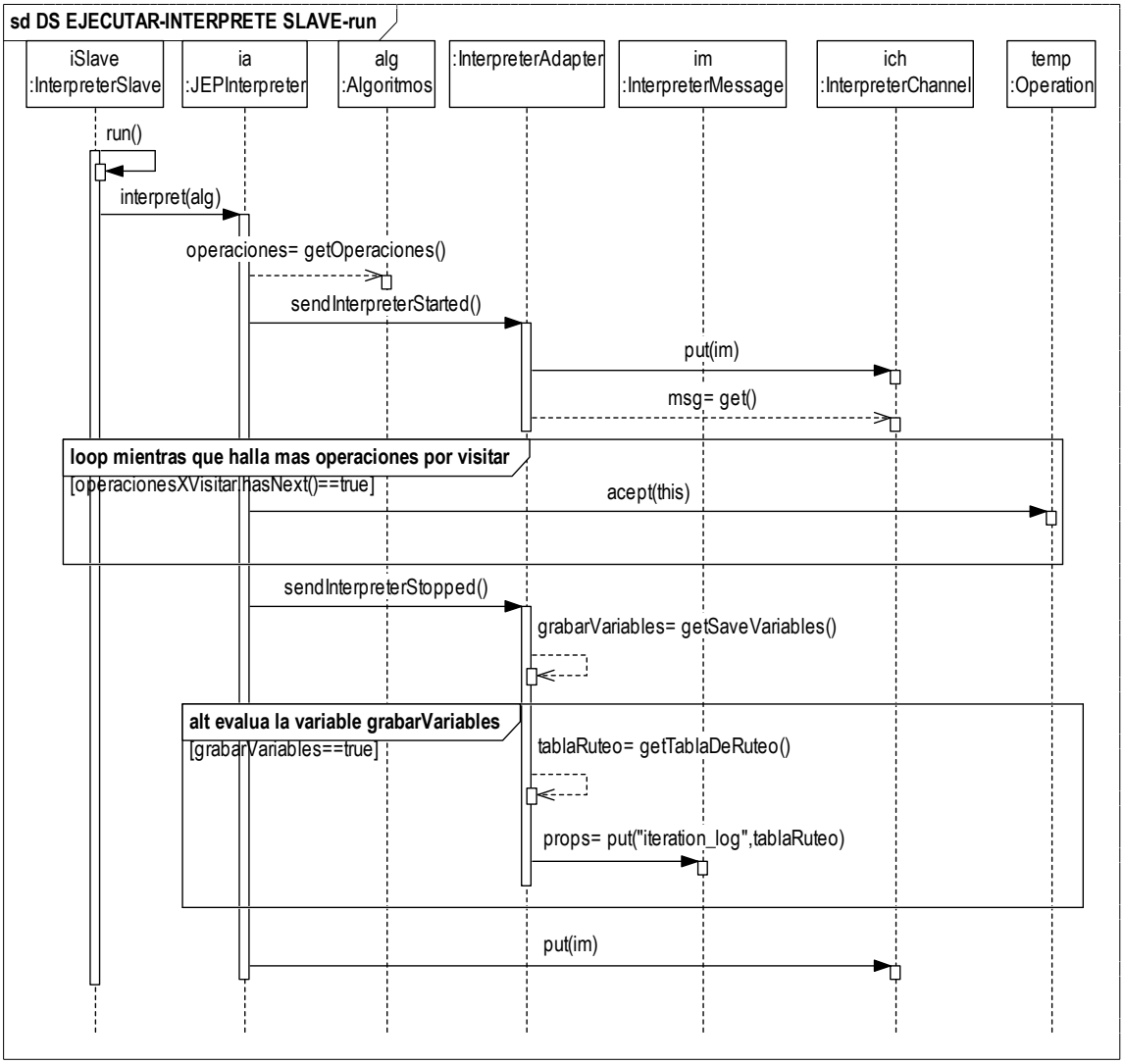

sd DS EJECUTAR HASTA-InterpreterMaster-runVII-WRITE_DATA
iMaster :InterpreterMaster
im :InterpreterMessage
iChannel :InterpreterChannel
chapin :Chapin
chapinGUI :GUI
lienzo :AreaTrabajo
fig :Figura
loop mientras que el msg retornado sea diferente a INTERPRETER_STARTED y a ERROR
[mode!= InterpreterMessage.INTERPRETER_STOPPED && mode!= InterpreterMessage.ERROR]
alt Evalua que tipo de mensaje es
[id==InterpreterMessage.WRITE_DATA]
boolean= contains("message")
alt Evalua si el mensaje puede ser contenido
[contains("message")==true]
props= getProperty("message")
chapinGui= getGUI()
appendToOutput(props+ " " +varValue + "\n")
[contains("message")==false]
chapinGui= getGUI()
appendToOutput(varValue + "\n")
cleanHighlightOf(input)
borderColor= getBorderColor()
setBorderColor(borderColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth)
setDrawAddSign(false)
alt Evalua la variable cont
[cont==true]
put(new InterpreterMessage(InterpreterMessage.CONTINUE))
[cont==false]
put(new InterpreterMessage(InterpreterMessage.STOP))
msg= get()

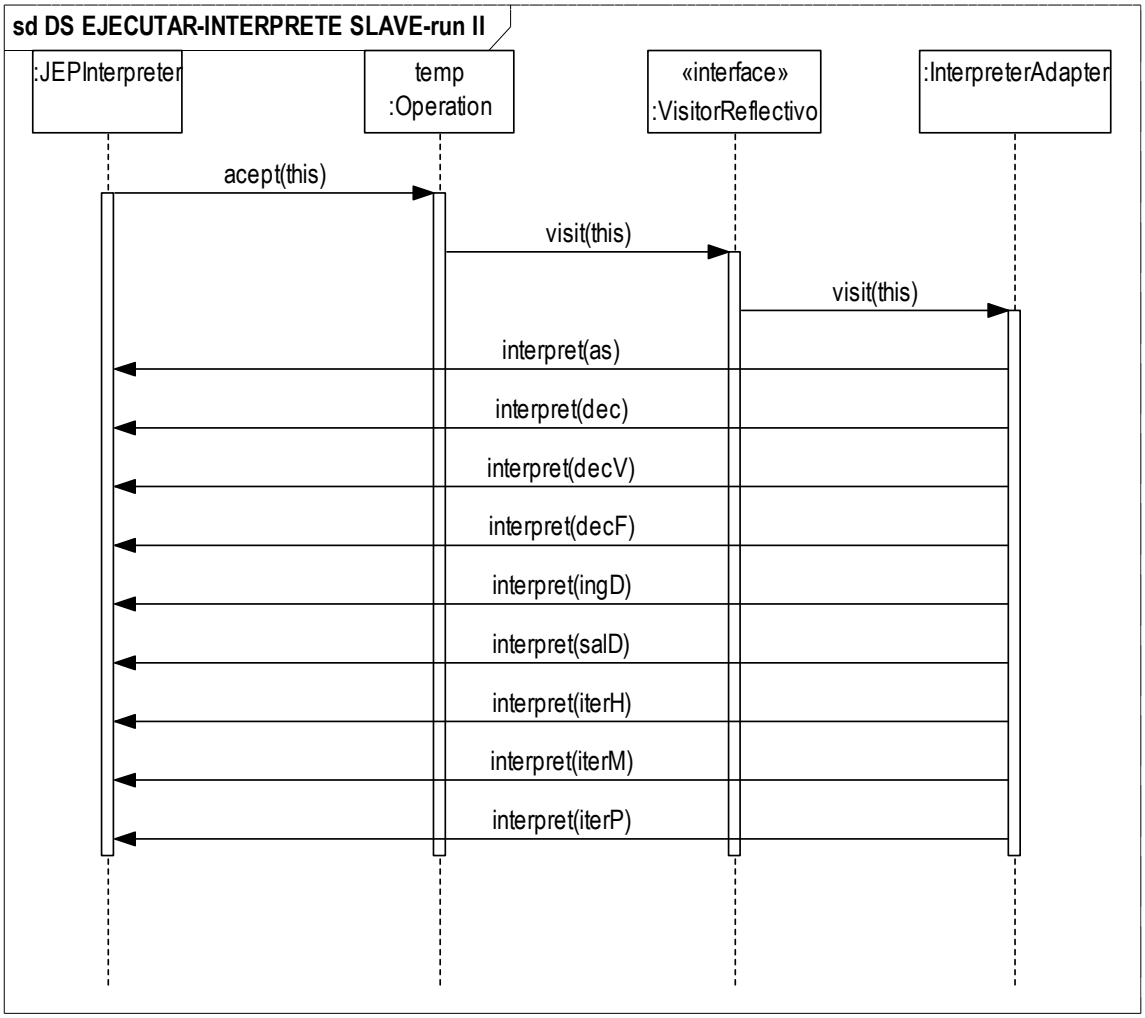

sd DS EJECUTAR HASTA-InterpreterMaster-run VIII
iMaster
:InterpreterMaster
im
:InterpreterMessage
chapin
:Chapin
chapinGUI
:GUI
:AreaTrabajo
props= getProperty("iteration_log")
setLastExecutionLog(props)
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
setModo(AreaTrabajo.NORMAL_MODE)
chapinGui= getGUI()
setExecMode(this.STOP)
mode= getId()
alt Evalua el modo que esta trabajando el interpretador
[mode==InterpreterMessage.INTERPRETER_STOPPED]
chapinGui= getGUI()
appendToOutput("Finalizado sin errores!!!\n")


sd DS EJECUTAR HASTA--InterpreterMessage ERROR
iMaster :InterpreterMaster
im :InterpreterMessage
chapin :Chapin
:GUI
lienzo :AreaTrabajo
temp :Figura
mode= getId()
alt Evalua el modo actual del InterpreterMessage
[mode == InterpreterMessage.ERROR]
props= getProperty("type")
props= getProperty("operation_id")
chapinGui= getGUI()
areaTrabajo= getAreaTrabajo()
figuras= getFigures()
loop Mientras que existan mas elementos en la lista figuras
[iter.hasNext()==true]
id= getId()
alt Evalua si el id retornado por Operation corresponde al id establecido
[id.equals(id)==true]
temp= findFigure(id,figuras)
[id.equals(id)==false]
boolean= esHoja()
alt Evalua si no es primer elemento
[esHoja != true]
figuras= getFigures()
temp= findFigure(id,figuras)
alt Evalua si el retorno de la operacion findFigure no es nulo
[temp != null]
temp= findFigure(id,figuras)

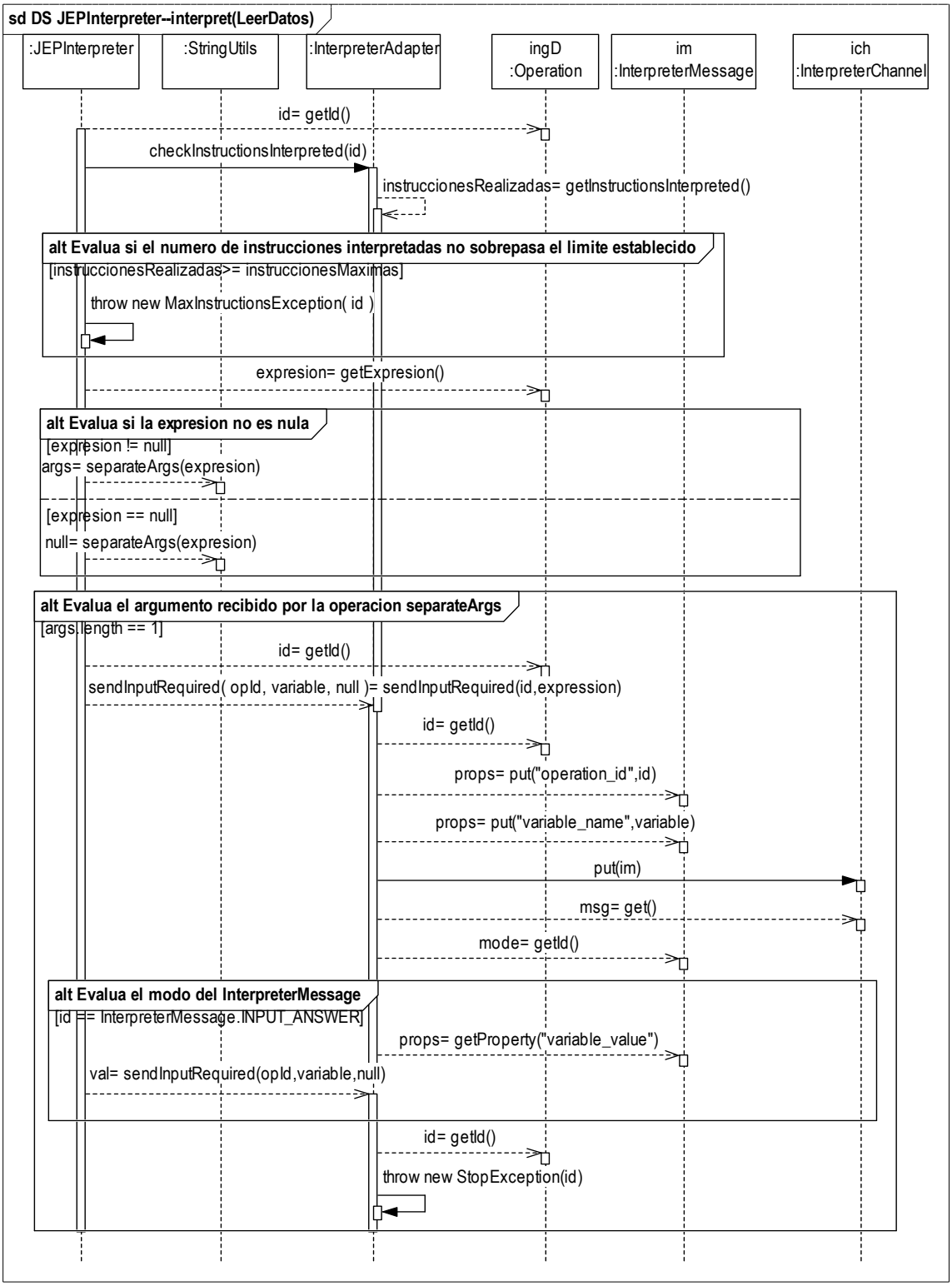

sd DS EJECUTAR HASTA--InterpreterMessage ERROR II
iMaster
:InterpreterMaster
im
:InterpreterMessage
chapin
:Chapin
:GUI
lienzo
:AreaTrabajo
temp :Figura
addHighlightTo(errorOp)
modoActual= getModo()
alt Evalua el modo actual del Area de Trabajo
[modoActual == RUN_UNTIL_MODE]
runUntilModeColor= getActualModeBorderColor()
setBorderColor(runUntilModeColor)
strokeWidth= getStrokeWidth()
setStroke(new BasicStroke(strokeWidth * 3))
alt Evalua el tipo de ERROR
[type.equals("StopException") == true]
props= getProperty("message")
chapinGui= getGUI()
setSwRun(0)

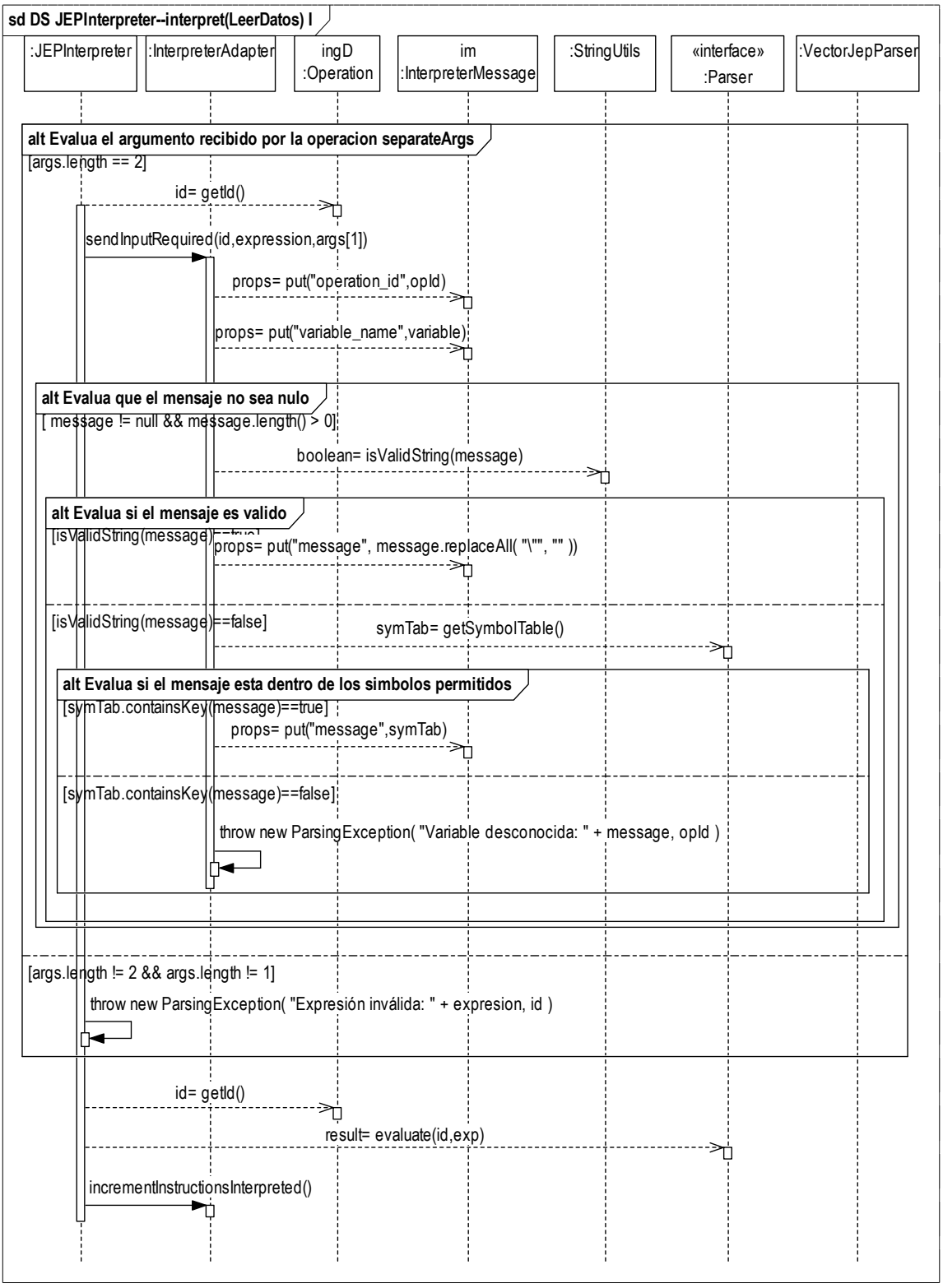
10.13 DIAGRAMA DE SECUENCIA DEL MÉTODO RUN() EN INTERPRETERSLAVE

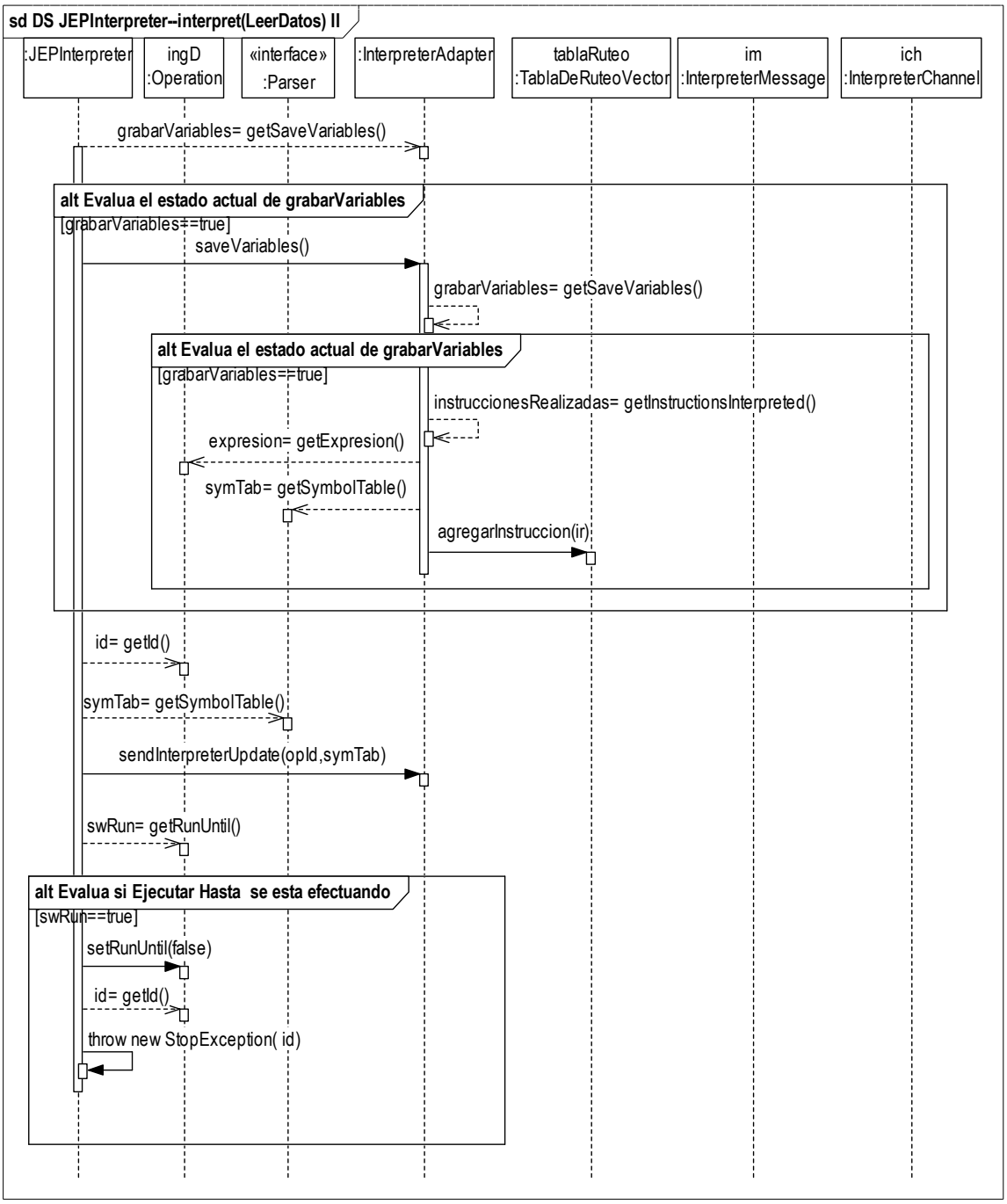



sd DS EJECUTAR-INTERPRETE SLAVE-run II
:JEPInterpreter
temp
:Operation
«interface»
:VisitorReflectivo
:InterpreterAdapter
acept(this)
visit(this)
visit(this)
interpret(as)
interpret(dec)
interpret(decV)
interpret(decF)
interpret(ingD)
interpret(salD)
interpret(iterH)
interpret(iterM)
interpret(iterP)

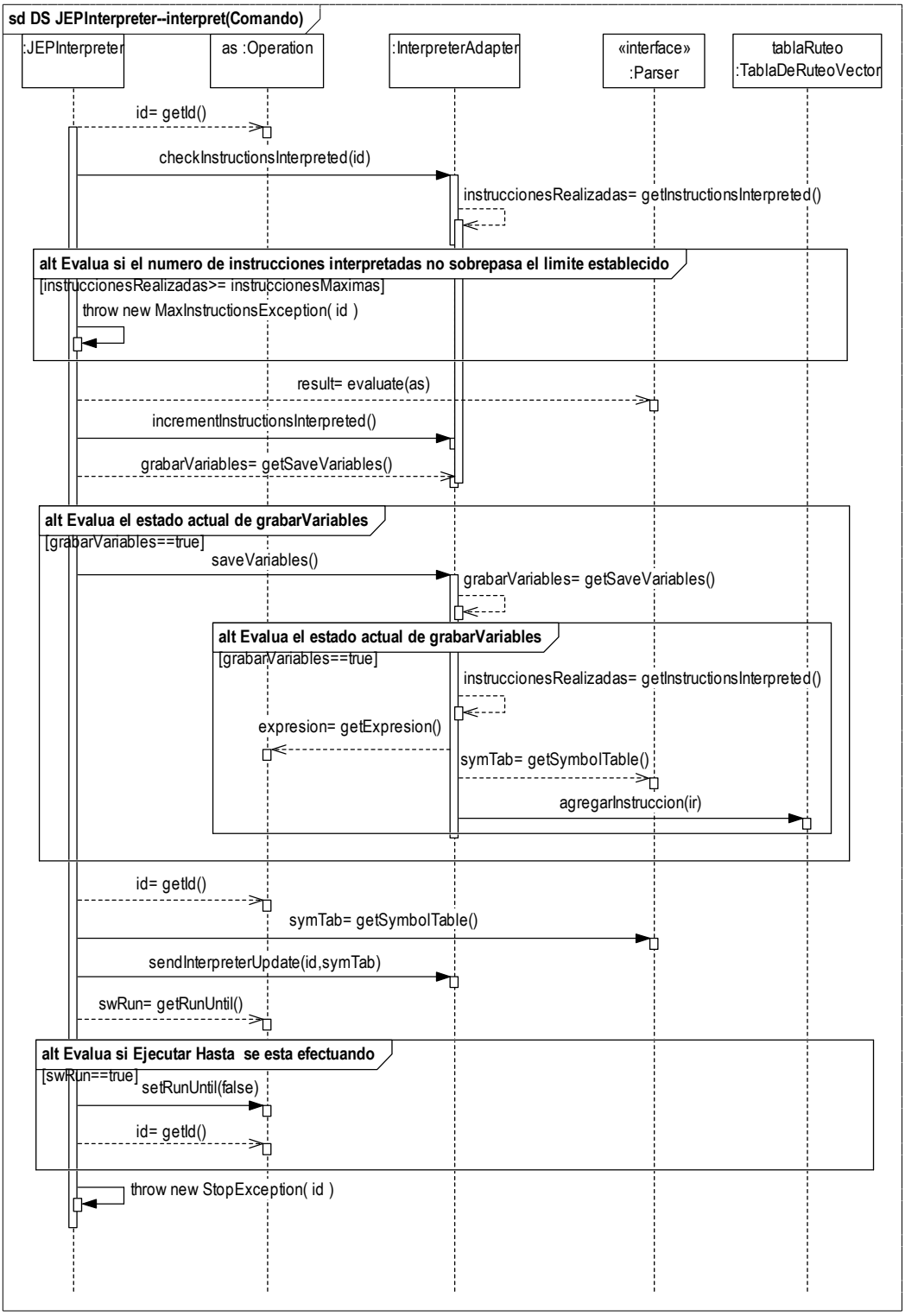
10.13.1 Diagrama de Secuencia de interpret(LeerDatos)



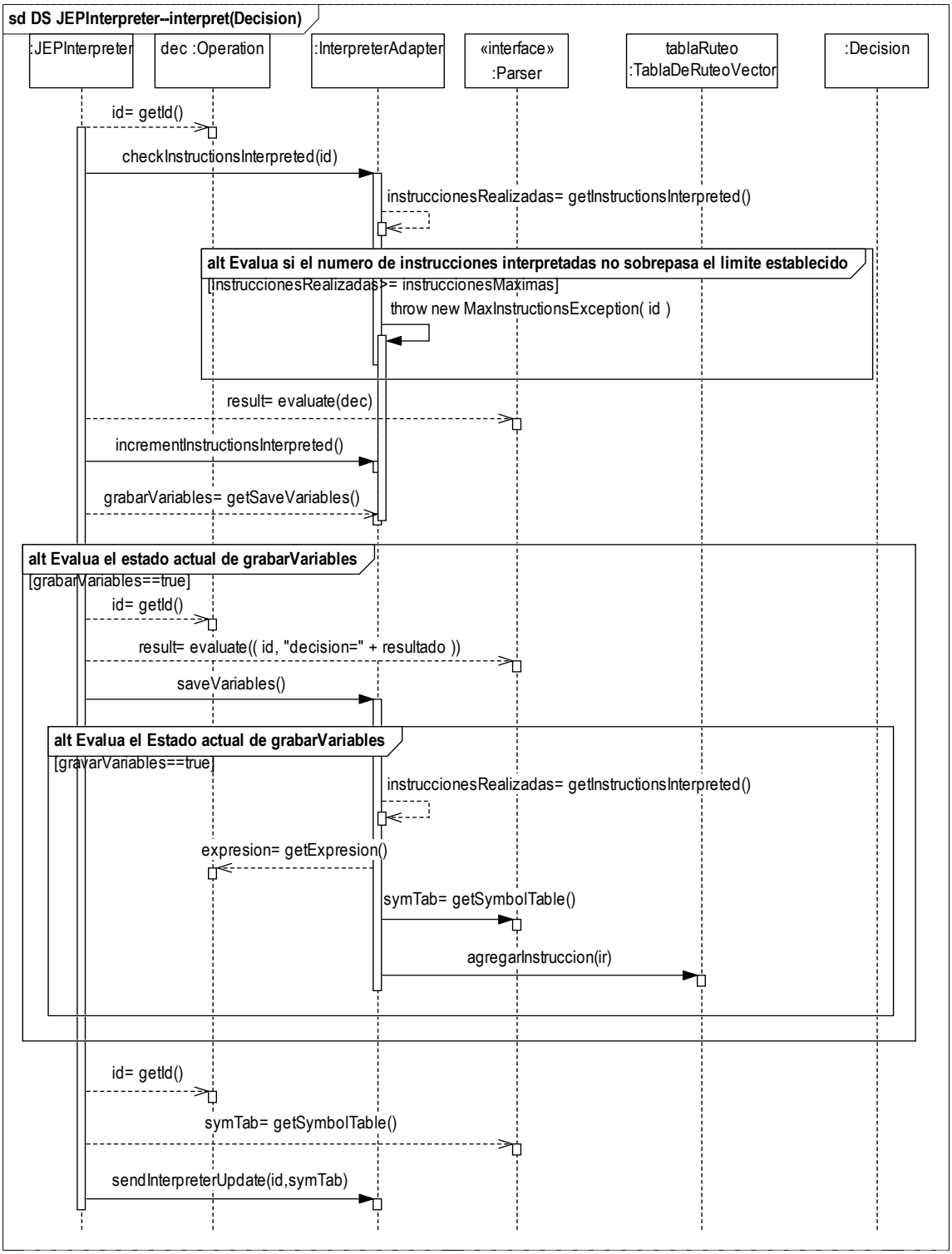

sd DS JEPInterpreter--interpret(LeerDatos) I
:JEPInterpreter
:InterpreterAdapter
ingD
:Operation
im
:InterpreterMessage
:StringUtils
«interface»
:Parser
:VectorJepParser
alt Evalua el argumento recibido por la operacion separateArgs
[args.length == 2]
id= getId()
sendInputRequired(id,expression,args[1])
props= put("operation_id",opId)
props= put("variable_name",variable);
alt Evalua que el mensaje no sea nulo
[message != null && message.length() > 0]
boolean= isValidString(message)
alt Evalua si el mensaje es valido
[isValidString(message)==true]
props= put("message", message.replaceAll("\"", ""))
[isValidString(message)==false]
symTab= getSymbolTable()
alt Evalua si el mensaje esta dentro de los simbolos permitidos
[symTab.containsKey(message)==true]
props= put("message",symTab)
[symTab.containsKey(message)==false]
throw new ParsingException("Variable desconocida: " + message, opId)
[args.length != 2 && args.length != 1]
throw new ParsingException("Expresión inválida: " + expresion, id)
id= getId()
result= evaluate(id,exp)
incrementInstructionsInterpreted()

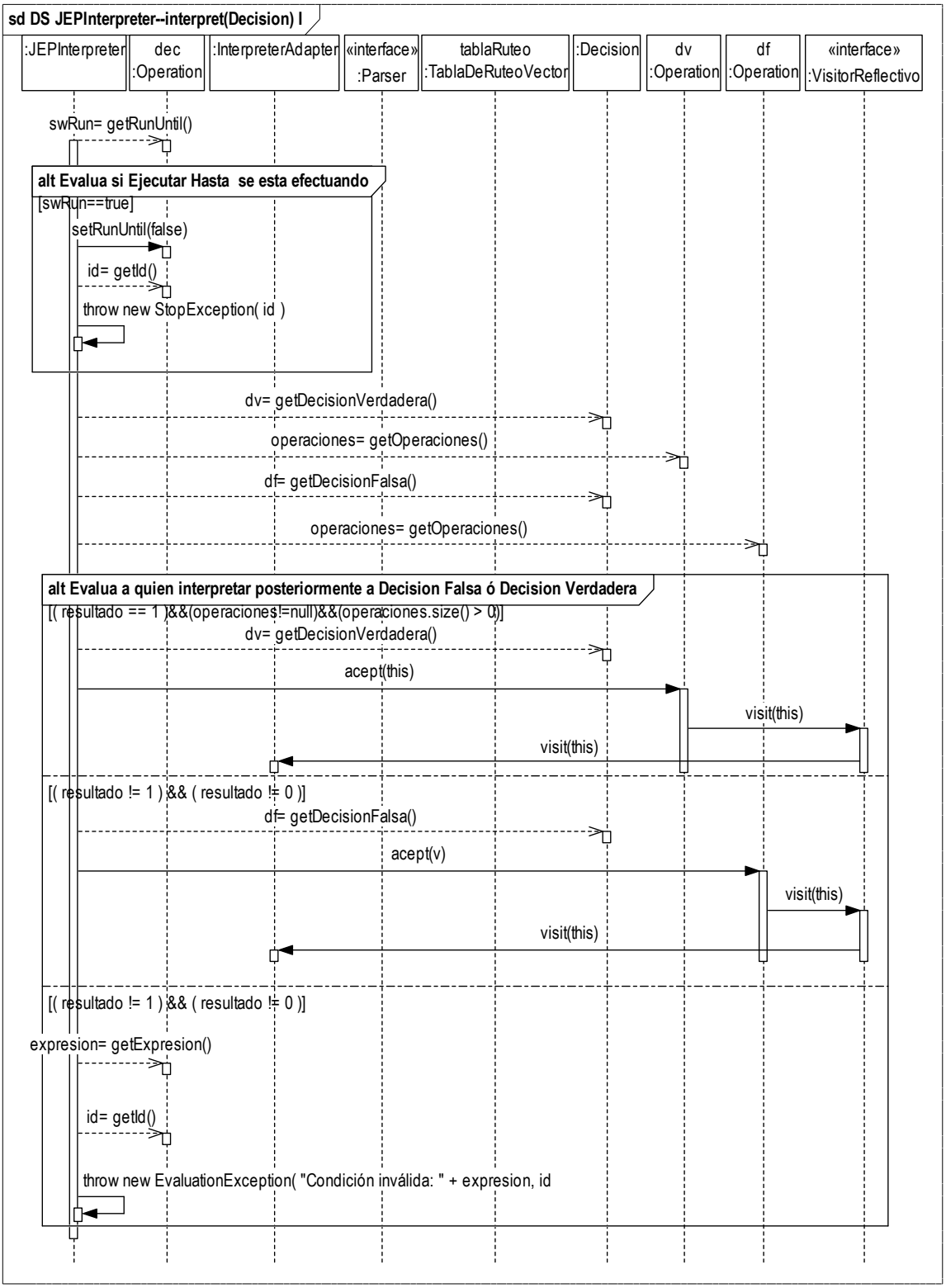

sd DS JEPInterpreter--interpret(LeerDatos) II
:JEPInterpreter
ingD
:Operation
«interface»
:Parser
:InterpreterAdapter
tablaRuteo
:TablaDeRuteoVector
im
:InterpreterMessage
ich
:InterpreterChannel
grabarVariables= getSaveVariables()
alt Evalua el estado actual de grabarVariables
[grabarVariables==true]
saveVariables()
grabarVariables= getSaveVariables()
alt Evalua el estado actual de grabarVariables
[grabarVariables==true]
instruccionesRealizadas= getInstructionsInterpreted()
expresion= getExpresion()
symTab= getSymbolTable()
agregarInstruccion(ir)
id= getId()
symTab= getSymbolTable()
sendInterpreterUpdate(opId,symTab)
swRun= getRunUntil()
alt Evalua si Ejecutar Hasta se esta efectuando
[swRun==true]
setRunUntil(false)
id= getId()
throw new StopException(id)

10.13.2 Diagrama de Secuencia de interpret(Comando)

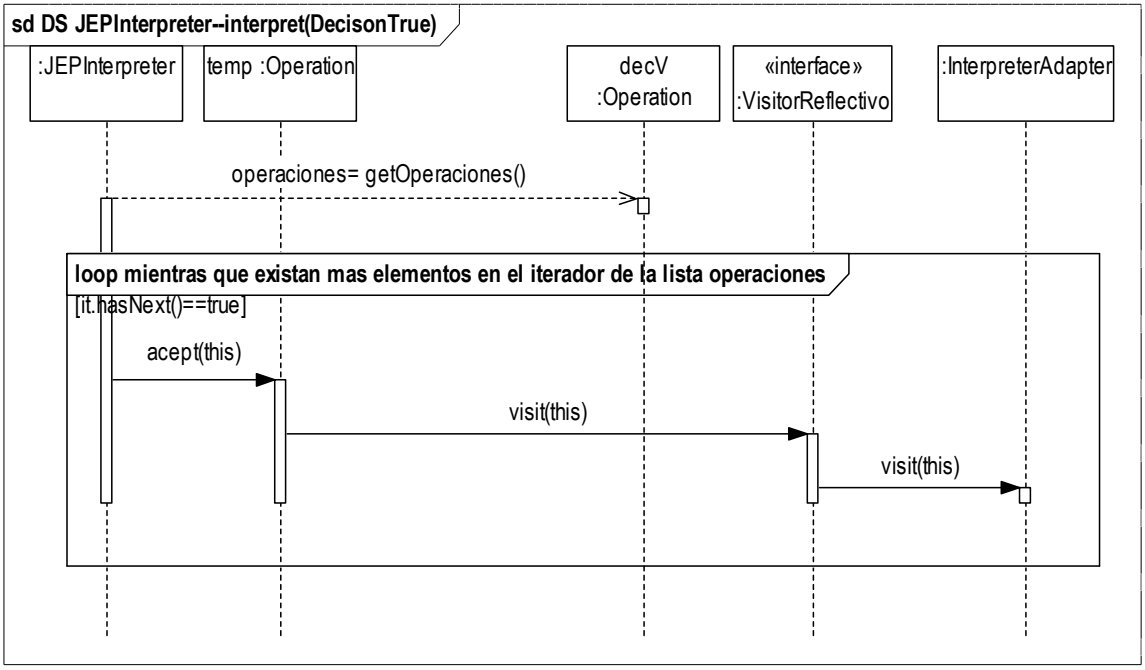


10.13.3 Diagrama de Secuencia de interpret(Decisión)

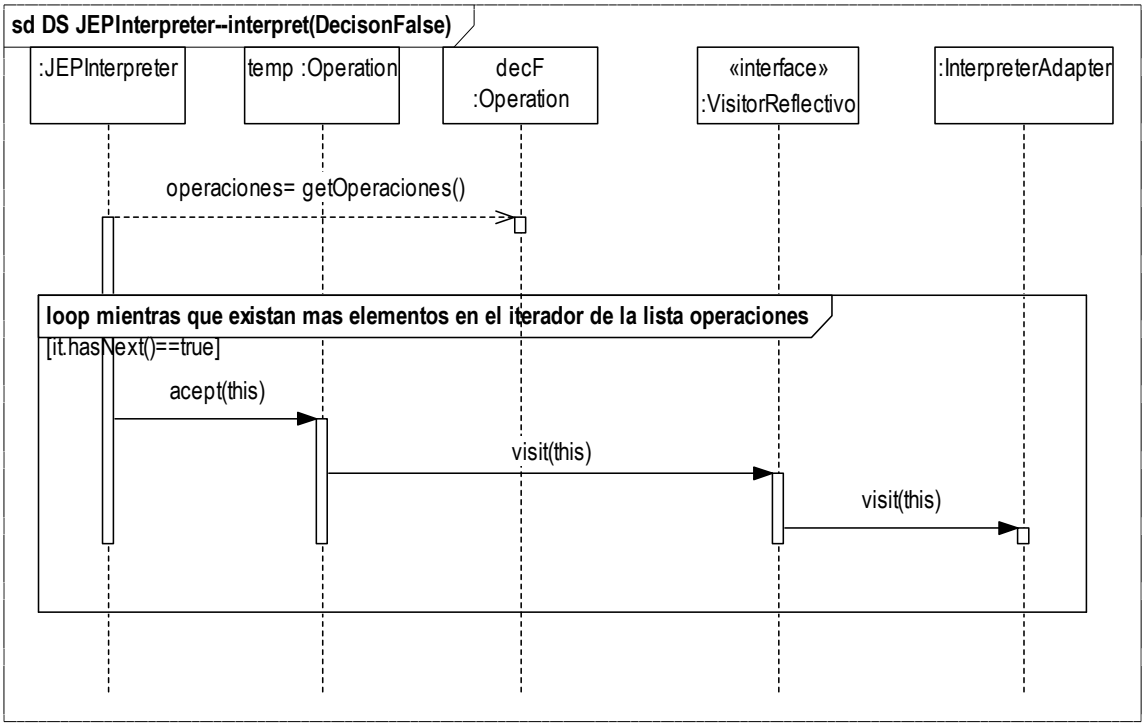



sd DS JEPInterpreter--interpret(Decision) I
:JEPInterpreter
dec :Operation
:InterpreterAdapter
«interface» :Parser
tablaRuteo :TablaDeRuteoVector
:Decision
dv :Operation
df :Operation
«interface» :VisitorReflectivo
swRun= getRunUntil()
alt Evalua si Ejecutar Hasta se esta efectuando
[swRun==true]
setRunUntil(false)
id= getId()
throw new StopException(id)
dv= getDecisionVerdadera()
operaciones= getOperaciones()
df= getDecisionFalsa()
operaciones= getOperaciones()
alt Evalua a quien interpretar posteriormente a Decision Falsa ó Decision Verdadera
[(resultado == 1)&&(operaciones!=null)&&(operaciones.size() > 0)]
dv= getDecisionVerdadera()
acept(this)
visit(this)
visit(this)
[(resultado != 1) && (resultado != 0)]
df= getDecisionFalsa()
acept(v)
visit(this)
visit(this)
[(resultado != 1) && (resultado != 0)]
expresion= getExpresion()
id= getId()
throw new EvaluationException("Condición inválida: " + expresion, id

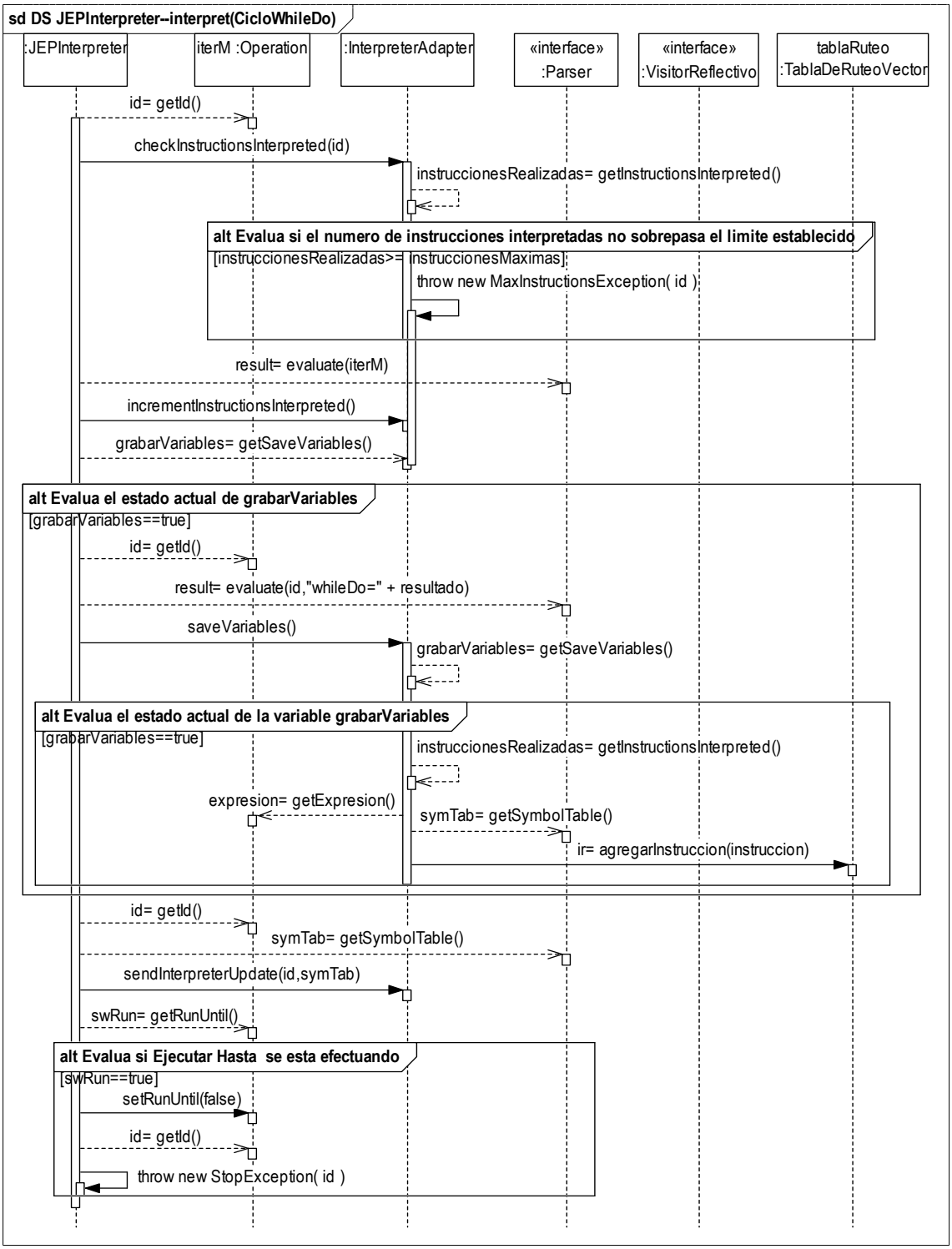
10.13.4 Diagrama de Secuencia de interpret(DecisionTrue)

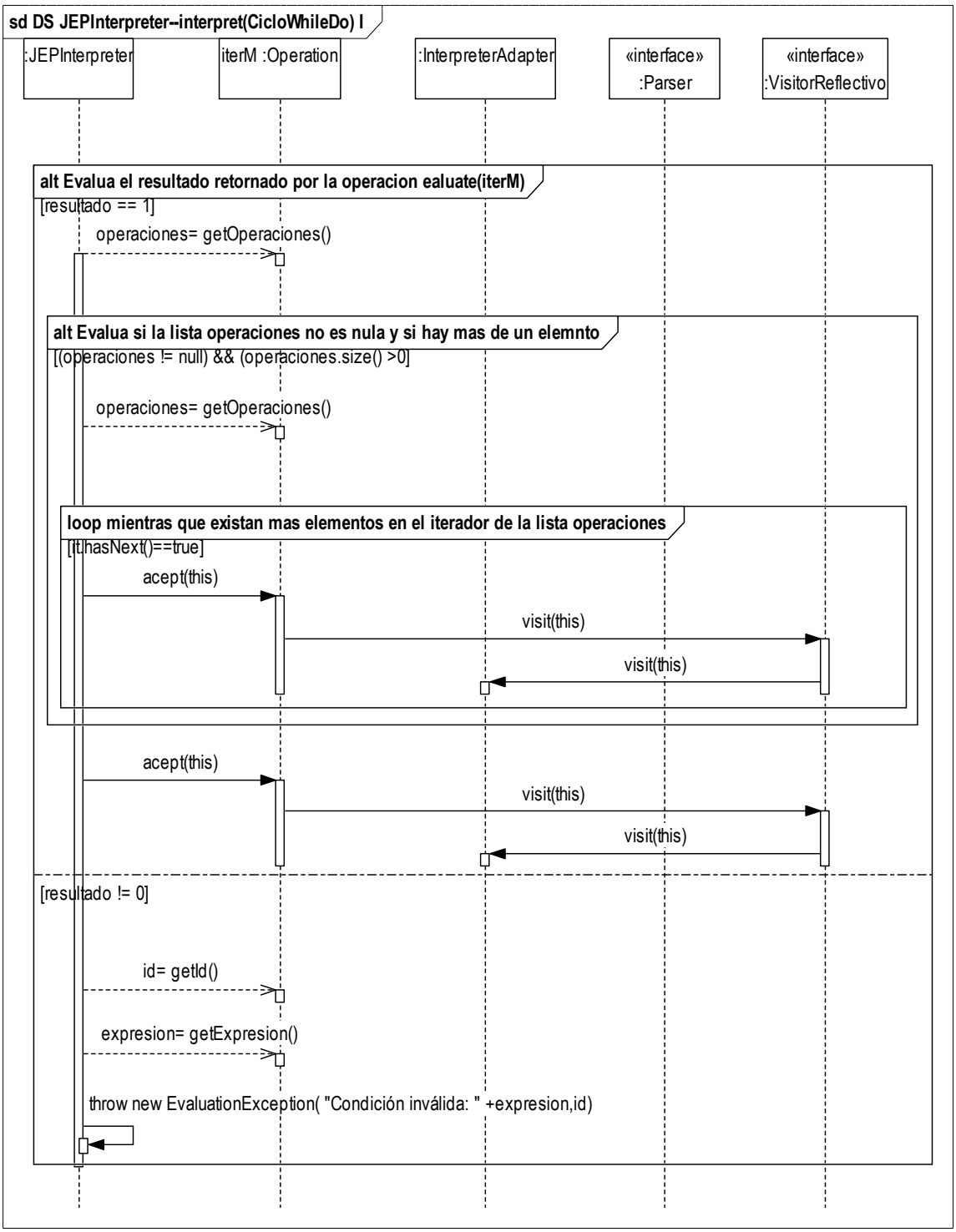


10.13.5 Diagrama de Secuencia de interpret(DecisionFalse)

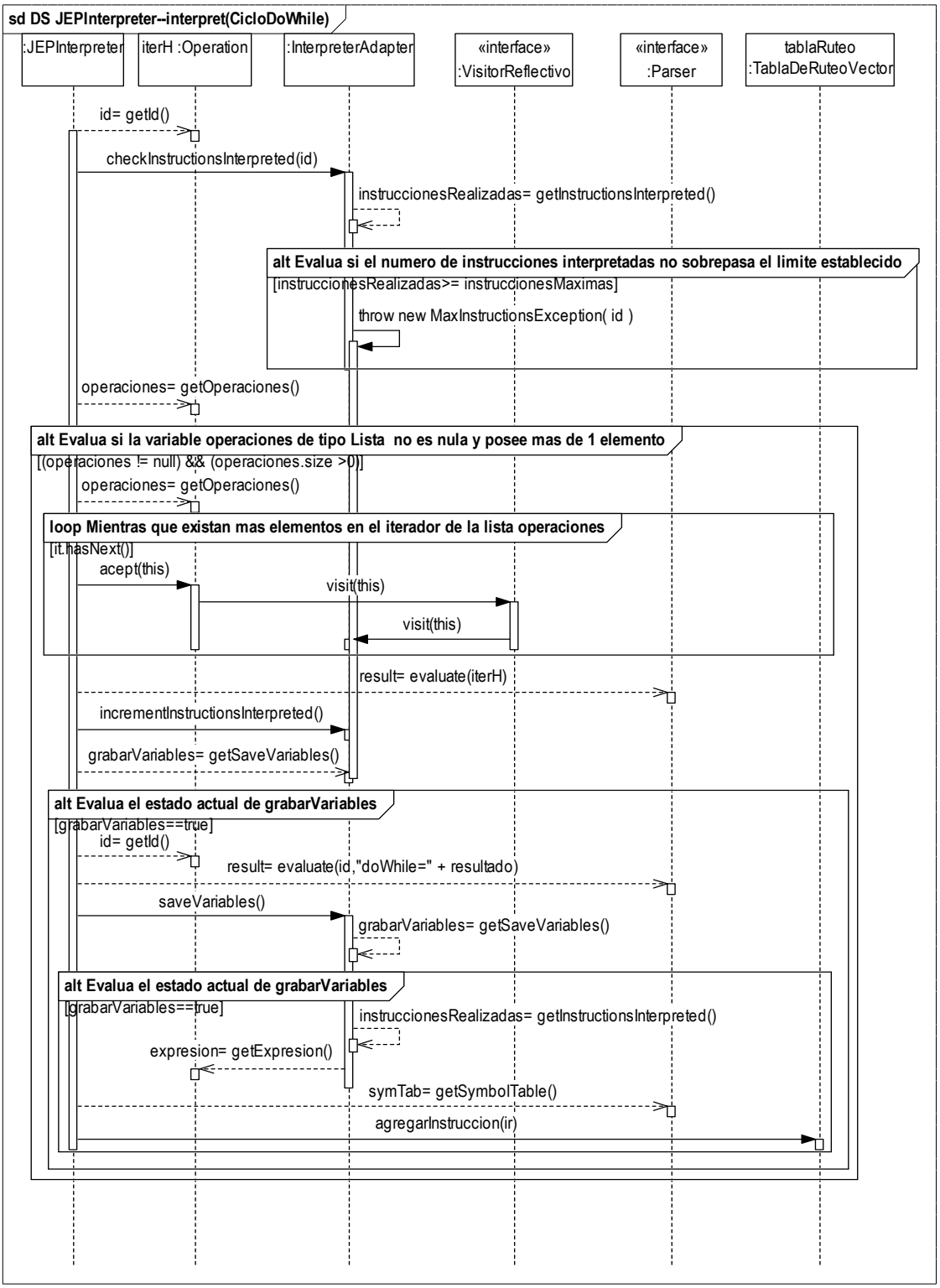


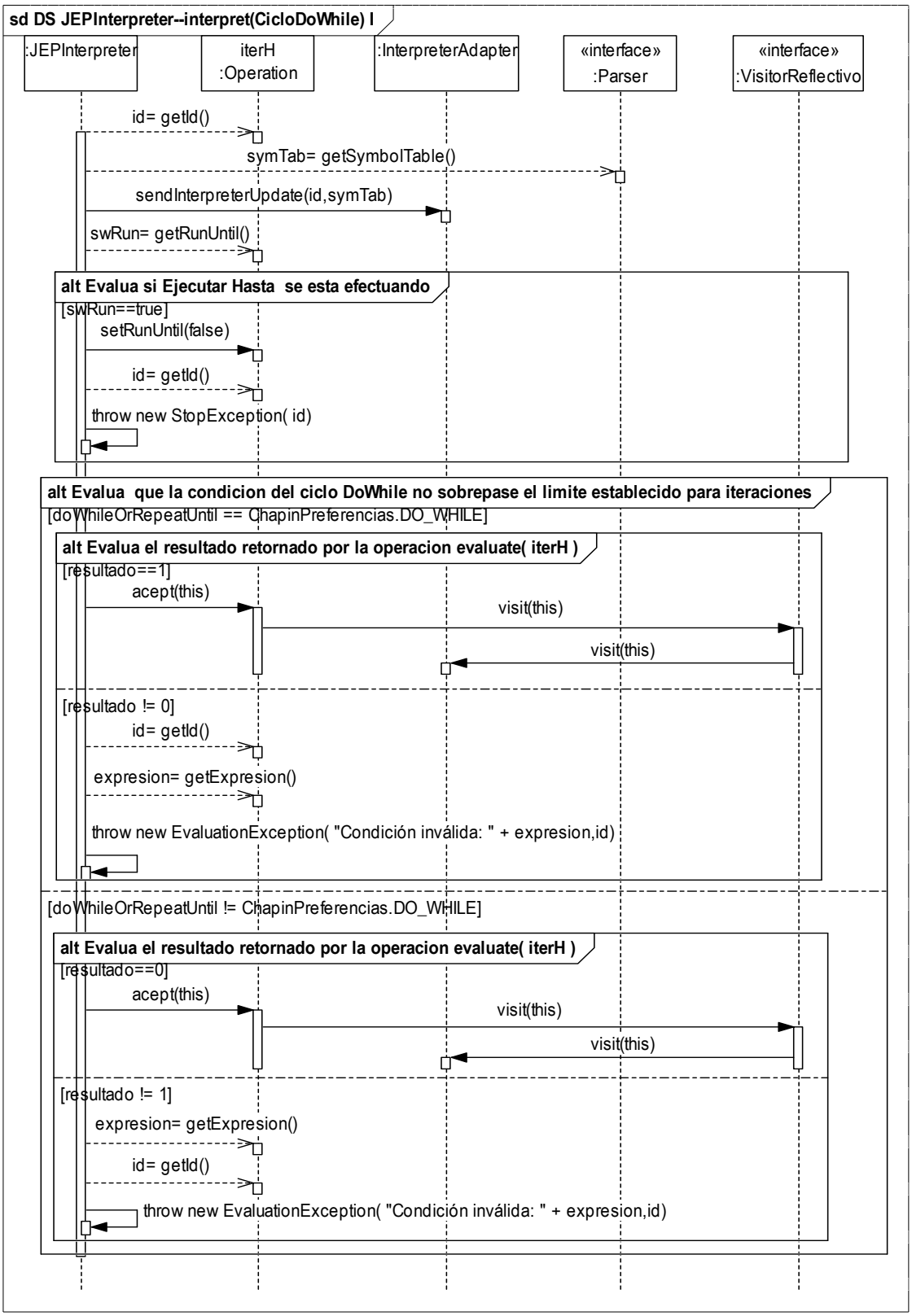
10.13.6 Diagrama de Secuencia de interpret(CicloWhileDo)



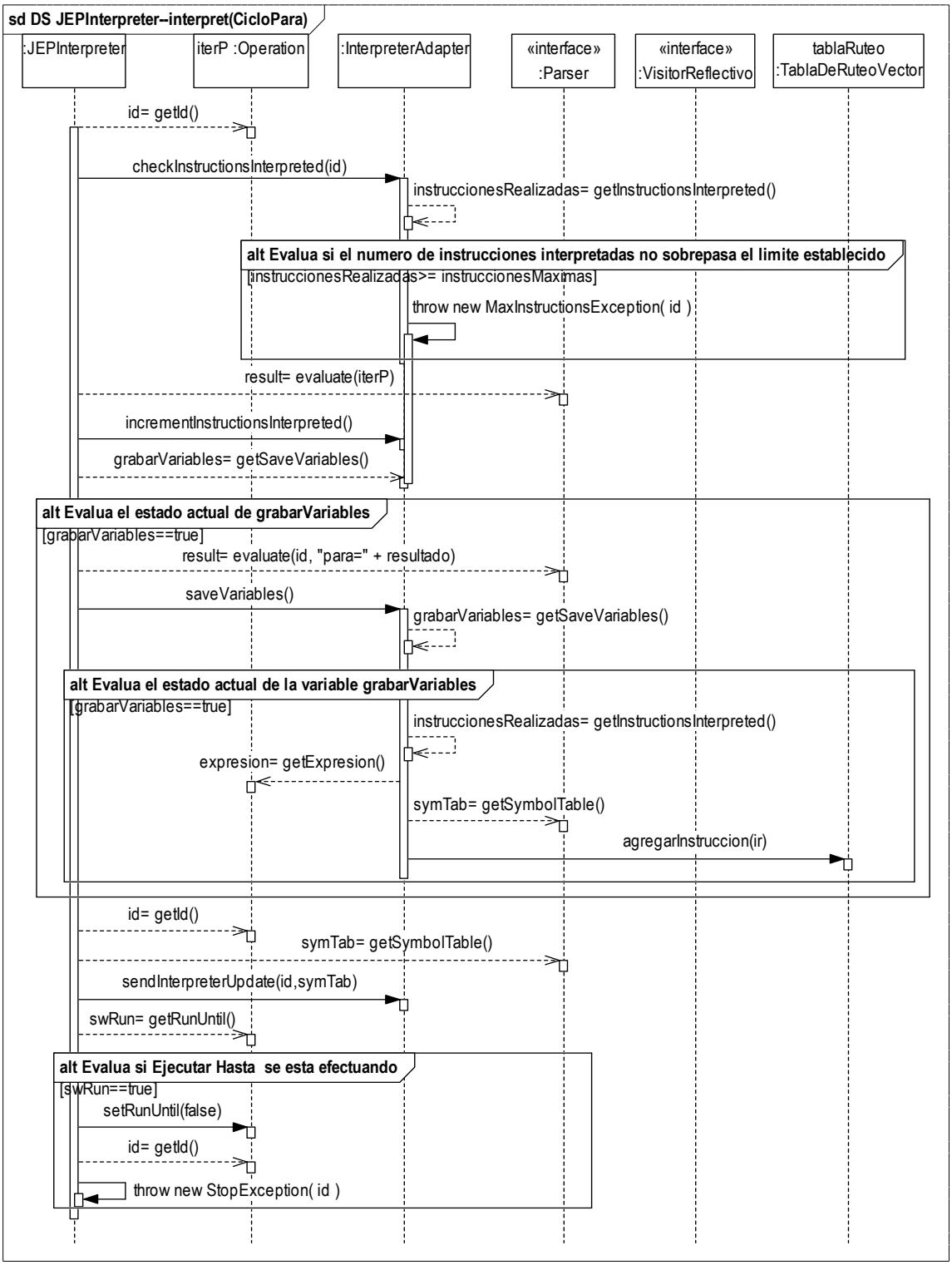

sd DS JEPInterpreter--interpret(CicloWhileDo) I
:JEPInterpreter
iterM :Operation
:InterpreterAdapter
«interface»
:Parser
«interface»
:VisitorReflectivo
alt Evalua el resultado retornado por la operacion ealuate(iterM)
[resultado == 1]
operaciones= getOperaciones()
alt Evalua si la lista operaciones no es nula y si hay mas de un elemnto
[(operaciones != null) && (operaciones.size() >0]
operaciones= getOperaciones()
loop mientras que existan mas elementos en el iterador de la lista operaciones
[it.hasNext()==true]
acept(this)
visit(this)
visit(this)
acept(this)
visit(this)
visit(this)
[resultado != 0]
id= getId()
expresion= getExpresion()
throw new EvaluationException("Condición inválida: " +expresion,id)

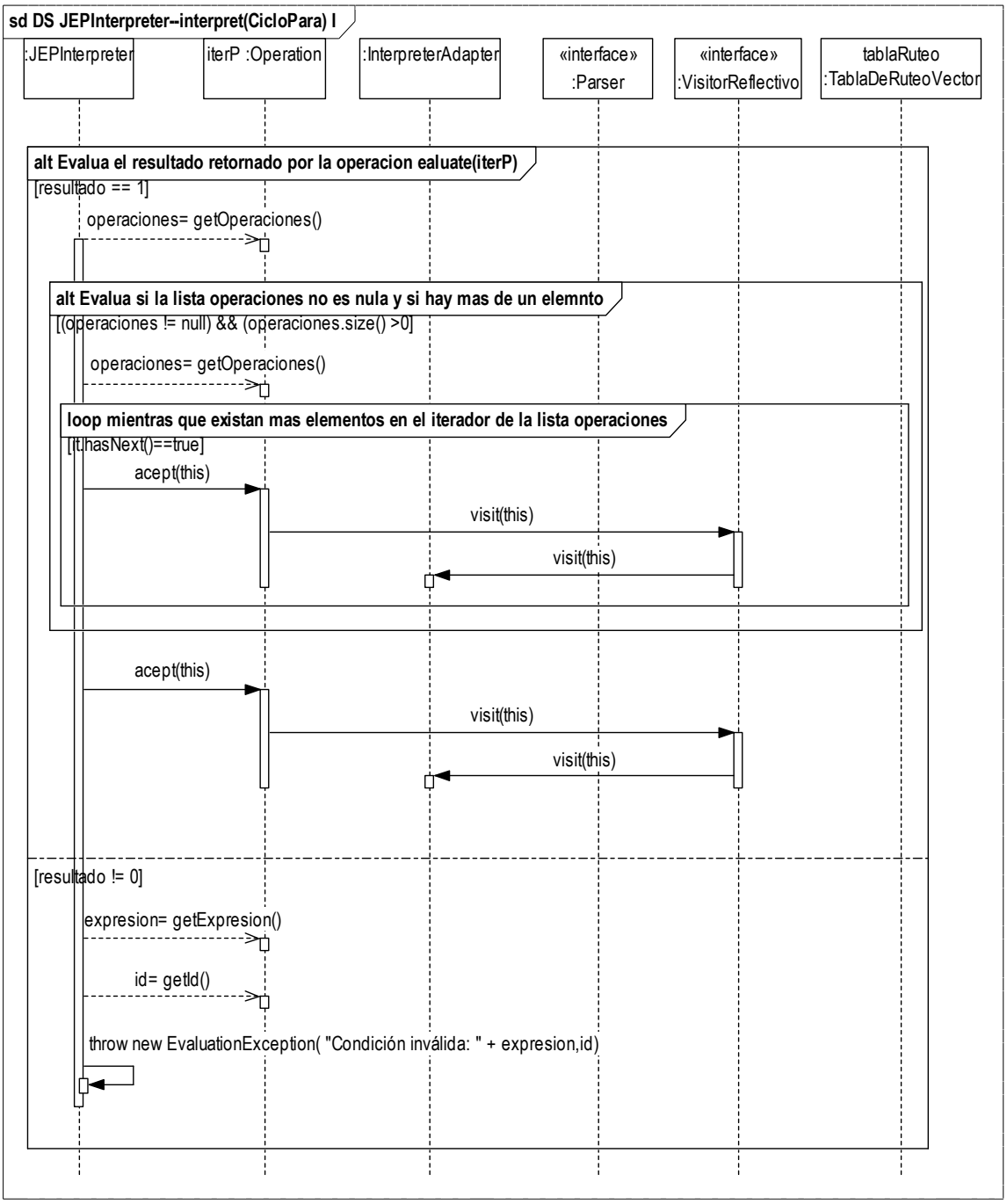
10.13.7 Diagrama de Secuencia de interpret(CicloDoWhile)



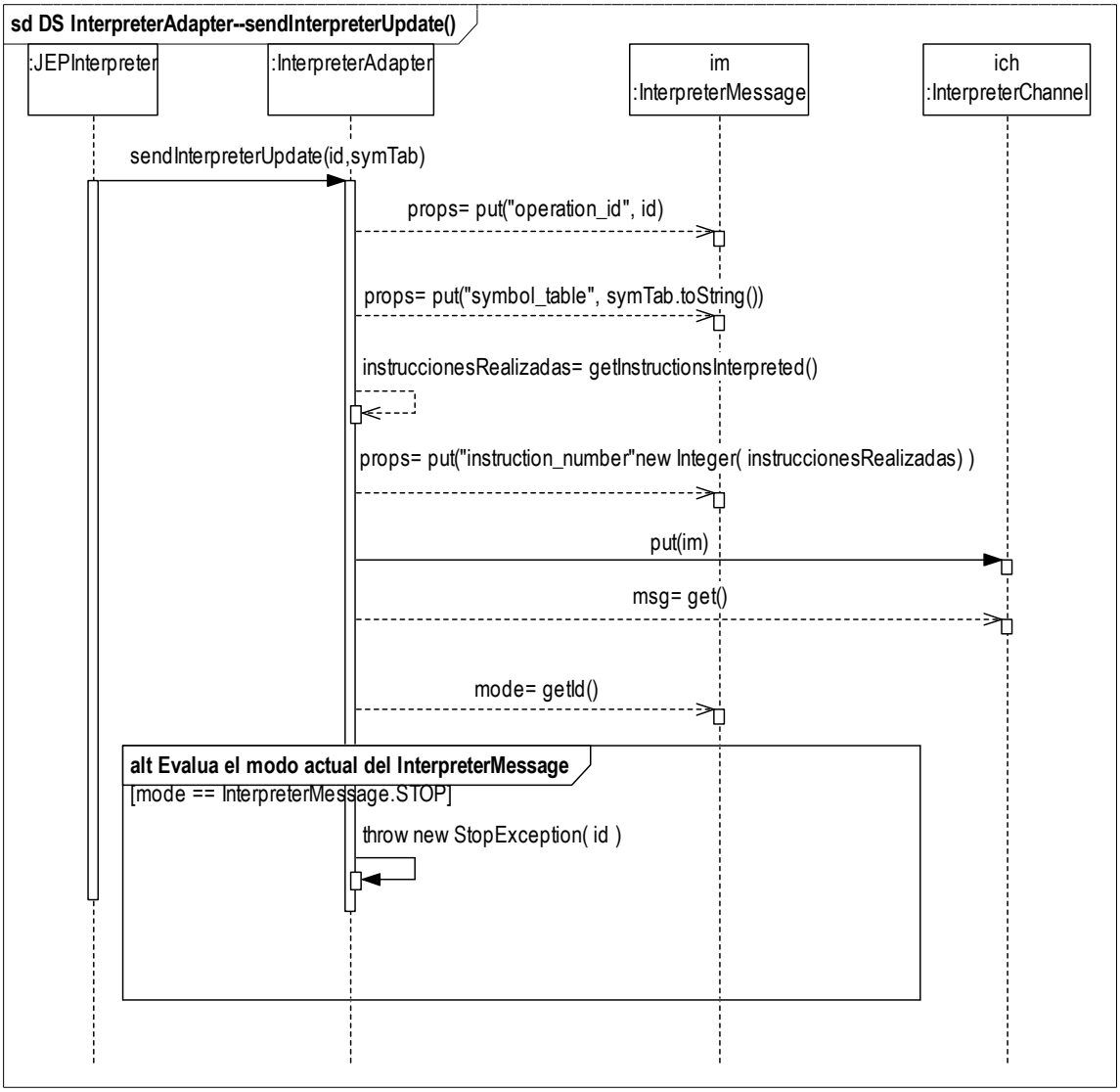

sd DS JEPInterpreter--interpret(CicloDoWhile) I
:JEPInterpreter
iterH
:Operation
:InterpreterAdapter
«interface»
:Parser
«interface»
:VisitorReflectivo
id= getId()
symTab= getSymbolTable()
sendInterpreterUpdate(id,symTab)
swRun= getRunUntil()
alt Evalua si Ejecutar Hasta se esta efectuando
[swRun==true]
setRunUntil(false)
id= getId()
throw new StopException(id)
alt Evalua que la condicion del ciclo DoWhile no sobrepase el limite establecido para iteraciones
[doWhileOrRepeatUntil == ChapinPreferencias.DO_WHILE]
alt Evalua el resultado retornado por la operacion evaluate(iterH)
[resultado==1]
acept(this)
visit(this)
visit(this)
[resultado != 0]
id= getId()
expresion= getExpresion()
throw new EvaluationException("Condición inválida: " + expresion,id)
[doWhileOrRepeatUntil != ChapinPreferencias.DO_WHILE]
alt Evalua el resultado retornado por la operacion evaluate(iterH)
[resultado==0]
acept(this)
visit(this)
visit(this)
[resultado != 1]
expresion= getExpresion()
id= getId()
throw new EvaluationException("Condición inválida: " + expresion,id)

10.13.8 Diagrama de Secuencia de interpret(CicloPara)




sd DS JEPInterpreter--interpret(CicloPara) I
:JEPInterpreter
iterP :Operation
:InterpreterAdapter
«interface»
:Parser
«interface»
:VisitorReflectivo
tablaRuteo
:TablaDeRuteoVector
alt Evalua el resultado retornado por la operacion ealuate(iterP)
[resultado == 1]
operaciones= getOperaciones()
alt Evalua si la lista operaciones no es nula y si hay mas de un elemnto
[(operaciones != null) && (operaciones.size() >0]
operaciones= getOperaciones()
loop mientras que existan mas elementos en el iterador de la lista operaciones
[it.hasNext()==true]
acept(this)
visit(this)
visit(this)
acept(this)
visit(this)
visit(this)
[resultado != 0]
expresion= getExpresion()
id= getId()
throw new EvaluationException("Condición inválida: " + expresion,id)

10.13.9 Diagrama de Secuencia de sendInterpreterUpdate() de InterpreterAdapter

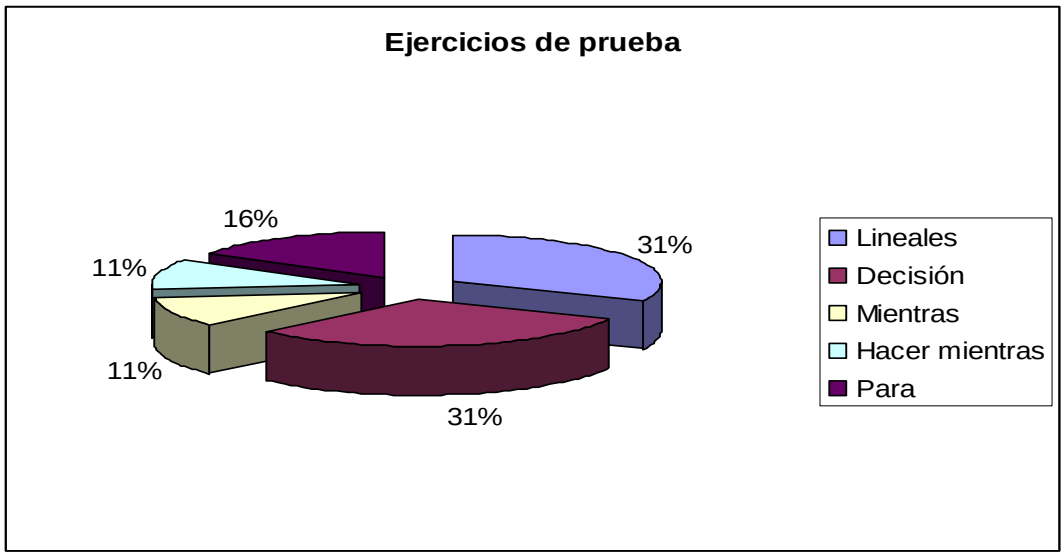


11. PRUEBAS Y RESULTADOS

Para el programa **"ICD-CHAPIN"** se han realizado diferentes pruebas en cuanto a la utilización y manejo de los algoritmos en la materia de Fundamentos de programación, utilizando la metodología de Diagramas N-S o también conocidos como diagramas de chapin.

En general se obtuvo un promedio de prueba de 4 diagramas o algoritmos por cada una de las estructuras, obteniendo resultados satisfactorios al no presentarse ningún error. Los ejemplos utilizados para las respectivas pruebas fueron tomadas del libro titulado: **ALGORITMOS: Pseudocódigo, Diagrama de flujo, Diagrama N-S** de la autoría del Ing. Anívar Chaves Torres, asesor de esta investigación, quien autorizó tomar fragmentos de texto para la ayuda del programa como también realizar las pruebas.[1]

En porcentaje se tiene que cada una de las estructuras se probó con un número de algoritmos expresados en el siguiente gráfico, para un total de 19 ejercicios que representarían el 100 por ciento de los algoritmos representados e interpretados en el software desarrollado:



En

[1] CHAVES TORRES, Anivar.Algoritmos Pseudocodigos, Diagramas de Flujo y Diagramas N-S. San Juan de Pasto, Colombia. Multigráfico impresores.2004.297p.

cuanto la forma de representar los algoritmos, el más común es que en el software la estructura escribir, no tiene la opción de sólo escribir un mensaje a la hora de interpretar un algoritmo si no que se debe escribir el nombre de la variable y después el texto opcional, o simplemente la variable; otra aclaración es que no existe declaración de variables con el comando Expresión, si no que depende de la asignación que se realice por parte del usuario, el programa toma el contenido y lo liga a esa variable, manejando por defecto dos tipos de variables: Real y cadena, las cuáles contienen a todos los demás tipos. Por el contrario el comando Leer, si maneja el tipo de variable para su respectiva utilización y generación de código.

Al momento de probar la interpretación de los ciclos, se genera un error de sintaxis cuando se trata de realizar la repetición de un ciclo demasiadas veces, esto se debe al paquete que implementamos denominado Parser JEP en su versión 2.3.0 del 3 de Octubre del año 2004, el cuál maneja tipos de datos **double** y en ejemplos como la serie de los 1000 primeros fibonnacci, se debe realizar la suma de dos números que superan la cantidad soportada por este tipo de dato; el Parser JEP se encarga de realizar las operaciones y comparaciones. De igual forma para el algoritmo de los números primos y del factorial. La solución a este problema está en la generación de código en cualquiera de los dos lenguajes para los cuáles se desarrolló esta función y luego compilar y ejecutar dicho archivo en el lenguaje y editor adecuado.

Se debe tener en cuenta que al momento de generar código y guardar el archivo, se crean dos caracteres especiales al principio de cada código generado, que se deben quitar o suprimir al abrirlo en el editor adecuado para el lenguaje; estos caracteres aparecen debido a que Java al momento de guardar un archivo, escribe una cadena de caracteres en formato UTF-8*; de los cuáles los dos primeros bytes especifican el número de bytes escritos y Java utiliza estos dos bytes para leer correctamente los archivos.[1]

De acuerdo con lo anterior, el margen de error es muy reducido teniendo en cuenta que los puntos nombrados sólo son de manejo de algoritmos y en el programa ICD-CHAPIN se pueden corregir con lo explicado anteriormente.

El parser o analizador implementado en el proyecto es uno que está disponible de manera gratuita en internet: JEP (Java Expression Parser). El JEP es una

---

[1] CEBALLOS SIERRA, Francisco Javier. Java 2 Curso de Programación. México, D.F. Alfaomega. 2000. 778 p.

* 8-bit Unicote Transformation Format.(UTF-8)

biblioteca de Java para analizar y evaluar expresiones matemáticas. Con este paquete se puede permitir que usuarios entren en una fórmula arbitraria como una cadena, y al instante evaluarlo. El JEP reconoce variables definidas del usuario, constantes y funciones. Varias funciones matemáticas comunes y constantes son incluidas. En cuanto a la gramática que se utiliza, es la misma que utiliza el JEP, la cuál se denomina JavaCC (Java Compiler Compiler) que son propiamente funciones que leen dependiendo de un parámetro que es pasado a JavaCC y de acuerdo al código de Java, un archivo o una línea escrita en ese momento. Dicha gramática puede trabajar con la máquina virtual 1.2 y superiores, además de que la especificación del lenguaje abarca tanto el aspecto léxico como el sintáctico y utiliza analizadores sintácticos descendentes de tipo LL (k) o por defecto LL (1).

En el desarrollo de un archivo de ayuda para el proyecto ICD-CHAPIN, se utilizó el paquete JavaHelp, que es muy completo debido a que facilita el desarrollo de ayuda on-line (formato HTML), es independiente de la plataforma (debido a que es implementado en Java) y permite generar una base de datos para realizar una búsqueda de palabras en la ayuda.

El formato utilizado para guardar los diagramas en forma de archivo plano, es haciendo una cadena en lenguaje XML, luego se comprime el archivo con las clases de java en el paquete **java.util.zip** de las cuáles utilizamos las clases ZipEntry, ZipOutputStream y por último se codifica en formato UTF8. La extensión de los archivos para el software ICD-CHAPIN es ***.dns** (Diagramas N-S) Para recuperar los diagramas se utiliza el paquete org.w3c.dom, con las clases necesarias para recuperar los datos y volver a dibujar el algoritmo.
Se implementaron las estructuras más utilizadas en programación para el desarrollo de cualquier algoritmo como son secuenciales, de decisión e iterativas.

En cuanto al proyecto desarrollado ICD-CHAPIN, mejora al Nessi, en primer lugar por la interfaz gráfica de usuario, la cuál es más llamativa, en referencia a iconos y botones; se puede cambiar la apariencia en tiempo de ejecución, mover un diagrama realizado y colocarlo en cualquier punto del lienzo; implementa la estructura repetitiva PARA; el zoom tiene una escala definida de manera que sea visible para el usuario. Por otro lado tiene la opción para ejecutar o interpretar un diagrama hasta un punto determinado (breakpoint), con el fin de encontrar posibles errores. Además de lo anterior, se desarrolló un módulo para generar código tanto para lenguaje C como para lenguaje Java, de acuerdo al diagrama que esté presente en pantalla, teniendo la posibilidad de edición (cortar, pegar, eliminar), imprimir el código generado, y guardarlo con la extensión propia del lenguaje elegido por el usuario. Otra mejora del proyecto ICD-CHAPIN, es la implementación de una ayuda didáctica, que consiste en Demos para que el usuario observe la manera adecuada de manejar el software y se relacione más

con el mismo; finalmente se tiene la ventana de presentación inicial del programa denominada ventana Splash.

El programa también fue probado en la plataforma Linux, con Ubuntu versión 7.04, y se obtuvo los mismos resultados, aunque existen algunas excepciones, las cuales son:

- En cuanto a la configuración de la apariencia, solo funciona la apariencia Unix y la apariencia metálica propia de Java.

- En cuanto a la configuración de los diagramas, existe inconvenientes con el tipo de letra que se vaya a cambiar, ya que varias fuentes no están presentes en dicha plataforma.

- Los demos no funcionan en Linux, ya que son en formato **.exe** no compatible con el sistema operativo Linux.

12. CONCLUSIONES

El proyecto desarrollado ICD-CHAPIN es una herramienta informática para diseñar, interpretar diagramas N-S o de Chapin y posteriormente generar código en lenguaje Java o C++, siendo un gran apoyo para estudiantes y profesores al momento de probar los algoritmos antes de ser codificados.

En la creación del proyecto ICD-CHAPIN se tuvo en cuenta varios puntos, como es la forma de dibujar los diagramas, que se realiza de manera atractiva al usuario, en otras palabras el software es didáctico en la realización de cualquier tipo de algoritmo que no exijan mucha complejidad.

En la forma de evaluar las expresiones de un diagrama realizado por el usuario se utiliza el JEP el cuál utiliza la gramática JavaCC. (Ver capítulo 10: Pruebas y resultados).

También la manera en que los diagramas se vuelven persistentes en un medio de almacenamiento, es decir que se transformen en un archivo para poder ser transportado y manejado de manera más fácil, es muy importante debido a que se puede guardar y luego recuperar un archivo o diagrama. (Ver capítulo 10: Pruebas y resultados).

Se utilizó el paquete JavaHelp para generar archivos de ayuda del software, este permite crear de manera rápida y gráfica, las ayudas para cualquier tipo de programa.

El generar código a partir de un diagrama, permite al usuario tener una perspectiva diferente del algoritmo implementado, teniendo la posibilidad de conocer más acerca de los lenguajes de programación C y Java (sintaxis respectiva), en los cuáles se ha hecho énfasis, sirviendo de avance para estudiantes en la rama de programación en el programa de ingeniería de sistemas.

Java y su máquina virtual es una herramienta de desarrollo de software, que permite ejecutar la aplicación en diferentes plataformas, permitiendo así adecuarse de acuerdo a las características que el usuario prefiera. De igual

manera, la utilización del entorno de desarrollo integrado NetBeans, en su versión 5.5, facilita el desarrollo del software en gran parte de modo que la interfaz gráfica se realiza de forma más rápida y sin complicaciones.

La utilización de UML en el desarrollo del proyecto, permitió visualizar y definir de forma clara cada componente del sistema para su adecuada construcción, además de poder documentar este proceso, lo cual es de gran ayuda para tener una visión clara del proyecto en si, y definir los puntos clave para su desarrollo.

Al ser Open Source y licencia GPL, se puede realizar modificaciones y redistribuciones del proyecto, acoplándolo a futuros requerimientos y nuevas investigaciones.

13. RECOMENDACIONES

Aprovechar las características y funcionalidad de la aplicación ICD-CHAPIN, para fomentar el uso de los diagramas N-S, y de esta manera tener más opciones al momento de solucionar un problema.

Aprovechar las demostraciones didácticas que tiene el software para la buena utilización del programa, de forma que los errores en su manejo sean los mínimos y pueda desempeñarse correctamente.

Fomentar la utilización de diagramas N-S ya que con esta herramienta gráfica, se puede realizar un buen estudio y análisis a la hora de aprender la solución de ejercicios por medio de algoritmos.

Fomentar la creación de un grupo de investigación para que trabaje en la ampliación del software ICD-CHAPIN, en aspectos tales como la implementación de arreglos unidimensionales (vectores), bidimensionales (matrices), implementación de subprogramas (funciones) y la implementación del compilador de C, para que el código generado sea compilado directamente desde el IDE que presenta el programa. En cuanto al manejo que tiene el Parser JEP en el manejo de Arreglos, este paquete tiene clases especiales para realizar las operaciones con arreglos, por ejemplo, se debe enviar dos vectores y la operación y el Parser JEP se encarga de ejecutar esta operación y devuelve el resultado de la operación indicada o el error en caso de que no se pueda realizar la solicitud especifica.

BIBLIOGRAFIA

BAASE, Sara & GELDER, Allen Van. Algoritmos computacionales Introducción al análisis y diseño 3era. Edición. México D.F. Pearson Educación. 2002. 704 p.

BOOCH, Grady. Análisis y diseño orientado a objetos con aplicaciones 2da. Edición. México D.F. Pearson Educación. 1996. 638 p.

BOOCH, Grady & RUMBAUGH, James & JACOBSON, Ivar. El lenguaje unificado de modelado UML. Madrid, España. Addison Wesley Iberoamericana.1999. 464 p.

CAIRÓ, Oswaldo. Metodología de la programación: Algoritmos, diagramas de flujo y programas Tomo 2. México D.F. Alfaomega. 1010 p.

CEBALLOS SIERRA, Francisco Javier. Java 2 Curso de Programación. México, D.F. Alfaomega. 2000. 778 p.

CHAVES TORRES, Anivar.Algoritmos Pseudocodigos, Diagramas de Flujo y Diagramas N-S. San Juan de Pasto, Colombia. Multigráfico impresores.2004.297p.

FERNANDEZ SASTRE, Sergio. Fundamentos del diseño y la programación orientada a objetos. Madrid, España. Prentice Hall. 145 p.

INSTITUTO COLOMBIANO DE NORMAS TÉCNICAS. Normas Colombianas para la presentación de trabajos de investigación. Quinta Actualización. Santa Fe de Bogotá, D.C. ICONTEC. 2004. 126 p. NTC 1486.

JOYANES AGUILAR, Luís. Fundamentos de Programación, Algoritmos y estructuras de datos. Santa Fe de Bogotá D.C. McGraw-Hill. 1998. 714 p.

MARTIN, James & ODELL, James. Análisis y diseño orientado a objetos. México. Prentice Hall. 1994. 546 p.

PERDITA STEVENS, Rob Pooley. Utilización de UML en Ingeniería del software con objetos y componentes. Madrid, España. Addison Wesley. 2002. 300 p.

SISA, Jaime. Estructura de datos y algoritmos con énfasis en programación orientada a objetos. Bogotá D.C. Pearson Educación de Colombia Ltda. 2002. 328 p.

Anexo A

MANUAL DE USUARIO

1. INTRODUCCIÓN

Este es el manual de usuario del programa ICD-CHAPIN, cuyo software tiene como objetivo ayudar en las bases de programación y principalmente de los fundamentos de la programación en el programa de Ingeniería de Sistemas y demás áreas que lo requieran.

Este manual es de mucha importancia para adentrarse en el manejo del software y el conocimiento de los algoritmos de Nassi-Shneiderman para el desarrollo de un determinado problema.

En particular, cuando se experimenta por primera vez el uso de algoritmos, es un poco difícil y teniendo esta herramienta como ayuda principal, no sólo se podrá dibujar, si no también interpretar el algoritmo, obteniendo el resultado de que el algoritmo en realidad funciona correctamente o no.

Ahora se espera la buena utilización de esta herramienta y el deseo de que la forma de aprender este tipo de algoritmos sea de una mejor manera y sobretodo que se comprenda de forma más rápida.

2. REQUERIMIENTOS MINIMOS PARA UN BUEN FUNCIONAMIENTO

- Sistema operativo: Microsoft Windows, Linux
- Dispositivos de Entrada: teclado y Mouse
- Máquina virtual Java 1.5.0 o superiores

- Memoria Ram mínima de 256 Mb y espacio en disco de 12 Mb.

3. INSTALACIÓN DEL PROGRAMA

Para comenzar a instalar el software denominado Intérprete de comandos para diagramas de Chapin (N-S) **"ICD-CHAPIN"**, se debe dar clic en el instalador de nombre **ICD-CHAPIN.exe**, que se presenta de la siguiente forma:

Luego de realizar el paso anterior, lo que aparece es la siguiente pantalla de inicio del instalador, en donde se observa el nombre del programa, los nombres de los desarrolladores y demás datos básicos:

Es necesario recordar que esta ventana está vigente sólo por 4 segundos y que el fondo para el demás proceso de instalación será el mismo. Luego se pasa a la

ventana en donde se puede leer la licencia, el usuario debe marcar la casilla de Acepto los términos del contrato, sólo así podrá habilitar el botón de siguiente para poder continuar:

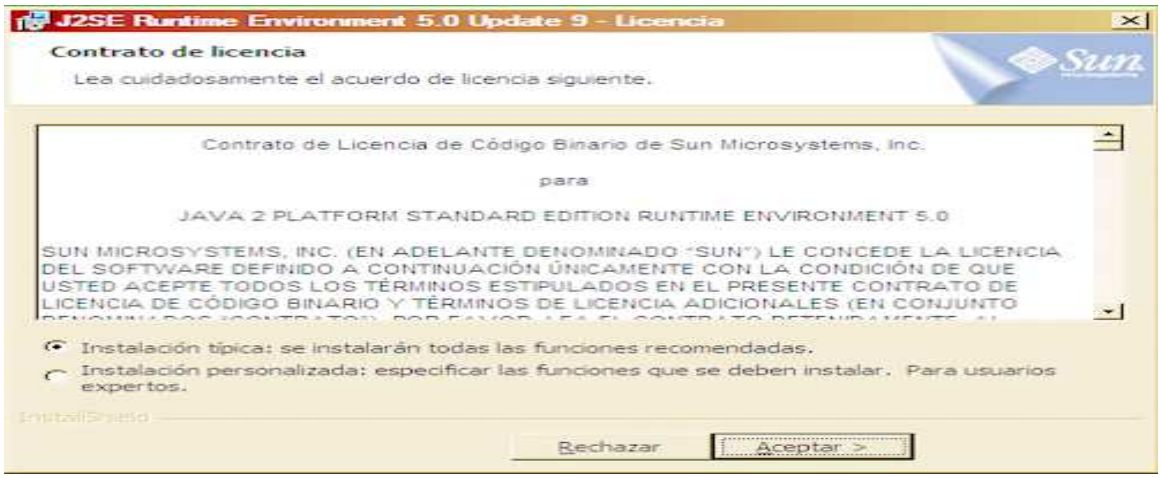
Luego de pulsar el botón siguiente lo que sigue es leer los agradecimientos, los autores correspondientes y dar clic en el botón siguiente:

En seguida se puede mirar la ruta en donde se va a instalar y mirar el espacio que se requiere para que el software sea instalado, además se debe observar el tamaño disponible en la unidad **C:** que se ha tomado como predeterminada:

Si se pulsa el botón comenzar el software se empezará a instalar en la ruta por defecto, indicando los archivos que se están copiando y el porcentaje en la barra de progreso. Cabe destacar que en cualquier momento se puede salir de la instalación o retroceder uno o varios pasos atrás:

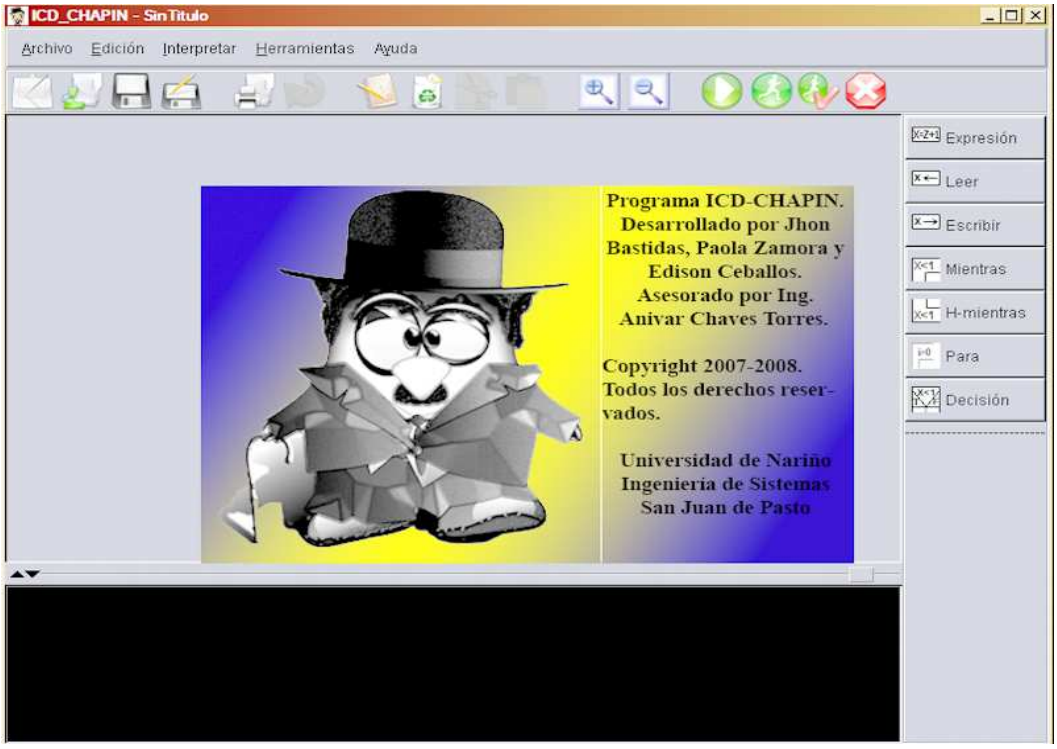
Si se da clic en el botón Aceptar presenta la siguiente ventana, en donde se puede seleccionar o no las opciones de ver el archivo Léeme, ejecutar la aplicación instalada y crear acceso directo del software en el escritorio. Es necesario dejar instalar la máquina virtual (si no se tiene), que es indispensable para que la aplicación funcione, su instalación comenzará de manera automática:

Este es el último paso para instalar el programa ICD-CHAPIN lo que sigue ahora es instalar la máquina virtual, se deja las opciones seleccionadas por defecto y se da clic en el botón Aceptar, para comenzar su instalación en la siguiente ventana:



La última ventana es la siguiente luego de esperar un momento mientras se instala y finalmente pulsando el botón Finalizar de la siguiente ventana:

La aplicación inicia con la siguiente pantalla, siendo una interfaz muy amigable y agradable al usuario:



4. MANEJO DEL SOFTWARE

El diagrama N-S cuenta con un conjunto limitado de símbolos para representar los pasos del algoritmo, por ello se apoya en expresiones del lenguaje natural; sin embargo, dado que el lenguaje natural es muy extenso y se presta para la ambigüedad, solo se utiliza un conjunto de palabras, a las que se denomina palabras reservadas. Las palabras reservadas más utilizadas son:

Inicio, Fin, Leer, Escribir
Mientras, Repita Hasta, Para
Incrementar, Decrementar, Hacer, Función
Entero, Real, Caracter, Cadena
Lógico, Retornar

Los símbolos utilizados en el diagrama de Chapin corresponden a cada tipo de estructura. Dado que se tienen tres tipos de estructuras, se utilizan tres símbolos. Esto hace que los procesos del algoritmo sean más fáciles de representar y de interpretar.[1]

4.1 ESTRUCTURAS SECUENCIALES

Son aquellas que se ejecutan una después de otra. Se tienen tres tipos de instrucciones secuenciales: la declaración de variables, asignación, instrucción Leer e instrucción Escribir.

4.1.1 Declaración de variables: Teniendo en cuenta la compatibilidad con la mayoría de los lenguajes, se recomienda que desde el diseño del programa se utilice una forma determinada para la declaración de las variables. Esta consiste en escribir el tipo de datos y la lista de identificadores que se tendrán de dicho tipo, separando cada identificador por medio de comas (,). Para mejorar la claridad de la declaración se puede colocar dos puntos (:) para separar el tipo de datos de la lista de identificadores.

Ejemplo:

[1] CHAVES TORRES, Anivar. Algoritmos Pseudocodigos, Diagramas de Flujo y Diagramas N-S. San Juan de Pasto, Colombia. Multigráfico impresores.2004.297p.

Entero: edad
Real: estatura, peso, sueldo
Cadena: nombre, direccion

En el programa denominado ICD-CHAPIN, la declaración de variables se realiza por medio de la barra de dibujo y el comando Expresión (por ejemplo, variable=0) y el comando leer, en donde se puede escoger el tipo de la variable y el identificador.

4.1.2 Asignación: Asignar un valor a una variable equivale a decir que se guarda dicho valor en la posición de memoria reservado para la variable en mención. Por lo tanto, para poder realizar una asignación es necesario primero haber declarado una variable, con lo cual se reserva un espacio de memoria suficiente para guardar un dato del tipo especificado.

Una expresión de asignación tiene la forma:

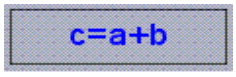
Variable = expresión

Una asignación tiene tres partes, una variable, el signo igual y la expresión cuyo valor se asigna a la variable.

Ejemplo:

```
Entero: X, Y          //Declaración de variables X, Y
X = 10
Y = X * 2 + 8
```

En este ejemplo, la variable Y contendrá el valor 28. Para realizar una asignación se debe ir a la barra de dibujo, presionar el comando Expresión y seleccionar con el ratón la ubicación del símbolo, para luego escribir por ejemplo:



4.1.3 Instrucción Leer: La instrucción LEER se utiliza para enviar información desde un dispositivo de entrada de datos hacia la memoria. En la memoria los datos son ubicados mediante el identificador (nombre de variable) utilizado como complemento de la instrucción LEER. En diagrama N-S la instrucción de entrada se representa así:

Leer < lista de identificadores de variables >

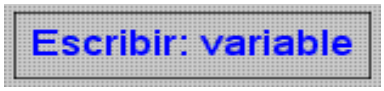
Esta instrucción se realiza dando clic en el comando Leer de la barra de dibujo, en donde se elige el tipo y el identificador de la variable, en la ventana que se despliega luego de dar clic en el diagrama para ubicar la estructura.

Ejemplo:

Leer: a, b
Donde "a" y "b" son las variables que recibirán los valores.

4.1.4 Instrucción Escribir: Esta instrucción permite enviar datos desde la memoria hacia un dispositivo de salida como la pantalla o la impresora. La información que se envía puede ser constante o también el contenido de variables.

Escribir < lista de constantes y variables >
Ejemplo: Escribir a, b



Cuando se escriben dos variables es necesario separarlas con comas (,) y los mensajes se escriben entre comillas dobles " ". Si una variable es escrita entre comillas se mostrará el identificador y no el contenido.

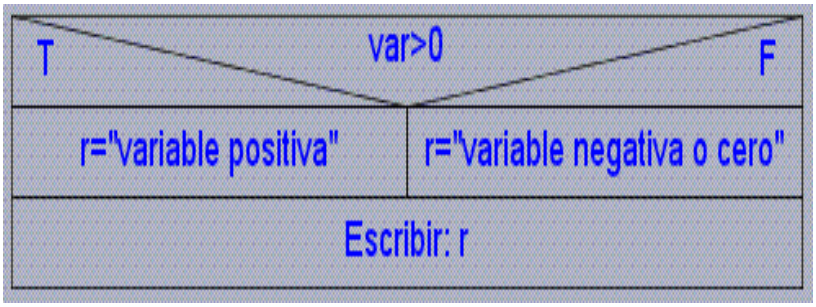
En el caso del programa, sólo se puede escribir una variable o la variable seguido de una coma **“,”** y el mensaje entre comillas dobles (**por ejemplo: num,“El resultado es:**”). Esta instrucción se la realiza en el comando Escribir de la barra de dibujo, en donde aparece una ventana para llenar la expresión.

4.2 ESTRUCTURAS DE DECISIÓN

Las estructuras de decisión o también llamadas de selección permiten que el algoritmo tome decisiones y ejecute u omita algunos procesos dependiendo del cumplimiento de una condición. Se puede manejar tres tipos de decisiones: simple, doble y múltiple. En el software no se maneja la decisión múltiple (o también llamados switch).

4.2.1 Decisión simple y doble: una decisión es simple, cuando solo se tiene determinado los pasos a seguir si el resultado de la condición es verdadero, mientras que si es falso no hace nada y el algoritmo continúa después de la estructura condicional.Una decisión es doble cuando se tiene un curso de acción para el caso que el resultado de la comparación sea verdadero y otro para cuando sea falso.

En diagrama de Chapin el símbolo para representar una decisión es el siguiente:



El software ICD-CHAPIN, permite realizar este tipo de estructuras, con el comando Decisión de la barra de dibujo, para luego llenar la decisión en una ventana que nos presenta, en el gráfico anterior se presenta como ejemplo la decisión de que si una variable es mayor que cero (var>0) y las dos alternativas para indicar un mensaje: verdadero y falso.

4.3 ESTRUCTURAS DE REPETICIÓN

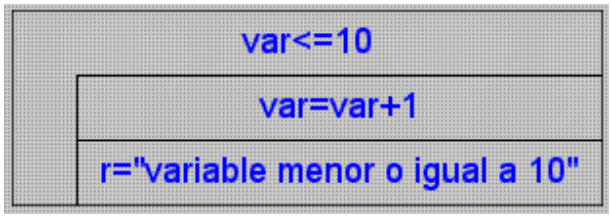
En la solución de algunos problemas es necesario ejecutar repetidas veces una instrucción o un conjunto de instrucciones. En algunos casos, el número de repeticiones se conoce con anterioridad, mientras que en otras depende de cálculos o estados de variables que se dan dentro de la solución del problema. Un ciclo consiste en un grupo de acciones que se ejecutan repetidas veces dependiendo del cumplimiento de una condición.

Para insertar un ciclo en el diagrama que se esté diseñando, se utiliza el comando Mientras y de igual forma el comando H-Mientras, ubicados en la barra de dibujo, los dos nos presenta una ventana para llenar la decisión, la diferencia es que según el resultado de la comparación se repiten las instrucciones; entonces, el ciclo mientras se repite cuando la condición sea verdadera y el ciclo Hacer mientras se repite cuando la condición sea falsa.

4.3.1 Ciclo Mientras: Este ciclo consiste en un conjunto de instrucciones que se repiten mientras se cumpla una condición. De igual manera que en los condicionales, la condición es evaluada y retorna un valor lógico, que puede ser verdadero o falso. En el caso del ciclo mientras las instrucciones contenidas en la estructura de repetición se ejecutarán solamente si al evaluar la condición se genera un valor verdadero; es decir, si la condición se cumple; en caso contrario, se ejecutará la instrucción que aparece después de Fin mientras.

A diferencia de otros ciclos, el ciclo mientras comienza evaluando la expresión condicional. Si en la primera pasada por el ciclo mientras la condición no se cumple las instrucciones que están dentro del ciclo no se ejecutarán ni una sola vez.

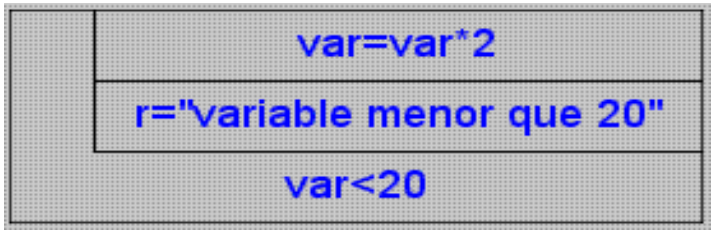
En diagrama de Chapin (N-S), esta estructura tiene la presentación:



4.3.2 Ciclo Hacer mientras: Este ciclo difiere del ciclo mientras en que se ejecuta como mínimo una vez las instrucciones que están dentro de él y luego compara la condición. De igual manera que en los condicionales, la condición es evaluada y retorna un valor lógico, que puede ser verdadero o falso. Las instrucciones contenidas en la estructura de repetición se ejecutarán solamente si al evaluar la condición se genera un valor Verdadero; es decir, si la condición se cumple; en caso contrario, se ejecutará la instrucción que aparece después de la estructura Hacer mientras.

Si en la primera vez que compara la condición, no es falsa por lo menos una sola vez se ha ejecutado, sin comparar la condición.

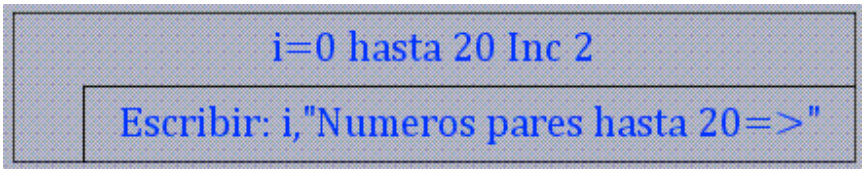
En diagrama de Chapin (N-S), esta estructura tiene la presentación:



4.3.3 Ciclo Para: a diferencia de los ciclos Mientras y Repetir hasta que, éste ciclo maneja el valor inicial, el valor final y el valor de incremento o decremento de la variable de control como parte de la instrucción.

Cuando al ejecutarse un algoritmo se encuentra una instrucción para la variable de control (contador) toma el valor inicial, se verifica que el valor inicial no sobrepase el valor final y luego se ejecutan las instrucciones del ciclo. Al encontrar la instrucción fin para, se produce el incremento y se vuelve a verificar que la variable de control no haya superado el límite admitido, y se vuelven a ejecutar las instrucciones que están dentro del ciclo, y así sucesivamente tantas veces como sea necesario hasta que se supere el valor final establecido.

En diagrama de Chapin (N-S), esta estructura tiene la presentación:



Este ciclo es más completo que los dos ciclos anteriores y puede ser implementado con el comando Para, ubicado en la barra de dibujo, en donde presenta una ventana para llenar la variable con su asignación inicial, el valor final de la variable, y el incremento o decremento de la variable.

5. INTERFAZ GRÁFICA DE USUARIO

La anterior es la ventana principal que contiene:

Área de trabajo: Es donde son dibujados los diagramas.

Consola: Es donde se presentan los resultados de la interpretación de los diagramas.

Barra de dibujo: Con esta barra usted puede dibujar las operaciones en el área de trabajo.

Menú: Seleccionando los ítems del menú le ayudará a guardar el diagrama, interpretarlo, imprimirlo, hacerle zoom, entre otros.

Barra de herramientas: Con esta barra de herramientas se puede realizar las operaciones más comunes que se encuentran en el menú.

5.1 LA BARRA DE HERRAMIENTAS

La barra de herramientas contiene los iconos con las operaciones más importantes, así:

 NUEVO: Permite crear un nuevo diagrama.

 ABRIR: Permite buscar un diagrama y cargarlo en el área de trabajo.

 GUARDAR: Permite guardar el diagrama actual o los cambios realizados.

 GUARDAR COMO: Permite guardar el diagrama actual con otro nombre.

 IMPRIMIR: Permite imprimir un diagrama.

 DESHACER: Permite deshacer la ultima acción realizada.

 EDITAR: Permite editar una estructura u operación seleccionada.

 ELIMINAR: Permite eliminar una estructura u operación seleccionada.

 CORTAR: Permite cortar una estructura u operación seleccionada.

 PEGAR: Permite pegar una estructura u operación que fue copiada.

 AUMENTAR: Permite aumentar el tamaño de un diagrama.

 DISMINUIR: Permite disminuir el tamaño de un diagrama.

 EJECUTAR TODO: Permite ejecutar todo el diagrama realizado.

 EJECUTAR PASO A PASO: Permite ejecutar de manera detenida un diagrama.

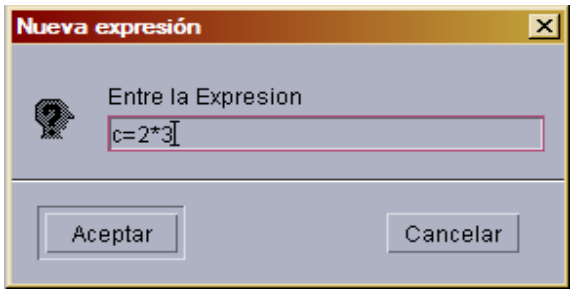
 EJECUTAR HASTA: Permite ejecutar de manera detenida un diagrama, hasta un determinado punto, seleccionado por el usuario (conocido como depurar).

 DETENER EJECUCIÓN: Detiene la ejecución de un diagrama.

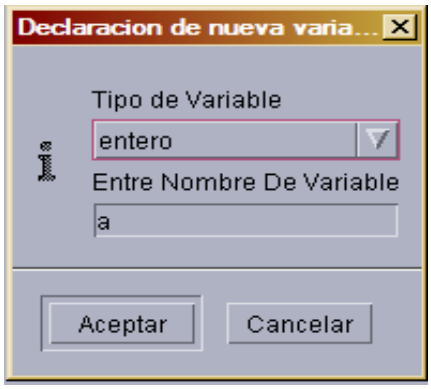
5.2 COMO DIBUJAR

El programa tiene una forma muy simple de realizar los dibujos o diagramas de los algoritmos, la barra de dibujo posee los comandos principales que se han planteado para el desarrollo de este proyecto, así:

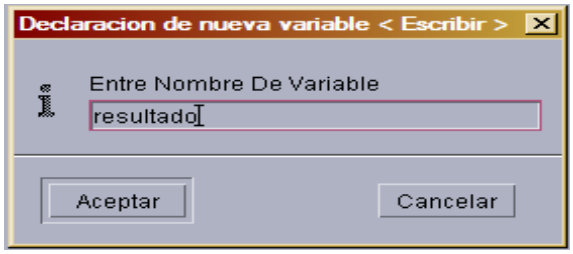
EXPRESIÓN: permite agregar una expresión de asignación al diagrama. En el programa aparece la siguiente ventana para escribir la asignación u operación por parte del usuario:



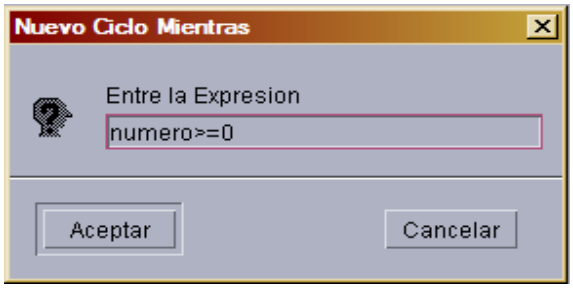
LEER: Permite agregar al diagrama la estructura leer, para que el usuario pueda entrar el valor de una variable al momento de la interpretación del diagrama, así:



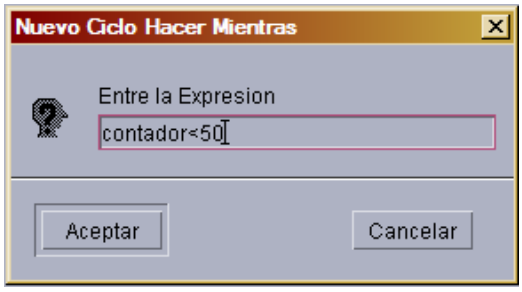
ESCRIBIR: Permite agregar al diagrama el comando escribir, se utiliza para poder presentar el valor final de una variable del algoritmo luego de un proceso y opcionalmente un mensaje.



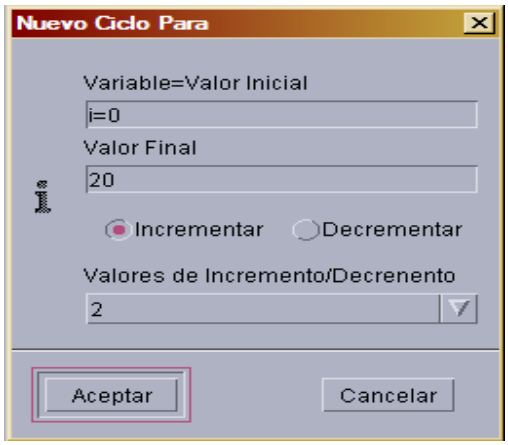
CICLO MIENTRAS: Permite agregar al diagrama el ciclo mientras, el cuál compara primero y si se cumple la condición ejecuta las acciones de lo contrario no hace nada.

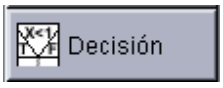


CICLO HACER MIENTRAS: Permite agregar al diagrama el ciclo hacer mientras, el cuál ejecuta una vez como mínimo las acciones y luego compara la condición.

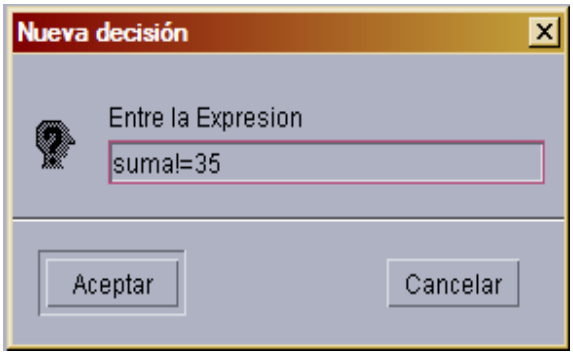


CICLO PARA: permite agregar al diagrama el ciclo para, el cuál permite colocar el valor inicial, el valor final y el incremento o decremento de la variable de control en un solo paso, y primero se compara la condición para ejecutar las instrucciones internas.





DECISIÓN: permite agregar una estructura de decisión sencilla, que consiste en que según la condición ejecuta determinadas acciones.



5.3 CONFIGURACIÓN DEL PROGRAMA

El programa ICD-CHAPIN se lo puede configurar de una manera muy cómoda y sencilla de acuerdo a la perspectiva del usuario, para acceder a la ventana de configuración del programa se debe dirigir a la barra de menú que está en la parte superior de la interfaz de usuario y seleccionar la opción Herramientas, en donde luego se despliega un submenú para elegir la opción de Configuración; otra forma más rápida de acceder a la ventana de configuración es presionando las teclas abreviadas (ctrl.+F). Esta sencilla ventana permite cambiar la apariencia del programa en cuatro formas diferentes, en donde se puede visualizar la apariencia del programa en las imágenes que se muestran de acuerdo a la opción elegida, así:

Dentro de la ventana de configuración no sólo se puede realizar el cambio de la apariencia del programa, si no también cambiar el color, el tipo o el tamaño de la fuente para los diagramas que se muestran en el área de trabajo, en esta pestaña se observa un ejemplo de los cambios que se realizan antes de aplicar la configuración al programa y como acción final podemos cancelar los cambios y el programa quedará con la configuración anterior. La ventana que se muestra es la siguiente:

6. INTÉRPRETE DE COMANDOS

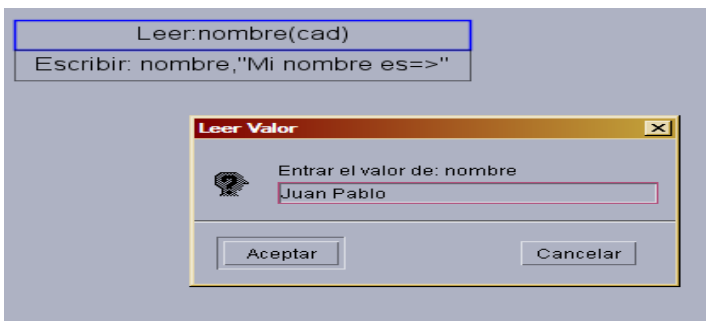
El programa ICD-CHAPIN, tiene como ventaja principal la interpretación de los algoritmos que el usuario realiza, de acuerdo con esto, el usuario puede comprobar si su algoritmo está bien o mal realizado y observar en que parte está equivocado teniendo la posibilidad de corregir posibles errores. Por otra parte, permite interpretar de tres formas diferentes un algoritmo, una herramienta que para sacar diferentes conclusiones tan sólo con el simple hecho de presionar uno de estos tres botones localizados en la barra de herramientas:

 Ejecutar todo  Ejecutar Paso a paso  Ejecutar Hasta

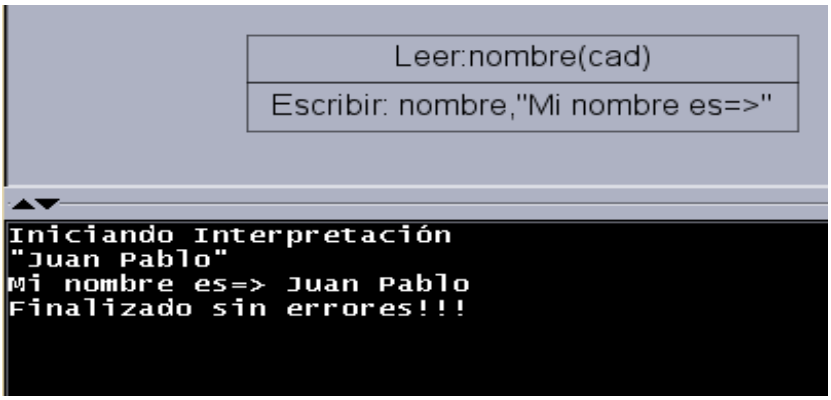
El botón ejecutar todo interpreta el algoritmo actual por completo, de una forma más rápida y sin pausas. El botón Ejecutar paso a paso interpreta el algoritmo actual de una forma pausada, indicando cada casilla por donde hace el recorrido. El botón Ejecutar hasta es muy importante debido a que el usuario escoge una estructura del algoritmo actual hasta donde se ejecutará y de forma pausada observará por donde hace el recorrido, precisamente hasta el punto elegido por el usuario, si el punto no aparece en el recorrido, entonces continúa su interpretación hasta el final del algoritmo.

Cabe destacar que en la interpretación de los algoritmos se detalla o resalta la estructura que se está interpretando con un borde de color AZUL.

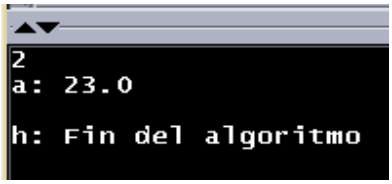
En cualquiera de los tres botones de ejecución o interpretación del algoritmo, es importante saber que la estructura de lectura, presenta una ventana para poder pedir los datos al usuario de una manera interactiva, la ventana presentada es la siguiente:



De igual manera para la estructura escribir, pero la interpretación de esta estructura aparece directamente en la consola y no en una ventana aparte, de la siguiente forma:



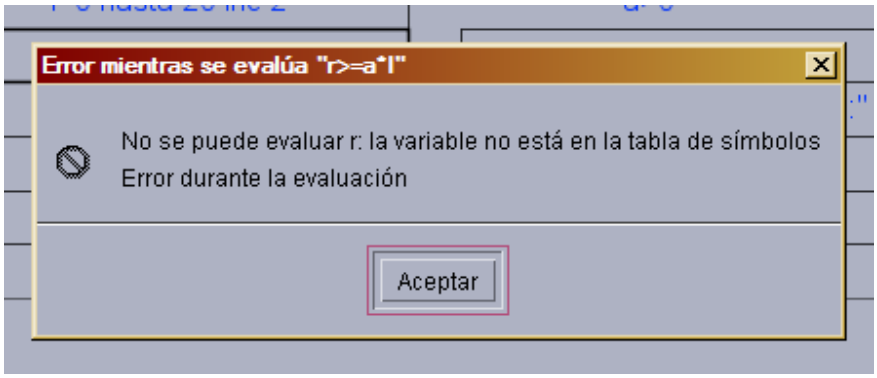
Por otra parte está la diferencia en los botones de ejecutar paso a paso y ejecutar hasta, para las demás estructuras implementadas, las cuáles solamente aparecen en la consola indicando el tipo de variable y su valor actual, para la muestra un ejemplo:



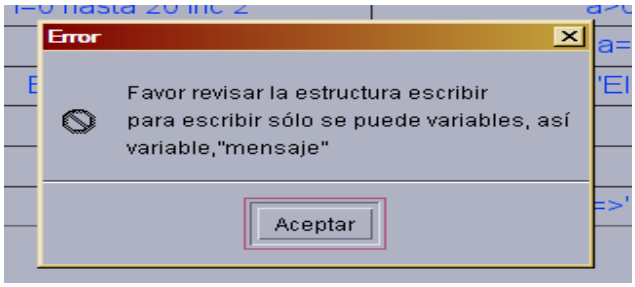
6.1 MANEJO DE ERRORES:

En la interpretación, de acuerdo al Parser JEP que ha sido desarrollado por Sun Microsystems, el programa toma la instrucción y la ejecuta, comparando si está bien o mal realizada, si por ejemplo, la instrucción está mal, la interpretación se detiene y pasa a presentar una ventana de error, existen diferentes tipos de errores entre los cuáles los más comunes y frecuentes, se presentan de la siguiente manera:

Cuando no se reconoce una variable, es decir que no haya sido inicializada o leída por el usuario, aparece:

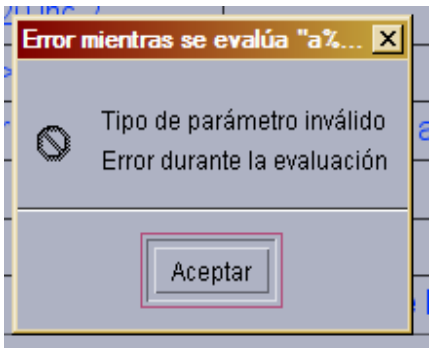


Cuando la estructura escribir está mal implementada por el usuario, aparece:

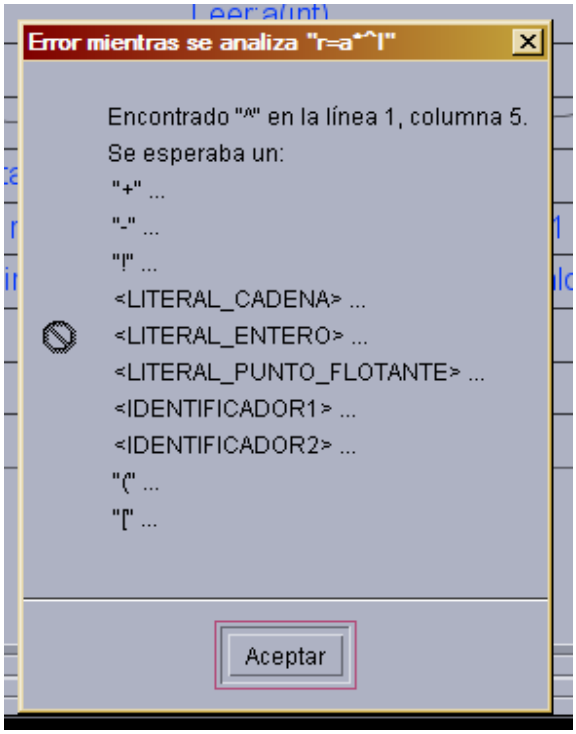


Cuando en cualquier estructura existe un error de sintaxis, aparece:

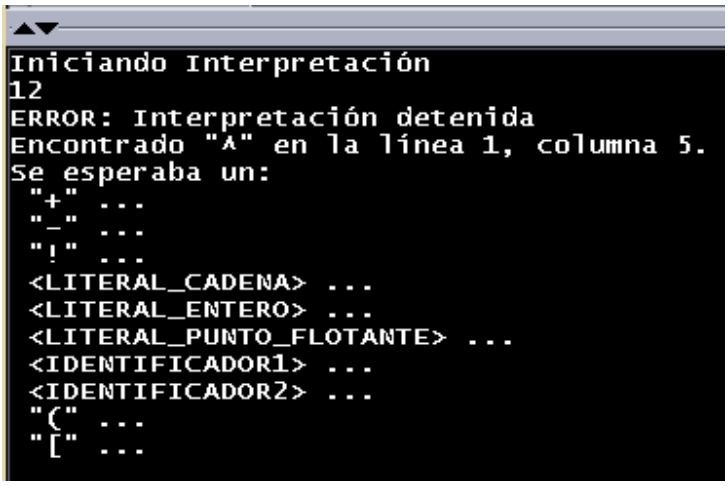
Cuando se implementan funciones básicas del Parser JEP y los parámetros no son válidos, aparece:



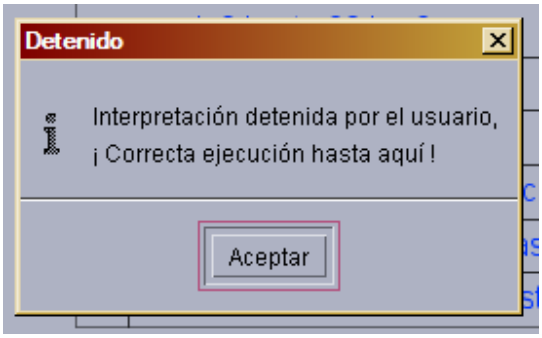
O cuando se implementa una estructura de condición o asignación y está mal escrita, aparece:



Es necesario observar que para cualquier tipo de error que se presente, el programa escribe en la consola el error y no solamente en una ventana exterior, por ejemplo para el error anterior, en consola aparecerá:



Un punto importante es el botón DETENER  el cuál detiene la interpretación en cualquier momento que el usuario desee y presenta un mensaje como el que sigue a continuación, dejando resaltado de color negro la estructura en la cuál se detuvo la interpretación:



6.2 VARIABLES QUE SE MANEJAN

Las variables que se manejan no son necesariamente inicializadas por el usuario con la estructura leer, si no que se pueden hacer automáticamente con la estructura expresión escribiendo el nombre de la variable, el operador " = " y el valor inicial.

En cuanto a los nombres de las variables, se debe tener en cuenta y como regla principal que contenga al inicio una letra preferiblemente en minúscula, porque para la programación el manejo de variables se realiza con minúsculas. En el nombre de la variable están permitidos los caracteres especiales, por ejemplo un asterisco (*), un más (+) ó una línea baja (_) entre otros, pero no se permite el ingreso de espacios en blanco o signos de puntuación. Para mejor comprensión del algoritmo es necesario colocar a las variables, nombres significativos, es decir que representen el trabajo para el cuál son creadas, permitiendo que una variable contenga más de una palabra, sin ser separadas por espacios en blanco.

6.2.1 Palabras reservadas

El programa maneja dos tipos de palabras reservadas, es decir que no se puedan utilizar como nombres de variables. En primer lugar están las constantes que se manejan, en caso de que el usuario quiera utilizar alguna de estas:

- **Constantes: *** e = 2,7182...; ***** pi = 3,1415...

- **Funciones:** Un error común es utilizar alguna de estás funciones como nombre de variables, lo que está incorrecto y al momento de interpretar presenta un mensaje de error en el análisis, el programa implementa funciones del Parser JEP que se encuentran en la siguiente tabla, a tener en cuenta en la implementación de un algoritmo:

Seno	sin(x)
Coseno	cos(x)
Tangente	tan(x)
Arco seno	asin(x)
Arco coseno	acos(x)
Arco tangente	atan(x)
Arco tangente (con 2 parámetros)	atan2(y, x)
Seno hiperbólico	sinh(x)
Coseno hiperbólico	cosh(x)
Tangente hiperbólica	tanh(x)
Seno hiperbólico inverso	asinh(x)
Coseno hiperbólico inverso	acosh(x)
Tangente hiperbólica inversa	atanh(x)
Logaritmo natural	ln(x)
Logaritmo base 10	log(x)
Exponencial	exp(x)
Valor / magnitud absoluta	abs()
Truncar	trunc(x)
Redondear	round(x)
Número aleatorio (entre 0 y 1)	rand()
Módulo	mod()
Raiz cuadrada	sqrt()
Sum	sum()
Si	if()
Convertir a string	str()

6.2.2 Operadores manejados

El software maneja la mayoría de operaciones para la implementación de cualquier tipo de algoritmo, esas operaciones se representan con los símbolos representados en la siguiente tabla:

NOMBRE Y SÍMBOLO	
Potencia	^
Negación lógica	!
Valor positivo o negativo	+x, -x
Módulo	%
División	
Multiplicación	*
Adición, Sustracción	+, -
Menor o igual, mayor o igual	<=, >=
Menor que, mayor que	<, >
Distinto, igual	!=, ==
Y lógico	&&
O lógico	||

Todas las operaciones destacadas en la tabla anterior se pueden implementar con datos de tipo numérico, pero solamente el operador adición (+), distinto (!=) e igual (==), se pueden aplicar a datos de tipo carácter o cadena.

NOTA: Para la multiplicación, se puede realizar de forma implícita, es decir, reemplazando el operador asterisco " * " por un espacio en blanco al momento de multiplicar por ejemplo, un número por una variable " x ", tenemos que:

4*x es igual que tener 4 x Nótese el espacio en blanco.

7. GENERAR CÓDIGO

Para generar código en el programa, se puede realizar de dos formas, la primera es dando clic en el menú Archivo, luego se selecciona Generar código y se elige el lenguaje de programación, que puede ser Java o lenguaje C, aparece una ventana en donde se escribe un nombre para el código generado. Lo que sigue, es editar o cambiar el código generado o si no se tienen cambios, se puede guardar en una ubicación específica con el nombre que se quiera y la extensión que el programa le coloca por defecto (*.cpp o *.java).

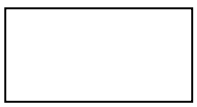
Si se quiere obtener más información acerca del proceso de generar código a partir de un algoritmo N-S diseñado, se debe observar el Demo que se encuentra en el programa, que se encuentra en la barra de menú **Ayuda** y a continuación pulsamos **Demos**, seleccionando de la ventana el Demo **Generar código** y damos clic en **OK**, para que nos indique la forma correcta de realizar este paso.

8. EJEMPLO DE DISEÑO DE DIAGRAMA

Diseñar un algoritmo para calcular el área y el perímetro de un rectángulo

- Definición del problema: Calcular área y perímetro de un rectángulo

- Análisis del problema: Para desarrollar este problema es necesario conocer las fórmulas para obtener tanto el área como el perímetro de un rectángulo.



Sea b = base y h = altura, las fórmulas a utilizar son:

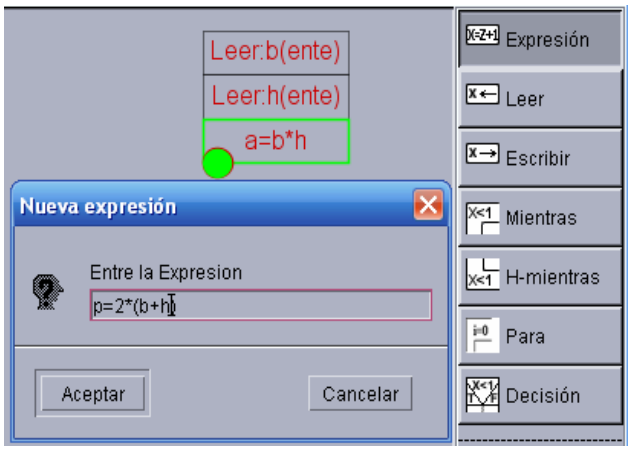
- Área = b * h
- Perímetro = 2 * (b + h)

- Datos de entrada: b y h (base y altura)
- Datos de salida: área y perímetro

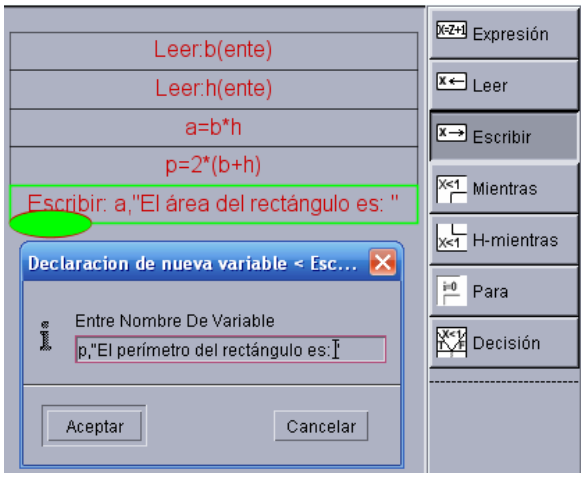
- Procesos: área = b * h Perímetro = 2 * (b + h)

- Diseño de la solución: Se selecciona el comando Leer de la barra de dibujo, se da un clic en el área de trabajo para que aparezca la ventana en donde llenamos el nombre de la variable, en este caso **b**, para indicar que es la base y de igual forma para la altura **h**.

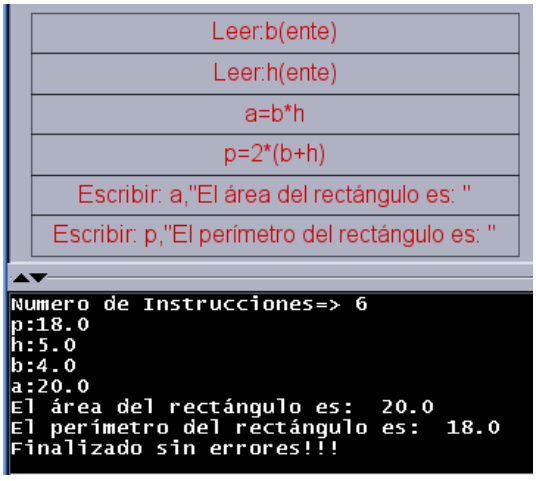
En seguida se elige el comando Expresión, se da clic en el área de trabajo y nuevamente aparece una ventana para llenar la expresión de cálculo para el área (**a**) y el perímetro (**p**), de acuerdo a sus fórmulas.



Por último se tiene que escribir las dos variables calculadas con su respectivo mensaje, quedando el algoritmo completo.



Luego de terminar el algoritmo se procede a interpretarlo con el fin de verificar su correcto diseño, entonces se puede ejecutar de tres formas diferentes y el programa guiará su interpretación de acuerdo a la estructura, en este caso se ha ejecutado paso a paso y en la consola se observa los valores de la variable y el resultado obtenido:



Anexo B

UNIVERSIDAD DE NARIÑO PASTO

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERIA DE SISTEMAS SEMESTRE IX

OBJETIVO: Esta encuesta tiene como objetivo recolectar determinar el nivel de conocimiento de los diagramas N-S y la utilización de herramientas para diseñarlos.

INTRUCCIONES: Responder las siguientes preguntas marcando con una **"X"** la respuesta que usted considere acorde con su opinión.

ENCUESTA Nº_____

I.¿Al momento de realizar un algoritmo, qué herramienta utiliza?

PSEUDOCODIGO____ DFD____ D N-S____

II. ¿Conoce la existencia de alguna herramienta software para representar algoritmos?

SI_____ NO_____ ¿Cuál?_____

III. Conoce el diseño de diagramas N-S

SI____ NO____

IV. ¿Utiliza con frecuencia los diagramas N-S

SI____ NO____

¿Porqué?_____

V. ¿Conoce la existencia de alguna herramienta software para diseñar diagramas N-S?

SI_____ NO_____ ¿Cuál?_____

VI. ¿Considera necesario desarrollar herramientas para el diseño de diagramas N-S?

SI_____ NO_____