Universidad de las Ciencias Informáticas
Facultad 6

Título: alasBioSyS 2.0: Plug-in de Análisis de Series Temporales

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Haymel Odio Domínguez

Yailen Delgado Reyes

Tutor(es): MsC. Yunet González Mulet

Consultante: MsC. Noel Moreno Lemus

Ciudad de La Habana, Cuba, 2010

"Año 52 de la Revolución."

"El futuro de nuestro país tendrá que ser necesariamente un futuro de hombres de ciencia, de hombres de pensamiento".

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Haymel Odio Domínguez

_____

Firma del Autor

Yailen Delgado Reyes

_____

Firma del Autor

MsC. Yunet González Mulet

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor:

MsC. Yunet González Mulet

Universidad de las Ciencias Informáticas, Habana, Cuba

Email: ygonzalezmu@uci.cu

AGRADECIMIENTOS DE YAILEN

Para todos mis familiares en especial a mi mamita Ana Maris y mi papito Alexis por ser la razón de mí existir, los quiero mucho son los mejores papas del mundo, sin ustedes yo no estaría hoy aquí ni seria nadie.
A mi hermano adorado del alma Alexei que aunque hoy no esté aquí compartiendo este gran momento de mi vida, eternamente estas y estarás en mi corazón, fuiste mi guía y mi ejemplo siempre.
A mi tío Juan Carlos que ha sido como un padre para mí, aunque con un carácter un poco fuerte sé que lo hacías por mi bien, te quiero mucho.
A mis abuelas Damaris y Anay por malcriarme y ser tan consentidoras, las adoro.
A todos mis tíos, primos y demás familiares que no he mencionado no piensen que no me acuerdo de ustedes es muy poca hoja y no alcanza para todos pero no se me olvidaron, muchos besos.
A Persy por ser mi compañero en las buenas y en las malas, gracias por soportarme estos 5 años y ayudarme tanto.
A mi compañera de tesis Haymel por tener que soportar mi carácter y mi forma, te mereces un premio. Gracias.
A todos los que he conocido en el trascurso de la universidad que aunque no se lo imaginen dieron un granito de arena para yo estar hoy aquí, Nela (la enana), Marisel (la chuchi), Imara (la suqui), Aliekna (Alik), Elisa (Eli), Yanet, Tatiana (tati), Yadiurvis, Mariela (marilú), Leanet (lili), Geidy (tuti), Yadira(tillan), Carlos (el pelu), Jorge, Juan Antonio(Pochy), Asencio(pasú), Edel, Carlos y Luis Grabiel por toda su ayuda y por los momentos que pasamos, los quiero mucho.
A los que fueron compañeros de aula durante el trascurso de la carrera,Raidel, Leysi, Tavo, Hermes, Yadira, Alicia, Sulay, Surama, Dayana, Alex , los pelú, las Anais, Robert, Guille, Yili, Acro, Angel, Bola, Cuan, y restantes es que son muchos, siempre me acordaré de todos y los tendré presente.
A la gente de Moa, el pikiri, carli, el enano, él feo, lila, daro y demás, a los que no mencioné no se pongan bravos me acuerdo de ustedes también, a mis vecinos que con su esfuerzo y ayuda también aportaron algo a la causa.
A mi tutora Yunet por haber estado presente siempre que la necesitamos y otro tutor también Noel que nos dio su ayuda y apoyo.
A mi otra familia aquí en la Habana que me acogieron en su casa como una más, besos y abrazos y la bendición de dios. A todos los que de una forma u otra contribuyeron a que hoy yo pueda decir que soy ya una ingeniera, los quiero y los llevo en mi corazón a todos, muchísimas gracias.

AGRADECIMIENTOS DE HAYMEL

A mi mami que aunque no estuvo aquí conmigo siempre la tuve presente en mi corazón.
A mi papi que se merece todo lo bueno de este mundo por ser tan bueno y por estar conmigo estos años tan difíciles sin mi mami.
A Oty mi hermanita por estar conmigo en las buenas y en las malas siempre apoyándome.
A mi abuelita linda Elvira que la quiero mucho mucho y que nunca la voy a olvidar.
A mi tío Raoul, el negro, Jiménez y mi tía Marcia por ayudarme en todo lo que han podido.
En general a toda mi familia los quiero mucho espero ser un orgullo para ustedes.
A mis amigos del barrio, para los que están aquí y los que están un poco lejos, los quiero a todos, en especial para mi amiga del alma Marta que ha estado conmigo desde que empezamos en la escuela y en toda mi vida, te quiero y te voy a extrañar mucho.
A mis amigos del aula los pelus, las Anais, Sulay, Yadira, Alicia, Yaneysi, Yili, Surama, Robert, Alex, Dayana, Bola, Jesús, Karelia, Acro, Angel, Guillermo y Yosbel.
A mis amigas del apto Marilú, Yadiurbis, Imara, Aliekna, Marisel, Yanet, Elisa y Tatiana.
A mis amigos desde primer año Yisel, Irina Rodríguez, Ailem, Melbita, Leticia, Leysi, Raidel, Frank, Piny, Tuti, Lily, Reinier, Vera, Fito, Danelys, Yasmani, Gracielita y Hermes.
A mis mejores amigas Nelin, Tillan y la Iri por haberme ayudado tanto y aguantado, por qué no?, las quiero mucho, nunca las voy a olvidar dondequiera que estén.
A mi compañera de tesis Yailen como le digo más cariñosamente la Yumi por estar ahí conmigo siempre.
A mis amigos que son tantos que sólo puedo mencionar a los más cercanos Jorge, Asencio, Asdrubal, Persy, Carlos, Osmay, David, Rene, Ale, Leo, Roniel, Reinier, Johan, Javier, Yohan, Carlos, Edel, Tavo y Abdel
A mi grupo Ilu Ashe y a todos los negros.
A mis profes Noel, Julio, Alexei, Niurka, Garnache, Liesner, Liudmila, Sara, Edgar y Beatriz.
A mi Pochy por estar conmigo en estos últimos meses.
A Yunet mi tutora por apoyarnos y ayudarnos en todo lo posible.
A mi familia en Holguín que me han acogido cariñosamente Juan y Graciela.
Para todos aquellos amigos, profes, vecinos y familiares, los quiero mucho y nunca los olvidaré.
Gracias a todos.
Pd: Espero no se me haya quedado alguien sin poner, esto es para todos los que he conocido a lo largo de mi carrera universitaria.

DEDICATORIA

A nuestros padres porque son el tesoro mas valioso de nuestras vidas.
A nuestras familias que nos han apoyado durante estos 5 años.
A nuestros amigos que han sabido compartir junto a nosotras momentos buenos y malos.
A nuestros profes por habernos enseñado todo lo que sabemos hasta hoy.
A todos los que ayudaron a que este sueño de ser ingenieras se hiciera realidad.
Y no por último es menos importante a nuestra Revolución Cubana y a Fidel por haber creado esta Universidad de excelencia que hoy en día forja grandes ingenieros.

Gracias a todos

RESUMEN

Las series temporales se analizan a través de métodos que ayudan a interpretarlas, al permitir extraer información representativa referente tanto a los orígenes o relaciones subyacentes como a predecir su comportamiento futuro. El análisis más clásico de las series temporales se basa en los valores que toma la variable de observación. Muchos son los software que han sido desarrollados en este sentido, unos de propósito general y otros más específicos como TISEAN que a pesar de contar con un conjunto de técnicas para el análisis de series temporales, aún presenta limitantes, como su interfaz poco amigable y su difícil utilización en el instante de realizar el análisis de la serie temporal. Con este trabajo se presenta una aplicación que busca solucionar estos problemas, proporcionando al usuario la interacción con una interfaz de fácil utilización y que permite la agilización del proceso de análisis de una serie temporal.

PALABRAS CLAVES
Series temporales, análisis, plug-in.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En los últimos cincuenta años el estudio de los sistemas dinámicos ha aportado valiosos conocimientos en ciencias como la Biología, la Química y la Física por sólo citar algunas (1). Un sistema dinámico es aquel que evoluciona en el tiempo a partir de un grupo de reglas que por lo general son no lineales. Una de las formas más utilizadas de representar un sistema dinámico es por medio de un sistema de ecuaciones diferenciales (SED), dichas ecuaciones, pocas veces se pueden solucionar analíticamente (2), (3), (4). Con el objetivo de darle solución a esta problemática se utilizan los métodos numéricos, los cuales son muy conocidos y se encuentran implementados en varios software. Como resultado de la resolución del SED, mediante este método, se obtiene una serie temporal, a partir de la cual se adquiere información cualitativa acerca del comportamiento de dichas soluciones.

Para el análisis de las series temporales se han desarrollado diversas técnicas, con el fin de reconocer patrones de comportamientos según la evolución de las variables y parámetros dentro de un sistema, a fin de conocer el comportamiento de los posibles valores futuros. Dentro de estas se encuentran la Dimensión Fractal, Análisis de Bifurcaciones, la Dimensión de Correlación, Ploteo Recurrente y el Espectro de Lyapunov.

Existen algunas herramientas que implementan estas técnicas, una de las más reconocidas por su utilización es el TISEAN, la cual es una aplicación informática para el análisis de series temporales con métodos basados en la teoría de sistemas no lineales dinámicos deterministas (5). Entre sus principales ventajas se encuentran su libre distribución y una gran documentación, brindando así información útil para los investigadores, como las referencias de los algoritmos utilizados. Sus principales debilidades radican en una interfaz poco amigable al usuario, las salidas que proporciona son ficheros pocos descriptivos y abstractos; el usuario necesita de vasta experiencia en el uso del TISEAN para realizar los análisis con fluidez, además de tener conocimiento básico de código MS-DOS para poder trabajar con el mismo (6).

Como parte del proceso de informatización de la sociedad cubana en los diferentes sectores que la componen y con el objetivo de potenciar la sustitución de importaciones de productos bio-informáticos, se desarrolla en la facultad 6 un software llamado alasBioSyS para la simulación de sistemas biológicos a partir de modelos matemáticos. Los resultados que este arroja constituyen series temporales que por el

momento son analizadas desde el punto de vista informático a través de la Minería de Datos, no mediante las técnicas clásicas utilizadas para el análisis del comportamiento de las series temporales.

El software alasBioSyS carece de un plug-in[1] para el análisis de series temporales por lo que se plantea para el presente trabajo el siguiente **problema científico:** ¿Cómo incorporar funcionalidades del TISEAN a la versión 2.0 de alasBioSyS para el análisis de series temporales?

Con el desarrollo de este trabajo se pretende dar solución al problema antes citado. Para la obtención de estos resultados se plantea como **objeto de estudio:** Aplicaciones informáticas basadas en plug-ins y como **campo de acción:** Desarrollo de plug-ins para el software biosys.

El **objetivo general** de la investigación es: Desarrollar un plug-in de análisis de series temporales para el software alasBioSyS.

Como **objetivos específicos:**

- Definir los requisitos del plug-in de análisis de series temporales.
- Diseñar el plug-in de análisis de series temporales.
- Implementar el plug-in de análisis de series temporales.
- Desarrollar el plug-in para alasBioSyS 2.0.
- Incorporar el plug-in a alasBioSyS 2.0.
- Realizar pruebas unitarias al plug-in.

Las tareas a desarrollar para cumplir los objetivos trazados son:

- Definición de los requisitos del plug-in.
- Definición de las herramientas y la metodología más convenientes para la realización de la aplicación.
- Diseño del plug-in de análisis de series temporales.
- Implementación del plug-in de análisis de series temporales.

[1] Plug-in: Módulo informático que interactúa con otro para aportarle una función o servicio específico.

- Desarrollo del plug-in para alasBioSyS 2.0.
- Incorporación del plug-in a alasBioSyS.
- Realización de pruebas unitarias al plug-in de análisis de series temporales.

Estructura del documento:

El documento está estructurado de la siguiente forma. **Capítulo 1:** Fundamentación teórica, incluye un estudio de los software de análisis de series temporales existentes, se mencionan las técnicas, metodologías y herramientas que serán usadas para dar solución al problema planteado. **Capítulo 2:** Características del Sistema, contiene los requerimientos funcionales y no funcionales que darán solución al problema planteado. Además, se identifican los casos de uso, el actor del sistema y se brinda una descripción textual de dichos casos de uso. **Capítulo 3:** Diseño del sistema, es donde se definen las clases del diseño. Se exponen los diagramas de clases del diseño y los diagramas de interacción realizados en el diseño. Se comentan los patrones de diseño y de arquitectura aplicados. Se muestra a través del diagrama de despliegue la distribución de los componentes físicos necesarios para la implantación del sistema. **Capítulo 4:** Implementación del sistema, contiene la representación de los diagramas de componentes, se muestran las principales interfaces de la aplicación y por último se muestran los resultados de las pruebas aplicadas y un análisis de los resultados obtenidos. Al final se incluyen las **Bibliografías** y un conjunto de **Anexos** que brindan información complementaria al lector.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Este capítulo es el resultado de la búsqueda y análisis de la información relacionada con el estudio del estado del arte del tema tratado, los software existentes de análisis de series temporales y se abarcan también las tendencias, técnicas, metodologías y herramientas usadas en la actualidad, puntualizando en las que sirven de apoyo para el estudio de las series temporales.

1.1 Sistemas Dinámicos

Eduard Groller plantea que un sistema dinámico es aquel, cuya evolución temporal de un estado inicial se establece por un juego de reglas (7).

El estudio de los sistemas dinámicos trata de comprender los sistemas deterministas, considera situaciones que dependen de algún parámetro dado, que frecuentemente sea el tiempo, y que varían de acuerdo a leyes establecidas. De manera que el conocimiento de la situación en un momento dado, permite reconstruir el pasado y predecir el futuro. Se podría decir que un sistema dinámico es un modo de describir el recorrido a lo largo del tiempo de todos los puntos de un espacio dado (8).

1.2 Series Temporales

Cuando se habla de una secuencia de valores observados a lo largo del tiempo, y por tanto ordenados cronológicamente, en un sentido amplio, es denominado serie temporal; la cual puede definirse como una sucesión ordenada en el tiempo de valores de una variable. Puede ser representada con una matriz donde una de las variables es el tiempo y las demás son las variables del sistema en estudio. No es más que la evolución temporal de un sistema. Uno de los usos principales de los modelos de series temporales ha sido el de proporcionar predicciones a corto y medio plazo de las variables (9).

1.3 . Análisis de series temporales

El análisis de series temporales se utiliza hoy día con profusión en muchas áreas de la ciencia, fundamentalmente en Física, Economía e Ingeniería. Los objetivos del análisis de series temporales son diversos, pudiendo destacar la predicción, el control de un proceso, la simulación de procesos y la generación de nuevas teorías. El análisis de series temporales incluye técnicas que permiten interpretar

este tipo de datos, al extraer información representativa referente a los orígenes o relaciones y que da la posibilidad de pronosticar comportamiento futuro. Este último constituye uno de los usos más habituales de las series temporales, entre algunas de estas se encuentran Dimensión de Correlación, Ploteo Recurrente y Espectro de Lyapunov.

1.3.1 Dimensión de Correlación

La dimensión de correlación se refiere al número mínimo de dimensiones necesarias para caracterizar al sistema. La baja dimensionalidad indica la existencia de un modelo simple que genera y rige el sistema en estudio.

Para el cálculo de la dimensión de correlación se han definido alternativas, pero el método propuesto por Grassberger y Procaccia (1983) es el más utilizado, puesto que no precisa de series temporales tan largas como el método originariamente propuesto por Takens (1981). Para su cálculo se elige al azar un punto perteneciente a la dinámica y sobre él se marca una esfera de radio r, luego se cuenta la cantidad de puntos dentro de la esfera y se estudia cómo varía esa cantidad en función del radio r de la esfera. Por ejemplo, para una distribución uniforme de puntos en un volumen (objeto de tres dimensiones), el número de puntos varía como r^3. Los exponentes de r son los valores de las dimensiones necesarias para describir la dinámica (10).

1.3.2 Ploteo Recurrente

Los gráficos de recurrencia representan la dinámica de una serie temporal en un espacio bidimensional. En este tipo de gráficos, las regularidades o irregularidades que se observan, serán muestra de esta misma característica en la dinámica del sistema. Este procedimiento se basa en buscar órbitas regulares (recurrencias), estables o inestables, de una forma gráfica. Además de ser una técnica muy utilizada para reconocer dinámicas deterministas no lineales o caóticas, tiene la ventaja de que pueden estudiarse en series cortas pues lo único que se busca es la presencia de alguna de estas recurrencias indicativas de un orden complejo (10).

Matemáticamente, el Ploteo Recurrente puede ser representado de la siguiente manera:

$$R_{i,j} = \Theta\left(\varepsilon_i - \left\|x_i - x_j\right\|\right), x_i \in R^m,\ i,j = 1 \dots N \quad [3]$$

donde N es el número de estados X_i considerados, ε_i es la distancia umbral y Θ(.) es la función de Heaviside (11). A continuación se muestra la gráfica de recurrencia del atractor de Lorenz (12), Fig. 1 (B):

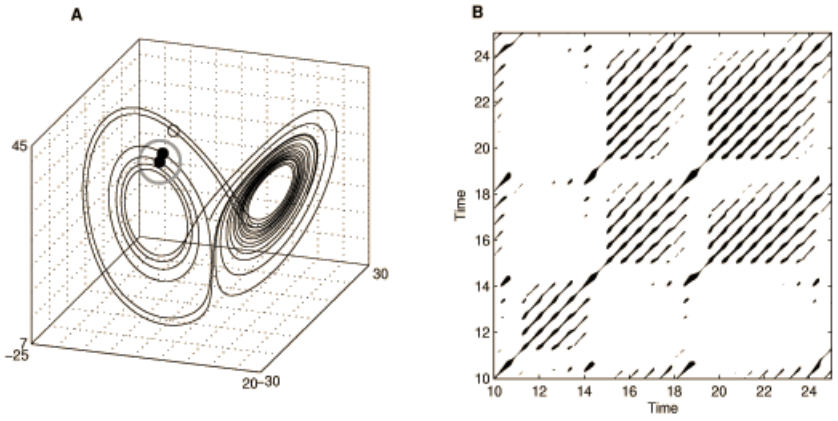


Figura 1: (A) Trayectoria del espacio de fase del modelo de Lorenz. (B) Ploteo Recurrente.

1.3.3 Espectro de Lyapunov

Aleksandr Mikhailovich Lyapunov es conocido por el desarrollo de la teoría de la estabilidad [2]así como por sus muchas contribuciones a la física matemática y la teoría de la probabilidad[3]. El cálculo del espectro de Lyapunov completo requiere un esfuerzo mucho más que sólo el máximo exponente.

El espectro de Lyapunov puede ser usado para dar una estimación de la tasa de producción de entropía y de la dimensión fractal del sistema dinámico considerado. En concreto, desde el conocimiento del espectro de Lyapunov es posible obtener el llamado método de Kaplan-Yorke dimensión D_{KY}, que se define como sigue:

[2] Teoría de estabilidad: Estudia la estabilidad de las soluciones de ecuaciones diferenciales y sistemas dinámicos, es decir, examina como difieren las soluciones bajo pequeñas modificaciones de las condiciones iniciales.

[3] Teoría de la Probabilidad: La definición clásica de la probabilidad, en su forma actual, está basada en el concepto de equiprobabilidad de los resultados, basado a su vez en la simetría. Se supone que un experimento se puede descomponer en *n* sucesos equiprobables y mutuamente excluyentes E1,…z, En, llamados sucesos 'elementales'. Así, la probabilidad de suceso aleatorio A es el número del intervalo [0,1] que expresa el cociente entre los *m* sucesos elementales que componen A y el número total *n* de posibles sucesos elementales.

$$D_{KY} = k + \sum_{i=1}^{k} \lambda_i / |\lambda_{k+1}|$$

donde k es el entero máximo de tal manera que la suma de los más grandes exponentes k todavía no negativos. D_{KY} representa una cota superior para la dimensión de información del sistema.

Considerando que el exponente de Lyapunov da una medida de la previsibilidad total de un sistema, a veces es interesante para estimar las previsibilidades locales alrededor de un punto x_0 en el espacio de fase. Estos valores propios son también llamados exponentes locales de Lyapunov (5).

Una de las principales características de los sistemas no lineales es su carácter impredecible en períodos de tiempo prolongados, como consecuencia de la inherente inestabilidad de las soluciones y de la dependencia de sus condiciones iniciales. Al igual que la dimensión de correlación, el máximo exponente de Lyapunov se obtiene a partir de la reconstrucción del registro en un espacio vectorial, permitiendo obtener una medida abstracta de la separación de las trayectorias a medida que evolucionan temporalmente (13).

1.4 . Aplicaciones informáticas existentes para el análisis de series temporales

1.4.1 Gepasi

EL software Gepasi permite la generación automática de una secuencia de simulaciones con diferentes combinaciones de valores de parámetros, de manera efectiva. Estas características permiten un estudio rápido y sistemático. El código fuente (en C) está disponible a petición del autor, y mientras que la interfaz de usuario depende de tener Windows como sistema operativo, la parte numérica es portable a otros sistemas operativos. GEPASI es adecuado tanto para la investigación como para fines educativos. Aunque fue diseñado con el objetivo de estudiar las vías bioquímicas, puede igualmente ser utilizado para simular otros sistemas dinámicos (14).

1.4.2 Copasi

Este software emplea métodos para garantizar soluciones con altos niveles de fiabilidad. Entre sus funciones están: editar modelos desde la perspectiva matemática o bioquímica, hallar estados

estacionarios, estimación y optimización de parámetros y realiza varios tipos de análisis como el del exponente de Lyapunov en una dinámica del sistema. Para el cálculo, esencialmente utiliza un conjunto de librerías estándares, las cuales implementan varios métodos numéricos. Se debe destacar que los procesos que se llevan a cabo para responder a los tipos de análisis son costosos computacionalmente, por el gran número de operaciones que realizan internamente (15).

1.4.3 Matlab

El asistente matemático matlab (abreviatura de Matrix Laboratory (laboratorio de matrices)), es un programa de análisis numérico creado por The MathWorks en 1984. Está disponible para las plataformas Unix, Windows y Mac OS X. El Matlab cuenta con paquetes de funciones que permiten resolver múltiples problemas, además, brinda la posibilidad de implementar funciones propias, haciendo uso del lenguaje de programación Fortran. Usando sus métodos e implementando funciones propias es posible realizar estudios de sistemas dinámicos, pero, para ellos hay que tener un dominio de Matlab y de su lenguaje de programación. Este software, muy usado en universidades, centros de investigación y desarrollo, incluye funcionalidades para el análisis de estabilidad mediante el cálculo de los exponentes de Lyapunov, específicamente valiéndose del espectro completo de los exponentes (16).

1.4.4 TISEAN

Es un proyecto de software creado para el análisis de series temporales con métodos basados en la teoría de sistemas no lineales dinámicos deterministas. Sus mayores ventajas son que es de libre distribución y que tiene una gran documentación, incorporando información útil para los investigadores como las referencias de los algoritmos utilizados, además tiene incorporada las tres técnicas utilizas para el análisis de series temporales. Su principal debilidad radica en interfaces poco amigables para el usuario, es necesario trabajar suficiente tiempo con él para ganar en experiencia y habilidad para realizar un estudio ágil y la necesidad de tener un conocimiento básico de MS-DOS para sacarle el máximo partido (5).

1.5 Metodología de desarrollo de software OpenUP

Existen diferentes metodología para el desarrollo de productos las que fueron analizadas y definidas en la arquitectura del proyecto alasBioSyS, quedando como propuesta final la metodología OpenUP.

OpenUP/Basic es un Framework de procesos de desarrollo de software de código abierto. Permite un desarrollo ágil del software con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles y tareas. Es un proceso interactivo de desarrollo de software simplificado, completo y extensible para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias (17). Es una versión más ágil de lo que es el Proceso Unificado del Rational (RUP), aplica propuestas iterativas e incrementales dentro del ciclo de vida, puede hacer frente a una amplia variedad de tipo de proyectos. Estructura el ciclo de vida del software en 4 fases: Inicio, Elaboración, Construcción y Transición (17).

1.6 . Lenguajes y herramientas para el desarrollo

1.6.1 Lenguaje de modelado UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para visualizar y documentar los elementos de los sistemas orientados a objetos. Permite modelar, visualizar, especificar y construir los artefactos necesarios. Al no ser un método de desarrollo, es independiente del ciclo de desarrollo que se vaya a seguir, puede encajar en un tradicional ciclo en cascada, o en un evolutivo ciclo en espiral o incluso en los métodos ágiles de desarrollo, soportando tanto el modelo lógico como el físico (18). UML al ser no propietario, es usado y refinado por muchas empresas, grupos de investigadores y desarrolladores a nivel mundial (19).

1.6.2 Herramienta CASE Visual Paradigm Suite 3.4

Las herramientas CASE (Computer Aided Software Engineering) es la mejor base para el proceso de análisis y desarrollo de un software. Estas constituyen un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

Visual Paradigm es una potente herramienta CASE empleada para visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML, proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de forma rápida. Facilita la interoperabilidad con otras herramientas CASE como Rational Rose. Se integra con diversos IDE´s como: NetBeans (de Sun),

Eclipse (de IBM), JDeveloper (de Oracle), JBuilder (de Borland). Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal. Genera código y realiza ingeniería inversa para diferentes lenguajes de programación como: Java, C++, CORBA IDL, PHP, XML Schema y ADA. En adición, se genera código para C#, Visual Basic.net, Object Definition Lenguaje (ODL), Flash Action Script, Delphi, Perl y Python. Se integra con el Visio para importar imágenes del mismo para realizar los diagramas de despliegue. Además, exporta e importa los diagramas en el estándar XML.

Visual Paradigm es la herramienta CASE que se empleará en la modelación de este proyecto por su característica de ser multiplataforma, por las facilidades que brinda y por ser una herramienta que se puede utilizar legalmente.

Visual Paradigm

- Ofrece entorno de creación de diagramas para UML 2.0.
- Disponibilidad en múltiples plataformas.
- Disponibilidad de integrarse en los principales IDEs.
- Soporta una gama de lenguajes en la generación de código e ingeniería inversa en Java, C++, CORBA IDL, PHP, Esquema de XML, Ada y Python.
- La generación de código soporta C #, VB .NET, Lenguaje de Definición de Objeto (ODL), Flash Action Script, Delphi, Perl, Objetivo-C, y Ruby (20).

1.6.3 Lenguaje de Programación Java

Pueden mencionarse muchas características de Java, sin embargo, para los propósitos del presente trabajo se mencionarán sólo las más significativas para la implementación del mismo:

- Orientado a Objetos: Java soporta las características esenciales del paradigma de la programación a objetos: encapsulación, herencia y polimorfismo. Java hace uso de la definición de entidades formadas por métodos y variables que reciben el nombre de clases, la instancia de alguna clase cualquiera en Java recibe el nombre de objeto.
- Robusto: Libera al desarrollador de la necesidad de desalojar la memoria que la aplicación ya no usa, asociado a esto, Java requiere la declaración explícita tanto de los tipos de datos como

de métodos. Estos factores antes mencionados, destacan como causas comunes de errores lo que resulta en aplicaciones poco fiables. Al implementar una aplicación en Java se reduce el porcentaje de errores ocasionados por las causas antes mencionadas lo que da como consecuencia programas más robustos y confiables. Es importante mencionar que Java verifica que no haya problemas tanto en tiempo de ejecución como en tiempo de compilación.

- Independiente de plataforma: Mientras que en lenguajes de programación como C++ existe la necesidad de recompilar el código fuente cada vez que se cambia de plataforma, Java ofrece la posibilidad de que los archivos que son generados para una aplicación sean independientes de plataforma, es decir, que se compilen una vez y se ejecuten en cualquier plataforma.
- Multitarea: A pesar de que las capacidades de multitarea que pueden ser implementadas en Java dependen en gran parte del sistema operativo en el cual se ejecuten, digamos Windows o Unix, dichas capacidades superan en gran manera a los entornos de flujo único (single-thread) que ofrecen otros lenguajes de programación. Al ser multitarea, Java permite la ejecución concurrente de varios procesos ligeros o hilos de ejecución.

Java soporta las características esenciales del paradigma de la programación a objetos: encapsulación, herencia y polimorfismo. También hace uso de la definición de entidades formadas por métodos y variables que reciben el nombre de clases (21).

1.6.4 Ambiente de Desarrollo Integrado (IDE). NetBeans versión 6.8

Constituye el ambiente idóneo para el desarrollo de aplicaciones de escritorio usando Java y un entorno de desarrollo integrado (IDE). Facilita todas las herramientas necesarias para crear aplicaciones de escritorio profesional, corriendo en muchas plataformas incluso Windows, Linux, el Mac OS X y Solaris. Posee un consumo más bajo de memoria y sensibilidad mientras se trabaja con proyectos grandes. NetBeans es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Este enfoque de bienes comunes creativos ha permitido una mayor capacidad de uso, con cada nueva versión, y ha proporcionado a los desarrolladores mayor flexibilidad, al modificar el IDE, si así lo desean (24). En cuanto al desarrollo de bases de datos permite correr las consultas SQL desde él mismo, revisa la sintaxis inmediatamente, y su depurador de errores es excelente; además de tener un buscador de ocurrencias en tiempo real. La ingeniería delantera e inversa permite al diseñador de bases de datos diseñar aplicaciones

usando el UML, generando el código desde Java, desde el modelo de UML o actualizando el modelo de cambios hechos en el código fuente (22).

1.7 Herramienta manejadora de Plug-in Front-End GRATO

Teniendo en cuenta que el proyecto alasBioSyS en su versión 2.0 tiene una arquitectura orientada a plug-ins, se definió utilizar como herramienta manejadora de plug-in Front-End, por sus ventajas dentro de las cuales se encuentran que es multiplataforma, es dinámico y se encarga de la portabilidad de los plug-ins garantizando:

- Eliminar impedancias en la interoperabilidad de los plug-in.
- Mantener el principio de conservación, pues en la evolución del Front-End los plug-ins desarrollados siempre serán compatibles.
- Ganar en soportes de escalabilidad para el ingreso de futuros plug-ins.
- Ganar en el manejo de memoria de ejecución controlando todos los componentes visuales desde un mismo marco (32).

1.8 Conclusiones

En este capítulo se realizó el estudio del estado del arte del problema a resolver y los temas principales que se relacionan con el Análisis de Series Temporales, dando una breve explicación de los conceptos fundamentales relacionados con los sistemas dinámicos y los distintos tipos de análisis, también se arribó a la conclusión de utilizar para el plug-in las 3 técnicas siguientes: Dimensión de Correlación, Ploteo Recurrente y por último Espectro de Lyapunov. Además de la herramienta manejadora de plug-in Front-End, como lenguaje de modelado UML, Visual Paradigm para la elaboración de los diagramas correspondientes, Java como lenguaje de programación y como ambiente de desarrollo integrado(IDE) NetBeans, resaltando los beneficios que estas brindan y la metodología OpenUP que posibilita un proceso controlado que guiará el desarrollo del software.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Este capítulo contiene los requerimientos funcionales y no funcionales que darán solución al problema planteado. Además, se identifican los casos de uso, el actor del sistema y se brinda una descripción textual de estos casos de uso.

2.1 Caracterización del problema

En la facultad 6 se desarrollan diversos proyectos, entre los que se encuentra alasBioSyS para el estudio de simulación a partir de modelos matemáticos, los resultados que arroja, constituyen series temporales que por el momento son analizadas desde el punto de vista informático a través de la Minería de Datos, no mediante las técnicas clásicas utilizadas para el análisis del comportamiento de las series temporales.

Por el problema antes expuesto se dio la tarea de implementar el plug-in de análisis de series temporales para alasBioSyS en su versión 2.0 donde se incorporan nuevas técnicas tales como Dimensión de Correlación, Ploteo Recurrente y Espectro de Lyapunov, con las cuales se podrá analizar el comportamiento de las series temporales en un período de tiempo determinado.

Solución propuesta

Actualmente se está implementando la versión 2.0 de alasBioSyS a la cual se le quiere incluir el plug-in de análisis de series temporales para facilitar la tarea del investigador.

Las técnicas antes mencionadas están implementadas en TISEAN, un software utilizado con el fin de analizar series temporales del cual se van a utilizar algunas técnicas en el plug-in, aunque ya están implementadas no cuentan con interfaces amigables al usuario por lo que se realizarán las mismas para mejorarla. El plug-in brinda dentro de sus opciones la posibilidad de seleccionar la técnica con que va a realizar el análisis el investigador, además de mostrar como fichero de salida un fichero de datos.

2.2 Modelo de Dominio

El modelo de dominio es una representación visual, se centra en una parte del negocio, la relacionada con el ámbito del proyecto. Ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar el plug-in.

2.3 Diagrama de conceptos de dominio

A continuación se muestra el modelo del dominio que recoge los principales conceptos que se identifican en el análisis de la herramienta propuesta.

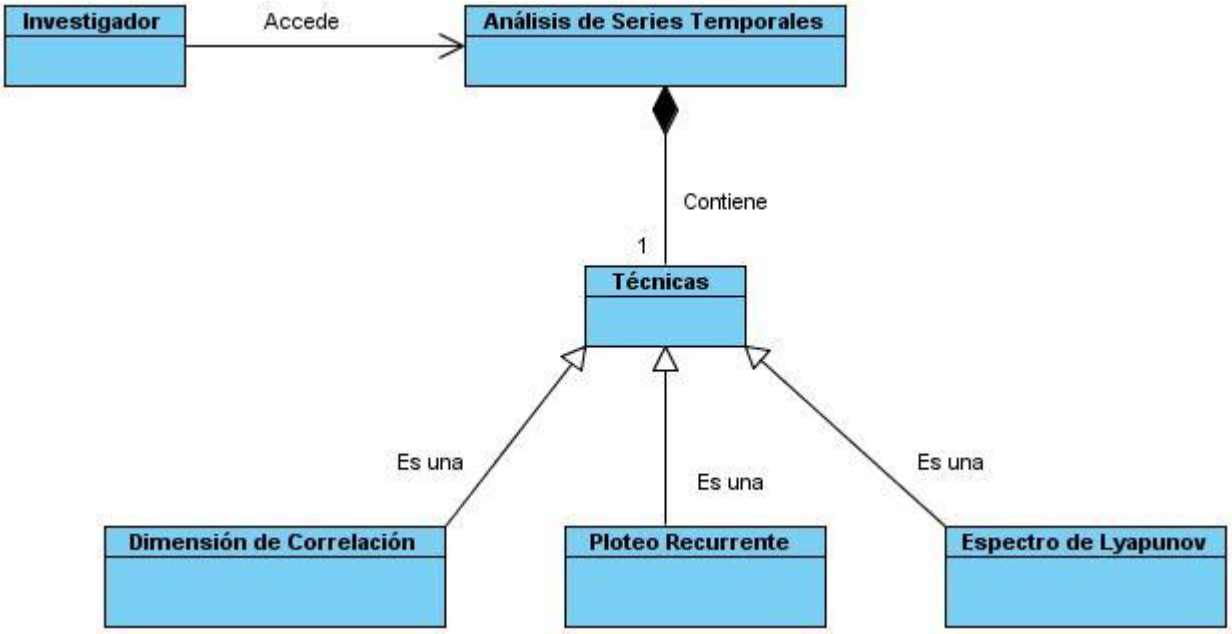


Figura 2: Modelo de Dominio

2.4 Definición de los conceptos del modelo de dominio

- **Investigador:** persona que interactúa con el sistema.
- **Análisis de series temporales:** proceso de estudio del análisis de las series temporales.
- **Técnicas:** técnicas que el investigador utiliza para el posterior análisis de las series temporales.
- **Dimensión de Correlación:** técnica utilizada para el análisis del comportamiento de una serie temporal.
- **Ploteo Recurrente:** técnica utilizada para el análisis del comportamiento de una serie temporal.

- **Espectro de Lyapunov:** técnica utilizada para el análisis de la estabilidad en una serie temporal.

2.5 Actor del sistema

Es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externo con los que el sistema interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el sistema para beneficiarse de sus resultados.

Tabla 1: Descripción del actor del sistema a automatizar.

Actor	**Descripción**
Investigador	Es el rol que podrá realizar un conjunto de tareas para hacer una investigación de determinado sistema biológico.

2.6 Requisitos Funcionales

R1 Analizar la serie mediante la técnica Dimensión de Correlación.

R2 Analizar la serie mediante la técnica Ploteo Recurrente.

R3 Analizar la serie mediante la técnica Espectro de Lyapunov.

2.7 Diagrama de Casos de Uso del Sistema

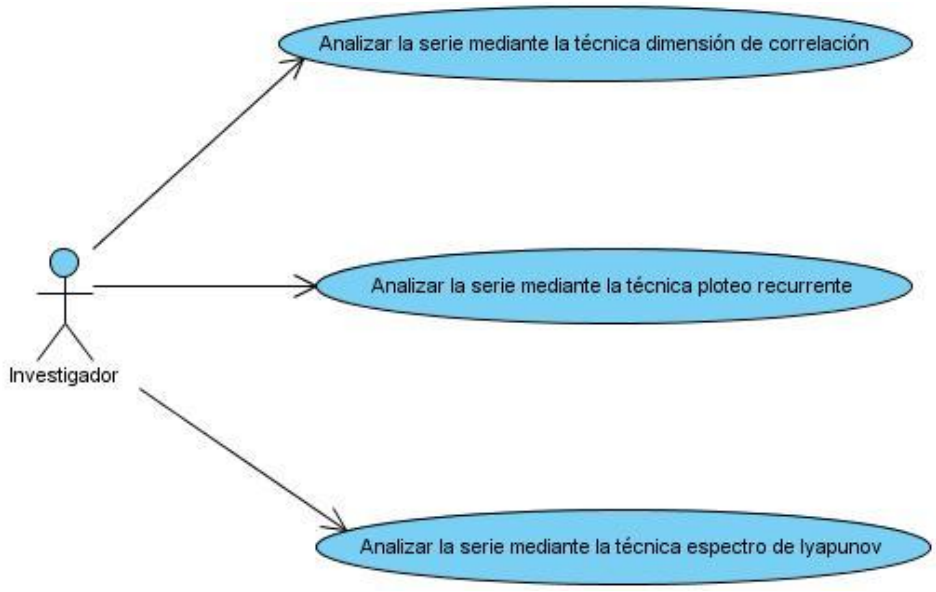


Figura 3: Diagrama de Casos de Uso del Sistema.

2.8 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, y que harán del mismo un sistema confiable y seguro.

Hardware:

- Se requiere de 512 MB de memoria RAM.
- Procesadores Pentium IV.
- Disco duro de 40 GB (puede variar dependiendo de la cantidad de información a almacenar).

Software:

- Se requiere del software TISEAN para el funcionamiento de la aplicación.

Usabilidad:

- El sistema podrá ser usado por aquellos usuarios que posean conocimientos básicos en el campo de la informática y de la modelación de sistemas biológicos.

Soporte:

- El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

Rendimiento:

- El sistema debe ser capaz de formular respuestas lo más rápido posible. Para hacer más rápida la obtención de los resultados de los análisis.

Identidad:

- El sistema debe mantener una misma identidad visual como producto informático garantizando en todo momento que el usuario no experimente cambios bruscos en las interfaces visuales.

2.9 Descripción de los casos de uso del sistema

Nombre del CU	**Realizar Análisis por Dimensión de Correlación**
Actor	Investigador (Inicia)
Resumen	El caso de uso se inicia cuando el investigador selecciona la dirección del TISEAN y el directorio de salida, carga el fichero de entrada y selecciona la técnica para el análisis del comportamiento de la serie temporal y manda a analizar. Finaliza el caso de uso cuando el sistema muestra los resultados en un fichero de salida.
Referencias	RF 1
Precondiciones	Debe existir el fichero de entrada.
Poscondiciones	Se muestra al investigador el fichero de salida.
Curso normal de los eventos	

Acción del actor	Respuesta del Sistema
1. El caso de uso se inicia cuando el investigador selecciona la técnica dimensión de correlación y se activa su interfaz.	2. El sistema muestra la interfaz de la técnica.
3. El investigador carga el fichero de entrada y entra las opciones necesarias para realizar el análisis: • Número de puntos a utilizar (l). • Número de líneas a ser ignoradas (x). • Retraso de los vectores (d). • Número de los componentes, incrustación de dimensión (M). • Columna a leer (c). • Ventana (t). • Escala de longitud máxima (R). • Escala de longitud mínima (r). • Número de valores de épsilon. • Máximo número de parejas que se utilizará (N). • Datos que se normalizarán a [0,1] para todos los componentes (E). • Nombre del archivo de salida(o). • Nivel de detalle (v). 0 sólo los mensajes de pánico. 1 agregar entrada/salida de mensajes. 2 tiempos de abertura de entrada/salida de mensaje de cada archivo. • Mostrar opciones (h).	4. El sistema verifica la validez de los datos.

Presiona el botón que permite el análisis, ícono “Analizar”.	
Prioridad	Crítico

Nombre del CU	**Realizar Análisis por Ploteo Recurrente**
Actor	Investigador (Inicia)
Resumen	El caso de uso se inicia cuando el investigador selecciona la dirección del TISEAN y el directorio de salida, carga el fichero de entrada y selecciona la técnica para el análisis del comportamiento de la serie temporal y manda a analizar. Finaliza el caso de uso cuando el sistema muestra los resultados en un fichero de salida.
Referencias	RF 2
Precondiciones	Debe existir el fichero de entrada.
Poscondiciones	Se muestra al investigador el fichero de salida.
Curso normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el investigador selecciona la técnica de Ploteo Recurrente y se activa su interfaz.	2. El sistema muestra la interfaz de la técnica.
3. El investigador carga el fichero de entrada y entra las opciones necesarias para realizar el análisis: • Número de puntos a utilizar (l). • Número de líneas a ser ignoradas (x).	4. El sistema verifica la validez de los datos.

• Columna a leer (c). • Número de los componentes, incrustación de dimensión (m). • Retraso de los vectores (d). • Tamaño de la sección (r). • Porcentaje de todos los números de puntos encontrados (%). • Nombre del archivo de salida (o). • Nivel de detalle (v). 0 sólo los mensajes de pánico. 1 entrada/salida de mensajes. • Mostrar opciones (h). Presiona el botón que permite el análisis, ícono “Analizar”.	
Prioridad	Crítico

Nombre del CU	**Realizar Análisis por Espectro de Lyapunov**
Actor	Investigador (Inicia)
Resumen	El caso de uso se inicia cuando el investigador selecciona la dirección del TISEAN y el directorio de salida, carga el fichero de entrada y selecciona la técnica para el análisis del comportamiento de la serie temporal y manda a analizar. Finaliza el caso de uso cuando el sistema muestra los resultados en un fichero de salida.
Referencias	RF 3
Precondiciones	Debe existir el fichero de entrada.
Poscondiciones	Se muestra al investigador el fichero de

	salida.
Curso normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el investigador selecciona la técnica Espectro de Lyapunov y se activa su interfaz.	2. El sistema muestra la interfaz de la técnica.
3. El investigador carga el fichero de entrada y entra las opciones necesarias para realizar el análisis: • Número de puntos a utilizar (l). • Número de líneas a ser ignoradas (x). • Columna a leer (c). • Número de los componentes, incrustación de dimensión (m). • Tamaño mínimo de sección (r). • Factor para aumentar el tamaño de la sección (f). • Número de secciones a usar (k). • Número de iteraciones (n). • Invertir el orden de las series de tiempo (l). • Nombre del archivo de salida (o). • Nivel de detalle (v). 0 sólo los mensajes de pánico. 1 entrada/salida de mensajes. • Mostrar opciones (h). Presiona el botón que permite el análisis, ícono “Analizar”.	4. El sistema verifica la validez de los datos.

Prioridad	Crítico

2.10 Conclusiones

En este capítulo se expuso la propuesta del sistema. Se trabajó en la fase de requerimientos, definiendo el modelo del dominio, así como los requerimientos funcionales y no funcionales que regirán el proceso de desarrollo del sistema. Se elaboró el diagrama de casos de uso del sistema y se describieron los mismos, para comenzar la fase de diseño, donde se refinarán los requisitos y se ejecutarán otras actividades propias de ese flujo de trabajo.

CAPÍTULO 3: DISEÑO DEL SISTEMA

Es donde se definen las clases del diseño. Se exponen los diagramas de clases del diseño y los diagramas de interacción realizados en el diseño. Se comentan los patrones de diseño y de arquitectura aplicados. Se muestra a través del diagrama de despliegue la distribución de los componentes físicos necesarios para la implantación del sistema.

3.1 Patrones Arquitectónicos utilizados

Modelo Vista Controlador (MVC) es un patrón de diseño que considera dividir una aplicación en tres módulos claramente identificables y con funcionalidades bien definidas: el Modelo, las Vistas y el Controlador (23).

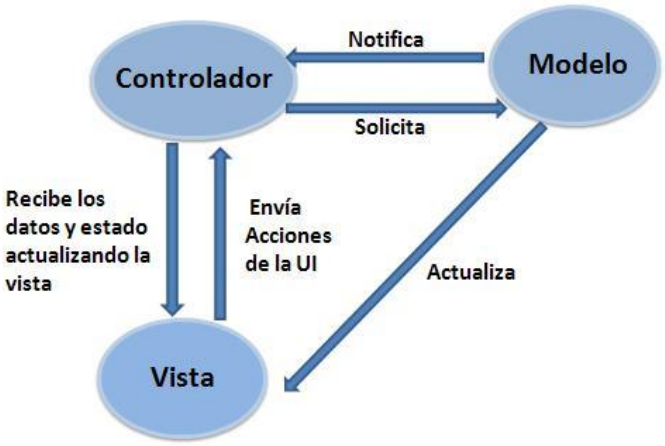


Figura 4 : Patrón Modelo-Vista-Controlador

- **El Modelo**

Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada (24). Es un conjunto de clases que representan la información del mundo real que el sistema debe procesar. El modelo desconoce la existencia de las vistas y del controlador (25).

- **La Vista**

Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador (25). Son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo, está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación (23).

- **El Controlador**

El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador (23).

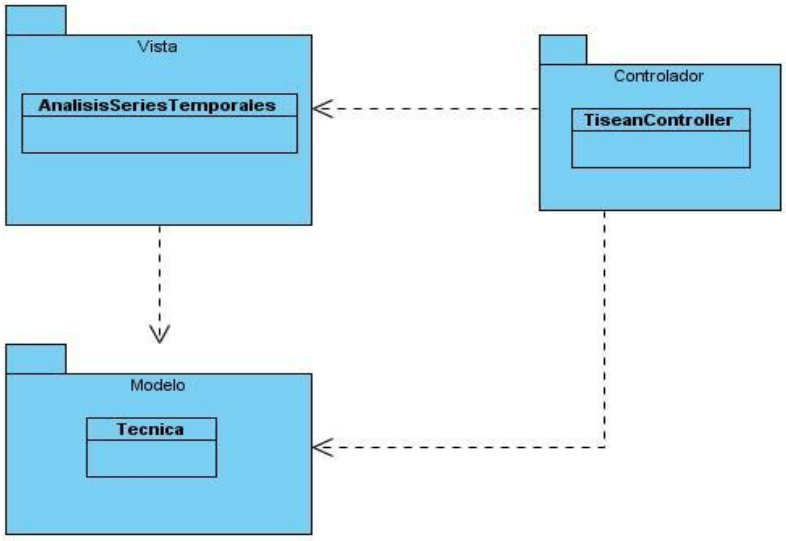


Figura 5 : Aplicación del patrón arquitectónico Modelo-Vista-Controlador.

3.2. Patrones de diseño utilizados

Los patrones de diseño (Design Patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. “Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, luego describe el núcleo de la solución a ese problema, de tal manera que puede usar esa solución un millón de veces más, sin hacer jamás la misma cosa dos veces” (26).

Existen diferentes clasificaciones: dentro de los patrones de producto de software se encuentran los de análisis, arquitectura, diseño y lenguaje de programación. Para el desarrollo de la solución se aplicaron diferentes patrones de diseño, fundamentalmente los patrones de diseño: GRASP (del inglés: General Responsibility Assignment Software Patterns), con el objetivo de facilitar el mantenimiento del software y contribuir a la realización de un producto reutilizable y escalable.

3.2.1 Patrón Experto

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada.

El uso de este patrón queda evidenciado en el diseño de la clase *Tecnica*. Esta clase puede catalogarse como un experto porque dispone de toda la información necesaria para coordinar todo el proceso del análisis de series temporales, o sea, cuenta con los datos necesarios para saber qué tipo de técnica a utilizar para realizar el mismo.

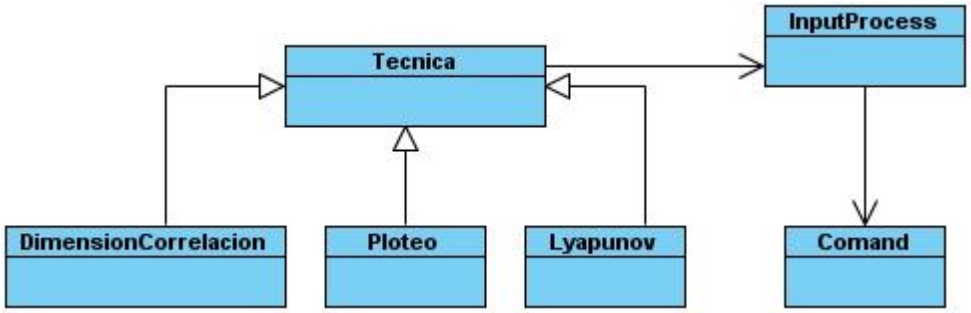


Figura 6: Aplicación del patrón de diseño Experto.

3.2.2 Patrón Creador

Se refiere a asignar responsabilidades a las clases de crear instancias de otras conociendo que las primeras son las que contienen la información para ello.

El uso de este patrón queda evidenciado en el diseño de las clases *DimensionCorrelacion, Lyapunov* y *Ploteo,* pues estas clases son la encargada de crear las instancias de *ImputProcess* a utilizar para el proceso de análisis de series temporales.

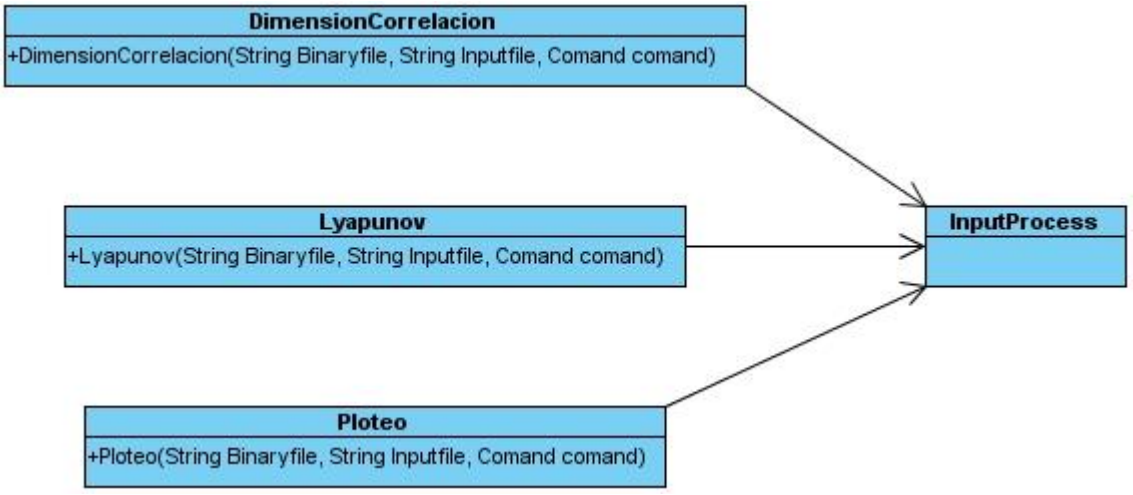


Figura 7: Aplicación del patrón de diseño Creador.

3.2.3 Patrón Bajo Acoplamiento

Cada clase está acoplada (relacionada) a las clases estrictamente necesarias, garantizando un bajo impacto de los cambios que se producen en una clase para las demás clases que se relacionan con ella.

El uso de este patrón queda evidenciado en la asignación de las relaciones entre las clases, pues cada una de ellas está relacionada de manera que se establezcan sólo las dependencias necesarias para cumplir con sus responsabilidades, esto favorece a la flexibilidad del diseño y a la actualización de los cambios del sistema pues las clases son menos dependientes entre sí. Ver más **(Anexo1)**

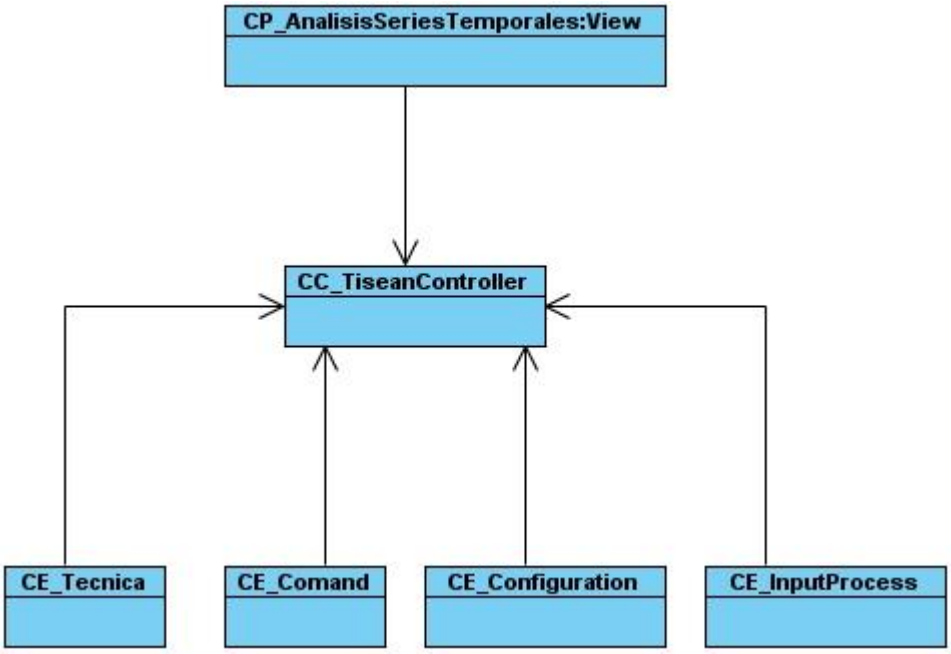


Figura 8: Aplicación del patrón de diseño Bajo Acoplamiento.

3.2.4 Alta Cohesión

Asignar responsabilidades a las clases de manera que todos sus métodos tuvieran un comportamiento bien definido.

El uso de este patrón queda evidenciado en el diseño de las clases, pues cada una de ellas contiene sólo las funcionalidades que le corresponden según la información que manejan. Ejemplo de la aplicación de este patrón en el sistema desarrollado se puede ver en las clases *TiseanContoller, ImputProcess, Configuration, Comand y Opción.*

3.2.5 Singleton

El Patrón Singleton se utiliza para garantizar la persistencia de una única instancia de objetos durante la ejecución de un programa. El uso del patrón queda evidenciado cuando en la clase *AnalisisSeriesTemporales* tiene un atributo estático denominado instance de tipo *AnalisisSeriesTemporales*; en el constructor de dicha clase se chequea si instance tiene valor o no y de no tenerlo se le asigna la propia instancia de la clase. Ver más **(Anexo2)**

Figura 9: Aplicación del patrón Singleton.

3.3 Modelo del Diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso. Se centra en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema a desarrollar (28).

Dentro de sus propósitos están: crear una entrada apropiada y un punto de partida para la implementación, descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia. El modelo de diseño puede estar conformado principalmente por paquetes, clases, subsistemas de diseño y sus relaciones.

Los subsistemas de diseño y las clases del diseño representan abstracciones del subsistema y componentes de implementación del sistema. Estas abstracciones son directas, y representan una sencilla correspondencia entre el diseño y la implementación (28).

3.4 Diagrama de clases del diseño

Un diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro (29). A continuación se muestran los diagramas de clases del diseño

- Diagrama de clases del diseño CU Analizar la serie mediante la técnica Ploteo Recurrente. **(Anexo 3)**
- Diagrama de clases del diseño CU Analizar la serie mediante la técnica Espectro de Lyapunov. **(Anexo 4)**
- Diagrama de clases del diseño CU Analizar la serie mediante la técnica Dimensión de Correlación. **(Anexo 5)**

3.5 Descripción de las clases del diseño

A continuación se ofrece una descripción detallada de los atributos y métodos de las clases principales.

Nombre: Tecnica	
Atributo	**Tipo**
content	String
Binaryfile	String
ejecutable	InputProcess
Para cada responsabilidad	
Nombre	Tecnica(String Binaryfile, String Inputfile)
Descripción:	Constructor de la clase recibe como parámetros dos variables de tipo cadena, una que es un fichero binario propio del Tisean y la otra variable el fichero de entrada de la aplicación.
Nombre	readFile(File file)
Descripción:	Método que recibe como parámetro una variable de tipo file encargado de efectuar el proceso de lectura del fichero.

Nombre	start()
Descripción:	Método encargado de inicializar el proceso de ejecución de la técnica.

Tabla 2: Descripción de la clase de diseño Tecnica.

Nombre: TiseanController	
Atributo	**Tipo**
listaTecnica	ListaSE<Tecnica>
tiseanconfig	Configuration
listaopciones	ListaSE<Opcion>
algoritmo	String
TiseanDir	String
OutPutDirectory	String
OutPutFileName	String
FileName	String
Path	String
PathSeparator	String
Para cada responsabilidad	
Nombre	TiseanController()
Descripción:	Constructor de la clase, método encargado de inicializar los atributos de la clase.
Nombre	CrearTecnicaDimensionCorrelacion (String fichero)
Descripción:	Método que recibe como parámetro una variable de tipo cadena que es el fichero de entrada, encargado de efectuar el proceso de crear una técnica de tipo DimensionCorrelacion.
Nombre	CrearTecnicaLyapunov(String fichero,String Comando)
Descripción:	Método que recibe como parámetros dos variables de tipo cadena, una que es el fichero de entrada y la otra los comando de acuerdo con la técnica utilizada, encargado de efectuar el proceso de crear una técnica de tipo Lyapunov.
Nombre	CrearTecnicaPlotero(String fichero)
Descripción:	Método que recibe como parámetros una variable de tipo cadena, que es el fichero de entrada, encargado de efectuar el proceso de crear una técnica de tipo Ploteo.
Nombre	AdicionarTecnica(Tecnica t)
Descripción:	Método que recibe como parámetro una variable de tipo Tecnica, encargado de adicionar cualquier tipo de técnica.
Nombre	RemoveTecnica(Tecnica tec)
Descripción:	Método que recibe como parámetro un variable de tipo Tecnica, encargado de eliminar cualquier tipo de técnica.
Nombre	CrearTecnica(String fileInput)

Descripción:	Método encardado de crear la técnica en dependencia de la que seleccione el usuario.
Nombre	CreateComandString()
Descripción:	Método encargado de crear la cadena de comando a partir de la lista de opciones introducida por el usuario.
Nombre	CreateComand(String cmd,int time)
Descripción:	Método encargado de crear un comando a partir de la cadena de comandos.

Tabla 3: Descripción de la clase de diseño TiseanContoller.

Nombre: Configuration	
Atributo	**Tipo**
ExecuteFile	File
TiseanDir	String
OutPutDir	String
Para cada responsabilidad	
Nombre	Configuration(METHODS algoritmo)
Descripción:	Constructor de la clase que recibe como parámetro una variable de tipo METHODS, encargado de inicializar las variables.
Nombre	ChangeMethod(METHODS algoritmo)
Descripción:	Método que recibe como parámetro una variable de tipo METHODS, encargado de cambiar el tipo de algoritmo en la configuración.
Nombre	isLinux()
Descripción:	Método encargado de indicar si el tipo de sistema operativo por el que está corriendo la aplicación es en Linux.
Nombre	isWindows()
Descripción:	Método encargado de indicar si el tipo de sistema operativo por el que está corriendo la aplicación es en Windows.

Tabla 4: Descripción de la clase de diseño Configuration.

Nombre: Comand	
Atributo	**Tipo**
Comando	String
time	int
Para cada responsabilidad	
Nombre	Comand(String comand,int time)
Descripción:	Constructor de la clase que recibe como parámetro una

	variable de tipo cadena que es el comando a ejecutar y otra de tipo entero pasa establecer el tiempo de ejecución del comando, método encargado de inicializar las variables.
Nombre	toString()
Descripción:	Método encargado de concatenar cadena o convertirla en cadena.

Tabla 5: Descripción de la clase de diseño Comand.

Nombre: InputProcess	
Atributo	**Tipo**
sysprocess	java.lang.Process
salida	Process
flag	boolean
Response	String
comand	String
Para cada responsabilidad	
Nombre	InputProcess(Comand c)
Descripción:	Constructor de la clase que recibe como parámetro una variable de tipo comand, método encargado de inicializar las variables.
Nombre	execute()
Descripción:	Método para ejecutar el proceso de entrada
Nombre	run()
Descripción:	Método encargado de correr los procesos de entrada
Nombre	createExecuteComand()
Descripción:	Método encargado de crear un comando ejecutable inicializando un hilo de proceso.

Tabla 6: Descripción de la clase de diseño ImputProcess.

Nombre: OutputProcess	
Para cada responsabilidad	
Nombre	OutputProcess(Comand comando, java.lang.Process sysprocess)
Descripción:	Constructor de la clase que recibe como parámetro una variable comando que sería el que se va a ejecutar y el proceso, método encargado de inicializar variables.
Nombre	run()
Descripción:	Método encargado de correr los procesos de salida

Tabla 7: Descripción de la clase de diseño OutputProcess.

Nombre: Process

Atributo	**Tipo**
bandera	boolean
comando	Comand
Salida	java.lang.Process
Para cada responsabilidad	
Nombre	Process(Comand comando,java.lang.Process Salida)
Descripción:	Constructor de la clase recibe como parámetro un comando que sería el que se va a ejecutar y un proceso que sería el de salida, método encargado de inicializar las variables.
Nombre	run()
Descripción:	Método encargado de correr los procesos.

Tabla 8: Descripción de la clase de diseño Process.

Nombre: Opcion	
Atributo	**Tipo**
valor	String
comando	String
Para cada responsabilidad	
Nombre	Opcion(String valor, String comando)
Descripción:	Constructor de la clase recibe como parámetro un comando que sería el que se va a ejecutar y un valor que sería el introducido por el usuario en los formularios, método encargado de inicializar las variables.

Tabla 9: Descripción de la clase de diseño Opcion.

Nombre: DimensionCorrelacion	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	DimensionCorrelacion(String Binaryfile, String Inputfile, Comand comand)
Descripción:	Constructor de la clase recibe como parámetro un fichero binario correspondiente al tipo de técnica a ejecutar, un fichero de entrada y un comando que sería el que se ejecuta.

Tabla 10: Descripción de la clase de diseño DimensionCorrelacion.

Nombre: Ploteo	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	Ploteo(String Binaryfile, String Inputfile, Comand comand)
Descripción:	Constructor de la clase recibe como parámetro un fichero

	binario correspondiente al tipo de técnica a ejecutar, un fichero de entrada y un comando que sería el que se ejecuta.

Tabla 11: Descripción de la clase de diseño Ploteo.

Nombre: Lyapunov	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	Lyapunov (String Binaryfile,String Inputfile,Comand comand)
Descripción:	Constructor de la clase recibe como parámetro un fichero binario correspondiente al tipo de técnica a ejecutar, un fichero de entrada y un comando que sería el que se ejecuta.

Tabla 12: Descripción de la clase de diseño Lyapunov.

Nombre: ComandConfig	
Atributo	**Tipo**
opciones	Opcion []
Para cada responsabilidad	
Nombre	ComandConfig(Opcion[] opciones)
Descripción:	Constructor de la clase recibe como parámetro una colección de opciones la vista, método encargado de inicializar las variables.
Nombre	String getComand(METHODS algoritmo)
Descripción:	Método de acceso que retorna el comando a ejecutar.
Nombre	getComand(Configuration conf)
Descripción:	Método de acceso que retorna el comando a ejecutar. (sobrecarga de método)

Tabla 13: Descripción de la clase de diseño ComandConfig.

Nombre: AnalisisSeriesTemporales	
Atributo	**Tipo**
formploteo	PloteoRecurrente
formLyapunov	EspectroLyapunov
formdimension	DimCorrelacion
ActivatedIndex	int
instance	AnalisisSeriesTemporales
tiseancontroller	TiseanController

Para cada responsabilidad	
Nombre	AnalisisSeriesTemporales()
Descripción:	Constructor de la clase inicializa los atributos de la clase.
Nombre	PrepareOptions()
Descripción:	Método encargado de coger los datos del formulario y setea la lista de opciones al controlador.
Nombre	initCmp()
Descripción:	Método encargado de inicializar los componentes visuales correspondientes a las técnicas.

Tabla 14: Descripción de la clase de diseño AnalisisSeriesTemporales.

Nombre: ConfigurationView	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	ConfigurationView()
Descripción:	Constructor de la clase inicializa los atributos de la clase.

Tabla 15: Descripción de la clase de diseño ConfigurationView.

Nombre: DimCorrelacion	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	DimCorrelacion ()
Descripción:	Constructor de la clase inicializa los atributos de la clase.
Nombre	getOpciones()
Descripción:	Método encargado una lista de opciones introducidas por el usuario en los formularios.
Nombre	Validate(JTextField txf, String expresionreg)
Descripción:	Método encargado de validar los datos introducidos por los usuarios usando expresiones regulares.

Tabla 16: Descripción de la clase de diseño DimCorrelacion.

Nombre: EspectroLyapunov	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	EspectroLyapunov ()
Descripción:	Constructor de la clase inicializa los atributos de la clase.
Nombre	getOpciones()
Descripción:	Método encargado una lista de opciones introducidas por el usuario en los formularios.

Nombre	Validate(JTextField txf,String expresionreg)
Descripción:	Método encargado de validar los datos introducidos por los usuarios usando expresiones regulares.

Tabla 17: Descripción de la clase de diseño EspectroLyapunov.

Nombre: PloteoRecurrente	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	PloteoRecurrente ()
Descripción:	Constructor de la clase inicializa los atributos de la clase.
Nombre	getOpciones()
Descripción:	Método encargado una lista de opciones introducidas por el usuario en los formularios.
Nombre	Validate(JTextField txf,String expresionreg)
Descripción:	Método encargado de validar los datos introducidos por los usuarios usando expresiones regulares.

Tabla 18: Descripción de la clase de diseño PloteoRecurrente.

Nombre: Explorer	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	Explorer ()
Descripción:	Constructor de la clase inicializa los atributos de la clase.
Nombre	init(String dir, DefaultMutableTreeNode root)
Descripción:	Método encargado dado un directorio listar de forma arbólica todo los ficheros de salida en dicho directorio.

Tabla 19: Descripción de la clase de diseño Explorer.

3.6 Diagramas de secuencias

Un diagrama de secuencia contribuye a la descripción de la dinámica del sistema en términos de la interacción entre sus objetos. Los diagramas de interacción son usados para identificar las clases y los métodos que ellos entregan.

- Diagrama de secuencia del caso de uso Analizar la serie mediante la técnica Dimensión de Correlación. **(Anexo 6)**
- Diagrama de secuencia del caso de uso Analizar la serie mediante la técnica Ploteo Recurrente. **(Anexo 7)**
- Diagrama de secuencia del caso de uso Analizar la serie mediante la técnica Espectro de Lyapunov. **(Anexo 8)**

3.7 Modelo de Despliegue

El modelo de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones. A continuación se muestra en la Figura 10 el modelo de despliegue perteneciente al proyecto alasBioSyS, del cual sólo la parte señalada en rojo se necesita para poner en ejecución el plug-in de análisis de series temporales.

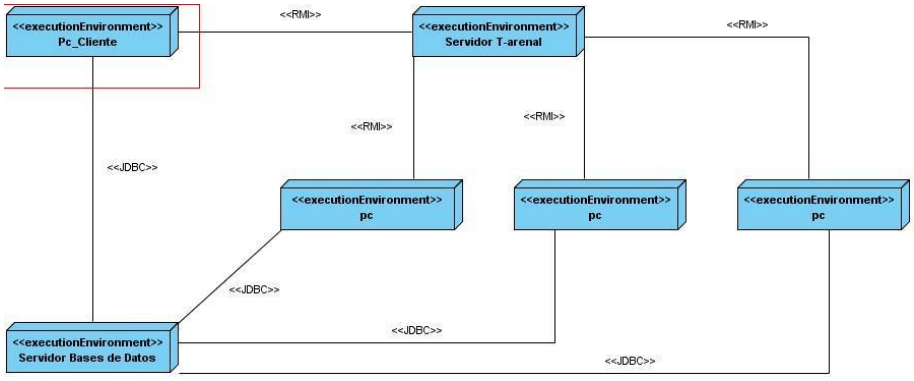


Figura 10: Modelo de Despliegue.

3.8 Conclusiones

En este capítulo se identificó como estilo arquitectónico el Modelo-Vista-Controlador que se utilizó para el desarrollo de las funcionalidades así como los patrones de diseño Experto, Creador, Bajo Acoplamiento, Alta Cohesión y Singleton así como la fundamentación de su aplicación. También fueron desarrollados los diagramas de clases del diseño y los diagramas de secuencia correspondientes.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

El capítulo contiene la representación de los diagramas de componentes, las diferentes interfaces de la aplicación y por último se muestran los resultados de las pruebas aplicadas y un análisis de los resultados obtenidos.

4.1. Implementación

4.1.1 Diagrama de Componentes

Los diagramas de componentes están compuestos por paquetes, componentes, interfaces y relaciones de dependencia, pueden mostrar distintas partes del sistema. Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre los componentes del software. Los componentes pueden ser de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (30).

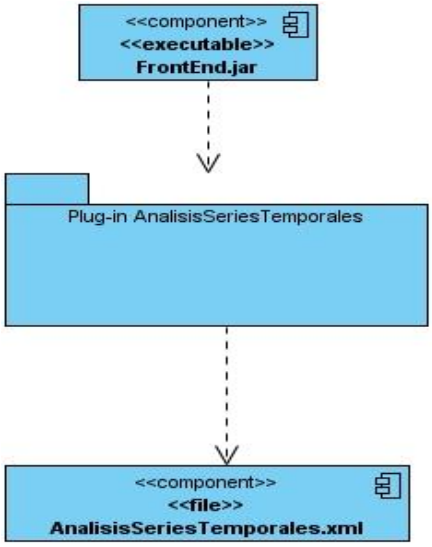


Figura 10 : Diagrama de Componente del plug-in Análisis de Series temporales.

Debido a lo extensos diagramas se agruparon los componentes en un paquete, el cual corresponde al plug-in desarrollado. A continuación se muestran los diagramas de componentes agrupados por caso de uso del plug-in correspondiente.

- Diagrama de componente del caso de uso Analizar la serie mediante la técnica Espectro de Lyapunov. **(Anexo 9)**
- Diagrama de componente del caso de uso Analizar la serie mediante la técnica Dimensión de Correlación. **(Anexo 10)**
- Diagrama de componente del caso de uso Analizar la serie mediante la técnica Ploteo Recurrente. **(Anexo 11)**

4.2 Prototipos funcionales del Pulg-in de Análisis de Series Temporales

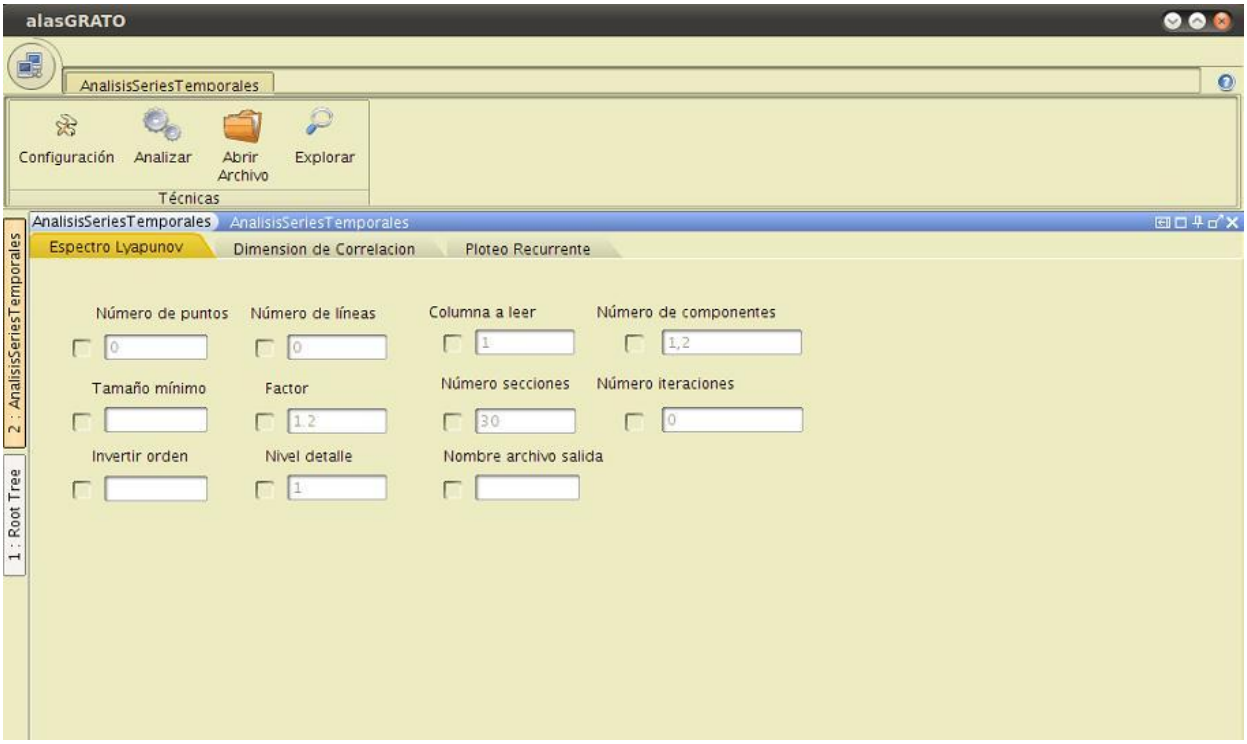


Figura 11: Interfaz principal del plug-in de Análisis de Series Temporales.

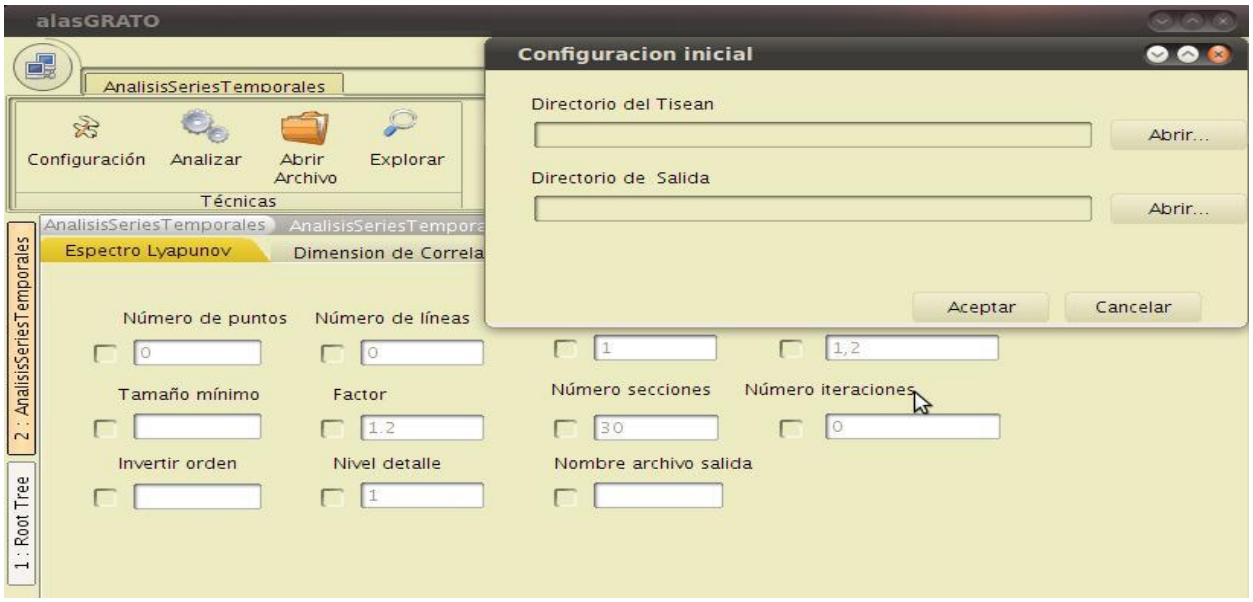


Figura 12: Interfaz de Configuración del Sistema.

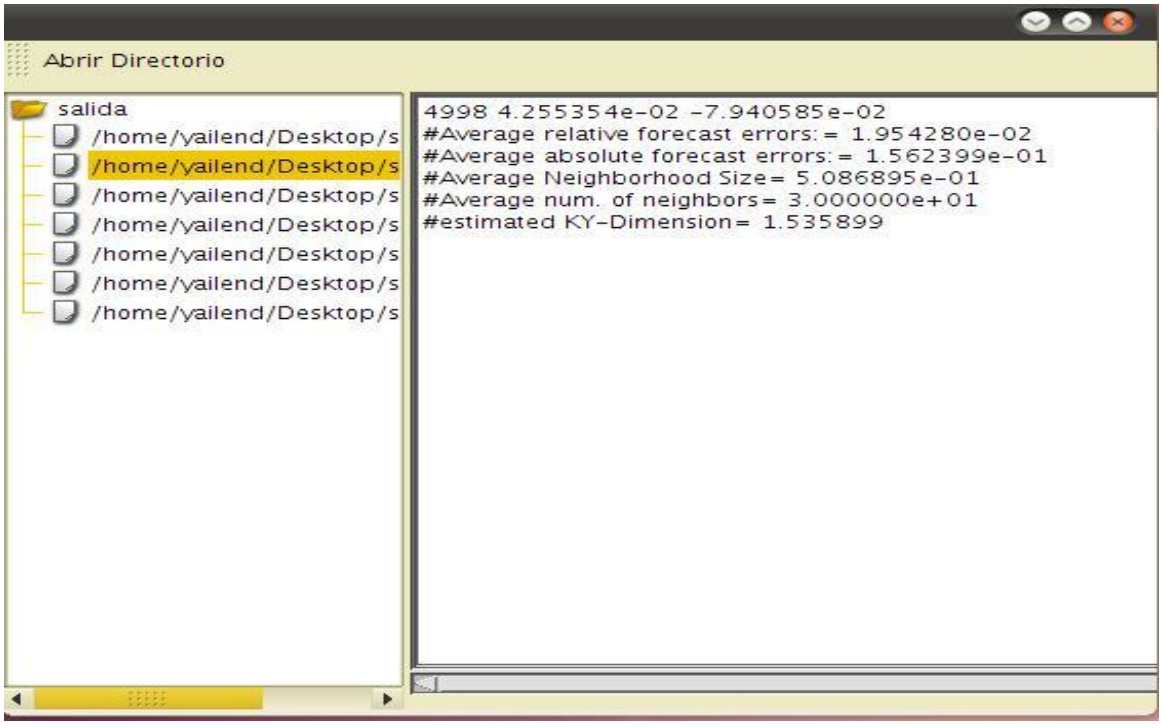


Figura 13: Interfaz donde se muestran los resultados de los ficheros de salida.

4.3. Pruebas del Sistema

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para lograr obtener mejoras en la calidad y representa una revisión final de las especificaciones del diseño y de la codificación. Uno de los objetivos de la fase de pruebas es verificar que el comportamiento del sistema satisfaga los requisitos establecidos por los clientes y futuros usuarios (31).

4.3.1 Plan de prueba

Un plan de pruebas está constituido por un conjunto de pruebas. Cada prueba debe hacerse de acorde al tipo de propiedad que se quiere probar (fiabilidad, rendimiento, seguridad, amigabilidad), como se mide el resultado, especificar en qué consiste la prueba y definir cuál es el resultado que se espera.

4.3.2 Configuración del entorno de prueba

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probarlo se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso. Los requerimientos que se consideraron necesarios para las pruebas se referencian a continuación:

Requerimientos de software:

- PC con Linux (Ubuntu).
- Máquina Virtual de Java 1.6.

Requerimientos de hardware:

- PC con Microprocesador Pentium IV o superior.
- 160 GB de Disco Duro o superior.
- 512 MB de memoria RAM o superior.

Fecha Inicio	Fecha Fin	Actividades	Personas Implicadas
17/5/2010	17/5/2010	Aprobación y firma del Plan de Prueba.	Haymel Odio Domínguez Yailen Delgado Reyes Yunet González Mulet
19/5/2010	19/5/2010	Comprobación de las condiciones previas para iniciar las pruebas.	Haymel Odio Domínguez Yailen Delgado Reyes Yunet González Mulet
Primera Iteración del Plug-in Análisis de Series Temporales			
20/5/2010	20/5/2010	Realización de las pruebas de caja negra para el caso de uso Analizar la serie mediante la técnica Dimensión de Correlación.	Haymel Odio Domínguez Yailen Delgado Reyes
21/5/2010	21/5/2010	Realización de las pruebas de caja negra para el caso de uso Analizar la serie mediante la técnica Ploteo Recurrente.	Haymel Odio Domínguez Yailen Delgado Reyes
22/5/2010	22/5/2010	Realización de las pruebas de caja negra para el caso de uso Analizar la serie mediante la técnica Espectro de Lyapunov.	Haymel Odio Domínguez Yailen Delgado Reyes

4.3.3 Diseño de las pruebas de caja negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba procuran demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto.

Nombre del caso de uso	Descripción	Flujo Central
ESC 1: Analizar la serie mediante la técnica Dimensión de Correlación.	El usuario introduce los datos de configuración (directorio de salida y directorio del TISEAN), selecciona el fichero de entrada, introduce los datos en el formulario y oprime el botón	Selecciona la opción analizar de la Barra de Herramientas del Front-End.

	Analizar. El sistema valida los datos y muestra la salida en un fichero txt.	

Nombre del caso de uso	**Descripción**	**Flujo Central**
ESC 2: Analizar la serie mediante la técnica Ploteo Recurrente.	El usuario introduce los datos de configuración (directorio de salida y directorio del TISEAN), selecciona el fichero de entrada, introduce los datos en el formulario y oprime el botón Analizar. El sistema valida los datos y muestra la salida en un fichero txt.	Selecciona la opción analizar de la Barra de Herramientas del Front-End.

Nombre del caso de uso	**Descripción**	**Flujo Central**
ESC 3: Analizar la serie mediante la técnica Espectro de Lyapunov.	El usuario introduce los datos de configuración (directorio de salida y directorio del TISEAN), selecciona el fichero de entrada, introduce los datos en el formulario y oprime el botón Analizar. El sistema valida los datos y muestra la salida en un fichero txt.	Selecciona la opción analizar de la Barra de Herramientas del Front-End.

Los casos de uso: Analizar la serie mediante la técnica Espectro de Lyapunov, Analizar la serie mediante la técnica Dimensión de Correlación y Analizar la serie mediante la técnica Ploteo Recurrente, realizan el mismo flujo de evento y como consecuencia el diseño de caso de prueba es aplicable a los tres casos de uso, arrojando similares respuestas por el sistema.

Diseño de caso de prueba

Variables:

1-V1 Fichero de entrada

2-V2 Directorio de Salida

3-V3 Directorio del TISEAN

4-V4 Comandos que generan ficheros de salidas

Id del Escenario	Escenario	V1	V2	V3	V4	Respuesta del Sistemas	Resultado de la prueba
ESC 1.1	Analizar la serie temporal	v	v	v	v	Se realizó un análisis satisfactorio.	Se obtiene el fichero de salida con los datos de análisis.
ESC 1.2	Analizar la serie temporal	f	v	v	v	Debe seleccionar un fichero de entrada.	No se obtiene fichero de salida.
ESC 1.3	Analizar la serie temporal	v	f	f	v	Usted no ha configurado correctamente el sistema.	No se configura el sistema.
ESC 1.4	Analizar la serie temporal	v	v	v	f	Ha seleccionado opciones que no generan ficheros de salida.	No se obtiene fichero de salida.

4.4 Conclusiones

En este capítulo se diseñaron los diagramas de componentes correspondientes al plug-in de Análisis de Series Temporales. Se realizaron las pruebas unitarias de caja negra al plug-in obteniendo 2 no conformidades las cuales fueron solucionadas, garantizando así el correcto funcionamiento del plug-in.

CONCLUSIONES

1. Fueron implementadas las interfaces de los requisitos definidos para el plug-in de análisis de series temporales, permitiendo al usuario que la interacción con el sistema fuera amigable.
2. Se incorporó el plug-in a la versión 2.0 de alasBioSyS posibilitando otro tipo de análisis.
3. Se realizaron pruebas de Caja Negra sobre las interfaces que implementan las funcionalidades para comprobar que se cumplieron los requerimientos especificados obteniendo un resultado satisfactorio.

RECOMENDACIONES

1. Realizar la implementación para graficar las salidas proporcionadas por la técnica Ploteo Recurrente.
2. Buscar alternativas para que el plug-in implementado sea utilizado en otras plataformas.

BIBLIOGRAFÍA

1. **González, Carbonell y Miguel, Félix.** *Cálculo numérico de Exponentes de Lyapunov en ecuaciones diferenciales.Tesis presentada en opción al grado científico de Doctor en Ciencias Matemáticas.* Ciudad de la Habana : s.n., Enero 2006.
2. *Biological system interactions Proccedings of the National Academy of Sciences of de United States of America. Physiological Sciences.* **Adomian, G, G. E. A. y. R. E. B.** 81, 1984, Vols. 2938-2940.
3. **Edelstein-Keshet and Escobar, V. H.** *Minería web de uso y perfiles de usuario: Aplicaciones con Lógica Difusa. Tesis de doctorado.* Universidad de Granada.España : s.n., 1988,2007.
4. *Computational physiology and the physiome project. Experimental physiology.* **Crampin, E. J., M. H, P. H, P. N, D. N, N. S y M.** 1-26, 2003, Vol. 89(1).
5. [En línea] [Citado el: 20 de noviembre de 2009.] http://www.mpipks-dresden.mpg.de/~tisean/Tisean_3.0.1/index.html..
6. [En línea] [Citado el: 20 de noviembre de 2009.] http://complexorganizations.blogspot.com/2007/03/tisean-30-ya-disponible.html..
7. **Gröller, E., H. L y R. W.** *Visualization of dynamical systems. Elsevier Science Publishers B. V.* Amsterdam : s.n., 1999.
8. **Rosales Ortega, Jose.** *Sistemas Dinámicos Elementales.* . Instituto Tecnológico de Costa Rica : s.n.
9. [En línea] [Citado el: 20 de noviembre de 2009.] http://www.seh-lelha.org/tseries.htm..
10. **Navarro, J.** *Dinámicas No Lineales: Algunas Técnicas de Análisis y Software Libre. Paper presented at the Ponencia en Congreso Psicología Social.* . 2007.
11. **Marwan, N., M. C. R., M. T. y. J. K.** *Recurrence plots for the analysis of complex systems. Elsevier.Physics Reports.* 2007. 438.
12. *Solución numérica del atractor de Lorenz por el método de Runge Kutta-Fehlberg . Revista de Investigación de Física.* **Llosa, M. y Gómez, J. s.l.** 2004, Vols. Vols. 7(1-2).
13. [En línea] [Citado el: 18 de diciembre de 2009.] www.scc.org.co/.../v13/v13/body/v13n3a6.htm..
15. [En línea] [Citado el: 12 de febrero de 2010.] http://bioinformatics.oxfordjournals.org/cgi/content/abstract/9/5/563..
16. **Garcia J, Rodriguez JI, Vidal J.** *Aprenda Matlab 7.0 como si estuviera en primero.* 2006.
17. [En línea] [Citado el: 24 de febrero de 2010.] http://epf.eclipse.org/wikis/openup/.
18. Características de UML. [En línea] [Citado el: 1 de marzo de 2010.] http://www.abcdatos.com/tutoriales/tutorial/l7157.html..
19. Características de XML. [En línea] [Citado el: 12 de febrero de 2010.] http://www.dcc.uchile.cl/~rbaeza/inf/xml.html .
20. Características de Visual Paradigm . [En línea] [Citado el: 12 de diciembre de 2010.] http://www.versioncero.com/noticia/210/visual-paradigm-for-um..
21. [En línea] [Citado el: 16 de marzo de 2010.] http://www.javahispano.org/contenidos/archivo/37/poo_java.pdf..
22. Conozca el nuevo NetBeans. [En línea] [Citado el: 13 de noviembre de 2009.] http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html..
23. [En línea] [Citado el: 12 de abril de 2010.] http://www.ucbcba.edu.bo/Publicaciones/revistas/actanova/documentos/v2n4/v2.n4.bascon.pdf.

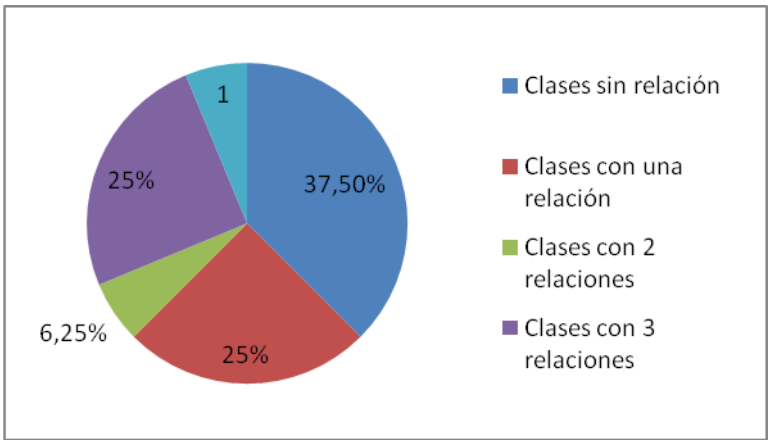
24. [En línea] [Citado el: 12 de abril de 2010.] http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista.shtml#resu.
25. [En línea] [Citado el: 12 de febrero de 2010.] http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista.shtml#resu.
26. **Gamma, Erich, Helm, Richard y Johnson, Ralph.** *Design Patterns: Elements of Reusable Object-Oriented Software. s.l. Pag 34. .* 1995.
27. [En línea] [Citado el: 20 de marzo de 2010.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.
28. **Brena, R.** *AUTOMATAS Y LENGUAJES: un enfoque de diseño. Tecnológico de Monterrey, 1ra edición. .* 2003.
29. **Pressman, Roges S.** Ingeniería de Software un Enfoque Práctico. . [En línea] 2005. [Citado el: 12 de febrero de 2010.] http://biblioteca.uci.cu/bives/titdigitales..
30. **Paz Noa, Eliane.** *Diseño y aplicación de pruebas al producto "Árbol Genealógico.". Universidad de las Ciencias Informáticas.Pág. 20.* Ciudad de la Habana : s.n., 2007.
31. **Martínez Figueredo, Alfredo y Luna Gallardo, Arisbel.** *BioSyS: Módulo de Modelación gráfica de Sistemas Biológicos. Universidad de las Ciencias Informáticas. Pág. 54.* Ciudad de la Habana : s.n., 2008.
32. *COmplex PAthway SImulator. Bioinformatics.* **Hoops, S., S. S., R. G., C. L., J. P., N. S., M. S., L. X., P. M. y U. K.** 2006, Vol. Vol. 22(24).
33. [En línea] [Citado el: 24 de febrero de 2010.] http://OpenUP.com/tutorial.
34. Características de UML. [En línea] [Citado el: 24 de febrero de 2010.] http://www.abcdatos.com/tutoriales/tutorial/l7157.html..
35. [En línea] [Citado el: 12 de abril de 2010.] http://www.ucbcba.edu.bo/Publicaciones/revistas/actanova/documentos/v2n4/v2.n4.bascon.pdf.
36. **Villaverde Martínez Julio Antonio.** *Un nuevo Front-End para la plataforma alasGRATO.Universidad de las Ciencias Informáticas*. Ciudad de la Habana:s,n.,2009.

ANEXOS

Anexo 1: Descripción detallada del patrón Bajo Acoplamiento.

Clases	*Cant Relaciones*	*Porciento*
TiseanController	9	60%
Tecnica	1	6,60%
ComandConfig	3	20%
InputProcess	1	6,60%
Process	1	6,60%
OutputProcess	1	6,60%
DimensionCorrelacion	3	20%
Lyapunov	3	20%
Ploteo	3	20%
ListaSE<T>	2	13,30%
Node <T>	0	0%
Configuration	0	0%
METHODS	0	0%
Comand	0	0%
Opcion	0	0%
UserData	0	0%

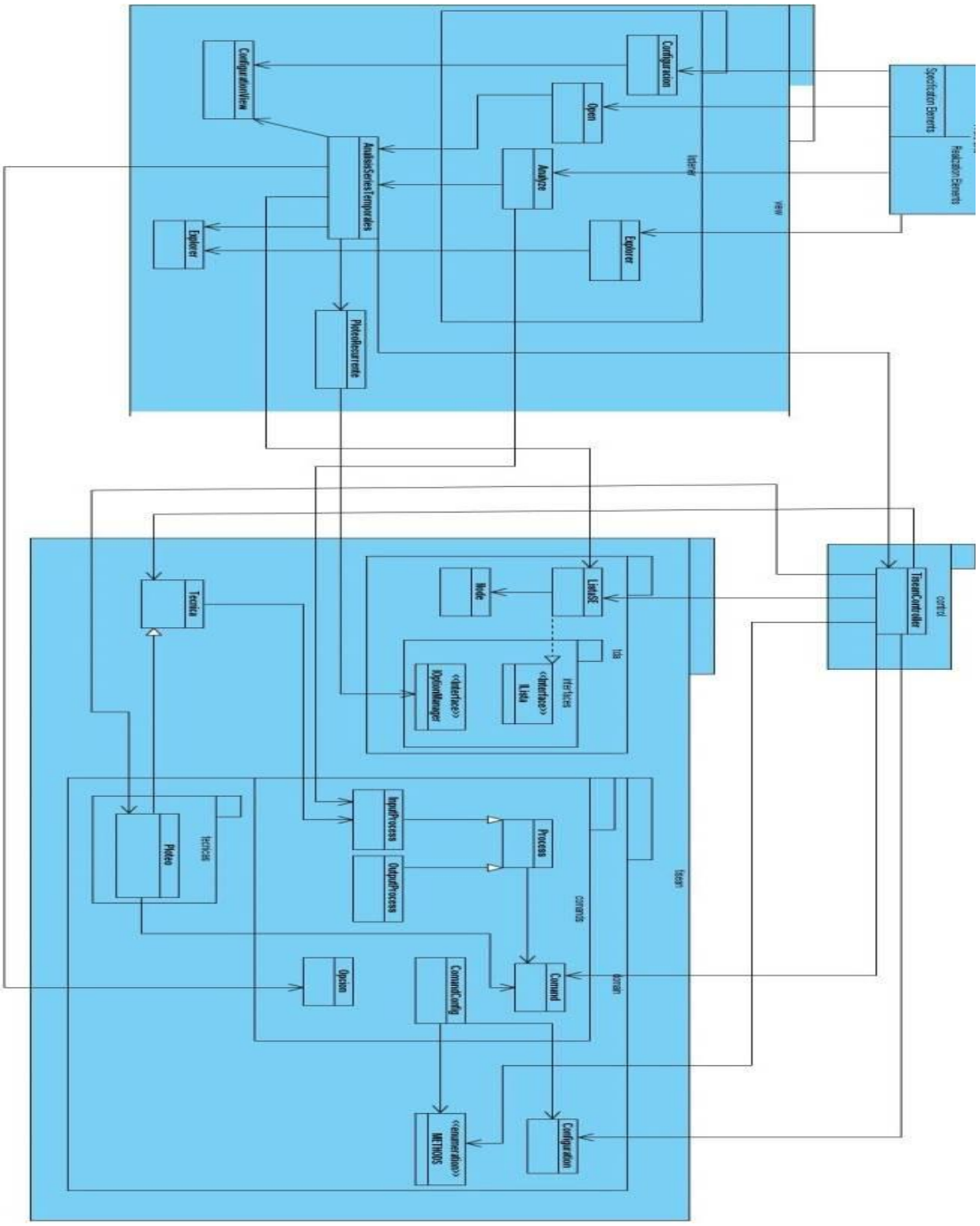
Clases sin relación	6
Clases con una relación	4
Clases con 2 relaciones	1
Clases con 3 relaciones	4
Clases con más de 3 relaciones	1



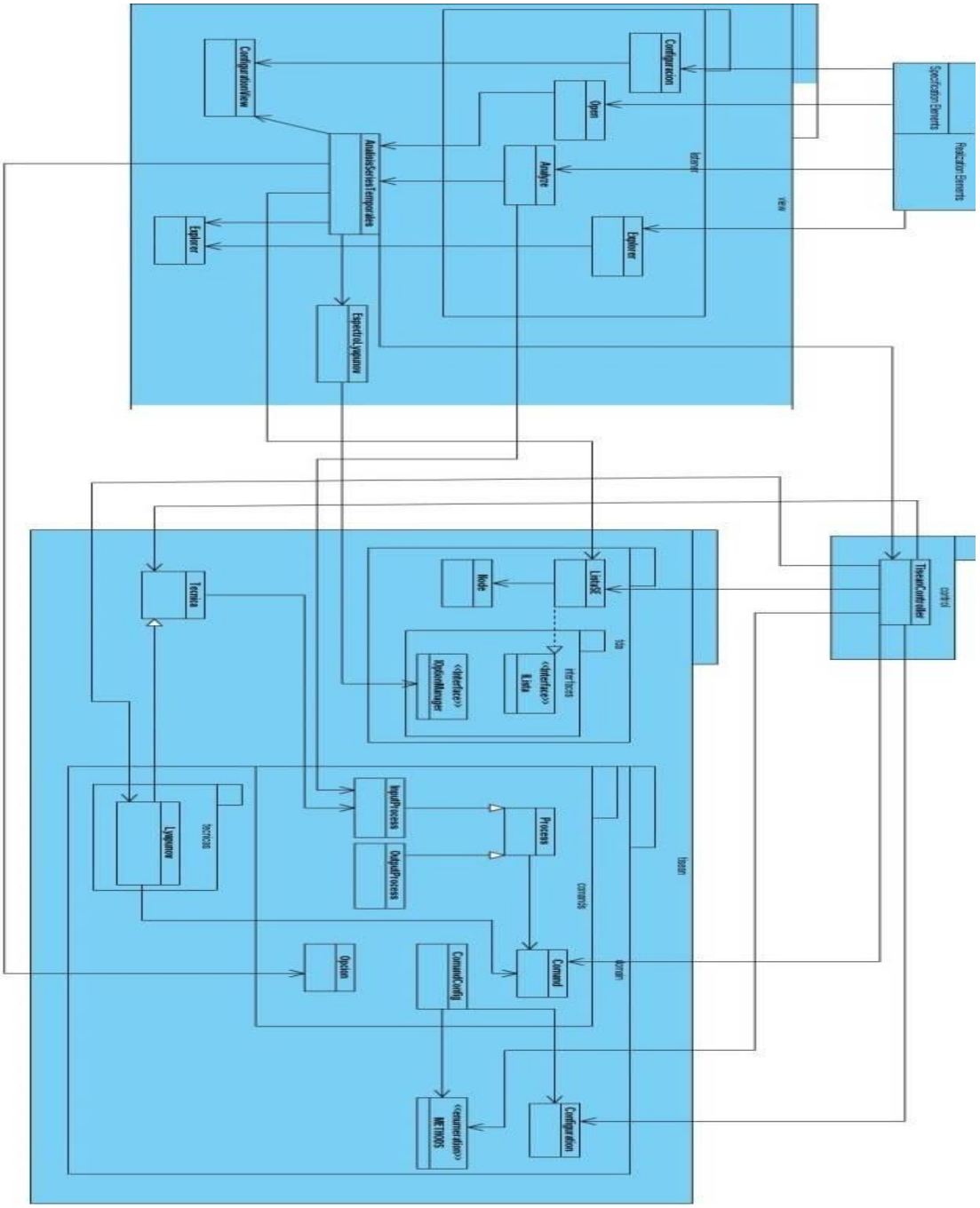
Anexo 2: Descripción detallada del patrón Singleton.

```
/** Creates new form AnalisisSeriesTemporales */
public AnalisisSeriesTemporales() {
    initComponents();
    this.ActivatedIndex=-1;
    this.initCmp();
    tabcontainer.addTab("Espectro Lyapunov", this.formLyapunov);
    tabcontainer.addTab("Dimension de Correlacion", this.formdimension);
    tabcontainer.addTab("Ploteo Recurrente", this.formploteo);
    if(AnalisisSeriesTemporales.instance==null)
    AnalisisSeriesTemporales.instance=this;
}
```

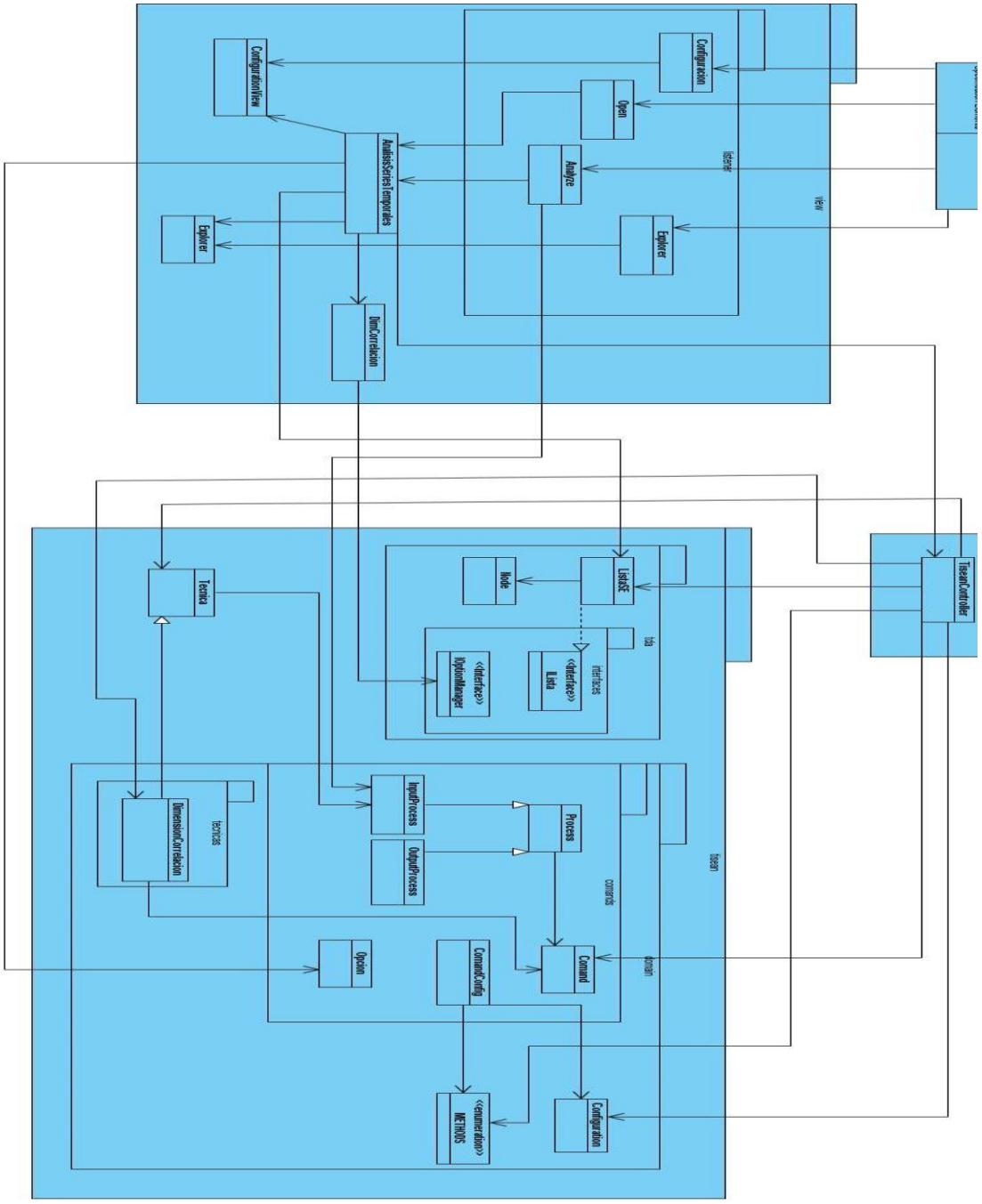
Anexo 3: Diagrama de clases del diseño CU Analizar la serie mediante la técnica Ploteo Recurrente.



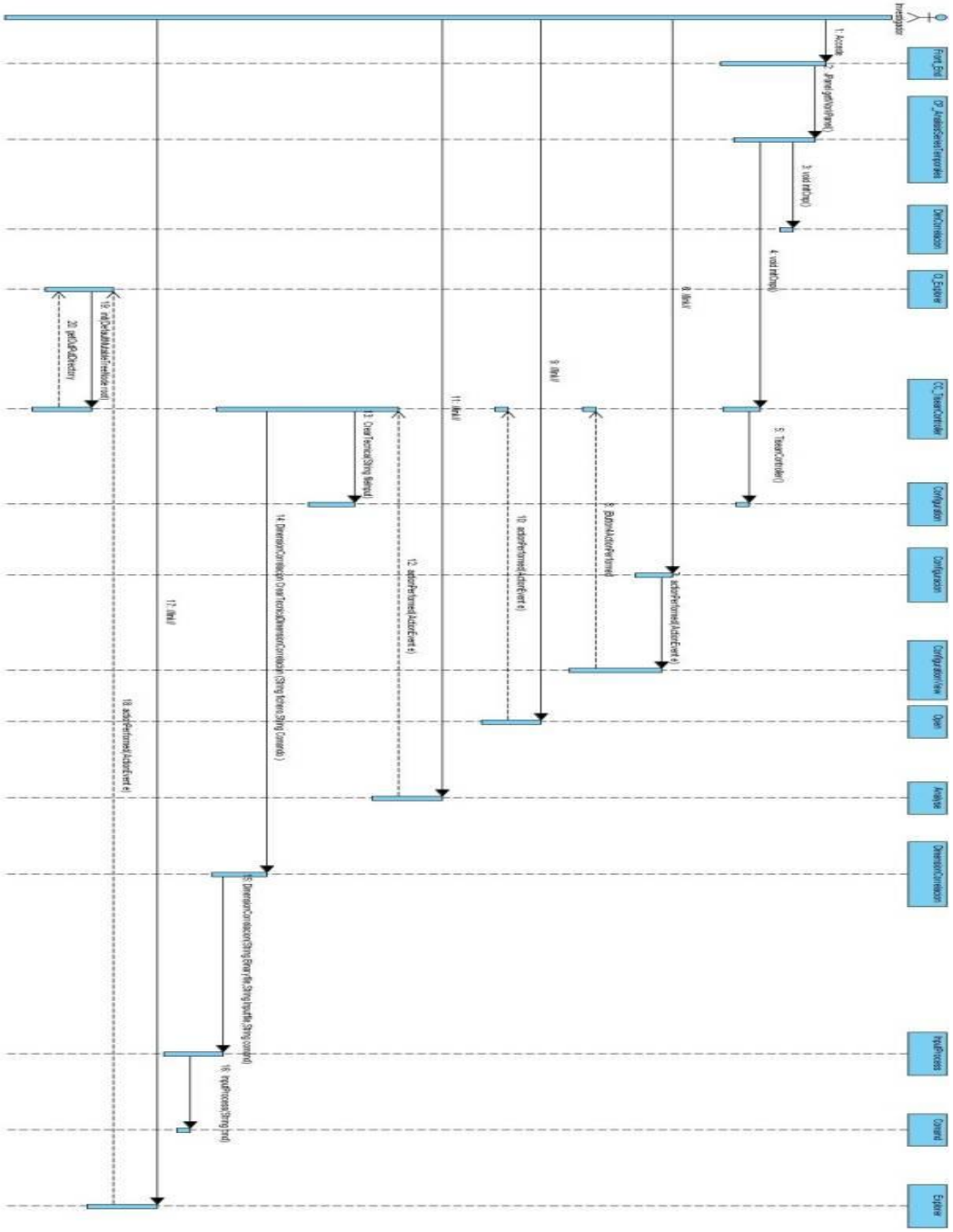
Anexo 4: Diagrama de clases del diseño CU Analizar la serie mediante la técnica Espectro de Lyapunov.



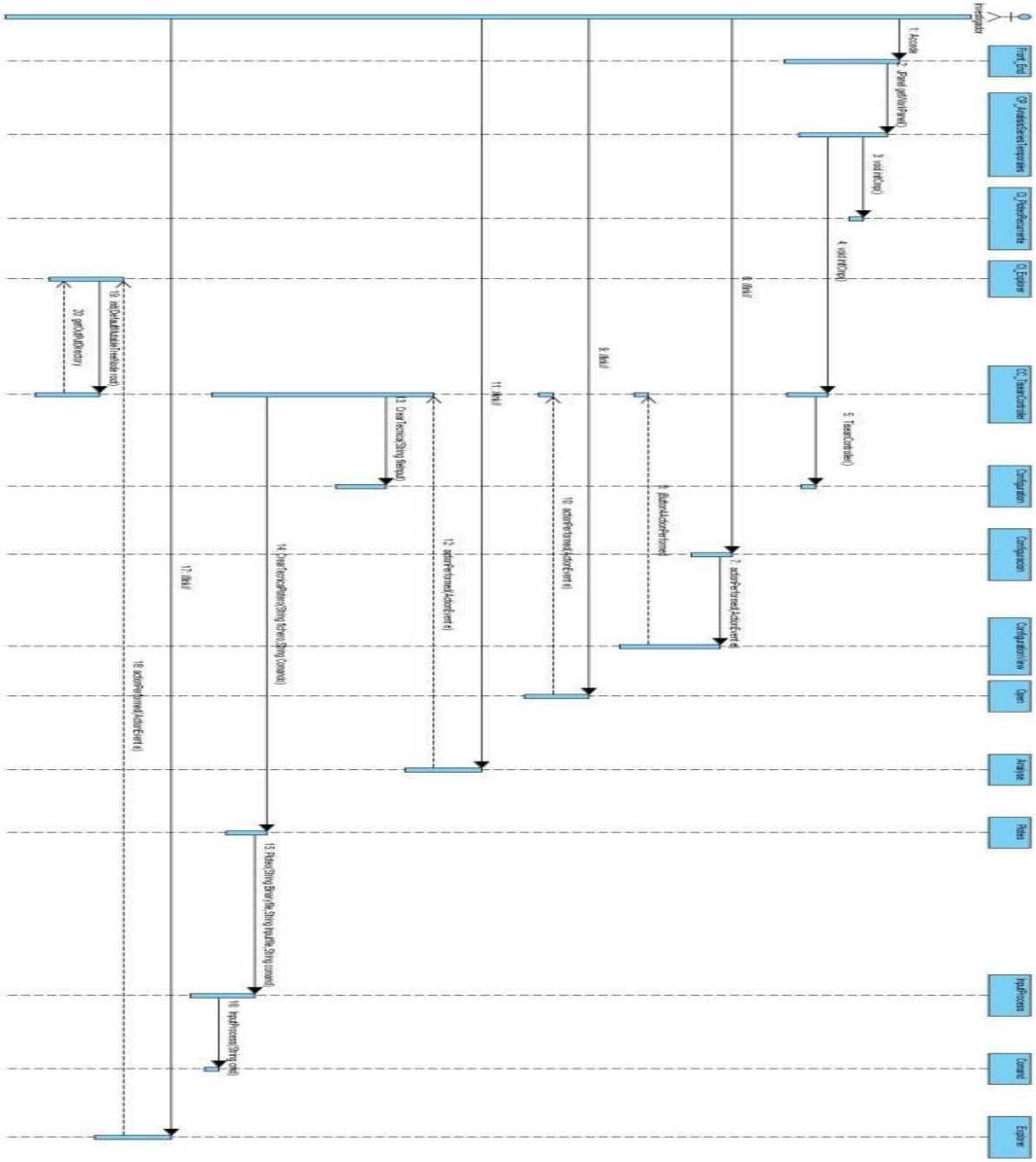
Anexo 5: Diagrama de clases del diseño CU Analizar la serie mediante la técnica Dimensión de Correlación



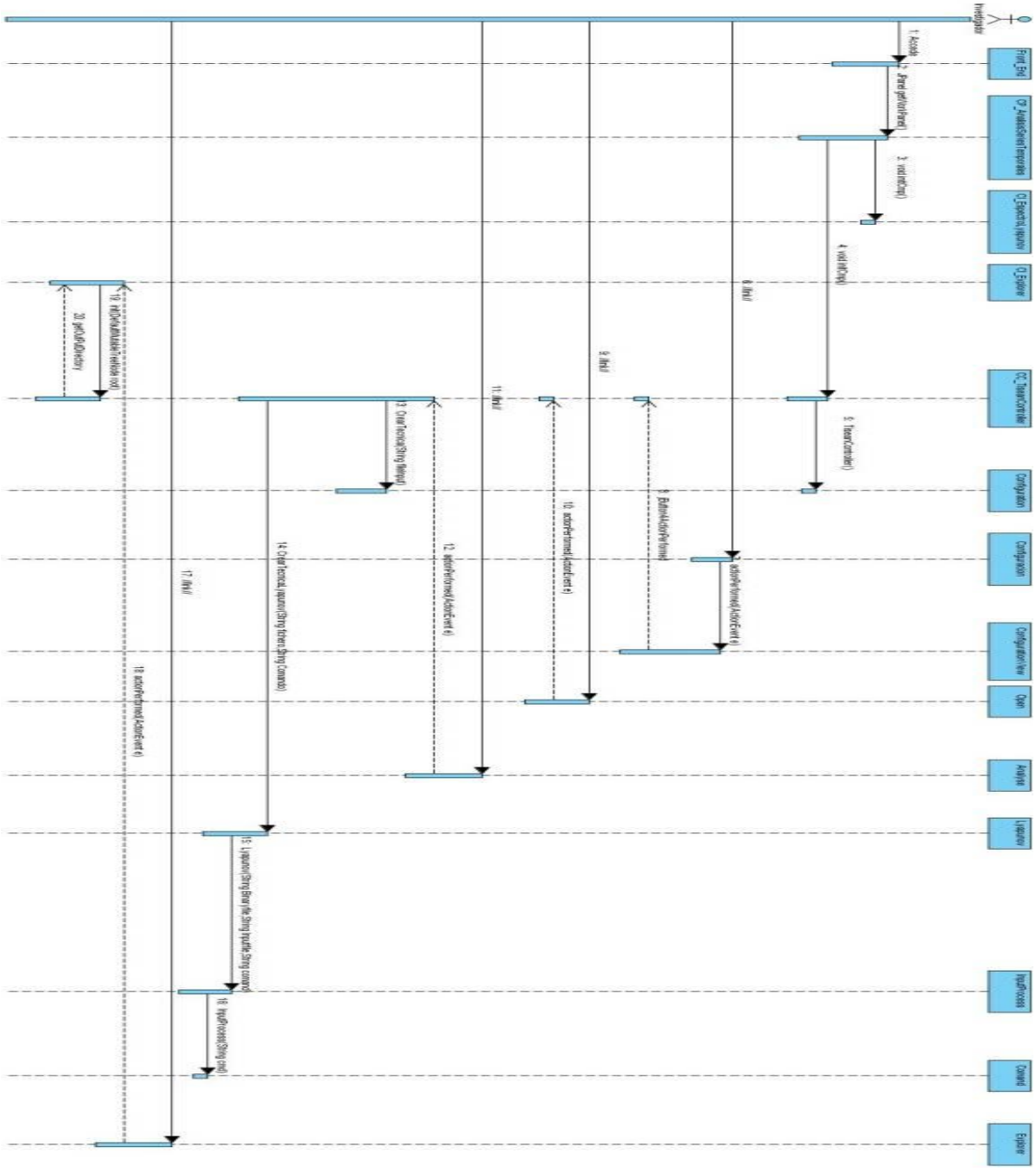
Anexo 6: Diagrama de Secuencia del CU Analizar la serie mediante la técnica Dimensión de Correlación.



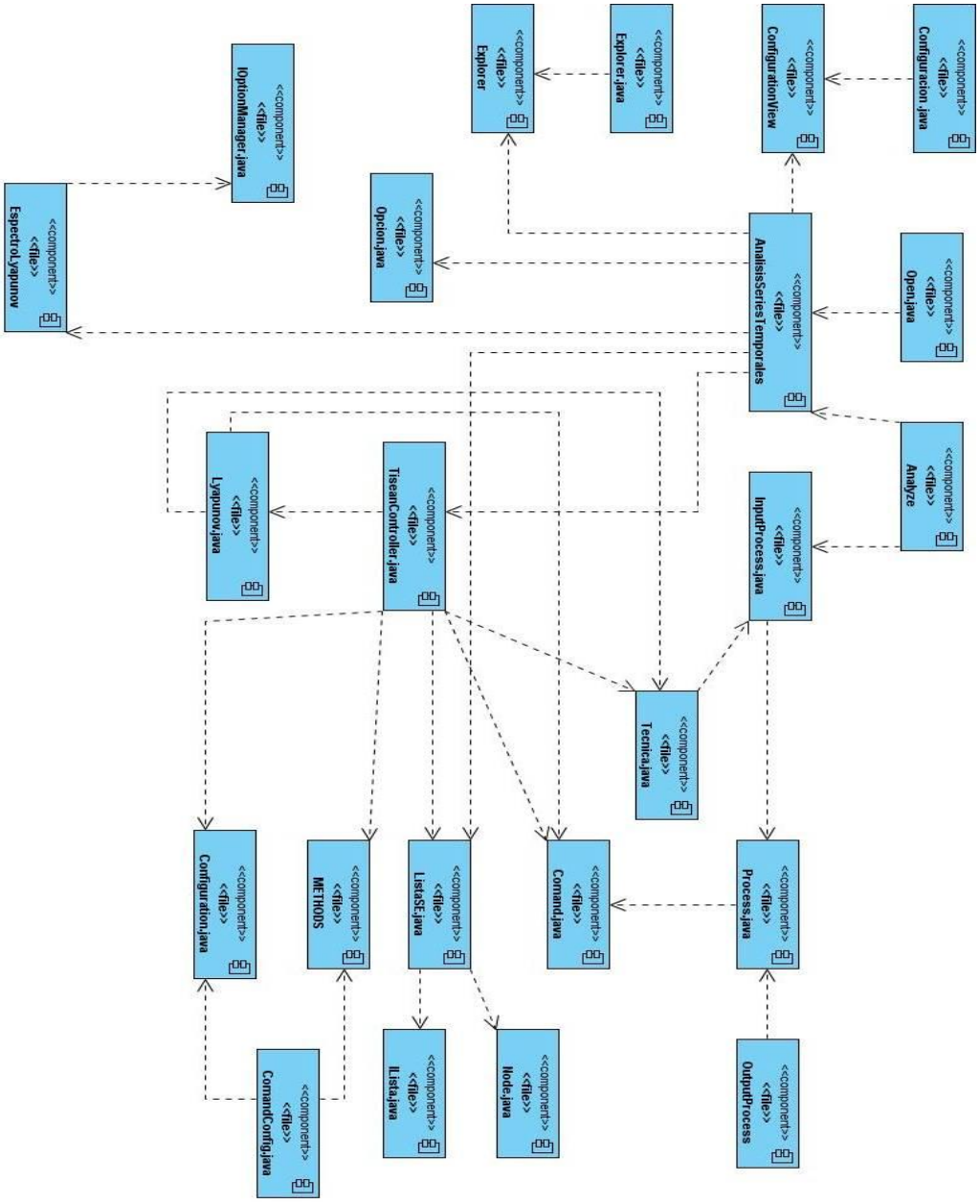
Anexo 7: Diagrama de Secuencia del CU Analizar la serie mediante la técnica Ploteo Recurrente.



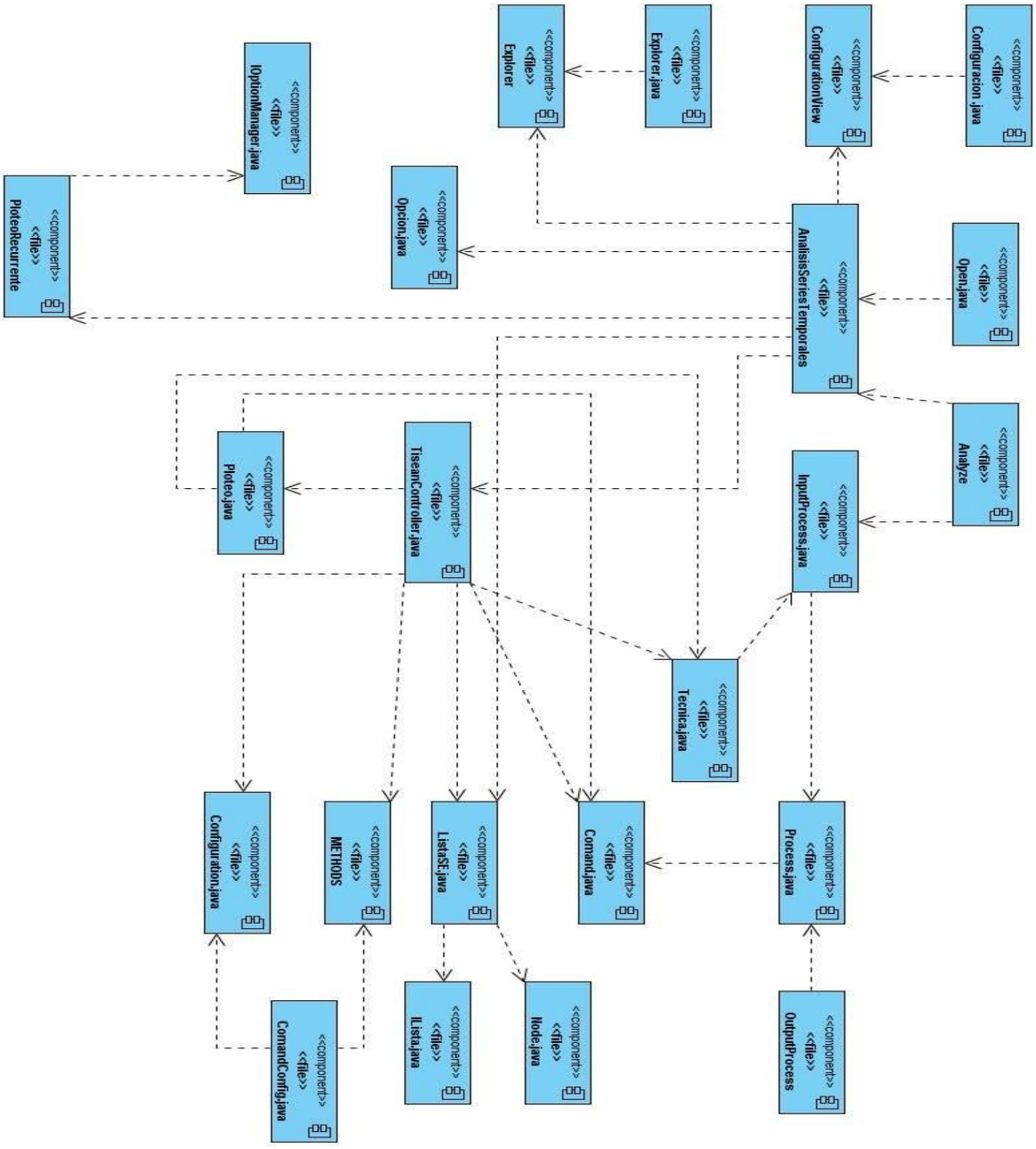
Anexo 8: Diagrama de Secuencia del CU Analizar la serie mediante la técnica Espectro de Lyapunov.



Anexo 9: Diagrama de Componente del CU Analizar la serie mediante la técnica Espectro de Lyapunov.



Anexo 10: Diagrama de Componente del CU Analizar la serie mediante la técnica Ploteo Recurrente.



Anexo 11: Diagrama de Componente del CU Analizar la serie mediante la técnica Dimensión de Correlación.