Sistema multiagente para el estudio y seguimiento técnico de los valores cotizados en el mercado bursátil del Ibex

David Lema Muñiz
Grado de Ingeniería Informática
Ingeniería del Software

Oriol Martí Girona
Santi Caballe Llobet

08/01/2019

Título del trabajo:	*Sistema multiagente para el estudio y seguimiento técnico de los valores cotizados en el mercado bursátil del Ibex*
Nombre del autor:	*David Lema Muñiz*
Nombre del consultor/a:	*Oriol Martí Girona*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega (mm/aaaa):	01/2019
Titulación::	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Ingeniería del Software*
Idioma del trabajo:	*Español*
Palabras clave	*Finanzas, AOP, SOA*

Resumen del trabajo:

En el presente trabajo se desarrollará un modelo de agentes deliberativos basado en las arquitecturas BDI y FIPA según el paradigma de programación AOP, con el fin de obtener un sistema multiagente autónomo y capaz de analizar la información bursátil del Mercado Continuo español (Ibex).

De tal forma, a partir de un conjunto de agentes especializados, el sistema accederá a la información del Ibex para estudiarla y plantear una operativa como oportunidad de inversión. Para ello, el sistema utilizará un servicio web como modelo distribuido de persistencia de datos que implementará siguiendo las especificaciones de la arquitectura REST y de acuerdo al modelo RM-ODP.

Abstract:

The technological proposal with greater acceptance for the design of applications within the scope of artificial intelligence at present is the one based on the paradigm of agent-oriented programming. In this way, the development of intelligent agents as entities with their own personality and capable of acting in a conscious way, presents a set of characteristics that allow to focus software developments towards autonomous models capable of responding dynamically to changes in their environment, showing a flexible behavior as response to unexpected situations, and associate and cooperate jointly with other agents to be more efficient.

The objective of this paper is to develop a multi-agent system with analytical capacity in stock market in accordance with the BDI and FIPA architectures and implemented according the AOP paradigm. This multi-agent system will access information in the Ibex stock market, analyze the information obtained and propose an investment transaction method that can be modified according to the success of the results obtained.

In addition, according to the RM-ODP model based on points of view and the paradigm of component-based programming, a web service based on REST will be built as a data persistence model that will allow the multi-agent system to access the resources remotely and distributed. In this way, system agents will invoke the set of web service interfaces through HTTP and obtain the Ibex market data in XML format.

INDICE

ANEXOS

INTRODUCCIÓN

Desde sus inicios hasta hoy, la complejidad de los desarrollos de los programas de computación ha ido aumentando a medida que daba respuesta a necesidades sociales cada vez más complejas. Así, el alcance de estos desarrollos, junto a la planificación de su coste y su tiempo de construcción, ha ido dando cada vez mayor importancia a los modelos arquitectónicos que facilitaban su estructuración y su desarrollo.

De tal manera, la ingeniería del software ha ido evolucionando hasta definir un conjunto de patrones estructurales o estilos arquitectónicos que mejoran tanto la interacción de los programas con su entorno físico, como facilitan su diseño y posterior revisión, es decir, que permiten, mediante la aplicación de una serie de principios y reglas, organizar las distintas fases de la vida del software y así obtener el mejor rendimiento posible durante su ejecución.

Por otra parte, desde aquellos inicios, también ha ido evolucionando la idea de dar al software capacidad propia de decisión, es decir, también han evolucionado los distintos estilos arquitectónicos que dan soporte al desarrollo de la inteligencia artificial. Así, aunque la disciplina que se encarga de estudiar la inteligencia artificial todavía está en sus primeras fases de su desarrollo, el modelo arquitectónico con mayor aceptación en la actualidad es el basado en agentes inteligentes. El presente trabajo, pretende desarrollar un sistema multiagente basado en este paradigma de la programación.

1. Agentes inteligentes

En economía, un agente[(1)] es cualquier entidad con personalidad propia con capacidad de actuar de forma consciente. Así, el conjunto de agentes económicos está formado por todo aquel actor que es capaz de tomar decisiones, en mayor o menor grado racionales, e interactuar en un mercado, que es el contexto social donde se hacen efectivos estos actos en forma de intercambios.

Ya en el ámbito de la ingeniería del software, si se entiende al mercado como el espacio de dominio que define y delimita la funcionalidad construida, el enunciado anterior es perfectamente trasladable a la definición de agente inteligente, con la paradoja de que los principales actores en economía son los seres humanos (familias, empresas y gobiernos) y en el ámbito del desarrollo del software son los sistemas desarrollados en el campo de la inteligencia artificial (IA).

De tal forma, los agentes inteligentes surgen a partir de la evolución del área de la IA y progresan, a lo largo del tiempo, desarrollando un paradigma de programación propio (AOP o programación orientada a agentes) que absorbe conceptos de paradigmas de programación ya existentes, en particular el paradigma de programación orientado a objetos y el orientado a eventos, que permite el desarrollo de aplicaciones complejas con un alto grado de abstracción mediante el desarrollo de agentes y cuyas características principales son las siguientes:

La autonomía

Los agentes tienen capacidad para actuar de forma consciente, es decir, tienen un cierto grado de control sobre sus acciones para obrar de forma independiente y responder ante cambios en su entorno. Además, por lo general, su comportamiento viene determinado por un conjunto de objetivos o fines, los cuales pueden tener su origen en deliberaciones de carácter racional.

La inteligencia

Los agentes tiene la capacidad de decidir de forma dinámica qué objetivos perseguir y durante cuánto tiempo hacerlo, es decir, son capaces de interpretar y valorar los cambios producidos en su entorno (generalmente a través de eventos) y reinterpretar los objetivos iniciales para continuar con ellos, suspenderlos o incluso abandonarlos para dedicarse a otros nuevos.

Así, algunas de las interacciones entorno/agente no son predecibles en tiempo de diseño, lo que provoca que su comportamiento sea emergente y se resuelva en tiempo de ejecución. Por tanto, los agentes muestran un comportamiento flexible como respuesta a situaciones imprevistas, se adaptan consecuentemente y cobran consciencia de ello.

La interacción (sistemas multiagentes)

Los agentes no actúan solos sino que son entidades especializadas que reparten y distribuyen sus responsabilidades de forma social. De tal forma, son capaces de reconocer la aparición de nuevos agentes, compartir la información, solicitar asistencia, coordinarse y actuar de forma conjunta.

Por otra parte, también es posible el diseño de modelos en competencia, es decir, los distintos agentes compiten entre ellos por cumplir un cometido partiendo de la base de que el conjunto de recursos de su entorno es escaso o limitado.

La habilidad social

Los agentes son capaces de cooperar para realizar algún tipo de trabajo en común con el fin de mejorar y ganar en eficiencia. Para ello, son capaces de dialogar unos con otros, delegar la realización de tareas, organizar los procesos y proponer acuerdos mediante negociación colectiva.

(1) del latín agens, -entis, part. act. de agĕre, hacer

A tal respecto, cabe señalar que son capaces de utilizar modelos de comunicación simbólica y definir formalmente el conjunto de términos del lenguaje a emplear mediante ontologías.

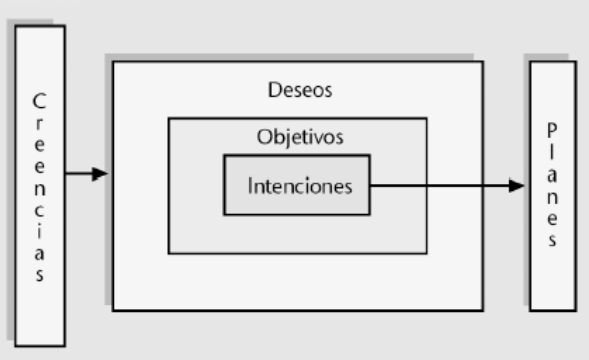
1.1. Agentes deliberativos

Según la forma de responder y reaccionar a los cambios en su entorno, es posible clasificar a los agentes inteligentes en agentes reactivos o agentes deliberativos. Los primeros son aquellos sistemas que no manejan representación simbólica, la información que manejan la reciben principalmente desde sensores y su comportamiento está basado en reglas simples. Los segundos son aquellos sistemas que a partir de una base de conocimientos y un motor de inferencias son capaces de mostrar un comportamiento inteligente.

Arquitectura BDI

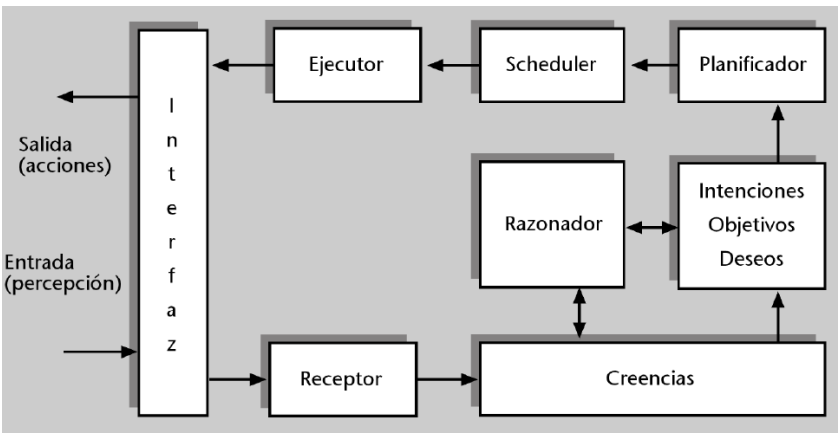
La arquitectura más utilizada en el diseño de los sistemas deliberativos es la arquitectura BDI (Beliefs-Desires-Intentions), formada por tres elementos básicos:

- Las creencias (beliefs), con los conocimientos del agente sobre su entorno
- Los deseos (desires), con los objetivos que se quieren alcanzar
- Las intenciones (intentions), con las acciones a ejecutar para alcanzar estos objetivos



Además de los tres elementos básicos, es posible implementar otros dos elementos: los objetivos y los planes, de tal forma que el primero constituye un subconjunto de los deseos del agente, y el segundo un conjunto de la secuencias de acciones a ejecutar a partir de las intenciones del agente.

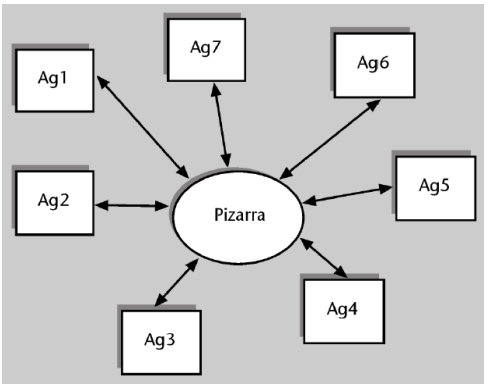
Así, el agente deliberativo recibe o recoge los datos de su entorno, que le sirven para actualizar sus propias creencias (base de conocimientos simbólica), a partir de las cuales, mediante un modelo racional de juicio de valor, infiere sus deseos y objetivos, para finalmente convertirlos en un conjunto de intenciones que le permitan planificar y ejecutar todas aquellas acciones que hagan posible alcanzar su fin último.



1.2. Sistemas multiagente

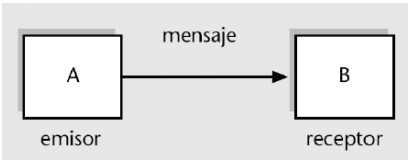
Anteriormente ya se ha visto que una de las características principales de los agentes es su capacidad de interactuar y cooperar de forma conjunta. De tal forma, una de sus habilidades más significativas es su capacidad de comunicación, para ello, las dos técnicas más utilizadas son los sistemas de pizarra y el paso de mensajes.

Los sistemas de pizarra (blackboard systems) se fundamentan en una estructura centralizada de datos (pizarra) a la que todos los agentes acceden para escribir y recuperar información, de tal forma que los agentes no se comunican entre sí sino que utilizan un área común de trabajo para compartir e intercambiar el conocimiento.



Los sistemas de paso de mensajes, sin embargo, se fundamentan en un modelo de comunicación descentralizado en el que cada agente decide con que agente comunicarse. Así, un agente emisor envía un mensaje a otro agente receptor, de tal forma que el mensaje tan solo es accesible por él.

Este tipo de comunicación consigue, por un lado una mayor flexibilidad debido a su arquitectura descentralizada, y por otra parte, abrir una serie de posibilidades en la comunicación más allá del simple hecho de informar. Es decir, los mensajes enviados por el emisor pueden tener diferentes intenciones respecto al receptor: informar, solicitar hacer una acción, abrir una negociación, etc…

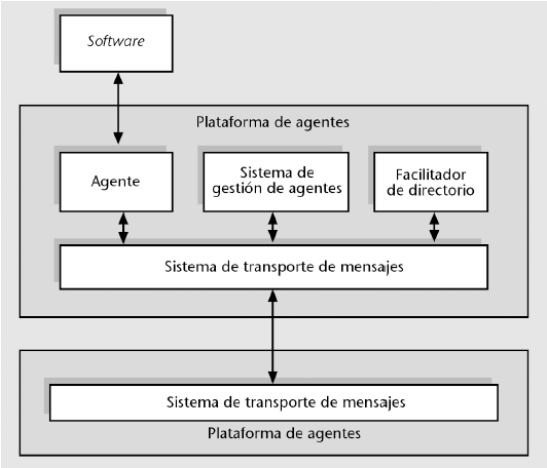


Foundation for Intelligent Physical Agents

FIPA es una organización que propone un conjunto de normas o estándares para el desarrollo de agentes inteligentes que se centran en aspectos como la arquitectura de los sistemas multiagente, el lenguaje o los protocolos de comunicación.

Así, en la arquitectura propuesta por FIPA nos encontramos tres elementos clave:

- La plataforma de agentes, que proporciona la infraestructura básica para la creación y ejecución de los agentes
- El facilitador de directorio (DF), en donde se publican y anuncian los servicios de cada agente que hay disponible en la plataforma
- El sistema de gestión de agentes (AMS), que además de controlar la plataforma, permite la creación y destrucción de agentes



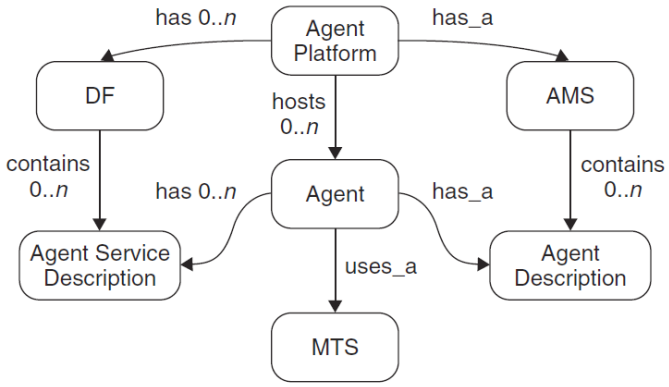
Por otra parte, FIPA proporciona una implementación estándar del lenguaje de comunicación FIPA-ACL (FIPA agent communication language) que está fundamentado en un sistema de paso de mensajes, de tal forma, que define una serie de protocolos de comunicación que definen el conjunto de mensajes a intercambiar para llevar a cabo una acción comunicativa concreta. Entre los protocolos más utilizados están:

- FIPA-Inform, es el más básico y se utiliza para que un agente informe de algo a otro
- FIPA-Query, se utiliza cuando un agente quiere preguntar algo a otro
- FIPA-Request, se utiliza cuando un agente quiere que otro agente haga algo
- FIPA-Contract Net, es el más complejo y se utiliza cuando varios agentes quieren coordinar algún tipo de actividad o negociación. Es habitual utilizar ontologías.

Java Agent Development Framework

JADE es un espacio de trabajo (middleware) distribuido como software libre que fue desarrollado originalmente por Telecom Italia y que, sobre una arquitectura FIPA, proporciona una plataforma para el desarrollo y la coordinación de múltiples agentes.

Así, JADE facilita la gestión y la comunicación multiagente mediante un conjunto de utilidades que permiten crear, registrar, ubicar, migrar u operar con agentes. Para ello, establece como modelo de administración los siguientes componentes:



Agent Platform (AP): proporciona la infraestructura física en la que se implementan los agentes y consta de las máquinas, los sistemas operativos, los componentes de administración FIPA y de los propios agentes. Además, una única AP puede estar distribuida en múltiples dispositivos físicos, de tal forma que los agentes no tienen que estar localizados en una misma ubicación.

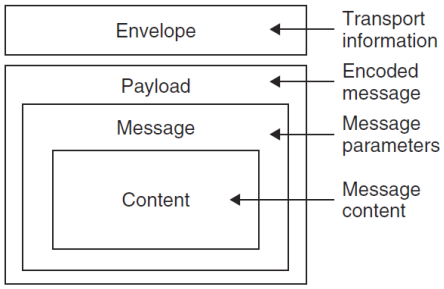
Agente: un agente es un proceso computacional habitado en una AP que ofrece uno o más servicios que pueden estar publicados o no en la propia plataforma. Cada uno de los agentes será

identificado de forma única e inequívoca dentro de la plataforma mediante el identificador de agente de FIPA (AID).

Directory Facilitator (DF): El DF es un componente opcional de una AP que proporciona servicios de páginas amarillas a otros agentes. Así, mantiene una lista de agentes y proporciona información sobre los agentes presentes en la plataforma, de tal forma que aquel agente que desee dar a conocer sus servicios a otros agentes debe solicitar el registro de su descripción de agente. Además, el DF puede restringir el acceso a la información en su directorio y solicitar permisos para su acceso.

Sistema de gestión de agentes (AMS): El AMS es un componente obligatorio y único de una AP ya que, por un lado, es el responsable de administrar su funcionamiento, y por otro lado, tan solo puede existir uno en cada AP, ya que en el caso de que la AP abarque varias máquinas, el AMS será la autoridad en todas ellas. De tal forma, cada agente debe registrarse en un AMS para obtener un AID lo que posibilitará su administración (creación, eliminación y supervisión de estados) y quedará subordinado a él, ya que el AMS puede solicitarle que realice una acción o una operación concreta y tiene la autoridad suficiente para hacer que se lleve a cabo.

Servicio de transporte de mensajes (MTS): El MTS es un servicio proporcionado por una AP para el transporte de mensajes FIPA-ACL entre agentes.



2. Funcionalidad de los agentes

El objetivo general del proyecto es obtener un sistema de agentes autónomos con capacidad de análisis dentro del ámbito técnico bursátil desarrollado a partir de las arquitecturas BDI y FIPA e implementado según el paradigma de programación AOP. Así, el sistema multiagente tendrá que poder acceder a la información de los mercados financieros bursátiles, analizar la información obtenida y plantear una operativa de inversión según el grado de acierto del propio sistema.

Además, su desarrollo se efectuará a partir de los estándares y áreas del conocimiento que PMI propone en su guía PMBOK y según el diseño arquitectónico en términos de puntos de vista propuesto por el modelo RM-ODP y el cumplimiento de los estándares IEE 1471.

2.1. Obtención de los requisitos del sistema

De tal forma, a partir del objetivo general del proyecto, es posible establecer como requisitos generales del producto aquellos que permitan al sistema analizar la información bursátil:

- Conocer el conjunto de empresas que forman el mercado financiero
- Acceder a la cotización actual de cada una de las empresas
- Acceder a la cotización histórica de cada una de las empresas
- Saber calcular un conjunto de indicadores técnicos
- Reconocer y priorizar cada una de las posibilidades de inversión

Así, será necesario, obtener una fuente externa de datos que nos permita acceder a información de los mercados financieros, iniciar las investigaciones necesarias que nos permitan obtener el conocimiento de cómo analizar esa información y establecer un conjunto de indicadores técnicos que nos permitan operar con los datos del mercado analizado.

Servicio externo de datos

Los datos de contenido bursátil será necesario obtenerlos de alguna fuente gratuita de Internet, ya que no se dispone de capacidad financiera para contratar un servicio externo. Así, se ha seleccionado la página web https://www.invertia.com por cumplir con los siguientes requisitos:

- Es posible reconocer el conjunto de empresas que forman un mercado financiero (índice bursátil) y conocer sus cotizaciones actuales con un retraso de quince minutos.
- Una vez conocido el conjunto de empresas que forman un índice, también es posible acceder a su cotización histórica

Obtención de fuentes de conocimiento

Con el fin de obtener tantas propuestas de ideas como fuera posible, se efectuó un brainstorming digital en el que se recogían, a partir de varios blogs de análisis e inversión bursátil, un conjunto de ideas iniciales sobre las que desarrollar la funcionalidad del producto:

- https://www.novatostradingclub.com
- http://www.estrategiasdebolsa.es
- https://www.rankia.com

Cálculo de indicadores técnicos

Para que el sistema sea capaz de analizar datos de contenido bursátil es necesario que maneje una serie de parámetros o indicadores que le permitan realizar este análisis. Más en concreto, el sistema manejará los siguientes indicadores técnicos:

- El precio de cotización, valor monetario de un valor o empresa cotizada
- El precio de entrada de la operación a realizar (compra)
- El precio de salida de la operación a realizar (venta)
- El precio objetivo de la operación a realizar
- La tendencia, que indicará la dirección o el rumbo del precio de un valor
- Los soportes, que indicarán el nivel de precios en el que se espera que la fuerza de compra supere a la fuerza de venta
- Las resistencias, que indicarán el nivel de precios en el que se espera que la fuerza vendedora supere a la compradora
- Indicadores técnicos de soporte: La media móvil exponencial (EMA) y la media móvil de convergencia/divergencia (MACD)

2.1.1. Generación de ideas: Juicio de expertos

Con el fin de establecer un modelo de reconocimiento de posibilidades de inversión sobre un conjunto de indicadores técnicos, se ha llevado a cabo un proceso de generación de ideas, seleccionando a partir de los blogs mencionados anteriormente, una serie de enunciados que servirán como base para la obtención de los distintos requisitos funcionales del producto.

- *Se conoce un soporte o una resistencia relevante cuando se aprecia perfectamente en un marco temporal superior*
- *Una tendencia alcista se define como una sucesión de máximos crecientes y mínimos crecientes*
- *Una tendencia bajista se define como una sucesión de máximos decrecientes y mínimos decrecientes*
- *Para trazar una tendencia alcista hay que unir al menos tres mínimos*
- *Para trazar una tendencia bajista hay que unir al menos tres máximos*
- *Una media móvil bien ajustada resume la tendencia en curso, basta ver el sentido del extremo derecho (si va hacia abajo o va hacia arriba)*
- *Un soporte es un nivel de precio, por debajo del actual, en el que se espera que la fuerza de compra supere a la fuerza de venta y esto haga que el precio se incremente*
- *Una resistencia es un nivel de precio, por encima del actual, en el que se espera que la fuerza de venta supere a la fuerza de compra y esto haga que el precio disminuya*
- *Habitualmente los soportes se corresponden con mínimos anteriores y las resistencias con máximos*
- *Cuando el volumen de contratación es alto hay interés en el valor, cuando es bajo hay poco interés*
- *El movimiento del precio tiende a ser entre soportes y resistencias relevantes*

- *Tendrán más relevancia aquellos soportes que más rebotes tengan y cuanto mayor sea el marco temporal al que pertenezcan*
- *Soportes y resistencias se comportan mejor cuanto más veces se prueban*
- *Un stop loss es un precio de salida que evita pérdidas mayores*
- *Un stop loss lo colocaremos donde no esperamos que llegue el precio, debajo de un soporte relevante*
- *El precio o punto de entrada más conveniente es aquel que se coloca el precio está próximo a un soporte relevante*
- *El precio o punto de entrada se coloca cuando el precio rompe una resistencia*
- *El nivel en el que el precio frena o rebota puede no ser un valor muy definido, sino que más bien es una franja de precios*
- *Los precios redondos se comportan mejor que otros precios como soportes o resistencias*
- *Los gaps o huecos en las cotizaciones se suelen comportar bien como soportes o resistencias*
- *Las velas muy largas (diferencia entre precio máximo y mínimo) se suelen comportar bien como soportes o resistencias*
- *Los retrocesos de Fibonacci suelen ser niveles de soporte*
- *Al operar, no queremos soportes o resistencias relevantes entre zona de compra y venta*
- *Compramos justo por encima del próximo soporte relevante*
- *Sin una estructura adecuada de soportes y resistencias es mejor no operar*
- *Es mejor operar a favor de tendencia*
- *El MACD es un indicador estable que permite identificar si la tendencia tiene fuerza que la respalde*

Así, lo primero que realizaremos es una revisión de todos los enunciados anteriores ordenando y combinando el conjunto de ideas inicial con el fin de obtener una lista de requisitos candidatos:

- *Una tendencia alcista se define como una sucesión de máximos crecientes y mínimos crecientes*
- *Una tendencia bajista se define como una sucesión de máximos decrecientes y mínimos decrecientes*
- *Para trazar una tendencia alcista hay que unir al menos tres mínimos*
- *Para trazar una tendencia bajista hay que unir al menos tres máximos*
- *Una media móvil bien ajustada resume la tendencia en curso, basta ver el sentido del extremo derecho (si va hacia abajo o va hacia arriba)*
- *El MACD es un indicador estable que permite identificar si la tendencia tiene fuerza que la respalde*

Primeramente, hemos agrupado el conjunto de enunciados cuya temática es el reconocimiento de la dirección del precio de un valor, es decir su tendencia (grupo 1 de objetivos). Del conjunto de enunciados anterior se extrae que para definir una tendencia es necesario primeramente distinguir y sintetizar el significado de los siguientes términos: *sucesión de máximos crecientes, sucesión de máximos decrecientes, sucesión de mínimos crecientes, sucesión de mínimos decrecientes, unión de máximos, unión de mínimos, media móvil, MACD*.

- *Se conoce un soporte o una resistencia relevante cuando se aprecia perfectamente en un marco temporal superior*
- *Un soporte es un nivel de precio, por debajo del actual, en el que se espera que la fuerza de compra supere a la fuerza de venta y esto haga que el precio se incremente*
- *Una resistencia es un nivel de precio, por encima del actual, en el que se espera que la fuerza de venta supere a la fuerza de compra y esto haga que el precio disminuya*
- *Habitualmente los soportes se corresponden con mínimos anteriores y las resistencias con máximos*
- *Tendrán más relevancia aquellos soportes que más rebotes tengan y cuanto mayor sea el marco temporal al que pertenezcan*
- *Soportes y resistencias se comportan mejor cuanto más veces se prueban*
- *El nivel en el que el precio frena o rebota puede no ser un valor muy definido, sino que más bien es una franja de precios*
- *Los precios redondos se comportan mejor que otros precios como soportes o resistencias*
- *Los gaps o huecos en las cotizaciones se suelen comportar bien como soportes o resistencias*
- *Las velas muy largas (diferencia entre precio máximo y mínimo) se suelen comportar bien como soportes o resistencias*
- *Los retrocesos de Fibonacci suelen ser niveles de soporte*

Seguidamente, hemos agrupado todos aquellos enunciados cuya temática es el reconocimiento de soportes o resistencias relevantes (grupo 2 de objetivos). De tal forma, es posible hacer una síntesis del alcance del significado de ambos términos a partir de los contenidos siguientes: *mínimos anteriores, máximos anteriores, número de rebotes, marco temporal, franja de precios, precio redondo, gap o hueco, vela larga, retroceso de Fibonacci.*

- *Cuando el volumen de contratación es alto hay interés en el valor, cuando es bajo hay poco interés*

A continuación, el grupo 3 de objetivos con un único enunciado que relaciona el volumen con el interés en el valor. Los términos recogidos son: *volumen, interés alto, interés bajo*.

- *Un stop loss es un precio de salida que evita pérdidas mayores*
- *Un stop loss lo colocaremos donde no esperamos que llegue el precio, debajo de un soporte relevante*
- *El precio o punto de entrada más conveniente es aquel que se coloca el precio está próximo a un soporte relevante*
- *El precio o punto de entrada se coloca cuando el precio rompe una resistencia*
- *Compramos justo por encima del próximo soporte relevante*

Después, el grupo 4 de objetivos que recoge el conjunto de enunciados que definen los distintos precios a tener en cuenta para definir una operativa de inversión: el precio de entrada y los precios de salida, ya sea para cerrar la operativa de forma positiva o para cerrar la operativa con el fin de evitar pérdidas. La terminología recogida en este grupo guarda una muy estrecha relación con la del grupo 2: *precio de entrada, precio de salida, stop loss*.

- *El movimiento del precio tiende a ser entre soportes y resistencias relevantes*
- *Al operar, no queremos soportes o resistencias relevantes entre zona de compra y venta*
- *Sin una estructura adecuada de soportes y resistencias es mejor no operar*
- *Es mejor operar a favor de tendencia*

Y finalmente, en el grupo 5 se han recogido aquellos enunciados que definen la forma de actuar para decidir operar sobre un valor, decisión que se tomará según el estudio del análisis del conjunto de términos obtenidos en los anteriores grupos.

2.1.2. Refinamiento y descomposición del conjunto inicial de ideas

Por otra parte, las características del paradigma AOP hacen que cada uno de los agentes que formarán parte del producto final, debido a su grado de autonomía e identidad propia, se conviertan en parte interesada del propio proyecto. Desde esta perspectiva, es posible resumir el conjunto de requisitos candidatos que definen la funcionalidad del sistema mediante la descomposición de los objetivos generales y del conjunto inicial de ideas:

- Los agentes tienen que poder conocer las empresas de un mercado
  - Los agentes tienen que poder seleccionar un mercado de valores o índice
  - Los agentes tienen que poder acceder a la información de cada valor del mercado
- Los agentes tienen que poder acceder a la cotización de estas empresas
- Los agentes tienen que poder acceder a la cotización histórica de estas empresas
- Los agentes tienen que poder mantener la información actualizada
- Los agentes tienen que poder acceder a herramientas de análisis técnico
  - Los agentes tienen que poder reconocer la tendencia de un valor
    - Los agentes tienen que poder calcular una media móvil
    - Los agentes tienen que poder calcular una media MACD
  - Los agentes tienen que poder reconocer los soportes y resistencias relevantes
    - Los agentes tienen que poder manejar varios marcos temporales
    - Los agentes tienen que poder manejar franjas de precios
    - Los agentes tienen que poder reconocer una colección de mínimos
    - Los agentes tienen que poder reconocer una colección de máximos
    - Los agentes tienen que poder reconocer el número de rebotes
    - Los agentes tienen que poder encontrar precios redondos
    - Los agentes tienen que poder encontrar gaps o huecos en los precios
    - Los agentes tienen que poder encontrar velar largas
    - Los agentes tienen que poder calcular el retroceso de Fibonacci
  - Los agentes tienen que poder reconocer el interés en un valor
    - Los agentes tienen que poder comparar volúmenes de contratación
- Los agentes tienen que poder reconocer posibilidades de inversión
- Los agentes tienen que poder priorizar las posibilidades de inversión
- Los agentes tiene que poder operar en un valor (comprar y vender)
  - Los agentes tienen que poder reconocer el precio de entrada o inversión
  - Los agentes tienen que poder reconocer el precio de salida
    - Los agentes tienen que poder reconocer un precio de salida con beneficios

        - Los agentes tienen que poder reconocer un stoploss
- Los agentes tiene que poder comunicarse entre ellos en tiempo real
- Los agentes tiene que poder recordar sus operaciones en cada valor
- Los agentes tiene que poder sacar conclusiones de la operativa realizada

Una vez obtenida la lista de requisitos candidatos es necesario, para revisar el conjunto de agentes definidos en la fase inicial y poder asignarles roles, volver a analizar la lista de requisitos obtenida. Para ello, utilizaremos una estrategia bottom-up y los agruparemos por temáticas:

- Grupo 1: Obtener la información del mercado
    - Los agentes tienen que poder conocer las empresas de un mercado
    - Los agentes tienen que poder acceder a la cotización de estas empresas
    - Los agentes tienen que poder acceder a la cotización histórica de estas empresas
    - Los agentes tienen que poder mantener la información actualizada

- Grupo 2: Analizar la información obtenida
    - Los agentes tienen que poder acceder a herramientas de análisis técnico
    - Los agentes tienen que poder reconocer posibilidades de inversión
    - Los agentes tienen que poder priorizar las posibilidades de inversión

- Grupo 3: Abrir una operativa en el mercado
    - Los agentes tiene que poder operar en un valor (comprar y vender)

- Grupo 4: Revisar las operaciones realizadas
    - Los agentes tiene que poder comunicarse entre ellos en tiempo real
    - Los agentes tiene que poder recordar sus operaciones en cada valor
    - Los agentes tiene que poder sacar conclusiones de la operativa realizada

Grupo 1: Obtener la información del mercado

La obtención de información de un mercado financiero es un proceso repetitivo y constante ya que es necesario tener la información siempre actualizada. Además, la información se obtiene de un servicio externo (http://www.invertia.com) por lo que habrá que conectarse con él, seleccionar y filtrar la información necesaria, y finalmente almacenarla en una BBDD.

Así, la preferencia será constituir una abstracción de los procesos de acceso al servicio web para poder acceder a ellos de forma remota mediante identificadores URI y a través del protocolo de comunicaciones HTTP, construir una la base de datos para almacenar la información y finalmente, recogerla mediante estructuras de datos XML.

Grupo 2: Analizar la información obtenida

El análisis de la información y el reconocimiento de posibilidades de inversión también es un proceso repetitivo y constante que seleccionará cíclicamente la información histórica de un valor y la estudiará mediante la aplicación de un conjunto de herramientas de análisis técnico.

Así, entre el conjunto de indicadores técnicos estará compuesto por:

- La media móvil exponencial (EMA)
- La media móvil de convergencia/divergencia (MACD)
- Cálculo de soportes y resistencias

<u>Media móvil exponencial (EMA)</u>

Si $X=\{x_1, x_2, x_3, \ldots, x_N\}$ es un conjunto de N valores numéricos, la media móvil de n periodos de X se calcula según el promedio de los n últimos valores de X:

$$\textit{Media móvil simple: } SMA_n = 1/n\sum_{i=1}^{n} x_i, \textit{ con } n \leq N$$

De tal forma, una media móvil descarta los valores más antiguos y promedia los n valores más recientes. Por tanto, una media móvil exponencial otorga un mayor peso al dato más reciente lo que hace que el promedio se ajuste mejor al último dato y en consecuencia se ajusten con mayor rapidez que una media simple. Así, definimos α como la constante de ajuste tal que α = 2/ (n + 1), donde n es el número de periodos de cálculo de la media.

$$\textit{Media móvil exponencial: } EMA_n = \alpha * \textit{Precio actual} + (1- \alpha) * EMA_{n-1}$$

Por otra parte, se establece un número de periodos de n= 8 para su cálculo, de tal forma que cuando el número de periodos coincida con el tamaño de la muestra (primer cálculo de la media móvil con n= N) se calculará como una media móvil simple (SMA). Su interpretación técnica es la de resumir la tendencia de los precios.

Media móvil de convergencia/divergencia (MACD)

El MACD es otro tipo de media móvil que consta de tres componentes:

- El MACD, que se calcula como la diferencia entre dos medias móviles exponenciales de diferente número de periodos: una media móvil rápida y más sensible a los movimientos del precio en el corto plazo (n=12) y otra media más lenta de mediano plazo (n=26)

$$MACD = EMA_{12} - EMA_{26}$$

- La Señal, que se calcula como la media móvil exponencial del MACD calculado anteriormente. El número de periodos que usaremos será n=9

$$Señal = EMA_9(MACD)$$

- Y el Histograma o MACDh, que se calcula como la diferencia entre el MACD y la Señal

$$MACDh = MACD - Señal$$

Su interpretación técnica se hará según el valor del MACDh, o del MACD y la señal:

- Cuando el MACDh > 0, entendemos que la tendencia al alza tiene fuerza
- Cuando el MACDh < 0, entendemos que la tendencia ya no tiene fuerza
- Cuando el MACDh crece, entendemos que hay posibilidad de operar con la tendencia
- Cuando el MACDh decrece, entendemos que es mejor no operar
- Cuando el MACD cruza hacia arriba la señal, lo entendemos como señal de compra
- Cuando el MACD cruza hacia abajo la señal, lo entendemos como señal de venta
- Cuando MACD y señal superan el cero, lo entendemos como señal de compra
- Cuando MACD y señal pasan a negativo, lo entendemos como señal de venta

Soportes y resistencias

Los soportes y las resistencias son niveles de precios en los que la cotización cambia su tendencia actual por la contraria. Además, se parte de la idea de que habitualmente coinciden con máximos o mínimos anteriores, es decir, se parte de la idea de que un precio que empieza en cero y sube alcanzará un máximo en el momento que retroceda y ese máximo será la resistencia de la cotización del valor.

De tal forma, para el cálculo de los soportes y las resistencias, el sistema primeramente revisará el histórico de un valor buscando sus máximos y sus mínimos. Para ello, se establecen dos marcos

temporales: cotizaciones semanales (último precio de la semana) y cotizaciones diarias, y dentro de las diarias, tres tamaños de muestras de datos: nueve, seis y tres meses (las cotizaciones semanales se calcularán siempre sobre un periodo de nueve meses). El siguiente algoritmo representa de forma más detallada la búsqueda de una colección de máximos:

```
1.  Inicio algoritmo búsqueda máximos
2.    Seleccionar el marco temporal;
3.    Seleccionar el tamaño de la muestra;
4.    max = 0;
5.    Hacer lectura de datos historicos i veces;
6.      Si maxi >max;
7.        Guardar MAX[j] = maxi;
8.      Fin si;
9.      max = maxi;
10.   Fin hacer;
11. Fin algoritmo
```

Una vez obtenidos los máximos y los mínimos, tendremos que comprobar cuantas veces sirvieron como soporte o como resistencia de los precios, es decir, tendremos que agruparlos y revisar cuantas veces un mismo precio ha sido máximo o mínimo, ya que allí donde el precio rebota al alza hay un mínimo, y allí donde el precio rebota a la baja hay un máximo.

Para ello, primeramente, se revisarán los valores obtenidos con el fin de trabajar con franjas de precios. Así, se aplicará un factor precisión al conjunto de máximos y mínimo, establecido inicialmente en f=±0,5%, de tal forma que el rango de máximos vendrá dado por la expresión (max-f*max, max+f*max).y el de mínimos por (min-f*min, min+f*min).

Finalmente, contaremos el número de veces que la colección completa de máximos y mínimos se ha probado (está dentro de un rango de precios) para, por último, seleccionar todos aquellos contadores que se encuentren por encima de un factor de precisión p = promedio(contador) +1.

```
1.  Inicio algoritmo búsqueda niveles
2.    MAX[] = buscar máximos;
3.    MIN[] = buscar mínimos;
4.    MAXMIN[] = MAX[] U MIN[];
5.    f = 0,5%;
6.    RANGOS[] = CalcularRangos(MAXMIN, f);
7.    CONT[] = ContarPertenenciaRango(RANGOS, MAXMIN);
8.    p = Promedio(CONT) + 1;
9.    Hacer por cada máximo o mínimo en MAXMIN;
10.     Si conti > p;
11.       NIVELES[j] = MAXMIN[i]
12.     Fin si
13.   Fin hacer
14. Fin algoritmo
```

Por otra parte, además de analizar la información de un valor, es necesario comunicar al sistemas cuando se haya encontrado una posibilidad de inversión (análisis positivo) y almacenar los datos más significativos del análisis para en un futuro poder revisarlos, valorarlos e intentar sacar algún tipo de conclusión de ellos.

Grupo 3: Abrir una operativa en el mercado

Una vez analizada la información y obtenida alguna posibilidad de inversión es necesario materializarla y operar sobre ella. Para ello, el sistema tendrá que reconocer cuando el resultado de un análisis es positivo, recuperar los datos del análisis y revisar si el conjunto de indicadores mantienen vigente su posibilidad para operar respecto a la cotización del valor actual.

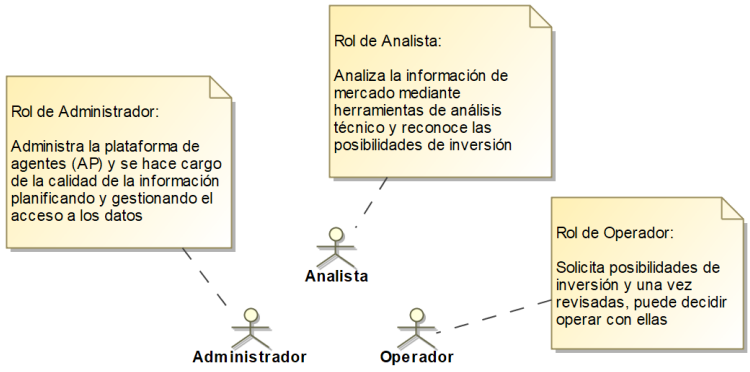
En caso afirmativo, el sistema debe calcular los precios de salida de la operación, el de ganancia y el de pérdida (stoploss) para abrir una operación de inversión. Una vez abierta, el sistema debe hacer un seguimiento de la operación hasta que decida cerrarla, así, tanto el proceso de seguimiento de operaciones como el de reconocimiento de la existencia de análisis positivos serán procesos de carácter repetitivo, de tal forma, que el sistema podrá elegir entre mantener una operación abierta o sustituirla por una nueva con una mejor perspectiva de rentabilidad.

Por último, el sistema informará y almacenará el conjunto de operaciones realizadas con sus resultados, enlazándolas con el estudio previo realizado (análisis).

Grupo 4: Revisar las operaciones realizadas

Además, el sistema está modelado desde una perspectiva de IA por lo que su finalidad es intentar aprender en cierta medida de los aciertos y errores cometidos en el conjunto de operaciones realizadas. De tal forma, los resultados de la aplicación del modelo de cálculo y decisión planteado (análisis y operaciones) deben persistir para poder estudiarse en el futuro.

Aún así, queda fuera del alcance de este proyecto la revisión de la calidad del sesgo del conjunto de indicadores aplicados, el estudio de su correcta parametrización (conjunto de indicadores técnicos analizados) y el análisis probabilístico de los resultados obtenidos.



Del análisis anterior, es posible reconocer el conjunto de roles de los agentes del sistema:

- Rol de administrador, para gestionar los aspectos generales del sistema como la planificación de sucesos repetitivos, el acceso a los datos, su distribución, almacenamiento, etc…
- Rol de analista, para analizar la información de un mercado y calcular el conjunto de indicadores técnicos establecido
- Rol de operador, para operar según las recomendaciones recibidas de los analistas y seguir la evolución de cada operación abierta

Por otra parte, a partir de la lista inicial de objetivos generales y requisitos candidatos también es posible conseguir los requisitos a nivel de datos necesarios para el desarrollo el producto:

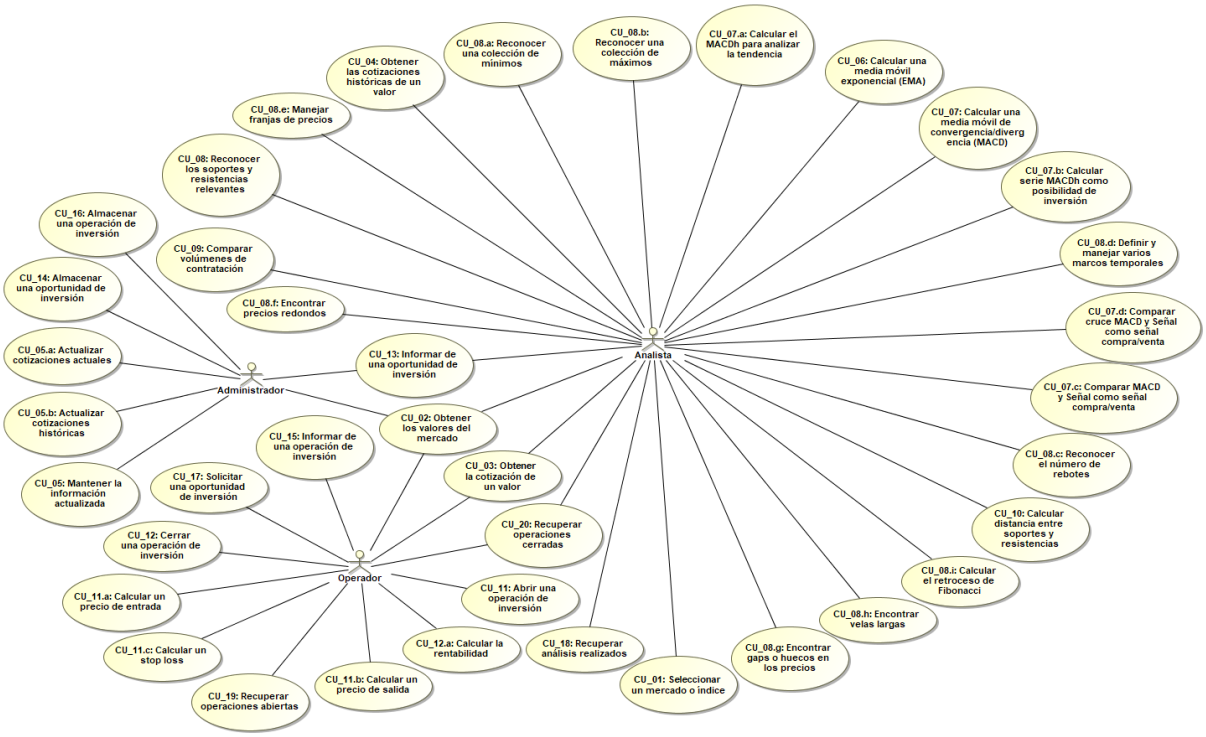
- Los agentes tienen que poder seleccionar un mercado de valores o índice
  - El sistema tiene que poder identificar un índice, conocer su descripción e identificador externo
- Los agentes tienen que poder acceder a la información de cada valor del mercado
  - El sistema tiene que poder identificar un valor dentro de un índice, conocer su nombre y su identificador externo para poder extraer la información de Internet
- Los agentes tienen que poder acceder a la cotización de cada valor del mercado
  - El sistema tiene que poder conocer el precio de mercado actual de un valor identificando la fecha y la hora de la cotización, la variación del precio respecto al día anterior y el volumen contratado
- Los agentes tienen que poder acceder a la cotización histórica de cada valor del mercado
  - El sistema tiene que poder conocer por cada día de cotización el precio de apertura del valor, el precio de cierre, el máximo alcanzado, el mínimo y el volumen cotizado
- Los agentes tienen que poder reconocer las posibilidades de inversión

    - El sistema tiene que poder identificar por cada valor su cotización, sus soportes y resistencias más relevantes, las medias móviles exponenciales de los precios y volúmenes, y el MACD
- Los agentes tiene que poder operar en un valor (comprar y vender)
    - El sistema tiene que poder reconocer el tipo de operación efectuada, el precio de entrada, el precio de salida, el precio stoploss, el precio de cierre definitivo y la rentabilidad obtenida
- Los agentes tiene que poder comunicarse entre ellos en tiempo real
    - El sistema tiene que poder clasificar los mensajes por tipo y poder encapsular la información analizada y la información de la operativa bursátil llevada a cabo

Y finalmente, con el fin de describir la funcionalidad del sistema mediante casos de uso e historias de usuario, revisamos, numeramos y relacionamos con un rol de agente los requisitos obtenidos:

Identificador	**Descripción**	**Agentes**
CU_01	Seleccionar un mercado o índice	An
CU_02	Obtener los valores del mercado	Ad, An, Op
CU_03	Obtener la cotización de un valor	An, Op
CU_04	Obtener las cotizaciones históricas de un valor	An
CU_05	Mantener la información actualizada	Ad
CU_05.a	Actualizar cotizaciones actuales	Ad
CU_05.b	Actualizar cotizaciones históricas	Ad
CU_06	Calcular una media móvil exponencial (EMA)	An
CU_07	Calcular una media móvil de convergencia/divergencia (MACD)	An
CU_07.a	Calcular el MACDh para analizar la tendencia	An
CU_07.b	Calcular el MACDh (serie) para analizar la posibilidad de inversión	An
CU_07.c	Comparar el MACD y la Señal para analizar señales de compra/venta	An
CU_07.d	Comparar el cruce del MACD y la Señal para analizar señales de compra/venta	An
CU_08	*Reconocer los soportes y resistencias relevantes*	An
CU_08.a	Reconocer una colección de mínimos	An
CU_08.b	Reconocer una colección de máximos	An
CU_08.c	Reconocer el número de rebotes	An
CU_08.d	Definir y manejar varios marcos temporales	An
CU_08.e	Manejar franjas de precios	An
CU_08.f	Encontrar precios redondos	An
CU_08.g	Encontrar gaps o huecos en los precios	An
CU_08.h	Encontrar velas largas	An
CU_08.i	Calcular el retroceso de Fibonacci	An
CU_09	Comparar volúmenes de contratación	An
CU_10	Calcular distancia entre soportes y resistencias	An
CU_11	Abrir una operación de inversión	Op
CU_11.a	Calcular un precio de entrada	Op
CU_11.b	Calcular un precio de salida	Op
CU_11.c	Calcular un stoploss	Op
CU_12	Cerrar una operación de inversión	Op
CU_12.a	Calcular la rentabilidad	Op
CU_13	Informar de una oportunidad de inversión	Ad, An
CU_14	Almacenar una oportunidad de inversión	Ad
CU_15	Informar de una operación de inversión	Op
CU_16	Almacenar una oporación de inversión	Ad
CU_17	Solicitar una oportunidad de inversión	Op
CU_18	Recuperar análisis realizados	An
CU_19	Recuperar operaciones abiertas	Op
CU_20	Recuperar operaciones cerradas	An, Op

Agentes -> Ad: Administrador, An: Analista, Op: Operador



2.2. Gestión de los requisitos funcionales

Una vez obtenido el conjunto de requisitos candidatos, seleccionaremos de todos ellos cuales van a formar parte del conjunto definitivo de requisitos para el diseño del producto según el valor que aporten, su coste en desarrollo y el riesgo que supone eliminarlo para el propio producto.

2.2.1. Estimación de los requisitos

Estimaremos el coste de cada requisito utilizando como unidad de medición la facilidad de desarrollo, en términos de velocidad, de cada uno de ellos. Así, siguiendo el modelo que propone Mike Cohn en su libro *Agile Estimating and Planning* asignaremos primeramente a dos de los requisitos candidatos un valor de la escala (de menor dificultad a mayor): 1, 2, 3, 5 y 8, y posteriormente compararemos por triangulación la valoración asignada con el resto de requisitos:

Identificador	Descripción	Estimación
CU_01	Seleccionar un mercado o índice	3
CU_02	Obtener los valores del mercado	3
CU_03	Obtener la cotización de un valor	3
CU_04	Obtener las cotizaciones históricas de un valor	3
CU_05	Mantener la información actualizada	3
CU_05.a	Actualizar cotizaciones actuales	3
CU_05.b	Actualizar cotizaciones históricas	3
CU_06	Calcular una media móvil exponencial (EMA)	2
CU_07	*Calcular una media móvil de convergencia/divergencia (MACD)*	*8*
CU_07.a	Calcular el MACDh para analizar la tendencia	5
CU_07.b	Calcular el MACDh (serie) para analizar la posibilidad de inversión	8
CU_07.c	Comparar el MACD y la Señal para analizar señales de compra/venta	5
CU_07.d	Comparar el cruce del MACD y la Señal como señal de compra/venta	8
CU_08	*Reconocer los soportes y resistencias relevantes*	*8*
CU_08.a	Reconocer una colección de mínimos	5

CU_08.b	Reconocer una colección de máximos	5
CU_08.c	Reconocer el número de rebotes	8
CU_08.d	Definir y manejar varios marcos temporales	2
CU_08.e	Manejar franjas de precios	1
CU_08.f	Encontrar precios redondos	2
CU_08.g	Encontrar gaps o huecos en los precios	3
CU_08.h	Encontrar velas largas	5
CU_08.i	Calcular el retroceso de Fibonacci	8
CU_09	Comparar volúmenes de contratación	3
CU_10	Calcular distancia entre soportes y resistencias	1
CU_11	Abrir una operación de inversión	2
CU_11.a	Calcular un precio de entrada	1
CU_11.b	Calcular un precio de salida	1
CU_11.c	Calcular un stoploss	1
CU_12	Cerrar una operación de inversión	2
CU_12.a	Calcular la rentabilidad	1
CU_13	Informar de una oportunidad de inversión	2
CU_14	Almacenar una oportunidad de inversión	2
CU_15	Informar de una operación de inversión	2
CU_16	Almacenar una operación de inversión	2
CU_17	Solicitar una oportunidad de inversión	2
CU_18	Recuperar análisis realizados	2
CU_19	Recuperar operaciones abiertas	2
CU_20	Recuperar operaciones cerradas	2

El proceso de valoración de costes realizado, además de clasificar cada requisito según su coste en desarrollo, nos ha permitido volver a refinar la lista de requisitos de tal manera que:

- La granularidad de los requisitos CU_05, CU_07 y CU_08 no es adecuada por ser demasiada gruesa para poder definir la funcionalidad del sistema y por tanto, no pueden formar parte de la lista de requisitos definitiva. Así, serán sustituidos por los de mayor especificación o granularidad más fina obtenidos de su refinamiento.
- Por el contrario, la granularidad de los requisitos CU_11 y CU_12 es demasiada fina para definir el funcionamiento del sistema, así que se descarta el refinamiento hecho con anterioridad ya que su coste en desarrollo no es significativo.

2.2.2. Priorización de los requisitos

Una vez estimado el coste de los requisitos, los priorizaremos utilizando la técnica MOSCOW propuesta por el método DSDM (dynamic systems development method) en el que se recomienda limitar el número de prioridades a cuatro:

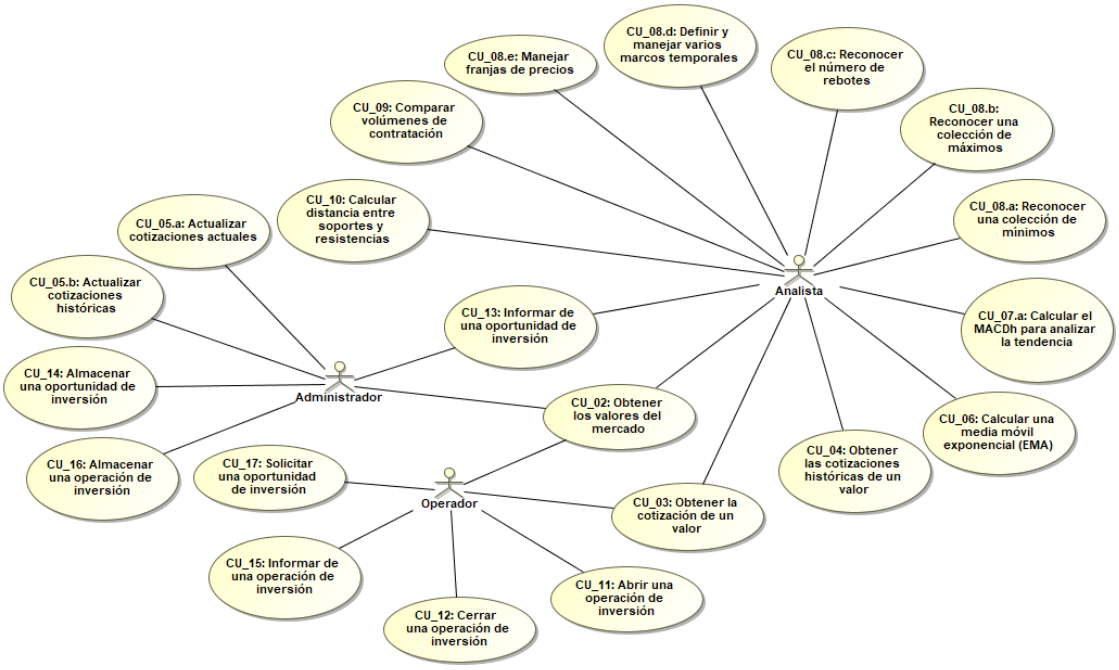
- Must have, sin estos requisitos el producto no se puede desarrollar
- Should have, son igual de importantes pero es posible prescindir de ellos temporalmente
- Coulhave, requisitos no prioritarios pero que mejoran el resultado final del producto
- Won't have, requisitos que se podrían no tener en cuenta

Identificador	Descripción	Estimación	Prioridad
CU_01	Seleccionar un mercado o índice	3	Could have
CU_02	Obtener los valores del mercado	3	Must have
CU_03	Obtener la cotización de un valor	3	Must have
CU_04	Obtener las cotizaciones históricas de un valor	3	Must have
CU_05.a	Actualizar cotizaciones actuales	3	Must have

CU_05.b	Actualizar cotizaciones históricas	3	Must have
CU_06	Calcular una media móvil exponencial (EMA)	2	Should have
CU_07.a	Calcular el MACDh para analizar la tendencia	5	Must have
CU_07.b	Calcular el MACDh (serie) como posibilidad de inversión	8	Could have
CU_07.c	Comparar el MACD y la Señal como señal de compra/venta	5	Could have
CU_07.d	Comparar cruce del MACD y la Señal como señal de compra/venta	8	Could have
CU_08.a	Reconocer una colección de mínimos	5	Must have
CU_08.b	Reconocer una colección de máximos	5	Must have
CU_08.c	Reconocer el número de rebotes	8	Must have
CU_08.d	Definir y manejar varios marcos temporales	2	Should have
CU_08.e	Manejar franjas de precios	1	Must have
CU_08.f	Encontrar precios redondos	2	Won't have
CU_08.g	Encontrar gaps o huecos en los precios	3	Could have
CU_08.h	Encontrar velas largas	5	Won't have
CU_08.i	Calcular el retroceso de Fibonacci	8	Won't have
CU_09	Comparar volúmenes de contratación	3	Should have
CU_10	Calcular distancia entre soportes y resistencias	1	Must have
CU_11	Abrir una operación de inversión	2	Must have
CU_12	Cerrar una operación de inversión	2	Must have
CU_13	Informar de una oportunidad de inversión	2	Must have
CU_14	Almacenar una oportunidad de inversión	2	Should have
CU_15	Informar de una operación de inversión	2	Should have
CU_16	Almacenar una oporación de inversión	2	Should have
CU_17	Solicitar una oportunidad de inversión	2	Must have
CU_18	Recuperar análisis realizados	2	Could have
CU_19	Recuperar operaciones abiertas	2	Could have
CU_20	Recuperar operaciones cerradas	2	Could have

2.2.3. Producto mínimo

Finalmente, se selecciona como producto mínimo toda aquella funcionalidad imprescindible, temporalmente o no, marcada con las prioridades *must have* o *should have*



Identificador	Descripción	Estimación	Prioridad	Agentes
CU_08.c	Reconocer el número de rebotes	8	Must have	An
CU_07.a	Calcular el MACDh para analizar la tendencia	5	Must have	An
CU_08.a	Reconocer una colección de mínimos	5	Must have	An
CU_08.b	Reconocer una colección de máximos	5	Must have	An
CU_02	Obtener los valores del mercado	3	Must have	Ad, An, Op
CU_03	Obtener la cotización de un valor	3	Must have	An, Op
CU_04	Obtener las cotizaciones históricas de un valor	3	Must have	An
CU_05.a	Actualizar cotizaciones actuales	3	Must have	Ad
CU_05.b	Actualizar cotizaciones históricas	3	Must have	Ad
CU_11	Abrir una operación de inversión	2	Must have	Op
CU_12	Cerrar una operación de inversión	2	Must have	Op
CU_13	Informar de una oportunidad de inversión	2	Must have	Ad, An
CU_17	Solicitar una oportunidad de inversión	2	Must have	Op
CU_08.e	Manejar franjas de precios	1	Must have	An
CU_10	Calcular distancia entre soportes y resistencias	1	Must have	An
CU_09	Comparar volúmenes de contratación	3	Should have	An
CU_06	Calcular una media móvil exponencial (EMA)	2	Should have	An
CU_08.d	Definir y manejar varios marcos temporales	2	Should have	An
CU_14	Almacenar una oportunidad de inversión	2	Should have	Ad
CU_15	Informar de una operación de inversión	2	Should have	Op
CU_16	Almacenar una oporación de inversión	2	Should have	Ad

Agentes -> Ad: Administrador, An: Analista, Op: Operador

2.3. Gestión de los requisitos no funcionales

El funcionamiento del sistema multiagente no depende solo de las capacidades de cada uno de los agentes, sino que también depende de aspectos propios del entorno en los que estos operan. Así, es necesario identificar el conjunto de requisitos no funcionales del producto para delimitar los aspectos funcionales de cada agente respecto a sus posibilidades operativas desde la perspectiva de la calidad que le ofrece su entorno de ejecución.

Así, el conjunto de cualidades esperadas del entorno como garantía de eficiencia, se desarrollarán principalmente, conforme a la aplicación de los distintos requisitos legales que hacen referencia al uso de estándares arquitectónicos y de diseño del software recogidos en los siguientes y anteriores apartados de este proyecto, y que facilitarán tanto la construcción del propio producto como su futuro mantenimiento y soporte.

2.3.1. Estándares de diseño

Según la norma IEEE 1471 la descripción arquitectónica de un sistema de software se organiza en vistas, que representan y describen el sistema desde una especificación (punto de vista) concreta. Es decir, la norma estructura el conjunto de modelos que documentan la arquitectura del sistema de software y los organiza en vistas de acuerdo a un propósito concreto con el fin de representar uno o más aspectos relevantes del sistema.

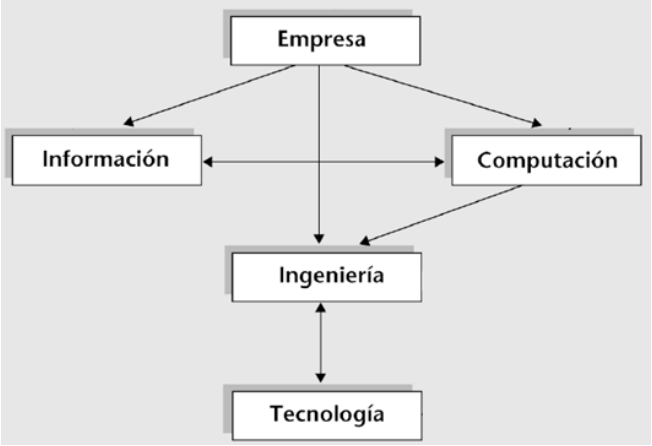
Así, cada vista se corresponde con un punto de vista, que es una forma de abstracción que permite concentrarse en algún aspecto concreto del sistema y que representa un aspecto relevante del mismo mediante el uso de una serie de conceptos, construcciones y reglas propias.

Más en concreto, la norma IEEE 1471 establece que la especificación de cualquier punto de vista ha de contener la siguiente información:

- El nombre del punto de vista
- Los stakeholders del sistema para los que el punto de vista es relevante
- Los aspectos del sistema tratados por el punto de vista
- Los lenguajes, notaciones, metodologías y técnicas de modelado a emplear

Reference Model–Open Distributed Processing

RM-ODP es un modelo de referencia para el procesamiento abierto y distribuido, definido por ISO e ITU-T que proporciona una arquitectura dirigida a la integración de aspectos como la distribución y la interoperabilidad de los objetos y componentes de un sistema. Así, el modelo establece un conjunto de normas que describen la arquitectura del software del sistema de acuerdo a cinco puntos de vista diferentes y mediante el uso de un modelo de objetos común. Estos cinco puntos de vista que define RM-ODP son: el punto de vista de la empresa, el de la información, el de la computación, el de la ingeniería y el de la tecnología.



• El punto de vista de la empresa describe los requisitos del sistema y de su entorno desde la perspectiva de la lógica de negocio: Se centra en los objetivos, el alcance, y el entorno y políticas que rigen las actividades del sistema dentro de la organización de la que forma parte.

• El punto de vista de la información describe la información que va a tratar el sistema, lo que incluye las estructuras de los datos, sus formatos y su diseño relacional: Se centra en los tipos de datos tratados por el sistema, su semántica y las restricciones impuestas sobre su uso.

• El punto de vista computacional describe la funcionalidad del sistema en términos de descomposición y organización del conjunto de objetos que lo componen. De tal forma, en este punto de vista, el sistema de software se ve como un conjunto de objetos computacionales que interactúan entre si a través de interfaces y cada interfaz especifica un conjunto de operaciones con la que el objeto interactúa con su entorno.

• El punto de vista de la ingeniería describe la infraestructura necesaria para soportar la distribución de la funcionalidad descrita en el punto de vista de la computación. De tal forma, en este punto de vista, el sistema de software se ve como un conjunto de objetos que se comunican entre si a través de canales.

• El punto de vista de la tecnología describe el conjunto de especificaciones tecnológicas que soportará el sistema con relación a la infraestructura del hardware, del software y de las comunicaciones que permitan implementar la funcionalidad del sistema, y procesar y distribuir el conjunto de datos según las especificaciones realizadas en el resto de los puntos de vista.

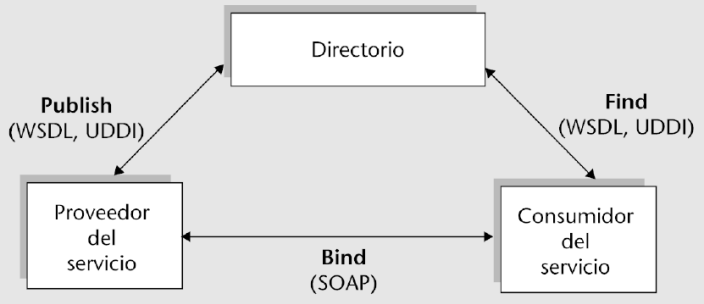
2.3.2. Arquitectura orientada a servicios

Por otra parte, en términos operacionales es necesario desarrollar con mayor detalle las garantías de eficiencia y robustez del entorno del sistema. Así, en el uso de la estrategia de análisis bottom-up empleada en el reconocimiento del conjunto de requisitos funcionales se identificó como mejor modelo para la obtención de los datos aquel que proporcionase un servicio de acceso a la información bursátil de manera independiente y específica, de tal forma que permitiese su uso de forma remota mediante HTTP, identificadores URI y XML

Modelo arquitectónico

La arquitectura SOA está caracterizada principalmente por constituir abstracciones de programas, bases de datos o procesos, y estar orientada, por un lado, al paso de mensajes que se intercambian entre el demandante y el propio servicio, y por otro lado, a la descripción de información mediante metadatos procesables reconocidos por el conjunto de consumidores (usuarios). Además, presentan la ventaja de estar orientados al uso en red (distribuidos) y ser independientes de la plataforma al usar estándares como los ya mencionados.

De tal forma, el principal desarrollo de esta arquitectura se da dentro del entrono web (Internet) mediante el desarrollo de los servicios web, que apoyados en los estándares HTTP, XML, SOAP, UDDI y WSDL identifican la prestación de un servicio determinado mediante URIs e interfaces públicas publicitadas por algún agente del propio servicio web. Así, lo más habitual es que estos sistemas se invoquen de forma remota mediante mensajes en formato XML y a través de protocolos estándares de Internet.



Así, el esquema de funcionamiento de los servicios web requiere tres elementos fundamentales:

- Un proveedor del servicio web, que es quien presta el servicio facilitando la invocación remota de un conjunto de operaciones (interface). Estas operaciones se publican permitiendo identificar tanto su descripción (el consumidor reconocen como invocar la operación) y su ubicación (localización física en un servidor).

- Un consumidor del servicio, que es quien accede y explota el servicio web. Es necesario que el consumidor conozca la ubicación del servicio para poder acceder a él, para ello, tiene que acceder primero al servicio de directorio (UDDI) donde se almacenan las listas de referencia a los servicios disponibles.

- Un agente de servicio, que sirve como enlace entre proveedor y consumidor para efectos de publicación, búsqueda y localización del servicio.

Estilos arquitectónicos

Entre los estilos arquitectónicos más importantes están los servicios web basados en SOAP y WSDL (WS-*) y los servicios web basados en recursos (RESTful web services).

Los servicios basados en SOAP (Simple object access protocol) permiten la interacción con servicios web basados en XML, y su especificación incluye la sintaxis para definir los mensajes, las reglas de codificación y el modelo de invocación remota. Además, utiliza WSDL (Web services description language) para describir la localización del servicio, el conjunto de operaciones proporcionadas, el formato de los mensajes y el modelo de invocación del servicio; y UDDI (universal description, discovery and integration) para proporcionar a los proveedores un mecanismo de publicación de servicios y que los consumidores puedan localizarlos e invocarlos.

En los servicios basados en REST (representational state transfer), los datos y la funcionalidad son considerados como recursos a los que se accede mediante la invocación remota de operaciones a partir de identificadores URI. El estilo arquitectónico REST se basa en una arquitectura cliente-servidor, y está diseñado para que clientes y servidores intercambien información mediante una interfaz estandarizada, mediante el uso de protocolos de comunicación sin estado (por lo general HTTP), y mediante el uso de tipos MIME como identificador del formato de los datos: XML, JSON, XHTML, HTML, PNG, etc…

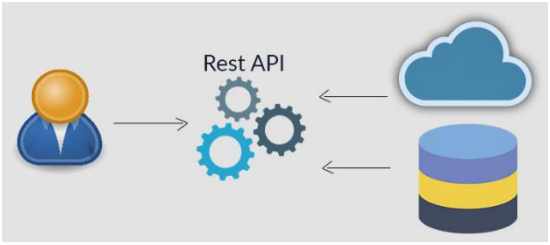
Por otra parte, los métodos más usados en REST para la serialización de la información (marshalling) son los métodos POX y JSON:

- Plain old XML (POX), conceptualmente sigue la misma filosofía que POJO (plain old java object) conforme a la relación entre los datos y la forma abstracta de organizarlos. En el caso de POJO, la información se organiza sobre una clase de Java y en el caso de POX, la información se estructura jerárquicamente utilizando un documento XML.

- JavaScript object notation (JSON), se utiliza cuando el tamaño del flujo de datos (stream) es de vital importancia. Su simplicidad ha dado lugar a la generalización de su uso, pero presenta el inconveniente de que penaliza la validación de la información recibida.

Diseño del modelo de acceso a los datos

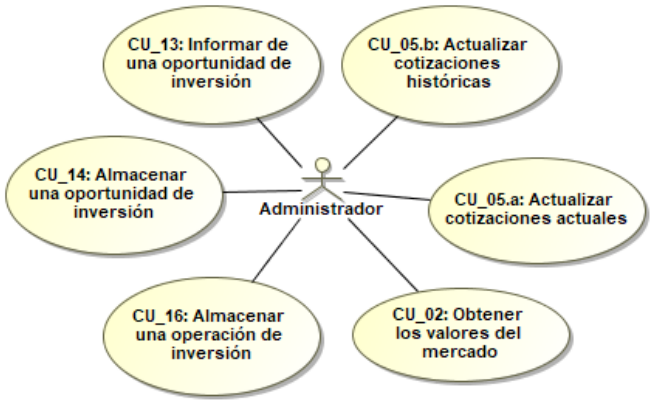
En consecuencia, en cumplimiento de las necesidades de calidad del entorno del sistema multiagente, se gestionará el modelo de acceso a los datos y su persistencia mediante la arquitectura REST y la implementación del método POX. Con ello se conseguirá, por una parte, construir un servicio eficiente e independiente de la operativa de los agentes para el acceso a la información externa desde Internet, dinámica por el uso de identificadores URI y HTTP, y de calidad por su estructuración en formato XML, y por otra parte, abstraer el recurso de la base de datos, configurando una única puerta de acceso para la lectura y el almacenamiento de los datos. Así, será el rol de agente administrador el que supervise la calidad de la información y mantenga la vigencia de los datos mediante la planificación de los accesos al servicio externo web.



2.4. Historias de usuario y casos de uso

Una vez definidos el producto mínimo y el modelo de acceso a los datos, documentaremos el conjunto de requisitos funcionales mediante el uso de una metodología ágil a partir de historias de usuario que por estar basadas en la comunicación verbal, facilitarán una mejor comprensión del funcionamiento del producto:

Agente administrador



Como administrador quiero poder obtener los valores de un mercado

• El agente administrador accederá al servicio web mediante una URI concreta solicitando así la lista de valores cotizados en el mercado continuo español

• El servicio web devolverá en formato XML la lista con los nombres de los valores cotizados

Como administrador quiero poder actualizar las cotizaciones de los valores

• El agente administrador accederá al servicio web mediante una URI concreta solicitando así que este se comunique con el servicio externo para obtener las últimas cotizaciones del mercado continuo

• El servicio web accederá a la web externa, obtendrá los datos y finalmente los grabará en la BBDD sustituyendo los datos existentes si los hubiera

Como administrador quiero poder actualizar las cotizaciones históricas de los valores

• El agente administrador accederá al servicio web mediante una URI concreta solicitando así que este se comunique con el servicio externo para obtener por cada valor sus cotizaciones de todo el año 2018

• El servicio web accederá a la web externa para obtener los datos y grabarlos en la BBDD

• El servicio web revisará si ha obtenido todo el histórico completo del año 2018 hasta la actualidad y si no lo ha hecho, repetirá las dos primeras operaciones tantas veces como haga falta

Como administrador quiero poder informar de una oportunidad de inversión

• El agente administrador recibirá un mensaje con la solicitud de envío de la mejor oportunidad de inversión que conozca

• El agente administrador buscará la oportunidad de inversión marcada como prioritaria y de mayor distancia entre el soporte más relevante y la resistencia más relevante

• El agente administrador informará de la oportunidad de inversión en el caso de que haya encontrado alguna

Como administrador quiero poder almacenar una oportunidad de inversión

• El agente administrador accederá al servicio web mediante una URI concreta solicitando así que este grabe un análisis en la base de datos

• El servicio web grabará en la base de datos el análisis

Como administrador quiero poder almacenar una operación de inversión

• El agente administrador accederá al servicio web mediante una URI concreta según el tipo de operación (apertura o cierre) solicitando así que este grase una operación en la base de datos

• El servicio web grabará en la base de datos la operación

Agente analista

Como analista quiero poder obtener los valores de un mercado

• Ver misma historia de usuario para agente administrador

Como analista quiero poder obtener la cotización de un valor

• El agente analista accederá al servicio web mediante una URI concreta solicitando así las cotizaciones de un valor del mercado continuo

• El servicio web devolverá en formato XML los datos con la cotización del valor

Como analista quiero poder obtener las cotizaciones históricas de un valor

• El agente analista accederá al servicio web mediante una URI concreta solicitando así las cotizaciones de los últimos nueve meses de un valor del mercado continuo

• El servicio web devolverá en formato XML los datos con la cotización histórica del valor

Como analista quiero poder calcular una media móvil exponencial (EMA)

• El agente analista seleccionará como muestra las últimas once cotizaciones históricas de un valor, de tal forma que en la undécima posición esté el último cierre conocido y en la décima posición el penúltimo

• El agente analista calculará la media móvil simple de las ocho primeras (más antiguas) cotizaciones como valor inicial para el cálculo de la media móvil exponencia: SMA_8

• El agente analista calculará la media móvil exponencial inicial (novena posición) sobre los valores SMA_8 y $\alpha=2/N+1$ con N=8, de tal forma que $EMA_{8\_ini} = \alpha*Precio_{cierre}+(1-\alpha)*SMA_8$

• El agente analista calculará la media móvil exponencial anterior (décima posición) con N=8 y $\alpha=2/N+1$, de tal forma que $EMA_{8\_ant} = \alpha*Precio_{cierre}+(1-\alpha)* EMA_{8\_ini}$

• El agente analista calculará la media móvil exponencial actual (undécima posición) con N=8 y $\alpha=2/N+1$, de tal forma que $EMA_{8\_actual} = \alpha*Precio_{cierre}+(1-\alpha)* EMA_{8\_ant}$

Como analista quiero poder calcular el histograma del MACD (MACDh)

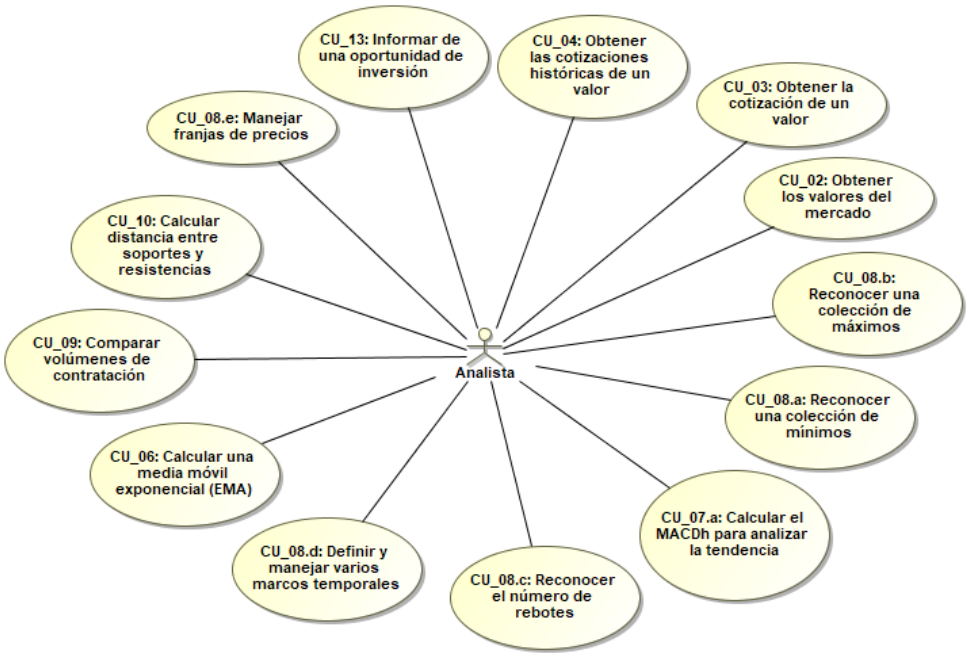
• El agente analista seleccionará las últimas treinta y ocho cotizaciones históricas de un valor

• El agente analista calculará el EMA_{12} y el EMA_{26}, tal y como se ha visto en el caso de uso CU_06

• El agente analista calculará el $MACD = EMA_{26} – EMA_{12}$

• El agente analista calculará la Señal como la EMA_9 de los nueve últimos valores del MACD teniendo en cuenta que el cálculo inicial se hará sobre la SMA_9 (décima posición empezando por el final) tal y como ocurre en el caso de uso CU_06

• El agente analista calculará el histograma MACDh = MACD - Señal



Como analista quiero poder obtener una colección de mínimos

• El agente analista seleccionará un marco temporal: Semanal o diario

• El agente analista seleccionará el tamaño de la muestra en función del marco temporal seleccionado: Si es semanal los últimos nueve meses de cotizaciones filtrando los precios de cierre semanal (última cotización de la semana que por lo regular coincide con el viernes) y si es diario, primeramente los últimos nueves meses, acto seguido los últimos seis meses y por último los últimos tres meses

• El agente analista revisará la muestra de datos y guardará cuatro colecciones de mínimos señalándolas por su marco temporal y tamaño, revisando los mínimos de la muestra de datos ordenados por fecha de tal forma que un mínimo guardado será aquel mínimo menor al último mínimo histórico conocido

Como analista quiero poder obtener una colección de máximos

• El agente analista seleccionará un marco temporal: Semanal o diario

• El agente analista seleccionará el tamaño de la muestra en función del marco temporal seleccionado: Si es semanal los últimos nueve meses de cotizaciones filtrando los precios de cierre semanal (última cotización de la semana que por lo regular coincide con el viernes) y si es diario, primeramente los últimos nueves meses, acto seguido los últimos seis meses y por último los últimos tres meses

• El agente analista revisará la muestra de datos y guardará cuatro colecciones de máximos señalándolas por su marco temporal y tamaño, revisando los máximos de la muestra de datos ordenados por fecha de tal forma que un máximo guardado será aquel máximo mayor al último máximo histórico conocido

Como analista quiero poder obtener una colección de rebotes en los máximos y mínimos

• El agente analista obtendrá una colección de máximos y mínimos tal y como se ha visto en los casos de uso CU_08.a y CU_08.b, y las unirá en una sola

• El agente analista calculará rangos de preciosa cada uno de los valores de la lista unificada, máximos y mínimos, según un factor de precisión de ±0,50%

• El agente analista contará cuantas veces se sitúa cada máximo o mínimo dentro del conjunto de rangos

• El agente analista seleccionará aquellos niveles de precios (máximos o mínimos) cuya cuenta sea mayor a la media más uno del resultado de cada uno de los contadores

Como analista quiero poder comparar volúmenes históricos de contratación

• El agente analista calculará el EMA_8 tal y como se ha visto en el caso de uso CU_06 sustituyendo el dato $Precio_{cierre}$ por Volumen

Como analista quiero poder calcular la distancia entre el soporte y la resistencia más relevantes

• El agente analista obtendrá una colección de niveles de precios a partir de sucesiones de máximos y mínimos tal y como se ha visto en el caso de uso CU_08.c

• El agente analista seleccionará de la colección de niveles obtenida, como resistencia más relevante el nivel más cercano al precio actual del valor y que además sea superior a este, y como soporte más relevante el nivel más cercano al precio actual del valor y que además sea inferior a este

• El agente analista calculará la distancia como la diferencia entre la resistencia y el soporte más relevantes

Como analista quiero poder informar de una oportunidad de inversión encontrada en el análisis

• El agente analista obtendrá la distancia entre soporte y la resistencia más relevantes (caso de uso CU_10) y preseleccionará el análisis cuando la resistencia sea al menos un 6% mayor que el soporte

• El agente analista obtendrá la media de móvil exponencial de precios (caso de uso CU_06) de un análisis preseleccionado y mantendrá la preselección siempre y cuando $EMA_{8\_Precio\ Actual} > EMA_{8\_Precio\ Anterior}$

• El agente analista obtendrá el histograma MACDh (caso de uso CU_07.a) de un análisis preseleccionado y lo seleccionará como oportunidad de negocio siempre y cuando $MACDh > 0$

• El agente analista obtendrá la media móvil exponencial de volúmenes (caso de uso CU_09) de un análisis seleccionado y lo marcará como prioritario cuando $EMA_{8\_Volumen\ Actual} > EMA_{8\_Volumen\ Anterior}$

• El agente analista enviará un mensaje informativo de un análisis seleccionado con el tipo 'Nuevo Análisis' al agente administrador según la especificación FIPA-ACL y como contenido, el propio análisis

• El agente administrador guardará los datos del análisis en una lista de análisis disponibles y llamará al servicio web según el caso de uso CU_14

Agente operador

Como operador quiero poder obtener los valores de un mercado

• Ver misma historia de usuario para agente administrador

Como operador quiero poder obtener la cotización de un valor

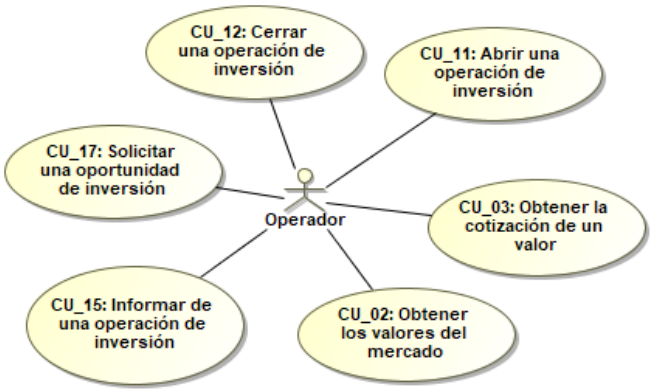
• Ver misma historia de usuario para agente analista

Como operador quiero poder solicitar una oportunidad de inversión

• El agente operador envía un mensaje al agente administrador solicitando una oportunidad de inversión según la especificación FIPA-ACL

• El agente administrador buscará en la lista de análisis disponibles un análisis de prioridad alta y de entre ellos el de mayor distancia entre el soporte y la resistencia más relevantes, y lo retornará al operador

• El agente administrador quitará el análisis de la lista de análisis disponibles y lo guardará en la lista de análisis entregados



Como operador quiero poder abrir una operación de inversión

• El agente operador actualizará la cotización del valor analizado (caso de uso CU_03)

• El agente operador calculará el precio de salida como el precio actual más el 3% y si el resultado es menor que la resistencia más relevante lo preseleccionará como operación de inversión

• El agente operador calculará el stoploss de una operación preseleccionada como el soporte más relevante actual más el 1% y el precio de entrada como el precio actual del valor, y dará la operación por abierta

• El agente operador informará de la operación abierta al agente administrador (caso de uso CU_15)

Como operador quiero poder cerrar una operación de inversión

• El agente operador vigila la operación abierta solicitando cada cinco minutos la cotización del valor (caso de uso CU_03)

• El agente operador comparará el precio del valor con el precio de salida y si es mayor cerrará la operación, y también comparará el precio con el stoploss y si es menor, cerrará la operación

• El agente operador guardará el precio de cierre de una operación cerrada y calculará su rentabilidad

Como operador quiero poder informar de una operación de inversión

• El agente operador enviará un mensaje informativo de una operación abierta con el tipo 'Nueva Operación' al agente administrador según la especificación FIPA-ACL y como contenido, la operación

• El agente administrador guardará los datos de la operación en una lista de operaciones abiertas y llamará al servicio web según el caso de uso CU_16

Por último, con el fin de clarificar y detallar el comportamiento de los agentes, se adjunta como anexo a esta memoria un modelo de documentación de requisitos más formal y mediante casos de uso, se detalla el ciclo de vida de una operación de inversión:

- Como analista quiero poder informar de una oportunidad de inversión (análisis)
- Como administrador quiero poder almacenar una oportunidad de inversión
- Como operador quiero poder solicitar una oportunidad de inversión
- Como administrador quiero poder informar de una oportunidad de inversión
- Como operador quiero poder abrir una operación de inversión
- Como operador quiero poder cerrar una operación de inversión

2.5. Síntesis de la funcionalidad de los agentes

Como se ha visto, el sistema multiagente está compuesto por tres agentes que interaccionan entre sí para intentar llevar a cabo una inversión financiera. Para ello, cada uno de los agentes tiene un rol y una misión específica dentro del sistema: El rol de agente analista plantea, a partir de la información recogida de los mercados financieros, un conjunto de posibilidades de inversión que el rol de agente operador materializa con el fin de ir poco a poco revisando los resultados obtenidos, para una vez ganada suficiente experiencia, poder llegar a enjuiciar los planteamientos y la parametrización inicial propuesta.

Además, el sistema necesita para mantener el nivel de especialización implícito de cada rol, un administrador del sistema para canalizar y dirigir el flujo de información intercambiada, su sincronización y su persistencia para su futuro análisis, algo que, en primera instancia, el alcance del proyecto contempla tan solo de forma teórica y no práctica.

Así, el diseño del producto se realizará sobre el conjunto de requisitos que conforman el producto mínimo y que en futuros apartados serán traducidos en las operaciones que darán al sistema las cualidades funcionales necesarias para cumplir con sus especificaciones. Además, el sistema está configurado para operar sobre un conjunto estático de indicadores técnicos y con una parametrización inicial que formarán la base técnica del sistema multiagente. De tal forma, es posible resumir desde varias perspectivas los valores por defecto del sistema:

• Según el modelo de análisis, los indicadores calculados son: la media móvil exponencial (EMA) tanto del precio de cotización como del volumen: la media móvil de convergencia/divergencia (MACD); el análisis de máximos, mínimos y niveles de precios (soportes y resistencias), tanto semanales como diarios, de periodos de nueve, seis y tres meses.

Indicador	Parámetro	Valor
EMA	Periodo	8 sesiones
MACD	Periodo EMA rápida	12 sesiones
MACD	Periodo EMA lenta	26 sesiones
MACD	Periodo Señal	9 sesiones
Máximos y mínimos	Cotización/Tamaño muestra	Semanal/9 meses
Máximos y mínimos	Cotización/Tamaño muestra	Diario/9 meses
Máximos y mínimos	Cotización/Tamaño muestra	Diario/6 meses
Máximos y mínimos	Cotización/Tamaño muestra	Diario/3 meses
Nivel de soporte o resistencia	Precisión del rango	1% (±0,5%)
Nivel de soporte o resistencia	Modelo de selección	> Media(contador)+1

• Según el modelo de identificación de posibilidades de inversión, los indicadores revisados son la distancia o diferencia entre la resistencia y el soporte más relevantes encontrados, la media móvil exponencial del precio, y el histograma del MACD. Además, se priorizan los análisis seleccionados a partir de la media móvil exponencial del volumen.

Indicador	Parámetro	Valor
EMA (Precio)	Diferencia(Actual, Anterior)	> 0
EMA (Volumen)	Diferencia(Actual, Anterior)	> 0
MACD	MACDh	> 0
Nivel de soporte o resistencia	Diferencia(Resistencia, Soporte)	≥ 5%

• Según la operativa de inversión, los indicadores revisados son el último precio de cotización, la resistencia más relevante y el soporte más relevante.

Indicador	Parámetro	Valor
Cotización	Precio salida	Precio + 2,5%
Cotización	Apertura operación	Precio salida < Resistencia
Cotización	Stoploss	Soporte - 1%

• Según la administración de la información y su planificación, la frecuencia con la que se actualice la información bursátil será, en horario de apertura, cada cinco minutos, y en horario de cierre, una vez al día, de lunes a viernes.

Indicador	Parámetro	Valor
Planificación	Información actual	Cada 5 minutos
Planificación	Información histórica	Cada día, de lunes a viernes

Por último, comentar algunas posibles futuras ampliaciones de la funcionalidad del modelo para dinamizar los procesos de análisis y así, la capacidad de ganar experiencia de los agentes:

- Añadir nuevos indicadores técnico o ampliar el uso de los indicadores actuales
- Modificar, ampliando o reduciendo la parametrización por defecto del modelo. Por ejemplo, se podría valorar el comportamiento del modelo sobre el cálculo semanal de máximos y mínimos, sin tener en cuenta los diarios (o viceversa)
- Avanzar sobre el modelo de prioridades de análisis técnicos y analizando los datos obtenidos, comprobar si hay configuraciones mejores y peores que otras
- Dinamizar la operativa de inversión, sustituyendo la inversión en curso por una nueva que haya sido marcada como más prioritaria
- Añadir referencias globales al modelo, como la tendencia general del mercado

3. Base de conocimiento de los agentes

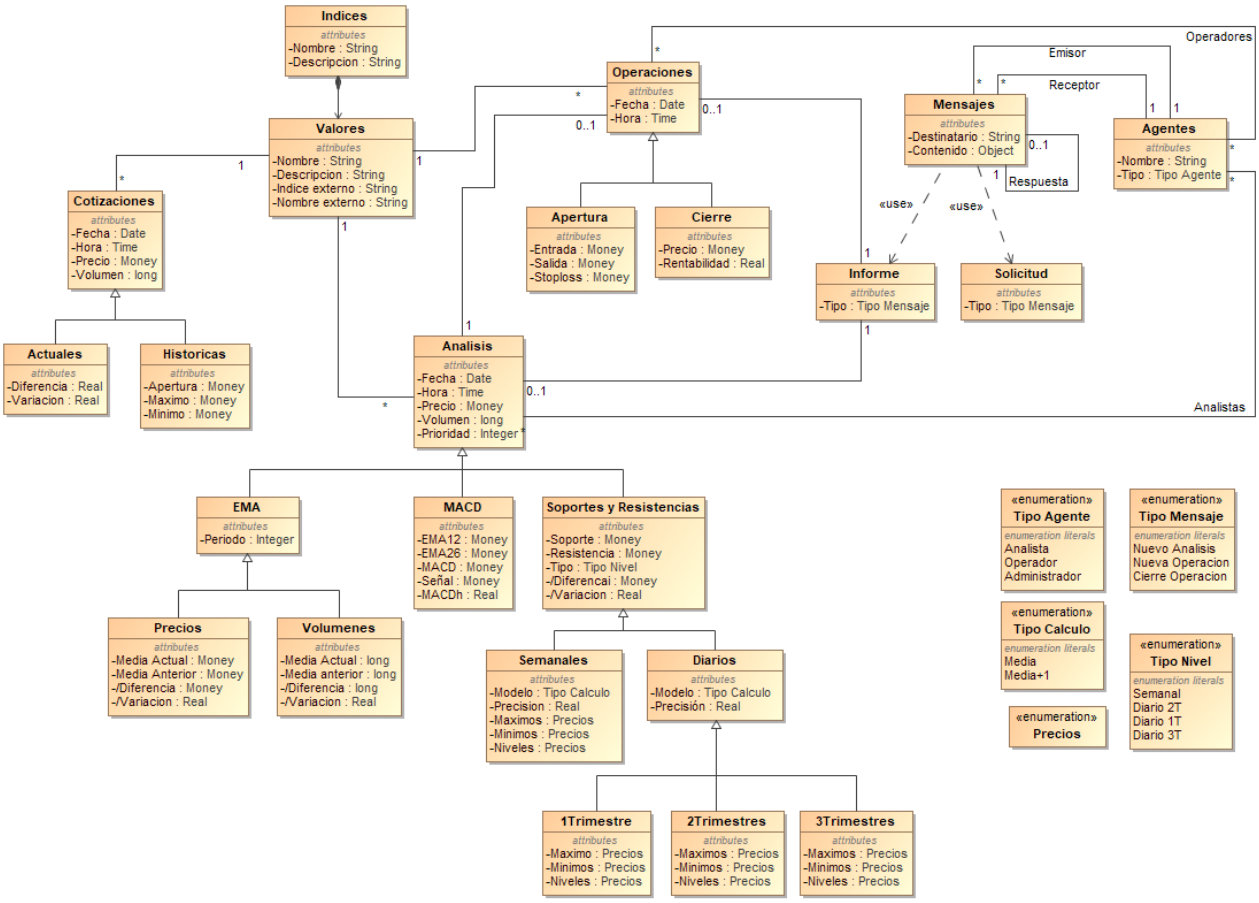
Siguiendo el diseño de puntos de vista que propone RM-OPD, revisaremos el diseño del producto desde el punto de vista de la información: Hasta ahora tan solo se ha visto el conjunto de especificaciones del producto desde un punto de vista de empresa, es decir, desde un punto de vista centrado únicamente en la descripción de los objetivos del producto, de sus requisitos, y de las políticas de diseño y estándares que se aplicarán a lo largo de su fase de diseño, sin embargo, en este apartado nos centraremos en una visión del producto desde los datos y sus estructuras.

Además, en el proceso de identificación y gestión del conjunto de requisitos funcionales, también se hizo una pequeña aproximación a aquellos que afectan únicamente a los datos y que serán en los que nos centremos para reconocer la base del conocimiento del sistema multiagente:

- *El sistema tiene que poder identificar un índice, conocer su descripción e identificador externo*
- *El sistema tiene que poder identificar un valor dentro de un índice, conocer su nombre y su identificador externo para poder extraer la información de Internet*
- *El sistema tiene que poder conocer el precio de mercado actual de un valor identificando la fecha y la hora de la cotización, la variación del precio respecto al día anterior y el volumen contratado*
- *El sistema tiene que poder conocer por cada día de cotización el precio de apertura del valor, el precio de cierre, el máximo alcanzado, el mínimo y el volumen cotizado*
- *El sistema tiene que poder identificar por cada valor su cotización, sus soportes y resistencias más relevantes, las medias móviles exponenciales de los precios y volúmenes, y el MACD*
- *El sistema tiene que poder reconocer el tipo de operación efectuada, el precio de entrada, el precio de salida, el precio stoploss, el precio de cierre definitivo y la rentabilidad obtenida*
- *El sistema tiene que poder clasificar los mensajes por tipo y poder encapsular la información analizada y la información de la operativa bursátil llevada a cabo*

3.1. Modelo conceptual de clases

La información que va a tratar el sistema la modelaremos primeramente mediante un diagrama de clases para posteriormente avanzar con la definición de la estructura de los datos, sus formatos y su diseño relacional mediante OCL (Object Constraint Language). Por último, diseñaremos el modelo de persistencia que dará soporte a la base de conocimientos de los agentes mediante el diseño de una base de datos y la construcción de un servicio web que nos permita acceder a ella.



Principios y patrones de diseño

En el diseño de las clases se han aplicado un conjunto de principios y patrones ya contrastados que van a permitir construir el sistema multiagentes con las mismas buenas cualidades que otros sistemas en donde ya han sido probados de forma satisfactoria:

Principio de bajo acoplamiento

El principio de bajo acoplamiento (low coupling) nos permitirá diseñar clases con menor grado de dependencia entre ellas, de tal forma que sean fáciles de entender de forma aislada, mejorando tanto su reutilización como sus futuras modificaciones, ya que se conseguirá que tengan la menor incidencia posible en el resto de clases.

Principio de abierto-cerrado (OCP)

El principio abierto-cerrado (open-closed principle) se fundamenta en la idea de que un clase tendría que estar abierta a la extensión pero cerrada a la modificación, es decir, que el diseño del sistema tendrá que tener en cuenta que para añadir nueva funcionalidad en el futuro es mejor hacerlo incorporando clases que modificando las que ya había, algo que nos permitirá hacer el uso de abstracciones e interfaces.

Principio de no repetición (DRY)

El principio de no-repetición (don't repeat yourself) nos indica que cada unidad de conocimiento debe tener una única representación en el sistema, evitando así información duplicada y ambigüedad en las responsabilidades de cada clase.

Además de los principios generales anteriores, aplicaremos los siguientes patrones de análisis y diseño que tienen un carácter más específico:

Patrón asociación histórica

El patrón de asociación histórica permitirá recuperar valores que una asociación entre clases ha ido tomando a lo largo del tiempo. Esto se consigue, añadiendo una nueva dimensión (de n-aria a n+1-aria) a la asociación que represente al tiempo.

Patrón estado

El patrón estado (state) permitirá instrumentalizar el principio de bajo acoplamiento en clases en las que su comportamiento varía en función de su estado, de tal forma que, cada comportamiento diferenciado formará una nueva clase.

Patrón estrategia

El patrón estrategia (strategy) permitirá instrumentalizar el principio de no repetición, al definir una familia de algoritmos y hacerlos intercambiables los unos con los otros. En la práctica, se puede llegar a sustituir por el patrón método plantilla (template method) ya que tienen la misma finalidad, el patrón estrategia mediante delegación y el patrón método plantilla mediante herencia.

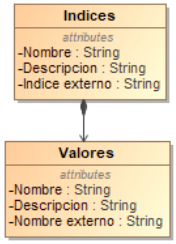
Estrategia de diseño

Al igual que hicimos con los requisitos de funcionamiento del sistema, utilizaremos una estrategia bottom-up para agrupar el conjunto de requisitos de datos por temáticas e ir aplicando los patrones de diseño enunciados al conjunto de clases inicial.

Grupo 1 – Valores del mercado

En este primer grupo irán aquellos requisitos de datos que identifican al conjunto de empresas o valores que cotizan en un mercado financiero, de tal forma, que representaremos la relación entre las clases *Indices* y *Valores* mediante una composición:

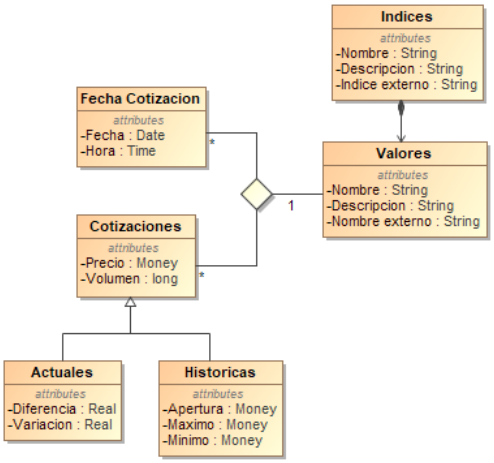
- *El sistema tiene que poder identificar un índice, conocer su descripción e identificador externo*
- *El sistema tiene que poder identificar un valor dentro de un índice, conocer su nombre y su identificador externo para poder extraer la información de Internet*



Grupo 2 – Cotizaciones de los valores

En este segundo grupo irán aquellos requisitos de datos que identifican las cotizaciones, actuales e históricas, de los valores bursátiles y estará compuesta principalmente por la clase *Cotizaciones* a la que aplicaremos el patrón asociación histórica, para tener una perspectiva temporal de las cotizaciones de los valores, y el patrón estado, para abstraer e independizar los dos tipos de cotizaciones que hay en dos subclases de comportamiento diferenciado:

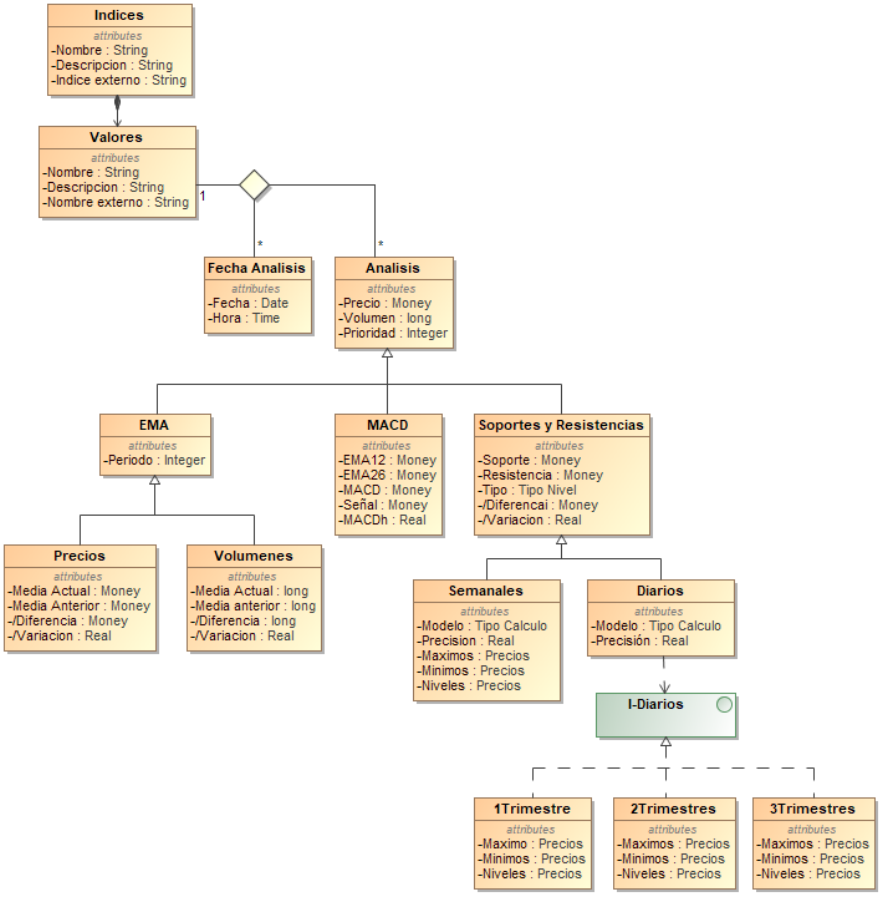
- *El sistema tiene que poder conocer el precio de mercado actual de un valor identificando la fecha y la hora de la cotización, la variación del precio respecto al día anterior y el volumen contratado*
- *El sistema tiene que poder conocer por cada día de cotización el precio de apertura del valor, el precio de cierre, el máximo alcanzado, el mínimo y el volumen cotizado*



Grupo 3 – Análisis de valores

En esta tercera agrupación irán aquellos requisitos de datos que identifican las oportunidades de inversión (análisis seleccionados para operar con ellos) y estará compuesta por la superclase *Analisis*, a la que se le aplicará el patrón asociación histórica, y por una colección de subclases correspondientes a los distintos métodos técnicos de análisis que serán diseñadas de forma independiente (comportamiento propio) mediante la aplicación del patrón estado. También, se aplica el patrón estrategia a la clase *Diarios* para evitar repetir la definición de los métodos en todas sus subclases y construirlos de forma común e intercambiable para cada una de ellas:

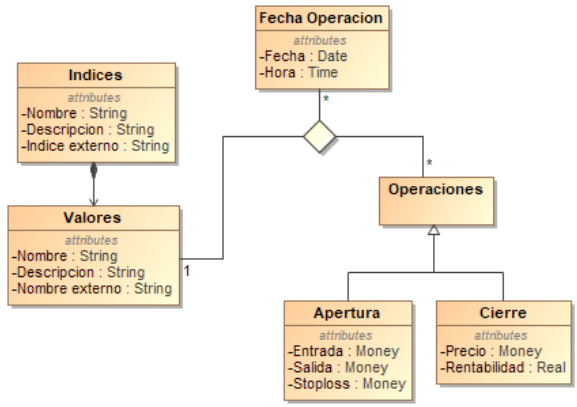
- *El sistema tiene que poder identificar por cada valor su cotización, sus soportes y resistencias más relevantes, las medias móviles exponenciales de los precios y volúmenes, y el MACD*



Grupo 4 – Operaciones con valores

En este cuarto grupo irán aquellos requisitos de datos que identifican las operaciones de inversión realizadas y estará compuesta por la clase *Operaciones* a la que se le aplicará el patrón asociación histórica y el patrón estado, para conseguir que sus subclases *Apertura* y *Cierre* se correspondan con los distintos tipos de operación existentes en el modelo:

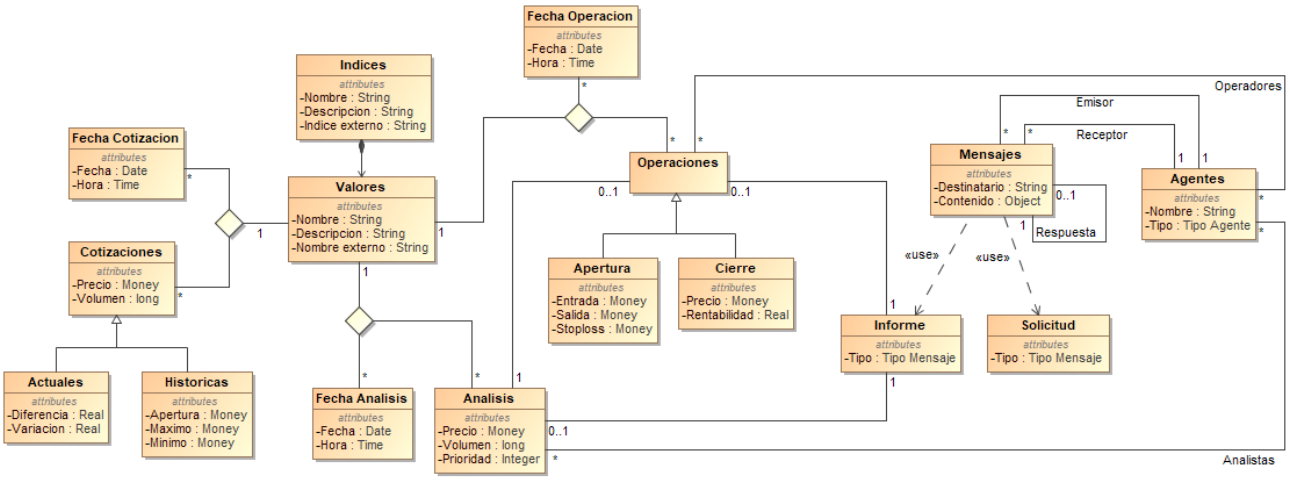
- *El sistema tiene que poder reconocer el tipo de operación efectuada, el precio de entrada, el precio de salida, el precio stoploss, el precio de cierre definitivo y la rentabilidad obtenida*



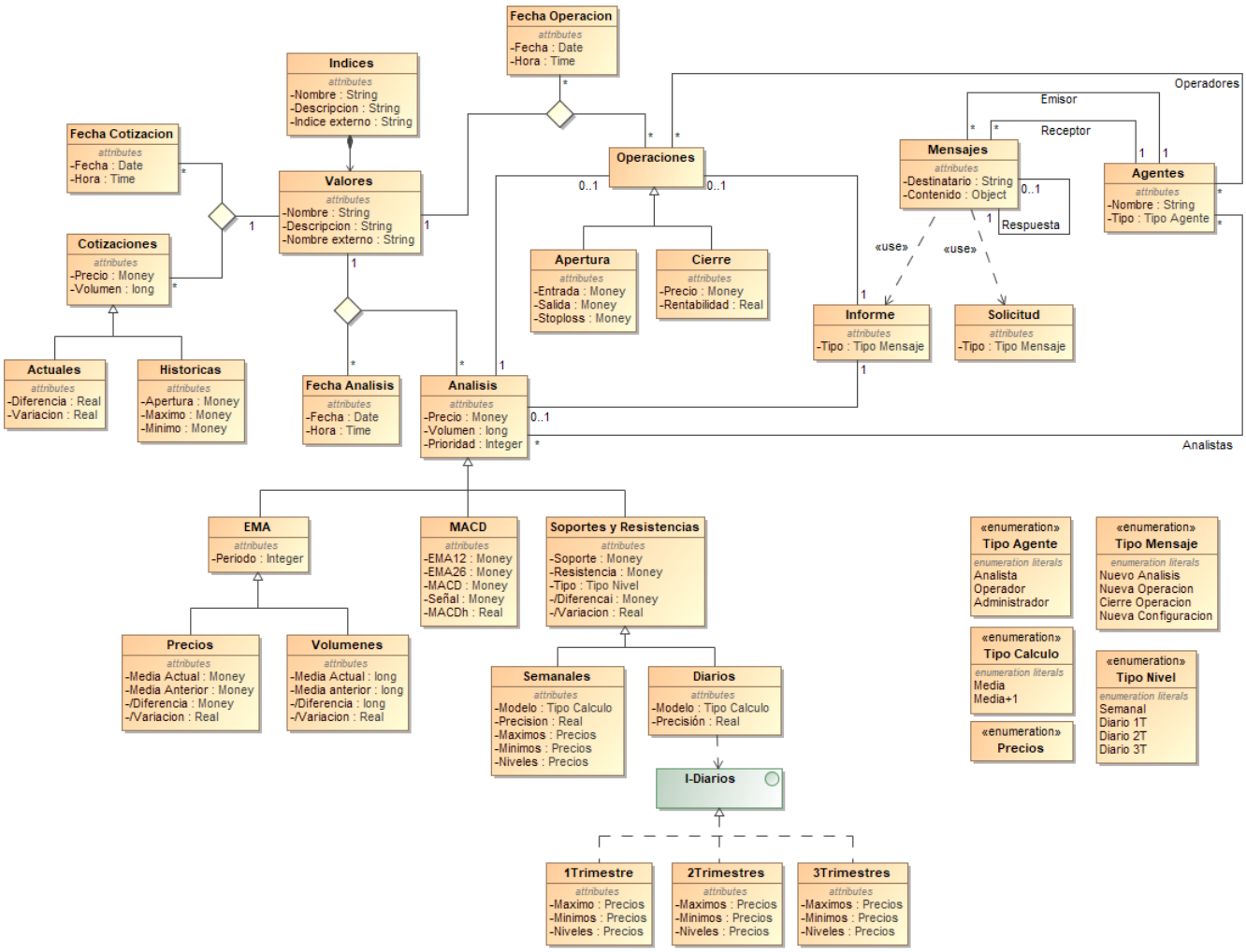
Grupo 5 – Comunicación entre agentes

En este último grupo irán aquellos requisitos de datos que identifican y dan soporte al modelo de comunicaciones entre los agentes que forman el sistema. A tal respecto, tal y como quedó establecido en apartados anteriores, hay que notificar que el modelo de comunicaciones entre agentes será el que proporciona el framework JADE en aplicación del estándar FIPA. Así, el modelo de clases propuesto tan solo dará soporte a la tipología de objetos que serán enviados mediante el uso de mensajes FIPA-ACL, es decir, el diagrama de clases tiene un fin meramente informativo para reconocer el futuro comportamiento y contenido de los mensajes intercambiados en el sistema, pero en ningún caso se llevará a la capa de persistencia el contenido de los mismos.

- *El sistema tiene que poder clasificar los mensajes por tipo y poder encapsular la información analizada y la información de la operativa bursátil llevada a cabo*



Finalmente, el diagrama de clases queda establecido de la siguiente manera:



3.2. Modelo relacional y lógico

El lenguaje UML (Unified Modeling Language), utilizado en el diseño del modelo conceptual y su diagrama de clases, no permite expresar con todo detalle las restricciones de integridad o las reglas de derivación de los de los atributos que componen cada una de las clases del modelo.

Así, para solucionar estas carencias y complementar el lenguaje UML, utilizaremos el lenguaje textual OCL (Object Constraint Language) para poder usar un conjunto de enunciados o expresiones que permitan condicionar la integridad del modelo relacional de clases mediante invariantes y aplicar modelos de cálculo a los atributos de las clases (reglas de derivación).

Para ello, es necesario hacer un diseño del modelo relacional a partir del modelo conceptual decidiendo que información será la que se almacene finalmente en la base de datos y como se tiene que estructurar para conseguir un modelo lógico robusto, rápido y fiable. De tal forma, identificaremos el conjunto de tablas con sus claves que formarán la base de datos y documentaremos sus relaciones mediante el lenguaje OCL.

Modelo lógico de la base de datos

El análisis del diagrama de clases recomienda que no todas las clases tengan que ser persistentes y que algunas de ellas compartan una estructura lógica común que va a permitir acceder a la información de forma más eficiente:

- La composición de las clase *Indices* y *Valores* formará dos tablas: *indices*, con clave el atributo *Nombre* y *valores*, con una clave doble fruto de la composición: el *Nombre* del índice y el *Nombre* del valor. Además, para facilitar la interpretación del modelo, al primer atributo le llamaremos *indice* y al segundo *valor*
- La superclase *Cotizaciones* y sus subclases formarán otras dos tablas. La tabla *cotizaciones*, que por el hecho de manejar un único registro tendrá la misma clave que la tabla valores: *indice* y *valor*; y la tabla *historico*, que añadirá a la clave el atributo *fecha*
- La clase *Agentes* formará una nueva tabla: *agentes* que tendrá como clave el atributo *Nombre* (al que llamaremos *agente* para una mejor interpretación del dato)
- La superclase *Analisis* y todas sus subclases de primer nivel formarán una única tabla: *analisis* que tendrá como clave un identificador numérico: *analisisid* y tendrá como claves foráneas, las claves de las tablas *valores* y *agentes*
- Las subclases de la clase *EMA* formarán otra tabla: *ema* que tendrá como clave un identificador numérico: *emaid* y un atributo *tipo* que permitirá reconocer los datos de cada una de las subclases de forma inequívoca. Además, tendrá como clave foránea la clave de la tabla *analisis*
- Las subclases de primer nivel de la clase *Soportes y Resistencias* formarán otra tabla: *soportesresistencias* que tendrá como clave un identificador numérico: *sopresid* y como clave foránea la de la tabla *analisis*
- Las subclases *1Trimestre, 2Trimestres, 3Trimestres y parte de los atributos de la subclase Semanales* formarán tres nuevas tablas. La tabla *maximos* que tendrá como clave un identificador numérico: *maximoid*, la tabla *minimos* cuya clave numérica será: *minimoid* y la tabla *niveles* con clave numérica: *nivelid*. Además, cada una de las tablas tendrá como clave foránea la clave de la tabla *soportesresistencias*
- La clase *Operaciones* formará una nueva tabla que tendrá como clave un identificador numérico: *operacionid* y que tendrá como claves foráneas las claves de las tablas *valores*, *analisis* y *agente*
- Las subclases de la clase *Operaciones* formarán dos tablas: compras, con clave numérica *compraid*; y ventas, con clave numérica *ventaid*. Además, cada una de las tablas tendrá como clave foránea la clave de la tabla *operaciones*

Invariantes y reglas de derivación

Además del modelo lógico, también es necesario detallar el modelo entidad-relación de la base de datos. Así, se adjunta como anexo a esta memoria los distintos enunciados en lenguaje OCL que forman el conjunto de restricciones de integridad y las reglas de derivación aplicadas al modelo conceptual de clases. A tal respecto, notificar que es posible expresar de más de una manera los enunciados recogidos en esta memoria.

3.3. Diseño de la base de datos

Sobre el diagrama de clases anterior, definiremos el conjunto de tablas que formarán parte de la base de datos del sistema y en la que almacenaremos los datos de forma permanente.

Nombre de la clase: *Indices*

Nombre de la tabla: *indices*

Descripción de la tabla: Permite almacenar los mercados financieros o índices

Clave de la tabla: La clave de la tabla es *indice*

Nombre del Campo	Tipo	Tipo BBDD
indice	String	text
descripcion	String	text
referencia	String	text

Nombre de la clase: *Valores*

Nombre de la tabla: *valores*

Descripción de la tabla: Permite almacenar el conjunto de valores cotizados en un mercado

Clave de la tabla: La clave de la tabla es la combinación de la clave foránea de la tabla *indices* y de la propia tabla: *indice+valor*

Notas/Comentarios:

Nombre del Campo	Tipo	Tipo BBDD
indice	String	text
valor	String	text
descripcion	String	text
referencia	String	text

Nombre de la clase: *Actuales*

Nombre de la tabla: *cotizaciones*

Descripción de la tabla: Permite almacenar la última cotización de un valor de un mercado financiero. Como el interés en el dato radica en su vigencia y actualidad, el almacenamiento de datos consiste en el borrado de los datos existentes y en la grabación de los nuevos.

Clave de la tabla: Como los accesos de lectura a la tabla se van a realizar principalmente para conocer la cotización de un valor concreto y se da la particularidad de que la tabla va a tener un único registro, se define la clave de la tabla de igual forma que en la tabla *valores*: *indice+ valor*

Notas/Comentarios: La clase *Actuales* se junta con su superclase *Cotizaciones* para evitar accesos redundantes al disco duro y la pérdida de eficiencia en el almacenamiento de los datos

Nombre del Campo	Tipo	Tipo BBDD
indice	String	text
valor	String	text
precio	Money	real
diferencia	Money	real
variacion	Real	numeric(5,2)
volumen	Long	long
fecha	Date	date
hora	Time	time

Nombre de la clase: *Historicas*

Nombre de la tabla: *historico*

Descripción de la tabla: Permite almacenar las cotizaciones históricas de un valor

Clave de la tabla: Como los accesos de lectura a la tabla se van a realizar principalmente para conocer la cotización histórica de un valor concreto, con el fin de mejorar los accesos a la BBDD se define la clave respetando la clave foránea de la tabla *valores* junto a la fecha de cotización (dato de periodicidad diaria): *indice+valor+fecha*

Notas/Comentarios: La clase *Historicas* se junta con su superclase *Cotizaciones* para evitar accesos redundantes al disco duro y la pérdida de eficiencia

Nombre del Campo	Tipo	Tipo BBDD
indice	String	text

valor	String	text
fecha	Date	date
ultimo	Money	real
apertura	Money	real
maximo	Money	real
minimo	Money	real
volumen	Long	long

Nombre de la clase: *Agentes*

Nombre de la tabla: *agentes*

Descripción de la tabla: Permite almacenar los datos de los agentes

Clave de la tabla: La clave de la tabla es *agente*

Notas/Comentarios: La enumeración Tipo Agente tomará los siguientes valores en la BBDD: Administrador = 0; Analista = 1; Operador = 2

Nombre del Campo	Tipo	Tipo BBDD
agente	String	text
tipo	Tipo Agente	char

Nombre de la clase: *Analisis*

Nombre de la tabla: *analisis*

Descripción de la tabla: Permite almacenar análisis identificados como posibilidad de inversión

Clave de la tabla: La clave de la tabla es *analisisid* que se definirá autoincremental enlazándola con el secuenciador *analisisid_seq*

Notas/Comentarios: Para evitar accesos redundantes al disco duro y la pérdida de eficiencia, la tabla recoge también los datos de todas sus subclases principales: *EMA*, *MACD* y *Soportes y Resistencias*. Además, el valor recogido en el atributo *ema* es el periodo de cálculo de las medias; los valores recogidos en los atributos *soporte*, *resistencia* y *nivel*, son los valores del soporte y la resistencia más relevantes junto al tipo de nivel de precios con los que han sido calculados; los valores de los atributos *índice* y *valor* se corresponden con la clave de la tabla *valores*; y el atributo *agente* representa el identificador del agente que ha realizado el análisis y se corresponde con la clave de la tabla *agentes*

La enumeración Tipo Nivel tomará los siguientes valores en la BBDD: Semanal = 0; Diario y estudio de un trimestre: 1; Diario y dos trimestres: 2; Diario y tres trimestres: 3

Nombre del Campo	Tipo	Tipo BBDD
analisisid	Integer	bigint
precio	Money	real
volumen	Long	long
ema	Integer	bigint
macd	Money	real
señal	Money	real
macdh	Real	real
soporte	Money	real
resistencia	Money	real
nivel	Tipo Nivel	char
prioridad	Integer	bigint
indice	String	text

valor	String	text
agente	String	text
fecha	Date	date
hora	Time	time

Nombre de las clases: *Precios, Volumenes*

Nombre de la tabla: ema

Descripción de la tabla: Permite almacenar los resultados del cálculo de medias

Clave de la tabla: La clave de la tabla es *emaid* que se definirá autoincremental enlazándola con el secuenciador *emaid_seq*

Notas/Comentarios: Para evitar accesos redundantes al disco duro y la pérdida de eficiencia, las subclases *Precios* y *Volumenes* se tratarán en una misma tabla, y su superclase *EMA* se tratará en la tabla *analisis*

El atributo *tipo* tomará los siguientes valores en la BBDD: Media de precios = 0; Media de volúmenes: 1

Nombre del Campo	Tipo	Tipo BBDD
emaid	Integer	bigint
tipo		char
media	Money/Long	real
analisisid	Integer	bigint

Nombre de las clases: *Semanales, Diarios*

Nombre de la tabla: *soportesresistencias*

Descripción de la tabla: Permite almacenar los resultados de los cálculos de los soportes y resistencias de precios

Clave de la tabla: La clave de la tabla es *sopresid* que se definirá autoincremental enlazándola con el secuenciador *sopresid_seq*

Notas/Comentarios: Para evitar accesos redundantes al disco duro y la pérdida de eficiencia, las subclases *Semanales* y Diarios se tratarán en una misma tabla, y su superclase *Soportes y Resistencias* se tratará en la tabla *análisis*, relacionandose mediante el atributo *analisisid*

La enumeración *Tipo Calculo* indica el tipo de modelo de selección de los niveles y tomará los siguientes valores en la BBDD: Media = 0; Media+1: 1; el atributo *precision* indica la amplitud porcentual del rango de precios para el cálculo de niveles; la enumeración *Tipo Nivel* tomará los siguientes valores en la BBDD: Semanal = 0; Diario y estudio de un trimestre: 1; Diario y dos trimestres: 2; Diario y tres trimestres: 3

Nombre del Campo	Tipo	Tipo BBDD
sopresid	Integer	bigint
modelo	Tipo Calculo	char
precision	Real	numeric(5,2)
tipo	Tipo Nivel	char
analisisid	Integer	bigint

Nombre de las clases: *Semanales, 1Trimestre, 2Trimestres, 3Trimestres*

Nombre de la tabla: *máximos*

Descripción de la tabla: Permite almacenar los precios máximos obtenidos en el análisis

Clave de la tabla: La clave de la tabla es *maximoid* que se definirá autoincremental enlazándola con el secuenciador *maximoid_seq*

Notas/Comentarios: La tabla recoge la clave foránea de *soportesresistencias*: *sopresid*

Nombre del Campo	Tipo	Tipo BBDD
maximoid	Integer	bigint
precio	Money	real
sopresid	Integer	bigint

Nombre de las clases: *Semanales, 1Trimestre, 2Trimestres, 3Trimestres*

Nombre de la tabla: *minimos*

Descripción de la tabla: Permite almacenar los precios mínimos obtenidos en el análisis

Clave de la tabla: La clave de la tabla es *analisisid* que se definirá autoincremental enlazándola con el secuenciador *analisisid_seq*

Notas/Comentarios: La tabla recoge la clave foránea de *soportesresistencias*: *sopresid*

Nombre del Campo	Tipo	Tipo BBDD
minimoid	Integer	bigint
precio	Money	real
sopresid	Integer	bigint

Nombre de las clases: *Semanales, 1Trimestre, 2Trimestres, 3Trimestres*

Nombre de la tabla: *niveles*

Descripción de la tabla: Permite almacenar los resultados de los cálculos de los niveles de precios (soportes y resistencias) realizados, seleccionando de entre los máximos y los mínimos, aquellos que cumplan las condiciones del modelo de cálculo y la precisión del rango de precios

Clave de la tabla: La clave de la tabla es *nivelid* que se definirá autoincremental enlazándola con el secuenciador *nivelid_seq*

Notas/Comentarios: La tabla recoge la clave foránea de *soportesresistencias*: *sopresid*

Nombre del Campo	Tipo	Tipo BBDD
nivelid	Integer	bigint
precio	Money	real
sopresid	Integer	bigint

Nombre de la clase: *Operaciones*

Nombre de la tabla: *operaciones*

Descripción de la tabla: Permite almacenar las operaciones realizadas

Clave de la tabla: La clave de la tabla es *operacionid* que se definirá autoincremental enlazándola con el secuenciador *operacionid_seq*

Notas/Comentarios: La tabla recoge la clave foránea de la clave foránea de *valores*: *indice* y *valor*, la clave foráneas de *analisis*: *analisisid*, y la clave foránea de *agentes*: *agente*. Además, cuando se grabe un registro en la tabla *operaciones*, también se grabará un registro bien en la tabla *compras*, bien en la tabla *ventas*

Nombre del Campo	Tipo	Tipo BBDD
operacionid	Integer	bigint
indice	String	text
valor	String	text
analisisid	Integer	bigint
agente	String	text
fecha	Date	date
hora	Time	time

Nombre de la clase: *Apertura*

Nombre de la tabla: *compras*

Descripción de la tabla: Permite almacenar las operaciones aperturadas

Clave de la tabla: La clave de la tabla es *compraid* que se definirá autoincremental enlazándola con el secuenciador *compraid_seq*

Notas/Comentarios: La tabla recoge la clave foránea de *operaciones*: *operacionid*

Nombre del Campo	Tipo	Tipo BBDD
compraid	Integer	bigint
entrada	Money	real
salida	Money	real
stoploss	Money	real
operacionid	Integer	bigint

Nombre de la clase: *Cierre*

Nombre de la tabla: *ventas*

Descripción de la tabla: Permite almacenar las operaciones cerradas

Clave de la tabla: La clave de la tabla es *ventaid* que se definirá autoincremental enlazándola con el secuenciador *ventaid_seq*

Notas/Comentarios: La tabla recoge la clave foránea de *operaciones*: *operacionid*

Nombre del Campo	Tipo	Tipo BBDD
Ventaid	Integer	bigint
Cierre	Money	real
diferencia	Money	real
rentabilidad	Real	numeric(5,2)
operacionid	Integer	bigint

3.4. Diseño del servicio web

Para mantener la responsabilidad del sistema respecto a su fiabilidad y robustez en el acceso a los datos, el modelo de persistencia se gestionará mediante un servicio web que permitirá a los agentes abstraerse de las tareas de mantener aspectos como la calidad y la vigencia de los datos con los que operan. De tal forma, se construirá un servicio web sobre el paradigma de programción REST (Representational State Transfer), al que se podrá acceder mediante HTTP y URI como identificador de recursos, y del que se obtendrá la información de salida en formato XML.

Al igual que en el diseño del diagrama de clases, en el diseño del servicio web se aplicarán una serie de patrones arquitectónicos y de diseño ya contrastados, y que permitirán construir el servicio con las mismas buenas cualidades que otros sistemas en donde ya han sido probadas de forma satisfactoria. Además, se modelara identificando aquellas unidades funcionales que proporcionan un comportamiento diferenciado dentro del servicio, es decir, se diseñará el servicio planteando una programación orientada a componentes que nos permitirá diseñar la funcionalidad del propio servicio con un correcto nivel de desacoplamiento y abstracción.

Desarrollo de software basado en componentes

Se define el término componente siguiendo lo enunciado por Cheesman y Daniels en su libro *Componentes UML*, en donde se observa que, al contrario que en la programación orientada a objetos en la que es posible distinguir claramente entre los conceptos de clase y de objeto, en la programación orientada a componentes el término componente se emplea tanto para referirse a su especificación como para referirse a su instancia. Así, definen el término asociándolo a diferentes etapas del proceso de desarrollo del software:

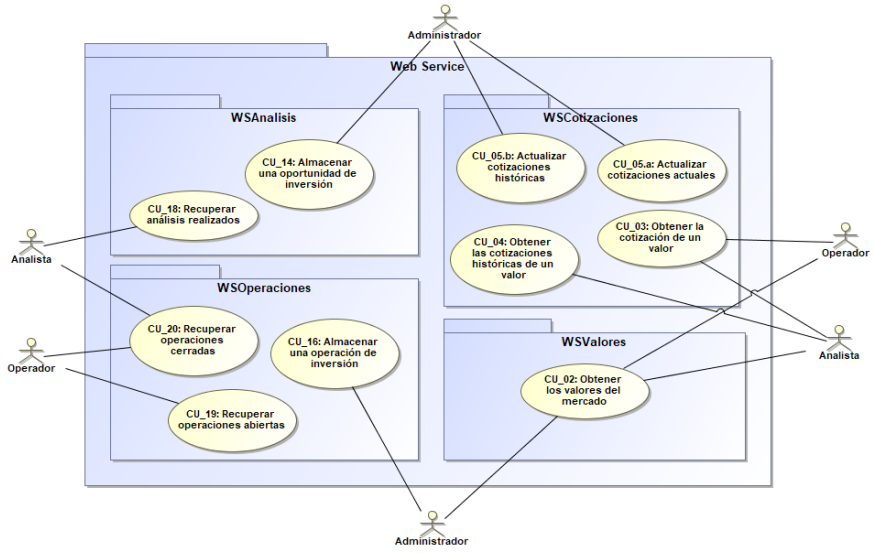
- Especificación del componente
- Implementación del componente
- Componente instalado
- Instancia del componente

• La etapa de especificación representa a un componente como una unidad de software y lo describe según los servicios que proporciona, los recursos que necesita y el comportamiento esperado de cualquiera de sus instancias, de tal forma que coincide con la especificación de sistema de software obtenido desde el punto de vista de la computación en RM-ODP

• La implementación de un componente (artefacto) es la realización de una especificación del componente que puede ser implantada, instalada y reemplazada de manera independiente en uno o más archivos, de tal manera, una especificación del componente se materializa por al menos una implementación del componente en una o varias plataformas distintas.

• Un componente instalado es una copia de una implementación del componente instalada en una máquina concreta. A la hora de instalar un componente (fase de despliegue), es necesario proporcionar la información que permite el despliegue de los ficheros (ejecutables y de configuración) y particularizar la instalación según los requisitos y características del sistema.

• Una instancia de componente es una instancia de un componente instalado y está compuesto por una colección de objetos con identidad única y estado propio que llevan a cabo el comportamiento y la funcionalidad desarrollada.



De tal forma, el resultado de la agrupación realizada anteriormente, nos permite identificar el conjunto de componentes del servicio web:

- Valores del mercado (WSValores), este componente agrupará toda aquella funcionalidad que haga referencia a la información identificativa de los valores cotizados
- Cotizaciones de los valores (WSCotizaciones), este componente agrupará toda aquella funcionalidad que haga referencia a las cotizaciones de un valor
- Análisis de valores (WSAnalisis), este componente agrupará toda aquella funcionalidad que haga referencia a las oportunidades de inversión encontradas
- Operaciones con valores (WSOperaciones), este componente agrupará toda aquella funcionalidad que haga referencia a las operaciones de inversión realizadas
- Comunicación entre agentes, este grupo queda fuera del alcance del servicio web, ya que el sistema utilizará para sus comunicaciones el estándar FIPA-ACL

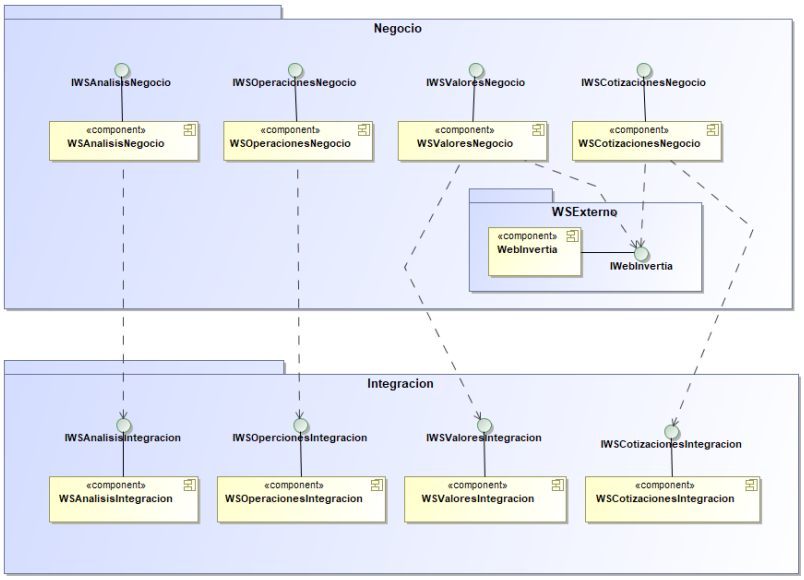
Patrones arquitectónicos

La arquitectura en capas (Layers) permite estructurar y organizar el sistema a gran escala, de tal manera que cada uno de sus componentes puede trabajar con un cierto nivel de abstracción, es decir, con responsabilidades propias e independientes del resto de componentes.

El modelo más empleado de arquitectura en capas es el diseño en tres capas:

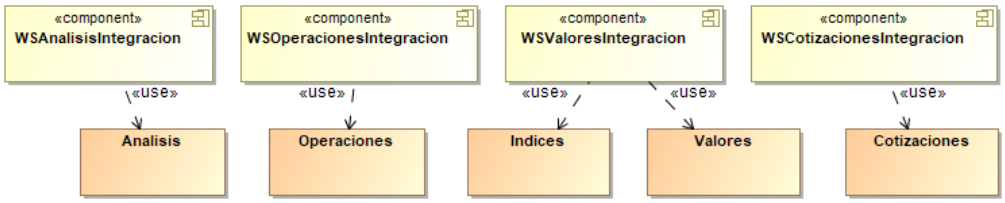
- La capa de presentación, cuya responsabilidad recae principalmente en la interacción con el usuario, es decir, en presentar y capturar la información de usuario.
- La capa de negocio, cuya responsabilidad es gestionar la información y las operaciones que determinan las reglas y la funcionalidad del propio sistema (sus programas)
- La capa de persistencia, cuya responsabilidad es el acceso y almacenamiento de los datos

En nuestro caso, debido a la existencia de agentes y no de usuarios, la presentación y captura de los datos no es un requisito necesario. De tal forma, los agentes llamarán de forma directa al conjunto de interfaces de la capa de negocio para interactuar con el servicio web, o lo que es lo mismo, nuestro sistema multiagente implementará un modelo de diseño en tan solo dos capas: la capa de negocio y la capa de persistencia.



Además, desde un punto de vista de la información, es posible relacionar cada uno de los componentes con cada una de las clases que fundamentan el diseño de la capa de persistencia

desde la perspectiva del almacenamiento de los datos. Así, el diagrama de dependencias de cada uno de los componentes del sistema es el siguiente:



Interfaces del servicio web

El conjunto de interfaces del servicio web recogen por completo la funcionalidad recogida en la fase de gestión de los requisitos, incluso aquella funcionalidad no prioritaria, ya que mejoran el resultado final del servicio web y su coste de desarrollo no es demasiado alto:

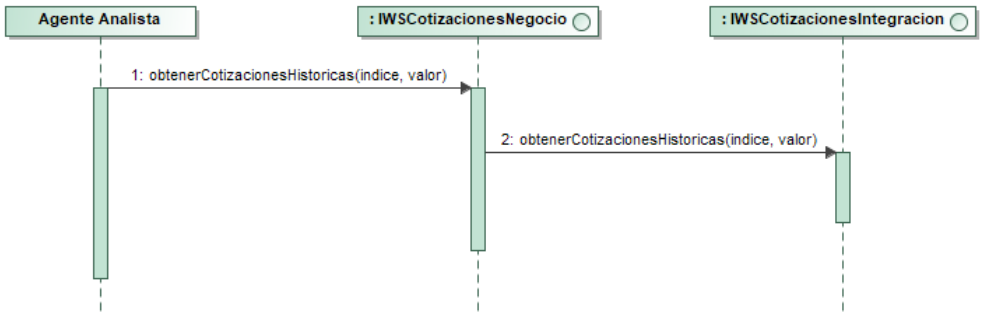
• El componente WSValores estará formado por una única operación(*) que permitirá obtener el conjunto de valores que cotizan en un mercado o índice concreto. A tal respecto, el alcance de este proyecto está limitado a recoger tan solo la información bursátil del Mercado Continuo español, lo que significa que en la práctica tan solo trabajaremos con un único índice: IBEX.

Además, el método tendrá un funcionamiento dinámico respecto a la actualización de la información, de tal forma que cada vez que sea invocado se conectará con el servicio externo, recogerá la información y añadirá a la BBDD aquella que no haya sido añadida con anterioridad.

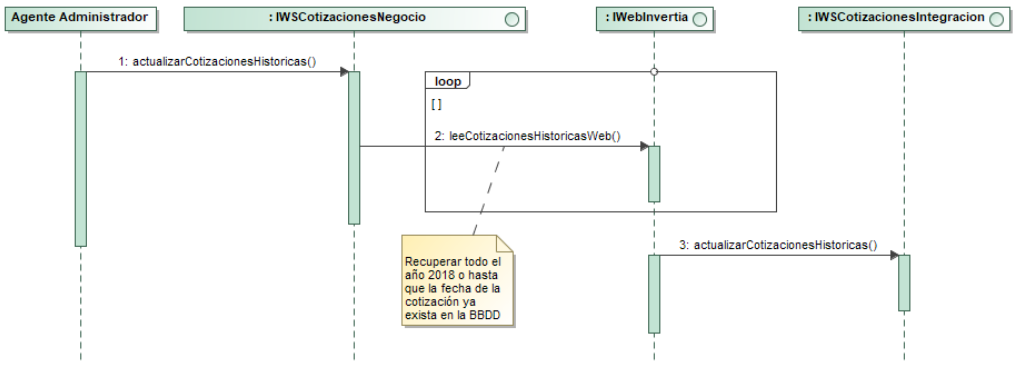
Identificador	Operación	Prioridad	Coste	Agentes
CU_02.a	List<Valores> obtenerCotizadas(String indice)	Must have	3	An, Op
CU_02.b	List<Valores> obtenerCotizadasAdmin(String indice)	Must have	3	Ad

(*) Para mejorar la eficiencia del sistema, se desdobla la operación en dos, de tal forma que solo cuando sea el Administrador quien la invoque se conecte con el WS externo (el resto de perfiles recuperarán los datos de la BBDD)

• El componente WSCotizaciones estará formado por cuatro operaciones que permitirán actualizar la información periódicamente y obtenerla desde la BBDD cuando sea necesaria por los agentes con rol de analista u operador.



De tal forma, el servicio web se conectará en cada cierre de mercado, de lunes a viernes, actualizando las cotizaciones históricas de cada valor, mientras que la frecuencia de actualización de las cotizaciones mientras el mercado esté abierto será de cada cinco minutos. Además, el proceso de actualización de información histórica revisará que se haya obtenido toda la información concerniente al año 2018.



Identificador	Operación	Prioridad	Coste	Agentes
CU_03	Actuales obtenerCotizacion(String indice, String valor)	Must have	3	An, Op
CU_04	List<Historicas> obtenerCotizacionHistorica(String indice, String valor)	Must have	3	An
CU_05.a	actualizarCotizaciones()	Must have	3	Ad
CU_05.b	actualizarCotizacionesHistoricas()	Must have	3	Ad

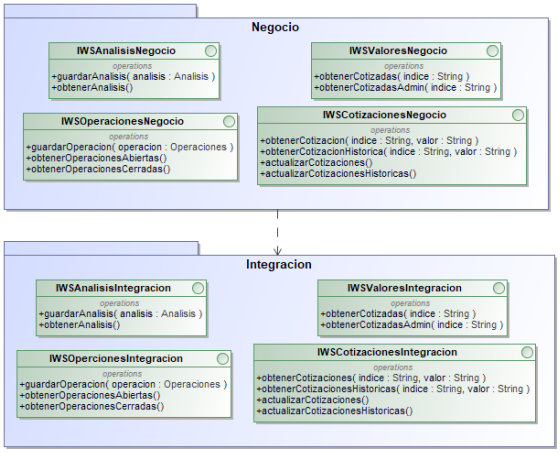
• El componente WSAnalisis estará formado por dos operaciones: Una primera operación que permitirá guardar los análisis realizados y señalados como oportunidad de inversión, y una segunda operación que se implementará pero que en la práctica no se usará, ya que su finalidad es recuperar las oportunidades de inversión para revisarlas y sacar conclusiones de ellas, algo que queda fuera del alcance de este proyecto.

Identificador	Operación	Prioridad	Estimación	Agentes
CU_14	guardarAnalisis(Analisis analisis)	Should have	2	Ad
CU_18	List<Analisis> obtenerAnalisis()	Could have	2	An

• El componente WSOperaciones, de forma similar al componente anterior, estará formado por las operaciones que permitan guardar y recuperar las operaciones realizadas. De igual forma, las operaciones que permiten recuperar la información se implementarán pero no se utilizarán en la práctica, ya que queda fuera del alcance de este proyecto revisar y analizar los resultados obtenidos al formular una operación y llevarla a cabo.

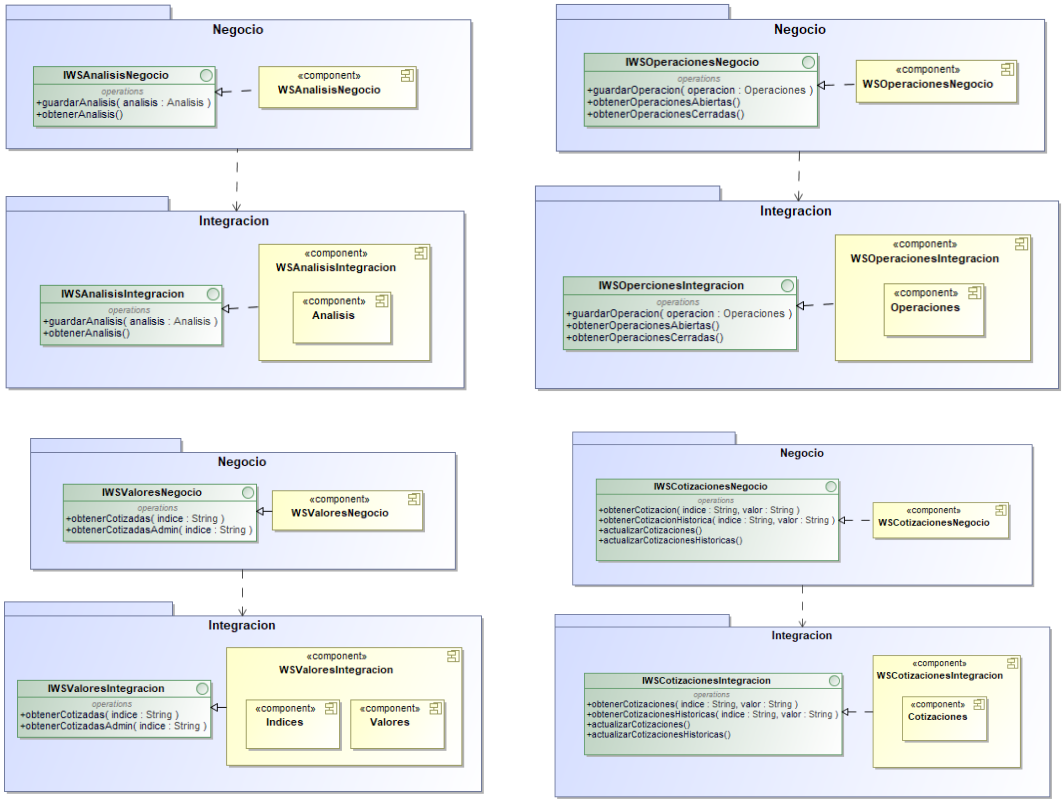
Identificador	Operación	Prioridad	Estimación	Agentes
CU_16	guardarOperacion(Operacion operacion)	Should have	2	Ad
CU_19	List<Operaciones> obtenerOperacionesAbiertas()	Could have	2	Op
CU_20	List<Operaciones> obtenerOperacionesCerradas()	Could have	2	An, Op

Por lo que el conjunto de interfaces del sistema distribuido por capas y por componente quedaría de la siguiente manera:



Detalle de cada uno de los componentes

Desde un punto de vista de la computación, es posible refinar la arquitectura del sistema y detallar con mayor precisión el diseño de cada uno de los componentes. Así, de forma general, la capa de negocio se encargará de procesar y gestionar la funcionalidad del sistema y trabajará de forma síncrona con los elementos de la capa de persistencia, que será la encargada de la gestión de los datos y de su integración en una base de datos relacional.



Además, mejoraremos el rendimiento del sistema posibilitando tanto el acceso remoto como el acceso local a cada uno de los componentes de la capa de negocio y mediante la aplicación del patrón de diseño fachada unificaremos la llamada a cada uno de los accesos definidos.

Patrón fachada (Facade)

El patrón fachada es un patrón de diseño que permite reducir el acoplamiento entre dos subsistemas mediante el uso de una clase que representa a la totalidad del subsistema. Así, mediante su aplicación, el conjunto de subsistemas o capas reducirán su complejidad y minimizarán la dependencia entre ellos, por estructurar y ofrecer un único punto común de entrada a cada uno de ellos que recibirá el nombre de fachada.

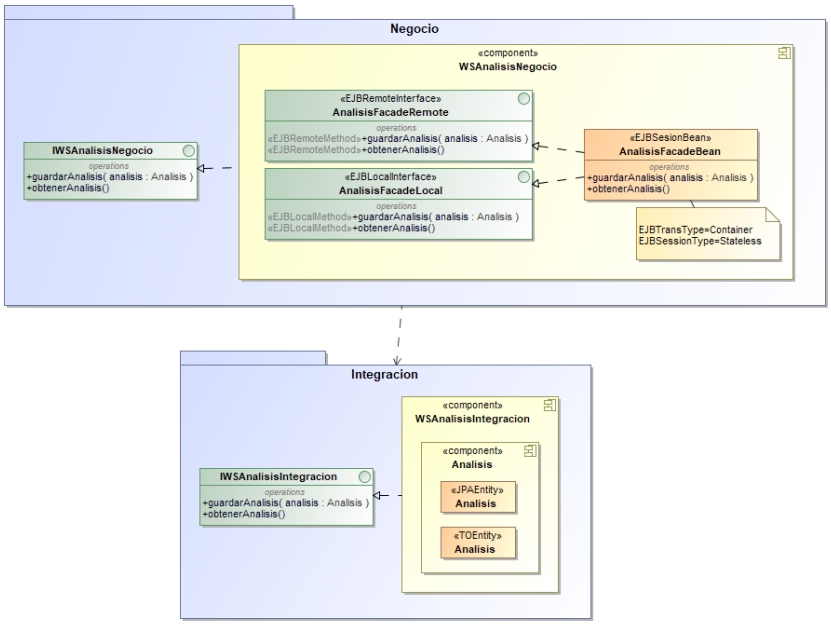
Así, aplicaremos este patrón de diseño a los distintos componentes de la capa de negocio definiendo un nuevo componente EJB de sesión que decidirá de forma dinámica cuál es el tipo de método, local o remoto, al que debe acceder cada petición realizada por el cliente, de tal forma que el conjunto de operaciones serán invocadas de forma síncrona y sin estado, permitiendo que las transacciones sean gestionadas por el contenedor.

De tal forma, si nos centramos en la capa de integración, los datos serán almacenados en una base de datos relacional a la que accederá cada una de las operaciones definidas en el servicio. Además, por una parte, se seguirá la especificación EJB 3.x y se utilizará JPA (Java Persistence API) para gestionar la persistencia y sus relaciones, y por otra parte, se dará soporte al servicio web

siguiendo el patrón de diseño TO (Transfer Object) que facilitará la conversión de los datos a XML, es decir, la capa de persistencia manejará la información sobre dos estándares: JPA para el tratamiento de los datos y su almacenamiento en la BBDD, y TO para la gestión del flujo de información mediante la conversión de los datos a formato XML.

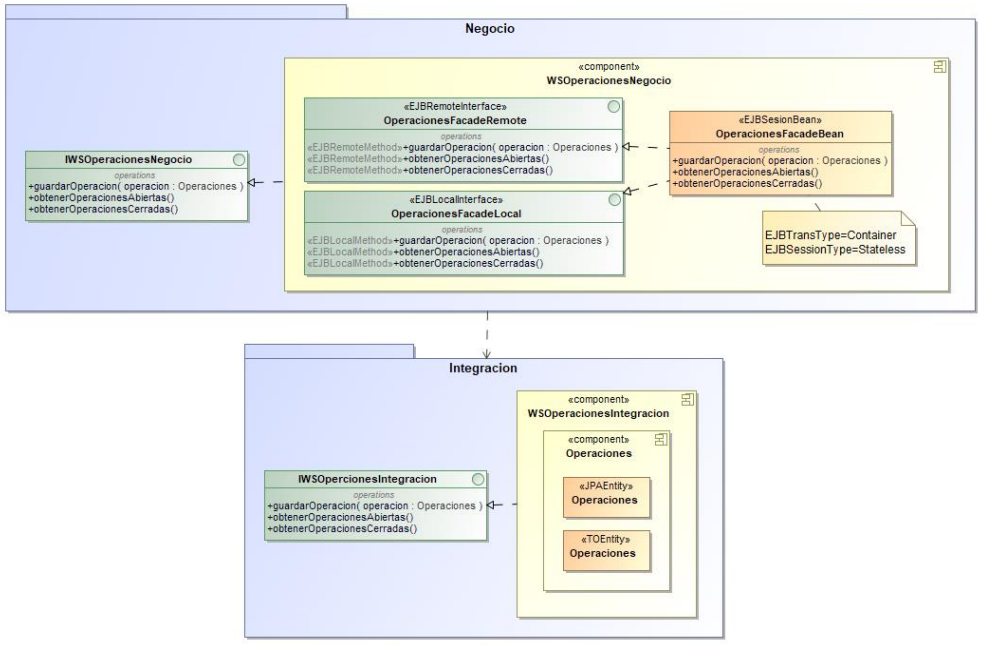
Componente WSAnalisis

Las capas de negocio e integración del componente que dará soporte a los análisis bursátiles se implementará de la siguiente forma:



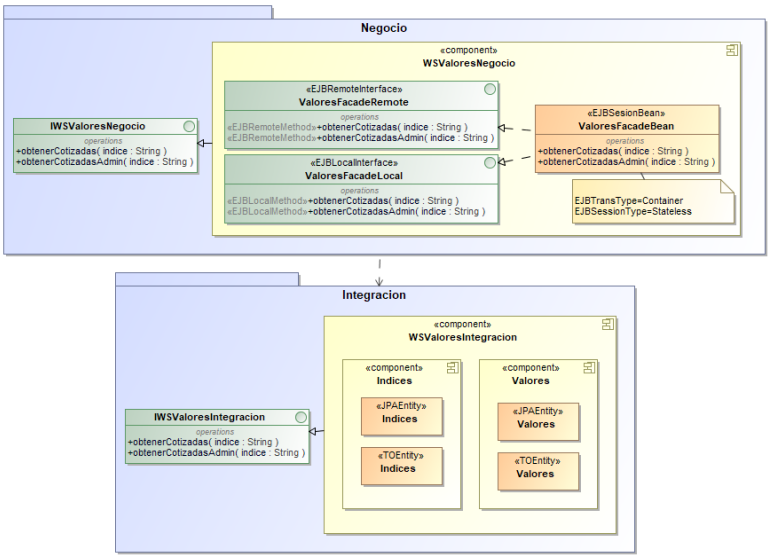
Componente WSOperaciones

Las capas de negocio e integración del componente operacional del sistema quedarán estructuradas de la forma siguiente:



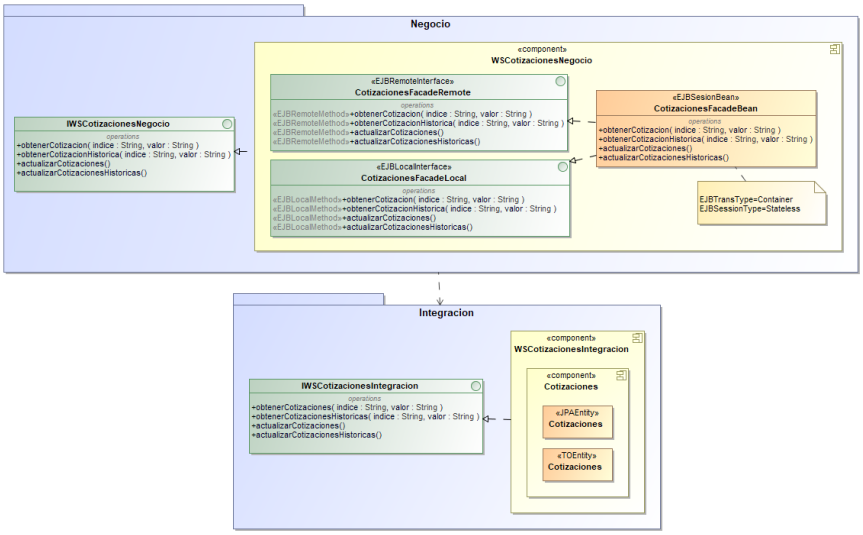
Componente WSValores

Las capas de negocio e integración del componente que permitirá reconocer el conjunto de empresas cotizadas en un mercado financiero serán:



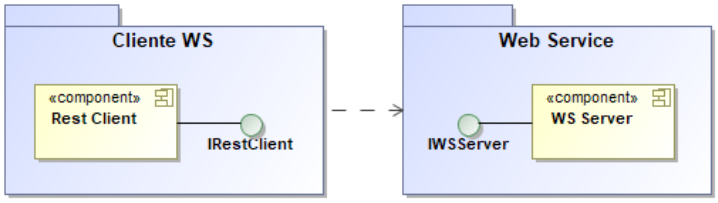
Componente WSCotizaciones

Finalmente, la estructura de las capas de negocio e integración del componente que permitirá reconocer las cotizaciones de cada valor cotizado será:



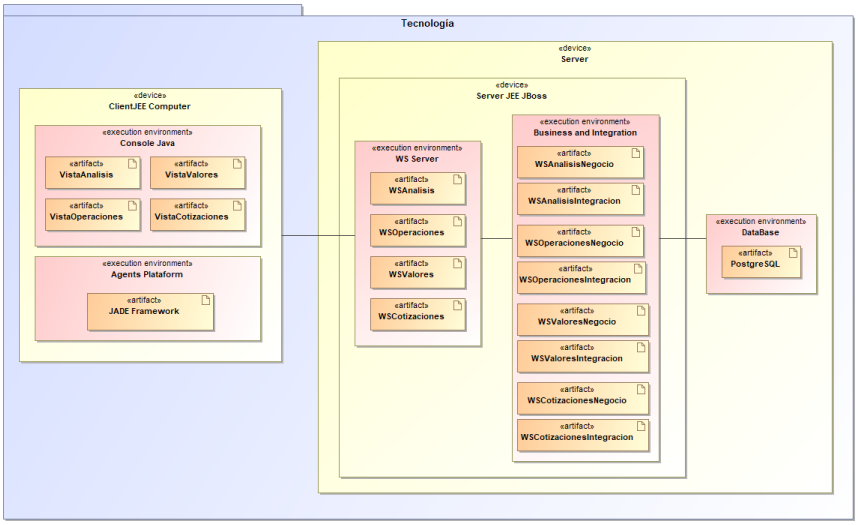
Perspectiva Cliente/Servidor

También presentamos el servicio web como un modelo a dos capas desde una perspectiva de arquitectura cliente/servidor:



Y finalmente, desde el punto de vista de la tecnología, se describirá el sistema según las especificaciones de su infraestructura y el conjunto de tecnologías a utilizar:

- Los clientes(*) no necesitarán ningún tipo de software especial para comunicarse con el servicio web, ya que lo harán mediante el protocolo de comunicaciones HTTP e intercambiarán los datos en formato XML
- Como contenedor de EJB utilizaremos el servidor de aplicaciones JBoss
- Como a gestor de bases de datos utilizaremos PostgreSQL



(*) Clientes: Programa Java para pruebas del WS y Sistema Multiagente

4. Inferencia de los agentes

Además de una base del conocimiento, el sistema necesita de un modelo de razonamiento sobre el que decidir cómo actuar. Así, el diseño del modelo de inferencia de los agentes seguirá las especificaciones recogidas en el estilo arquitectónico BDI y se estructurará siguiendo el sistema de principios del entendimiento recogidos en la obra *Crítica de la razón pura* de Immanuel Kant, que servirá como base conceptual para el modelado de la experiencia de cada agente.

Sistema de los principios del entendimiento

Immanuel Kant en su obra *Crítica de la razón pura*, más en concreto en el capítulo II del libro segundo, *Sistema de todos los principios del entendimiento puro*, propone un sistema para la conexión entre los juicios y los conceptos como base fundamental del entendimiento. Así, en el mencionado capítulo, se presenta una tabla de reglas o principios sobre los que se sustenta la capacidad de análisis trascendental y su síntesis en forma de entendimiento humano:

Todas las intuiciones son magnitudes extensivas

Este principio enuncia que cualquier fenómeno es intuido según su forma en el tiempo y en el espacio, es decir, que cualquier representación conceptual que nos hagamos de un objeto está condicionada a un tiempo y a un espacio

En todos los fenómenos, lo real tiene una magnitud intensiva, es decir un grado

Este principio enuncia que cualquier representación subjetiva que nos hagamos de un objeto a través de los sentidos tiene que tener un grado, es decir, somos capaces de representar un objeto percibido gracias a una magnitud intensiva o unidad de medida física.

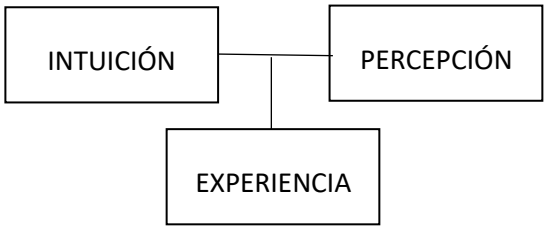
La experiencia solo es posible mediante la representación de un enlace necesario de las percepciones

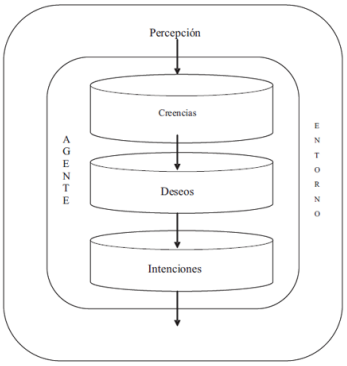
Este principio determina que la experiencia es un conocimiento empírico de los objetos de los sentidos mediante su percepción individual y conjunta, es decir, identificamos la existencia de un objeto dentro de un marco temporal cambiante gracias a tres reglas que preceden a toda experiencia y hacen que esta sea posible:

- La condición de permanencia, que enuncia que aunque un objeto de los sentidos cambie, su sustancia (identidad) permanece
- La condición de secuenciación, que enuncia que todo cambio (empezar a ser) tiene lugar según el principio de causalidad: toda causa tiene su efecto
- La condición de simultaneidad, que enuncia que todas las sustancias que podamos percibir en el espacio de forma simultánea, se hallan en comunidad o acción recíproca

Finalmente, sobre estos tres principios enuncia los postulados del pensar empírico en general, de los que nos quedaremos con los dos primeros que enuncian sobre aquello que es posible y aquello que es real:

- Es posible, aquello que concuerda con las condiciones de la experiencia dese el punto de vista de la intuición
- Es real, aquello que además está relacionado con las condiciones materiales de la experiencia (de la sensación)





4.1. Síntesis de la experiencia de los agentes

Siguiendo como modelo los postulados del pensar empírico en general de Kant, aplicaremos la arquitectura BDI, que implementa las ideas que el filósofo Michael Bratman recoge en su libro *Theory of human practical reasoning*, al sistema multiagente. Así, la arquitectura BDI está formada por tres elementos básicos: las creencias (beliefs), los deseos (desires) y las intenciones (intentions), de tal manera que, según Bratman, la creencia y el deseo son actitudes mentales relacionadas con el comportamiento (la pro-actividad), mientras que la intención está relacionada con el control del propio comportamiento. Además, el deseo y la intención tendrían como factor diferencial el compromiso, que conduce a la persistencia temporal de los planes a realizar.

Creencias: Síntesis de la posibilidad

Se pueden resumir las creencias como los conocimientos que tiene el agente sobre su entorno o dominio de actuación, que en nuestro caso son los mercados financieros. Así, el economista austriaco Ludwig von Mises presenta en su obra *La acción humana* una investigación sobre la manera en que los mercados actúan de forma espontánea sin necesidad de que haya objetivos planificados por los distintos agentes que operan en él (cataláctica) y que se fundamenta en el estudio de la estructura lógica trascendental de la acción humana consciente (praxeológia).

Así, la praxeología entiende que el ser humano, ante un escenario de medios escasos, actúa de forma racional dándole un uso a esos medios, con el propósito de evolucionar de un estado de menos satisfacción a un estado de más satisfacción. De tal forma, nuestro sistema multiagente actuará de forma racional mediante el uso de un conjunto de herramientas o indicadores técnicos con el fin de alcanzar el mayor grado de satisfacción posible, es decir, de rentabilidad.

Además, el modelo de creencias que se plantea tan solo pretende sustentar un conjunto de hipótesis que intentan reconocer, en mayor o menor medida, el grado de interés que hay sobre un valor en un momento concreto, teniendo en consideración la explicación cataláctica anterior, que los precios de mercado son los que son y no se acomodan a ningún modelo ideal, sino más bien a los fines, buenos o malos, que cada uno de los agentes que en él intervienen le hayan querido dar.

Base de conocimiento

La base del conocimiento de los agentes se fundamenta en la información recogida de los mercados, el conjunto de indicadores técnicos empleados y su parametrización. De tal forma, a partir de estos tres elementos, el sistema tendrá que plantear un método para medir con mayor o menor acierto, el grado de interés que hay en el mercado para que el precio de un valor respete la dirección o tendencia que venía mostrando hasta el momento, de tal forma, no se puede afirmar con certeza alguna que su aplicación tengan validez práctica para la obtención de resultados positivos de las operaciones que se efectuarán en los mercados financieros.

Deseos: Materialización de la experiencia

Siguiendo el modelo BDI, los deseos estarían enfocados hacia la planificación del conjunto de objetivos que se quieren alcanzar, es decir, a formular los pasos concretos a seguir para la realización de nuestras creencias. Así, es necesario constituir un modelo de comportamiento que sirva para poner en práctica los conocimientos y poderlos enjuiciar, o lo que es lo mismo, plantear un método de inferencia que nos permita poner en duda, tanto la base de conocimientos como los planteamientos del propio método.

De tal forma, el método plantea como principio de acción del sistema multiagente un conjunto de conjeturas técnicas fundamentadas en la idea de que los precios no son el resultado de una función (algoritmo) de mercado sino que son el resultado de un conjunto de operaciones abstractas que persiguen un fin propio, el cual se pretende distinguir mediante las proposiciones siguientes:

- Medimos la tendencia según la variación del EMA (Precio)
- Medimos el grado de interés según el MACDh y la variación del EMA (Volumen)
- Proponemos un modelo de interés propio: Operar entre dos niveles de precios (soporte y resistencia) cuya relación sea igual o mayor a un 5%, solamente cuando la tendencia sea alcista, y cuando la relación entre el precio actual y la resistencia sea al menos un 2,5%

Intenciones: Análisis y juicio de los datos

Finalmente, las intenciones estarían enfocadas con la realización de los objetivos, es decir, a ejecutar todas las acciones que permitan alcanzar el fin último del agente. Así, si los deseos subjetivizan, en cierta forma, el comportamiento de los agentes mediante la elección de parte de sus conocimientos técnicos como método a poner en práctica, las intenciones llevan a cabo y realizan ese comportamiento. Es, si cabe, donde el sistema multiagente fundamenta una conciencia por su contacto con la realidad, y a partir de los resultados y datos obtenidos, establecer una base de juicio para poner en duda su propia actividad y sus conocimientos.

Así, a partir de las condiciones materiales de la experiencia del sistema multiagente, será posible enjuiciar su propio método de operar y establecer si el sistema plantea como modelo de interés propio aquel que le permita obtener una rentabilidad positiva, es decir, desde una

perspectiva técnica, será posible plantear un modelo de aprendizaje a partir de la supervisión de los datos obtenidos para modificar de forma dinámica el método que fundamenta la manera de actuar del sistema.

De todas formas, queda fuera del alcance de este proyecto modelar un algoritmo de aprendizaje supervisado que permita enjuiciar el método, ya que, entre otras cosas, es necesario estudiar y analizar con detalle los datos obtenidos por el sistema, y revisar la tasa general de acierto de las operaciones en su conjunto, pero de todas formas, se plantearán algunas de las cuestiones que habría que tener en cuenta para ello:

- Supervisar los datos obtenidos y clasificarlos según su rentabilidad
- Incrementar la base de conocimiento con nuevas herramientas técnicas
- Proponer modelos de parametrización alternativos
- Probar modelos aleatorios de selección de parametrización
- Simultanear operaciones sobre un valor con varias parametrizaciones distintas

4.2. Modelización del sistema multiagente

Una de las disciplinas más enfocada al uso de agentes reactivos es la dedicada al diseño de sistemas integrados (embedded systems), que por lo general, son sistemas de computación de uso específico y encapsulados completamente en un dispositivo, es decir, combinan de una manera concreta el hardware y el software para llevar a cabo la funcionalidad que se les ha encomendado.

De tal forma, aunque el actual sistema multiagente está enfocado principalmente al diseño de agentes inteligentes deliberativos y además, también está totalmente abstraído del hardware en donde se implemente mediante el uso de un framework, el conjunto de paradigmas del software empleados en el diseño de sistemas integrados nos servirá como modelo para estructurar la estrategia de programación empleada en el modelado de las capacidades de razonamiento e inferencia de cada uno de los agentes inteligentes que conforman nuestro sistema.

Bucle de control simple

Una de las estrategias de programación de los sistemas integrados es el llamado bucle de control simple que se caracteriza por estar orientada al diseño de tareas que se ejecutan de forma cíclica y continua. Así, este paradigma se basa, principalmente, en la ejecución de un bucle continuo que se encarga de llamar de manera sucesiva a las diferentes subrutinas que gestionan los diversos elementos de hardware o de software, y que mediante algún tipo de rutina de control que verifica su estado, detecta los cambios en su entorno y permite responder a ellos.

Control por eventos

Esta estrategia de programación asume que todo el funcionamiento del sistema está determinado por la sucesión de un conjunto de eventos que están previstos detectar y procesar. Estos eventos, por lo general, ocurren de manera impredecible y el sistema debe atender a ellos de manera inmediata, interrumpiendo si es necesario la tarea que esté realizando en ese momento.

Máquinas de estado

Además del uso de los dos paradigmas anteriores, en los sistemas integrados es habitual la organización de la programación en función del estado del sistema que se ha de controlar. Así, es posible diseñar el software a partir de un conjunto de máquinas de estados sobre la premisa de que el sistema solo se puede encontrar en un único estado en cada instante.

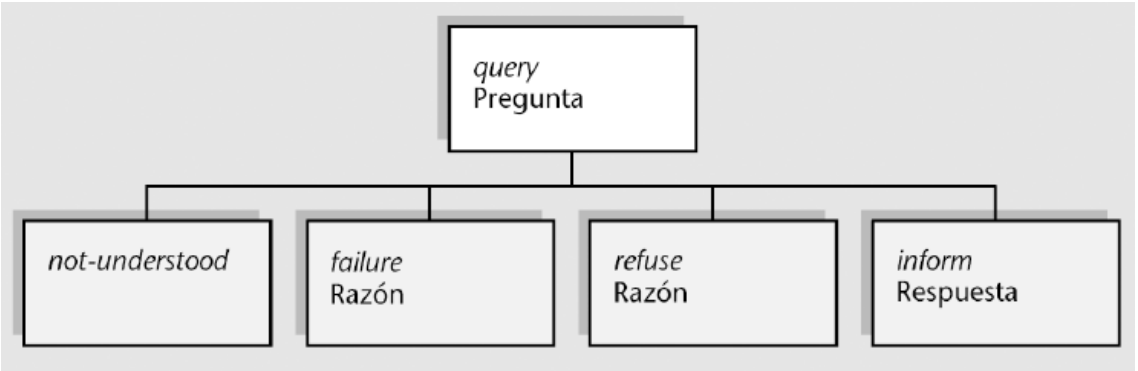
Así, cada uno de los estados en los que se puede encontrar el sistema se asocia a una situación de funcionamiento determinada, independiente y desacoplada del resto del funcionamiento del sistema, de tal forma que es posible modificar el código inherente a un estado sin afectar al funcionamiento del resto de estados.

Técnicas de comunicación

Una de las características más importantes del sistema multiagente es la capacidad de comunicación entre los distintos agentes, de tal forma que pueden pasarse o solicitar información unos a otros para resolver un problema de forma conjunta.

Así, el estándar FIPA proporciona una implementación del lenguaje de comunicación FIPA-ACL (FIPA agent communication language) fundamentado en un modelo de pase de mensajes y en un conjunto de protocolos que definen el volumen de mensajes a intercambiar en una comunicación entre agentes, entre los cuales, el sistema implementará los siguientes:

- FIPA-Inform, para informar de algo a otro agente
- FIPA-Query, para consultar algo a otro agente



Estrategia de programación

El sistema multiagente estará formado por tres tipos o roles de agentes distintos según sus funciones específicas dentro del sistema: el rol de administrador, el de analista y el de operador. De tal forma, no todos los agentes tendrán las mismas capacidades, sino que según su rol en el sistema, los agentes tendrán unas u otras facultades para interactuar, administrar, planificar o estudiar y analizar el modelo desarrollado.

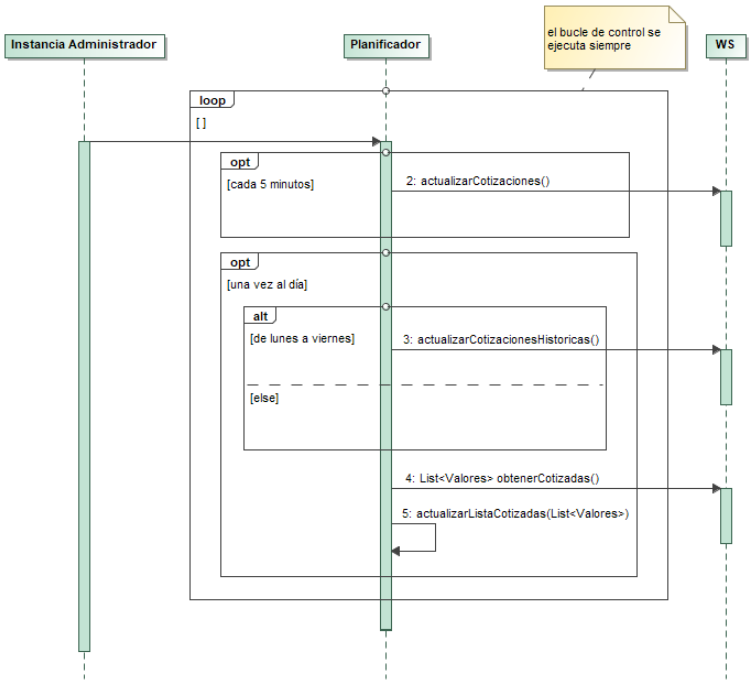
Así, en función del rol del agente dentro del sistema, la programación se orientará más hacia un modelo de agente deliberativo o se orientará más hacia un modelo de agente reactivo, de tal forma que su estructuración seguirá, en mayor medida, una de las estrategias mencionadas con anterioridad. De todas formas, por norma general, el diseño de cada agente seguirá un modelo de programación híbrido, es decir, una combinación de paradigmas de programación.

4.2.1. Rol de agente administrador

El agente administrador es la conclusión de la búsqueda de una mayor eficiencia en el diseño del sistema según los principios de bajo acoplamiento y especialización en las responsabilidades de cada uno de los componentes y agentes que lo forman. Así, la separación de la actividad de los agentes según su utilidad dentro del modelo, las capacidades comunicativas dentro del sistema y el diseño distribuido del modelo de integración de datos, necesitan de la figura del administrador del sistema para gestionar todos los aspectos comunes como la planificación de los sucesos repetitivos, la distribución de la información y su sincronización.

De tal forma, las principales tareas del agente están enfocadas, bien al mantenimiento de los datos mediante la invocación al servicio web de forma periódica, bien al soporte y la ayuda al resto de agentes mediante el almacenamiento de información común y su distribución a través de mensajes FIPA. Por otra parte, para mejorar el rendimiento del sistema, cada agente tendrá un espacio de memoria propio en el que, mediante listas, guardará la información relacionada con su actividad.

Así, el agente implementará mediante un bucle de control simple los ciclos recursivos que actualizan la información de mercado y mediante la gestión de eventos el manejo de las comunicaciones entre agentes a través de un modelo mixto entre el paso de mensajes FIPA y el uso de una pizarra central representada por el propio rol de agente administrador.



Por tanto, tal y como se ha mencionado, el administrador manejará un conjunto de listas que le permitirán manejar desde memoria la información y la parametrización del sistema. Así, el administrador manejará las siguientes operaciones identificadas en los casos de uso:

Caso Uso	Descripción	Operación
CU_02	Obtener los valores del mercado	List<Valores> obtenerCotizadas(String indice)
CU_05.a	Actualizar cotizaciones actuales	actualizarCotizaciones()
CU_05.b	Actualizar cotizaciones históricas	actualizarCotizacionesHistoricas()
CU_14	Almacenar una oportunidad de inversión	guardarAnalisis(Analisis analisis)
CU_16	Almacenar una operación de inversión	guardarOperacion(Operacion operacion)

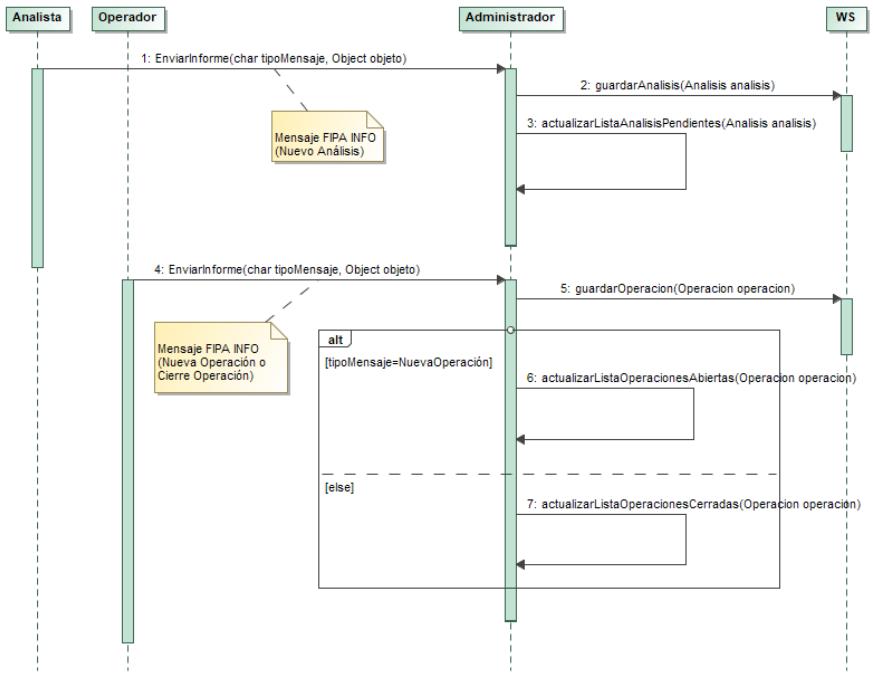
Y las siguientes operaciones enfocadas en el uso y manejo de listas de datos:

Operación	Descripción
inicializarConfiguraciónSistema()	Inicializa parámetros por defecto del sistema
actualizarListaCotizadas(List<Valores>)	Manejo lista de valores de un mercado
actualizarListaAnalisisPendientes(Analisis analisis)	Manejo lista análisis pendientes
actualizarListaOperacionesAbiertas(Operacion operacion)	Manejo lista operaciones abiertas
actualizarListaOperacionesCerradas(Operacion operacion)	Manejo lista operaciones cerradas
actualizarConfiguración(Metodo método)	Manejo de la configuración del sistema

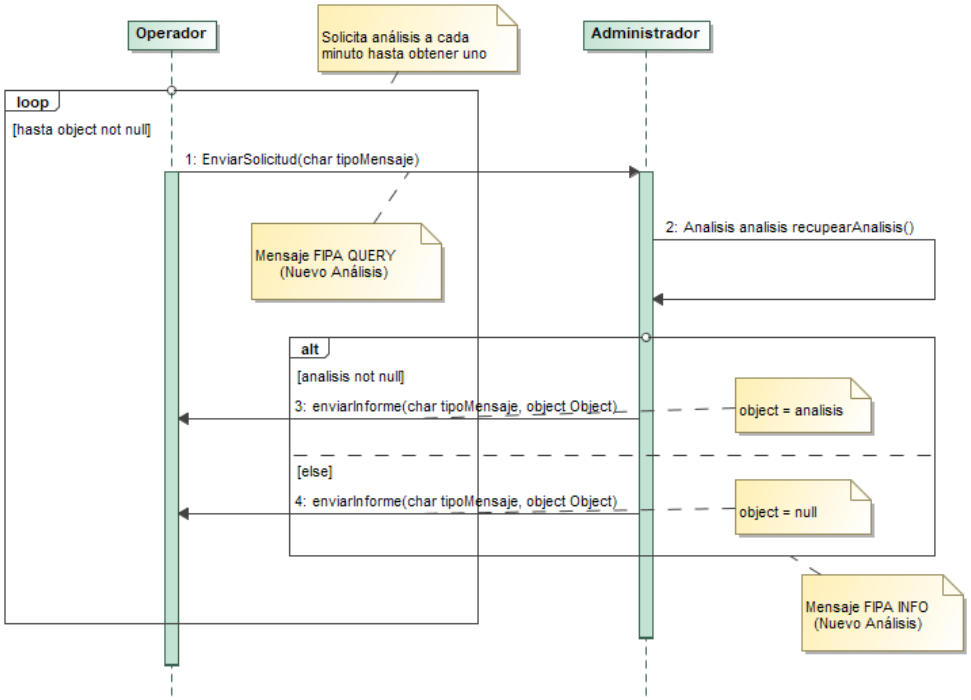
También manejará y canalizará los mensajes entre los distintos agentes del sistema. Así, los casos de uso referentes al manejo de informes y solicitudes, se gestionarán según el paradigma de la programación orientada a eventos y el conjunto de especificaciones que FIPA implementa para la comunicación por medio de mensajes.

Identificador	Descripción	Agentes
CU_13	Informar de una oportunidad de inversión	Ad, An
CU_17	Solicitar una oportunidad de inversión	Op
CU_15	Informar de una operación de inversión	Op

Informes (mensajes tipo INFO)



Solicitudes (mensajes tipo QUERY)



Por otra parte, el diseño del modelo sobre una arquitectura BDI requiere que el sistema maneje una terminología y un conjunto de herramientas técnicas común. Así, el sistema multiagente estará configurado para utilizar y compartir la parametrización y el método de aprendizaje empleados, por lo que será posible modificar esta configuración de forma dinámica según los resultados obtenidos y el grado de experiencia de los agentes.

Para ello, será necesario un nuevo tipo de mensaje que permita trasladar a todos los agentes el modelo conceptual vigente y que estará centrado tanto en la capacidad para el análisis técnico como para la operativa bursátil, por lo que su configuración estará compuesta por una parametrización, un conjunto de reglas y un modelo de priorización:

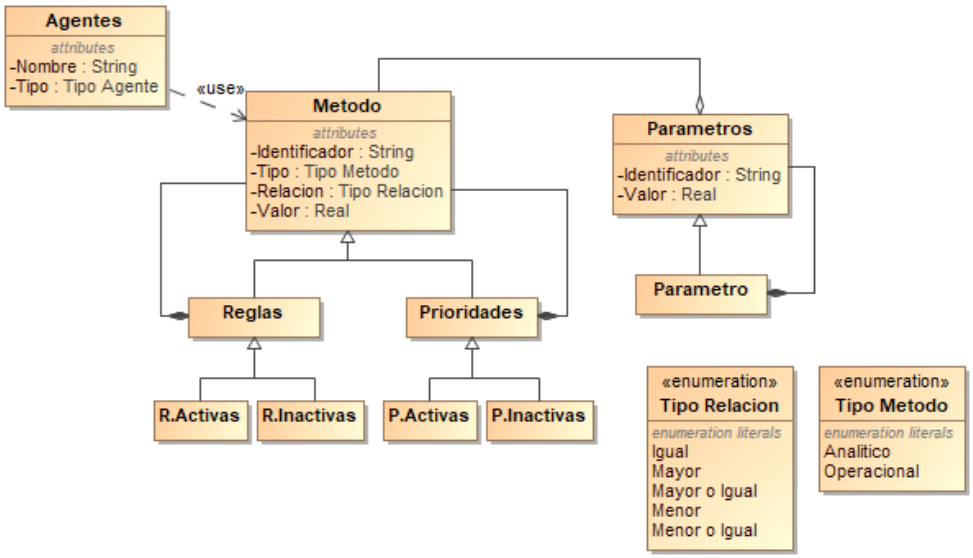
MÉTODO DE ANÁLISIS

PARÁMETRO	***REGLA***	***PRIORIDAD***
Nº SESIONES EMA PRECIO	*EMA PRECIO*	*EMA VOLUMEN*
Nº SESIONES EMA VOLUMEN	*HISTOGRAMA MACD*	
Nº SESIONES EMA CORTA MACD	*RANGO ENTRE NIVELES*	
Nº SESIONES EMA LARGA MACD		
Nº SESIONES SEÑAL MACD		
COMBINACIÓN MÁXIMOS Y MÍNIMOS		
PRECISIÓN DEL RANGO		
MODELO SELECCIÓN DE NIVEL		

MÉTODO OPERACIONAL

PARÁMETRO	***REGLA***	***PRIORIDAD***
VARIACIÓN PRECIO SALIDA	*APERTURA OPERACIÓN*	
VARIACIÓN STOPLOSS		

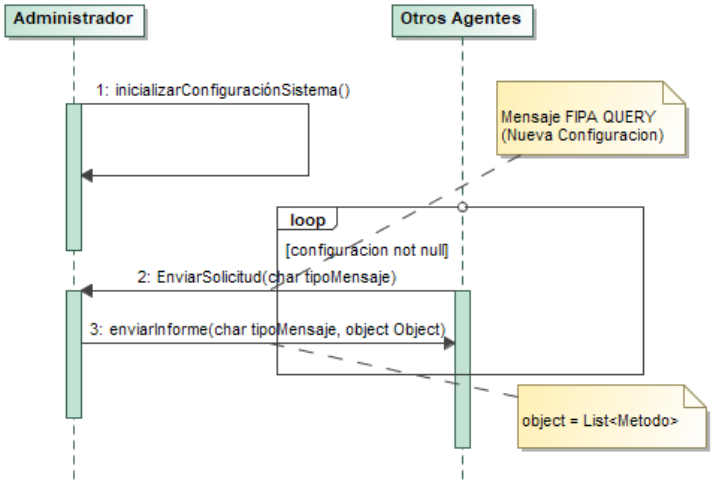
De tal forma, es posible ampliar el diagrama de clases y desarrollar de forma detallada el método sobre el que se fundamenta la actividad del sistema multiagente (ver apartado 3.5, síntesis de la funcionalidad de los agentes). Para ello, además del patrón de diseño Estado (State) visto con anterioridad, se aplicará el patón Objeto compuesto (Composite) ya que necesitamos tratar colecciones de elementos de forma uniforme.



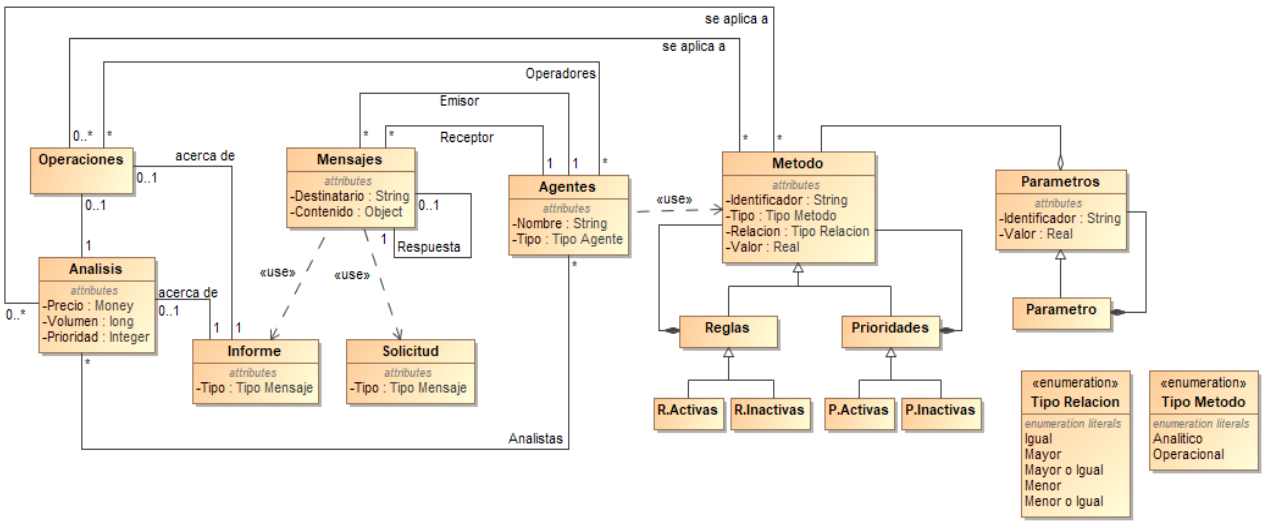
En consecuencia, la configuración por defecto del sistema será inicializada en la memoria del agente administrador y distribuida al resto de los agentes, que la duplicarán en sus respectivas

memorias, mediante mensajes FIPA tipo QUERY según una tipología. Así, la tipología completa de los mensajes FIPA estará compuesta por los siguientes tipos:

- Nueva Configuración (0), para informar o solicitar una nueva parametrización
- Nuevo Análisis (1), para informar o solicitar un nuevo análisis
- Nueva Operación (2), para informar de la apertura de una operación
- Cierre Operación (3), para informar del cierre de una nueva operación



Por otra parte, si se quisiera almacenar la metodología empleada a la hora de obtener un análisis o de ejecutar una operación, habría que ampliar el número de tablas de nuestra base de datos y hacer unos pequeños cambios en la funcionalidad existente:



Cambios en la funcionalidad

Habría que modificar los casos de uso CU_14 y CU_16, añadiendo como parámetros de las operaciones a la clase *Metodo*

Identificador	Operación	Agentes
CU_14	guardarAnalisis(Analisis analisis, Metodo metodo)	Ad
CU_16	guardarOperacion(Operacion operacion, Metodo metodo)	Ad

Cambios en la BBDD

El marco conceptual por defecto del sistema está definido por los siguientes métodos, es decir, por el siguiente conjunto de reglas y de prioridades:

- Regla EMA PRECIO o diferencia entre la EMA actual y la EMA anterior del precio. Para su cálculo, necesita como parámetro el número de sesiones.

REGLA	*RELACIÓN*	*VALOR*
EMA PRECIO	*>*	*0*

PARÁMETRO	*VALOR DEFECTO*
Nº SESIONES EMA PRECIO	*8*

- Regla HISTOGRAMA MACD, que compara si el MACDh es positivo. Necesita como parámetros aquellos necesarios para el cálculo del MACD.

REGLA	*RELACIÓN*	*VALOR*
HISTOGRAMA MACD	*>*	*0*

PARÁMETRO	*VALOR DEFECTO*
Nº SESIONES EMA CORTA MACD	*12*
Nº SESIONES EMA LARGA MACD	*26*
Nº SESIONES SEÑAL MACD	*9*

- RANGO ENTRE NIVELES, que compara si la diferencia entre la resistencia y el soporte es mayor o igual al 5%. Necesita como parámetros los definidos en la tabla *soportesresistencias* como *Precisión* y *Tipo Calculo* (Media = 0; Media+1: 1), además de una combinatoria posicional del *Tipo Nivel* (Semanal = 0; Diario y estudio de un trimestre: 1; Diario y dos trimestres: 2; Diario y tres trimestres: 3), de tal forma que si el valor de la combinación es 1001 se calcularán los máximos 0001=Semanales y 1000=Diarios y tres trimestres.

REGLA	*RELACIÓN*	*VALOR*
RANGO ENTRE NIVELES	*≥*	*5*

PARÁMETRO	*VALOR DEFECTO*
COMBINACIÓN MÁXIMOS Y MÍNIMOS (Combinación Tipo Nivel)	*1111*
PRECISIÓN DEL RANGO	*1*
MODELO SELECCIÓN DE NIVEL (Tipo calculo)	*1*

- La prioridad EMA VOLUMEN seguirá el mismo modelo que el precio.

REGLA	*RELACIÓN*	*VALOR*
EMA VOLUMEN	*>*	*0*

PARÁMETRO	*VALOR DEFECTO*
Nº SESIONES EMA VOLUMEN	*8*

- Regla APERTURA OPERACIÓN, que compara si la diferencia entre la resistencia y el precio es mayor o igual que el 2,5%.

REGLA	*RELACIÓN*	*VALOR*
APERTURA OPERACIÓN	*≥*	*2,5*

PARÁMETRO	*VALOR DEFECTO*
VARIACIÓN PRECIO SALIDA	*+2,5*
VARIACIÓN STOPLOSS	*-1*

De tal forma, habría que modificar la base de datos y añadir cuatro tablas más, en las que el tipo de enumeración *Tipo Relacion* agrupará los valores siguientes: Igual=0; Mayor=1; Mayor Igual=2; Menor=3; Menor Igual=4

Nombre de la tabla: *analitica*

Descripción de la tabla: Permite almacenar los métodos analíticos

Clave de la tabla: La clave de la tabla es doble *analisisid* + *metodoid*

Nombre del Campo	Tipo	Tipo BBDD
analisisid	Integer	bigint
metodoid	String	text
relacion	Tipo Relacion	char
valor	Real	real

Nombre de la tabla: *parametrosanalitica*

Descripción de la tabla: Permite almacenar los métodos operacionales

Clave de la tabla: La clave de la tabla es triple *analisisid* + *metodoid* + *parametroid*

Nombre del Campo	Tipo	Tipo BBDD
analisisid	Integer	bigint
metodoid	String	text
parametroid	String	text
valor	Real	real

Nombre de la tabla: *operativa*

Descripción de la tabla: Permite almacenar los métodos operacionales

Clave de la tabla: La clave de la tabla es doble *operacionid* + *metodoid*

Nombre del Campo	Tipo	Tipo BBDD
operacionid	Integer	bigint
metodoid	String	text
relacion	Tipo Relacion	char
valor	Real	real

Nombre de la tabla: *parametrosoperativa*

Descripción de la tabla: Permite almacenar los métodos operacionales

Clave de la tabla: La clave de la tabla es doble *operacionid* + *metodoid* + *parametroid*

Nombre del Campo	Tipo	Tipo BBDD
operacionid	Integer	bigint
metodoid	String	text
parametroid	String	text
valor	Real	real

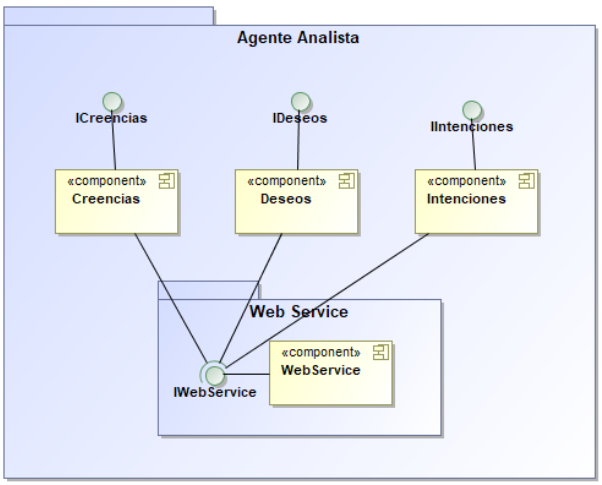
4.2.2. Rol de agente analista

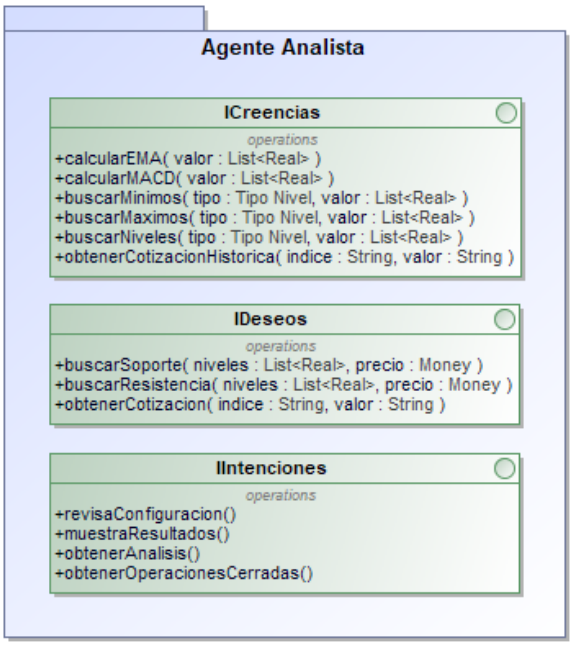
El agente analista es el resultado de la estructuración del sistema multiagente sobre un modelo de bajo acoplamiento y la abstracción de la capacidad de ganar experiencia en relación a las facultades de intuir y percibir. De tal forma, las principales tareas del agente están enfocadas, al análisis de la información y al cálculo a partir de una serie de conocimientos técnicos según la terminología descrita anteriormente: la parametrización de un conjunto indicadores, la aplicación de un determinado método de análisis y el establecimiento de un modelo de prioridades.

Así, el agente se diseñará, principalmente, según las especificaciones BDI y el paradigma de programación orientada a componentes, aunque también implementará mediante un bucle de control simple los ciclos recursivos de los que tenga que hacerse cargo. De tal forma, la distribución de las operaciones identificadas en los casos de uso por componente es la siguiente:

Caso Uso	Operación	Componente
CU_02	List<Valores> obtenerCotizadas(String indice)	
CU_04	List<Historicas> obtenerCotizacionHistorica(String indice, String valor)	Creencias
CU_06	EMA calcularEMA(List<real> datos)	Creencias
CU_07.a	MACD calcularMACD(List<real> datos)	Creencias
CU_08.a	List<Real> buscarMinimos(char tipoNivel, List<real> datos)	Creencias
CU_08.b	List<Real> buscarMaximos(char tipoNivel, List<real> datos)	Creencias
CU_08.c	List<Real> buscarNiveles(char tipoNivel, List<real> datos)	Creencias
CU_09	EMA calcularEMA(List<real> datos)	Creencias
CU_03	Actuales obtenerCotizacion(String indice, String valor)	Deseos
CU_10	Real buscarSoporte(List<real> niveles, Real precio)	Deseos
CU_10	Real buscarResistencia(List<real> niveles, Real precio)	Deseos
CU_18	List<Analisis> obtenerAnalisis()	Intenciones
CU_20	List<Operaciones> obtenerOperacionesCerradas()	Intenciones

(*) Nota 1: el caso de uso CU_10: Calcular distancia entre soportes y resistencias, ha sido desdoblado en dos operaciones
(*) Nota 2: además al componente Intenciones se añaden las operaciones revisaConfiguración() y muestraResultados()





Además, para hacerse cargo de la interacción entre los distintos componentes, se han diseñado tres nuevas clases que se responsabilizarán de la ejecución de la metodología inherente a cada uno de ellos. Así, siguiendo el patrón Experto (Expert) se han diseñado tres nuevas clases con toda la información necesaria para el tratamiento de cada componente: ControladorCreencias, ControladorDeseos, ControladorIntenciones.

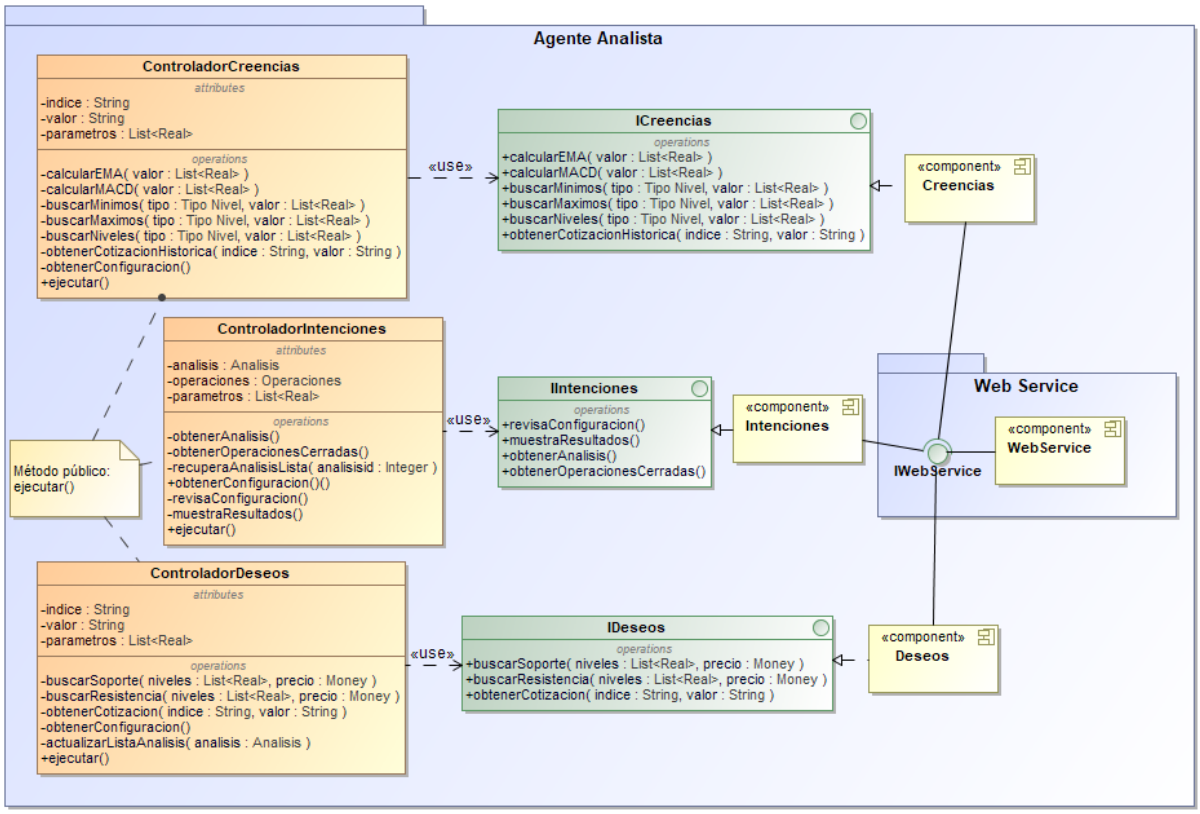
Así, siguiendo el modelo de uso de listas que permitirán a cada agente duplicar la información en su memoria y por lo tanto ganar en eficiencia, será necesario definir un nuevo conjunto de operaciones, de las que el bucle de control del rol de analista usará:

Operación	Descripción
seleccionarValorLista()	Manejo lista de valores de un mercado
actualizarListaCotizadas(List<Valores>)	Manejo lista operaciones abiertas

Y las clases expertas en el tratamiento de cada componente BDI usarán:

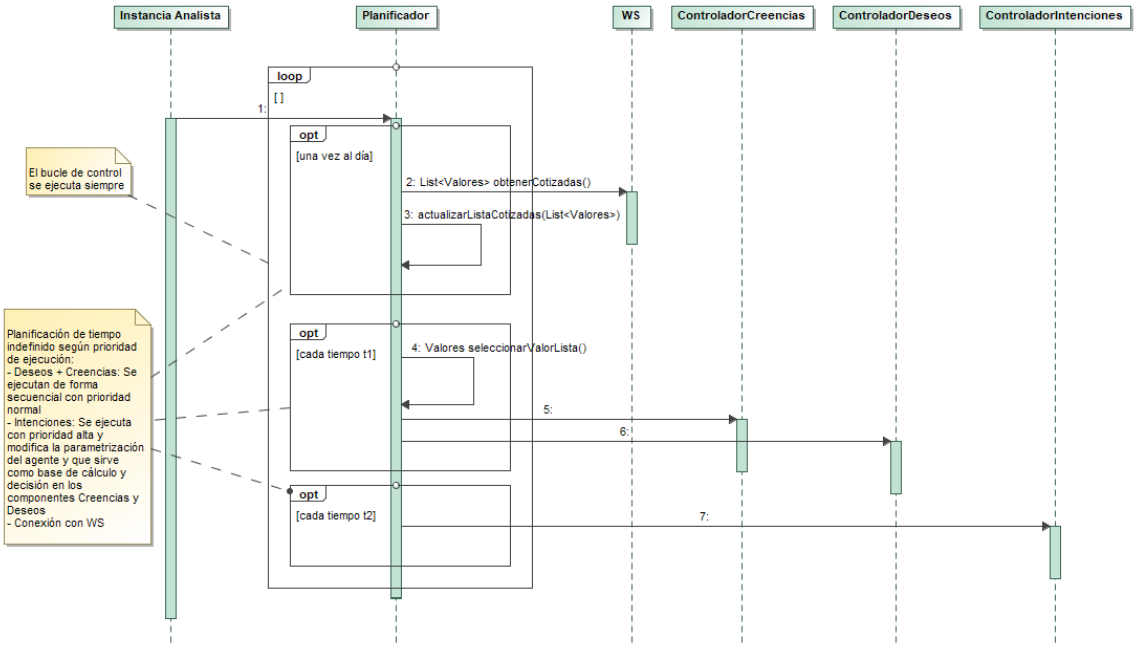
Operación	Ámbito	Descripción
obtenerConfiguracion()	General	Recuperación configuración común sistema
actualizarListaAnalisis(Analisis analisis)	Deseos	Manejo lista análisis (oportunidades inversión)
recuperaAnalisisLista(Integer analisisid)	Intenciones	Manejo lista análisis (oportunidades inversión)
revisaConfiguracion()	Intenciones	Estudio configuración del sistema
muestraResultados()	Intenciones	Mostrar pantall y log resultados revisión configuración
ejecutar()	General	Método público invocación clase experta

De tal forma, el agente desde un punto de vista de la computación estará compuesto por cada uno de los componentes BD, sus interfaces y cada uno de los expertos diseñados:



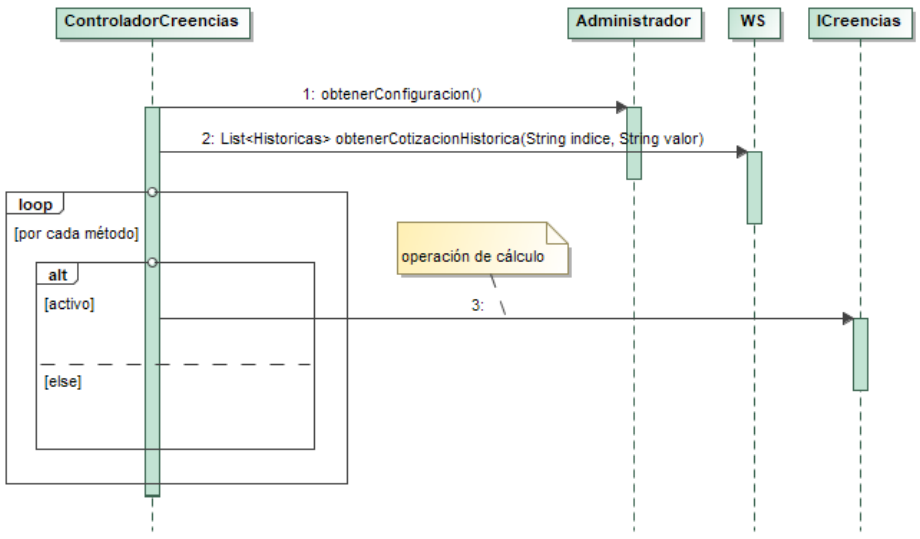
En consecuencia, el comportamiento del agente se implementará mediante un bucle de control simple que invocará cada tiempo t_1 a los controladores (clases expertas) de los componentes de dan soporte a las creencias y los deseos del agente, y cada tiempo t_2 al controlador del componente que estructura sus intenciones. Para ello, se definirá en cada clase experta un único método público (*ejecutar*) que permitirá invocarlas y a su vez, invocar a un componente concreto.

Así, en la planificación del bucle de control será importante definir un modelo de prioridades que garantice que los recursos del sistema serán dedicados al manejo de intenciones antes que a las actividades de análisis del conocimiento, lo que permitirá modificar con total garantía la configuración del sistema (metodología y parametrización) de tal forma que todo el sistema en general pueda obtener en cualquier momento la versión más actualizada de ella y así utilizar un modelo conceptual común.



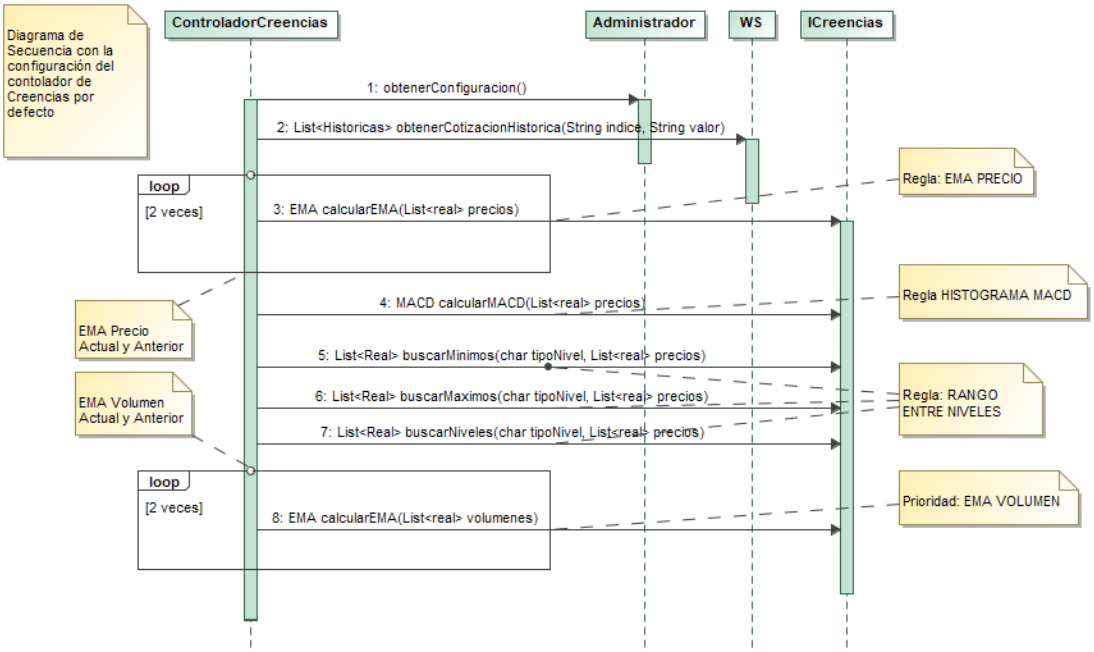
Controlador de Creencias

El controlador de creencias será invocado cada tiempo t_1 por el planificador del agente una vez se haya seleccionado un valor para analizar. Su misión es realizar, según la parametrización del sistema, todos los cálculos necesarios para poder analizar un valor bursátil, de tal forma, que primeramente, obtendrá las cotizaciones históricas del valor y seguidamente, ejecutará las operaciones necesarias para el cálculo de cada método activo en la configuración del sistema.



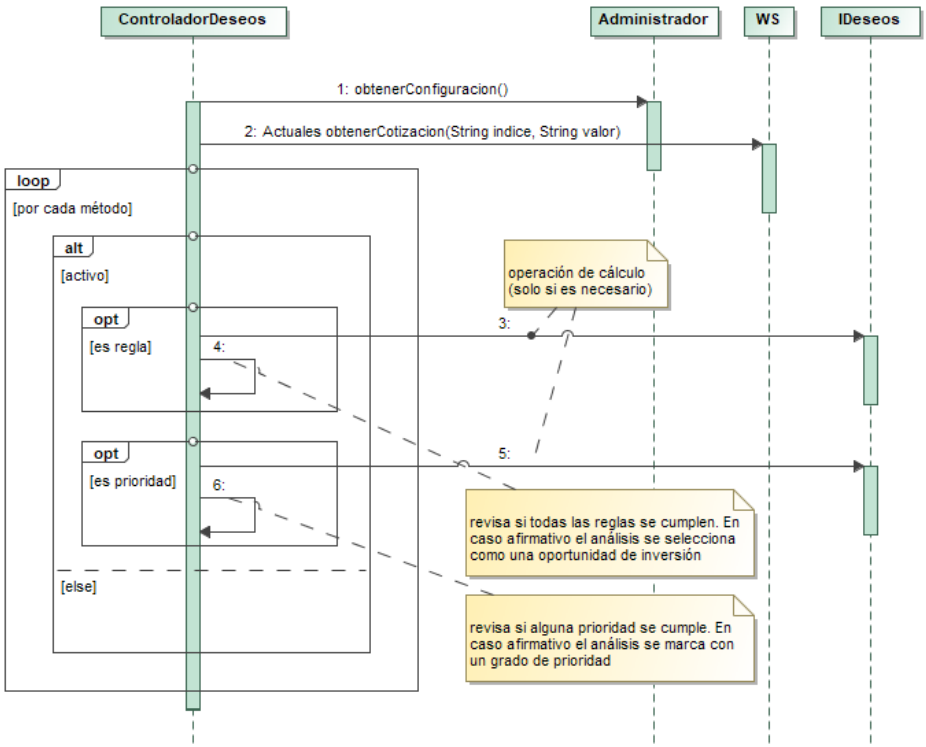
Seguidamente se muestra la secuencia de operaciones para el conjunto de reglas y prioridades definidas por defecto en términos de análisis bursátil:

REGLAS	***PRIORIDADES***
EMA PRECIO	*EMA VOLUMEN*
HISTOGRAMA MACD	
RANGO ENTRE NIVELES	



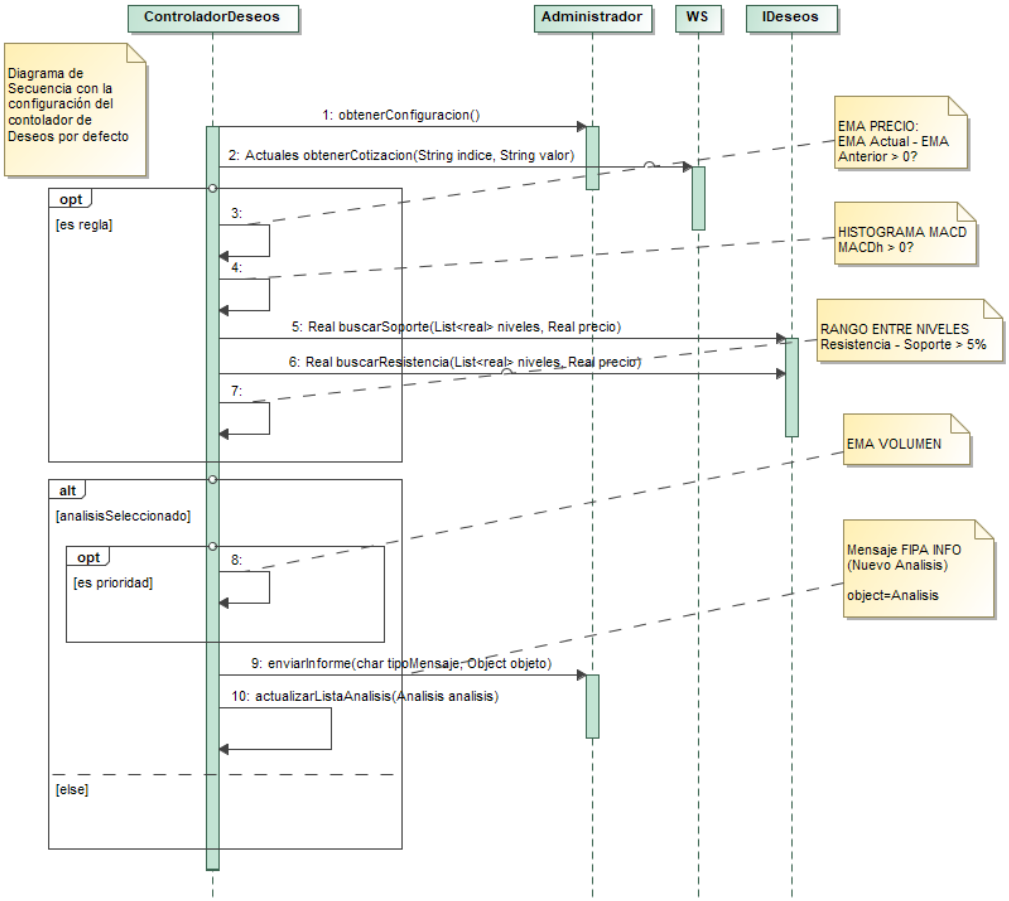
Controlador de Deseos

El controlador de deseos será invocado a continuación del controlador de creencias, una vez finalicen todos los procesos llevados a cabo por este. Su principal misión es comprobar si las reglas definidas se cumplen, es decir, si el análisis cumple con todas y cada una de las relaciones establecidas en ellas y se puede seleccionar como una oportunidad de inversión.



Además, los análisis serán marcados con una prioridad según el grado de cumplimiento con los métodos establecidos para ello. Así, la oportunidad de inversión tendrá una prioridad 0 cuando no cumpla ningún criterio, tendrá prioridad 1 si cumple uno, 2 con dos, y así sucesivamente.

Como en el caso de las creencias, también se muestra la secuencia de operaciones para el conjunto de reglas y prioridades definidas por defecto:



Controlador de Intenciones

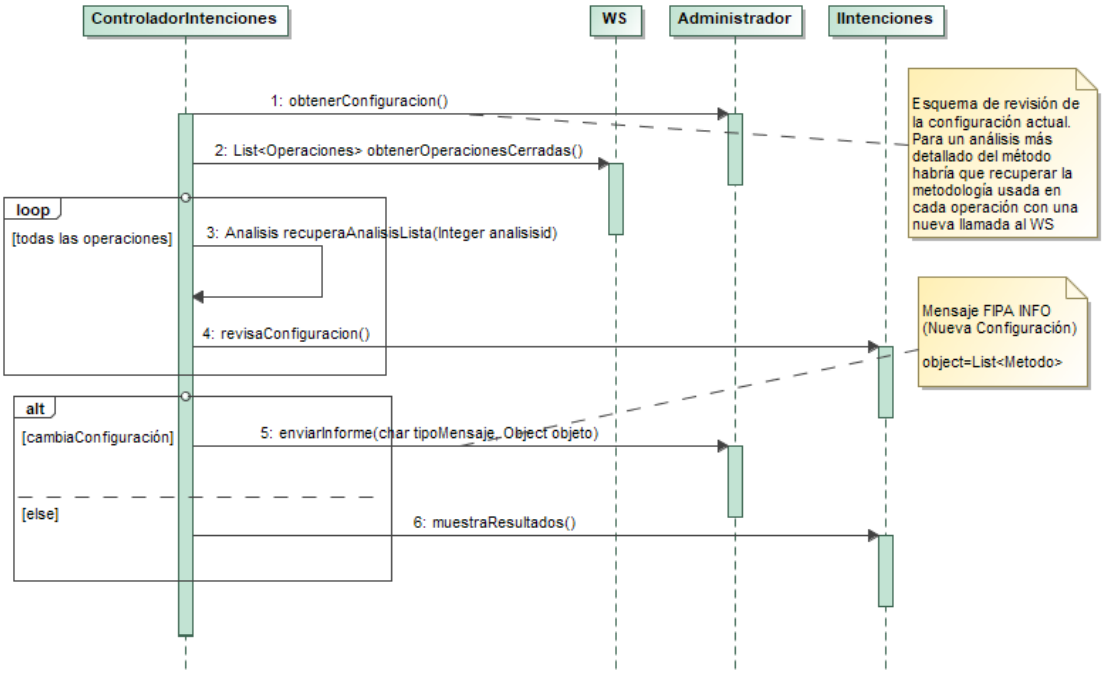
Por último, el controlador de intenciones será invocado cada tiempo t_2 por el planificador del agente de forma independiente de las interacciones con los otros dos controladores. Su misión es analizar a partir de los resultados obtenidos si el método empleado o los parámetros utilizados están obteniendo buenos resultados.

Así, este controlador tiene la facultad de modificar tanto el método de reconocimiento de un análisis como una oportunidad de inversión (activando y desactivando reglas o prioridades, y cambiando sus valores de referencia), como los resultados del uso de las herramientas técnicas (modificando la parametrización del cálculo de cada indicador).

De tal forma, es importante que el planificador del agente tenga en cuenta la importancia del controlador dentro del sistema multiagente y garantice que los recursos físicos del sistema estarán a su disposición de forma prioritaria. Así, como modelo general de revisión de la configuración, el controlador primeramente obtendrá la configuración actual enviando una solicitud FIPA al administrador, seguidamente se comunicará con el servicio web para obtener el resultado de todas las operaciones cerradas y posteriormente, estudiará cada operación junto con su análisis para, finalmente, analizar si la metodología o la parametrización son las adecuadas.

Como ya se ha avanzado anteriormente, queda fuera del alcance de este proyecto proponer un modelo detallado de estudio de resultados, sino que tan solo se proponen sus aspectos generales:

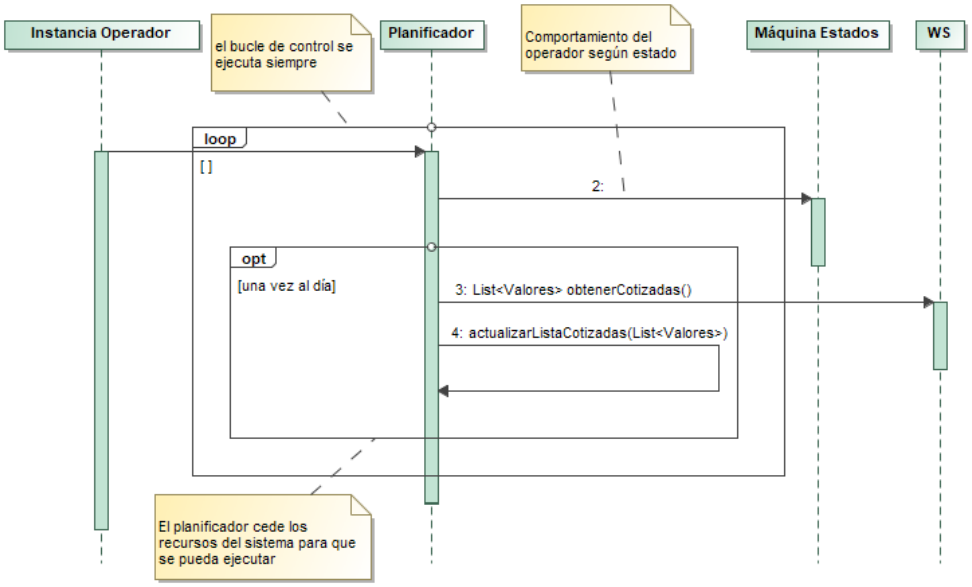
- Acceso de ámbito global a la metodología y a la parametrización del modelo
- Asincronía entre el componente Intenciones y los componentes Creencias y Deseos
- Priorización de la ejecución del componente Intenciones sobre el resto de componentes



4.2.3. Rol de agente operador

El agente operador es el encargado de poner en práctica y materializar el modelo conceptual del sistema, ya que su función es la de vigilar la evolución de las operaciones que el sistema realiza. Así, de entre todos los roles de agentes, es el que tiene una mayor proximidad a la capacidad de percepción del sistema, de tal forma que la delegación de la base de conocimientos en el rol de analista y su propia especialización, le convierten prácticamente en un agente reactivo.

En consecuencia, el agente se diseñará, principalmente, según el comportamiento de una máquina de estados, aunque, al igual que el resto de roles, también implementará a partir de un bucle de control simple los ciclos recursivos de los que tenga que hacerse cargo.



Así, siguiendo un modelo de máquinas de estado, se definirán tres estados posibles en los que el rol de agente operador podrá estar:

• Reconocimiento, en este estado el operador reconoce una oportunidad de inversión mediante la comunicación activa (pull) con el resto del sistema. Así, el agente contacta periódicamente (cada minuto) con el agente administrador para comprobar si las condiciones del mercado son las propicias para empezar a operar.

• Validación, en este estado el operador revisa la oportunidad informada y calcula los parámetros de la operación a realizar.

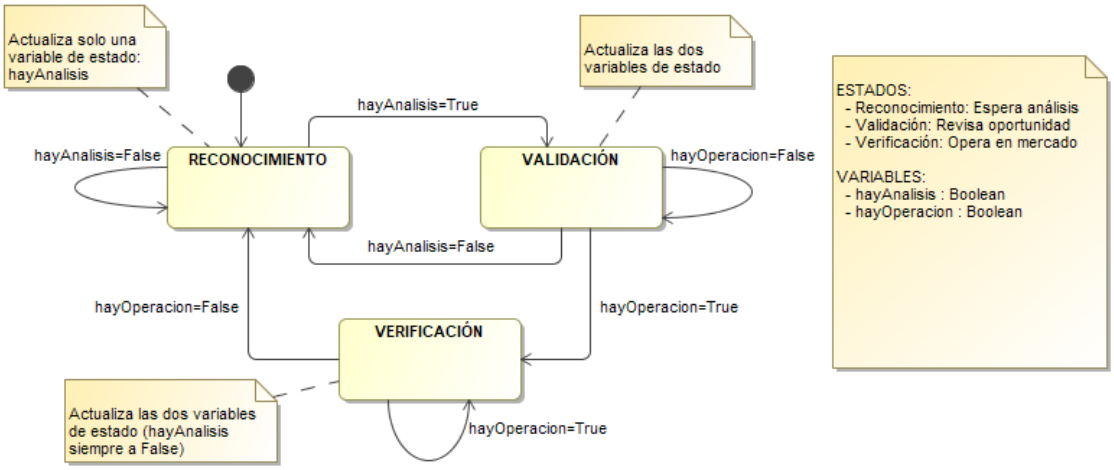
• Verificación, en este estado el operador se encuentra plenamente dentro de su fase de actividad, recogiendo la información del mercado y comparándola con los parámetros establecidos. Así, el agente interactúa con su entorno externo (mercado) y percibe los distintos cambios acontecidos.

Además, se definirán dos variables booleanas, necesarias para la transición entre los estados:

- hayAnalisis, que indica si el agente ha identificado una oportunidad de inversión
- hayOperacion, que indica si el agente ha iniciado una operación bursátil

De tal forma, se puede establecer la siguiente tabla de transiciones, en las que se enumeran los saltos de un estado a otro y se identifican las condiciones para hacer efectiva cada transición:

Transición	Condición
Reconocimiento -> Reconocimiento	hayAnalisis = False
Reconocimiento -> Validación	hayAnalisis = True
Validación -> Reconocimiento	hayAnalisis = False & hayOperacion = False
Validación -> Validación	hayAnalisis = True & hayOperacion = False
Validación -> Verificación	hayAnalisis = True & hayOperacion = True
Verificación -> Reconocimiento	hayOperacion = False
Verificación -> Verificación	hayOperacion = True



Además, al igual que el resto de agentes, el operador manejará un conjunto de listas que le permitirán manejar desde memoria la información y la parametrización del sistema. Así, el agente manejará las siguientes operaciones identificadas en los casos de uso:

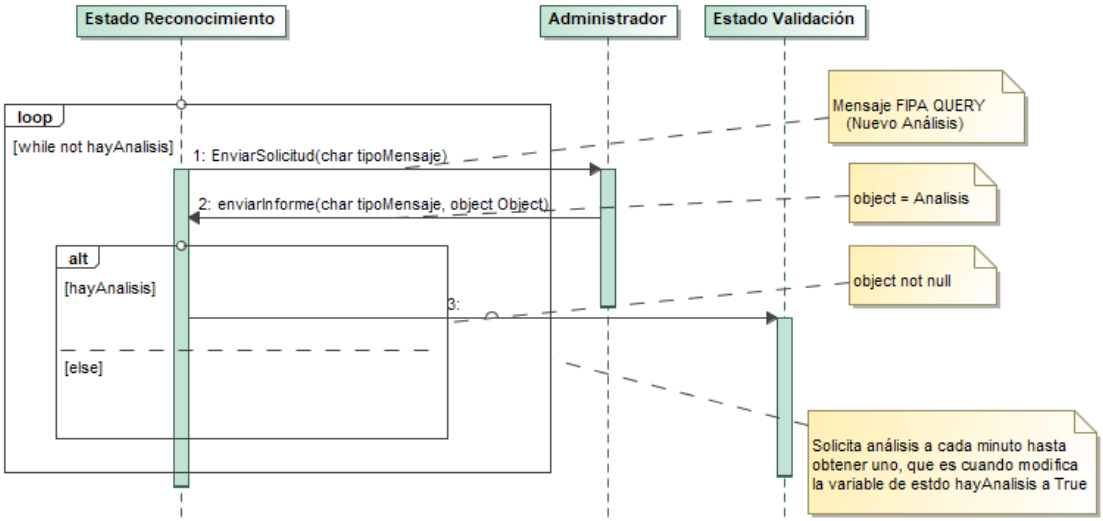
Caso Uso	Descripción	Operación
CU_02	Obtener valores del mercado	List<Valores> obtenerCotizadas(String indice)
CU_03	Obtener cotización de un valor	Actuales obtenerCotizacion(String indice, String valor)

Y las operaciones siguientes para el manejo de listas de datos en memoria:

Operación	Descripción
actualizarListaCotizadas(List<Valores>)	Manejo lista de valores de un mercado
actualizarListaOperacionesAbiertas(Operacion operacion)	Manejo lista operaciones abiertas
actualizarListaOperacionesCerradas(Operacion operacion)	Manejo lista operaciones cerradas
obtenerParametros()	Recuperación configuración común sistema
calcularOperacion()	Cálculo atributos de una operación

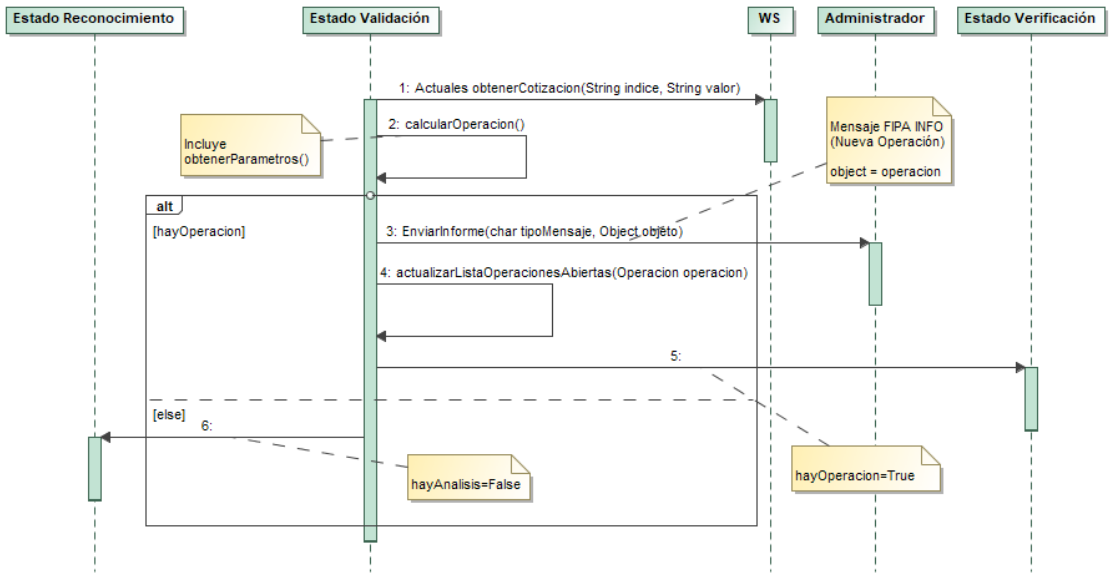
Estado de Reconocimiento

El agente comprobará cada minuto si existe alguna oportunidad de inversión. Una vez la obtenga, activará la variable *hayAnalisis* para poder evolucionar de estado.



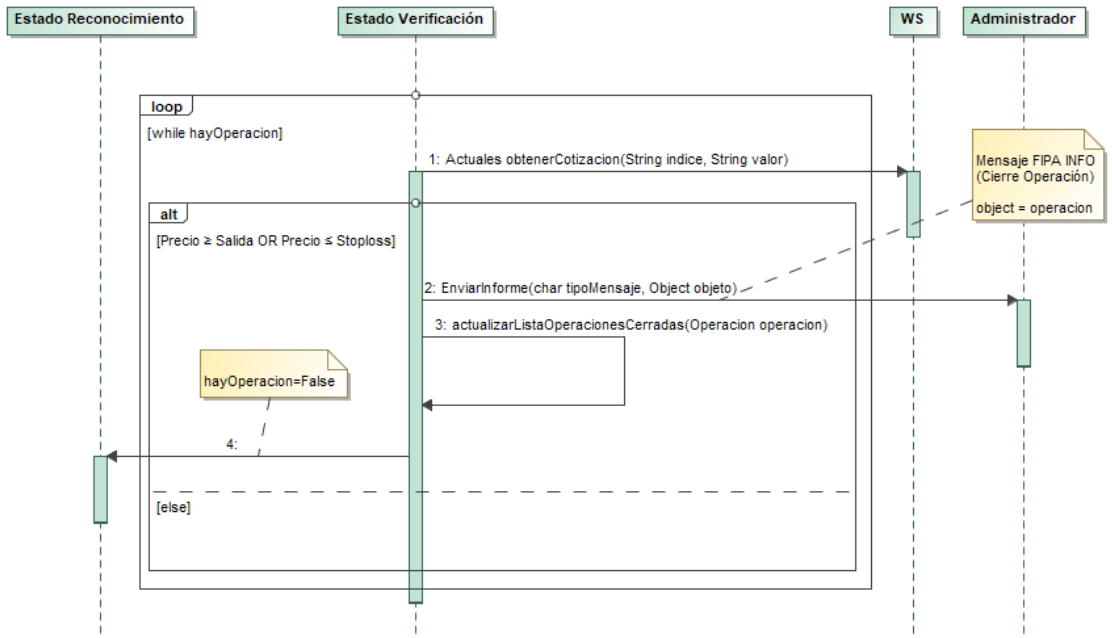
Estado de Validación

El agente revisará la oportunidad de inversión según las condiciones actuales del mercado y calculará los parámetros de la operación, activando la variable *hayOperacion* para pasar al siguiente estado o desactivando la variable *hayAnalisis* para regresar al estado anterior.



Estado de Verificación

El agente comprobará cada cinco minutos la evolución de la operación según los parámetros establecidos y una vez cierre la operación desactivará las variables *hayAnalisis* y *hayOperacion* para modificar su estado.



4.3. Síntesis del modelo de inferencia

En los apartados anteriores se definieron y establecieron las principales características de una arquitectura de software basada en agentes con el fin de modelar y diseñar un sistema multiagente con capacidades deliberativas, es decir, con capacidad para actuar de forma racional y autónoma a partir de una base de conocimientos y un motor de inferencia.

Así, primeramente se gestionó la funcionalidad del sistema hasta establecer unas características mínimas del producto y se documentaron los requisitos de esta funcionalidad mediante historias de usuario y casos de uso. Seguidamente, se identificaron los distintos roles de agentes y sus características operacionales, además se estructuró una base de conocimiento común según el modelo conceptual y lógico de la estructura de clases definida, a partir del cual, se pudieron establecer las propiedades y características de la base de datos del sistema.

Seguidamente, a partir de una serie de principios arquitectónicos y patrones de diseño reconocidos, se desacopló del diseño funcional de los propios agentes toda la estructura de persistencia de datos, y mediante el diseño de un servicio web se facilitó el acceso a la información a través del uso de un conjunto de operaciones de ejecución remota, sin estado y mediante protocolos de comunicación aceptados y reconocidos dentro del ámbito de Internet.

Finalmente, se estableció el modelo de inferencia de los agentes a través de la especialización de cada uno de los roles de agente sobre un espacio de actividad común. De tal forma, se estableció un método de actuación que afecta a la totalidad de los agentes del sistema fundamentado en la arquitectura BDI y estructurado sobre un conjunto de reglas, prioridades y parámetros que el componente intencional es capaz de administrar, modificándolo si es preciso, y trasladar al resto de componentes consiguiendo un modelo de comportamiento común.

Para ello, es importante tener en cuenta las siguientes consideraciones:

- El sistema multiagente necesita de una memoria persistente de ámbito global en la que almacenar el método de comportamiento del sistema para conseguir la cooperación entre agentes y la aparición del colectivismo y la habilidad social
- Además, el sistema necesita contemplar un modelo de prioridades entre componentes de tal forma que garantice la capacidad de enjuiciar su propio comportamiento, es decir, poner en duda el método establecido
- De tal forma, el modelo de componentes BDI necesita ser ejecutado de forma asíncrona y establecer un marco de ejecución prioritario para el componente intencional, que es capaz de poner en duda el comportamiento general del sistema y subordinar en cierta forma los componente de creencias y deseos

5. Perspectiva teórica de los costes

Por lo general, en la gestión de costes de un proyecto se incluirían todos aquellos procesos involucrados en estimar, presupuestar y controlar los gastos económicos inherentes al desarrollo y finalización del propio proyecto, sin embargo, el ámbito académico en el que se realiza este proyecto y su carácter personal (falta de grupo de trabajo) hace que la perspectiva con la que se aborde la gestión de costes sea una perspectiva teórica desde el punto de vista del modelo organizativo ideal del proyecto y desde el punto de vista del análisis marginal de costes sobre el desarrollo de un producto diferenciado y único.

Modelo organizativo del proyecto

En los últimos años, principalmente dentro del ámbito de la innovación y las TIC, ha surgido el término *startup* para describir un modelo de empresa emergente que busca montar un negocio a partir del desarrollo de un conjunto de ideas nuevas y que están enfocadas principalmente al uso de nuevas tecnologías. Así, algunas particularidades que nos podemos encontrar en estos modelos de empresa son:

- La estrategia de innovación se fundamenta en la búsqueda por la obtención de un producto distinto y mejor tecnológicamente a los ya existentes en el mercado, y en caso de poder desarrollarlo, explotarlo en exclusiva de forma monopolística.
- Por tanto, el riesgo presente en la inversión en innovación tecnológica es elevado y su tasa de fracaso alta, de tal forma, es habitual ver la presencia del capital de riesgo en forma de startups o la figura del business angel como inversionistas a fondo perdido.
- Por la relación riesgo/rentabilidad, el factor costes también es un condicionante muy presente en este tipo de industrias, ya que toda la inversión se dedica a producir el nuevo producto, y no en cubrir otro tipo de gastos de carácter largo-placista como los que nos podemos encontrar en la industria tradicional.

De tal forma, desde una perspectiva empresarial el proyecto tomaría forma de startup y todos los esfuerzos económicos estarían enfocados al desarrollo de la propia empresa (proyecto). Así, desde el punto de vista de costes habría que minorar lo máximo posible los costes fijos (costes capital) que son los relacionados con la propia estructura productiva de la empresa (por ejemplo utilizando solo software libre: GNL, LGPL, etc) e intentar gestionar de forma óptima los costes variables, producidos principalmente por las variaciones en el nivel de actividad (coste del trabajo).

Análisis marginal de costes

Una gestión de los costes que se centra en los costes variables y no en los fijos, los que mantendremos al mínimo posible mediante el uso de licencias de software libre o compra de hardware dentro del presupuesto inicial establecido, es una gestión enfocada al control de los

costes a corto plazo, por lo que el seguimiento periódico del control de los costes cobra si cabe mayor importancia en una organización de empresa como la elegida (startup).

Por otra parte, el resultado obtenido del proyecto (producto final) no es un bien tangible sino que es un nuevo software, único en el mercado, que interacciona con un mercado financiero y obtiene una serie de resultados. Supongamos ahora, que los resultados son positivos y que podría haber interés en ofrecerlo como una herramienta de inversión mediante algún modelo de suscripción de usuarios o mediante la descarga del software por un plazo limitado de tiempo.

De tal forma, definimos ingreso marginal como el incremento en los ingresos por incrementar la producción en una unidad más (vender una nueva licencia) y coste marginal como el incremento en los costes por incrementar la producción en una unidad. Cuando los ingresos por incrementar en una unidad la producción sean mayores que los costes (tecnológicos) el beneficio aumentará, es decir, cuando el ingreso marginal sea mayor al coste marginal la empresa obtendrá mayores beneficios, en caso contrario la empresa, aun produciendo más, obtendrá menores beneficios.

En consecuencia, cabría determinar el nivel óptimo de producción de la empresa desde una perspectiva de producto único, es decir, desde una perspectiva monopolística teniendo en cuenta que es posible incrementar la producción aumentando el gasto (inversión) pero no infinitamente (ley de rendimientos marginales decrecientes[2]). Así, como monopolio, el nivel óptimo de producción se establece en el momento en que el ingreso y el coste marginal se igualan, de tal forma que la empresa admitiría nuevos usuarios mientras el ingreso marginal fuese mayor al coste marginal y dejaría de aceptarlos en el momento que el coste marginal superase al ingreso marginal, situando el óptimo de usuarios cuando ingreso y coste se igualan.

6. Gestión de la calidad

En los anteriores apartados se mostraban desde varias perspectivas el desarrollo del proyecto para poder desarrollar con éxito el sistema multiagente. Así, se consideraba la gestión del proyecto desde varios punto de vista como el estudio del alcance del proyecto (requisitos), la planificación de un conjunto de actividades (ver anexo) y desde una perspectiva teórica de los costes.

Además de los anteriores puntos de vista, es necesario establecer una visión del proyecto desde una perspectiva de gestión de calidad, es decir, desde una visión que nos permita establecer si el producto desarrollado responde a las expectativas planteadas por los interesados y recogidas en el conjunto de requisitos técnicos y funcionales.

6.1. Perspectivas de la calidad

Así, la RAE define calidad como "*la propiedad o conjunto de propiedades inherentes a algo, que permite juzgar su valor*", de tal forma, enfocaremos primeramente la calidad del software desde el punto de vista de su fabricación, es decir, desde el cumplimiento de los estándares del proceso de desarrollo, y desde el punto de vista del producto, es decir, desde el conjunto de las características internas del software que permitan establecer su valía estructural.

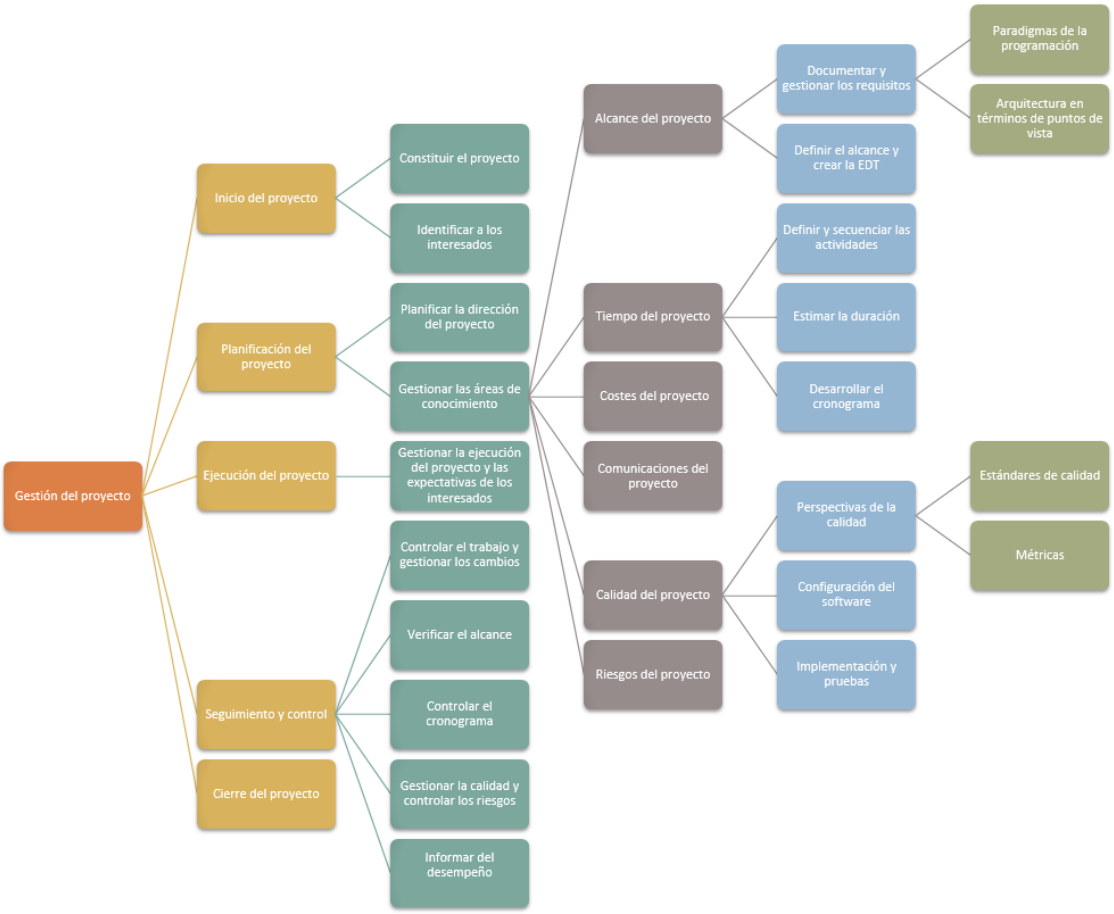
Punto de vista de la fabricación

En el proceso de desarrollo del software se han seguido las especificaciones y consejos que el Project Management Institute (PMI) publica en su guía PMBOK, en la que se recogen un conjunto de buenas prácticas para lograr una gestión del proyecto eficaz.

De tal forma, la gestión del proyecto se realiza en cinco fases interactivas diferenciadas (inicio, planificación, ejecución, seguimiento y cierre) mediante el desarrollo de un conjunto de áreas del

(2) La ley de rendimientos marginales decrecientes establece una disminución en el incremento marginal de producción ocasionado por un incremento en los factores de producción

conocimiento y procesos, que a modo de buenas prácticas, gozan del reconocimiento internacional como modelo de gestión y dirección para el desarrollo de un proyecto.



Punto de vista del producto

Por otra parte, el diseño del software ha seguido un conjunto de paradigmas (principalmente AOP y SOA) y de modelos arquitectónicos (como BDI y FIPA) que nos han permitido identificar el conjunto de módulos del sistema, la funcionalidad y responsabilidades de cada módulo, así como todas las interacciones posibles. Además, a partir del estándar de diseño RM-ODP se han establecido un conjunto de normas para describir de forma correcta y ordenada la arquitectura del software a desarrollar, de tal forma, sus especificaciones se han descrito a partir de un conjunto de puntos de vista que permitían concentrarse en aspectos concretos del sistema (norma IEEE 1471) y ya desde una perspectiva de acceso a los recursos, nos hemos apoyado en un estilo arquitectónico basado en un modelo cliente-servidor (REST) y en el uso de estándares de comunicaciones de Internet (HTTP, URI, MIME).

Y finalmente, se han diseñado el conjunto de clases a partir de una serie de principios y patrones de diseño contrastados con el fin de estructurar el modelo desde una perspectiva de bajo acoplamiento, de alto grado de abstracción y de fácil reutilización. Así, principalmente se han empleado los siguientes patrones de diseño:

- Arquitectura en capas, para el diseño estructurado y con cierto grado de abstracción
- Patrón estado, para el diseño de clases con un bajo grado de dependencia entre ellas
- Patrón estrategia, para evitar información duplicada y ambigüedades
- Patrón fachada, para el diseño con bajo grado de acoplamiento
- Patrón experto, para la correcta asignación de responsabilidades

Análisis de la calidad desde una perspectiva AOP

Aunque es fácilmente aceptable una definición de la calidad del software como la empleada por Roger Pressman, en donde se enuncia como "la conformidad con los requisitos funcionales y de rendimiento, con los estándares de desarrollo prefijados y con las características implícitas que se esperan de cualquier software profesional desarrollado", es necesario comentar algunos aspectos de la definición respecto a la gestión de la calidad enfocada hacia el desarrollo del software sobre el paradigma AOP y sus características.

Así, el paradigma de programación de agentes se suele usar para el desarrollo de software complejo, con componentes autónomos que interactúan y se organizan de forma dinámica, y que cooperan entre ellos mostrando cierto grado de inteligencia y habilidad social, lo que hace que exista, entre las expectativas de sus requerimientos, un cierto punto de incertidumbre en los resultados que se van a obtener.

De tal forma, las propias especificaciones del paradigma y su especialización en un comportamiento emergente, hace que muchas de las situaciones en las que se encuentran los agentes no se hayan previsto en las fases de diseño del software y se resuelvan en tiempo de ejecución. Esta particularidad lleva a matizar el significado de *característica implícita esperada del software*, ya que debido a la flexibilidad y adaptación mostrada por los agentes, la gestión de la calidad debe enfocarse hacia la evaluación del comportamiento (proactividad) y no de los resultados, es decir, hacia las actitudes relacionadas con el comportamiento de los agentes, así como al control de ese comportamiento.

Defectos, errores, fallos e incidencias

Siguiendo con la definición de calidad de la RAE, es necesario precisar una serie de juicios de valor que nos permitan cuestionar la calidad del software. Así definimos los siguientes términos como anomalías no deseadas en el resultado final del software:

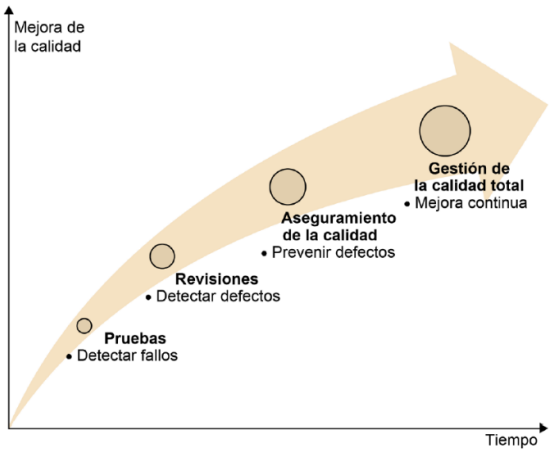
- Defecto: anomalía del comportamiento del software respecto a su especificación
- Error: equivocación que produce defectos en el proceso de desarrollo del software
- Fallo: consecuencia de un defecto que produce resultados inesperados
- Incidencia: cualquier situación que difiera del comportamiento esperado y provoque mantenimientos (correctivos del software)

Así, la gestión de la calidad del software se centrará, desde un punto de vista del producto, en detectar y corregir los defectos, y desde un punto de vista de la fabricación (proceso) en revisar la metodología empleada, ya sea en la estructuración arquitectónica del diseño del software, ya sea en el seguimiento de los hitos y objetivos marcados en la gestión del proyecto.

Por otra parte, vemos que en la terminología anterior también cabe matizar desde una perspectiva AOP lo que es un fallo o una incidencia. Así, el modelo arquitectónico BDI que se estructura en tres elementos básicos: creencias, deseos e intenciones, se fundamenta, como ya se ha notificado, en el conjunto de actitudes funcionales y en el control del comportamiento (pro-actividad) de los agentes, de tal forma, los resultados inesperados deben enfocarse a anomalías propias de ese comportamiento, es decir, a fallos en la capacidad de los agentes para actuar y desplegar sus habilidades, pero no hacia al signo de sus resultados, ya que la capacidad de evolucionar a partir de unos resultados negativos es una de las características inherentes al paradigma AOP.

A tal respecto, notificar que el término *evolucionar* se emplea partiendo de la idea de que el problema que gestionan los agentes tiene solución, aunque no siempre es el caso. Así, en el ámbito de la inversión bursátil, hasta el momento, no existe ningún método que garantice el éxito, es

decir, una de las cuestiones a tener en cuenta como resultado de la actividad de los agentes es que en ningún caso son capaces de conseguir resultados positivos.



Así, gestionaremos la calidad del software desarrollado mediante un conjunto de técnicas o buenas prácticas, que estarán enfocadas principalmente a la verificación y validación del comportamiento de los agentes (análisis cualitativo del producto), a la mejora de los procesos de desarrollo (análisis cualitativo del proceso) y a la medición a partir de un conjunto de métricas del funcionamiento desarrollado (análisis cuantitativo del producto).

De tal forma, es fundamental detectar los fallos y los defectos del producto mediante las una fase de pruebas y revisiones del software para comprobar que se cumplen tanto con sus especificaciones técnicas como con las expectativas de su desarrollo. Ambas técnicas se verán en fases más avanzadas del proyecto, cuando el software este desarrollado e implementado, de momento nos centraremos en establecer algunos de los parámetros que nos van a permitir asegurar la calidad del software y en consecuencia prevenir sus defectos.

6.2. Medidas y métricas de la calidad

Entre los objetivos principales del aseguramiento de la calidad del software (SQA) está la planificación de sus actividades de desarrollo, la verificación de estas actividades mediante estándares y procedimientos establecidos (ver perspectivas de la calidad desde los punto de vista de la fabricación y del producto), el establecimiento de un conjunto de métricas sobre las cuales medir distintos atributos del software y la gestión de la configuración del propio software que permitirá asegurar su calidad a través del control de los cambios y del control de versiones.

Así, como modelo de calidad sobre el que realizar un conjunto de evaluaciones y mediciones, tomaremos como referencia el estándar ISO 9126 en los que se definen seis factores que nos van a permitir enfocar tanto el análisis cualitativo de calidad mediante las técnicas de prueba y revisión del software, como el análisis cuantitativo a partir de un conjunto de métricas.

De tal forma, de los seis factores de calidad que presenta la ISO 9126, por las particularidades del sistema multiagente desarrollado, plantearemos en este y posteriores apartados un análisis cualitativo y cuantitativo de los factores de funcionalidad, fiabilidad, eficiencia y facilidad de mantenimiento del software. Para ello, mediante el software de análisis de estructuras para Java

STAN y el software de evaluación de código SonarQube estableceremos un conjunto de métricas que nos permitirán analizar a nivel de proceso y de producto la calidad alcanzada por el software:

6.2.1. Métricas generales

El tamaño de un software puede definirse según el número de sus componentes y de sus líneas de código, de tal forma que cuanto más componentes y líneas de código haya, en principio, más complejo será el software. Así, recogeremos cinco medidas que hacen referencia a estas métricas básicas de carácter general:

- Número de paquetes:

Medirá el número de paquetes contenidos en la aplicación.

- Número de unidades o clases de nivel superior:

Medirá el número de clases de nivel superior de la aplicación, que son clases de Java, interfaces o enumeraciones, que no están contenidas en otra clase. Reciben también el nombre de unidades, ya que generalmente corresponden a unidades de compilación.

- Número de clases miembros de otra clase:

Medirá la cantidad de clases miembro contenidas en otra clase.

- Número de métodos:

El número de métodos de una clase.

- Número de campos:

El número de campos de una clase.

- Número estimado de líneas de código (ELOC):

Aproximación de número de líneas de código fuente sin tener en cuenta las líneas de comentarios o las líneas vacías. La aproximación asume algunas reglas comunes del estilo de codificación de Java y es más significativa que un valor exacto, ya que no depende del estilo de programación de un desarrollador concreto, lo que hace que los valores sean menos comparables.

6.2.2. Métricas de calidad del proceso

Además, evaluaremos el progreso y la cobertura de las pruebas realizadas desde una perspectiva de producto mínimo y del conjunto de componentes del sistema. Para ello, graficaremos la curva de progreso de las pruebas y calcularemos la tasa de cobertura de las pruebas:

- Curva de progreso de las pruebas:

Graficaremos el progreso de las pruebas realizadas a los siguientes componentes del sistema: WSAnalisis, WSOperaciones, WSCotizaciones, WSValores, FIPA, Creencias, Deseos, Intenciones, Operador, Configuración según el funcionamiento establecido en la definición del producto mínimo.

- Métricas de cobertura de pruebas:

Mediremos la cantidad de elementos que se han probado respecto al total de elementos existentes. Para ello utilizaremos como referencia los mismos componentes que los definidos para graficar la curva de progreso de las pruebas.

$$Cobertura\ pruebas = \frac{Nº\ elementos\ cubiertos}{Nº\ total\ elementos} * 100$$

6.2.3. Métricas de calidad del producto

Para medir la calidad del producto, primeramente, utilizaremos una métrica de complejidad que nos indicará el grado de dificultar para entender y verificar los distintos componentes del sistema.

Así, la métrica más conocida y usada para medir la complejidad es la llamada complejidad ciclomática o complejidad de McCabe, que mide el número de caminos independientes existentes en un programa mediante una representación en grafo y se calcula a partir del número de arcos y nodos del grafo obtenido:

- Complejidad Ciclomática (CC):

Mide el número de puntos de decisión en el gráfico de flujo de control del método incrementado en uno. Los puntos de decisión se producen en las sentencias if/for/while, cláusulas case/catch y elementos de código fuente similares, donde el flujo de control no es solo lineal.

Además, utilizaremos algunas métricas específicas de la programación orientada a objetos que nos permitirán complementar las mediciones y así poder evaluar la calidad del producto. De tal forma, revisaremos las seis métricas que proponen Chidamber y Kemerer para clases e interfaces:

- Métodos ponderados por clase (WMC):

Se mide mediante la suma de la métrica de Complejidad Ciclomática sobre todos los métodos de una clase.

- Profundidad del árbol de herencia (DIT):

Se mide a partir del árbol de herencias de una clase y viene determinado por la longitud de la ruta desde la propia clase hasta la raíz (java.lang.Object), sin tener en cuenta las implementaciones de interfaces, ya que no forman parte del árbol de herencia. Si se quisiera medir la profundidad de una interface, se mediría de forma parecida: la longitud de la ruta más larga desde la propia interface a su interface raíz.

- Número de descendientes (children) (NOC):

El número de hijos de una clase es el número de subclases que extienden la clase. El número de hijos de una interface es el número de interfaces derivadas que extienden la interface.

- Acoplamiento entre objetos (CBO):

Mide el acoplamiento entre clases teniendo en cuenta que dos clases están acopladas si se llaman métodos de una de las clases a métodos de la otra clase sin haber una relación de herencia entre las dos clases.

- Responsabilidad de una clase (RFC):

Mide el número de métodos que se llaman desde una clase, dentro y fuera de la propia clase (cada método se cuenta solo una vez, incluso si se llama varias veces).

- Carencia de cohesión en los métodos (LCOM):

Mide la falta de cohesión entre los métodos de una clase teniendo en cuenta que dos métodos de una clase están relacionados, si acceden a un campo común. De tal forma, la falta de cohesión se calcula como el número de pares de métodos que no están relacionados menos el número de pares de métodos que están relacionados (0 si el resultado es negativo).

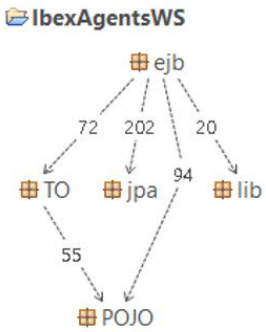
6.3. Gestión de configuración del software

Las actividades de la configuración del proyecto, es decir, de sus características funcionales y físicas, se han centrado principalmente en estructurar, identificar y definir los elementos del producto de forma jerárquica, ya que actividades como la gestión de versiones y el control de cambios se han simplificado notablemente debido a que el proyecto ha sido diseñado y elaborado tan solo por un único desarrollador, de tal forma, estas actividades han sido desarrolladas principalmente desde una perspectiva de copia respaldo (backup).

Así, la principal tarea de control de modificaciones en los elementos de la configuración del software consiste en numerar la nueva versión del proyecto y en subirla a un servidor de respaldo que en este caso ha sido el servidor OneDrive de Microsoft. De tal forma, este apartado del proyecto se centrará principalmente en desarrollar la estructura de la jerarquía de los elementos del software compuesto por el servicio web y el modelo de agentes diseñados con JADE.

Estructura del web service

El servicio web, de nombre IbexAgentsWS, está compuesto por cinco paquetes de clases:



- Paquete ejb, en donde se localizarán todas las clases Enterprise JavaBeans que dan funcionalidad al WS mediante la implementación del estándar de programación JEE de Oracle. De tal forma, las clases de este paquete tendrán una relación directa con las reglas de negocio.
- Paquete jpa, en donde se localizarán todas las entidades JPA (Java Persitence API) que están relacionadas con el acceso a la BBDD, es decir, en este paquete se encontrar las clases relacionadas de forma directa con la persistencia de los datos.
- Paquete TO, en donde se localizarán aquellas entidades TO (Transfer Object) relacionadas con la encapsulación y el transporte de los datos en formato XML.
- Paquete POJO, en donde se localizarán aquellas clases que para una mejor estructuración de los datos, bien sea para conseguir un menor grado de acoplamiento, bien sea para evitar información duplicada o ambigua, apoyan a los paquetes ejb y TO definiendo nuevas clases POJO (Plain Old Java Objects).
- Paquete lib, de forma parecida al paquete POJO este paquete está compuesto por entidades de auxilio (utilidades) para una mejor estructuración del funcionamiento del sistema.

<u>Paquete ejb</u>

Este paquete está compuesto por los EJB Session sin estado de los distintos componentes definidos en el web service:

• *AnalisisFacadeBean y su interface remota AnalisisFacadeRemote que implementa los métodos obtenerAnalisis y guardarAnalisis.*

• *CotizacionesFacadeBean y su interface remota CotizacionesFacadeRemote que implementa los métodos actualizarCotizaciones y actualizarCotizacionesHistoricas.*

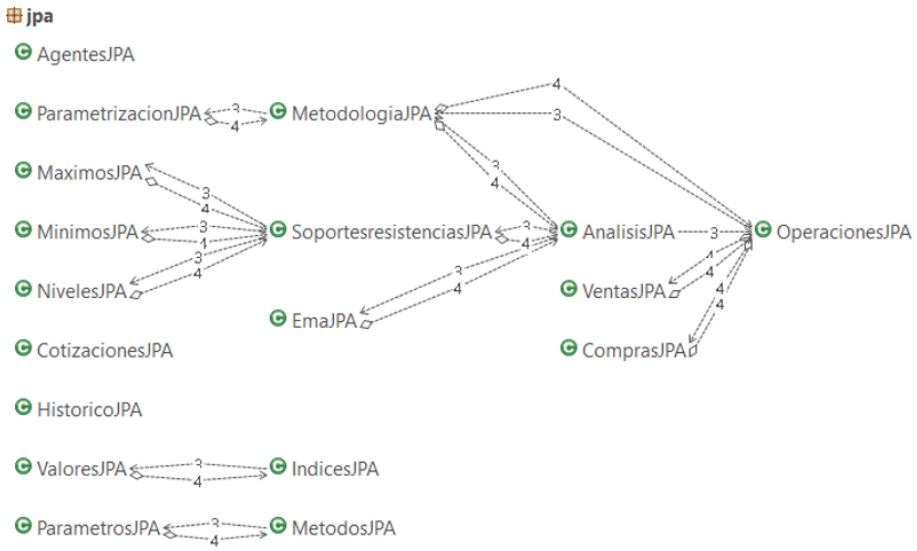
• *MetodosFacadeBean y su interface remota MetodosFacadeRemote que implementa el método obtenerConfiguracion.*

• *OperacionesFacadeBean y su interface remota OperacionesFacadeRemote que implementa los métodos guardarCompra, obtenerOperacionesCerradas y obtenerOperacionesAbiertas.*

• *ValoresFacadeBean y su interface remota ValoresFacadeRemote que implementa los métodos obtenerCotizadas y obtenerCotizadasAdmin*

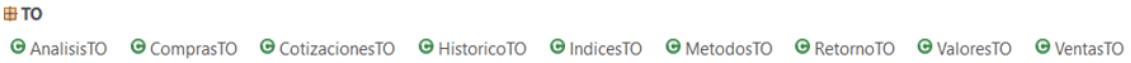
Paquete jpa

Está compuesto por cada una de las entidades JPA necesarias para el almacenamiento de los datos:



Paquete TO

Está compuesto por cada una de las entidades TO necesarias para la transferencia de datos en XML con el sistema de agentes. Estos datos podrán ser de entrada para su almacenamiento mediante el método POST o de salida para su lectura mediante el método GET.



Paquete POJO

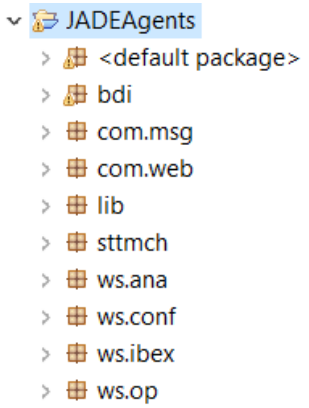
Está compuesto por un conjunto de clases que simplifican el manejo de los datos y su transferencia mediante clases TO.

Paquete lib

Está compuesto por una única clase *Utils* que simplifican el procesamiento de los datos, ya que agrupa un conjunto de métodos de acceso común por los distintos componentes del servicio web. Algunos de los métodos que implementa son: getFecha, getHora, getHtml, getHtmlData, etc…

Estructura del sistema de agentes

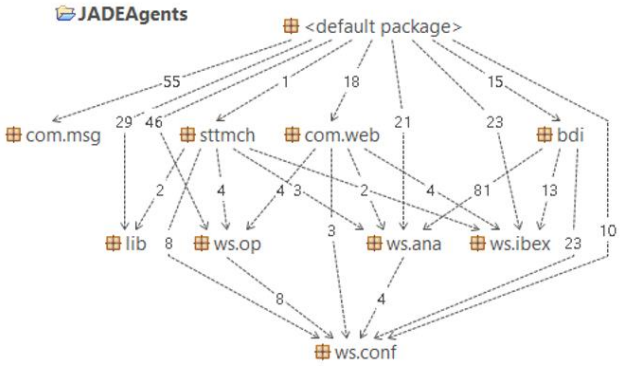
El sistema multiagente, de nombre JADEAgents, está compuesto por seis paquetes principales de clases, dos de los cuales están compuestos a su vez por dos y cuatro paquetes respectivamente, lo que hace un total de diez paquetes.



- Paquete por defecto, en donde se localizarán las clases que dan soporte a los tres tipos de agentes: Admnistradores, Analistas y Operadores. Estas clases estarán compuestas principalmente por varias subclases, cada una de ellas correspondiente a un proceso de carácter repetitivo.
- Paquete bdi, en donde se localizarán todas las clases relacionadas con el modelo BDI (Beliefs, desires and intentions) con el que se estructura el modelo de agente deliberativo (agentes con rol de Analista).

- Paquete sttmch, en donde se localizarán las clases que dan soporte estructural a los agentes diseñados sobre máquinas de estados (agentes con rol de Operadores).
- Paquete com.*, en donde se localizarán todas aquellas clases relacionadas con las comunicaciones y que estará compuesto por el paquete com.msg, en donde se localizan las clases auxiliares para la comunicación por mensajes FIPA, y el paquete com.web, en donde se localizan las clases que permite la comunicación web con el WS.
- Paquete ws.*, en donde se localizarán todas las clases relacionadas con los datos a intercambiar con el web service. Así, este paquete estará compuesto por cuatro paquetes: ws.ana que replica la información para las comunicaciones con el componente remoto AnalsisiWS; ws.conf, que replica la información para comunicarse con el componente MetodosWS; ws.op que contiene la réplica para las comunicaciones con OperacionesWS; y ws.ibex que contiene la información para poder comunicarse con los componentes remotos ValoresWS y CotizacionesWS.
- Paquete lib, compuesto por un conjunto de utilidades del tipo getFecha, getHora, etc..

El modelo relacional entre los distintos paquetes se muestra en el siguiente grafo, en donde se puede observar que el paquete por defecto (los agentes) está relacionado con todas las entidades y que el uso del web service y de una configuración común también son rasgos característicos del sistema multiagente diseñado:

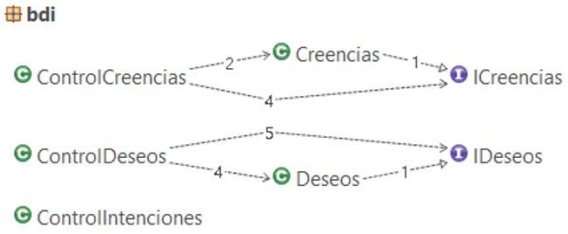


Paquete por defecto

Está compuesto por cada una de los roles que forma el conjunto de agentes JADE: Administrador, Analista y Operador

Paquete bdi

Está compuesto por el conjunto de entidades que permiten instanciar el paradigma BDI teniendo en cuenta que se ha aplicado el patrón de diseño Experto para su implementación. Así, cada una de las unidades del paradigma se gestionará mediante una clase controladora (driver):



Paquete sttmch

Está compuesto por una única clase que contiene métodos auxiliares a los distintos estados en el que se puede encontrar el agente Operador.

Paquete com.*

Está compuesto por todas aquellas entidades que permiten implementar el modelo de comunicaciones entre agentes mediante mensajes FIPA (clase auxiliar ya que FIPA está integrado en el framework de JADE) y el modelo de comunicaciones REST sobre HTTP con el WS.

Paquete ws.*

Está compuesto por un conjunto de réplicas de todas aquellas clases TO y POJO necesarias para el intercambio de datos con entre los distintos agentes y el servicio web. De tal forma, el sistema multiagente (principalmente el rol de agente Administrador) puede invocar los métodos remotos del servicio web tanto para recibir información como para enviarla para ser almacenada.

Paquete lib

Está compuesto por una única clase *Utils* que simplifican el procesamiento de los datos, ya que agrupa un conjunto de métodos de utilidad general.

Roles de los agentes

Rol de Agente Administrador

El rol de administrador se estructura principalmente sobre un modelo de bucle de repetición simple para las comunicaciones con el servicio web mediante los procesos *wsManagement* y *wsHistoricalManagement*. El primero, se comunicará cada cinco minutos con el WS para actualizar las cotizaciones de los valores, y el segundo, se comunicará una vez al día para actualizar las cotizaciones históricas una vez haya cerrado el mercado.

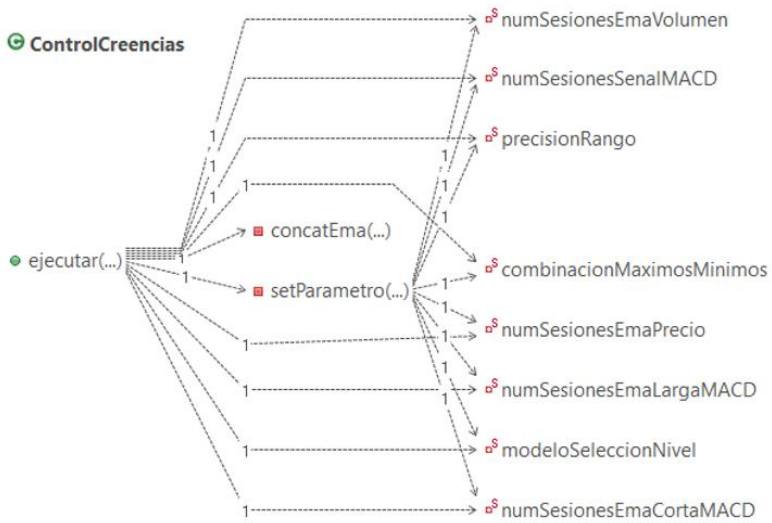
Por otra parte, también se estructura sobre un modelo de gestión de eventos para la gestión de las comunicaciones entre agentes. Así, mediante el proceso *messageManagement* se revisará las distintas solicitudes FIPA que le hagan el resto de agentes:

Rol de Agente Analista

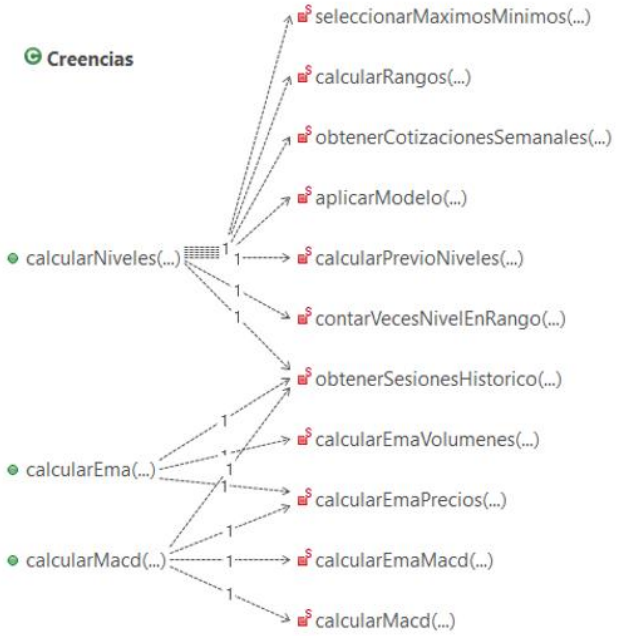
El rol de analista se estructura principalmente bajo la implementación del paradigma BDI y el patrón de diseño Experto, de tal forma, que el agente estará constituido por los tres componentes BDI (Creencias, Deseos e Intenciones) y por tres controladores expertos (drivers) de cada uno de los componentes, que implementarán un único método público (*ejecutar*()) para poder ser invocados por el agente.

Así, el agente implementa dos procesos principales: *BDStateMachine* para dar soporte a los componentes de Creencias y Deseos que se ejecutarán cada diez minutos de forma secuencial, e *IStateMachine* que se ejecutará cada minuto para dar soporte al componente de Intenciones. Además, también se ejecutarán los procesos: *messageManagement*, que se ejecutará una única vez cuando arranque el agente para obtener, mediante solicitud FIPA al Administrador, la lista de valores y la parametrización inicial, y *updateIbexStock*, que se ejecutará una vez volviendo a consultar al Administrador si hay cambios en la lista de valores cotizados.

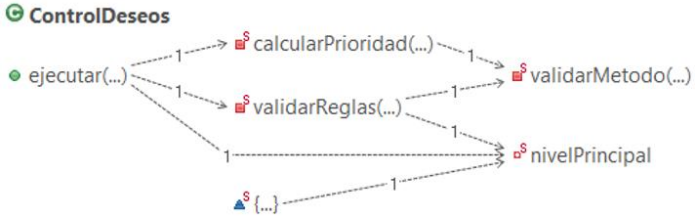
Así, una vez el agente sea capaz de obtener la configuración inicial, empezará a actuar según el modelo BDI invocando al controlador de Creencias, que a su vez invocará al componente de creencias BDI mediante su interface pública.



De tal forma, los métodos principales para el cálculo de indicadores técnicos estarán localizados en este componente:



Y una vez invocada la clase experta en Creencias, en caso de que los cálculos hayan obtenido los valores deseados, se invocará al controlador de Deseos para aplicar el método vigente:



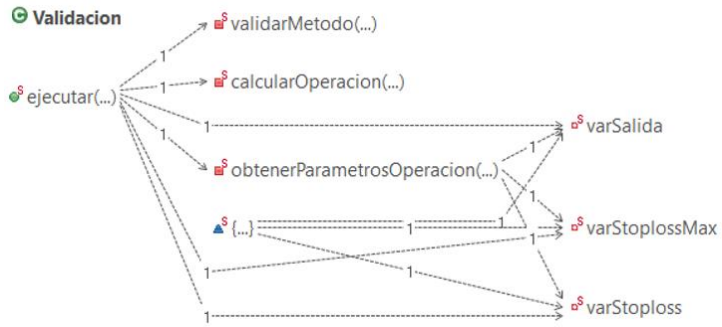
Por otra parte, el agente ejecutará de forma preferente el proceso *IStateMachine* para invocar a la clase experta en Intenciones con el fin de revisar la configuración general del sistema a partir de los datos obtenidos. Como se ha mencionado varias veces, queda fuera del alcance de este proyecto implementar un modelo concreto de revisión del método deliberativo del sistema multiagente, por lo que tan solo se ha programado un controlador experto con un único método que permite aproximarse a modo de ejemplo a la funcionalidad deseada: *nuevaConfiguracion()*

Rol de Agente Operador

El rol de operador se estructura principalmente bajo la implementación de una máquina de estados con tres estados posibles (además del inicial): Reconocimiento, Validación y Verificación recogidos en el proceso *stateMachine* que se ejecuta de forma continua.

Además, el agente implementa dos procesos más: *getInitialConfiguration*, que se ejecuta una única vez para obtener la configuración inicial del sistema (métodos para operar) y *updateConfiguration*, que se ejecuta cada diez minutos para recoger los cambios realizados por el agente analista en la metodología. Tanto la configuración inicial como sus cambios se obtienen mediante la comunicación por mensajes FIPA con el Administrador.

Por otra parte, con el fin de mejorar la estructuración y el grado de desacoplamiento de la máquina de estados, se añade una clase experta del estado de *Validación*, que es el estado en el que el agente revisa la oportunidad de inversión recibida (análisis) y comprueba si efectivamente puede efectuar una operación de compra. En caso afirmativo, la efectuará y pasará al estado de *Verificación* hasta que se cumpla algún requisito en los cálculos efectuados (precios de salida). En caso negativo, desechará el análisis recibido y retrocederá al estado anterior, *Reconocimiento*, en el mantendrá una comunicación constante con el Administrador hasta que se le proporcione un nuevo análisis.



7. Implementación y despliegue

Como ya se ha comentado con anterioridad, se ha construido el proyecto desde una perspectiva cliente/servidor donde el servicio web es accesible desde tres posibles clientes que no necesitarán ningún tipo de software especial para comunicarse con el servidor, ya que lo harán mediante el protocolo de comunicaciones HTTP e intercambiarán los datos en formato XML. De tal forma, será posible acceder al servicio web bien desde la plataforma multiagente desarrollada sobre JADE, bien desde el cliente Java para pruebas del servicio web, o bien desde cualquier navegador web que invoque al servicio para hacer consultas.

Así, siguiendo el paradigma de programación REST, se han habilitado las siguientes direcciones para que los distintos clientes puedan hacer uso del servicio:

Componente WSVALORES

Se podrán obtener el conjunto de valores o empresas cotizadas en el Mercado Continuo español invocando alguna de las direcciones web siguientes, de tal forma que cuando invoca el rol de agente Administrador, el WS se comunicará con el servicio externo (www.invertia.com) para actualizar la información existente, en caso contrario, tan solo recuperará la información de la BBDD:

http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerCotizadas/IBEX
http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerCotizadasAdmin/IBEX

Componente WSCOTIZACIONES

Se podrán obtener las cotizaciones, actuales e históricas, del Mercado Continuo español para cada una de las empresas cotizadas invocando alguna de las direcciones web siguientes. De tal forma, los métodos de consulta recuperarán la información ya existente en la BBDD y los métodos de actualización, que serán invocados por el rol de agente Administrador, provocarán que el WS se comunique con el servicio externo y actualice los datos existentes en la BBDD:

http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerCotizacion/IBEX/ACS
http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerCotizacionHistorica/IBEX/ACS
http://localhost:8080/IbexAgentsWS/AgentRestWS/actualizarCotizaciones
http://localhost:8080/IbexAgentsWS/AgentRestWS/actualizarCotizacionHistorica;indice=IBEX;valor=ACS;referencia=RV011ACS

(*) Se ha utilizado el código de empresa ACS como ejemplo de consulta. Invocando los métodos del componente WSValores es posible obtener los datos de las empresas

Componente WSOPERACIONES

Se podrán obtener y consultar las operaciones realizadas por los distintos agentes de rol Operador invocando al servicio web tanto para obtener las compras de valores efectuadas, como las ventas de los valores comprados con anterioridad. En caso de que una operación esté ya cerrada (venta) esta operación se dejará de presentar en la lista de operaciones abiertas (compras):

http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerOperacionesAbiertas
http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerOperacionesCerradas

Además, tanto desde la plataforma de agentes como del programa de pruebas Java se podrán invocar mediante el método POST del protocolo de comunicaciones HTTP las operaciones de almacenamiento de las distintas operaciones realizadas:

http://localhost:8080/IbexAgentsWS/AgentRestWS/guardarCompra
http://localhost:8080/IbexAgentsWS/AgentRestWS/guardarVenta

Componente WSANALISIS

Se podrán obtener y consultar los análisis realizados por el rol de agente Analista y seleccionados como oportunidades de inversión con independencia de que se haya operado con ellos (el análisis puede ser rechazado por el agente Operador). Para ellos se podrá invocar la siguiente dirección web:

http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerAnalisis
Además, tanto la plataforma de agentes como el programa de pruebas podrán invocar al WS mediante el método POST para el almacenamiento de estos análisis:

http://localhost:8080/IbexAgentsWS/AgentRestWS/guardarAnalisis

Metodología y Parametrización

Por último, dando respuesta a las necesidades de la plataforma de agentes para emplear una metodología común, se ha preferido almacenar en la BBDD su parametrización. De tal forma, será posible obtener esta configuración desde la dirección web:

http://localhost:8080/IbexAgentsWS/AgentRestWS/obtenerConfiguracion

7.1. Implementación del proyecto

Para la correcta implementación del proyecto, habrá que tener en cuenta que el software (web service y sistema multiagente) ha sido desarrollado con la siguiente tecnología:

Lenguaje de programación

Se ha utilizado el kit de software de Java *JDK 1.8 versión 1.8.0-121* para el desarrollo de la programación tanto del servicio web como de los agentes. Es muy importante no utilizar una versión inferior a Java8 ya que el código ha sido optimizado para reducir su complejidad mediante el uso de expresiones lambda y API Stream en el manejo de colecciones de datos.

Además, será necesario para el manejo y uso de la plataforma de agentes el framework *JADE 4.5.0* para Java. Se puede consultar entre los anexos de esta PAC los detalles para su obtención, instalación e integración con el resto de tecnologías del sistema.

Gestor de BBDD

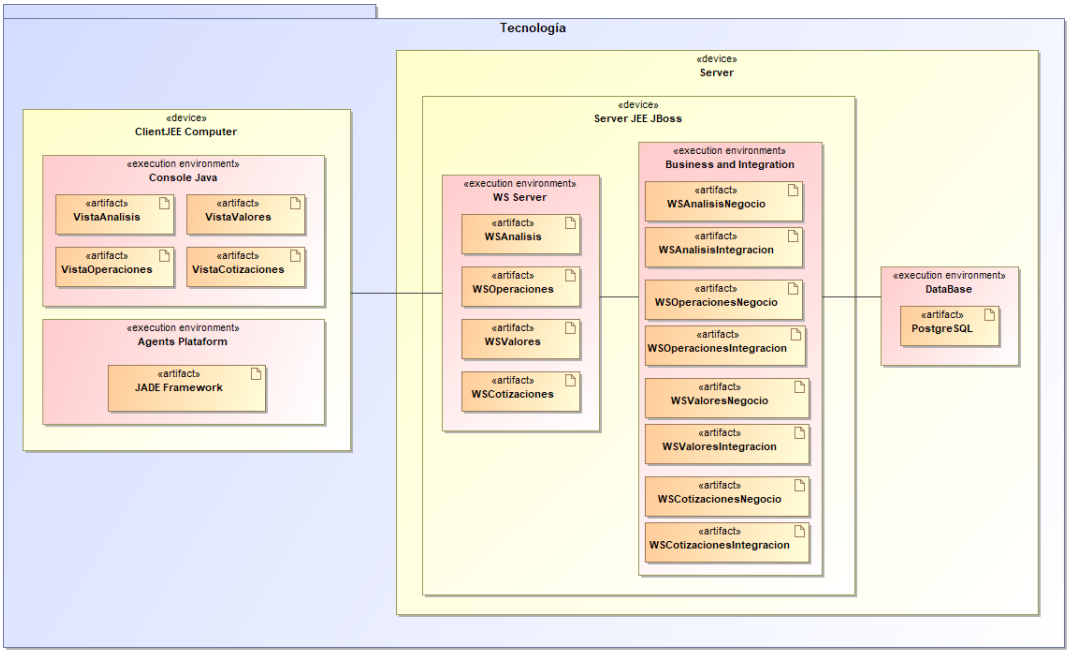
Como sistema gestor de base de datos relacional se ha utilizado el software de sistema abierto PostgreSQL versión 9.6.2-2. Al final de esta PAC se podrá consultar como anexo toda la configuración de la BBDD así como un conjunto de scripts que van a facilitar la creación de sus tablas, junto a la importación/exportación y el mantenimiento de los datos.

Servidor de aplicaciones

Para poder implementar el servicio web necesitamos un servidor de aplicaciones. De tal forma, utlizaremos *WildFly Aplication Server, versión 10.1.0.Final* (antiguamente JBoss) que es un servidor de apliaciones Java EE de código abierto y el conector *Connector Java Postgresql-9.4.1209* para que se pueda comunicar con el gestor de BBDD PostgreSQL.

Entorno de desarrollo

Como entorno de desarrollo se ha empleado el IDE *Eclipse, versión Neon 4.6* y la herramienta de compilación y construcción de paquetes *Apache Ant, versión 1.10.1*. Además, se ha instalado el plugin para Eclipse *JBoss Tools 4.4.3.Final* que facilitará la programación del WS.



7.2. Despliegue del proyecto

En términos generales, toda la configuración del software anterior se ha realizado según las especificaciones descritas en el manual de configuración de las asignaturas *Ingeniería del software de componentes y sistemas distribuidos* y *Proyecto de desarrollo de software*, teniendo en cuenta que el sistema se ha construido en los siguientes términos:

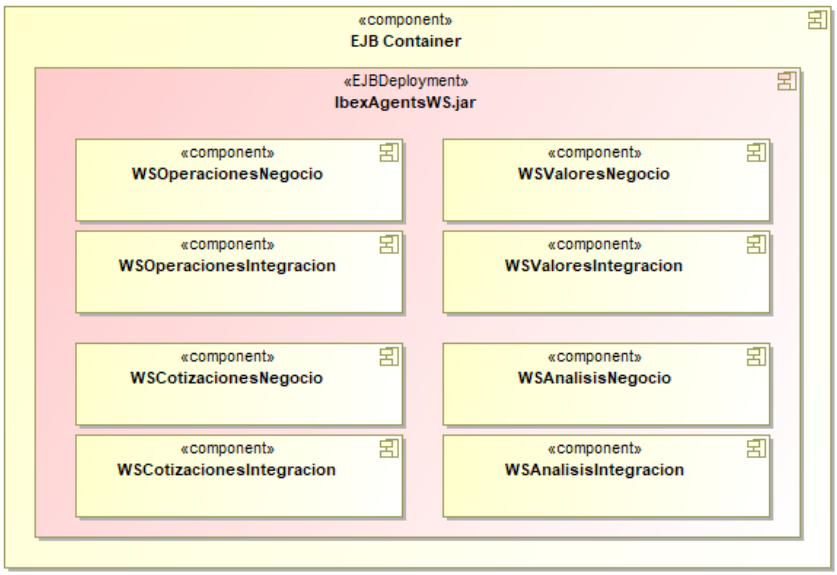
- El nombre del proyecto de desarrollo del web service es IbexAgentsWS
- El nombre del proyecto de desarrollo del sistema de agentes es JADEAgents
- El nombre del esquema PostgreSQL es ibexdata
- El usuario de acceso a PostgreSQL es USER

Así, para realizar el despliegue del servidor web se ha construido la aplicación ejecutando Ant sobre el fichero build.xml (desde Eclipse: botón derecho, RunAs, Ant) y se han obtenido los distintos paquetes que conforman la aplicación y que serán instalados de forma automática en el servidor de aplicaciones JBoss:

```
<property name="buildjar" value="${build}/jar"/>
<property name="buildwar" value="${build}/war"/>
<property name="deploy" value="${jboss.home}\standalone\deployments"/>
```

De tal forma, se empaquetan las distintas capas de la solución en dos contenedores .war y .jar, para ser, a su vez, empaquetados en un fichero .ear contenedor de toda la estructura del WS:

```
<!-- Deploy the ear. Copy the ear of the JBoss deployment directory -->
<target name="deployear" depends="ear">
  <copy file="${dist}/IbexAgentsWS.ear" todir="${deploy}"/>
</target>
```

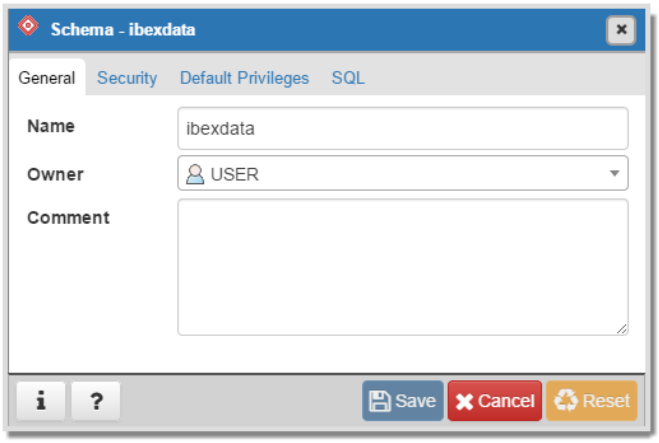


Respecto al despliegue del sistema multiagente, tan solo es necesario añadir al proyecto la librería del framework JADE que se puede obtener en el apartado de descargas de http://jade.tilab.com/ después de haber aceptado los términos generales para su uso (licencia GNU).

Es posible consultar entre los anexos a este proyecto todas las particularidades para la instalación y ejecución tanto del servicio web como de la plataforma de agentes JADE.

Creación de la BBDD sobre PostgreSQL

Para crear la BBDD que dé soporte a la capa de persistencia del sistema multiagente se ejecutará la script *ibexdata.sql* que permitirá obtener de forma automática toda la estructura de la BBDD (tabla, campos, formatos, etc).



Además, la script añade algunas instrucciones SQL para incorporar los siguientes datos por defecto necesarios para la correcta ejecución del sistema:

- Valor del índice del Mercado Continuo español al que se ha llamado IBEX
- Metodología y parametrización inicial del sistema

De tal forma, la estructura final de tablas de la BBDD debería quedar tal y como se muestra en la imagen inferior:

Type	Name
Table	ibexdata.agentes
Table	ibexdata.analisis
Table	ibexdata.compras
Table	ibexdata.cotizaciones
Table	ibexdata.ema
Table	ibexdata.historico
Table	ibexdata.indices
Table	ibexdata.maximos
Table	ibexdata.metodologia
Table	ibexdata.metodos
Table	ibexdata.minimos
Table	ibexdata.niveles
Table	ibexdata.operaciones
Table	ibexdata.parametrizacion
Table	ibexdata.parametros
Table	ibexdata.soportesresistencias
Table	ibexdata.valores
Table	ibexdata.ventas

Por último, se han añadido dos nuevos ficheros .sql que permitirán probar la solución con mayor eficiencia:

- *historical.sql*, fichero que se acompaña con el fichero *historicaldata.xml* para la importación de los datos históricos desde Enero del 2018 de los distintos valores que conforman el índice IBEX. Con ello se evita tener que descargar por programa la información, proceso de duraría alrededor de dos horas.
- *restartdata.sql*, fichero que borra los datos de todas la tablas de la BBDD con excepción de la tabla de índices, de métodos y de parámetros (datos por defecto en ibexdata.sql)

Instalación en Eclipse de las aplicaciones

Para importar las soluciones a Eclipse se recomienda seguir los siguientes pasos:

1. Haber configurado todo el software descrito en el apartado anterior incluido el framework de JADE. Es posible consultar los anexos adjuntos a este proyecto como guía de instalación.

2. Crear un proyecto vacío en Eclipse bien con el nombre de *IbexAgentsWS* si se va a importar el web service, bien con el nombre de *JADEAgents* si se va a importar el sistema multiagente:

Desde Eclipse: - *File/New/Project*
- *nombre: IbexAgentsWS*

3. Importar las soluciones adjuntas seleccionando todas las subcarpetas que presenta el asistente:

Desde Eclipse: - *Import File Systemt*
- *seleccionar todas las subcarpetas*

8. Ejecución y pruebas

Una vez configurado el software, es posible ejecutarlo tanto desde Eclipse como desde una ventana de comandos del sistema. En este caso, el software ha sido ejecutado y probado sobre un sistema operativo Microsoft Windows 10 Pro.

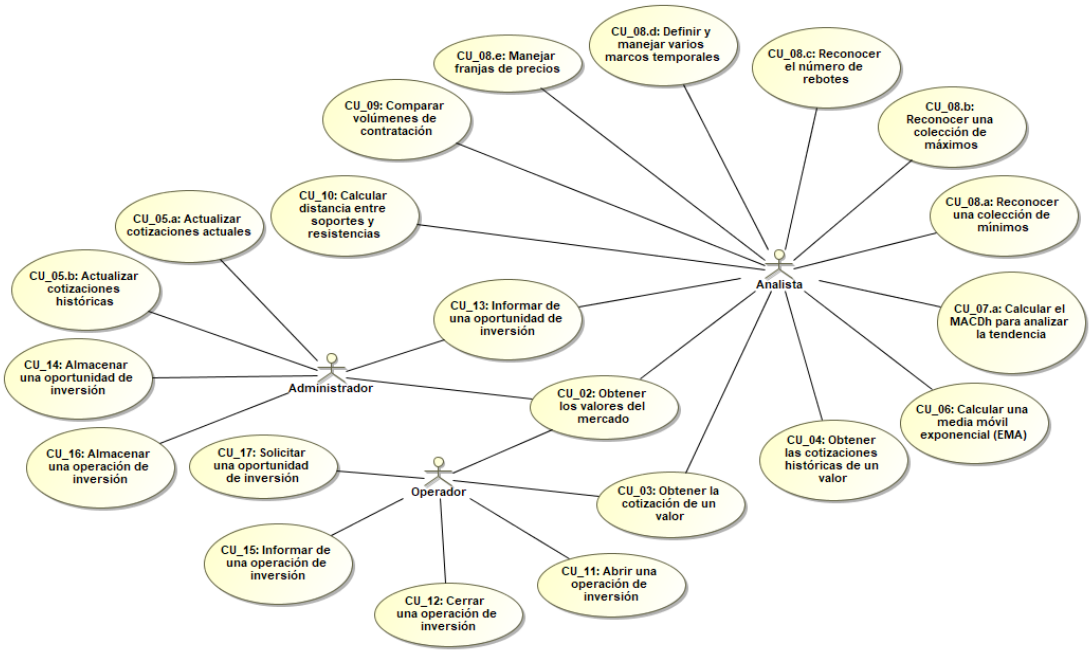
8.1. Análisis cualitativo del proyecto

Como se ha visto en la fase de gestión de requisitos del proyecto, las cualidades del software fueron estimadas y priorizadas sobre varios modelo y técnicas de metodología Agile hasta la obtención de un producto mínimo documentado sobre historias de usuario y casos de uso.

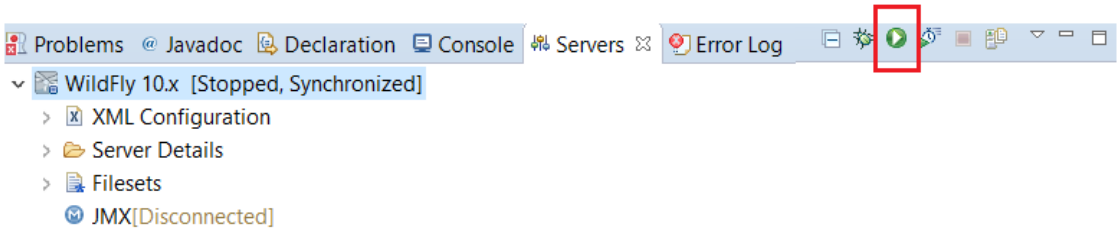
De tal forma, el sistema multiagente desarrollado está compuesto por tres agentes que interaccionan entre sí para intentar llevar a cabo una inversión financiera. Para ello, cada uno de los agentes tiene un rol y una misión específica dentro del sistema:

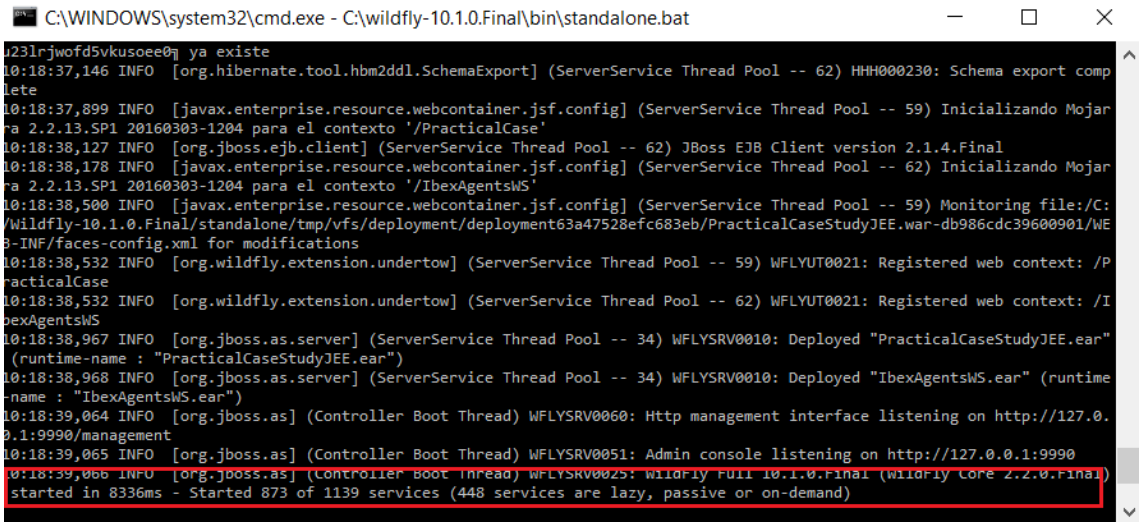
- El rol de agente analista, que plantea a partir de la información recogida de los mercados financieros un conjunto de posibilidades de inversión
- El rol de agente operador, que materializa las posibilidades de inversión con el fin de ir poco a poco revisando los resultados obtenidos y que el modelo vaya ganando en experiencia.
- Y el rol de agente administrador, que canaliza y dirige el flujo de información intercambiada, su sincronización y su persistencia, bien sea entre los distintos agentes, bien sea con el servicio web.

Así, el conjunto de requisitos que conforman el producto mínimo, se han traducido en un conjunto de operaciones y métodos que dan al sistema las cualidades funcionales necesarias para cumplir con sus especificaciones.

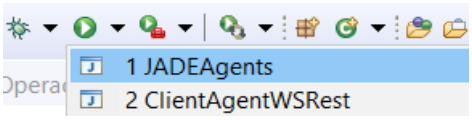


Y que se pueden evaluar mediante la ejecución del software obtenido y así probar si su funcionamiento se corresponde con las especificaciones desarrolladas con anterioridad. Para ello, primeramente ejecutaremos el servicio web, bien sea desde Eclipse, bien sea desde una ventana de comandos del sistema mediante la instrucción: *C:\wildfly-10.1.0.Final\bin\standalone.bat*, en la que habrá que sustituir la ruta de instalación del servidor JBoss por la adecuada en cada caso.



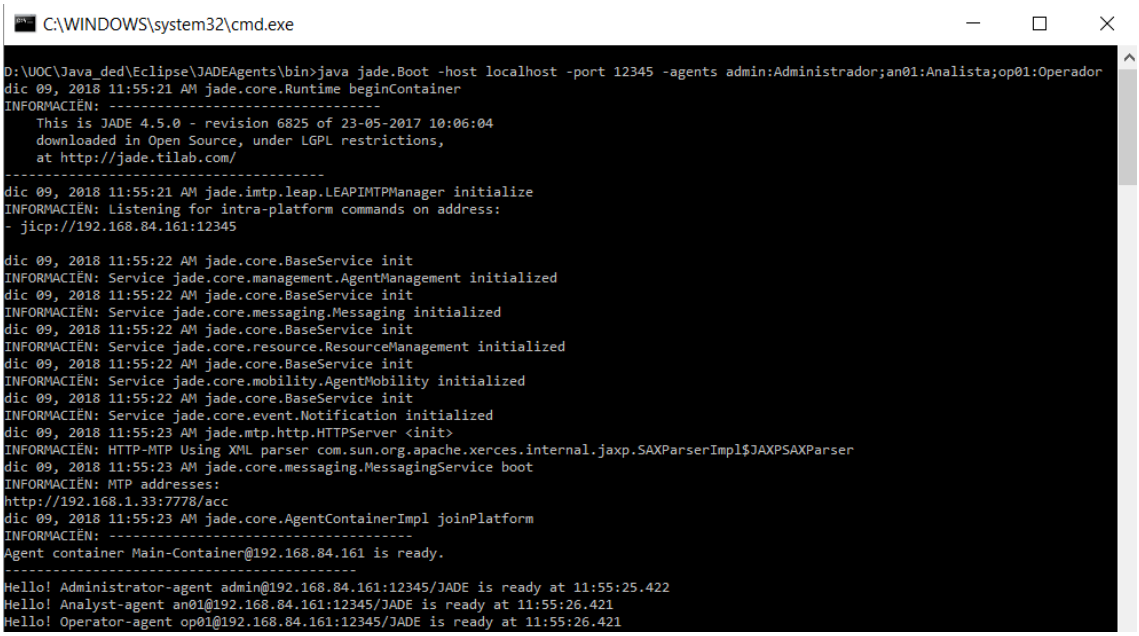


Y de igual forma, será posible ejecutar el sistema JADE multiagente, bien desde Eclipse mediante la configuración realizada con anterioridad y de la que se pueden consultar los detalles en los ficheros anexos a esta PAC:



O bien desde la línea de comandos del sistema, accediendo a la carpeta donde están las clases compiladas de la aplicación y llamando a la clase *jade.Boot* de JADE que se encuentra dentro de su librería y que habrá que instalar según las especificaciones descritas en el anexo, ya que será necesario (al igual que con Eclipse) haber configurado previamente las variables del sistema.

cd D:\UOC\Java_ded\Eclipse\JADEAgents\bin
java jade.Boot -host localhost -port 12345 -agents admin:Administrador;an01:Analista;op01:Operador



El sistema FIPA empieza presentando al conjunto de los agentes instanciados en la plataforma y continua con la obtención del agente administrador de la configuración por defecto del sistema y el conjunto de empresas cotizadas mediante la comunicación con el servicio web. Al mismo tiempo, los agentes con rol de analista y operador, solicitan mediante mensajes FIPA al administrador la configuración y la lista de valores obtenida. Una vez todos los agentes la obtengan, el sistema estará inicializado y comenzará a operar. Este inicio del sistema se puede ver desde una ventana de Eclipse tal y como se muestra en la imagen inferior:

```
Hello! Administrator-agent admin@192.168.84.161:12345/JADE is ready at 12:04:24.569
admin@192.168.84.161:12345/JADE at 12:04:24.691 says: Configuration ready
admin@192.168.84.161:12345/JADE at 12:04:25.340 says: Ibex stocks ready
Hello! Analyst-agent an001@192.168.84.161:12345/JADE is ready at 12:04:25.569
Hello! Operator-agent op001@192.168.84.161:12345/JADE is ready at 12:04:25.569
an001@192.168.84.161:12345/JADE at 12:04:25.570 says: Trying get configuration...
op001@192.168.84.161:12345/JADE at 12:04:25.570 says: Trying get configuration...
op001@192.168.84.161:12345/JADE at 12:04:25.588 says: Waiting for configuration
admin@192.168.84.161:12345/JADE at 12:04:26.345 says: New configuration query message from op001@192.168.84.161:12345/JADE
op001@192.168.84.161:12345/JADE at 12:04:26.348 says: Initial configuration received
op001@192.168.84.161:12345/JADE at 12:04:26.355 says: Trying get analysis...
an001@192.168.84.161:12345/JADE at 12:04:26.588 says: Trying get Ibex stocks...
an001@192.168.84.161:12345/JADE at 12:04:26.591 says: Waiting for initial data
admin@192.168.84.161:12345/JADE at 12:04:27.344 says: New configuration query message from an001@192.168.84.161:12345/JADE
an001@192.168.84.161:12345/JADE at 12:04:27.347 says: Initial configuration received
admin@192.168.84.161:12345/JADE at 12:04:28.345 says: New analysis query message from op001@192.168.84.161:12345/JADE
admin@192.168.84.161:12345/JADE at 12:04:28.345 says: Analysis not found. List is empty
op001@192.168.84.161:12345/JADE at 12:04:28.346 says: Null analysis received
admin@192.168.84.161:12345/JADE at 12:04:29.346 says: New Ibex stock query message from an001@192.168.84.161:12345/JADE
an001@192.168.84.161:12345/JADE at 12:04:29.372 says: Ibex stocks received
```

Con el sistema ya activo, el agente con rol analista irá analizando cada uno de los valores de forma secuencial según su parametrización. En el caso de que algún análisis sea considerado una oportunidad de inversión será enviada al agente administrador, que a su vez, lo almacenará en memoria y lo guardará de forma persistente en la BBDD invocando al servicio web. Al mismo tiempo, el agente con rol de operador irá solicitando un análisis al administrador hasta recibirlo de forma efectiva y así poder empezar a operar.

```
an001@192.168.84.161:12345/JADE at 11:58:41.638 says: Calling beliefs driver with IBEX.FCC
an001@192.168.84.161:12345/JADE at 11:58:41.651 says: Calling desires driver with IBEX.FCC
an001@192.168.84.161:12345/JADE at 11:58:41.651 says: Reported a new analysis of IBEX.FCC
admin@192.168.84.161:12345/JADE at 11:58:42.626 says: New analysis inform message from an001@192.168.84.161:12345/JADE
admin@192.168.84.161:12345/JADE at 11:58:42.792 says: New analysis save at WS (Id: 138, Share: IBEX.FCC)
admin@192.168.84.161:12345/JADE at 11:58:52.126 says: Updating Ibex stock exchange at WS
admin@192.168.84.161:12345/JADE at 11:58:52.354 says: New Ibex stock exchange saved at WS
op001@192.168.84.161:12345/JADE at 11:58:54.138 says: Checking price of IBEX.AEDAS: 22.4
an001@192.168.84.161:12345/JADE at 11:58:58.521 says: Calling beliefs driver with IBEX.FDR
an001@192.168.84.161:12345/JADE at 11:58:58.537 says: Calling desires driver with IBEX.FDR
an001@192.168.84.161:12345/JADE at 11:58:58.537 says: IBEX.FDR analysis rejected
op003@192.168.84.161:12345/JADE at 11:58:58.611 says: Trying get analysis...
admin@192.168.84.161:12345/JADE at 11:58:58.645 says: New analysis query message from op003@192.168.84.161:12345/JADE
admin@192.168.84.161:12345/JADE at 11:58:58.645 says: IBEX.FCC analysis found
op003@192.168.84.161:12345/JADE at 11:58:58.646 says: IBEX.FCC analysis received
op004@192.168.84.161:12345/JADE at 11:58:59.612 says: Trying get analysis...
admin@192.168.84.161:12345/JADE at 11:58:59.646 says: New analysis query message from op004@192.168.84.161:12345/JADE
admin@192.168.84.161:12345/JADE at 11:58:59.646 says: Analysis not found. List is empty
```

La programación de los ciclos de comunicación entre agentes es una parte muy importante del sistema, ya que el acceso a recursos comunes (acceso al WS o a la cola de mensajes FIPA) puede llegar a bloquear el sistema o incluso provocar algún tipo de resultado no esperado. Así, el ciclo de análisis se ha establecido en quince segundos:

```
an001@192.168.84.161:12345/JADE at 11:58:41.651 says: Reported a new analysis of IBEX.FCC
admin@192.168.84.161:12345/JADE at 11:58:42.626 says: New analysis inform message from an001@192.168.84.161:12345/JADE
admin@192.168.84.161:12345/JADE at 11:58:42.792 says: New analysis save at WS (Id: 138, Share: IBEX.FCC)
```

Y el ciclo de solicitudes de un nuevo análisis se ha establecido en treinta segundos.

```
op003@192.168.84.161:12345/JADE at 11:58:58.611 says: Trying get analysis...
admin@192.168.84.161:12345/JADE at 11:58:58.645 says: New analysis query message from op003@192.168.84.161:12345/JADE
admin@192.168.84.161:12345/JADE at 11:58:58.645 says: IBEX.FCC analysis found
op003@192.168.84.161:12345/JADE at 11:58:58.646 says: IBEX.FCC analysis received
```

Finalmente, cuando un agente operador recibe una oportunidad de inversión, lo revisa y comienza a operar sobre él, es decir, se comunica cada minuto con el servicio web para obtener la cotización actual, al mismo tiempo que el agente administrador también se comunica cada minuto para indicarle al servicio que actualice las cotizaciones de mercado.

```
op003@192.168.84.161:12345/JADE at 11:59:08.661 says: Reported a new buy of IBEX.FCC at price: 12.04
admin@192.168.84.161:12345/JADE at 11:59:09.659 says: New buy inform message from op003@192.168.84.161:12345/JADE
admin@192.168.84.161:12345/JADE at 11:59:09.699 says: New buy save at WS (Id: 76, Share: IBEX.FCC)
```

Respecto a la temporalidad definida, hay que notificar que ha sido establecida para una mejor interacción y prueba del sistema multiagente, ya que estos ciclos no tienen nada que ver con los ciclos de actualización de datos del servicio externo, ya que según se ha podido comprobar, actualiza las cotizaciones cada diez minutos.

```
admin@192.168.84.161:12345/JADE at 12:35:52.174 says: Updating Ibex stock exchange at WS
admin@192.168.84.161:12345/JADE at 12:35:52.422 says: New Ibex stock exchange saved at WS
```

Por último, notificar que las validaciones realizadas por el agente con rol de operador sobre los análisis recibidos pueden resultar, como ya se ha mencionado, en una oportunidad de inversión:

```
op001@192.168.84.161:12345/JADE at 12:35:52.890 says: Checking price of IBEX.AEDAS: 22.16
op003@192.168.84.161:12345/JADE at 12:35:53.886 says: Checking price of IBEX.FCC: 12.06
```

O bien pueden resultar fallidos a causa del precio actual del valor, en cuyo caso, el agente mantendrá el análisis vigente en memoria revisando el precio hasta que lo considere aceptable:

```
op004@192.168.84.161:12345/JADE at 12:35:54.842 says: Temporal rejected analysis of IBEX.FER at price: 18.08
op002@192.168.84.161:12345/JADE at 12:35:55.892 says: Temporal rejected analysis of IBEX.ITX at price: 26.81
```

8.2. Pruebas y detección de fallos

Las pruebas del software son una estrategia de gestión del riesgo en la que se valida si los requisitos se han cumplido y se buscan defectos evaluando las condiciones reales del software que se ha desarrollado. Así, primeramente, resumiremos agrupando por entidades la funcionalidad principal del sistema seleccionado los métodos más representativos de cada uno de los componentes del sistema. A tal respecto, notificar que la gran mayoría de estos métodos están compuestos a su vez por métodos de menor grado de dificultad:

WSOperaciones
- *public Collection<ComprasTO> obtenerOperacionesAbiertas();*
- *public Collection<VentasTO> obtenerOperacionesCerradas();*
- *public Collection<RetornoTO> guardarCompra(ComprasTO operacion);*
- *public Collection<RetornoTO> guardarVenta(VentasTO operacion);*

WSValores
- *public Collection<ValoresTO> obtenerCotizadas(@PathParam("indice") String indice);*
- *public Collection<ValoresTO> obtenerCotizadasAdmin(@PathParam("indice") String indice);*

WSAnalisis
- *public Collection<AnalisisTO> obtenerAnalisis();*
- *public Collection<RetornoTO> guardarAnalisis(AnalisisTO analisis);*

WSCotizaciones
- *public Collection<CotizacionesTO> obtenerCotizacion(@PathParam("indice") String indice, @PathParam("valor") String valor);*
- *public Collection<HistoricoTO> obtenerCotizacionHistorica(@PathParam("indice") String indice, @PathParam("valor") String valor);*
- *public java.util.Collection<RetornoTO> actualizarCotizaciones();*
- *public Collection<RetornoTO> actualizarCotizacionHistorica(@MatrixParam("indice") String indice, @MatrixParam("valor") String valor, @MatrixParam("referencia") String referencia)*

FIPA
- *public static ACLMessage newQuery(Solicitud solicitud, String destinatario)*
- *public static ACLMessage replyQuery(Solicitud solicitud, AID destinatario)*
- *public static ACLMessage newInform(Solicitud solicitud, String destinatario)*

ICreencias/Creencias
- *public Collection<Ema> calcularEma(short tipo, short numSesiones, Collection<HistoricoTO> historico)*
- *public Macd calcularMacd(short numSesionesCorto, short numSesionesLargo, short numSesionesSenal, Collection<HistoricoTO> historico)*
- *public Collection<Soportesresistencias> calcularNiveles(short tipo, short modelo, float precision, Collection<HistoricoTO> historico)*
- *private static List<HistoricoTO> obtenerSesionesHistorico(List<HistoricoTO> historico, short numSesiones)*
- *private static List<HistoricoTO> obtenerCotizacionesSemanales(List<HistoricoTO> cotizaciones)*
- *private static float calcularEmaPrecios(List<Float> cotizaciones, short numSesiones)*
- *private static List<Macd> calcularMacd(List<Float> listaEmaCorto, List<Float> listaEmaLargo)*
- *private static Soportesresistencias calcularPrevioNiveles(List<Float> maximos, List<Float> minimos)*
- *private static List<Float> seleccionarMaximosMinimos(List<Float> maxmin)*
- *private static List<Rangos> calcularRangos(float precision, List<Float> maximos, List<Float> minimos)*

IDeseos/Controlador Deseos
- *public float valorMetodoEmaPrecio(Collection<Ema> ema);*
- *public NivelPrincipal valorMetododRangoNiveles(Collection<Soportesresistencias> sopres, float precio);*
- *public long valorMetodoEmaVolumen(Collection<Ema> ema);*
- *private static boolean validarReglas(Collection<Metodo> reglas, CotizacionesTO cotizacion, AnalisisTO analisis)*
- *private static short calcularPrioridad(Collection<Metodo> prioridades, AnalisisTO analisis)*

Control Intenciones
- *public Collection<Metodo> nuevaConfiguracion(Collection<Metodo> listaMetodos)*

Operador - Validación
- *private static Compra calcularOperacion(float precio, float soporte, float varPrecioSalida, float varStoploss, float varStoplossMaxima)*

Configuracion

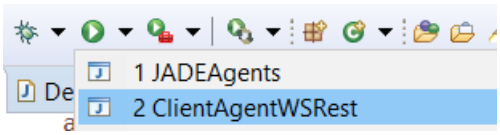
- *public Collection<MetodosTO> obtenerConfiguracion();*

Comunicaciones

- *private static synchronized Document getDoc(String nameUrl)*
- *private static String analisisToString(AnalisisTO analisis)*

Así, para poder evaluar con mayor certeza cada una de las funcionalidades descritas y poder comprobar si se cumplen con los objetivos del proyecto, se ha desarrollado un software de pruebas que nos permitirá probar algunos de los métodos (principalmente las comunicaciones con el servicio web) de forma aislada a los procesos de interacción continua entre agentes.

De tal forma, se ha desarrollado la aplicación *TestIAWS* en Java que nos permitirá automatizar alguna de las pruebas a realizar sobre el sistema. Su instalación y configuración se describen al detalle en los anexos de esta PAC:



En este caso, se ha ejecutado la aplicación desde el IDE Eclipse, de tal forma que nos muestra un menú con varias opciones para probar el sistema:

```
A G E N T S  W S
UOC TFG based on Rest Web Service

0 - Salir

1 - Ver valores de la BBDD
2 - Ver cotizacion actual de un valor desde la BBDD
3 - Ver histórico de un valor desde la BBDD

4 - Obtener valores desde proveedor externo
5 - Obtener cotizaciones actuales desde prov. externo
6 - Obtener TODAS cotizaciones históricas desde prov. ext.

7 - Grabar un análisis
8 - Grabar una operación de compra
9 - Grabar una operación de venta (HAY QUE AÑADIR ids AL CÓDIGO)

10 - Test (Pruebas dinámicas cambiando el código)
11 - Test+ (Pruebas dinámicas cambiando el código)
12 - Test++ (Pruebas dinámicas cambiando el código)
 Escoja una opción:
```

A modo de resumen, las opciones del 1 al 6 interactúan con el servicio web sobre los componentes WSValores y WSCotizaciones, en donde la opción 6 tiene una especial utilidad, ya que actualiza la BBDD histórica de cotizaciones sin esperar a que lo haga el agente administrador (lo hace una vez al día, pasadas 24h desde que se ha iniciado el sistema multiagente).

Por otra parte, las opciones del 7 al 9 interactúan con los componentes WSAnalisis y WSOperaciones invocando los métodos POST para su almacenamiento. Es importante notificar que podría ser necesario modificar el código y actualizar los identificadores (claves) empleados. Notificar también, que es posible invocar los métodos GET de estos componentes mediante llamadas desde cualquier navegador web (ver apartado 4 de esta PAC).

Por último, las opciones 10 al 12 son fragmentos de código del componente BDI del proyecto JADE sin ningún orden ni preestablecido: pruebas de repeticiones, cálculos, etc…

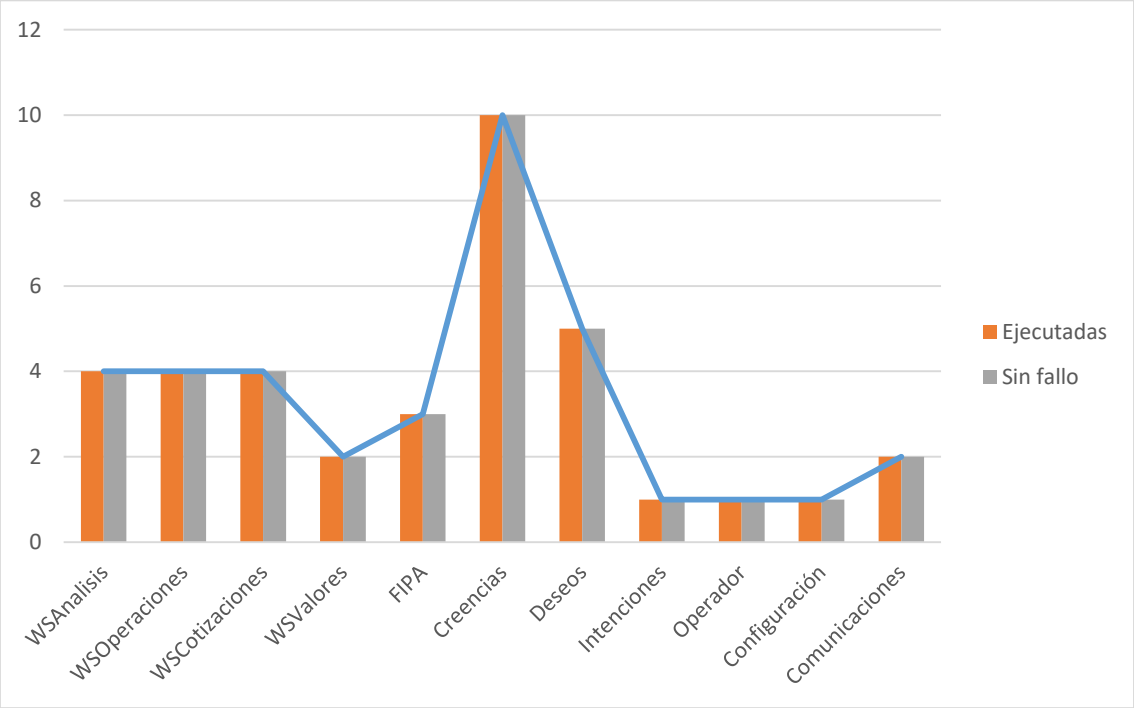
8.2.1. Requisitos mínimos del sistema

Para realizar las pruebas del sistema, es necesario que existan en la BBDD los datos necesarios para poder identificar como índice al Mercado Continuo español (IBEX), la metodología y parametrización empleada por los agentes por defecto, y las cotizaciones históricas con un mínimo de tres meses de los valores del índice anterior.

Para el cumplimento de los dos primeros requisitos, se añade en la script *ibexdata.sql* las instrucciones SQL que añaden el índice IBEX y la metodología por defecto a la BBDD. Para el cumplimiento del tercer requisito, se añade la script *historical.sql* con el historial de cotizaciones desde Enero del 2018, aunque también es posible recogerlas bien desde la aplicación de pruebas *TestIAWS*, bien desde el propio sistema FIPA esperando 24h desde su iniciación.

8.2.2. Coberturas de las pruebas

Siguiendo el esquema de métodos más representativos expuesto con anterioridad, la cobertura de las pruebas realizadas se puede resumir en el siguiente gráfico:



A tal respecto, cabe señalar que al no existir un conjunto de usuarios finales de la aplicación, la detección de defectos mediante pruebas de aceptación no es posible, por lo que el gráfico anterior debe interpretarse desde una perspectiva de pruebas del sistema, es decir, pruebas dirigidas a probar el sistema de forma completa, en función de los requisitos y objetivos establecidos. Para ello, como ya se ha mencionado, se ha facilitado un software de pruebas unitarias y de componente (unidades del sistema) y se han realizado diversas pruebas de integración entre componentes, así como diversas pruebas de rendimiento y responsabilidad del sistema.

De tal forma, es preciso señalar que las pruebas de consistencia del sistema se han realizado variando bien la frecuencia de ejecución de las distintos subprocesos de cada uno de los agentes que definen su comportamiento (*behaviours*), bien el número de agentes con distintos roles instanciados en el sistema. Así, los bloqueos y tiempos de ejecución entre los distintos componentes del sistema deben tener en cuenta el comportamiento de las hebras que los componen en referencia a su sincronización y acceso a los recursos compartidos, por lo que, cambios en los tiempos de estos bloqueos podrían afectar a la consistencia del sistema.

De tal manera, las pruebas de concurrencia máxima de agentes se han realizado con un agente administrador, un agente analista y diez agentes operadores, obteniéndose resultados satisfactorios durante toda la vida útil de la prueba (doce horas).

```
an001@192.168.84.161:12345/JADE at 12:35:49.225 says: Calling beliefs driver with IBEX.EDR
an001@192.168.84.161:12345/JADE at 12:35:49.236 says: Calling desires driver with IBEX.EDR
an001@192.168.84.161:12345/JADE at 12:35:49.236 says: IBEX.EDR analysis rejected
admin@192.168.84.161:12345/JADE at 12:35:52.174 says: Updating Ibex stock exchange at WS
admin@192.168.84.161:12345/JADE at 12:35:52.422 says: New Ibex stock exchange saved at WS
op001@192.168.84.161:12345/JADE at 12:35:52.890 says: Checking price of IBEX.AEDAS: 22.16
op003@192.168.84.161:12345/JADE at 12:35:53.886 says: Checking price of IBEX.FCC: 12.06
op004@192.168.84.161:12345/JADE at 12:35:54.842 says: Temporal rejected analysis of IBEX.FER at price: 18.08
op002@192.168.84.161:12345/JADE at 12:35:55.892 says: Temporal rejected analysis of IBEX.ITX at price: 26.81
op005@192.168.84.161:12345/JADE at 12:35:56.866 says: Checking price of IBEX.IBG: 31.7
an001@192.168.84.161:12345/JADE at 12:36:05.982 says: Calling beliefs driver with IBEX.EKTL
an001@192.168.84.161:12345/JADE at 12:36:05.993 says: Calling desires driver with IBEX.EKTL
an001@192.168.84.161:12345/JADE at 12:36:05.993 says: IBEX.EKTL analysis rejected
an001@192.168.84.161:12345/JADE at 12:36:22.747 says: Calling beliefs driver with IBEX.ELE
an001@192.168.84.161:12345/JADE at 12:36:22.763 says: Calling desires driver with IBEX.ELE
an001@192.168.84.161:12345/JADE at 12:36:22.763 says: IBEX.ELE analysis rejected
```

8.2.3. Historias de usuario y casos de uso

Por último, los distintos requisitos documentados, bien en historias de usuario bien en casos de uso, en fases anteriores del proyecto han sido probados, en especial el caso de uso que recogía el ciclo de vida de una operación de inversión. Prueba de ello son las distintas capturas de pantalla de este y el anterior apartado.

- Como analista quiero poder informar de una oportunidad de inversión (análisis)
- Como administrador quiero poder almacenar una oportunidad de inversión
- Como operador quiero poder solicitar una oportunidad de inversión
- Como administrador quiero poder informar de una oportunidad de inversión
- Como operador quiero poder abrir una operación de inversión
- Como operador quiero poder cerrar una operación de inversión

8.3. Análisis cuantitativo del proyecto

Uno de los objetivos principales del aseguramiento de la calidad del software (SQA) es el establecimiento de un conjunto de métricas sobre las cuales medir distintos atributos del software y cuya finalidad es poder realizar diferentes evaluaciones durante el ciclo de vida del software para comprobar que los requisitos de calidad se han alcanzado, es decir, recogiendo de nuevo la definición que ofrece la RAE sobre calidad, comprobar que el conjunto de propiedades inherentes al software nos permiten juzgar su valor y concluir que técnicamente se ha cumplido con ellos.

De tal forma, evaluamos primeramente el conjunto de componentes de los programas y su código con la herramienta de análisis estructural para Java STAN (http://stan4j.com/) y seguidamente, con la herramienta de análisis de código SonarQube (www.sonarqube.org) que permite evaluar el seguimiento de los estándares de codificación y buenas prácticas en la programación.

Así, una vez revisada la codificación de los programas según los resultados presentados por las herramientas de análisis de calidad anteriores y corregidos algunos problemas en las mediciones de algunas de las métricas:

- Problemas de tamaño en algunos componentes del servicio web según la métrica de número estimado de líneas de código (ELOC) que los identificaban como demasiado extensos con lo que su mantenibilidad podría suponer dificultades futuras.

- Problemas de complejidad ciclomática (CC) en algunos componentes del sistema JADE, con la consiguiente dificultad para para entender y verificar su codificación.

- Problemas con bugs (como por ejemplo, excepciones en threads que no se finalizan o salidas de métodos en decisiones if con la instrucción return y el resto del código escrito fuera de la condición) y vulnerabilidades (como por ejemplo catch sin código).

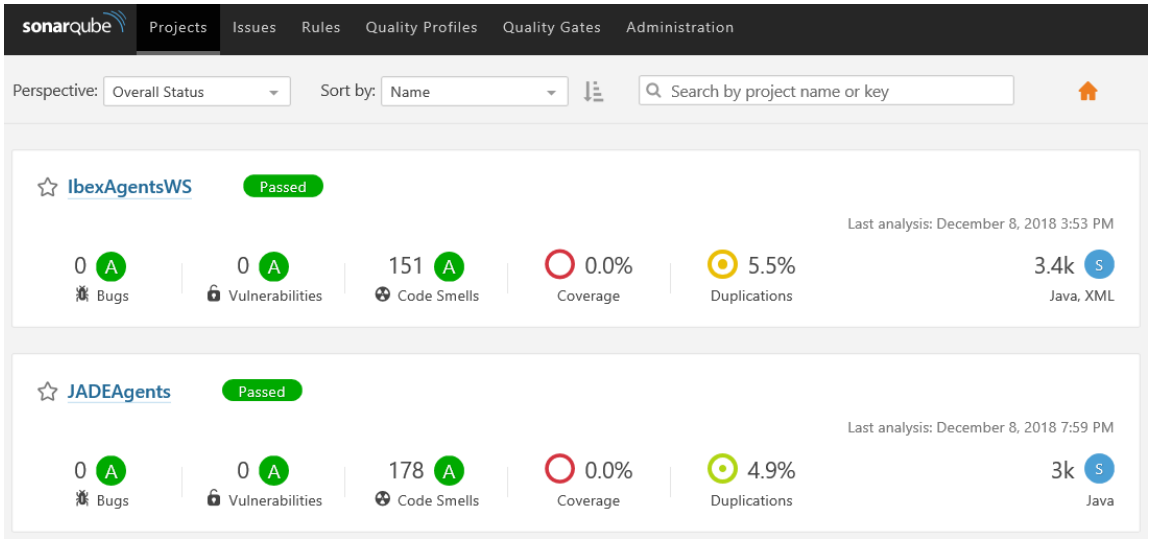
Se obtienen, **c**omo resultado del análisis estructural del servicio web, únicamente dos avisos: El primero indicando el grado de responsabilidad del componente WSAnalisis, algo por otra parte esperable, debido al número de entidades ejb enlazadas a un análisis, y el segundo en referencia a la complejidad de la obtención de las cotizaciones, fruto del conjunto de decisiones que hay que tomar para clasificar los datos obtenidos desde el servicio web externo.

Artifact	Metric	Value
IbexAgentsWS	DIT	0.8913
ejb.AnalisisFacadeBean	RFC	102
ejb.CotizacionesFacadeBean.obtenerCotizacion(String[])	CC	16

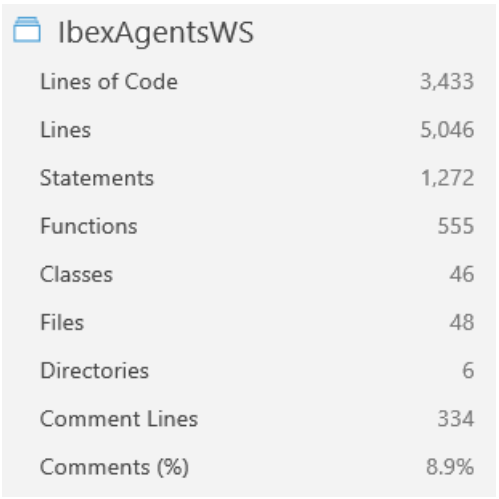
Como resultado de la revisión estructural del sistema multiagente, se obtienen también una serie de avisos que hacen mención, principalmente, al componente Creencias, algo que también es de esperar ya que el conjunto de cálculos de indicadores de inversión se realizan en este componente y en consecuencia su codificación es más extensa en número de métodos, algo que se puede apreciar de manera más inmediata en el gráfico de cobertura de las pruebas.

Artifact	Metric	Value
JADEAgents	DIT	0.96
bdi.Creencias	ELOC	370
Operador.stateMachine.action()	ELOC	88
bdi.ControlCreencias.ejecutar(Collection<Metodo>, Collection<HistoricoTO>)	CC	15
bdi.ControlCreencias.setParametro(String, float)	CC	17

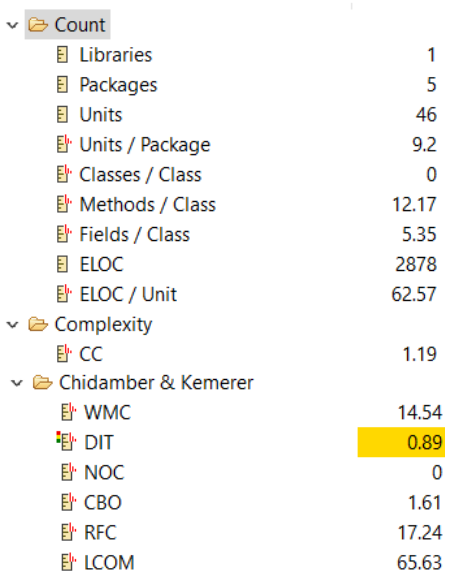
Finalmente, una vez revisado el estilo de codificación de los programas según las vulnerabilidades y bugs encontradas por SonarQube, se obtiene la más alta calificación (A) en el cumplimiento de los estándares y recomendaciones de buenas prácticas propuestas por el propio software.



Por último, se adjuntan algunas de las métricas finales por cada uno de los sistemas de software desarrollados. Es posible consultar más al detalle entre los anexos de esta PAC los resultados de las métricas por paquete y por clase del conjunto de componentes principales del software.



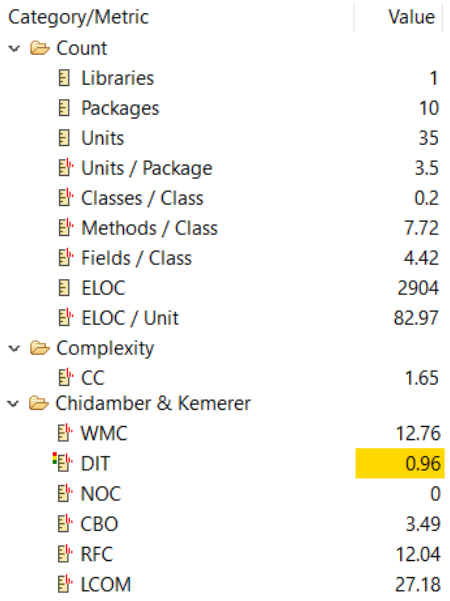
IbexAgentsWS	
Lines of Code	3,433
Lines	5,046
Statements	1,272
Functions	555
Classes	46
Files	48
Directories	6
Comment Lines	334
Comments (%)	8.9%



Category/Metric	Value
Count	
Libraries	1
Packages	5
Units	46
Units / Package	9.2
Classes / Class	0
Methods / Class	12.17
Fields / Class	5.35
ELOC	2878
ELOC / Unit	62.57
Complexity	
CC	1.19
Chidamber & Kemerer	
WMC	14.54
DIT	0.89
NOC	0
CBO	1.61
RFC	17.24
LCOM	65.63

El conjunto de mediciones de la aplicación IbexAgentsWS muestra que el mayor número de métricas se sitúa dentro de los valores óptimos establecidos. Así, la complejidad ciclomática (CC) o puntos de decisión del grafo obtenido a partir del código fuente se sitúa dentro de los valores recomendados por el software STAN, es decir, entre 0 y 15; el número de métodos ponderados por clase (WMC) que suma la CC sobre todos los métodos de las clases se sitúa entre 0 y 100; el acoplamiento entre objetos (CBO) recoge un valor menor que 25 y la responsabilidad media de las clases (RFC) un valor menor que 100. Tan solo, la métrica que recoge la media de profundidad del árbol de herencia (DIT), es decir, la mayor longitud desde las clases de la solución hasta la clase raíz, recoge un valor próximo a su óptimo, que es cuando se sitúa por encima de uno.

JADEAgents	
Lines of Code	2,986
Lines	4,051
Statements	1,402
Functions	353
Classes	45
Files	35
Directories	10
Comment Lines	223
Comments (%)	6.9%



Category/Metric	Value
Count	
Libraries	1
Packages	10
Units	35
Units / Package	3.5
Classes / Class	0.2
Methods / Class	7.72
Fields / Class	4.42
ELOC	2904
ELOC / Unit	82.97
Complexity	
CC	1.65
Chidamber & Kemerer	
WMC	12.76
DIT	0.96
NOC	0
CBO	3.49
RFC	12.04
LCOM	27.18

Respecto al conjunto de mediciones de la aplicación JADEAgents vemos que también todas las métricas están dentro de sus valores óptimos a excepción de la media de profundidad del árbol de herencia, que está todavía más próximo a su óptimo que en el caso del servicio web.

8.4. Conclusiones y continuidad del proyecto

El software presentado junto a esta memoria[3] hay que considerarlo como una versión Beta del sistema multiagente, de tal forma, aunque se trata de una versión del producto completa y útil, hay que considerarla como una primera versión preliminar para poder ser evaluada y a la que habrá que ir añadiendo funcionalidad respecto a varios aspectos que permitirán, principalmente, mejorar el producto desde el punto de vista de su eficiencia, la interacción con su entorno, la gestión de los fallos y la optimización del estudio de los datos.

Así, a raíz de las pruebas realizadas, es posible obtener algunas conclusiones respecto a la configuración del sistema en referencia al número de agentes a implementar en función de su rol dentro del modelo:

- Las pruebas realizadas muestran que con la configuración actual no es necesario más de una agente analista en el sistema, ya que los análisis se realizan sobre la información histórica del mercado y esta no se actualiza hasta pasadas 24 horas. Así, en esta primera versión, una vez que el agente analista revisa toda la lista de valores cotizados en el mercado, siempre que al menos haya encontrado una oportunidad de inversión, la actividad del agente será bloqueada durante el tiempo de una hora.

- Además, las pruebas también indican que es necesario un número elevado de operadores, ya que los resultados obtenidos muestran que en función de las cotizaciones del mercado, el número de posibles inversiones puede llegar hasta siete diarias. Notificar también que en el plazo temporal de un día ninguna de las compras efectuadas ha podido ser cerrada al no haber alcanzado la cotización del valor ninguno de los precios de venta.

Por otra parte, también será necesario corregir ciertos aspectos del software en referencia a la logística del mercado, como puede ser el manejo de los tiempos de actividad, interacción, etc…

- Esta primera versión Beta del sistema está configurada temporalmente para poder ser evaluada y no para interactuar con los plazos y tiempos del mercado de valores. Así, las operaciones efectuadas por el conjunto de agentes se realizarán muchas veces sin que, en las condiciones actuales de obtención de información, los datos de mercado hayan podido variar. Este comportamiento se debe únicamente para una mejor evaluación del sistema, ya que en la actualidad el servicio externo empleado actualiza los datos de mercado en un tiempo aproximado de diez minutos y adaptar el programa a estos plazos haría que la tarea de revisar el comportamiento del programa se prolongase más de lo deseado.

- A tal respecto, se puede afirmar que el servicio externo empleado en la actualidad para la obtención de datos de mercado en tiempo real no cumple con las necesidades futuras del sistema. Si bien la web www.invertia.com sí que puede emplearse como proveedor de datos históricos, el retraso en las cotizaciones de quince minutos y la tasa de refresco de estas cotizaciones de unos diez minutos, hacen que uno de los requisitos futuros de mejora del sistema sea obtener un nuevo proveedor externo de información en tiempo real.

Continuando con algunas de las mejoras futuras reconocidas en la fase de pruebas del sistema, además de la ya mencionada sustitución del proveedor externo de datos, la versión definitiva del sistema debería avanzar sobre los siguientes aspectos:

- Es relativamente fácil ampliar la funcionalidad del agente analista para darle responsabilidad sobre la información obtenida en tiempo real. De tal forma, el estudio de la EMA o del MACD se puede realizar intercalando como último dato el obtenido en tiempo real, con lo que se conseguiría ampliar el funcionamiento del rol de analista y simplificar con ello el manejo de tiempos de bloqueo.

(3) Se presenta junto a esta versión Beta, un segundo modelo de software que implementa varias mejoras en referencia a la gestión de fallos, el registro en el servicio de directorio (DF) y la gestión de agentes (AMS): Ver detalles en el fichero adjunto con el software JADEAgentsPro.txt

- Además, en las actualizaciones futuras del sistema, se pueden añadir nuevos métodos de análisis a los ya implementados, algunos de ellos recogidos ya en algunos de los requisitos funcionales del sistema desechados en la fase de priorización de requisitos como pueden ser el estudio de soportes a partir del retroceso de Fibonacci, el estudio de velas, etc…

- Otra futura mejora será la optimización temporal del manejo de análisis que por precio no es aconsejable interactuar con ellos y que en la actualidad son gestionados por un operador e informados con el título de "*temporal rejected analysis*". Según los resultados obtenidos por el sistema, habrá que proponer soluciones del tipo desechar el análisis temporal una vez haya pasado un tiempo *t*

- También habrá que ampliar según los resultados obtenidos el componente de intenciones del modelo BDI del que en la actualidad tan solo se ha probado su capacidad para modificar el método operacional común del sistema multiagente: En el caso de que no se haya encontrado ninguna oportunidad con los márgenes de rentabilidad propuestos por defecto, este margen se disminuirá hasta el 4%.

Por último, algunos de los aspectos en los que habría que centrarse en el futuro en referencia a la mejora de eficiencia, recuperación de fallos y administración de roles de agente son:

- Ya se ha comentado que el software es una versión de evaluación del sistema, por lo que una vez activado el sistema, por lo general este interactúa con el mercado de forma continua y sin descanso. A tal respecto, habría que ampliar el sistema para que reconociera las horas de apertura de cada uno de los mercados y mejorase así su eficiencia a la hora de interactuar con ellos. Así, habría que modificar la tabla *Indices* añadiendo la hora de apertura y cierre, y bloquear los agentes mientras ningún índice estuviera abierto.

- Respecto a la recuperación de fallos, el modelo multiagente no propone soluciones ante posibles caídas o fallos del sistema, de tal forma, cada vez que haya que reiniciar el sistema la plataforma de agentes reiniciará su comportamiento como si nunca hubieran operado. A tal respecto, el servicio web ya ha sido programado para dar soporte a un modelo de recuperación de fallos y cabría implementar la recuperación de las operaciones en curso por los distintos agentes en caso de fallo mediante su obtención desde el WS y distribución a partir de un nuevo tipo de mensaje FIPA.

- Siguiendo con el modelo de recuperación de fallos, la versión actual de software, no implementa ningún modelo de reinicio del servicio web en caso de caída, tan solo revisa en el arranque del sistema su disponibilidad y cada rol de agente se desactivará en caso de mal funcionamiento. Algunas mejoras al respecto serían la implementación de la operativa de administración del servicio web y la centralización de la activación y desactivación del conjunto de roles de agente por el agente administrador.

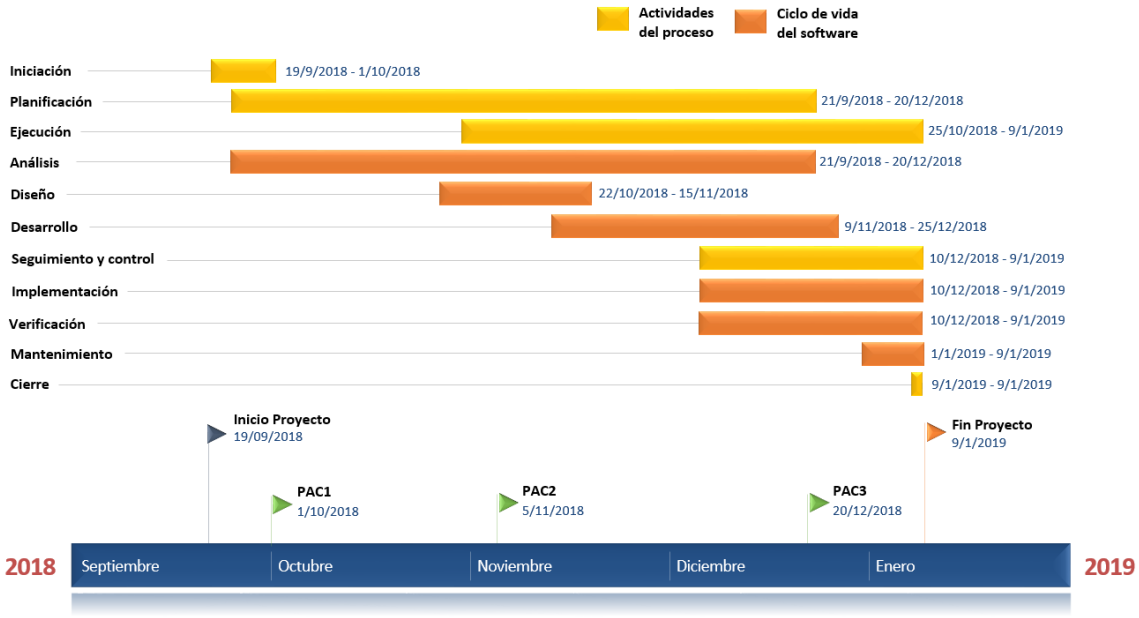
- Tampoco hay una gestión activa respecto a posibles caídas del servicio externo de datos, ya que en la actualidad en caso de poder recuperar la información actualizada del sistema se utiliza la última obtenida evitando un mal funcionamiento del software. De todas formas, también cabría plantear mejoras respecto al mal funcionamiento del servicio externo, como el cierre automático de operaciones, etc…

- Finalmente, también se podrían plantear mejoras en referencia a la eficiencia operativa del sistema: En la actualidad el número de agentes se definen parametrizando la instrucción de arranque del sistema, de tal forma que una vez implementados los agentes su número no puede variar. Así, otra mejora futura sería instrumentar en el agente administrador la capacidad de reconocer el número ideal de agentes en el sistema y que estos fueran implementados o retirados de la plataforma en tiempo real.

ANEXOS

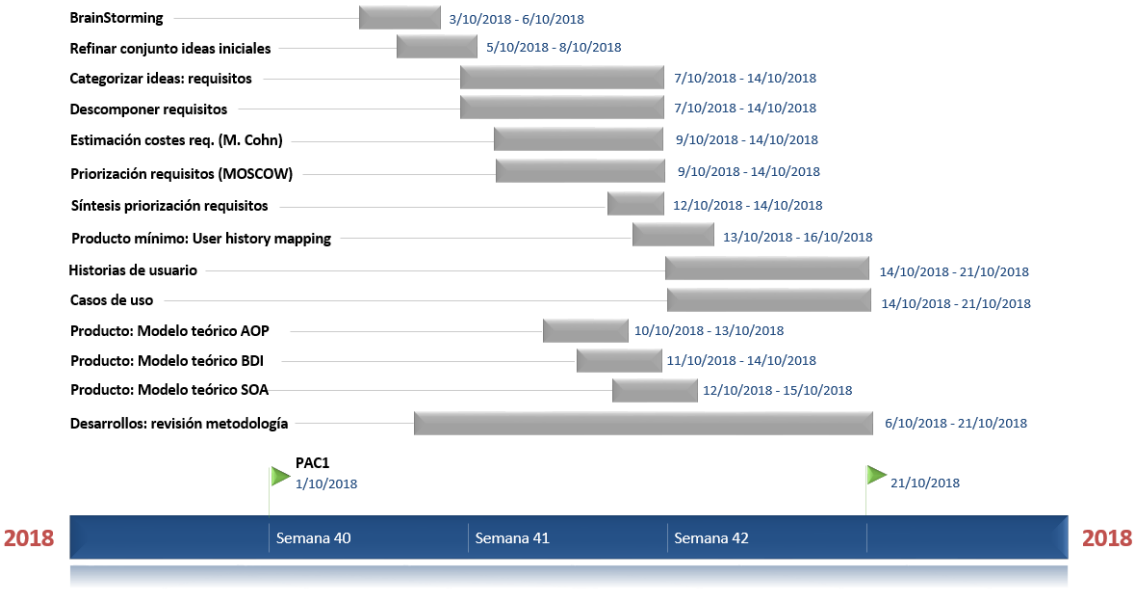
CRONOGRAMA Y ACTIVIDADES DEL PROYECTO

Cronograma a largo plazo de las actividades del proyecto según las actividades de gestión del proyecto (PMBOK) y el ciclo de vida del software:

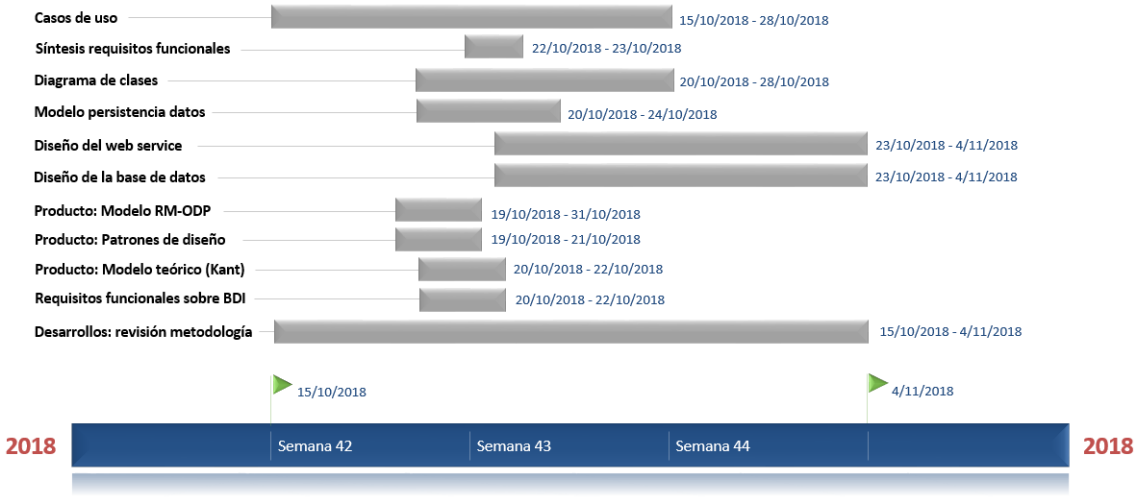


Cronogramas a corto plazo de las actividades del proyecto según el alcance del producto:

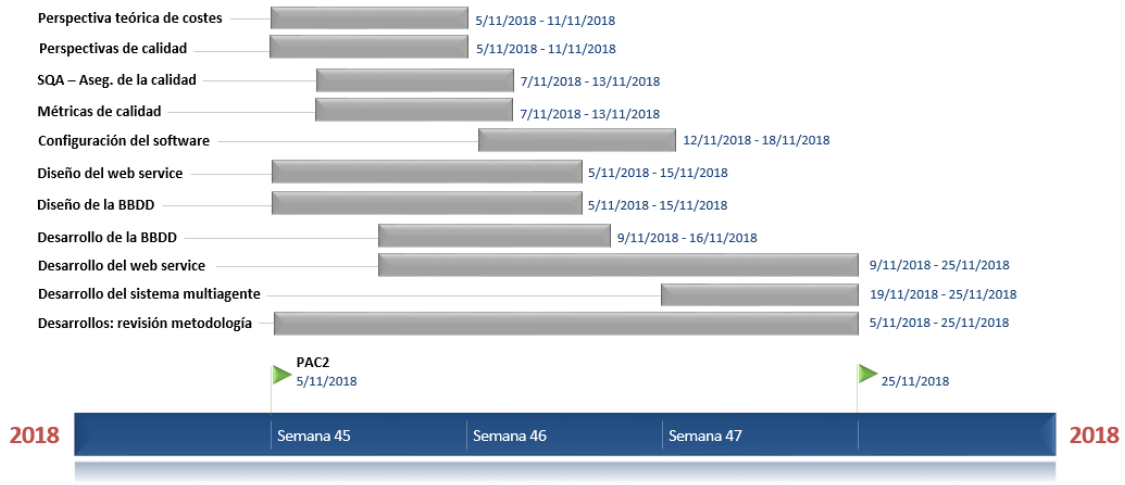
- Semanas 40 a la 42:



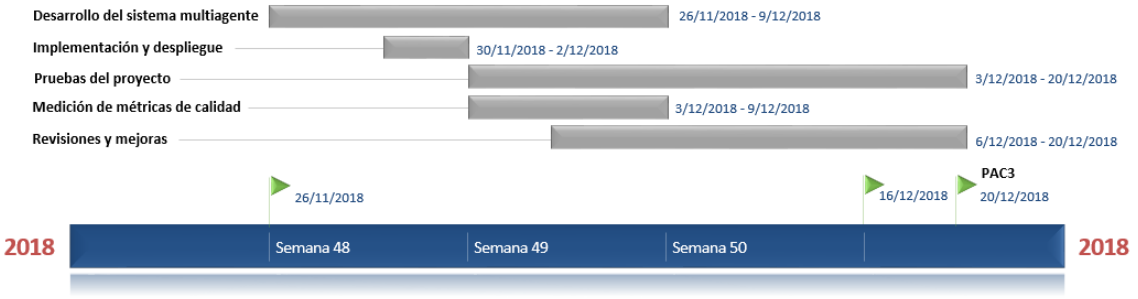
- Semanas 42(*) a la 44:



- Semanas 45 a la 47:



- Semanas 48 a la 50:



(*) Actualización de la planificación semanal anterior

DOCUMENTACIÓN FORMAL DE REQUISITOS: CASO DE USO DEL CICLO DE VIDA DE UNA OPERACIÓN DE INVERSIÓN

Caso de uso: Informar de un oportunidad de inversión (análisis positivo)

Actor principal: Agente con rol de analista

Ámbito: Sistema multiagente

Nivel de objetivo: Análisis de información

Stakeholders e intereses:

El analista estudia y analiza la cotización de un valor y resuelve que es una oportunidad de inversión, por lo que lo informa al resto de agentes que delegan en la figura del administrador del sistema la gestión y el mantenimiento de la información común

Precondición: El analista ha recuperado y analizado los datos históricos y la cotización actual de un valor, es decir, conoce los máximos y mínimos históricos del valor, los niveles de rebote, la media móvil exponencial del precio y del volumen, y el histograma del MACD

Garantías en caso de éxito: El analista enviará al administrador un informe (mensaje FIPA-ACL del tipo INFORM) con la oportunidad de inversión

Escenario principal de éxito:

1) El analista calcula la distancia porcentual entre la resistencia y el soporte más relevantes, y esta es mayor o igual que el 6%

2) El analista calcula la EMA actual y anterior de precios de ocho sesiones, y la actual es mayor a la anterior

3) El analista calcula el MACDh y este es positivo

4) El analista calcula la EMA actual y anterior del volumen de ocho sesiones y la actual es mayor, por lo que marca el análisis como prioritario

Extensiones:

1a. El analista calcula la distancia entre la resistencia y el soporte más relevantes y es menor al 6%, por lo que abandona el análisis finaliznado el caso de uso, para seleccionar el siguiente valor de su lista de valores y analizarlo

2a. El analista calcula la EMA actual y anterior de precios de ocho sesiones, y la anterior es mayor, por lo que abandona el análisis finalizando el caso de uso, para seleccionar el siguiente valor de su lista de valores y analizarlo

3a. El analista calcula el MACDh y es negativo, por lo que abandona el análisis finalizando el caso de uso, para seleccionar el siguiente valor de su lista de valores y analizarlo

4a. El analista calcula la EMA actual y anterior de volúmenes de ocho sesiones, y la anterior es mayor, por lo que no añade ninguna marca al análisis

Caso de uso: Almacenar (guardar) una oportunidad de inversión

Actor principal: Agente con rol de administrador

Ámbito: Sistema multiagente

Nivel de objetivo: Administración de la información (servicio web)

Stakeholders e intereses:

El administrador almacena en la BBDD un análisis clasificado como oportunidad de inversión

Precondición: El administrador recibe un análisis clasificado como oportunidad de inversión y el servicio web es accesible de forma remota

Garantías en caso de éxito: La oportunidad de inversión se almacenará en la BBDD

Escenario principal de éxito:

1) El administrador invoca la operación de grabación de un análisis de forma remota, mediante una llamada al servicio web sobre HTTP y una URI concreta

2) El servicio web recibe, codifica y graba el análisis retornando el identificador del análisis

3) El administrador guarda el análisis en su lista de oportunidades de inversión pendientes, en el que incluye el código de análisis recibido del servicio web

Extensiones:

1a. El administrador invoca la operación mediante una llamada al servicio web y este no está disponible, por lo que se da un mensaje de error general del sistema y el caso de uso finaliza

1b. El administrador invoca la operación mediante una llamada al servicio web y hay un error con la integridad de los datos, por lo que se da un mensaje de error acceso a los datos del sistema y el caso de uso finaliza

Caso de uso: Solicitar una oportunidad de inversión

Actor principal: Agente con rol de operador

Ámbito: Sistema multiagente

Nivel de objetivo: Operativa de inversión

Stakeholders e intereses:

El operador solicita al administrador si tiene conocimiento de alguna oportunidad de inversión

Precondición: -

Garantías en caso de éxito: El operador envía una solicitud (mensaje FIPA-ACL del tipo QUERY) de oportunidad de inversión al administrador

Escenario principal de éxito:

1) El operador envía una solicitud de nueva oportunidad de inversión y el administrador la recibe

Extensiones:

1a. El operador envía una solicitud al administrador pero este no la recibe, por lo que se da un mensaje de error general del sistema y el caso de uso finaliza

Caso de uso: Informar de una oportunidad de inversión (búsqueda de análisis positivos)

Actor principal: Agente con rol de administrador

Ámbito: Sistema multiagente

Nivel de objetivo: Operativa de inversión

Stakeholders e intereses:

El administrador informa al operador de la oportunidad de inversión de mayor prioridad que conozca

Precondición: El administrador recibe una solicitud de oportunidad de inversión de un operador

Garantías en caso de éxito: El administrador envía al operador un informe (mensaje FIPA-ACL del tipo INFORM) con la oportunidad de inversión más prioritaria

Escenario principal de éxito:

1) El administrador busca en su lista de oportunidades de inversión pendientes la de mayor prioridad: de entre las marcadas como prioritarias (volumen creciente) la de mayor distancia entre la resistencia y el soporte más relevantes

2) El administrador quita el análisis de la lista de oportunidades de inversión pendientes y lo incorpora a la lista de oportunidades entregadas

3) El administrador envía un informe al operador con la oportunidad de inversión

Extensiones:

1a. El operador no encuentra ninguna oportunidad de inversión, se lo comunica al operador y el caso de uso finaliza

3a. El administrador envía un informe al administrador pero este no lo recibe, por lo que se da un mensaje de error general del sistema

Caso de uso: Abrir una operación de inversión

Actor principal: Agente con rol de operador

Ámbito: Sistema multiagente

Nivel de objetivo: Operativa de inversión

Stakeholders e intereses:

El operador abre una operación de inversión para reconocer si la aplicación de la función de análisis y su parametrización producen un rendimiento positivo

Precondición: El operador recibe una posibilidad de inversión del administrador

Garantías en caso de éxito: El operador envía un informe (mensaje FIPA-ACL del tipo INFORM) de operación de inversión abierta al administrador

Escenario principal de éxito:

1) El operador invoca al servicio web solicitando la cotización actual del valor analizado

2) El operador calcula el precio de salida incrementando un 3% el precio actual del valor y además la resistencia más relevante es mayor que el precio de salida calculado

3) El operador calcula el precio stoploss disminuyendo un 1% el soporte más relevante

Extensiones:

1a. El operador invoca al servicio web y este no está disponible, por lo que se da un mensaje de error general del sistema y el caso de uso finaliza

2a. El operador calcula el precio de salida y la resistencia relevante es menor que el precio calculado, por lo que el operador rechaza el análisis y el caso de uso finaliza

Caso de uso: Cerrar una operación de inversión

Actor principal: Agente con rol de operador

Ámbito: Sistema multiagente

Nivel de objetivo: Operativa de inversión

Stakeholders e intereses:

El operador cierra una operación de inversión por cumplirse la condición de salida establecida

Precondición: El operador ha abierto una operación de inversión

Garantías en caso de éxito: El operador envía un informe (mensaje FIPA-ACL del tipo INFORM) de operación de inversión cerrada al administrador

Escenario principal de éxito:

1) El operador se conecta al servicio web solicitando la cotización actual del valor analizado

2) El operador comprueba que el precio actual del valor es mayor que el precio de salida o menor que el stoploss

3) El operador calcula la rentabilidad de la operación

Extensiones:

1a. El operador invoca al servicio web y este no está disponible, por lo que se da un mensaje de error general del sistema y el caso de uso finaliza

2a. El operador comprueba el precio actual del valor y es menor al precio salida y mayor al stoploss, por lo que el operador espera cinco minutos y repite el caso de uso

INVARIANTES Y REGLAS DE DERIVACIÓN DEL DISEÑO RELACIONAL DE LA BASE DE DATOS

Claves

-- Indices: indice
context Indices inv:
self.allInstances()->isUnique(indice)

-- Indices: índice y valor (clave doble)
context Valores inv:
self.allInstances()->forAll(v1,v2 | v1<>v2 implies v1.Indices<>v2.Indices or v1.valor<>v2.valor)

-- Actuales (Cotizaciones): índice y valor (clave doble)
context Cotizaciones inv:
self.allInstances()->forAll(c1,c2 | c1<>c2 implies c1.Indices<>c2.Indices or c1.Valores<>c2.Valores)

-- Historicas (Cotizaciones): índice, valor y fecha (clave triple)
context Historicas inv:
self.allInstances()->forAll(h1,h2 | h1<>h2 implies h1.Indices<>h2.Indices or h1.Valores<>h2.Valores or h1.fecha<>h2.fecha)

-- Agentes: agente
context Agentes inv:
self.allInstances()->isUnique(agente)

-- Analisis: analisisid
context Analisis inv:
self.allInstances()->isUnique(analisisid)

-- EMA: emaid
context EMA inv:
self.allInstances()->isUnique(emaid)

-- Soportes y Resistencias: sopresid
context 'Soportes y Resistencias' inv:
self.allInstances()->isUnique(sopresid)

-- Operaciones: operacionid
context Operaciones inv:
self.allInstances()->isUnique(operacionid)

-- Apertura (Operaciones): compraid
context Apertura inv:
self.allInstances()->isUnique(compraid)

-- Cierres (Operaciones): ventaid
context Cierres inv:
self.allInstances()->isUnique(ventaid)

Atributos obligatorios

-- Atributos obligatorios de la clase Indices
context Indices inv:
self.Descripcion <> "" and self.'Indice externo' <> ""

-- Atributos obligatorios de la clase Valores
context Valores inv:
self.Descripcion <> "" and self.'Nombre externo' <> ""

-- Atributos obligatorios de la clase Cotizaciones y sus subclases
context Cotizaciones inv:
self.Precio > 0 and self.Volumen > 0

context Actuales inv:
self.Diferencia > 0 and self.Variacion > 0

context Actuales inv:
self.Apertura > 0 and self.Maximo > 0 and self.Minimo > 0

-- Atributos obligatorios de la clase Analisis y sus subclases
context Analisis inv:
self.Precio > 0 and self.Volumen > 0

context EMA inv:
self.Periodo > 0

context Precios inv:
self.'Media Actual' > 0 and self.'Media Anterior' > 0

context Volumenes inv:
self.'Media Actual' > 0 and self.'Media Anterior' > 0

context MACD inv:
self.EMA12 > 0 and self.EMA26 > 0 and self.MACD > 0 and self.Señal > 0 and self.MACDh > 0

context 'Soportes y Resistencias' inv:
self.Soporte > 0 and self.Resistencia > 0 and self.Tipo <> ""

context Semanales inv:
self.Modelo <> "" and self.Precision > 0

context Semanal inv:
not(self.Maximos.oclIsUndefined())

context Semanal inv:
not(self.Minimos.oclIsUndefined())

context Semanal inv:
not(self.Niveles.oclIsUndefined())

context Diarios inv:
self.Modelo <> "" and self.Precision > 0

context xTrimestre inv:
not(self.Maximos.oclIsUndefined())

context xTrimestre inv:
not(self.Minimos.oclIsUndefined())

context xTrimestre inv:
not(self.Niveles.oclIsUndefined())

-- Atributos obligatorios de la clase Operaciones y sus subclases
context Apertura inv:
self.Entrada > 0 and self.Salida > 0 and self.Stoploss > 0

context Cierre inv:
self.Precio > 0 and self.Rentabilidad > 0

context Diarios inv:
self.Nombre <> "" and self.Tipo <> ""

Derivaciones

-- Regla derivación atributos: Diferencia y Variacion de la clase Precios
context Precios::Diferencia:Money
derive: self.'Media Actual' - self.'Media Anterior'

context Precios::Variacion:Real
*derive: (self.'Media Actual' - self.'Media Anterior') / self.'Media Anterior' * 100*

-- Regla derivación atributos: Diferencia y Variacion de la clase Volumenes
context Volumenes::Diferencia:Money
derive: self.'Media Actual' - self.'Media Anterior'

context Volumenes::Variacion:Real
*derive: (self.'Media Actual' - self.'Media Anterior') / self.'Media Anterior' * 100*

-- Regla derivación atributo: Variacion de la clase MACD
context MACD::Variacion:Real
*derive: (self.'Media Actual' - self.'Media Anterior') / self.'Media Anterior' * 100*

-- Regla derivación atributos: Diferencia y Variacion de la clase Soportes y Resistencias
context 'Soportes y Resistencias'::Diferencia:Money
derive: self.Resistencia - self.Soporte

context 'Soportes y Resistencias'::Variacion:Real
*derive: (self.Resistencia - self.Soporte) / self.Soporte * 100*

Definiciones

-- Enumeración Tipo Nivel: Semanal = 0; Diario 1T: 1; Diario 2T: 2; Diario 3T: 3
context 'Soportes y Resistencias' def:
tipoNivel = Set{0,1,2,3}

-- Enumeración Tipo Calculo: Media = 0; Media+1: 1
context Semanal def:
tipoCalculo = Set{0,1}

context Diarios def:
tipoCalculo = Set{0,1}

-- Enumeración Tipo Agente: Administrador = 0; Analista = 1; Operador = 2
context Agentes def:
tipoAgente = Set{0,1,2}

Otras restricciones

-- Atributos tipoNivel
context 'Soportes y Resistencias' inv:
self.Tipo.oclIsTypeOf(tipoNivel)

-- Atributos tipoCalculo
context Semanal inv:
self.Modelo.oclIsTypeOf(tipoCalculo)

context Diarios inv:
self.Modelo.oclIsTypeOf(tipoCalculo)

-- Atributos tipoAgentel
context Agente inv:
self.Tipo.oclIsTypeOf(tipoAgente)

-- Una compra/venta se identifica por el analisisid común
context Cierre inv:
self.Operaciones->forAll(op | op.oclIsTypeOf(Apertura) implies op.Analisis = self.Analisis)

-- El contenido de un mensaje es un Analisis o una Operacion
context Mensajes inv:
self.Contenido.oclIsTypeOf(Analisis) or self.Contenido.oclIsTypeOf(Operaciones)

METRICAS DEL SERVICIO WEB

Paquete ejb

Category/Metric	Value
Count	
Units	10
Classes / Class	0
Methods / Class	6.5
Fields / Class	0.5
ELOC	914
ELOC / Unit	91.4
Complexity	
CC	2.43
Fat	5
ACD - Unit	5.56%
Chidamber & Kemerer	
WMC	15.8
DIT	0.5
NOC	0
CBO	3.7
RFC	29.8
LCOM	6.4

Componente WSAnalisis

Category/Metric	Value
Count	
Classes	0
Methods	16
Fields	1
ELOC	285
Complexity	
Fat	25
Chidamber & Kemerer	
WMC	43
DIT	1
NOC	0
CBO	15
RFC	102
LCOM	0

Componente WSCotizaciones

Category/Metric	Value
Count	
Classes	0
Methods	13
Fields	1
ELOC	242
Complexity	
Fat	14
Chidamber & Kemerer	
WMC	51
DIT	1
NOC	0
CBO	3
RFC	60
LCOM	48

Componente WSMétodos

Category/Metric	Value
Count	
Classes	0
Methods	2
Fields	1
ELOC	28
Complexity	
Fat	1
Chidamber & Kemerer	
WMC	4
DIT	1
NOC	0
CBO	2
RFC	13
LCOM	1

Componente WSOperaciones

Category/Metric	Value
Count	
Classes	0
Methods	14
Fields	1
ELOC	242
Complexity	
Fat	22
Chidamber & Kemerer	
WMC	38
DIT	1
NOC	0
CBO	14
RFC	86
LCOM	0

Componente WSValores

Category/Metric	Value
Count	
Classes	0
Methods	7
Fields	1
ELOC	94
Complexity	
Fat	8
Chidamber & Kemerer	
WMC	22
DIT	1
NOC	0
CBO	3
RFC	24
LCOM	15

METRICAS DEL SISTEMA MULTIAGENTE

Agente administrador

Category/Metric	Value
Count	
Classes	3
Methods	8
Fields	11
ELOC	276
Complexity	
Fat	26
Chidamber & Kemerer	
WMC	19
DIT	1
NOC	0
CBO	14
RFC	39
LCOM	14

Agente analista

Category/Metric	Value
Count	
Classes	4
Methods	4
Fields	14
ELOC	244
Complexity	
Fat	32
Chidamber & Kemerer	
WMC	5
DIT	1
NOC	0
CBO	12
RFC	13
LCOM	6

Agente operador

Category/Metric	Value
Count	
Classes	3
Methods	7
Fields	19
ELOC	264
Complexity	
Fat	31
Chidamber & Kemerer	
WMC	14
DIT	1
NOC	0
CBO	11
RFC	19
LCOM	11

Controlador de creencias

Category/Metric	Value
Count	
Classes	0
Methods	4
Fields	10
ELOC	113
Complexity	
Fat	18
Chidamber & Kemerer	
WMC	35
DIT	1
NOC	0
CBO	6
RFC	22
LCOM	4

Controlador de deseos

Category/Metric	Value
Count	
Classes	0
Methods	6
Fields	8
ELOC	110
Complexity	
Fat	7
Chidamber & Kemerer	
WMC	32
DIT	1
NOC	0
CBO	6
RFC	24
LCOM	1

Controlador de intenciones

Category/Metric	Value
Count	
Classes	0
Methods	2
Fields	0
ELOC	41
Complexity	
Fat	0
Chidamber & Kemerer	
WMC	12
DIT	1
NOC	0
CBO	3
RFC	8
LCOM	1

Componente de creencias

Category/Metric	Value
Count	
Classes	0
Methods	15
Fields	10
ELOC	370
Complexity	
Fat	14
Chidamber & Kemerer	
WMC	67
DIT	1
NOC	0
CBO	4
RFC	31
LCOM	6

Componente de deseos

Category/Metric	Value
Count	
Classes	0
Methods	5
Fields	4
ELOC	77
Complexity	
Fat	1
Chidamber & Kemerer	
WMC	19
DIT	1
NOC	0
CBO	3
RFC	13
LCOM	10

Estado de validación del operador

Category/Metric	Value
Count	
Classes	0
Methods	6
Fields	8
ELOC	98
Complexity	
Fat	12
Chidamber & Kemerer	
WMC	29
DIT	1
NOC	0
CBO	7
RFC	18
LCOM	0

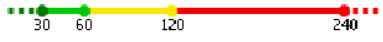
FIPA

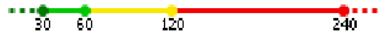
Category/Metric	Value
Count	
Classes	0
Methods	4
Fields	0
ELOC	29
Complexity	
Fat	0
Chidamber & Kemerer	
WMC	7
DIT	1
NOC	0
CBO	3
RFC	4
LCOM	0

Comunicaciones WS

Category/Metric	Value
Count	
Classes	0
Methods	17
Fields	0
ELOC	261
Complexity	
Fat	16
Chidamber & Kemerer	
WMC	50
DIT	1
NOC	0
CBO	5
RFC	23
LCOM	0

VALORES POR DEFECTO DE LAS MÉTRICAS

Count Metrics	Rating	Linear
Number of Top Level Classes (Units)		✓
Number of Methods		✓
Number of Fields		✓
Estimated Lines of Code (ELOC)		✓
Estimated Lines of Code (ELOC)		✓

Complexity Metrics	Rating	Linear
Cyclomatic Complexity (CC)		✓
Fat		✓
Fat		✓
Fat		✓
Tangled		✓
Tangled for Library Dependencies (Tangled - Libraries)		✓
Tangled for Flat Package Dependencies (Tangled - Packages)		✓
Average Component Dependency between Libraries (ACD - Library)		✓
Average Component Dependency between Packages (ACD - Package)		✓
Robert C. Martin Metrics	**Rating**	**Linear**
Distance (D)		✓
Average Absolute Distance		✓
Chidamber & Kemerer Metrics	**Rating**	**Linear**
Weighted Methods per Class (WMC)		✓
Depth of Inheritance Tree (DIT)		✓
Average Depth of Inheritance Tree		✓
Coupling between Objects (CBO)		✓
Response for a Class (RFC)		✓

INSTRUCCIONES SQL - CREACIÓN TABLAS Y DATOS POR DEFECTO

```
CREATE SCHEMA ibexdata
   AUTHORIZATION "USER";

GRANT ALL ON SCHEMA ibexdata TO "USER";

CREATE TABLE ibexdata.indices(
   indice text not null,
   descripcion text not null,
   referencia text not null,
   CONSTRAINT "indices_pkey" PRIMARY KEY (indice));

INSERT INTO ibexdata.indices(indice, descripcion, referencia)
   values ('IBEX', 'Mercado Continuo de la bolsa española', 'IB011CONTINU');

CREATE TABLE ibexdata.valores(
   indice text not null,
   valor text not null,
   descripcion text not null,
   referencia text not null,
   CONSTRAINT "valores_pkey" PRIMARY KEY (indice,valor));

CREATE TABLE ibexdata.cotizaciones(
   indice text not null,
   valor text not null,
   precio real not null,
   diferencia real not null,
   variacion numeric(5,2) not null,
   volumen bigint not null,
   fecha date,
   hora time,
   CONSTRAINT "cotizaciones_pkey" PRIMARY KEY (indice, valor));

CREATE TABLE ibexdata.historico(
   indice text not null,
   valor text not null,
   fecha date not null,
   ultimo real not null,
   apertura real not null,
   maximo real not null,
   minimo real not null,
   volumen bigint not null,
   CONSTRAINT "historico_pkey" PRIMARY KEY (indice, valor,fecha));

CREATE TABLE ibexdata.agentes(
   agente text not null,
   tipo smallint not null,
   CONSTRAINT "agentes_pkey" PRIMARY KEY (agente));

CREATE TABLE ibexdata.analisis(
   analisisid serial not null,
   precio real not null,
   volumen bigint not null,
   macd real not null,
   senal real not null,
   macdh real not null,
   soporte real not null,
   resistencia real not null,
   nivel smallint not null,
   prioridad smallint not null,
   indice text not null,
   valor text not null,
   agente text not null,
   fecha date not null,
```

```
    hora time not null,
    CONSTRAINT "analisis_pkey" PRIMARY KEY (analisisid));


CREATE TABLE ibexdata.ema(
    emaid serial not null,
    tipo smallint not null,
    clase smallint not null,
    periodo smallint not null,
    media real not null,
    analisisid int not null,
    CONSTRAINT "ema_pkey" PRIMARY KEY (emaid));

CREATE TABLE ibexdata.soportesresistencias(
    sopresid serial not null,
    modelo smallint not null,
    precision numeric(5,2) not null,
    tipo smallint not null,
    analisisid int not null,
    CONSTRAINT "soportesresistencias_pkey" PRIMARY KEY (sopresid));

CREATE TABLE ibexdata.maximos(
    maximoid serial not null,
    precio real not null,
    sopresid int not null,
    CONSTRAINT "maximos_pkey" PRIMARY KEY (maximoid));

CREATE TABLE ibexdata.minimos(
    minimoid serial not null,
    precio real not null,
    sopresid int not null,
    CONSTRAINT "minimos_pkey" PRIMARY KEY (minimoid));

CREATE TABLE ibexdata.niveles(
    nivelid serial not null,
    precio real not null,
    sopresid int not null,
    CONSTRAINT "niveles_pkey" PRIMARY KEY (nivelid));

CREATE TABLE ibexdata.operaciones(
    operacionid serial not null,
    indice text not null,
    valor text not null,
    analisisid int not null,
    agente text not null,
    CONSTRAINT "operaciones_pkey" PRIMARY KEY (operacionid));

CREATE TABLE ibexdata.compras(
    compraid serial not null,
    entrada real not null,
    salida real not null,
    stoploss real not null,
    fecha date not null,
    hora time not null,
    operacionid int not null,
    CONSTRAINT "compras_pkey" PRIMARY KEY (compraid));

CREATE TABLE ibexdata.ventas(
    ventaid serial not null,
    cierre real not null,
    diferencia real not null,
    rentabilidad numeric(5,2) not null,
    fecha date not null,
    hora time not null,
    operacionid int not null,
    CONSTRAINT "ventas_pkey" PRIMARY KEY (ventaid));
```

```
CREATE TABLE ibexdata.metodos(
    metodo text not null,
    tipo smallint not null,
    ambito smallint not null,
    relacion smallint not null,
    valor float not null,
    CONSTRAINT "metodos_pkey" PRIMARY KEY (metodo));

INSERT INTO ibexdata.metodos(metodo, tipo, ambito, relacion, valor)
    values ('EMA PRECIO', 0, 0, 1, 0.0);
INSERT INTO ibexdata.metodos(metodo, tipo, ambito, relacion, valor)
    values ('HISTOGRAMA MACD', 0, 0, 1, 0.0);
INSERT INTO ibexdata.metodos(metodo, tipo, ambito, relacion, valor)
    values ('RANGO ENTRE NIVELES', 0, 0, 2, 6.0);
INSERT INTO ibexdata.metodos(metodo, tipo, ambito, relacion, valor)
    values ('EMA VOLUMEN', 1, 0, 1, 0.0);
INSERT INTO ibexdata.metodos(metodo, tipo, ambito, relacion, valor)
    values ('APERTURA OPERACION', 0, 1, 2, 3.0);

CREATE TABLE ibexdata.parametros(
    parametro text not null,
    valor float not null,
    metodo text not null,
    CONSTRAINT "parametros_pkey" PRIMARY KEY (parametro));

INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('NUMERO SESIONES EMA PRECIO', 8.0, 'EMA PRECIO');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('NUMERO SESIONES EMA CORTA MACD', 12.0, 'HISTOGRAMA MACD');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('NUMERO SESIONES EMA LARGA MACD', 26.0, 'HISTOGRAMA MACD');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('NUMERO SESIONES SENAL MACD', 9.0, 'HISTOGRAMA MACD');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('COMBINACION MAXIMOS Y MINIMOS', 1111.0, 'RANGO ENTRE NIVELES');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('PRECISION DEL RANGO', 1.0, 'RANGO ENTRE NIVELES');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('MODELO SELECCION DE NIVEL', 1.0, 'RANGO ENTRE NIVELES');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('NUMERO SESIONES EMA VOLUMEN', 8.0, 'EMA VOLUMEN');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('VARIACION PRECIO SALIDA', 3.0, 'APERTURA OPERACION');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('VARIACION STOPLOSS', -1.0, 'APERTURA OPERACION');
INSERT INTO ibexdata.parametros(parametro, valor, metodo)
    values ('VARIACION STOPLOSS MAXIMA', -5.0, 'APERTURA OPERACION');

CREATE TABLE ibexdata.metodologia(
    metodoid serial not null,
    metodo text not null,
    tipo smallint not null,
    ambito smallint not null,
    relacion smallint not null,
    valor float not null,
    analisisid int,
    operacionid int,
    CONSTRAINT "metodologia_pkey" PRIMARY KEY (metodoid));

CREATE TABLE ibexdata.parametrizacion(
    paramid serial not null,
    parametro text not null,
    valor float not null,
    metodoid int not null,
    CONSTRAINT "parametrizacion_pkey" PRIMARY KEY (paramid));
```

INSTRUCCIONES SQL – BORRADO DATOS E IMPORTACIÓN HISTORICO

Borrado de datos de pruebas

```
DELETE FROM ibexdata.parametrizacion;
DELETE FROM ibexdata.metodologia;
DELETE FROM ibexdata.minimos;
DELETE FROM ibexdata.maximos;
DELETE FROM ibexdata.niveles;
DELETE FROM ibexdata.soportesresistencias;
DELETE FROM ibexdata.ema;
DELETE FROM ibexdata.ventas;
DELETE FROM ibexdata.compras;
DELETE FROM ibexdata.operaciones;
DELETE FROM ibexdata.analisis;
```

Importación datos históricos

```
COPY ibexdata.historico FROM 'C:\historico.csv' DELIMITER ';' CSV HEADER;
```

(Nota: Cambiar ruta por carpeta contenedora del fichero)

GUÍA DE CONFIGURACIÓN DEL SOFTWARE

Esta guía es un breve resumen de las especificaciones descritas en el manual de configuración del software de las asignaturas *Ingeniería del software de componentes y sistemas distribuidos* y *Proyecto de desarrollo de software*.

Enlaces descarga del software

SDK Java
https://www.oracle.com/technetwork/java/javase/downloads/index.html

Apache Ant
http://apache.rediris.es//ant/binaries/apache-ant-1.10.1-bin.zip

PostgreSQL
https://www.postgresql.org/download

Jboss
http://download.jboss.org/wildfly/10.1.0.Final/wildfly-10.1.0.Final.zip
https://jdbc.postgresql.org/download/postgresql-9.4.1209.jar

Jade
http://jade.tilab.com/download/jade/license/jade-download/

Eclipse
http://www.eclipse.org/downloads/
http://www.jboss.org/tools/download.html

Roles de usuario y contraseñas

PostgreSQL
Usuario: USER
Contraseña: PASSWORD

Jboss
Tipo Usuario: ApplicationRealm
Usuario: USER
Contraseña: PASSWORD

Configuración conector PostgreSQL para JBoss

1. Crear el fichero JBOSS_HOME\modules\system\layers\base\org \postgresql\main\module.xml con el siguiente contenido:

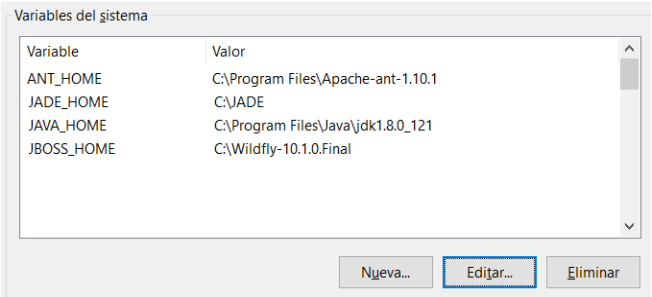
```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="org.postgresql">
    <resources>
        <resource-root path="postgresql-9.4.1209.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
   </dependencies>
</module>
```

2. Modificar el fichero JBOSS_HOME\standalone\configuration\standalone.xml para que la etiqueta *datasources* quede con el siguiente contenido:

```
<datasources>
   <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enabled="true" use-
java-context="true">
       <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
       <driver>h2</driver>
       <security>
           <user-name>sa</user-name>
           <password>sa</password>
       </security>
    </datasource>
   <datasource jta="false" jndi-name="java:jboss/postgresDS" pool- name="postgresDS" enabled="true" use-java-
context="true" use-ccm="false">
       <connection-url>jdbc:postgresql://localhost:5432/postgres</connection-url>
       <driver-class>org.postgresql.Driver</driver-class>
       <driver>postgresql</driver>
       <pool/>
       <security>
           <user-name>USER</user-name>
           <password>PASSWORD</password>
       </security>
       <statement/>
    </datasource>
    <drivers>
       <driver name="h2" module="com.h2database.h2">
           <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
       </driver>
       <driver name="postgresql" module="org.postgresql">
           <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource- class>
       </driver>
    </drivers>
 </datasources>
```

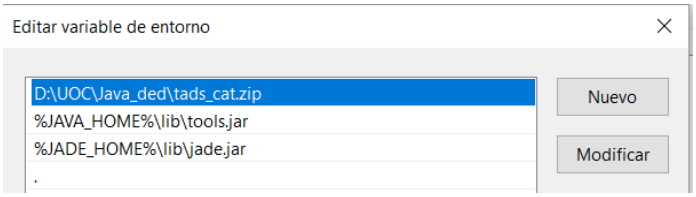
Variables del sistema

JAVA_HOME: contiene el directorio de instalación del JDK
ANT_HOME: contiene el directorio de instalación de Apache Ant
JBOSS_HOME: contiene el directorio de instalación de JBoss
JADE_HOME: contiene el directorio con los ficheros del framework JADE

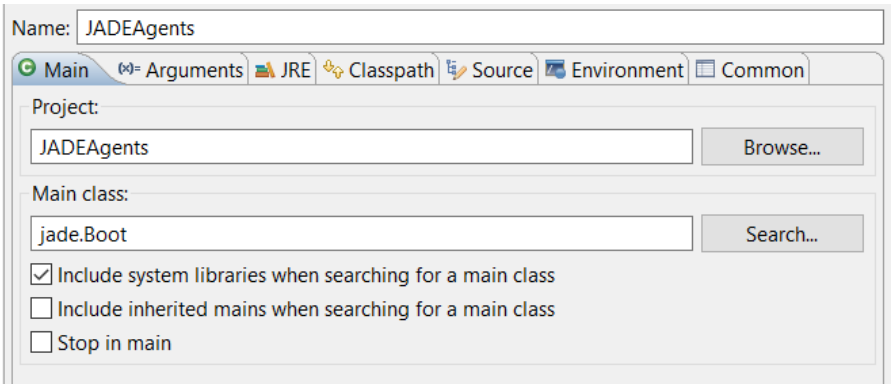


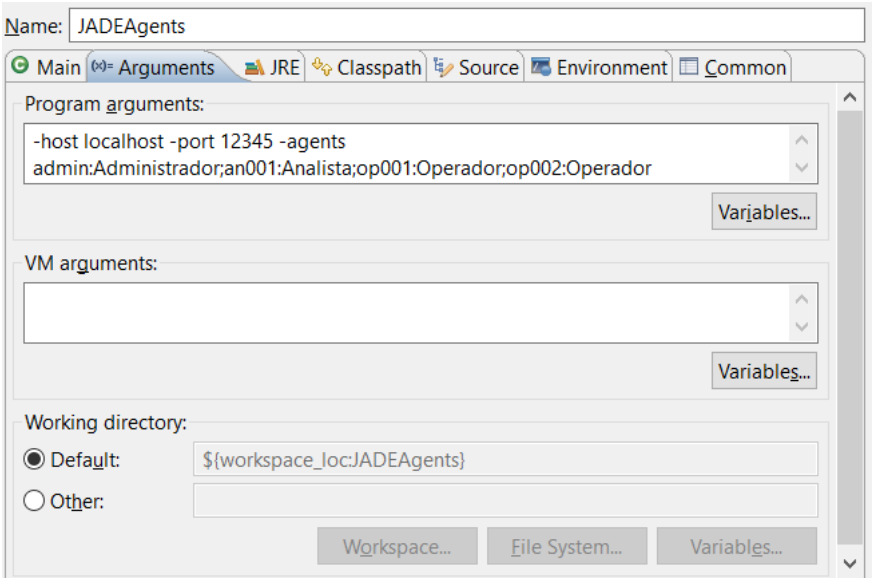
PATH: contiene el directorio bin de todos los directorios anteriores

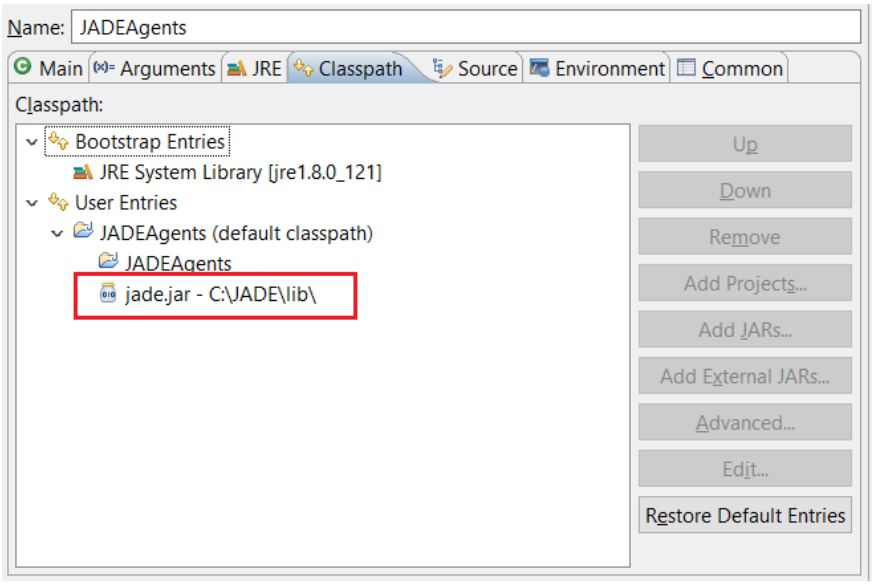
CLASSPATH: contiene una referencia a la carpeta de trabajo y los ficheros tools.jar del JDK y jade.jar del JADE



ENTORNO DE COMPILACIÓN/EJECUCIÓN DE JADE EN ECLIPSE







SOFTWARE DE MEDICIÓN DE MÉTRICAS DE CALIDAD

STAN 2.1 - Structure Analysis for Java

http://stan4j.com/download/app/
http://download.stan4j.com/app/stan-app-help-2.2.pdf

SONARQUBE 7.4: Server & Scanner para Windows10

https://www.sonarqube.org/downloads/
https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner

Configuración servidor

1. Iniciar servidor Sonarqube:
   C:\Metrics\sonarqube-7.4\bin\windows-x86-64\StartSonar.bat
2. Acceder interface web:
   http://localhost:9000/projects
3. Administrar proyectos:
   Usuario: admin
   Contraseña: admin

Configuración scanner

1. Configurar scanner: En carpeta raíz de la solución crear fichero sonar-project.properties (por ejemplo, C:\IbexAgentsWS\sonar-project.properties) con el siguiente contenido:

```
# must be unique in a given SonarQube instance
sonar.projectKey=IbexAgentsWS
# this is the name and version displayed in the SonarQube UI. Was mandatory prior to SonarQube 6.1.
sonar.projectName=IbexAgentsWS
sonar.projectVersion=1.0
# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
sonar.sources=./src
sonar.java.binaries=./bin
```

2. Modificar el fichero conf\sonar-scanner.properties que se encuentra en la carpeta donde se ha instalado el scanner:

```
#----- Default SonarQube
serversonar.host.url=http://localhost:9000
```

3. Modificar el fichero bin\sonar-scanner.bat que se encuentra en la carpeta de instalación del scanner, indicando la dirección del proyecto a analizar:

```
set PROJECT_HOME=C:\IbexAgentsWS
```

4. Ejecutar el scanner mediante el fichero de lotes *bin\sonar-scanner.bat* y refrescar la interface web del servidor

BIBLIOGRAFÍA

Recursos bibliográficos

IMMANUEL KANT. (2017) Crítica de la razón pura: Editorial Gredos
LUDWIG VON MISSES (2011) La acción humana: Unión Editorial
RENÉ DESCARTES (2017) Discurso del método: Alianza Editorial
FABIO BELLIFEMINE, GIOVANNI CAIRE, DOMINIC GREENWOOD (2004) Developing Multi-Agent Systems with JADE: John Wiley & Sons

Recursos didácticos

UOC Aprendizaje Computacional: Agentes y sistemas multiagentes
UOC Ingeniería de requisitos: Gestión de requisitos
UOC Ingeniería de requisitos: Documentación de requisitos
UOC Análisis y diseño con patrones: Catálogo de patrones
UOC Ingeniería del software de componentes y sistemas distribuidos: Diseño de aplicaciones distribuidas
UOC Ingeniería del software de componentes y sistemas distribuidos: Arquitectura del software
UOC Sistemas integrados: El software
UOC Ingeniería del software de componentes y sistemas distribuidos: Desarrollo de software basado en componentes
UOC Mercados y conducta: La empresa
UOC Mercados y conducta: Estructura de mercado
UOC Proyecto de desarrollo de software: Calidad del software: gestión de la calidad y métricas
UOC Proyecto de desarrollo de software: Gestión de la configuración del software
UOC Proyecto de desarrollo de software: Calidad del software: técnicas de prevención, detección y corrección de defectos

Recursos web

http://www.lsi.upc.edu/~bejar/ecsdi/Teoria/ECSDI03a-AOSE.pdf
http://www.fdi.ucm.es/profesor/jpavon/doctorado/metodologias.pdf
https://es.wikipedia.org/wiki/La_acción_humana
https://es.wikipedia.org/wiki/Praxeología
https://en.wikipedia.org/wiki/Belief–desire–intention_software_model
http://dle.rae.es/srv/fetch?id=6nVpk8P
https://es.wikipedia.org/wiki/Ciclo_de_vida_del_lanzamiento_de_software

Otros recursos web

https://www.novatostradingclub.com/
https://www.invertia.com/es/portada
https://www.iro.umontreal.ca/~vaucher/Agents/Jade/
https://stackoverflow.com
http://stan4j.com/
http://www.sonarqube.org