Universidad Carlos III de Madrid
Escuela Politécnica Superior
Ingeniería Superior en Informática

PLEX: Una herramienta para realizar experimentos en Planificación Automática

Proyecto Fin de Carrera

Autor: Samuel Bartolomé López
Tutor: Raquel Fuentetaja Pizán

5 de noviembre de 2010

Agradecimientos

Quiero agradecer a mis padres y a mi hermano el apoyo durante toda la carrera ya que siempre han estado cuando los he necesitado, dándome ayuda y consejo, y agradecerles que me hayan dejado la libertad de sacar la carrera a mi manera y con mis tiempos.

Quiero agradecer a Raquel toda su entrega, disponibilidad, ayudas, consejos y amabilidad durante todo el proyecto y agradecerle la libertad que me ha dado a la hora de realizarlo. Me gustaría también dar las gracias a su futuro bebé, ya que su inminente llegada al mundo ha conseguido que termine el proyecto. Os deseo todo lo mejor.

Quiero agradecer a mi hermano el que siempre me haya ido abriendo puertas y allanado el camino haciendo que todo sea mas sencillo para mi.

Quiero agradecer a Sori que haya cambiado mi vida. Agradecerla que durante todos los años de la universidad me haya motivado, apoyado y ayudado estando siempre a mi lado. Y agradecerla el que cada día me haga un poco más feliz.

Quiero agradecer a mis amigos Buba, Chivo, Fur, Gordo, Kojo y Kurras el que no me hayan faltado ni una sola vez desde que los conozco. Sin vosotros nunca hubiese sido el "amigo informático".

Quiero agradecer a Jess, Maru, Patri, Xana y Sandrich todas las noches de parranda, todas las tardes de cañas y su imprescindible compañía.

Quiero agradecer a Vidal todos los ratos que hemos pasado dentro y fuera de la universidad, todas las experiencias que hemos vivido y las que vendrán.

Quiero agradecer a Adri, a Antonio, a Jaime y a Juan todas las mañanas y las tardes pasadas en las aulas hablando, jugando, apoyándonos y "cagando fuego" una práctica tras otra.

Quiero agradecer a Carlos, David y a Isra todas las partiditas de mus y todos y cada uno de los buenos momentos que hemos pasado juntos.

Quiero agradecer a todos los miembros de la ATL el buen rollo de cada uno de los partidos de baloncesto jugados (con sus previos y postpartidos) y el haberme llevado a lo más alto en mi carrera "baloncestística".

Por último dar las gracias a todos los que han formado parte de mi vida universitaria, cada granito que han aportado me han hecho llegar hasta aquí.

Os quiero ☺

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este documento se explica el desarrollo de la herramienta PLEX (PLanning EXperimenter). El contexto de esta herramienta es la Planificación Automática. PLEX proporciona facilidades para plantear y realizar experimentos con la finalidad de comparar el comportamiento de distintos planificadores. Además, permite analizar los resultados de los experimentos proporcionando distintos tipos de gráficas y tablas. Una de las características principales de PLEX es la flexibilidad, puesto que su diseño está pensado para que se puedan incorporar nuevos planificadores fácilmente.

El documento empieza con el capítulo 2 dónde se explican los motivos que justifican el desarrollo de una herramienta como PLEX. A continuación, en el capítulo 3, se detallan los objetivos planteados para el Proyecto Fin de Carrera.

Después de estos dos putos se realiza una breve introducción al mundo de la Planificación Automática. Así, en el capítulo 4, se describen los planificadores más importantes y las técnicas que aplican. A continuación, se estudian las herramientas existentes, orientadas a facilitar el uso y la comprensión de distintos planificadores desde el punto de vista del usuario.

El documento continua con el capítulo 5 en el que se detallan las principales características de la herramienta. Con este capítulo se pretende dar al lector una visión general de lo que se ha desarrollado. El desarrollo técnico de la herramienta PLEX se detalla en el capítulo 6. En este capítulo se describen los requisitos, el diseño y el funcionamiento de la aplicación. Finalmente, se incluyen las conclusiones (capítulo 7) y los trabajos futuros (capítulo 8).

El documento concluye con dos apéndices. El apéndice A es un pequeño manual de usuario, mientras que el apéndice B detalla la gestión del proyecto.

Capítulo 2

Motivación

En este capítulo se explican los motivos por los que se cree necesario el diseño e implementación de la herramienta PLEX. Estos motivos son:

- Hoy en día la creación de experimentos en planificación automática se puede convertir en una tarea que requiere demasiado tiempo. Tiempo que podría ser utilizado en el análisis de los resultados de esos experimentos en vez de en su creación. Principalmente esto ocurre porque la salida de los planificadores no está estandarizada y eso produce dificultades a la hora de obtener los resultados.
- Como se explica en el estado de la cuestión (sección 4.4), para obtener la salida de los diferentes planificadores de forma automática se necesita crear código ad-hoc para cada planificador. Este código es difícil de mantener ya que los planificadores pueden cambiar y hay que crear nuevos códigos si aparecen nuevos planificadores. Además este código se ha de modificar dependiendo de lo que queramos obtener de cada planificador.
- El código ad-hoc de cada planificador no nos muestra una visión global del experimento, por lo que si necesitamos datos globales como puede ser el tiempo total de un planificador, necesitamos obtener esos datos a mano o con otros códigos.
- La herramienta ItSimple explicada en el estado de la cuestión (sección 4.3.2), nos permite crear dominios y problemas, a la vez que probarlos con diferentes planificadores. Pero a la hora de crear experimentos no nos permite introducir de forma sencilla nuevos planificadores, así como seleccionar qué datos de la solución queremos obtener. Tampoco permite un trabajo posterior con los datos obtenidos.
- La herramienta PLTooL cuya funcionalidad se refleja en el estado de la cuestión (sección 4.3.1) sí que permite la configuración de experimentos, pero sólo con una serie de planificadores ya incluidos en la aplicación no permitiendo el uso de otros. Además no permite la selección de las variables a estudiar, ni permite la creación de gráficas.
- Debido a que en la actualidad no existe una herramienta que trate de forma genérica a los planificadores, tanto en la entrada como en la salida para la obtención de los datos que el usuario requiera y así poder hacer un estudio de esto a través de tablas y gráficas, se puede afirmar que el diseño e implementación de una herramienta que permita realizar estas acciones está suficientemente motivado.
- Se puede afirmar que el diseño e implementación de de una herramienta que permita tratar de forma genérica a los planificadores, tanto en la entrada como en la salida, obteniendo los datos que el usuario requiera y así poder hacer un estudio de esto a través de tablas y gráficas, acciones está suficientemente motivado

Capítulo 3

Objetivos

El objetivo principal que se ha planteado para este proyecto fin de carrera es la construcción de la herramienta PLEX (PLanning EXperimenter), cuya finalidad es facilitar la realización de experimentos para comparar planificadores en el ámbito de la Planificación Automática. Para ello, la herramienta ofrecerá un interfaz sencillo y amigable, y servirá para automatizar tareas para las que usualmente los investigadores utilizan sus propios *scripts*.

El objetivo principal se puede dividir en los siguientes objetivos específicos:

1. Realizar el diseño e implementación de la herramienta que, después del estudio del estado de la cuestión y para cubrir las carencias de las herramientas existentes, tendrá la siguientes características:

   (.1) Deberá aceptar cualquier archivo de dominio o problema.

   (.2) Deberá aceptar cualquier planificador independientemente de la entrada que requiera y de la salida que genere. La herramienta no debe preocuparse de si el dominio y problemas son compatibles con el planificador. Este aspecto quedará bajo responsabilidad del usuario.

   (.3) Será capaz de procesar la salida de los planificadores de una forma semi-automática y sencilla, eliminado la necesidad de realizar programas *ad-hoc* individuales. Esto se hará definiendo un método general para facilitar esta tarea al usuario. Este método general permitirá elegir las variables del experimento que se quieren examinar.

   (.4) Será capaz de lanzar los experimentos que se programen y ofrecerá los resultados de su ejecución tanto a través de tablas, como de gráficas. El número de dominios, problemas y planificadores no estará limitado.

   (.5) Las gráficas se construirán *online*, según se obtiene el resultado de la ejecución de cada planificador y la herramienta ofrecerá los tipos de gráficas que usualmente se utilizan en los experimentos de Planificación Automática.

   (.6) La herramienta ofrecerá datos relativos al análisis global de resultados como los utilizados en las últimas competiciones de Planificación Automática. Además, el usuario ha de poder crear diferentes tipos de operaciones con los datos en tiempo de ejecución a través de código JAVA 1.6. Las operaciones más comunes estarán preprogramadas.

   (.7) Los experimentos se guardarán en formato XML. Así se permitirá al usuario importar experimentos antiguos a través de la creación de archivos de experimento XML con el formato de los archivos XML que se definirá para la herramienta.

   (.8) La herramienta podrá recuperar experimentos que se han interrumpido. Esto permitirá controlar la pérdida de datos producida por cualquier error externo a la herramienta durante su ejecución.

   (.9) La herramienta debe permitir el cambio de idioma de una forma sencilla, así como la creación de nuevos idiomas.

2. Documentar el trabajo realizado, incluyendo:

   (.1) Un resumen de las características principales de la herramienta, sus requisitos y diseño.

   (.2) Manual de usuario que especifique como usar toda la funcionalidad de la aplicación.

   (.3) Puesto que la herramienta se programará en JAVA, se generará el correspondiente *javadoc* de todo el código JAVA implementado.

Capítulo 4

Estado de la cuestión

En este apartado se realizará una introducción a la Planificación Automática explicando sus orígenes, su utilidad y los principales algoritmos que se aplican. Además, se comentarán las principales características de algunos de los más importantes planificadores y su funcionamiento de forma breve para lograr un acercamiento a estos y así poder comprender mejor la finalidad de la herramienta PLEX.

4.1. Planificación automática

La Planificación Automática es una rama de la Inteligencia Artificial que se refiere a la generación automática de planes compuestos por secuencias de acciones, normalmente para ser ejecutadas por agentes inteligentes, robots y vehículos autónomos no tripulados. Este área de la Inteligencia Artificial surgió hacia 1971, fecha en la que nace el planificador STRIPS (Fikes and Nilsson, 1971). STRIPS fue creado con la finalidad de controlar el robot Shakey y conseguía que éste resolviese distintos problemas.

Un planificador típicamente toma tres entradas: una descripción del estado inicial del mundo, un conjunto de acciones posibles y una descripción de las metas deseadas. Las acciones constituyen el dominio de planificación, mientras que la descripción compone el estado inicial y las metas forman la parte del problema de planificación a resolver. Tanto dominio como problema se codifican en un lenguaje formal. Un planificador produce una secuencia de acciones que conducen desde el estado inicial a un estado en el que todas las metas se han conseguido.

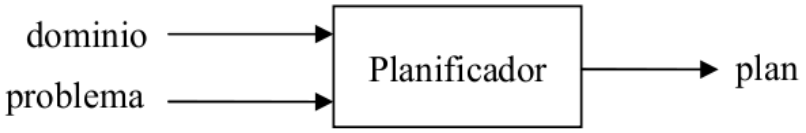

Figura 4.1: Proceso de planificación

La especificación de los problemas y del dominio se estandarizó en un lenguaje conocido como PDDL (*Planning Domain Definition Language)* (Fox and Long, 2003), gracias a la IPC (Competición Internacional de Planificación) de la conferencia ICAPS (*International Conference on Automated Planning and Scheduling)* [1], que viene repitiéndose desde 1998.

Aunque las entradas de los planificadores se han estandarizado, no ocurre así con las salidas. Es decir, con la forma en que expresan los planes obtenidos y ciertos detalles de la ejecución. Esta falta de estandarización provoca que para procesar la salida haya que realizar programas *ad-hoc*.

[1] `http://www.icaps-conference.org/index.php/Main/Competitions`

Aunque existen planificadores dependientes del dominio, en el mundo de la planificación se ha realizado un gran esfuerzo para conseguir planificadores independientes del dominio. Un planificador independiente del dominio cuenta con algoritmos generales que sirven para planificar en cualquier problema de cualquier dominio. Existen distintos tipos de técnicas para construir un planificador independiente del dominio. Una de las más relevantes es la búsqueda heurística. Otras son la planificación de orden parcial, los grafos de planificación (Blum and Furst, 1995), la planificación SAT, etc.

Los planificadores han ido poco a poco mejorando hasta llegar a los que podemos considerar de alto rendimiento, capaces de resolver problemas más complejos con resultados muy satisfactorios como pueden ser: LPG (Gerevini et al., 2004), SGPLAN (Chen et al., 2006) o METRIC-FF (Hoffmann, 2003).

Hoy en día se sigue investigando en Planificación Automática, creando nuevos planificadores o modificando los antiguos con el objetivo de mejorar los planes que obtienen y/o el tiempo que se tarda en conseguirlos. Gracias a esto, hoy en día se puede plantear utilizar un planificador automático en problemas del mundo real como en los videojuegos, en las empresas al obtener las secuencias de uso de recursos más óptimas, en los mapas ayudando a la obtención de rutas, etc...

4.2. Planificadores

En este punto se van a describir las principales características de varios planificadores, para acercar al lector un poco más a la idea de lo que es un planificador.

4.2.1. FF

FF (Hoffmann and Nebel, 2001) es un planificador totalmente independiente del dominio que hace búsqueda heurística. Fue desarrollado por Jörg Hoffmann. Es un sistema implementado íntegramente en C y participó en la tercera IPC mostrando un comportamiento muy competitivo.

Metric-FF es el sucesor de FF. Tiene sus mismas características pero permite utilizar variables de estado numéricas.

4.2.2. SAYPHI

SAYPHI se trata de un planificador desarrollado por Tomás de la Rosa para la investigación en planificación heurística y aprendizaje automático. Su nombre se obtiene de las siglas de Sistema de Aprendizaje Y Planificación Heurística. Es un planificador similar a FF y desarrollado en el lenguaje de programación LISP.

SAYPHI se puede considerar un sistema de planificación y aprendizaje que trata de integrar diferentes métodos para obtener un conocimiento de control que guíe el proceso de búsqueda de un planificador heurístico común. Este planificador trabaja con dominios y problemas especificados en el lenguaje PDDL.

SAYPHI puede utilizar distintos tipos de heurísticas y utilizar distintos algoritmos de búsqueda, como búsqueda en escalada, búsqueda *Enforced Hill-Climbing* y búsqueda A*.

4.2.3. SGPlan

SGPLAN (Chen et al., 2006) desarrollado por Chih-Wei Hsu, Benjamin W.Wah y Yixin Chen. Este planificado quedó en primer lugar en la IPC de 2006.

Este planificador divide en subproblemas el problema a resolver. Después resuelve los subproblemas y a partir de las soluciones parciales construye la solución global del problema.

Utiliza como base el planificador Metric-FF y desarrolla técnicas para que consiga resolver las restricciones, optimizar las preferencias de los objetivos y alcanzar submetas en una representación multivalor.

4.2.4. LPG

LPG (Gerevini et al., 2004) fue desarrollado por Alfonso Gerevini, Alessandro Saetti e Ivan Serina, su nombre se obtiene de Local Search in Planning Graphs. LPG es un planificador que participó en la IPC en 2004. Se trata de un planificador estocástico.

El espacio de búsqueda de LPG consiste en una serie de planes parciales (subgrafos del grafo de planificación). En cada paso de la búsqueda, LPG aplica ciertas modificaciones para transformar un plan parcial en otro. La evaluación de estos planes se lleva a cabo mediante una función parametrizada que hace uso de diversas heurísticas.

LPG puede producir planes de buena calidad mediante un proceso que produce una secuencia de planes, en el que cada plan es de mejor calidad que el anterior. LPG también integra un algoritmo de búsqueda mejor primero, similar al empleado por FF. Aparte de los buenos resultados ofrecidos por LPG, también destaca su versatilidad, ya que permite manejar los tres primeros niveles de PDDL 2.1, así como acciones con costes y duraciones asociadas.

4.3. Herramientas en planificación automática

En esta sección se van a explicar algunas de las herramientas que se usan hoy en día en la planificación automática. Así pues se estudiaran las aplicaciones PLTooL (Fernánndez et al., 2007) e ItSimple (Vaquero et al., 2007), explicando su funcionalidad y su capacidad de realizar experimentos de planificación.

4.3.1. PLTooL Versión 2.0

En este punto se va a describir el funcionamiento de la herramienta PLTooL desarrollada por el Grupo de Planificación y Aprendizaje (PLG) de la Universidad Carlos III de Madrid. Esta herramienta esta compuesta por diferentes módulos que se explicarán a continuación.

4.3.1.1. Módulo de planificación

Este módulo se compone de la interfaz mostrada en la figura 4.2, en ella se puede seleccionar un planificador, ya sea IPSS, SAYPHI, LPG, SGPLAN o METRIC-FF. Una vez elegido uno la interfaz se adapta para mostrar los parámetros que aceptan cada uno de ellos. Después, se puede seleccionar un dominio, un problema y opcionalmente un fichero de reglas de control.

Como podemos observar en la figura 4.3, la aplicación permite ejecutar el planificador y observar el plan que se ha obtenido si ha conseguido encontrar solución. También nos muestra el tiempo total, el número de nodos y el coste de la solución.

En el caso de que el planificador elegido fuese IPSS nos permite consultar cual ha sido el árbol de búsqueda construido para llegar a la solución del problema como se muestra en la figura 4.4.

4.3.1.2. Módulo de aprendizaje de reglas de control

El módulo de aprendizaje se basa en el sistema Hamlet que permite el aprendizaje de reglas de control para IPSS. La reglas de control se usan para guiar la búsqueda que lleva a cabo el planificador. Estas reglas sirven para recomendar el operador a elegir en un momento dado del proceso de búsqueda. En la figura 4.5 podemos observar la interfaz destinada a este propósito. En ella podemos seleccionar un dominio, y un conjunto de problemas de entrenamiento. Además se ha de introducir el fichero de reglas de control. Por último, permite ajustar ciertos parámetros del planificador, así como medidas para optimizar el aprendizaje de las reglas.

4.3.1.3. Módulo de aprendizaje de macro-operadores

En este módulo se permite el aprendizaje de macro-operadores partiendo de planes generados anteriormente con el planificador IPSS. Un macro-operador es una secuencia de operadores que pueden tratarse en conjunto como un único operador.

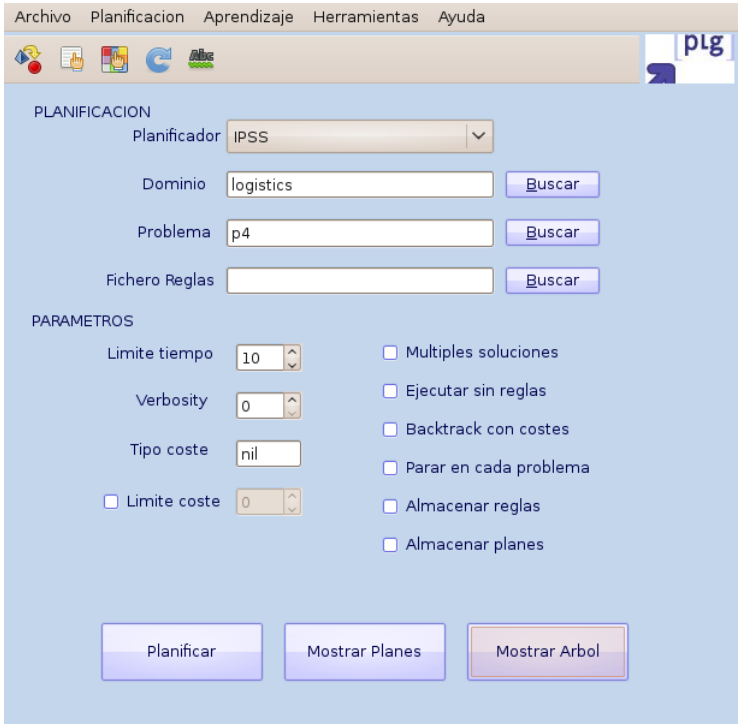

Figura 4.2: Módulo de planificación

La interfaz que permite la ejecución de este módulo se puede observar en la figura 4.6. En ella se puede introducir dónde comienza y finaliza el macro-operador y si se quiere que se instale en el dominio y/o que se muestre por pantalla.

4.3.1.4. Módulo de traducción

La herramienta PLTooL permite la traducción de PDDL 2.1 a IPSS y viceversa. Así pues la interfaz de este módulo, que se muestra en la figura 4.7, aparece dividida en dos partes. La primera esta dedicada a la traducción de PDDL a IPSS, permitiendo la traducción de un dominio o de un dominio y varios problemas; y la segunda se destina a la traducción de IPSS a PDDL permitiendo la traducción de un dominio y un conjunto de problemas.

4.3.1.5. Módulo de experimentación

Este módulo permite realizar experimentos con múltiples dominios y problemas combinándolos con múltiples planificadores. La herramienta permite la inclusión de varios dominios con sus respectivos problemas y una serie de planificadores para después ejecutarlos y recoger las soluciones.

En la interfaz gráfica mostrada en la figura 4.8 podemos crear, guardar o abrir un nuevo experimento, así como comenzar la ejecución de un experimento, especificar el modo de ejecución y el número de veces que se quiere repetir dicho experimento. En el modo de ejecución se puede especificar si se ejecuta el experimento por dominio o por planificador y si se quiere o no almacenar los planes.

Se pueden introducir o borrar nuevos dominios y planificadores gracias a los botones que aparecen en cada sección de la interfaz. Al añadir un nuevo dominio se muestra una nueva ventana que podemos observar en la figura 4.9 en la que podemos introducir el dominio, los problemas y las reglas de control. Cuando queremos añadir un nuevo planificador nos aparece una ventana en la que elegimos qué planificador usar y nos permite configurar los parámetros que cada uno de éstos puede usar. Esta ventana se puede observar en la figura 4.10

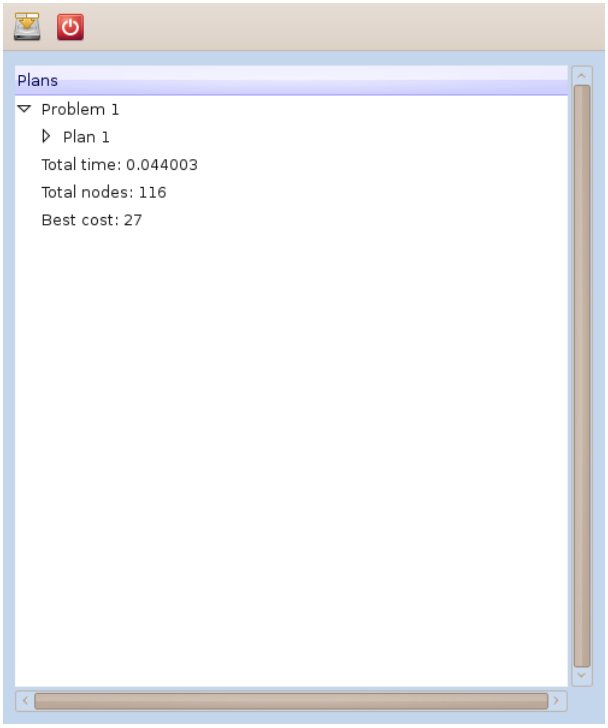

Figura 4.3: Plan obtenido después de la ejecución del planificador

Para obtener los resultados del experimento hay que buscar en la carpeta "pltool/experimenter" el fichero generado por la herramienta PLTooL. Cada vez que el usuario ejecuta un experimento se crea una carpeta cuyo nombre será especificado por: <nombre del experimento>-<día>-<mes>-<año>-<hora>-<minutos>-<segundos>. En la carpeta se crearán archivos con nombre: <resultados>-<nombre del experimento>.cvs. En el que se guardaran los resultados del experimento separados por <dominio-problema>-<planificador>. Los valores que se guardan para cada problema de un dominio ejecutado por un planificador son:

1. Path completo del dominio.
2. Path completo del problema.
3. Nombre que el usuario ha dado al planificador en la configuración del experimento.
4. Parámetros del planificador escogidos.
5. Tiempo que el planificador a utilizado para encontrar solución para el problema actual.
6. Tiempo acumulado por el planificador en todo el experimento.
7. Nodos totales expandidos por el planificador para el problema actual.
8. Nodos acumulados por el planificador en todo el experimento.
9. Coste del plan obtenido para el problema actual.
10. Coste acumulado de todos los planes obtenidos por el planificador en todo el experimento.
11. Longitud del plan obtenido para el problema actual.
12. Longitud acumulada de todos los planes obtenidos por el planificador en todo el experimento.

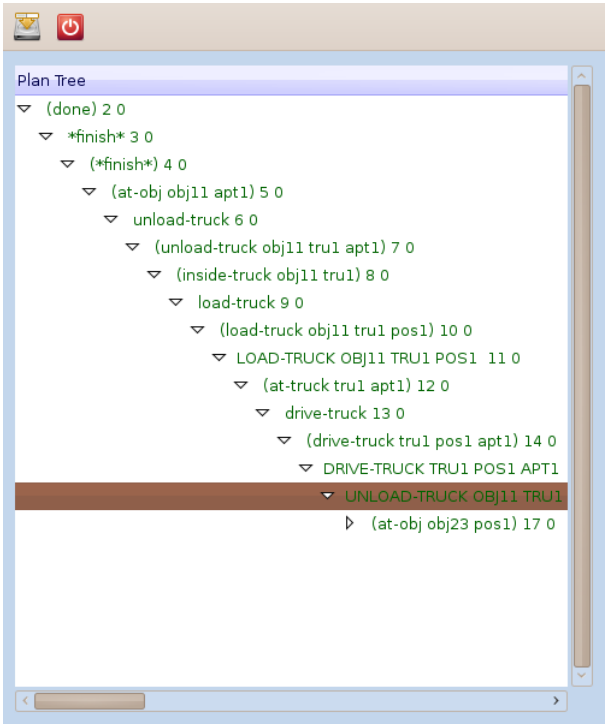

Figura 4.4: Árbol de búsqueda obtenido con el planificador IPSS

13. Variable que indica si se ha encontrado solución o no el experimento.

14. Número de problemas que ha resuelto el planificador hasta el momento.

4.3.2. ItSimple

En esta sección se explica la herramienta de planificación ItSIMPLE [2]. Se va a estudiar que es capaz de hacer y si puede realizar experimentos con diferentes dominios y planificadores para poder realizar estudios sobre éstos.

Esta herramienta esta creada para dar soporte a los usuarios durante la construcción de un dominio de planificación y sus problemas, principalmente en las fases iniciales de la etapa de diseño. Estos estados iniciales abarcan la especificación del dominio, el modelado, el análisis, el testeado y el mantenimiento.

Para la especificación y el modelado, la herramienta ItSimple usa UML (Unified Modeling Language), XML (eXtensible Markup Language), redes Petri y PDDL (Planning Domain Definition Language). Así pues la herramienta es capaz de transformar los modelos UML en representaciones PDDL para que los usuarios puedan usar sus modelos con distintos planificadores (Metric-FF, FF, SGPlan, MIPS-xxl, LPG-td, LPG, hspsp, SATPlan y Plan-A). El lenguaje XML se utiliza para la traducción de los modelos UML a representaciones como PDDL o redes Petri.

El entorno también incluye una interfaz para el análisis y la administración de los planes, que permite a los diseñadores observar el comportamiento del modelo durante la simulación de éstos. A través de diferentes imágenes el usuario puede observar como va cambiando el dominio según se ejecuta cada paso del plan.

En la figura 4.11 podemos ver cómo se especifica a través de un diagrama UML de casos de uso las diferentes acciones que puede realizar un camión y un avión.

[2] `http://dlab.poli.usp.br/twiki/bin/view/ItSIMPLE/OverView`

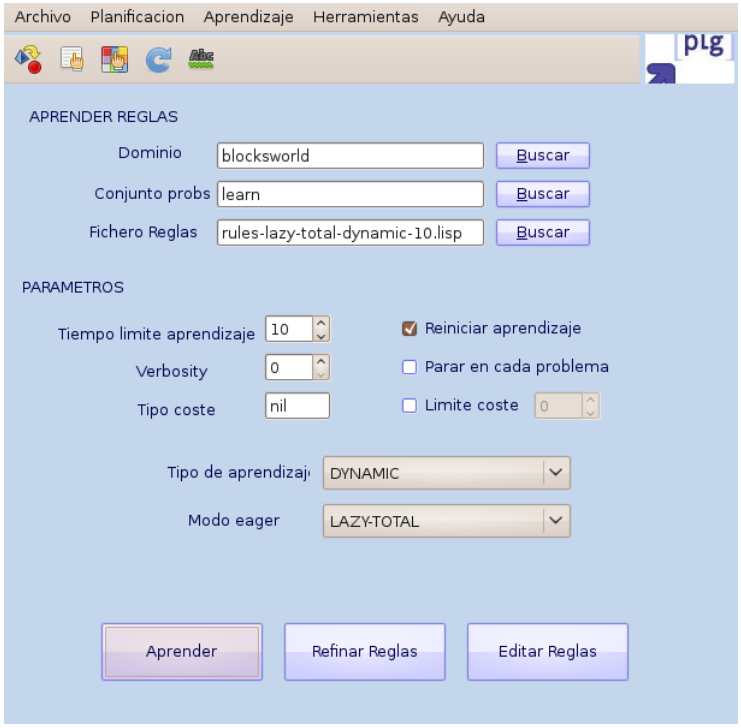

Figura 4.5: Módulo de aprendizaje de reglas de control

En la figura 4.12 se aprecia cómo se especifican las diferentes piezas del dominio y cómo se relacionan gracias a un diagrama UML de clases.

Por último, en la figura 4.13, se refleja cómo, a través de un diagrama UML de estados, se añade el cambio realizado por el estado de un "paquete" bajo diferentes acciones ("cargar un camión", "cargar un avión", "descargar un camión" o "descargar un avión").

Después de crear el dominio, ItSIMPLE permite probarlo a través de una serie de planificadores incorporados. Estos planificadores son:

1. **Metric-FF (windows y linux).**
2. **FF (v2.3) (linux).**
3. **SGPlan 5.2.2 (linux).**
4. **SGPlan 6 (linux).**
5. **MIPS-xxl 2006 (linux).**
6. **MIPS-xxl 2008 v1.1 (linux).**
7. **LPG-td 1.0 (windows y linux).**
8. **LPG 1.2 (linux).**
9. **hspsp 2008 (linux).**
10. **Plan-A 1.0 (linux).**

La herramienta permite la inclusión de nuevos planificadores, introduciendo el ejecutable de éste en una carpeta determinada y modificando un archivo XML en el cual se introducen las características del

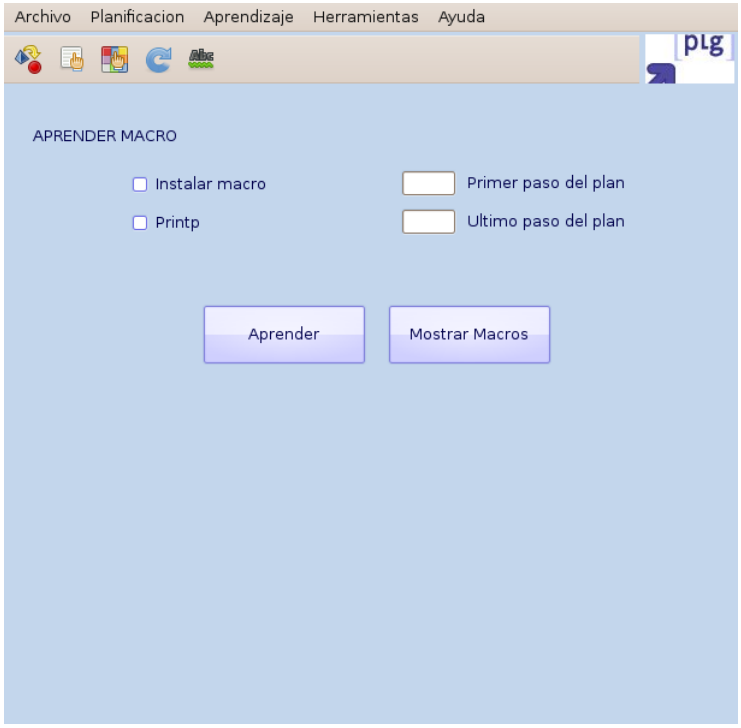

Figura 4.6: Módulo de aprendizaje de reglas de control

planificador, como pueden ser sus argumentos o dónde escribe la salida. También permite hacer peticiones al equipo creador de la aplicación para que se introduzcan planificadores de forma permanente.

Después de la ejecución del planificador la herramienta es capaz de dar el tiempo que ha tardado el planificador en obtener la solución, enumerar los pasos de los ésta que se compone y cargar en la aplicación el plan generado. La herramienta puede realizar esto gracias a que trabaja sobre un dominio creado dentro de ella, pudiendo detectar el plan en la salida del planificador porque es capaz de reconocer los elementos que aparecen en éste.

ItSimple permite la realización de gráficos con el tiempo y la cantidad de pasos de cada planificador y para cada problema, pero sólo permite la generación de las gráficas con estos dos datos. Por otro lado, una vez ejecutado el planificador, la herramienta nos muestra el plan resultante y permite estudiar a través de gráficas cómo las diferentes variables de los dominios cambian a medida que el plan se va ejecutando (figura 4.14).

Por último, destacar que la aplicación puede generar una película a través de diagramas UML dónde nos muestra cómo se va ejecutando el plan (figura 4.15).

Para concluir, diremos que ItSimple se trata de una aplicación orientada al diseño de dominios de forma gráfica a través de lenguaje UML. Nos permite diseñar las diferentes variables que determinen un "mundo" como puede ser el "mundo de los bloques", especificar problemas para los dominios creados y por último observar como se comporta el dominio tras la ejecución de los planes obtenidos por algún planificador. Aunque la herramienta sea capaz de generar gráficas con datos como el tiempo o el número de pasos del plan generado, no se puede decir que sea una herramienta completa para la experimentación con planificadores ya que no permite recoger más variables de los planificadores ni ejecutar varios dominios y planificadores al mismo tiempo. Se puede decir que la herramienta da la funcionalidad de ejecutar planificadores con la finalidad de que el usuario observe el comportamiento del dominio ante diversos planes, pero no está destinada a estudiar cómo se comportan los diferentes planificadores ante los dominios.

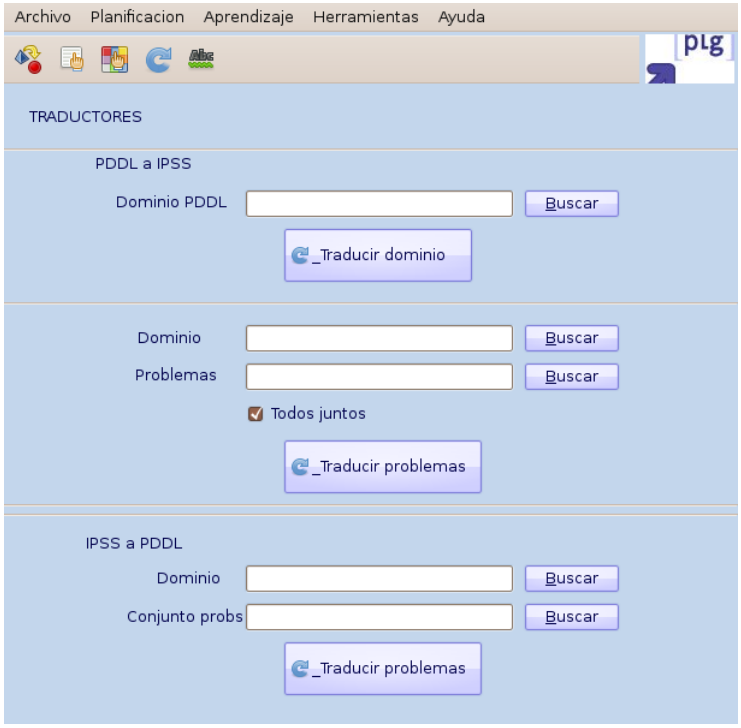

Figura 4.7: Módulo de traducción

4.4. Experimentos en planificación automática

En esta sección se va a explicar de qué manera se realizan experimentos en la planificación automática. Los experimentos en planificación automática se componen de uno o más dominios, con uno o más problemas que se ejecutan en uno o más planificadores, con diversos objetivos, como pueden ser la comparación de planificadores, estudiar el comportamiento de un dominio, o simplemente tratar de encontrar solución a un problema.

Lo primero que se realiza a la hora de crear un experimento es decidir qué dominio/s, problema/s y planificador/es van a formar parte de éste. Una vez escogidos, se ejecuta cada problema de cada dominio con cada planificador o cada planificador con cada problema. Para la ejecución de cada planificador es necesario especificar qué parámetros de éste se van a usar y cómo. La salida de los planificadores suele mostrar si se ha encontrado solución, y si es así el tiempo necesario para obtenerla, su coste, su tamaño y el plan obtenido.

En este punto se presenta un problema: la salida de los planificadores, en contraposición con los dominios o los problemas que tienen un lenguaje determinado, no está estandarizada por lo que cada planificador puede mostrar diferentes datos, en formatos y lugares diferentes. Esto es un problema debido a que para la realización de un experimento en la mayoría de los casos es necesario comparar los mismos valores, ya sea tiempo, coste, etc... de diferentes planificadores.

Actualmente para poder realizar esta acción la persona que realiza el experimento tiene que tratar cada salida de cada planificador de forma específica y así obtener los datos que necesita en cada momento. Habitualmente se usa código ad-hoc para realizar esta tarea, pero no es una solución muy eficaz ya que cada vez que aparece un nuevo planificador, se modifica la salida de uno ya existente o se necesitan obtener otros valores, es necesario rehacer todo este código.

Una vez se han obtenido los datos de un problema determinado para un planificador es posible que se necesite más información, como por ejemplo el tiempo o coste acumulado por un planificador en todo

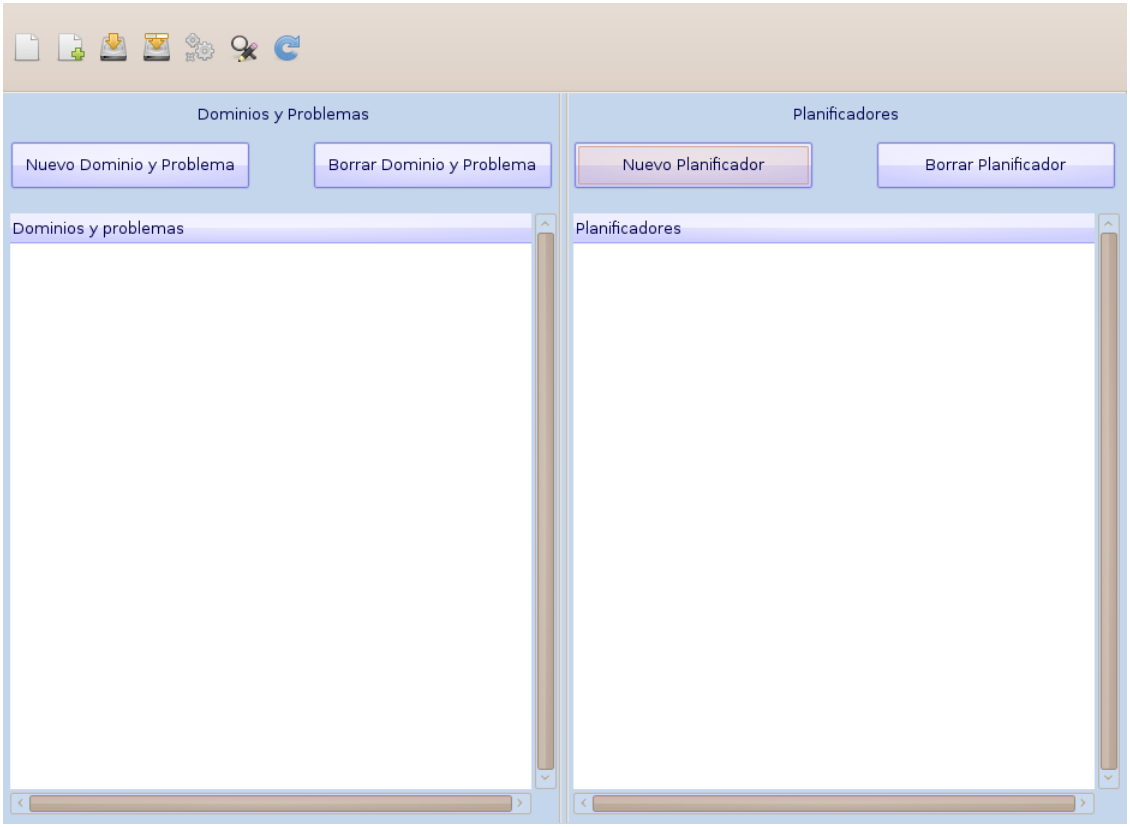

Figura 4.8: Módulo de experimentación

Figura 4.9: Nuevo dominio

el experimento. Si es necesario, esto se debe calcular obteniendo los datos de las salidas del planificador o de la salida obtenida del uso de los códigos ad-hoc.

Por último, el investigador que realiza el experimento puede necesitar crear gráficas y tablas para realizar comparaciones más visuales o para la publicación de los experimentos en diversos medios. Para ello ha de emplear alguna herramienta existente para la creación de gráficos como puede ser GNUplot para lo cual es necesario volver a formatear los datos de la manera correcta para que estas aplicaciones los acepten.

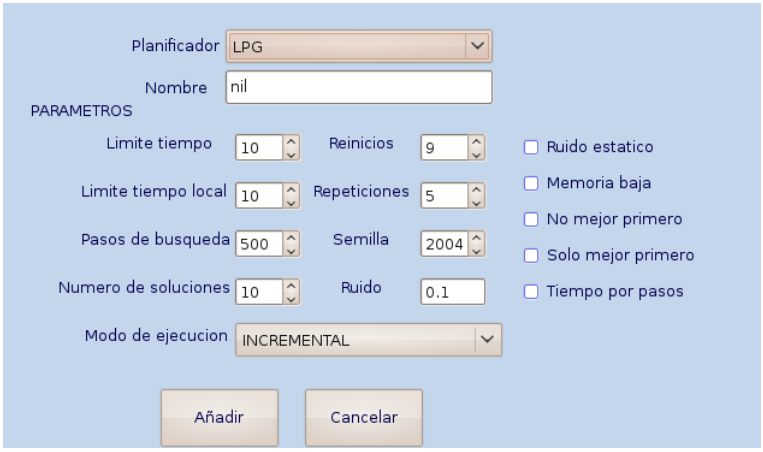

Figura 4.10: Nuevo planificador

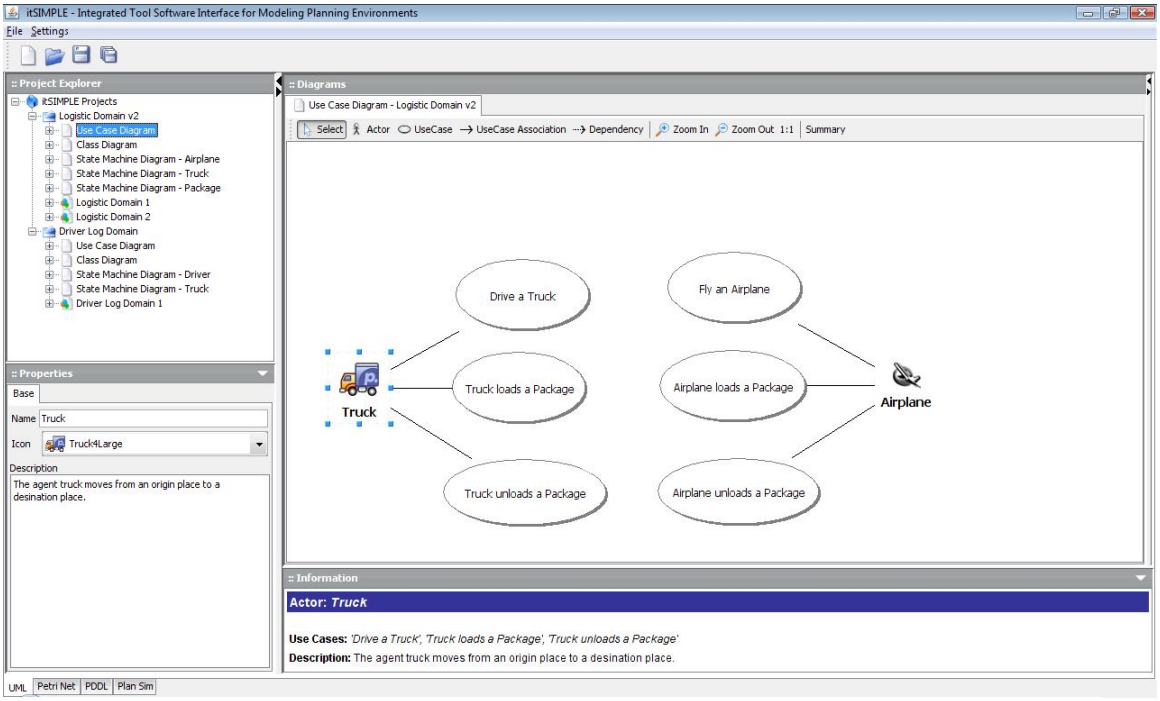

Figura 4.11: Diagrama UML de casos de uso

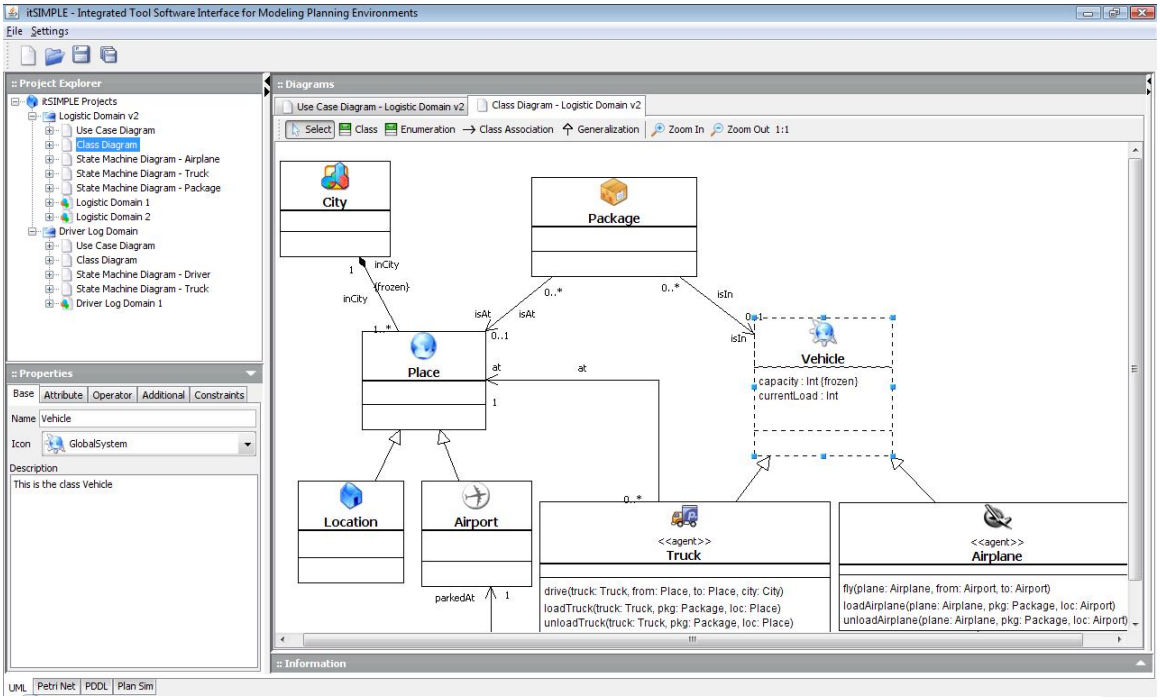

Figura 4.12: Diagrama UML de clases

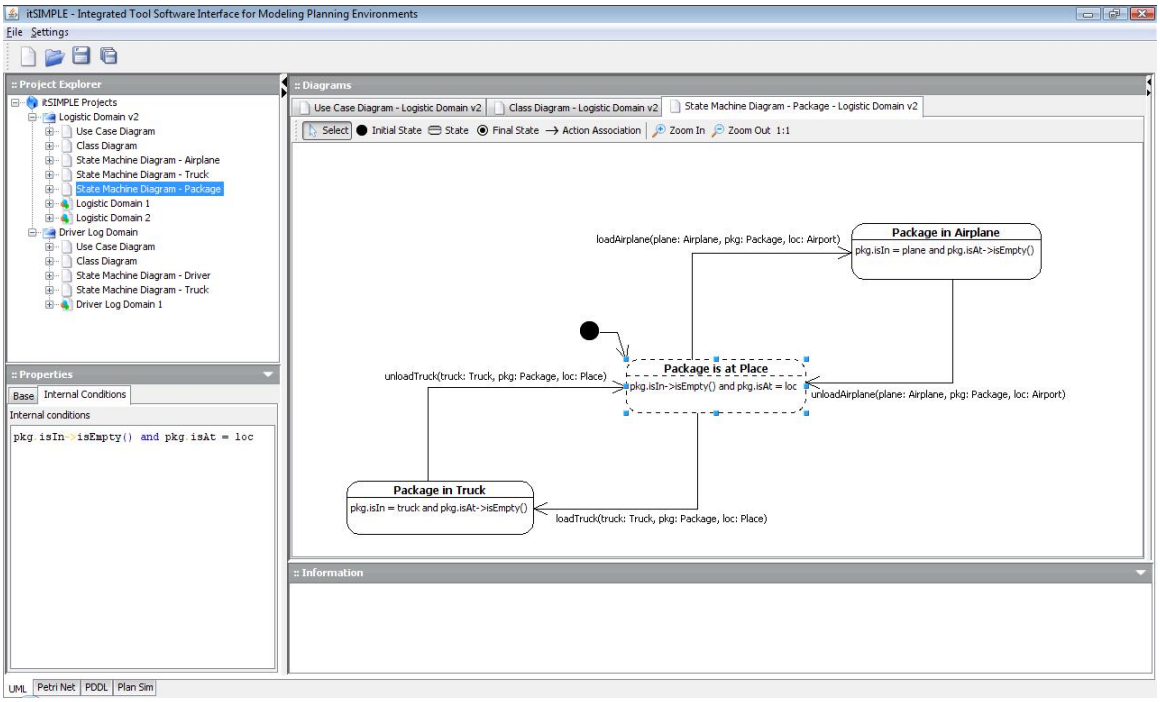

Figura 4.13: Diagrama UML de estados

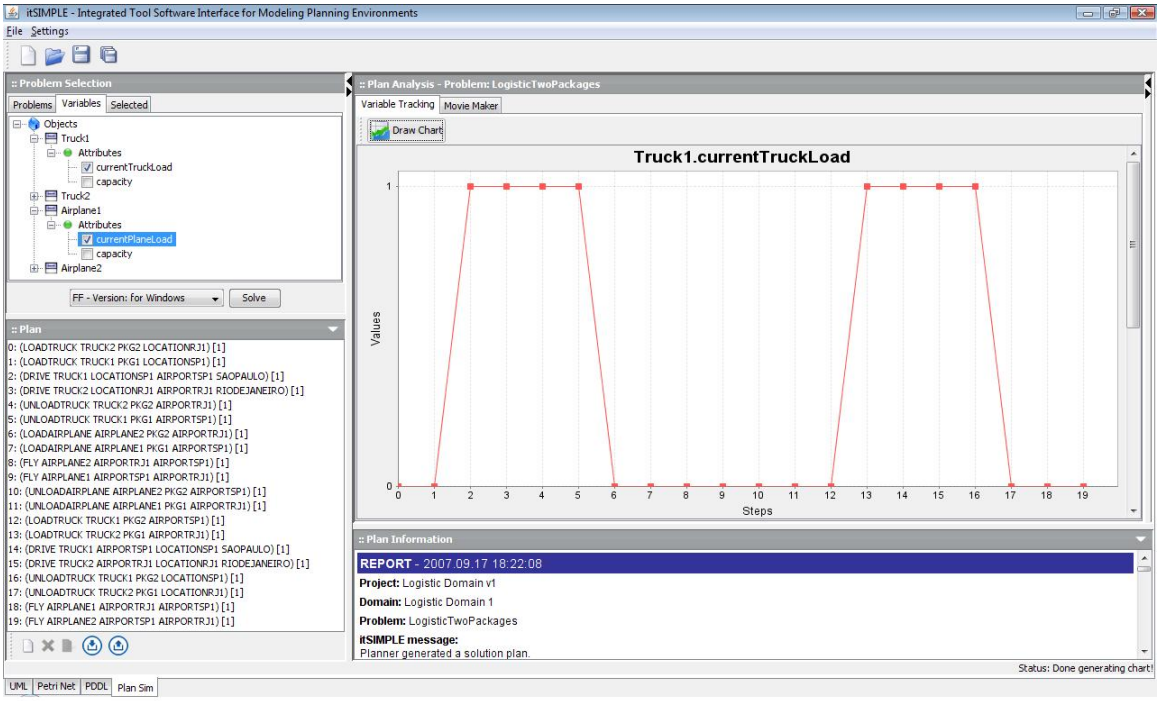

Figura 4.14: Gráfica para el estudio de variables

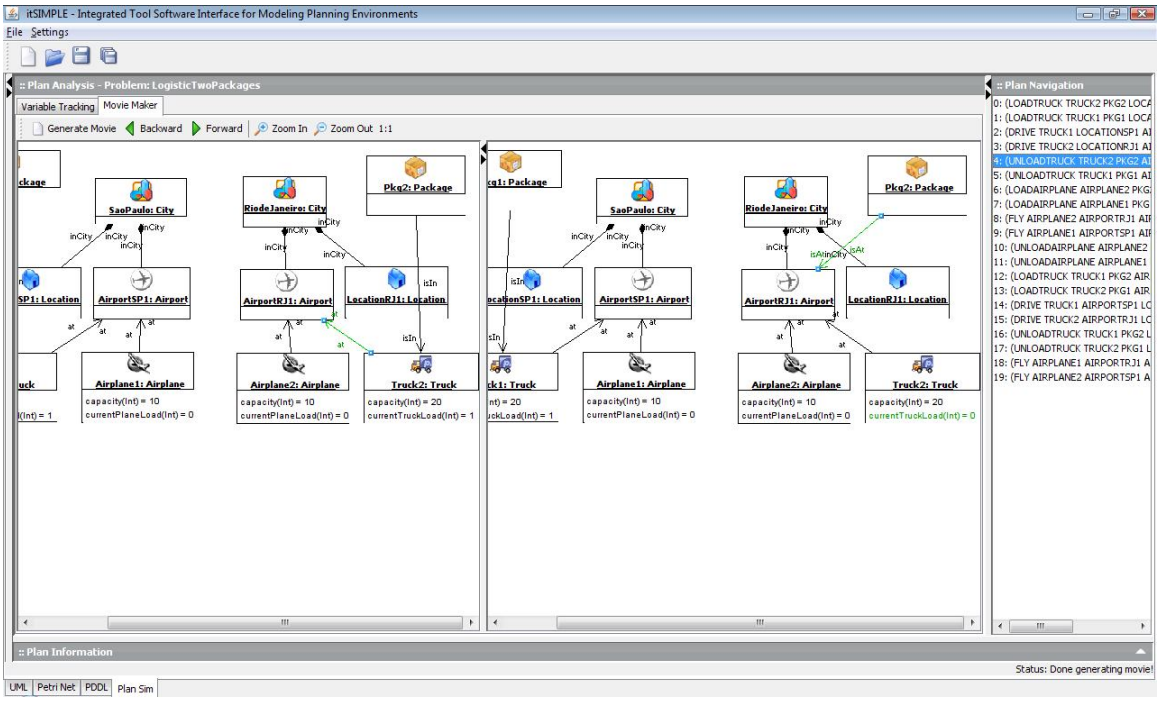

Figura 4.15: Creación de película a partir del plan ejecutado

Capítulo 5

Características principales de la herramienta PLEX

Para el desarrollo de la herramienta PLEX se ha tomado como punto de partida el Experimenter de la herramienta PLTOOL. Sin embargo, PLEX amplía las capacidades de esta herramienta, principalmente en que ofrece funcionalidades tanto para ejecutar experimentos como para analizar sus resultados (PLTOOL sólo permitía la ejecución). Las principales características de la herramienta PLEX en este sentido se detallan a continuación en este capítulo.

Por otro lado la tecnología con la que se ha construido PLEX es distinta. PLTOOL estaba programada en LISP combinado con GTK pero esto dificultaba principalmente el uso de hilos para mantener el interfaz de la herramienta activo mientras se ejecutaban los distintos planificadores. Este problema se soluciona, en gran medida con JAVA, y por ello se optó por utilizar este último lenguaje para construir la herramienta PLEX.

5.1. Procesado semi-automático de la salida de los planificadores

La herramienta Plex permite el uso de cualquier planificador gracias a que sus salidas no están estandarizadas. Para que la herramienta pueda obtener los valores que el planificador da como solución (como puede ser el coste de generar la solución, el tiempo que ha tardado en obtenerla, etc...) es necesario usar un método de lectura de la salida lo más genérico posible. Para solucionar este problema la herramienta utiliza un sistema de clave-valor que se explica a continuación.

La ejecución de un planificador normalmente produce por pantalla una salida en texto. La herramienta Plex solicita al usuario el comando para ejecutar el planificador y mostrar un ejemplo de esta salida. Sobre este ejemplo se deben identificar las palabras clave que indican dónde está el valor de las variables a considerar. La base del sistema clave-valor diseñado es que para poder obtener un valor es necesario que tenga delante o detrás un texto único en la salida e invariable para cada solución en las diversas ejecuciones (la clave). Esta clave permite a la herramienta recorrer la salida del planificador diferenciando y obteniendo todos los valores que el usuario necesite para su posterior análisis. Las claves de la herramienta pueden coincidir o no con las etiquetas que habitualmente incluyen los planificadores para indicar que valor esta mostrando en la salida.

En el caso de que se trate de un planificador con varias soluciones se usa un procedimiento parecido para diferenciarlas. En este caso se necesita una clave de texto también única e invariable que separe las soluciones, pero la clave se puede encontrar delante, detrás o en medio de cada solución.

Con este método para obtener valores y para separar soluciones la herramienta puede usar cualquier planificador, con cualquier formato de salida, siempre y cuando disponga de porciones de texto que permitan diferenciar de forma automática los valores y las soluciones.

En la figura 5.1, podemos ver resaltado en rojo las claves escogidas para obtener el tiempo y coste

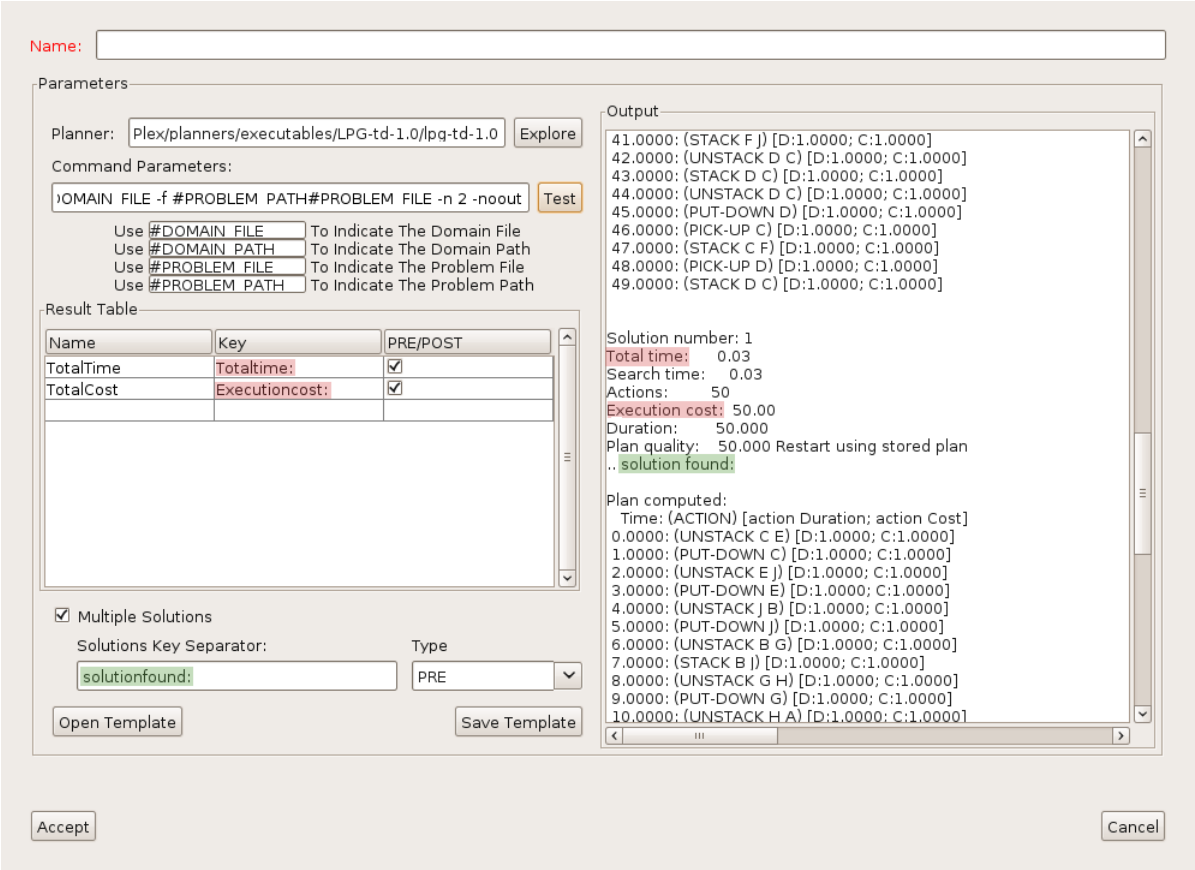

Figura 5.1: Ejemplo del procesado semi-automático de la salida de los planificadores

que ha necesitado el planificador para dar con la solución. Estas claves son únicas para cada solución del planificador y son de tipo "pre" (han de estar delante del valor), por lo que al escogerlas la herramienta podrá obtener los valores 0.03 y 50.00 que se corresponden con el tiempo y el coste, y los guardará en las variables "TotalTime" y "TotalCost" respectivamente. Por otro lado, en esta misma figura se puede observar resaltada la clave de separación de soluciones en color verde. Esta clave es de tipo "pre" por lo que se ha de encontrar al comienzo de cada solución. La clave es única ya que sólo se encuentra separando soluciones y siempre es el mismo texto, por lo que es una clave de separación de soluciones válida.

5.2. Tabla de resultados

Mientras es ejecutado el experimento, la herramienta PLEX va incluyendo los valores que el usuario ha marcado para cada planificador en una tabla. La tabla se compone de una serie de columnas que serán las variables que el usuario decidió obtener y una serie de filas que se corresponden con cada solución de cada planificador para cada problema en cada dominio. En esta tabla el usuario podrá realizar una serie de operaciones sobre los datos para estudiar como ha sido el resultado del experimento:

- El usuario podrá eliminar planificadores, dominios o problemas que considere irrelevantes una vez obtenidos los resultados.
- En el caso de que haya múltiples soluciones el usuario podrá indicar a la herramienta que solución es la mejor y cual se deberá usar para dibujar los gráficos. Esto se realiza a través de un parámetro llamado "prioridad de solución" que permite escoger una variable e indicar si es mejor la solución con mayor o menor valor. En la figura 5.2 se puede observar como funciona este operador. En la

sección "Solution Priority" se ha seleccionado la variable "TotalCost" y el operador "<", esto significa que la solución que la herramienta ha de escoger es la de menor valor para la variable "TotalCost". Así pues, en la tabla de resultados se observa que para el problema "pfile01" del dominio "blocks", el planificador "lpg" ha obtenido dos soluciones y que se escoge la solución 1 (marcada en color verde), ya que su valor para la variable "TotalCost" es menor.

- La herramienta PLEX también permite al usuario restringir las soluciones dejando fuera de los gráficos las que no cumplan ciertos criterios. Para ello el usuario puede crear reglas en las que, a partir de una variable, un operador lógico y un valor, se crea una condición que excluirá a las soluciones que no la cumplan. Como ejemplo, podemos ver la regla que se encuentra en la sección "Solution Selector Rules" (figura 5.2). Esta regla se compone por la variable "TotalCost", el operador "<" y el valor "45", produciendo que en la tabla de resultados se marquen como soluciones no válidas (en color rojo) todas aquellas cuyo valor para la variable "TotalCost" sea menor que 45

- Para el estudio de planificadores estocásticos, que cada vez que se ejecutan dan una solución diferente, la herramienta permite una composición de planificadores por mediana. Esta composición consiste en realizar la mediana para los valores de las soluciones válidas (escogidas por el operador de prioridad y que cumplan las reglas de selección de soluciones) de una serie de planificadores elegidos por el usuario para una determinada variable, que también puede seleccionar. Para que las soluciones no se corrompan cuando hace la mediana sobre un número par de variables, el planificador resultante de la operación tendrá dos soluciones, una para cada valor central, en vez de tener una conseguida a través de realizar la media aritmética entre los dos valores centrales. Podemos encontrar un ejemplo en la figura 5.3. En ella se observan los planificadores "lpg1", "lpg2", "lpg3" y "composition". Los tres primeros son ejecuciones con los mismos parámetros del planificador "lpg" y el último es una composición de éstos sobre la variable "TotalCost". Se puede observar que debido a la regla de selección de soluciones para el problema "pfile01" dispone de tres soluciones válidas para los planificadores que forman la composición ("lpg1", "lpg2" y "lpg3") por lo que el planificador "composition" únicamente tiene una solución que se corresponde con la mediana de

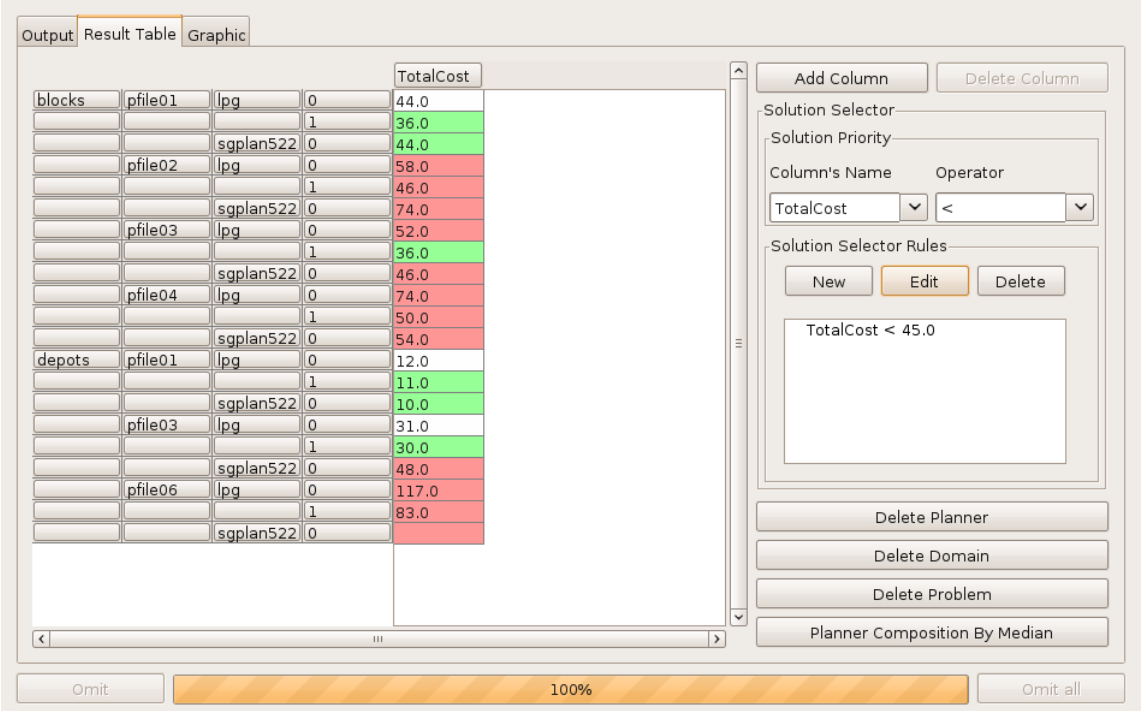

Figura 5.2: Ejemplo de selección de soluciones

las tres soluciones. En cambio para el problema "pfile10" la regla de selección de soluciones sólo ha permitido disponer de dos soluciones válidas para los tres planificadores que forman la composición, por lo que el planificador "composition" tiene dos soluciones con los valores de las dos únicas soluciones de los planificadores en vez de uno como ocurre en el problema "pfile01".

- La herramienta PLEX también permite al usuario añadir columnas a la tabla de resultados. Para especificar a la herramienta con qué valores se va a generar la nueva columna, el usuario puede generar código JAVA que será cargado en tiempo de ejecución. Además la herramienta provee al usuario una serie de métodos JAVA que le permiten acceder a la tabla de resultados, pudiendo crear métodos que permitan el añadido de columnas que acumulen o realicen escalas logarítmicas sobre otras, o incluso que puntúen cómo de buenas son las soluciones. Las operaciones que incluye la herramienta están explicadas en la sección 5.4

5.3. Gráficas de resultados

Durante la ejecución del experimento, la herramienta PLEX, además de añadir los valores a la tabla de resultados, va generando dos tipos de gráficos. A continuación se definen estos tipos de gráficos:

- **Gráfico estándar:** Se trata de un gráfico de líneas en el que usuario selecciona una variable y un dominio, mostrando así los valores de la tabla de resultados sin ninguna modificación. En este gráfico el eje de abscisas marca cada uno de los problemas del dominio, el eje de ordenadas representa los valores, y las líneas muestran cada uno de los planificadores. En la figura 5.4 se puede encontrar un ejemplo de este tipo de gráficos. El gráfico se corresponde a la tabla de resultados de la figura 5.3 y en él se representa el valor de las soluciones válidas para la variable "TotalCost". En el planificador "lpg1" no se representa ningún valor para el problema "pfile10" porque la regla de selección de soluciones no tienen ninguna solución valida.

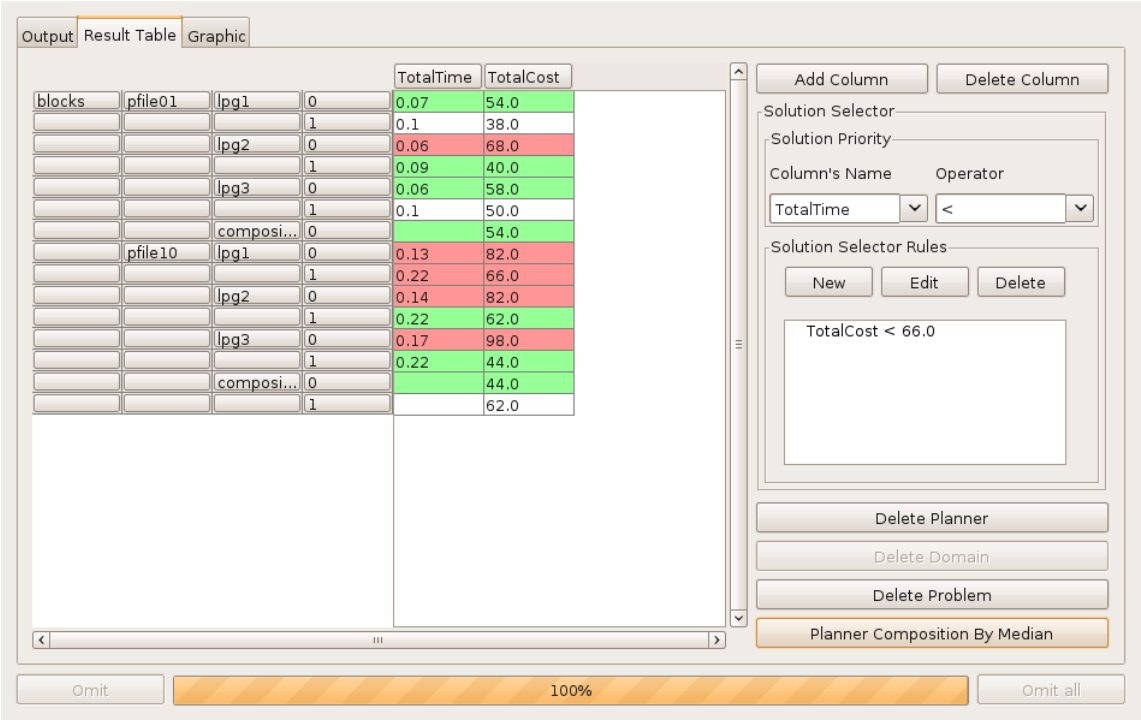

Figura 5.3: Ejemplo de composición de planificadores

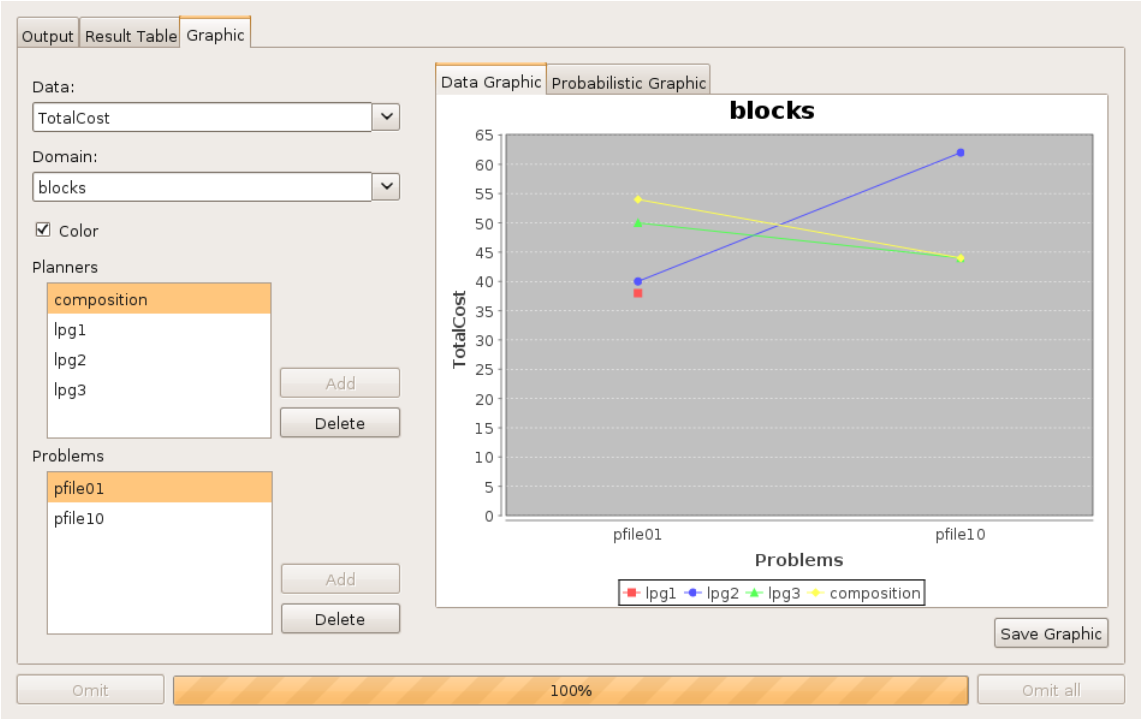

Figura 5.4: Ejemplo gráfico estándar

- **Gráfico probabilístico:** Se trata también de un gráfico de líneas en el que, a partir de la variable y el dominio seleccionado por el usuario, se muestra el porcentaje de problemas resueltos que ha obtenido un planificador para valores de la variable menores o iguales a un determinado valor. Así pues, en este gráfico el eje de abscisas representa los valores de la variable, el eje de ordenadas muestra el porcentaje de problemas resueltos y las líneas representan a cada uno de los planificadores. En la figura 5.5 se observa un ejemplo de gráfico probabilístico que se basa en los datos de la tabla de resultados de la figura 5.3. En este gráfico cabe destacar que todos los planificadores han encontrado solución a todos los problemas, menos el planificador "lpg1" que se queda en el cincuenta, mientras que todos los demás alcanzan el cien por cien.

La herramienta PLEX permite al usuario elegir qué problemas y planificadores se van a representar en el gráfico, pudiendo simplificar gráficos en experimentos grandes o destacar ciertas partes del experimento.

Todos los gráficos generados se pueden guardar como imágenes en formato PNG y JPEG.

5.4. Análisis global de resultados

Además de permitir al usuario generar operaciones en JAVA para la creación de nuevas variables o columnas, la herramienta PLEX viene con una serie de operaciones genéricas que facilitan el análisis global de los resultados. Estas operaciones son:

- **Operación de acumulado por dominio y planificador sólo para problemas resueltos:** Esta operación realiza un acumulado sobre otra columna. Los valores acumulados serán por dominio y por planificador. Si algún planificador no resuelve un problema para un dominio, no se acumulará ese problema para ningún planificador del dominio. En las figuras 5.6 y 5.7 encontramos un ejemplo de esta operación. Observamos cómo se han acumulado las soluciones de todos los problemas menos la del problema "pfile06" del dominio "depots" para el planificador "sgplan" ya que no ha tenido solución. Se puede observar también que el acumulado se hace por dominio y planificador.

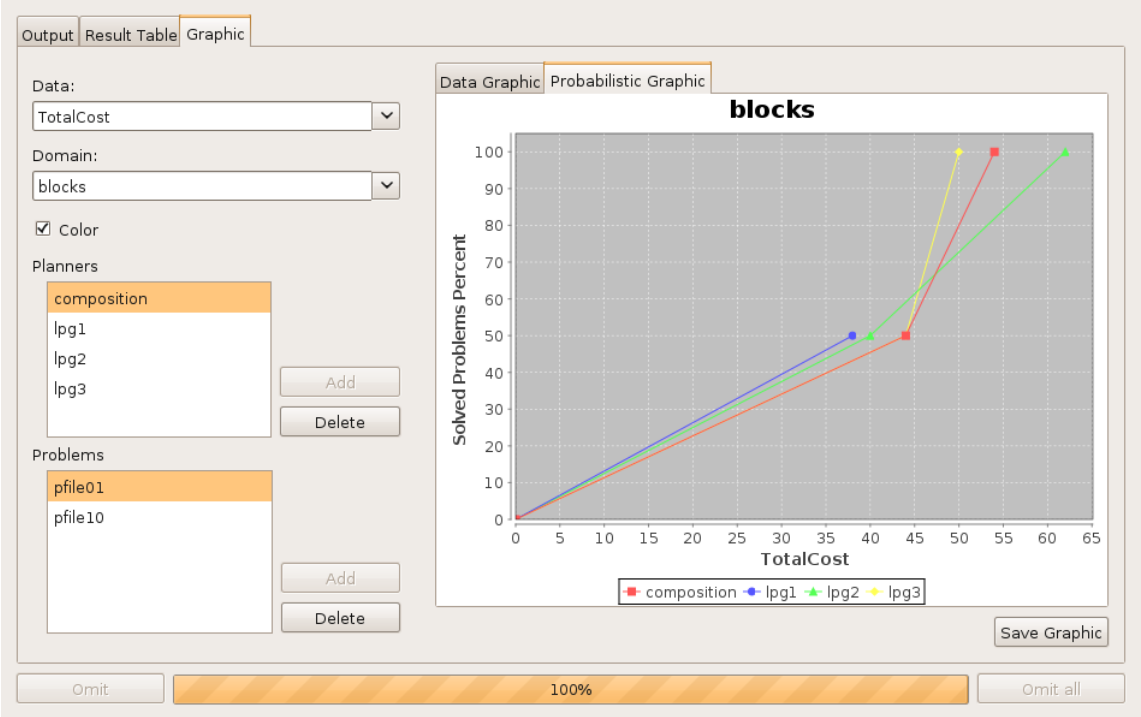

Figura 5.5: Ejemplo gráfico probabilístico

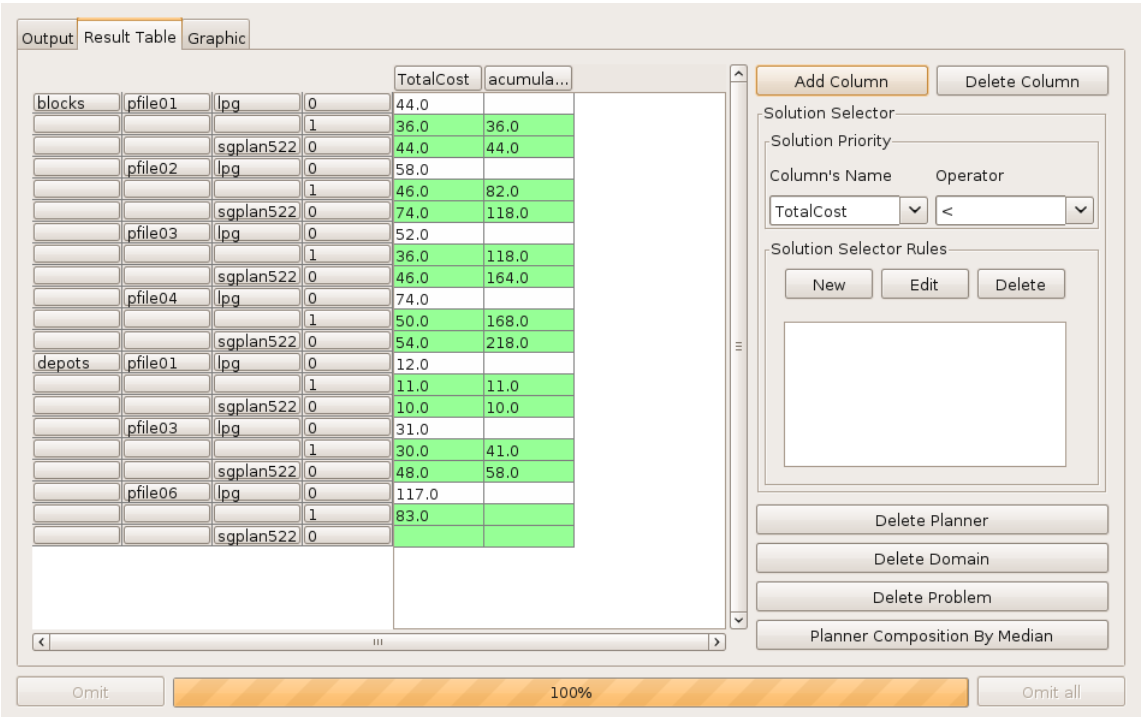

Figura 5.6: Ejemplo de operación de acumulado por dominio y planificador sólo para problemas resueltos

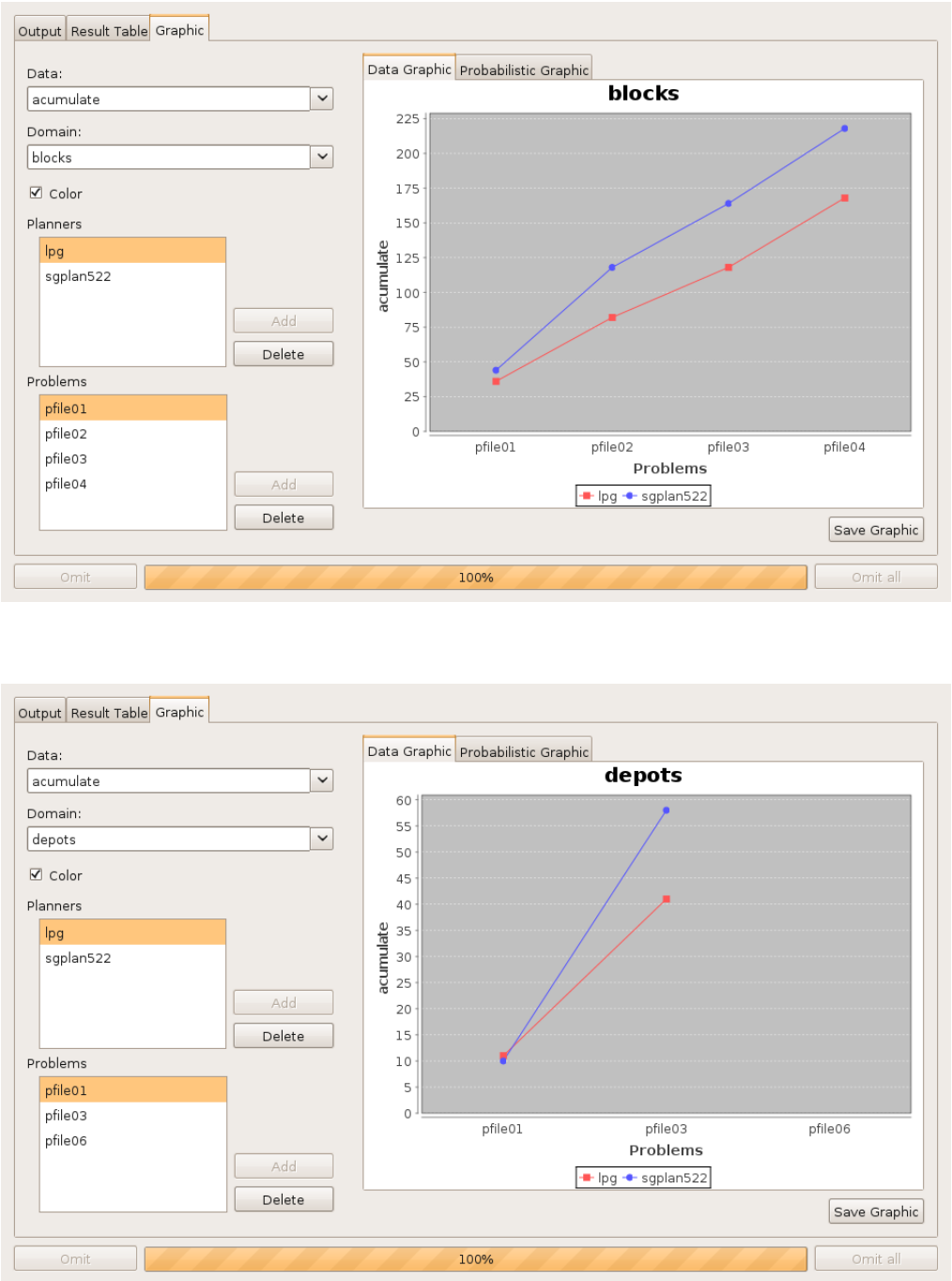

Figura 5.7: Ejemplo de operación de acumulado por dominio y planificador sólo para problemas resueltos

- **Operación de acumulado por dominio y planificador si todos se han resuelto:** Esta operación realiza un acumulado sobre otra columna. Los valores acumulados serán por dominio y por planificador. Si algún planificador no resuelve un problema para un dominio, no se acumulará ningún valor para el planificador en el dominio. En las figuras 5.8 y 5.9 observamos un ejemplo de esta operación. En ella vemos cómo para el dominio "blocks" se ha realizado un acumulado sobre las soluciones válidas para cada planificador. En cambio, para el dominio "depots" sólo se ha realizado el acumulado para las soluciones válidas del planificador "lpg" pero no para el planificador "sgplan" ya que éste no ha conseguido solucionar todos los problemas del dominio.

- **Operación de escala logarítmica:** Esta operación realiza un logaritmo sobre los valores de otra columna. En la figura 5.10 se puede observar un ejemplo en el que simplemente se hace un logaritmo en base dos a los valores de la columna "TotalCost".

- **Operación de puntuado de soluciones:** Esta operación divide el mejor valor obtenido por todos los planificadores para un problema de un dominio por el valor obtenido por el planificador en ese problema de ese dominio o a la inversa, dependiendo de si el mejor valor es el menor o el mayor respectivamente. Con el resultado de esta función y un acumulado se puede puntuar qué planificador ha funcionado mejor. En la figura 5.11 podemos observar un ejemplo de esta operación. En este caso vemos cómo para cada problema de cada dominio el mejor valor de la columna "TotalCost" genera un uno en la columna "score" y para los demás valores genera números entre cero y uno dependiendo de lo buena que sea la solución. Con la columna "score" podemos ver qué planificador se ha comportado mejor para cada problema. Si además hacemos un acumulado para la columna "score" podremos observar qué planificador se ha comportado mejor para cada dominio, como se puede observar en la columna "acumulate". En las figuras 5.12, 5.13 y 5.14 se refleja este ejemplo con gráficos para el dominio blocks. Así pues, en la figura 5.12, se muestran los valores para el "TotalCost" del dominio "Blocks". En la figura 5.13 se muestra el resultado de esta operación. Se puede apreciar que los planificadores con mejores resultados obtienen un uno como valor de score, y el resto, valores entre cero y uno dependiendo de lo buena que sea la solución (notar que para

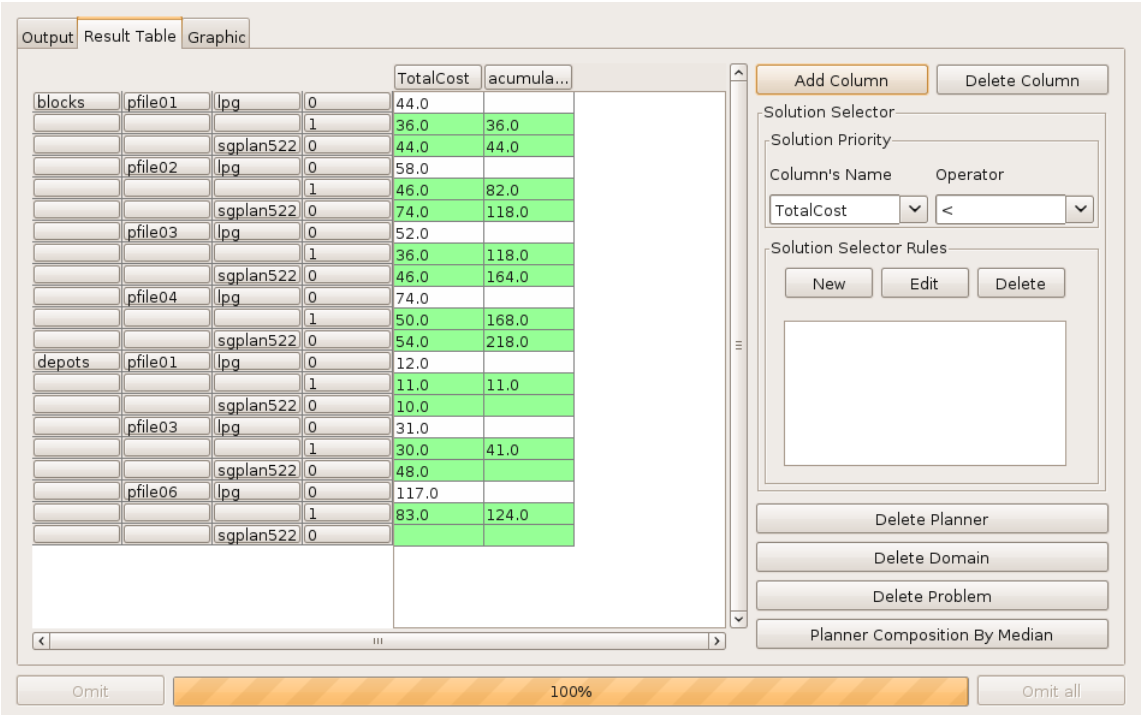

Figura 5.8: Ejemplo de operación de acumulado por dominio y planificador si todos se han resuelto

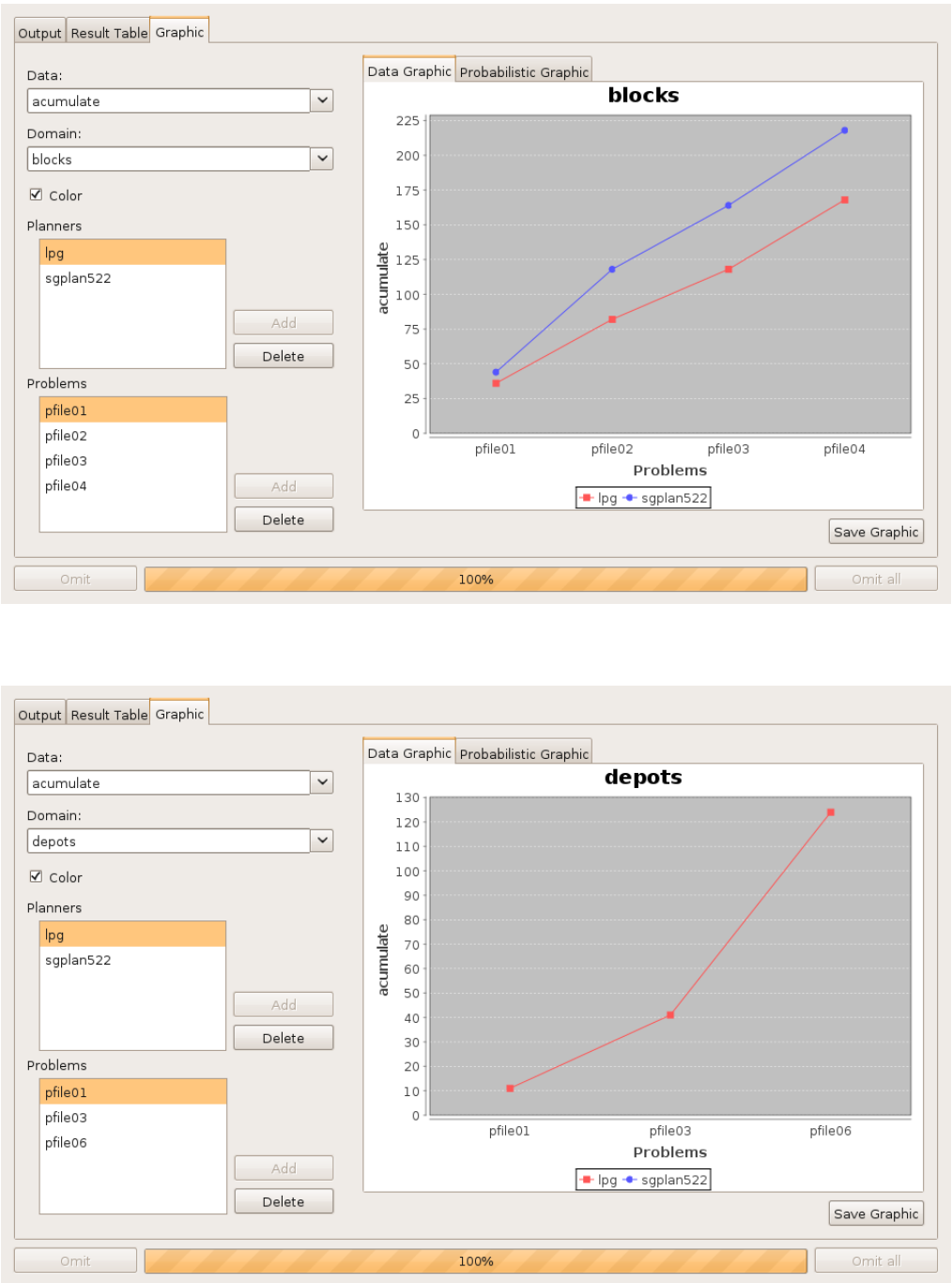

Figura 5.9: Ejemplo de operación de acumulado por dominio y planificador si todos se han resuelto

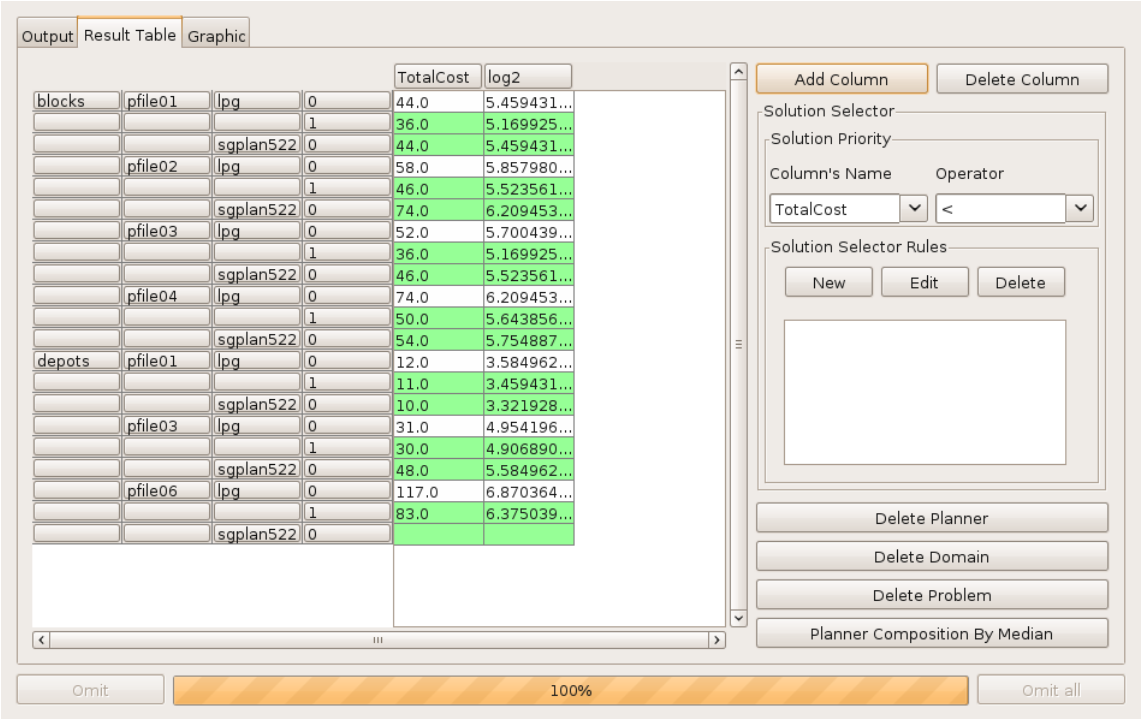

Figura 5.10: Ejemplo de operación de escala logarítmica

el coste total el resultado es mejor cuanto menor sea). Por último, en la figura 5.14 se muestra un acumulado de la columna "score" mostrando qué planificador ha conseguido mejores resultados y en este caso es "sgplan".

5.5. Finalización por límite excedido independiente del planificador

Para poder limitar la ejecución de los planificadores la herramienta PLEX permite al usuario introducir un tiempo en minutos máximo de CPU o un tamaño máximo de memoria en KiloBytes para todos los planificadores. Así pues, cuando uno de ellos sobrepase alguno de estos valores se detendrá su ejecución y se analizará lo que el planificador haya obtenido hasta el momento.

5.6. Guardado automático de los experimentos durante la ejecución

PLEX guarda el experimento cada vez que un planificador completa su ejecución. De esta manera se evita una posible pérdida de datos, así como tener que repetir el experimento en el caso de que se produjera un fallo en el sistema que detuviera la ejecución de la herramienta. Por este motivo siempre que se quiere ejecutar un experimento es necesario indicar un archivo dónde guardarlo si no se había hecho previamente.

En el caso de que una ejecución se haya detenido, intencionada o inesperadamente, se podrá retomar la ejecución tan sólo con abrir el archivo de experimento. Esta ejecución continuará desde el último planificador que haya completado su ejecución.

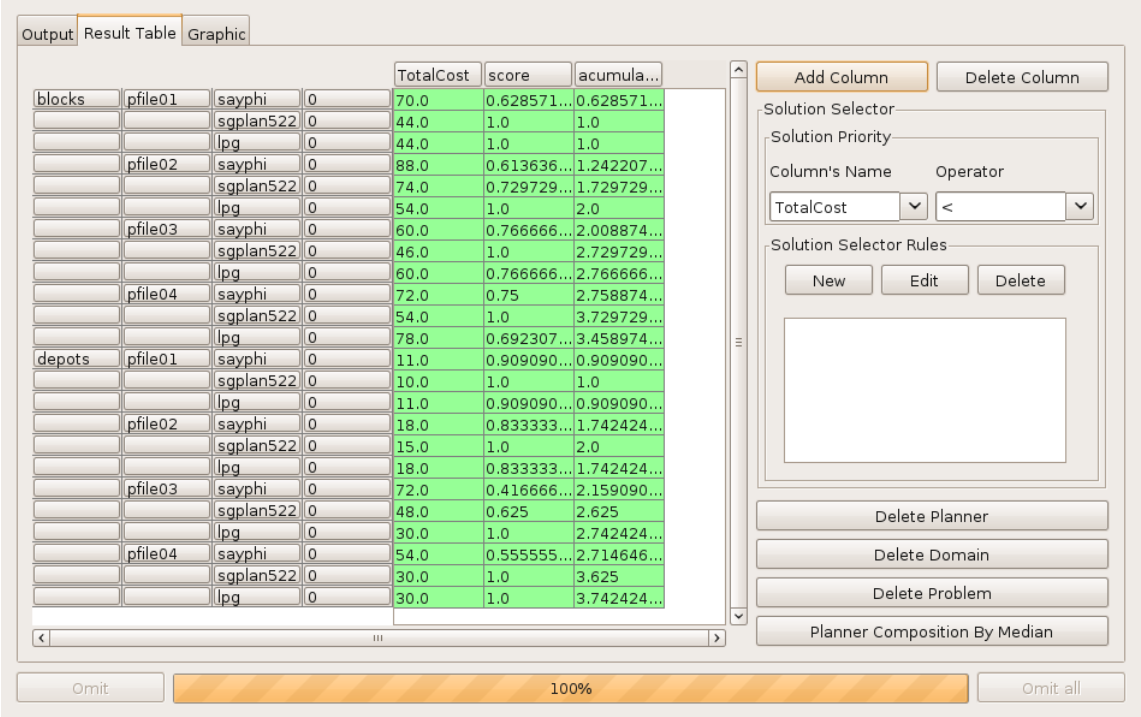

Figura 5.11: Ejemplo de operación de puntuado de soluciones

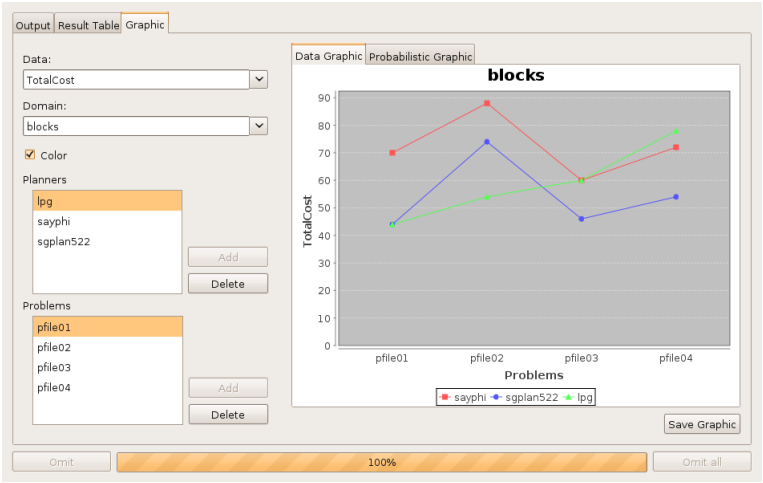

Figura 5.12: Ejemplo de operación de puntuado de soluciones

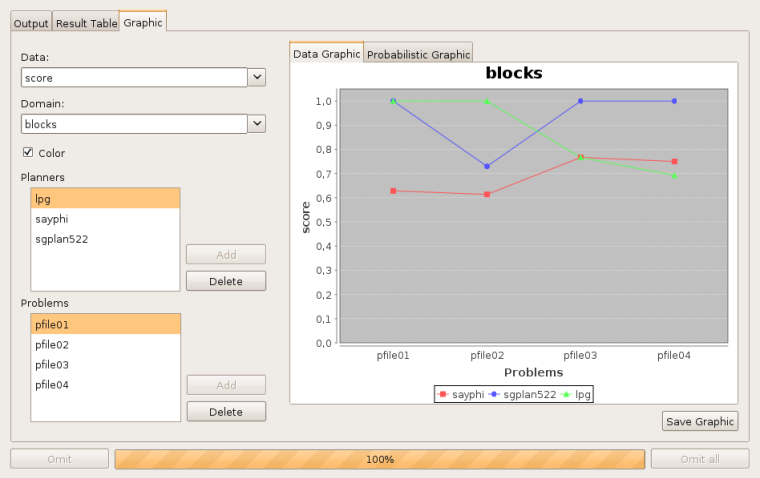

Figura 5.13: Ejemplo de operación de puntuado de soluciones

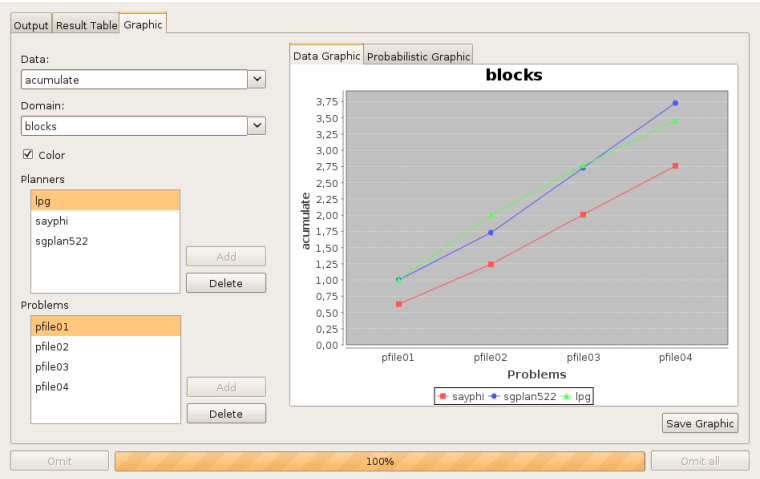

Figura 5.14: Ejemplo de operación de puntuado de soluciones

5.7. Cambio automático de idioma

La herramienta PLEX permite que se cambie el idioma de forma dinámica durante el uso de la aplicación. Para ello lo único necesario es la inclusión de un archivo de idiomas como se indica en la sección A.7.1 del manual de usuario.

5.8. Formato de los archivos XML

La herramienta PLEX usa el formato XML para guardar los archivos de experimentos y los archivos de plantilla de planificador.

5.8.1. Archivos de experimento

Los archivos de experimento contienen toda la información sobre la configuración, ejecución y resultados del experimento en el caso de que los tenga. Estos archivos se generan al guardar un experimento. La herramienta PLEX aceptará todos los experimentos que sigan el formato XML definido en el archivo "experimenter.dtd" contenido en la carpeta "dtds" de la herramienta. En la tabla 5.1 se muestra el DTD del archivo de experimento. En él encontramos los siguientes elementos:

Tabla 5.1: DTD del archivo de experimento

```
<!ELEMENT EXPERIMENTER ( EXPERIMENT?, RESULTS? )>
    <!ELEMENT EXPERIMENT ( DOMAINS, PLANNERS )>
        <!ELEMENT DOMAINS ( DOMAIN* )>
            <!ELEMENT DOMAIN ( NAME, FILE_PATH, PROBLEMS )>
                <!ELEMENT PROBLEMS ( PROBLEM+ )>
                    <!ELEMENT PROBLEM ( FILE_PATH )>
        <!ELEMENT PLANNERS ( CPU_TIME_LIMIT?, MEMORY_LIMIT?, PLANNER* )>
            <!ELEMENT PLANNER ( NAME, PLANNER_PARAMETERS )>
                <!ELEMENT PLANNER_PARAMETERS ( FILE_PATH, COMMAND,
NAME_KEYS, MULTIPLE_SOLUTION_KEY_DATA? )>
                    <!ELEMENT NAME_KEYS ( NAME_KEY+ )>
                        <!ELEMENT NAME_KEY ( NAME, KEY, PRE_POST )>
                    <!ELEMENT MULTIPLE_SOLUTION_KEY_DATA ( KEY, TYPE
)>
    <!ELEMENT RESULTS ( COLUMNS_NAMES, DOMAINS_PROBLEMS_NAMES, PLANNERS-
_NAMES, RESULTS_VALUES, RUNNER?, SOLUTION_SELECTOR? )>
        <!ELEMENT COLUMNS_NAMES ( COLUMN_NAME+ )>
        <!ELEMENT DOMAINS_PROBLEMS_NAMES ( DOMAIN_PROBLEMS_NAMES+ )>
            <!ELEMENT DOMAIN_PROBLEMS_NAMES ( DOMAIN_NAME, PROBLEMS-
_NAMES )>
                <!ELEMENT PROBLEMS_NAMES ( PROBLEM_NAME+ )>
        <!ELEMENT PLANNERS_NAMES ( PLANNER_NAME+ )>
        <!ELEMENT RESULTS_VALUES ( RESULT+ )>
            <!ELEMENT RESULT ( PLANNER_NAME, DOMAIN_NAME, PROBLEM_-
NAME, OUTPUT, COMMAND, SOLUTIONS )>
                <!ELEMENT SOLUTIONS ( SOLUTION+ )>
                    <!ELEMENT SOLUTION ( DATA* )>
                        <!ELEMENT DATA ( NAME, VALUE )>
        <!ELEMENT RUNNER ( DOMAIN_COUNT, PLANNER_COUNT, PROBLEM_COUNT
)>
        <!ELEMENT SOLUTION_SELECTOR ( SOLUTION_PRIORITY, SOLUTION_SELEC-
TOR_RULES )>
            <!ELEMENT SOLUTION_PRIORITY ( COLUMN_NAME, OPERATOR )>
            <!ELEMENT SOLUTION_SELECTOR_RULES ( SOLUTION_SELECTOR_RU-
LE* )>
```

Tabla 5.1 – continúa en la página siguiente

Tabla 5.1 – continúa de la página anterior

```
                    <!ELEMENT SOLUTION_SELECTOR_RULE ( COLUMN_NAME, OPE-
RATOR, VALUE )>
<!ELEMENT NAME ( #PCDATA )>
<!ELEMENT FILE_PATH ( #PCDATA )>
<!ELEMENT CPU_TIME_LIMIT ( #PCDATA )>
<!ELEMENT MEMORY_LIMIT ( #PCDATA )>
<!ELEMENT COMMAND ( #PCDATA )>
<!ELEMENT KEY ( #PCDATA )>
<!ELEMENT PRE_POST ( #PCDATA )>
<!ELEMENT TYPE ( #PCDATA )>
<!ELEMENT COLUMN_NAME ( #PCDATA )>
<!ELEMENT DOMAIN_NAME ( #PCDATA )>
<!ELEMENT PROBLEM_NAME ( #PCDATA )>
<!ELEMENT PLANNER_NAME ( #PCDATA )>
<!ELEMENT OUTPUT ( #PCDATA )>
<!ELEMENT VALUE ( #PCDATA )>
<!ELEMENT DOMAIN_COUNT ( #PCDATA )>
<!ELEMENT PLANNER_COUNT ( #PCDATA )>
<!ELEMENT PROBLEM_COUNT ( #PCDATA )>
<!ELEMENT OPERATOR ( #PCDATA )>
```

Tabla 5.1: DTD del archivo de experimento

- **EXPERIMENTER:** Representa un experimento completo, tanto la parte de configuración como la de resultados.
- **EXPERIMENT:** Representa la parte de configuración de un experimento. Esta información es la que se introduce antes de ejecutar el experimento y se compone de todos los dominios y planificadores configurados.
- **DOMAINS:** Lista los dominios que se han configurado en el experimento.
- **DOMAIN:** Representa un dominio configurado en el experimento que se compone de un nombre, una ruta al fichero de dominio y una serie de problemas.
- **PROBLEMS:** Lista los problemas configurados para un dominio.
- **PROBLEM:** Representa un problema configurado para un dominio y se compone de una ruta al fichero del problema.
- **PLANNERS:** Lista de los planificadores que se han configurado en el experimento. Además incluye la información del tiempo límite de CPU y el máximo de memoria a utilizar.
- **PLANNER:** Representa un planificador configurado en el experimento. Se compone de un nombre de planificador y una serie de parámetros que el usuario configura.
- **PLANNER_PARAMETERS:** Representa la configuración que se ha dado a un planificador para un experimento. Los parámetros que incluye son una ruta al fichero ejecutable del planificador, el comando de ejecución del planificador, la información de los nombres-claves y la información sobre planificadores con varias soluciones.
- **NAME_KEYS:** Lista cada uno de los nombre-clave que se han introducido en un planificador
- **NAME_KEY:** Representa el nombre de cada variable a obtener, la clave para obtenerla y el tipo de clave usada.
- **MULTIPLE_SOLUTION_KEY_DATA:** Representa la clave usada para obtener las diferentes soluciones y el tipo de esta clave.

- **RESULTS:** Enumera los resultados obtenidos tras la ejecución de un planificador. Contiene la información de los nombres de las columnas, los nombres de los dominios con los nombres de sus problemas y los nombres de los planificadores que aparecerán en la tabla de resultados. Además contiene cada uno de los resultados, información sobre si la ejecución ha terminado y cómo el usuario esta seleccionando soluciones.
- **COLUMNS_NAMES:** Lista los nombres de columna que aparecen en la tabla de resultados.
- **DOMAINS_PROBLEMS_NAMES:** Especifica los nombres de dominio con sus respectivos problemas que aparecen en la tabla de resultados.
- **DOMAIN_PROBLEMS_NAMES:** Lista para cada nombre de dominio los nombres de los problemas que aparecen en la tabla de resultados.
- **PROBLEMS_NAMES:** Lista los nombres de problemas que aparecen en la tabla de resultados.
- **PLANNERS_NAMES:** Enumera los nombres de los planificadores que aparecen en la tabla de resultados.
- **RESULTS_VALUES:** Detalla cada uno de los resultados que se han obtenido para la tabla de resultados.
- **RESULT:** Representa un resultado de la tabla de resultados. Se compone de un nombre de dominio, de problema y de planificador. Además contiene la salida del planificador, el comando utilizado para ejecutarlo y las soluciones obtenidas.
- **SOLUTIONS:** Enumera las soluciones obtenidas para un planificador.
- **SOLUTION:** Representa los datos de cada una de las soluciones que puede obtener un planificador.
- **DATA:** Representa los datos de una solución. Se compone de un nombre de variable y de su valor.
- **RUNNER:** Representa la información sobre en que punto se encuentra la ejecución. Para ello almacena el numero de planificador, domino y problema que se está ejecutando.
- **SOLUTION_SELECTOR:** Contiene la configuración del usuario para seleccionar soluciones. Se compone de un selector de prioridad y de una serie de reglas de selección de soluciones.
- **SOLUTION_PRIORITY:** Representa la prioridad con la que se escogen las soluciones. Se compone de un nombre de columna y un operador.
- **SOLUTION_SELECTOR_RULES:** Enumera las reglas de selección de soluciones.
- **SOLUTION_SELECTOR_RULE:** Representa una regla de selección de soluciones. Se compone de un nombre de columna, un operador, y un valor.
- **NAME:** Representa un nombre.
- **FILE_PATH:** Representa la ruta a un fichero.
- **CPU_TIME_LIMIT:** Representa el número de minutos que un planificador puede usar la CPU como máximo.
- **MEMORY_LIMIT:** Representa el número de KiloBytes que puede usar un planificador como máximo.
- **COMMAND:** Representa el comando con el que se ejecuta el planificador.
- **KEY:** Representa una clave de texto inequívoco e invariable.
- **PRE_POST:** Representa el tipo de clave para las que se encuentran en la tabla nombre-clave.
- **TYPE:** Representa el tipo de clave para las múltiples soluciones.

- **COLUMN_NAME:** Representa un nombre de columna para la tabla de resultados.
- **DOMAIN_NAME:** Representa un nombre de dominio para la tabla de resultados.
- **PROBLEM_NAME:** Representa un nombre de problema para la tabla de resultados.
- **PLANNER_NAME:** Representa un nombre de planificador para la tabla de resultados.
- **OUTPUT:** Representa la salida de un planificador.
- **VALUE:** Representa un valor numérico.
- **DOMAIN_COUNT:** Representa el número de dominio que se está ejecutando.
- **PLANNER_COUNT:** Representa el número de planificador que se está ejecutando.
- **PROBLEM_COUNT:** Representa el número de problema que se está ejecutando.
- **OPERATOR:** Representa un operador.

5.8.2. Archivos de plantilla de planificador

Los archivos de plantilla de planificador contienen toda la información sobre la configuración de un planificador y se generan al guardar una plantilla dentro de la configuración de un planificador. Estos archivos permiten al usuario tener configuraciones genéricas de planificadores para facilitar la configuración de experimentos. La herramienta PLEX aceptará todos las plantillas de planificador que sigan el formato XML definido en el archivo "planner.dtd" contenido en la carpeta "dtds" de la herramienta. En la tabla 5.2 se muestra el DTD del archivo de plantilla de planificador. Los elementos que encontramos en él se pueden ver definidos en la sección "Archivos de experimento" (5.8.1).

Tabla 5.2: DTD del archivo de plantilla de planificador

```
<!ELEMENT PLANNER_PARAMETERS ( FILE_PATH, COMMAND, NAME_KEYS, MULTIPLE_-
SOLUTION_KEY_DATA? )>
        <!ELEMENT NAME_KEYS ( NAME_KEY+ )>
                <!ELEMENT NAME_KEY ( NAME, KEY, PRE_POST )>
                        <!ELEMENT MULTIPLE_SOLUTION_KEY_DATA ( KEY, TYPE )> <!ELE-
MENT FILE_PATH ( #PCDATA )>
<!ELEMENT COMMAND ( #PCDATA )>
<!ELEMENT NAME ( #PCDATA )>
<!ELEMENT KEY ( #PCDATA )>
<!ELEMENT PRE_POST ( #PCDATA )>
<!ELEMENT TYPE ( #PCDATA )>
```

Tabla 5.2: DTD del archivo de plantilla de planificador

Capítulo 6

Desarrollo

En este capítulo se especifica el análisis y el diseño de la herramienta PLEX a través de diversas tablas y diagramas que detallan el trabajo desarrollado para la elaboración del proyecto.

6.1. Análisis del sistema

Para desarrollar el análisis de la herramienta PLEX se han generado diagramas de casos de uso y se han obtenido los requisitos software. Ambos se definen y detallan en este punto.

6.1.1. Casos de Uso

En esta sección se van identificar los casos de uso que la herramienta PLEX implementará. Estos casos de uso son necesarios para la posterior identificación de los requisitos software y definirán la funcionalidad de la aplicación.

Los casos de uso se han separado en varios grupos para facilitar el estudio de éstos. Así pues, en la figura 6.1, podemos observar los casos de uso generales del experimentador, en la figura 6.2 y la figura 6.3 podemos observar todos los casos de uso relacionados con los dominios y los planificadores respectivamente. Los casos de uso que aparecen durante la ejecución se observan en la figura 6.4. En la figura 6.5 y la figura 6.7 se representan los casos de uso sobre la tabla de resultados y las gráficas respectivamente. Por último, en la figura 6.6 se observan los casos de uso que permiten el rellenado de columnas nuevas.

A continuación se muestra una tabla en la que se identifica cada caso de uso, se indica sus objetivos y se explica cuáles son las precondiciones necesarias para poder llegar a ese caso de uso, así como las postcondiciones o efectos que tendrá en el sistema. Para ello se usa la plantilla que aparece en la tabla 6.1.

Identificador	
Caso de uso:	
Objetivo:	
Precondiciones:	
Postcondiciones:	

Tabla 6.1: Plantilla casos de uso

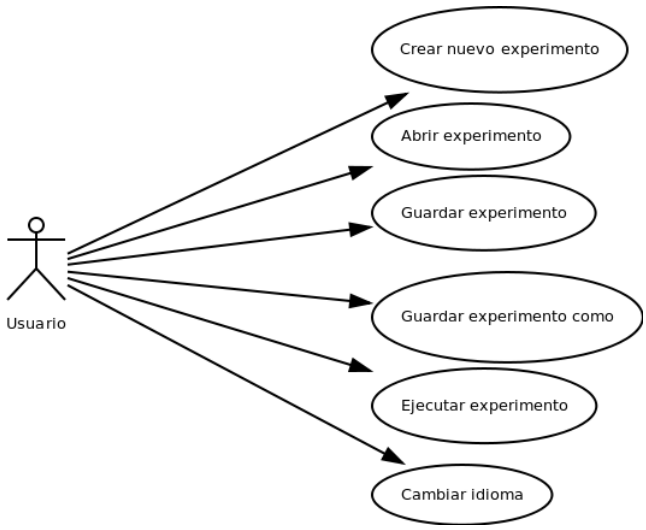

Figura 6.1: Casos de uso del experimentador

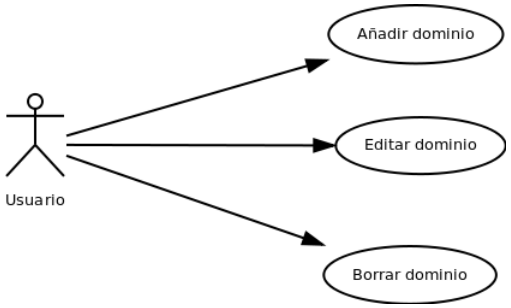

Figura 6.2: Casos de uso sobre los dominios

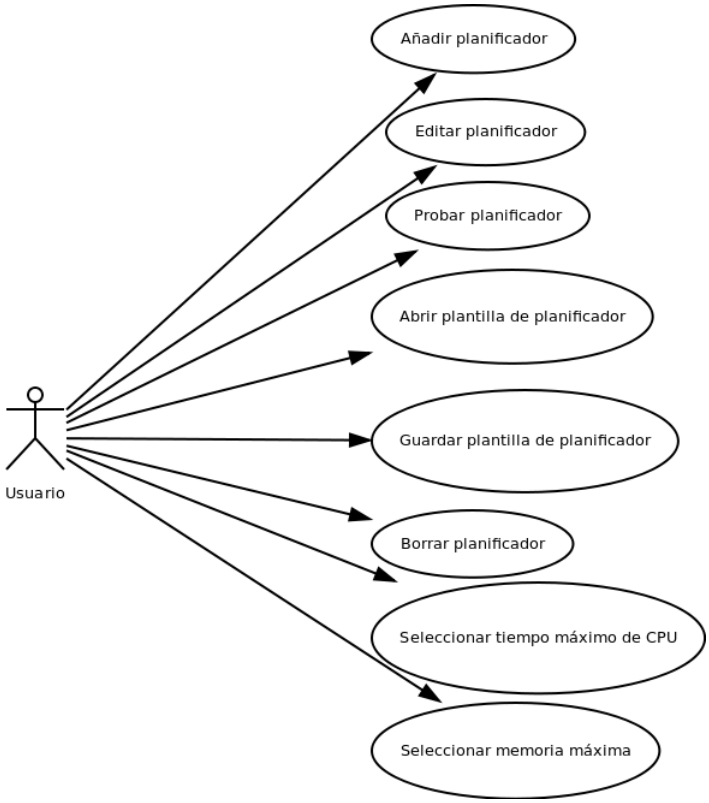

Figura 6.3: Casos de uso sobre los planificadores

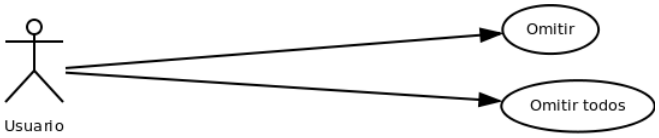

Figura 6.4: Casos de uso durante la ejecución

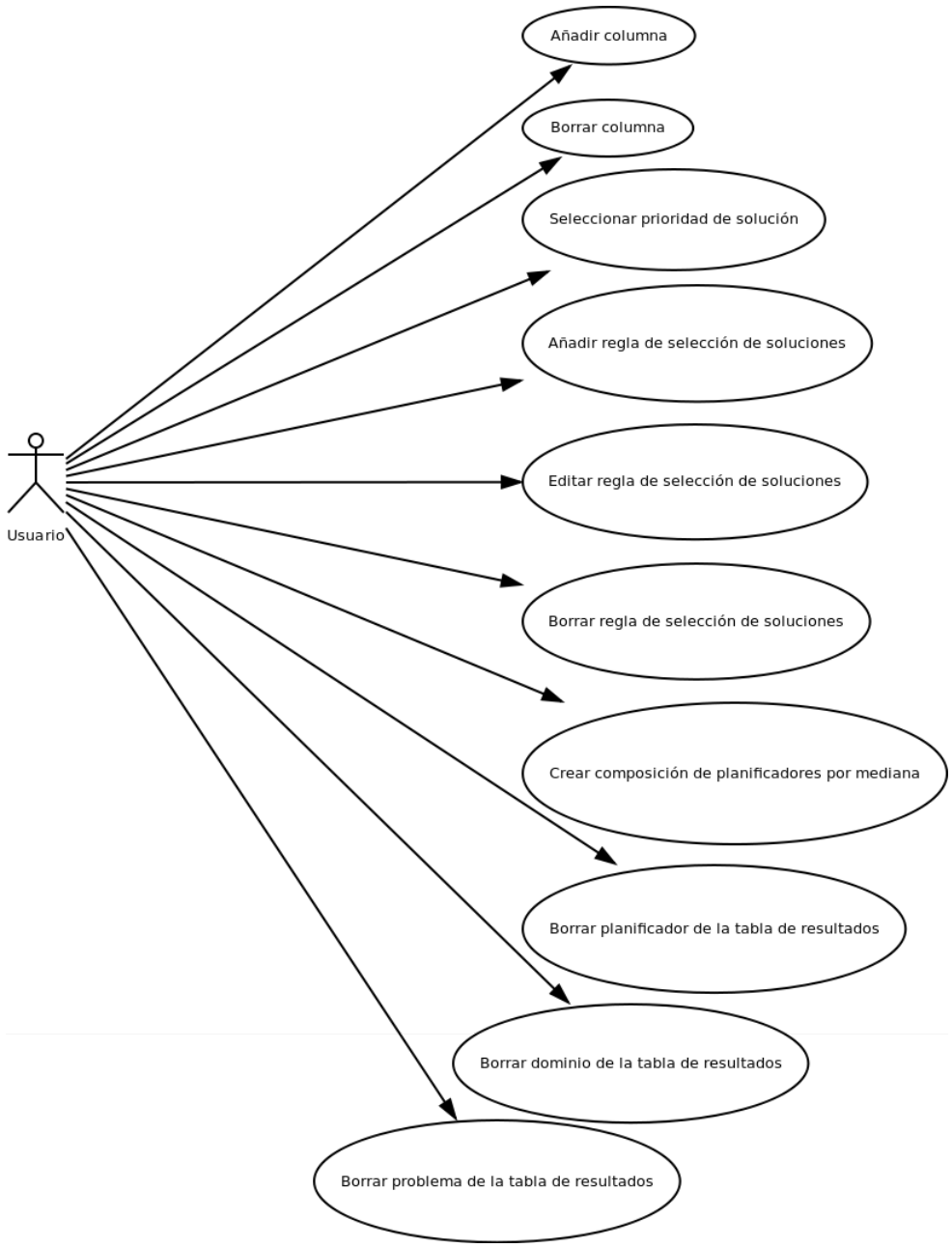

Figura 6.5: Casos de uso sobre la tabla de resultados

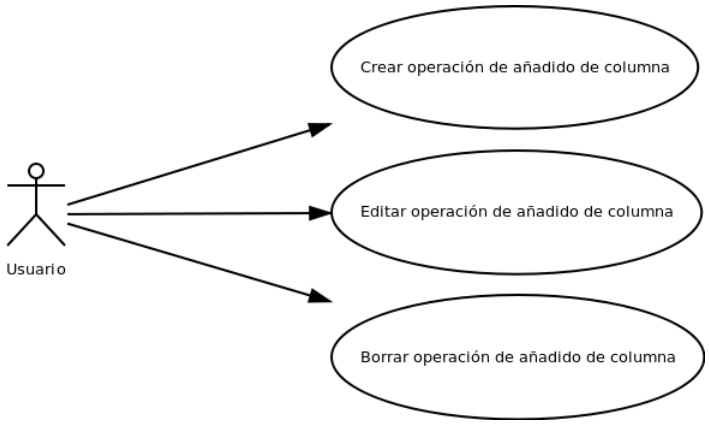

Figura 6.6: Casos de uso sobre la creación de operaciones para el añadido de columnas

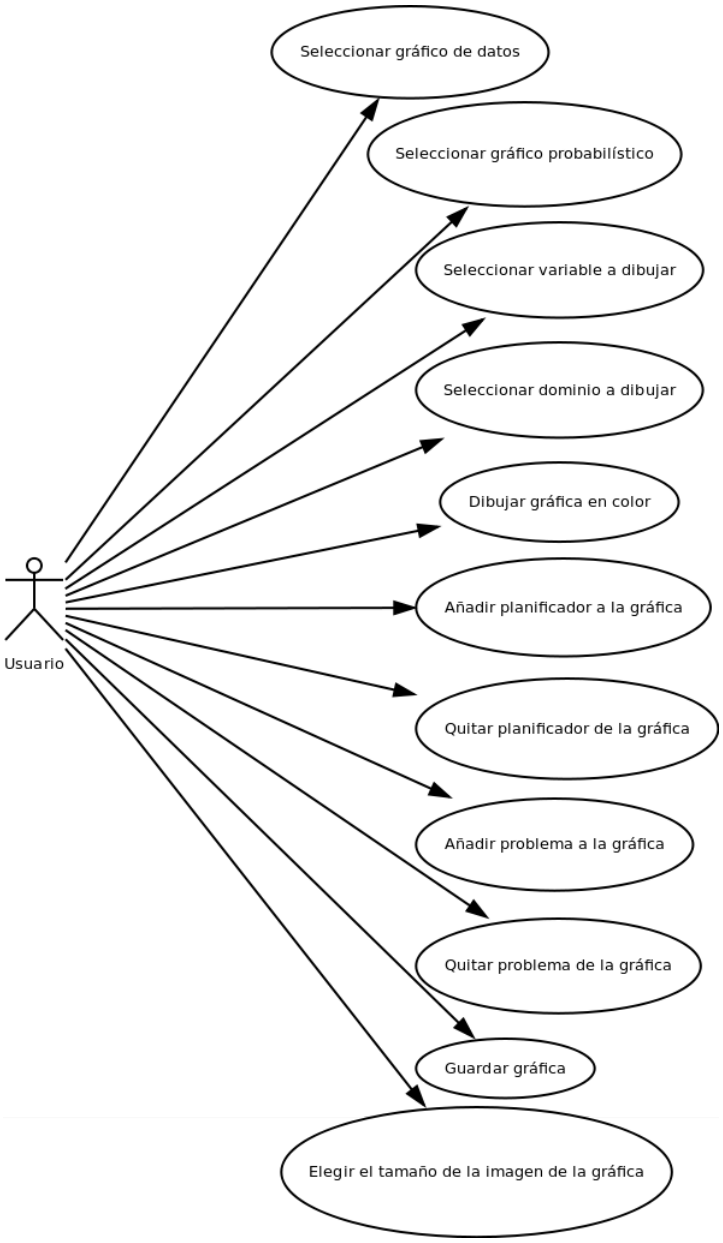

Figura 6.7: Casos de uso sobre las gráficas

Los atributos de la tabla se definen como:

- **Identificador:** Cada identificador será unívoco y se nombrará bajo el esquema CU-nnn,donde:
  - **CU:** Representa que se trata de un caso de uso.
  - **nnn:** Las letras "nnn" se remplazarán por el número de caso de uso. Estos números serán seleccionados de forma consecutiva desde el 001 hasta el 999.
- **Caso de uso:** En este campo se introducirá el nombre del caso de uso que se estudia en la tabla.
- **Objetivo:** Identifica el objetivo del caso de uso.
- **Precondiciones:** Describe las condiciones que deben darse para que se pueda proporcionar este caso de uso al usuario.
- **Postcondiciones:** Describe cómo queda el sistema después de utilizar el caso de uso.

CU-001	
Caso de uso:	Crear nuevo experimento.
Objetivo:	Se crea un nuevo experimento, sacando de la aplicación toda la información del experimento que estuviese abierto.
Precondiciones:	Es necesario que no esté abierto un resultado.
Postcondiciones:	La aplicación limpiará toda la información sobre cualquier experimento abierto. Se cargara un experimento vacío. Se marca el experimento como guardado.

Tabla 6.2: CU-001, Crear nuevo experimento.

CU-002	
Caso de uso:	Abrir experimento.
Objetivo:	Carga en la aplicación el experimento elegido por el usuario.
Precondiciones:	El usuario debe haber escogido un fichero de experimento válido.
Postcondiciones:	Se carga la información del experimento en la aplicación. Se marca el experimento como guardado.

Tabla 6.3: CU-002, Abrir experimento.

CU-003	
Caso de uso:	Guardar experimento.
Objetivo:	Guarda la información del experimento en un fichero XML con el formato que se especifica en el punto 5.8.1 de este documento.
Precondiciones:	El experimento ya tiene un nombre. El experimento ha de estar marcado como no guardado.
Postcondiciones:	Se crea un archivo XML con toda la información del experimento. Se marca el experimento como guardado.

Tabla 6.4: CU-003, Guardar experimento.

CU-004	
Caso de uso:	Guardar experimento como.
Objetivo:	Guarda la información del experimento en un fichero XML.
Precondiciones:	El usuario debe de haber escogido un nombre de experimento.
Postcondiciones:	Se crea un archivo XML con toda la información del experimento. Se marca el experimento como guardado.

Tabla 6.5: CU-004, Guardar experimento como.

CU-005	
Caso de uso:	Ejecutar experimento.
Objetivo:	Ejecuta cada planificador con cada problema de cada dominio y recoge los resultados de las variables a estudiar. La ejecución se realiza por planificador, lanzando el archivo ejecutable de éste.
Precondiciones:	Es necesario que no este abierto un resultado. El experimento al menos debe tener un planificador. Las rutas de los archivos ejecutables de los planificadores han de ser correctos. El experimento al menos debe tener un dominio. Las rutas de los archivos de los dominios han de ser correctas.
Postcondiciones:	Lanza cada ejecución de cada planificador con cada problema de cada dominio de manera secuencial. Muestra la salida de cada planificador. Recoge las variables a estudiar en la tabla de resultados Se guarda y se marca el experimento como guardado tras la ejecución de cada planificador para cada problema de cada dominio.

Tabla 6.6: CU-005, Ejecutar experimento.

CU-006	
Caso de uso:	Cambiar idioma.
Objetivo:	Cambia el idioma de la aplicación.
Precondiciones:	Es necesario que no este abierto un resultado.
Postcondiciones:	La aplicación carga el nuevo idioma.

Tabla 6.7: CU-006, Cambiar idioma.

CU-007	
Caso de uso:	Añadir dominio.
Objetivo:	Añade un dominio al experimento.
Precondiciones:	Tener seleccionado un nombre único para el dominio. Tener seleccionado el archivo del dominio. Tener seleccionado al menos un problema. Los nombres de los problemas seleccionados han de ser únicos. No tener ningún resultado abierto.
Postcondiciones:	El dominio es añadido al experimento actual. Se marca el experimento como no guardado.

Tabla 6.8: CU-007, Añadir dominio.

CU-008	
Caso de uso:	Editar dominio.
Objetivo:	Modifica uno de los dominios del experimento para permitir al usuario introducir cambios en dominios ya introducidos
Precondiciones:	Tener seleccionado un dominio del experimento actual. Tener seleccionado un nombre único para el dominio. Tener seleccionado el archivo del dominio. Tener seleccionado al menos un problema. Los nombres de los problemas seleccionados han de ser únicos. No tener ningún resultado abierto.
Postcondiciones:	El dominio es modificado en el experimento actual. Se marca el experimento como no guardado.

Tabla 6.9: CU-008, Editar dominio.

CU-009	
Caso de uso:	Borrar dominio.
Objetivo:	Elimina un dominio del experimento.
Precondiciones:	Tener seleccionado un dominio del experimento actual. No tener ningún resultado abierto.
Postcondiciones:	El dominio es eliminado del experimento actual. Se marca el experimento como no guardado.

Tabla 6.10: CU-009, Borrar dominio.

CU-010	
Caso de uso:	Añadir planificador.
Objetivo:	Introduce un planificador al experimento.
Precondiciones:	Tener seleccionado un nombre único para el planificador. Tener seleccionado un archivo ejecutable para el planificador. Tener seleccionado un comando correcto para el planificador. Tener seleccionado al menos una clave-nombre para una variable. No tener ningún resultado abierto.
Postcondiciones:	El planificador es introducido en el experimento actual. Se marca el experimento como no guardado.

Tabla 6.11: CU-010, Añadir planificador.

CU-011	
Caso de uso:	Editar planificador.
Objetivo:	Modifica uno de los planificador del experimento para permitir al usuario introducir cambios en planificadores ya introducidos.
Precondiciones:	Tener seleccionado un dominio del experimento actual. Tener seleccionado un nombre único para el planificador. Tener seleccionado un archivo ejecutable para el planificador. Tener seleccionado un comando correcto para el planificador. Tener seleccionado al menos una clave-nombre para una variable. No tener ningún resultado abierto.
Postcondiciones:	El planificador es modificado en el experimento actual. Se marca el experimento como no guardado.

Tabla 6.12: CU-011, Editar planificador.

CU-012	
Caso de uso:	Probar planificador.
Objetivo:	Prueba el comando de un planificador para comprobar que el planificador produce una salida esperada.
Precondiciones:	Tener seleccionado un archivo ejecutable para el planificador. Tener seleccionado un comando correcto para el planificador. No tener ningún resultado abierto.
Postcondiciones:	Muestra la salida del planificador para el comando introducido.

Tabla 6.13: CU-012, Probar planificador.

CU-013	
Caso de uso:	Abrir plantilla de planificador.
Objetivo:	Carga la información de una plantilla de planificador. Una plantilla de planificador contiene la información sobre la configuración de un planificador en un fichero.
Precondiciones:	Tener seleccionado un archivo con la configuración del planificador. No tener ningún resultado abierto.
Postcondiciones:	Carga en el la aplicación la información sobre la plantilla del planificador.

Tabla 6.14: CU-013, Abrir plantilla de planificador.

CU-014	
Caso de uso:	Guardar plantilla de planificador.
Objetivo:	Crea un archivo XML con la configuración del planificador. El formato del XML se especifica en el punto 5.8.2 de este documento.
Precondiciones:	Tener seleccionado un archivo ejecutable para el planificador. Tener seleccionado un comando correcto para el planificador. Tener seleccionado al menos una clave-nombre para una variable. Tener seleccionado el nombre del archivo dónde se guardará. No tener ningún resultado abierto.
Postcondiciones:	Crea un archivo XML e introduce la información sobre el planificador.

Tabla 6.15: CU-014, Guardar plantilla de planificador.

CU-015	
Caso de uso:	Borrar planificador.
Objetivo:	Elimina un planificador del experimento.
Precondiciones:	Tener seleccionado un planificador del experimento actual. No tener ningún resultado abierto.
Postcondiciones:	Elimina el planificador del experimento actual. Se marca el experimento como no guardado.

Tabla 6.16: CU-015, Borrar planificador.

CU-016	
Caso de uso:	Seleccionar tiempo máximo de CPU.
Objetivo:	Introduce en el experimento un tiempo máximo de CPU en minutos para la ejecución de cada planificador con cada problema de cada dominio.
Precondiciones:	No tener ningún resultado abierto.
Postcondiciones:	Introduce la información en el experimento. Se marca el experimento como no guardado.

Tabla 6.17: CU-016, Seleccionar tiempo máximo de CPU.

CU-017	
Caso de uso:	Seleccionar memoria máxima.
Objetivo:	Introduce en el experimento un límite de memoria máximo en KiloBytes para la ejecución de cada planificador con cada problema de cada dominio.
Precondiciones:	No tener ningún resultado abierto.
Postcondiciones:	Introduce la información en el experimento. Se marca el experimento como no guardado.

Tabla 6.18: CU-017, Seleccionar memoria máxima.

CU-018	
Caso de uso:	Omitir.
Objetivo:	Detiene la ejecución del planificador actual y comienza la del siguiente.
Precondiciones:	Un planificador debe estar ejecutándose.
Postcondiciones:	Termina la ejecución del planificador actual. Comienza la ejecución del siguiente planificador si quedan.

Tabla 6.19: CU-018, Omitir.

CU-019	
Caso de uso:	Omitir todos.
Objetivo:	Detiene la ejecución del planificador actual y termina la ejecución.
Precondiciones:	Un planificador debe estar ejecutándose.
Postcondiciones:	Termina la ejecución del experimento.

Tabla 6.20: CU-019, Omitir todos.

CU-020	
Caso de uso:	Añadir columna.
Objetivo:	Añade una columna a la tabla de resultados. Esto permite al usuario crear nuevas variables a partir de las que se han conseguido con el resultado de los planificadores
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Tener seleccionado un nombre único para la columna. Tener seleccionada una operación de añadido de columna con sus argumentos.
Postcondiciones:	A partir del código de la operación de añadido de columna agrega los valores a una columna nueva que se añade a la tabla de resultados. Se marca el experimento como no guardado.

Tabla 6.21: CU-020, Añadir columna.

CU-021	
Caso de uso:	Borrar columna.
Objetivo:	Elimina una columna de la tabla de resultados.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. La tabla de resultados ha de tener más de una columna. Tener seleccionado el nombre de una columna.
Postcondiciones:	Elimina de la tabla de resultados la columna seleccionada con todos sus valores. Se marca el experimento como no guardado.

Tabla 6.22: CU-021, Borrar columna.

CU-022	
Caso de uso:	Seleccionar prioridad de solución.
Objetivo:	A partir de una variable y a través de un operador de "mayor que" o "menor que" se decide, en caso de múltiples soluciones válidas, cuál es la que se desea seleccionar.
Precondiciones:	Debe tener un resultado abierto. Tener seleccionada una variable. Tener seleccionado un operador.
Postcondiciones:	Se resaltan las soluciones escogidas. Las soluciones escogidas son las que se usarán para dibujar las gráficas y en las operaciones de añadido de columna. Se marca el experimento como no guardado.

Tabla 6.23: CU-022, Seleccionar prioridad de solución.

CU-023	
Caso de uso:	Añadir regla de selección de soluciones.
Objetivo:	Se añade una regla que permite excluir soluciones del experimento. Estas reglas se forman a partir de una variable, un operador lógico y un valor.
Precondiciones:	Debe tener un resultado abierto. Tener seleccionada una variable. Tener seleccionado un operador ("mayor que", "mayor o igual que", "igual que", "menor o igual que" o "menor que"). Tener introducido un valor numérico.
Postcondiciones:	Todas las soluciones que no cumplan la regla serán resaltadas. Todas las soluciones que no cumplan la regla no se tendrán en cuenta en ningún caso en el experimento. Se marca el experimento como no guardado.

Tabla 6.24: CU-023, Añadir regla de selección de soluciones.

CU-024	
Caso de uso:	Editar regla de selección de soluciones.
Objetivo:	Modifica los valores de alguna regla.
Precondiciones:	Debe tener un resultado abierto. Tener seleccionada una regla. Tener seleccionada una variable. Tener seleccionado un operador ("mayor que", "mayor o igual que", "igual que", "menor o igual que" o "menor que"). Tener introducido un valor numérico.
Postcondiciones:	La regla seleccionada será modificada. Todas las soluciones que no cumplan la regla serán resaltadas. Todas las soluciones que no cumplan la regla no se tendrán en cuenta en ningún caso en el experimento. Se marca el experimento como no guardado.

Tabla 6.25: CU-024, Editar regla de selección de soluciones.

CU-025	
Caso de uso:	Borrar regla de selección de soluciones.
Objetivo:	Elimina una regla de selección de soluciones del experimento.
Precondiciones:	Debe tener un resultado abierto. Tener seleccionada una regla.
Postcondiciones:	La regla seleccionada será eliminada Se marca el experimento como no guardado.

Tabla 6.26: CU-025, Borrar regla de selección de soluciones.

CU-026	
Caso de uso:	Crear composición de planificadores por mediana.
Objetivo:	Añade a la tabla de resultados un planificador que se crea al realizar la mediana sobre una serie de planificadores seleccionados. Para realizar esta mediana se extraen los valores de una variable de dichos planificadores.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Tener seleccionado un nombre de planificador único. Tener seleccionados al menos dos planificadores sobre los que realizar la mediana. Tener una variable seleccionada de la cual se obtendrán los valores.
Postcondiciones:	Se introduce un nuevo planificador que sólo tendrá valores para la columna que representa la variable seleccionada. Estos valores son los obtenidos de realizar la mediana sobre los planificadores seleccionados. Si el número de planificadores es impar el valor resultante es el central, si son pares, el planificador resultante tendrá dos soluciones con los dos valores centrales. Se marca el experimento como no guardado.

Tabla 6.27: CU-026, Crear composición de planificadores por mediana.

CU-027	
Caso de uso:	Borrar planificador de la tabla de resultados.
Objetivo:	Se eliminan todas las apariciones de un planificador en la tabla de resultados.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Tener seleccionado un planificador.
Postcondiciones:	Se elimina el planificador de la tabla de resultados con todos sus valores. Se marca el experimento como no guardado.

Tabla 6.28: CU-027, Borrar planificador de la tabla de resultados.

CU-028	
Caso de uso:	Borrar dominio de la tabla de resultados.
Objetivo:	Se eliminan todas las apariciones de un dominio en la tabla de resultados.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Tener seleccionado un dominio.
Postcondiciones:	Se elimina el dominio de la tabla de resultados con todos sus valores. Se marca el experimento como no guardado.

Tabla 6.29: CU-028, Borrar dominio de la tabla de resultados.

CU-029	
Caso de uso:	Borrar problema de la tabla de resultados.
Objetivo:	Se eliminan todas las apariciones de un problema en la tabla de resultados.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Tener seleccionado un dominio. Tener seleccionado un problema.
Postcondiciones:	Se elimina el problema de la tabla de resultados con todos sus valores. Se marca el experimento como no guardado.

Tabla 6.30: CU-029, Borrar problema de la tabla de resultados.

CU-030	
Caso de uso:	Crear operación de añadido de columna.
Objetivo:	Crea una operación a partir de código java que permite el rellenado de las nuevas columnas en tiempo de ejecución.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Debe tener un nombre válido de clase java que sea único. Debe tener un código sin errores de compilación de java.
Postcondiciones:	Crea una carpeta con un archivo con un archivo que contiene la información de la operación y un .jar con el código java compilado.

Tabla 6.31: CU-030, Crear operación de añadido de columna.

CU-031	
Caso de uso:	Editar operación de añadido de columna.
Objetivo:	Modifica una operación de añadido de columna que previamente ha creado el usuario.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Debe tener una operación seleccionada. Debe tener un nombre válido de clase java que sea único. Debe tener un código sin errores de compilación de java.
Postcondiciones:	Modifica el archivo con la información de la operación y crea un .jar con el código java compilado eliminando el anterior.

Tabla 6.32: CU-031, Editar operación de añadido de columna.

CU-032	
Caso de uso:	Borrar operación de añadido de columna.
Objetivo:	Elimina del sistema una operación de añadido de columna.
Precondiciones:	La ejecución del experimento ha de estar terminada. Debe tener un resultado abierto. Debe tener una operación seleccionada.
Postcondiciones:	Elimina la carpeta de la operación con todo su contenido.

Tabla 6.33: CU-032, Borrar operación de añadido de columna.

CU-033	
Caso de uso:	Seleccionar gráfico de datos.
Objetivo:	Muestra un gráfico con los planificadores sobre una variable y un dominio.
Precondiciones:	Debe tener un resultado abierto.
Postcondiciones:	Muestra la gráfica al usuario.

Tabla 6.34: CU-033, Seleccionar gráfico de datos.

CU-034	
Caso de uso:	Seleccionar gráfico probabilístico.
Objetivo:	Crea una gráfica a partir de una variable y un dominio. El eje "y" de esta gráfica muestra el porcentaje de problemas resueltos y el eje "x" los valores de la variable en orden creciente pudiendo representar en qué valor de la variable cada planificador ha resuelto un problema.
Precondiciones:	Debe tener un resultado abierto.
Postcondiciones:	Muestra la gráfica al usuario.

Tabla 6.35: CU-034, Seleccionar gráfico probabilístico.

CU-035	
Caso de uso:	Seleccionar variable a dibujar.
Objetivo:	Permite cambiar la variable que se dibuja en el gráfico.
Precondiciones:	Debe tener un resultado abierto.
Postcondiciones:	Dibuja la gráfica con la nueva variable.

Tabla 6.36: CU-035, Seleccionar variable a dibujar.

CU-036	
Caso de uso:	Seleccionar dominio a dibujar.
Objetivo:	Permite cambiar el dominio que se dibuja en el gráfico.
Precondiciones:	Debe tener un resultado abierto.
Postcondiciones:	Dibuja la gráfica con el nuevo dominio.

Tabla 6.37: CU-036, Seleccionar dominio a dibujar.

CU-037	
Caso de uso:	Dibujar gráfica en color.
Objetivo:	Permite seleccionar si la gráfica se va a mostrar en color o en escala de grises.
Precondiciones:	Debe tener un resultado abierto.
Postcondiciones:	Dibuja la gráfica con el color seleccionado.

Tabla 6.38: CU-037, Dibujar gráfica en color.

CU-038	
Caso de uso:	Añadir planificador a la gráfica.
Objetivo:	Permite añadir planificadores a la gráfica mostrada cuando no están todos incluidos.
Precondiciones:	Debe tener un resultado abierto. No deben estar incluidos todos los planificadores.
Postcondiciones:	Redibuja la gráfica con el planificador incluido.

Tabla 6.39: CU-038, Añadir planificador a la gráfica.

CU-039	
Caso de uso:	Quitar planificador de la gráfica.
Objetivo:	Permite eliminar planificadores de la gráfica mostrada.
Precondiciones:	Debe tener un resultado abierto. Debe tener como mínimo un planificador
Postcondiciones:	Redibuja la gráfica sin el planificador.

Tabla 6.40: CU-039, Quitar planificador de la gráfica.

CU-040	
Caso de uso:	Añadir problema a la gráfica.
Objetivo:	Permite añadir problemas a la gráfica mostrada cuando no están todos incluidos.
Precondiciones:	Debe tener un resultado abierto. No deben estar incluidos todos los problemas.
Postcondiciones:	Redibuja la gráfica con el problema incluido.

Tabla 6.41: CU-040, Añadir problema a la gráfica.

CU-041	
Caso de uso:	Quitar problema de la gráfica.
Objetivo:	Permite eliminar problemas de la gráfica mostrada.
Precondiciones:	Debe tener un resultado abierto. Debe tener como mínimo un problema.
Postcondiciones:	Redibuja la gráfica sin el problema.

Tabla 6.42: CU-041, Quitar problema de la gráfica.

CU-042	
Caso de uso:	Guardar gráfica.
Objetivo:	Crea un archivo png o jpeg con la imagen de la gráfica mostrada.
Precondiciones:	Debe tener un resultado abierto. Debe tener un nombre para la imagen seleccionado.
Postcondiciones:	Crea el archivo de la imagen de la gráfica.

Tabla 6.43: CU-042, Guardar gráfica.

CU-043	
Caso de uso:	Elegir el tamaño de la imagen de la gráfica.
Objetivo:	Permite indicar el tamaño de la imagen de la gráfica a generar.
Precondiciones:	Debe tener un resultado abierto. Debe tener seleccionado un ancho en píxeles. Debe tener seleccionado un alto en píxeles.
Postcondiciones:	Guarda la información sobre el tamaño de la imagen para posteriormente generarla.

Tabla 6.44: CU-043, Elegir el tamaño de la imagen de la gráfica.

6.1.2. Descripción de requisitos software

En este punto se van a describir todos los requisitos software, para ello se empleará la plantilla que se muestra en la tabla 6.45. El campo identificador será único para cada requisito y se nombrarán bajo el esquema SRx-nnn, dónde:

- **SR:** Representa que se trata de un requisito de software y aparecerá en todos los requisitos de este tipo.
- **x:** La letra "x" se remplazara dependiendo de cada tipo de requisito:
  - **F:** Representa un requisito funcional.
  - **R:** Representa un requisito de rendimiento.
  - **I:** Representa un requisito de interfaz.
  - **O:** Representa un requisito de operación.
  - **Re:** Representa un requisito de recursos.
  - **C:** Representa un requisito de comprobación.
  - **A:** Representa un requisito de aceptación.
  - **D:** Representa un requisito de documentación.
  - **M:** Representa un requisito de mantenimiento.
  - **P:** Representa un requisito de portabilidad.
- **nnn:** Las letras "nnn" se remplazarán por el número de requisito. Estos números serán seleccionados de forma consecutiva desde el 001 hasta el 999.

6.1.2.1. Requisitos funcionales

En los requisitos funcionales se especifica cuáles son las funciones que va a poder realizar el software desarrollado. Así pues, en este punto quedará definida la funcionalidad de la herramienta PLEX.

Identificador	
Nombre	
Descripción:	

Tabla 6.45: Plantilla requisitos software

SRF-001	
Crear un nuevo experimento	
Descripción:	Un usuario de PLEX ha de poder crear nuevos experimentos.

Tabla 6.46: SRF-001, Crear un nuevo experimento

SRF-002	
Abrir un experimento	
Descripción:	Un usuario de PLEX ha de poder abrir experimentos cuyos datos se encuentren en XML en un determinado formato.

Tabla 6.47: SRF-002, Abrir un experimento

SRF-003	
Guardar un experimento	
Descripción:	Un usuario de PLEX ha de poder guardar un experimento ya sea cuando sólo se ha configurado o cuando ya se haya ejecutado. Los datos se guardarán en XML con un formato determinado.

Tabla 6.48: SRF-003, Guardar un experimento

SRF-004	
Guardar un experimento como	
Descripción:	Un usuario de PLEX ha de poder guardar un experimento con el nombre que el usuario elija ya sea cuando sólo se ha configurado o cuando ya se haya ejecutado. Los datos se guardarán en XML con un formato determinado.

Tabla 6.49: SRF-004, Guardar un experimento como

SRF-005	
Ejecutar experimento	
Descripción:	Un usuario de PLEX ha de poder ejecutar un experimento cuya configuración sea correcta para poder analizar sus resultados.

Tabla 6.50: SRF-005, Ejecutar experimento

SRF-006	
Añadir dominios a un experimento	
Descripción:	Un usuario de PLEX ha de poder añadir uno o varios dominios a un experimento no ejecutado. Cada dominio ha de tener un nombre único, uno o más archivos de problemas y un archivo de dominio. Los archivos de dominio y problemas pueden estar en cualquier formato aceptado por los planificadores que el usuario quiera usar en el experimento.

Tabla 6.51: SRF-006, Añadir dominios a un experimento

SRF-007	
Editar dominios de un experimento	
Descripción:	Un usuario de PLEX ha de poder editar cualquier dominio que se haya introducido anteriormente en un experimento no ejecutado. Una vez editado, el dominio ha de tener un nombre único, uno o más archivos de problemas y un archivo de dominio. Los archivos de dominio y problemas pueden estar en cualquier formato aceptado por los planificadores que el usuario quiera usar en el experimento

Tabla 6.52: SRF-007, Editar dominios de un experimento

SRF-008	
Eliminar dominios de un experimento	
Descripción:	Un usuario de PLEX ha de poder eliminar cualquiera de los dominios introducidos en un experimento no ejecutado

Tabla 6.53: SRF-008, Eliminar dominios de un experimento

SRF-009	
Añadir planificadores a un experimento	
Descripción:	Un usuario de PLEX ha de poder añadir uno o varios planificadores a un experimento no ejecutado. Cada planificador ha de tener un nombre único, un ejecutable, una línea de comandos,una o más claves para obtener las variables que se quieren estudiar y, opcionalmente, una clave capaz de distinguir las múltiples soluciones que un planificador ha de tener. Estas claves son porciones únicas de texto de la salida del planificador.

Tabla 6.54: SRF-009, Añadir planificadores a un experimento

SRF-010	
Editar planificadores de un experimento	
Descripción:	Un usuario de PLEX ha de poder editar los planificadores que tenga un experimento no ejecutado. Una vez editado, el planificador ha de tener un nombre único, un ejecutable, una línea de comandos,una o más claves para obtener las variables que se quieren estudiar y opcionalmente una clave capaz de distinguir las múltiples soluciones que un planificador ha de tener. Estas claves son porciones únicas de texto de la salida del planificador.

Tabla 6.55: SRF-010, Editar planificadores de un experimento

SRF-011	
Borrar planificadores de un experimento	
Descripción:	Un usuario de PLEX ha de poder borrar cualquier planificador de un experimento que aún no ha sido ejecutado.

Tabla 6.56: SRF-011, Borrar planificadores de un experimento

SRF-012	
Guardar plantilla de planificador	
Descripción:	Un usuario de PLEX ha de poder almacenar cualquier configuración de un planificador para su posterior uso en un formato XML determinado. Para poder almacenarlo ha de tener una configuración correcta.

Tabla 6.57: SRF-012, Guardar plantilla de planificador

SRF-013	
Abrir plantilla de planificador	
Descripción:	Un usuario de PLEX ha de poder abrir configuraciones de planificadores cuyos datos estarán en ficheros XML con un formato determinado.

Tabla 6.58: SRF-013, Abrir plantilla de planificador

SRF-014	
Probar el comando del planificador	
Descripción:	Un usuario de PLEX ha de poder probar el comando del planificador. Está funcionalidad es necesaria para que el usuario pueda comprobar que los argumentos que quiere introducir en el planificador son correctos y producen el efecto deseado.

Tabla 6.59: SRF-014, Probar el comando del planificador

SRF-015	
Limitar el tiempo de CPU	
Descripción:	Un usuario de PLEX ha de poder introducir el tiempo máximo que un planificador puede estar ejecutándose en la CPU. Para ello la herramienta usa el comando de Linux ulimit.

Tabla 6.60: SRF-015, Limitar el tiempo de CPU

SRF-016	
Limitar la memoria	
Descripción:	Un usuario de PLEX ha de poder introducir la memoria máxima que un planificador puede usar mientras se ejecuta. Para ello la herramienta usa el comando de Linux ulimit.

Tabla 6.61: SRF-016, Limitar la memoria

SRF-017	
Omitir la resolución de un problema por un planificador	
Descripción:	Un usuario de PLEX ha de poder terminar la ejecución de un planificador que intenta resolver un problema, permitiendo que se ejecute el siguiente.

Tabla 6.62: SRF-017, Omitir la resolución de un problema por un planificador

SRF-018	
Omitir todos los planificadores	
Descripción:	Un usuario de PLEX ha de poder terminar la ejecución de todos los planificadores mientras se están ejecutando, impidiendo que se ejecuten los demás.

Tabla 6.63: SRF-018, Omitir todos los planificador

SRF-019	
Añadir reglas de selección de soluciones	
Descripción:	Un usuario de PLEX ha de poder añadir reglas para la selección de soluciones en la tabla de resultados.

Tabla 6.64: SRF-019, Añadir reglas de selección de soluciones

SRF-020	
Editar reglas de selección de soluciones	
Descripción:	Un usuario de PLEX podrá editar las reglas para la selección de soluciones en la tabla de resultados.

Tabla 6.65: SRF-020, Editar reglas de selección de soluciones

SRF-021	
Eliminar reglas de selección de soluciones	
Descripción:	Un usuario de PLEX podrá eliminar las reglas para la selección de soluciones en la tabla de resultados.

Tabla 6.66: SRF-021, Eliminar reglas de selección de soluciones

SRF-022	
Elegir una solución entre múltiples	
Descripción:	Un usuario de PLEX ha de seleccionar una solución en caso de que un planificador devuelva varias. Para ello podrá elegir qué variable será la diferenciadora y si prefiere escoger la mayor o la menor. La solución escogida será la usada en las posteriores operaciones que el usuario requiera ya sea sobre los datos o sobre las gráficas.

Tabla 6.67: SRF-022, Elegir una solución entre múltiples

SRF-023	
Borrar un planificador de la solución	
Descripción:	Un usuario de PLEX ha de poder eliminar un planificador de la solución del experimento. Como mínimo ha de quedar siempre un planificador.

Tabla 6.68: SRF-023, Borrar un planificador de la solución

SRF-024	
Borrar un dominio de la solución	
Descripción:	Un usuario de PLEX ha de poder eliminar un dominio de la solución del experimento. Como mínimo ha de quedar siempre un dominio.

Tabla 6.69: SRF-024, Borrar un dominio de la solución

SRF-025	
Borrar un problema de la solución	
Descripción:	Un usuario de PLEX ha de poder eliminar un problema de la solución del experimento. Como mínimo ha de quedar siempre un problema.

Tabla 6.70: SRF-025, Borrar un problema de la solución

SRF-026	
Crear composiciones de planificadores por mediana	
Descripción:	Un usuario de PLEX ha de poder seleccionar una serie de planificadores, elegir una variable y, dando un nombre de planificador único, crear un nuevo resultado de un planificador para cada problema y dominio dónde el valor de la variable escogida sea la mediana de los valores que tenía esa variable en cada resultado. Si el número de valores para escoger la mediana es impar el valor escogido será el intermedio. En caso de que los valores sean pares, los escogidos serán los dos centrales y se mostrará en la tabla como si fuera un planificador con varias soluciones.

Tabla 6.71: SRF-026, Crear composiciones de planificadores por mediana

SRF-027	
Añadir columna	
Descripción:	Un usuario de PLEX ha de poder añadir columnas a la tabla de resultados con nuevos datos. Para crear la columna se ha de introducir un nombre de columna único, una operación para obtener los datos, y los parámetros que requiera la operación.

Tabla 6.72: SRF-027, Añadir columna

SRF-028	
Eliminar Columna	
Descripción:	Un usuario de PLEX ha de poder eliminar columnas de la tabla de resultados. Como mínimo siempre ha de quedar una columna.

Tabla 6.73: SRF-028, Eliminar Columna

SRF-029	
Crear una operación para añadir columnas	
Descripción:	Un usuario de PLEX ha de poder crear una operación que tenga como resultado una columna de la tabla. Esta operación se compondrá de un nombre y un código java que será el encargado de crear la columna. Opcionalmente se podrá introducir una descripción

Tabla 6.74: SRF-029, Crear una operación para añadir columnas

SRF-030	
Editar una operación para añadir columnas	
Descripción:	Un usuario de PLEX ha de poder editar una operación de añadido de columnas. Así pues, podrá cambiar el nombre, código o descripción de ésta.

Tabla 6.75: SRF-030, Crear una operación para añadir columnas

SRF-031	
Borrar una operación para añadir columnas	
Descripción:	Un usuario de PLEX ha de poder borrar una operación de añadido de columnas desapareciendo ésta de la herramienta

Tabla 6.76: SRF-031, Crear una operación para añadir columnas

SRF-032	
Obtener una gráfica con los datos de la tabla de resultados	
Descripción:	Un usuario de PLEX ha de poder obtener una gráfica por dominio para cada variable seleccionada en cada planificador en la que se muestren los resultados obtenidos para cada problema por cada planificador sobre esa variable.

Tabla 6.77: SRF-032, Obtener una gráfica con los datos de la tabla de resultados

SRF-033	
Obtener una gráfica probabilística con los datos de la tabla de resultados	
Descripción:	Un usuario de PLEX ha de poder obtener una gráfica probabilística por dominio para cada variable seleccionada en cada planificador. En esta gráfica se mostrará el porcentaje de problemas resueltos según crece la variable escogida (desde 0 hasta el valor máximo obtenido).

Tabla 6.78: SRF-033, Obtener una gráfica probabilística con los datos de la tabla de resultados

SRF-034	
Guardar gráficas	
Descripción:	Un usuario de PLEX ha de poder guardar cualquiera de las gráficas generadas en formato jpeg o png. Además ha de poder decidir qué tamaño tendrá la imagen guardada.

Tabla 6.79: SRF-034, Guardar gráficas

SRF-035	
Seleccionar variable a dibujar en la gráfica	
Descripción:	Un usuario de PLEX ha de poder elegir qué variable quiere que se represente en la gráfica a partir de los datos de la tabla.

Tabla 6.80: SRF-035, Seleccionar variable a dibujar en la gráfica

SRF-036	
Seleccionar dominio a dibujar en la gráfica	
Descripción:	Un usuario de PLEX ha de poder elegir qué dominio quiere que se represente en la gráfica a partir de los datos de la tabla.

Tabla 6.81: SRF-036, Seleccionar dominio a dibujar en la gráfica

SRF-037	
Añadir planificadores a la gráfica	
Descripción:	Un usuario de PLEX ha de poder añadir a la gráfica cualquier planificador que se encuentre en la tabla de resultados.

Tabla 6.82: SRF-037, Añadir planificadores a la gráfica

SRF-038	
Quitar planificadores de la gráfica	
Descripción:	Un usuario de PLEX ha de poder quitar de la gráfica cualquier planificador que se encuentre en ésta.

Tabla 6.83: SRF-038, Quitar planificadores de la gráfica

SRF-039	
Añadir problemas a la gráfica	
Descripción:	Un usuario de PLEX ha de poder añadir a la gráfica cualquier problema que se encuentre en la tabla de resultados para el dominio seleccionado.

Tabla 6.84: SRF-039, Añadir problemas a la gráfica

6.1.2.2. Requisitos de rendimiento

Los requisitos de rendimiento especifican los valores que han de poder tomar ciertas variables medibles.

SRF-040	
Quitar problemas de la gráfica	
Descripción:	Un usuario de PLEX podrá elegir generar la gráfica en color o en escala de grises.

Tabla 6.85: SRF-040, Quitar problemas de la gráfica

SRF-041	
Seleccionar la gráfica en color o en escala de grises	
Descripción:	Un usuario de PLEX ha seleccionar si quiere que la gráfica se genere en color o en escala de grises.

Tabla 6.86: SRF-041, Cambiar idioma

SRF-042	
Cambiar idioma	
Descripción:	Un usuario de PLEX ha de poder cambiar el idioma de la herramienta.

Tabla 6.87: SRF-042, Cambiar idioma

SRF-043	
Seleccionar el tamaño de la imagen de la gráfica a guardar	
Descripción:	Un usuario de PLEX ha de poder elegir el tamaño en píxeles de la imagen de la gráfica cuando se disponga a guardarla.

Tabla 6.88: SRF-043, Seleccionar el tamaño de la imagen de la gráfica a guardar

SRR-001	
Número ilimitado de dominios y problemas	
Descripción:	Un usuario de PLEX ha de poder introducir todos los dominios y problemas que crea necesario. No existe un limite máximo.

Tabla 6.89: SRR-001, Número ilimitado de dominios y problemas

6.1.2.3. Requisitos de interfaz

Especifican el hardware, software o elementos de bases de datos con los que el sistema debe interactuar o comunicarse. Pueden incluir sistemas operativos, ambientes de software, formatos de archivos, sistemas administradores de bases de datos, sistemas de gestión y otras aplicaciones de software, además de especificar configuraciones de hardware.

SRR-002	
Número ilimitado de planificadores	
Descripción:	Un usuario de PLEX ha de poder introducir todos los planificadores que crea necesario. No existe un limite máximo.

Tabla 6.90: SRR-002, Número ilimitado de planificadores

SRR-003	
Número ilimitado de operaciones de añadido de columna	
Descripción:	Un usuario de PLEX ha de poder usar y crear todas las operaciones de añadido de columna que crea necesarias. No existe un limite máximo.

Tabla 6.91: SRR-003, Número ilimitado de operaciones de añadido de columna

SRI-001	
Sistema operativo	
Descripción:	La herramienta PLEX ha de funcionar correctamente en un sistema operativo Linux que implemente las llamadas al sistema ulimit y kill.

Tabla 6.92: SRI-001, Tiempo máximo de CPU con ulimit

SRI-002	
Tiempo máximo de CPU con ulimit	
Descripción:	Para controlar el tiempo máximo de CPU que puede consumir un planificador se ha de usar el comando ulimit de Linux.

Tabla 6.93: SRI-002, Tiempo máximo de CPU con ulimit

SRI-003	
Gasto de memoria máximo con ulimit	
Descripción:	Para controlar el gasto máximo de memoria que puede consumir un planificador se ha de usar el comando ulimit de Linux.

Tabla 6.94: SRI-003, Gasto de memoria máximo con ulimit

SRI-004	
Detener ejecución de planificadores con kill	
Descripción:	Para detener la ejecución de un planificador se ha de usar el comando kill de Linux. Además de matar al proceso del planificador se han de matar todos los procesos que éste haya lanzado en el sistema.

Tabla 6.95: SRI-004, Detener ejecución de planificadores con kill

SRI-005	
Experimentos en XML	
Descripción:	Los experimentos se almacenarán en XML con el formato especificado por el DTD llamado experimenter.dtd en la carpeta dtds del proyecto.

Tabla 6.96: SRI-005, Experimentos en XML

SRI-006	
Plantillas de planificador en XML	
Descripción:	Las plantillas de los planificadores se almacenarán en XML con el formato especificado por el DTD llamado planner.dtd en la carpeta dtds del proyecto.

Tabla 6.97: SRI-006, Plantillas de planificador en XML

SRI-007	
Guardado automático durante la ejecución	
Descripción:	La herramienta PLEX ha de guardar automáticamente el estado del experimento después de la ejecución de un planificador para un problema y dominio determinado, con la finalidad de no perder datos por una parada inesperada de la ejecución. La herramienta ha de almacenar todas las variables necesarias para recuperar la ejecución en el punto en el que se interrumpió.

Tabla 6.98: SRI-007, Guardado automático durante la ejecución

6.1.2.4. Requisitos de operación

Especifican cómo va a ejecutar el sistema, y cómo se va a comunicar con los operadores humanos. Incluyen todos los requisitos de interfaces de usuario.

SRI-008	
Gráficas en jpeg	
Descripción:	La herramienta PLEX ha de permitir guardar en formato jpeg las gráficas generadas a partir de la tabla de datos.

Tabla 6.99: SRI-008, Gráficas en jpeg

SRI-009	
Gráficas en png	
Descripción:	La herramienta PLEX ha de permitir guardar en formato png las gráficas generadas a partir de la tabla de datos.

Tabla 6.100: SRI-009, Gráficas en png

SRI-010	
Java 1.6	
Descripción:	Para que la aplicación pueda funcionar correctamente el sistema dónde se ejecute ha de tener instalada la máquina virtual y el compilador de java versión 1.6

Tabla 6.101: SRI-010, Java 1.6

SRI-011	
Desarrollo en Java 1.6	
Descripción:	La herramienta PLEX ha de ser desarrollada en lenguaje Java versión 1.6

Tabla 6.102: SRI-011, Desarrollo en Java 1.6

SRO-001	
Mensajes de error	
Descripción:	La herramienta PLEX ha de generar un mensaje de error siempre que aparezca alguno con la finalidad de informar al usuario.

Tabla 6.103: SRO-001, Mensajes de error

SRO-002	
Interfaz del experimentador	
Descripción:	En la figura 6.8 se observa como ha de ser la interfaz de usuario del experimentador. Desde ella se ha de poder crear, abrir, guardar, ejecutar un experimento y modificar el idioma. Además se ha de poder configurar añadiendo, editando o eliminando tanto dominios como planificadores. En esta interfaz también se debe poder introducir los límites de tiempo de CPU y de memoria usada.

Tabla 6.104: SRO-002, Interfaz del experimentador

SRO-003	
Interfaz de selección de archivos XML	
Descripción:	En la figura 6.9 se observa cómo ha de ser la interfaz de usuario que nos permite escoger un fichero XML, ya sea para abrir o guardar un experimento como una plantilla de planificador.

Tabla 6.105: SRO-003, Interfaz de selección de archivos XML

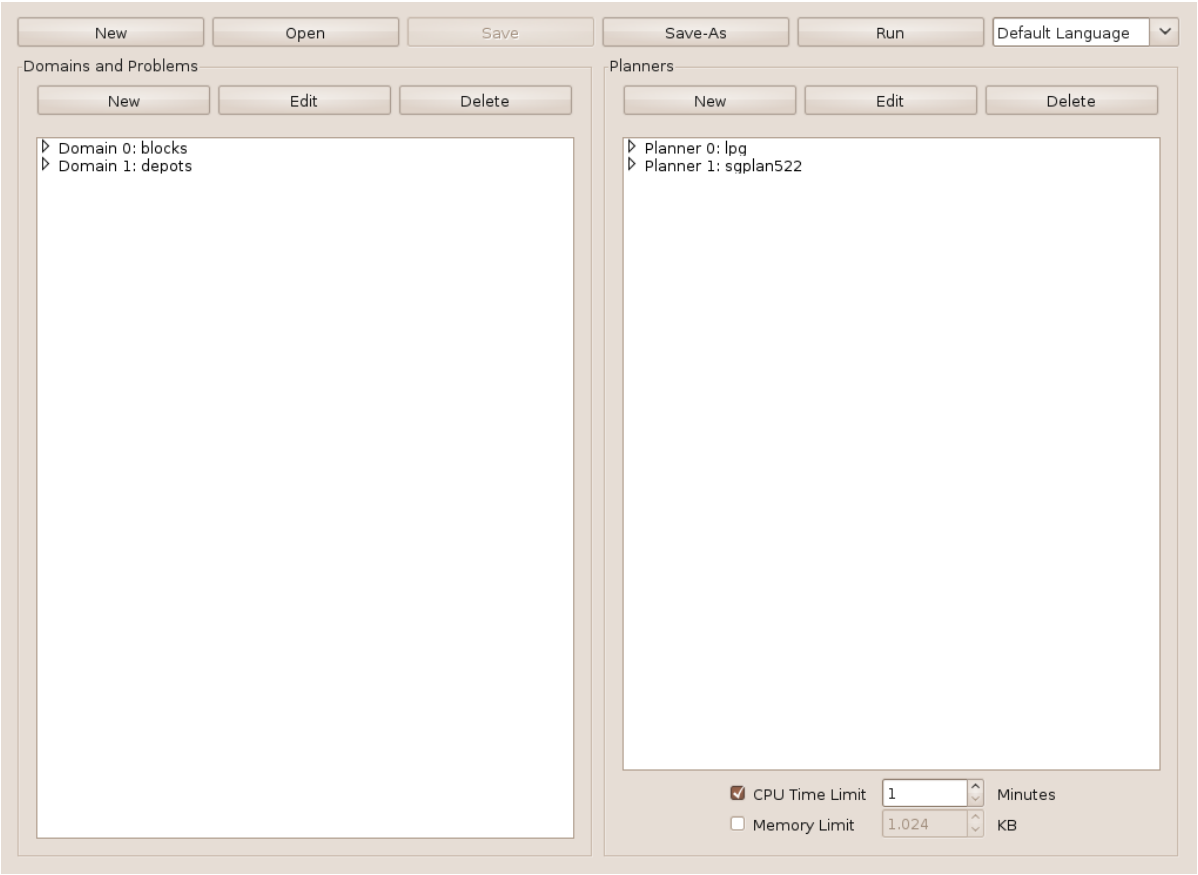

Figura 6.8: Interfaz del experimentador

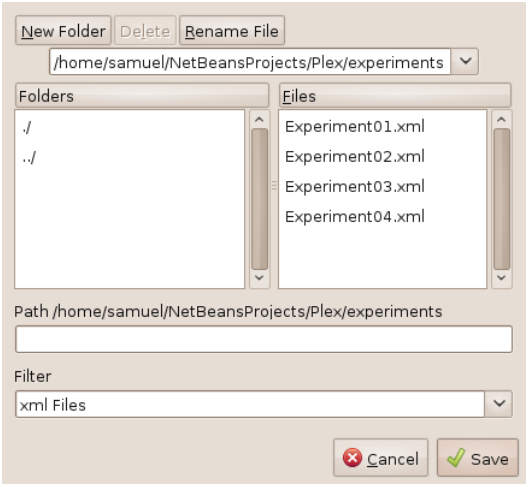

Figura 6.9: Interfaz de selección de archivos XML

SRO-004	
Interfaz de añadido y editado de dominios	
Descripción:	En la figura 6.10 se observa como ha de ser la interfaz de usuario que nos permite añadir o editar dominios en un experimento. La interfaz ha de permitir introducir un nombre para el dominio, seleccionar el archivo donde está definido, así como los archivos de problemas que el usuario crea oportuno.

Tabla 6.106: SRO-004, Interfaz de añadido y editado de dominios

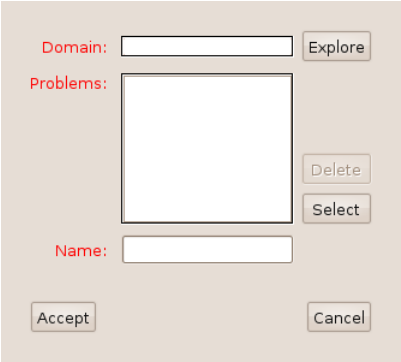

Figura 6.10: Interfaz de añadido y editado de dominios

SRO-005	
Interfaz de selección de archivos con cualquier extensión	
Descripción:	En la figura 6.11 se observa cómo ha de ser la interfaz de usuario que nos permite escoger un fichero sin importar su extensión, ya sea para seleccionar un dominio o un problema.

Tabla 6.107: SRO-005, Interfaz de selección de archivos con cualquier extensión

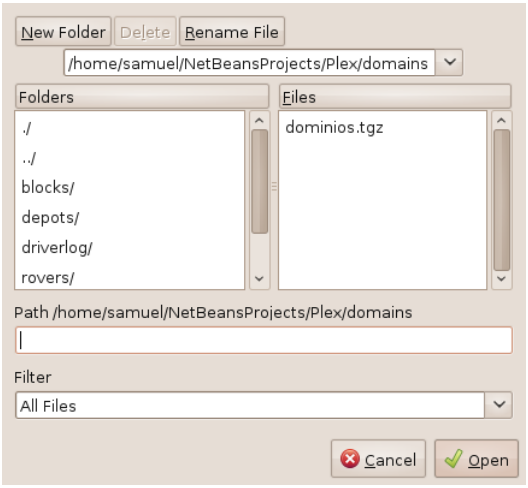

Figura 6.11: Interfaz de selección de archivos con cualquier extensión

SRO-006	
Interfaz de añadido y editado de planificadores	
Descripción:	En la figura 6.12 se observa cómo ha de ser la interfaz de usuario que nos permite añadir o editar planificadores en un experimento. La interfaz nos ha de permitir introducir un nombre para el planificador y seleccionar el archivo dónde esta definido el comando con el que se ejecutará el planificador. Además, se ha de poder introducir las variables que el usuario quiere obtener del planificador a través de unas claves que pueden encontrarse delante o detrás del valor a obtener. Por último, se ha de permitir diferenciar varias soluciones si el usuario lo requiere. En la parte derecha de la interfaz aparece un campo de texto que se tiene que rellenar con la salida de la ejecución del planificador sin parámetros, cuando se escoge su ejecutable, y con la salida que se obtiene cuando se realiza el test del comando.

Tabla 6.108: SRO-006, Interfaz de añadido y editado de planificadores

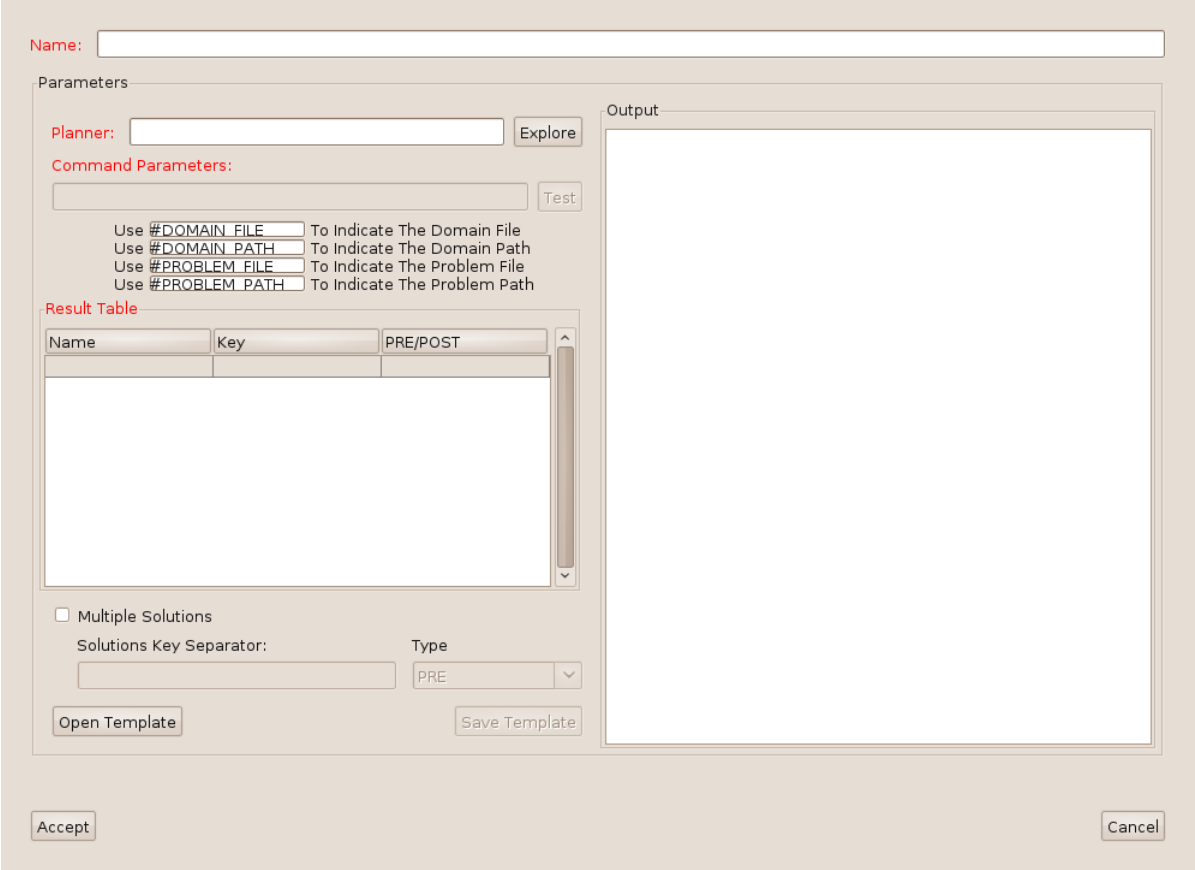

Figura 6.12: Interfaz de añadido y editado de planificadores

SRO-007	
Interfaz de selección de archivos ejecutables	
Descripción:	En la figura 6.13 se observa como ha de ser la interfaz de usuario que nos permite escoger un fichero sea ejecutable. Se usa para seleccionar el ejecutable de un planificador.

Tabla 6.109: SRO-007, Interfaz de selección de archivos ejecutables

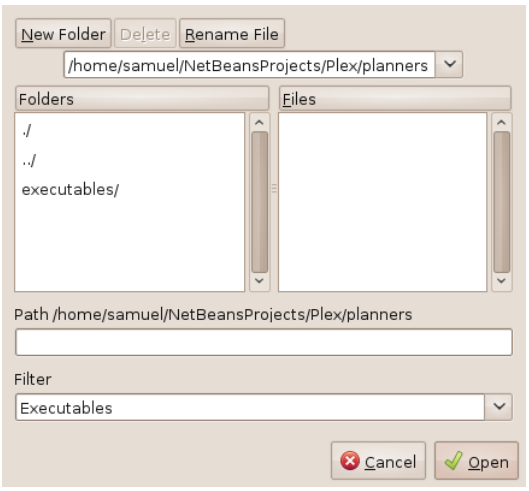

Figura 6.13: Interfaz de selección de archivos ejecutables

SRO-008	
Interfaz de resultado con la salida de los planificadores	
Descripción:	En la figura 6.14 se observa cómo ha de ser la interfaz de usuario qué nos permite observar que salida están produciendo los planificadores. Desde esta interfaz se ha de poder permitir detener la ejecución de un planificador para un problema de un dominio determinado o la de todos los planificadores.

Tabla 6.110: SRO-008, Interfaz de resultado con la salida de los planificadores

SRO-009	
Interfaz de resultado con la tabla de variables	
Descripción:	En la figura 6.15 se observa cómo ha de ser la interfaz de usuario que nos permite observar las variables que se han obtenido de la ejecución de los planificadores. Desde esta interfaz se ha de poder permitir detener la ejecución de un planificador para un problema de un dominio determinado o la de todos los planificadores, así como seleccionar el criterio adecuado para elegir una solución cuando un planificador a obtenido varias e introducir criterios para la aceptación de soluciones. Esta interfaz también nos permite modificar la tabla añadiendo columnas, eliminando dominios, planificadores o problemas, y permitiendo la introducción de composición de planificadores por mediana.

Tabla 6.111: SRO-009, Interfaz de resultado con la tabla de variables

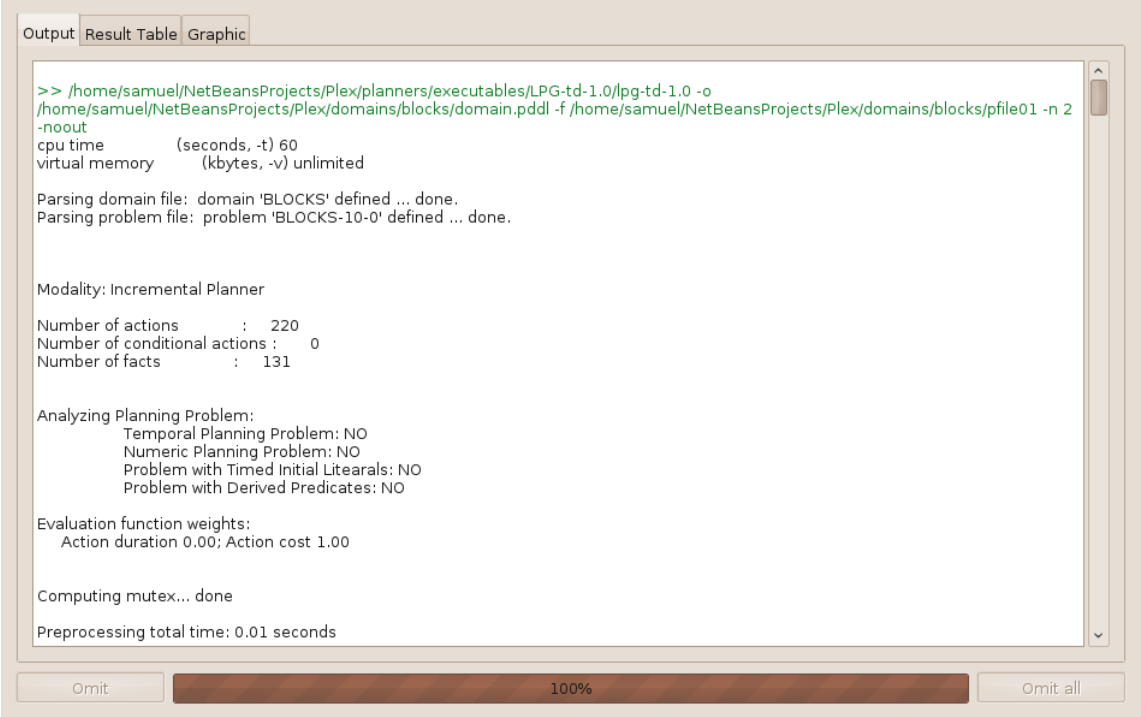

Figura 6.14: Interfaz de resultado con la salida de los planificadores

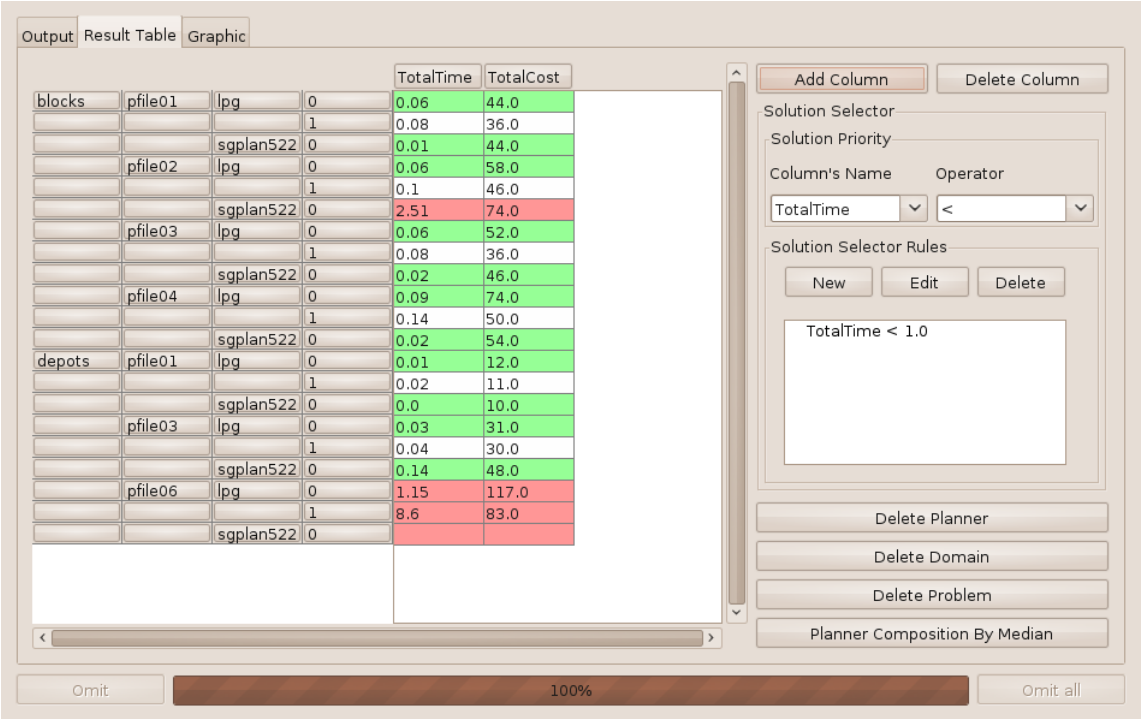

Figura 6.15: Interfaz de resultado con la tabla de variables

SRO-010	
Interfaz de composición de planificadores por mediana	
Descripción:	En la figura 6.16 se observa cómo ha de ser la interfaz de usuario que nos permite crear una composición de planificadores por mediana. Para ello se ha de poder introducir un nombre para la composición y seleccionar los planificadores que se usarán, así como la variable sobre la que se realizará la mediana.

Tabla 6.112: SRO-010, Interfaz de composición de planificadores por mediana

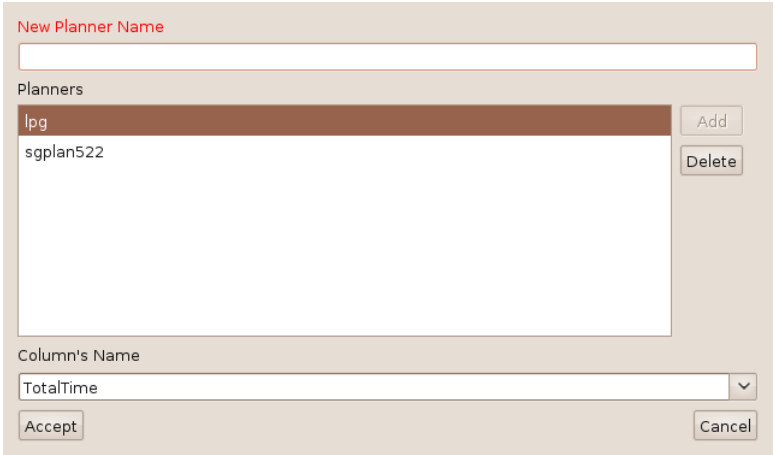

Figura 6.16: Interfaz de composición de planificadores por mediana

SRO-011	
Interfaz de introducción de criterios para la selección de soluciones	
Descripción:	En la figura 6.17 se observa cómo ha de ser la interfaz de usuario que nos permite introducir criterios para desechar soluciones que no los cumplan. Para ello se ha de introducir el nombre de una columna, un operador y un valor.

Tabla 6.113: SRO-011, Interfaz de introducción de criterios para la selección de soluciones

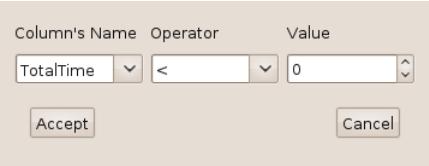

Figura 6.17: Interfaz de introducción de criterios para la selección de soluciones

SRO-012	
Interfaz de añadido de columnas	
Descripción:	En la figura 6.18 se observa cómo ha de ser la interfaz de usuario que nos permite añadir nuevas columnas a la tabla. Para ello se ha de introducir el nombre de la columna a crear, la operación que rellenará la columna y los parámetros que ésta necesite. En la parte derecha de la interfaz se mostrará la descripción de la operación seleccionada para que el usuario pueda saber cómo funciona.

Tabla 6.114: SRO-012, Interfaz de añadido de columnas

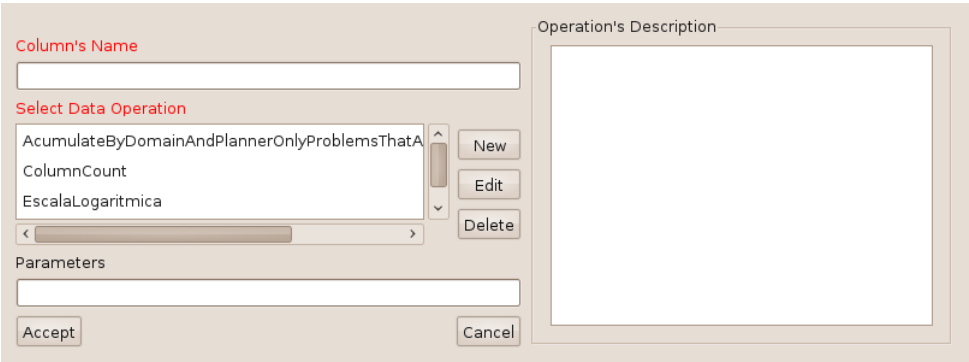

Figura 6.18: Interfaz de añadido de columnas

SRO-013	
Interfaz de creación y editado de las operaciones de añadido de columnas	
Descripción:	En la figura 6.19 se observa cómo ha de ser la interfaz de usuario que nos permite crear o editar las operaciones de añadido de columnas. Para ello se ha de poder introducir el nombre de la operación, una descripción y el código java de esta.

Tabla 6.115: SRO-013, Interfaz de creación y editado de las operaciones de añadido de columnas

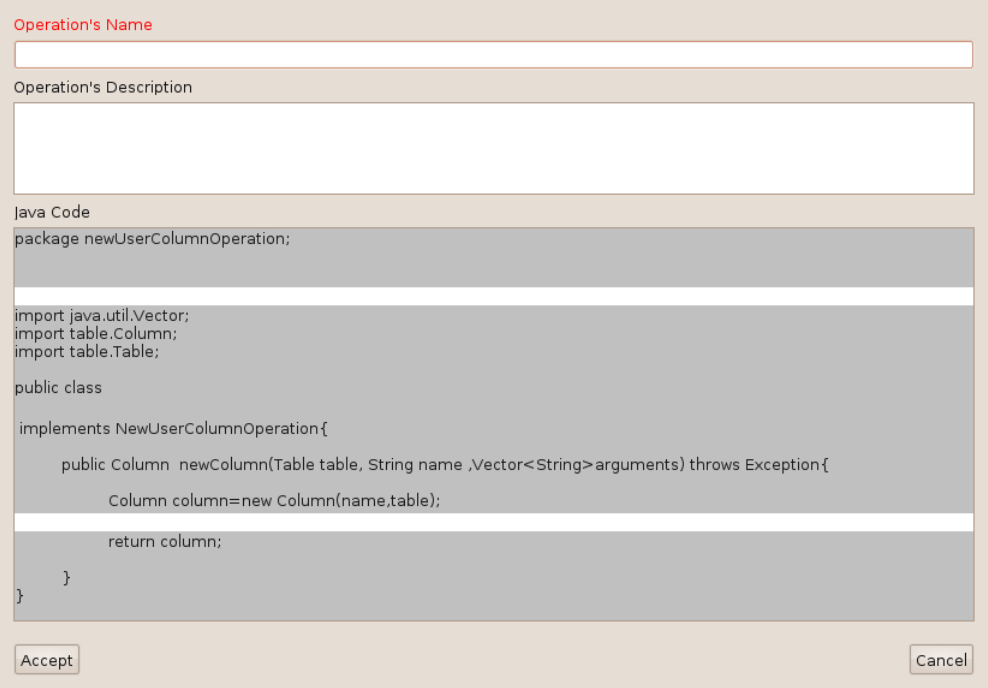

Figura 6.19: Interfaz de creación y editado de las operaciones de añadido de columnas

SRO-014	
Interfaz de resultado con las gráficas	
Descripción:	En la figura 6.20 se observa cómo ha de ser la interfaz de usuario que nos permite observar las gráficas que se han generado a partir de las variables que se encuentran en la tabla de resultados. En esta interfaz se ha de poder seleccionar el dominio y la variable sobre la que se van a generar las gráficas, además de los problemas y planificadores que se van a mostrar. Se ha de permitir al usuario observar la gráfica tanto en color como en blanco y negro. Además, desde esta interfaz se ha de poder guardar la gráfica.

Tabla 6.116: SRO-014, Interfaz de resultado con las gráficas

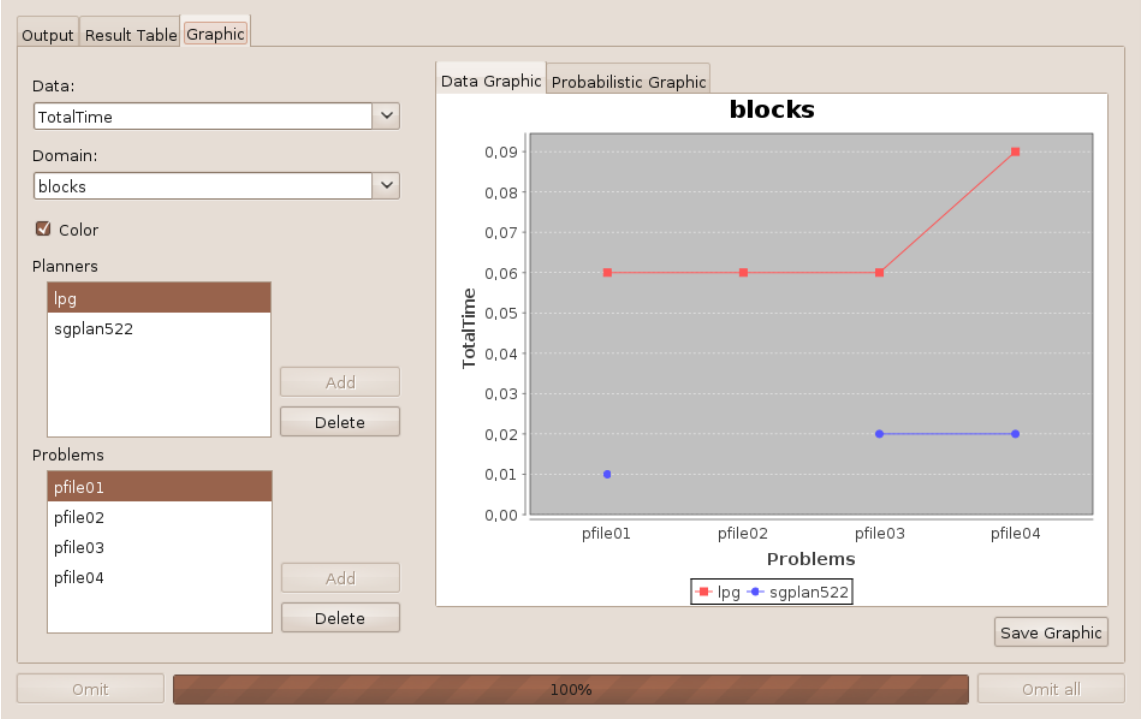

Figura 6.20: Interfaz de resultado con las gráficas

SRO-015	
Interfaz de guardado de las gráficas	
Descripción:	En la figura 6.21 se observa cómo ha de ser la interfaz de usuario que nos permite seleccionar el tamaño de la imagen en la que se va a guardar la gráfica. Para ello se permite escoger el alto y el ancho de esta imagen.

Tabla 6.117: SRO-015, Interfaz de guardado de las gráficas

Figura 6.21: Interfaz de guardado de las gráficas

SRO-016	
Interfaz de selección de archivos JPEG	
Descripción:	En la figura 6.22 se observa cómo ha de ser la interfaz de usuario que nos permite escoger un fichero JPEG. Esta interfaz se utiliza para seleccionar cómo guardar la imagen de una gráfica.

Tabla 6.118: SRO-016, Interfaz de selección de archivos JPEG

6.1.2.5. Requisitos de recursos

Especifican límites superiores sobre recursos físicos, como poder de procesamiento, memoria principal, espacio en disco, etc. El análisis de la herramienta PLEX no requiere ningún requisito de este tipo.

6.1.2.6. Requisitos de comprobación

Especifican restricciones sobre cómo el software va a ser verificado. Son requisitos que exigen características que facilitan la comprobación del sistema o dicen como se va a realizar.

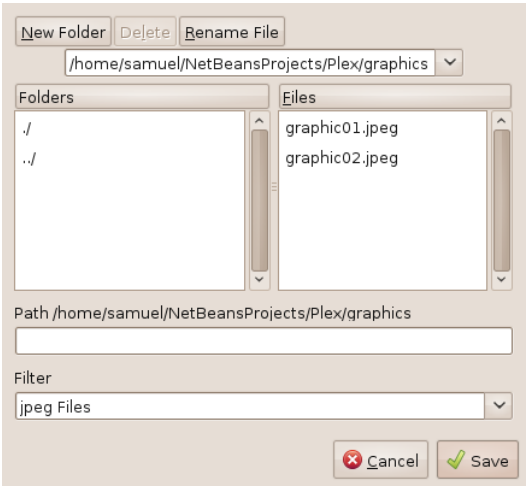

Figura 6.22: Interfaz de selección de archivos JPEG

SRO-017	
Interfaz de selección de archivos PNG	
Descripción:	En la figura 6.23 se observa cómo ha de ser la interfaz de usuario que nos permite escoger un fichero PNG. Esta interfaz se utiliza para seleccionar cómo guardar la imagen de una gráfica.

Tabla 6.119: SRO-017, Interfaz de selección de archivos PNG

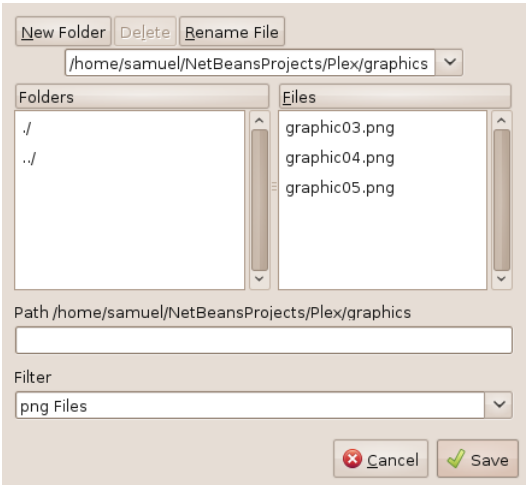

Figura 6.23: Interfaz de selección de archivos PNG

SRC-001	
Control de dominios	
Descripción:	La herramienta PLEX no permitirá que se introduzcan varios dominios con el mismo nombre. Cada dominio deberá tener un archivo que lo describa y por lo menos un problema. Los problemas dentro de un dominio deberán tener nombres de archivo diferentes.

Tabla 6.120: SRC-001, Control de dominios

SRC-002	
Control de planificadores	
Descripción:	La herramienta PLEX no permitirá que se introduzcan varios planificadores con el mismo nombre. Cada planificador deberá tener un archivo ejecutable y una línea de comandos que tenga por lo menos una etiqueta para la introducción del nombre del problema y otra para el nombre del dominio. . Cada planificador deberá, por lo menos, tener un nombre de variable con su respectiva clave de texto único para su posterior estudio. En el caso de que se marquen múltiples soluciones en el planificador será necesario diferenciarlas a través de una clave de texto único.

Tabla 6.121: SRC-002, Control de planificadores

SRC-003	
Control de columnas	
Descripción:	Cada columna ha de tener un nombre diferente, y como mínimo, la tabla de resultados siempre deberá tener una columna.

Tabla 6.122: SRC-003, Control de columnas

SRC-004	
Control de operaciones de añadido de columnas	
Descripción:	El nombre de la operación deberá ser un nombre válido de clase JAVA y no podrá acabar con un número.

Tabla 6.123: SRC-004, Control de operaciones de añadido de columnas

SRC-005	
Control de la composición de planificadores por mediana	
Descripción:	El nombre del planificador resultante de la composición deberá ser único. Para realizar la composición, como mínimo, se deberán escoger dos planificadores.

Tabla 6.124: SRC-005, Control de la composición de planificadores por mediana

SRC-006	
Control de experimento	
Descripción:	Para que un experimento pueda ejecutarse deberá incluir, como mínimo, un dominio y un planificador.

Tabla 6.125: SRC-006, Control de experimento

SRC-007	
Control de errores	
Descripción:	Todos los errores que se produzcan en la herramienta PLEX deberán ser reportados al usuario a través de la interfaz gráfica.

Tabla 6.126: SRC-007, Control de errores

6.1.2.7. Requisitos de aceptación

Estos requisitos especifican restricciones de cómo el software ha de ser validado.

6.1.2.8. Requisitos de documentación

Estos requisitos especifican cómo va a ser la documentación que se generará durante el proyecto.

SRC-008	
Compatibilidad de planificador y dominio	
Descripción:	La herramienta PLEX no deberá comprobar que un planificador sea compatible con el dominio escogido. Quedará bajo la responsabilidad del usuario configurar experimentos con dominios válidos para los planificadores usados.

Tabla 6.127: SRC-008, Compatibilidad de planificador y dominio

SRC-009	
Control de la tabla de resultados	
Descripción:	La tabla de resultados ha de tener siempre, como mínimo, un planificador, un dominio y un problema.

Tabla 6.128: SRC-009, Control de la tabla de resultados

SRA-001	
Pruebas a realizar	
Descripción:	Se deberán realizar pruebas en la herramienta PLEX con la finalidad de comprobar que todos los requisitos se han implementado y que la herramienta funciona correctamente.

Tabla 6.129: SRA-001, Pruebas a realizar

SRA-002	
Documentación pruebas	
Descripción:	No se realizará ninguna documentación sobre las pruebas realizadas.

Tabla 6.130: SRA-002, Documentación pruebas

SRD-001	
Lenguaje de la documentación	
Descripción:	Toda la documentación del proyecto deberá ser redactada en español.

Tabla 6.131: SRD-001, Lenguaje de la documentación

SRD-002	
Índices a incluir	
Descripción:	Se debe incluir un índice de documento, uno de tablas y uno de figuras.

Tabla 6.132: SRD-002, Índices a incluir

SRD-003	
Nombre de tablas y figuras	
Descripción:	Todas las tablas y figuras deberán tener un nombre asociado que aparecerá a continuación de éstas.

Tabla 6.133: SRD-003, Nombre de tablas y figuras

SRD-004	
Cabecera de página	
Descripción:	En las páginas pares, en la parte izquierda de la cabecera, aparecerá la sección en la que se encuentra el lector y en la parte derecha el número de página. En las páginas impares, en la parte derecha de la cabecera, se mostrará el capítulo en el que se encuentra el lector y en la parte izquierda el número de página.

Tabla 6.134: SRD-004, Cabecera de página

SRD-005	
Pie de página	
Descripción:	En el pie de página de las paginas impares se mostrará el nombre del documento y el escudo de la universidad Carlos III de Madrid. En el de las páginas pares se colocará en la parte izquierda el nombre del autor y en la derecha la fecha de creación del documento.

Tabla 6.135: SRD-005, Pie de página

6.1.2.9. Requisitos de mantenimiento

En los requisitos de mantenimiento se especifica cómo debe realizarse el software para permitir su futura modificación.

6.1.2.10. Requisitos de portabilidad

Especifican facilidad de modificación del software para ser ejecutado en otros ordenadores o sistemas operativos.

6.2. Diseño del sistema

En esta sección se detallará el diagrama de clases y se explicarán las principales funcionalidades de la herramienta PLEX.

Para facilitar la comprensión, el diagrama de clases se dividirá por librerías y éstas a su vez por paquetes. Dentro de cada paquete sólo se mostrarán los detalles de las clases pertenecientes a ellos pero solamente el nombre y el paquete de las clases externas con las que se relacionan.

SRM-001	
Nombres de programación	
Descripción:	Los nombres para los métodos y las variables han de estar en inglés y ser lo suficientemente descriptivos para dar una idea de qué función realizan o de qué dato almacenan.

Tabla 6.136: SRM-001, Nombres de programación

SRM-002	
Comentarios de código	
Descripción:	El código ha de estar debidamente comentado para que en todo caso se entienda qué es lo que se ha programado y cómo funciona.

Tabla 6.137: SRM-002, Comentarios de código

SRP-001	
Sistemas operativos	
Descripción:	La herramienta PLEX ha de funcionar en el sistema operativo Linux en cualquier versión que implemente las llamadas al sistema ulimit, kill y ps.

Tabla 6.138: SRP-001, Sistemas operativos

SRP-002	
Idiomas de la interfaz	
Descripción:	La herramienta PLEX ha de estar implementada en español e inglés. Además se podrán añadir nuevos lenguajes sin necesidad de recompilar la aplicación.

Tabla 6.139: SRP-002, Idiomas de la interfaz

Por claridad no se va a explicar en este documento cada uno de los atributos y métodos de las clases, por lo que si se quieren más detalles se puede acudir a la documentación de la herramienta.

6.2.1. Librería Plex

Esta librería es la que contiene toda la parte funcional de la aplicación, así pues, incluye el kernel, la entrada y salida formada por la interfaz gráfica y el modelo encargado de leer y escribir datos. A continuación se mostrará el diseño de los distintos paquetes.

6.2.1.1. Paquete kernel

En el paquete kernel podemos observar la clase Experimenter (figura 6.24) que contiene el método main, por lo que es la encargada de ejecutar la aplicación. Una vez se instancia, esta clase es la encargada de construir y mostrar la interfaz gráfica. También es la encargada de crear experimentos, abrirlos, guardarlos y ejecutarlos.

La clase Experimenter, además, implementa el patrón Singleton para que sólo se pueda crear un objeto de esta clase y para que se pueda acceder a ella desde cualquier punto de la aplicación.

6.2.1.2. Paquete kernel.experiment

En la figura 6.25 se puede observar la clase Experiment de la que se compone este paquete. Esta clase es la encargada de guardar toda la información del experimento, tanto la configuración inicial, como los resultados.

6.2.1.3. Paquete kernel.experiment.domains

En la figura 6.26 podemos observar las clases Domains, Domain, Problems y Problem pertenecientes a este paquete. En estas clases, el experimento mantiene los datos de configuración que conciernen al dominio y a los problemas usados para éste en el experimento.

La clase Domains almacena los diferentes dominios que el usuario introduce en el experimento, Domain guarda la información de cada dominio, Problems guarda todos los problemas que se quieren utilizar con un determinado experimento y Problem almacena la información de cada problema.

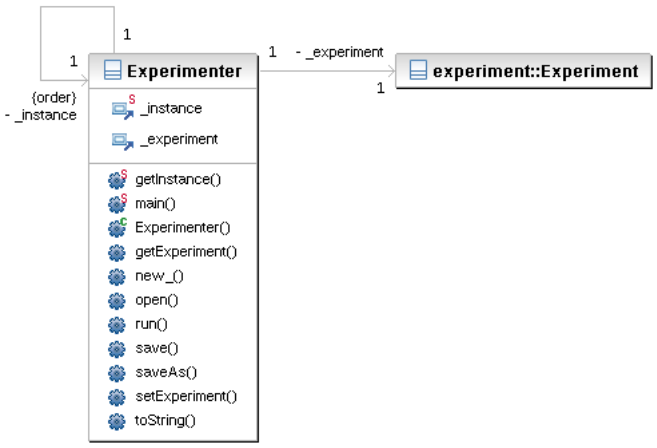

Figura 6.24: Diagrama de clases del paquete kernel

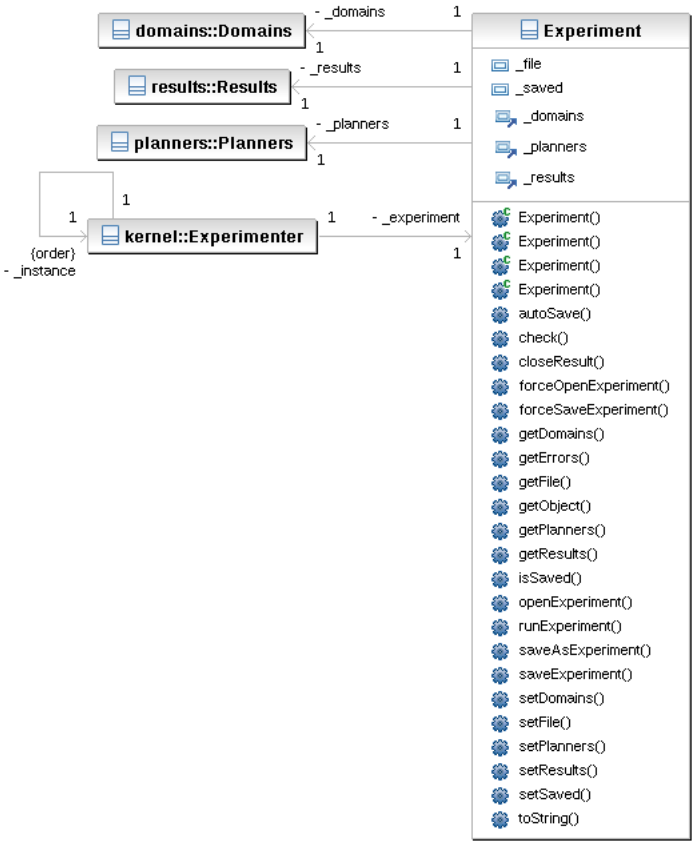

Figura 6.25: Diagrama de clases del paquete kernel.experiment

6.2.1.4. Paquete kernel.experiment.planners

Este paquete está formado por las clases Planners, Planner, PlannerParameters, MultipleSolutionsKey, NameKey y RunPlanner como se puede observar en la figura 6.27.

La clase RunPlanner es la encargada de obtener la solución que se desprende de la ejecución de un planificador para un dominio y un problema determinado. El resto de las clases se encargan de almacenar la configuración del experimento tal y como hacen las clases del paquete kernel.experiment.domains (sección 6.2.1.3).

La clase Planners almacena los diferentes planificadores que el usuario ha introducido en el experimento, Planner almacena la información de un planificador, PlannerParameters las características particulares de un planificador, MultipleSolutionsKey almacena la información necesaria para diferenciar las soluciones en el caso de que el planificador pueda obtener varias soluciones y NameKey almacena los nombres de las variables que se quieren obtener como resultado en el experimento y especifica cómo obtenerlas.

6.2.1.5. Paquete kernel.experiment.results

Las clases que forman este paquete se especifican en la figura 6.28 y son: Results, Result, Solutions, Solution, Data, Runner y RunExperiment.

La clase Results guarda todos los resultados del experimento una vez este ha sido ejecutado; Result contiene el dominio, problema y planificador que ha sido ejecutado; Solutions almacena una o varias soluciones dependiendo de las que el planificador devuelva; Solution guarda los resultados de cada una

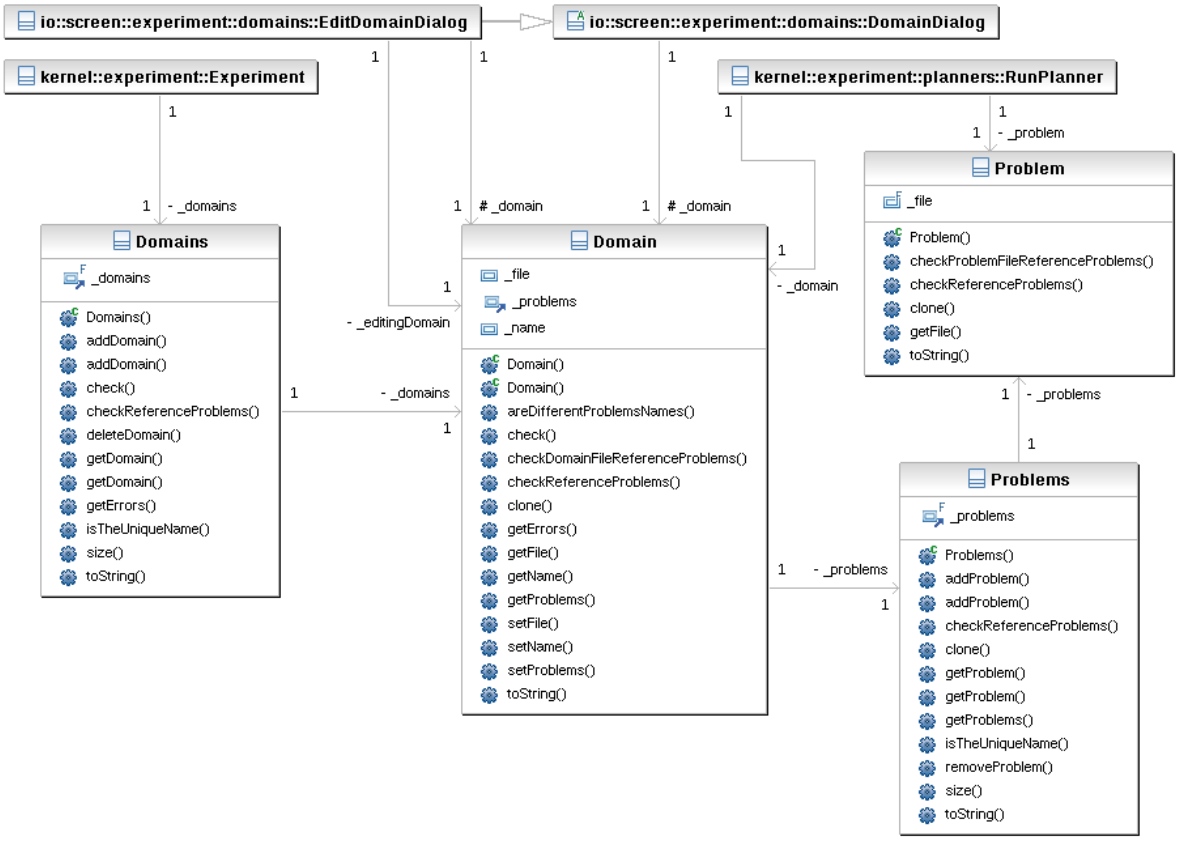

Figura 6.26: Diagrama de clases del paquete kernel.experiment.domains

de las soluciones; y Data almacena los pares de nombre de variable y resultado que el usuario introdujo en el planificador a la hora de configurar el experimento.

Por otro lado la clase Runner se encarga de almacenar la información que se refiere a la parte en la que se haya la ejecución del experimento. Esta clase permite que, si se cierra el programa por algún motivo, se pueda continuar la ejecución desde el último resultado obtenido.

Por último, la clase RunnerExperiment es la encargada de ejecutar cada uno de los planificadores para cada dominio y problema y guardar el resultado.

6.2.1.6. Paquete kernel.experiment.results.columns

En la figura 6.29 se pueden observar las clases NewColumn, NewColumnOperations y NewColumnOperation. Estas clases permiten al usuario introducir nuevas columnas en la tabla de resultados. Esto quiere decir que el usuario puede introducir nuevas variables en la solución de un planificador para un dominio y un problema concreto. Estas nuevas variables podrán depender o no de otras variables ya resueltas.

La clase NewColumn es la encargada de añadir esa nueva columna o variable a los resultados, siendo necesario que el usuario indique el nombre de la nueva variable y qué operación dará los resultados. Para ello se usan las clases NewColumnOperations que se encargan de almacenar y eliminar las operaciones que el usuario haya creado a través de la clase NewColumnOperator. Ésta es la encargada de recoger código java del usuario en tiempo de ejecución, permitiendo así que se use para crear nuevas columnas.

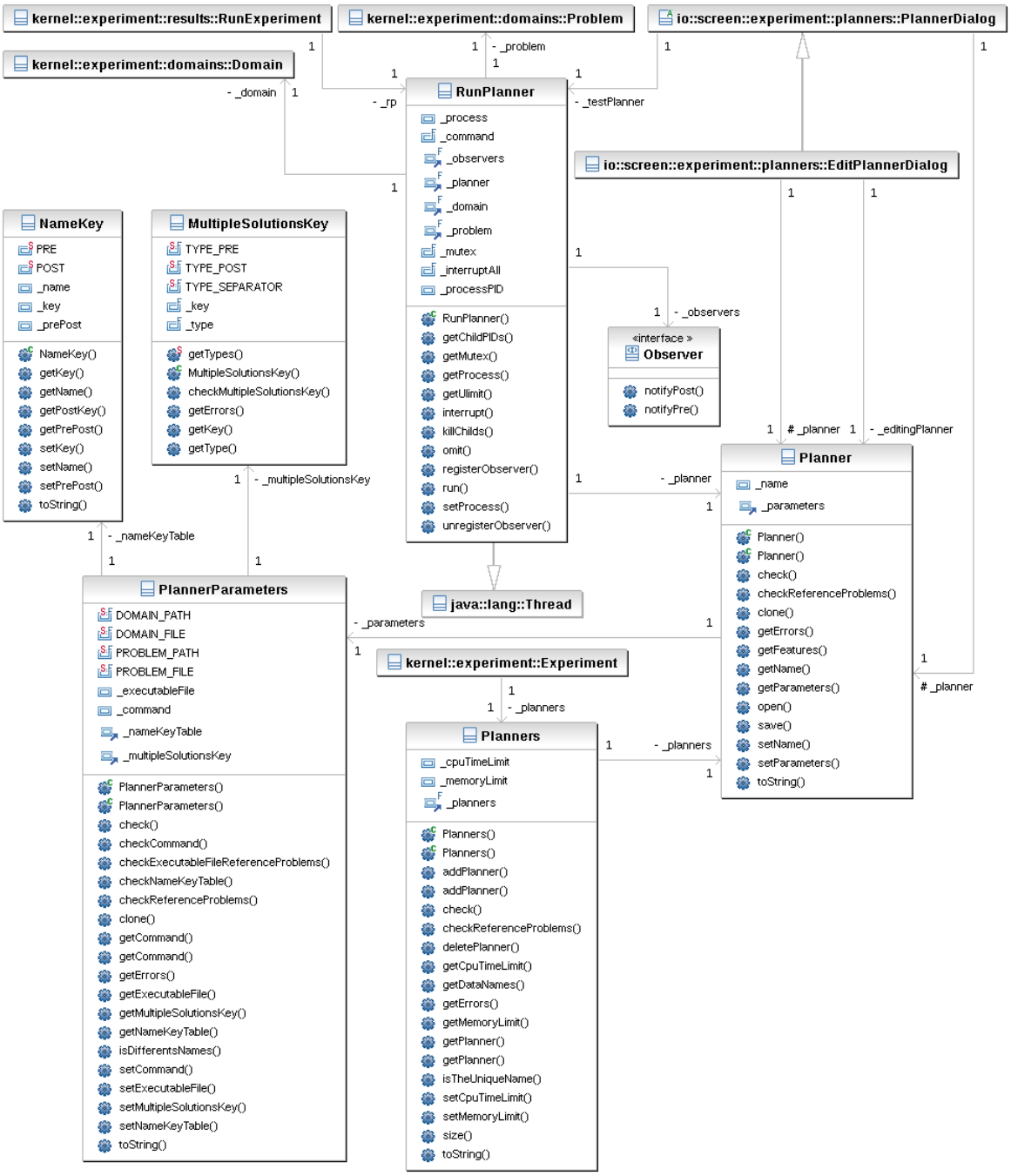

Figura 6.27: Diagrama de clases del paquete kernel.experiment.planners

Figura 6.28: Diagrama de clases del paquete kernel.experiment.results

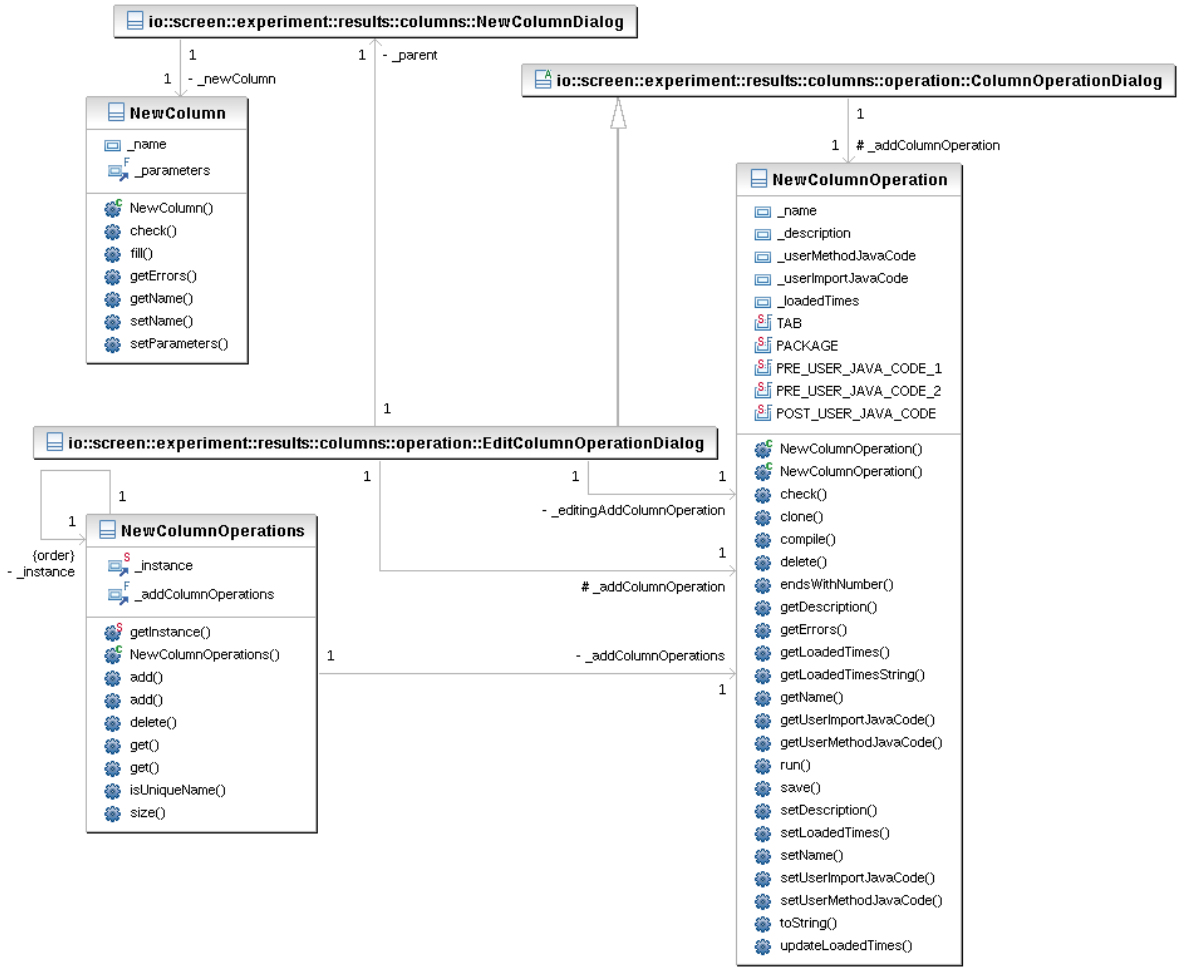

Figura 6.29: Diagrama de clases del paquete kernel.experiment.results.columns

6.2.1.7. Paquete kernel.experiment.results.solutionSelector

Las clases mostradas en la figura 6.30 permiten al usuario elegir una solución, en el caso de que hubiera varias, o excluir del resultado del experimento las que no le interesen. Esto es útil para la generación de gráficas ya que, para el resultado de un planificador ejecutado con un dominio y un problema que tenga múltiples soluciones, es necesario escoger una solución para su representación.

La clase SolutionSelector es la encargada de obtener la solución prioritaria y las que quedan fuera del experimento. Para ello utiliza la clase SolutionPriority que escoge una solución a partir de todas las soluciones válidas para las múltiples soluciones de un planificador dependiendo de una variable y de un operador de prioridad. Por otro lado, SolutionSelectorRule se encarga de crear reglas a partir de nombres de variables, operadores lógicos y valores que deciden qué soluciones son válidas y cuáles quedan fuera del experimento.

6.2.1.8. Paquete kernel.experiment.results.solutionSelector.operators

En este paquete se engloban las interfaces PriorityOperator y LogicOperator, así como las clases que las implementan que son: Mayor, MayorQue, Igual, MenorQue y Menor (figura 6.31). La única función de estas clases es la de comparar valores.

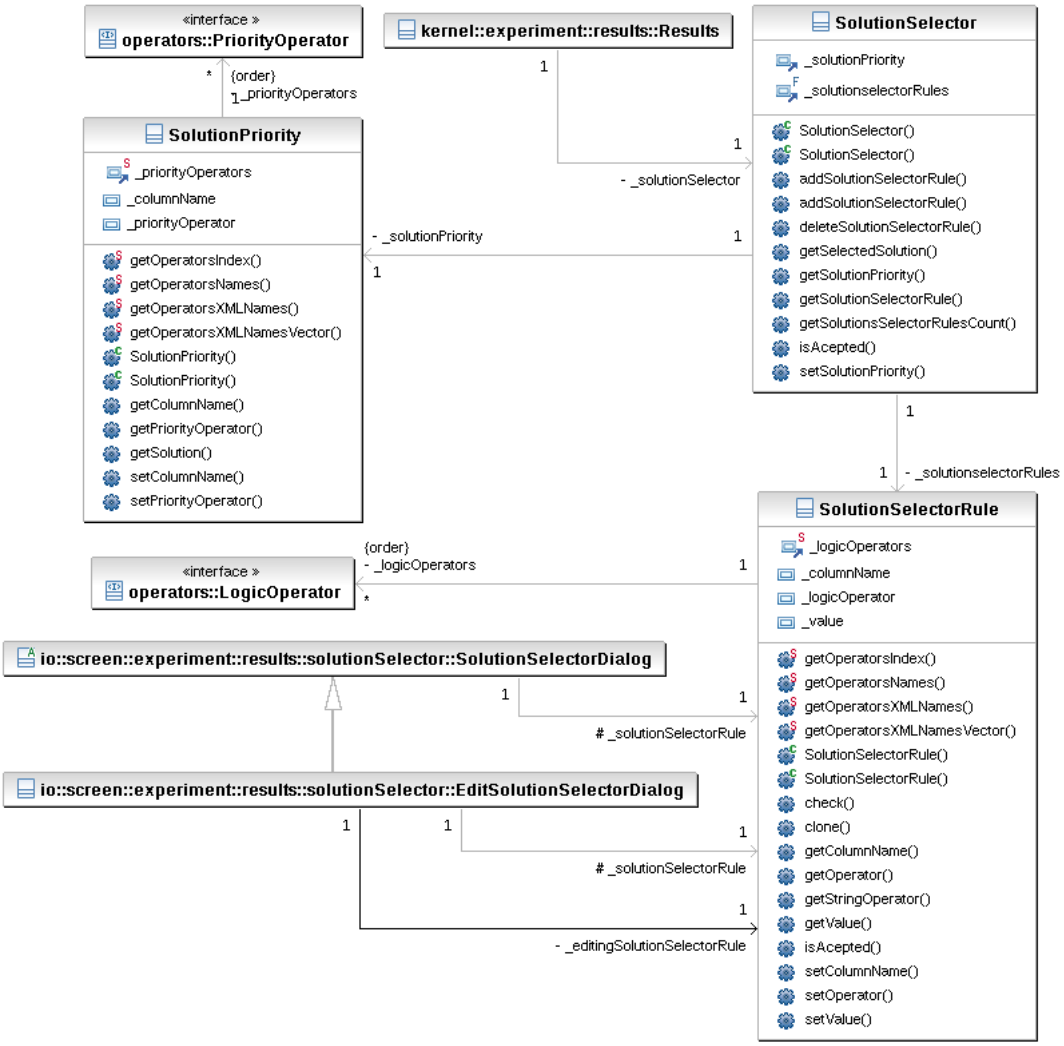

Figura 6.30: Diagrama de clases del paquete kernel.experiment.results.solutionSelector

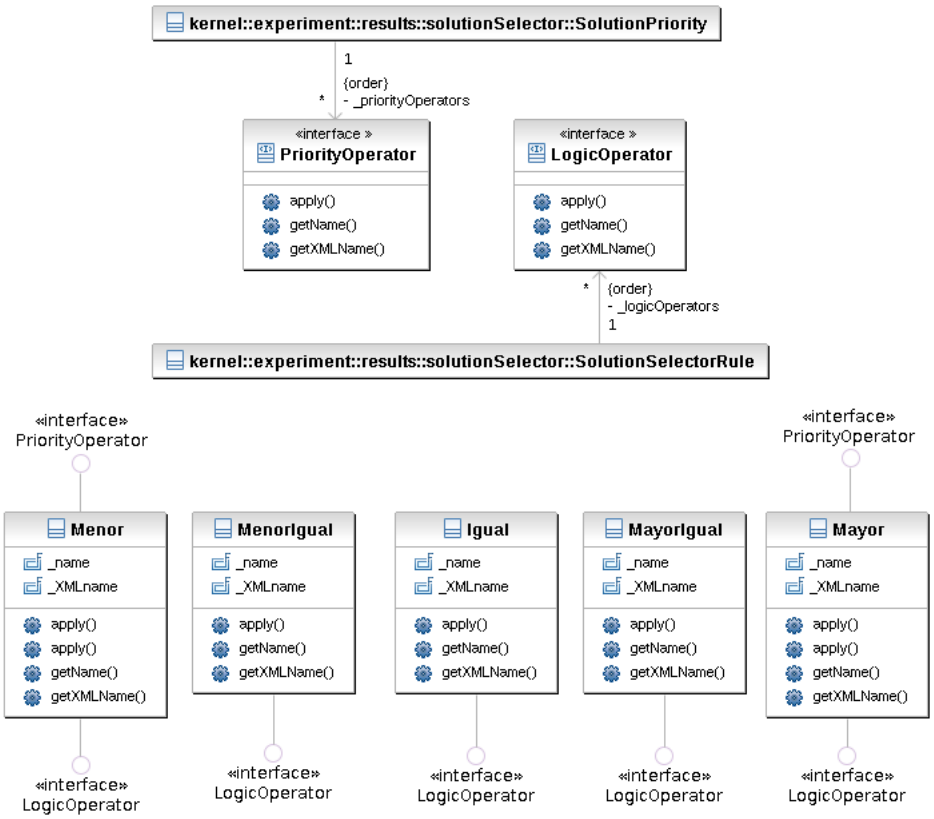

Figura 6.31: Diagrama de clases del paquete kernel.experiment.results.solutionSelector.operators

6.2.1.9. Paquete kernel.language

Este paquete se compone de una única clase llamada CurrentLanguage, como se observa en la figura 6.32. En el diagrama de esta clase se han omitido los métodos y los atributos por ser una clase demasiado grande.

La clase CurrentLanguage tiene como función almacenar el idioma que la herramienta tiene en uso. Implementa el patrón Singleton y guarda en cada atributo cada uno de los textos que tiene la herramienta. Como métodos tiene todos los de acceso y modificación de sus atributos.

6.2.1.10. Paquete io.file

El paquete io.file se compone de dos clases, tal y como podemos observar en la figura 6.33. Estas clases son WorkingDirectory y ModificadorClassPath.

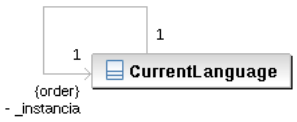

Figura 6.32: Diagrama de clases del paquete kernel.language

La clase WorkingDirectory se compone de un método estático que nos permite obtener la ruta absoluta de la aplicación en el sistema ejecutado, independientemente de cómo se haya ejecutado ésta. Esta clase es útil para que ejecuciones como "java -jar Plex.jar" y "java -jar carpeta/Plex.jar" den rutas idénticas.

La clase ModificadorClassPath implementa el patrón Singleton y añade archivos en tiempo de ejecución a la herramienta PLEX. Esta clase se utiliza para introducir el código generado por el usuario y crear así nuevas columnas en tiempo de ejecución.

6.2.1.11. Paquete io.file.data

Este paquete se compone de veinte clases que están representadas en la figura 6.34. La función de estas clases es la de cargar los datos de los ficheros XML y la de crearlos a partir de los datos del experimento. La única clase que tiene un comportamiento diferente es la de NewColumnOperationData que es la única que no lee y guarda los objetos como XML ya que estos datos se cargan directamente como un objeto.

6.2.1.12. Paquete io.file.experiment

Este paquete se compone de una única clase con dos métodos estáticos como se observa en la figura 6.35. La clase llamada ExperimentIO es la encargada de que, a través de sus métodos estáticos, el kernel pueda indicar, a la entrada y salida de la aplicación, que se ha de cargar o guardar un experimento.

6.2.1.13. Paquete io.file.experiment.planner

Este paquete también esta formado por una sola clase con dos métodos estáticos. Esta clase llamada PlannerParametersIO está representada en la figura 6.36 y es la encargada de recibir la orden del kernel de guardar o cargar una plantilla de planificador para ser usada en la configuración del experimento.

6.2.1.14. Paquete io.file.experiment.results.columns

Este paquete se encuentra representado en la figura 6.37 en la que podemos observar la clase NewColumnOperationsIO. Esta clase implementa el patrón Singleton y se encarga de ver qué operaciones de creado de nueva columna se encuentran en la carpeta del disco destinada a este fin y a cargarlas en el sistema.

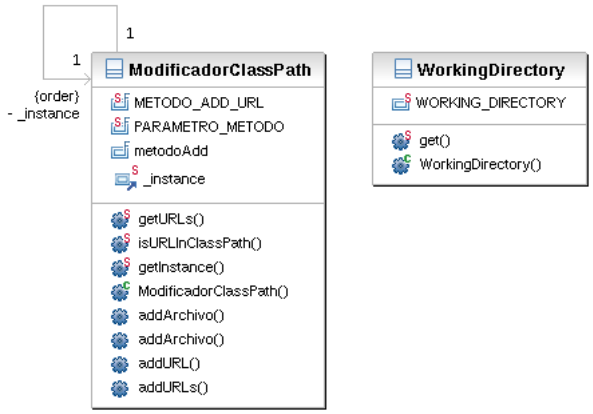

Figura 6.33: Diagrama de clases del paquete io.file

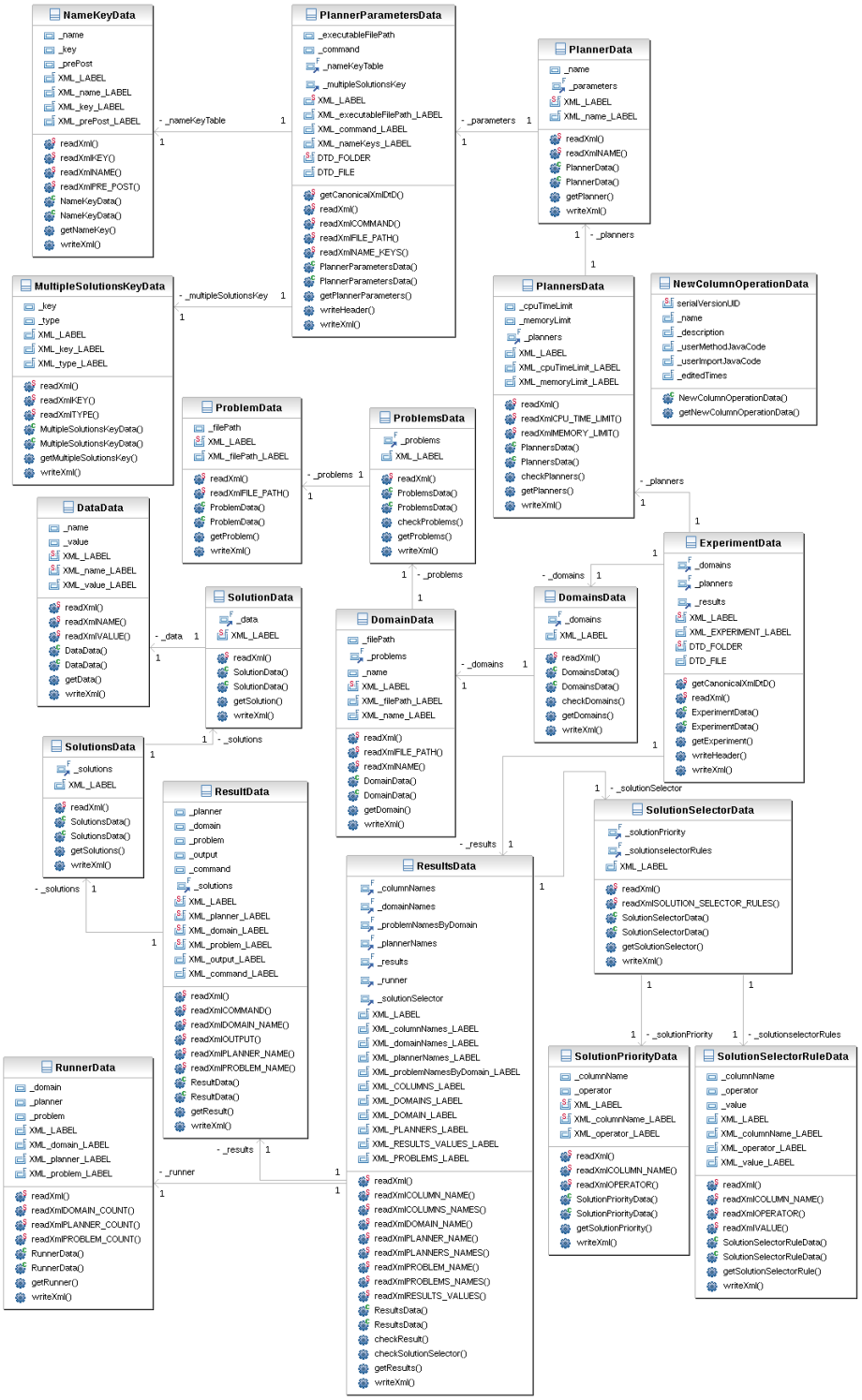

Figura 6.34: Diagrama de clases del paquete io.file.data

Figura 6.35: Diagrama de clases del paquete io.file.experiment

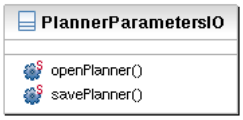

Figura 6.36: Diagrama de clases del paquete io.file.experiment.planner

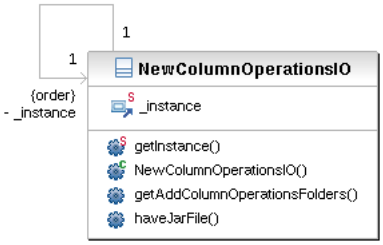

Figura 6.37: Diagrama de clases del paquete io.file.experiment.results.columns

6.2.1.15. Paquete io.file.experiment.results.columns.operation

La clase mostrada en la figura 6.38 permite al kernel de la aplicación cargar, guardar, compilar y borrar operaciones de creación de nuevas columnas. Esta clase, además, implementa el patrón Singleton.

6.2.1.16. Paquete io.file.filter

Como se puede observar en la figura 6.39 todas las clases de este paquete heredan de la clase FileFilter. Estas clases se encargan de gestionar las extensiones de todos los archivos usados por la herramienta PLEX.

Así pues, la clase PngFileFilter se encarga de la extensión de los archivos ".png", la clase NewColumnOperationFileFilter se encarga de los archivos de operaciones para el creado de nuevas columnas con extensión ".ColumnOperationfeatures", la clase JpegFileFilter es la encargada de los archivos ".jpeg", la clase XmlFileFilter de los archivos ".xml" y por último, la clase ExecutableFileFilter se encarga de los archivos ejecutables con extensión, o sin ella, dependiendo del sistema operativo.

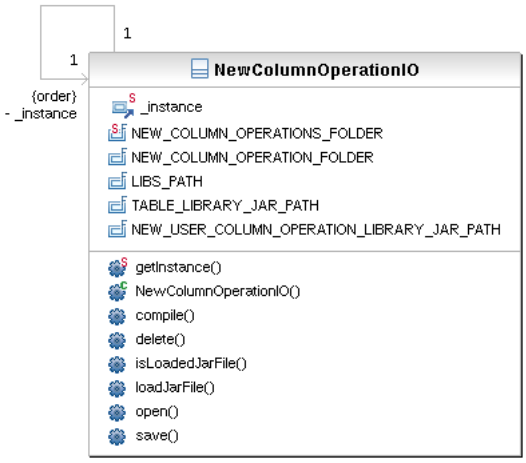

Figura 6.38: Diagrama de clases del paquete io.file.experiment.results.columns.operation

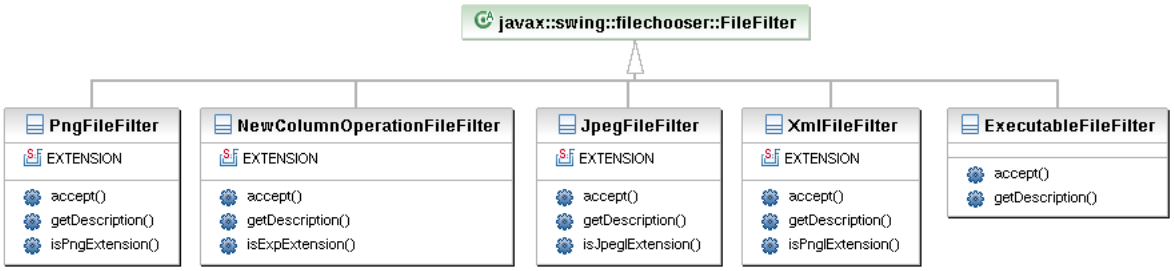

Figura 6.39: Diagrama de clases del paquete io.file.filter

6.2.1.17. Paquete io.file.language

Este paquete mostrado en la figura 6.40 se compone de la clase LanguageIO que implementa el patrón Singleton y que se encarga de leer los textos traducidos de un archivo cargándolo en la aplicación.

6.2.1.18. Paquete io.file.xml

En la figura 6.41 se pueden observar las clases Xml, ExperimenterScanner, XMLException y DTDErrorHandler.

La clase Xml es la encargada de leer y escribir ficheros XML.

La clase ExperimenterScanner trabaja con la clase DTDErrorHandler que implementa la interfaz DTDErrorHandler. La primera se encarga de comprobar la estructura del archivo XML con respecto al esquema DTD del archivo. A su vez va obteniendo la información contenida en éste y manda los errores a la clase DTDErrorHandler que los registra.

Por último, la clase XMLException, que hereda de Exception, se encarga de crear las excepciones relacionadas con los ficheros XML.

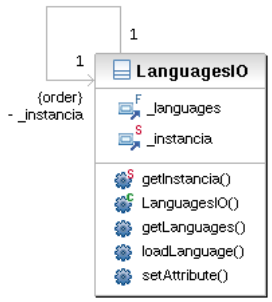

Figura 6.40: Diagrama de clases del paquete io.file.language

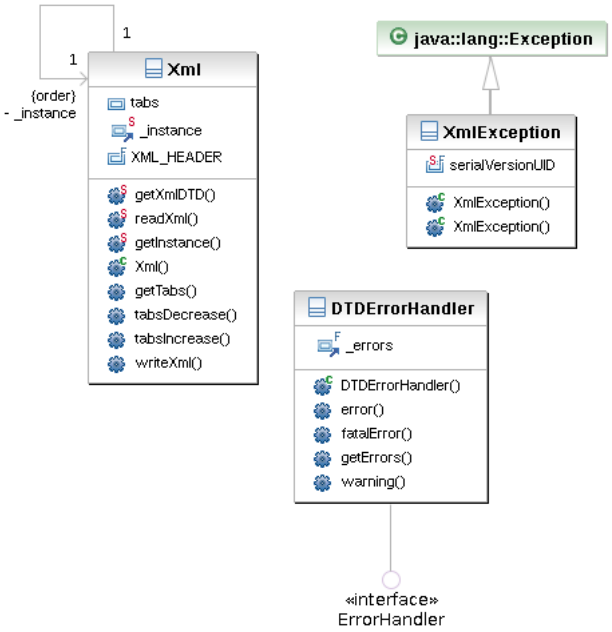

Figura 6.41: Diagrama de clases del paquete io.file.xml

6.2.1.19. Paquete io.screen.error

Este paquete está reflejado en la figura 6.42 y se compone de la clase ErrorDialog que se encarga de crear una interfaz que muestre al usuario los posibles errores que surjan con el uso de la herramienta PLEX.

6.2.1.20. Paquete io.screen.experiment

En este paquete se encuentran las clases que crean la interfaz de inicio de la aplicación y que permite crear, guardar, abrir y ejecutar experimentos. Las clases de las que se compone se muestran en la figura 6.43. Por último, resaltar que la clase ExperimenterFrame implementa el patrón Singleton y es por esta clase por la que el kernel se comunica con la interfaz gráfica.

6.2.1.21. Paquete io.screen.experiment.domains

En la figura 6.44 se pueden ver representadas las clases de este paquete. Estas clases son las encargadas de crear la interfaz gráfica para la introducción y modificación de los dominios de un experimento.

6.2.1.22. Paquete io.screen.experiment.planners

Las clases de este paquete se encuentran representadas en la figura 6.45. Estas clases producen la interfaz gráfica necesaria para la introducción y modificación de los planificadores del experimento. También permiten escoger las opciones de éstos y cargar plantillas de planificador previamente guardadas.

6.2.1.23. Paquete io.screen.experiment.results

En este paquete se encuentran las clases encargadas de la creación de la interfaz gráfica que muestra los resultados del experimento al usuario. Además, se encuentran las clases que permiten la gestión de estos resultados. Estas clases se pueden observar en la figura 6.46.

6.2.1.24. Paquete io.screen.experiment.results.columns

Las clases que forman este paquete se encargan de las interfaces gráficas que la aplicación muestra al usuario para la creación y borrado de las columnas de la tabla de resultados. Se pueden ver representadas en la figura 6.47

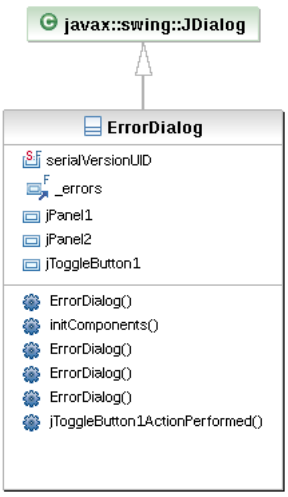

Figura 6.42: Diagrama de clases del paquete io.screen.error

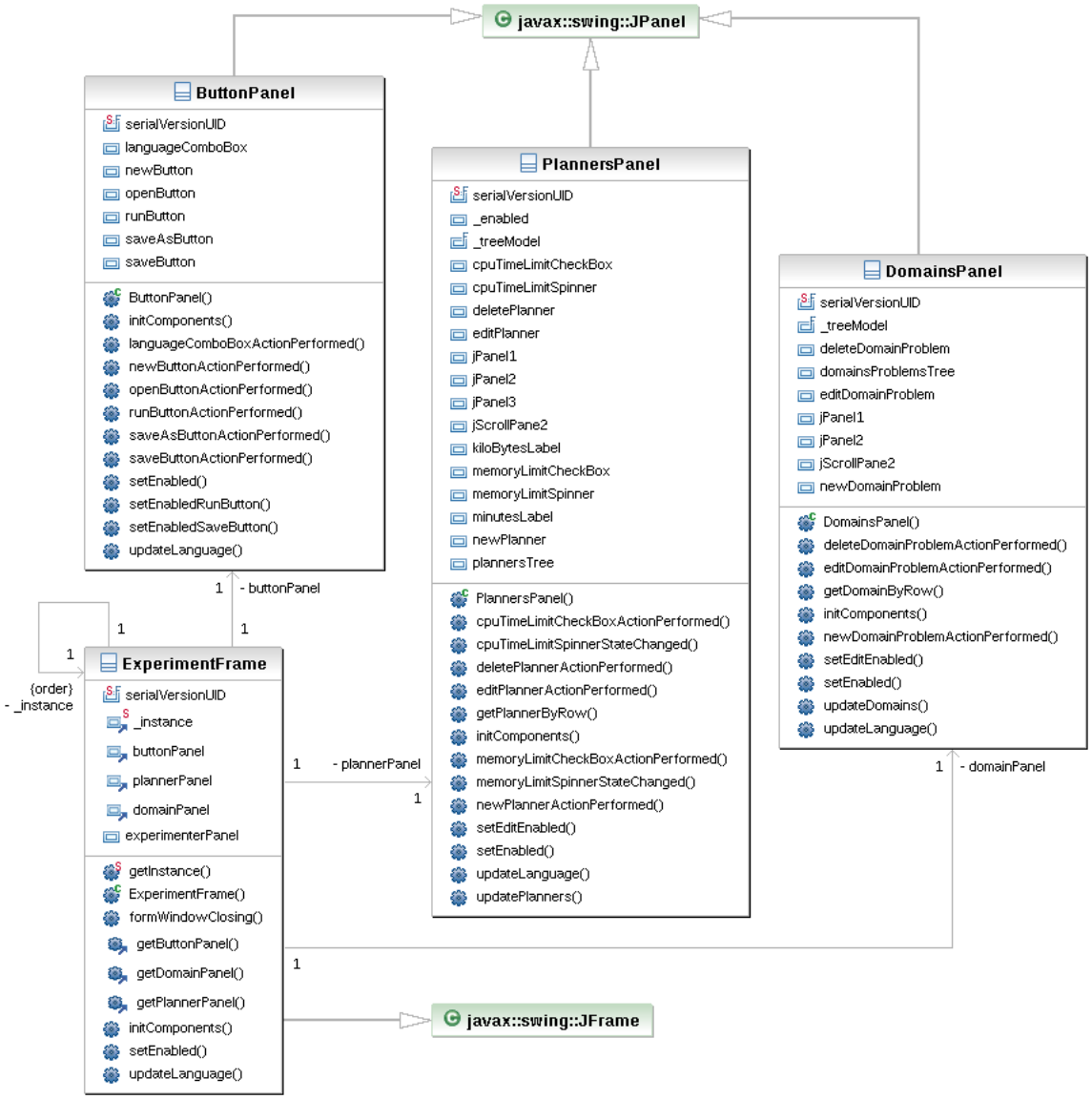

Figura 6.43: Diagrama de clases del paquete io.screen.experiment

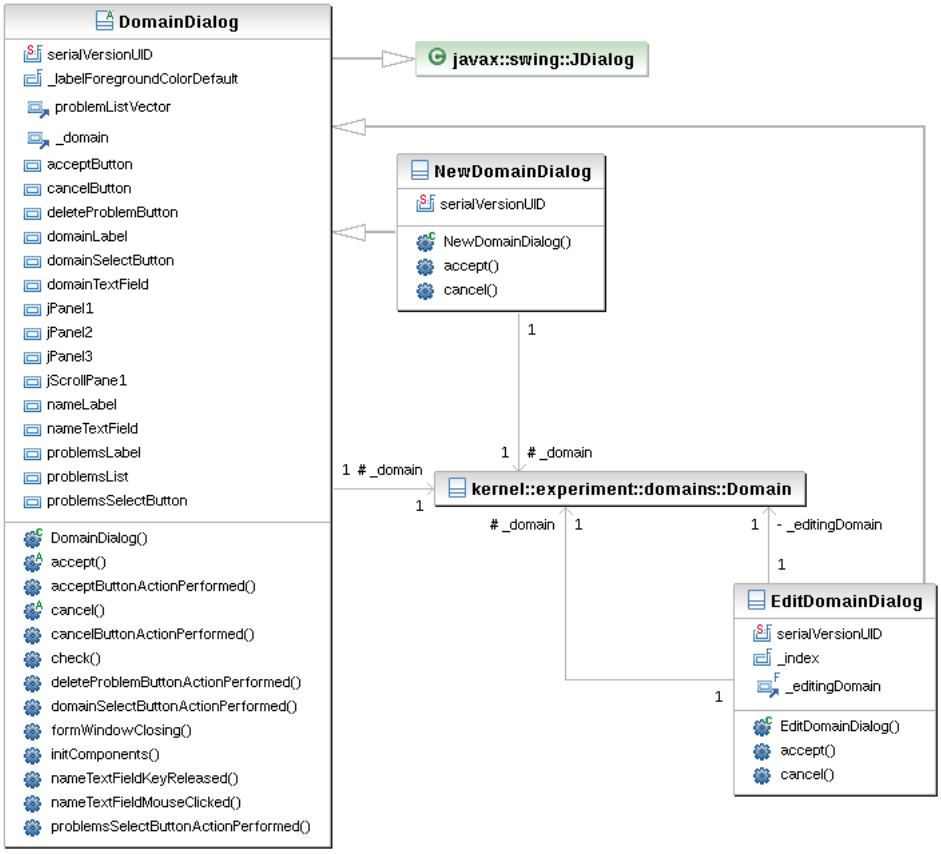

Figura 6.44: Diagrama de clases del paquete io.screen.experiment.domains

6.2.1.25. Paquete io.screen.experiment.results.columns.operation

El diagrama de clases de este paquete se puede ver representado en la figura 6.48. Las clases que lo forman se encargan de las interfaces gráficas necesarias para que el usuario pueda crear y modificar diferentes operaciones de creación de columnas.

6.2.1.26. Paquete io.screen.experiment.results.graphics

Este paquete queda representado en la figura 6.49 y se compone de las clases que generan las interfaces gráficas necesarias para que el usuario de la herramienta pueda obtener diferentes gráficos a partir de la tabla de resultados, también para que pueda añadir planificadores y problemas a éstos y para que pueda guardar las gráficas como imágenes.

6.2.1.27. Paquete io.screen.experiment.results.plannerComposition

Las clases que componen este paquete forman las interfaces gráficas necesarias para que el usuario pueda crear composiciones de planificadores, útiles para planificadores estocásticos. Estas clases se pueden observar en la figura 6.50

6.2.1.28. Paquete io.screen.experiment.results.solutionSelector

Todas las interfaces gráficas necesarias para la creación de reglas de selección de soluciones son creadas por las clases pertenecientes a este paquete y que se pueden ver representadas en la figura 6.51.

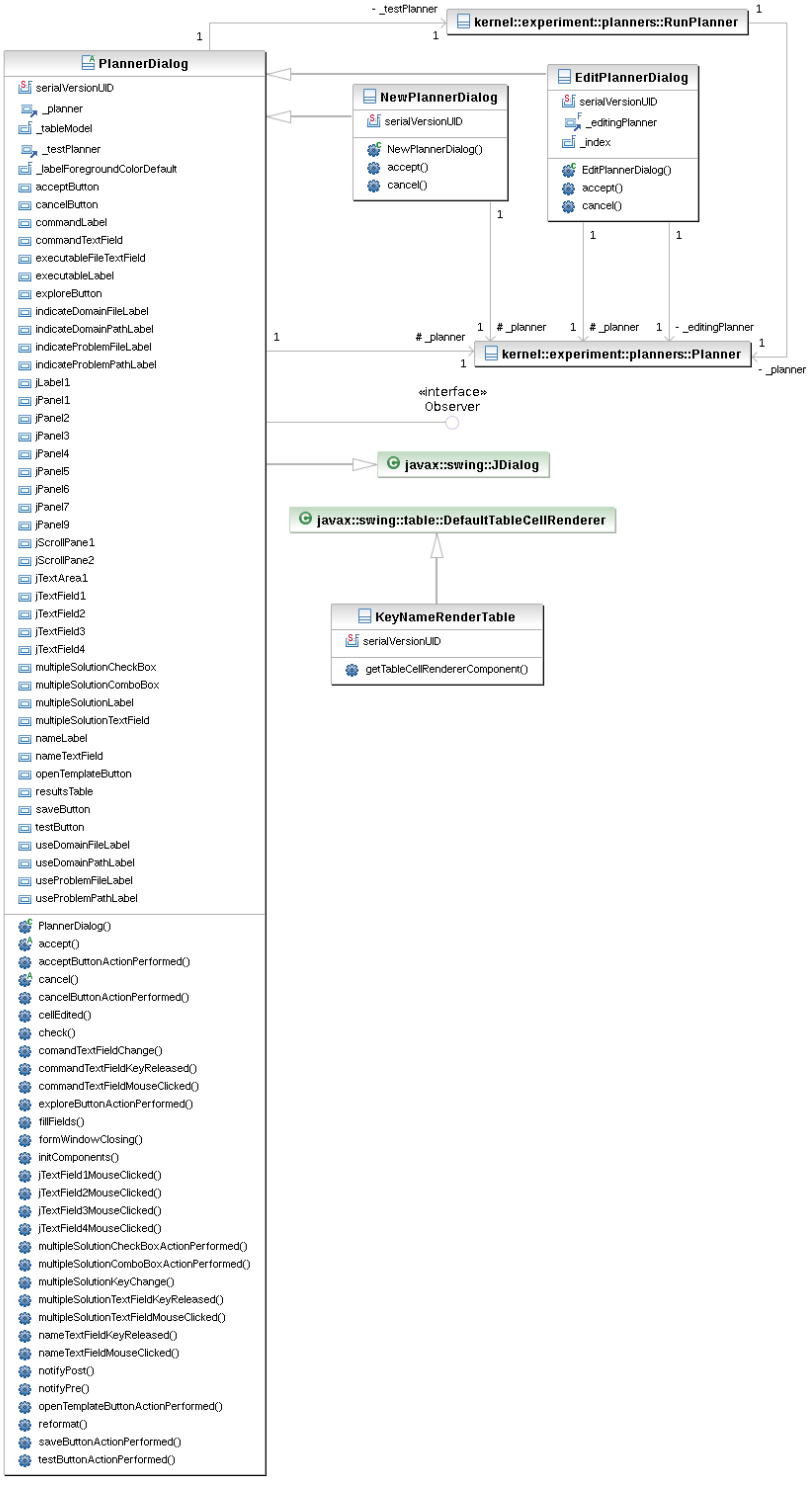

Figura 6.45: Diagrama de clases del paquete io.screen.experiment.planners

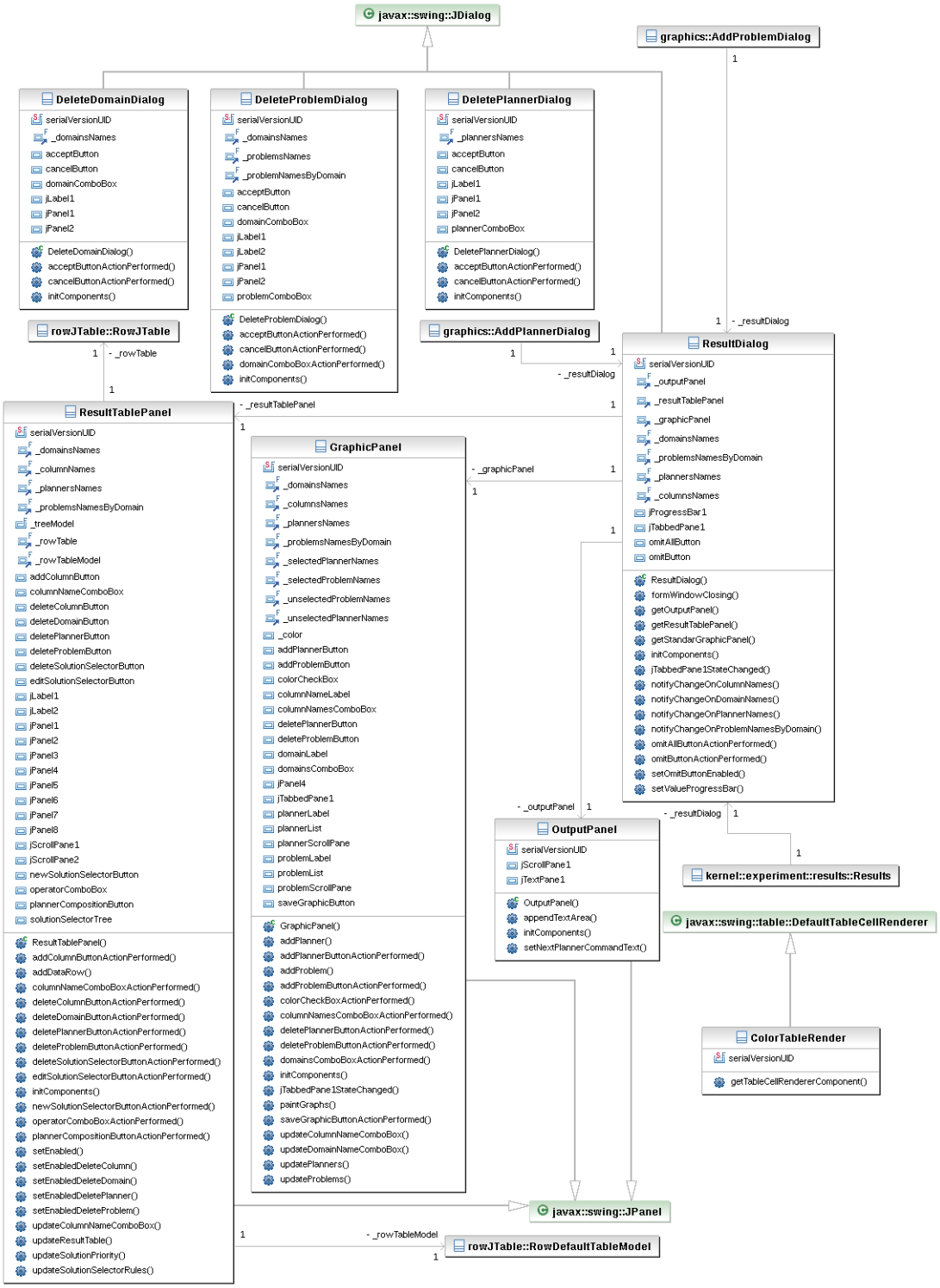

Figura 6.46: Diagrama de clases del paquete io.screen.experiment.results

Figura 6.47: Diagrama de clases del paquete io.screen.experiment.results.columns

6.2.1.29. Paquete io.screen.experiment.save

Este paquete esta representado en la figura 6.52 y en ella se puede observar la clase SaveQuestionDialog que es la encargada de crear una interfaz gráfica que permite realizar una pregunta al usuario.

6.2.2. Librería NewUserColumnOperation

Esta librería se utiliza para poder combinar el código que el usuario introduce en la aplicación en tiempo de ejecución y la propia aplicación.

6.2.2.1. Paquete newUserColumnOperation

En la figura 6.53 podemos observar el diagrama de clases que define este paquete, en el que sólo se encuentra una interfaz cuyo método ha de implementar el código de usuario, y que la aplicación va a poder usar una vez compilado.

6.2.3. Librería RowJTableLibrary

Las clases que forman esta librería permiten la creación de tablas con encabezado tanto en las filas como en las columnas.

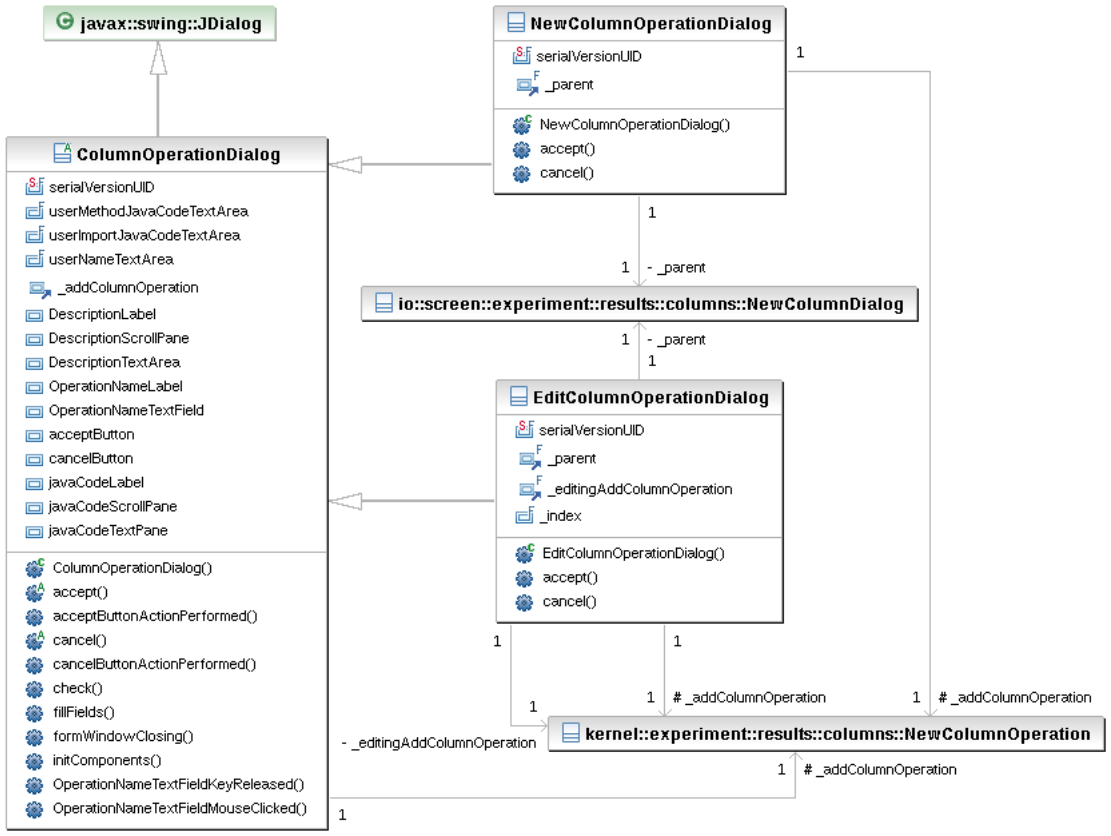

Figura 6.48: Diagrama de clases del paquete io.screen.experiment.results.columns.operation

6.2.3.1. Paquete rowJTable

Las clases que se observan en la figura 6.54 son las que componen este paquete. Se trata de clases que añaden funcionalidad a las clases JAVA de creación de tablas, con la finalidad de incluir encabezado de filas.

6.2.4. Librería TableLibrary

Esta librería se forma por las clases que permiten al usuario acceder a los resultados del experimento cuando crea código para la introducción de nuevas columnas.

6.2.4.1. Paquete table

Las clases de este paquete se encuentran representadas en la figura 6.55.Estas clases son Table, Column e IndexTableIterator.

La clase Table proporciona una serie de métodos que el usuario podrá usar para generar su código, la clase column permite guardar y obtener datos del experimento en forma de columna y por último, la clase IndexTableIterator permite la creación de iteradores para recorrer los resultados del experimento de diferentes formas.

Figura 6.49: Diagrama de clases del paquete io.screen.experiment.results.graphics

6.2.5. Librerías externas

La aplicación usa dos librerías externas que son jfreechart-1.0.12 [1] y jcommon-1.0.15 [2]. La librería jfreechart necesita para su funcionamiento a la librería jcommon. Ambas librerías son usadas para la creación de los gráficos que utiliza la herramienta PLEX.

[1] http://www.jfree.org/jfreechart/
[2] http://www.jfree.org/jcommon/

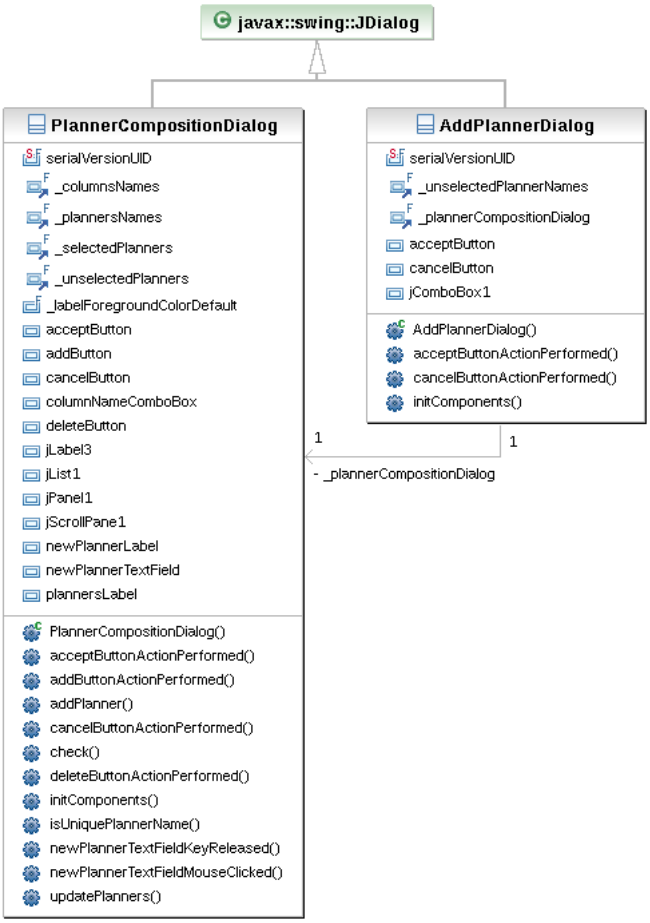

Figura 6.50: Diagrama de clases del paquete io.screen.experiment.results.plannerComposition

Figura 6.51: Diagrama de clases del paquete io.screen.experiment.results.solutionSelector

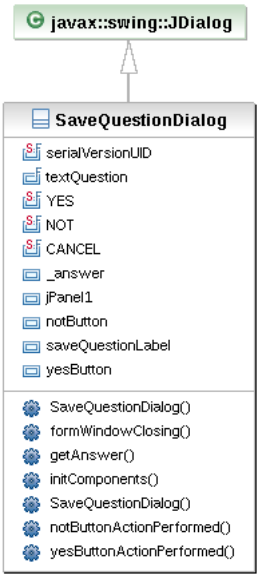

Figura 6.52: Diagrama de clases del paquete io.screen.experiment.save

Figura 6.53: Diagrama de clases del paquete newUserColumnOperation

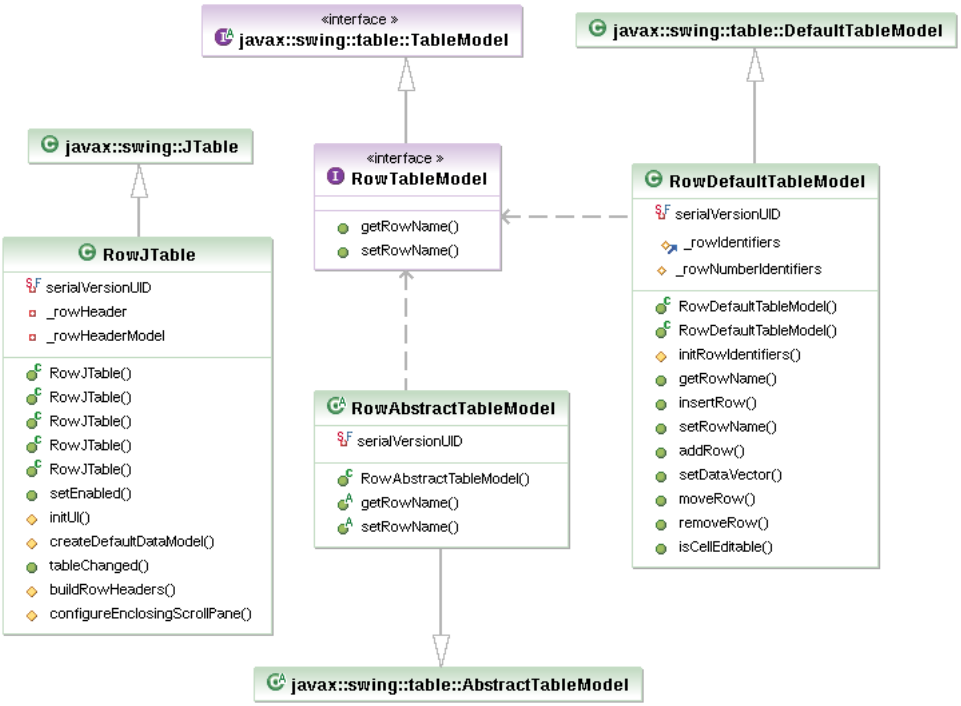

Figura 6.54: Diagrama de clases del paquete rowJTable

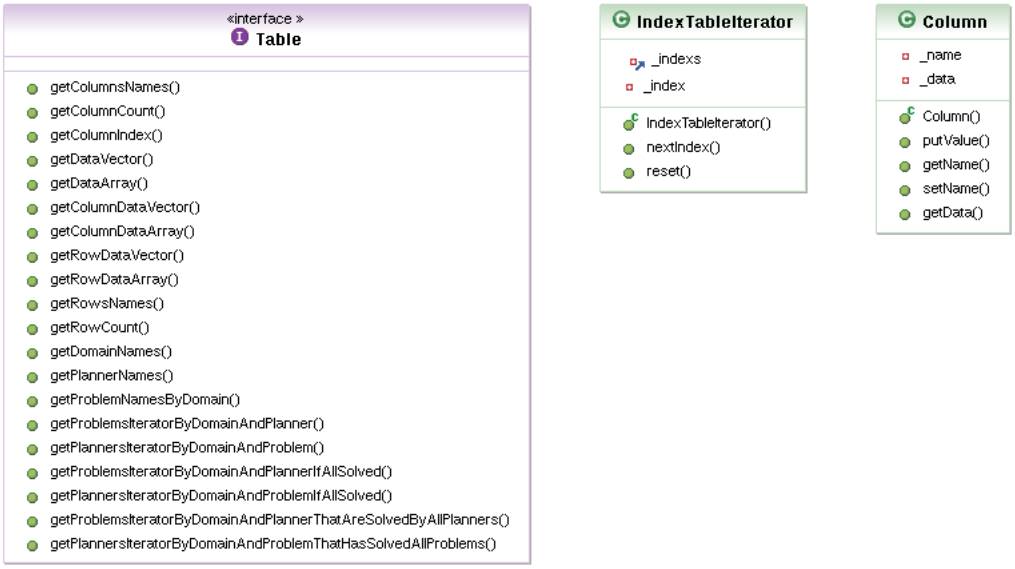

Figura 6.55: Diagrama de clases del paquete table

Capítulo 7

Conclusiones

Tras la realización del Proyecto Fin de Carrera se pueden extraer las conclusiones que se describen a continuación:

1. Se buscaba que la herramienta PLEX fuese totalmente independiente de los dominios, problemas y planificadores. Esto se ha conseguido de manera satisfactoria, aunque con una única salvedad: resulta necesario que las salidas de los planificadores contengan porciones de texto único e invariable que permitan al usuario identificar tanto soluciones como valores de resultados.

2. A través de la herramienta PLEX el usuario es capaz de generar experimentos con diferentes dominios y planificadores y ejecutarlos, pudiendo obtener como resultado las variables que requiera y permitiendo su posterior estudio a través de tablas y gráficas. Para ello la herramienta permite al usuario introducir diferentes problemas con sus respectivos dominios e incluir diferentes planificadores indicando cómo deben ejecutarse y cómo se debe tratar su salida.

3. Además, la herramienta PLEX permite realizar estudios de los resultados añadiendo variables a los resultados tales como: valores acumulados por dominio y planificador, escalas logarítmicas, puntuación de planificadores y cualquier operación que el usuario quiera crear. Para esto se permite al usuario programar estas funciones con código java que se cargará en tiempo de ejecución en la aplicación.

4. El usuario puede observar los resultados de los planificadores en gráficas que puede guardar como imágenes en formato PNG y JPEG.

5. La aplicación permite al usuario guardar el experimento en cualquier momento y además realiza un auto-guardado durante la ejecución previniendo la perdida de información producida por cualquier error. Además, se puede recuperar la ejecución del experimento simplemente volviéndolo a abrir.

6. Gracias a que la aplicación puede guardar y abrir experimentos en formato XML cualquier usuario puede crear un fichero XML con la información de un experimento no generado con PLEX y cargarlo en la aplicación.

Por lo tanto, prácticamente todos los objetivos que se plantearon al inicio del Proyecto Fin de Carrera se han alcanzado. Sin embargo, el desarrollo de la herramienta ha dado también lugar a nuevas ideas que se podrían incorporar y que se detallan en el siguiente capítulo.

Capítulo 8

Trabajos futuros

En este capítulo se describen posibles mejoras que se podrían introducir en la herramienta para ampliar sus funcionalidades.

1. Lanzar la aplicación de forma distribuida sería una posible mejora de la herramienta que permitiría a un usuario lanzar experimentos y estudiar sus resultados en diferentes servidores desde un mismo ordenador. Para ello sería necesario introducir en la herramienta un cliente y un servidor con un lenguaje a través del cual el cliente pueda indicar al servidor cuál es la configuración del experimento y, a su vez, el servidor al cliente cuál es el resultado del experimento.

2. Producir gráficas *gnuplot* mediante *scripts* generados por la herramienta. Aunque PLEX permite la creación de gráficos, en las publicaciones científicas se suelen utilizar gráficos generados con la herramienta *gnuplot*. Para que la herramienta sea capaz de generar estos gráficos bastaría con generar automáticamente los *scripts* de entrada a *gnuplot*. Para que estos *scripts* fueran parametrizables sería necesario abstraer los parámetros más usuales y generar las correspondientes interfaces gráficas para introducirlos.

3. Exportar la tabla de resultados en diferentes formatos (texto, hoja de calculo, latex, etc) permitiría al usuario poder utilizarla, bien en publicaciones, bien para realizar estudios con otras herramientas, etc...

4. Introducir una descripción en los experimentos permitiría al usuario poder definir textualmente qué es lo que se está haciendo, cuál es el resultado esperado e incluso el estudio de los resultados. Para ello en los experimentos sería necesario introducir un campo que incluyese esa descripción.

5. Mejorar el editor de código java de la herramienta facilitaría al usuario la creación de nuevas operaciones de añadido de columna. Las mejoras del editor podrían incluir:

   - El coloreado de las palabras clave de java.
   - El tabulado de las diferentes líneas de código.
   - El auto-compilado del código que subraye los errores mientras se escribe el código.

6. Añadir otro modo de obtener los valores de salida de un planificador cuando no es posible definir una clave. Un modo podría ser permitir al usuario introducir cómo debe obtener particularmente los datos para ese planificador.

7. Permitir al usuario crear nuevos tipos de gráfica en tiempo de ejecución usando código java. Para ello sería necesario realizar un proceso parecido a la creación de nuevas operaciones de añadido de columnas, siendo necesario crear una librería de funciones que permitiesen al usuario crear gráficas a partir de los datos de la tabla de resultados.

Bibliografía

Blum, A. L. and Furst, M. L. (1995). Fast planning through planning graph analysis. In Mellish, C. S., editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95*, volume 2, pages 1636–1642, MontrÃ©al, Canada. Morgan Kaufmann.

Chen, Y., Hsu, C., and Wah, B. (2006). Temporal planning using subgoal partitioning and resolution in SGPlan. 26:323–369.

Fernánndez, S., Borrajo, D., Fuentetaja, R., Arias, J. D., and Veloso, M. (2007). PLTOOL: a knowledge engineering tool for planning and learning. *The Knowledge Engineering Review*, 22(2):153–184.

Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. 2:189–208.

Fox, M. and Long, D. (2003). PDDL2.1: An extension of PDDL for expressing temporal planning domains. 20:61–124. ISSN 11076-9757.

Gerevini, A., Saetti, A., and Serina, I. (2004). Planning with numerical expressions in LPG. pages 667–671.

Hoffmann, J. (2003). The METRIC-FF planning system: Translating "ignoring delete lists" to numeric state variables. 20:291–341.

Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. 14:253–302.

Vaquero, T. S., Romero, V., Tonidandel, F., and Silva, J. R. (2007). itsimple 2.0: An integrated tool for designing planning domains. In Boddy, M. S., Fox, M., and ThiÃ©baux, S., editors, *ICAPS*, pages 336–343. AAAI.

Apéndice A

Manual de Usuario

En este manual de usuario se va a explicar cómo usar toda la funcionalidad de la herramienta Plex. La herramienta Plex se compone de una serie de carpetas con diferentes utilidades:

- **Carpeta dist:** Esta carpeta contiene el ejecutable que arranca la herramienta así como las librerías que ésta necesita.
- **Carpeta doc:** En esta carpeta se encuentra la documentación de la aplicación.
- **Carpeta domains:** Esta carpeta está destinada a que los usuarios de la herramienta Plex puedan introducir los dominios y problemas que suelen usar, facilitando su acceso desde la aplicación, ya que es la carpeta que, por defecto, usará la aplicación para cargarlos.
- **Carpeta dtds:** Esta carpeta contiene los dtds que usa la aplicación para la creación y lectura de los archivos xml (experimentos y plantillas de planificador).
- **Carpeta experiments:** Ésta es la carpeta por defecto para abrir y guardar experimentos.
- **Carpeta graphics:** Esta es la carpeta que por defecto usará la aplicación para guardar como imágenes los gráficos de los resultados.
- **Carpeta languages:** En esta carpeta se encuentran los idiomas en los que se podrá traducir la herramienta.
- **Carpeta NewUserColumnOperations:** La aplicación generará una carpeta dentro de ésta con toda la información sobre las operaciones de añadido de columna que el usuario genere durante el uso de la herramienta.
- **Carpeta planners:** Esta carpeta esta destinada a que los usuarios de la herramienta Plex puedan introducir los planificadores que suelen usar, facilitando su acceso desde la aplicación ya que es la carpeta que por defecto usará la aplicación para cargarlos. Esta carpeta es la que se utiliza por defecto para abrir y guardar las plantillas de planificador.

Para arrancar la herramienta se debe ejecutar el archivo Plex.jar que se encuentra en la carpeta dist. Para ello se ha de escribir el comando "`java -jar Plex.jar`" en un terminal.

A.1. Nuevo experimento

Con el botón 'New" el usuario creará un nuevo experimento borrando toda la información de cualquier otro que estuviese abierto. En el caso de que haya un experimento abierto cuando se crea un experimento, la herramienta primero preguntará si se desea guardar el experimento actual y después creará el nuevo.

Para crear un nuevo experimento la aplicación no ha de tener ningún resultado abierto.

A.2. Abrir experimento

El botón "Open" permite al usuario abrir experimentos guardados anteriormente. Estos experimentos pueden ser de tres tipos:

- **Experimento en configuración:** Este tipo de experimento es aquel que no ha sido ejecutado todavía, por lo que se podrán modificar, añadir y eliminar planificadores, dominios y problemas. Cuando abrimos este experimento la aplicación nos muestra la ventana de configuración con los datos del mismo.
- **Experimento ejecutándose:** Este tipo de experimento se genera cuando se interrumpe la ejecución de un experimento, por lo que este no termina su ejecución, ya sea de forma manual o inesperada. Gracias al auto guardado que la herramienta Plex utiliza después de la ejecución de cada planificador, la aplicación tiene la capacidad de continuar con la ejecución una vez que se ha abierto el experimento. Cuando abrimos este tipo de experimento la aplicación abre la ventana de configuración del experimento y la ventana de resultados, continuando la ejecución por el último planificador que terminó.

- **Experimento ejecutado:** Este experimento es aquel que ha completado la ejecución de todos los planificadores para todos los dominios y problemas. Cuando se abre este experimento la aplicación muestra la ventana de configuración del experimento y la ventana de resultados.

Si cuando se abre un experimento la aplicación tiene cargado otro que no está guardado, al igual que cuando se crea uno nuevo, la herramienta nos ofrecerá la posibilidad de guardarlo antes de abrir el nuevo.

Puede ocurrir que cuando se abra un experimento alguno de los paths configurados y guardados en el experimento, tanto de dominio, de problema o de planificador, ya no existan. En este caso la aplicación mostrará un error de referencias y se comportará de diferente manera dependiendo del tipo de experimento:

- **Experimento en configuración:** En este caso la herramienta simplemente mostrará que paths son incorrectos para que el usuario pueda corregirlos.
- **Experimento ejecutándose:** En este caso la aplicación, en vez de seguir con su ejecución, permitirá al usuario arreglar los path que están marcados como erróneos. Una vez solucionados los problemas de referencia el usuario podrá continuar con la ejecución mediante el botón "Run".
- **Experimento ejecutado:** En este caso la aplicación permite arreglar los problemas de referencia de los path erróneos, aunque su modificación no repercutirá en el resultado y sólo tendrá validez para poder observar cómo fue la configuración del experimento. Una vez terminados los problemas de referencia el usuario puede pulsar el botón "Run" para que los cambios queden almacenados.

A.3. Guardar experimento

El usuario podrá guardar los experimentos con los botones "Save" y "Save-As". El primero de ellos se usa para guardar el experimento con el nombre que tenga actualmente, en caso de que el experimento no tenga ningún nombre se le pedirá al usuario. Este botón sólo será accesible cuando haya algún cambio que no haya sido guardado en el experimento.

El botón "Save-As" estará disponible para el usuario en todo momento y permitirá guardar el experimento con el nombre que el usuario elija. Una vez utilizado el "Save-As" el experimento que quedará abierto será el que se ha elegido para guardar, quedando cerrado el anterior sin los cambios que se hayan producido desde el último guardado sobre éste.

A.4. Configurar dominios y problemas

Los dominios y problemas podrán ser configurados sólo para "experimentos en configuración" y para experimentos con problemas de referencia, en cuyo caso sólo se permitirá el editado.

A.4.1. Añadir dominio y problemas

Para añadir un dominio y sus problemas se ha de pulsar el botón "New" de la sección "Domains and problems". En ese momento se abrirá una ventana en la que se podrá seleccionar el path del dominio y de los problemas y se podrá seleccionar un nombre de dominio.

Para seleccionar un dominio se mostrará la carpeta "domains" de la herramienta siempre que no se haya seleccionado antes un problema, en cuyo caso se mostrará la carpeta del problema para facilitar la selección del dominio.

En el caso de seleccionar un problema se mostrará la carpeta "domains" de la herramienta siempre que no se haya seleccionado antes un dominio, en cuyo caso se mostrará la carpeta del dominio para facilitar la selección los problemas.

Cuando se elige un dominio, si el usuario aún no ha introducido ningún nombre para el dominio, la aplicación tomará como nombre el de la carpeta que le contiene.

A.4.2. Editar dominio y problemas

Para editar un dominio y sus problemas se ha de pulsar el botón "Edit" de la sección "Domains and problems" y tener un dominio seleccionado en esta misma sección. El pulsado del botón nos mostrará una ventana con toda la información del dominio que se podrá modificar.

A.4.3. Borrar dominio y problemas

Para borrar un dominio y sus problemas se ha de pulsar el botón "Delete" de la sección "Domains and problems" y tener un dominio seleccionado en esta misma sección. Una vez realizada está acción el dominio y sus problemas desaparecerán del experimento.

A.5. Configurar planificadores

Los planificadores podrán ser configurados sólo para "experimentos en configuración" y para experimentos con problemas de referencia, en cuyo caso sólo se permitirá el editado.

A.5.1. Añadir planificador

Para añadir un planificador se ha de pulsar el botón "New" de la sección "Planners". Una vez pulsado se podrá configurar un nuevo planificador introduciendo un nombre, un path, un comando de ejecución, una tabla de nombre-clave y si se obtendrán múltiples soluciones.

A.5.1.1. Seleccionar path del ejecutable del planificador

Para seleccionar un planificador se ha de pulsar el botón "Explore" que nos permitirá la selección de un ejecutable. Una vez seleccionado, si el planificador no tiene un nombre asignado, tomará el del ejecutable.

Además en la sección "Output" se nos mostrará la salida de la ejecución del planificador sin ningún parámetro. Esto se hace porque en la mayoría de los planificadores se muestra de esta manera cuáles son los argumentos que se han de introducir, facilitando al usuario generar el comando de ejecución.

A.5.1.2. Añadir parámetros del comando

Para añadir los parámetros de ejecución del planificador la aplicación muestra al usuario una serie de etiquetas que serán sustituidas por los datos de dominios y problemas:

- **Etiqueta "#DOMAIN_FILE":** Esta etiqueta representa el nombre del dominio (por ejemplo "domain.pddl").
- **Etiqueta "#DOMAIN_PATH":** Esta etiqueta representa el path absoluto hasta el archivo de dominio sin incluirlo (por ejemplo "/home/usuario/Plex/domains/blocks/").
- **Etiqueta "#PROBLEM_FILE":** Esta etiqueta representa el nombre del problema (por ejemplo "pfile1").
- **Etiqueta "#PROBLEM_PATH":** Esta etiqueta representa el path absoluto hasta el archivo de problema sin incluirlo(por ejemplo "/home/usuario/Plex/domains/blocks/").

Con estas etiquetas y con los datos característicos de cada planificador se pueden configurar los argumentos del planificador para su posterior ejecución (por ejemplo para el planificador Lpg:"-o #DOMAIN_PATH#DOMAIN_FILE-f #PROBLEM_PATH#PROBLEM_FILE -n 2 -noout" o para el planificador sgplan: "-o #DOMAIN_PATH#DOMAIN_FILE -f #PROBLEM_PATH#PROBLEM_FILE").

Para que los argumentos del comando sean válidos han de tener como mínimo las etiquetas "#DOMAIN_FILE" y "#PROBLEM_FILE".

Una vez seleccionados unos argumentos válidos, el usuario puede comprobar el funcionamiento del planificador con esos argumentos con el botón "Test". Una vez pulsado, se pedirá un dominio y un problema (se recomienda que sea uno de fácil solución para que no se demore mucho la espera). Después se ejecutará el planificador y se mostrará la salida en la sección "Output" permitiendo al usuario comprobar que funciona correctamente.

A.5.1.3. Crear tabla de nombres-claves

Para que la herramienta Plex pueda obtener los resultados de los planificadores de forma correcta, éstos han de cumplir una sola característica: han de tener un texto único e invariable delante o detrás del valor a obtener.

De este modo para indicar a la aplicación qué valores ha de recoger se ha de rellenar la tabla de nombres-claves. Para ello, en cada fila se debe introducir un nombre para la variable en la primera columna, la clave de texto inequívoca e invariable en la segunda y en la tercera la palabra "pre", si la clave se encuentra por delante del valor a guardar, o la palabra "post" si se encuentra por detrás.

Los nombres de variable han de ser diferentes dentro de un planificador y deberán coincidir con los nombres de variables de los otros planificadores que se configuren en el experimento y que representen la misma variable, para que la herramienta los considere el mismo tipo de datos en el resultado.

Para rellenar la tabla se recomienda usar la salida del planificador que el usuario tendrá en la sección "Output" después de realizar el test de los argumentos del planificador.

A.5.1.4. Múltiples soluciones

En el caso de que se trate de un planificador que obtenga varias soluciones, se le deberá indicar a la herramienta y además decirle como diferenciarlas. Para que la herramienta Plex pueda diferenciarlas la salida del planificador ha de tener una clave de texto único e invariable que separe las soluciones. Esta clase puede ser de varios tipos:

- **Etiqueta "PRE":** Con este tipo de clave, la herramienta ignorará toda la salida hasta que la encuentre, una vez encontrada tomará como solución toda la salida del planificador hasta que la vuelva a encontrar o se termine esa salida (TEXTO_IGNORADO - CLAVE - SOLUCIÓN - CLAVE - SOLUCIÓN - ... - CLAVE SOLUCIÓN).
- **Etiqueta "POST":** Este tipo de clave considera como solución toda la salida encontrada hasta la clave (SOLUCIÓN - CLAVE - SOLUCIÓN - CLAVE - ... - CLAVE - TEXTO_IGNORADO).
- **Etiqueta "SEPARATOR":** Esta etiqueta considera solución todo lo que encuentre antes y después de la clave (SOLUCIÓN - CLAVE - SOLUCIÓN - CLAVE - ... - CLAVE - SOLUCIÓN).

A.5.1.5. Plantillas de planificador

Si todos los valores introducidos en la configuración del planificador son válidos, la herramienta Plex permite guardar éstos como una plantilla de planificador. Esto permite al usuario tener guardadas sus configuraciones más usadas.

Para guardar estas Plantillas el usuario ha de pulsar el botón "Save template" y para cargarlas el botón "Open Template".

A.5.2. Editar planificador

Para editar un planificador se ha de pulsar el botón "Edit" de la sección "Planners" y tener un planificador seleccionado en esta misma sección. El pulsado del botón nos mostrará una ventana con toda la información del planificador que se podrá modificar.

A.5.3. Borrar planificador

Para borrar un planificador se ha de pulsar el botón "Delete" de la sección "Planners" y tener un planificador seleccionado en esta misma sección. Una vez realizada está acción el planificador desaparecerá del experimento.

A.5.4. Añadir tiempo máximo de CPU para cada planificador

El usuario puede poner un límite de tiempo de CPU que cuando sea sobrepasado por un planificador hará que se detenga. Este valor se puede seleccionar en la sección "Planners" de la pantalla de configuración de experimento.

Para realizar esta acción la herramienta usa el comando ulimit, por lo que es necesario que esté instalado en el sistema operativo donde se ejecute la aplicación si se quiere usar esta opción.

A.5.5. Añadir cantidad máxima de memoria para cada planificador

El usuario puede poner una cantidad máxima de memoria que cuando sea sobrepasada por un planificador hará que se detenga. Este valor se puede seleccionar en la sección "Planners" de la pantalla de configuración de experimento.

Para realizar esta acción la herramienta usa el comando ulimit, por lo que es necesario que esté instalado en el sistema operativo donde se ejecute la aplicación si se quiere usar esta opción.

A.6. Ejecutar experimento

Para iniciar la ejecución, el usuario ha de tener configurado un experimento válido y pulsar el botón "Run". Un experimento válido es aquel que tiene uno o más dominios válidos y uno o más planificadores válidos.

Una vez iniciada la ejecución del experimento la herramienta lanzará cada planificador con cada problema para cada dominio. De este modo la herramienta tomará el primer problema del primer dominio ejecutándolo con el primer planificador, después con el segundo, y así hasta completar todos los planificadores; momento en el que se pasará al segundo problema. Cuando acaben los problemas del primer dominio se continuará por el siguiente hasta completar todos los dominios.

Según termine la ejecución de cada planificador su salida se mostrará en la pestaña "Output" y sus resultados se introducirán en la tabla de resultados y en los gráficos contenidos en las pestañas "Result Table" y "Graphic".

A.6.1. Omitir ejecución

Durante la ejecución de los planificadores los botones "Omit" y "Omit All" permanecerán activos. Estos botones permiten al usuario detener la ejecución de un planificador o de todos respectivamente.

En el caso de que se omita un planificador cuando ya ha empezado a ejecutarse, se tendrá en cuenta su solución si le ha dado tiempo a encontrar una antes de que se lleve a cabo su interrupción. Por el contrario si se omite sin que haya comenzado su ejecución el planificador ni si quiera será lanzado.

A.6.2. Gestionar columnas

La gestión de columnas sólo se puede realizar cuando la ejecución del experimento se ha completado desde la pestaña "Result Table".

A.6.2.1. Añadir columna

Para añadir una columna el usuario ha de pulsar el botón "Add Column" y se abrirá una ventana en la que tendrá que elegir el nombre de la nueva columna que tendrá que ser único y elegir de qué modo se va a rellenar esa columna a través de una operación y una serie de argumentos para esa operación. Los

argumentos de la operación han de estar separados por el símbolo " %".

A.6.2.1.1. Crear operación de añadido de columna

Dentro de la ventana de añadido de columnas el usuario puede crear nuevas operaciones de añadido de columnas, para ello ha de pulsar el botón "New". Esto abrirá una ventana en la que se puede dar un nombre a la operación que deberá ser único, una descripción de lo que la operación va a realizar y de los argumentos que necesita y por ultimo, el código java que el usuario debe usar para introducir el suyo.

Para que el usuario pueda acceder a los resultados de los experimentos para generar las nuevas columnas, se le da una serie de métodos java que están definidos en la interfaz Table y definidos en el javadoc de la aplicación.

Si el usuario quiere en su código informar al usuario de su operación de añadido de columnas de algún error deberá hacerlo a través de alguna excepción de java.

Una vez introducido el código java, la herramienta Plex lo compila y prepara para su utilización, pero sólo se carga en la aplicación cuando el usuario lo selecciona para su uso.

A.6.2.1.2. Editar operación de añadido de columna

Para editar una operación de añadido de columna, se ha de seleccionar y pulsar el botón "Edit" de la ventana de añadido de columnas. Esta acción producirá que se abra una ventana con toda la información de la operación permitiendo al usuario modificarla.

Una vez aceptados los cambios se compilarán y se prepararán para su uso, pero no se cargarán en la herramienta hasta que el usuario use esa operación.

A.6.2.1.3. Borrar operación de añadido de columna

El borrado de una operación de añadido de columna se realiza seleccionando y pulsando el botón "Delete" de la ventana de añadido de columnas. Después de pulsar el botón, la herramienta borra la operación seleccionada del disco, no siendo posible su recuperación.

A.6.2.2. Borrar columna

Para borrar una columna el usuario ha de pulsar el botón "Delete Column" de la pestaña "Result Table". Con esto se permitirá al usuario elegir la columna a borrar. Una vez elegida, ésta se eliminará de los resultados del experimento.

A.6.3. Seleccionar soluciones

La herramienta Plex permite al usuario realizar una selección de las soluciones que se han obtenido de los diferentes planificadores para los dominios y problemas con los que fueron ejecutados.

En la tabla de resultados las celdas pueden tener tres colores diferentes:

- **Blanco:** De este color se muestran todas las soluciones que no han sido bloqueadas por el usuario pero que no son las escogidas para la creación de los gráficos.
- **Verde:** De este color se muestran las soluciones escogidas para la creación de los gráficos.
- **Rojo:** De este color se muestran las soluciones que el usuario a desechado a través de alguna regla de selección de soluciones.

A.6.3.1. Seleccionar la prioridad de soluciones

La selección de la prioridad de soluciones se puede realizar en la sección "Solution Priority" de la pestaña "Result Table". En esta sección se permite la selección de una columna y de un operador de prioridad.

El operador de prioridad permite a la herramienta saber qué solución es mejor y el nombre de columna indica en qué variable se ha de fijar para aplicar ese operador. Con esto el usuario podrá elegir qué solución escoger en el caso de que haya varias soluciones para un planificador y las coloreará de verde dejando al resto en color blanco.

A.6.3.2. Gestionar reglas de selección de soluciones

La herramienta Plex permite al usuario rechazar algunas soluciones bajo varios criterios. Las soluciones que el usuario rechaza se pintaran de color rojo. Esta gestión se realiza en la sección "Solution Selector Rules" de la pestaña "Result Table".

A.6.3.2.1. Añadir regla

Para añadir una regla el usuario ha de pulsar el botón "New" de la sección "Solution Selector Rules". Esto abrirá una ventana en la que el usuario podrá elegir una columna, un operador lógico y el valor de una variable.

Una vez creada, la regla aparecerá en el cuadro de texto de la sección y se aplicará en el resultado, dejando fuera del experimento las soluciones que no cumplan la condición introducida en la regla.

En el caso de que una regla se cree sobre una columna y después la columna se borre, esta regla aparecerá de color rojo en el cuadro de texto y se desactivará para que el usuario solucione el conflicto.

A.6.3.2.2. Editar regla

Para editar una regla el usuario ha de seleccionar una de ellas y pulsar el botón "Edit" de la sección "Solution Selector Rules". Esta acción abrirá una ventana con toda la información de la regla que podrá ser modificada por el usuario.

A.6.3.2.3. Borrar regla

Para borrar una regla el usuario ha de seleccionar una de ellas y pulsar el botón "Delete" de la sección "Solution Selector Rules". En este momento la regla quedará inactiva y será borrada de la lista de reglas.

A.6.4. Gestionar filas

La herramienta Plex permite al usuario borrar filas en la tabla de resultados y componer éstos para un resultado común. La gestión de filas sólo se puede realizar cuando la ejecución del experimento se ha completado desde la pestaña "Result Table".

A.6.4.1. Borrar planificador

Para borrar un planificador el usuario ha de pulsar el botón "Delete Planner" en la pestaña "Result Table". Se abrirá una ventana en la que se seleccionará el planificador a borrar. Éste se eliminará de la solución para todos los dominios y problemas.

A.6.4.2. Borrar dominio

Para borrar un dominio el usuario ha de pulsar el botón "Delete Domain" en la pestaña "Result Table". Se abrirá una ventana en la que se seleccionará el dominio a borrar. Éste se eliminará de la solución para

todos los planificadores.

A.6.4.3. Borrar problema

Para borrar un problema el usuario ha de pulsar el botón "Delete Problem" en la pestaña "Result Table". Se abrirá una ventana en la que se seleccionará el problema a borrar y el dominio al que pertenece. El problema se eliminará de la solución para todos los planificadores pero sólo para el dominio seleccionado.

A.6.4.4. Crear composición de planificadores por mediana

La herramienta Plex permite la generación de composición de planificadores por su mediana. Esto es especialmente útil para planificadores estocásticos (que en cada ejecución dan una solución) ya que se permite al usuario crear un experimento con el mismo planificador y después escoger el resultado que indica la mediana de todos estos planificadores.

Para la creación de una composición es necesario pulsar el botón "Planner Composition By Median". Este botón genera una ventana en la que podemos elegir un nombre para el planificador que se creará, de qué planificadores estará compuesto y sobre qué columna se estudiarán los valores. Una vez seleccionadas las características de la composición aparecerá un nuevo planificador en la tabla de resultados con el nombre que se ha elegido. Este nuevo planificador tendrá sólo valores para la columna seleccionada. Los valores obtenidos serán el resultado de hacer la mediana sobre los valores de la misma columna de los planificadores seleccionados para la composición.

La mediana se realizará dependiendo de la cantidad de datos sobre los que se realice. Así pues si se tienen, para un dominio y un problema, un número par de soluciones seleccionadas (soluciones en verde en la tabla) con valor (la celda de la tabla no puede estar vacía), el planificador creado tendrá dos soluciones que se corresponderán con los valores centrales de las soluciones seleccionadas con valor. En caso de que el número sea impar el planificador creado sólo tendrá una solución con el valor central de las soluciones seleccionadas con valor.

A.6.4.5. Modificar gráficos

La herramienta Plex genera dos tipos de gráficos con los datos obtenidos en la solución, un gráfico con los datos de la tabla y uno probabilístico. Estos gráficos se pueden observar y modificar en la pestaña "Graphics".

En esta pestaña se puede elegir qué dominio, qué variable, qué planificadores y qué problemas se van a dibujar en el gráfico. Los planificadores y problemas no seleccionados para ser dibujados se considerarán como si no estuviesen en el resultado para la creación de los gráficos.

También se permite al usuario guardar el gráfico como imagen jpeg o png. Pulsando el botón "Save Graphic" nos da la opción de elegir el tamaño de la imagen y dónde se desea guardar; por defecto se guardará en la carpeta "Graphics" de la herramienta.

A.7. Cambiar idioma de la herramienta

Para cambiar de idioma el usuario ha de seleccionarlo en la parte superior izquierda de la ventana de configuración de experimentos. Sólo se puede modificar el idioma cuando el experimento se encuentra en fase de configuración.

A.7.1. Crear un nuevo idioma para herramienta

Los usuarios de la herramienta Plex podrán traducir la aplicación a distintos idiomas. Para ello lo único que ha de hacer es incluir un archivo en la carpeta "languages". Este archivo ha de tener como nombre el idioma y tendrá que contener traducidas todas las claves que se pueden observar en los ficheros "english" y "español".

A.8. Requisitos de software

Para poder utilizar toda la funcionalidad de la herramienta Plex es necesario que el sistema operativo donde se ejecute tenga las llamadas "kill", "ps" y "ulimit", necesarias para que se puedan ejecutar los planificadores sin que sobrepasen un límite de tiempo, un límite de memoria y poder parar su ejecución en caso de error evitando que queden procesos "zombies".

Por otro lado, es necesario el compilador de java 1.6 para poder compilar el código que el usuario introduzca en tiempo de ejecución y así poder cargarlo en la aplicación.

Apéndice B

Gestión del proyecto

En este anexo se describe la gestión y planificación del proyecto de construcción de la herramienta PLEX.

B.1. Recursos humanos

En esta sección se describen las personas que han intervenido en este proyecto. Además se detalla el coste asociado a cada uno de sus puestos de acuerdo a las tarifas estipuladas en el BOE 063, de 13/03/2008, sección 3, páginas 15242 a 15243. Se considera el importe bruto para cada participante.

Nombre	Siglas	€/Hora
Jefe de Proyecto	JP	35
Analista	AN	33
Programador	PR	29

Tabla B.1: Recursos humanos

B.2. Planificación

En esta sección se muestra el diagrama de Gantt (figuraB.1) que comprende la planificación de la consecución de las distintas fases del proyecto, así como la tabla B.2 con las diferentes tareas, su duración y el coste humano asociado a ellas.

Tarea	Inicio	Fin	Trabajo	Duración	Coste	Asignado a
1 Diseño e implementación de PLEX	oct 12	oct 18	230d	228d	57032,57	
1.1 Motivación	oct 21	nov 9	5d	13d 7h	1331,11	AN, JP
1.2 Objetivos	oct 21	nov 9	5d	13d 7h	1331,11	AN, JP
1.3 Estado de la cuestión	oct 12	nov 16	14d	25d 3h	3716,6	
1.3.1 Planificación Automática	oct 12	oct 20	7d	7d	1853,6	AN, JP
1.3.2 Herramientas	oct 21	nov 16	7d	18d 3h	1863	AN, JP
1.4 Análisis	nov 16	ene 20	35d	35d	9268	AN, JP
1.5 Diseño	ene 20	mar 8	33d	33d	8738,4	AN, JP
1.6 Implementación	mar 8	oct 7	68d	128d 2h	16083,95	JP, PR
1.7 Validación	mar 8	oct 18	70d	134d 4h	16563,41	JP, PR
2 Redacción de la memoria	nov 19	oct 18	30d	200d	7760	AN, JP, PR

Tabla B.2: Tareas del proyecto

B.3. Recursos materiales

En la tabla B.3 se enumeran los recursos materiales consumidos en el desarrollo del proyecto teniendo en cuenta la devaluación e impuestos pertinentes sobre los precios que se muestran. Además, se señalan directamente el coste total que ha supuesto cada artículo al proyecto, es decir, se han tenido en cuenta todas las unidades requeridas:

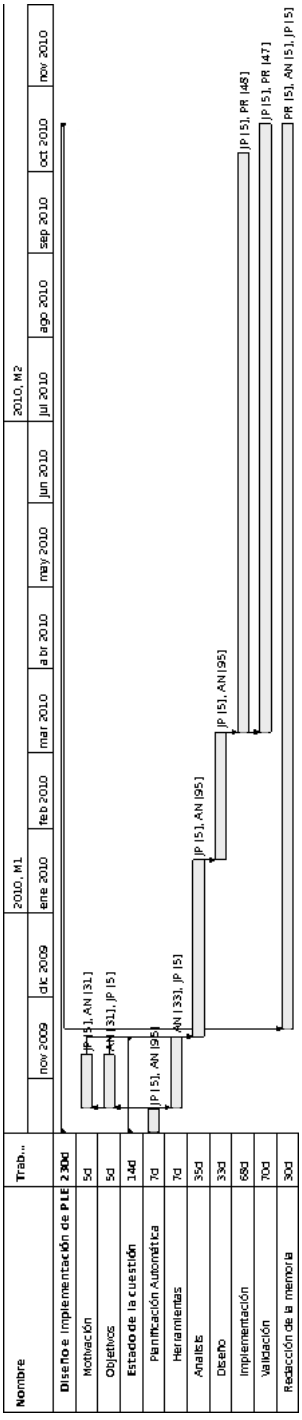

Figura B.1: Diagrama de Gantt

Recurso	**Precio Final**
Ordenador	1500
Memoria USB	20
Material de oficina	50
Impresora de inyección	50
Cartuchos de tinta	30
Total	1650

Tabla B.3: Recursos materiales