Proyecto de
Sistemas Informáticos

SERVICIO DE META-PLANIFICACIÓN GRID

2005-2006

Autores:
Alberto González Martínez
Isabel Fernández Utrilla
Verónica Gonzalo Sanz

Profesor director:
Ignacio Martín Llorente

Universidad Complutense Madrid

Facultad de Informática

RESUMEN

La Grid Computing es una nueva tecnología que comienza a crecer, junto con las múltiples posibilidades que puede ofrecer a cualquier desarrollador. Junto con las opciones que ofrece esta tecnología en cuanto a compartición de recursos surge la aplicación Word.

Esta aplicación se aprovecha de los recursos de las distintas máquinas del Grid para la ejecución de un trabajo. Parte de una máquina supervisora en la que comienza la ejecución y a partir de ella se va copiando y ejecutando en otros equipos.

A la par que Worm se ejecuta puede ir acompañada de cualquier otro trabajo programado que el desarrollador pretenda correr en las distintas máquinas.

Esta aplicación es capaz de clonarse y ejecutarse a través de los equipos establecidos, siempre bajo el control de la máquina supervisora que monitoriza sus trazas y posibles errores.

La máquina supervisora mantiene parte de la aplicación en constante ejecución para poder interaccionar con el usuario a través de la interfaz gráfica. La información principal que se refleja son las trazas de evolución de la aplicación y la muestra de errores que serán detectados y/o gestionados.

La aplicación se ha desarrollado en Java y se ha basado en las API`s de Globus Toolkit 4.0 para todo lo relacionado con Grid.

SUMMARY

Grid Computing is a new technology that right now is growing together with the multiple possibilities that it can offer to any developer. The application Worm arises together with the options that this technology can offer as for sharing resources**.**

This application takes advantages of the resources of the different machines of the Grid for the execution of a job. It begins its execution in a machine which supervises all the cycle of execution and from that machine it is copied and executing in other equipments. At the same time that Worm is executed it can be accompanied of any another programmed job that the developer tries to run in the different machines.

Worm application is capable to transfer the code and execute it across the established equipments, always under the control of the machine that supervises and monitors all the traces and possible errors.

The first machine supports a part of the application in constant execution so that user has access to the graphical interface without any interruption. The traces of evolution of the application and the errors that will be detected and/or managed for the application are the principal information that the interface reflects.

Worm application has been developed in Java and has been based on the APIs that Globus Toolkit 4 offer for everything related to Grid.

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Firma de los autores:

Alberto G. M. Isabel F. U. Verónica G. S.

ESPECIFICACIÓN DE REQUISITOS SOFTWARE

S R S

1. INTRODUCCIÓN

El primer paso de este documento es dar una idea general de qué se encuentra en su interior, así como introducir al lector del mismo para proporcionarle una idea sobre en que consiste el proyecto y los pasos de construcción del mismo.

1.1. PROPÓSITO

Esta Especificación de Requisitos se realiza con la finalidad de definir de la forma más clara posible el funcionamiento y manejo, en base al que se diseña el producto.

Esta documentación se desarrolla principalmente para establecer un escrito en el que se registren todas las condiciones y restricciones de funcionamiento de la aplicación. Como resulta de gran dificultad predefinir de forma exacta todos los temas relacionados con la funcionalidad del proyecto ha de entenderse que esta especificación se ha ido ampliando según se han superado las distintas etapas de construcción del mismo.

Está orientado a cualquier persona que desee conocer el funcionamiento de la aplicación que se va a desarrollar, y en específico al cliente al que va a ir destinado el producto.

1.2. ALCANCE

El producto que se espera resulte de la realización de este proyecto consiste en una aplicación que realiza una función de autoejecución entre una secuencia de máquinas pertenecientes al Grid.

Esta *aplicación Worm* tiene como principal funcionalidad el propagarse por otras máquinas predefinidas siendo capaz de ejecutarse en ellas, recopilar información de las mismas y regresar finalmente a la primera en la que comenzó a ejecutarse obteniendo como resultado el recopilatorio de información de su recorrido para mostrársela al usuario por pantalla.

La ejecución y propagación de la *aplicación gusano* a través del Grid se basa en las directrices de juego limpio que implica la no realización de acciones perjudiciales para otras máquinas de esta red.

1.3. DEFINICIONES

1.3.1. CONTRATO

Antes del inicio del proyecto, junto con la solicitud de adjudicación del mismo, todos los miembros del equipo firmaron un escrito que restringía los posibles derechos posteriores sobre los resultados de la realización de la aplicación.

A parte, el principal contrato se basa principalmente en el compromiso existente entre el grupo de proyecto y el profesor que lo supervisa, que se limita a un compromiso verbal.

1.3.2. CLIENTE

Se puede considerar cliente al profesor Ignacio Martín Llorente, que es la persona responsable de supervisar y verificar todo el desarrollo del proyecto. Así mismo es también la persona que debe aprobar todo el trabajo realizado a la finalización del proyecto.

1.3.3. PROVEEDOR

Como proveedor se considera al grupo de proyecto, constituido por tres miembros, que son los responsables de que el proyecto se lleve a cabo.

1.3.4. USUARIO

El usuario resulta difícil de definir ya que al tratarse de un proyecto de fin de carrera no está dedicado a un usuario comercial.

Se podría decir que inicialmente el usuario resulta ser el propio cliente, pero mirando hacia el futuro el usuario del producto puede centrarse principalmente en personas dedicadas a la investigación o desarrolladores de Globus.

1.4. REFERENCIAS

El desarrollo de este documento está basado en:

IEEE Std. 830–1998: Especificaciones de los requisitos del software.

1.5. APRECIACIÓN GLOBAL

Este documento pretende documentar la descripción global inicial así como cualquier otra ampliación realizada con la evolución del trabajo. Para ello este documento contiene dos apartados principales:

- **Descripción Global**: que describe los factores generales del producto y hace más comprensible las funciones y requisitos que engloba la aplicación.Se desglosa a su vez en los siguientes subapartados:
  - *Perspectiva del producto:* establece posibles relaciones con otros productos definiendo así un contexto.
  - *Funciones del producto*: hace una descripción inicial de las funciones que el software realizará.
  - *Características de usuario*: describe características generales de los usuarios del producto.
  - *Restricciones*: realiza una apreciación de aspectos que pueden limitar el trabajo de los diseñadores.
  - *Supuestos y dependencias*: identifica factores que afectan a los requisitos de la SRS.
  - *Requisitos futuros*.

- **Requisitos Específicos**: contiene la descripción de los requisitos identificables del software con el suficiente detalle para permitir el diseño de un sistema que los satisfaga. Se desglosa a su vez en los siguientes subapartados:
  - *Requisitos funcionales*: define las acciones fundamentales que tendrán lugar en el software.
  - *Requisitos de rendimiento*: posibles requerimientos numéricos de rendimiento.
  - *Requisitos de la base de datos*.
  - *Restricciones de diseño*: señala restricciones que se imponen en el diseño.
  - *Atributos del sistema software*: atributos del sistema software que pueden servir como requisitos.

2. DESCRIPCIÓN GLOBAL

Este apartado pretende dar una descripción general de aquellos factores y requisitos que afectan al producto.

2.1. PERSPECTIVA DEL PRODUCTO

2.1.1. INTERFAZ DEL SISTEMA

El producto resultante de este proyecto no es un componente de un sistema mayor sino que es una aplicación independiente en sí misma.

La aplicación no depende de ninguna otra aplicación, si no que tendrá un funcionamiento autónomo.

2.1.2. INTERFAZ HARDWARE

La aplicación requiere que todas las máquinas por las que se extienda el gusano deben pertenecer al GRID, principalmente la máquina inicial ya que tendrá un papel especial.

2.1.3. INTERFAZ SOFTWARE

Para la gestión: La aplicación se ve restringida al sistema operativo Linux.

2.1.4. INTERFAZ DE COMUNICACIÓN

Los equipos deben pertenecer a Grid.

2.1.5. MEMORIA

No se requiere gran capacidad para la carga de la aplicación.

2.2. FUNCIONES DEL PRODUCTO

La aplicación gusano debe de realizar una serie de funcionalidades principales que se recogen en el siguiente listado:

- **Capacidad de reiniciar su ciclo de vida en una nueva máquina**:Debe de tener la capacidad de preparar todo aquello que sea necesario para que una vez finalice su ciclo de vida en la máquina actual pueda continuar con otro ciclo de vida en una nueva máquina.Esta es la principal característica que hace que se pueda definir a la aplicación como *gusano*. Englobará dos subcapacidades principales que hacen posible el reiniciar su ciclo de vida:
  - *Capacidad de clonación*:El propio gusano debe de ser capaz de clonarse a sí mismo en la siguiente máquina en la que se va a propagar. Esta capacidad constituye el primer paso de la regeneración en otra máquina.
  - *Capacidad de ejecución*: La aplicación debe de ser capaz de lanzarse de nuevo en la siguiente máquina a partir del momento en el que ha finalizado su clonación.

- **Capacidad de recopilación de información**:Está directamente relacionada con la capacidad de ejecución aunque supone una función adicional a ella. A partir de la ejecución de la aplicación se prodrá recopilar, desde la máquina en la que reside, la información que sea necesaria para irla propagando hasta el momento de su regreso a la máquina de partida.

▪ **Capacidad de destrucción**:Una vez el gusano ha realizado todas sus funciones en la máquina en la que reside, se destruirá a sí mismo dejando así el mínimo rastro de su paso y sus efectos.

2.3. CARACTERÍSTICAS DEL USUARIO

Se pueden distinguir dos tipos de usuarios de la aplicación:

▪ **Usuario con conocimientos técnicos**:Es el principal tipo de usuario final al que va dirigida la realización del proyecto.Este usuario tendrá conocimientos medios/altos de programación así como conocimientos previos sobre Globus Toolkit y Grid Computing. Toda la base técnica que el usuario posea será la que le hará apreciar en menor o mayor medida el contenido de la aplicación y lo que es capaz de realizar. A su vez, este usuario podrá utilizarla o basarse en ella para futuros trabajos de investigación y/o desarrollo si fuera preciso.

▪ **Usuario con escasos/ninguno conocimientos técnicos**: Es un tipo de usuario que de forma secundaria puede tener en algún momento acceso a la aplicación. Al carecer de conocimiento técnicos será incapaz de reconocer de forma clara gran parte del funcionamiento del "Worm" ya que la mayor parte de su ejecución completa no es inmediatamente visible para el usuario. Este tipo de usuario sólo podrá valerse de la información obtenida del ciclo completo de vida del gusano a través de su recorrido.

2.4. RESTRICCIONES

2.4.1. POLÍTICAS DE REGULACIÓN

Debido a que la finalidad básica de la aplicación consiste en expandirse y ejecutarse a través de varias máquinas, queda previamente establecido que su ejecución en cualquiera de las máquinas no debe tener efectos nocivos para ésta y sus contenidos.

2.4.2. REQUISITOS DE LENGUAJE DE ALTO NIVEL

Java, RSL, GSI. Uso de Linux como sistema operativo y el API de Globus Toolkit.

2.5. LIMITACIONES HARDWARE

Uso de equipos cuyo sistema operativo es Linux, y tienen conectividad a la red GRID.

2.5.1. REQUISITOS DE FIABILIDAD

La aplicación debe propagarse, ejecutarse en cada una de las máquinas sin que su anidamiento en esta suponga problemas para la misma.

2.5.2. CREDIBILIDAD DE LA APLICACIÓN

La posibilidad de recuperación de la aplicación en caso de verse interrumpida sin haber finalizado el recorrido del gusano queda pendiente para posteriores iteraciones.

2.5.3. FUNCIONES DE CONTROL

La ejecución del gusano a través de las distintas máquina llevará un control que permita conocer datos mínimos de su ejecución como son: en qué máquinas se ha anidado y el orden de ejecución en las mismas.

2.5.4. SEGURIDAD Y CONSIDERACIONES DE SEGURIDAD

La seguridad la proporciona la red GRID a través de GSI (Grid Security Infraestructure): no se podrá hacer nada si no se ha inicializado el proxy y las transferencias de datos son seguras desde el momento que se usa GridFTP.

2.6. REQUISITOS FUTUROS

La descripción dada hasta el momento dibuja la base del proyecto junto con sus funcionalidades básicas. Con posterioridad se espera incluir nuevas funcionalidades en la aplicación o perfeccionar las ya definidas.

3. REQUISITOS ESPECÍFICOS

En esta sección se entra en un nivel de detalle que permita el posterior diseño del sistema.

3.1. INTERFACES EXTERNAS

- **Con el hardware de los sistemas**: La aplicación se ejecuta sobre una arquitectura Grid haciendo uso del middleware Globus Toolkit 4.0.

- **Con el usuario**: El usuario no podrá tener gran control sobre el gusano ya que actuará de una forma programada y automatizada.Inicialmente se establece la principal comunicación con el usuario a través del fichero que le proporcione la información que se recopile en la ejecución.No se prevee que el usuario proporcione estímulos al gusano ya que las entradas, principalmente el listado de máquinas, estarán preestablecidas.

3.2. FUNCIONES

Función 1: Manejo del listado de máquinas para el recorrido del gusano.

Función: Obtención del listado de máquinas

Descripción: La aplicación debe de tener acceso a un listado de máquinas preestablecido que será el mapa de ruta del gusano.
Entrada: Dirección del archivo de datos con el listado de máquinas.
Salida: Operatividad del gusano para seguir el recorrido de dicho listado.
Origen: Localización en la que se encuentre el archivo.
Destino: La propia aplicación y la nueva reubicación.
Necesita: Archivo que contenga el listado.
Precondición: Que el archivo exista y sea accesible por la aplicación.
Poscondición: Posibles modificaciones en el listado.
Efectos laterales: El archivo o su localización pueden verse modificados.

Función 2: Manejo de la información de salida que debe proporcionar el gusano.

Función: Inferencia de la información de salida.

Descripción: Con el paso del gusano por cada una de las máquinas añade al resultado de la aplicación cierta información de su recorrido.
Entrada: Localización del archivo de salida de información.
Salida: Inclusión de nueva información en el archivo de salida.
Origen: Localización en la que se encuentre el archivo.
Destino: La propia aplicación y la nueva reubicación.
Necesita: El archivo en caso de que ya exista o su creación en caso de no existir.
Precondición: Que la localización del archivo sea accesible por la máquina para poder utilizarlo.
Poscondición: Modificaciones en el archivo de salida consistentes en nueva información sobre el recorrido.
Efectos laterales: El archivo y su localización se verán modificados.

Función 3: Transmisión de los archivos que se precisan para el funcionamiento de la aplicación.

Función: Transmisión de archivos.

Descripción: Transmisión de toda la jerarquía de archivos de la que hace uso la aplicación a través de dos máquinas de Grid.
Entrada: Localización original de la jerarquía de archivos.
Salida: Nueva ubicación para la jerarquía de archivos.
Origen: Localización original de la jerarquía de archivos.
Destino: La siguiente máquina en el recorrido del gusano.
Necesita: Conocer la siguiente máquina a la que se debe transferir.
Precondición: Que la localización de los archivos sea accesible y que la siguiente máquina cumpla todos los requisitos para una transmisión viable.
Poscondición: La siguiente máquina dispondrá de todos los ficheros necesarios para poder ejecutar el gusano.

Función 4: Relanzamiento del gusano en una nueva máquina.

Función: Relanzamiento de la aplicación.

Descripción: Desde la máquina en la que actualmente reside el gusano y una vez realizada la transmisión de archivos a la siguiente máquina relanza su ejecución en esta antes de finalizarla en la actual.
Entrada: Siguiente máquina en el ciclo de ejecución.
Salida: Ejecución del gusano en otra máquina.
Origen: Listado de máquinas para conocer los datos de la siguiente.
Destino: Ejecución en la nueva máquina.
Necesita: El listado de máquinas y archivos transmitidos.
Precondición: Que la siguiente máquina esté activa y accesible y que la transmisión de archivos se haya finalizado satisfactoriamente.
Poscondición: Nueva ejecución en el ciclo de vida del gusano en otra máquina.
Efectos laterales: Todos los que conlleva la ejecución de la aplicación.

3.3. REQUISITOS DEL DESARROLLO

3.3.1. Número de usuarios del sistema

La aplicación gusano será monousuario. La aplicación no podrá ser accedida y/o manejada por varios usuarios, sólo accederá a ella y a los datos de su ciclo de vida aquél que la lanza.

3.3.2. 3.3.2 Tipo de información que se maneja

La aplicación se manejará principalmente a través de archivos que contengan los datos necesarios para la ejecución del gusano. A continuación se describen los principales archivos que se estiman necesarios para la aplicación.

- **Archivos de máquinas**: uno o más archivos (dependiendo del diseño y/o implementación) debe contener el listado de las máquinas por las que ha de pasar el gusano. Este archivo realiza la función de mapa de ruta y por lo tanto debe describir claramente el orden en el que el gusano trascurrirá por las distintas máquinas.

- **Archivos de salidas**: uno o más archivos (dependiendo del diseño y/o implementación) que contendrán los datos referentes al ciclo de vida del gusano. Este archivo/s contendrá los datos que la aplicación irá acumulando como resultado de su paso por los distintos equipos.

- **Archivos de código**: uno o más archivos (dependiendo del diseño y/o implementación) que contendrán el propio código de la aplicación que deberá ir clonándose según transcurra la ruta para poder se ejecutado.

Por lo tanto, los datos que manejará la aplicación estarán registrados en archivos para que se puedan pasar de forma cómoda de una máquina a otra. Los anteriormente enumerados serán los tipos de archivos de los que se servirá la aplicación para gestionar su funcionamiento pero queda pendiente para las etapas de diseño e implementación la decisión de determinar de forma exacta el número de archivos necesarios y la distribución concreta de los datos dentro de ellos.

3.4. REQUISITOS DEL BANCO DE DATOS LÓGICOS

La aplicación carece de una base de datos claramente establecida. En ausencia de esta tal y como ya se ha nombrado en este documento, los datos necesarios serán especificados a través de la gestión de archivos, distinguiendo dos flujos de datos distintos:

- **Datos de configuración de la aplicación***:*Los datos de configuración de la aplicación consistirán principalmente en mantener registrado el listado de las máquinas que han de seguir durante el recorrido de ejecuciones. Se puede estimar que probablemente sea necesario precisar algunas propiedades más que sean constantes mientras dure la vida de la aplicación. Cualquier propiedad que se necesite establecer cuando llegue el momento de la implementación será registrada en un archivo ya que ésta es la forma más práctica de transmitir de una máquina a la siguiente.

- **Datos de salida de la aplicación***:*Igual que los datos de configuración los datos de salida se irán registrando de forma acumulativa en un archivo.El archivo será creado o reseteado (en caso de existir) al iniciar la vida del gusano y será ampliado con cada paso que se realice a través de la ruta.

De esta forma el banco de datos sobre el que se establece la aplicación no es una base de datos, como suele ocurrir en aplicaciones más comerciales o de otra índole, sino que se fija sobre un sistema de archivos que en su momento estará claramente definido tanto en estructura como en formato (dependiendo de las posteriores necesidades de implementación).

3.5. RESTRICCIONES DEL DISEÑO

La aplicación no consta de demasiadas restricciones de diseño, a continuación se anotan las más importantes a tener en cuenta.

- El sistema operativo sobre el se establece la aplicación ha de ser Linux.

- La aplicación consiste en un gusano que se clona y ejecuta a través de Grid, por lo tanto queda como restricción que absolutamente todas las máquinas del listado de ruta deben de pertenecer a Grid.

- Para todas las máquinas del Grid por las que debe pasar la aplicación debe tener la autorización necesaria que le permita obtener las credenciales para poder ejecutarse en ella.

3.6. ATRIBUTOS DEL SOFTWARE DEL SISTEMA

3.6.1. FIABILIDAD

La aplicación no requiere ningún punto concreto de fiabilidad, su correcto funcionamiento está principalmente motivado por la disponibilidad y seguridad.

3.6.2. DISPONIBILIDAD

Como principal punto crítico de disponibilidad hay que destacar el caso más probable que es aquel en el que alguna de las máquinas de la lista de ruta no se encuentre accesible, por el motivo que sea, para la aplicación.

La aplicación debe garantizar que en caso de que haya un contratiempo como el citado en su ruta su ciclo de ejecuciones no se vea interrumpido de forma que el gusano quede parado sin terminar el recorrido.

En este caso, dependiendo de la naturaleza de la interrupción, el gusano debe ignorar la máquina por la que no puede pasar o bien ir directamente al último paso de su recorrido que es nuevamente la primera de las máquinas.

3.6.3. SEGURIDAD

La aplicación no contará con medidas concretas de recuperación como protección de accesos accidentales o malintencionados ya que el único caso que se contempla es la pérdida del archivo que contiene los datos de la ruta y este resulta fácil de rehacer, por su simplicidad, en caso de pérdida.

La propia aplicación debe respetar a su vez cualquier aspecto que pueda perjudicar a la máquina por la que pasa basándose en una política de *fair play*.

3.6.4. MANTENIMIENTO

Para facilitar las modificaciones y mantenimiento de la aplicación esta será organizada en módulos distinguibles.

Deben mantenerse de forma diferenciada todo lo que refiera a:

- *Gestión de información*:
  - Información de configuración.
  - Información de salida.
- *Gestión de la vida del gusano*:
  - Clonación del gusano.
  - Ejecución del gusano.

De esta forma la modularidad del sistema permitirá de forma más localizada cualquier modificación futura.

3.6.5. PORTABILIDAD

El principal problema de portabilidad de la aplicación reside en que se soporta sobre Linux, Grid y Globus Toolkit.

3.7. ORGANIZACIÓN DE REQUISITOS ESPECÍFICOS

3.7.1. MODO DEL SISTEMA

La aplicación gusano consta de un único modo de funcionamiento. No distingue entre usuarios comunes y administradores como típicamente ocurre en otro tipo de sistemas.

De esta forma en su único modo de funcionamiento el gusano será iniciado en su transcurso a través de las distintas máquinas preestablecidas y a posteriori se obtendrán los datos de salida que proporciona la aplicación.

3.7.2. CLASES DE USUARIO

Al existir un único modo de funcionamiento no resulta posible distinguir los usuarios, en función de sus opciones a la hora de acceder a la aplicación.

De esta forma, como ya hemos nombrado anteriormente, las distintas clases de usuarios se fundan en la medida en la que van a comprender el funcionamiento de la aplicación más allá de los simples resultados que se muestren como salida. Esta comprensión dependerá de su formación técnica que cuanto más completa sea le permitirá analizar con mayor conocimiento y/o precisión cualquier resultado que la ejecución de la aplicación le pueda proporcionar.

3.7.3. ESTÍMULO

El principal, y se podría decir que único, estímulo de la aplicación es la orden por parte del usuario de iniciar la ruta de ejecuciones. Esto se debe a que en su mayor parte todas las variables que pueden afectar a la ejecución estarán preestablecidas para asegurar en la medida de lo posible su correcto funcionamiento.

3.7.4. CONTESTACIÓN

Como resultado la aplicación ofrecerá todos aquellos datos que ha recabado a lo largo de su paso por el itinerario de máquinas establecido.

WORM APPLICATION

Contenido Proyecto

1. INTRODUCCIÓN

La presentación de este documento supone el último paso del trabajo que hemos realizado en nuestro proyecto de fin de carrera.

Estas páginas pretenden describir y detallar todo lo que se puede suponer sobre el desarrollo de este proyecto. Se pretende documentar de forma clara todos los aspectos generales y técnicos que engloba la aplicación gusano que ha sido diseñada, especificada e implementada durante el transcurso de este año.

De esta forma esperamos que toda pregunta que se pueda plantear sobre nuestro trabajo quede aclarada después de analizar la documentación que aquí se presenta.

2. FUNCIONAMIENTO GENERAL DEL PROYECTO

La primera descripción del funcionamiento general de la aplicación fue planteada en el documento de especificación de requisitos software con carácter previo a la implementación del proyecto. Ahora con la aplicación final en la mano resulta posible concretar de forma más exacta los principales aspectos del funcionamiento de la misma.

2.1. USUARIOS DE LA APLICACIÓN

Al ser esta una aplicación que inicialmente no posee fin comercial alguno resulta más difícil clasificar las características de un posible usuario final. No obstante podríamos diferenciar nuevamente, como ya se hizo en la especificación de requisitos, a los usuarios en función de qué conocimientos técnicos, útiles en el dominio de la aplicación, poseen.

Entendiendo que para clasificar dichos usuarios es de utilidad detallar los principales conocimientos que abarca el dominio del proyecto. Los conocimientos más importantes son los relacionados con Grid, Globus Toolkit 4 y programación en Java.

En base a estos, y otros conocimientos técnicos del usuario entenderemos que facilita la mejor apreciación y análisis de la aplicación.

En la especificación de requisitos inicial se distinguió entre dos tipos de usuarios, aquellos que poseían conocimientos técnicos y aquellos que no tenían, o estos eran escasos. Ahora vamos a hilar más fino y vamos a distinguir, bajo el mismo concepto de clasificación, entre tres tipos de usuarios.

- Usuarios con conocimientos técnicos generales de informática y con conocimientos específicos sobre Grid y Globus Toolkit. → Estarán en posición de poder analizar cualquier aspecto de la aplicación y de entender de forma completa su funcionamiento.Se posibilita la fácil reutilización de código en caso de ser preciso.

- Usuarios con conocimientos técnicos generales de informática pero sin conocimientos específicos sobre Grid y Globus Toolkit. → Entenderán de forma general el funcionamiento de la aplicación pero no apreciarán detalles técnicos relacionados con el uso de Globus Toolkit.Se posibilita la reutilización de código, aunque no parece probable.

- Usuarios sin conocimientos técnicos generales de informática. → No apreciarán los detalles técnicos de la aplicación y serán meros espectadores de la ejecución de la misma.

- No realizará reutilización de código. Realmente estos usuarios serán muy esporádicos y prácticamente inexistentes puesto que la aplicación será destinada a entornos de investigación y desarrollo.

A pesar de esta clasificación de posibles usuarios hay que destacar que el uso de la aplicación a través del interfaz de usuario es el mismo independientemente de en cual de las categorías encaje. La diferencia radica en la apreciación técnica y en la posible reutilización de código por parte del usuario en posteriores desarrollos.

2.2. ESTRUCTURA GENERAL Y FUNCIONALIDADES

Recordaremos una vez más antes de profundizar en el diseño de la aplicación en qué consiste su funcionamiento general. La aplicación se estructura en una jerarquía de carpetas que contienen principalmente ficheros para el establecimiento de la configuración y ficheros de código. Todos estos ficheros se encuentran inicialmente en la primera máquina del recorrido que juega un papel especial en la aplicación. De esta forma podemos ver la lista de máquinas como una cola circular en la que ésta máquina ocupa el primer y último lugar del ciclo. La máquina inicial representa un papel especial que pasaremos a desglosar a continuación.

- **Máquina inicial del recorrido**:Esta máquina debe cumplir principalmente dos requisitos iniciales:
  - Ha de contener de forma íntegra la jerarquía de archivos que constituyen la aplicación.
  - Debe estar conectada al Grid.

En esta máquina se van a llevar a cabo las siguientes funciones de la aplicación:
- Inicio de la ejecución de la aplicación.
- Información del listado de máquina que conforman la ruta.
- Monitorización de todos los resultados que la aplicación obtenga del ciclo de ejecuciones del gusano a través de las máquinas.
- Finalización de la ejecución de la aplicación.
- Monitorización y corrección de los posibles errores de la ejecución.

Todas las tareas en las que participa activa (ej. establecimiento de máquinas del recorrido) o pasivamente (ej. observación de la información) el usuario se gestionan en esta máquina ya que es la que ejecuta la interfaz gráfica.

Por lo tanto, podemos ver como esta es la máquina en la que comienza y termina la aplicación realizando la función de cabeza visible de cara al usuario.

El usuario verá toda la información relacionada con las ejecuciones a través de esta máquina que juega un papel activo desde que comienza hasta que finaliza el ciclo.

- **Resto de máquinas de la cola**:Son las máquinas que la aplicación va extrayendo del listado que posee.Estas máquinas no tienen por qué contener los archivos que precisa la aplicación ya que la máquina anterior se los traspasará para posteriormente lanzar la ejecución en ella.El usuario de la aplicación no accederá de forma directa a esta máquina en ningún momento, sólo a través de la máquina principal obtendrá información de cómo ha evolucionado la clonación y ejecución en esta máquina.

Llegado este punto se espera que el lector pueda formarse una idea general de cómo el gusano va a ir clonándose y ejecutándose una máquina tras otra para regresar siempre a la primera de ellas.

A continuación se verá una estructuración en módulos de la aplicación analizando las funcionalidades que se encuentran en cada una de ellas.

2.2.1. Módulo de credenciales

Este módulo acredita la aplicación para su funcionamiento dentro de cada una de las máquinas de Grid. Estas credenciales se han de establecer en todas y cada una de las máquinas.

En este módulo se trabaja con el certificado X.509 que es el formato de datos estándar para los certificados GSI. Este certificado se crea con un certificado digital de usuario y la clave privada de usuario.

2.2.2. Módulo de ejecución

Es el módulo que gestiona el proceso por el que un nuevo clon del gusano pasa a ejecutarse en la siguiente máquina.

Este módulo requiere que, por una parte que las credenciales estén establecidas en la máquina en la que se va a ejecutar y por otra que la estructura de archivos que componen la aplicación resida ya de forma adecuada en la siguiente máquina.

Este módulo necesita utilizar la información de configuración que se contiene en los ficheros, por eso dentro de este módulo se puede englobar el de gestión de propiedades.

- **Módulo de gestión de propiedades**

En este módulo se engloba la gestión de los ficheros de propiedades que utiliza la aplicación para su funcionamiento.

Dentro de él se podrían diferenciar a grandes rasgos tres funcionalidades:

- *Acceso a fichero de propiedades*:Este módulo localiza y accede al fichero para su posterior lectura.

- *Lectura del fichero*:La lectura conlleva la obtención de las propiedades que se registran dentro del fichero.

- *Modificación del fichero*:En caso de ser preciso, este módulo gestiona cualquier modificación que deba quedar registrada en estos ficheros para que quede actualizada en posteriores ejecuciones del gusano.

- **Comprobación de la conectividad antes de la ejecución**.

En el lanzamiento de la aplicación a una máquina lo primero que se debe hacer es comprobar la conectividad de esta máquina ya que por distintas razones puede haberse desconectado del GRID y así evitar posibles situaciones anómalas.

Si no existen problemas de conectividad se procede al lanzamiento de la aplicación. En el caso de un fallo en la conexión se salta dicha máquina y se procede a actualizar el fichero de propiedades de las máquinas poniendo su entrada a "INCORRECTO" para determinar así que no se ha podido ejecutar la aplicación en dicha máquina y se salta a la siguiente máquina de la lista.

2.2.3. Módulo de notificaciones

A través de este módulo se gestionan de forma centralizada todas las notificaciones precisas en la ejecución de la aplicación.

Este módulo tiene especial significado para la primera máquina de la lista de orden de ejecución ya que realiza la función supervisora/controladora.

2.2.4. Módulo de interfaz de usuario

Es el módulo más cercano al usuario como su propio nombre indica.

Se encarga de la interacción de la aplicación con el usuario, tanto la aportación de usuario a la aplicación como la información que la aplicación ofrece al usuario se gestionan a través de él.

Sus principales funcionalidades son:

- Establecimiento/modificación del listado de máquinas sobre las que va a trabajar la aplicación.
- Inicio de la ejecución del gusano a través de la ruta de máquinas establecidas.
- Muestra la información que recoge la aplicación a través de su ciclo de vida a través de las máquinas. Esta información se centra en la evolución temporal de la ejecución del gusano.

2.2.5. Módulo de errores

Este es el módulo que se encarga del reconocimiento de errores de la aplicación.

El gusano a lo largo de su ciclo de vida es susceptible de encontrase con la aparición de ciertos errores. Los errores más típicos han sido identificados y registrados para que el propio gusano pueda reconocerlos. La finalidad de esto es la de gestionarlos si fuera posible y/o informar a la máquina supervisora en caso de no poder realizar ninguna gestión para solucionarlo o esquivarlo.

2.2.6. Módulo de transferencia

Es el módulo que se encarga de lo relacionado con la transferencia de ficheros.

Como ya se ha explicado, para que la aplicación se pueda ejecutar en una máquina remota es preciso que el sistema de archivos que la conforman haya sido copiado en esa máquina.

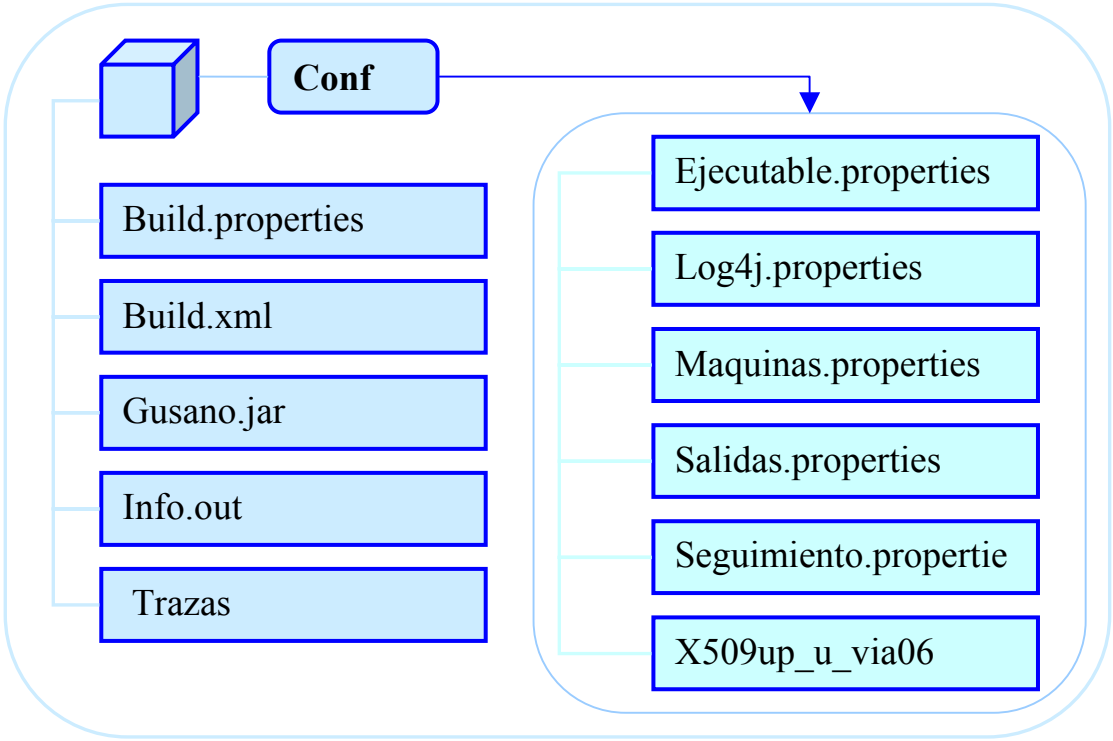
Por lo tanto este módulo se va a encargar de que de forma previa a la ejecución todos los ficheros necesarios (de configuración, código, etc.) se encuentren en la máquina/ruta pertinentes.

Es un módulo de vital importancia dentro de la aplicación ya que sin el la ejecución y progreso del gusano a través de las máquinas resultaría imposible.

2.3. ESTRUCTURAS DE DATOS

La gestión de la aplicación no se basa en una base de datos como ocurre típicamente en aplicaciones de otra índole. Esta aplicación se soporta, como se ha comentado en apartados previos, sobre una estructura de archivos claramente identificada.

A continuación se va a desglosar esta jerarquía y se irá viendo uno por uno cual es el papel que juega cada archivo en la aplicación.



Los archivos se agrupan en dos tipos diferenciados en base al uso que la aplicación hace de ellos:

- **Archivos de ejecución**:
  - **Gusano.jar**:
    Contiene el código de la aplicación necesario para su ejecución.
  - **Build.xml**:
    Es el script a través del cual se lanza el código java de la aplicación utilizando *ant*.

- **Archivos de configuración**:
  - **Archivos de propiedades**:
    - **Build.properties**:
      Establece las propiedades necesarias para que build.xml se ejecute correctamente, principalmente *locations* de librerías y código.
    - **X509**:
      Contiene las credenciales X509 que se crean en la primera de las máquinas y que hay que ir transfiriendo para que la aplicación pueda ejecutarse.
  - Carpeta **conf**:
    - **Ejecutable.properties**:
      Contiene datos necesarios para la ejecución de la aplicación.
      - Código RSL para la creación del directorio raíz en la que se encuentran ubicados los archivos.

- Código RSL para la creación del directorio *conf*.
- Código RSL a través del cual se va a ejecutar haciendo uso de *ant* el script de ejecución build.xml.

• **Log4j.properties**:
La función de este archivo es la de configurar las propiedades referentes a *log4j* para toda la gestión del fichero de trazas que nos ayudará a resolver situaciones anormales.

• **Maquinas.properties**:
Este archivo juega un papel importante dentro de la configuración de la aplicación. Previamente se había hecho referencia a este fichero que contiene la lista de máquinas. De forma ordenada indica inequívocamente qué máquinas y en qué secuencia se han de visitar, o intentarlo al menos, durante la ejecución de la aplicación.

• **Salidas.properties**:
Registra las máquinas en las que el gusano ha sido capaz de transferirse y ejecutarse. En caso de que la aplicación no se encuentre ningún problema y pueda pasar por todas las máquinas este será una imagen similar a maquinas.properties.
Este archivo también podría incluir otro tipo de información general si se quisiera, como por ejemplo los usuarios de una máquina.

• **Seguimiento.properties**:
Conforma la principal salida que ofrece la aplicación. Este fichero recoge toda la información que la aplicación ha ido recabando a lo largo de su ejecución.
Cuando el gusano reside en una máquina es este el fichero en el que registra los datos obtenidos.

o **Archivos de información:**

• **Info.out**:
Guarda de forma acumulativa las salidas por consola resultantes de la ejecución de la aplicación en cada una de las máquinas del recorrido.

• **Trazas**:
Contiene la descripción de los pasos que va siguiendo la aplicación traza por traza. En caso de fallo de la aplicación se puede seguir fácilmente la pista de los pasos que ha realizado el gusano y así analizar el error posible. Es un archivo de información que en caso de no producirse error alguno resulta totalmente prescindible para el resto del funcionamiento.

• **Errores.xml**:
En caso de error en la ejecución éste queda registrado en el archivo. En él el error queda completamente identificado de forma que el módulo de errores puede proceder a tratarlo solucionándolo si es reparable o simplemente informando al usuario de la situación de error.

Más a delante se adjuntará una copia del contenido de cada uno de estos ficheros con la que se podrá formar una idea más completa de cómo son y se estructuran.

Destacar también que principalmente se han utilizado ficheros de tipo *properties* ya que son fácilmente manejables desde código java, tanto para acceder a ellos como para modificarlos.

2.3.1. Fichero de seguimiento

Por su importancia y su complejidad de gestión a continuación se pasan a desglosar los principales puntos del funcionamiento de este fichero.

Cuando se requiere el fichero de seguimiento para la muestra de resultados, se procede a conseguirlo de:

1. Si es la maquina de partida lo coge de su directorio local.
2. Si no estamos ante el caso anterior:
    a. Mira la conectividad que existe de la maquina en la que se ha ejecutado el gusano justo antes que la presente. Si tiene conectividad (es decir Gram.ping) entonces procede a la copia desde esta ubicación.
    b. Si no posee conectividad va siguiendo la secuencia de maquinas que ha recorrido el gusano a la inversa empezando desde su posición hasta que encuentre alguna máquina con conexión al GRID y efectúa la copia del fichero de seguimiento desde esta ubicación.

Cuando se termina la ejecución en una máquina, se procede a la copia del fichero de seguimiento en la máquina origen para poder realizar el estudio de la ejecución de la aplicación.

- **Tiempos de ejecución y copia**

Para el cálculo de tiempos y poder efectuar un estudio de la ejecución, se van insertando mensajes con los tiempos de comienzo y finalización de los dos grandes bloques de ejecución: COPIADO y EJECUCIÓN.

3. ESPECIFICACIÓN DE REQUISITOS SOFTWARE FINAL

Antes del desarrollo e implementación de la aplicación se redactó una Especificación de Requisitos Software inicial. Esta SRS se ha visto ampliada durante el desarrollo del proyecto, posibilitada principalmente por la ampliación de conocimientos técnicos del grupo de desarrollo y por las necesidades de la propia aplicación que inicialmente no se reflejaron.

3.1. AMPLIACIÓN DE LA ESPECIFICACIÓN DE REQUISITOS SOFTWARE INICIAL

3.1.1. Autonomía de Worm Application

En la primera especificación se contempló la ejecución de la aplicación en cada una de las máquinas como un hecho autónomo. Después del desarrollo de Worm esta autonomía ha resultado no ser total.

La aplicación, en cada una de sus ejecuciones puntuales en los distintos equipos, mantiene siempre un hilo de información y/o comunicación con la máquina supervisora. Ésta máquina, que juega el papel principal del ciclo de vida de la aplicación, se estableció como nuevo requisito cuando se detectó su necesidad durante el desarrollo.

No obstante, el grado de autonomía dentro de cada máquina es casi total ya que la supervisora sólo participa como foco de monitorización de trazas y, en caso de producirse, de gestión de error.

3.1.2. Control de errores

Al comienzo no se especificó ningún requisito sobre el posible control de errores, esto se debió al desconocimiento de las excepciones que podría sufrir la aplicación. Estas situaciones de errores no quedaron definidas hasta el momento de realizar las pruebas del primer prototipo de la aplicación.

Junto con la identificación de errores se estableció la necesidad de un módulo que realizara la gestión de las excepciones, el cual está directamente relacionado con la máquina supervisora, que se encarga de este trabajo.

3.1.3. Interfaz Gráfica

Inicialmente la observación de los efectos de la ejecución, así como su manejo, resultaban demasiado complicados de ver. Para facilitar toda la monitorización de trazas de ejecución de Worm se diseñó la interfaz gráfica que ofrece un entorno amigable para el usuario.

La interfaz de usuario se aloja, de forma ininterrumpida durante todo el ciclo de vida de la aplicación Worm, en la máquina supervisora.

Principalmente, su funcionalidad radica en la muestra de información y en el control de la aplicación por parte del usuario.

NOTA: Más adelante, en este documento de contenido del proyecto, se desglosa con mucho más detalle el funcionamiento y desarrollo que involucran estas ampliaciones de requisitas realizadas a posteriori.

4. GARANTÍA DE CALIDAD SOFTWARE

4.1. LÍNEAS BASE

En este apartado se van a indicar las líneas base que se han establecido como directrices de la aplicación.

- **Ejecución**

Esta es quizá la más importante de las líneas que dirigen la aplicación. Engloba todo aquello que se ve involucrado en la ejecución de un job en otra máquina del Grid. En este caso dicho job será el código fuente de la propia aplicación de forma que el gusano se relanza en su siguiente punto del recorrido.

Esta línea se puede identificar con el módulo de ejecución de la aplicación. Tanto en el apartado de estructura como en el de diseño del proyecto se puede ver como se compone y estructura este módulo así como el modo en el que interacciona con otros.

- **Transferencia**

La transferencia de archivos resulta vital para el funcionamiento de la aplicación, sin ella resultaría imposible llevar a cabo la ejecución. Como ya se ha comentado, la aplicación se soporta sobre una estructura de archivos con una jerarquía establecida.

Este sistema de ficheros necesitará ser clonado en la siguiente máquina del recorrido antes de que se indique el comienzo de la ejecución en la máquina siguiente en el recorrido de la ejecución.

El módulo de transferencia debe garantizar que el traspaso de ficheros es completo y correcto y que respeta también la estructura jerárquica establecida para los ficheros. Si todo esto se cumple la siguiente ejecución será posible.

- **Credenciales**

Para realizar cualquier operación dentro de Grid debe poseerse la autorización necesaria y aquí es donde entra en juego el módulo de credenciales. El uso de este módulo es totalmente imprescindible.

Las credenciales se establecen en todas y cada una de las máquinas en las que residirá la aplicación y gracias a ellas esta estará autorizada para realizar las operaciones pertinentes.

Cada uno de los módulos de la aplicación son útiles e importantes, pero estos tres que se han presentado como líneas base son quizá los más destacados e imprescindibles. Si alguna de las líneas base no funciona correctamente el resto de la aplicación resulta inútil ya que son de vital necesidad.

4.2. OCIs (Informes de Cambios)

Todos los cambios realizados sobre la primera especificación y diseño de la aplicación han sido supervisados y aprobados por todos los miembros del grupo.

A continuación se registran los informes de cambios de las principales funcionalidades incluidas en la aplicación después de su visión inicial.

- **Log4j**

  - *Motivación*: A medida que el código se iba haciendo más grande su implementación iba resultando más compleja su depuración. Por esta razón se planteó como necesario alguna forma de registrar los pasos que iba dando la ejecución de la aplicación en relación con el código de forma que en caso de fallo se pudiesen seguir los pasos de ejecución y detectar de forma rápida los puntos de error. Además, el hecho de consistir en una ejecución distribuida acentuaba más esta necesidad puesto que no había manera de depurar los errores ocurridos en las máquinas remotas.

  - *Consecuencia*: Se planteó como solución más eficaz el uso de *log4j* que forma parte del API de Java. La principal ventaja de utilizarlo es la capacidad de activar ciertos *log* que van definiendo las trazas de la ejecución de la aplicación.De esta forma entra a formar parte de la jerarquía de ficheros de la aplicación el fichero *Trazas* en el que log4j va añadiendo cada uno de los pasos y/o errores que refleja la aplicación. El log4j me permiten distinguir entre una serie de tipos distintos de trazas como son INFO, ERROR, WARN… Esta distinción entre los tipos de traza permite distinguir las situaciones de correcta ejecución de las que tienen errores o warnings.

- **InterfazGUI**

  - *Motivación*: Con el prototipo inicial resultaba imposible apreciar de forma inmediata la evolución del gusano a través de su recorrido. Así es como aparece la idea de que debe existir algún interfaz al que el usuario tenga fácil acceso y a través del cual pueda ver algún resultado procedente de la ejecución.

  - *Consecuencia*:Se diseña e implementa un interfaz de usuario que se integra con el código de la aplicación. Este interfaz se localiza en la primera máquina del recorrido del gusano que es de la que parte y a la que regresa después de su ciclo de vida a través del resto de máquinas. El hecho de que la primera máquina también sea la última del recorrido, y que a través de ella el usuario acceda a la aplicación usando la interfaz, hace que esta se puede considerar como máquina supervisora de la aplicación.

- **Control de excepciones**

  - *Motivación*:Cuando la aplicación produce algún error y se lanza una excepción del sistema resultaba muy complicado registrarla de forma clara a fin de poder esquivarla y/o resolverla, si fuera posible.

  - *Consecuencia*:Se establece un nuevo módulo que gestiona las excepciones que puede producir la aplicación. Se tratan como errores que serán notificados y/o tratados en caso de ser posible. Este cambio aporta seguridad y fiabilidad a la ejecución de la aplicación.

- **Parser e hilo de notificaciones**

  - *Motivación:*Todos los errores y/o excepciones que puede provocar la aplicación quedan inicialmente gestionados por el módulo de errores que se incluye tras el cambio de diseños que conlleva el control de excepciones. Esta gestión lleva a ver la necesidad de que se realice de una forma centralizada que dirija todo lo relacionado con este tema.

  - *Consecuencia*:Se especifica, diseña e implementa el módulo de notificaciones que gestiona, como su propio nombre indica, la notificación de cualquier error para que la parte de la aplicación residente en la primera máquina centralice y controle todo lo relacionado con esa información. El control, que se realiza a través de un hilo, informará al usuario de la excepción ocurrida y, si es capaz, la solventará en tiempo real.

5. SEGUIMIENTO DE RIESGOS

Este apartado se centra en la clasificación de los riesgos de cinco tipos: tecnológicos, personales, de organización, de requisitos y de especificación.

En cada una de las categorías se han identificado inicialmente los riesgos que se podían producir y después se ha registrado el seguimiento de dichos riesgos para detallar si se han dado o no a lo largo del desarrollo del proyecto.

5.1. RIESGOS TECNOLÓGICOS

5.1.1. IDENTIFICACIÓN Y ANÁLISIS

A. Falta de disponibilidad del punto de trabajo en el seminario de Laboratorio Grid de la facultad.

B. Dificultad y lentitud en el acceso remoto a los equipos del Laboratorio Grid, ya sea desde los equipos de los laboratorios de la facultad o desde el equipo personal de los miembros del equipo, en estos últimos el problema es mucho mayor.

5.1.2. SEGUIMIENTO

A. Este es un riesgo con el que el grupo se ha encontrado a lo largo de la realización del proyecto. Los puestos de trabajo en el seminario eran normalmente uno o dos en el mejor de los casos de forma que en el momento en el que los miembros del grupo deseaban trabajar por separado se han encontrado con escasos recursos.
   Incluso en alguna ocasión ha resultado imposible acceder a estos puestos por estar completo el uso del mismo o por encontrarse cerrado y no tener posibilidad alguna de conseguir la llave.
   Esta barrera se ha solventado siempre trabajando de forma remota accediendo desde máquinas externas a las de Laboratorio Grid. Unas veces se ha trabajado desde la propia facultad y otras desde los propios domicilios de los miembros del grupo.

B. El cumplimiento del riesgo identificado en punto A ha agravado este segundo riesgo ya que aunque inicialmente era estimable tener que trabajar en algún momento con conexión remota ha hecho que esta forma de trabajo haya supuesto una tónica general y continuada.
Al ser la forma habitual de trabajo en el proyecto la conexión remota la dificultad de acceso y lentitud de conexión se han hecho más patentes y molestas para la evolución del trabajo.
La conexión remota ha sido un problema mayor para el desarrollo del interfaz gráfico debido a la necesidad de importar el display: el consumo de recursos de memoria y de red aumentaban considerablemente y empobrecía el rendimiento, hasta el punto de no ser viable esta conexión remota (desde fuera de la facultad).

5.2. RIESGOS PERSONALES

5.2.1. IDENTIFICACIÓN Y ANÁLISIS

A. Falta de base de formación en las tecnologías utilizadas (Grid Computing).

B. Posibles discrepancias entre los miembros del grupo de proyecto.

C. Posibles ausencias de alguno/s miembros del grupo por causas personales, por ejemplo enfermedad.

D. Abandono del proyecto de algún miembro del grupo por cualquier causa que se pueda dar.

E. Incompatibilidad de horarios de los distintos miembros del grupo para realizar trabajo común y/o reunirse para seguimiento común de cualquier tarea individual.

5.2.2. SEGUIMIENTO

A. Era un riesgo previsto y asumido por todos los miembros del grupo. Desde el primer momento, desde la elección y aceptación del proyecto a realizar, los miembros del grupo eran conscientes de que la tecnología Grid Computing era desconocida para ellos.
La forma en la que se ha afrontado este riesgo ha sido la formación desde el principio y prácticamente continuada. Cada una de las tareas de diseño y desarrollo han hecho necesaria la formación tecnológica de lo miembros del grupo. Todo esto ha ralentizado el avance del trabajo productivo pero dentro de límites razonables y asumibles sin haber supuesto mayores trastornos para la evolución del trabajo que los que ya se esperaban inicialmente por el total desconocimiento del tema.

B. Afortunadamente no se han producido discrepancias dentro del grupo ya que la tónica de trabajo ha sido de forma habitual compartida y aceptada por todos los miembros.
Destacar que esto ha sido posible debido a que el número de miembros es solamente tres y que su coordinación y comunicación ha sido totalmente favorable.

C. Este riesgo no ha supuesto ningún problema ya que si alguno de los miembros se ha ausentado en algún momento ha sido exclusivamente por enfermedad y nunca por periodo superior a dos días. Este periodo máximo de ausencia ha resultado totalmente aceptable y en ningún caso ha supuesto ningún retraso en la evolución del proyecto.

D. No se ha producido el abandono de ninguno de los miembros del grupo por lo que este riesgo no se ha visto contemplado en la realidad.

E. A pesar de que la compatibilidad de los horarios de los miembros del grupo no ha sido total, algo que resulta prácticamente imposible aún en el mejor de los casos, ha resultado ser muy cómoda. Los horarios de disponibilidad para trabajar en común

sobre el proyecto se han organizado sin problemas gracias la actitud positiva al respecto de todos los miembros.

5.3. RIESGOS DE ORGANIZACIÓN

5.3.1. IDENTIFICACIÓN Y ANÁLISIS

A. Mala coordinación a la hora de integrar trabajos independientes de miembros del grupo.

B. Desconocimiento y/o desinterés por el que algún miembro del grupo no quede informado de fechas de finalización de tareas y/o reuniones del grupo y/o con el profesor responsable de la supervisión.

C. Pérdida de documentos o archivos.

D. Descontrol en el uso de versiones por parte de los distintos miembros.

5.3.2. SEGUIMIENTO

A. Este riesgo no ha supuesto ningún problema mencionable ya que al ser reducido el número de miembros del grupo de trabajo la buena coordinación entre los mismos ha resultado cómoda a la hora de integrar partes diferenciadas del proyecto.

B. Todos los miembros han mantenido la comunicación constante con los demás y en todo momento ha sido de conocimiento común la evolución del trabajo de cada uno de ellos para los demás.
Esto ha hecho que la organización y comunicación entre los distintos miembros haya sido fluida y cómoda. Ha influido muy favorablemente la creación de un grupo a través de la Web para que cualquiera pudiera acceder a las últimas novedades y/o información que otros miembros pudieran aportar sobre el proyecto. De la misma forma la comunicación vía mail y directa ha sido constante y fácil de establecer en cualquier caso.

5.4. RIESGOS DE REQUISITOS

5.4.1. IDENTIFICACIÓN Y ANÁLISIS

A. Inestabilidad y/o identificación incorrecta de los requisitos básicos necesarios del proyecto.

B. Requisitos poco realistas que resultan imposibles de satisfacer.

C. Ampliación de requisitos vitales cuando el desarrollo del proyecto esté ya bastante avanzado.

5.4.2. SEGUIMIENTO

A. Cuando se realizó la especificación inicial de requisitos el grupo fue consciente de que el desconocimiento de la tecnología podía limitar la implementación del proyecto. Por este motivo la identificación de requisitos del prototipo se ajustó sólo a las necesidades básicas que establecían posibles de realizar.
Este planteamiento favoreció a que las especificaciones principales se ajustaran correctamente a las bases de funcionamiento de la aplicación de forma que ampliaciones y adhesiones posteriores se fueron acoplando sin mayor problema.

B. Como ya se ha hecho patente en este documento, con la necesidad de formación inicial el grupo se mostró bastante escéptico al proponer los requisitos básicos. Debido a esto y a la evolución razonablemente satisfactoria de la formación en conocimientos técnicos los requisitos básicos han sido cumplidos de acuerdo a lo establecido. Además han podido ser ampliados a la par que la formación técnica de los miembros ha ido evolucionando y posibilitando su implementación.

C. Este riesgo no se ha cumplido debido a que los requisitos básicos se establecieron claramente desde el principio y que en base a ellos se han ido introduciendo nuevas funcionalidades de forma incremental y controlada.

5.5. RIESGOS DE ESTIMACIÓN

5.5.1. IDENTIFICACIÓN Y ANÁLISIS

A. Estimaciones temporales poco realistas e imposibles de cumplir en el plazo estipulado.

B. Adelanto de la fecha límite para la realización de proyecto de forma imprevista.

5.5.2. SEGUIMIENTO

A. Al igual que en el tratamiento de riesgos de requisitos los riesgo de estimación se han tratado con la misma cautela y realismo. Las fechas auto impuestas por los miembros del grupo para la finalización de tareas parciales del proyecto han sido cumplidas en la gran mayoría de los casos. El único caso que cabe destacar y que supuso pequeños trastornos para la evolución del trabajo ha sido una vez más la necesidad de formación tecnológica de los miembros del grupo. La formación que al principio se estimó inicial a pesar de haber sido al comienzo cuando más importancia ha tenido ha resultado ser prácticamente continuada a lo largo del proyecto.

B. La fecha límite para la entrega del proyecto ha sido la estimada al comienzo, lo que ha supuesto que el resto de la planificación planteada haya sido válida de acuerdo a las fechas establecidas.

6. SEGUIMIENTO DE PLANIFICACIÓN TEMPORAL

6.1. TABLA Y GRÁFICA TEMPORAL DE LA PLANIFICACIÓN DE TAREAS

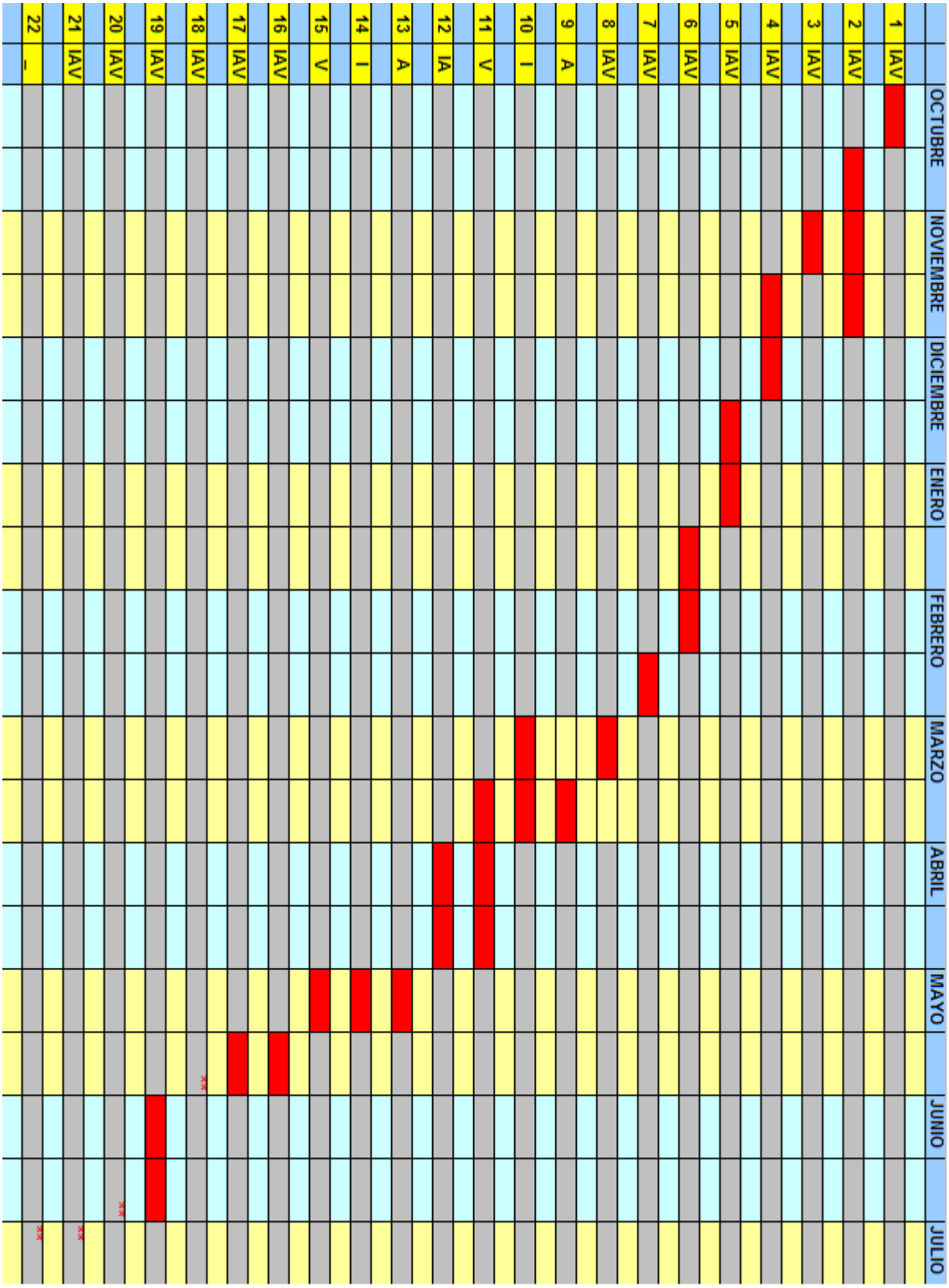
Duración estimada del proyecto:

Inicio	01-10-2005
Finalización	30-06-2006

Periodo de tiempo		Tarea estipulada	ID tarea
Inicio	Final		
01-10-2005	15-10-2005	Primera toma de contacto con los conocimientos técnicos de Grid y Globus Toolkit.	**1**
15-10-2005	01-12-2005	Continuación de la investigación y formación en el entorno de Globus Toolkit	**2**
01-11-2005	15-11-2005	Especificación y diseño inicial de la aplicación	**3**
15-11-2005	15-12-2008	Prácticas y ensayos de prototipos	**4**
15-12-2005	15-01-2006	Prototipo inicial	**5**
15-01-2006	15-02-2006	Ampliación y cierre del prototipo básico de la aplicación	**6**
15-02-2006	01-03-2006	Pruebas y mejoras de la aplicación	**7**
01-03-2006	15-03-2006	Inicio de la segunda ampliación de funcionalidades	**8**
15-03-2006	01-04-2006	Ampliación de funcionalidades	**9**

01-03-2006	01-04-2006	Diseño e implementación de interfaz gráfica	**10**
15-03-2006	01-05-2006	Diseño y desarrollo de la primera versión de documentación del proyecto	**11**
01-04-2006	01-05-2006	Implementación de viabilidad de integración entre aplicación e interfaz gráfica	**12**
01-05-2006	15-05-2006	Verificación y prueba de las funcionalidades de la aplicación	**13**
01-05-2006	15-05-2006	Verificación y prueba de la interfaz gráfica	**14**
01-05-2006	15-05-2006	Desarrollo continuo de la documentación del proyecto	**15**
15-05-2006	30-05-2006	Integración final de la aplicación con la interfaz gráfica	**16**
15-05-2006	30-05-2006	Revisión a nivel de código y ampliación continua de la documentación.	**17**
31-05-2006		Presentación de demostración con el profesor que supervisa el proyecto	**18**
01-06-2006	30-06-2006	Verificación, prueba y ultimación de detalles del código y documentación de la aplicación. Posibles cambios de urgencia imprevistos.	**19**
30-06-2006		Fecha límite para la realización de cualquier cambio imprevisto.	**20**
05-07-2006		Fecha de presentación del proyecto	**21**
07-07-2006		Fecha límite para la entrega de la documentación del proyecto.	**22**

En la figura que se muestra a continuación se puede ver de forma más gráfica como se distribuyen en el tiempo las distintas tareas. (Junto al número de tarea aparecen las iniciales de los responsables de cada una de ellas).


OCTUBRE
NOVIEMBRE
DICIEMBRE
ENERO
FEBRERO
MARZO
ABRIL
MAYO
JUNIO
JULIO
1 IAV
2 IAV
3 IAV
4 IAV
5 IAV
6 IAV
7 IAV
8 IAV
9 A
10 I
11 V
12 IA
13 A
14 I
15 V
16 IAV
17 IAV
18 IAV
19 IAV
20 IAV
21 IAV
22 –

6.1.1. TAREA 1: Toma de contacto con los conocimientos técnicos

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** El proyecto se desarrolla sobre la tecnología Grid Computing y en concreto apoyándose sobre Globus Toolkit 4, estos conocimientos era desconocidos para los miembros del grupo de forma que fue precisa una primera toma de contacto que diera un idea general sobre las tecnologías sobre las que se iba a trabajar.

6.1.2. TAREA 2: Investigación y formación

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** Una vez superada la primera toma de contacto fue preciso profundizar en estos conocimientos técnicos. Principalmente se dedicó esta tarea al manejo y conocimiento del Globus Toolkit 4 y todas las posibilidades que ofrecía.

6.1.3. TAREA 3: Especificación y diseño inicial de la aplicación

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** Conociendo ya el manejo básico de estas tecnologías se estaba ya en disposición de realizar la especificación y diseño básicos de la aplicación. La primera especificación supone la base de la aplicación y el diseño dibuja lo que serán después sus funcionalidades.

6.1.4. TAREA 4: Prácticas y ensayos de prototipos

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** Aproximadamente durante un mes se realizaron bastantes prácticas simples para introducir al grupo en el manejo de Globus lo que llevó a los primeros intentos de realizar un prototipo inicial de la aplicación.

6.1.5. TAREA 5: Prototipo inicial

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** En este punto se conocían ya muchos de los pormenores de Globus como resultado de la continua formación y práctica del grupo en este tema. Se realizó el primer prototipo de la aplicación que realizaba de forma básica la función del gusano que consistía en crearse, clonarse e irse ejecutando de una máquina a otra de forma sencilla pero aún sin auto destruirse ni recopilar información.

6.1.6. TAREA 6: Ampliación y cierre del prototipo básico de la aplicación

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** El código que en el primer prototipo inicial resultaba bastante básico y poco estructurado por lo que se fue reformando para que el código estuviera bien modulado y fuera más simple sus posteriores ampliaciones.

6.1.7. TAREA 7: Pruebas y mejoras de la aplicación

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** Una vez reestructurado el primer prototipo se realizaron pruebas de este para que una vez verificado su correcto funcionamiento se pudiera pasar a la siguiente ampliación o iteración. La principal mejora que se introdujo fue registrar como el gusano iba pasando por las distintas máquinas. A través de un fichero de propiedades el gusano iba registrando de forma incremental el nombre de la máquina en la que se alojaba. De esta forma al final del recorrido se podía verificar que efectivamente el gusano había visitado todas las máquina que se suponían debía visitar.

6.1.8. TAREA 8: Inicio de la segunda ampliación de funcionalidades

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

- **Contenido:** La segunda ampliación que comenzó con el mes de marzo supuso un gran reto ya que eran muchos los nuevos detalles a incluir en la aplicación. Se introdujo en el código el uso de log4j que permitía registrar datos sobre la ejecución del código y que era de gran utilidad en caso de fallo en la ejecución de la aplicación y también para ampliaciones del código.También destacar como principal incorporación la recopilación de información del seguimiento del gusano a través de su recorrido por las máquinas. Se comienza a plantear el manejo de las credenciales a través de la aplicación y se empieza a estudiar y probar como implementar un módulo para esta nueva funcionalidad. En esta etapa del proyecto se formaliza la idea de incluir una interfaz gráfica que funcione en al primera máquina del recorrido que como ya se ha nombrado juega un papel de supervisor y/o controlador de la ejecución del gusano durante su ciclo de vida. Sobre esta idea se hace una especificación y diseño de cómo debe ser y funcionar esta interfaz.

6.1.9. TAREA 9: Ampliación de funcionalidades

- **Responsable:** Alberto González

- **Contenido:** Se implementa el manejo de las credenciales desde la aplicación para que pueda autorizarse a través de su recorrido. Esta implementación involucra la inclusión de un nuevo módulo en el proyecto.

6.1.10. TAREA 10: Diseño e implementación de la interfaz gráfica

- **Responsable:** Isabel Fernández

- **Contenido:** Se procede en esta etapa a finalizar el diseño de la interfaz de forma paralelamente al inicio de su implementación, ya que ciertos inconvenientes en esta implementación hacen variar de forma leve el diseño. Una vez claro el diseño de esta interfaz se implementa de forma básica a la espera de su integración con el resto del código.

6.1.11. TAREA 11: Diseño y desarrollo de la primera versión de la documentación

- **Responsable:** Verónica Gonzalo

- **Contenido:** Con las principales funcionalidades de la aplicación ya implementadas y verificadas, y con los nuevos objetivos diseñados y en proceso ya de implementación

es el momento de formalizar y dar formato a la primera versión algo esquemática pero ya formal de la documentación del proyecto.En esta etapa se constituye ya una base sólida sobre la que ir recabando la documentación del proyecto que en este punto ya incluye la información básica del mismo.

6.1.12. TAREA 12: Implementación y viabilidad de integración entre aplicación e interfaz gráfica

- **Responsable:** Isabel Fernández, Alberto González

- **Contenido:** Una vez se tiene la versión inicial de la interfaz de usuario y el código de la aplicación se encuentra ya en un estado avanzado de implementación se estudian las posibilidades de cómo integrar este nuevo módulo de interfaz con el resto de la aplicación.

6.1.13. TAREA 13: Verificación y prueba de las funcionalidades de la aplicación

- **Responsable:** Alberto González

- **Contenido:** Se realizan pruebas de los nuevos módulos que se han incluido en la aplicación. Las pruebas involucran principalmente el correcto funcionamiento e interacción de los módulos de credenciales, notificaciones y errores.

6.1.14. TAREA 14: Verificación y prueba de la interfaz gráfica

- **Responsable:** Isabel Fernández

- **Contenido:** Se realizan también pruebas sobre la interfaz gráfica que requieren finalmente retocar pequeños de talles de la misma. Con esta tarea la interfaz queda implementada y a la espera de ser totalmente integrada en la aplicación.

6.1.15. TAREA 15: Desarrollo continuo de la documentación del proyecto

- **Responsable:** Verónica Gonzalo

- **Contenido:** Todas las implementaciones y pruebas se supervisan a la par que se reflejan en la documentación del proyecto. Esto lleva al igual que en el código a incluir documentación de todos los nuevos módulos.

6.1.16. TAREA 16: Integración final de la aplicación con la interfaz gráfica

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo
- **Contenido:** Se pasa a integrar la interfaz con el código realizando las adaptaciones necesarias que son mínimas debido a que previamente se había estudiado la viabilidad de esta integración. Las modificaciones resultantes son fruto de pequeños imprevistos.

6.1.17. TAREA 17: Revisión a nivel de código y ampliación contínua de la documentación

- **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

▪ **Contenido:** De forma previa a la demostración ante el profesor se realiza una revisión completa de toda la aplicación, incluyendo código y documentación, de forma que todos los miembros del grupo tenga la oportunidad de participar en todos los cierres de implementación y documentación de los módulos desarrollados.

6.1.18. TAREA 18: Presentación de demostración con el profesor supervisor del proyecto

▪ **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

▪ **Contenido:** Antes de la finalización del proyecto y con una versión prácticamente final se proyecta una demostración para asegurar el visto bueno del profesor que supervisa el proyecto. Esta presentación establece así una fecha obligada para que todos los temas de implementación abiertos hasta ahora queden totalmente cerrados.

6.1.19. TAREA 19: Verificación, prueba y ultimación de detalles de código y documentación

▪ **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

▪ **Contenido:** Este periodo que coincide con la época de exámenes se plantea como el menos productivo de toda la planificación. Este es el principal motivo por el cual el 99'9% del trabajo del proyecto se debe finalizar para la fecha de la demostración ante el profesor.Este periodo queda establecido para la resolución de imprevistos y detalles de última hora, así como para posibles retoques en la documentación.

6.1.20. TAREA 20: Fecha límite para la realización de cualquier cambio imprevisto

▪ **Contenido:** Llegada esta fecha el grupo establece como base clara no prolongar el desarrollo del proyecto.

6.1.21. TAREA 21: Fecha de presentación del proyecto

▪ **Responsable:** Isabel Fernández, Alberto González, Verónica Gonzalo

▪ **Contenido:** La fecha de presentación queda fijada para el día 05-07-2006 a las 16:30 horas. Esta presentación se realizará en el salón de actos de la facultad en acto público.

6.1.22. TAREA 22: Fecha límite para la entrega de la documentación del proyecto

▪ **Contenido:** El 07-07-2006 es la fecha que la secretaría ha publicado como límite para presentar la documentación del proyecto de la asignatura de Sistemas Informáticos.

7. DISEÑO DEL PROYECTO SOFTWARE

7.1. ESTRUCTURA DE CLASES

7.1.1. Módulo de credenciales

- Credentials
  - Credencial.java

7.1.2. Módulo de ejecución

- Listener
  - InteractiveJobListener.java
  - JobListener.java
  - TransferListener.java

- Worm
  - ExecutionWorm.java

7.1.3. Módulo de gestión de propiedades

- FileProperties.java
- ManagementProperties.java

7.1.4. Módulo de interfaz de usuario

- WormGUI
  - WormGUI.java

- Util
  - InitialConfiguration.java
  - MachineDataGUI.java
  - FormatDataTime.java

- View
  - GraphsPanel.java
  - ColumMachine.java
  - TimeRule.java
  - TableMacGridView.java
  - TableMacWormView.java

- Themes
  - BlueTheme.java
  - GrayTheme.java
  - DefaultTheme.java
  - WhiteSatinTheme.java

- Módulo de notificaciones

- NotificationsGUI
  - NotificationsGUI.java
  - NotificationsThread.java

7.1.5. Módulo de errores

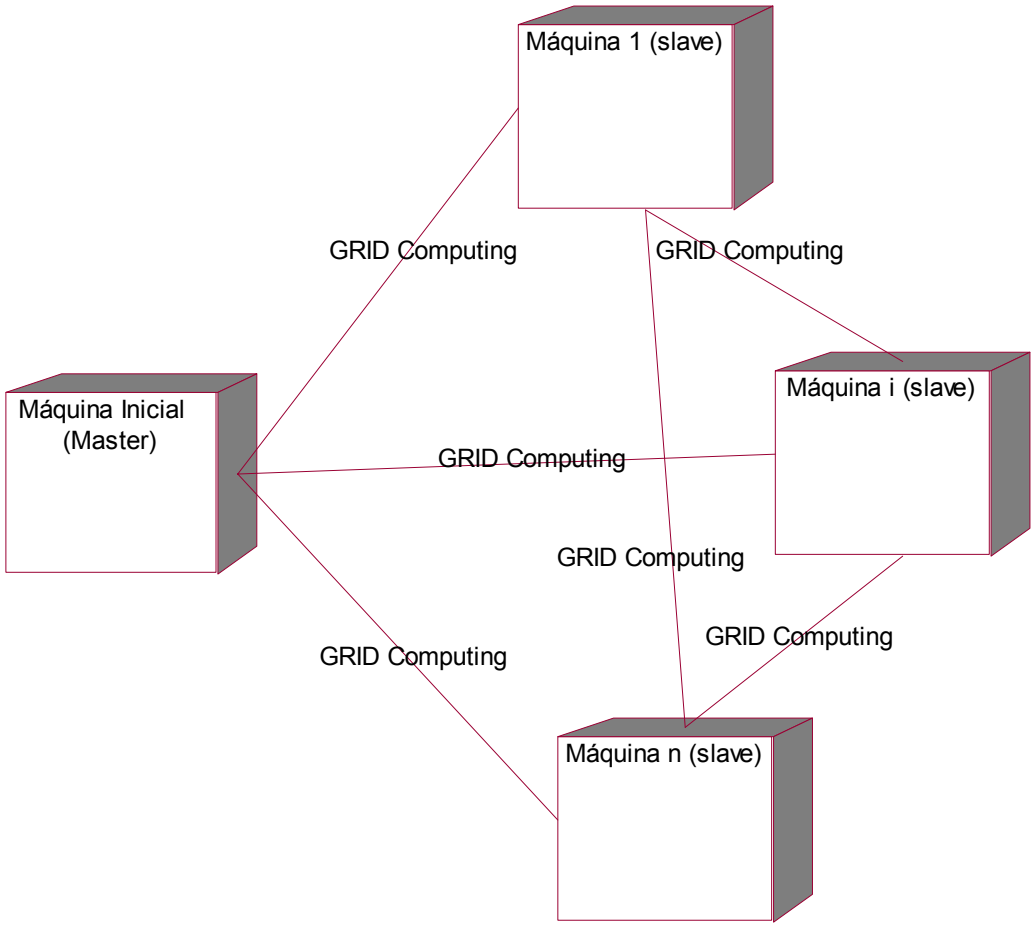
- ControlErrors
  - DataError.java
  - XMLExceptions.java
  - ParserError.java

7.1.6. Módulo de transferencia

- Transfers
  - FileTransfer.java

7.2. DIAGRAMAS UML

7.2.1. DIAGRAMA DE DESPLIEGUE



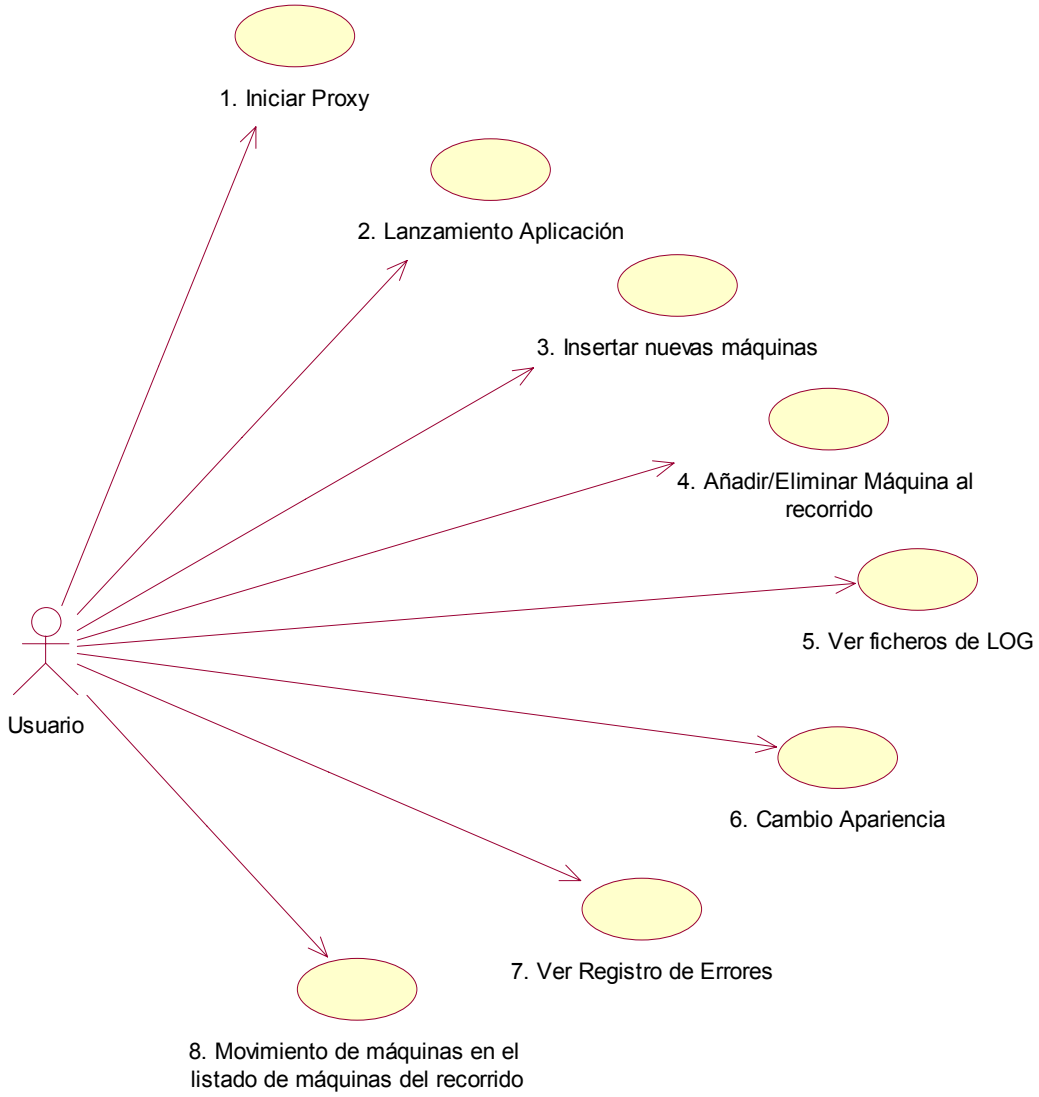
El diagrama de despliegue muestra la configuración de los nodos de procesamiento que participan en la aplicación y los procesos software que residen en ellos.

La distribución de los componentes es: Las distintas máquinas que se muestran en la parte superior tienen la siguiente distribución:

La máquina Inicial (Master) en general es la encargada del control de la aplicación sin embargo en algunos casos las decisiones son tomadas por los nodos intermedios (slave).

La conexión entre los nodos es total y se realiza a través de tecnología GRID.

7.2.2. DIAGRAMA DE CASOS DE USO DE WORM



A continuación expondremos un breve comentario sobre cada uno de los casos de uso y posteriormente describiremos el flujo de eventos desencadenados en la ejecución de cada uno de ellos. Así mostraremos el flujo normal de ejecución y las posibles alternativas, si existen, de ejecución que se puedan dar.

- **Actores del Sistema**

  Usuario: Sólo existe un único tipo de usuario.

- **Casos de uso del usuario**

  - *Iniciar el Proxy*

  ➢ Descripción: Con esta acción se podrá crear el "grid-proxy-init" necesario para la obtención de permisos de ejecución en el GRID.

- Secuencia Estimulo/ Respuesta: Una vez ejecutado esta opción se consigue la habilitación del resto de operaciones para ejecutar la aplicación ya que de esta manera se obtienen los permisos necesarios para trabajar en el GRID.
- Requerimientos funcionales: Se requieren autorización en el GRID a través del certificado digital de usuario y de la clave privada de usuario necesarios para realizar la operación y tener conectividad al GRID.

Flujo Normal

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable.	
	2. El sistema solicita los ficheros que contienen el certificado digital de usuario (userCert.pem), la clave privada de usuario (userkey.pem) y rellenar una serie de parámetros de configuración así como la contraseña para la creación del "grid-proxy-init".
3. El usuario indica estos datos y pulsa "Grid-proxy-Init".	
	4. El sistema comprueba que se han rellenado los campos y que los ficheros y la contraseña son correctos.
	5. Procede a realizar el "grid-proxy-init".

Alternativa 1

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable.	
	2. El sistema solicita los ficheros que contienen el certificado digital de usuario (userCert.pem), la clave privada de usuario (userkey.pem) y rellenar una serie de parámetros de configuración así como la contraseña para la creación del "grid-proxy-init".
3. El usuario indica estos datos y pulsa "Cancelar".	
	4. Se cancela la operación y no se producen modificaciones.

Alternativa 2

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable.	
	2. El sistema solicita los ficheros que contienen el certificado digital de usuario (userCert.pem), la clave privada de usuario (userkey.pem) y rellenar una serie de parámetros de configuración así como la contraseña para la creación del "grid-proxy-init".

	proxy-init".
3. El usuario indica estos datos y pulsa "Grid-proxy-Init".	
	4. El sistema comprueba que se han rellenado los campos y que los ficheros y la contraseña son correctos.
	5. Se devuelve error en caso de que los datos no se hayan completado correctamente o si los ficheros o contraseña no son válidos.

- *Lanzamiento de la aplicación*

- Descripción: Consiste en la puesta en marcha del núcleo de ejecución de la aplicación que irá recorriendo las distintas máquinas seleccionadas en la lista que conforma el recorrido e ir ejecutando las operaciones correspondientes en cada una de las máquinas por las que va pasando.
- Secuencia Estimulo/ Respuesta: Se consigue la ejecución en las distintas máquinas de los jobs incluidos en el fichero de ejecución y los tiempos en cada una de las máquinas. Además se van introduciendo una serie de valores en las distintas máquinas que me permiten controlar el flujo de ejecución de la aplicación.
- Requerimientos funcionales: Se requiere conectividad al GRID además de que el Grid_proxy_init se haya efectuado con anterioridad al lanzamiento de la ejecución.

Flujo Normal

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable o a través del botón diseñado a tal fin.	
	2. El sistema comprueba los datos, realiza la creación y transferencia de los ficheros necesarios para la ejecución al directorio establecido por defecto.
	3. Se comprueban todos los parámetros y en caso de que todo sea correcto se procede al lanzamiento de la aplicación que se irá autoejecutando en las distintas máquinas.
4. El usuario puede ir comprobando la ejecución de la aplicación a través de las herramientas de visualización.	

Alternativa 1

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable o a través del botón diseñado a tal fin.	
	2. El sistema comprueba los datos, realiza la creación y transferencia de los ficheros necesarios para la ejecución al directorio

	establecido por defecto.
	3. Se comprueban todos los parámetros y en caso de que ocurra algún tipo de error grave se cancelará la operación de lanzamiento y se mostrará el error al usuario. En otro caso si el error se puede solventar, se mostrara el mensaje de error al usuario para que este informado de la ejecución.

- *Selección de las máquinas que conforman el recorrido*

- Descripción: Permite añadir una máquina al listado de máquinas que están disponibles para ser usadas por la aplicación.
- Secuencia Estimulo/ Respuesta: Al añadir una máquina, se inserta a la lista de máquinas que conforman la secuencia de ejecución y también se añade a la lista de máquinas disponibles escribiendo su dirección en el fichero de configuración correspondiente.
- Requerimientos funcionales: Para poder añadir una máquina se requiere que tenga conectividad en el GRID para lo cual se hacen una serie de comprobaciones previas.

Flujo Normal

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable o a través del icono correspondiente.	
	2. El sistema solicita al usuario los datos de la máquina a añadir (la URL).
3. El usuario indica estos datos y pulsa "Aceptar"	
	4. El sistema comprueba que se ha rellenado la URL de la máquina correspondiente y comprueba la conectividad de la máquina.
	5. Si la existe conectividad entonces la añade a los listados de máquinas.

Alternativa 1

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable o a través del icono correspondiente.	
	2. El sistema solicita al usuario los datos de la máquina a añadir (la URL).
3. El usuario indica estos datos y pulsa "Aceptar"	
	4. El sistema comprueba que se ha rellenado la URL de la máquina

	correspondiente y comprueba la conectividad de la máquina.
	5. Cancela la adicción de la máquina en caso de que no exista conectividad a la máquina indicada y lo muestra al usuario.

Alternativa 2

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario elige esta opción en el menú desplegable o a través del icono correspondiente.	
	2. El sistema solicita al usuario los datos de la máquina a añadir (la URL).
3. El usuario indica estos datos y pulsa "Cancelar"	
	4. Se cancela la operación.

- *Inserción/Eliminación máquinas en la lista de recorrido*

- Descripción: Consiste en la selección por parte del usuario de las máquinas que se desean que formen parte del recorrido de ejecución de la aplicación.
  Al mismo tiempo se pueden eliminar máquinas de la lista que conforma el recorrido para evitar que la aplicación pase por ellas.
- Secuencia Estimulo/ Respuesta: Al hacer doble clic sobre el listado de la parte superior, la máquina seleccionada se añade y pasa a formar parte del listado de máquinas que conforman el recorrido.
  Si se efectúa el doble clic sobre alguna de las máquinas que conforman el recorrido, está se eliminará de este listado.
- Requerimientos funcionales: No se requieren requisitos funcionales.

Flujo Normal → Operación Añadir.

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario las máquinas en el listado de máquinas disponibles en el GRID efectuando un doble clic sobre alguna de ellas	
	2. Procede a la inserción de la máquina en el listado que conforma el recorrido de ejecución de la aplicación.

Flujo Normal-> Operación eliminar

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario las máquinas en el listado de máquinas configuradas en ese momento para el recorrido de ejecución efectuando un doble clic sobre alguna de ellas	
	2. Procede a la eliminación de la máquina

	del listado que conforma el recorrido de ejecución.

- *Ver ficheros de trazas de ejecución*

- Descripción: Permite el visionado de las trazas de ejecución de las distintas máquinas para conseguir de forma centralizada una visualización global del flujo de ejecución.
- Secuencia Estimulo/ Respuesta: Al pulsar sobre el botón de visualización de los logs, se muestran los ficheros de trazas de cada una de las máquinas que conformaron el recorrido de ejecución para así comprobar la ejecución de la aplicación en estas máquinas.
- Requerimientos funcionales: Se requieren que los ficheros de trazas sean reportados por las máquinas según terminen la ejecución.

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario pulsa sobre el icono que proporciona el acceso a los ficheros de trazas.	
	2. Permite visualizar todos los ficheros de trazas que haya ido recibiendo de las distintas máquinas en las que se ha ejecutado la aplicación.

- *Cambio de apariencia*

- Descripción: Permite cambiar la configuración de la ventana consiguiendo que sea más atractiva al usuario.
- Secuencia Estimulo/ Respuesta: Al pulsar sobre el menú desplegable de "Configuración->Aspecto->" se puede cambiar la apariencia de la interfaz gráfica para que ésta sea más amigable.
- Requerimientos funcionales: No tiene requerimientos funcionales.

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario el menú desplegable y elige la opción de "Aspecto" y dentro de la lista de aspectos el aspecto deseado	
	2. El sistema procede al cambio completo del aspecto de la interfaz gráfica.

- *Ver registro de errores*

- Descripción: Al igual que en el Caso de uso 3.5 la funcionalidad de éste es similar pero para poder visualizar el registro de errores donde se almacenan todos los errores que se han podido generar en la ejecución sobre las distintas máquinas.
- Secuencia Estimulo/ Respuesta: Mediante un "oyente" se va creando el registro de errores según se van produciendo estos. Se puede visualizar la máquina que produjo el error, el tipo de error y la descripción del mismo:
- Requerimientos funcionales: No tiene requisitos funcionales.

Flujo de Eventos	
Usuario	**Sistema**

1. Se inicia cuando el usuario pulsa sobre el icono que proporciona el acceso al registro de errores.	
	2. Se muestra el registro de errores que se ha podido ir construyendo a lo largo de la ejecución de la aplicación. Si no se ha producido ningún error mostrará el registro en blanco.

- *Movimiento en el listado de las máquinas que conforman el recorrido*

- Descripción: Para la modificación del orden de ejecución en las distintas máquinas se usarán los botones "Subir" y "Bajar" que alteran la secuencia de máquinas del recorrido.
- Secuencia Estimulo/ Respuesta: Se produce la alteración del orden de ejecución para que este de acuerdo con el criterio del usuario.
- Requerimientos funcionales: Se requiere que haya más de una máquina para comprobar el movimiento puesto que con menos no se efectuará ninguna alteración.

Flujo Normal.

Flujo de Eventos	
Usuario	**Sistema**
1. Se inicia cuando el usuario pulsa sobre los botones de "Subir" o "Bajar".	
	2. El sistema procede a la modificación del listado de máquinas que conforman el recorrido.

7.3. DIAGRAMA DE CLASES WORM

A continuación se va a proceder a realizar un breve comentario de las tareas a las que se van a dedicar cada una de las clases contenidas en este diagrama.

- **ParserError**: Esta clase permite realizar la interpretación y construcción de un objeto del tipo "DataError" con el fin de almacenar la información de los errores que se generan en la aplicación y que son almacenados en un fichero XML por medio de la clase "XMLError".

- **XmlException**: Esta clase permite la creación de un fichero de extensión XML que contendrá los errores que se van a ir produciendo a lo largo de la ejecución de la aplicación. Una vez creado el fichero "Errores.xml" se encarga de reportarlo a la máquina de partida, mediante el uso de la clase "FileTransfer", para que se solventen las situaciones de error y se pueda continuar con la ejecución de la aplicación.

- **NotificationsThread**: Clase que extiende de "Thread" y que realiza un testeo del fichero de errores.

- Cuando se detecta un error se trata en esta clase efectuando las acciones necesarias para crear un nuevo contexto de ejecución y volver a lanzar la aplicación. Requiere acceso a los ficheros de configuración para realizar las modificaciones oportunas.

▪ **DataError**: Clase que contiene los datos identificativos de los errores que se puedan producir durante la ejecución.

▪ **FileTransfer**: La transferencia de ficheros entre máquinas del GRID se realiza a través de esta clase. Requiere del uso del proxy creado mediante la clase "Credentials". Además requiere de un "listener" que se encarga de comprobar y emitir una señal cuando la transferencia se ha completado.

▪ **TransferListener**: "Listener" encargado de comunicar a lo objetos de la clase "FileTransfer" que la transferencia a la máquina remota se ha realizado.

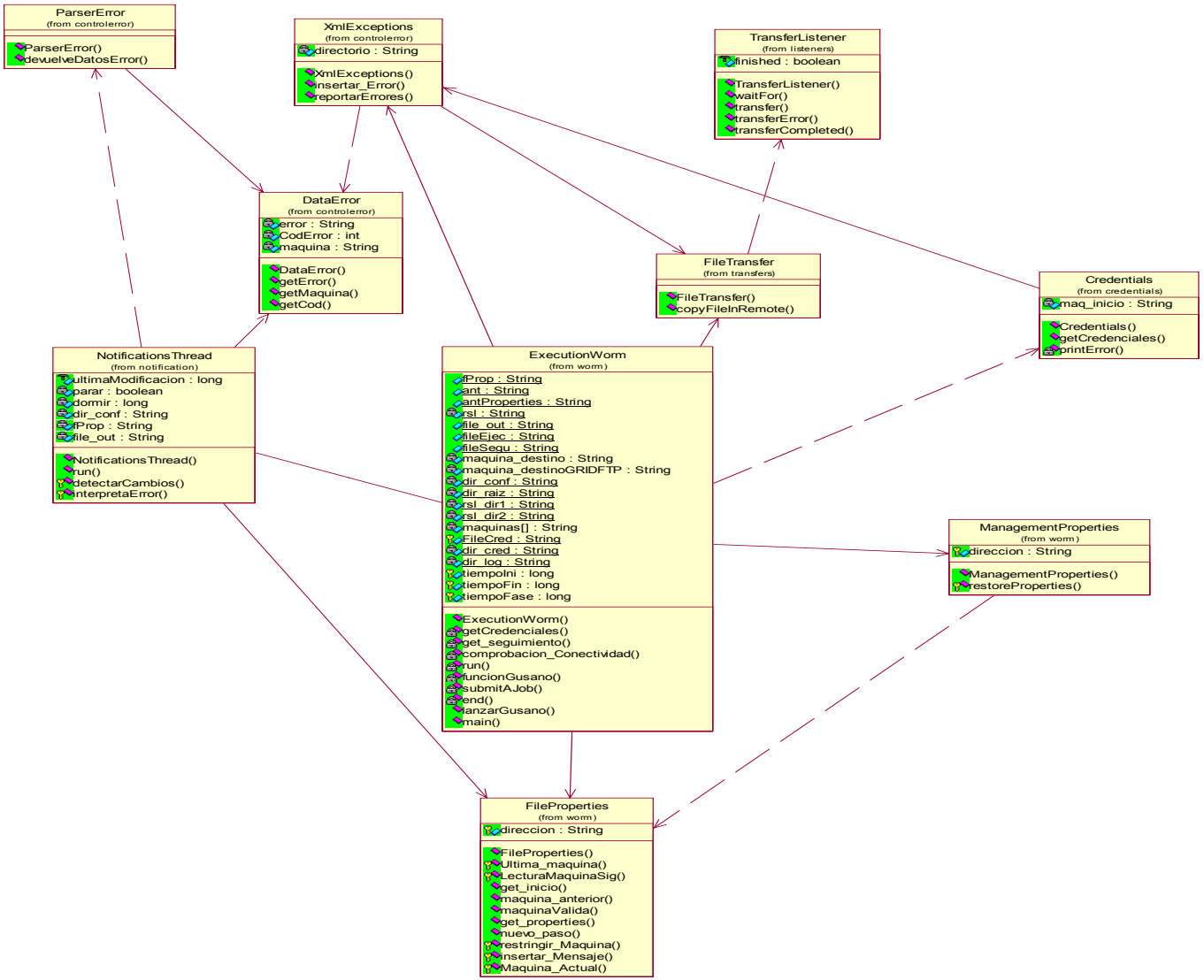
▪ **Credentials**: Permite la creación de un proxy necesario para realizar las operaciones de transferencia, submit etc… sobre el GRID.

▪

▪ **ExecutionWorm**: Es la clase principal del núcleo de la aplicación y se encarga de realizar las operaciones principales y de gestionar todo el flujo de ejecución. En esta clase se crean todos los objetos necesarios de las otras clases y se realizan los mandatos de las operaciones correspondientes. También requiere el uso de un fichero "Ejecución.properties" que contiene los jobs necesarios en la ejecución.

▪ **ManagementProperties**: Se encarga de efectuar la gestión de fichero de salida "salidas.properties" que contiene la información sobre las máquinas por las que ha pasado la aplicación durante la ejecución.

▪ **FileProperties**: Hace uso de los ficheros de propiedades para obtener los datos necesarios para la ejecución tales como el listado de máquinas, inserción de mensajes en el fichero de "Seguimiento.properties" que almacena la información de ejecución (maquinas, tiempos de ejecución, tiempos de transferencia etc…).


ParserError
(from controlerror)
ParserError()
devuelveDatosError()
XmlExceptions
(from controlerror)
directorio : String
XmlExceptions()
insertar_Error()
reportarErrores()
TransferListener
(from listeners)
finished : boolean
TransferListener()
waitFor()
transfer()
transferError()
transferCompleted()
DataError
(from controlerror)
error : String
CodError : int
maquina : String
DataError()
getError()
getMaquina()
getCod()
FileTransfer
(from transfers)
FileTransfer()
copyFileInRemote()
Credentials
(from credentials)
maq_inicio : String
Credentials()
getCredenciales()
printError()
NotificationsThread
(from notification)
ultimaModificacion : long
parar : boolean
domir : long
dir_conf : String
fProp : String
file_out : String
NotificationsThread()
run()
detectarCambios()
interpretaError()
ExecutionWorm
(from worm)
fProp : String
ant : String
antProperties : String
rsl : String
file_out : String
fileEjec : String
fileSegu : String
maquina_destino : String
maquina_destinoGRIDFTP : String
dir_conf : String
dir_raiz : String
rsl_dir1 : String
rsl_dir2 : String
maquinas[] : String
FileCred : String
dir_cred : String
dir_log : String
tiempoIni : long
tiempoFin : long
tiempoFase : long
ExecutionWorm()
getCredenciales()
get_seguimiento()
comprobacion_Conectividad()
run()
funcionGusano()
submitAJob()
end()
lanzarGusano()
main()
ManagementProperties
(from worm)
direccion : String
ManagementProperties()
restoreProperties()
FileProperties
(from worm)
direccion : String
FileProperties()
Ultima_maquina()
LecturaMaquinaSig()
get_inicio()
maquina_anterior()
maquinaValida()
get_properties()
nuevo_paso()
restringir_Maquina()
insertar_Mensaje()
Maquina_Actual()

7.4. DIAGRAMA DE CLASES WORMGUI

▪ **NotificationsGUI:**Hilo que comprueba si ha cambiado el fichero de seguimiento para mostrarlo en la gráfica en caso de que lo haya hecho.Cuando en un determinado tiempo no detecta cambios, el hilo se destruye.

▪ **NotificationsThread:**Hilo que comprueba si ha cambiado el fichero de errores.En caso de detectar este cambio, se muestra un pequeño aviso en la esquina inferior derecha y se añade a la tabla de errores detectados.

▪ **BlueTheme, DefaultTheme, GrayTheme, WhiteSatinTheme:**Cada uno de ellos sobreescriben DefaultMetalTheme para cambiar el look & feel de la aplicación. ealmente, lo únicamente que cambia son los colores y las fuentes que se utilizan en la gráfica.

▪ **FormatDataTime:**Clase utilizada para pintar la regla de la gráfica de evolución.En ella se indica el número de rayas pequeñas entre dos principales, el número de unidades consideradas entre dos rayas, número de pixeles entre una raya y otra de la regla y el número de pixeles que ocupa una unidad de tiempo. Además, es en esta clase donde se lleva a cabo el zoom, tanto in como out de la gráfica.

▪ **InitialConfiguration:**Al iniciar la aplicación, se utiliza esta clase para obtener todas las máquinas de los ficheros "maquinas.properties" y "maquinasGrid.properties". Al lanzar el gusano, necesita esta clase para inicializar y cargar todos los ficheros necesarios en la ruta elegida.

▪ **MachineDataGUI:**Estructura de datos necesaria para guardar la información de la evolución del gusano.La fase indica lo que hace el gusano y el intervalo durante cuánto tiempo.

▪ **ColumMachine**: Clase que extiende de JPanel y que representa la columna de todas las máquinas del gusano. Es el panel que se muestra a la izquierda de la gráfica.

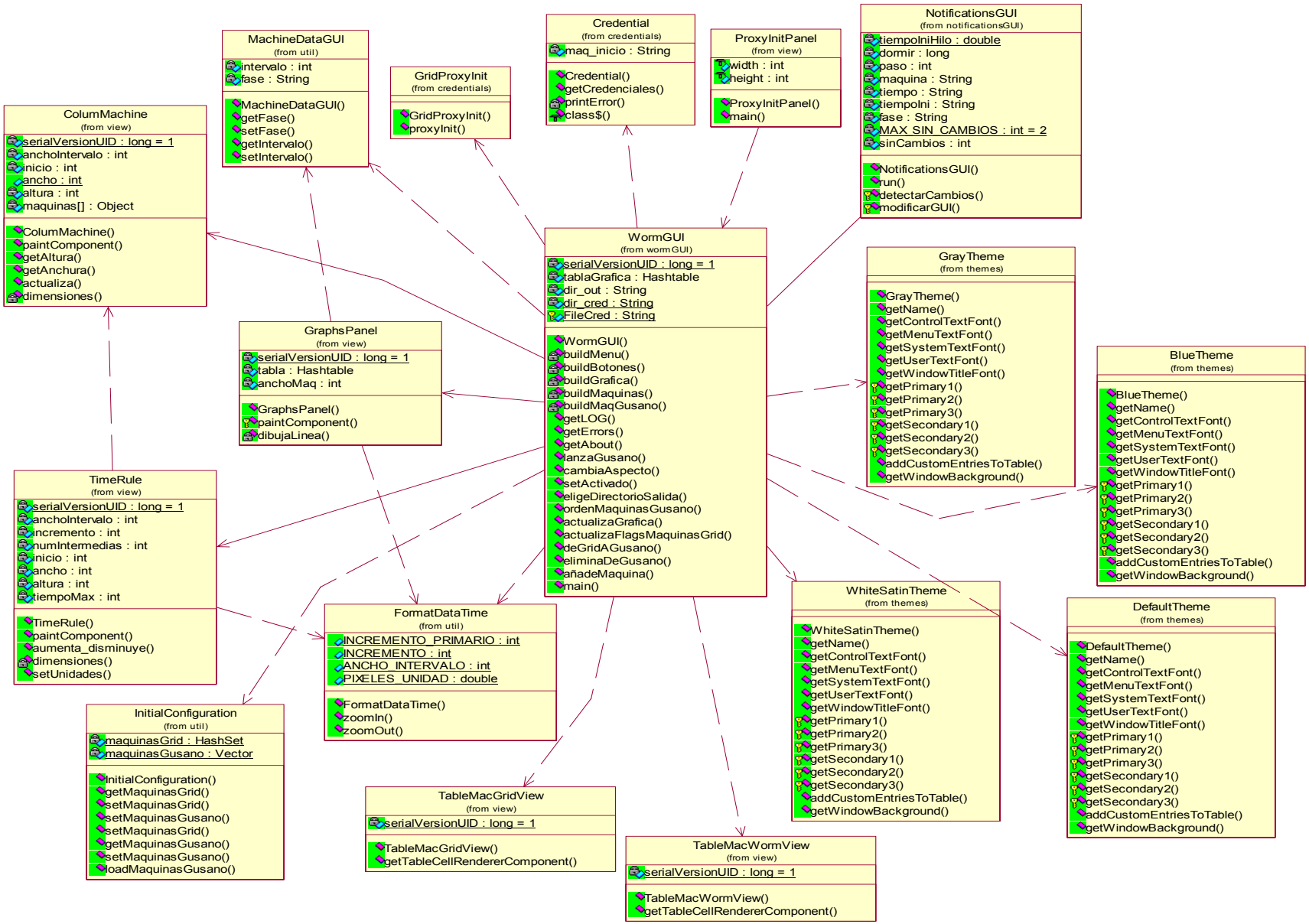
▪ **TimeRule**:Clase que extiende de JPanel y que representa la regla temporal que se muestra sobre la gráfica de evolución del gusano.

▪ **GraphsPanel**:Panel sobre el que se va pintando las barras de evolución del gusano

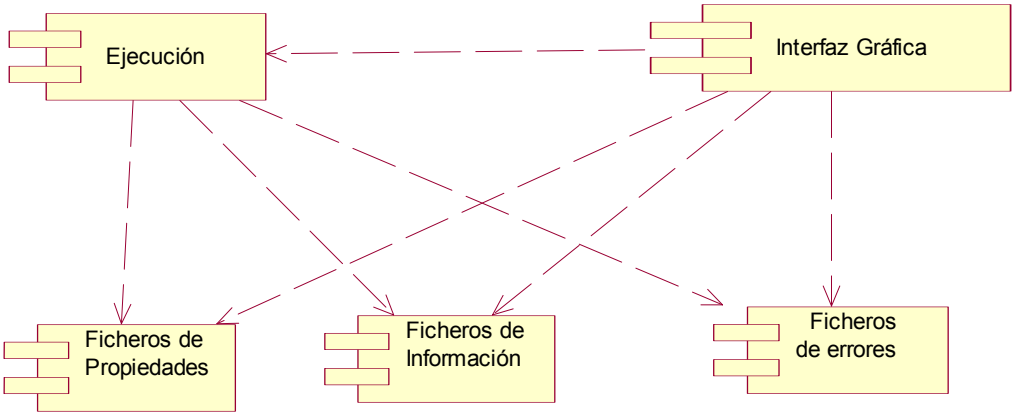
▪ **TableMacGridView:**Extensión de DefaultTableCellRenderer para la representación de la JTable que contiene la lista de todas las máquinas que hay sobre el Grid.

▪ **TableMacWormView:**Extensión de DefaultTableCellRenderer para la representación de la JTable que contiene la lista de todas las máquinas que pasarán sobre el gusano.

▪ **WormGUI:**Clase principal. En ella se crea todos los componentes visuales de la GUI.


MachineDataGUI
(from util)
intervalo : int
fase : String
MachineDataGUI()
getFase()
setFase()
getIntervalo()
setIntervalo()
GridProxyInit
(from credentials)
GridProxyInit()
proxyInit()
Credential
(from credentials)
maq_inicio : String
Credential()
getCredenciales()
printError()
class$()
ProxyInitPanel
(from view)
width : int
height : int
ProxyInitPanel()
main()
NotificationsGUI
(from notificationsGUI)
tiempoIniHilo : double
dormir : long
paso : int
maquina : String
tiempo : String
tiempoIni : String
fase : String
MAX_SIN_CAMBIOS : int = 2
sinCambios : int
NotificationsGUI()
run()
detectarCambios()
modificarGUI()
ColumMachine
(from view)
serialVersionUID : long = 1
anchoIntervalo : int
inicio : int
ancho : int
altura : int
maquinas[] : Object
ColumMachine()
paintComponent()
getAltura()
getAnchura()
actualiza()
dimensiones()
WormGUI
(from wormGUI)
serialVersionUID : long = 1
tablaGrafica : Hashtable
dir_out : String
dir_cred : String
FileCred : String
WormGUI()
buildMenu()
buildBotones()
buildGrafica()
buildMaquinas()
buildMaqGusano()
getLOG()
getErrors()
getAbout()
lanzaGusano()
cambiaAspecto()
setActivado()
eligeDirectorioSalida()
ordenMaquinasGusano()
actualizaGrafica()
actualizaFlagsMaquinasGrid()
deGridAGusano()
eliminaDeGusano()
añadeMaquina()
main()
GrayTheme
(from themes)
GrayTheme()
getName()
getControlTextFont()
getMenuTextFont()
getSystemTextFont()
getUserTextFont()
getWindowTitleFont()
getPrimary1()
getPrimary2()
getPrimary3()
getSecondary1()
getSecondary2()
getSecondary3()
addCustomEntriesToTable()
getWindowBackground()
BlueTheme
(from themes)
BlueTheme()
getName()
getControlTextFont()
getMenuTextFont()
getSystemTextFont()
getUserTextFont()
getWindowTitleFont()
getPrimary1()
getPrimary2()
getPrimary3()
getSecondary1()
getSecondary2()
getSecondary3()
addCustomEntriesToTable()
getWindowBackground()
GraphsPanel
(from view)
serialVersionUID : long = 1
tabla : Hashtable
anchoMaq : int
GraphsPanel()
paintComponent()
dibujaLinea()
TimeRule
(from view)
serialVersionUID : long = 1
anchoIntervalo : int
incremento : int
numIntermedias : int
inicio : int
ancho : int
altura : int
tiempoMax : int
TimeRule()
paintComponent()
aumenta_disminuye()
dimensiones()
setUnidades()
FormatDataTime
(from util)
INCREMENTO_PRIMARIO : int
INCREMENTO : int
ANCHO_INTERVALO : int
PIXELES_UNIDAD : double
FormatDataTime()
zoomIn()
zoomOut()
WhiteSatinTheme
(from themes)
WhiteSatinTheme()
getName()
getControlTextFont()
getMenuTextFont()
getSystemTextFont()
getUserTextFont()
getWindowTitleFont()
getPrimary1()
getPrimary2()
getPrimary3()
getSecondary1()
getSecondary2()
getSecondary3()
addCustomEntriesToTable()
getWindowBackground()
DefaultTheme
(from themes)
DefaultTheme()
getName()
getControlTextFont()
getMenuTextFont()
getSystemTextFont()
getUserTextFont()
getWindowTitleFont()
getPrimary1()
getPrimary2()
getPrimary3()
getSecondary1()
getSecondary2()
getSecondary3()
addCustomEntriesToTable()
getWindowBackground()
InitialConfiguration
(from util)
maquinasGrid : HashSet
maquinasGusano : Vector
InitialConfiguration()
getMaquinasGrid()
setMaquinasGrid()
setMaquinasGusano()
setMaquinasGrid()
getMaquinasGusano()
setMaquinasGusano()
loadMaquinasGusano()
TableMacGridView
(from view)
serialVersionUID : long = 1
TableMacGridView()
getTableCellRendererComponent()
TableMacWormView
(from view)
serialVersionUID : long = 1
TableMacWormView()
getTableCellRendererComponent()

7.5. DIAGRAMA DE COMPONENTES



En este diagrama podemos distinguir una serie de componentes que pasamos a describir a continuación:

Componente "Ejecución":

En esta componente podemos distinguir dos partes esenciales para la ejecución de la aplicación:

Worm.jar: Que contiene el código necesario para la ejecución de la aplicación en las distintas máquinas (Código java).

Build.xml: Encargado del lanzamiento a ejecución del código descrito en el punto previo.

Componente "Ficheros de Propiedades"

En este componente incluimos todos aquellos ficheros que contienen información necesaria para la ejecución de la aplicación tales como las máquinas en las que se va a ejecutar, el fichero de credenciales, los jobs a ejecutar, el configurador del API de trazas Log4j, y el fichero de propiedades asociado a "build.xml" llamado "build.properties".

Componente "Ficheros de información"

El seguimiento de la ejecución se realizará a partir de una serie de ficheros que contienen información de esta índole. Un ejemplo claro es el fichero de trazas generado por "log4j" que permite realizar un seguimiento del curso de la ejecución.

Componente "Ficheros de errores"

La posible ocurrencia de errores desencadena la necesidad de usar un fichero de extensión .XML que me permita guardar un registro de errores y solventar aquellas situaciones en las que sea posible hacerlo.

Componente "Interfaz Gráfica"

Para facilitar el uso de toda esta aplicación se ha desarrollado una interfaz gráfica que me permite ver todos los detalles de la ejecución de un modo más intuitivo y familiar.

7.6. DIAGRAMAS DE SECUENCIA WORMGUI

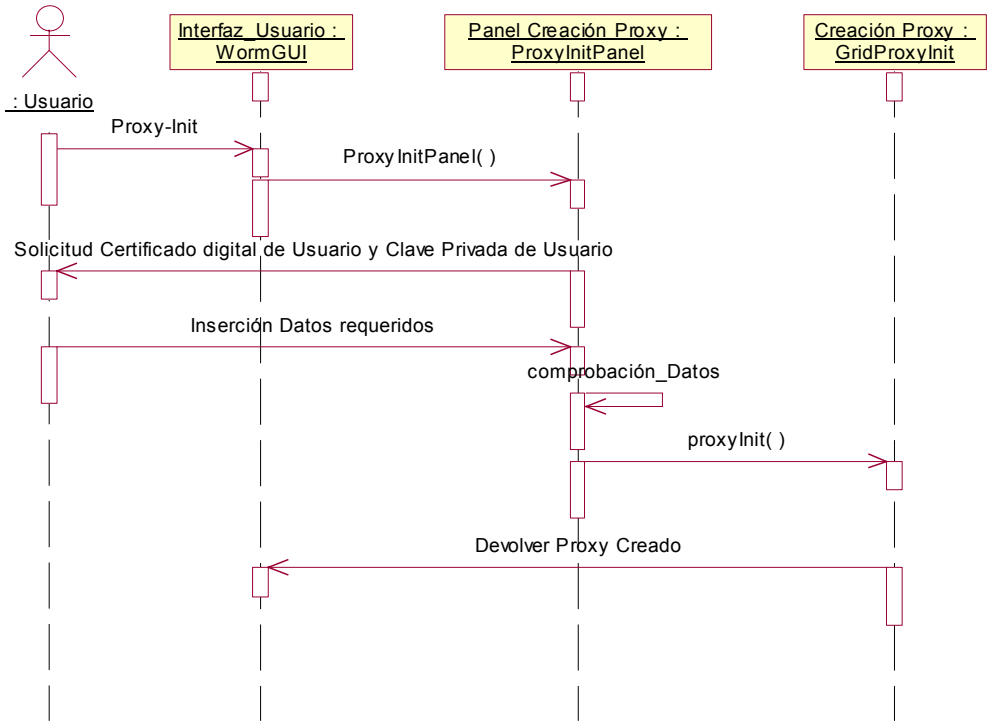
7.6.1. Diagrama de secuencia: Proxy-Init.

Para poder ejecutar el Grid_proxy_Init se requieren una serie de parámetros que han de ser introducidos por el usuario y posteriormente comprobados con el fin de que se pueda crear de forma correcta el proxy que permitirá al usuario la ejecución en el GRID.

En la primera figura se muestra la ejecución de flujo normal para el caso de uso "Proxy-Init" en la que los datos proporcionados por el usuario son correctos y se puede crear el proxy sin ningún tipo de problema

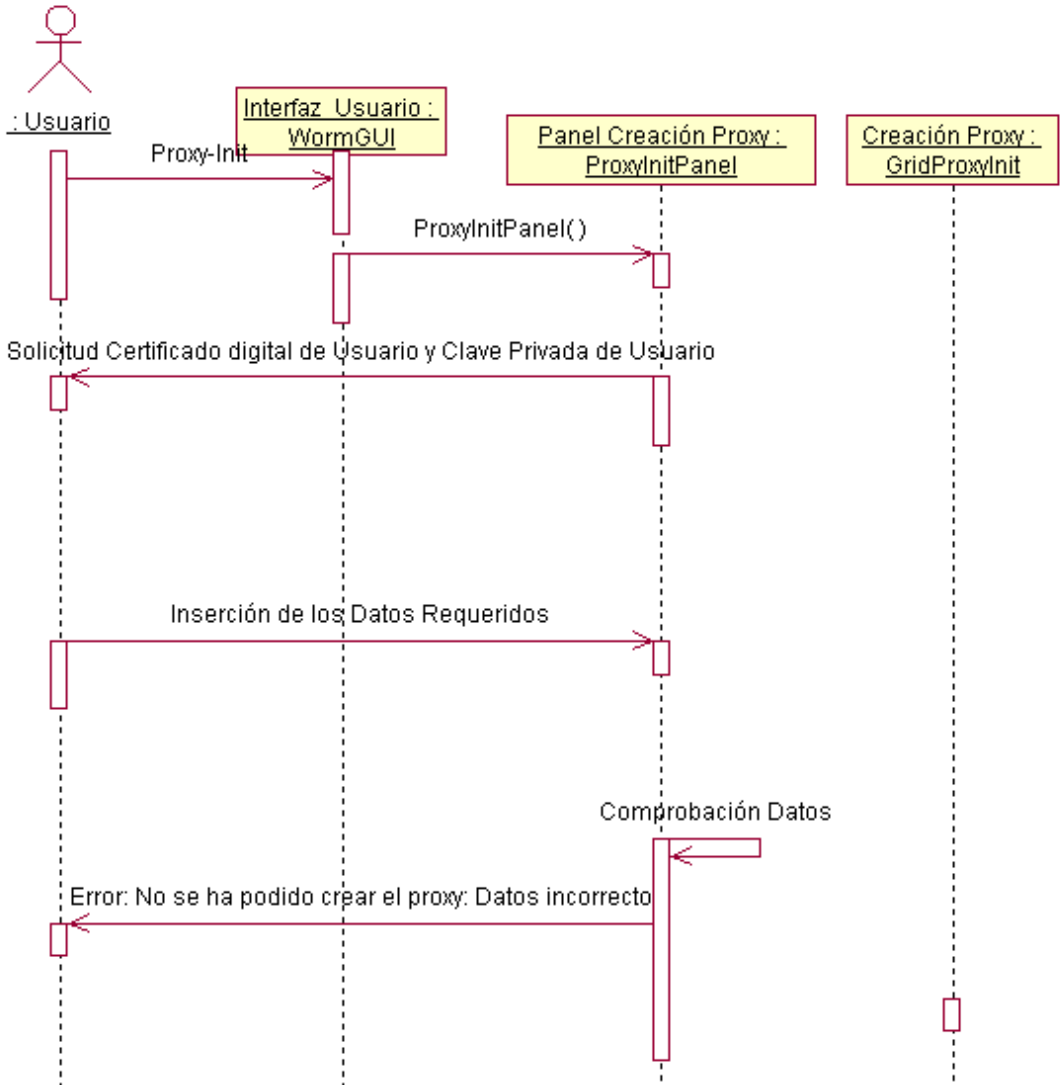
El resto de figuras muestran ejecuciones en las que se ha producido algún tipo de error.

1. Secuencia correcta



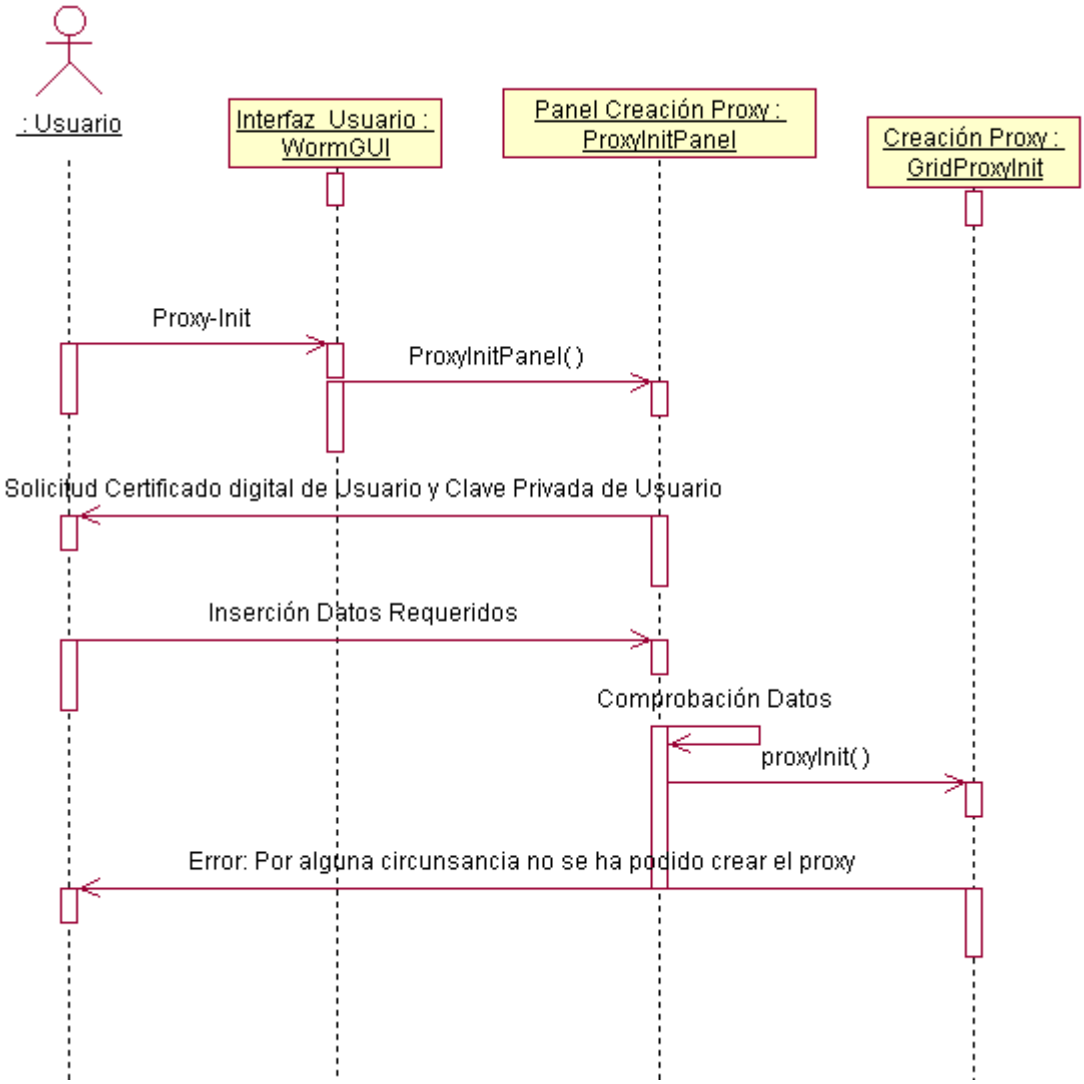
2. Secuencia Error: *Los datos proporcionados por el usuario son incorrectos.*

En este caso se recogen los datos del usuario y durante la comprobación se detecta alguna anomalía que hace detener el proceso e informar a usuario de esta situación.



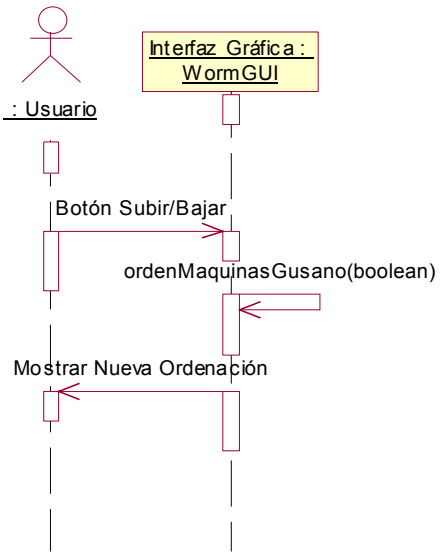
3. ***Secuencia Error:*** *Error durante la creación del proxy*

En este caso la validación de los datos ha sido correcta, sin embargo por alguna circunstancia no se ha podido efectuar la creación del proxy. Esta situación tiene que ser informada al usuario con el fin de que no proceda al lanzamiento de la aplicación ya que provocaría un error durante su ejecución por no tener los permisos de acceso al GRID creados.



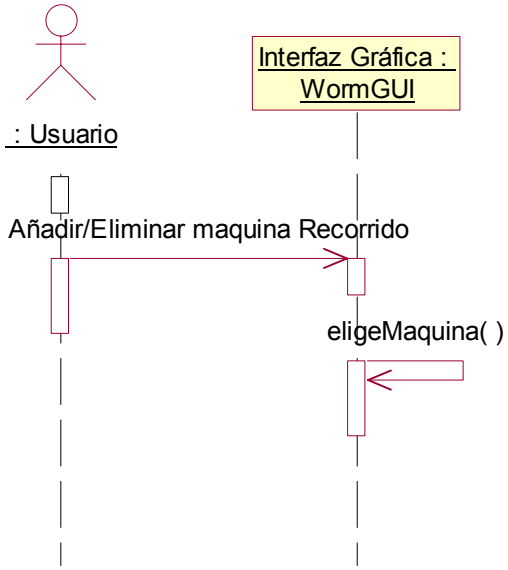
7.6.2. Diagrama Secuencia: Ordenación Máquinas recorrido

La secuencia de ejecución consiste en realizar la ordenación de las máquinas de acuerdo a las necesidades del usuario. Para realizar estas acciones se usan unos botones que permiten mover las máquinas de forma ascendente o descendente. No se muestran situaciones anómalas puesto que es una operación trivial y no puede fallar.



7.6.3. Diagrama Secuencia: Añadir/Eliminar máquinas al recorrido

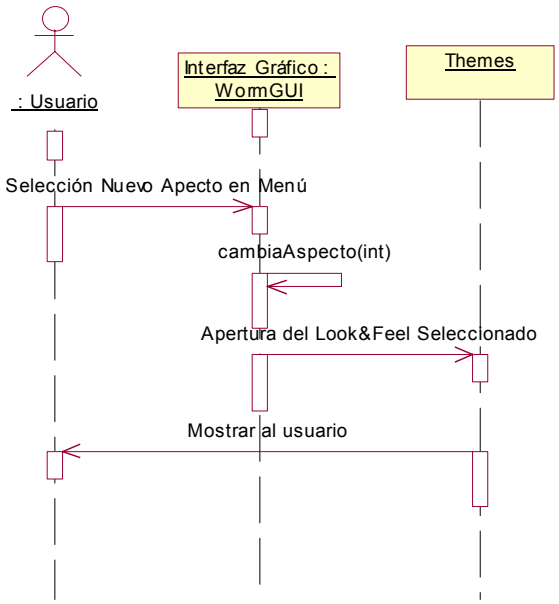
La configuración del recorrido que seguirá la aplicación consiste en añadir las máquinas seleccionadas a un listado que conformará el recorrido final. Para añadir se pulsa sobre el listado de máquinas disponibles en los listados y se colocarán en el listado de máquinas del recorrido. Para eliminar se efectúa la misma operación pero sobre las máquinas del listado que conforma el recorrido.



7.6.4. Diagrama Secuencia: Cambiar Aspecto

Se podrá realizar un cambio de aspecto de la interfaz de usuario mediante este caso de uso. La secuencia de ejecución consiste en seleccionar en la Interfaz gráfica la opción correspondiente. Una vez elegida está opción procederá a elegir el "THEME" correspondiente de entre los que hay disponibles.

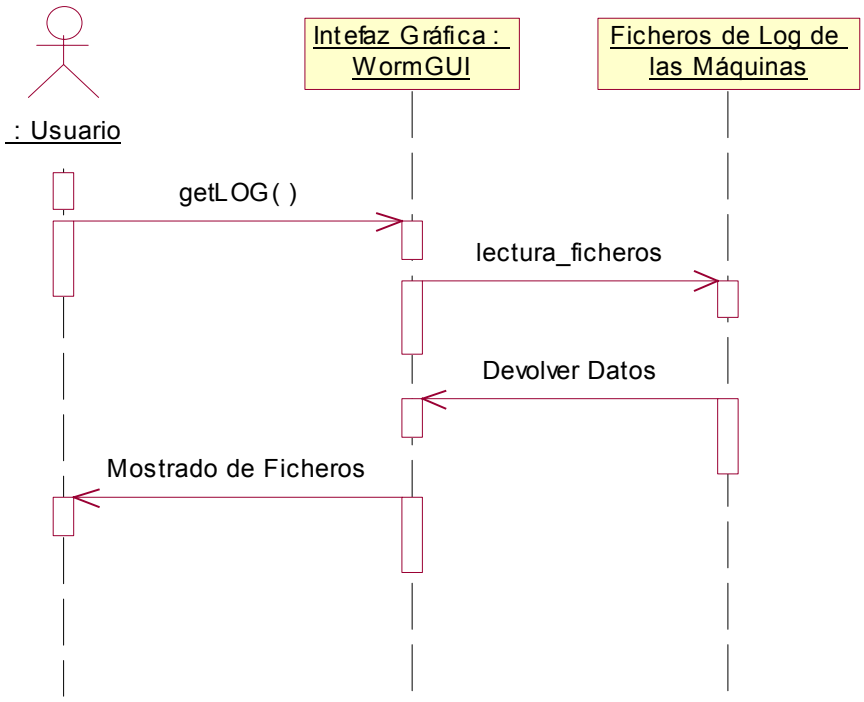
El flujo normal de ejecución es siguiente, y no cabe la posibilidad de situaciones de error.



7.6.5. Diagrama Secuencia: Ver Ficheros de LOG

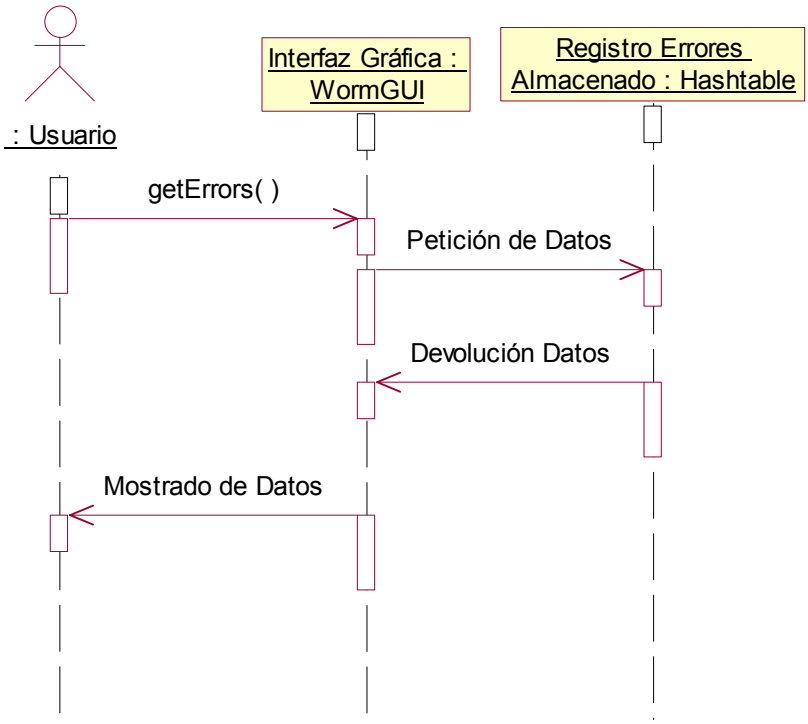
El visionado de las trazas generadas en las distintas máquinas ha de estar disponible para el usuario. Esto se efectúa mediante el visionado de los "logs".

En la interfaz de usuario se elige esta opción que cargara los datos situados en el directorio correspondiente y serán mostrados al usuario.



7.6.6. Diagrama Secuencia: Lectura registro de errores

Al igual que en el caso de uso previo ("Ver Ficheros de Log") el usuario deseará tener acceso a los posibles errores que se hayan ido produciendo a lo largo de la ejecución. Para todo ello se elige esta opción en la interfaz que acudirá al registro de errores que se ha ido creando durante la ejecución y podrá visualizarlos.



7.6.7. Diagrama Secuencia: Insertar nuevas máquinas

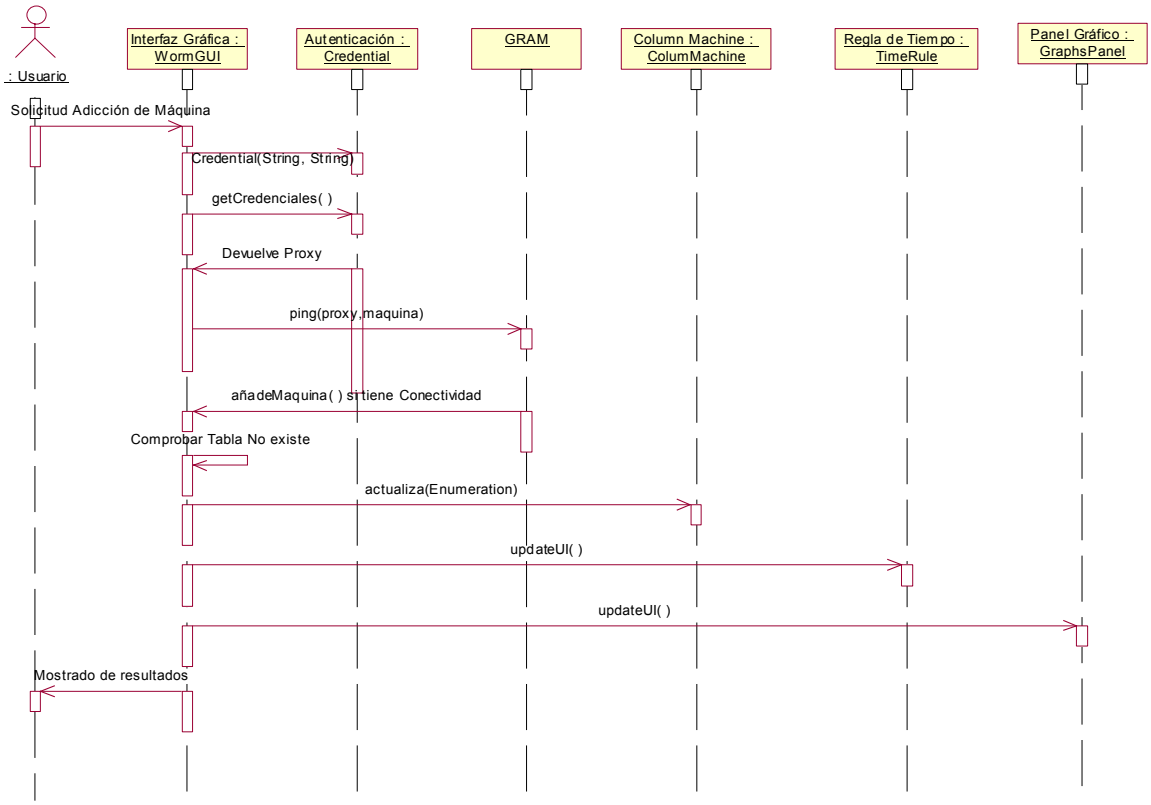
A la hora de añadir máquinas que no están incluidas en la lista de máquinas disponibles para la ejecución en el GRID, se puede efectuar mediante esta opción de la interfaz de usuario.

Para añadir las máquinas hay que realizar una serie de comprobaciones, ya que se requiere que el proxy este creado para ese usuario y también comprobar la conectividad con la máquina para evitar situaciones de error.

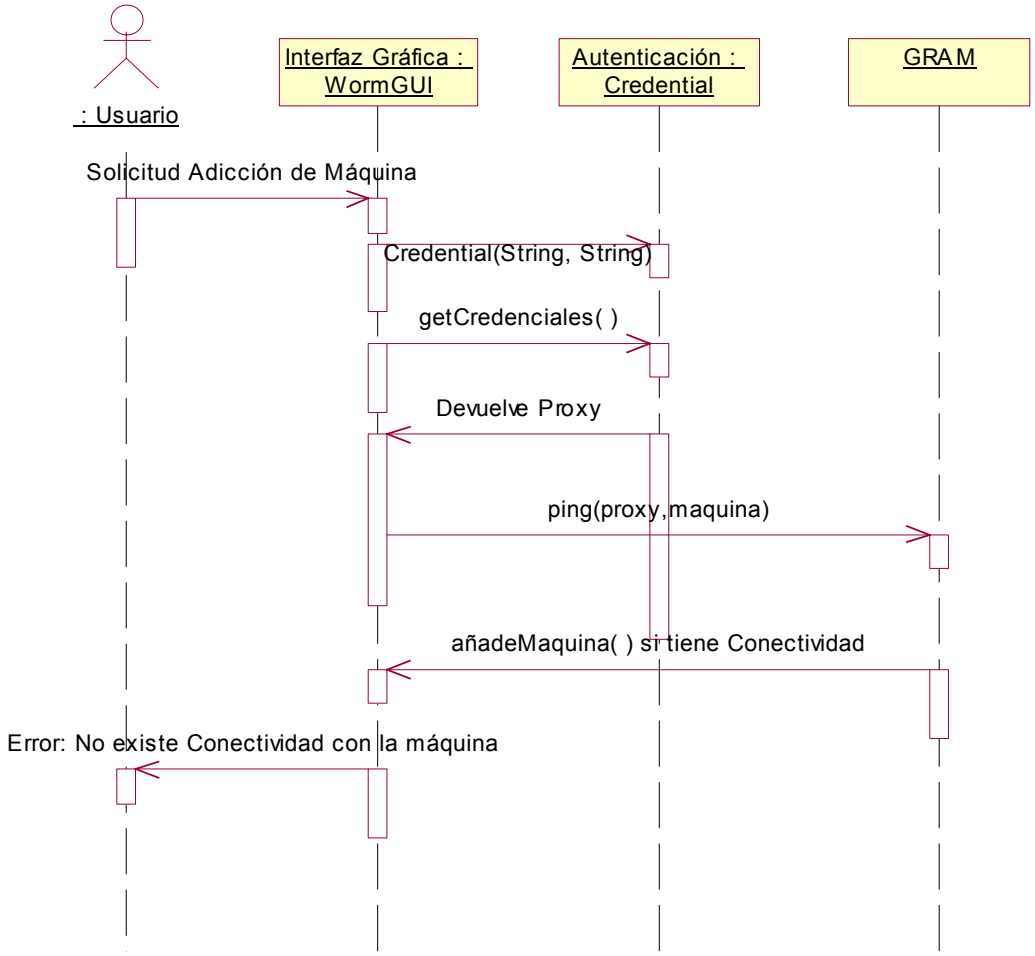
Si cualquiera de las dos situaciones previas no se dan, se devolverá un mensaje al usuario para indicarle que no ha sido posible añadir la máquina deseada. Otra situación de error que puede producirse es que la máquina añadida ya exista. Ante esta última situación se procederá a informar al usuario y terminar con la operación correspondiente.

A continuación se muestran los diagramas de secuencia para un flujo de ejecución correcto y para las situaciones erróneas antes descritas.

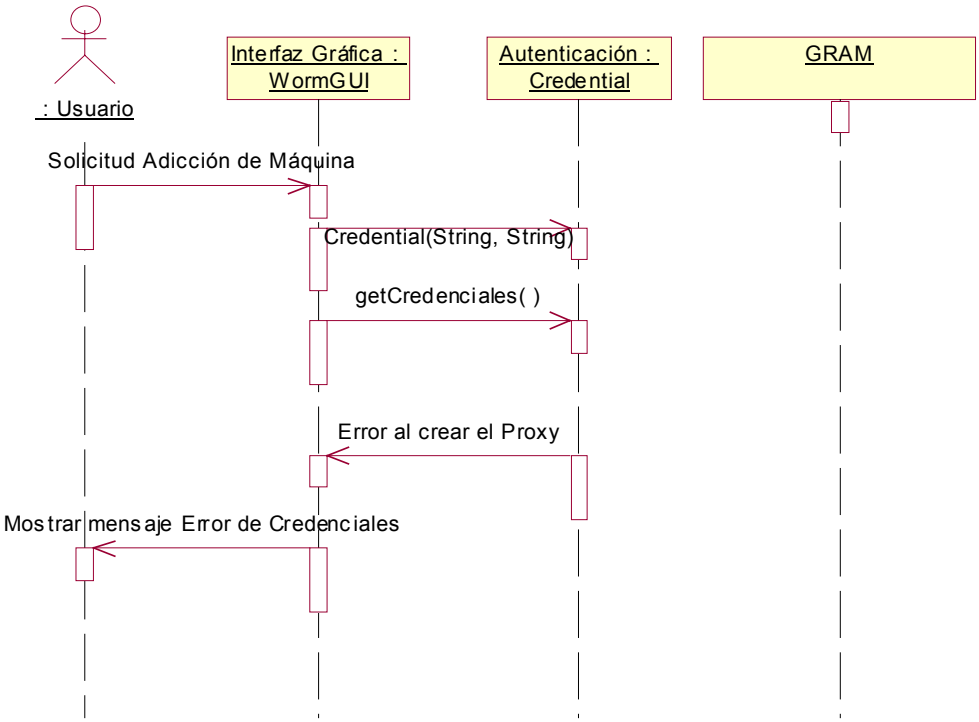
1. Secuencia Correcta



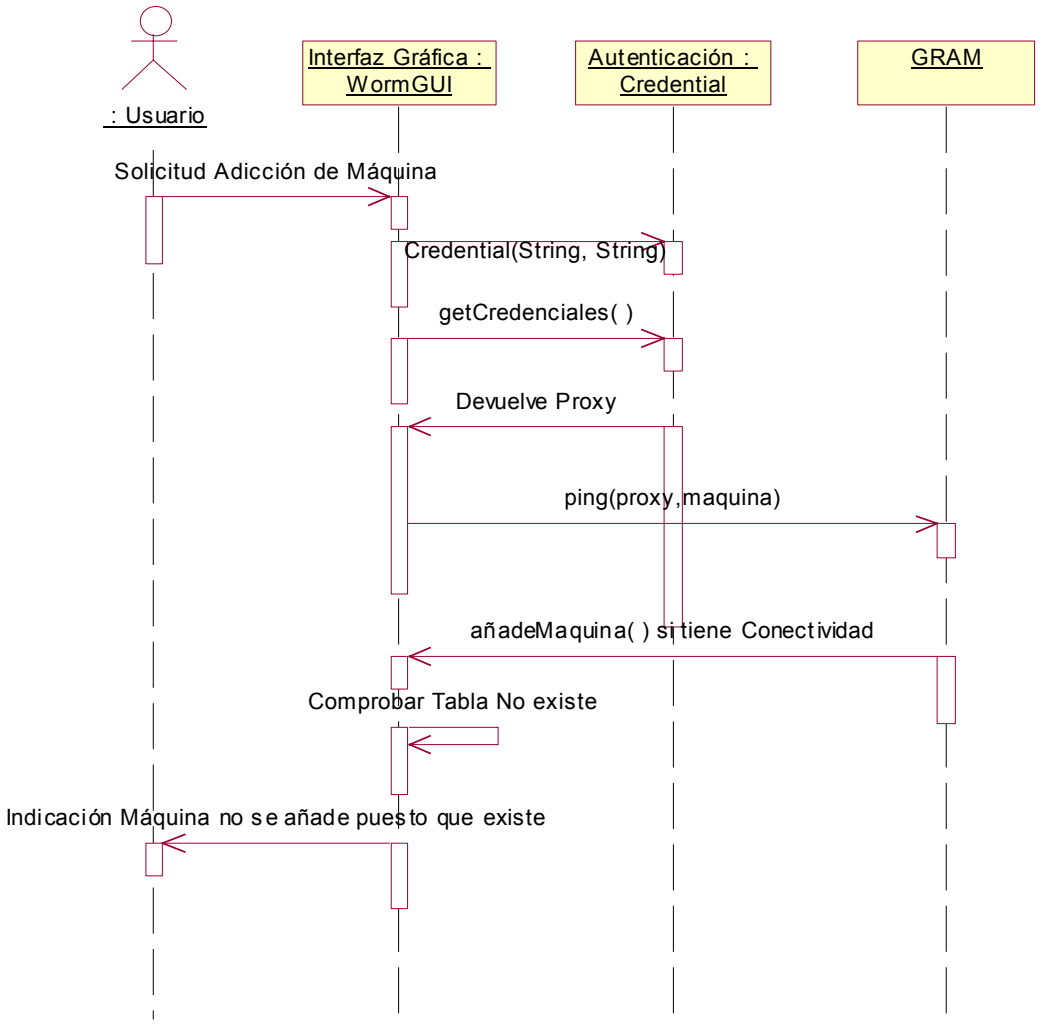
2. Secuencia Error: ***No se ha podido realizar el PING a la máquina***



3. Secuencia Error: No se puede crear las credenciales: No se ha iniciado el proxy



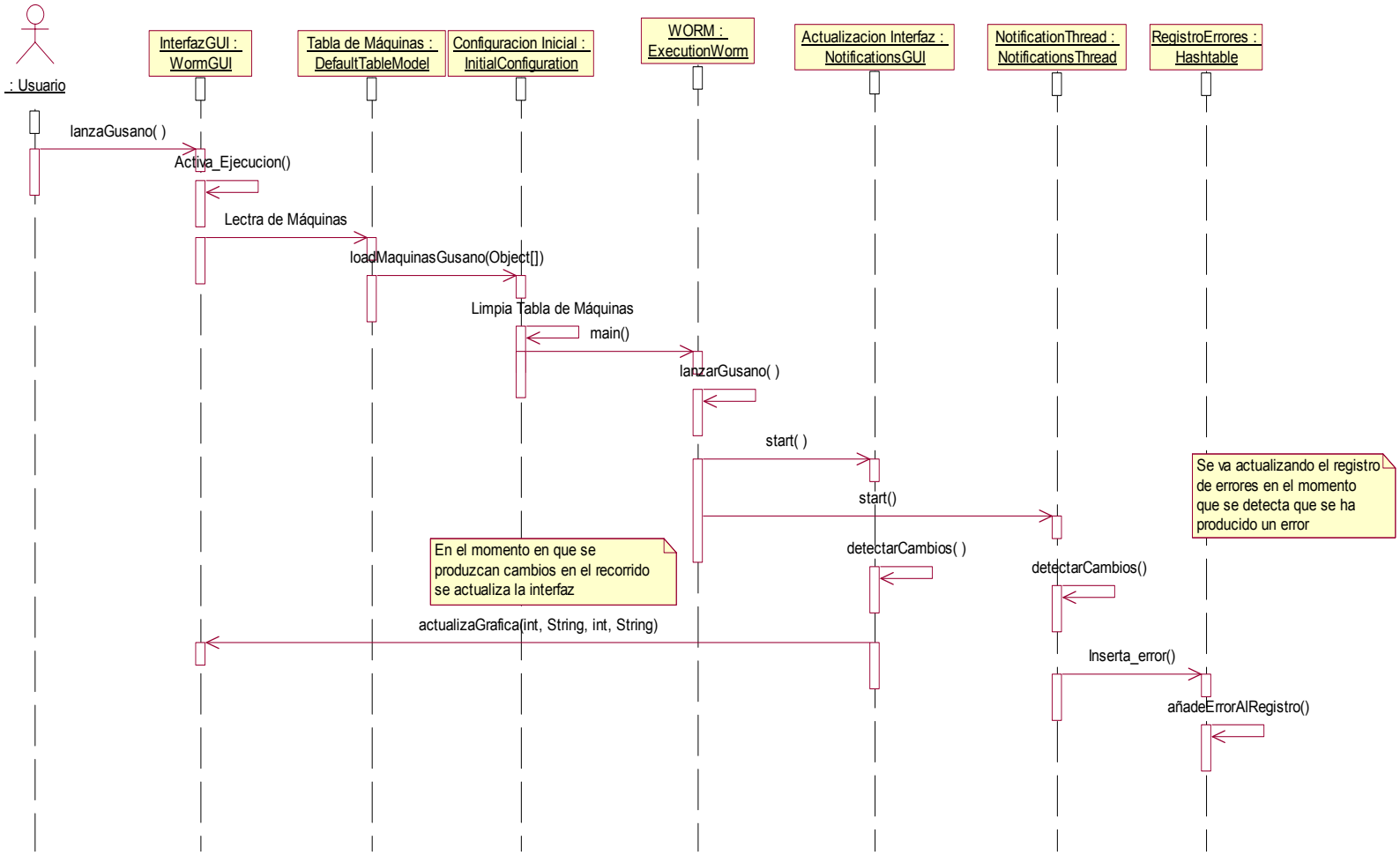
4. Secuencia Error: La máquina ya existe en el listado de máquinas disponibles



7.6.8. Diagrama de Secuencia: Ejecuta Worm "Lanzar gusano"

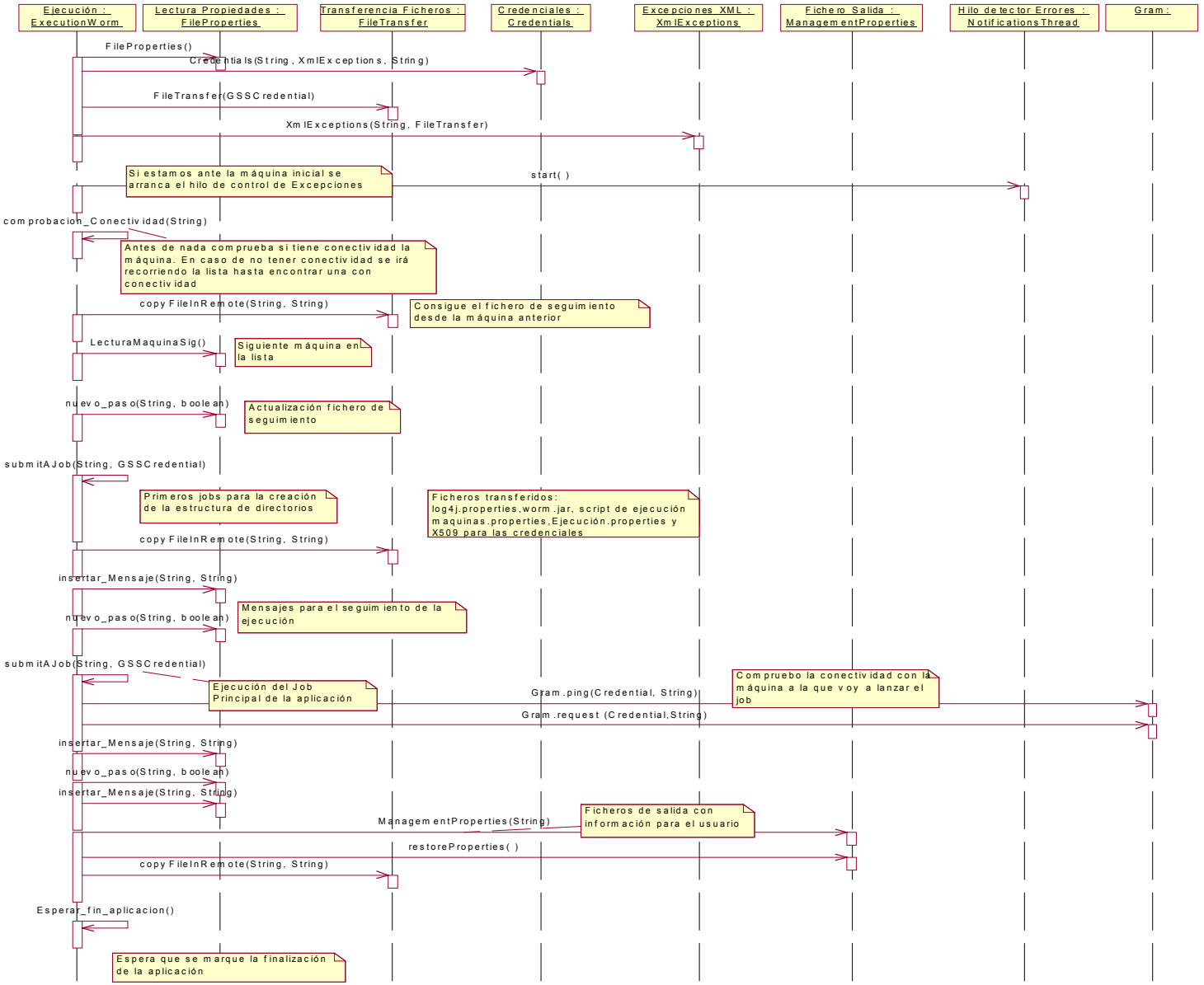
A continuación se muestra el diagrama de secuencia de una ejecución correcta del caso de uso que lanza a ejecutar el núcleo de la aplicación. Las situaciones de error que se puedan producir se determinaran más adelante con los diagramas de secuencia del otro módulo de la aplicación, es decir, los diagramas de secuencia de ejecutar el "Worm.jar".

1. Secuencia Correcta:

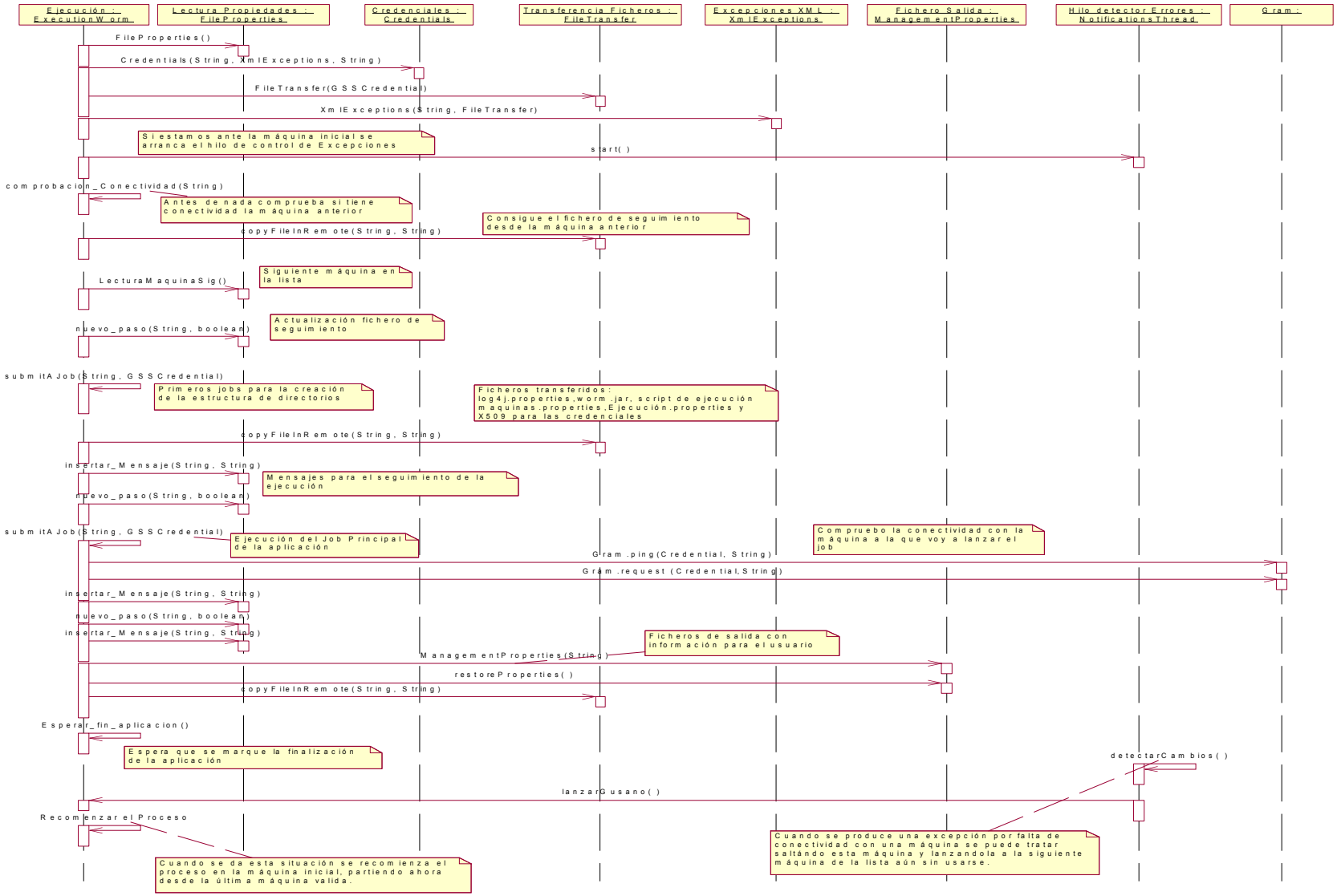


7.6.9. Diagrama Secuencia: WORM (Núcleo de ejecución)

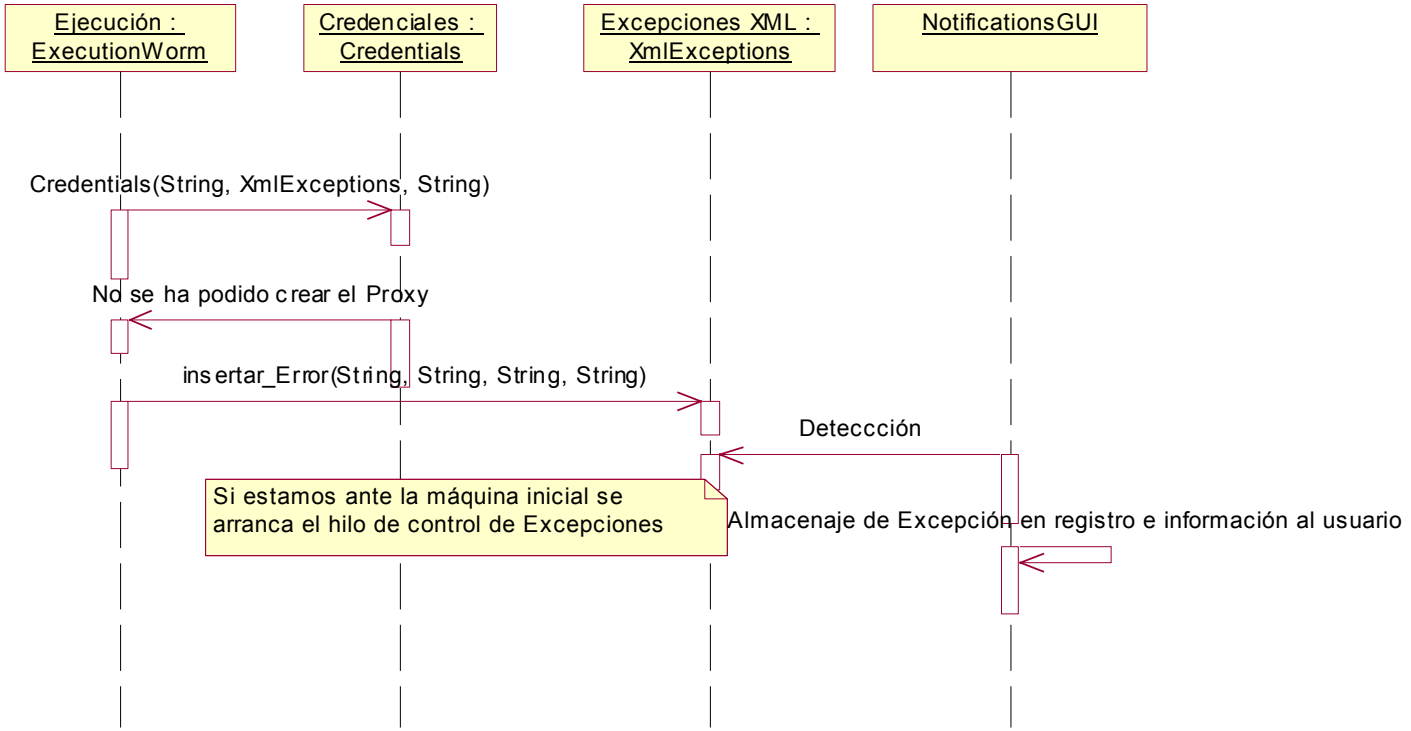
1. Flujo normal de Ejecución



2. Recuperación de la ejecución frente a los errores de conectividad con las distintas máquinas del recorrido.



3. Secuencia de Error: No se puede crear el Proxy



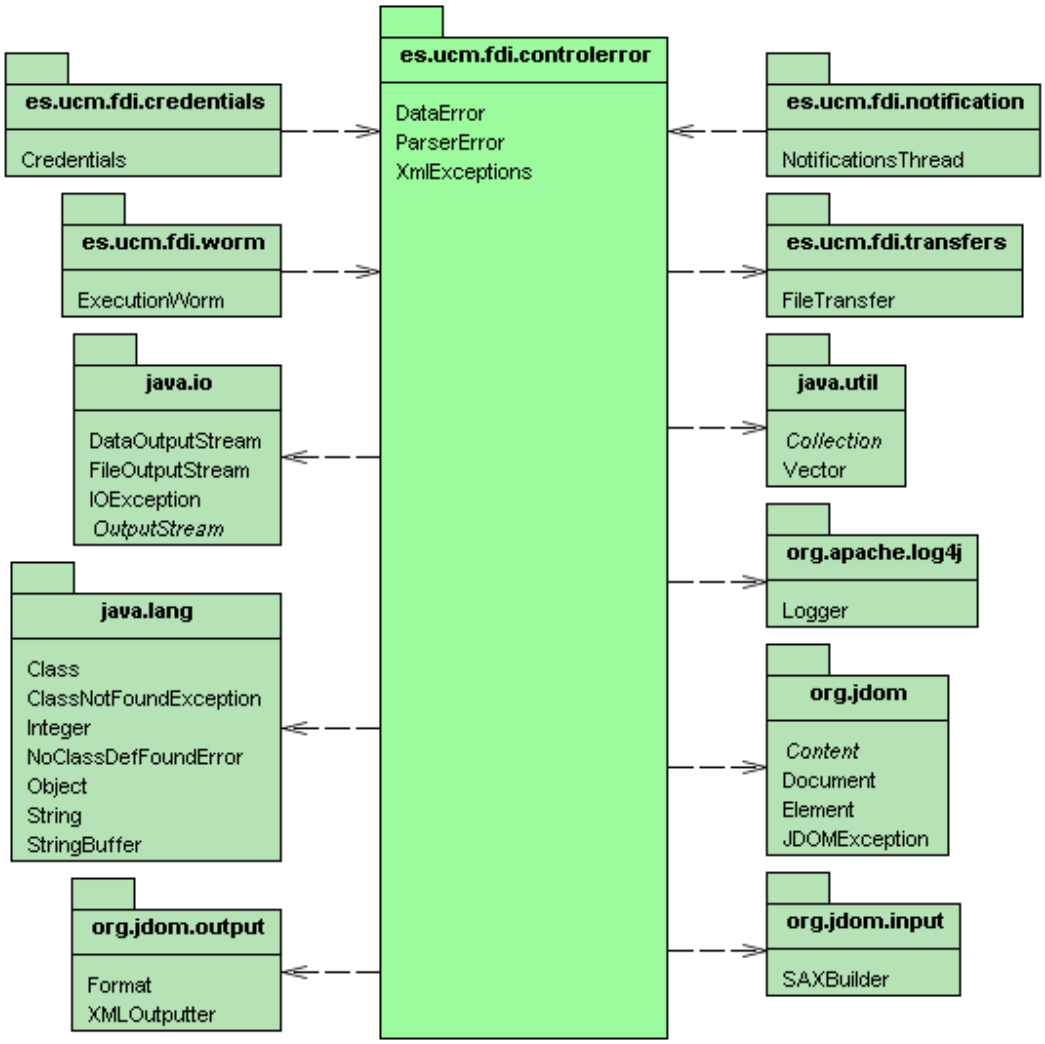
Existen otras situaciones de error que no se muestran mediante diagramas de Secuencia, que a grandes rasgos son:

1. Fallo en la lectura de los ficheros de configuración o de salida.
2. Fallo durante las transferencias de los ficheros.
3. Identificación incorrecta de la máquina del usuario. Este error es totalmente ajeno al usuario y en principio no tiene porqué darse pero es algo que se debe plantear.

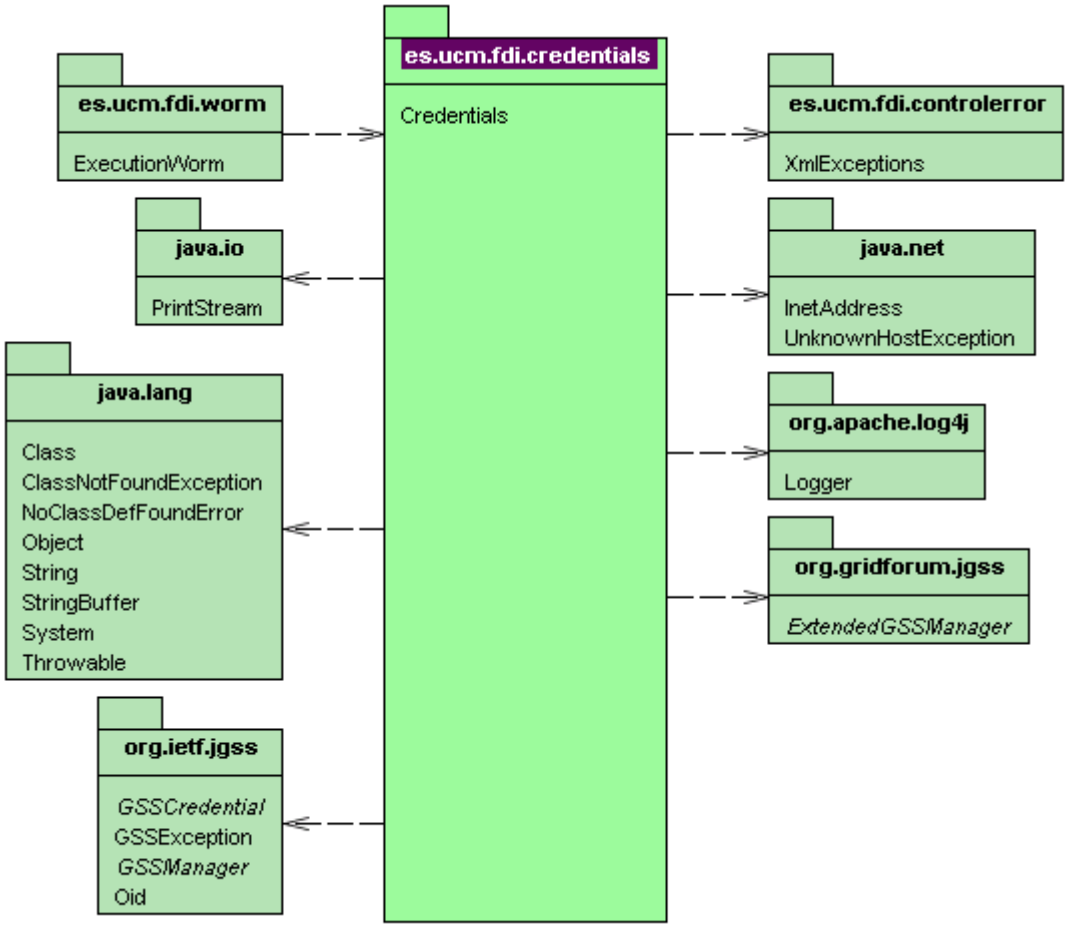
7.7. DIAGRAMAS DE PAQUETES

7.7.1. DIAGRAMAS PAQUETES WORM

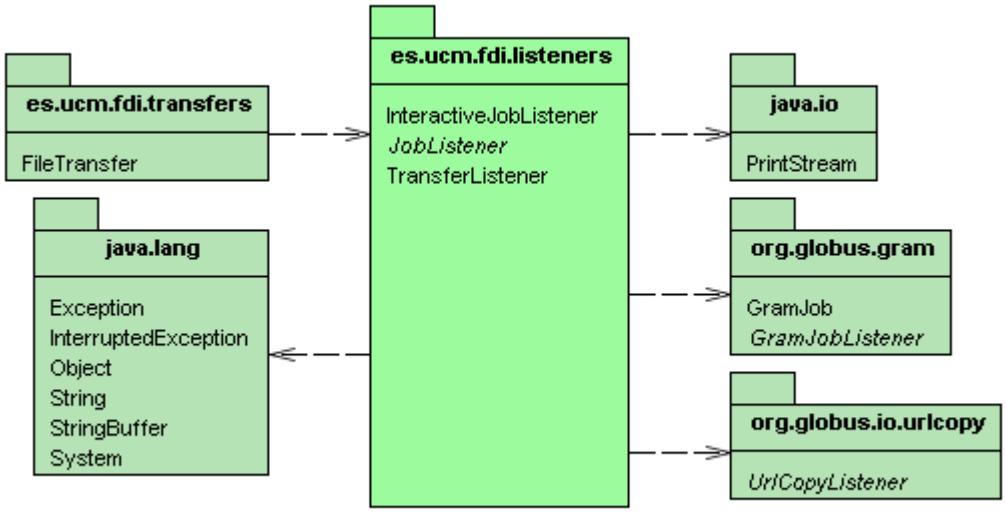
DIAGRAMA PAQUETE CONTROLERROR



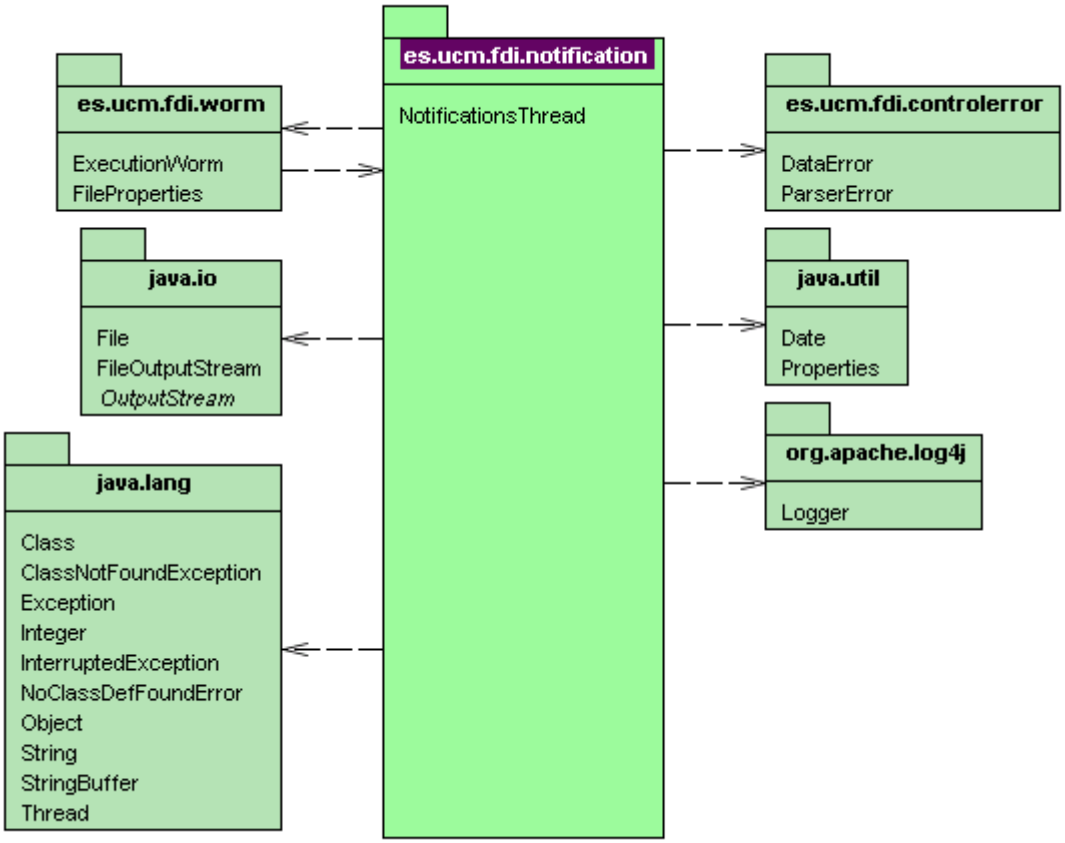
- **DIAGRAMA CREDENTIALS**



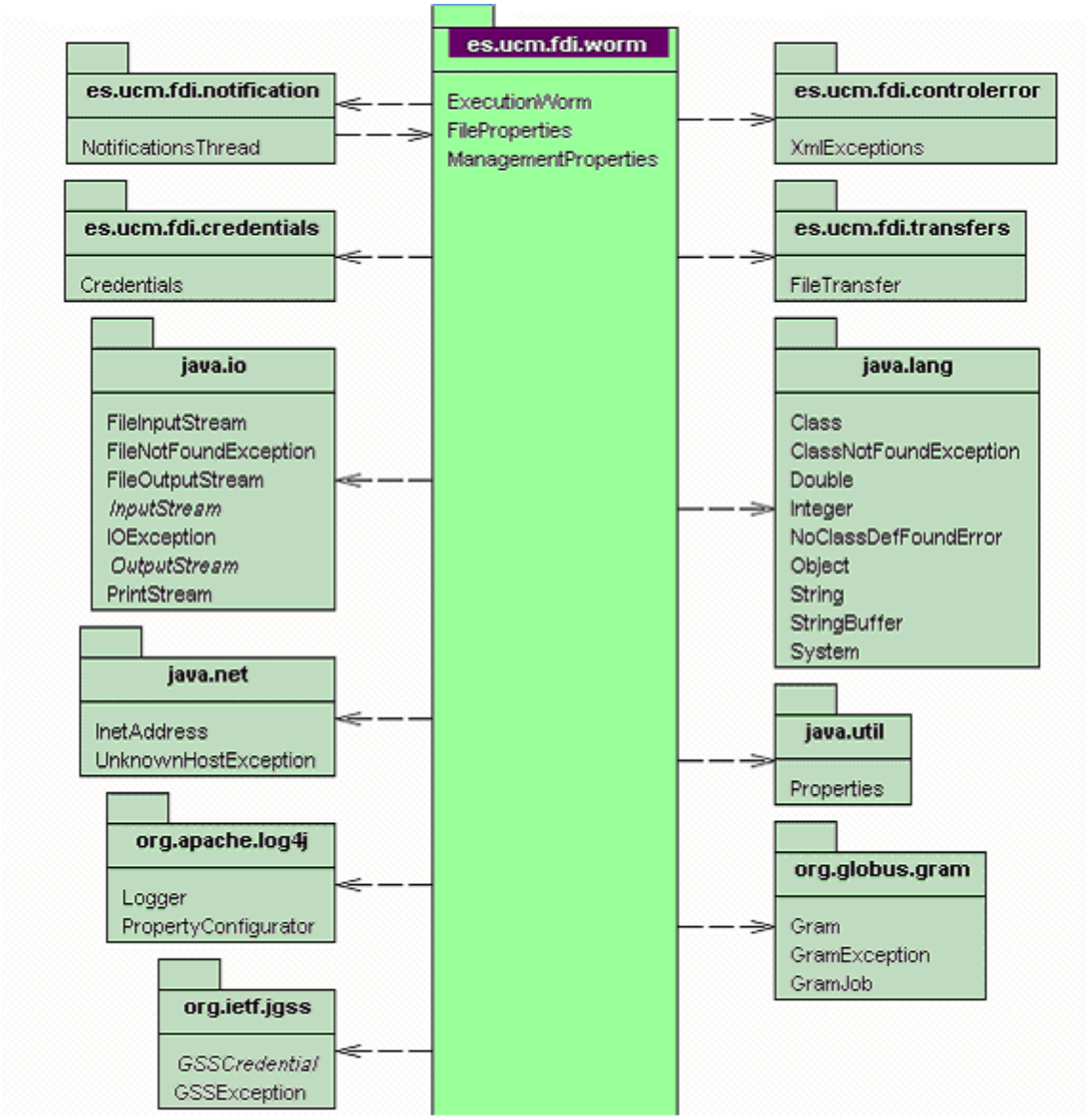
- **DIAGRAMA LISTENERS**



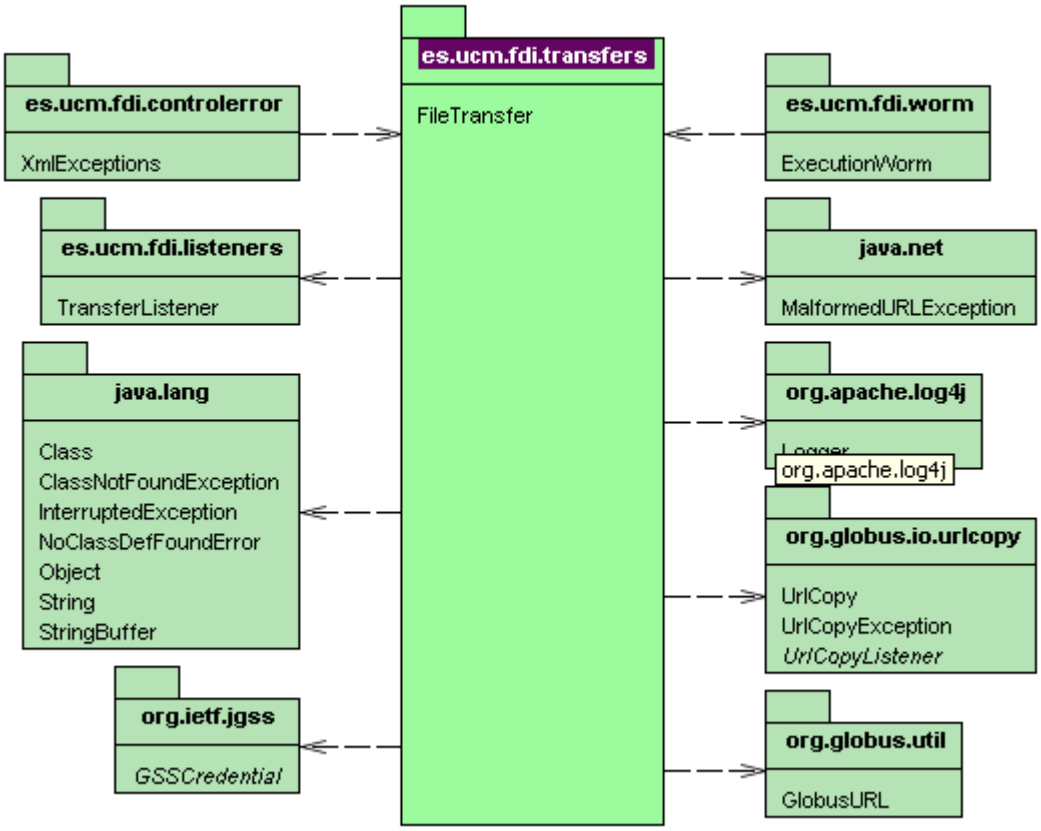
- **DIAGRAMA NOTIFICATIONS**



- **DIAGRAMA WORM**

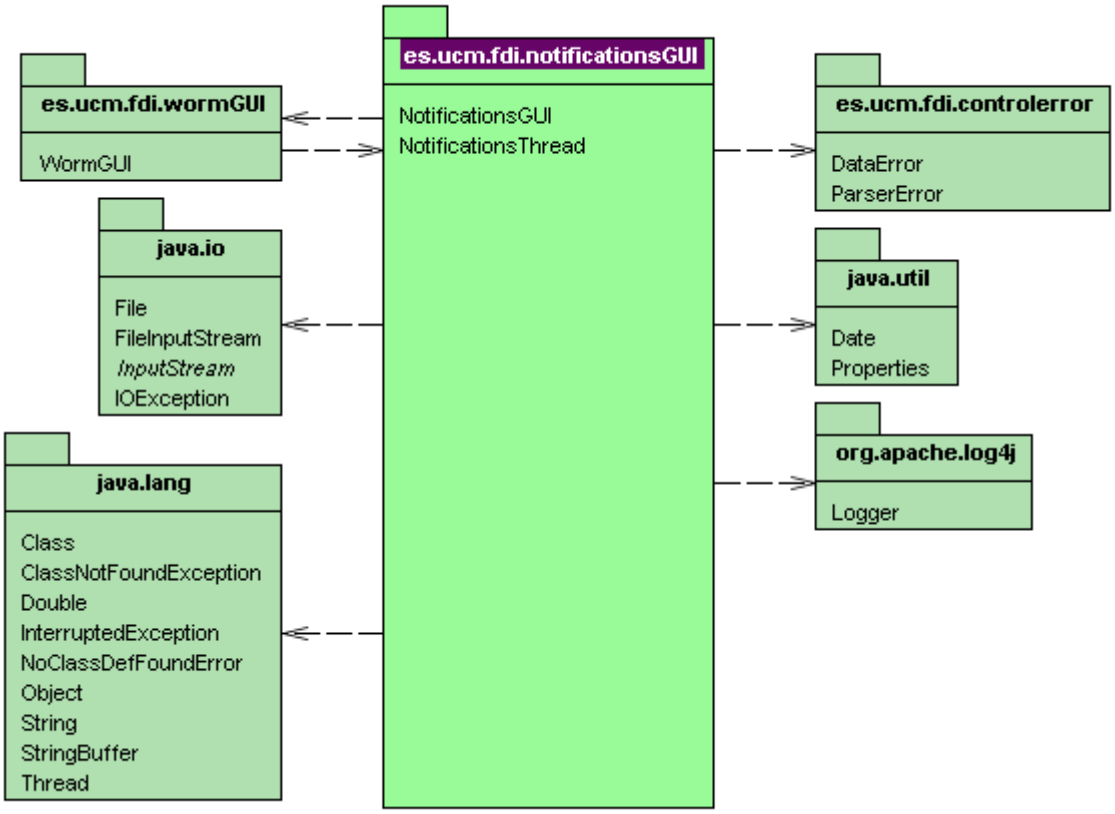


- **DIAGRAMA TRANSFER**

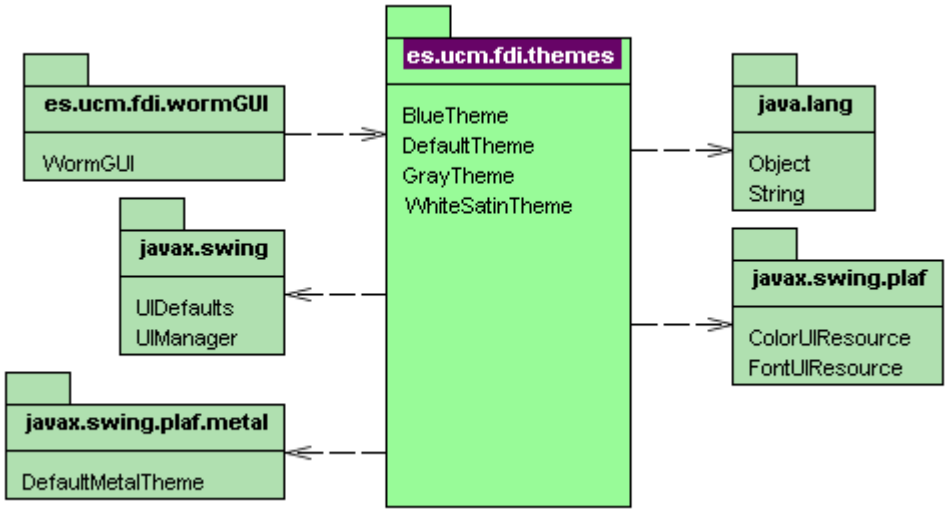


7.7.2. DIAGRAMAS PAQUETES WORMGUI

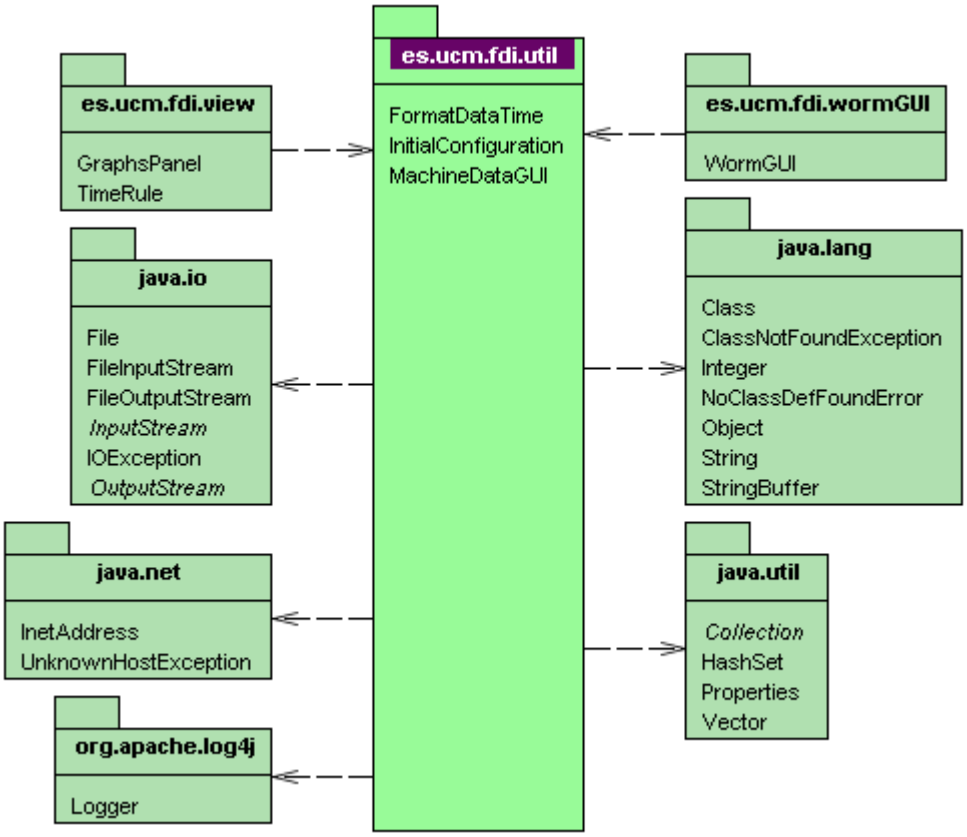
- **NotificationsGUI**



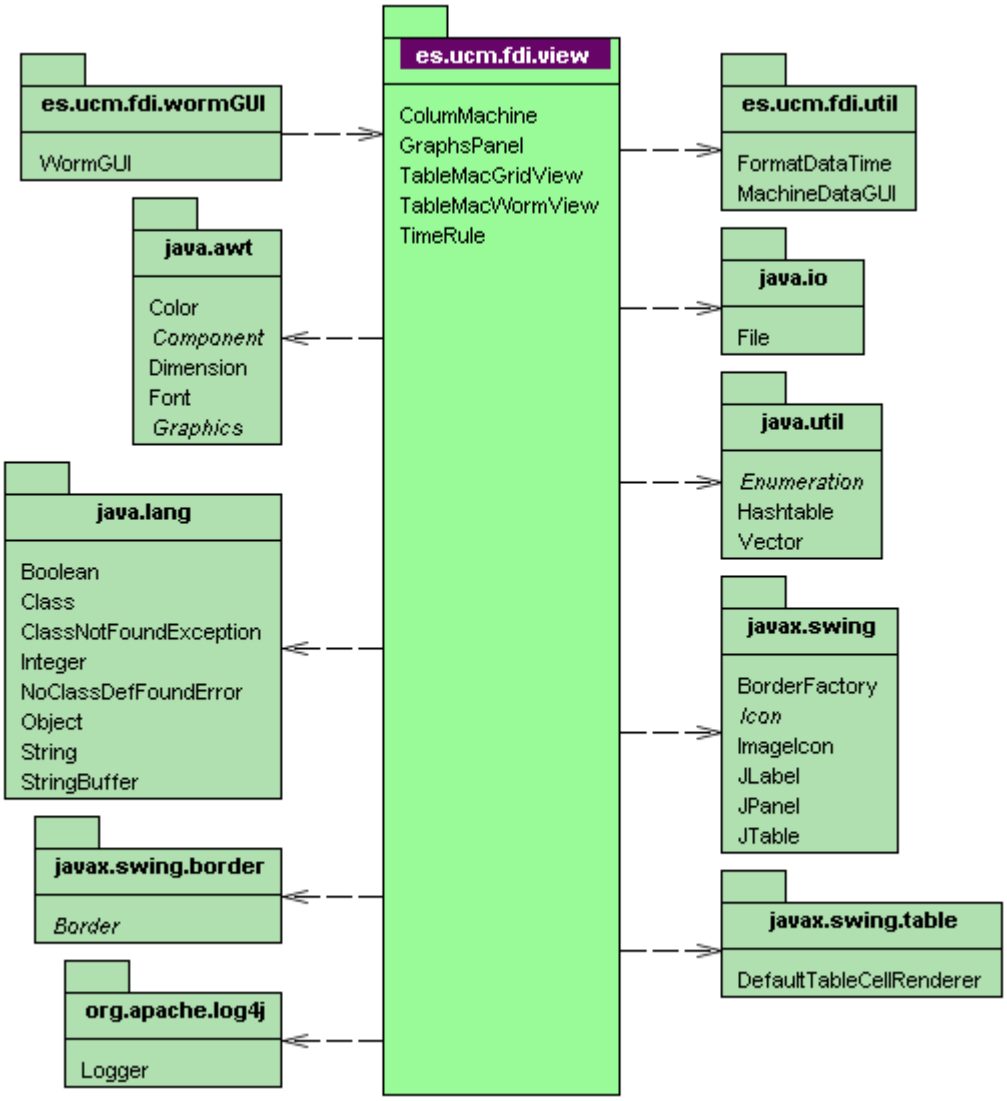
- **THEMES**



- **UTIL**


es.ucm.fdi.util
FormatDataTime
InitialConfiguration
MachineDataGUI
es.ucm.fdi.view
GraphsPanel
TimeRule
es.ucm.fdi.wormGUI
WormGUI
java.io
File
FileInputStream
FileOutputStream
InputStream
IOException
OutputStream
java.lang
Class
ClassNotFoundException
Integer
NoClassDefFoundError
Object
String
StringBuffer
java.net
InetAddress
UnknownHostException
java.util
Collection
HashSet
Properties
Vector
org.apache.log4j
Logger

- **VIEW**


es.ucm.fdi.wormGUI
WormGUI
es.ucm.fdi.view
ColumMachine
GraphsPanel
TableMacGridView
TableMacWormView
TimeRule
es.ucm.fdi.util
FormatDataTime
MachineDataGUI
java.awt
Color
Component
Dimension
Font
Graphics
java.io
File
java.util
Enumeration
Hashtable
Vector
java.lang
Boolean
Class
ClassNotFoundException
Integer
NoClassDefFoundError
Object
String
StringBuffer
javax.swing
BorderFactory
Icon
ImageIcon
JLabel
JPanel
JTable
javax.swing.border
Border
javax.swing.table
DefaultTableCellRenderer
org.apache.log4j
Logger

8. DESCRIPCIÓN DETALLADA DE ALGUNOS MÓDULOS DE INTERÉS

8.1. MÓDULO DE ERRORES

8.1.1. SITUACIONES DE ERROR DADA EN LA APLICACIÓN.

Clase Ejecutable.java

En esta clase se pueden dar diferentes situaciones de error, en algunos casos se podrá salvar la situación realizando una serie de operaciones alternativas, mientras que otras la gravedad de las mismas provocará la finalización del flujo de ejecución.

- Método GetCredentials:

Las situaciones de error que pueden darse consisten en excepciones de tipo GSSExcepcion como consecuencia de que el proxy no este creado, por lo tanto esta situación de error imposibilita la ejecución y producirá una entrada en el registro de errores que será mostrado al usuario mediante un mensaje para que la solvente creando el proxy y proceder a la ejecución.

- Método Comprobación_Conectividad:

Las situaciones de errores son completamente controladas ya que me permiten controlar determinadas situaciones y solventarlas. Las excepciones que puede provocar son: "GramException" y "GSSException". Se reportará el error correspondiente, para que sea almacenado en el registro de errores, y se procederá a solventar el error saltando la máquina sin conexión al GRID y marcándola como INCORRECTA para evitar futuras situaciones de error.

- Método Run:

En este método se pueden producir errores de lectura/escritura de los ficheros de propiedades. Esto es un error grave ya que no puede efectuar la lectura de ficheros esenciales en la ejecución y por lo tanto se procederá a terminar la ejecución.

Las excepciones que se pueden dar en este caso son: "IOException", "FileNotFoundException".

En este método se hace una llamada "(java.net.InetAddress.getLocalHost()). getHostName();" que me proporciona el nombre host. Si no se puede identificar este entonces se producirá una situación de error que será tratada obteniendo el nombre del host del fichero de propiedades que contiene el nombre de todas las máquinas por las que se ha ido pasando. De esta manera podemos salvar la situación sin necesidad de terminar completamente la ejecución de la aplicación. La excepción que se produce en este caso es un: "UnknownHostException". Se creará una entrada en el registro de errores para tener constancia de la misma.

- Método Función_Gusano:

Al igual que en el método descrito anteriormente se puede producir una excepción del tipo "UnknownHostException".

- Método Submitajob:

En este método se pueden dar situaciones de error debido a que haya algún Host que no tenga conectividad al GRID. En este caso se insertará una nueva entrada en el registro de errores, se indicará al usuario cual es la máquina que ha dado la situación de error y por

último se saltará a la máquina siguiente de la lista para proseguir con la ejecución de la aplicación. La máquina que provoca el error será marcada como "INCORRECTA".

En este caso las Excepciones que se pueden dar son "GRAMException" y "GSSException" que serán controladas.

- Método LanzarGusano:

Al igual que en el método descrito anteriormente se puede producir una excepción del tipo "UnknownHostException".

Clase FileProperties.java

- Método Lectura_MaquinaSig:

Se producen excepciones del tipo: "FileNotFoundException" e "IOException".

En este método se pueden producir errores de lectura/escritura de los ficheros de propiedades. Esto es un error grave ya que no puede efectuar la lectura de ficheros esenciales en la ejecución y por lo tanto se procederá a la finalización de la ejecución.

- Método Get_Properties:

Se producen excepciones del tipo: "FileNotFoundException" e "IOException".

En este método se pueden producir errores de lectura/escritura de los ficheros de propiedades. Esto es un error muy grave ya que no puede efectuar la lectura de ficheros esenciales en la ejecución y por lo tanto se procederá a la terminación de la ejecución.

- Método Nuevo_Paso:

Se producen excepciones del tipo: "FileNotFoundException" e "IOException".

En este método se pueden producir errores de lectura/escritura de los ficheros de propiedades. Esto es un error muy grave ya que no puede efectuar la lectura de ficheros esenciales en la ejecución y por lo tanto se procederá a la terminación de la ejecución.

- Método Insertar_Mensaje:

Se producen excepciones del tipo: "FileNotFoundException" e "IOException".

En este método se pueden producir errores de lectura/escritura de los ficheros de propiedades. Esto es un error muy grave ya que no puede efectuar la lectura de ficheros esenciales en la ejecución y por lo tanto se procederá a la terminación de la ejecución.

Clase ManagementProperties.java

- Método Restore_Properties:

Se producen excepciones del tipo: "FileNotFoundException" e "IOException".

En este método se pueden producir errores de lectura/escritura de los ficheros de propiedades. Esto es un error muy grave ya que no puede efectuar la lectura de ficheros esenciales en la ejecución y por lo tanto se procederá a la terminación de la ejecución.

Codificaciones de errores para el hilo de control

- *GSSCredentials*: Código error: 1 : Error en la obtención de credenciales.
- *GSSException*: Código error: 2: Error en la identificación de host del GRID para la conectividad.

- *GSSException*: Código error: 3: En el caso de que es imposible hacer un SUBMIT del job.
- *GramException*: Código error: 4: No submit a job.
- *UnknownHostException*: Código error: 5: Problemas al obtener el host para lecturas de propiedades.
- *FileNotFoundException*: Código error: 6: No se puede leer algún fichero necesario en la configuración.
- *IOException*: Código error: 7: Problemas en lectura y escritura sobre los ficheros de propiedades.

8.1.2. Control de errores

El control de errores consiste en un hilo que observa la creación o modificaciones de un fichero "Errores.xml" que contendrá todos los errores que se van a ir generando a lo largo de la ejecución de la aplicación. Este fichero será leído por el motor gráfico para mostrar los errores correspondientes al usuario e introducirlos en el registro de errores. Así el usuario obtendrá información de errores de ejecución en tiempo real.

Ese mismo fichero de errores será leído por el hilo de control de errores que solventará aquellas situaciones en las que el error no implique la terminación de la ejecución.

Ej: Si una máquina da un error de que no puede leer los ficheros, pues que del error y la parte de la aplicación encargada de esto recoja el error, vea que es y en que maquina se da y la ponga como incorrecta y siga por la lista para evitar que se paralice toda la ejecución de la aplicación.

Así se conseguiría un planificador que se comportase bien frente a los errores y se recupere de ellos.

8.2. MÓDULO DE INTERFAZ DE USUARIO

Para facilitar tanto la lectura de los resultados obtenidos por el núcleo de la aplicación "Worm" como facilitar al usuario el control total de ejecución y el seguimiento de la misma se planteó añadir un módulo de interfaz gráfica para tales efectos.

Las funcionalidades de la interfaz gráfica son las siguientes:

1. Creación del Proxy: Permite crear el proxy correspondiente a un usuario para permitir la ejecución del núcleo de la aplicación. Este será el primer paso ya que si no se crea el proxy se producirá un error que impedirá al usuario la correcta ejecución.

2. Inserción de nuevas máquinas que formen parte de la lista de máquinas disponibles para la configuración del recorrido de ejecución: En este caso permite incorporar nuevas máquinas que constituyan el recorrido, siempre y cuando se compruebe su conectividad al GRID evitando así posibles situaciones de error derivadas de la falta de conexión de las mismas. Si la nueva máquina ya esta incluida en la lista de disponibilidad se mostrará al usuario un mensaje informativo de la situación y no se incluirá la máquina en el listado.

3. Configuración de las máquinas al recorrido: Mediante esta opción permite añadir y eliminar las máquinas que conforman el recorrido de la aplicación.

4. Modificación orden recorrido: Permite modificar el orden de las maquinas en el recorrido debido a decisiones del usuario.

5. Visualización ficheros de Trazas: Permite al usuario monitorizar el flujo de ejecución, información relevante, así como comprobar el correcto funcionamiento y en caso de error comprobar que tipo de error se ha producido.

6. Visualización del registro de errores: Cuando se produce un error en el núcleo de la aplicación, se reporta hasta la máquina inicio o de control. Una vez recibido en la máquina de control se procede a insertarlo en un registro de errores que el usuario puede visualizar a través de la opción correspondiente.

7. Cambio de Apariencia: Permite crear un entorno más cómodo al usuario cambiando la apariencia del entorno gráfico.

8. Lanzamiento del núcleo de la aplicación: Este es el punto que incluye más funcionalidades, puesto que permite proceder al lanzamiento de la ejecución en las máquinas del GRID. Previo al lanzamiento crea todos los ficheros de propiedades necesarios para la ejecución y los inicializa a los datos precisados y que han sido configurados previamente. Una vez que se inicia la ejecución se crea un hilo que se encarga de ir mostrando los datos obtenidos durante la ejecución para que el usuario vaya observando el flujo. Una vez terminada la proceso el usuario tendrá acceso a los ficheros de log ya que estos no están disponibles hasta ese momento. En cuanto a los errores, la interfaz se encarga de ir mostrando los mensajes de error correspondiente, bien para informar de la situación de error al usuario (cuando no se paraliza la ejecución si el módulo de control de errores a solventado la situación) o mostrar la causa de la parada de la ejecución para que el usuario solvente la situación y vuelva a iniciar el proceso.

9. MANTENIMIENTO

9.1. ADAPTATIVO

Hay que asegurarse que todas las máquinas implicadas en la ejecución de la aplicación están dentro del Grid y tienen Globus Toolkit 4. Del mismo modo el usuario debe estar autorizado para poder ejecutar en estas máquinas.

Como el código fuente de la aplicación está programado en Java es preciso que todas las máquinas tengan la máquina virtual de Java.

El sistema operativo sobre el que funciona la aplicación es Linux.

9.2. PERFECTIVO

Sobre la aplicación hemos contemplado varias posibilidades de ampliación de las que vamos a destacar las siguientes:

- La aplicación es capaz de acreditarse en las máquinas para poder clonarse y ejecutarse pero podría implementarse la *delegación de credenciales*.

- La aplicación se lanza y se ejecuta a sí misma pero con leves ampliaciones podría ser capaz de ejecutar distintas tareas programadas en Java. De la misma forma que se clona y ejecuta a sí misma a través de las maquinas podría hacerlo con otras tareas.

- El módulo de notificaciones se organiza y gestiona a través del uso de threads, existe otra posibilidad que podría ser implementada basándose en *listeners*.

- La lista de máquinas que dibujan la ruta de ejecución es estática desde el momento en el que el usuario la establece en la interfaz gráfica. Esta lista podría obtenerse de forma dinámica ya que existe un comando a través del cual se pueden ver las máquinas que están conectadas en el Grid. El motivo por el que no hemos

implementado esta opción era no teníamos acceso a varias de ellas debido a los permisos.

- La aplicación antes de lanzarse a una nueva máquina comprueba que si es posible o no pero siempre a una máquina preestablecida por el orden de lista de máquinas. Una ampliación posible sería que entre las máquinas de la lista la aplicación pudiera determinar el estado de funcionamiento en el que se encuentran y decidir en base a sus propios criterios cuál es la máquina indicada para la próxima ejecución.

9.3. CORRECTIVO

Como mantenimiento correctivo de la aplicación se puede establecer el incremento de los tipos de error que es capaz de reconocer y tratar el gusano.

La aplicación es capaz de reconocer ciertos errores pero existirán errores que esta no pueda identificar. Por esto, una opción correctiva es la de ampliar este reconocimiento, y tratamiento en caso de ser posible, de los errores.

Otra parte incluida en este mantenimiento consiste en solventar posibles problemas que errores que puedan aparecen durante su uso.

9.4. PREVENTIVO

- El código se ha estructurado en varios módulos clasificados en base a la funcionalidad que realizan. Se ha diseñado e implementado de esta forma para que en caso se desee analizar, modificar y/o ampliar la aplicación sea mucho más fácil y accesible.La estructura de módulos proporciona mayor flexibilidad y legibilidad.

- Los comentarios de código de las clases para que quede mejor documentado establece dos ventajas:
  - Facilita el incluir nuevos cambios o ampliaciones por parte de los desarrolladores del equipo.
  - Facilita la comprensión del código en caso de otra persona ajena al grupo de proyecto desee analizarlo y/o reutilizarlo.

- Todos los miembros del grupo conocen con exactitud el desarrollo, motivo y funcionamiento de cada una de las implementaciones realizadas. Esto hace que en caso de cambio o ampliación en cualquier módulo de la aplicación pueda realizarla cualquier de los miembros sin miedo a un desconocimiento de ese módulo.

MANUAL DE USUARIO

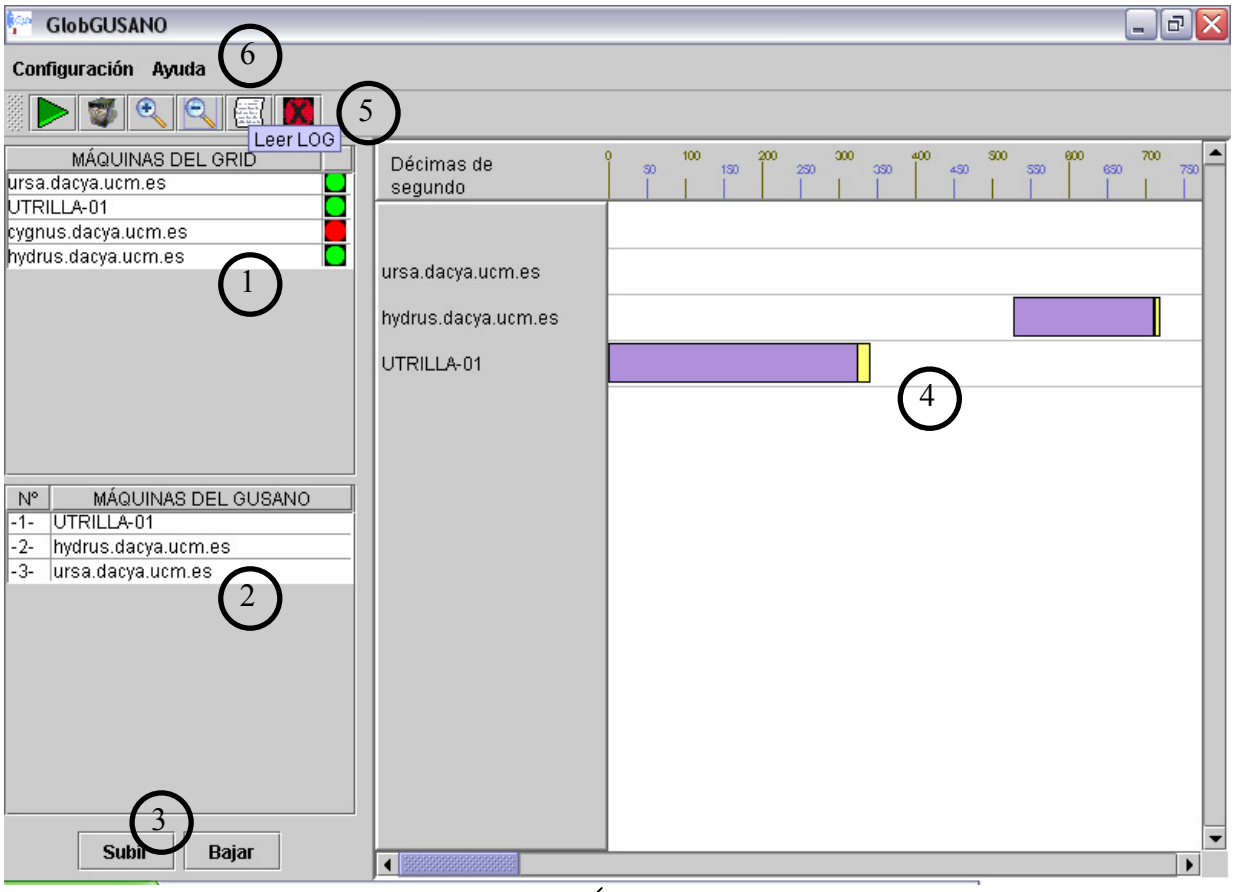
Guía rápida de la aplicación

1. Instalación de Worm Application

Worm Application no necesita instalación en su máquina.

- **Requisitos del sistema:**
  - SO Linux
  - Máquina perteneciente a una red GRID
  - Instalación de GlobusToolkit 4 en cada máquina del gusano

2. Esquema de Worm Application



1.- Lista de las máquinas del grid
2.- Lista de las máquinas del gusano
3.- Botones para cambiar el orden

4.- Gráfica de la evolución
5.- Menú rápido de iconos
6.- Menú desplegable

3. Las tablas de Worm Application

La ventana que aparece al arrancar Worm Application contiene dos tablas iniciales: la que muestra todas las máquinas existentes en el grid (1) y la que muestra el orden de las máquinas por las que pasará el gusano (2).

La tabla 1 recoge todos sus datos a partir del fichero **maquinasGrid.properties** y permanece invariable durante toda la ejecución de la aplicación. Si se desea añadir o cambiar esta tabla para otras ejecuciones, tan solo hay que modificar dicho fichero.

El gusano pasa por esta máquina

El gusano no pasa por esta máquina

MÁQUINAS DEL GRID	
ursa.dacya.ucm.es	
UTRILLA-01	
cygnus.dacya.ucm.es	
hydrus.dacya.ucm.es	

Formato maquinasGrid.properties

```
maq1=<nombre_maquina_1>
maq2=<nombre_maquina_2>
...
maqn=<nombre_maqunan_>
```

La tabla 2, al igual que la 1, recoge los datos del fichero **maquinas.properties**, pero en esta tabla se permiten modificaciones una vez lanzada la aplicación.

El número contenido en la primera columna indica el orden según el cual pasará el gusano. La máquina que siempre estará en la posición 1 será la máquina local1.

Formato de maquinas.properties.

```
maquina1=<nombre_maquina_1>
maquina2=<nombre_maquina_2>
...
maquinan=<nombre_maquna_n>
```

4. Añadir máquinas al gusano

Para añadir nuevas máquinas a la tabla **Máquinas del gusano** hay dos posibilidades:

- **Seleccionarla en la tabla máquinas del grid**

  Haz "**doble clic**" sobre la máquina de la tabla **Máquinas del grid** que deseas añadir.

  La máquina seleccionada pasará a la última posición libre de **Máquinas del gusano**.

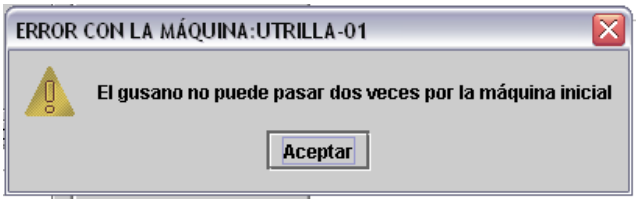
MÁQUINAS DEL GRID	
ursa.dacya.ucm.es	
UTRILLA-01	
cygnus.dacya.ucm.es	
hydrus.dacya.ucm.es	

Nº	MÁQUINAS DEL GUSANO
-1-	UTRILLA-01
-2-	hydrus.dacya.ucm.es
-3-	ursa.dacya.ucm.es
-4-	hydrus.dacya.ucm.es

1 Si en maquinas.properties maquina1 es distinta de la máquina local se ignora esta entrada del fichero.

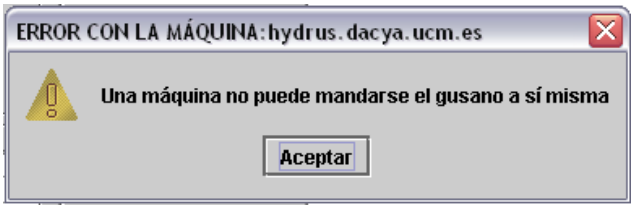
Hay restricciones a la hora de añadir máquinas.

- El gusano sólo puede pasar una vez por la máquina local.
- Si se intentara añadir *localhost* se mostraría el siguiente mensaje:



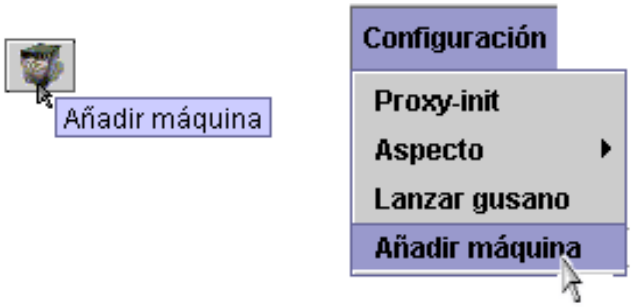
La siguiente máquina para el gusano no puede ser la máquina en la que está.

Al intentar añadir la misma máquina que está en la última posición de **Máquinas del Gusano** aparecerá el siguiente mensaje:

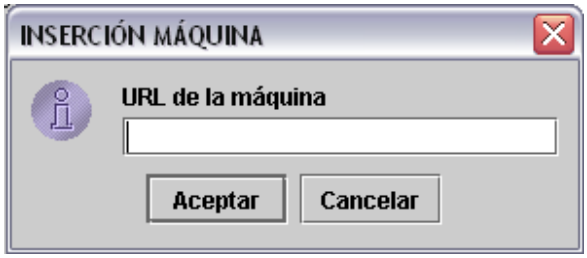


- **A través de la opción "Añadir máquina"**

Se puede añadir cualquier máquina, incluso si no aparece en la tabla de máquinas del Grid, pulsando sobre el botón del menú de los iconos o accediendo al menú Configuración->Añadir Máquina

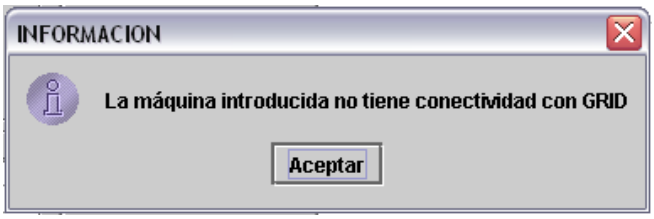


Con cualquiera de las dos opciones, nos aparecerá este cuadro de diálogo:



Si el nombre de la máquina es correcto, al aceptar el cuadro anterior, dicha máquina pasará inmediatamente a la última fila de la tabla "**Máquinas del gusano**".

Puede ser que intentemos añadir una máquina que no pertenece a la red Grid; en este caso, se muestra el mensaje:



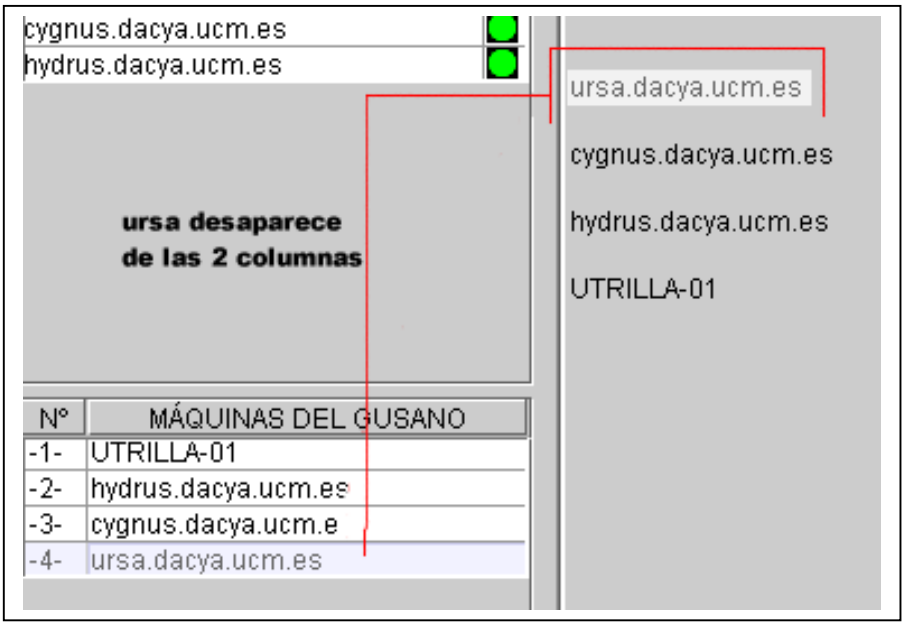
5. Eliminar máquinas del gusano

Si se desea eliminar alguna máquina por las que pasará el gusano basta con hacer "**doble clic**" sobre la que interese2.

En este punto, se distinguen dos casos:

Quedan más instancias de esa máquina: Sólo se elimina de la tabla "**Máquinas del gusano**".

Era la última instancia de la máquina: Como el gusano no pasará por esa máquina, se elimina de la columna de máquinas de la gráfica de evolución.



6. Cambiar el orden de las máquinas

Seleccionar las máquinas que se quieren para cambiar el orden3.

Pulsar sobre el botón **Subir** para adelantar una posición a todas las máquinas seleccionadas4.

Pulsar sobre el botón **Bajar** para retrasar una posición a todas las máquinas seleccionadas.

---

2 No se permite eliminar la primera máquina puesto que es la máquina local.

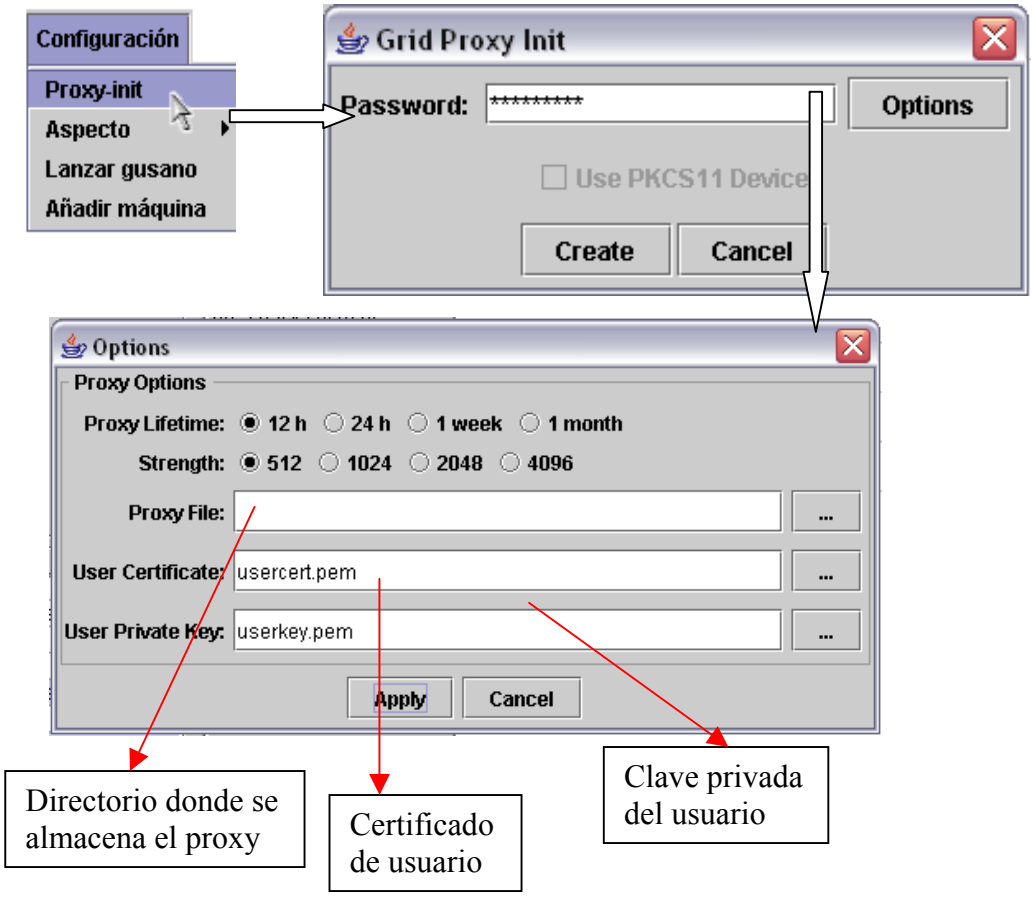
3 Se puede seleccionar un intervalo consecutivo de máquinas haciendo **click** sobre la inicial y la final y manteniendo pulsada la tecla **SHIFT**. Se puede seleccionar varias máquinas al azar haciendo **click** sobre ellas y manteniendo pulsada la tecla **Control**.

4 La máquina inicial (*localhost*) no puede ser movida.

7. Inicializar el proxy

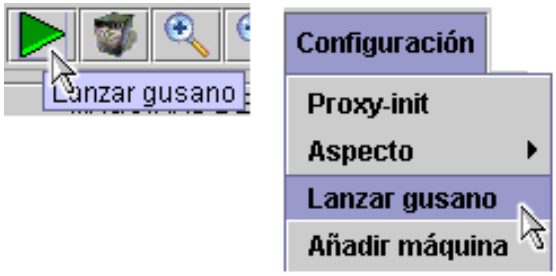
Globus, para mantener la seguridad e identificar a los usuarios del grid, obliga a crear un proxy antes de realizar cualquier otra tarea.

Desde nuestra aplicación se puede crear a través del menú **Configuración -> Proxy init**

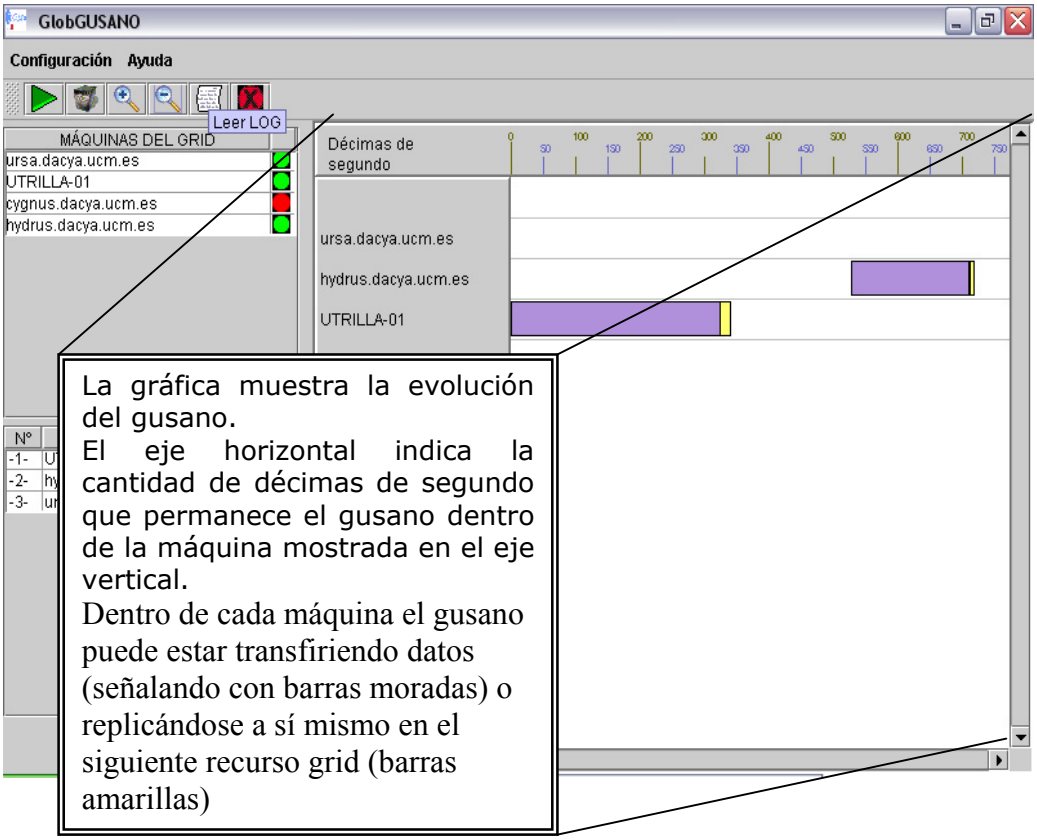


8. Lanzar el gusano

Una vez hayamos configurado todos los aspectos del gusano podremos lanzarlo a ejecución. Recuerde que no podrá funcionar si el proxy no está creado.



Al pulsar sobre una de estas dos opciones, empieza la ejecución del gusano por todas las máquinas.



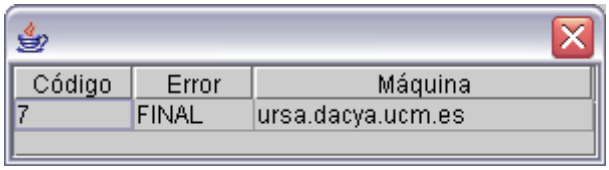
9. Ver errores producidos

Durante la ejecución, el gusano se puede encontrar problemas que desencadenen en un error, algunos de ellos irremediables.

Cada vez que se produce algún tipo de incidencia aparece una ventana emergente en la esquina inferior de sus monitor informándole de ello.

La estructura que tiene es:

- **Código:** Código asignado a ese error
- **Error:** Tipo del error producido
- **Máquina:** Máquina en la que se ha producido el error



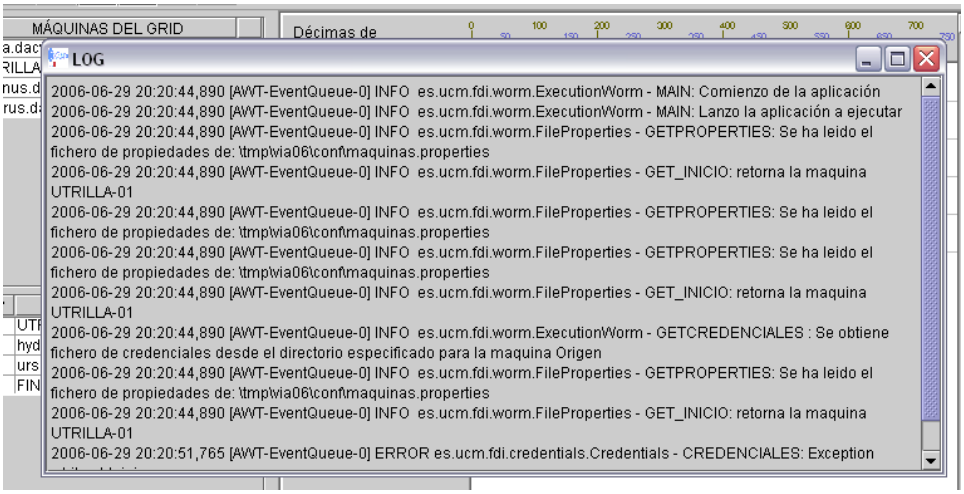
Esta ventana es temporalmente visible, con lo que después de unos segundos desaparece.

Para poder ver en cualquier momento todos los errores producidos a lo largo de la vida del gusano se puede pulsar sobre el botón  de la barra superior de iconos.

Nos aparecerá una tabla similar a la de la ventana emergente aunque en esta ocasión podrá contener más de un registro y permanecerá abierta mientras el usuario lo desee.

10. Ver fichero LOG

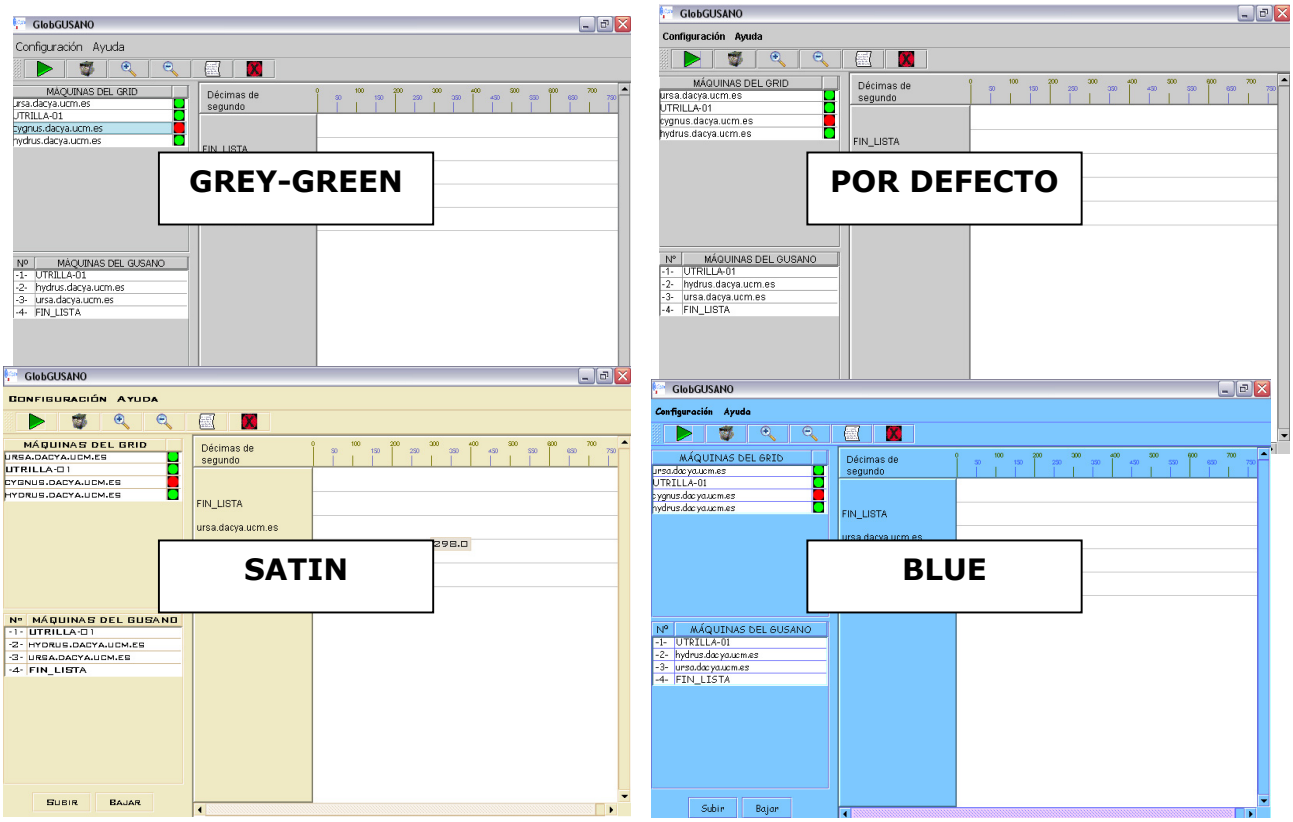
Una funcionalidad que tiene Worm Application es la de ir almacenando las trazas que genera el gusano para facilitar a posibles desarrolladores toda la información más.



Se puede acceder a esta información pulsando sobre el botón 

11. Cambiar el aspecto de la aplicación

En el menú **Configuración→ Aspecto** podemos elegir entre las siguientes ventanas.



BIBLIOGRAFÍA

- **Páginas Web**

www.globus.org
www.gridway.org
The Globus Toolkit 4 Programmer's Tutorial :
URL: http://gdp.globus.org/gt4-tutorial/multiplehtml/index.html

- **Libros**

Globus Toolkit 4 : programming Java services
Autor: Borja Sotomayor, Lisa Childers
Editorial: San Francisco : Morgan Kaufman, cop. 2006

- **Documentos y Artículos**

  - *Analisis de la Arquitectura de Globus Toolkit 4*
    Autor: José Luis Vázquez Poletti.
    Grupo de Arquitectura de Sistemas Distribuidos y Seguridad Dpto. Arquitectura de Computadores y Automática (DACYA) Universidad Complutense de Madrid

  - *Sun Grid Engine Sun Grid Engine en clusters bajo Linux*
    Autor: Isabel Campos Plasencia
    Grids y ECiencia: Santander(2004)

  - *Globus Middleware*
    Autor: Daniel Cano, Instituto de Física de Cantabria IFCA/CSIC
    Grids & E-Ciencia

  - *GRAM, RFT & Job Submission, Execution Management for GT4 Developers*
    Autores: Stuart Martin Argonne National Lab y Pawel Plaszczak gridwise

  - *Sistemas Grid basados en GT3: Seguridad en GT3*
    Autor: Borja Sotomayor (2004)

GLOSARIO DE TÉRMINOS

APÉNDICE

A continuación se mostrará la estructura de los diferentes ficheros que proporcionan soporte para la ejecución de la aplicación "WORM".

Build.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<project name="globusVIA" default="init" basedir=".">
<property file="build.properties" />

<path id="classpath">
   <fileset dir="${project.lib}" includes="**/*.jar"/>
   <fileset dir="${project.home}" includes="*gusano.jar"/>
   <fileset dir="${project.lib2}" includes="*jdom.jar"/>
</path>


<target name="jar">
        <jar destfile="gusano.jar" >
            <manifest>
                <attribute name="Main-Class" value="worm.ExecutionWorm"/>
            </manifest>
        </jar>
    </target>

<target name="ejecutar" description="ejecución de la aplicación">
   <echo>***Ejecución de la aplicación***</echo>
   <java fork="true" classname="${project.Main}" >
          <classpath>
                <path refid="classpath"/>
                <path location="${project.home}/gusano.jar"/>
          </classpath>
   </java>
</target>
<target name="borrar" description="borrar archivos temporales">
          <echo>***Se borran los archivos una vez ejecutados***</echo>
                <delete file="/lib/ant.jar"/>
                <delete file="${project.delete}/build.properties"/>
                <delete file="${project.delete}/build.xml"/>
                <delete file="${project.delete}/gusano.jar"/>
                <delete file="${project.delete}/info.out"/>
                <delete file="${project.delete}/trazas.log"/>
                <delete dir="${project.delete}/conf"/>
          <!--<delete includeemptydirs="true">
                <fileset dir="${project.delete}/"/>
          </delete>-->
</target>


<target name="init">
   <antcall target="ejecutar"/>
   <echo>***Proceso completado***</echo>
</target>
</project>
```

Build.properties

```
project.home=/tmp/via06
project.version=1.0
project.lib=/usr/local/gt4.0.0/lib
project.lib2=/home/via06/via06
project.delete=/tmp/via06
## Clase principal que se ejecutará
project.Main=es.ucm.fdi.worm.ExecutionWorm
```

Ejecutable.properties

```
dirRaiz=&(executable=/bin/mkdir)(arguments="//tmp/via06")
dirConf=&(executable=/bin/mkdir)(arguments="//tmp/via06/conf")
gusano=&(executable=/usr/local/ant1.6.2/bin/ant)(arguments="-buildfile"
    "/tmp/via06/build.xml")(environment= (JAVA_HOME
    /usr/local/j2sdk1.4.2))(stdout=/tmp/via06/info.out)
```

log4j.properties

```
#Propiedades del traceado
#Sun May 28 18:14:18 CEST 2006
log4j.appender.R.Append=false
log4j.rootLogger=info, R
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.File=trazas.log
log4j.appender.R.MaxFileSize=100KB
log4j.appender.R.layout=org.apache.log4j.PatternLayout
```

maquinas.properties

```
#Máquinas del gusano
#Sun May 28 18:14:18 CEST 2006
maquina3=ursa.dacya.ucm.es
maquina2=hydrus.dacya.ucm.es
maquina1=UTRILLA-01
indice=0
```

salidas.properties

```
#Maquinas visitadas
#Sat May 20 17:52:25 CEST 2006
indice_salida=6
maquina_salida5=hydrus.dacya.ucm.es
maquina_salida4=cygnus.dacya.ucm.es
maquina_salida3=ursa.dacya.ucm.es
maquina_salida2=hydrus.dacya.ucm.es
maquina_salida1=cygnus.dacya.ucm.es
```

seguimiento.properties

```
#Monitorización del gusano
#Sun May 28 18:27:31 CEST 2006
#Propiedades
#Sat May 20 17:52:20 CEST 2006
paso=6
TiempoFase5=543.0
TiempoFase4=654.0
TiempoFase3=18323.0
TiempoFase2=1710.0
TiempoFase1=32562.0
TiempoIni5=1.148127456986E12
TiempoIni4=1.148127449968E12
TiempoIni3=1.148127431642E12
TiempoIni2=1.148127411293E12
TiempoIni1=1.148127378729E12
paso5=Copiando
paso4=Ejecutando
paso3=Copiando
paso2=Ejecutando
paso1=Copiando
maquina5=ursa.dacya.ucm.es
maquina4=hydrus.dacya.ucm.es
maquina3=hydrus.dacya.ucm.es
maquina2=cygnus.dacya.ucm.es
maquina1=cygnus.dacya.ucm.es
```