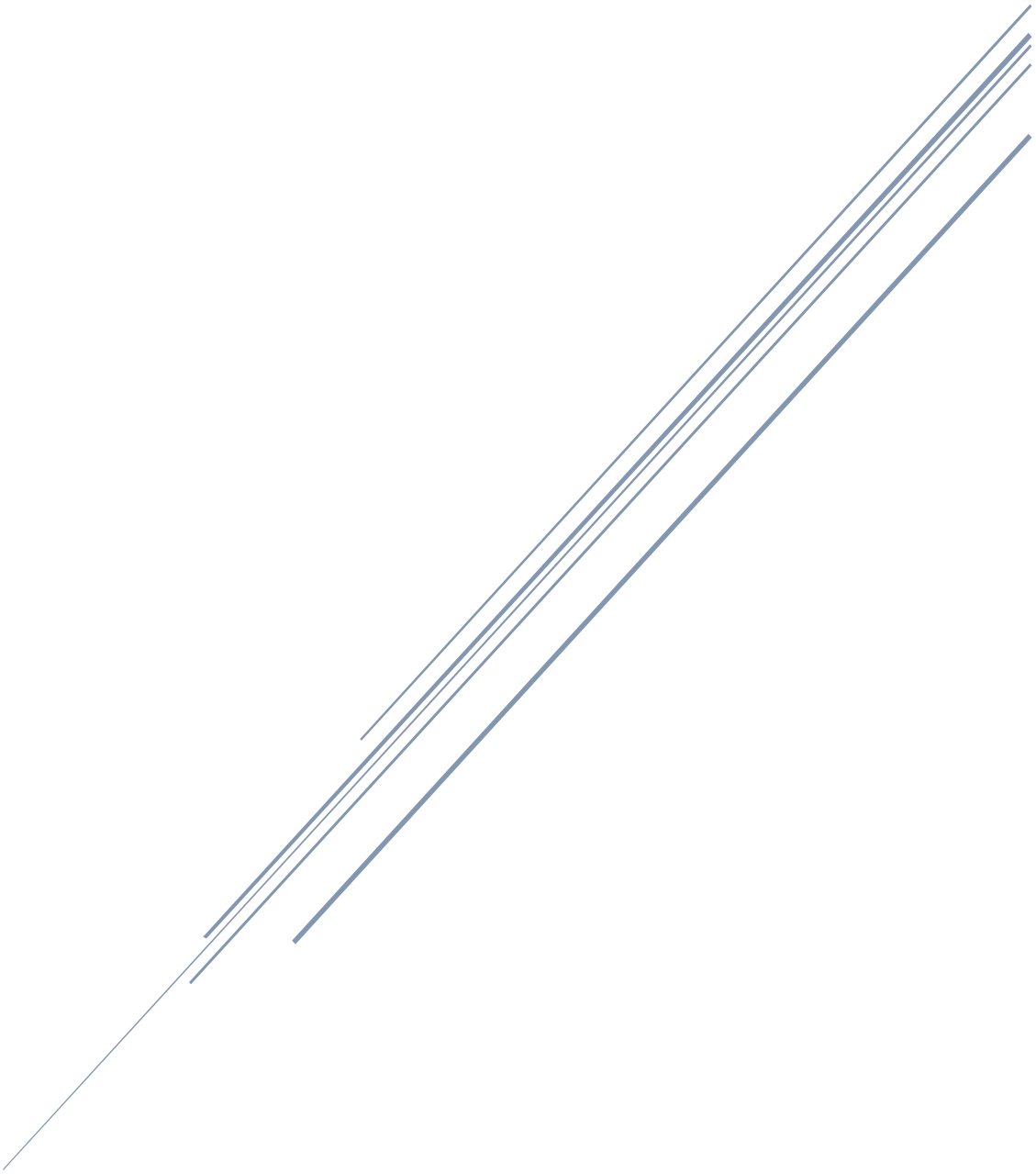
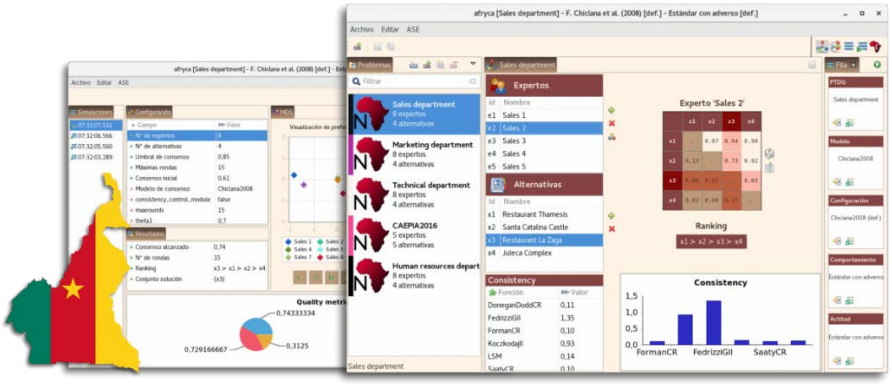
DOCUMENTACIÓN AFRYCA



1. Contenido

1. Introducción



1-Interfaz de AFRYCA

AFRYCA (acrónimo de **A FR**amework for the anal**Y**sis of **C**onsensus **A**pproaches) es un framework enfocado en la investigación de los **Procesos de Alcance de Consenso (PAC)** en la resolución de problemas de **Toma de Decisión en Grupo (TDG)**.

AFRYCA está desarrollado como una aplicación **Eclipse Rich Client Platform (Eclipse RCP)** de la rama **E4**, lo cual redunda en múltiples beneficios tanto para el usuario final como para el desarrollador:

- Arquitectura orientada a componentes fácil de extender y adaptar.
- Multiplataforma, con interfaz de usuario nativa para cada sistema.
- Ejecución en contenedor de inyección de dependencias.
- Soporte para preferencias.
- Diseño CSS.

El framework viene equipado con diferentes modelos de PACs propuestos en la literatura científica, así como con diversos patrones de comportamientos de expertos con los que llevar a cabo la simulación y el análisis de un PAC en TDG.

AFRYCA integra **ASE** y **BIRT**, los cuales lo convierten en una potente herramienta de análisis para los usuarios.

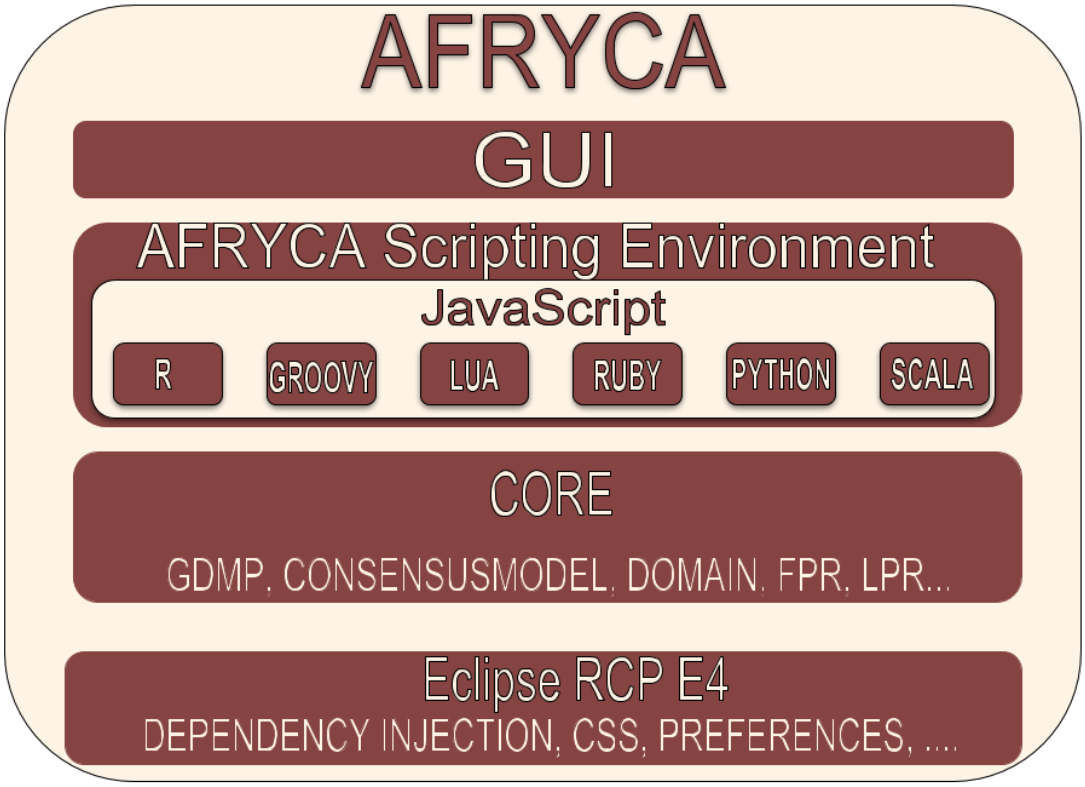
ASE (acrónimo de **AFRYCA S**cripting **E**nvironment) es un entorno de scripting construido sobre la especificación **JSR 223: Scripting for the JavaTM Platform**. ASE permite incorporar cualquier nueva funcionalidad mediante el uso de fragmentos de código en tiempo de ejecución, lo cual deja total libertad al usuario para analizar los PACs. El lenguaje nativo del entorno es **JavaScript** en su implementación **Nashorn**, y en él, ASE añade entornos para programar en **Groovy, Scala, Ruby, Python y Lua.** Adicionalmente, si el sistema donde se ejecuta AFRYCA cuenta con una instalación del entorno estadístico **R**, ASE facilita la integración con él para utilizarlo como un entorno de programación más.

BIRT (acrónimo de **B**usiness **I**ntelligence and **R**eporting Tools) es un proyecto Eclipse para la generación de gráficos e informes. El uso combinado de ASE y BIRT brinda al usuario la posibilidad de crear, en tiempo de ejecución, gráficos a medida con los que representar los datos obtenidos.

La versión **4.0** supone un punto de inflexión al añadir la posibilidad de trabajar con diferentes relaciones de preferencia, lo que lo hace uno de los softwares más completos en el ámbito de la toma de decisión en grupo y procesos de alcance de consenso.

2. Tecnologías

La arquitectura de AFRYCA está compuesta por diferentes tecnologías, éstas son Eclipse RCP en la rama E4 con la que esta programada la payor parte de su código, por otro lado, tenemos mediante ASE (acrónimo de AFRYCA Scripting Enviroment) la posibilidad de ejecutar código de diferentes lenguajes de programación basado en un entorno nativo de JavaScript, estos lenguajes son: Groovy, Lua, Ruby, Python Scala y R.



2-Tecnologías

Eclipse RCP E4: Plataforma para construir herramientas basadas en Eclipse y aplicaciones de escritorio. Esta nueva versión facilita a los desarrolladores ensamblar aplicaciones y herramientas basadas en la plataforma de eclipse. Presenta una interfaz basada en modelos y un nuevo mecanismo declarativo basado en CSS para el diseño de las aplicaciones. Estas instalaciones facilitan el diseño y la personalización de la interfaz de usuario de la aplicación.

JavaScript: Lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas4 aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

Groovy: Lenguaje de programación orientado a objetos implementado sobre la plataforma Java. Tiene características similares a Python, Ruby, Perl y Smalltalk. La especificación JSR 241 se encarga de su estandarización para una futura inclusión como componente oficial de la plataforma Java. Usa una sintaxis muy parecida a Java, comparte el mismo modelo de objetos, de hilos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java. El bytecode generado en el

proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto, puede usarse directamente en cualquier aplicación Java. Todo lo anterior unido a que la mayor parte de código escrito en Java es totalmente válido en Groovy hacen que este lenguaje sea de muy fácil adopción para programadores Java; la curva de aprendizaje se reduce mucho en comparación con otros lenguajes que generan bytecode para la JVM, tales como Jython o JRuby. Groovy puede usarse también de manera dinámica como un lenguaje de scripting.

Scala: Lenguaje de programación multi-paradigma diseñado para expresar patrones comunes de programación en forma concisa, elegante y con tipos seguros. Integra sutilmente características de lenguajes funcionales y orientados a objetos. La implementación actual corre en la máquina virtual de Java y es compatible con las aplicaciones Java existentes.

Ruby: Lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995. Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU. Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre.

Python: Lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,1 que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Lua: Lenguaje de programación extensible diseñado para una programación procedimental general con utilidades para la descripción de datos. También ofrece un buen soporte para la programación orientada a objetos, programación funcional y programación orientada a datos. Se pretende que Lua sea usado como un lenguaje de script potente y ligero para cualquier programa que lo necesite. Lua está implementado como una biblioteca escrita en C limpio (esto es, en el subconjunto común de ANSI C y C++).

R: Entorno y lenguaje de programación con un enfoque al análisis estadístico. Es una implementación de software libre del lenguaje S pero con soporte de alcance estático. Se trata de uno de los lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy popular en el campo de la minería de datos, la investigación biomédica, la bioinformática y las matemáticas financieras. A esto contribuye la posibilidad de cargar diferentes bibliotecas o paquetes con funcionalidades de cálculo y gráficas. R es parte del sistema GNU y se distribuye bajo la licencia GNU GPL. Está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux.

3. La interfaz de AFRYCA y su utilización

3.1. Interfaz

AFRYCA, como aplicación de escritorio basada en Eclipse RCP, hace uso de los paradigmas de interfaz definidos por esta plataforma.

En esta sección se listan los diferentes tipos de elementos usados en la interfaz de AFRYCA.

3.1.1. Perspectivas

Si se observa la interfaz de usuario de cualquier aplicación como una jerarquía de elementos, en la que unos elementos están contenidos dentro de otros, una perspectiva constituiría el nivel más alto en dicha jerarquía.

En AFRYCA se emplean diferentes perspectivas para separar la funcionalidad ofrecida por la aplicación, pudiendo verse cada una de estas perspectivas como un contexto de tareas (definición del problema, configuración del consenso, visualización de resultados, ...).

Las diferentes perspectivas definidas por AFRYCA pueden ser seleccionadas usando el selector de perspectivas, el cual se encuentra en la esquina superior derecha de la interfaz de usuario y proporciona, para cada una de las perspectivas, un acceso directo que permitirá cambiar la perspectiva activa.

3-Selector de perspectivas

Las diferentes perspectivas presentes en AFRYCA, siguiendo el orden mostrado por el selector de perspectivas de izquierda a derecha son:

PTDG: Gestión de Problemas de Toma de Decisión en Grupo (PTDG). Permite realizar tareas como la creación de un PTDG, su selección, modificación, clonación, importación, etc.

Modelos de consenso: Gestión de modelos de consenso y comportamientos a utilizar en caso necesario. Tareas como la selección del modelo, su configuración, la selección de un comportamiento y la gestión de su actitud se realizan desde aquí.

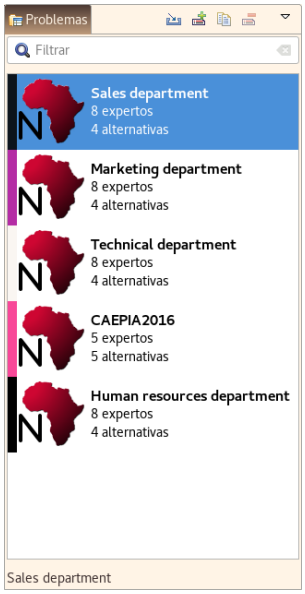
Simulaciones: Visualización de información sobre simulaciones realizadas. Se enfoca en el análisis comparativo de diferentes simulaciones.

Simulación: Visualización del progreso de una simulación. Se enfoca en el estudio del proceso de consenso realizado en una simulación.

ASE: (AFRYCA Scripting Environment) Es el entorno de scripting integrado de AFRYCA. La definición de snippets, funciones, enlaces a módulos, ejecución de fragmentos de código y otras funciones ofrecidas por ASE son gestionadas desde esta perspectiva.

3.1.2. Partes

El siguiente nivel en la jerarquía de elementos de la interfaz son las partes. Las partes son componentes de la interfaz que permiten visualizar información, navegar entre ella y modificarla. En una aplicación Eclipse RCP como AFRYCA, las partes son elementos dinámicos que pueden ser redimensionados y reposicionados para personalizar la interfaz a las preferencias del usuario.



4-Ejemplo de parte: Problemas en perspectiva PTDG

Las partes se dividen gráficamente en tres partes:

Título: Contiene un icono representativo (opcionalmente) y el nombre de la parte, pudiendo adaptarse su nombre en función de diversos factores. Por cuestiones prácticas o de diseño, en algunas ocasiones AFRYCA oculta el título de las partes al usuario.

Menú: Acceso a diferentes comandos relacionados con la parte. Los comandos pueden estar contenidos tanto en una barra de comandos como en un menú de comandos.

Cuerpo: Contenido de la parte.

Las partes pueden a su ver clasificarse como **vistas** y **editores**, estando las primeras más enfocadas en la manipulación directa de la información (los cambios realizados se aplican en el momento) mientras que las segundas se emplean en aquellos casos en los que se busca una edición (los cambios realizados no se aplican hasta que el usuario no lo indica). Si bien el concepto establece una distinción clara, en la práctica la mayoría de las vistas permiten realizar tareas de edición, y algunas tareas llevadas a cabo en los editores no permiten ser editadas y se realizan de forma directa, por lo que para el usuario final puede ser difícil discernir si está empleando una vista o un editor. Por ello, en esta guía para no confundir al usuario nos referiremos en exclusiva a vistas o a partes y evitaremos siempre y cuando sea posible la utilización del concepto de editor.

Las diferentes partes que conforman AFRYCA, siguiendo las diferentes perspectivas definidas con anterioridad son:

- **PTDG**
  - **Problemas**: Gestión de los archivos de PTDG.
  - **Problema**: Edición del PTDG seleccionado.
  - **Pila**: Información sobre la simulación a ejecutar. Compartida con la perspectiva de modelos de consenso.
- **Modelos de consenso**
  - **Modelos de consenso**: Selección del modelo de consenso.
  - **Configuración**: Configuración del modelo de consenso.
  - **Configuraciones**: Configuraciones almacenadas para un modelo de consenso.
  - **Comportamientos**: Comportamiento usado si el modelo de consenso usa feedback.
  - **Actitud**: Actitud del comportamiento.
  - **Actitudes**: Actitudes almacenadas para un comportamiento.
  - **Pila de simulación**: Información sobre la simulación a ejecutar. Compartida con la perspectiva de PTDG.
- **Simulaciones**
  - **Simulaciones**: Selección de la simulación.
  - **Configuración**: Datos de configuración de la simulación.
  - **Resultados**: Resultados de la simulación.
  - **MDS**: Visualización de preferencias ronda a ronda usando escalado multidimensional.
  - **PCA**: Visualización de preferencias ronda a ronda usando
  - **Métricas**: Métricas de análisis de la simulación.
  - **Gráficas**: Gráficas de la simulación.
- **Simulación**
  - **Ronda final**: Selección de la ronda final de la simulación.
  - **Ronda inicial**: Selección de la ronda inicial de la simulación.
  - **PTDG**: Visualización del problema durante la simulación.
  - **Gráficas**: Gráficas de la simulación.
- **ASE**
  - **Enlaces**: Variables de ejecución asociadas con ASE.
  - **Clases precargadas**: Clases java que son incluidas por defecto en cada script ejecutado en ASE.
  - **Variables**: Variables generadas en ASE.
  - **Evaluaciones**: Evaluaciones realizadas desde la consola de ASE.
  - **Consola**: Consola interactiva de ASE.
  - **Snippets**: Gestión de snippets y repositorios de snippets.
  - Funciones: Gestión de funciones, módulos y enlaces a módulos.

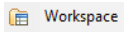
3.1.3. Ventanas de diálogo

Determinadas acciones realizadas con la aplicación requieren el uso de ventanas de diálogo o diálogos que permiten al usuario interactuar con la aplicación de forma detallada en un contexto determinado.

En adición a los diálogos habituales como preguntas de confirmación, advertencias o notificaciones de errores, AFRYCA incluye varios diálogos que facilitan la realización de tareas avanzadas.

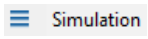
Estos diálogos, que podríamos denominar **diálogos avanzados** son:

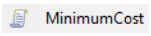
- Dentro de la pestaña **file**:
  - **Importación de PTDG**: Importación de un PTDG a partir de un archivo físico o la ruta de un archivo remoto.
  - **Nuevo PTDG:** Asistente para la creación de un PTDG.

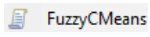
- Dentro de **Editar -> Preferencias**
  - 

Espacio de trabajo: Gestión de preferencias del espacio de trabajo de AFRYCA.
  - 

Preferencias del GDMP: Gestión de preferencias de PTDG.
  - 

Exportar preferencias: Para exportar preferencias una vez realizada la simulación.
  - 

Preferencias de simulación: Gestión de preferencias de simulaciones.
  - 

Preferencias para Mínimo Coste: Gestión de las preferencias para la ejecución de la métrica de mínimo coste.
  - 

Preferencias de FuzzyCMeans: Gestión de las preferencias para la ejecución del FuzzyCMeans en modelos de consenso.
  - 
Preferencias de las estructuras: Gestión de preferencias de las relaciones de preferencia.

- Desde la **Perspectiva de ASE**:

  - 
Repositorio: Creación o modificación de un repositorio de snippets.
  - 
Snippet: Creación o modificación de un snippet.
  - 
Módulo: Creación o modificación de un módulo de funciones.
  - 
Función: Creación o modificación de una función.
  - 
Enlace a módulo: Creación o modificación de un enlace a módulo.

3.1.4. Comandos

En AFRYCA, la interacción con la aplicación se modela haciendo uso de comandos, los cuales permiten encapsular una funcionalidad y aplicarla sobre unos datos específicos para obtener un resultado determinado. Así, en AFRYCA existen comandos para realizar todo tipo de tareas tales como crear un PTDG, seleccionar un modelo de consenso o ejecutar una simulación. Algunos comandos son invocados de forma directa con la interacción del usuario, tales como la selección o la modificación de una FPR, mientras que otros son invocados haciendo uso de los menús y combinaciones de teclas habilitadas.

Cabe notar que los comandos responden de modo inteligente al estado de la aplicación, lo cual implica que:

- Un comando únicamente podrá ser ejecutado cuando su ejecución tenga sentido. Por ejemplo, si no se ha seleccionado ningún PTDG no será posible ejecutar el comando que permite eliminar un PTDG.
- Si una combinación de teclas es utilizada para invocar varios comandos, la combinación se utilizará para ejecutar el comando asociado con el contexto actual, entendiéndose por contexto el elemento seleccionado en la interfaz. Así, si se ha seleccionado un PTDG y la vista de problemas está activa, al pulsar la tecla suprimir se invocará el comando para eliminar el PTDG seleccionado, no invocándose este comando si no se cumplen las condiciones anteriores.

De forma automática Eclipse RCP modifica los menús de la aplicación y añade a los elementos las combinaciones de teclas activas en cada momento. Sin embargo, algunas versiones de Eclipse RCP no realizan correctamente esta modificación, mostrándose en los menús combinaciones de teclas incorrectas. Por ello, incluimos a continuación las diferentes combinaciones de teclas habilitadas desde AFRYCA y recomendamos que, en caso de duda, se use la información de esta sección.

- Perspectiva de PTDG

Importar PTDG	Ctrl + I
Nuevo PTDG	Ctrl + N
Clonar PTDG	Ctrl + C
Eliminar PTDG	Suprimir
Añadir experto	Ctrl + N
Eliminar experto	Suprimir
Seleccionar colectiva	Ctrl + C
Retroceder experto	Ctrl + Fl. Subir
Avanzar experto	Ctrl + Fl. Bajar
Generar Preferencia	Ctrl + G
Añadir alternativa	Ctrl + N
Eliminar alternativa	Suprimir
Retroceder alternativa	Ctrl + Fl. Subir
Avanzar alternativa	Ctrl + Fl. Bajar
Guardar PTDG	Ctrl + S
Guardar todos los PTDGs	Shift + Ctrl + S

- Perspectiva de ASE

Pegar enlace	Ctrl + V
Reemplazar contenido del editor con enlace	Shift + Ctrl + V
Pegar instrucción	Ctrl + B
Reemplazar contenido del editor con instrucción	Shift + Ctrl + B

- Perspectiva de Modelos de Consenso

Guardar configuración	Ctrl + R
Restaurar configuración	Ctrl + Z
Eliminar configuración	Suprimir
Guardar actitud	Ctrl + R
Restaurar actitud	Ctrl + Z
Eliminar actitud	Suprimir
Preferencias del elemento activo	Ctrl + P
Simular	F11

3.2. El espacio de trabajo

3.2.1. Estructura

En AFRYCA todos los elementos de la aplicación se almacenan en un directorio denominado espacio de trabajo o workspace. En este directorio la aplicación crea diferentes carpetas en la que almacenar los archivos creados. En concreto, dentro del espacio de trabajo existirán los siguientes directorios:

problems/: Archivos de PTDG.

configurations/: Configuraciones de modelos de consenso.

attitudes/: Actitudes de comportamientos.

scripts/: Archivos generados por ASE.

A su vez, dentro del directorio **scripts** se emplean diferentes subdirectorios para el almacenamiento de snippets, funciones y enlaces a modulos. De este modo, cualquier PTDG creado por el usuario será almacenado en el directorio **<espacio de trabajo>/problems/,** las configuraciones de modelos de consenso en **<espacio de trabajo>/configurations/** y así con el resto de elementos.

Los elementos generados por AFRYCA se almacenan internamente haciendo uso de marcado xml, aunque para facilitar su gestión cada tipo de elemento se denota con una extensión distinta. Es posible encontrar archivos con las siguientes extensiones:

.afryca: PTDGs.

.configuration: Configuraciones de modelos de consenso.

.attitude: Actitud de comportamiento.

.script: Archivo de script. Puede ser un snippet o una función.

.category: Agrupación de scripts. Puede ser un repositorio o un módulo.

.mb: Enlace a módulo.

Cabe notar que en los archivos de PTDGs, configuraciones y actitudes, los nombres de los archivos coinciden con los nombres de los identificadores de los elementos en la aplicación, es decir, si un PTDG se denomina 'problema_x', el archivo físico se denominará 'problema_x.afryca'. En los archivos generados por ASE, para facilitar su invocación desde el mismo entorno se permite que el nombre del archivo sea diferente al identificador del elemento.

3.2.2. Preferencias

El espacio de trabajo cuenta con soporte para preferencias que permiten su personalización. Específicamente, desde sus preferencias puede configurarse:

- **Directorio**: Directorio empleado para el espacio de trabajo. Por defecto se establece el directorio workspace dentro del directorio de la aplicación. Admite la utilización de directorios de solo lectura. Para más información véase la sección de despliegue en la guía de instalación para linux o windows.
- **Modo de vista de los problemas del espacio de trabajo**: Permite conmutar el tipo de visualización empleada en la vista de problemas, pudiendo seleccionarse una vista normal más detallada o una vista reducida más compacta. Por defecto se emplea la vista normal.

5-Preferencias del espacio de trabajo

Para modificar las preferencias del espacio de trabajo desde AFRYCA se ofrece una ventana de diálogo que puede ser abierta de múltiples modos:

1. Desde el menú **'Editar > Preferencias > Espacio de trabajo'**.
2. Desde los menús de las vistas **Problemas, Configuraciones, Actitudes, Snippets o Funciones.**
3. Pulsa la combinación de teclas 'Ctrl + P' siempre y cuando la vista seleccionada sea alguna de las indicadas en el punto anterior.

3.2.3. Elementos nativos

AFRYCA soporta la inclusión de elementos precargados o nativos, los cuales son archivos integrados en las bibliotecas de la aplicación y sobre los que el usuario no puede realizar modificaciones.

La versión actual de AFRYCA incluye varios elementos nativos tales como:

- PTDGs empleados en publicaciones anteriores en las que se ha utilizado el framework.
- Módulos, funciones y enlaces a módulos empleados en la aplicación.
- Bibliotecas matemáticas en formato snippet desarrolladas desde ASE.
- Repositorios de snippets de ejemplo del uso de ASE.

Para diferenciar fácilmente los elementos nativos del resto de elementos del espacio de trabajo, un elemento nativo se marca con la letra 'N'.

6- Elementos nativos

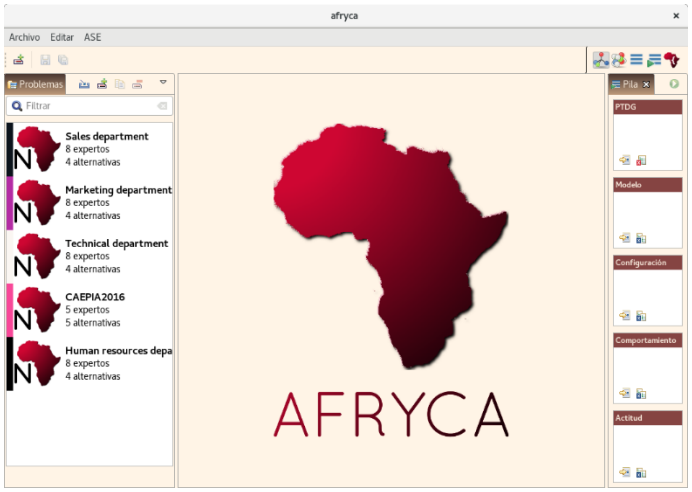
3.3. Definición de PTDGs

AFRYCA permite simular procesos de alcance de consenso en PTDG, para lo cual la primera etapa pasa por la definición del problema. La versión actual del framework soporta la definición de PTDGs en los que las valoraciones de los expertos son expresadas haciendo uso de relaciones de preferencia difusas (FPR), relaciones de preferencia difusas dudosas (HPR), relaciones de preferencias lingüísticas (LPR), relaciones de preferencias lingüísticas dudosas (HLPR), relaciones de preferencia multiplicativas (MPR) y matrices de decisión (DM).

Esta sección revisa la funcionalidad ofrecida por AFRYCA para la gestión de PTDGs, lo cual se lleva a cabo mediante una exploración de la perspectiva PTDG y de las utilidades proporcionadas por la herramienta para trabajar con las diferentes estructuras.

3.3.1. Perspectiva PTDG

La perspectiva PGDG es la que se mostrará al abrir la aplicación por primera vez, siendo su apariencia similar a la mostrada a continuación.



7-Perspectiva PTDG

En la perspectiva pueden apreciarse tres grandes zonas a modo de columnas, las cuales de izquierda a derecha cumplen las siguientes funciones:

- Vista de problemas: Esta vista muestra los PTDG existentes en el espacio de trabajo y permite realizar diversas operaciones sobre ellos tales como la creación, selección, importación, clonación y eliminación.
- Vista de problema: Permite la visualización y edición del PTDG seleccionado. Cuando no se selecciona ningún PTDG la vista muestra el logotipo de AFRYCA.
- Vista de pila de simulación: Informa sobre el estado de la simulación a ejecutar. Esta vista es compartida con la perspectiva de modelos de consenso.

A continuación, se detalla la funcionalidad de las dos primeras vistas. La vista de pila de simulación está documentada en la sección dedicada a las simulaciones.

3.3.2. Vista de problemas

La vista de problemas contiene un listado de los problemas existentes en el espacio de trabajo y permite trabajar con ellos. Para cada problema la vista muestra múltiple información:

8-Problema

- **Nombre del problema.**
- **Número de expertos.**
- **Número de alternativas.**
- **Marca del problema**. Se corresponde con la franja de color vertical mostrada a la izquierda del elemento y permite diferenciar visualmente de forma rápida un elemento de otro. El color de la franja es calculado empleando una función hash que toma como entrada los valores internos del problema al iniciar la aplicación. Si se modifica un PTDG, al reiniciar AFRYCA este se mostrará con una marca de un color diferente.
- **Icono del elemento.** Para elementos del tipo PTDG el icono se corresponde con el logotipo de AFRYCA. Si el elemento es nativo el icono tendrá una letra **'N'** superpuesta.

Si se desea puede modificarse el modo de visualización de la vista para emplear un estilo más compacto desde las preferencias del espacio de trabajo.

La vista permite establecer el PTDG que será empleado al realizar una simulación, el cual se selecciona al hacer click sobre él.

9- Problemas

Cuando un elemento es seleccionado su identificador es mostrado en la parte inferior de la vista y abierto en la vista de problema.

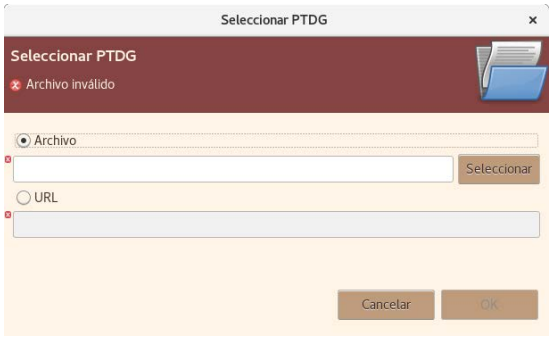
Es posible limitar los PTDGs mostrados en la vista a aquellos que contengan en su nombre una sucesión de caracteres dada. Cabe notar que, si se ha seleccionado un PTDG y al filtrar los elementos visibles en la vista queda excluido de los elementos visibles, será automáticamente deseleccionado. De igual modo, al realizar la creación de un nuevo PTDG cualquier filtro existente es eliminado de forma automática para evitar que el problema quede excluido de los elementos mostrados

10-Búsqueda de un problema

La vista de problemas es el elemento de la interfaz idóneo para realizar tareas de gestión sobre los PTDGs del espacio de trabajo, ya que habilita menús contextuales y combinaciones de teclas cuando está activa y lo que es más importante, permite observar el resultado de la acción realizada en el momento. Por ello, las acciones aplicables a PTDGs se describen dentro de esta sección. No obstante, mencionar que estas acciones pueden ser realizadas en cualquier punto de la aplicación empleando el menú **'Archivo'**.

3.3.3. Importación de un PTDG

La importación permite añadir un PTDG existente al espacio de trabajo de la aplicación, estando esta opción activa únicamente si el espacio de trabajo es de lectura y escritura. Al abrir la opción de importación se mostrará una ventana de diálogo con dos opciones:



11-Importar PTDG

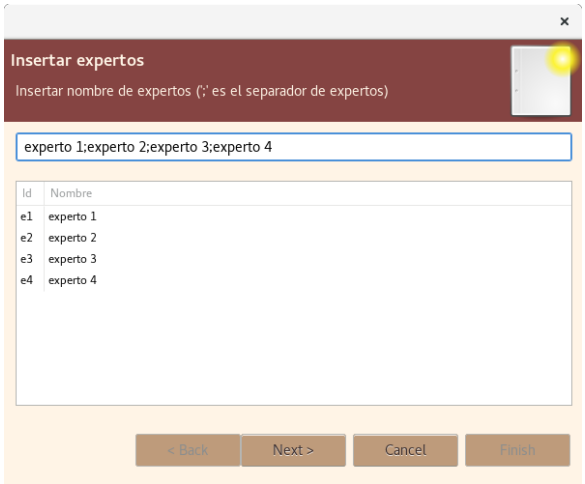
- **Archivo**: Importación de un archivo PTDG existente en el sistema.
- **URL**: Importación de un archivo PTDG remoto a partir de su URL.

Es adecuado mencionar que la importación de un PTDG a través de este comando únicamente es una facilidad que se le brinda al usuario. Puede añadirse un PTDG al espacio de trabajo de AFRYCA simplemente copiando el archivo dentro del directorio, apareciendo este como un elemento más de la aplicación la próxima vez que se inicie.

3.3.4.Creación de un PTDG

AFRYCA ofrece un asistente que permite crear de forma rápida un PTDG, el cual únicamente estará activo si el espacio de trabajo es de lectura y escritura. El asistente está dividido en dos páginas:

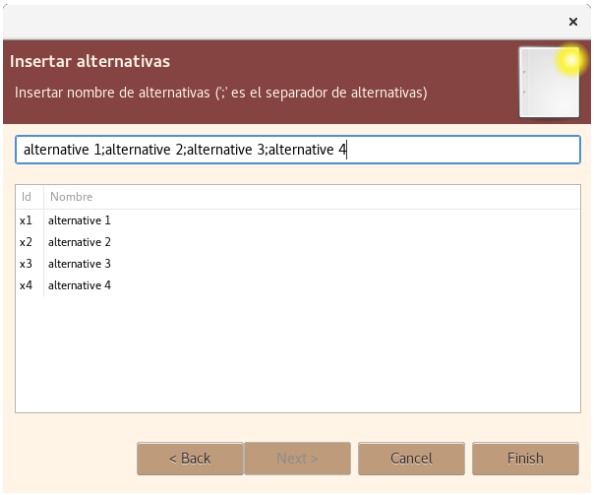
Página de expertos



12-Página de expertos

Desde esta página el usuario puede introducir los identificadores de los expertos que tomarán parte en el problema. Para facilitar su introducción, los identificadores se introducen en un cuadro de texto separando cada uno de ellos con el signo '**;**'. La página no permite que se empleen identificadores duplicados ni menos de dos expertos.

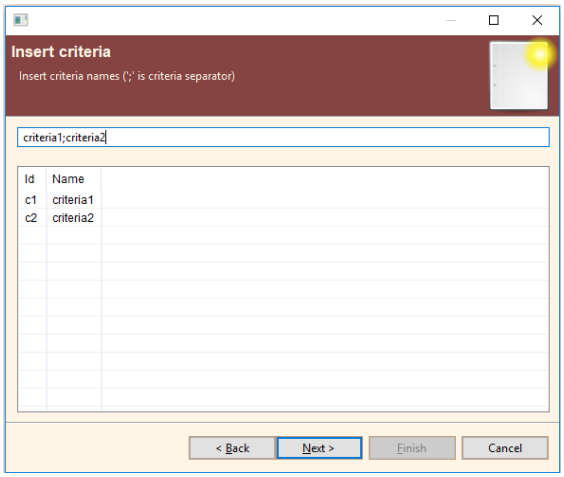
Página de alternativas



13-Página de alternativas

Desde esta página el usuario puede introducir las alternativas a valorar en el problema. Al igual que con la página de expertos, para facilitar su introducción los identificadores se introducen en un cuadro de texto separando cada uno de ellos con el signo '**;**'. La página no permite que se empleen identificadores duplicados ni menos de dos alternativas.
Una vez introducidos los expertos y alternativas, tras pulsar el botón '**Finish**' se solicitará el nombre para el nuevo PTDG y se creará el archivo en el espacio de trabajo.

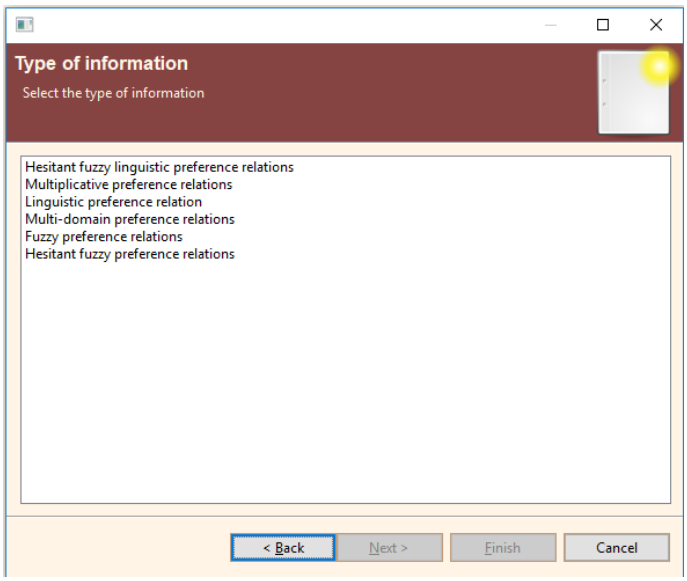
Página de criterios



14-Página de criterios

Desde esta página el usuario puede introducir los criterios a valorar en el problema. Al igual que con la página de expertos o alternativas, para facilitar su introducción los identificadores se introducen en un cuadro de texto separando cada uno de ellos con el signo '**;**'. La página no permite que se empleen identificadores duplicados.

Página de selección del tipo de información



15-Página de selección del tipo de información

Desde esta página el usuario especifica el tipo de información que va a utilizar en el problema. Los diferentes tipos de información son obtenidos desde los puntos de extensión de las estructuras.

Una vez introducidos los expertos, alternativas, criterios y el tipo de información tras pulsar el botón '**Finish**' se solicitará el nombre para el nuevo PTDG y se creará el archivo en el espacio de trabajo.

3.3.5.Clonación de un PTDG

Con la clonación de un PTDG AFRYCA facilita la creación de un problema a partir de uno ya existente, resultando especialmente útil cuando se desea realizar modificaciones sobre algún PTDG que, o bien no se desea modificar, o bien no es posible modificar al tratarse de un elemento nativo. Al igual que con el resto de opciones que implican modificaciones del espacio de trabajo, la clonación únicamente se podrá emplear si el espacio de trabajo es de lectura y escritura.

Para clonar un PTDG basta con seleccionar uno ya existente y ejecutar el comando desde alguna de las muchas vías habilitadas. Si es posible utilizar el comando se solicitará el nombre para el nuevo PTDG y se realizará su creación a partir de la información del PTDG seleccionado.

3.3.6.Eliminación de un PTDG

Con la eliminación de un PTDG se suprime el archivo del espacio de trabajo. Como es lógico, la opción estará disponible siempre y cuando un PTDG se haya seleccionado, este no se trate de

un elemento nativo de la aplicación, y se dispongan de permisos de escritura sobre el directorio del espacio de trabajo.

Para eliminar un PTDG basta con seleccionar el problema y ejecutar el comando. Si es posible eliminarlo se solicitará confirmación y se eliminará el PTDG.

3.3.7.Guardar un PTDG

Al realizar cualquier acción en un PTDG que implique que el problema difiera a la versión almacenada en el espacio de trabajo, el PTDG estará marcado como '**modificado**'. Cuando un PTDG está marcado como modificado este estado es indicado en múltiples puntos de la aplicación con el carácter '*****':

- Junto al nombre del problema en la vista de problemas.
- En el título de la vista de problema.
- En el título de la aplicación.

Un PTDG modificado puede ser guardado siempre y cuando no se trate de un elemento nativo de la aplicación y se dispongan de permisos de escritura sobre el directorio del espacio de trabajo.

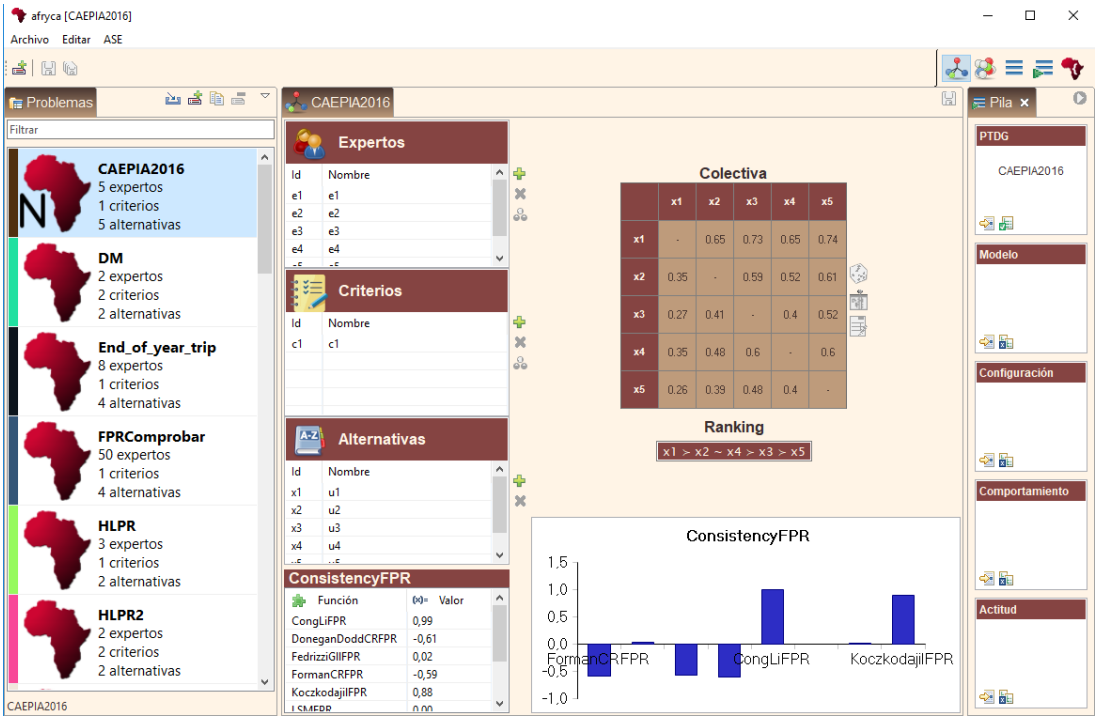
Para guardar un PTDG basta con seleccionar el problema y ejecutar el comando '**guardar PTDG**'. Si es posible invocar el comando se almacenarán los cambios y el PTDG dejará de estar marcado como modificado.

3.3.8.Guardar todos los PTDGs

El comando '**guardar todos los PTDGs**' guarda los cambios de todos los PTDGs marcados como '**modificados**' que no sean elementos nativos. El comando se podrá ejecutar siempre y cuando exista al menos un PTDG modificado que no se trate de un elemento nativo de la aplicación y se dispongan de permisos de escritura sobre el directorio del espacio de trabajo.
Si al salir de la aplicación existen PTDGs modificados y el comando '**guardar todos los PTDGs**' puede ser ejecutado, se le ofrecerá al usuario la opción de ejecutarlo para evitar perder los cambios realizados.

3.3.9.Vista de problema

La vista de problema es el lugar desde el que visualizar y modificar un PTDG. Su contenido será diferente en función de si existe un PTDG seleccionado o no, mostrándose en el primer caso los datos del PTDG seleccionado y en el segundo caso el logotipo de la aplicación.



16-Vista de problema

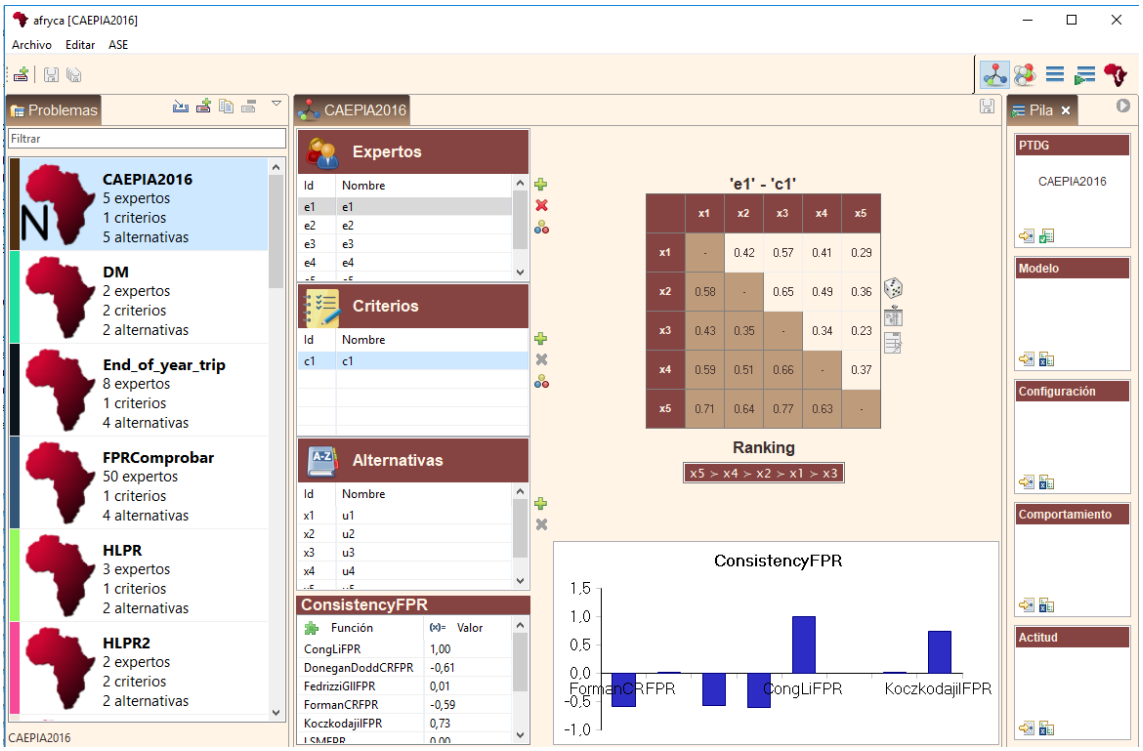
La vista de problema se subdivide en cuatro zonas, las cuales ofrecen diferente información del problema seleccionado:

- **Esquina superior izquierda** - **Elementos del problema**: Expertos y alternativas del PTDG.
- **Esquina superior derecha** - **Relación de preferencia**: FPR del experto seleccionado y ranking de alternativas calculado a partir de esta relación de preferencias.
- **Esquina inferior izquierda** - **Enlaces a módulos**: Enlaces a módulos seleccionados para los PTDG.
- **Esquina inferior derecha** - **Gráficas**: Gráficas seleccionadas para los PTDG.

3.3.9.1. Elementos del problema

Los elementos del problema son los expertos que toman parte en el problema y las alternativas que se valoran en el mismo. La vista de problema permite realizar diferentes acciones con los elementos.

3.3.9.2. Selección de un experto



17-Selección de un experto

Pulsando sobre cualquiera de los expertos mostrado, este pasará a ser el experto seleccionado. La selección de un experto implica que:

- Las preferencias mostradas en la vista de problema serán las de este experto, siendo la colectiva de ese experto si existen varios criterios o la preferencia del expertos i sólo existe un criterio.
- El ranking mostrado será calculado para este experto.
- Cualquier enlace a módulo en el que el experto seleccionado tenga relevancia será recalculado.

3.3.9.3. Deselección de un experto

Un experto seleccionado puede deseleccionarse invocando el comando '**seleccionar colectiva**' desde alguna de las vías habilitadas al efecto. La deselección implica que, toda la información mostrada en el resto de elementos de la vista que operan con las preferencias del experto seleccionado, se llevarán a cabo empleando una preferencia colectiva calculada a partir de las preferencias individuales de todos los expertos. En la versión actual de AFRYCA la preferencia colectiva se calcula en base a la selección del usuario en las preferencias de cada estructura o preferencia.

3.3.9.4. Añadir un experto

Puede añadirse un experto invocando el comando '**añadir experto**'. Esta acción creará un nuevo experto en el PTDG y creará para él una estructura vacía para valorar las alternativas del problema.

3.3.9.5. Eliminar un experto

Si se selecciona un experto y en el problema existen más de dos, el seleccionado puede eliminarse invocando el comando '**eliminar experto**'. La eliminación conlleva la supresión del mismo y de sus preferencias.

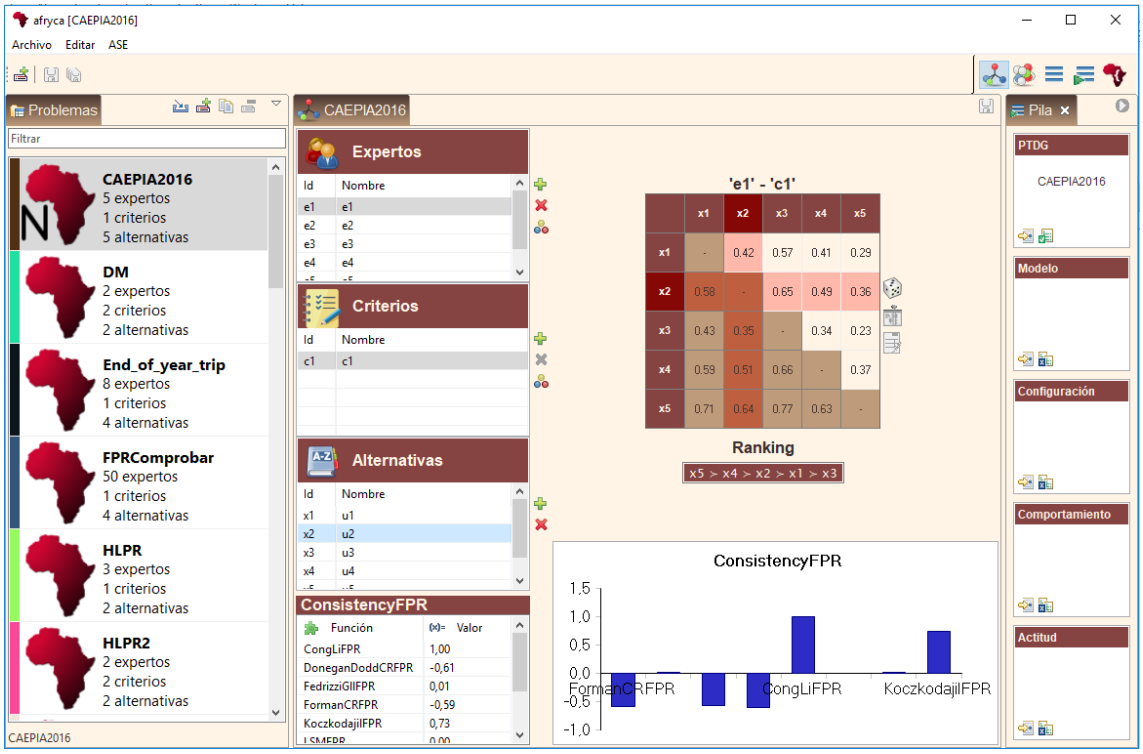
3.3.9.6. Retroceder un experto

El retroceso y avance de expertos permite ordenar los expertos del problema en la interfaz de la aplicación. Estando un experto seleccionado y siempre y cuando este no sea el primero en el PTDG, se puede retroceder su posición invocando el comando '**retroceder experto**'.

3.3.9.7. Avanzar un experto

De forma análoga al retroceso, si un experto está seleccionado y este no es el último en el PTDG, se puede avanzar su posición invocando el comando '**avanzar experto**'.

3.3.9.8. Selección de una alternativa



18-Selección de una alternativa

Al igual que con los expertos del problema, es posible seleccionar cualquier alternativa, lo cual puede realizarse de dos modos:

1. Pulsando sobre la fila correspondiente en la tabla de alternativas.
2. Pulsando sobre la cabecera de la alternativa en la fila o columna correspondiente en la estructura mostrada.

Al seleccionar una alternativa, la fila y columna de esta se resaltarán en la tabla de preferencias y cualquier enlace a módulo en el que la alternativa seleccionada tenga relevancia será recalculado.

3.3.9.9. Deselección de una alternativa

Una alternativa seleccionada puede deseleccionarse pulsando en la celda que ocupa la esquina superior izquierda de la estructura mostrada.

3.3.9.10. Añadir una alternativa

Puede añadirse una alternativa invocando el comando '**añadir alternativa**'.
Debido a que la introducción de una nueva alternativa afecta a la consistencia de las valoraciones existentes, al realizar esta acción se deben eliminar todas las valoraciones del problema. Por ello, se recomienda no realizar esta acción salvo en las fases iniciales de definición del problema.

3.3.9.11. Eliminar una alternativa

Si se selecciona una alternativa y en el problema existen más de dos, la alternativa puede eliminarse invocando el comando '**eliminar alternativa**'. La eliminación conlleva la supresión de la misma así como de las valoraciones en las que esté implicada.
Como esta acción puede realizarse sin que la consistencia del resto de valoraciones se vea afectada, al eliminar una alternativa no es necesario eliminar las valoraciones en las que no está implicada como si ocurre al añadir una nueva.

3.3.9.12. Retroceder una alternativa

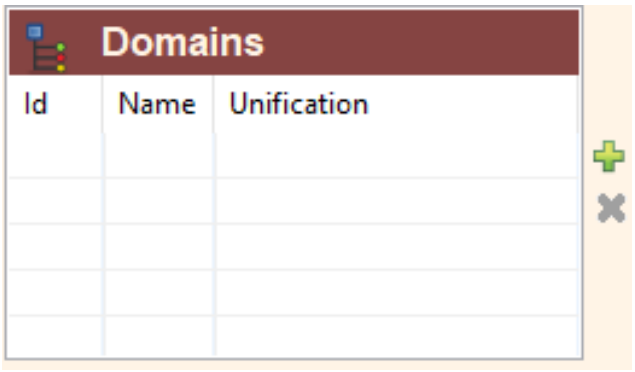
El retroceso y avance de alternativas permite ordenar las alternativas del problema en la interfaz de la aplicación. Estas acciones afectarán siempre a un par de alternativas, cuyas filas y columnas serán intercambiadas en las correspondientes relaciones de preferencia empleadas en el problema.

Estando una alternativa seleccionada y siempre y cuando esta no sea la primera en el PTDG, se puede retroceder su posición invocando el comando '**retroceder alternativa**'.

3.3.9.13. Avanzar una alternativa

De forma análoga al retroceso, si una alternativa está seleccionada y esta no es la última en el PTDG, se puede avanzar su posición invocando el comando '**avanzar alternativa**'.

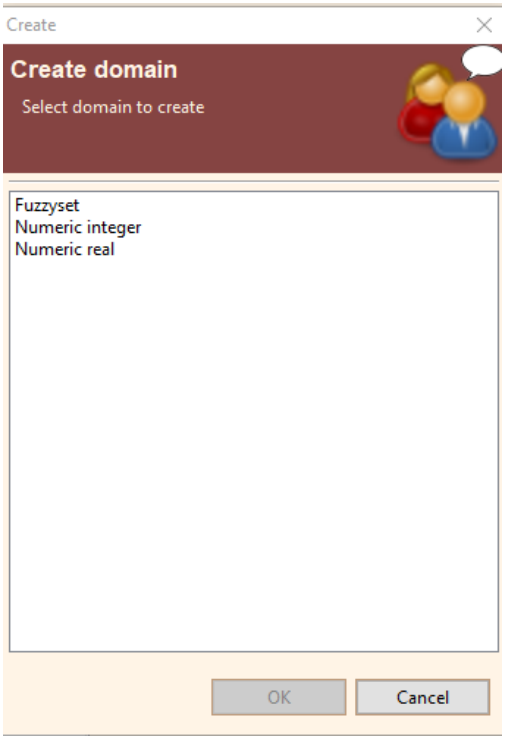
3.3.9.14. Añadir un dominio



19-Tabla de dominios

Puede añadirse un dominio invocando el comando '**añadir dominio'**.
La tabla sólo aparecerá en aquellas estructuras que necesitan los dominios para su uso como son LPR, HLPR o DM.

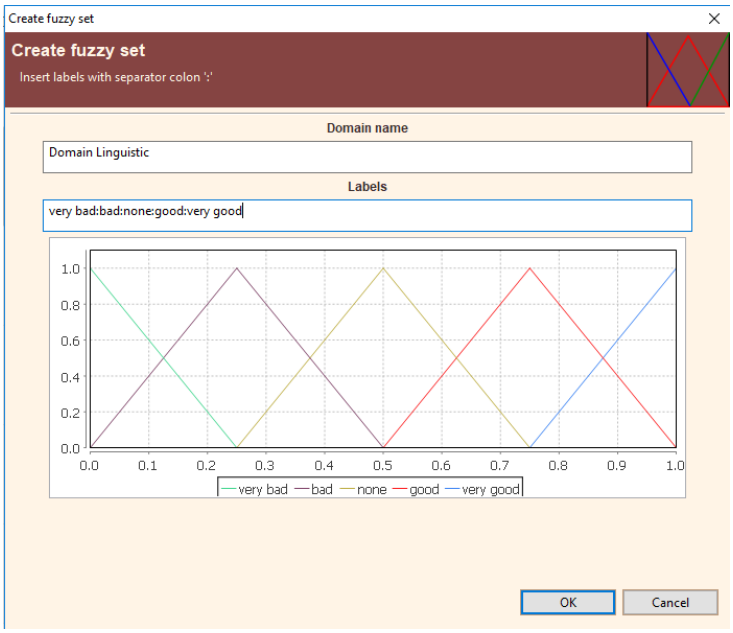
Al pulsar sobre añadir dominio, se abrirá un dialogo que mostrará los distintos tipos de dominios posibles a crear, en este caso son: FuzzySet (Dominio lingüístico), Numeric Intenger (Dominio numérico entero) o Numeric real (Dominio numérico real).



20-Selección de dominio

3.3.9.14.1. Añadir un dominio FuzzySet

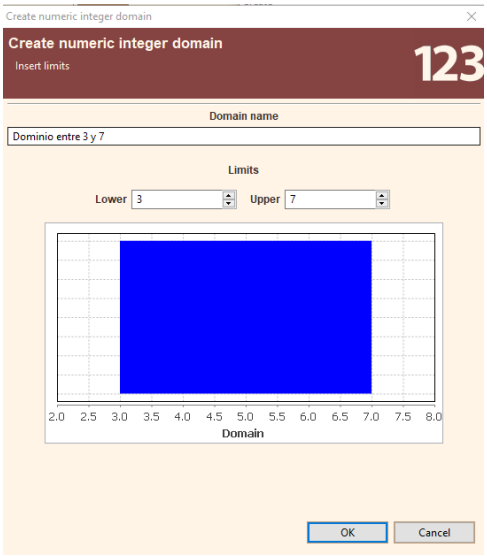
Si seleccionamos crear un dominio FuzzySet nos mostrará un dialogo el cual nos pide el nombre de dominio y las etiquetas del dominio separadas por "**:**".



21-Dominio FuzzySet

3.3.9.14.2. Añadir un dominio Numerico Entero

Si seleccionamos crear un dominio numérico entero se mostrará un dialogo, el cual nos permite introducir el nombre y el rango del dominio mediante dos spinners:

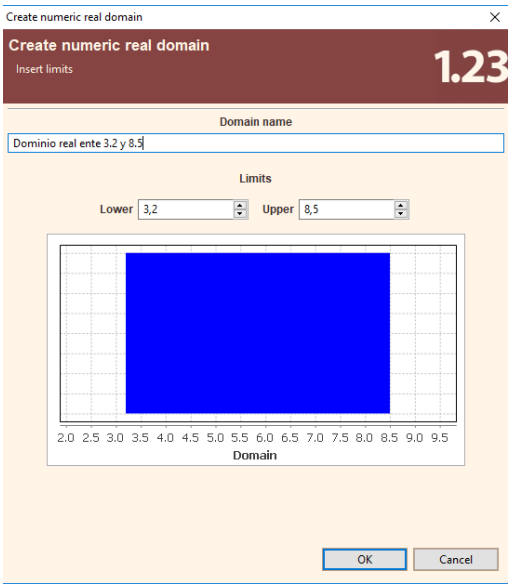


22-Dominio Numeric Integer

Cabe destacar que el spinner inferior siempre será menor que el superior, no pudiéndose modificar mientras el superior no tenga un valor mayor.

3.3.9.14.3. Añadir un dominio Numérico Real

Si seleccionamos crear un dominio numérico entero se mostrará un dialogo, el cual nos permite introducir el nombre y el rango del dominio mediante dos spinners:



23-Dominio Numeric Real

Cabe destacar que el spinner inferior siempre será menor que el superior, no pudiéndose modificar mientras el superior no tenga un valor mayor.

3.3.9.15. Preferencias

La relación de preferencia muestra las valoraciones realizadas por el experto seleccionado para las diferentes alternativas. Estas preferencias varían su estructura y comportamiento dependiendo del tipo de estructura empleada en el problema.

3.3.9.15.1. Preferencias de FPR

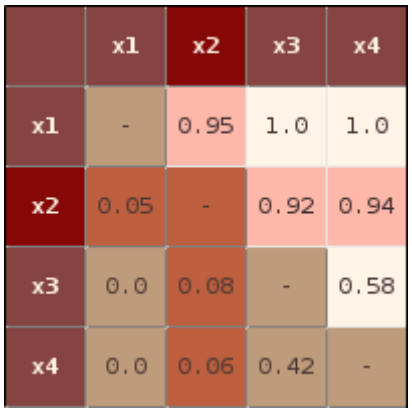
En una **FPR** el valor $\mathbf{x_{ij}}$ representa el grado de preferencia de la alternativa $\mathbf{x_i}$ sobre la alternativa $\mathbf{x_j}$ o $\mathbf{x_i/x_j}$, estando los valores x_{ij} comprendidos entre 0 y 1 y representado un valor:

- Mayor que 0.5 preferencia de x_i sobre x_j.
- Menor que 0.5 preferencia de x_j sobre x_i.
- Igual a 0.5 indiferencia entre x_i y x_j.

Las FPR se muestran haciendo uso de una tabla en la cual:

- La primera fila y columna contiene los identificadores de las diferentes alternativas. La selección de cualquiera de los identificadores conlleva la selección de la alternativa correspondiente.
- La selección de la celda correspondiente a la intersección de la primera fila con la primera columna implica la deselección de cualquier alternativa seleccionada previamente.

- Se admiten dos estados, '**no editable**' y '**editable**'. Una FPR '**no editable**' no permite la modificación de las valoraciones contenidas. Una FPR '**editable**' permite la modificación de las valoraciones contenidas en la diagonal superior de la tabla. La modificación de un valor conlleva la modificación del valor simétrico correspondiente en la diagonal inferior de la tabla.



	x1	x2	x3	x4
x1	-	0.95	1.0	1.0
x2	0.05	-	0.92	0.94
x3	0.0	0.08	-	0.58
x4	0.0	0.06	0.42	-

24-FPR editable en la que se ha seleccionado x2

Modificar una valoración

Seleccionando una celda de la diagonal superior de una FPR editable correspondiente al grado de preferencia de un par de alternativas, es posible modificar su valor de forma similar a como se realizaría en una hoja de cálculo.

El comportamiento de AFRYCA variará en función del valor introducido:

- Si el valor redondeado a dos decimales es igual al valor inicial no se realizará ninguna modificación.
- De igual modo, si se intenta establecer un valor no permitido la modificación no tendrá efecto.
- Si se borra el contenido de la celda o se introduce el valor '**NaN**' se entenderá que no se ha valorado el par de alternativas correspondiente y la celda se mostrará en blanco.

3.3.9.15.2. Preferencias de HPR

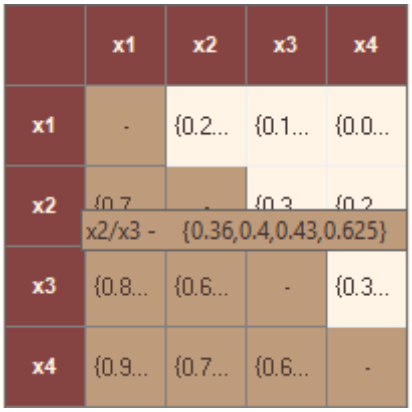
En una **Hesitant Preference Relation** el conjunto de valores $\mathbf{x_{ij}}$ representa el grado de preferencia de la alternativa $\mathbf{x_i}$ sobre la alternativa $\mathbf{x_j}$ o $\mathbf{x_i/x_j}$, estando los valores x_{ij} comprendidos entre 0 y 1 y representado un valor:

- Mayor que 0.5 preferencia de x_i sobre x_j.
- Menor que 0.5 preferencia de x_j sobre x_i.
- Igual a 0.5 indiferencia entre x_i y x_j.

Las HPR se muestran haciendo uso de una tabla en la cual:

- La primera fila y columna contiene los identificadores de las diferentes alternativas. La selección de cualquiera de los identificadores conlleva la selección de la alternativa correspondiente.
- La selección de la celda correspondiente a la intersección de la primera fila con la primera columna implica la deselección de cualquier alternativa seleccionada previamente.

- Se admiten dos estados, '**no editable**' y '**editable**'. Una HPR '**no editable**' no permite la modificación de las valoraciones contenidas. Una HPR '**editable**' permite la modificación de las valoraciones contenidas en la diagonal superior de la tabla. La modificación de un conjunto de valores conlleva la modificación del conjunto de valores simétrico correspondiente en la diagonal inferior de la tabla.
- Las valoraciones son mostradas dejando el ratón encima de la valoración.



25-HPR Editable

Modificar una valoración

Seleccionando una celda de la diagonal superior de una HPR editable correspondiente al grado de preferencia de un par de alternativas, es posible modificar su valor de forma similar a como se realizaría en una hoja de cálculo.

El comportamiento de AFRYCA variará en función del valor introducido:

- El valor debe estar comprendido por números comprendidos entre el 0 y 1, cuyo carácter de puntuación es el **punto** y de separación entre valores es la **coma**.

  Un ejemplo de un valor válido sería: 0,0.1,0.4,0.6,0.8…1

  También se permite un valor que comience y acabe con { }.

  Ejemplo: {0,0.1,0.4,0.6,0.8}.

- De igual modo, si se intenta establecer un valor no permitido la modificación no tendrá efecto.

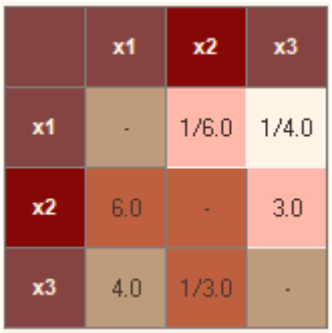
3.3.9.15.3. Preferencias de MPR

En una **Multiplicative Preference Relation** el valor $\mathbf{x_{ij}}$ representa el grado de preferencia de la alternativa $\mathbf{x_i}$ sobre la alternativa $\mathbf{x_j}$ o $\mathbf{x_i/x_j}$, estando los valores x_{ij} comprendidos entre 1/9 y 9 y representado un valor:

- Mayor que 1 preferencia de x_i sobre x_j.
- Menor que 1 preferencia de x_j sobre x_i.
- Igual a 1 indiferencia entre x_i y x_j.

Las MPR se muestran haciendo uso de una tabla en la cual:

- La primera fila y columna contiene los identificadores de las diferentes alternativas. La selección de cualquiera de los identificadores conlleva la selección de la alternativa correspondiente.
- La selección de la celda correspondiente a la intersección de la primera fila con la primera columna implica la deselección de cualquier alternativa seleccionada previamente.
- Se admiten dos estados, '**no editable**' y '**editable**'. Una MPR '**no editable**' no permite la modificación de las valoraciones contenidas. Una MPR '**editable**' permite la modificación de las valoraciones contenidas en la diagonal superior de la tabla. La modificación de un valor conlleva la modificación del valor simétrico correspondiente en la diagonal inferior de la tabla.



26-MPR editable en la que se ha seleccionado x2

Modificar una valoración

Seleccionando una celda de la diagonal superior de una MPR editable correspondiente al grado de preferencia de un par de alternativas, es posible modificar su valor de forma similar a como se realizaría en una hoja de cálculo.

El comportamiento de AFRYCA variará en función del valor introducido:

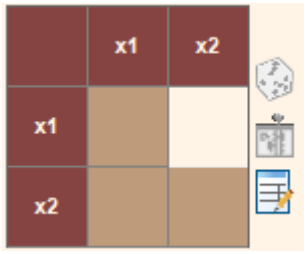
- Los valores posibles son los números enteros [1-9] o 1/[1-9]
- De igual modo, si se intenta establecer un valor no permitido la modificación no tendrá efecto.

Para mostrar la colectiva si se permiten otros posibles valores.

3.3.9.15.4. Preferencias de LPR

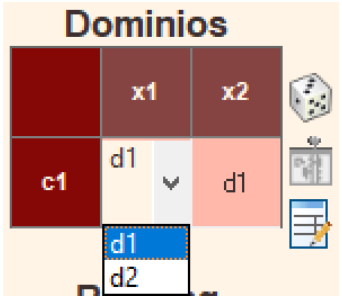
En una **Linguistic Preference Relation** el valor $\mathbf{x_{ij}}$ representa el grado de preferencia de la alternativa $\mathbf{x_i}$ sobre la alternativa $\mathbf{x_j}$ o $\mathbf{x_i/x_j}$, para poder valorar mediante LPR es necesario la creación de un dominio lingüístico, como se ha visto en apartados anteriores.

Una vez creado uno o varios domnios linguisticos es posible asignar estos dominios a las preferencias. Cada preferencia sólo puede tener un dominio asignado.
Para asignar los dominios mostraremos la preferencia a la cual queremos asignar el domino y se pulsará sobre el botón “cambiar tabla”, que como podemos ver es el único habilitado.



27-Cambiar tabla

Al pulsar sobre él nos mostrará una tabla parecida a la de la preferencia, la cual nos permite seleccionar un dominio creado mediante un desplegable.



28-Selección de dominio

Una vez asignado el dominio podemos introducir las valoraciones.

Modificar una valoración

Seleccionando una celda de la diagonal superior de una LPR editable correspondiente al grado de preferencia de un par de alternativas, es posible modificar su valor mediante un desplegable que muestra las valoraciones del dominio asignado anteriormente.

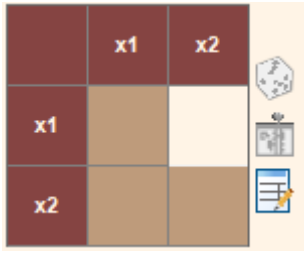
29-Selección de valoración

Una vez asignada la valoración se modificará su simétrica, con el valor opuesto.

3.3.9.15.5. Preferencias de HLPR

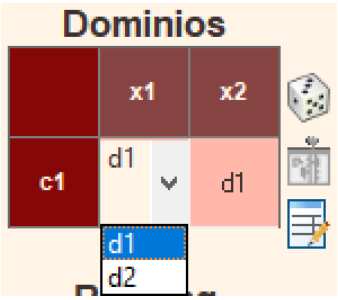
En una **Hesitant Linguistic Preference Relation** el valor $\mathbf{x_{ij}}$ representa un valor difuso de preferencia de la alternativa $\mathbf{x_i}$ sobre la alternativa $\mathbf{x_j}$ o $\mathbf{x_i/x_j}$, para poder valorar mediante HLPR es necesario la creación de un dominio lingüístico, como se ha visto en apartados anteriores.

Una vez creado uno o varios dominios lingüísticos es posible asignar estos dominios a las preferencias. Cada preferencia sólo puede tener un dominio asignado.
Para asignar los dominios mostraremos la preferencia a la cual queremos asignar el domino y pulsaremos sobre el botón “cambiar tabla”, que como podemos ver es el único habilitado.



30-Cambiar Tabla

Al pulsar sobre él nos mostrará una tabla parecida a la de la preferencia, la cual nos permite seleccionar un dominio creado mediante un desplegable.



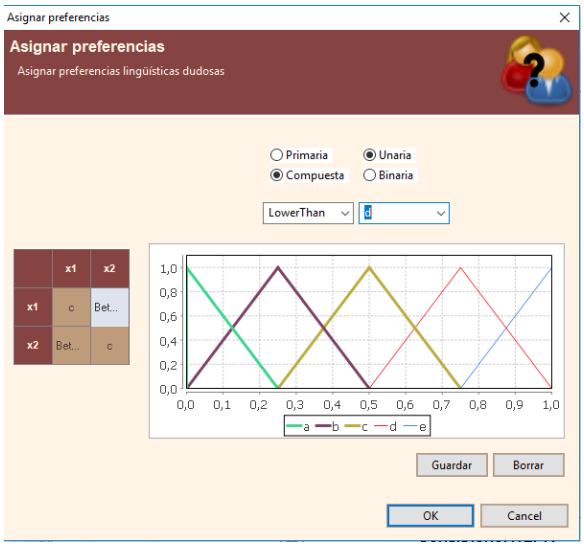
31-Selección de dominio

Una vez asignado el dominio podemos introducir las valoraciones.

Modificar una valoración

Seleccionando una celda de la diagonal superior de una HLPR editable correspondiente al grado de preferencia de un par de alternativas, es posible modificar su valor mediante una ventana de dialogo que será lanzada, en esta podemos seleccionar si la valoración será primaria o compuesta, en el caso de que fuese compuesta nos permite seleccionar si será unaria o binaria.

Para asignar la valoración iremos seleccionando la casilla a modificar y pulsando el botón guardar para almacenar la valoración deseada.

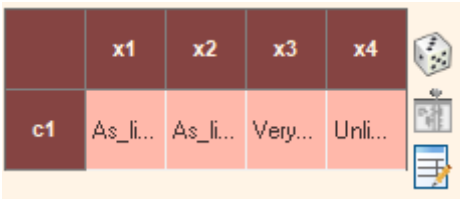


32- Modificar valoración HLPR

Una vez asignada la valoración se modificará su simétrica, con el valor opuesto.

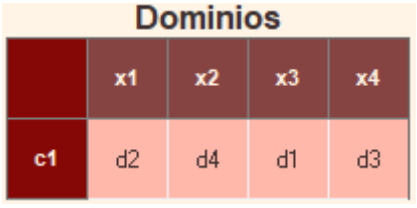
3.3.9.15.6. Preferencias de DM

En una **Decission Matrix** las valoraciones se realizan para cada criterio con las distintas alternativas existentes, como podemos ver en la tabla.



33- Preferencias Decission Matrix

Para la asignación de valoraciones a una DM es necesaria la creación de uno o varios dominios. En el caso de la DM podemos asignar un dominio a cada alternativa.



34- Dominios Decission Matrix

Una vez asignado el dominio podemos introducir las valoraciones.

Modificar una valoración

Seleccionando una celda de una DM, es posible modificar su valor. Esta modificación dependerá del dominio que esté asignado a esa alternativa y criterio.

3.3.10. Generar una estructura aleatoria

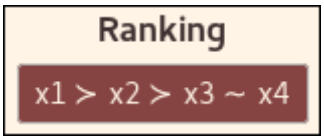
AFRYCA permite generar estructuras de forma aleatoria empleando diferentes métodos matemáticos encapsulados en funciones ASE. Si se selecciona un experto y se invoca el comando '**generar structura'** se generará una estructura de forma aleatoria empleando la función establecida en las preferencias de la misma para ello.

3.3.11. Completar una estructura

AFRYCA también permite completar los valores perdidos o no asignados de una estructura de forma consistente haciendo uso de métodos matemáticos encapsulados en funciones ASE. Si la relación de preferencia del experto seleccionado contiene valores perdidos, pero cuenta con un conjunto generador a partir del cual asignar estos valores, la invocación del comando '**completar estructura**' permitirá utilizar el método establecido en las preferencias de la estructura para asignar los grados de preferencia de las valoraciones no asignadas.

3.3.12. Ranking de alternativas

La relación de preferencia mostrada en la vista de problema se acompaña de un ranking de alternativas calculado a partir de los grados de no-dominancia.



35- Ranking de alternativas

Este ranking, el cual se lee de izquierda a derecha, muestra los identificadores de las alternativas y las relaciones de preferencia que existen entre ellas, pudiendo aparecer dos tipos de relaciones:

- Preferencia de la alternativa de la izquierda frente a la de la derecha. Se representa con el carácter '≻'.
- Indiferencia entre las alternativas. Se representa con el carácter '~'.

3.3.13. Enlaces a módulos

La vista soporta la visualización de enlaces a módulos ASE a través de los cuales analizar diferentes parámetros de los PTDG. Los enlaces a módulos a incluir pueden personalizarse desde las preferencias de PTDG.

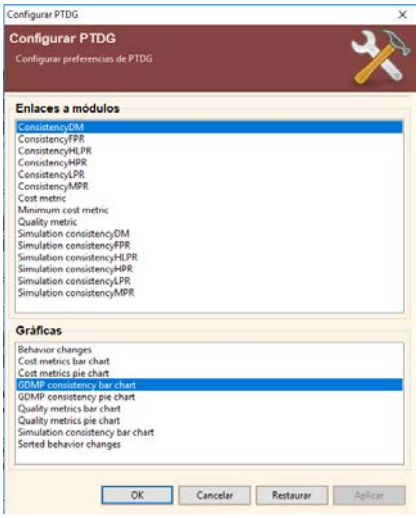
3.3.14. Gráficas

Al igual que con los enlaces a módulos, es posible visualizar diferentes gráficos BIRT con los que visualizar los parámetros que se considere apropiado. Las gráficas a mostrar se pueden personalizar desde las preferencias de PTDG.

3.3.15. Preferencias de PTDG

AFRYCA permite personalizar determinados aspectos de los PTDG usando preferencias. En concreto, las preferencias de PTDG permiten configurar:

- **Enlaces a módulos**: Enlaces a módulos de ASE que se mostrarán en la vista de problema. Por defecto se muestra el enlace '**Consistency**'.
- **Gráficas**: Gráficas BIRT que se mostrarán en la vista de problema. Por defecto se muestran las gráficas '**GDMP consistency bar chart**' y '**GDMP consistency pie chart**'.



36-Preferencias de PTDG

Para modificar las preferencias de PTDG se ofrece una ventana de diálogo que puede ser abierta desde el menú '**Editar > Preferencias > PTDG**'.

3.3.16. Preferencias de las estrcuturas

Al igual que con los PTDG, AFRYCA permite personalizar determinados aspectos de las estructuras usando dialogos. Específicamente, es posible configurar:

- **Función de generación de de la estructura**: Función ASE que se empleará para generar estructuras aleatorias.
- **Función para completar una estructura**: Función ASE que se empleará para completar una estructura con valores perdidos.
- **Función de agregación de una estructura**: Función ASE que se empleará para a agregación de la estructura seleccionada.

37-Preferencias de las estructuras

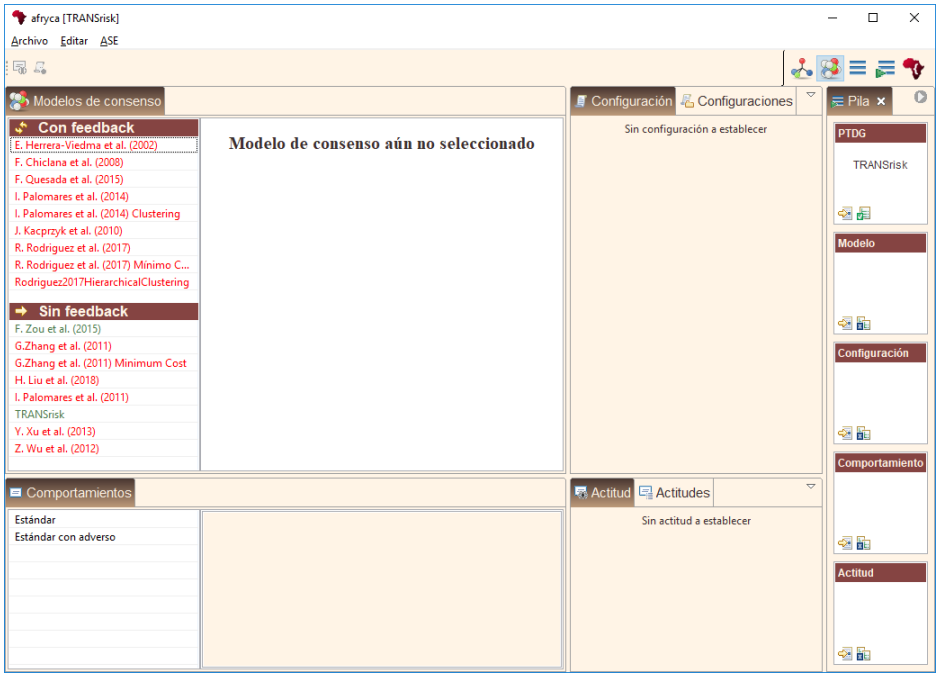
Para modificar las preferencias de una estructura se ofrece una ventana de diálogo que puede ser abierta desde el menú '**Editar > Preferencias > Nombre Estrcutra**.

3.4. Modelos de consenso

Tras definir y establecer el PTDG a emplear en la simulación a realizar, la siguiente etapa es la selección y adaptación del modelo de consenso a utilizar en el proceso de alcance de consenso. En esta sección de la web se recorre la perspectiva de modelos de consenso, punto desde el que AFRYCA permite llevar a cabo estas tareas.

3.4.1. Perspectiva de modelos de consenso

La perspectiva de modelos de consenso es la segunda perspectiva accesible desde el selector de perspectivas, siendo su apariencia inicial similar a la de la siguiente captura.



38-Perspectiva de modelos de consenso

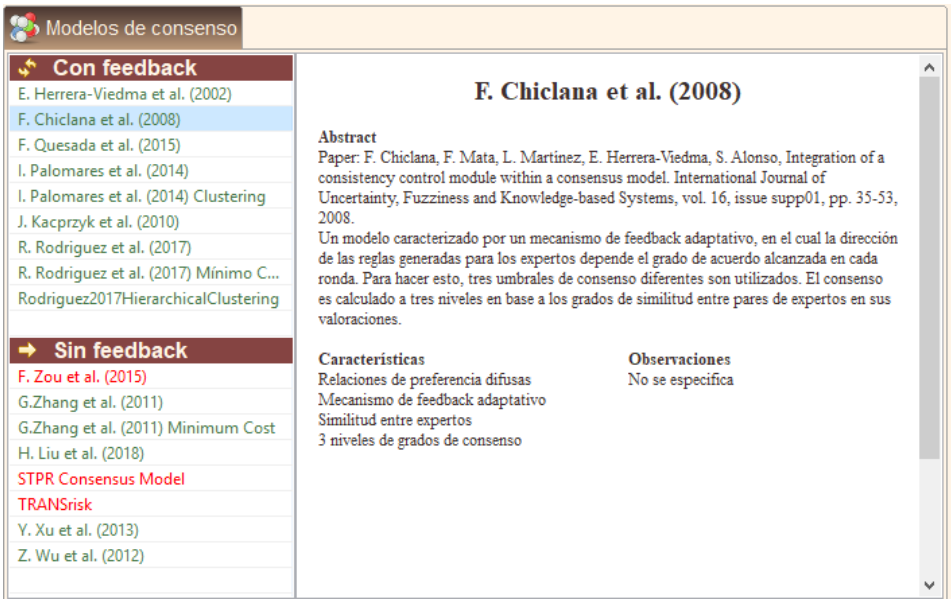
En la perspectiva pueden apreciarse diferentes vistas, estando algunas de ellas apiladas. Las funciones que cumplen las diferentes vistas en la configuración y adaptación de la simulación a realizar son:

1. **Vista de modelos de consenso**: Muestra los modelos de consenso disponibles junto con una descripción de los mismos. Permite seleccionar el modelo a emplear en la simulación. Los modelos en verde son modelos que usan la estructura seleccionada en el gdmp, mientras que los que están en rojo no pueden usarse, ya que son modelos preparados para trabajar con estructuras de diferente tipo.
2. **Vista de configuración**: Permite configurar los diferentes parámetros del modelo de consenso seleccionado.
3. **Vista de configuraciones**: Permite gestionar las diferentes configuraciones disponibles para los modelos de consenso.
4. **Vista de comportamientos**: Muestra los patrones de comportamiento disponibles para utilizar en los modelos de consenso que emplean un mecanismo de feedback. Permite seleccionar el comportamiento a emplear en estos modelos.
5. **Vista de actitud**: Permite configurar los parámetros del comportamiento seleccionado para definir la actitud de los expertos. Cabe destacar que si queremos que todos los expertos acepten todas las valoraciones sería necesario modificar el parámetro p=1 y el parámetro fix=1.
6. **Vista de actitudes**: Permite gestionar las diferentes actitudes disponibles para los comportamientos.
7. **Vista de pila de simulación**: Informa sobre el estado de la simulación a ejecutar. Esta vista es compartida con la perspectiva de PTDGs.

A continuación se revisa la funcionalidad de las vistas específicas de esta perspectiva, esto es, las vistas de modelos de
consenso, configuración, configuraciones, comportamientos, actitud y actitudes. La vista de pila de simulación está documentada en la sección de la web dedicada a las simulaciones.

3.4.2. Vista de modelos de consenso

La vista de modelos de consenso ofrece un listado de los modelos de consenso implementados en AFRYCA y permite seleccionar el modelo a emplear para llevar a cabo la simulación de un proceso de alcance de consenso.



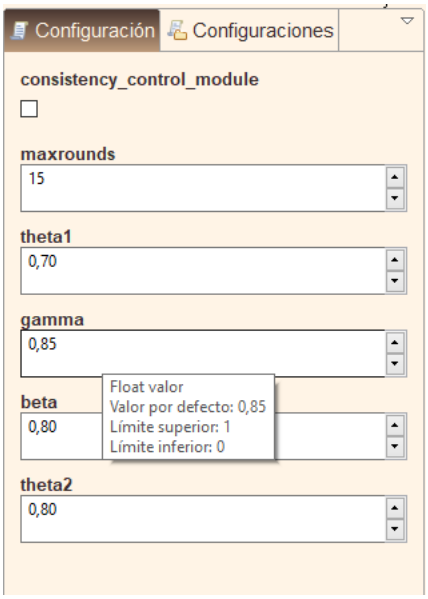
39-Vista de modelos de consenso

La vista está dividida en dos columnas, las cuales de izquierda a derecha son las siguientes:

1. **Lista de modelos de consenso**: Muestra los modelos de consenso disponibles y permite seleccionar cualquiera de ellos. La lista se subdivide en dos categorías:
   A. **Con feedback**: Modelos de consenso que emplean un mecanismo de feedback en el cual se realizan recomendaciones a los expertos para que modifiquen sus valoraciones a fin de incrementar el grado de consenso. Debido a que AFRYCA es una herramienta enfocada en la simulación y el análisis de los procesos de alcance de consenso, el comportamiento de los expertos ante las recomendaciones es modelado haciendo uso de diferentes patrones de comportamiento estadísticos que pueden seleccionarse en la vista de comportamientos.
   B. **Sin feedback**: Modelos de consenso que utilizan mecanismos automáticos para la actualización de la información y que no requieren que los expertos vuelvan a participar en el proceso una vez iniciado. Como es lógico, para este tipo de modelos no es necesario establecer el comportamiento de los expertos.
2. **Información del modelo**: Muestra diferente información sobre el modelo de proceso de alcance de consenso seleccionado.
   A. **Nombre del modelo**: Nombre dado al modelo. Usualmente los modelos se corresponden con publicaciones científicas, por lo que los identificadores empleados son similares a referencias bibliográficas.
   B. **Abstract** - **Descripción**: Datos básicos del modelo tales como artículos en los que está basado o una síntesis de su funcionamiento.
   C. **Características**: Puntos clave del modelo.
   D. **Observaciones**: Advertencias sobre peculiaridades del modelo, cambios realizados sobre el mismo por motivos prácticos o de implementación, etc.

3.4.3. Vista de configuración

La vista de configuración permite configurar los diferentes parámetros del modelo de consenso seleccionado.



40-Vista de configuración

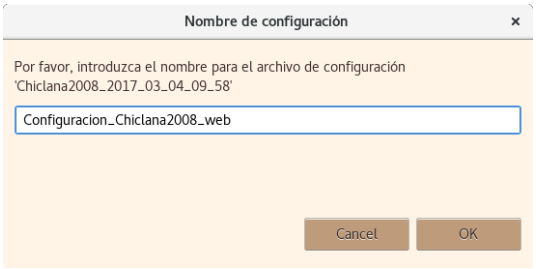
La vista se divide en diferentes controles etiquetados con el nombre del parámetro correspondiente, variando su apariencia en función de múltiples aspectos tales como la naturaleza del parámetro, el valor por defecto establecido para el mismo en la implementación del modelo o su valor en relación a otros parámetros. En concreto, para cada parámetro de la configuración del modelo, la vista incluye un control etiquetado con el siguiente funcionamiento:

- **Etiqueta**: Nombre del parámetro según se define en el artículo correspondiente en el que se ha propuesto el modelo. Si se coloca el cursor del ratón sobre la etiqueta se mostrará una ventana emergente con una descripción del parámetro.
- **Valor**: Control en el que establecer el valor del parámetro. Su formato y contenido variará atendiendo a diferentes criterios.
  - **Tipo**: El control puede ser una *casilla de activación* para valores booleanos, un *cuadro de texto* para listas de valores separados por el carácter '**;**', o un *spinner* (tipo de control numérico que permite incrementar y decrementar los valores contenidos) para valores numéricos.
  - **Valor por defecto**: Valor por defecto dado al parámetro tomado del artículo en el que se ha propuesto el modelo.
  - **Restricciones**: Limitación establecida sobre el parámetro para el correcto funcionamiento del modelo. Las restricciones pueden darse tanto a nivel de valores individuales como a nivel de listas de valores.
    - **Individuales**
      - Límite inferior
      - Límite superior
    - **Listas de valores**
      - Límite inferior de un elemento
      - Límite superior de un elemento
  - **Relaciones**: Relación de un parámetro con el resto. Al igual que las restricciones, las relaciones pueden darse tanto a nivel de valores individuales como a nivel de listas de valores.
    - **Individuales**
      - Igual a
      - Menor a
      - Mayor a
      - Menor o igual a
      - Mayor o igual a
    - **Listas de valores**
      - Número de elementos igual a
  - **Color**: Un control que contenga un parámetro válido se mostrará de modo habitual, sin embargo, un control que contenga un valor que incumpla alguna de las relaciones o restricciones impuestas sobre el parámetro se mostrará de color rojo.
  - **Ventana emergente**: Colocando el cursor del ratón sobre el control se mostrará una ventana emergente con diferente información como su tipo, valor por defecto, relaciones, restricciones o problemas existentes en caso de que los hubiese.

41-Configuración inválida

Guardar una configuración

Si se cuenta con permisos de escritura en el espacio de trabajo, cualquier cambio realizado sobre la configuración por defecto del modelo seleccionado podrá guardarse en el espacio de trabajo invocando el comando '**guardar configuración**'. Al invocar el comando se mostrará una ventana a través de la cual AFRYCA solicita el identificador de la configuración a almacenar. En caso de que la modificación se haya realizado sobre una configuración previamente guardada que no se corresponda con un elemento nativo de la aplicación, se dará la opción de sobrescribir la configuración guardada. Téngase en cuenta que el nombre de la configuración es el identificador único de la misma, por lo que no es posible emplear un identificador en uso aunque sea utilizado para guardar la configuración de un modelo diferente.



42-Solicitud de nombre para configuración

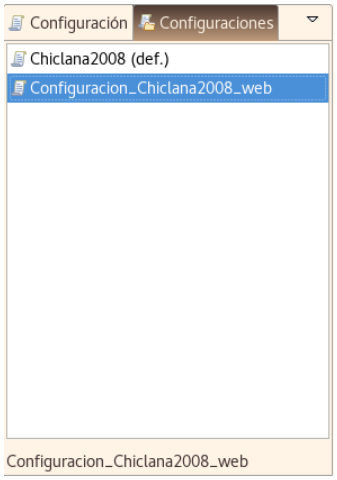
Pulsando el botón '**OK**' se guardará la configuración en el espacio de trabajo.

Restaurar una configuración

Si se han realizado modificaciones en una configuración y se desea volver a la configuración original, esto puede hacerse rápidamente invocando el comando '**restaurar configuración**'.

3.4.4. Vista de configuraciones

La vista de configuraciones permite navegar entre las configuraciones disponibles en el espacio de trabajo para el modelo seleccionado y escoger la deseada.



43-Vista de configuraciones

La vista siempre mostrará una *configuración virtual* con el sufijo '*(def.)*' correspondiente a la configuración por defecto del modelo. Dicha configuración es seleccionada de modo automático cada vez que se modifica el modelo de consenso seleccionado.

Selección de una configuración

Pulsando sobre cualquiera de las configuraciones mostradas esta pasará a ser la configuración seleccionada, lo cual implica que:

- Si se realiza una simulación el modelo tomará como parámetros iniciales los establecidos en esta configuración.
- Si la configuración anteriormente seleccionada contenía cambios estos **serán descartados** y no podrán ser recuperados.
- Cualquier enlace a módulo en el que la configuración seleccionada tenga relevancia será recalculado.

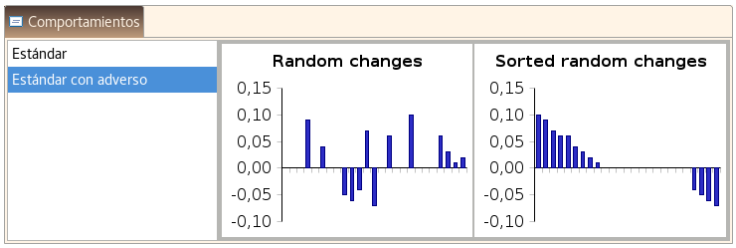
Eliminación de una configuración

Si se ha seleccionado una configuración que no se corresponde con un elemento nativo ni con la configuración por defecto y se cuenta con permisos de escritura en el espacio de trabajo, la configuración puede ser eliminada invocando el comando '**eliminar configuración**'. Tras invocar este comando y confirmar que se desea eliminar la configuración esta será eliminada del espacio de trabajo y será seleccionada de modo automático la configuración por defecto del modelo seleccionado.

3.4.5. Vista de comportamientos

La vista de comportamientos ofrece el listado de los patrones de comportamiento disponibles. AFRYCA emplea comportamientos basados en distribuciones de probabilidad estadísticas para simular la actitud de los expertos en aquellos procesos de alcance de consenso que hacen uso de un mecanismo de feedback y solicitan a los expertos que modifiquen sus valoraciones para incrementar el grado de consenso. Desde la vista de comportamientos es posible seleccionar aquel

patrón de comportamiento que, en caso de realizar una simulación empleando un modelo de consenso con feedback, será utilizado.



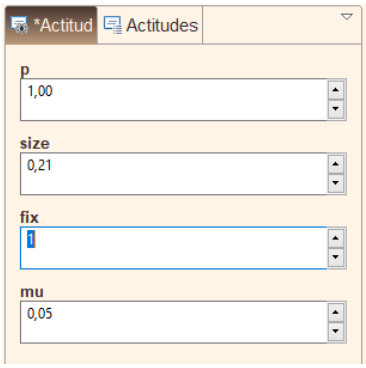
44-Vista de comportamientos

La vista se divide en dos columnas, las cuales de izquierda a derecha son:

1. **Lista de comportamientos**: Muestra los comportamientos disponibles y permite seleccionar cualquiera de ellos.
2. **Gráficos ilustrativos**: Ilustran cuales serían los posibles cambios realizados por un experto cuando se le solicitase realizar modificaciones en sus valoraciones. Cualquier cambio en la actitud del comportamiento provocará que los gráficos sean recalculados.
   A. **Random changes** - **Cambios aleatorios**: Muestra cual sería el cambio realizado por el experto ante 25 solicitudes de cambio, representando:
      - Un valor igual a 0 que el experto no acepta realizar el cambio.
      - Un valor superior a 0 que el experto acepta realizar el cambio que se le solicita (incrementar o decrementar su valoración).
      - Un valor inferior a 0 que el experto actúa de forma opuesta a lo que se le solicita, es decir, incrementando el valor si se le solicita que lo decremente o decrementandolo si se le solicita que lo incremente.
      - El valor absoluto del valor la magnitud del cambio realizado.
   B. **Sorted random changes** - **Cambios aleatorios ordenados**: Muestra los valores de la gráfica '**Random changes**' siguiendo un orden decreciente.

3.4.6. Vista de actitud

La vista de actitud es muy similar a la vista de configuración. La vista permite configurar la actitud de los expertos ante las recomendaciones mediante el ajuste de los parámetros de las distribuciones de probabilidad asociados.



45-Vista de actitud

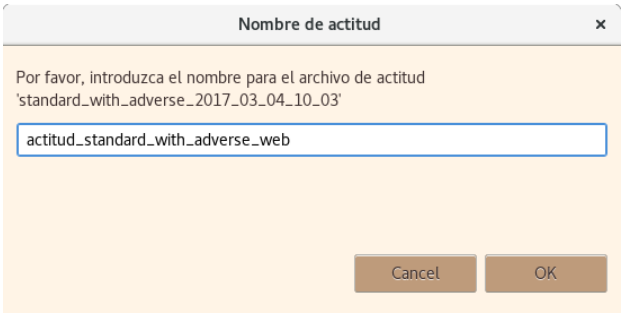
La vista se divide en diferentes controles etiquetados con el nombre del parámetro correspondiente, incluye para cada uno un control etiquetado con el siguiente funcionamiento:

- **Etiqueta**: Nombre del parámetro. Si se coloca el cursor del ratón sobre él se mostrará una ventana emergente con una descripción del parámetro.
- **Valor**: Control en el que establecer el valor del parámetro.

  Nota: Si fijamos p=1 y fix=1 se establecerá que todos los usuarios aceptan el cambio. Estos valores son importantes para comprar diferentes modelos de consenso, en los que los cambios son fijos.

Guardar una actitud

Al igual que las configuraciones de los modelos de consenso, si se cuenta con permisos de escritura en el espacio de trabajo, cualquier cambio realizado sobre la actitud por defecto del comportamiento seleccionado podrá guardarse en el espacio de trabajo invocando el comando '**guardar actitud**'. Al invocar el comando se mostrará una ventana a través de la cual AFRYCA solicita el identificador de la actitud a almacenar. En caso de que la modificación se haya realizado sobre una actitud previamente guardada que no se corresponda con un elemento nativo de la aplicación, se dará la opción de sobrescribir la actitud guardada. Téngase en cuenta que el nombre de la actitud es el identificador único de la misma, por lo que no es posible emplear un identificador en uso aunque sea utilizado para guardar la actitud de un comportamiento diferente.



46-Solicitud de nombre para actitud

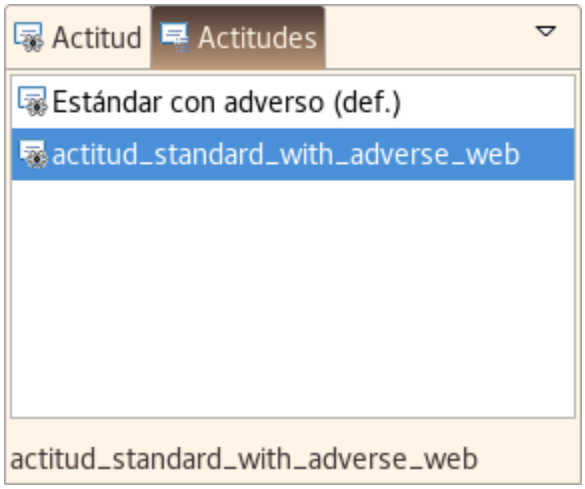
Pulsando el botón '**OK**' se guardará la actitud en el espacio de trabajo.

Restaurar una actitud

Si se han realizado modificaciones en una actitud y se desea volver a la actitud original, esto puede hacerse invocando el comando '**restaurar actitud**'.

3.4.7. Vista de actitudes

La vista de actitudes es el equivalente de la vista de configuraciones para comportamientos. La vista permite navegar entre las actitudes disponibles en el espacio de trabajo para el comportamiento seleccionado y escoger la deseada.



47-Vista de actitudes

La vista siempre mostrará una *actitud virtual* con el sufijo '*(def.)*' correspondiente a la actitud por defecto del comportamiento. Dicha actitud es seleccionada de modo automático cada vez que se modifica el comportamiento seleccionado.

Selección de una actitud

Pulsando sobre cualquiera de las actitudes mostradas esta pasará a ser la actitud seleccionada, lo cual implica que:

- Si se realiza una simulación en la que se emplea un modelo de consenso con feedback, el comportamiento usado en el modelo tomará como parámetros iniciales los establecidos en esta actitud.
- Si la actitud anteriormente seleccionada contenía cambios estos **serán descartados** y no podrán ser recuperados.
- Los gráficos ilustrativos del comportamiento serán recalculados.
- Cualquier enlace a módulo en el que la actitud seleccionada tenga relevancia será recalculado.

Eliminación de una actitud

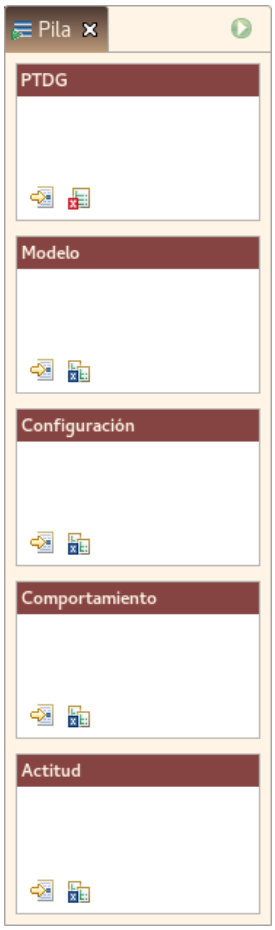
Si se ha seleccionado una actitud que no se corresponde con un elemento nativo ni con la actitud por defecto y se cuenta con permisos de escritura en el espacio de trabajo, la actitud puede ser eliminada invocando el comando '**eliminar actitud**'. Tras invocar este comando y confirmar que se desea eliminar la actitud esta será eliminada del espacio de trabajo y será seleccionada de modo automático la actitud por defecto del comportamiento seleccionado

3.5. Simulaciones

Establecido el PTDG a resolver, seleccionado y configurado el modelo de consenso a emplear y en caso necesario, fijado el comportamiento y su actitud, tan solo resta simular el proceso de alcance de consenso.
En esta entrada de la web son revisadas las funcionalidades que AFRYCA ofrece al usuario tanto para llevar a cabo una simulación como para analizar sus resultados. Para ello, en primer lugar se describe la vista de pila de simulación, la cual ofrece diferente información sobre la simulación a realizar y facilita su gestión. Posteriormente se recorrerán las dos perspectivas disponibles para el análisis de las simulaciones, la perspectiva de simulaciones y la perspectiva de simulación.

3.5.1. La pila de simulación

La pila de simulación es una vista que ofrece información sobre la simulación a realizar y que está disponible en las perspectivas de PTDG y de modelos de consenso. A través de esta vista es posible gestionar los diferentes elementos involucrados en la simulación. En concreto, la vista ofrece cinco **bloques** apilados verticalmente, representando cada uno de ellos un elemento de la simulación.



48-Pila de simulación

Los bloques contenidos en la vista, recorriendo esta de arriba a abajo son los siguientes:

1. **PTDG**: PTDG seleccionado.
2. **Modelo**: Modelo de consenso seleccionado.
3. **Configuración**: Configuración del modelo de consenso seleccionado.

4. **Comportamiento**: Patrón de comportamiento seleccionado.
5. **Actitud**: Actitud del patrón de comportamiento seleccionado.

Cada uno de los bloques que conforma la pila ofrece diferente información sobre el estado del elemento representado:

- **Nombre del bloque**: Identificador del elemento gestionado en el bloque.
- **Valor de la selección**: Identificador del elemento seleccionado. En blanco si no existe selección.
- **Enlace rápido**: Vínculo a la vista que permite realizar la selección del elemento gestionado en el bloque. Su activación puede implicar un cambio de la perspectiva visible.
- **Estado del elemento**: Información sobre la validez del elemento. Existen tres estados:
  - **Válido**: El elemento es válido para la simulación. Se representa con un icono en el que se incluye una marca de color verde.
  - **Inválido**: El elemento no es válido para la simulación por algún motivo. Se representa con un icono en el que se incluye una marca de color rojo. Si se coloca el cursor del ratón sobre la etiqueta se mostrará una ventana emergente con una descripción del problema.
  - **Indiferente**: Debido a los estados de los elementos previos de la pila el valor del elemento es indiferente para la simulación. Se representa con un icono en el que se incluye una marca de color azul.

La siguiente imagen muestra una pila en la que se pueden observar elementos en los tres estados. Su lectura sería como sigue:

1. Se ha seleccionado el PTDG '**Human resources department**' y es ***válido***.
2. Se ha seleccionado el modelo de consenso '**Chiclana2008**' y es ***válido***.
3. El modelo de consenso se ha configuración con la configuración '**Chiclana2008 (def.)**' y es ***válida***.
4. El comportamiento no se ha seleccionado y es ***inválido***. Esto indica que el modelo de consenso seleccionado utiliza un mecanismo de feedback y requiere que se seleccione un comportamiento.
5. La actitud no se ha seleccionado y su estado es ***indiferente*** al no haberse seleccionado un comportamiento.

49-Indicación de que es necesario seleccionar un comportamiento en la pila

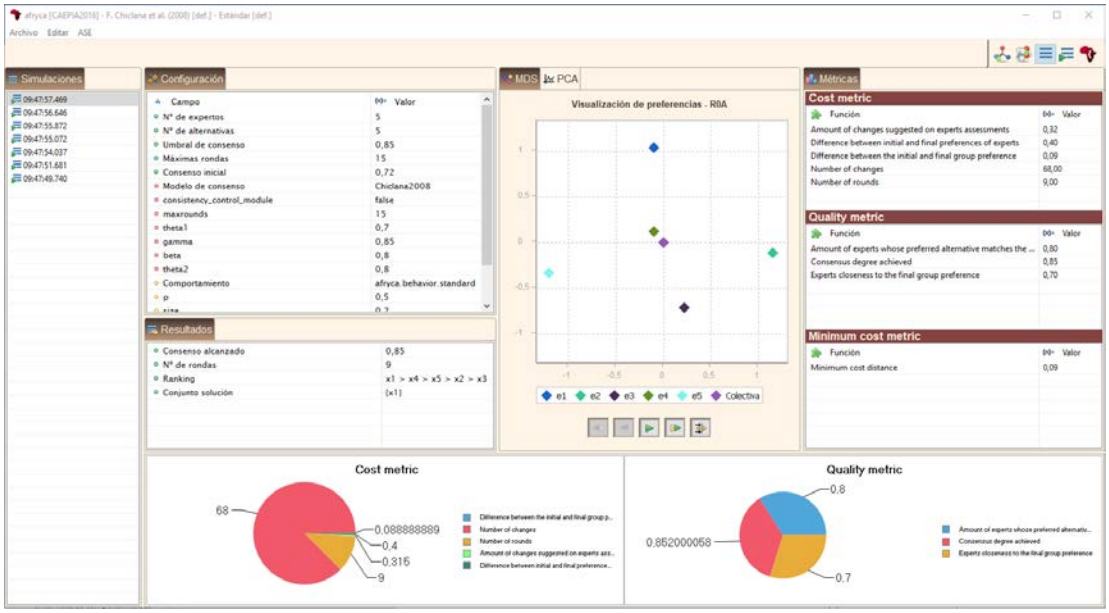
Mientras en la pila de simulación exista un elemento en el estado ***inválido*** no será posible realizar una simulación y el comando '**simular**' no podrá ser invocado. Si todos los elementos de la pila están en el estado ***válido*** o ***indiferente*** el comando '**simular**' estará activo y podrá ser invocado.

50-Pila de simulación válida

Al invocar el comando '**simular**' AFRYCA realiza la simulación configurada en la pila y una vez finalizada, mostrará una ventana de diálogo informando de la finalización del proceso y del identificador de la simulación, el cual se corresponderá con la hora exacta en la que se completó la simulación.
Mencionar que debido al fallo 400771 de la versión actual de Eclipse RCP, no es posible inhabilitar el botón cerrar en las vistas compartidas. En caso de cierre accidental, la vista puede ser recuperada nuevamente eliminando el workspace de Eclipse RCP, para lo cual remitimos a la sección de despliegue de AFRYCA de la guía de instalación para linux o windows.

3.5.2. Perspectiva de simulaciones

La perspectiva de simulaciones ofrece al usuario diferente información para el análisis de las simulaciones realizadas.



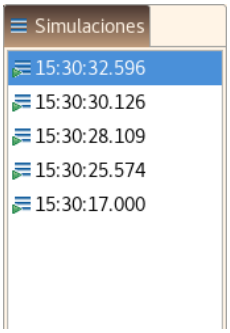
51-Perspectiva de simulaciones

La perspectiva contiene siete vistas, algunas de ellas configurables por el usuario a través de las preferencias. Las funciones que cumplen las vistas para el análisis de las simulaciones son:

1. **Vista de simulaciones**: Muestra las simulaciones realizadas y permite seleccionar la que se desee analizar.
2. **Vista de configuración**: Muestra los datos de configuración de la simulación seleccionada.
3. **Vista de resultados**: Muestra los resultados de la simulación seleccionada.
4. **Vista de MDS**: Permite visualizar las preferencias de los expertos utilizando en escalado multidimensional.
5. **Vista de métricas**: Ofrece diferentes métricas de análisis de la simulación.
6. **Vista de gráficas**: Gráficas de la simulación.

3.5.3. Vista de simulaciones

La vista de simulaciones ofrece un listado con las simulaciones realizadas desde la ejecución de la aplicación ordenadas por su hora de ejecución.

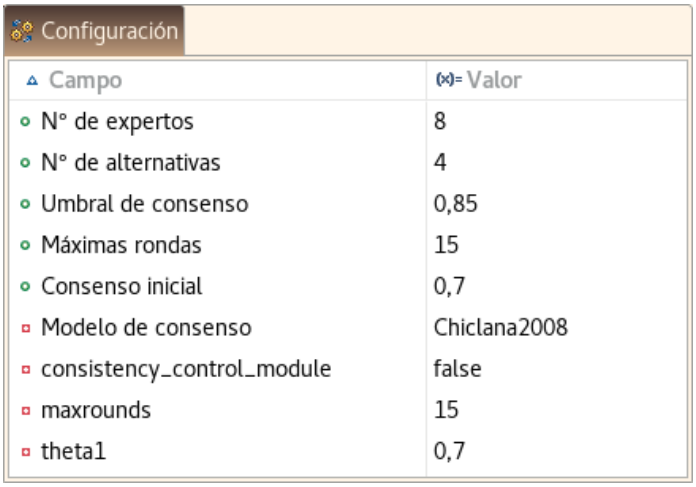


52-Vista de simulaciones

La vista se refresca automáticamente cada vez que es realizada una nueva simulación, la cual es seleccionada de forma automática.

3.5.4. Vista de configuración

La vista de configuración muestra los datos de la configuración de la simulación seleccionada.



53-Vista de configuración

No se debe confundir la configuración del modelo de consenso con la configuración de la simulación. La diferencia es simple:

- **Configuración del modelo de consenso**: Valores de configuración para los parámetros del modelo de consenso usado en la simulación.
- **Configuración de la simulación**: Suma de los datos del PTDG, el modelo de consenso, su configuración, el comportamiento utilizado y su actitud.

La vista de configuración emplea diferentes iconos para denotar la procedencia de cada uno de los valores mostrados:

- Un círculo verde identifica un valor relativo al PTDG.
- Un cuadro rojo un valor asociado al modelo de consenso.
- Un triángulo amarillo un valor vinculado al patrón de comportamiento.

3.5.5. Vista de resultados

La vista de resultados muestra los valores obtenidos al finalizar la simulación.

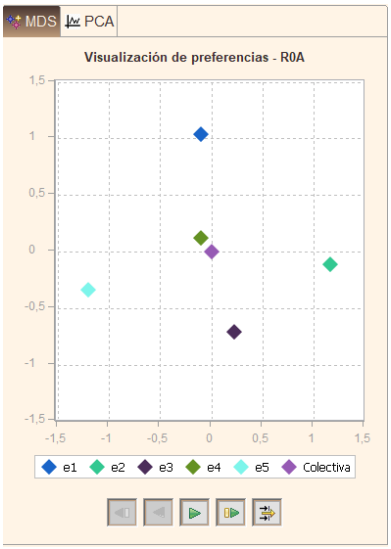
54-Vista de resultados

En concreto, la vista ofrece la siguiente información:

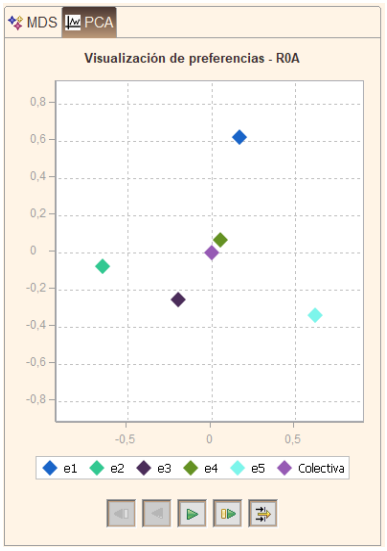
- **Consenso alcanzado**: Grado de consenso alcanzado.
- **Nº de rondas**: Número de rondas de consenso realizadas.
- **Ranking**: Ranking de alternativas calculado a partir de los grados de no-dominancia.
- **Conjunto solución:** Conjunto de alternativas que constituyen la solución al problema.

3.5.6. Vista de visualización de las preferencias MDS/PCA

La vista de la visualización de las preferencias permite visualizar de forma sencilla las preferencias de los expertos que toman parte en el proceso y su evolución durante el mismo, para esto podemos usar MDS (Multi-Dimensional Scaling) o PCA.



56-Visualización MDS



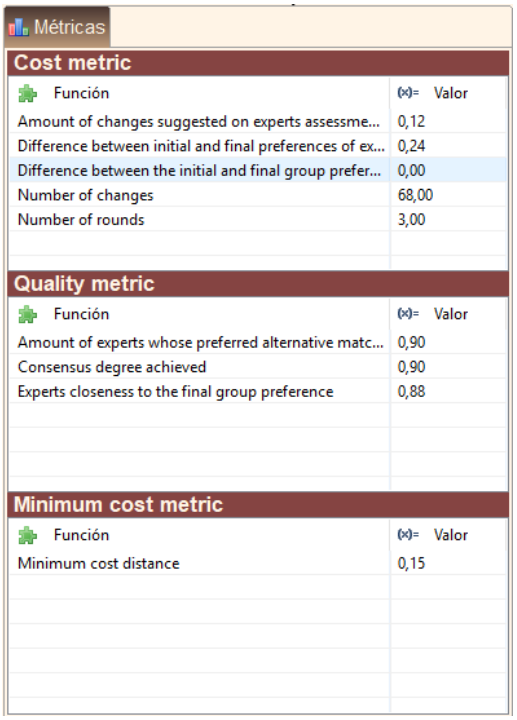
55-Visualización PCA

Para la visualización de la ronda inicial existen dos visualizaciones diferentes, la ronda 0A y 0B. La ronda 0A ofrece una visualización con una agrupación igual para problemas diferentes, mientras que la ronda 0B es la agrupación utilizada en el modelo de consenso. Con esto podemos comprar diferentes problemas viendo cómo sería la ronda inicial usando un mismo algoritmo de agrupación y cómo se trata en el modelo de consenso.

La vista ofrece la posibilidad de exportar la gráfica generada en cualquiera de las rondas. No obstante, debido a un falló en la implementación de SWTChartpara GTK, la exportación no es correcta en las actuales versiones de AFRYCA para linux.

3.5.7. Vista de métricas

La vista de métricas muestra los resultados de las funciones ASE de los módulos '**Cost metric**' , '**Quality metric**' y **Minimum cost metric**' a través de los enlaces a módulos ASE del mismo nombre.

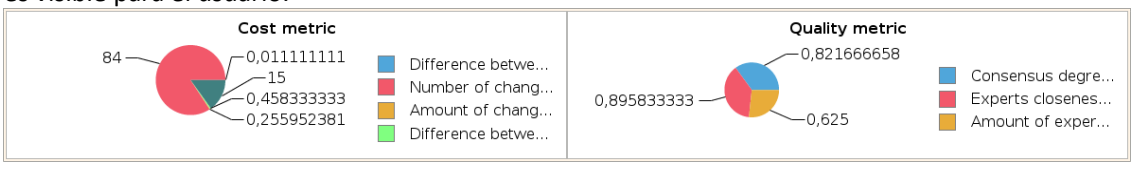


57-Vista de métricas

La versión actual de AFRYCA incluye un conjunto nativo experimental de métricas de ejemplo que puede ser ampliado por el usuario.

3.5.8. Vista de gráficas de simulaciones

La vista de gráficas incluye un conjunto de gráficos BIRT con los que visualizar los parámetros que se considere apropiado de la simulación seleccionada. Por motivos prácticos el título de la vista no es visible para el usuario.

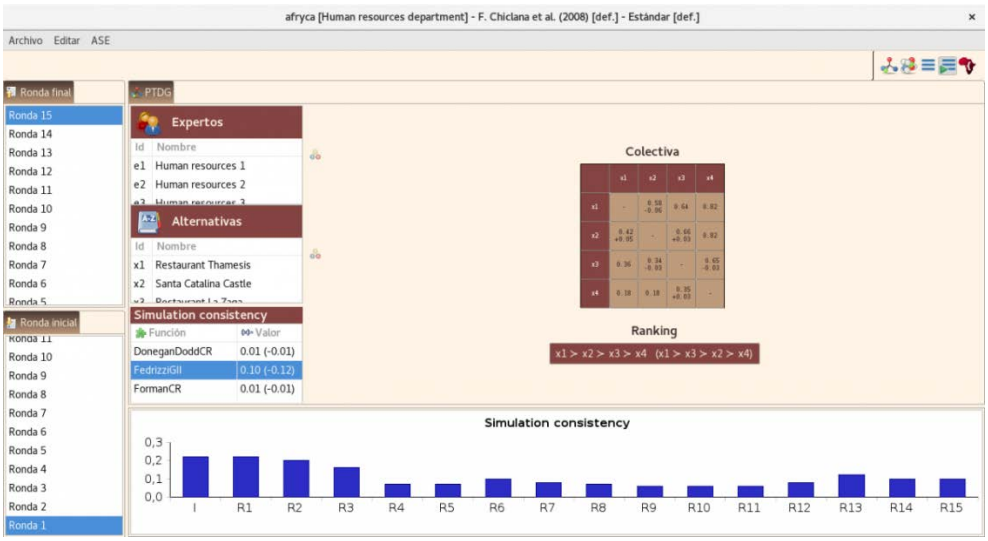


58-Vista de gráficas en perspectiva simulaciones

Las gráficas a mostrar se pueden personalizar desde las preferencias.

3.6. Perspectiva de simulación

La perspectiva de simulación complementa a la perspectiva de simulaciones y se enfoca en el análisis de la evolución de la simulación.



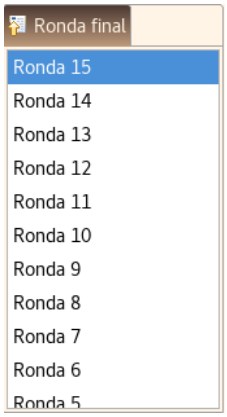
59-Perspectiva de simulación

La perspectiva está formada por cuatro vistas, algunas de ellas configurables por el usuario a través de las preferencias. Las funciones que cumplen las vistas para el análisis de la simulación son:

1. **Vista de ronda final**: Permite seleccionar la ronda final de la simulación.
2. **Vista de ronda inicial**: Permite seleccionar la ronda inicial de la simulación.
3. **Vista de PTDG**: Permite visualizar las preferencias del problema durante la simulación.
4. **Vista de gráficas**: Gráficas de la simulación.

3.6.1. Vista de ronda final

La vista de ronda final permite establecer la última ronda a considerar al analizar el proceso de alcance de consenso.

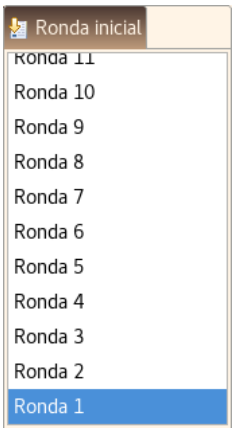


60-Vista de ronda final

Al analizar el proceso de consenso se tomarán como resultados del mismo los valores existentes al finalizar la ronda seleccionada.

3.6.2. Vista de ronda inicial

De forma análoga a la vista de ronda final, la vista de ronda incial permite establecer la primera ronda a considerar al analizar el proceso de alcance de consenso.



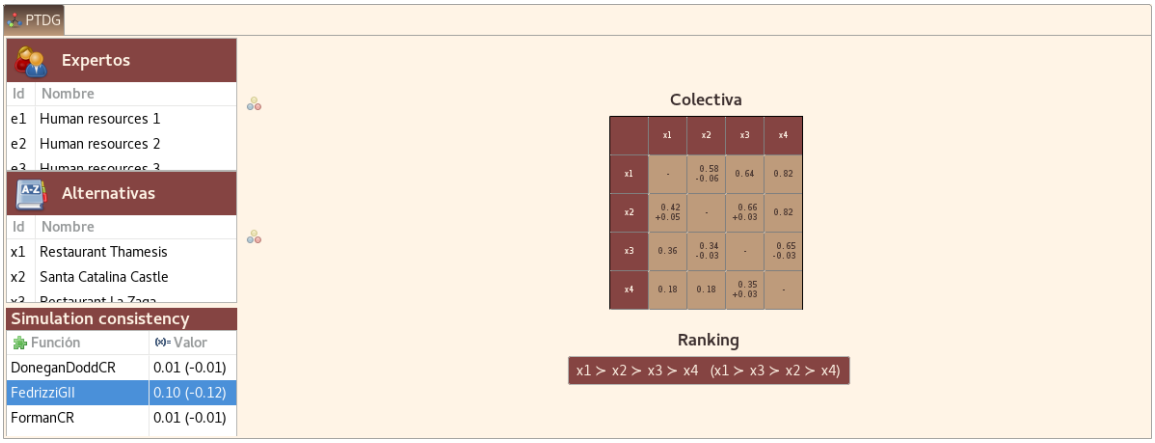
61-Vista de ronda inicial

La ronda establecida determina los valores que serán tomados como valores iniciales en el proceso.

Cabe notar que en AFRYCA, para cada ronda realizada se almacenan tanto los valores de partida como los valores obtenidos al concluir la ronda. Por tanto, al seleccionar la ronda final, los valores que se utilizarán como valores finales serán los obtenidos al concluir la misma. De igual modo, al seleccionar la ronda inicial los valores empleados serán los existentes antes de llevar a cabo dicha ronda. En caso de que la ronda inicial y final coincidan, los valores obtenidos serán relativos a la evolución del proceso en esta ronda.

3.6.3. Vista de PTDG

La vista de PTDG es una versión adaptada de la vista de problema de la perspectiva PTDG. La vista permite visualizar la evolución del PTDG en la simulación seleccionada y comprobar su evolución desde la ronda inicial seleccionada hasta la ronda final establecida.



62-Vista de PTDG

La vista se subdivide en tres zonas, las cuales ofrecen diferente información:

- **Esquina superior izquierda** - **Elementos del problema**: Expertos y alternativas del PTDG.
- **Esquina inferior izquierda** - **Enlaces a módulo**s: Enlaces a módulos seleccionados en las preferencias de simulación.
- **Columna derecha** - **Relación de preferencias**: FPR del experto seleccionado y ranking de alternativas calculado a partir de esta relación de preferencias.

Elementos del problema

Al igual que en la vista de problema de la perspectiva PTDG, los elementos del problema son los expertos que toman parte en el mismo y las alternativas que se valoran. A diferencia de la vista mostrada en la perspectiva PTDG, esta vista no permite realizar modificaciones sobre los elementos del problema, siendo las únicas opciones soportadas la selección y deselección.

Enlaces a módulos

De modo similar a la vista de la perspectiva PTDG, esta vista soporta la visualización de enlaces a módulos ASE a través de los cuales analizar diferentes parámetros de la simulación. Los enlaces a módulos a incluir pueden personalizarse desde las preferencias.

Relación de preferencias

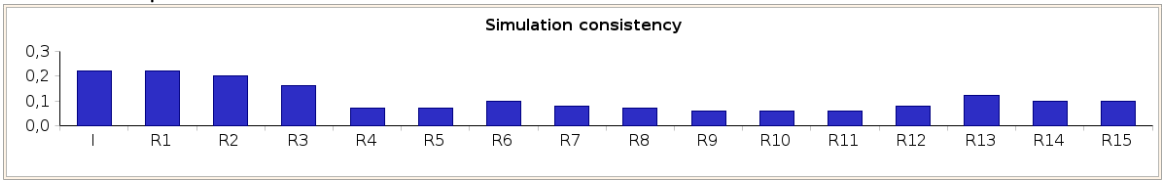
Tanto la tabla de relaciones de preferencias como el ranking de alternativas mostrados en esta vista difieren significativamente de los incluidos en la vista de la perspectiva PTDG. En la presente vista ambos elementos gráficos están adaptados para ilustrar la diferencia existente entre la ronda inicial y la ronda final establecidas:

- **Tabla de relaciones de preferencia**: Para cada par de alternativas se muestra la valoración dada por el experto al concluir la ronda final seleccionada. Si el valor se ha incrementado en relación al existente antes de llevar a cabo la ronda inicial seleccionada, debajo del valor se mostrará la magnitud del incremento con el prefijo '**+**'. Si por el contrario el valor se ha decrementado, se mostrará la magnitud del decremento con el prefijo '**-**'. Si el valor no ha variado, no se mostrará nada más que el valor final.
- **Ranking de alternativas**: Se muestra el ranking de alternativas ***final***, el cual es calculado a partir de los grados de no-dominancia de los resultados existentes al concluir la ronda final seleccionada. También es calculado el ranking de alternativas ***inicial***, considerando los resultados antes de iniciar la ronda inicial seleccionada. Si el ranking inicial difiere con el final,

el inicial será mostrado a la derecha del final entre signos de paréntesis. Si ambos rankings son iguales únicamente se mostrará el final.

3.6.4. Vista de gráficas en perspectiva simulación

Al igual que con la perspectiva de simulaciones, la perspectiva de simulación incluye su propia vista de gráficas en las que es posible incluir un conjunto de gráficos BIRT con los que visualizar los parámetros que se considere apropiado. Igual que con la otra vista, por motivos prácticos el título no es visible para el usuario.



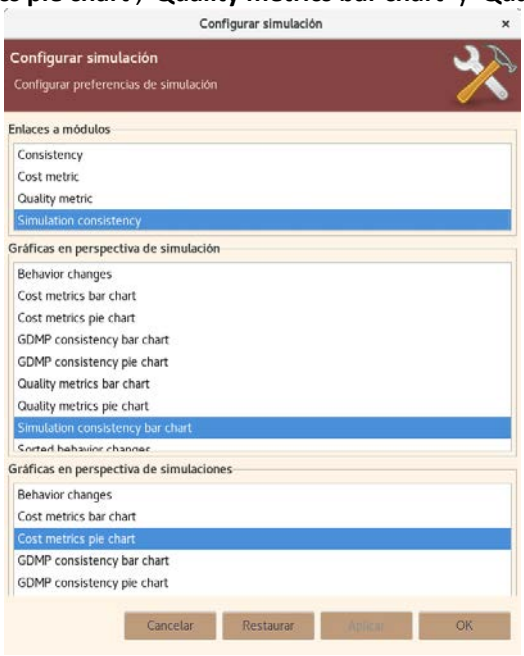
63-Vista de gráficas en perspectiva simulación

Las gráficas a mostrar se pueden personalizar desde las preferencias.

3.6.5. Preferencias

AFRYCA permite personalizar varios aspectos de las simulaciones desde las preferencias. Específicamente, las preferencias proveen soporte para configurar:

- **Enlaces a módulos**: Enlaces a módulos ASE que se mostrarán en la vista de PTDG de la perspectiva de simulación. Por defecto se muestra el enlace a módulo '**Simulation consistency**'.
- **Gráficas en perspectiva de simulación**: Gráficas BIRT que se mostrarán en la vista de gráficas en la perspectiva de simulación. Por defecto se muestra la gráfica '**Simulation consistency bar chart**'.
- **Gráficas en perspectiva de simulaciones**: Gráficas BIRT que se mostrarán en la vista de gráficas en la perspectiva de simulaciones. Por defecto se muestran las gráficas '**Cost metrics bar chart**', '**Cost metrics pie chart**', '**Quality metrics bar chart**' y '**Quality metrics pie chart**'.



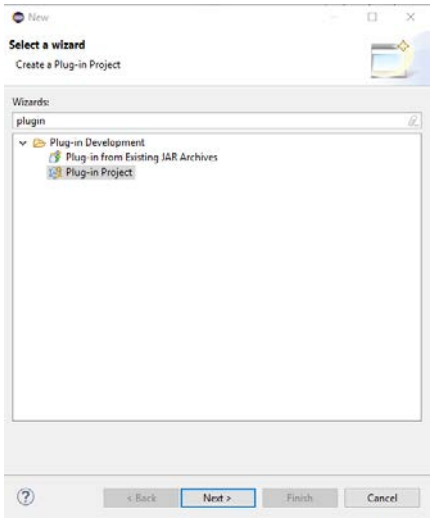
64-Preferencias de simulación

Para modificar las preferencias se ofrece una ventana de diálogo que puede ser abierta desde el menú '**Editar > Preferencias > Simulación**'.

4. Plug-ins

Creación de un nuevo plug-in en Afryca

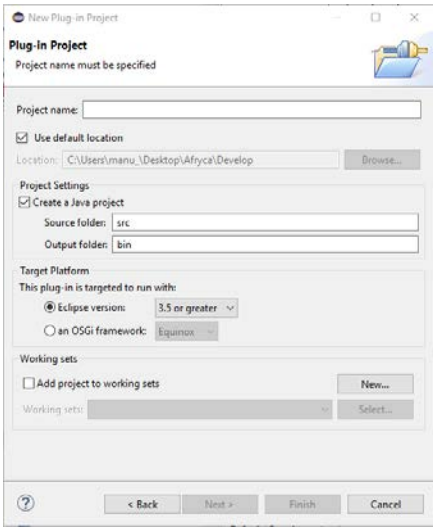
Para la creación de un nuevo plug-in en afryca, le daremos a Nuevo -> Plug-in Project



65-Nuevo Plug-in Project

Pulsamos en Next

EL nombre del plug-in siempre tendrá el formato:
afryca.nombrePlug-in: Para plug-ins con funcionalidad
afryca.nombrePlug-in.gui: Para plug-ins que se encarguen del entorno gráfico
afryca.nombrePlug-in.feature: Para features plug-ins asociados a un plug-in con funcionalidad o a un plug-in encargado del entorno gráfico.



66-Configuración Plug-in Project

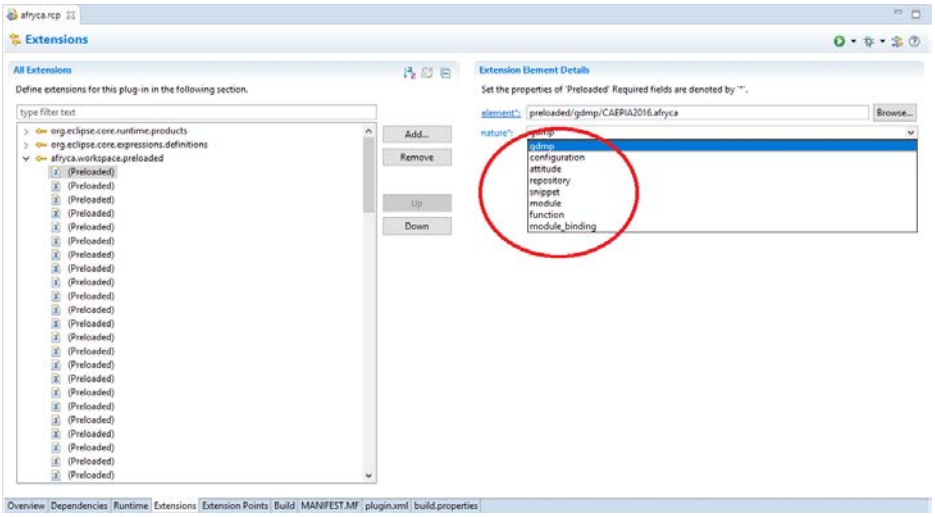
5. Puntos de extensión en afryca

Para añadir una extensión a un punto de extensión, nos iremos al **plugin.xml** de nuestro plug-in, a la pestaña de **Extensions** y pulsaremos sobre el botón **Add**, aquí nos aparecerán los puntos de extensión de nuestro proyecto.

En afryca existen los siguientes puntos de extensión con las siguientes características:

5.1 afryca.workspace.preloaded

Usado en **afryca.rcp**, mediante este punto de extensión podemos añadir ficheros externos precargados a afryca, estos archivos se pueden ser de 8 tipos distintos.

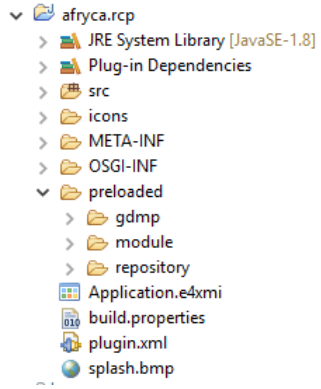


67-Punto de extension afryca.workspace.preloaded

Tipos:

5.1. **gdmp:** Problemas precargados en afryca
5.2. **configuration:**
5.3. **attitude:**
5.4. **repository:**
5.5. **snippet:** Snippets de ASE
5.6. **module:** Módulos de ASE
5.7. **function:** Funciones dentro de módulos de ASE
5.8. **module_binding:** Enlaces a módulos de ASE

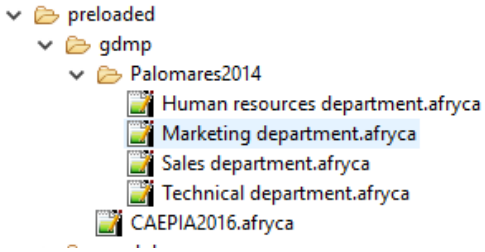
Los nuevos archivos siempre se guardarán en la ruta: **afryca.rcp->preloaded**



68-Archivos precargados

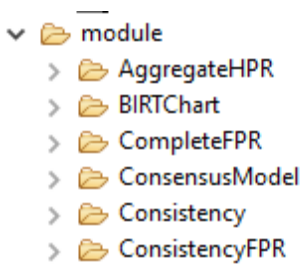
Como vemos existen dentro tres carpetas diferentes, cada una sigue patrones diferentes para su uso.

Dentro de **gdmp** se almacenarán los problemas precargados en afryca.



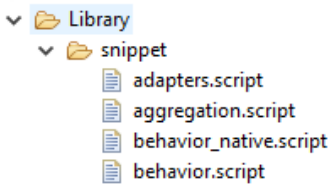
69- Problemas precargados

Dentro de **module** se almacenarán los diferentes módulos creados para ASE.



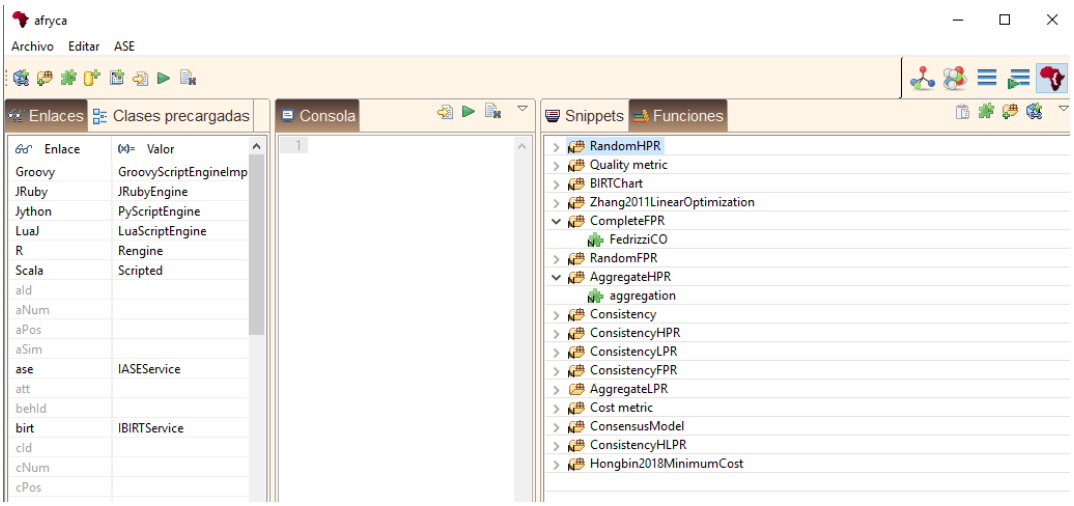
70-Módulos precargados

Por último, dentro de **Library** almacenaremos los snippets.



71-Snippets precargados

Una vez añadidos nos aparecerán en la aplicación

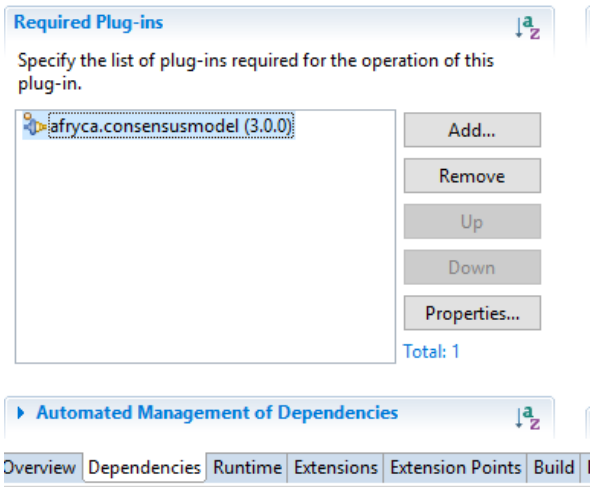


72-Funciones y Módulos

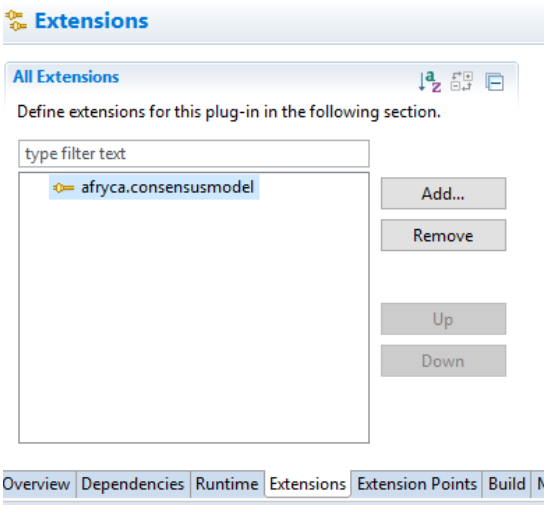
5.2 afryca.consensusmodel

Este punto de extensión se utilizará en cada uno de los modelos de consenso que se añadan a afryca, su uso es el siguiente:

1. Añadimos la dependencia a nuestro plug-in de **afryca.consensusmodel**

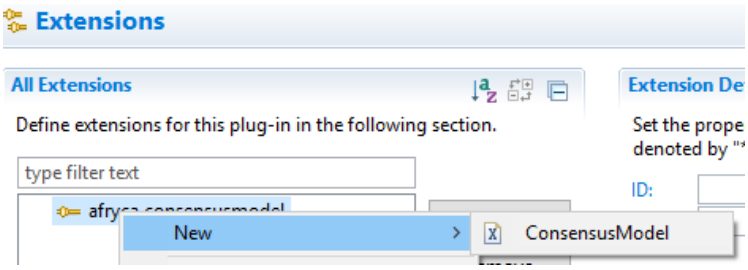


2. Añadimos el punto de extensión **afryca.consensusmodel** a nuestro plug-in



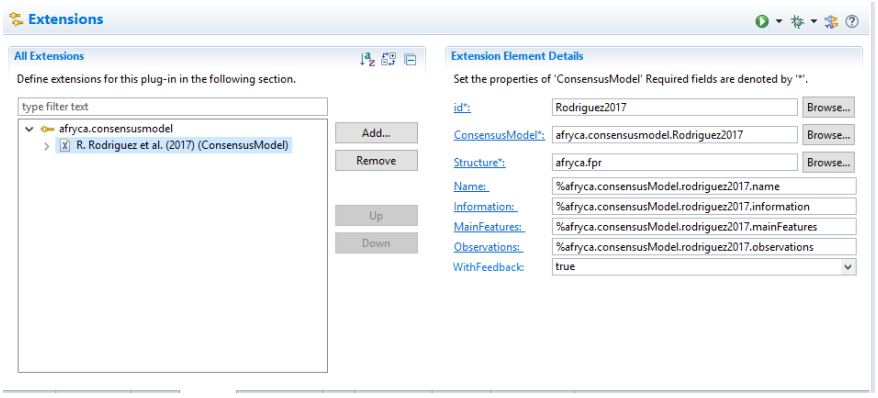
73-Extension afryca.consensusmodel

3. Añadimos los parámetros de configuración del punto de extensión
   Para esto pulsamos con el botón derecho sobre **afryca.consensumodel**, seleccionamos **New -> ConsensusModel**



74-Añadir Modelo de consenso a la extensión

Aquí debemos configurar nuestro plug-in, tenemos que darle un id, la clase java que usará, que será la de nuestro nuevo plug-in, la estructura que usa el modelo de consenso, el nombre, información, características, observaciones y si tiene o no feedback, este último parámetro hará que se nos muestre en la tabla de feedback o en la tabla de sin feedback.

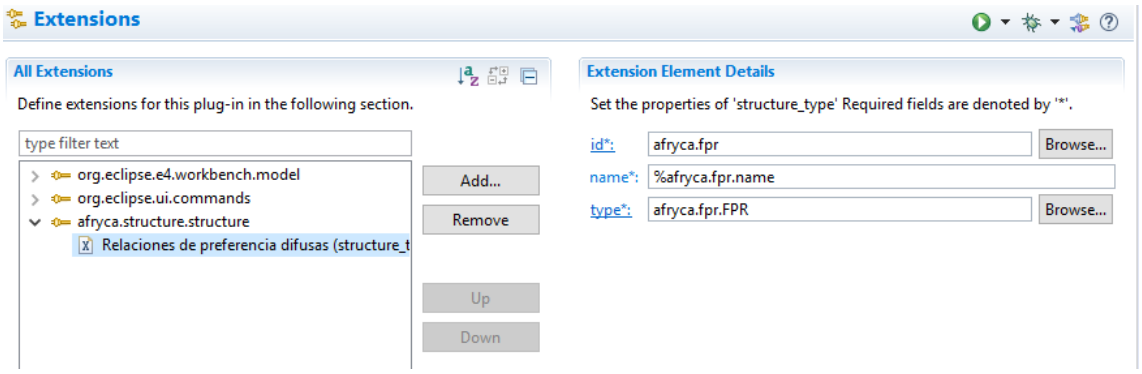


75-Configuración del Modelo de consenso

5.3 afryca.structure.structure

Este punto de extensión será utilizado para añadir una nueva estructura, los parámetros de configuración son los siguientes:

El identificador de la estructura, el nombre y la clase java que será la de nuestro nuevo plug-in.

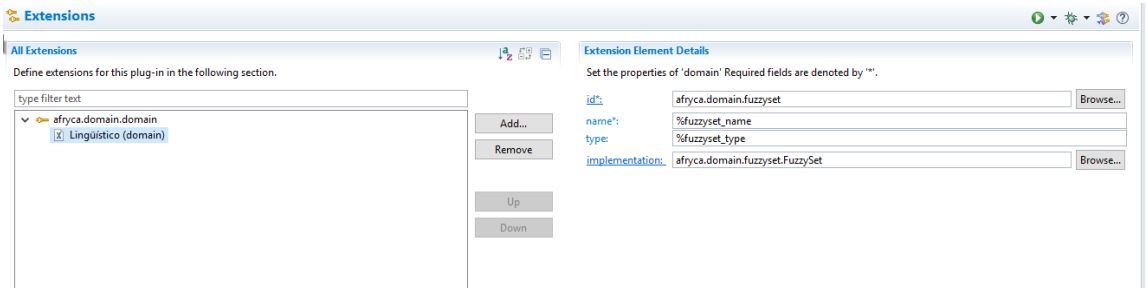


76-Extensión afryca.structure.structure

5.4 afryca.domain.domain

Este punto de extensión será utilizado para añadir nuevos dominios a los ya creados, como son Fuzzyset, Numérico o Real.

Para este debemos introducir el id, el nombre, el tipo y la clase java que se utilizará.



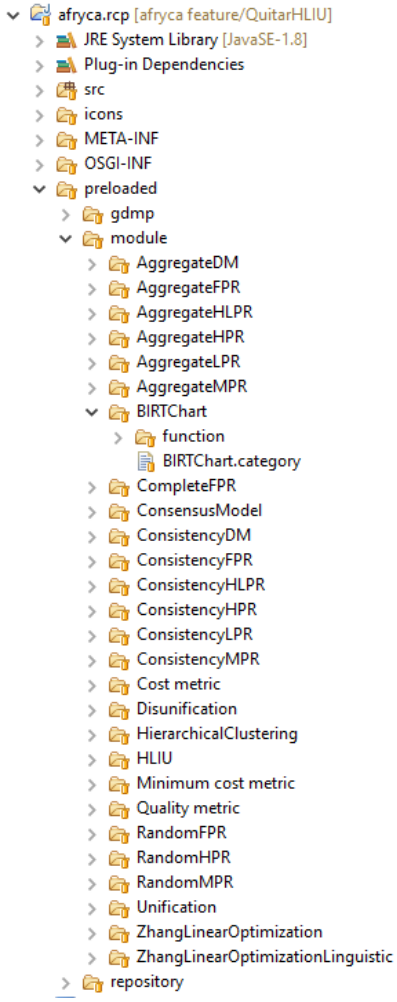
77-Extensión afryca.domain.domain

5.5 afryca.workspace.preloaded

Este punto de extensión está habilitado en el plugin afryca.rcp, mediante él podemos establecer archivos precargados en afryca. Estos archivos serán cargados al ejecutarse AFRYCA y no podrán ser modificados mediante la interfaz. Los tipos de archivos usados para esa extensión son:

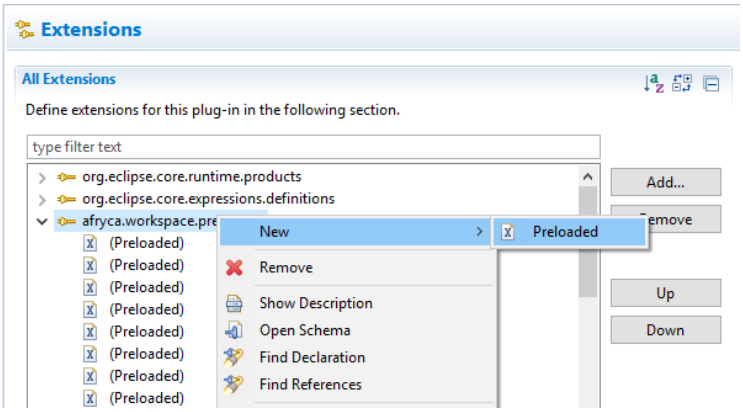
- GDMP
- Configuration
- Attitude
- Repository
- Snippet
- Module
- Function
- Module_binding

Generalmente estos archivos serán almacenados dentro del plugin afryca.rcp en el directorio preloaded, siguiendo la jerarquía ya establecida.



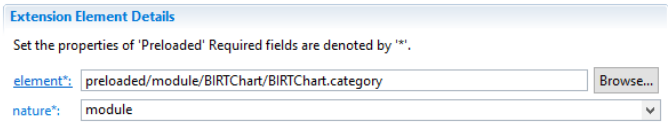
78-Jerarquía Preloaded

Una vez añadido nos iremos a las extensiones del plguin y añadimos un nuevo preloaded.



79-New Preloaded

En element añadimos la ruta del fichero y en nature seleccionamos el tipo de fichero.



80-Element Nature

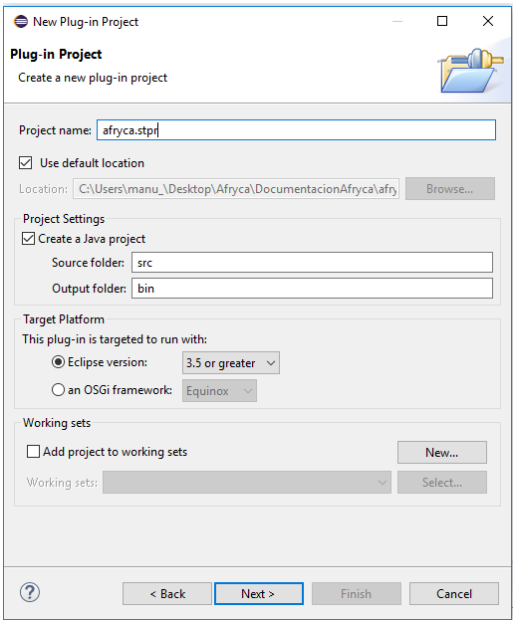
Al ejecutar ASE nos deben aparecer los nuevos elementos añadidos.

6. Añadir una nueva estructura(Structure)

En este capítulo veremos cómo añadir una nueva estructura en afryca.

6.1 Añadiendo una nueva estructura en afryca

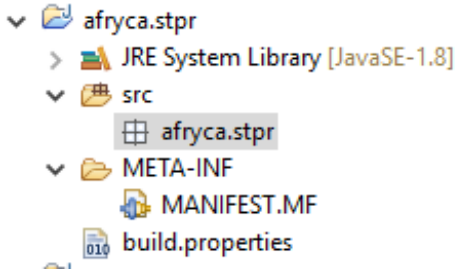
Creamos un nuevo plug-in, en este caso como estructura de prueba se llamará afryca.stpr y pulsamos en next y finish.



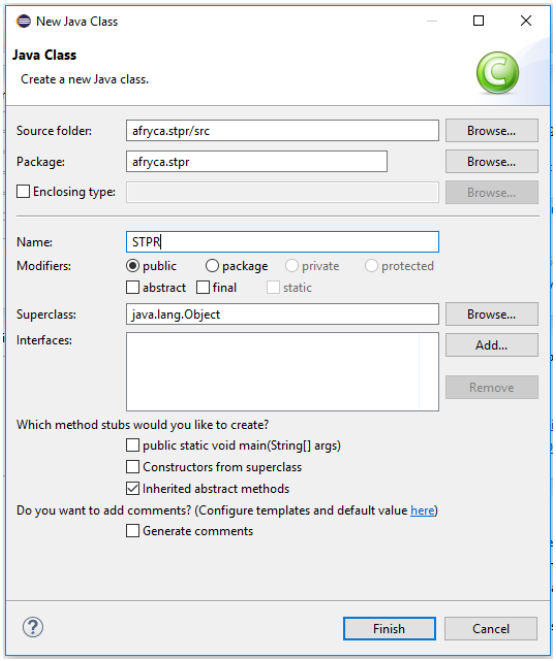
81-Plug-in Project Structure

Una vez hecho esto ya tenemos nuestro plug-in en afryca.

Ahora vamos a crear una clase java que le dará la funcionalidad a la estructura, para esto nos vamos a nuestro plug-in y creamos un nuevo package que se llamará igual que nuestro plug-in, y dentro de este crearemos nuestra clase java, en este ejemplo la llamaremos STPR.java

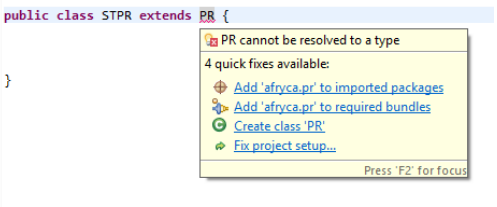


82-Package estrctura



83-Clase estructura

Lo primero que debemos analizar es si nuestra nueva estructura heredará directamente de **structure** o de **pr**, según la funcionalidad que queramos darle, en afryca todas las estructuras como **fpr**, **lpr**, **hpr**, **hlpr** heredan de **PR**, excepto **DECISSIONMATRIX** que hereda de **strcuture** directamente. En este ejemplo lo haremos de PR.



84- Herencia estrcutura

Nos aparecerá como queremos la dependencia del paquete PR, en eclipse existen dos tipos de dependencias, **"imported packages"** y **"required bundles"**, imported packages tan solo importará los paquetes necesarios, mientras que con required blundles creará una dependencia completa del plug-in, con lo que tendríamos el plug-in completo con todos sus paquetes.

Normalmente required bundles se usará para plug-ins creados por nosotros mismo, mientras que imported packages para plug-ins externos usados en la aplicación.

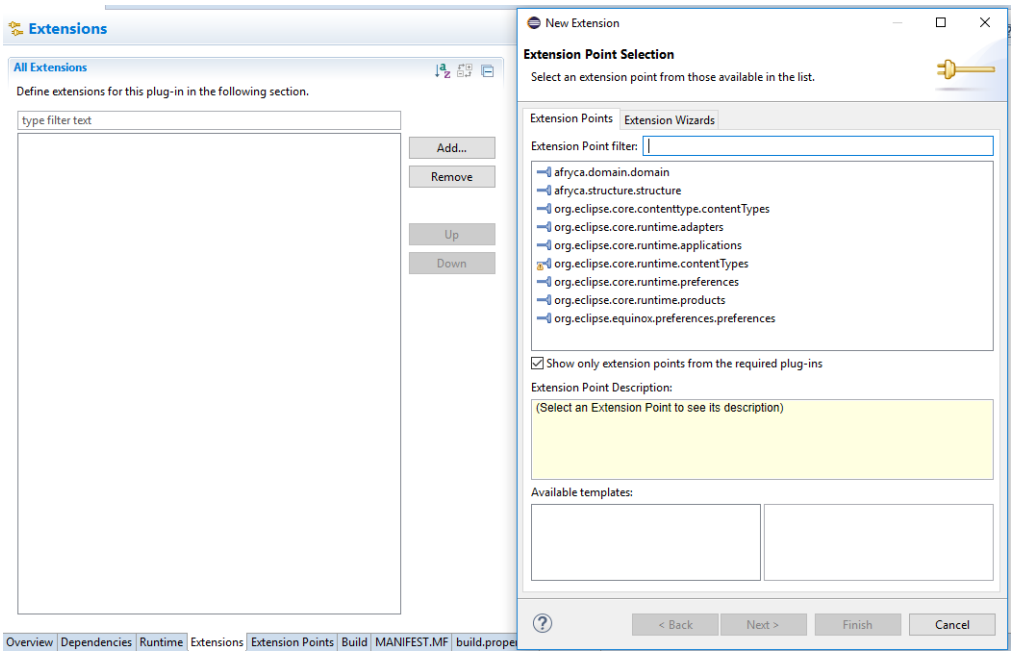
Nosotros elegiremos required bundles, ya que el plug-in PR ha sido creado por nosotros.

Una vez hecho esto añadimos todas las funciones necesarias en nuestra clase.

NOTA: Debemos estudiar la clase **STPR**Parser.java y el método toString() de STPR.java que son los encargados de leer y almacenar desde fichero los problemas creados.

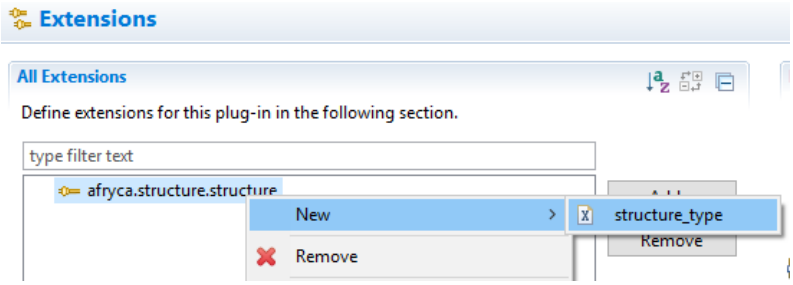
Ahora vamos a añadir nuestra estructura al punto de extensión, este paso se explicó antes, pero lo vuelvo a dejar a continuación.

Nos vamos al MANIFEST.MF de nuestro plug-in, a la pestaña de Extension y pulsamos en Add, y seleccionamos afryca.structure.structure



85-Añadir extensión

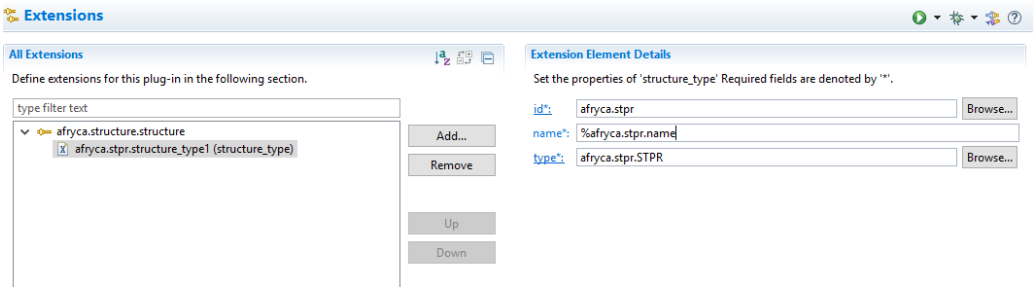
Ahora sobre el punto de extensión botón derecho->New -> structure_type



86-Nueva estructura

Aquí configuraremos nuestra nueva estructura, quedando así:

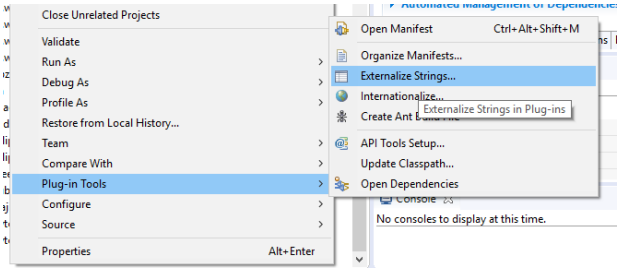
NOTA: si la estructura tiene un dominio, se indicará en el punto de extensión, mediante un **click derecho sobre la structura -> Add -> Domain**



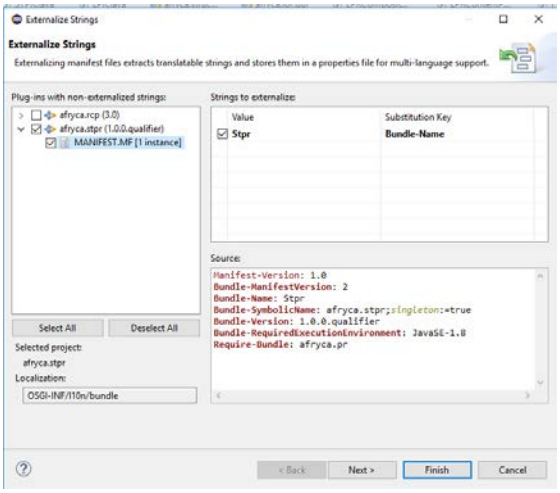
87-Configuración de la estructura

Como vemos, el name utiliza una nomenclatura %afryca….., esto quiere decir que vamos a usar un fichero externo que se encarga de los string para la externalización de cadenas.

Para crear este archivo pulsaremos con el botón derecho en nuestro plug-in -> **Plug-in Tools ->Externalize Strings**

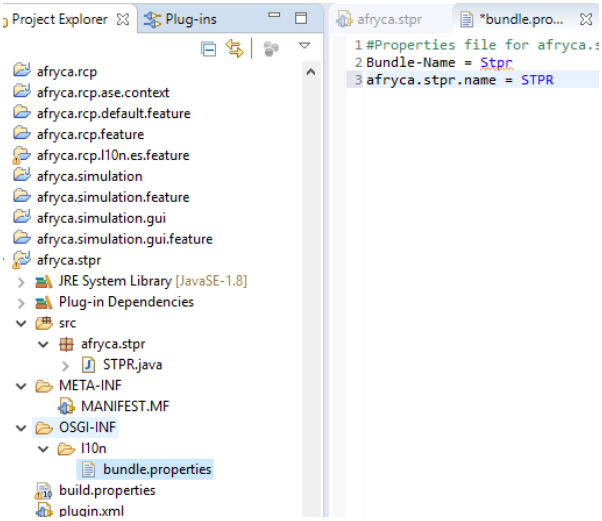


88-Externalize Strings



89-Edición de las cadenas

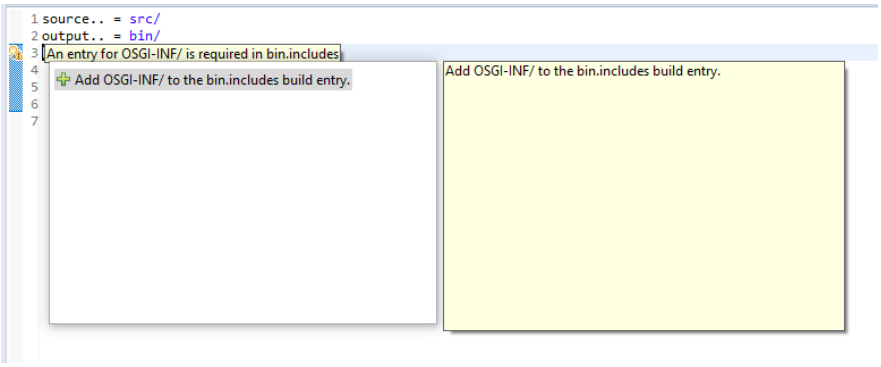
Aquí podemos ver que cadenas se van a externalizar, Pulsamos sobre Finish y nos creará el archivo con la externalización dentro de OSGI-INF -> l10n -> bundle.properties tendremos las externalizaciones, en este archivo añadimos la nueva quedando así.



90-Archivo externalización

NOTA: A veces es necesario cerrar el plugin.xml y volverlo a abrir para que obtenga el nombre desde el fichero.

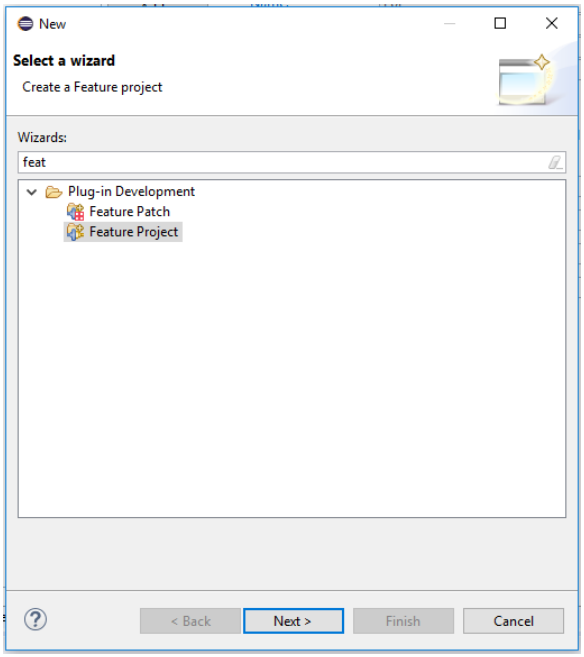
Por último, no vamos al fichero build.properties, donde añadimos la carpeta con las externalizaciones.



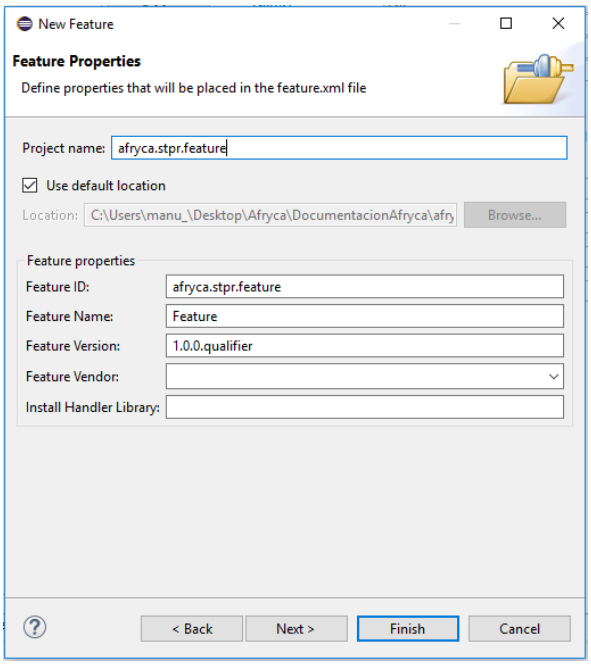
91-Build.propierties externalización

Ahora debemos crear el plug-in feature asociado al plug-in que acabamos de crear.

Para esto creamos un nuevo **Feature Project,** y le damos el mismo nombre pero acabado en feature, en este caso afryca.stpr.feature

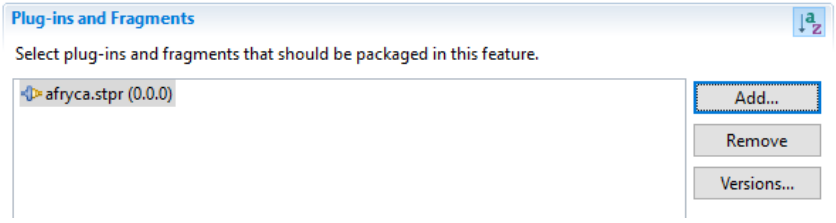


92-Nuevo Feature Project



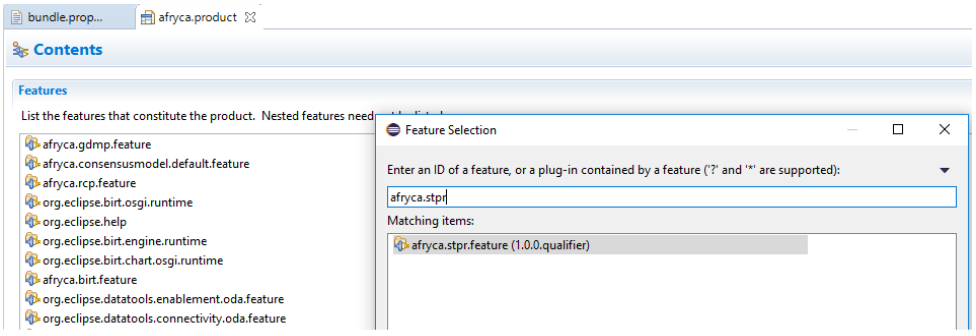
93-Nombre del feature

Accedemos a la pestaña de Included Plug-ins y añadimos nuestro plug-in afryca.stpr.



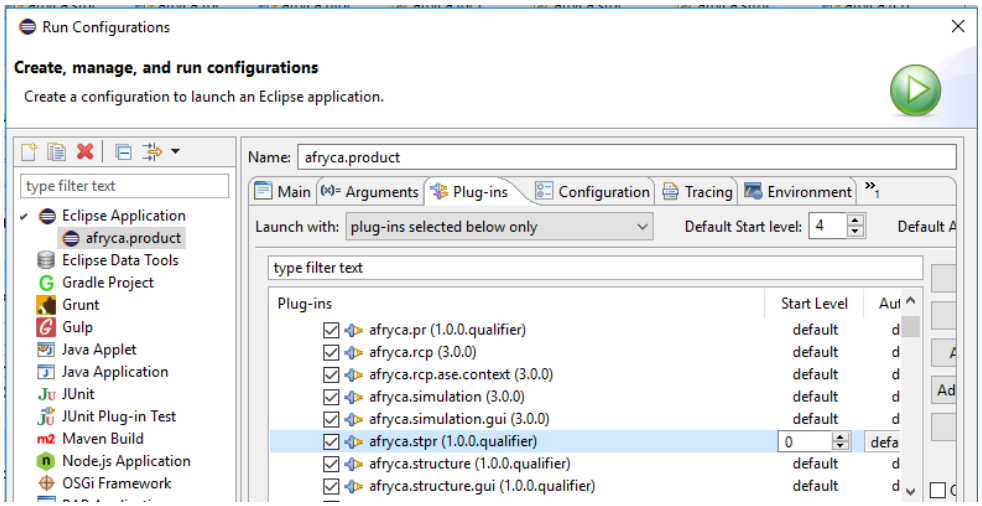
94-Selección del plug-in

Lo siguiente será añadir nuestro plug-in para que lo reconozca nuestra aplicación, ya que hasta ahora no sabe nada de él. Para esto añadimos el plug-in feature a la pestaña Contents de nuestro afryca.product y guardar todos los cambios.



95-Afryca.product plug-in

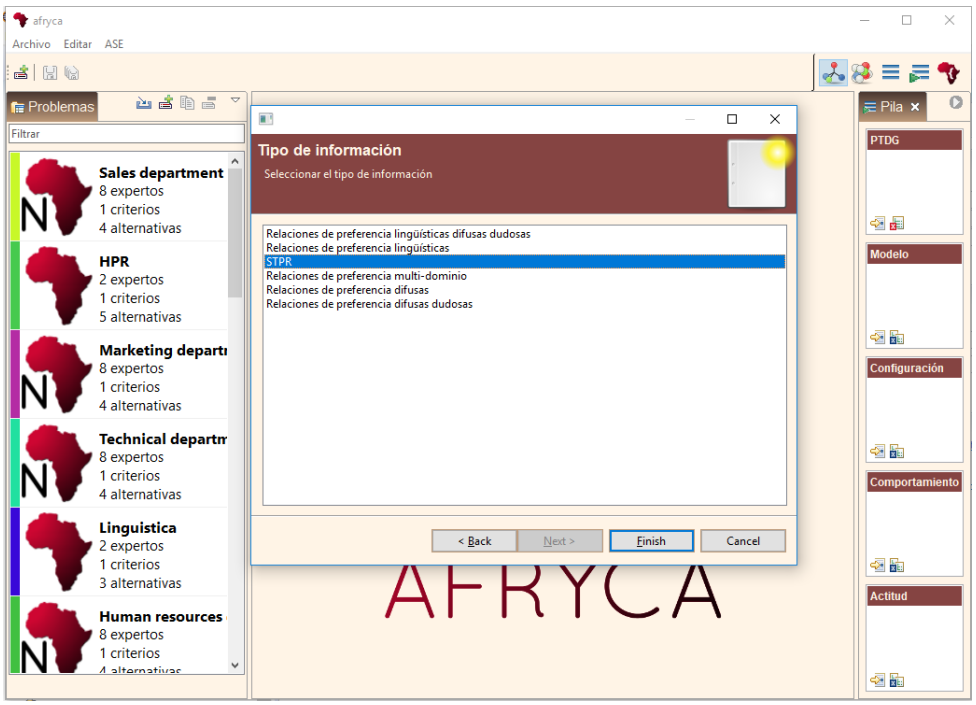
Por último, marcamos como seleccionado nuestro plug-in en **Run-> Run Configurations -> Plug-ins**



96-Run Configurations

Pulsamos en apply -> Run.

Si ejecutamos afryca podemos ver que, si creamos un nuevo problema, en la pantalla de Tipo de información nos aparecerá STPR, aunque todavía no podríamos visualizarla ya que no tenemos su correspondiente gui, que lo veremos más adelante.



97-Selección de estructura

NOTA: Si no aparece puede ser por:

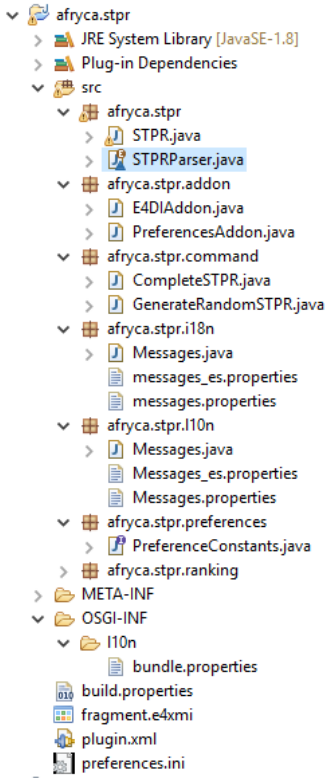
a) No se ha añadido el plug-in al plug-in afryca.stpr.features
b) No se ha añadido el plug-in afryca.stpr.features al afryca.product
c) No se ha marcado como seleccionado en el Run Configurations

IMPORTANTE: A veces, si marcamos el paquete como seleccionado en el Run Configurations y ejecutamos desde afryca.product puede desmarcarse el paquete.

6.2 Añadir visualización de la estructura creada

Primero vamos a añadir las clases necesarias en nuestra nueva estructura, para que pueda funcionar la visualización, para esto miraremos el plug-in de FPR como referencia.

¿Qué necesitamos ahora?



98-Clases de una estructura

STPRParser.java: Esta clase es la encargada de crear las estructuras desde el archivo en el que se guarda el problema

Addons: Se encargan de la lectura del fichero preferences.ini, y de las funciones que se van a ejecutar en ASE para generar estructuras aleatorias, sus gráficas de índices de consistencia o para completar preferencias incompletas.

Commands: Se encargan de ejecutar las funciones de ASE para completar o generar aleatorios.

I18n: Aquí se externalizarán los mensajes de error.

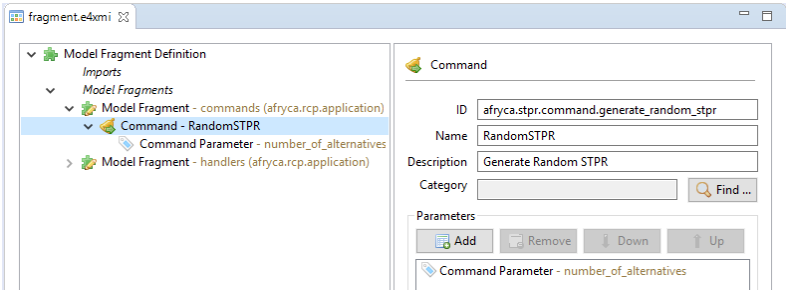
I10n: Externalizaciones de las cadenas de texto.

PreferenceConstants: Constantes con las asignaciones para las llamadas a ASE, esta clase está muy ligada al fichero preferences.ini, ya que de este leerá las cadenas.

Ranking: Aquí se almacenarán las clases que gestionarán el ranking de cada estructura.

Fragment.e4xmi: Aquí crearemos los comandos y handler para generar nuestra estructura aleatoriamente y completarla.

- **IMPORTANTE:** En el caso del comando para generar nuestra estructura aleatoria, como parámetro tenemos el número de alternativas



99-Random STPR configuración

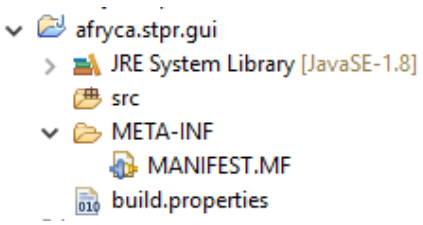
Debemos indicar en el XML del fragment el tipo de dato que es number_of_alternatives, ya que desde la interfaz gráfica no se puede, y quedaría así. (NOTA: usamos el de fpr, pero podemos crear nuestro propio comando en cada structure).

```
elementId="number_of_alternatives" name="number_of_alternatives" typeId="afryca.fpr.command.parameter.integer" optional="false"/>
```

100-Tipo de variable integer

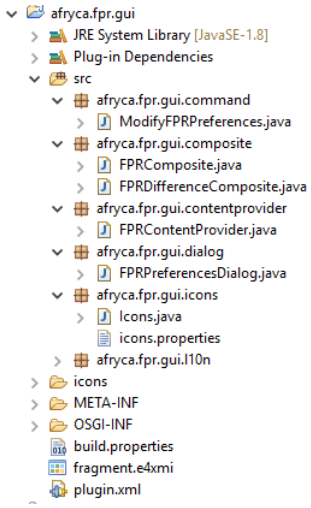
preferences.ini: Se asignará la correspondencia de las constantes creadas en el PreferenceConstants con las funciones o módulos que se llamarán en ASE.

Ahora vamos a crear el plug-in que implementará la visualización de la estructura creada, para esto crearemos un plug-in nuevo, y le daremos como nombre el mismo del plug-in de la estructura acabado en **.gui**, en este caso quedaría como **afryca.stpr.gui.**



101-Plug-in visualización de la estructura

Lo primero haremos será crear las clases necesarias, para esto podemos mirar cómo están las demás estructuras, en este caso vamos a mirar la de FPR



102-Clases visualización FPR

afryca.fpr.gui.command -> ModifyFPRPreferences.java : Esta clase representa a un comando, el cual se ejecuta para modificar las preferencias de las FPR.

afryca.fpr.gui.composite -> FPRComposite.java : Este es el composite encargado de la visualización de la estructura.

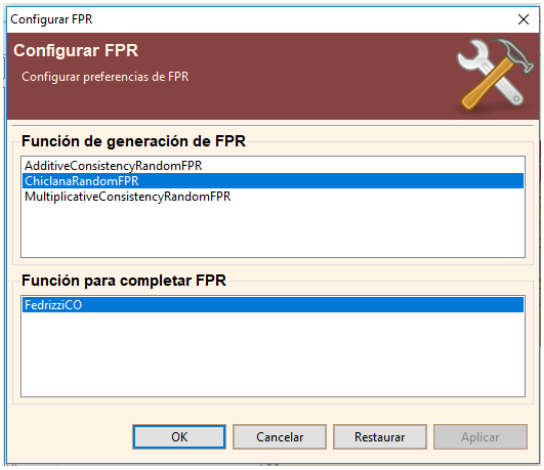
afryca.fpr.gui.composite -> FPRDifferenceComposite.java : Este es el composite encargado de la visualización de la estructura en la perspectiva de simulaciones.

afryca.fpr.gui.contentprovider -> FPRContentProvider.java : Esta clase se encarga del funcionamiento de la visualización, a la hora de editar los datos de una estructura, mostrarlos y modificarlos. Esta clase es la más importante para la visualización por lo que haremos un poco más de hincapié.

Funciones a tener en cuenta en la clase XXXContentProvider.java

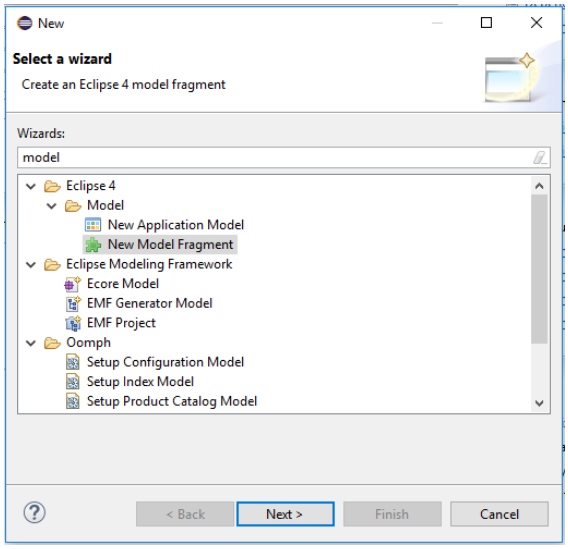
a) **doGetContentAt(int col, int row) :** Se encarga de mostrar los datos de nuestra estructura, aquí debemos obtener el dato que irá en cada casilla(col,row).
b) **doSetContentAt(int col, int row, Object value):** Se encarga de modificar nuestra estructura con los nuevos datos.
c) **doGetCellEditor(int col, int row):** Se encarga del método de modificación de los datos, ya sea mediante la misma tabla, generando un dialog nuevo, etc.
d) **doGetTooltipAt(int col, int row):** Se encarga del Tooltip creado al posicionar el cursor del ratón encima de una casilla.

afryca.fpr.gui.dialog -> FPRPreferencesDialog.java : Este es el Dialog para la modificación de las preferencias de las FPR (imagen abajo), es ejecutado desde el comando **ModifyFPRPreferences.java.**



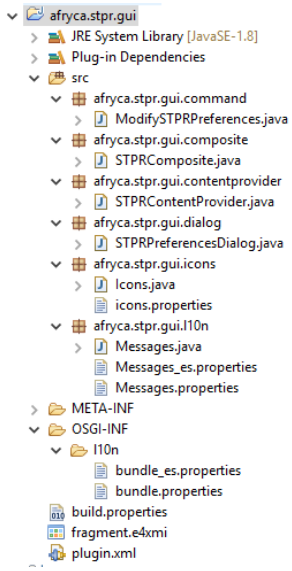
103-Configuración FPR

Añadimos el fragment, donde se configurarán los comandos, handlers y preferencias de la estructura.



104-New Model Fragment

Por lo tanto, debemos crear estas clases para nuestra nueva estructura, quedando así:

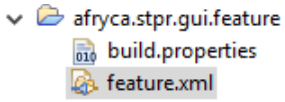


105-Clases Visualización de una estructura

Añadimos el nuevo plug-in al punto de extensión para las structuras.gui, para esto nos iremos a **Extension** de **afryca.stpr.gui**, y configuramos nuestro plug-in.

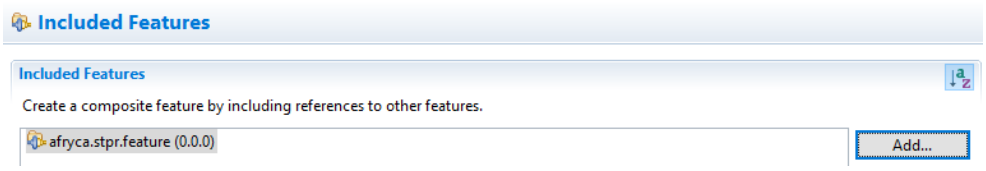
106-Extension estructrura

Una vez creadas y configuradas las clases crearíamos el feature que irá asociado a este plug-in.



107-Feature estructura

Ahora añadimos la feature de STPR en **Included Features** de la nueva feature creada**:**

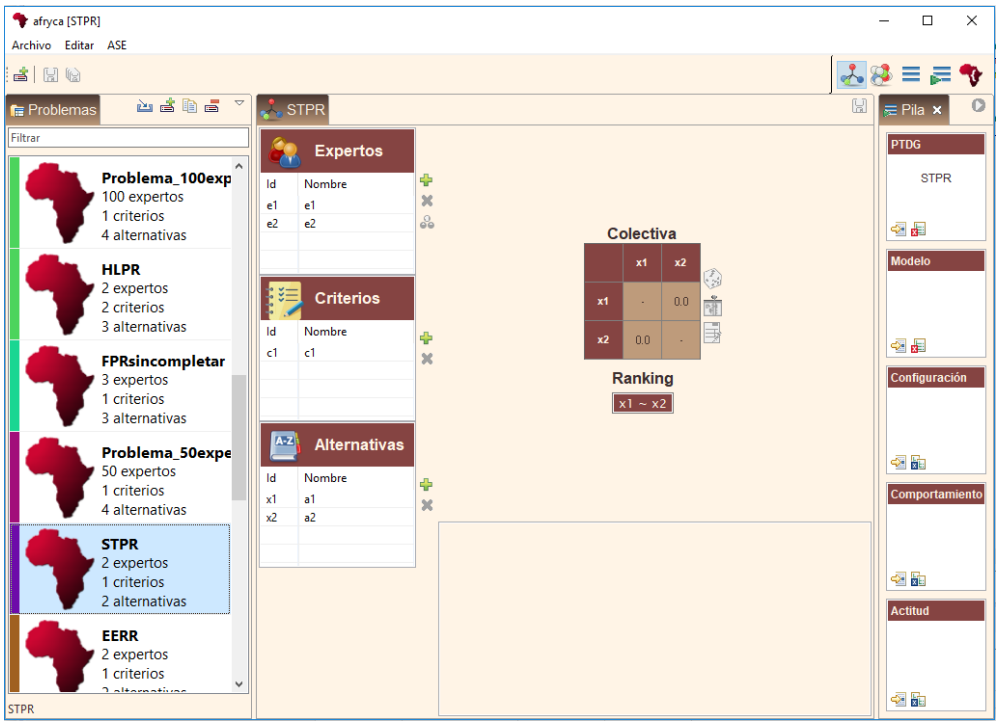


108-Included Features

Además, en la nueva feature añadimos en included Plug-ins nuestro plug-in **afryca.stpr.gui.**

Por último, y como siempre que creemos un plug-in nuevo, debemos añadir el feature al plug-in afryca.product además de al Run configurations y ejecutar.

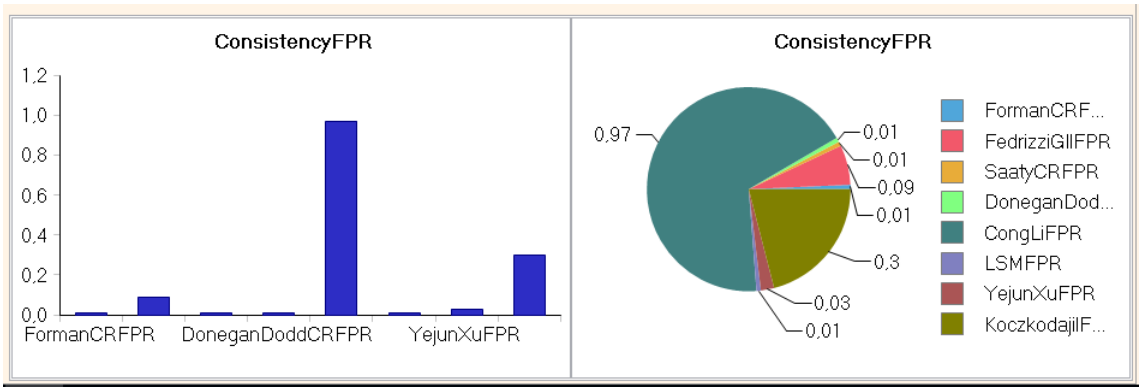
Ya tenemos la visualización de nuestra estructura.



109-Visualización de una estructura

IMPORTANTE: Para que se visualicen las gráficas de índices de consistencia es necesario crear en ASE un módulo y una función que calcule estos datos, en principio llamaremos una función que devuelva 0 para que al menos se visualicen las gráficas.

7. Configurar las gráficas de Afryca



110-Gráficas de consistencia

Las gráficas están presentes en varías perspectivas de AFRYCA y en las cuales se puede configurar cada una de ellas. Estas perspectivas son PTDG y Simulation.

Para configurar las gráficas en el PTDG necesitamos el archivo preferences.ini del plugin afryca.gdmp.gui y el de la estructura como podemos ver en las siguientes imágenes.

```
module.bindings=ConsistencyFPR
module.bindings.fpr=ConsistencyFPR
module.bindings.lpr=ConsistencyLPR
module.bindings.hlpr=ConsistencyHLPR
module.bindings.hpr=ConsistencyHPR
module.bindings.mpr=ConsistencyMPR
module.bindings.decisionmatrix=ConsistencyDM
module.bindings.stpr=ConsistencySTPR
gdmp.chart.functions=GDMP consistency bar chart
```

111-Preferences.ini del plugin afryca.gdmp.gui

```
random.fpr.function=ChiclanaRandomFPR
RandomFPRcomplete.fpr.function=FedrizziCO
aggregation.fpr.function=arithmeticAggregationFPR
gdmp.chart.functions.fpr=GDMP consistency bar chart;GDMP consistency pie chart
```

112-Preferences.ini de la estructura

Mediante el fichero preferences.ini del plugin afryca.gdmp.gui obtenemos el Module Binding que será lanzado para cada una de las estructuras y la visualización por defecto, mientras que con el fichero preferences.ini de la estructura obtenemos la visualización de la estructura, pudiendo ser modificada con el dialogo de configuración de la estructura.

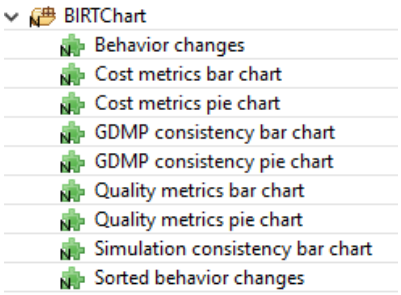
Para la configuración de las gráficas en las simulaciones habría que modificar el fichero preferences.ini del plugin afryca.simulation.gui como podemos ver en la imagen.

```
module.bindings=Simulation consistencyFPR
module.bindings.fpr=Simulation consistencyFPR
module.bindings.lpr=Simulation consistencyLPR
module.bindings.hpr=Simulation consistencyHPR
module.bindings.mpr=Simulation consistencyMPR
module.bindings.hlpr=Simulation consistencyHLPR
module.bindings.decisionmatrix=Simulation consistencyDM
module.bindings.stpr=Simulation consistencySTPR
simulation.chart.functions=Simulation consistency bar chart
simulations.chart.functions=Cost metrics pie chart;Quality metrics pie chart
```

113-Preferences.ini del plugin afryca.simulation.gui

Con esto especificamos el Module Binding que será lanzado para cada una de las estructuras y la visualización.

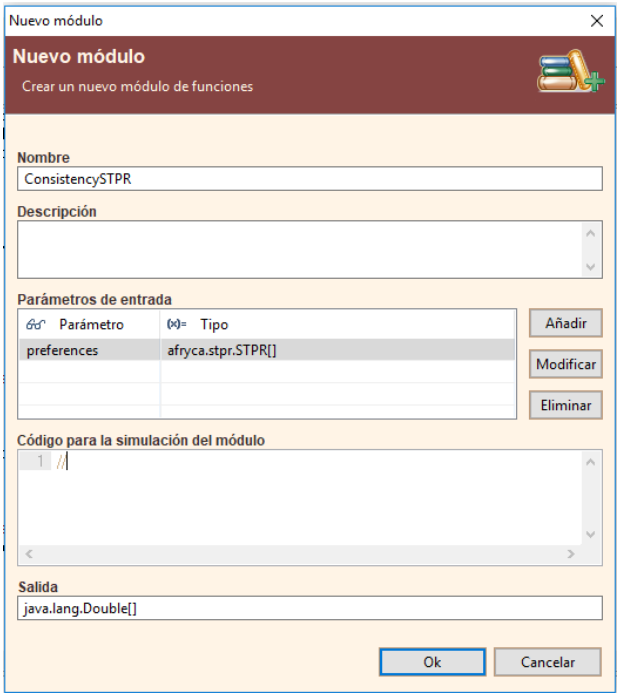
Estas funciones son definidas en ASE, dentro del módulo **BIRTChart**



114-Módulo BIRTChart

Y a su vez estas funciones se encargar de llamar a otro módulo, donde están todas las funciones de cálculo de los índices, pudiendo así tener varios datos en la misma gráfica.

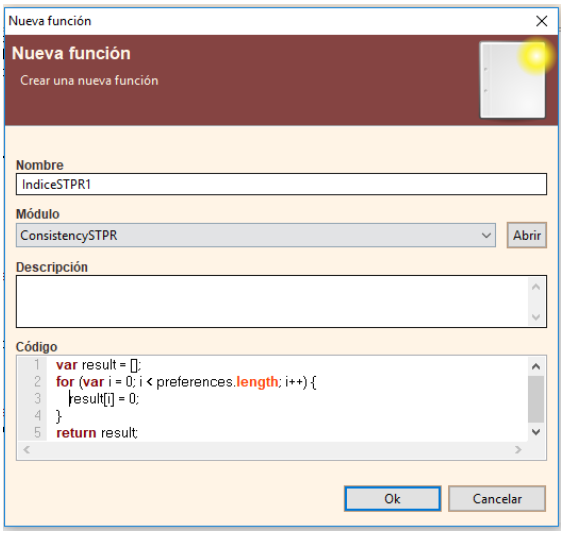
Nosotros ahora vamos a crear las gráficas para nuestra nueva estructura STPR, lo primero que haremos será crear el módulo en el que estarán cada una de las funciones de consistencia, este módulo lo llamaremos ConsistencySTPR



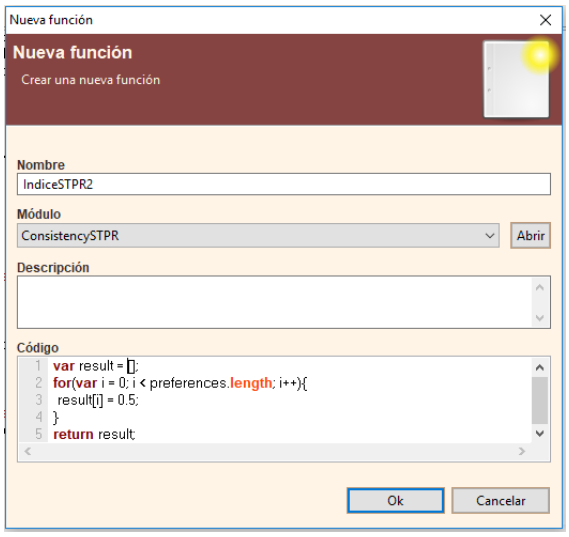
115-Nuevo módulo de consistencia

Cómo parámetro de entrada tendrá un array de nuestra estructura, y como salida devolverá un array de Double, que será los datos de los índices de consistencia.

Una vez creado el módulo, dentro de él crearemos tantas funciones queramos como distintos índices de consistencia, para este ejemplo vamos a crear dos.



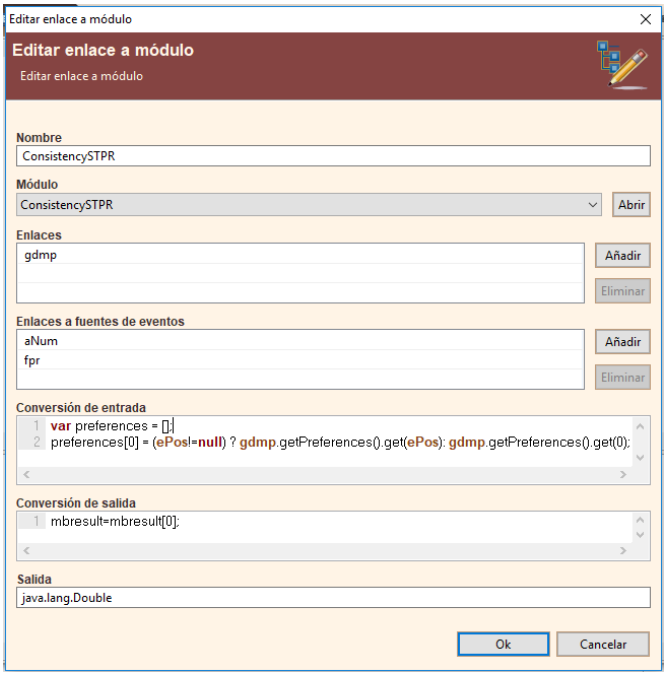
116-Función de ejemplo de consistencia 1



117-función de ejemplo de consistencia 2

Como vemos en el ejemplo en result[i] estamos almacenando un valor, aquí llamaríamos a nuestra función que calcule el indice y se asignaria.

También debemos crear un enlace a este módulo, que lo llamaremos igual que el modulo que acabamos de crear.



118-Enlace a módulo

En este enlace debemos devolver, si hay algun experto seleccionado, la preferencia del experto y si no la preferencia agrupada.

La fúncion de un enlace a módulo, implica que si se produce algún cambio en los Enlaces a fuentes de eventos, se llamará de nuevo al módulo que indica el enlace, ejecutando sus funciones.

Ahora añadimos las funciones al fichero preferences.ini del plug-in afryca.gdmp.gui, quedando así:

```
module.bindings=ConsistencyFPR
module.bindings.fpr=ConsistencyFPR
module.bindings.lpr=ConsistencyLPR
module.bindings.hlpr=ConsistencyHLPR
module.bindings.hpr=ConsistencyHPR
module.bindings.mpr=ConsistencyMPR
module.bindings.decisionmatrix=ConsistencyDM
module.bindings.stpr=ConsistencySTPR
gdmp.chart.functions=GDMP consistency bar chart
```

119-Preferences.ini afryca.gdmp.gui

Además, debemos configurar la clase PreferencesConstants.java de la estructura:

```java
public interface PreferenceConstants {
    public static final String NODEPATH = "afryca.stpr"; //$NON-NLS-1$
    public static final String RANDOM_STPR_FUNCTION_KEY = "random.stpr.function"; //$NON-NLS-1$
    public static final String COMPLETE_STPR_FUNCTION_KEY = "complete.stpr.function"; //$NON-NLS-1$
    public static final String AGGREGATE_STPR_FUNCTION_KEY = "aggregation.stpr.function"; //$NON-NLS-1$
    public static final String CHART_FUNCTION_KEY="gdmp.chart.functions.stpr"; //$NON-NLS-1$
    public static final String MODULE_BINDINGS_KEY = "module.bindings.stpr"; //$NON-NLS-1$
}
```

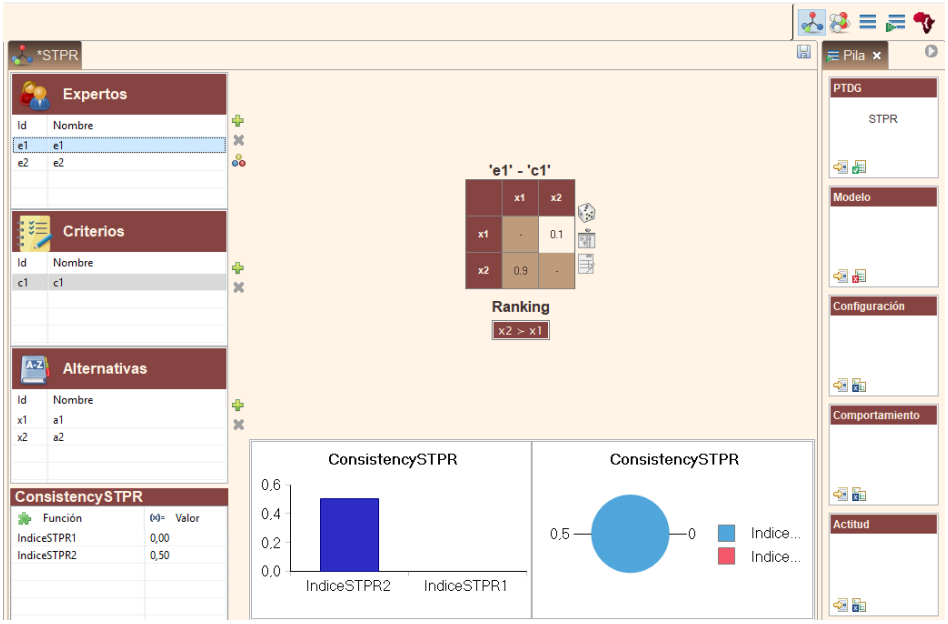
120-PreferenceConstans de la estructura

Para que la función `getPreferencesKeys` () devuelva las cadenas necesarias.

```java
@Override
public String [] getPreferencesKeys() {
    String [] result=new String[2];
    result[0] = PreferenceConstants.CHART_FUNCTION_KEY;
    result[1] = PreferenceConstants.MODULE_BINDINGS_KEY;
    return result;
}
```

121-Función getPreferencesKeys() de la estructura

Para finalizar ejecutamos afryca y comprobamos que aparecen las gráficas.



122-Visualización de las gráficas

8. Añadir un nuevo modelo de consenso

En este capítulo se explicarán los principales métodos a implementar para desarrollar un modelo de consenso, su proceso y se creará un modelo de consenso a modo de ejemplo.

8.1 Métodos a implementar de la clase ConsensusModel

Antes de comenzar, es importante saber el orden de ejecución de los métodos de nuestro modelo de consenso, para esto podemos ver el método **execute()** de la clase ConsensusModel:

```
public Map<EResultElements, Object> execute() {
    if (validConfiguration) {
        obtainConfigurationValues();
        boolean firstRound = true;
        do {
            if (firstRound) {
                preFirstSaveRoundResults();
                firstRound = false;
            } else {
                preSaveRoundResults();
            }
            consensusRound();
            posSaveRoundResults();
        } while (mustBeCarriedOutAnotherRound());
        computeExecutionResults();
    } else {
        return null;
    }

    return getResult();
}
```

123-Función execute de ConsensusModel

Debemos destacar, que la ejecución del método preFirstSaveRoundResult(), solo se ejecuta la primera vez.
Siendo la primera vez **preFirstSAveRoundResult -> consensusRound -> posSaveRoundResults.**
Mientras que posteriormente sería **preSaveRoundResult -> consensusRound -> posSaveRoundResults.**

Los métodos necesarios para nuestro nuevo modelo de consenso son:

A. **setModelConfiguration():** Mediante este método podemos validar cualquier restricción que deba cumplir los datos de entrada, o cualquier otro dato.

```
@Override
protected void setModelConfiguration() {
    if(datoIntroducidoA < datoIntroducidoB){
        validConfiguration=false;
    }

}
```

124-Función setModelConfiguration

Con esto comprobamos que el datoIntroducidoA no sea menor que el datoIntroducidoB, y de ser así, no se habilitará el botón de ejecución del modelo de consenso.

NOTA: Esta función no se suele utilizar, ya que la mayoría de las restricciones se puede poner desde el punto de extensión del modelo de consenso como veremos más adelante.

B. obtainConfigurationValues(): Este método se encargará de obtener los parámetros de entrada del modelo de consenso, que se definen en la pantalla de ejecución de dicho modelo.
Para esto debemos hacer lo siguiente:

a) Crearemos una variable de clase estática de tipo String que almacenará el identificador del parámetro que queremos obtener

```
private static final String POINT_A = "pointA";
```

125-Identificador de una variable

b) Crearemos una variable de clase del tipo de dato que queremos obtener, en este caso es un float.

```
private float pointA;
```

126-Variable

c) Dentro del método **obtainConfigurationValues,** asignaremos a la variable el parámetro.

```
pointA = (Float) configuration.getValue(POINT_A);
```

127-Obtener valor de la variable

d) En el punto de extensión definiremos **el parámetro con el mismo identificador que el dado en el punto a**, además de completar los demás campos como veremos más adelante en el ejemplo.

128-Identificador en el punto de extensión

Veamos varios ejemplos de asignación de valores:

```
cost = (Integer[]) configuration.getValue(COST);

p = (Integer) configuration.getValue(DISTANCE_MEASURE_MINKOWSKI);

weights = (Float[]) configuration.getValue(WEIGHTS);
```

129-Obtener valores de las variables

Además, en esta función podemos inicializar variables como el consensusDegree, currentRound o las recomendaciones para los expertos.

C. **obtainVisualizeValues():** Este método se encarga de calcular los valores para la representación gráfica de las preferencias, generalmente se usará las función que implementa cada estructura, aunque puede ser modificada por el usuario a la hora de implementar el modelo de consenso.

D. **preFirstSaveRoundResult():** Este método se encarga de los valores iniciales antes de hacer nada en el modelo de consenso, aquí normalmente, se calcula el consenso inicial de los expertos, se establece el número de rondas máximas, el nivel de consenso a alcanzar y el nombre del modelo de consenso. Todo esto se almacenará en el mapa **result**, utilizando siempre como clave las definidas en **EResultElements.**

```
result.put(EResultElements.initial_consensus_degree, consensusDegree);
result.put(EResultElements.maxround, maxRounds);
result.put(EResultElements.consensus_threshold, consensusThreshold);
result.put(EResultElements.consensus_model, CONSENSUS_MODEL_NAME);
```

130-Almacenamiento de datos

E. **preSaveRoundRsults():** Esta función se ejecuta antes de cada ronda, excepto antes de la ronda inicial (como hemos visto anteriormente al inicio de este punto), en este método llamaremos a la función preSaveRoundResult() de ConsensusModel, pasándole como parámetros la ronda incrementada en uno, las preferencias y el grado de consenso antes de comenzar la ronda.

```
preSaveRoundResult(currentRound + 1, preferences, (float) consensusDegree);
```

131-preSaveRoundRsult

F. **consensusRound():** Esta es el método más importante y donde se harán los cálculos y modificaciones de las preferencias de los expertos.

El proceso que se realiza en este método, generalmente será el siguiente

1) **Cálculo de la colectiva**
2) **Cálculo del grado de consenso**
3) **Modificación de las preferencias según el modelo**

Cabe destacar del punto **3- Modificación de las preferencias según el modelo**, que para la modificación de las preferencias afryca dispone de un mecanismo de comportamiento para expertos, este comportamiento es definido en la pantalla de los modelos de consenso

132-Comportamiento Estándar

133-Comportamiento Estándar con Adverso

Los comportamientos podrán ser Estándar, que nos dirá si un experto acepta o rechaza el cambio, o puede ser Estándar con adverso, en el que un usuario podrá aceptar, rechazar o contradecir el cambio.

El uso de los comportamientos se trabajará mediante la función **getNChanges(numCambios),** donde nosotros, en nuestro modelo, calcularemos anteriormente qué expertos deben cambiar sus preferencias y los contabilizaremos. El número de expertos a cambiar será el parámetro que reciba el método getNChanges(), y nos devolverá un array donde un 0 significa que el experto no acepta el cambio, u otro valor si el experto acepta el cambio.

Debemos procesar si el cambio para el experto se hará incrementando o disminuyendo su preferencia, por tanto, nos encargaremos de sumar o restar los valores recibidos mediante el getNChanges(), dándose el caso de que si hemos seleccionado Estándar con adverso getNChanges() devolverá valores negativos que al sumarlos o restarlos producirán el efecto contrario y podemos decir que este experto ha contradicho la recomendación.

G. **posSaveRoundResults():** En esta función calcularemos la colectiva con las preferencias modificadas y el grado de consenso, por último, llamaremos a la función:
posSaveRoundResult(preferencias, gradoDeConsenso, advises, colectiva),
Donde advises será un array de 0 y 1, donde 0 significa que un experto no aceptó el cambió su preferencia y 1 si aceptó. Este array es muy importante de cara a los datos finales, ya que nos calculará el número de cambios totales producidos a lo largo de todo el modelo.

```
protected void posSaveRoundResults() {
    computeCollective();
    computeConsensusDegree();
    posSaveRoundResult(preferences, consensusDegree, advises, preferences[numberOfExperts]);

}
```

134-posSaveRoundResults

H. **mustBeCarriedOutAnotherRound():** Este método devolverá un booleano, indicando si el modelo de consenso continua o no, normalmente se comprobará tanto si se han alcanzado el máximo de rondas y si se ha alcanzado el grado de consenso.

```
protected boolean mustBeCarriedOutAnotherRound() {
    return (((float) consensusDegree < consensusThreshold) && (currentRound < maxRounds));
}
```

135-mustBeCarriedOutAnotherRound

I. **saveExecutionResults():** En este método se almacenarán los datos finales obtenidos más relevantes, como son las preferencias finales, el grado de consenso y el número de rondas realizadas.

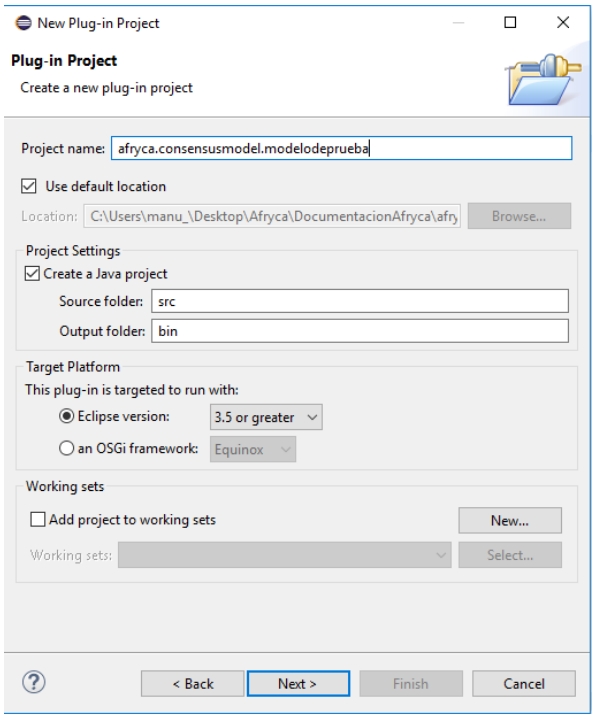
```
this.configuration.setValue(PREFERENCES, preferences);
result.put(EResultElements.number_of_rounds_required, h);
result.put(EResultElements.consensus_degree_achieved, gci);
```

136-saveExecutionResults

8.2 Implementación de un modelo de consenso

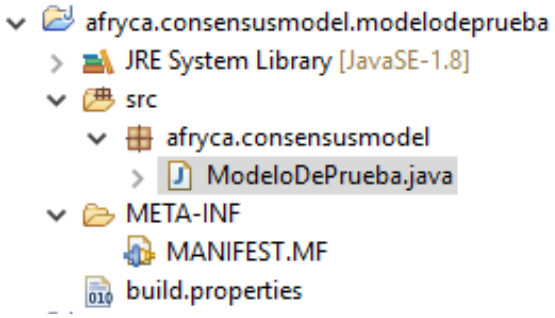
Para añadir un nuevo modelo de consenso lo primero será crearnos un nuevo plug-in, el nombre de los plug-ins de los modelos de consenso tiene el siguiente patrón:
afryca.consensusmodel.nombredelmodelo

Para este ejemplo lo llamaremos afryca.consensusmodel.modelodeprueba



137-Nuevo Plug-in para modelo de consenso

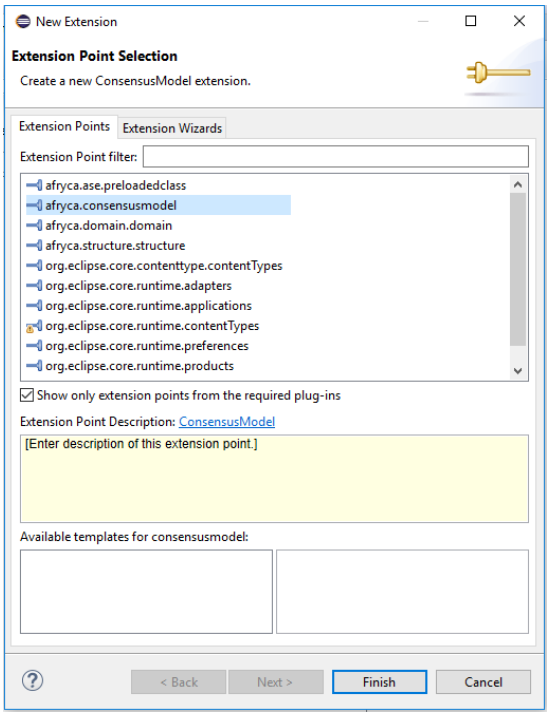
En este plug-in crearemos el paquete afryca.consensusmodel y en el paquete la clase que contendrá el modelo de consenso, quedando así:



138-Clase para el modelo de consenso

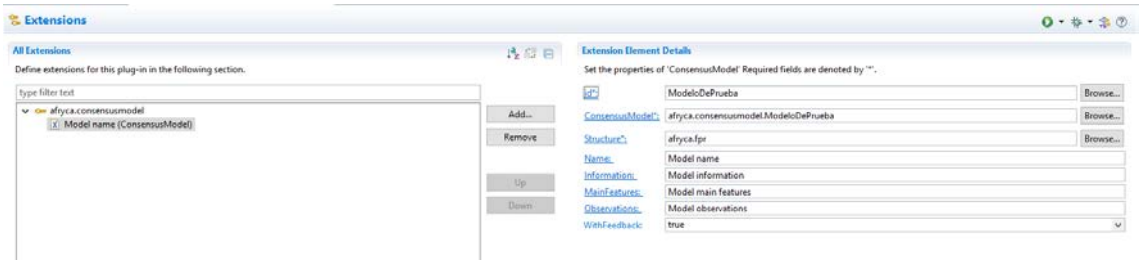
Esta clase tiene que heredar de ConsensusModel, por lo que importaremos el plug-in y añadiremos los métodos no implementados.

Una vez implementada la clase debemos añadir el punto de extensión, para esto nos vamos al MANIFEST.MF -> Extensions -> Add y seleccionamos afryca.consensusmodel



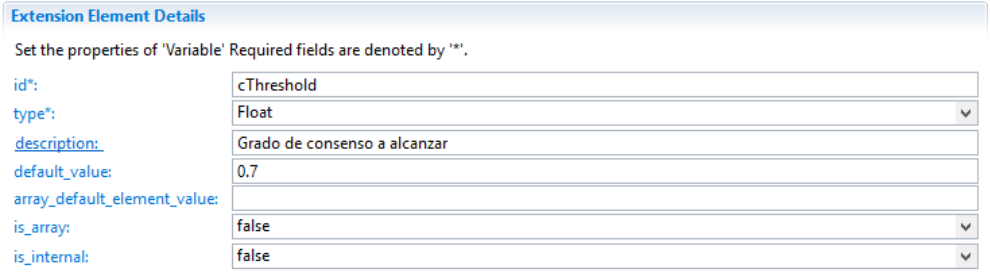
139-Extensión del modelo de consenso

Y completamos la configuración:



140-configuración de la extensión del modelo de consenso

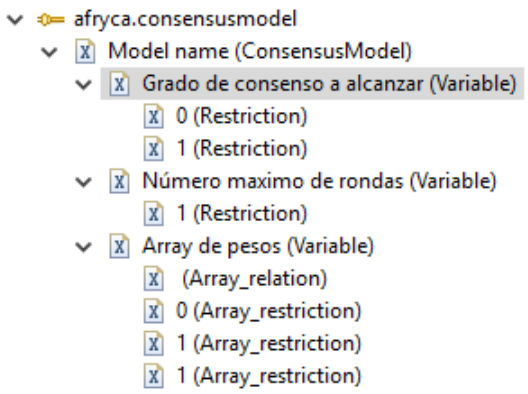
También debemos añadir las variables que hemos definidos en el modelo con los mismo identificadores.



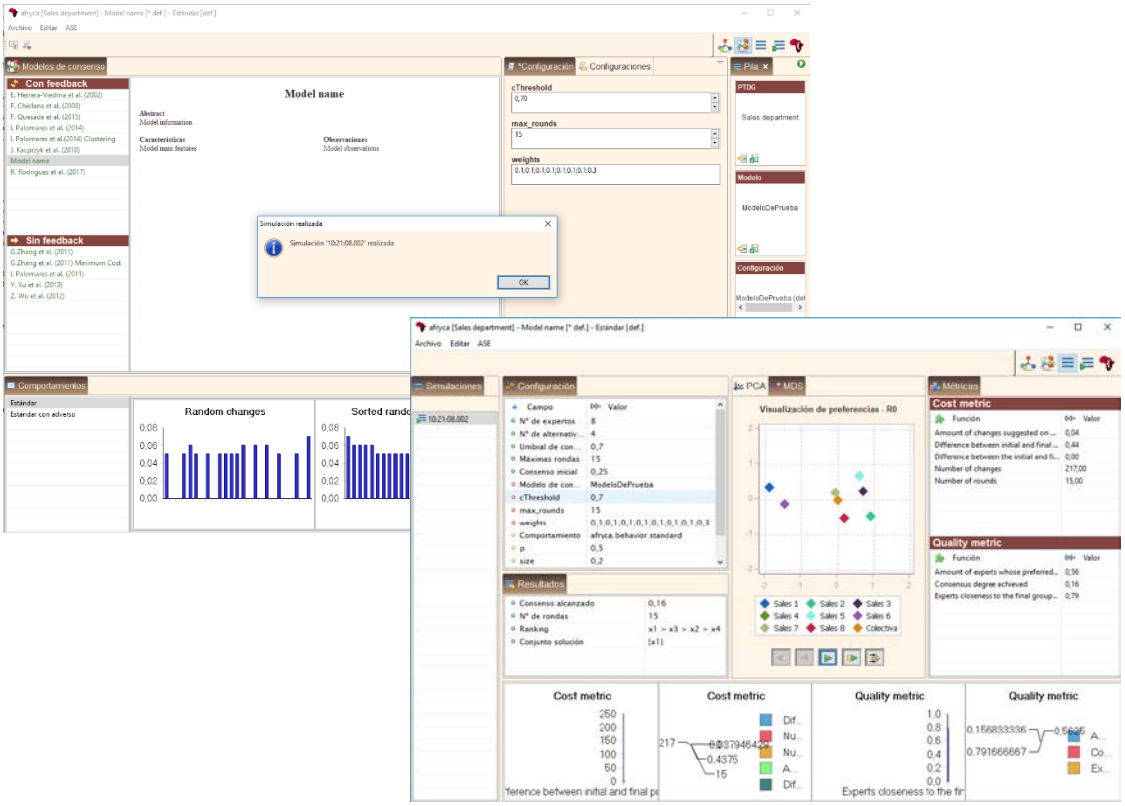
141-Variables del modelo de consenso

Y sus restricciones.

Como paso final, añadir el modelo de consenso al plug-in **afryca.consensusmodel.default.feature -> Included Plug-ins**, marcar el plug-in en Run Configurations y ejecutar.



142-Restricciones de las variables del modelo de consenso



143-Ejecución y visualización de los resultados del modelo de consenso

9. Añadir un nuevo Dominio

Este capítulo veremos cómo añadir un nuevo dominio a afryca.

9.1 Añadir Dominio

El nombre del plug-in tiene el siguiente patrón:

afryca.domain.nombredeldominio

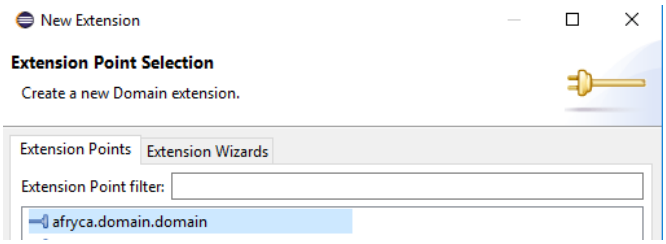
Para este ejemplo crearemos el dominio IntegerEven, en el que solo se aceptaran números enteros dentro de un rango cuyo mínimo y máximo sean pares.
Los dominios creados deben extender de la clase Domain por lo que debemos implementar sus métodos.
Además, debe tener un identificador de la clase:

```
public static final String ID = "afryca.domain.integereven";
```

Para comprobar que sean números pares, usaremos la clase ValueForcer de afryca:

```
public void setMinMax(int min, int max) {
    ValueForcer.notGreaterThan(min, max);
    ValueForcer.isEven(min);
    ValueForcer.isEven(max);
    this.min = min;
    this.max = max;
}
```

Por último, lo añadimos al punto de extensión afryca.domain.domain.



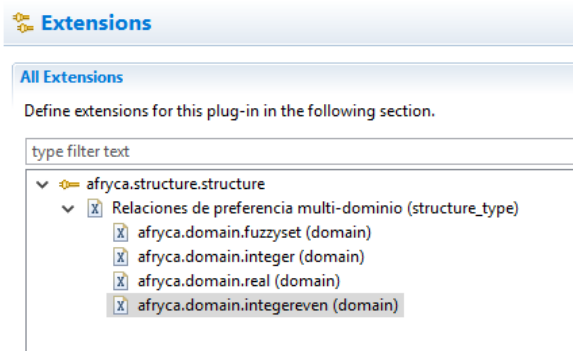
144-Extensión afryca.domain.domain

Y configuramos la extensión.

145-Configuración de la extensión

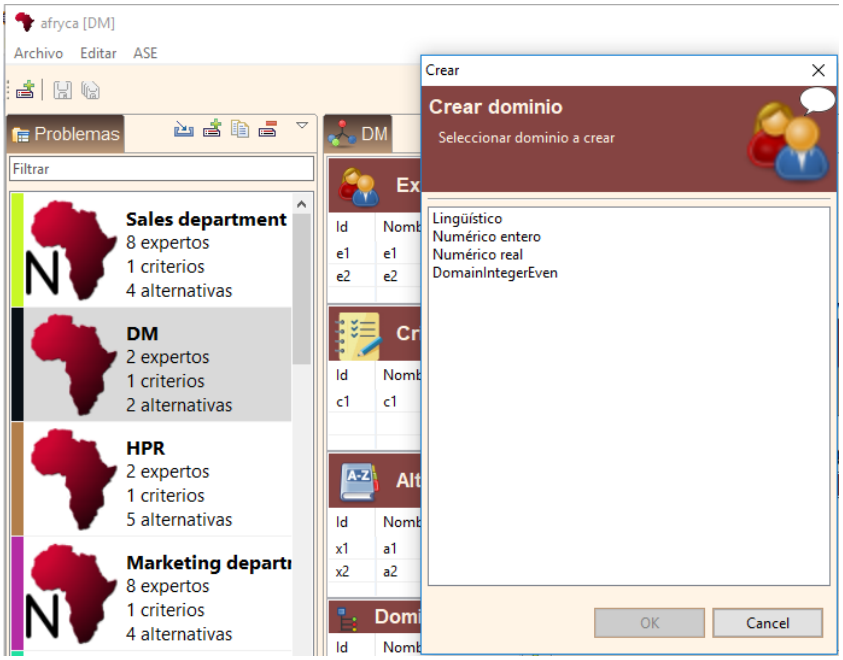
Ahora crearemos su feature, añadimos el plug-in a esta feature y la feature a afryca.product y Run Configurations como vimos anteriormente.

Para comprobar que ha funcionado añadimos nuestro dominio a una estructura, en este caso en DecisionMatrix para ver que nos aparece.



146- Añadiendo un dominio a una estructura

Si ejecutamos AFRYCA y comprobamos los dominios de una estructura nos aparece el creado.

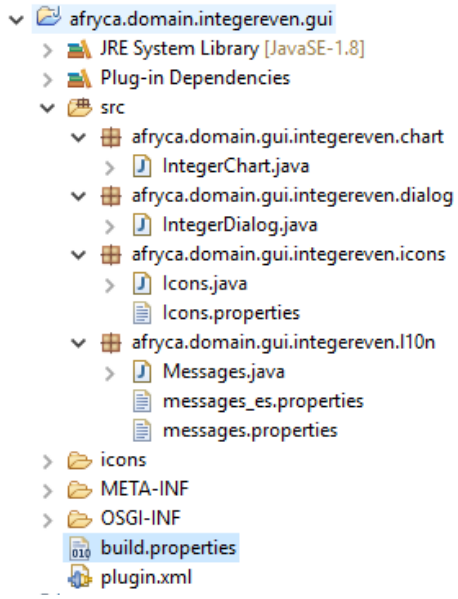


147-Dominios asignados a una estructura

9.2 Añadir interfaz al dominio

Ahora vamos a crear la interfaz con la que configuraremos los datos de nuestro dominio, para esto nos crearemos el plug-in
afryca.domain.integereven.gui

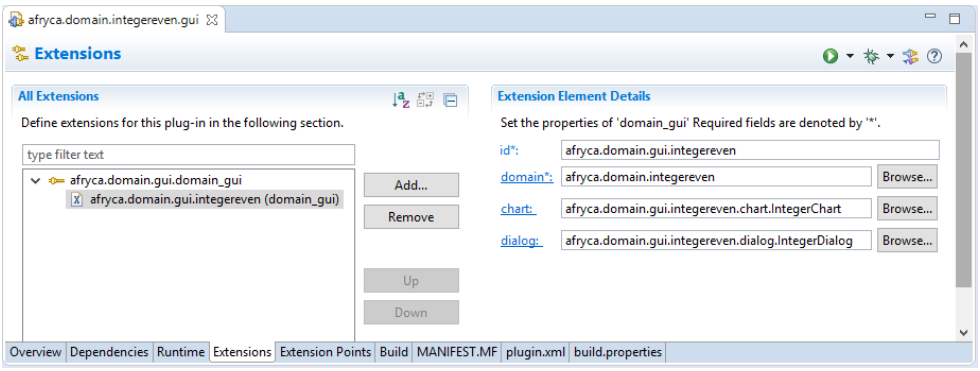
Para este ejemplo la interfaz será muy parecida a la de los dominios Integer o Real que ya existen en afryca.
Vamos a comentar que necesita nuestro plug-in para funcionar:



148-Clases necesarias para un dominio

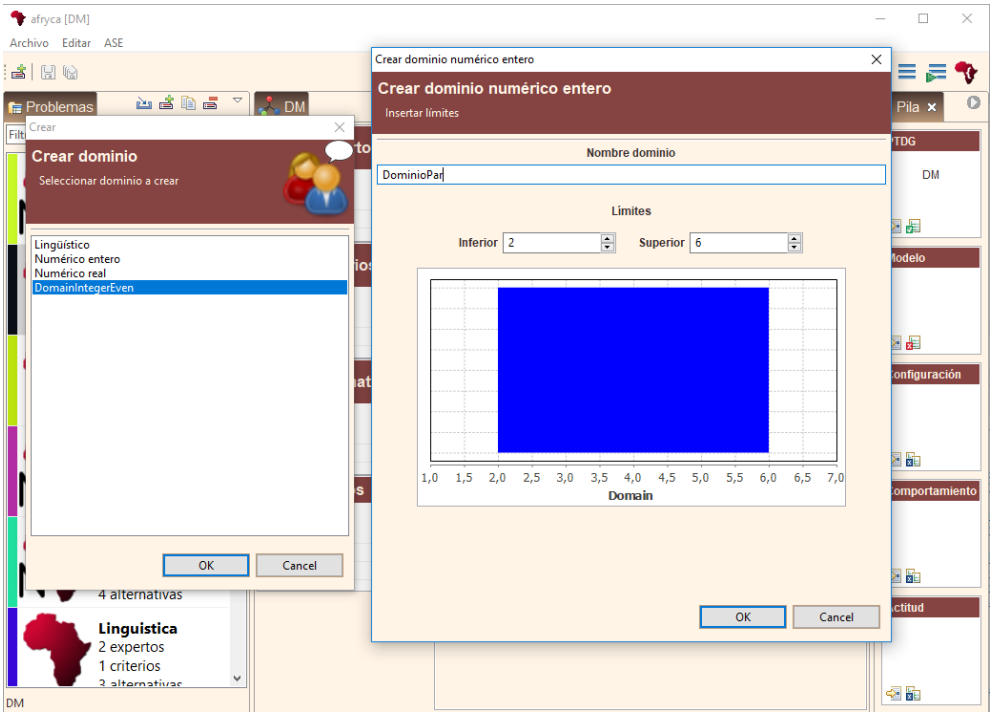
- **IntegerChart**: Deberá heredar de DomainChart, esta clase se encargará de la representación gráfica del dominio.
- **IntegerDialog**: Deberá heredar de DomainDialog, esta clase muestra el dialogo para la ceración del dominio y en que se incluirá el IntegerChart anterior. En esta clase se establecerá el identificador y se encargará de controlar los números pares.
- **Icons**: Iconos del dialogo
- **Messages**: Externalización.

Debemos configurar el punto de extensión, que en este caso será afryca.domain.gui.domain_gui, dónde debemos seleccionar el id que se ha establecido en la clase Dialog, el dominio que utiliza, el chart creado y el dialog creado.



149-Extension afryca.domain.gui.domain_gui

Como último paso, crearemos su feature correspondiente y lo incluiremos en **afryca.product y Run Configurations**.



150-Nuevo dominio creado

10. ASE

10.1 Introducción

ASE (acrónimo de **A**FRYCA **S**cripting **E**nvironment) es un entorno de scripting construido sobre la especificación **JSR 223: Scripting for the JavaTM Platform**.

ASE permite incorporar cualquier nueva funcionalidad mediante el uso de fragmentos de código en tiempo de ejecución, lo cual deja total libertad al usuario para analizar los PACs. El lenguaje nativo del entorno es JavaScript en su implementación **Nashorn**, y en él, ASE añade entornos para programar en **Groovy, Lua, Ruby, Python y Scala.**

Adicionalmente, si el sistema donde se ejecuta AFRYCA cuenta con una instalación del entorno estadístico **R**, ASE facilita la integración con él para utilizarlo como un entorno de programación más.

Por motivos prácticos, ASE emplea como lenguaje nativo el lenguaje de programación **JavaScript** en su implementación **Nashorn**, al venir incluida por defecto en las últimas versiones de la plataforma Java. Además, haciendo uso de la adaptación OSGi para la inclusión de engines realizada, se han desarrollado plug-ins para dar soporte a engines que permiten ejecutar código fuente en lenguaje **Groovy**, **Lua**, **Ruby**, **Python** o **Scala**.

Los engines pueden ser inyectados en cualquier punto de la aplicación para ejecutar código en cualquiera de estos lenguajes, y a su vez, han sido inyectados en ASE, por lo que desde el propio entorno de ASE se puede interactuar con ellos. Adicionalmente, si el sistema donde se ejecuta AFRYCA cuenta con una instalación del entorno estadístico **R**, ASE facilita la integración con él para utilizarlo como un entorno de programación más. La guía de instalación incluye una sección dedicada a la integración de R en AFRYCA.

Pero lo más importante para el investigador que desee usar AFRYCA, no es que pueda ejecutar sus propios scripts en ASE, si no que ASE le facilita un conjunto de herramientas para la gestión de los diferentes scripts desarrollados y que le permitirán realizar tareas sumamente complejas de forma extremadamente sencilla una vez se haya familiarizado con los conceptos y el funcionamiento de ASE. Debe tenerse en cuenta que:

- **ASE está desarrollado dentro de un framework E4, lo cual hace que se ejecute dentro de un contenedor de inyección de dependencias.** En esencia, esto significa que ASE puede ser utilizado desde cualquier punto de la aplicación y que, a su vez, cualquier elemento de la aplicación puede ser inyectado en ASE.
- **ASE puede modificar el código existente de la aplicación**. Es posible tomar cualquier clase y modificar la misma incluyendo nuevas funciones o modificando las existentes. Adaptar un modelo de consenso para que una función se ejecute de un modo diferente es una tarea muy sencilla de realizar usando ASE.
- **ASE permite la compilación de los scripts**. Cualquier funcionalidad desarrollada usando ASE puede ser compilada para que su ejecución sea tan eficiente como el código nativo.
- **ASE cuenta con soporte para el almacenamiento, gestión e invocación de los scripts**. Es posible almacenar desde fragmentos de código con cualquier contenido a funciones que esperen unas entradas y salidas determinadas, funciones que pueden ser enlazadas al framework para ser invocadas automáticamente ante los cambios en la aplicación.

- **ASE está soportado por el workspace de AFRYCA y sus elementos pueden precargase**. Los elementos precargados se incluyen directamente en los plug-ins de AFRYCA haciendo uso del punto de extensión ofrecido por el workspace. Precargar un script desarrollado en ASE es como incrementar la propia API de AFRYCA. Gran parte de la funcionalidad existente hasta el momento en las versiones de AFRYCA previas a ASE ha sido portada a scripts ASE y precargada en la aplicación.

En esta sección se expone la funcionalidad ofrecida por ASE, para lo cual se aclararán en primer lugar los conceptos usados en ASE así como sus principales elementos precargados. Tras ello se llevará a cabo una exploración de la perspectiva ASE, así como de las vistas y editores contenidas en la misma.

Conceptos

En ASE se usan conceptos que pueden resultar ambiguos para un nuevo usuario de la aplicación. Para evitar confusiones, en esta sección se establece a qué se está haciendo referencia cuando se emplea cada término en esta sección:

- **Script**: Guion para la ejecución de un programa.
- **Evaluación**: Resultado de la ejecución de algún script.
- **Script categorizable**: Tipo de script que admite su categorización
- **Categoría**: Agrupación de scripts categorizables.
- **Snippet**: Script categorizable con formato libre.
- **Repositorio**: Categoría para snippets.
- **Función**: Script categorizable que recibe parámetros y devuelve un resultado.
- **Módulo**: Categoría para funciones que define el nombre y tipo de los parámetros y el tipo de salida.
- **Parámetro de ejecución**: Argumento pasado por valor a una función.
- **Variable del entorno**: Variable creada en el entorno en alguna evaluación.
- **Variable de ejecución**: Argumento pasado por referencia a una función.
- **Enlace**: Elemento del framework vinculado a ASE y del que ASE es notificado ante sus cambios.
- **Enlace a módulo**: Envoltura de un módulo que es invocada cuando los enlaces fijados cumplen unas condiciones.
- **Clase precargada**: Clase que es importada de forma automática en cada evaluación.
- **Biblioteca**: Tipo especial de snippet que es compilado al inicio de la aplicación.
- **Simulación**: Evaluación usando un código de entrada predefinido.
- **Fragmento**: Script ejecutado en aislamiento que admite variables de ejecución.

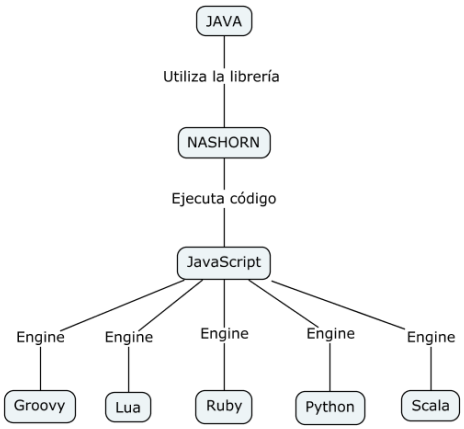
10.2 ASE-API

Al utilizar ASE no existe un único modo correcto de hacer las cosas, si no que el usuario dispone de un amplio abanico de opciones para realizar las mismas tareas. Para un correcto desempeño empleando ASE, es aconsejable:

1. Disponer de unos conocimientos mínimos sobre Eclipse RCP y E4, pilares sobre los que se construye AFRYCA. Desde la sección de fundamentos de la guía de desarrollo es posible acceder a diversa documentación sobre las bases de AFRYCA.
2. Disponer de un conocimiento medio de la API de AFRYCA, conociendo al menos cuales son los servicios que ofrece AFRYCA y cuáles son las clases de su modelo. La sección de desarrollo ofrece toda la información necesaria para salvar este punto.
3. Disponer de un conocimiento completo de la API de AFRYCA. Desde AFRYCA se incluye el repositorio precargado '**ASE-API**' el cual contiene diversos snippets de ejemplo de la API de ASE:
    - **Execute function examples**: Ejemplos de cómo ejecutar funciones de forma manual.
    - **Execute module binding example**: Ejemplo de la ejecución de un enlace a módulo de forma manual.
    - **Execute module examples**: Ejemplos de cómo ejecutar módulos de forma manual.
    - **Fragments examples**: Ejemplos de ejecución de fragmentos. De especial utilidad para el uso de ASE desde el código fuente de los plug-ins.
    - **Load snippet examples**: Ejemplo de la carga de snippet. De gran utilidad para la definición de funciones.
    - **Simulate function example**: Ejemplo para la simulación de funciones.
    - **Simulate module example**: Ejemplo para la simulación de módulos.
    - **Scripting examples**: Ejemplos de uso de los diferentes engines.
    - **Eval snippet examples**: Ejemplos de evaluación de snippets en diferentes lenguajes de programación usando los engines.

10.3 Tecnologías

Como se ha comentado anteriormente, ASE emplea como lenguaje nativo el lenguaje de programación **JavaScript** en su implementación **Nashorn**, además, se han desarrollado plug-ins para dar soporte a diferentes tecnologías que permiten ejecutar código fuente en lenguajes **Groovy**, **Lua**, **Ruby**, **Python** o **Scala**.



151-Tecnologías de ASE

Mediante ASE, tenemos tres formas distintas de ejecutar código para cada lenguaje, **modo 1** es ligeramente más rápido que el modo 2, sólo puede ser usado desde ASE y además, si el engine no esté cargado falla, **el modo 2** es más rápido que el modo 3, puede ser usando tanto en ASE como en Afryca y en caso de que no esté cargado el engine también fallaría, por último, el **modo 3** puede ser usado tanto en ASE como en Afryca y devuelve null si el engine no está cargado.

- **Groovy:** Lenguaje de programación orientado a objetos implementado sobre la plataforma Java. La especificación JSR 241 se encarga de su estandarización para una futura inclusión como componente oficial de la plataforma Java.
  Groovy usa una sintaxis muy parecida a Java, comparte el mismo modelo de objetos, de hilos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java.

  Ejemplo de uso en ASE:

```
var groovy = 'def product(Double x, Double y) {\n';
groovy += '    return x * y\n';
groovy += '}';
var notLoaded = 'is not loaded';
```

  a) Modo 1

```
var groovyResult = (typeof Groovy !== 'undefined') ? Groovy.eval(groovy) : notLoaded;
```

  b) Modo 2

```
var groovyResult = (typeof Groovy !== 'undefined') ? ase.evalCode(Groovy, code) :
notLoaded;
```

  c) Modo 3

```
var groovyResult = ase.evalCode('Groovy', code);
```

- **Lua:** Lenguaje de programación suficientemente compacto para usarse en diferentes plataformas. En Lua las variables no tienen tipo, sólo los datos y pueden ser lógicos, enteros, números de coma flotante o cadenas. Estructuras de datos como vectores, conjuntos, tablas hash, listas y registros pueden ser representadas utilizando la única estructura de datos de Lua: la tabla.

Ejemplo de uso en ASE:

```
var lua    = 'function product(x, y)\n';
lua += '    return x * y\n';
lua += 'end';
var notLoaded = 'is not loaded';
```

a) Modo 1

```
if (typeof LuaJ !== 'undefined') LuaJ.eval('result = ' + code);
// LuaJ does not return eval result
var luaj = (typeof LuaJ     !== 'undefined') ? LuaJ.get('result') : notLoaded;
```

b) Modo 2

```
luaj = (typeof LuaJ     !== 'undefined') ? ase.evalCode(LuaJ, 'result = ' + code + ' -- ase get result') : notLoaded; // ASE shortcut
```

c) Modo 3

```
Luaj = ase.evalCode('LuaJ', 'result = ' + code + ' -- ase get result'); // ASE shortcut
```

- **Ruby**: Lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995. Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU. Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre.

Ejemplo de uso en ASE:

```
var ruby    = 'def product(x, y)\n';
ruby += '    return x * y\n';
ruby += 'end';
var notLoaded = 'is not loaded';
```

a) Modo 1

```
var jruby = (typeof JRuby   !== 'undefined') ? JRuby.eval(code) : notLoaded;
```

b) Modo 2

```
jruby = (typeof JRuby!=='undefined') ? ase.evalCode(JRuby, code): notLoaded;
```

c) Modo 3

```
jruby = ase.evalCode('JRuby', code);
```

- **Python**: lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.
  Ejemplo de uso en ASE:

```
var python    = 'def product(x , y):\n';
python += '    return x * y';
var notLoaded = 'is not loaded';
```

  a) Modo 1

```
var jython = (typeof Jython  !== 'undefined') ? Jython.eval(code) : notLoaded;
```

  b) Modo 2

```
var jythonv= (typeof Jython  !== 'undefined') ? ase.evalCode(Jython, code) : notLoaded;
```

  c) Modo 3

```
jython = ase.evalCode('Jython', code);
```

- **Scala:** Lenguaje de programación multi-paradigma diseñado para expresar patrones comunes de programación en forma concisa, elegante y con tipos seguros. Integra sutilmente características de lenguajes funcionales y orientados a objetos. La implementación actual corre en la máquina virtual de Java y es compatible con las aplicaciones Java existentes.
  Ejemplo de uso en ASE:

```
var scala    = 'def product(x:Double, y:Double) : Double = {\n';
scala += '    return x * y\n';
scala += '}';
var notLoaded = 'is not loaded';
```

  a) Modo 1

```
var scalaResult    = (typeof Scala   !== 'undefined') ? Scala.eval(code) : notLoaded;
```

  b) Modo 2

```
scalaResult      = (typeof  Scala     !== 'undefined')  ?  ase.evalCode(Scala,  code)  :
notLoaded;
```

  c) Modo 3

```
scalaResult    = ase.evalCode('Scala', code);
```

10.4 Bibliotecas

Como se ha introducido en la sección de conceptos, en ASE el término biblioteca hace referencia a un tipo especial de snippet. Un snippet será una biblioteca siempre y cuando se haya definido en el repositorio precargado '**Library**'. Una biblioteca será compilada al iniciar la aplicación, por lo que su desempeño será equivalente al del resto del código nativo de AFRYCA.

Aunque no es estrictamente necesario, es muy aconsejable que las bibliotecas se añadan al ámbito global de la aplicación al realizar su definición, y que en su última línea se devuelvan ellas mismas para poder ser cargadas mediante la función '**loadSnippet**' del servicio de ASE. Para ello, basta con adaptar la siguiente plantilla, reemplazando '**<nombre>**' por el nombre de la biblioteca y '**<código>**' por su código fuente:

```
var <nombre> = (typeof exports === "undefined")?(function <nombre>() {}):(exports);

if(typeof global !== "undefined") { global.<nombre> = <nombre>; }

<código>

<nombre>
```

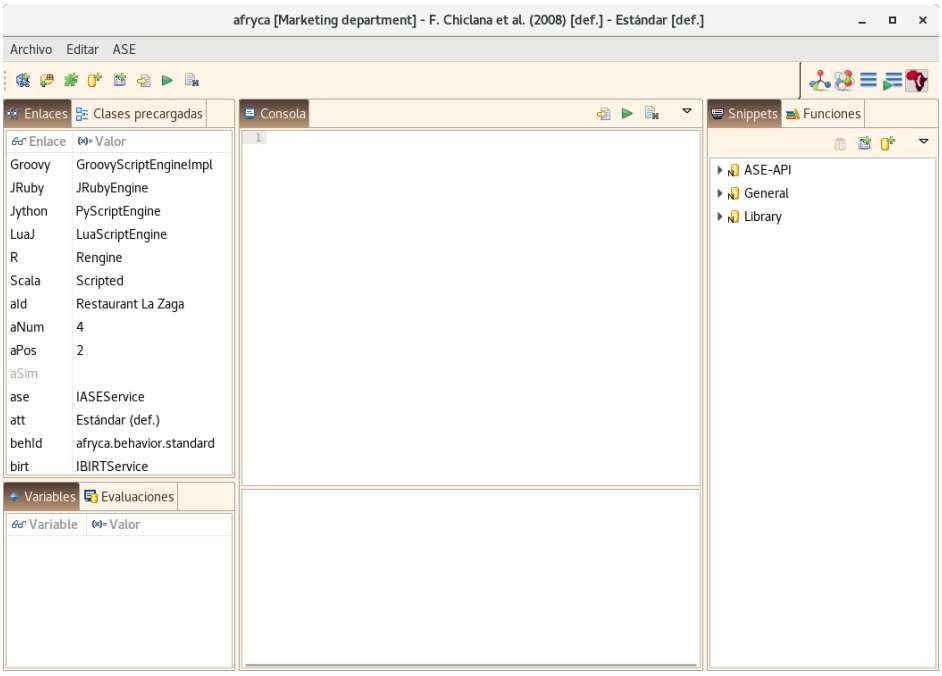
La versión actual de AFRYCA incluye varias bibliotecas precargadas, varias de las cuales reemplazan funcionalidad anteriormente proporcionada desde plug-ins. Es posible revisar el código y funciones de las diferentes bibliotecas desde la propia aplicación:

- **BIRTChartManager**: Biblioteca de asistencia para la gestión de gráficos BIRT. Simplifica el acceso a la interfaz de AFRYCA y la actualización de forma asíncrona de las gráficas.
- **HLPR**: Biblioteca experimental de soporte a relaciones de preferencia lingüísticas hesitant.
- **LPR**: Biblioteca experimental de soporte a relaciones de preferencia lingüísticas.
- **adapters**: Biblioteca con funciones de conversión de diverso tipo.
- **behavior**: Biblioteca usada por los comportamientos para el cálculo de distribuciones de probabilidad. Emplea internamente las clases precargadas de **org.apache.commons.math**.
- **behavior_native**: Biblioteca usada por los comportamientos para el cálculo de distribuciones de probabilidad. Emplea las funciones de la biblioteca **mctad**.
- **consensus**: Biblioteca con funciones de utilidad en los procesos de alcance de consenso.
- **consistency**: Biblioteca con funciones para el análisis de la consistencia en relaciones de preferencia.
- **distance**: Biblioteca con funciones para el cálculo de la distancia entre relaciones de preferencia.
- **linear_algebra**: Biblioteca con funciones de álgebra lineal.

- **mctad**: Biblioteca de terceros con funciones estadísticas. Más documentación en su sitio web: http://erictheise.com/mctad.js/
- **mds**: Biblioteca para el escalado multidimensional de las preferencias.
- **metric**: Biblioteca experimental de métricas.
- **numeric**: Biblioteca de terceros para el análisis numérico. Más documentación en su sitio web: http://numericjs.com/

Perspectiva de ASE

La perspectiva de ASE es la última perspectiva accesible desde el selector de perspectivas y permite gestionar todos los elementos relacionados con ASE e interactuar de forma directa con él usando la consola. Su apariencia inicial será similar a la de la siguiente captura.



152-Perspectiva de ASE

La perspectiva está formada por siete partes, seis vistas más un editor. La utilidad de cada una de las vistas en la gestión de ASE es la siguiente:

1. **Consola de ASE**: Editor que permite evaluar cualquier código introducido en ASE y visualizar el resultado de la ejecución. Realiza el resaltado de sintaxis del código.
2. **Vista de enlaces**: Muestra los enlaces de AFRYCA en ASE y facilita su inclusión en la consola.
3. **Vista de clases precargadas**: Muestra las clases precargadas en ASE y sus diferentes métodos, facilitando su inclusión en la consola y la autogeneración de código.
4. **Vista de variables**: Muestra las variables que se han originado al evaluar código en la consola y permite incluirlas en la consola o eliminarlas.
5. **Vista de evaluaciones**: Muestra un histórico de las evaluaciones realizadas en la consola y permite visualizar su resultado y volver a incluirlas en la consola.

6. **Vista de snippets**: Muestra los snippets y repositorios existentes y permite realizar su gestión.

7. **Vista de funciones**: Muestra las funciones, módulos y enlaces a módulos existentes y permite realizar su gestión.

10.5 Consola de ASE

La consola de ASE es un editor que permite la introducción de código y su evaluación en el engine de ASE, mostrando los resultados obtenidos.

153-Consola de ASE

De forma vertical, la vista se divide en dos paneles:

- **Entrada**: Editor para la introducción de código a ejecutar en ASE. El editor realiza el resaltado de sintaxis del código introducido, mostrando de un modo diferente:
    - Palabras reservadas del lenguaje JavaScript.
    - Comentarios.
    - Cadenas de caracteres.
    - Variables.
    - Enlaces.
    - Clases precargadas.
- **Salida**: Muestra los resultados obtenidos en cada ejecución.

Guardar como snippet

Si se cuenta con permisos de escritura en el espacio de trabajo, cualquier código introducido en la entrada de la consola puede ser guardado como snippet invocando el comando '**guardar como snippet**'. Al invocar el comando se mostrará el editor de snippets y se empleará como código el valor de entrada existente en la consola.

Evaluar editor

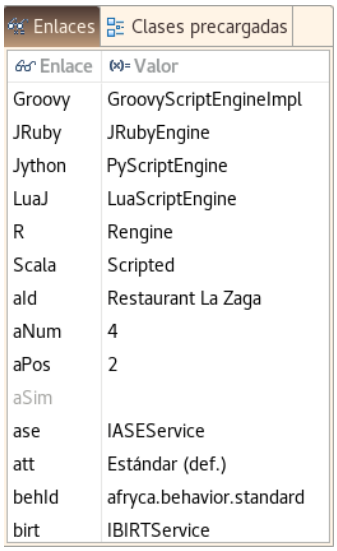
Cualquier código introducido en la entrada de la consola puede ser evaluado invocando el comando '**evaluar editor**'. El comando creará de forma interna un snippet que será configurado con todas las clases precargadas de ASE y que será evaluado en el engine de ASE.

Limpiar editor

Para eliminar el contenido de la entrada de la consola puede invocarse el comando '**limpiar editor**'. El comando no borra el contenido de la salida.

10.6 Vista de enlaces

La vista de enlaces muestra los enlaces establecidos por AFRYCA en ASE.



154-Vista de enlaces

Para cada enlace, la vista muestra su nombre y un valor abreviado del mismo. Dejando el cursor del ratón sobre el valor durante unos segundos la vista mostrará una ventana emergente con una descripción detallada del valor. En caso de que un enlace no tenga valor establecido, este se mostrará en la vista en color gris claro.

Téngase en cuenta que los nombres de los enlaces corresponden a variables creadas y gestionadas por AFRYCA en ASE, por lo que son variables en uso que no deberían ser modificadas por el usuario. En tal caso el comportamiento de la aplicación podría ser imprevisible.

Para evitar utilizar los nombres de variables más habituales como podrían ser *preferences*, *alternative* o *expert*, y por cuestiones de brevedad, AFRYCA emplea

identificadores compactos para los enlaces. Actualmente, los enlaces establecidos por AFRYCA en ASE son los siguientes:

- **Groovy**: Engine Groovy para código groovy.
- **JRuby**: Engine JRuby para código ruby.
- **Jython**: Engine Jython para código python.
- **LuaJ**: Engine LuaJ para código lua.
- **Scala**: Engine Scala para código scala.
- **R**: Engine de soporte para la JRI de R para código r.
- **ase**: Servicio ASE.
- **birt**: Servicio BIRT.
- **ui**: Acceso a la interfaz de Eclipse.
- **sWork**: Servicio Workspace.
- **sGDMP**: Servicio GDMP.
- **gdmp**: GDMP seleccionado.
- **fpr**: FPR del experto seleccionado.
- **eNum**: Número de expertos del GDMP seleccionado.
- **eId**: Identificador del experto seleccionado.
- **ePos**: Posición del experto seleccionado en el GDMP.
- **aNum**: Número de alternativas del GDMP seleccionado.
- **aId**: Identificador de la alternativa seleccionada.
- **aPos**: Posición de la alternativa seleccionada en el GDMP.
- **sCM**: Servicio ConsensusModel.
- **cmId**: Identificador del modelo de consenso seleccionado.
- **conf**: Configuración seleccionada.
- **sBeh**: Servicio Behavior.
- **behId**: Identificador del comportamiento seleccionado.
- **att**: Actitud seleccionada.
- **sSim**: Servicio Simulation.
- **sim**: Simulación seleccionada.
- **simStack**: Pila de simulaciones.
- **irSim**: Ronda inicial seleccionada en la simulación.

- **frSim**: Ronda final seleccionada en la simulación.
- **eSim**: Experto seleccionado en la simulación.
- **aSim**: Alternativa seleccionada en la simulación.

Pegar enlace

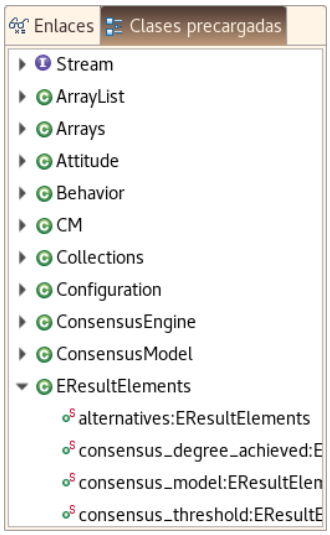
Invocando el comando '**pegar enlace**' el nombre del enlace seleccionado en la vista será pegado en la entrada de la consola. El comando será también invocado haciendo doble clic sobre cualquier enlace.

Reemplazar contenido del editor con enlace

Invocando el comando '**reemplazar contenido del editor con enlace**' el nombre del enlace seleccionado en la vista reemplazará cualquier contenido existente en la entrada de la consola.

Vista de clases precargadas

La vista de clases precargadas muestra el conjunto de clases que son precargadas en los scripts ejecutados en ASE.



155-Vista de clases precargadas

Como se ha especificado con anterioridad y se expone con mayor detalle en la guía de desarrollo, AFRYCA permite establecer diferentes clases a precargar extendiendo un punto de extensión. Las clases precargadas no son más que clases Java que son importadas automáticamente en cada script, lo cual redunda en que:

- Se evita importar de forma más anual las clases más comúnmente empleadas.
- El editor de ASE será capaz de resaltarlas en el código al reconocerlas como parte de la funcionalidad de ASE.
- Serán mostradas en la vista de clases precargadas lo cual facilitará conocer su interfaz.

La vista muestra las clases en modo de árbol, empleando diferentes iconos para representar interfaces, clases, constructores, métodos y campos estáticos.

Pegar elemento precargado

Invocando el comando '**pegar elemento precargado**' el nombre de la clase o campo de esta seleccionado en la vista será pegado en la entrada de la consola.

Reemplazar contenido del editor con elemento precargado

Invocando el comando '**reemplazar contenido del editor con elemento precargado**' el nombre de la clase o campo de esta seleccionado en la vista reemplazará cualquier contenido existente en la entrada de la consola.

Pegar instrucción

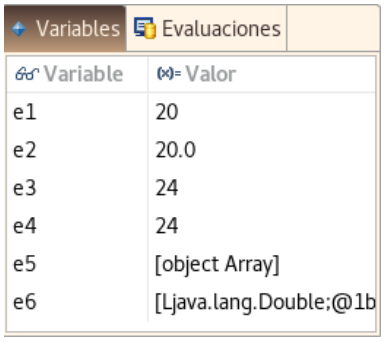
Invocando el comando '**pegar instrucción**' se generará una instrucción válida para ASE a partir del nombre de la clase o campo de esta seleccionado en la vista y será pegada en la entrada de la consola. El comando será también invocado haciendo doble clic sobre cualquier enlace.

Reemplazar contenido del editor con instrucción

Invocando el comando '**reemplazar contenido del editor con instrucción**' se generará una instrucción válida para ASE a partir del nombre de la clase o campo de esta seleccionado en la vista que reemplazará cualquier contenido existente en la entrada de la consola.

10.7 Vista de variables

La vista de variables muestra las variables que se han creado en la consola de ASE al realizar evaluaciones.



156-Vista de variables

Para cada variable la vista muestra su identificador y el valor del mismo.

Pegar variable

Invocando el comando '**pegar variable**' el nombre de la variable seleccionada en la vista será pegado en la entrada de la consola. El comando será también invocado haciendo doble clic sobre cualquier enlace.

Reemplazar contenido del editor con variable

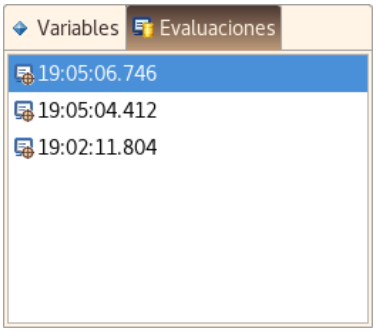
Invocando el comando '**reemplazar contenido del editor con variable**' el nombre de la variable seleccionada en la vista reemplazará cualquier contenido existente en la entrada de la consola.

Eliminar variable

Invocando el comando '**eliminar variable**' la variable seleccionada en la vista será eliminada. Cabe notar que la implementación de Nashorn usada no realiza correctamente la eliminación de variables, por lo que su eliminación es simulada estableciendo su valor a **null**.

10.8 Vista de evaluaciones

La vista de evaluaciones muestra una lista cronológica que permite navegar entre las diferentes evaluaciones realizadas en la consola.



157-Vista de evaluaciones

Por cada evaluación realizada la vista muestra una entrada con la hora exacta de ejecución de la misma. Dejando el cursor del ratón sobre cualquiera de ellas, la vista mostrará una ventana emergente que contendrá el código ejecutado y el resultado obtenido en la ejecución.

Pegar evaluación

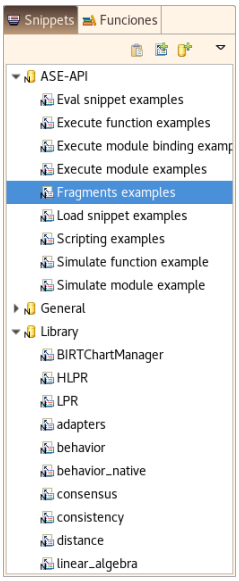
Invocando el comando '**pegar evaluación**' el código de la evaluación seleccionada será pegado en la entrada de la consola. El comando será también invocado haciendo doble clic sobre cualquier enlace.

Reemplazar contenido del editor con evaluación

Invocando el comando '**reemplazar contenido del editor con evaluación**' el código de la evaluación seleccionada en la vista reemplazará cualquier contenido existente en la entrada de la consola.

10.9 Vista de snippets

La vista de snippets muestra los snippets y repositorios existentes en el espacio de trabajo de la aplicación y permite trabajar con ellos.



158-Vista de snippets

La vista se organiza de forma jerárquica en forma de árbol, mostrándose en el primer nivel de la jerarquía los diferentes repositorios, los cuales pueden ser desplegados para visualizar los snippets contenidos.

La vista permite establecer el elemento seleccionado haciendo clic sobre él, no permitiéndose la selección múltiple de elementos.

Pegar snippet

Invocando el comando '**pegar snippet**' el código del snippet seleccionado será pegado en la entrada de la consola. El comando será también invocado haciendo doble clic sobre cualquier snippet.

Reemplazar contenido del editor con snippet

Invocando el comando '**reemplazar contenido del editor con snippet**' el código del snippet seleccionado en la vista reemplazará cualquier contenido existente en la entrada de la consola.

Eliminar snippet

Con la eliminación del snippet este será suprimido del espacio de trabajo. Como es lógico, la opción estará disponible siempre y cuando se haya seleccionado un snippet, no se trate de un elemento nativo de la aplicación y se dispongan de permisos de escritura en el directorio de trabajo.

Si se cumplen las condiciones anteriores, invocando el comando '**eliminar snippet**' será solicitada confirmación y eliminado el snippet.

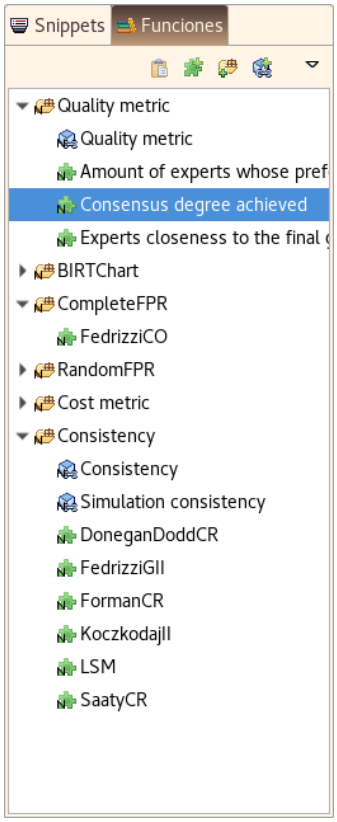
Eliminar repositorio

La eliminación de un repositorio suprime tanto el elemento como sus snippets del espacio de trabajo. Por ello, la opción estará disponible siempre y cuando se haya seleccionado un repositorio, no se trate de un elemento nativo de la aplicación, se dispongan de permisos de escritura en el directorio de trabajo y se cumplan las condiciones necesarias para poder eliminar sus snippets.

Si se cumplen las condiciones anteriores, invocando el comando '**eliminar repositorio**' será solicitada confirmación y eliminado el repositorio y sus snippets.

10.10 Vista de funciones

La vista de funciones muestra las funciones, módulos y enlaces a módulos existentes en el espacio de trabajo de la aplicación y permite trabajar con ellos.



159-Vista de funciones

De forma similar a la vista de snippets, la vista se organiza de forma jerárquica en forma de árbol, estando en el primer nivel de la jerarquía los diferentes módulos y en el segundo nivel los enlaces a módulos y funciones, los cuales son mostrados de arriba a abajo en este orden.

De igual modo que la vista de snippet, permite establecer el elemento seleccionado haciendo clic sobre él, no permitiéndose la selección múltiple de elementos.

Pegar función

Invocando el comando '**pegar función**' será generada una función a partir del código de la función seleccionada y los parámetros especificados por su módulo que será pegada en la entrada de la consola. El comando será también invocado haciendo doble clic sobre cualquier snippet.

Reemplazar contenido del editor con función

Invocando el comando '**reemplazar contenido del editor con función**' será generada una función a partir del código de la función seleccionada y los parámetros especificados por su módulo que reemplazará cualquier contenido existente en la entrada de la consola.

Eliminar función

Al eliminar una función esta es suprimida del espacio de trabajo. Al igual que con los snippets, la opción estará disponible siempre y cuando se haya seleccionado una función, no se trate de un elemento nativo de la aplicación y se dispongan de permisos de escritura en el directorio de trabajo.

Si se cumplen las condiciones anteriores, invocando el comando '**eliminar función**' será solicitada confirmación y eliminada la función.

Eliminar enlace a módulo

La eliminación de un enlace a módulo suprime el elemento del espacio de trabajo. Al igual que antes, la opción estará disponible siempre y cuando se haya seleccionado un enlace a módulo, no se trate de un elemento nativo de la aplicación y se dispongan de permisos de escritura en el directorio de trabajo.

Si se cumplen las condiciones anteriores, invocando el comando '**eliminar enlace a módulo**' será solicitada confirmación y eliminado el enlace a módulo.

Eliminar módulo

La eliminación de un módulo suprimirá tanto el elemento como sus funciones y enlaces a módulos del espacio de trabajo. Por ello, la opción estará disponible siempre y cuando se haya seleccionado un módulo, no se trate de un elemento nativo de la aplicación, se dispongan de permisos de escritura en el directorio de trabajo y se cumplan las condiciones necesarias para poder eliminar sus funciones y enlaces a módulos.

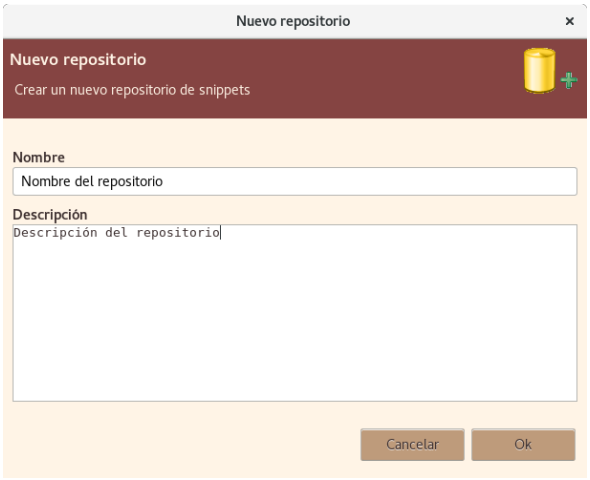
Si se cumplen las condiciones anteriores, invocando el comando '**eliminar módulo**' será solicitada confirmación y eliminado el módulo y sus funciones y enlaces a módulos.

10.11 Creación de un repositorio

AFRYCA ofrece una ventana de diálogo con la que es posible crear y editar repositorios.

Crear repositorio

Para crear un repositorio únicamente es necesario contar con permisos de escritura en el espacio de trabajo. Si se disponen de estos permisos será posible invocar el comando '**nuevo repositorio**', el cual desplegará una ventana de diálogo que permitirá realizar su creación.



160-Nuevo repositorio

Para la creación se deben especificar los siguientes campos:

- **Nombre**: Identificador del repositorio.
- **Descripción**: Descripción del repositorio.

Introducida la información solicita y tras pulsar el botón '**Ok**' AFRYCA solicitará el nombre del archivo para el nuevo repositorio y realizará su creación en el espacio de trabajo.

Editar repositorio

AFRYCA permite modificar los repositorios del espacio de trabajo, para lo cual es empleado el mismo cuadro de diálogo anterior en modo edición.

161-Edición del repositorio ASE-API

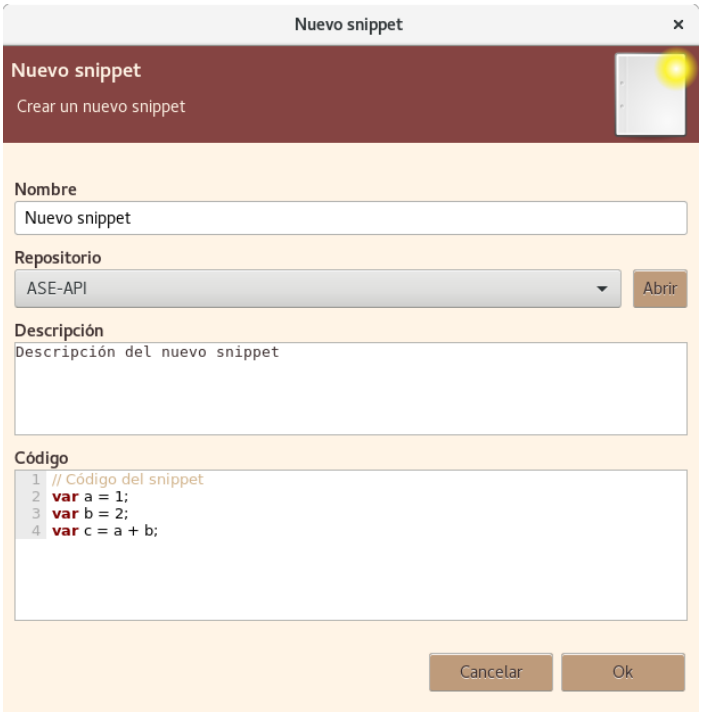
Siempre y cuando se haya seleccionado un repositorio será posible invocar el comando '**editar repositorio**' para realizar su edición. En caso de que no se dispongan de permisos para editar el elemento por tratarse de un elemento nativo o no disponerse de permisos en el espacio de trabajo, el cuadro de diálogo no permitirá su modificación, pero si permitirá visualizar su contenido.

10.12 Creación de un snippet

Igual que con los repositorios, AFRYCA ofrece una ventana de diálogo desde la que es posible crear y editar snippets.

Crear snippet

Para crear un snippet es necesario que exista al menos un repositorio y contar con permisos de escritura en el espacio de trabajo. Si se cumplen estas condiciones será posible invocar el comando '**nuevo snippet**', el cual desplegará una ventana de diálogo que permitirá realizar su creación.



162-Nuevo Snippet

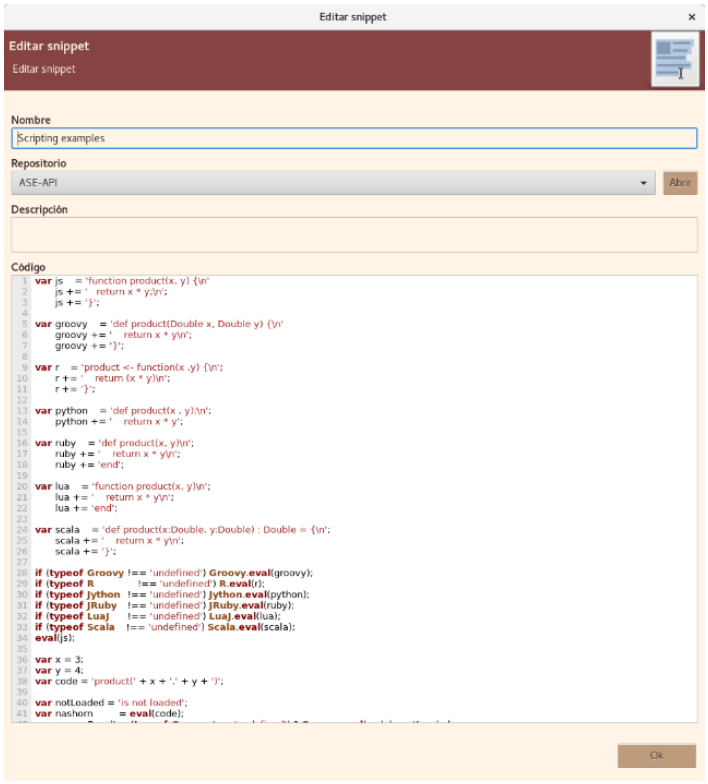
Para la creación se deben especificar los siguientes campos:

- **Nombre**: Identificador del snippet.
- **Repositorio**: Identificador del repositorio en el que estará contenido el snippet. Puede pulsarse el botón '**Abrir**' para visualizar sus datos.
- **Descripción**: Descripción del snippet.
- **Código**: Código fuente del snippet.

Introducida la información solicita y tras pulsar el botón '**Ok**' AFRYCA solicitará el nombre del archivo para el nuevo snippet y realizará su creación en el espacio de trabajo.

Editar snippet

Al igual que con los repositorios, AFRYCA permite modificar los snippets del espacio de trabajo, para lo cual es empleado el mismo cuadro de diálogo anterior en modo edición.



163-Edición del snippet

Scripting examples

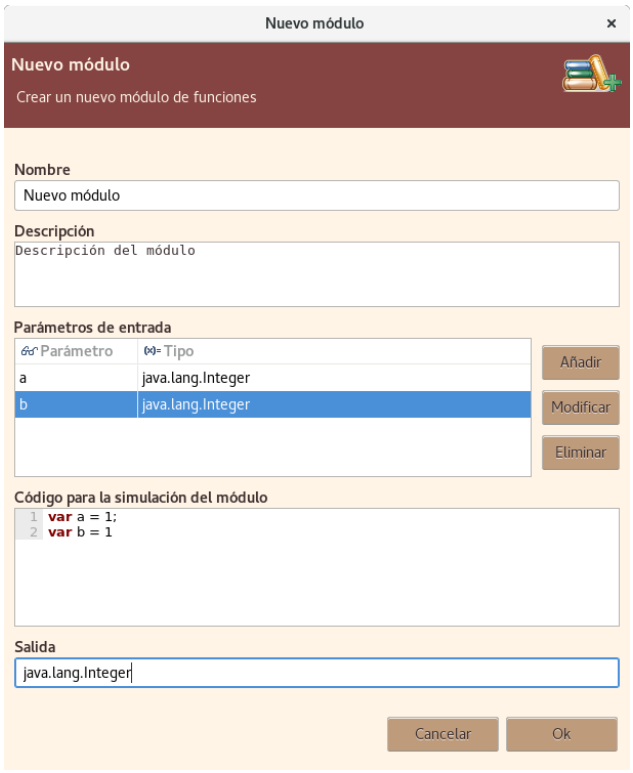
Nuevamente, será posible invocar el comando '**editar snippet**' siempre que exista un snippet seleccionado. Si el snippet se trata de un elemento nativo o no se disponen de permisos de escritura en el espacio de trabajo el cuadro de diálogo no permitirá su modificación.

10.13 Creación de un módulo

AFRYCA ofrece una ventana de diálogo desde la que es posible crear y editar módulos.

Crear módulo

Para crear un módulo únicamente es necesario contar con permisos de escritura en el espacio de trabajo. Si se disponen de estos permisos será posible invocar el comando '**nuevo módulo**', el cual desplegará una ventana de diálogo que permitirá realizar su creación.

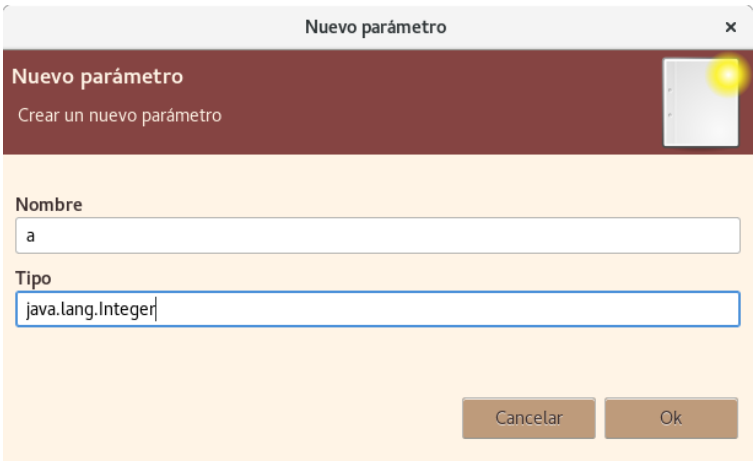


164-Nuevo módulo

Para la creación se deben especificar los siguientes campos:

- **Nombre**: Identificador del módulo.
- **Descripción**: Descripción del módulo.
- **Parámetros de entrada**: Parámetros de entrada para las funciones del módulo. Un parámetro estará caracterizado por su identificador y su tipo, ofreciendo AFRYCA una ventana de diálogo que facilita su definición.
- **Código para la simulación del módulo**: Código fuente para la simulación del módulo.
- **Salida**: Tipo de la salida del módulo.

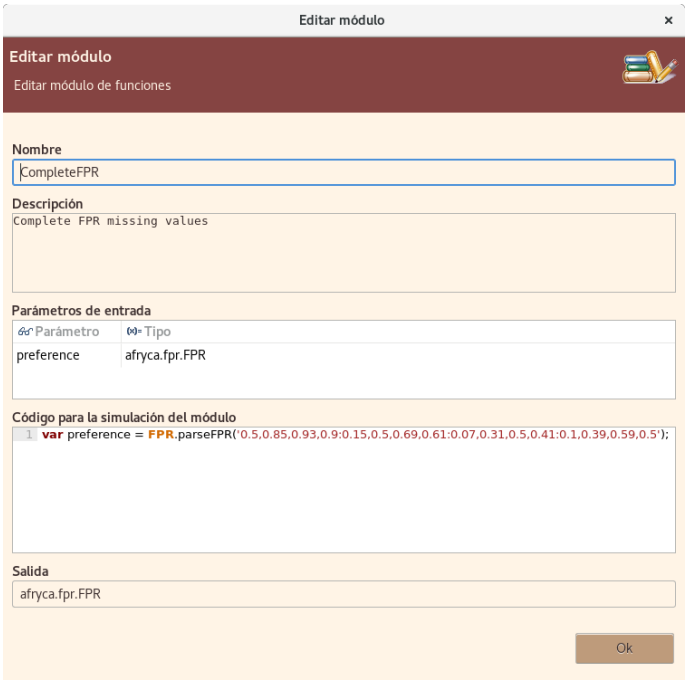
Introducida la información solicita y tras pulsar el botón '**Ok**' AFRYCA solicitará el nombre del archivo para el nuevo módulo y realizará su creación en el espacio de trabajo.



165-Nuevo parámetro de módulo

Editar módulo

Al igual que en los otros casos, AFRYCA permite modificar los módulos del espacio de trabajo empleado el mismo cuadro de diálogo anterior en modo edición.



166-Edición del módulo CompleteFPR

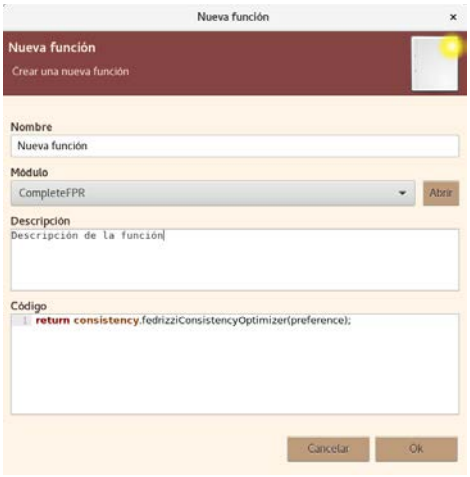
Nuevamente, será posible invocar el comando '**editar módulo**' siempre que exista un módulo seleccionado, pero si se trata de un elemento nativo o no se disponen de permisos de escritura en el espacio de trabajo el cuadro de diálogo no permitirá su modificación.

10.14 Creación de una función

Como en los casos anteriores, AFRYCA ofrece una ventana de diálogo desde la que es posible crear y editar funciones.

Crear función

De forma similar a la creación de un snippet, para crear una función es necesario que exista al menos un módulo y contar con permisos de escritura en el espacio de trabajo. Si se cumplen estas condiciones será posible invocar el comando '**nueva función**', el cual desplegará una ventana de diálogo que permitirá realizar su creación.



Nueva función

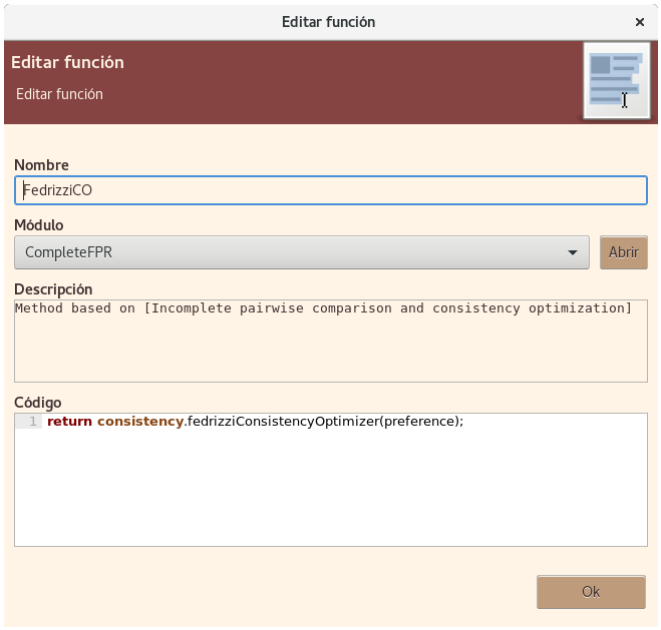
Para la creación se deben especificar los siguientes campos:

- **Nombre**: Identificador de la función.
- **Módulo**: Identificador del módulo en el que estará contenida la función. Puede pulsarse el botón '**Abrir**' para visualizar sus datos.
- **Descripción**: Descripción de la función.
- **Código**: Código fuente de la función. Su última línea debe de comenzar con la palabra reservada '**return**' seguida del valor a devolver por la función.

Introducida la información solicita y tras pulsar el botón '**Ok**' AFRYCA solicitará el nombre del archivo para la nueva función y realizará su creación en el espacio de trabajo.

Editar función

Como en los otros casos, AFRYCA permite modificar las funciones del espacio de trabajo empleado el mismo cuadro de diálogo anterior en modo edición.



167-Edición de la función FedrizziCO

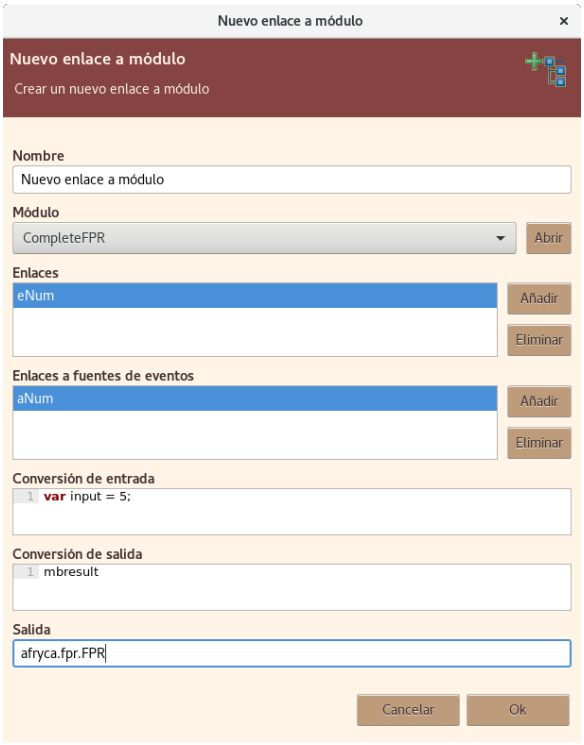
Siempre y cuando se haya seleccionado una función será posible invocar el comando '**editar función**', pero como en los otros casos, si se trata de un elemento nativo o no se disponen de permisos de escritura en el espacio de trabajo el cuadro de diálogo no permitirá su modificación.

Creación de un enlace a módulo

Para la creación y edición de los enlaces a módulos AFRYCA ofrece una nueva ventana de diálogo.

Crear enlace a módulo

De forma idéntica a las funciones, para crear un enlace a módulo es necesario que exista al menos un módulo y contar con permisos de escritura en el espacio de trabajo. Si se cumplen estas condiciones será posible invocar el comando '**nuevo enlace a módulo**', el cual desplegará una ventana de diálogo que permitirá realizar su creación.

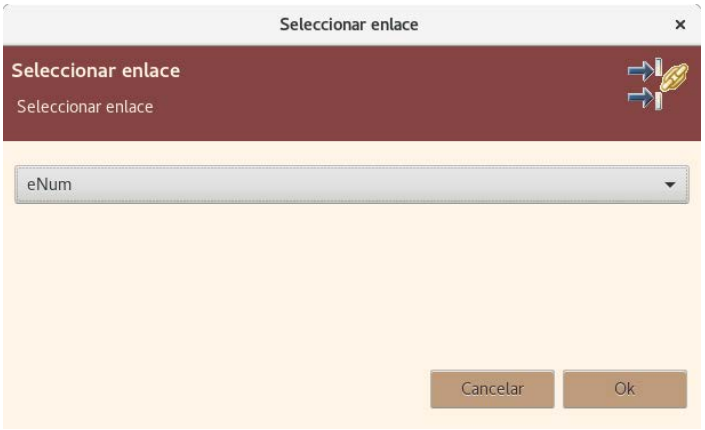


168-Nuevo enlace a módulo

Para la creación se deben especificar los siguientes campos:

- **Nombre**: Identificador del enlace a módulo.
- **Módulo**: Identificador del módulo que será enlazado. Puede pulsarse el botón '**Abrir**' para visualizar sus datos.
- **Enlaces**: Variables de AFRYCA enlazadas desde ASE que deben tener un valor asignado para que el módulo pueda evaluarse. AFRYCA ofrece una ventana de diálogo para su selección.
- **Enlaces a fuentes de eventos**: Variables de AFRYCA enlazadas desde ASE cuya modificación implicará la actualización del enlace a módulo. AFRYCA ofrece una ventana de diálogo para su selección
- **Conversión de entrada**: Código para la preparación de la invocación del módulo enlazado.
- **Conversión de salida**: Código para la adaptación de los resultados a partir de los resultados del módulo enlazado. El resultado del módulo está disponible en la variable reservada '**mbresult**'.
- **Salida**: Tipo de la salida del enlace a módulo.

Introducida la información solicita y tras pulsar el botón '**Ok**' AFRYCA solicitará el nombre del archivo para el nuevo enlace a módulo y realizará su creación en el espacio de trabajo.



169-Selección de enlace

Editar enlace a módulo

Al igual que en todos los casos vistos, AFRYCA permite modificar los enlaces a módulos del espacio de trabajo empleado el mismo cuadro de diálogo anterior en modo edición.

170-Edición del enlace a módulo Simulation consistency

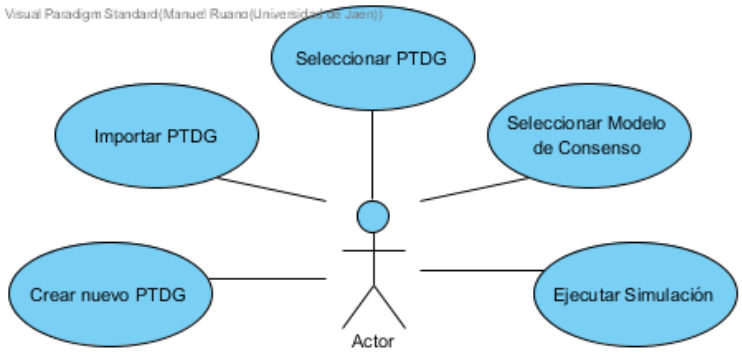
Si se ha seleccionado un enlace a módulo será posible invocar el comando '**editar enlace a módulo**', pero como siempre, si se trata de un elemento nativo o no se disponen de permisos de escritura en el espacio de trabajo el cuadro de diálogo no permitirá su modificación

11. Diseño

En esta sección analizaremos los diferentes diagramas de casos de uso y los diagramas de secuencia detallando, para cada uno de ellos, mediante un breve texto su funcionalidad.

11.1 Diagramas de caso de uso

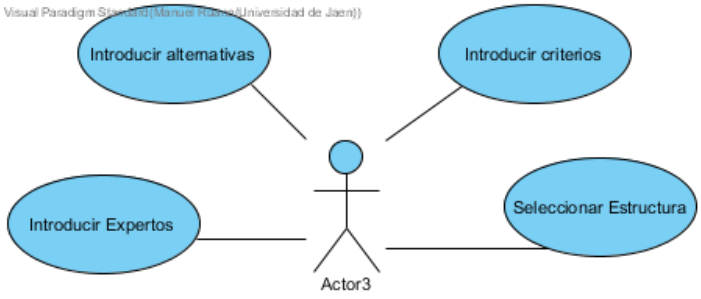
11.1.1 Caso de uso principal



171-Caso de uso pincipal

El caso de uso principal representa la funcionalidad general de la aplicación de AFRYCA, en la cual el actor una vez ejecutada AFRYCA podrá “Crear un nuevo PTDG” o “Importar PTDG”, una vez creado o importado podrá “Seleccionar PTDG”, tambien deberá “Seleccionar Modelo de Consenso” para finalmente “Ejecutar Simulación”.

11.1.2 Caso de uso Crear Nuevo PTDG

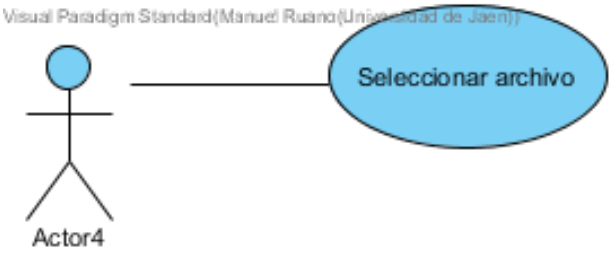


172-Caso de uso crear nuevo PTDG

El usuario puede crear un nuevo PTDG, para esto se le pedirá la siguiente información:

- Introducir Expertos
- Introducir Alternativas
- Introducir Criterios
- Seleccionar Estructura

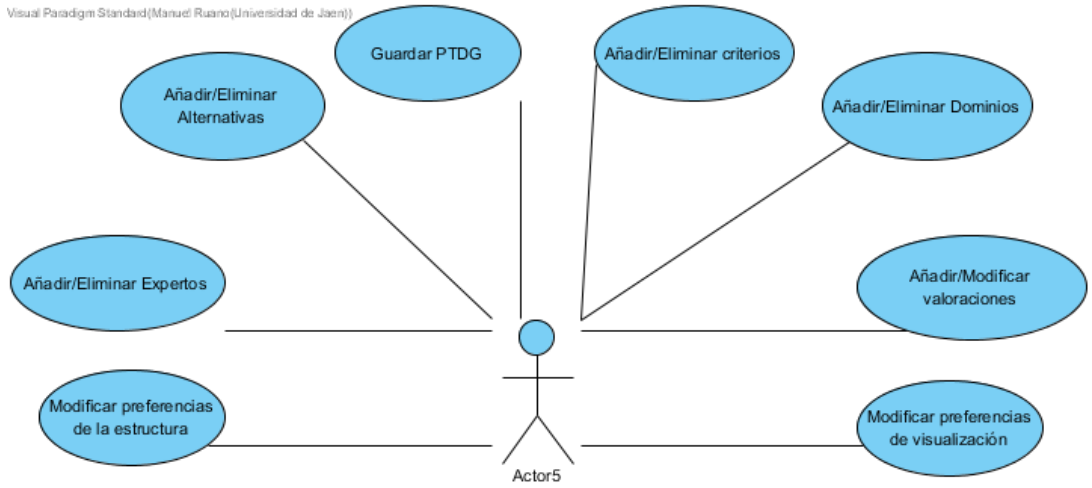
11.1.3 Importar PTDG



173-Caso de uso Importar PTDG

Para importar un PTDG se le pedirá al usuario que seleccione el archivo con el que desea trabajar.

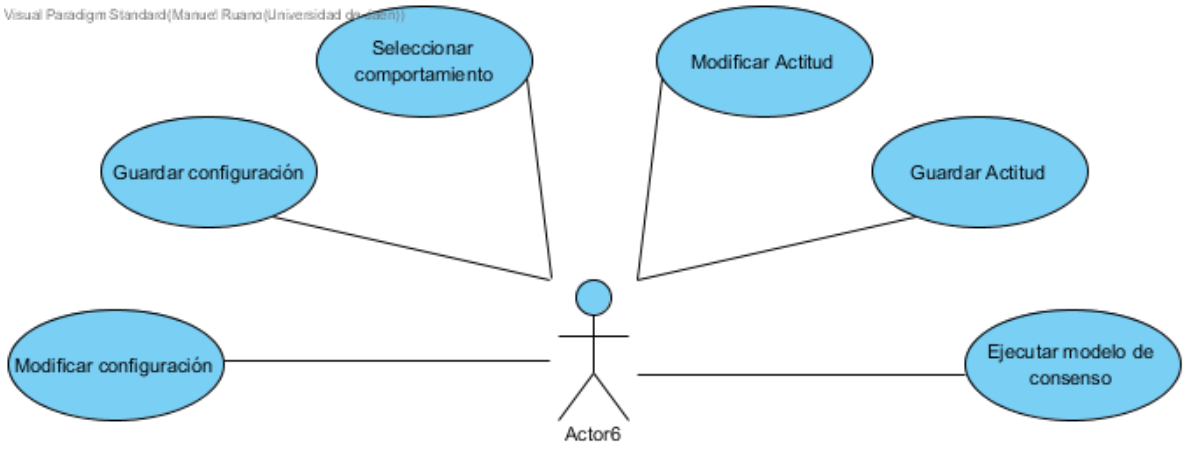
11.1.4 Seleccionar PTDG



174-Caso de uso seleccionar PTDG

Una vez el usuario ha seleccionado un PTDG, el usuario podrá Modificar las preferencias de visualización, Modificar las preferencias de la estructura del PTDG, Añadir/Eliminar Expertos, Alternativas, Criterios y Dominios en el caso de que sea soportado por la estructura del PTDG y añadir o modificar las valoraciones de los expertos.

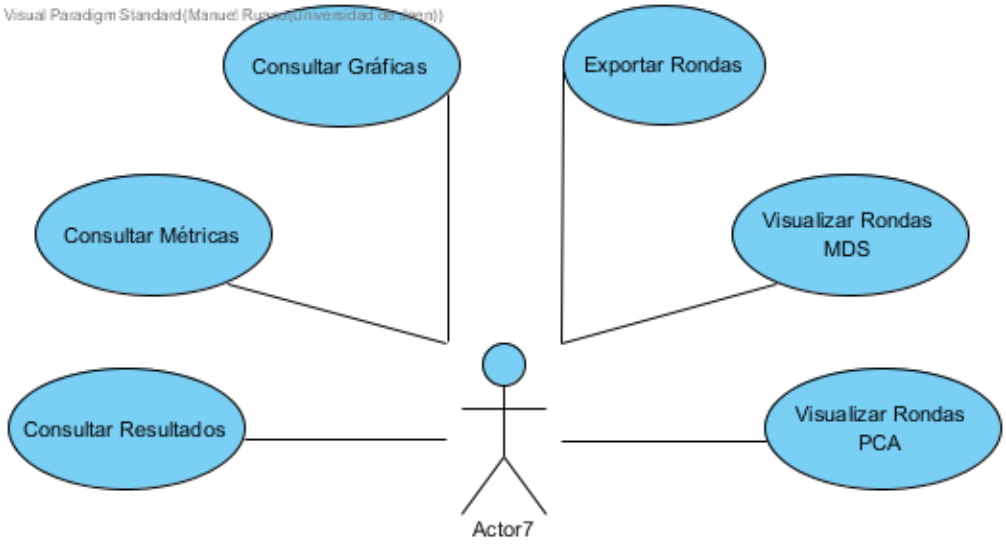
11.1.5 Seleccionar Modelo de Consenso



175-Caso de uso seleccionar modelo de consenso

Una vez el usuario ha seleccionado un modelo de consenso compatible con la estructura del PTDG seleccionado, el usuario podrá modificar la configuración del modelo de consenso asi como guardarla para posteriores ejecuciones, seleccionar un comportamiento, modificar una actitud y guardarla y por último ejecutar el modelo de consenso.

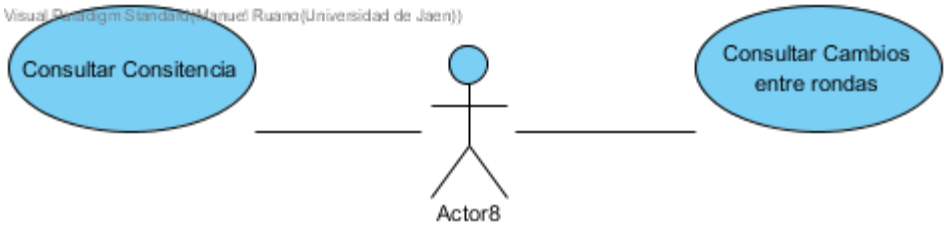
11.1.6 Caso de Uso Ejecutar Simulación



176-Caso de uso ejecutar simulación

Una vez el usuario a ejecutado la simulación, el usuario podrá consultar los resultados obtenidos, las métricas, las gráficas, también podrá visualizar las preferencias de cada experto y la colectiva en cada ronda tanto en MDS como en PCA y podrá exportar esta visualización.

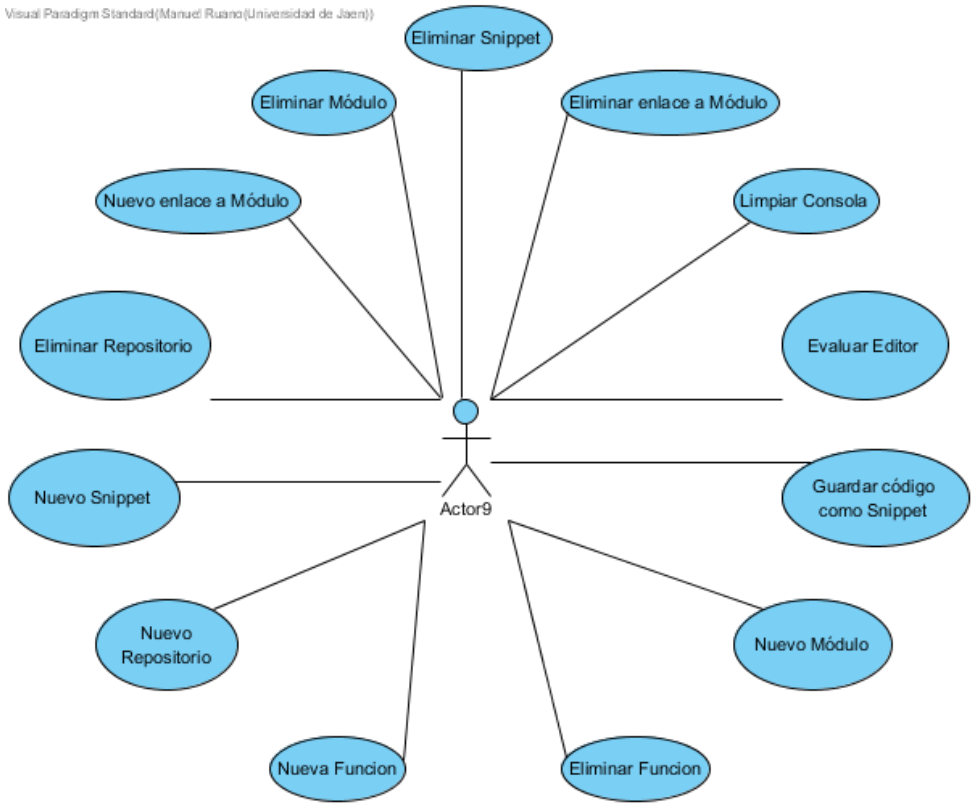
11.1.7 Caso de uso Simulaciones



177-Caso de uso simulaciones

En la perspectiva de simulaciones el usuario podrá consultar la consistencia en cada ronda y los cambios en las preferencias entre diferentes rondas.

11.1.8 Caso de uso ASE



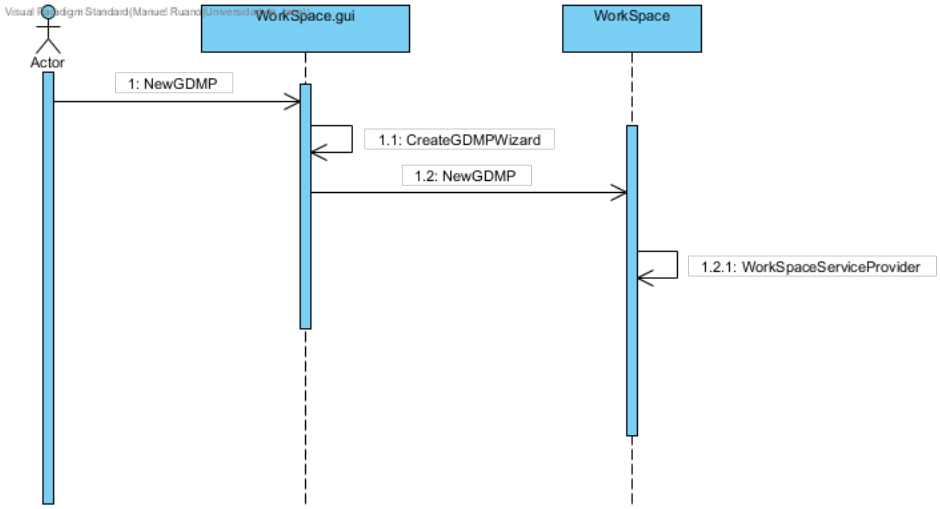
178-Caso de uso ASE

Desde la perspectiva de ASE el usuario puede crear/eliminar una función, un repositorio, un snippet o un enlace a módulo, si está usando la consola podrá limpiar la consola, evaluar el editor que ejecutará el código de la consola o guardar el código como un Snippet.

11.2 Diagramas de secuencia

En esta sección se analizarán los diagramas de secuencia de la aplicación de AFRYCA, para facilitar la compresión de estos diagramas y dado que estamos utilizando una tecnología basada en plugin, se ha optado por diseñar los diagramas de secuencia en base a los plugin y a las clases afectadas por cada uno de ellos.

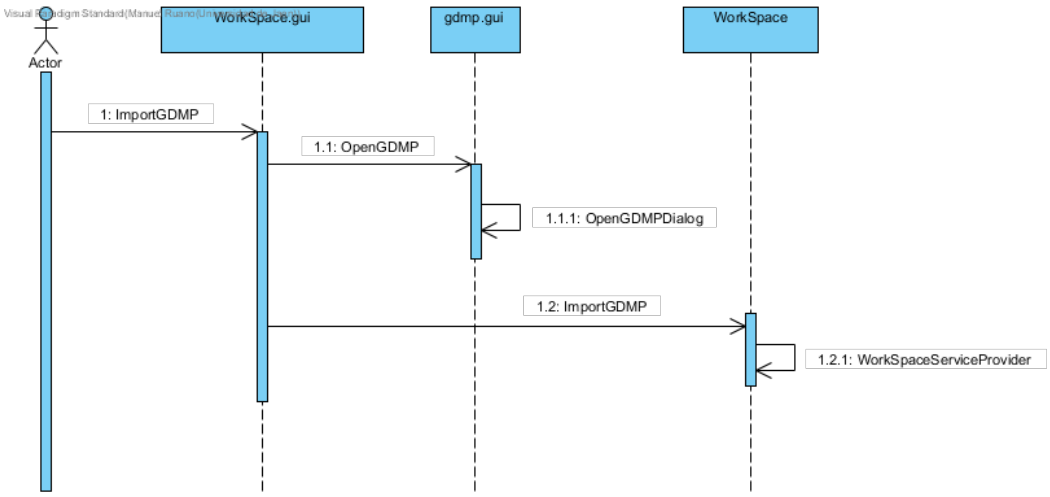
11.2.1 Crear Nuevo PTDG



179-Diagrama de secuencia nuevo PTDG

Cuando el usuario pulsa sobre el botón de crear nuevo PTDG el comando NewGDMP es lanzado, éste llama a la clase CreateGDMPWizard que se encarga de crear un Wizard para que el usuario introduzca los datos del PTDG como son los expertos, alternativas, criterios, el tipo de estructura y el nombre. Por último, el PTDG creado es añadido al servicio WorkSpaceServiceProvider.

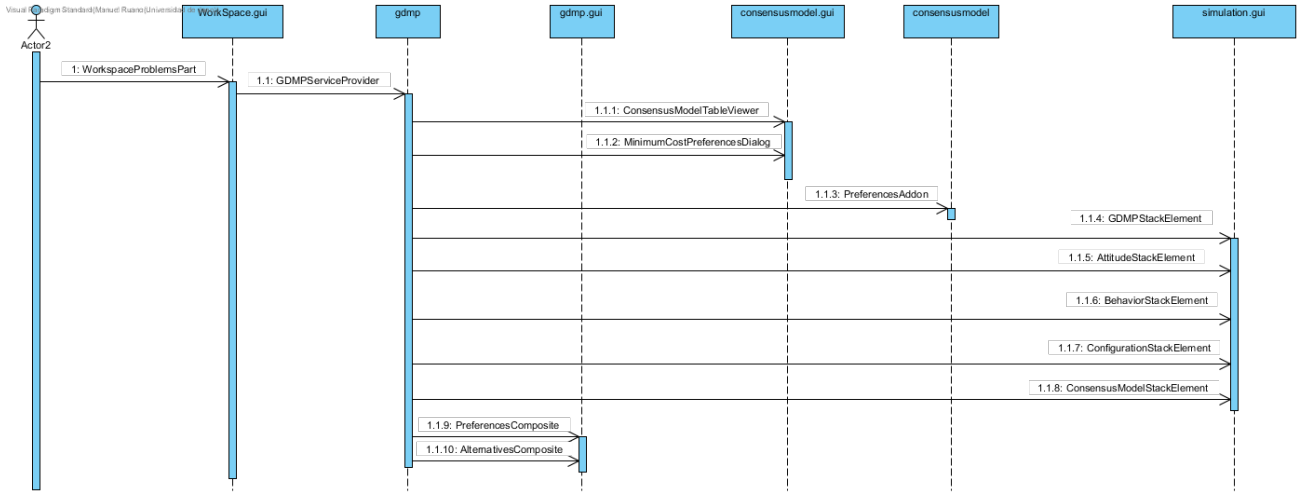
11.2.2 Importar PTDG



180-Diagrama de secuencia impoartar PTDG

Al pulsar sobre importar PTDG se activa el comando ImportGDMP que mediante la clase OpenGDMP lanza un dialog para seleccionar el archivo a importar, después es llamada la clase ImportGDMP del plugin workspace que, usando el sevicio WorkSpaceServiceProvider, almacena el GDMP.

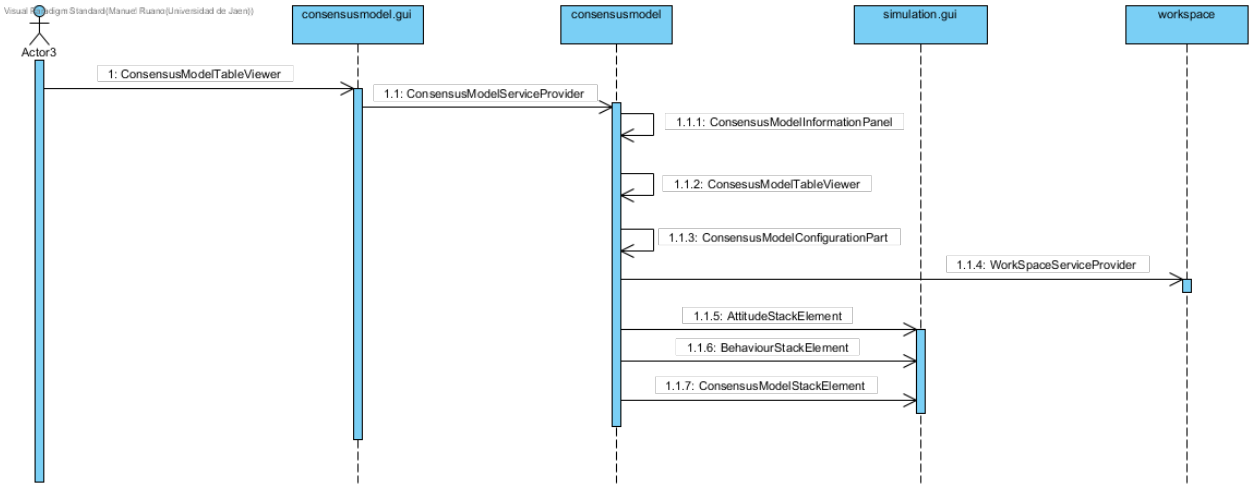
11.2.3 Seleccionar GDMP



181-Diagrama de secuencia seleccionar GDMP

Cuando el usuario selecciona un GDMP la tabla contiene un listener que llama a la clase GDMPServiceProvider del plugin gdmp donde se actualiza la selección en el servicio, al ser actualizada esta selección se mandan eventos a otras clases como son ConsensusModelTableViewer que actualizará la tabla de los modelos de consenso cambiando el color del nombre de los modelos, GDMPStackElement, AttitudeStackElemnt, BehaviorStackElement,ConfigurationStackElement y ConsensusModelStackElement actualizan la pila con la información del PTDG seleccionado, PreferencesComposite y AlternativesComposite se actualizarán en función de la preferencia mostrada y por último MinimumCostPreferencesDialog y PreferencesAddon que calcularán los pesos para el mínimo coste.

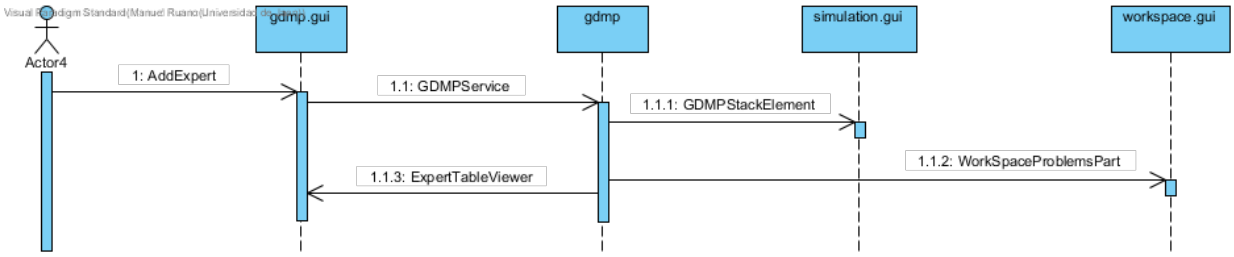
11.2.4 Seleccionar Modelo de Consenso



182-Diagrama de secuencia seleccionar modelo de consenso

Cuando un usuario selecciona un modelo de consenso se ejecuta un listener asociado a la tabla, en él se llama al servicio ConsensusModelServiceProvider el cual actualiza la selección. Al actualizar la selección se actualizan las clases ConsensusModelInformationPanel, ConseussModelTableViewer y ConsensusModelConfigurationPart. Además, son lanzados unos eventos son llamados en las clases de AttitudeStackElement, BehaviourStackElement y ConsensusStackElement para actualizar los datos de la pila y al servicio de WorkSpaceServiceProvider del plugin workspace que muestra la configuración del modelo de consenso seleccionado.

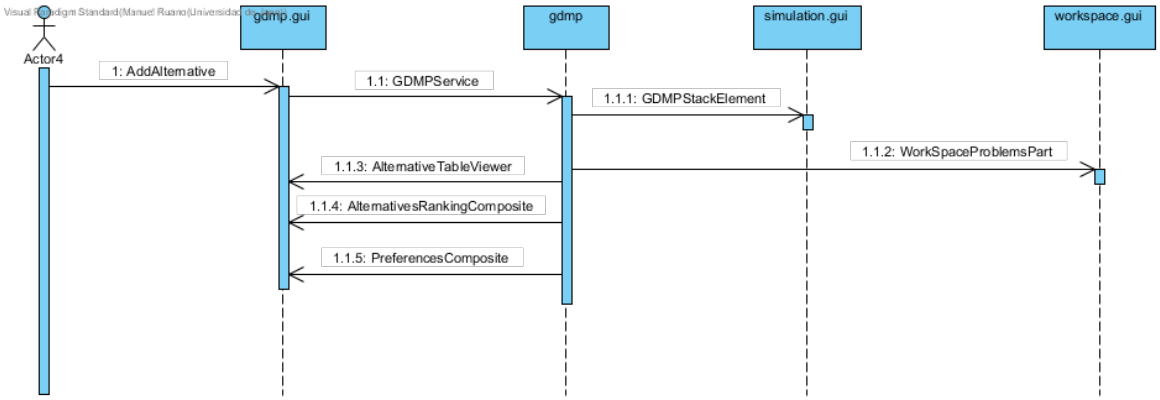
11.2.5 Añadir Experto



183-Diagrama de secuencia añadir experto

Al pulsar sobre añadir experto el comando AddExpert es ejecutado, genera un dialog para introducir el nombre, una vez finalizado el dialog, el experto es añadido al problema mediante el servicio GDMPServiceProvider del plugin gdmp, ahora son lanzados unos eventos en la clase ExpertTableViewer que actualiza la tabla de los expertos, en la clase GDMPStackElement que actualiza la pila y la clase WorkSpcaeProblemsPart para redibujar la pantalla con los nuevos datos.

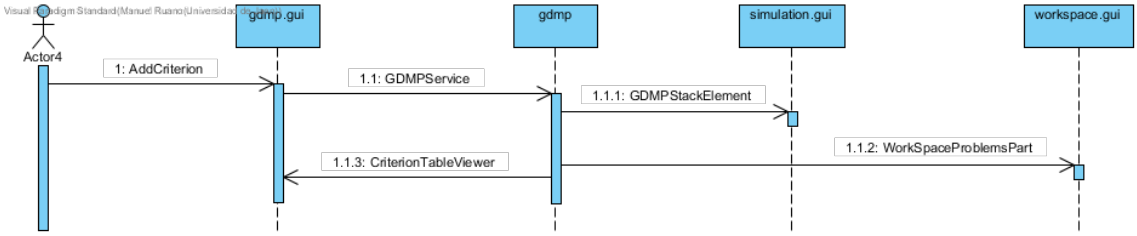
11.2.6 Añadir Alternativa



184-Diagrama de secuencia añadir alternativa

Al pulsar sobre añadir alternativa el comando AddAlternative es ejecutado, genera un dialog para introducir el nombre, una vez finalizado el dialog, la alternativa es añadida al problema mediante el servicio GDMPServiceProvider del plugin gdmp, ahora son lanzados unos eventos en la clase AlternativeTableViewer que actualiza la tabla de las alternativas, en la clase GDMPStackElement que actualiza la pila, en la clase alternativesRankingcomposite que actualiza el ranking, la clase PreferencesComposite que modifica la preferencia activa con una alternativa más y la clase WorkSpaceProblemsPart para redibujar la pantalla con los nuevos datos.

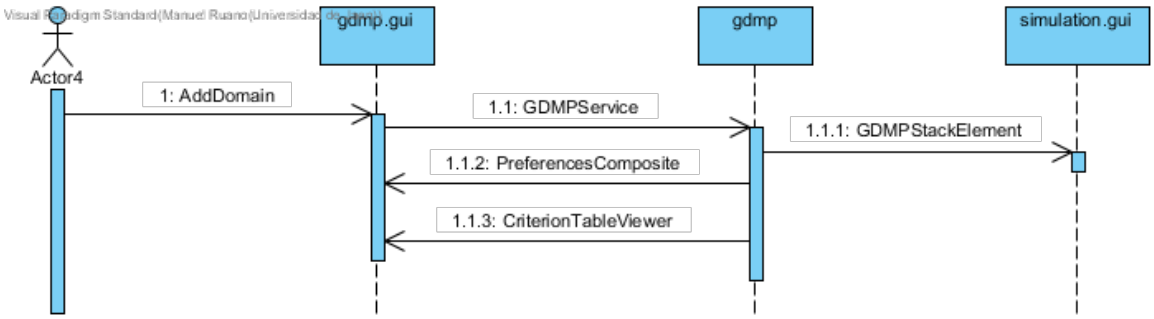
11.2.7 Añadir criterio



185-Diagrama de secuencia añadir criterio

Al pulsar sobre añadir criterio el comando AddCriteria es ejecutado, genera un dialog para introducir el nombre, una vez finalizado el dialog, el criterio es añadido al problema mediante el servicio GDMPServiceProvider del plugin gdmp, ahora son lanzados unos eventos en la clase CriteriaTableViewer que actualiza la tabla de los criterios, en la clase GDMPStackElement que actualiza la pila y la clase WorkSpaceProblemsPart para redibujar la pantalla con los nuevos datos.

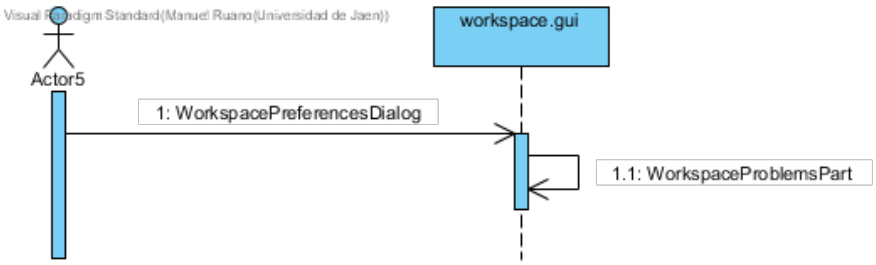
11.2.8 Añadir Dominio



186-Diagrama de secuencia añadir dominio

Al pulsar sobre añadir dominio el comando AddDomain es ejecutado, genera un dialog para seleccionar el tipo y configurarlo, una vez finalizado el dialog, el dominio es añadido al problema mediante el servicio GDMPServiceProvider del plugin gdmp, ahora son lanzados unos eventos en la clase DomainTableViewer que actualiza la tabla de los dominios, en la clase GDMPStackElement que actualiza la pila, la clase PreferencesComposite y la clase WorkSpaceProblemsPart para redibujar la pantalla con los nuevos datos.

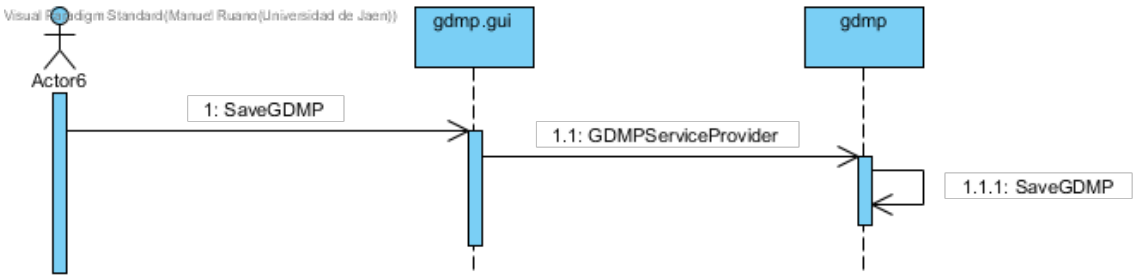
11.2.9 Modificar Preferencias del Espacio de Trabajo



187-Diagrama de secuencia modificar preferencias del espacio de trabajo

Cuando el usuario selecciona Modificar Preferencias del Espacio de Trabajo la clase WorkspacePreferencesDialog crea un dialogo en el que el usuario puede modificar la ruta del espacio de trabajo y la visualización de los problemas, una vez modificados mediante la clase WorkspaceProblemsPart es visualización es modificada y actualizada.

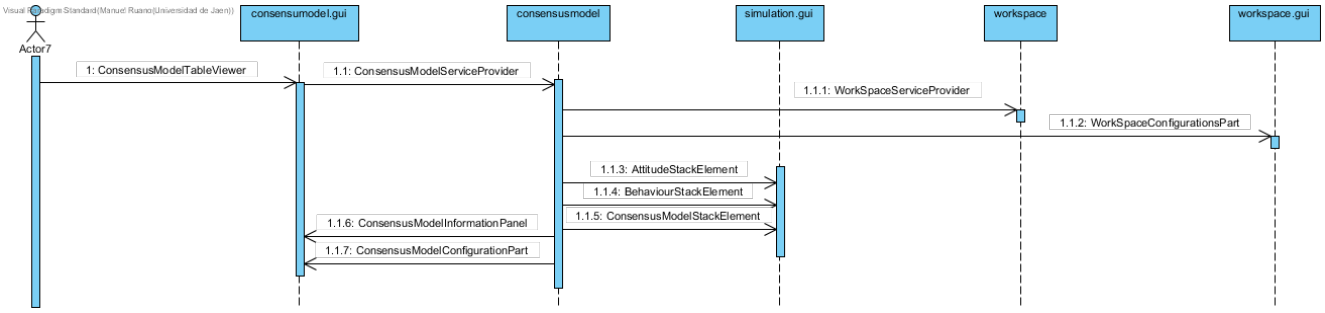
11.2.10 Guardar PTDG



188-Diagrama de secuencia guardar PTDG

Cuando el usuario pulsa sobre GuardarPTDG el commando SaveGDMP es lanzado, este consulta al servicio GDMPServiceProvider para obtener el GDMP a guardar y mediante el la clase SaveGDMP del plugin gdmp es almacenada en un fichero.

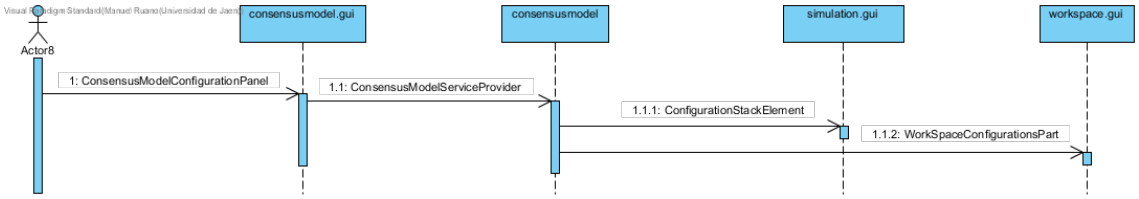
11.2.11 Seleccionar Modelo de Consenso



189-Diagrama de secuencia seleccionar modelo de consenso

Cuando el usuario selecciona un modelo de consenso la clase ConsensusModelTableViewer del plugin consensusmodel.gui actualiza la selección del modelo de consenso en el servicio ConsensusModelServiceProvider, el servicio se encarga de enviar unos eventos a las siguientes clases para actualizar información. La clase ConsensusModelInformationPanel actualiza la información mostrada acerca del modelo de consenso, la clase ConsensusModelConfigurationPart es actualizada para mostrar la configuración del modelo de consenso, la clase AttitudeStackElement, BehaviourStackelement y ConsesusModelStackElement actualizan la pila y mediante las clases WorkSpaceServiceProvider y WorkSpaceConfigurationPart se actualizan las configuraciones del modelo.

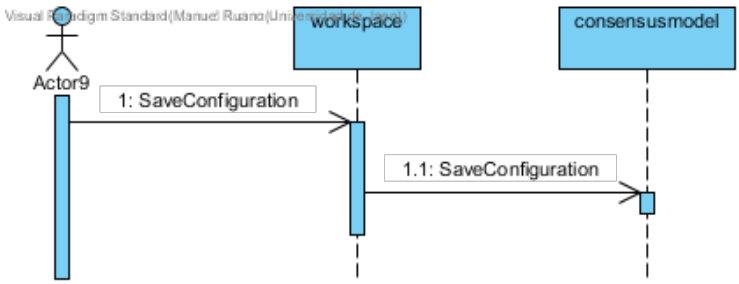
11.2.12 Modificar Configuración Modelo de Consenso



190-Diagrama de secuencia modificar condiguracion modelo de consenso

Cuando el usuario modifica la configuración en un modelo de consenso la clase ConsensusModelConfigurationPanel mediante un job llama al servicio ConsensusModelServiceProvider que actualiza la configuración almacenada, este mediante eventos llama a la clase ConfigurationStackElement para actualizar la pila y a la clase WorkScapeConfigurationPArt que actualiza la vista de la configuración.

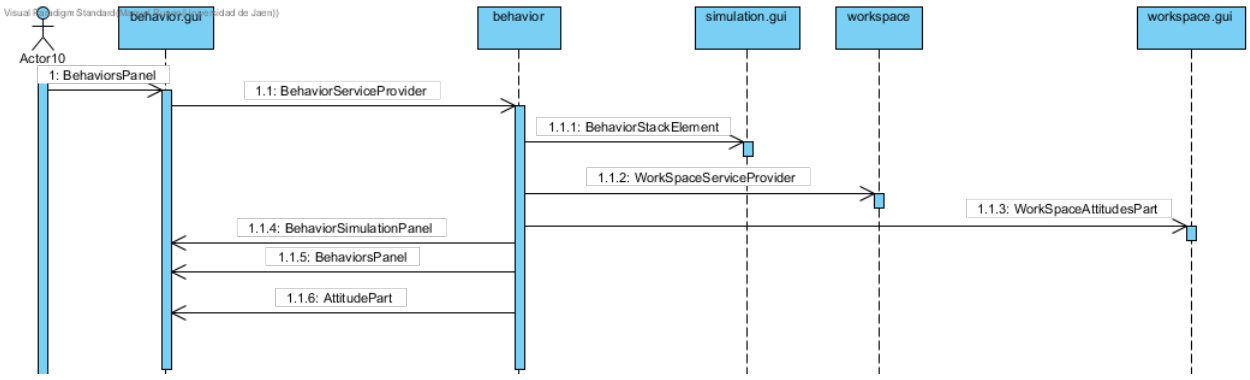
11.2.13 Guardar configuración Modelo de Consenso



191-Diagrama de secuencia guardar configuración modelo de consenso

Cuando el usuario pulsa sobre guardar configuración el comando SaveConfiguración del plugin workspace es lanzado, este obtiene la información a guardar y llama a la clase SaveConfiguration del plugin consensusmodel que es la encargada de escribir en el fichero.

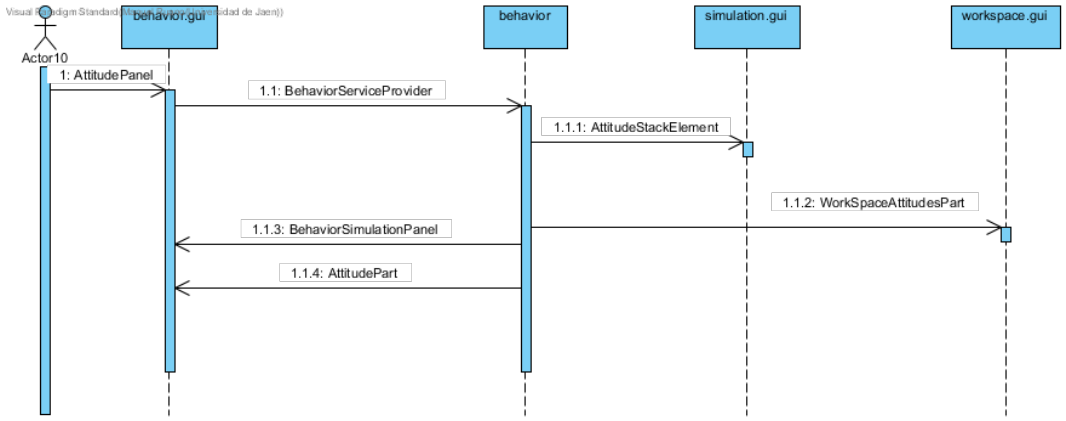
11.2.14 Seleccionar comportamiento



192-Diagrama de secuencia seleccionar comportamiento

Cuando el usuario selecciona un comportamiento la clase BehaviorsPanel se encarga de llamar al servicio BehaviorServiceProvider para actualizar la selección, el servicio también se encarga de enviar una serie de eventos a diferentes clases como son, la clase BehaviorStackElement actualiza la pila, mediante el servicio WorkSpaceServiceProvider se obtiene la información por defecto de la actitud que también es actualizada en la clase WorkSpaceAttitudesPart, la clase BehaviorSimulationPanel se encarga de generar los cambios, la clase BehaviorsPanel es llamada para actualizar la selección y la clase AttitudePart activa los paneles con las actitudes y la configuración.

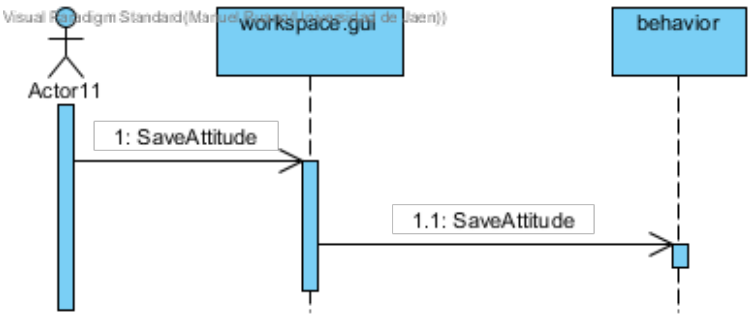
11.2.15 Modificar Actitud/Seleccionar Actitud



193-Diagrama de secuencia modificar/seleccionar actitud

Cuando el usuario selecciona una actitud la clase AttitudePanel se encarga de llamar al servicio BehaviorServiceProvider para actualizarla, el servicio también se encarga de enviar una serie de eventos a diferentes clases como son la clase AttitudeStackElement para actualizar la pila, a la clase WorkSapceAttitudesPart que actualiza la selección, a la clase BehaviorSimulationPanel que genera los cambios y la clase AttitudePanel que actualiza los datos de la selección.

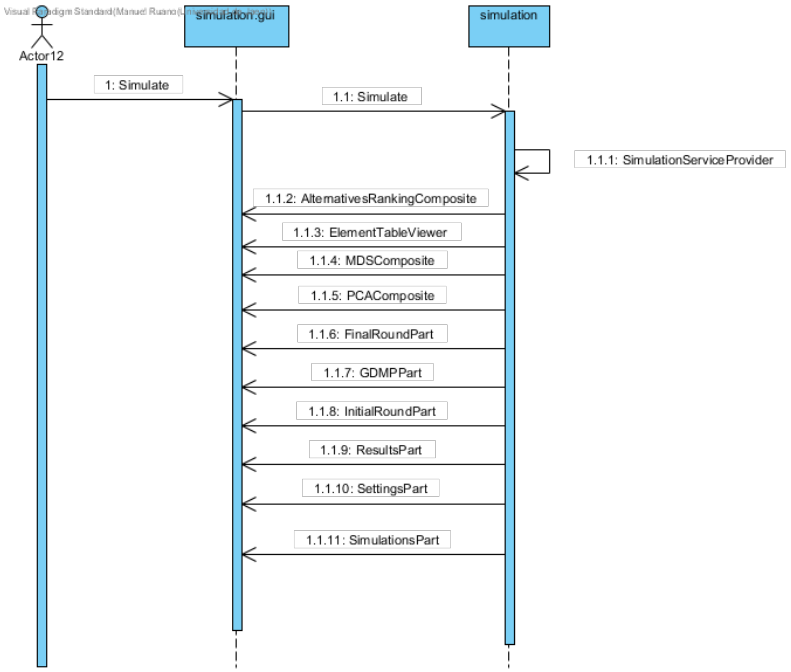
11.2.16 Guardar Actitud



194-Diagrama de secuencia guardar actitud

Cuando el usuario pulsa sobre guardar actitud, el comando SaveAttitude del plugin workspace.gui es ejecutado que se encarga de llamar a la clase SaveAttitude del plugin behavior que se encarga de almacenar la actitud en un fichero.

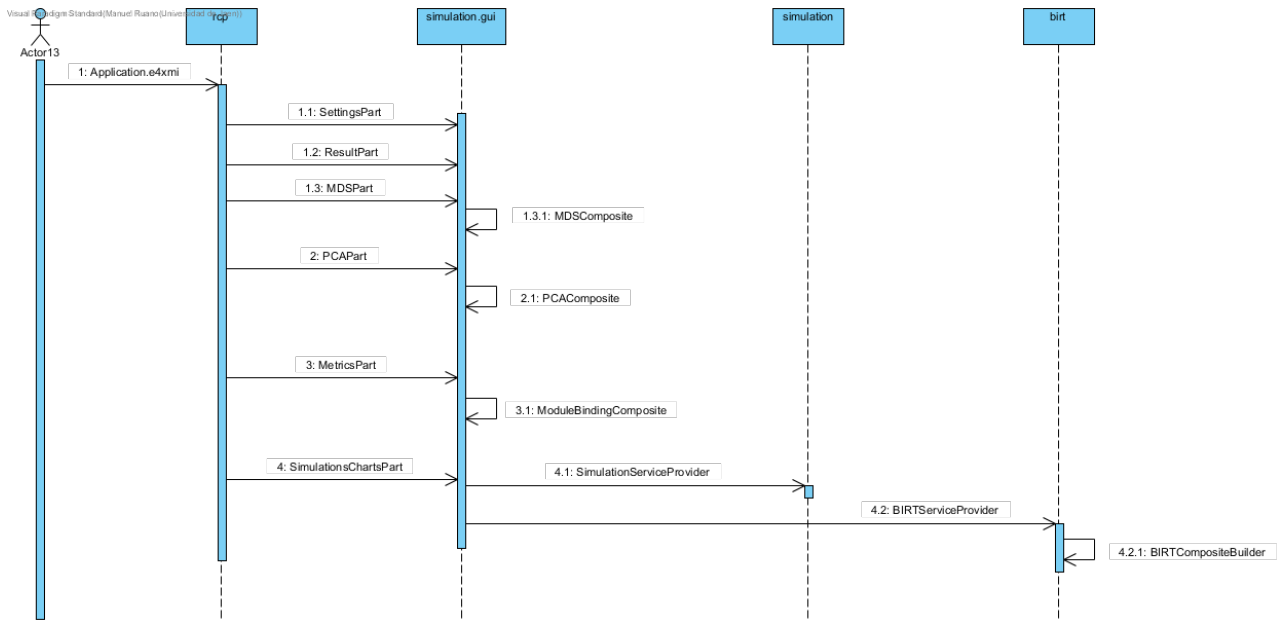
11.2.17 Ejecutar Simulación



195-Diagrama de secuencia ejecutar simulación

Cuando el usuario pulsa sobre ejecutar simulación se ejecuta el comando simulate del plugin simulation.gui que este a su vez llama a la clase Simulate del plugin simulation, esta clase es la encargada de llamar al servicio SimulationServiceProvider que ejecuta el modelo de consenso. Una vez ejecutado el servicio genera unos eventos que llaman a las clases de la imagen las cuales se actualizan para mostrar la información de la ejecución.

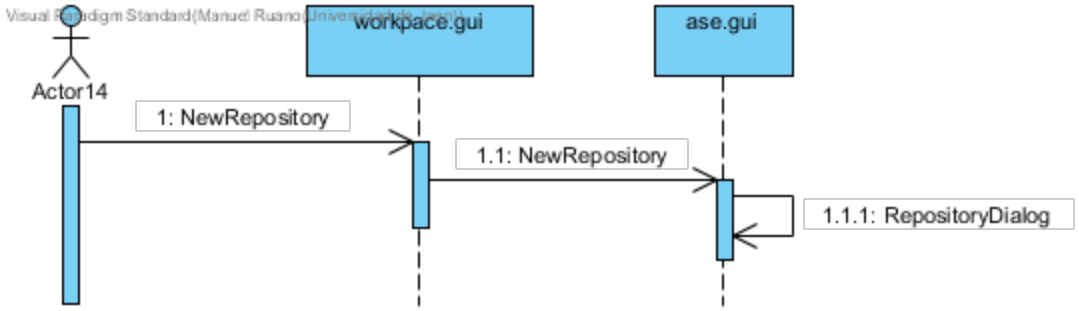
11.2.18 Visualizar Simulación



196-Diagrama de secuencia visualizar simulación

Cuando el usuario accede a la perspectiva de simulación el Application.e4xmi del plugin rcp es el encargado de llamar a las clases ya que esta perspectiva esta desarrollada haciendo uso de la interfaz que dispone Eclipse E4 para la visualización. En este caso se llama a cada una de las clases de la imagen ejecutando la función createControls de cada una de las clases.

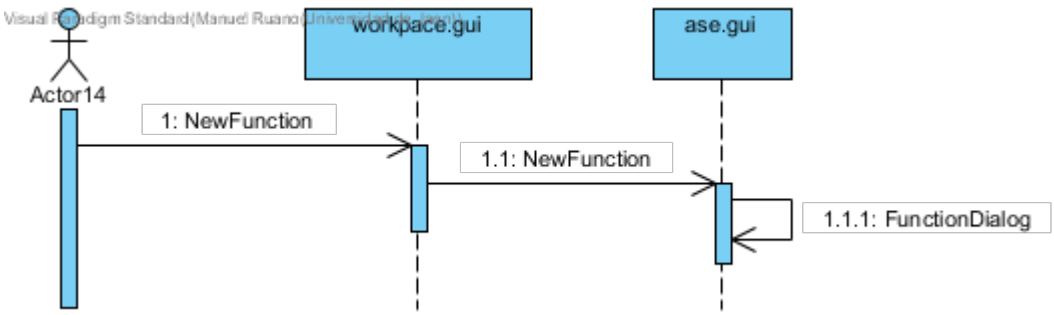
11.2.19 Nuevo Repositorio



197-Diagrama de secuencia nuevo repositorio

Cuando el usuario pulsa sobre Nuevo Repositorio se ejecuta el comando NewRepository del plugin workspace.gui que se encarga de llamar a la case NewRepository del plugin ase.gui. Ésta se encarga de crear un dialog donde el usuario insertará los datos.

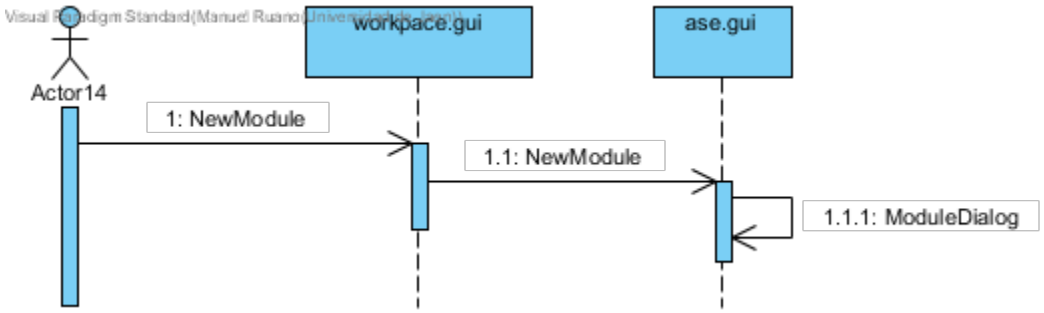
11.2.20 Nueva Función



198-Diagrama de secuencia nueva función

Cuando el usuario pulsa sobre nueva función se ejecuta el comando NewFunction del plugin workspace.gui que se encarga de llamar a la case NewFunction del plugin ase.gui. Ésta se encarga de crear un dialog donde el usuario insertará los datos.

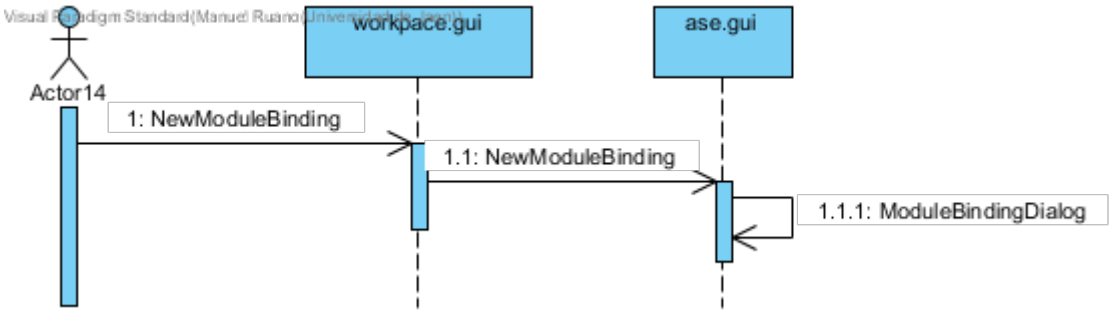
11.2.21 Nuevo Módulo



199-Diagrama de secuencia nuevo módulo

Cuando el usuario pulsa sobre nuevo módulo se ejecuta el comando NewModule del plugin workspace.gui que se encarga de llamar a la case NewModule del plugin ase.gui. Ésta se encarga de crear un dialog donde el usuario insertará los datos.

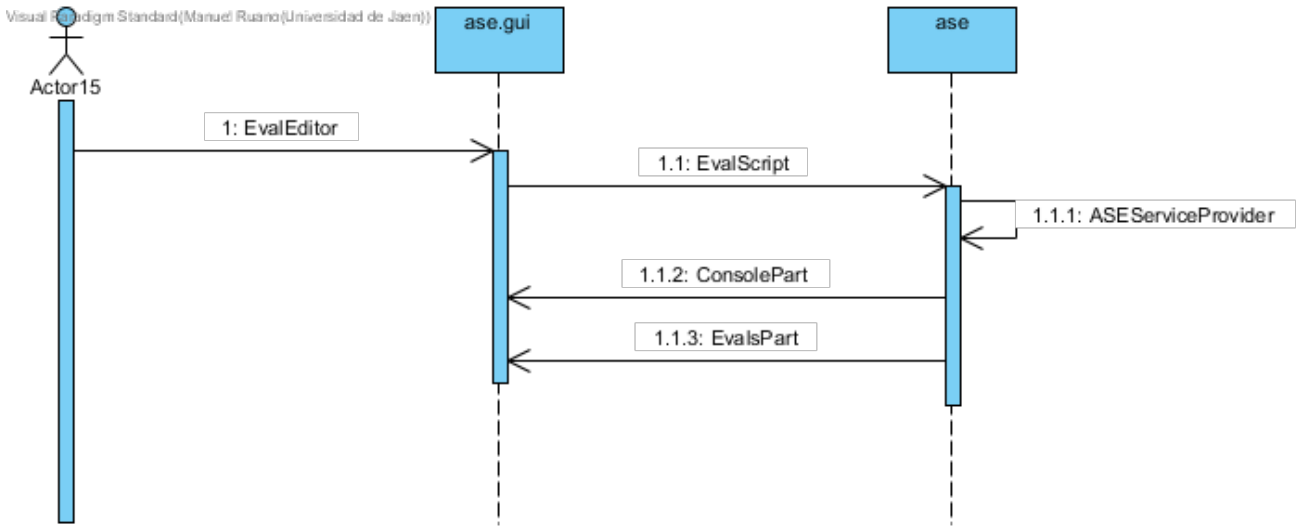
11.2.22 Nuevo Enlace a Módulo



200-Diagrama de secuencia nuevo enlace a módulo

Cuando el usuario pulsa sobre nuevo enlace a módulo se ejecuta el comando NewModuleBinding del plugin workspace.gui que se encarga de llamar a la case NewModuleBinding del plugin ase.gui. Ésta se encarga de crear un dialog donde el usuario insertará los datos.

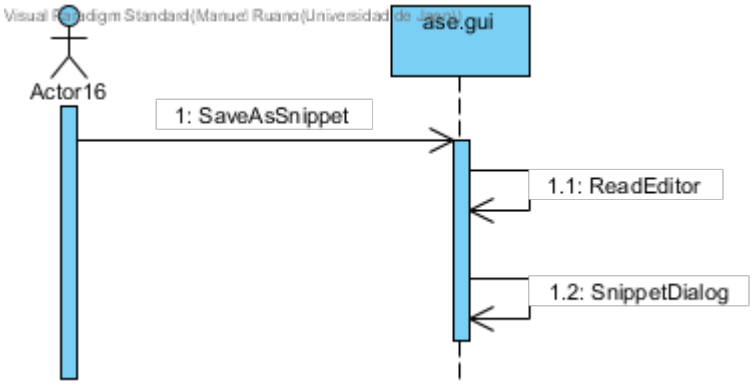
11.2.23 Evaluar Editor



201-Diagrama de secuencia evaluar editor

Cuando el usuario pulsa sobre Evaluar Editor el comando EvalEditor es lanzado, que este a su vez llama a la clase EvalScript que se encarga de llamar al servicio ASEServiceProvider que genera un RunnableScript. El servicio lanza unos eventos que llaman a las clases de ConsolePart y EvalsPart actualizando su visualización, lo que hace que se muestre por consola el código lanzado.

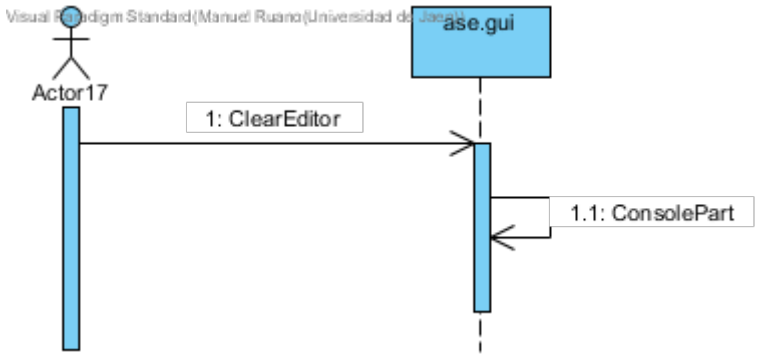
11.2.24 Guardar como Snippet



202-Diagrama de secuencia guardar como Snippet

Cuando el usuario pulsa sobre el botón Guardar como snippet el comando SaveAsSnippet es ejecutado, este se encarga de leer el editor mediante la clase ReadEditor que crea un SnippetDialog con la información que hay en la consola.

11.2.25 Limpiar Editor

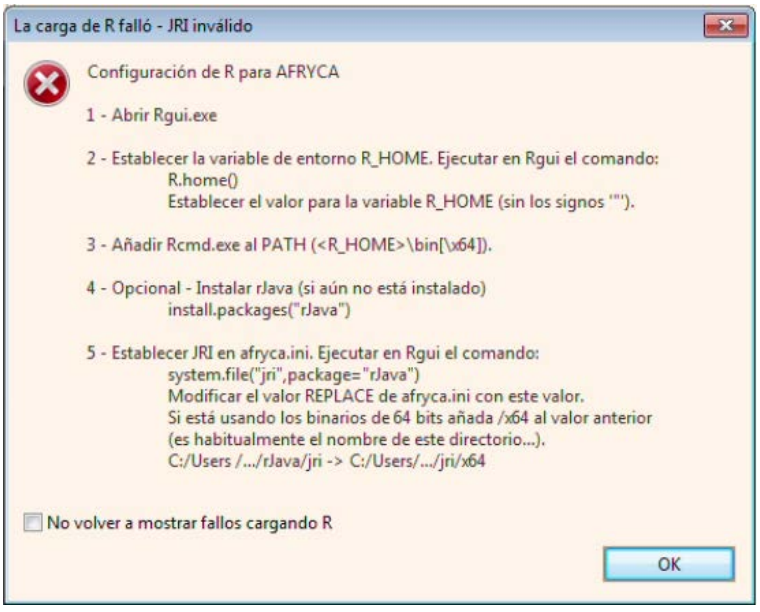


203-Diagrama de secuencia limpiar editor

Cuando el usuario pulsa sobre Limpiar Editor el comando ClearEditor del plugin ase.gui es lanzado, éste llama a la clase ConsolePart que se encarga de eliminar el contenido.

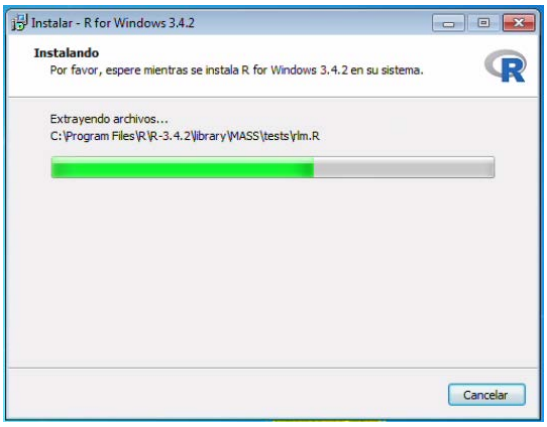
12. Instalación y configuración de R

Si al ejecutar afryca nos aparece el siguiente mensaje, quiere decir que no tenemos instalado R en nuestro equipo.



204-JRI Inválido

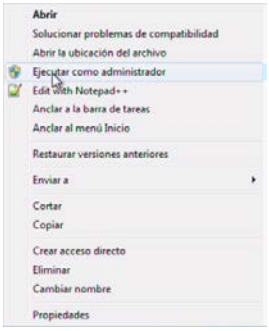
1) Descargamos su instalador desde la web: http://cran.r-project.org , obtenemos la versión para Windows y ejecutamos su instalación siguiendo las pociones por defecto.



205-Instalación R

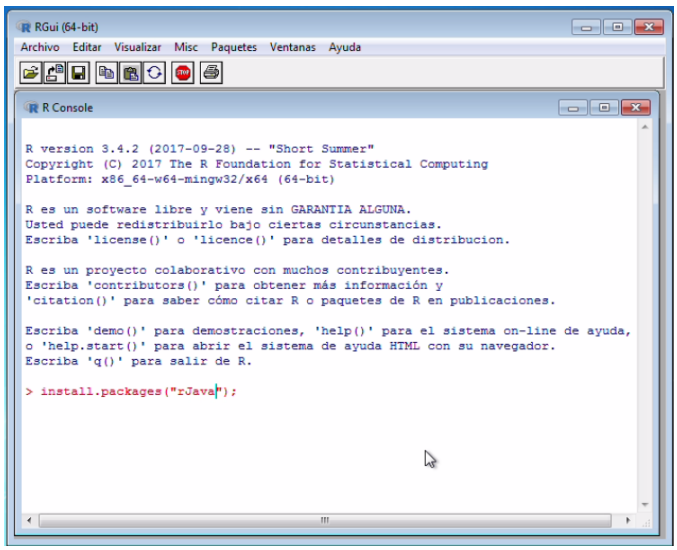
2) Instalamos el paquete rJava para R, lo primero será ejecutar la consola de R con permisos de administrador.

206-Consola R



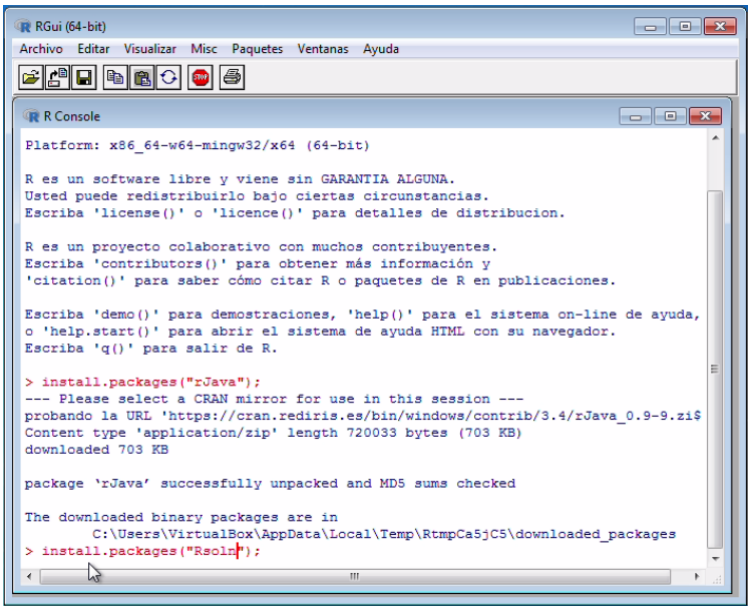
207-Ejecutar como administrador

Para instalar el paquete rJava ejecutamos el siguiente comando en consola: **install.packages("rJava");**



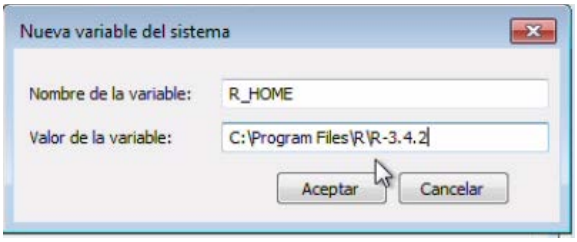
208-Paquete rJava

3) Instalamos el paquete Rsolnp con el comando: **install.packages("Rsolnp");**



209-Paquete Rsolnp

4) Creamos la variable de entorno R_HOME cuya ruta será la carpeta de instalación de R

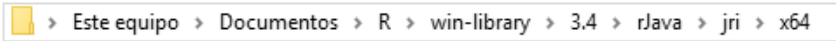


210-Variable R_HOME

5) Editamos la variable de entorno PATH con la ruta del paquete de rJava instalador anteriormente.

211-Ruta rJava

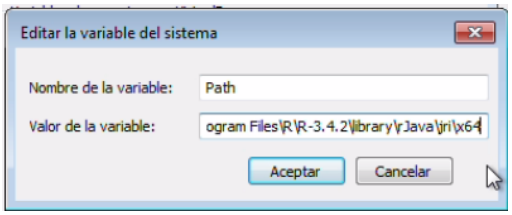
En algunas ocasiones esta ruta puede variar y las librerías de R se instalan en la carpeta de documentos del usuario.



212Ruta rJAva Documentos

Hay que destacar que la ruta escogida es x64 ya que el sistema operativo en el que se está instalando cumple estos requisitos.

La variable PATH quedará algo así:

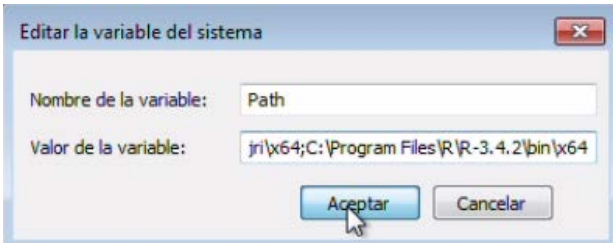


213-Variable PATH rJava

6) Añadir al PATH la ruta de lso binarios de R

214-Binarios R

La variable PATH quedará así:



215-PATH binario R

7) Modificamos el fichero afryca.ini reemplazando REPLACE

```
-startup
plugins/org.eclipse.equinox.launcher_1.3.201.v20161025-1711.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.401.v20161122-1740
-vmargs
-Declipse.log.level=ALL
-Dpython.console.encoding=UTF-8
-Djava.library.path=REPLACE
```

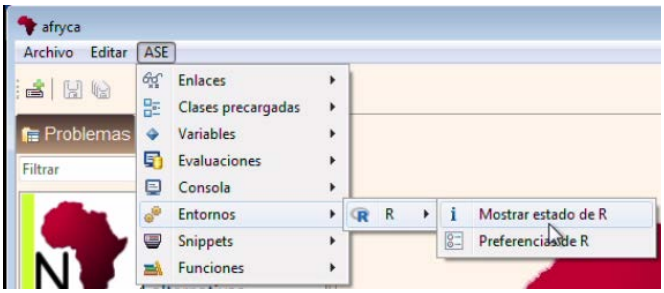
216-Afryca.ini REPLACE

Lo modificaremos por la ruta de rJava, quedando así:

```
-startup
plugins/org.eclipse.equinox.launcher_1.3.201.v20161025-1711.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.401.v20161122-1740
-vmargs
-Declipse.log.level=ALL
-Dpython.console.encoding=UTF-8
-Djava.library.path=C:\Program Files\R\R-3.4.2\library\rJava\jri\x64
```

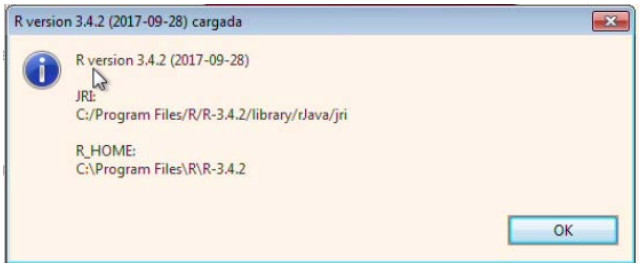
217-Afryca.ini rJava

Una vez hecho esto podemos ejecutar AFRYCA, para comprobar que R esta correctamente cargado podemos hacerlo desde:



218-Estado R

Si todo está correcto nos aparecerá el siguiente mensaje:



219-R cargado

En el caso de que R no este cargado nos mostrará el mensaje inicial, con lo que debemos comprobar que hemos seguido todos los pasos y volver a configurarlo.

13. Índice de imágenes