UNIVERSIDAD DE DEUSTO

FACULTAD DE INGENIERÍA

TESIS DOCTORAL

NUEVAS CAPACIDADES EN EL DESCUBRIMIENTO DE RECURSOS *GRID* BASADAS EN TECNOLOGÍAS SEMÁNTICAS

Presentada por David Buján Carballal
dentro del Programa de Doctorado en Ciencias de la Computación
Dirigida por el Dr. Óscar Corcho García y el Dr. Josuka Díaz Labrador

Los Directores El Doctorando

BILBAO, SEPTIEMBRE DE 2015

A mi familia y amigos

Resumen

Uno de los problemas abiertos en el contexto de las arquitecturas orientadas a servicios, como es el caso de la *grid*, es el descubrimiento de recursos y/o servicios adecuados para llevar a cabo una tarea determinada. Los proveedores de información *grid* normalmente ofrecen información funcional sobre los recursos que monitorizan, por lo que los modelos de información básicamente representan esta información sintáctica, y los consumidores de información usan normalmente dichas propiedades funcionales para seleccionar recursos. En la práctica, muchos trabajos se reinician debido a fallos en los recursos, aunque existen iniciativas que tratan de usar técnicas aisladas para manejar algunas propiedades no funcionales o de calidad de servicio.

En la presente tesis se proponen nuevas capacidades en el modelado de recursos *grid* relacionadas con sus propiedades no funcionales para que sean utilizadas tanto por proveedores como consumidores de información en la *grid*, de manera que sea un punto de partida para mejorar tanto las prestaciones de la infraestructura como las de los trabajos que se ejecutan sobre la misma.

Para ello, en primer lugar, se ha desarrollado un modelo basado en tecnologías semánticas para integrar los modelos existentes tanto a nivel de representación de información *grid* como de propiedades no funcionales en general, con el objetivo de enriquecer la información existente sobre cada uno de los recursos de diferente nivel existentes en los nodos de la *grid*. En segundo lugar, también se propone una metodología para la representación de propiedades no funcionales en el dominio *grid* que pueda ser utilizada por los administradores de los diferentes nodos para enriquecer la definición de sus recursos y mejorar la experiencia de sus usuarios respecto a la ejecución de sus trabajos. En tercer lugar, se aplica dicha metodología en la representación de dos propiedades no funcionales a nivel de recurso de computación (disponibilidad y fiabilidad) y una a nivel de un nodo *grid* (fiabilidad). Y en cuarto lugar, se ofrece un

conjunto de algoritmos modificados para incluir las propiedades no funcionales a nivel de recurso de computación en el descubrimiento y selección de los mismos para un modelo de pruebas dentro de un entorno de simulación de un nodo *grid*, de tal manera que permita comprobar los resultados que se obtienen mediante la utilización combinada o no de dichas propiedades no funcionales en el descubrimiento y selección de los recursos.

Abstract

An open problem within the domain of service oriented architectures, such as grid infrastructures, is the discovery of resources and/or services suitable for the execution of a specific task. Grid information producers generally provide functional information on the resources they monitor. Therefore information models represent this syntactic information whereas information consumers use such functional properties in order to select resources. In practice many jobs are generally restarted due to errors in the allocated resources. However various initiatives which try to use ad-hoc techniques in order to manipulate non-functional properties have been proposed.

This thesis proposes a new approach in the modelling of non-functional properties of grid resources, so that non-functional properties can be used both by the producers and the consumers of grid-based information in order to improve both the performance of the infrastructure layer and the performance of the jobs executed on such infrastructure.

Various innovations have been required regarding this new approach. In the first place, a model based on semantic technologies has been developed in order to integrate both the existing models focused on representation of grid-based information and those focused on non-functional properties, with the aim of enriching the existing information on every resource at every level present on each grid node. Secondly, a new methodology for the representation of non-functional properties in the grid domain has been proposed so that it can be used by the administrators of the various nodes in order to enrich the definition of their resources and improve the user experience with respect to the execution of their jobs. Thirdly, such methodology has been applied to the representation of three non-functional properties: two at computational resource level (availability and reliability) and one at grid node level (reliability). Finally, a set of algorithms modified by including non-functional properties at computational resource level which allows the discovery and

selection of such resources within the context of a testing model for a grid simulation tool has been developed. This set of modified algorithms allows the comparison of the results obtained by using such non-functional properties in a stand-alone or combined fashion during the discovery or selection of resources.

Agradecimientos

En primer lugar, quiero agradecer a mi familia y amigos el apoyo y amor incondicional que me ofrecen sin esperar nada a cambio. Mis hijos y mi mujer, junto a mis padres y mi hermana, son la razón por la que no he desfallecido en el largo camino que he recorrido para llegar a presentar mi tesis y el motivo por el que seguiré adelante con mi labor de investigación. Mateo, Alba, Rebeca, Mamá, Papá, Ana... Quiero agradeceros que hayáis estado siempre a mi lado en este viaje, animándome en todo momento y dándome la fuerza necesaria para avanzar y no quedarme en el camino. Quiero pediros perdón por todo el tiempo que no he pasado con vosotros y por cualquier efecto secundario que haya provocado esta tesis en nuestra relación. Prometo devolveros con creces todas las inversiones que habéis hecho en mí para que haya podido llegar hasta donde estoy y para que pueda seguir desarrollándome profesionalmente. En cualquier caso, nunca os lo agradeceré lo suficiente. En segundo lugar, quiero agradecer a mis codirectores de tesis, al Dr. Óscar Corcho García y al Dr. Josuka Díaz Labrador, que hayan querido acompañarme y guiarme todo este tiempo a pesar de las circunstancias personales y profesionales que han ido condicionando el desarrollo de mi tesis y que han hecho que se demorara más de lo deseable e incluso lo aceptable. Aun así, vuestro trabajo y esfuerzo han sido claves para llegar a este punto. En tercer lugar, gracias a todos mis compañeros de trabajo en la Universidad de Deusto y en DeustoTech, tanto los actuales como los antiguos. Gracias Ana, Iñaki, Asier, Pablo, Diego y tantos otros compañeros. Gracias grupo de investigación DELi por iniciarme en el noble oficio de la investigación. Gracias grupo de investigación MORElab por acogerme y ayudarme a desarrollarme como investigador. Gracias Diego por dejarte la piel por este grupo y mostrarnos el camino de la excelencia. He disfrutado y disfruto de los conocimientos que compartís conmigo y mucho más de vuestra compañía. Por último, pero no por ello menos importante, gracias a todos los investigadores con los que me he relacionado y en cuyos trabajos me he apoyado para llevar a cabo la presente tesis. Gracias al Dr. Luis

Manuel Vilches por su ayuda con la metodología, y al grupo de investigación IMG de la Universidad de Manchester por su propuesta S-OGSA y por acogerme para colaborar en el proyecto OntoGrid. Gracias a Erik Torres y Katia Leal por su esmero en preparar buenos estados del arte para otros investigadores. Gracias Roberto por tu ayuda con las traducciones. Y gracias a todas esas personas que no he nombrado, pero que forman parte de mi vida. Gracias a todos y todas por vuestro apoyo.

Tabla de contenidos

Índice de tablas

Índice de figuras

Índice de programas

Cuando te encuentres de camino a Ítaca,
desea que sea largo el camino, lleno de
aventuras, lleno de conocimientos.

Constantino Cavafis

Capítulo 1

Introducción

En este capítulo se presenta una visión general del contexto en el que se enmarca esta tesis, la motivación para su realización, los objetivos que se persiguen con la misma, las asunciones, hipótesis y restricciones que guían su desarrollo, así como sus principales aportaciones. Finalmente, se proporciona al lector una visión general de la estructura de esta memoria junto con un breve resumen del contenido de cada uno de sus capítulos.

1.1 Contexto

En este apartado se describen brevemente las áreas generales de conocimiento relacionadas con la presente tesis: el paradigma de computación *grid*, las propiedades no funcionales o de calidad en arquitecturas orientadas a servicio y la *grid* semántica.

1.1.1 Computación grid

La computación *grid* es un paradigma de computación que permite utilizar de forma coordinada distintos tipos de recursos que no están sujetos a un control centralizado, como son los tradicionales recursos de computación y de almacenamiento de cualquier sistema de computación distribuido y otros tipos de *hardware* y *software* específicos que puedan estar expuestos a través de una serie de servicios para su utilización. Por ello, se trata de un paradigma de computación distribuida en el que los recursos suelen ser heterogéneos

(supercomputadores, clústers, etcétera) y con diferentes arquitecturas, todos ellos conectados generalmente vía Internet.

La idea original bajo el término de "la *grid*" nació en ámbitos científicos a principios de 1990, haciendo referencia a una infraestructura que permitiese la integración y el uso coordinado de bases de datos, recursos de computación de alto rendimiento y de almacenamiento pertenecientes a diferentes propietarios y administrados por diferentes instituciones, tales como universidades, centros y laboratorios de investigación, e incluso algunas empresas.

Estas entidades se asocian para crear *grids* que les permitan compartir algunos recursos de sus diferentes dominios administrativos, utilizando para ello algún tipo de *middleware* o *software* específico que implemente este concepto y que posibilite la configuración de diferentes *organizaciones virtuales* (*Virtual Organizations* o VO).

La idea de "la *grid*" desarrollada por Foster y Kesselman (1999a) estaba inspirada por la analogía con las infraestructuras de distribución energética, especialmente la red eléctrica, donde la ubicación y procedencia de las fuentes de energía es completamente irrelevante para el consumidor. De esta forma, la primera idea de *grid* se dio a conocer a través de una visión de computación bajo demanda donde "la energía computacional" se convertiría en un recurso básico comercializable.

Sin embargo, la computación en *grid* se ha ido alejando de esta noción inicial y enfocándose hacia un modelo orientado a utilizar estándares de servicios web. Mediante estas tecnologías se accede a recursos computacionales de almacenamiento y cálculo, así como a otros dispositivos más especializados (sensores e instrumentos científicos), en forma de servicios *grid*.

Esta nueva noción de *grid* como plataforma de recursos virtuales en forma de servicios sobre Internet se apoya en una "arquitectura orientada a servicios" (*Service Oriented Architecture* o SOA), así como en los estándares y tecnologías ampliamente utilizados en los servicios web, como el *Protocolo Simple de Acceso a Objetos* (*Simple Object Access Protocol* o SOAP), el *Lenguaje de Descripción de Servicios Web* (*Web Services Description Language* o WSDL), y el protocolo de *Descripción, Descubrimiento e Integración Universal* (*Universal Description Discovery and Integration* o UDDI).

La comunidad de usuarios, desarrolladores y fabricantes que promueve el desarrollo de la computación distribuida, especialmente en el campo de la computación *grid*, es el *Open Grid Forum* (OGF), fundado tras la fusión del

Global Grid Forum y la *Enterprise Grid Alliance*. El OGF tiene dos funciones principales que se centran en el desarrollo de estándares para la computación *grid* y el desarrollo de tres comunidades diferentes dentro de la comunidad *grid*: grupos de trabajo para la elaboración de estándares, grupos de investigación para el desarrollo de casos de uso y grupos de usuarios de la comunidad *grid*.

El trabajo del OGF generó una familia de especificaciones para la definición y construcción de la *Arquitectura Abierta de Servicios Grid* (*Open Grid Services Architecture* u OGSA), la *Infraestructura Abierta de Servicios Grid* (*Open Grid Services Infrastructure* u OGSI), el *Marco de Recursos de Servicios Web* (*Web Services Resource Framework* o WSRF, también válido para los servicios *grid*), el *Lenguaje de Descripción de Envío de Trabajos* (*Job Submission Description Language* o JSDL) y otros. Estos grupos de especificaciones extienden a los servicios web tradicionales con una definición estándar para nuevas características, como el estado y el tiempo de vida, haciéndolos más apropiados para gestionar y compartir recursos en entornos de computación *grid*.

El modelo de computación *grid* presenta limitaciones que condicionan las aplicaciones que pueden beneficiarse del modelo. Normalmente, para que una aplicación pueda ser ejecutada de forma eficiente en la *grid*, se considera que se deberían cumplir requisitos como los siguientes: *a)* debería poder ser dividida en varias subtareas que puedan ser ejecutadas en paralelo sin que haya apenas comunicación, de forma que las subtareas sean prácticamente independientes entre sí o tengan muy pocas dependencias con otras; *b)* que el coste por continuar un trabajo que ejecuta una subtarea en la *grid* sea muy bajo si se retiran algunos nodos *grid*; y *c)* que el trabajo pueda utilizar nodos *grid* que se añadan dinámicamente al sistema. Los sistemas *grid* existentes no permiten explotar la segunda característica al máximo. En consecuencia, el coste asociado a la migración de un trabajo a la *grid* es, generalmente, elevado.

1.1.2 Propiedades no funcionales o de calidad de servicio en arquitecturas orientadas a servicio

En el sentido tradicional, la *calidad de servicio* (*quality of service* o QoS) se refiere al conjunto de tecnologías y técnicas utilizadas para garantizar que las operaciones de un sistema tengan resultados predecibles. Por ejemplo, en el caso de una red, se definen como atributos principales de QoS la disponibilidad, el ancho de banda, la latencia y la tasa de error.

En el dominio de los servicios web (y por consiguiente, también en el de los servicios *grid* que dan acceso a los diferentes recursos de la misma), la QoS se refiere a los atributos que caracterizan la calidad de un servicio web, que además de los mencionados con la red suelen incluir atributos relacionados con las prestaciones, la productividad, la fiabilidad, la escalabilidad, la robustez, la gestión de excepciones, la exactitud, la integridad, la accesibilidad, la disponibilidad, la interoperabilidad, y la seguridad de los servicios.

La utilización de estas propiedades no funcionales o atributos de calidad de servicio tiene como objetivo dar soporte a técnicas que permitan a las diferentes aplicaciones seleccionar varios servicios equivalentes en funcionalidad, pero con diferente nivel de servicio (*level of service* o LoS) o comportamiento esperado en cuanto a las prestaciones del servicio. En general, los parámetros que deben ser considerados para describir dicho comportamiento son: el tiempo de respuesta promedio, la productividad esperada y la disponibilidad del servicio en el tiempo.

1.1.3 Grid semántica

La *grid semántica* (de manera análoga a la web semántica respecto a la web) es definida por Goble, Kesselman y Sure (2005) como una extensión de la *grid* en la que la información y los servicios se presentan con un significado bien definido a través de descripciones que puedan ser procesadas por una máquina y cuyo objetivo es maximizar el potencial de la compartición y reutilización de los recursos descritos.

La *grid* semántica propone que la información sobre la *grid* sea descrita usando un modelo de datos semántico de manera que se facilite el descubrimiento de recursos y la creación de organizaciones virtuales. Las descripciones semánticas del modelo constituyen metadatos sobre los recursos que normalmente son representados mediante tecnologías de la web semántica como el modelo de datos para metadatos denominado *Marco de Descripción de Recursos* (*Resource Description Framework* o RDF). La *grid* semántica también propone la utilización de tecnologías de la web semántica y otras relacionadas dentro del *middleware* de la *grid*.

El concepto de la *grid* semántica fue articulado por primera vez en el contexto de la denominada *e-Ciencia* (*e-Science*), observando que este enfoque es necesario para alcanzar un alto grado de facilidad de uso y automatización de la *grid*, permitiendo así colaboraciones y computaciones más flexibles a escala global.

Algunas actividades de la *grid* semántica fueron coordinadas a través del *Semantic Grid Research Group* del *Open Grid Forum* (OGF), aunque existieron otros grupos de investigación que también realizaron aportaciones muy interesantes para intentar articular una infraestructura semántica explícita sobre la infraestructura *grid* que, potencialmente, permitiera incrementar la interoperabilidad y ganar flexibilidad para la misma.

1.2 Motivación

Debido a que en los últimos años se han ido consolidando las infraestructuras de computación *grid* como plataformas de acceso a recursos de computación y de almacenamiento en forma de servicios sobre en Internet, los usuarios de estas infraestructuras han ido generando nuevas necesidades que inicialmente no existían, como por ejemplo, la posibilidad de vincular sus solicitudes de ejecución de trabajos en la *grid* con los recursos más apropiados para ello, con el objetivo de mejorar las prestaciones de dichos trabajos, así como la eficiente utilización de los recursos *grid*. Es por ello que son necesarios nuevos mecanismos de planificación de trabajos y de administración de recursos que permitan dotar a las infraestructuras *grid* de nuevas capacidades basadas en la utilización de propiedades no funcionales o de calidad de servicio en dichos mecanismos.

Uno de los problemas abiertos en el contexto de las arquitecturas orientadas a servicios, como la *grid*, es el del descubrimiento de recursos adecuados para llevar a cabo una tarea determinada dentro de una aplicación (Chung 1993, Brooke et al. 2004a, 2004b).

La mayor parte de los enfoques de descubrimiento de servicios utilizados en estas arquitecturas están basados en el análisis de sus propiedades funcionales, es decir, en aquellas características que definen qué es lo que hace el servicio o qué ofrece. Sin embargo, existen otro tipo de propiedades no funcionales o de calidad de servicio que se pueden utilizar con este objetivo y sobre las cuales no se ha trabajado en profundidad aún, aunque hay iniciativas que han intentado trabajar de manera aislada con propiedades sencillas de obtener como la disponibilidad y el rendimiento de los recursos.

En muchos casos, nos encontramos con situaciones en las que los trabajos que se envían a la *grid* fallan debido a que los recursos que se les asignaron no estaban disponibles o habían caído. Ello provoca que los trabajos deban ser reenviados de nuevo y que se realice una nueva búsqueda y asignación de recursos. Además de la pérdida del trabajo realizado, también se produce un

derroche en el uso de los recursos, así como un incremento de tiempos de ejecución nada deseable.

Por un lado, existen herramientas *grid* que permiten reanudar algunos tipos de trabajos desde el punto de fallo, pero no es el caso general y además tampoco resuelve el problema de la pérdida de trabajo y tiempo. Aun así, los sistemas tolerantes a fallos basados en la recuperación de trabajos son muy costosos en sistemas distribuidos, por lo que en muchos casos se suele optar por cancelar aquellos trabajos que no han terminado al cabo de cierto tiempo límite y volver a enviarlos de nuevo.

Por otro lado, se suele recurrir también a monitorizar los recursos que fallan y ponerlos manualmente como requisitos del trabajo que se lanza, de manera que nunca se envía el trabajo a esos "malos" recursos. También se suele especificar como requisito del trabajo el tiempo esperado para la finalización del mismo, de manera que se le pueda asignar un recurso cuyo tiempo de respuesta sea menor que dicho tiempo esperado de finalización del trabajo. Y la misma idea se suele aplicar a la hora de considerar el tamaño de los ficheros utilizados o la salida esperada por un trabajo, en relación al espacio físico disponible en los recursos de almacenamiento.

Sin embargo, todas estas soluciones son muy específicas y claramente mejorables. Gran parte de los servicios desarrollados para el *middleware* de la *grid*, que también podemos denominar como consumidores de información *grid*, necesitan tener conocimiento sobre el comportamiento de los recursos *grid* para poder realizar una adecuada selección y asignación de los mismos que permita ejecutar trabajos en sistemas tan dinámicos y complejos como los sistemas *grid*.

De cara a obtener estas propiedades no funcionales o de calidad de servicio sobre los recursos para construir dicho conocimiento, las herramientas que podemos denominar proveedores de información *grid* tienen que seleccionar, medir y analizar distintas métricas sobre los recursos (sobre su comportamiento ejecutando trabajos, almacenando datos, etcétera) para ofertarlas a los consumidores de información *grid*. Sin embargo, no se ha realizado hasta la fecha un amplio estudio sobre cómo representar, calcular y medir las propiedades no funcionales o de calidad de servicio que pueda ser utilizado para determinar el nivel de servicio de un recurso *grid*.

Además, dada la complejidad tanto de los sistemas *grid* como de los conceptos relacionados con las propiedades no funcionales o de calidad de servicio, se ha de realizar un esfuerzo por especificar semánticamente las relaciones entre los distintos tipos de recursos *grid* y sus propiedades no funcionales o de calidad

de servicio, así como el significado de cada uno de los términos utilizados. Existen iniciativas para describir modelos de información *grid* por un lado, y conceptos de calidad de servicio por otro, pero no hay apenas relación entre ellos más allá de algunas propuestas con limitaciones de algunos grupos de trabajo del OGF.

Por un lado, los modelos de información *grid* existentes no usan el mismo lenguaje para representar los mismos conceptos sobre recursos, aunque varios grupos de la comunidad *grid* han detectado la necesidad de integrar sus modelos y hacerlos interoperables. Además, existe una falta de representación de propiedades no funcionales o de calidad de servicio sobre los recursos. En el mejor de los casos, algunos modelos especifican un rango de valores para medir y comparar el rendimiento de los recursos *grid*, pero faltan muchos detalles en general. Algunos proveedores y consumidores de información *grid* desarrollados específicamente para algunos proyectos usan técnicas aisladas para tratar el rendimiento de los recursos y su disponibilidad en un momento dado.

Por otro lado, la mayoría del trabajo existente en cuanto a modelos de calidad de servicio en general se centran en flujos de trabajo y servicios web en arquitecturas SOA, pero algunas de las métricas especificadas en estos modelos no son válidas para entornos *grid*, dado que los recursos y sus características no funcionales son más diversas, dinámicas e inter-organizacionales. Esa es la razón por la que es necesario un nuevo modelo de representación con métricas adaptadas a la *grid* y una metodología para la población del modelo.

Es por ello que, a partir de los trabajos existentes, parece razonable la necesidad de proponer nuevos enfoques y desarrollos tanto de modelos unificados como de metodologías que facilite la labor de utilizar propiedades no funcionales en el dominio *grid*, sin olvidar la oportunidad de impulsar con estas iniciativas el paradigma de la *grid* semántica.

Por último, en cuanto a los mecanismos de planificación de trabajos y de administración de recursos existentes en las infraestructuras *grid*, a pesar de los avances de los últimos años, el soporte para las propiedades no funcionales o de QoS en entornos de computación *grid* es aún muy limitado y, hasta el momento, no existe una solución definitiva para el problema.

En este trabajo, en primer lugar, se propone un nuevo modelo basado en tecnologías semánticas para integrar los modelos existentes tanto a nivel de representación de información *grid* como de propiedades no funcionales en general, con el objetivo de enriquecer la información existente sobre cada uno de los recursos de diferente nivel existentes en los nodos de la *grid*.

En segundo lugar, también se propone una metodología para la representación de propiedades no funcionales en el dominio *grid* que pueda ser utilizada por los administradores de los diferentes nodos para enriquecer la definición de sus recursos y mejorar la experiencia de sus usuarios respecto a la ejecución de sus trabajos.

En tercer lugar, se aplica dicha metodología en la representación de dos propiedades no funcionales a nivel de recurso de computación (fiabilidad y disponibilidad) y una a nivel de un nodo *grid* (fiabilidad).

Y en cuarto lugar, se ofrece un conjunto de algoritmos modificados para incluir las propiedades no funcionales a nivel de recurso de computación en el descubrimiento y selección de los mismos para un modelo de pruebas dentro de un entorno de simulación de un nodo *grid*, de tal manera que permita comprobar los resultados que se obtienen mediante la utilización combinada o no de dichas propiedades no funcionales en el descubrimiento y selección de los recursos.

1.3 Hipótesis, objetivos y contribuciones previstas

Con el fin de delimitar con mayor precisión el ámbito del trabajo a realizar, presentamos en este apartado la hipótesis cuya verificación promueve la presente investigación, los objetivos fijados para la tesis, dividiendo estos en generales, específicos y operacionales, así como los requisitos tanto operacionales como arquitectónicos que debe recoger la solución propuesta.

1.3.1 Premisas, hipótesis y restricciones

El trabajo descrito en este documento está basado en un conjunto de premisas y restricciones que contribuyen a contextualizar el trabajo realizado. Las principales aportaciones se resumen mediante una serie de hipótesis de partida, cuya verificación se realiza en el capítulo de evaluación de este trabajo.

1.3.1.1 Premisas

En el caso de las premisas, estas ayudan a explicar las decisiones tomadas para el desarrollo de las soluciones adoptadas con la finalidad de alcanzar los objetivos propuestos en este trabajo que se mencionarán más adelante.

Premisa P1. Tanto el modelo como la metodología de la tesis estarán orientados a unificar criterios relacionados con la representación de propiedades no funcionales en el dominio *grid*, de manera que el concepto definido sea entendible por la comunidad *grid* y se ajuste en lo posible a definiciones y conceptos manejados por los organismos de estandarización correspondientes, siendo especialmente consecuentes con las directrices expuestas por el OGF a través de la especificación OGSA y tomando como referencia los documentos estándar para la definición de recursos (GLUE) y trabajos (JSDL) en la *grid*.

Premisa P2. Tanto el modelo como la metodología de la tesis tomarán en consideración las recomendaciones de la propuesta S-OGSA para la extensión de la *grid* con capacidades semánticas, al considerar que es la iniciativa más completa en el estado de la cuestión estudiado en la presente tesis, así como una de las pocas que han sido implementadas integrándose con la especificación OGSA.

Premisa P3. La fiabilidad y la disponibilidad son las propiedades no funcionales más mencionadas en la bibliografía en el estado de la cuestión estudiado en la presente tesis, así como algunas de las menos aplicadas en las iniciativas analizadas también en la presente tesis.

Premisa P4. Acceder a *grids* reales como usuario no permite realizar modificaciones en el *middleware* desplegado en las mismas porque son sistemas en producción.

Premisa P5. Acceder a *grids* reales como proveedor de recursos es un proceso complicado que exige una serie de obligaciones y compromisos que implican a toda una entidad, además de un elevado esfuerzo de administración y mantenimiento de los recursos por parte de personal especializado.

Premisa P6. Desplegar un entorno *grid* virtual es una labor muy costosa frente al uso de herramientas de simulación y en ningún caso llegará a reproducir las características de un entorno *grid* real, a pesar de requerir prácticamente el mismo volumen de tareas de administración y mantenimiento.

Premisa P7. Las herramientas de simulación para el dominio *grid* aparecen en publicaciones aceptadas por la comunidad científica porque han madurado lo suficiente en los últimos años como para poder ser utilizadas por investigadores con dificultades como las mencionadas en las premisas anteriores o por usuarios que quieren realizar pruebas antes de usar la *grid* en producción a la que tienen acceso.

Premisa P8. Los datos reales que se pueden obtener a través de portales públicos de monitorización de entornos *grid* en producción suelen ser de alto nivel y estar agregados ofreciendo solamente cierta información sobre su rendimiento global y sobre su estado actual, por lo que en la mayoría de los casos resulta muy complicado obtener datos de bajo nivel sobre el comportamiento de los recursos individuales de un nodo en concreto. Además, no se suelen publicar este tipo de datos de bajo nivel como tasas de fallos, tiempo medio entre fallos, tiempo medio de reparación, etcétera, ya sea por prohibición explícita del proveedor de su *hardware* o porque exige una dedicación de recogida y organización de los mismos que no suelen realizar los responsables de los nodos porque están totalmente volcados en las tareas de administración de los mismos. Son contados los equipos de investigación que realizan este tipo de tareas y en la mayoría de los casos guardan con celo sus datasets para sus investigaciones y publicaciones.

1.3.1.2 Hipótesis

Las hipótesis cuya verificación promueve la presente investigación pueden enunciarse de la siguiente manera:

Hipótesis H1. Es posible utilizar propiedades no funcionales en la selección de recursos de computación *grid* para, en general, mejorar el tiempo de finalización medio o la tasa de fallos de un mismo trabajo lanzado a ejecución sobre un mismo nodo *grid*.

Hipótesis H2. Es posible utilizar herramientas matemáticas para la predicción y simulación de propiedades no funcionales sobre recursos de computación *grid*.

1.3.1.3 Restricciones

Las restricciones definen los límites de la contribución de este trabajo a la vez que permiten determinar futuras líneas de trabajo.

Restricción R1. El modelo de representación de información se desarrollará mediante una ontología fundacional que permita la reutilización de otras ontologías existentes como artefacto para crear una taxonomía y vocabulario unificado en el dominio *grid*, quedando fuera del alcance de la tesis la posibilidad de ser utilizada por un razonador para realizar inferencias y otras tareas relacionadas con tecnologías semánticas.

Restricción R2. La metodología de representación de propiedades no funcionales en el dominio *grid* se presentará como un marco general que permita plantear diferentes métodos para una propiedad no funcional sobre un recurso *grid* de cualquier nivel, pero en la presente tesis únicamente se desarrollarán tres métodos: dos para la representación de sendas propiedades no funcionales a nivel de recurso de computación (fiabilidad y disponibilidad) y otro más para la representación de propiedades no funcionales a nivel de nodo *grid* (fiabilidad), si bien en ningún caso se podrán desarrollar completamente debido a las características de la evaluación de la tesis cuya naturaleza se describe en siguientes restricciones.

Restricción R3. Dada la dificultad de reproducir un entorno *grid* real y de conseguir datos reales de rendimiento sobre toda una infraestructura *grid*, los experimentos de evaluación de la tesis se realizarán con una herramienta de simulación implementando planificación y selección de recursos a nivel de nodo *grid*, quedando fuera del alcance de la tesis la realización de experimentos a nivel de metaplanificación.

Restricción R4. Dada la naturaleza de la evaluación de la tesis, no ha lugar a la utilización del modelo de representación propuesto en la misma. Así mismo, se adaptarán al contexto y la arquitectura del modelo de pruebas *SimDag* de *SimGrid* aquellas actividades de los métodos de representación de propiedades no funcionales desarrollados a partir de la metodología de la tesis.

Restricción R5. Los experimentos de evaluación de la tesis se realizarán con la herramienta *SimGrid*, modificando su modelo de pruebas *SimDag* que provee funcionalidades para simular la planificación y ejecución de tareas paralelas, dependientes o no, descritas siguiendo un modelo de grafos acíclicos dirigidos (*Direct Acyclic Graphs* o DAG) sobre una plataforma de recursos de computación descrita siguiendo un formato XML determinado y la implementación del popular algoritmo de planificación min-min en el que para cada tarea ejecutable se selecciona su mejor recurso de computación (aquel que minimiza su tiempo de finalización o completion time) para posteriormente seleccionar la tarea que tenga el menor tiempo de finalización de entre todas y planificarla sobre dicho mejor recurso de computación seleccionado al principio para la tarea.

Restricción R6. Los resultados de los experimentos de evaluación de la tesis se circunscriben al algoritmo de planificación mencionado anteriormente y las diferentes modificaciones realizadas sobre el mismo para la selección de recursos de computación mediante la utilización independiente o combinada de propiedades no funcionales fiabilidad y disponibilidad de los recursos de computación.

Restricción R7. Las propiedades no funcionales que se utilizarán en las modificaciones del algoritmo de planificación mencionado anteriormente y que se probarán en los diferentes escenarios que se planteen en los experimentos de evaluación de la tesis serán la fiabilidad y la disponibilidad de los recursos de computación como indicadores estáticos, dada la restricción planteada anteriormente respecto al modelo de pruebas.

Restricción R8. Los datos sobre fallos de los recursos que serán utilizados en la evaluación de la tesis para simular escenarios realistas serán generados siguiendo los modelos matemáticos más plausibles encontrados en el estado de la cuestión analizado.

1.3.2 Objetivos

De cara a la demostración de la validez de las hipótesis mencionadas, fijamos a continuación el objetivo general del presente trabajo de investigación así como los objetivos específicos y operacionales que deberán llevar a la consecución del primero.

1.3.2.1 Objetivo general

El principal objetivo que se persigue con el presente trabajo es el siguiente:

Objetivo general. Desarrollar un modelo y una metodología de representación de propiedades no funcionales sobre recursos *grid* que permita a la comunidad de administradores, desarrolladores y usuarios de la *grid* integrar fácilmente este tipo de información en los sistemas *grid* existentes de manera que enriquezca el descubrimiento y selección de recursos en la *grid* usando esta extensión de información semántica asociada a los mismos y siguiendo el paradigma de la *grid* semántica para impulsar su adopción.

Los desarrollos conducentes a lograr el objetivo propuesto deberán cumplir una serie de requisitos que hemos clasificado en dos tipos:

- **Requisitos operacionales**

  Hacen referencia al propio funcionamiento de las soluciones propuestas. Los requisitos identificados son:

  - **RO1.** Las soluciones estarán orientadas a unificar criterios relacionados con la representación de propiedades no funcionales en el dominio *grid*, de manera que el concepto definido sea entendible por la comunidad *grid* y se ajuste en lo posible a definiciones y conceptos manejados por los organismos de estandarización correspondientes.
  - **RO2.** Las soluciones, en la medida de lo posible, deberán ajustarse a estándares.
  - **RO3.** Las soluciones deben dar soporte a diferentes modelos y aplicaciones de selección y asignación de recursos *grid*, así como de planificación de trabajos.
  - **RO4.** Las soluciones deberán poder ser probadas con herramientas de simulación.

- **Requisitos arquitectónicos**

  Hacen referencia a los requisitos que se deberían tener en cuenta a la hora de diseñar las soluciones. Los requisitos identificados son:

  - **RA1.** Las soluciones deberán ser fácilmente integrables con otras soluciones existentes, evitando modificar en lo posible los sistemas de información *grid* actuales, así como los estándares y aplicaciones existentes.
  - **RA2.** Las soluciones deberán ser fáciles de implementar y de usar por parte de los desarrolladores.
  - **RA3.** Las soluciones deberán ser fáciles de mantener, es decir, cualquier cambio solo debería afectar a los desarrollos propuestos y no alterar al resto de elementos del ecosistema *grid*.
  - **RA4.** Las soluciones deberán ser extensibles permitiendo añadir fácilmente capacidades adicionales que pueden ser requeridas por diferentes tipos de aplicaciones.

1.3.2.2 Objetivos específicos y operacionales

En consonancia con el objetivo general definido y con las acciones necesarias para la confirmación de la hipótesis enunciada, los objetivos específicos de esta tesis son los siguientes:

- **OE1**. Definir la filosofía y el enfoque deseado para el modelo y la metodología de la tesis, así como para el entorno de pruebas. Este objetivo se logra realizando un examen exhaustivo del estado de la cuestión con la pretensión de identificar aquellas características de las iniciativas precedentes que puedan desembocar en un buen resultado en cuanto a los desarrollos que se lleven a cabo en la tesis.
- **OE2.** Desarrollar un sistema basado en un modelo y una metodología para la representación de propiedades no funcionales o de calidad de servicio en el dominio *grid*.
- **OE3**. Seleccionar las propiedades no funcionales que van a emplearse en la experimentación. Para seleccionar las propiedades no funcionales que van a utilizarse para este propósito es indispensable conocer en profundidad el estado de la cuestión relacionado. Es importante determinar qué propiedades cuentan con un mayor interés científico. Este objetivo ha de establecerse antes de comenzar con la implementación de la técnica, ya que esta dependerá de las propiedades seleccionadas para la fase de pruebas.
- **OE4**. Diseñar e implementar la técnica de evaluación siguiendo la filosofía establecida en el objetivo OE1 y teniendo en cuenta varios subobjetivos de obligado cumplimiento como, por ejemplo, evitar un excesivo consumo de recursos computacionales, resolver los problemas en un tiempo competitivo o carecer de un diseño excesivamente complejo.
- **OE5**. Configurar un entorno de pruebas. Este objetivo consiste en conseguir identificar las alternativas existentes para la configuración de un entorno de pruebas válido para la tesis. Es importante que el conjunto de tecnologías utilizadas y de escenarios planteados para la experimentación de la tesis estén avalados por la comunidad *grid* y la comunidad científica. Para este propósito es necesario un exquisito conocimiento de la literatura disponible.
- **OE6**. Analizar y evaluar los resultados logrados en la experimentación. Una vez realizada la experimentación pertinente se analizarán los resultados obtenidos. Este análisis de los resultados se llevará a cabo tanto para el tiempo de finalización de los trabajos como para el número de fallos que se producen. Para realizar una comparación apropiada se utilizarán variables propias de la estadística descriptiva, como la media, la mediana y la desviación típica. Además de esto, con la intención de realizar una equiparación objetiva, rigurosa y fiable, se

llevarán a cabo varios tests estadísticos, como el test de Shapiro-Wilk, el test de Kruskal-Wallis y el test de los rangos con signo de Wilcoxon.

Para lograr los objetivos específicos mencionados, se plantean los siguientes objetivos operacionales:

- **OO1**. Estudiar los modelos de representación de información sobre el dominio *grid*.
- **OO2**. Estudiar los modelos de representación de información sobre propiedades no funcionales o de calidad de servicio en las arquitecturas orientadas a servicio.
- **OO3**. Diseñar un modelo común para la representación de propiedades no funcionales o de calidad de servicio en el dominio *grid*.
- **OO4**. Estudiar el *middleware grid* y las infraestructuras de computación *grid* para determinar quiénes son los proveedores y consumidores de información en el dominio *grid*, concretamente aquellos encargados de organizar y gestionar los recursos *grid*.
- **OO5**. Estudiar las soluciones planteadas en la literatura que abordan el problema de la selección de recursos en entornos de computación *grid*, especialmente aquellas que hagan referencia a la utilización de propiedades no funcionales o de calidad de servicio.
- **OO6**. Definir una metodología que formalice las actividades y tareas a realizar para poder llevar a cabo la representación de propiedades no funcionales de los recursos *grid*, basándose en el modelo de representación desarrollado. Esta metodología facilitará a los administradores de nodos *grid* la publicación de este tipo de información sobre sus recursos.
- **OO7.** A partir del estudio realizado sobre las soluciones planteadas en la literatura que abordan el problema de la utilización de propiedades no funcionales en la selección de recursos en entornos de computación *grid*, determinar las propiedades no funcionales de diferente nivel (recurso de computación y nodo *grid*) sobre las que se aplicará la metodología de la tesis.
- **OO8**. Aplicar la metodología desarrollada para demostrar cómo trabajar con propiedades no funcionales sobre recursos de distinto nivel, realizando las actividades y tareas correspondientes en cada caso.

- **OO9**. Seleccionar y configurar el entorno de pruebas adecuado para la evaluación de la tesis, determinando tanto las herramientas, como los algoritmos y modelos de pruebas a utilizar.
- **OO10**. Modificar el modelo de pruebas para comprobar qué efecto produce la utilización de propiedades no funcionales en la selección de recursos de computación en un nodo *grid* simulado. Para este fin se diseñarán experimentos o pruebas en diferentes escenarios que permitan verificar las hipótesis de la tesis así como las características y bondades de los desarrollos llevados a cabo en la misma.

En cuanto a los objetivos operacionales que posibilitarán la consecución de los objetivos específicos mencionados anteriormente, indicamos su vínculo con el objetivo específico al que contribuyen en la tabla 1.1 y los requisitos con los que se relacionan en la tabla 1.2.

	Objetivos operacionales	**Objetivo específico**
OO1	Estudiar modelos de representación de información del dominio *grid*	OE1, OE2
OO2	Estudiar modelos de representación de propiedades no funcionales	OE1, OE2
OO3	Estudiar el middleware *grid* y las infraestructuras de computación *grid*	OE1, OE2
OO4	Diseñar un modelo común para la representación de propiedades no funcionales en la *grid*	OE2
OO5	Estudiar las soluciones que abordan el problema de la selección de recursos en la *grid*	OE2, OE3
OO6	Definir una metodología para la representación de propiedades no funcionales en la *grid*	OE2
OO7	Determinar las propiedades no funcionales sobre las que se aplicará la metodología de la tesis	OE3, OE4
OO8	Aplicar la metodología con propiedades no funcionales sobre recursos de distinto nivel	OE3, OE4
OO9	Seleccionar y configurar el entorno de pruebas adecuado para la evaluación de la tesis	OE4, OE5
OO10	Probar el uso de propiedades no funcionales en la selección de recursos *grid*	OE5, OE6

Tabla 1.1 Objetivos operacionales más relevantes y objetivo específico al que contribuyen

Cabe decir que la información mostrada no pretende ser un exhaustivo plan del trabajo seguido en esta tesis, sino tan solo mostrar algunos de los más relevantes desde el punto de vista científico.

	Objetivos operacionales									
Requisitos	**OO1**	**OO2**	**OO3**	**OO4**	**OO5**	**OO6**	**OO7**	**OO8**	**OO9**	**OO10**
RO1	✓	✓	✓	✓						
RO2				✓		✓		✓		
RO3				✓		✓		✓		
RO4						✓	✓		✓	✓
RA1	✓	✓	✓	✓	✓	✓				
RA2				✓		✓		✓	✓	✓
RA3				✓		✓		✓		✓
RA4				✓		✓		✓		

Tabla 1.2 Relación entre los requisitos y los objetivos operacionales

Además de todos estos objetivos necesarios para alcanzar el objetivo general de la tesis que permitirá corroborar la hipótesis planteada en la misma, se tendrán en consideración los siguientes objetivos personales en el desarrollo de la tesis:

- Maximizar, en la medida de lo posible, la contribución a la comunidad científica mediante la publicación de artículos en congresos y revistas, tanto nacionales como internacionales, de carácter tanto científico como divulgativo.
- Maximizar la claridad y reproducibilidad de los experimentos planteados para que puedan ser estudiados, utilizados y/o modificados posteriormente por cualquier investigador o desarrollador, ya sea con el objetivo de aportar algún valor añadido o aplicarlos en alguna otra herramienta.

1.3.3 Contribuciones previstas

En la presente tesis pueden encontrarse las siguientes contribuciones mayores y menores:

Contribución C1. Un modelo para la representación de información sobre recursos *grid* y propiedades no funcionales.

C1.1. Ontología fundacional para la representación de información sobre recursos *grid* y propiedades no funcionales.

Contribución C2. Una metodología para la representación de propiedades no funcionales en el dominio *grid*.

C2.1. Proceso de desarrollo de métodos para la representación de propiedades no funcionales en el dominio *grid*.

C2.2. Herramienta para la generación de datos simulados sobre fallos de recursos de computación a partir de un modelo matemático.

C2.3. Método para la representación de la fiabilidad de un recurso de computación.

C2.3.1. Herramienta para la estimación del *Tiempo Medio Entre Fallos* (*Mean Time Between Failures* o MTBF).

C2.3.2. Herramienta para la predicción de la fiabilidad a partir del MTBF.

C2.3.3. Especificación y diseño de la aplicación necesaria para la publicación de datos relativos a la fiabilidad de un recurso de computación, siguiendo la arquitectura S-OGSA.

C2.4. Método para la representación de la disponibilidad de un recurso de computación.

C2.4.1. Herramienta para la estimación del *Tiempo Medio de Reparación* (*Mean Time to Repair* o MTTR).

C2.4.2. Herramienta para la predicción de la disponibilidad a partir del MTBF y el MTTR.

C2.4.3. Especificación y diseño de la aplicación necesaria para la publicación de datos relativos a la disponibilidad de un recurso de computación, siguiendo la arquitectura S-OGSA.

C2.5. Método para la representación de la fiabilidad de un nodo *grid*.

Contribución C3. Framework tecnológico de pruebas para *SimGrid*.

C3.1. Herramienta para la generación de ficheros sobre instantes de fallos realistas de un determinado host o recurso de computación con datos simulados a partir de un modelo matemático dado.

C3.2. Herramienta para la inclusión de propiedades no funcionales en los ficheros de plataformas o infraestructuras simuladas.

C3.3. Nueva versión del código del modelo de pruebas *SimDag*.

C3.2.1. Conjunto de variantes del algoritmo de planificación min-min que utilizan diferentes propiedades no funcionales para comprobar los resultados que se obtienen mediante su utilización combinada o no en el descubrimiento y selección de los recursos.

C3.2.2. Nuevas funcionalidades para la ampliación del informe de resultados y estadísticas de las simulaciones.

C3.4. Herramienta para la automatización de experimentos.

1.4 Metodología de investigación

En este apartado exponemos tanto el proceso de investigación como el método de trabajo que se seguirán para cumplir con los objetivos marcados en esta tesis. El proceso de investigación que se seguirá puede organizarse en cinco etapas o fases, que por otra parte son características en este tipo de trabajos (ver figura 1.1).

Durante la etapa de *identificación del problema* llevaremos a cabo el análisis del estado de la cuestión con el fin de detectar algunas limitaciones, necesidades u oportunidades de mejora presentes en el área de conocimiento en la que se entronca esta tesis. Este estudio permitirá además establecer el marco teórico sobre el cual basar la solución del problema. El *enunciado de la hipótesis* constituirá el enlace de unión entre los problemas y la solución, así como el timón que guiará los pasos de la investigación hacia su objetivo.

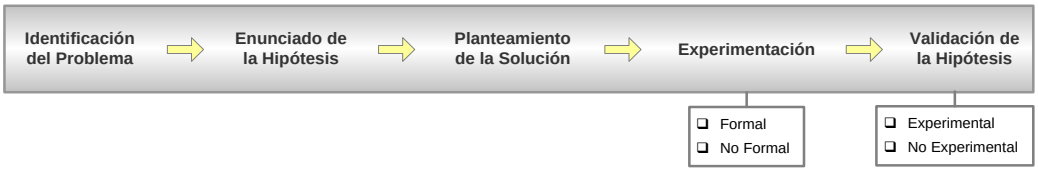

Figura 1.1 Etapas generales del proceso de investigación

El *planteamiento de la solución* perseguirá la consecución del objetivo propuesto en la tesis.

La *experimentación*, en términos generales, ayudará a la recogida y el análisis de datos que posibiliten incrementar el conocimiento sobre el ámbito de trabajo y sobre los diferentes avances en la solución. Estos datos constituirán indicadores que permitan seguir avanzando, replantearse estrategias y objetivos, etcétera, así como finalmente la *validación de la hipótesis* planteada. La experimentación que se realizará durante esta tesis puede separarse en dos categorías: una experimentación *no formal* (gobernada por el método de investigación-acción) y una experimentación *formal* (materializada a través de experimentos en laboratorio).

El método de trabajo se encuentra inspirado en los dos métodos de investigación siguientes: el método de *investigación-acción* (French et al. 1996, Wadsworth 1998), es decir, un enfoque cualitativo que se aplica para la definición, refinamiento continuo y validación de la solución planteada, y el método basado en *experimentos de laboratorio* (Straub et al. 2004, Boudreau

et al. 2001, Dix et al. 1998), es decir, una técnica cuantitativa que ha sido aplicada para la validación formal de los resultados.

1.4.1 Método investigación-acción

Este método se empleará a lo largo de todas las fases del proyecto aprovechando sus bondades. Por una parte, posibilitará la participación de los beneficiarios o evaluadores de la investigación durante el proceso de la misma. A través de la exposición de sus opiniones y sentimientos sobre la misma, permitirán obtener de esta manera el feedback necesario para replantearse o reconducir cualquier fase del proyecto y por tanto, afectar sobre ella. Sin embargo, será más relevante su uso en la fase de planteamiento de la solución, permitiendo el refinamiento y la mejora de ésta en base a ese conocimiento, y en la fase de experimentación no formal, a través de la preparación de las acciones que permitan obtener esos datos.

Tres son los ámbitos en los que el uso de este método se podrá apreciar con mayor claridad:

- **Colaboración en proyectos, redes de investigación y estancias en otros grupos de investigación.** En la medida de lo posible, ciertas tareas de investigación se realizarán en el marco de proyectos y redes de investigación, así como realizando visitas y estancias en otros grupos de investigación. En este escenario la participación de otros agentes en la elaboración de soluciones en los ámbitos de actuación de la tesis y fundamentalmente sus impresiones y valoraciones serán de incalculable valor en relación a los enfoques a adoptar para los desarrollos de la tesis y el establecimiento de nuevas líneas de trabajo (o modificaciones) a ser abordadas en las siguientes iteraciones.
- **Publicación en congresos.** De una forma análoga al caso anterior, las valoraciones recibidas por los evaluadores de los congresos en los que se irán presentando las distintas etapas de resultados de la investigación así como las opiniones vertidas en los foros de discusión de estos congresos, serán de incalculable valor, fundamentalmente en las fases iniciales de la investigación, para orientar su desarrollo.
- **Pruebas de carácter interno.** Durante distintas etapas del desarrollo de la tesis se llevarán a cabo distintos tipos de pruebas, unas más rigurosas y otras menos, unas involucrando a más personas y otras a menos, pero en todas ellas los resultados obtenidos y el análisis

de éstos permitirán obtener conclusiones sobre la validez o no de los resultados y sobre las siguientes acciones a llevar a cabo en las posteriores iteraciones del proceso cíclico.

Tomando como base las fases definidas dentro de este método (ver figura 1.2), establecidas dentro de un proceso cíclico y de refinamiento continuo, se componen de las siguientes actividades si las observamos desde el prisma de la presente tesis doctoral:

- **Planificación.** Su cometido variará en función del ciclo de iteración en el que nos encontremos, incluyendo desde la planificación inicial (o posterior) del análisis de las fuentes bibliográficas, pasando por la definición de las actividades a abordar durante las distintas iteraciones de desarrollo de la solución, la participación en proyectos relacionados, y hasta la planificación de nuevas pruebas y experimentos.
- **Acción.** Implica la ejecución de las actividades planificadas.
- **Observación.** Abarcará acciones como la lectura de información correspondiente a la fase inicial de revisión del estado de la cuestión para el planteamiento del problema y de la hipótesis y la elaboración del marco teórico base, el análisis de los resultados de las pruebas internas, las reuniones e informes de las empresas involucradas en los proyectos, evaluaciones y captura de opiniones en congresos, etc.
- **Reflexión.** En este proceso de reflexión, siguiendo las directrices del método investigación-acción, se hace partícipe al grupo crítico de referencia. Es decir, a los compañeros del grupo de investigación y principalmente, a los codirectores de la tesis, aunque también se podrá involucrar a investigadores de otros grupos de investigación participantes en proyectos y redes de investigación relacionados, a los participantes de foros de discusión en los distintos congresos, etcétera.

1.4.2 Método basado en experimentos en laboratorio

Este método será empleado durante la fase de experimentación formal con el fin de contrastar aquellas hipótesis que puedan ser verificadas mediante la realización de distintas pruebas. Se llevarán a cabo pruebas en distintos escenarios sobre una plataforma de experimentación, previamente definida, que permitirá evaluar diferentes parámetros relacionados con la hipótesis de partida. Un análisis de los resultados, bajo distintas perspectivas, permitirá concluir la bondad o no de los desarrollos llevados a cabo en la tesis.

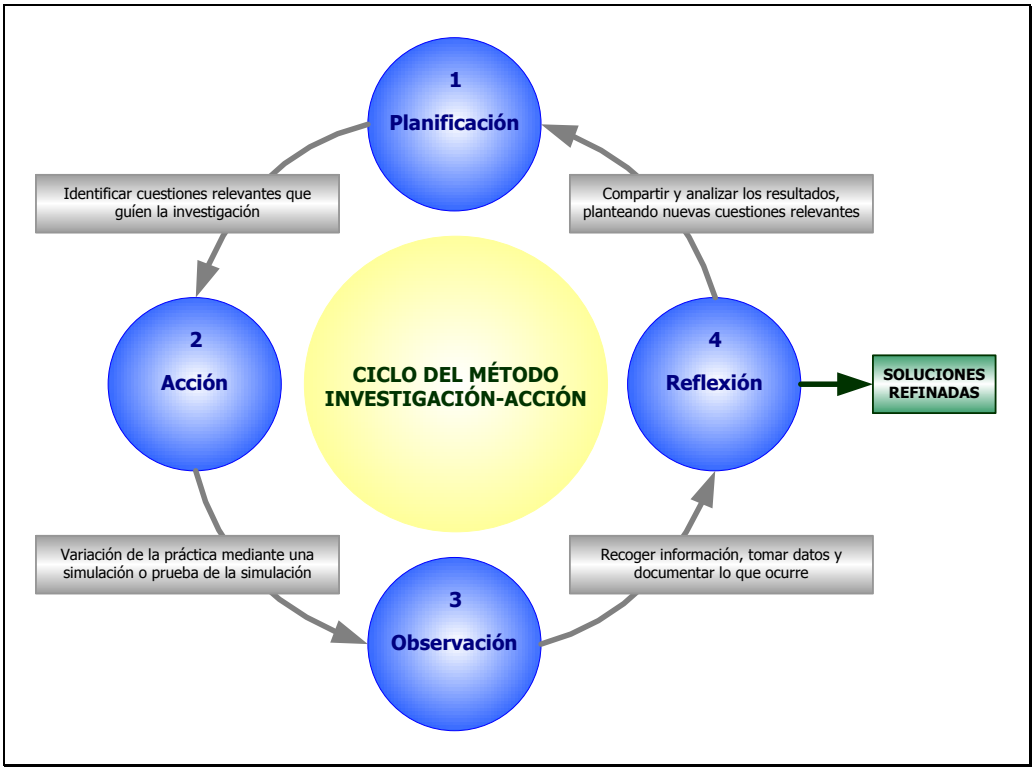

Figura 1.2 Fases y carácter cíclico del método investigación-acción

1.5 Organización del documento

La presente memoria se encuentra estructurada en cinco capítulos cada uno de los cuales aborda los siguientes contenidos:

- El *presente capítulo* proporciona una visión inicial del ámbito de trabajo de la tesis, así como las motivaciones generales que han propiciado su realización. Se han identificado además los problemas que esta tesis pretende resolver así como las características más relevantes de la solución propuesta, la hipótesis de trabajo de la tesis, los objetivos que guían su desarrollo y sus principales aportaciones.
- El *segundo capítulo* proporciona al lector una visión general del estado de la cuestión sobre las áreas de trabajo relacionadas con esta tesis, fijando de este modo los pilares conceptuales del trabajo desarrollado en la misma. Se abordan, primeramente, los modelos de representación de información tanto sobre recursos en el dominio *grid* como sobre propiedades no funcionales en las arquitecturas orientadas a servicio. En segundo lugar, se estudia la utilización de propiedades no funcionales por parte de los

proveedores y consumidores de información existentes en el dominio *grid*. En tercer lugar, se aborda el tema de la fiabilidad de un sistema en general y de la *grid* en particular, como una de las propiedades no funcionales más importantes y controvertidas mencionadas en la bibliografía revisada en la presente tesis. Y por último, se analizan diferentes alternativas para crear un entorno de pruebas *grid* que permita realizar la evaluación de la tesis. Así mismo, el capítulo finaliza con un apartado de conclusiones sobre el estado de la cuestión analizado.

- El *tercer capítulo* describe el grueso del trabajo de tesis: la propuesta de representación y utilización de propiedades no funcionales para el descubrimiento de recursos en el dominio *grid*. En primer lugar, un modelo de representación de propiedades no funcionales para el dominio *grid*. En segundo lugar, una metodología para la representación de propiedades no funcionales en el dominio *grid*. En tercer lugar, ejemplos de aplicación de dicha metodología en la representación de dos propiedades no funcionales a nivel de recurso de computación (*fiabilidad* y *disponibilidad*) y una a nivel de un nodo *grid* (*fiabilidad*). Y en cuarto lugar, un conjunto de algoritmos modificados para incluir las propiedades no funcionales a nivel de recurso de computación en el descubrimiento y selección de los mismos para un modelo de pruebas dentro de un entorno de simulación de un nodo *grid*, de tal manera que permita comprobar los resultados que se obtienen mediante la utilización combinada o no de dichas propiedades no funcionales en el descubrimiento y selección de los recursos.
- El *cuarto capítulo* aborda la experimentación llevada a cabo como medio para la validación de la hipótesis enunciada en la tesis. Se analiza la hipótesis, se seleccionan y justifican las estrategias de verificación más apropiadas, se describen las actividades y los experimentos realizados, se muestran los resultados de los mismos y se concluye la veracidad o no de la hipótesis.
- El *quinto capítulo* recoge finalmente las conclusiones de la tesis, así como las líneas futuras de trabajo y mejora.

Se concluye con un apartado dedicado a la bibliografía, con todas las referencias mencionadas en esta tesis, incluyendo otro apartado con referencias web, así como varios apéndices con contenidos extra para una mejor comprensión de algunas secciones de la presente memoria.

Desea que sea largo el camino.
Que sean muchas las mañanas estivales
en que con qué alegría, con qué gozo
arribes a puertos nunca antes vistos.

Constantino Cavafis

Capítulo 2

Estado de la cuestión

En este capítulo se realiza una revisión del estado de la cuestión en relación a los ámbitos de actuación de la tesis. En primer lugar se hace un breve repaso al *middleware* de la *grid* con el objetivo de analizar las tecnologías y herramientas más relevantes sobre las que se apoyan las infraestructuras *grid*, prestando especial atención a los sistemas de información *grid*.

En segundo lugar, se estudian qué iniciativas existen para la extensión semántica de la *grid*, prestando especial atención al criterio de mínima intrusión o máxima compatibilidad con la especificación OGSA del OGF.

En tercer lugar, se presenta un estudio sobre los diferentes modelos encontrados en la literatura tanto para la representación de información sobre recursos en el dominio *grid* en particular y sobre las propiedades no funcionales o de calidad de servicio en el contexto más amplio de las arquitecturas orientadas al servicio o SOA.

En cuarto lugar, se analizan qué otras aplicaciones presentes en la *grid* se consideran proveedores y/o consumidores de los sistemas de información *grid*, de cara a conocer su funcionamiento y el tipo de información que proveen y consumen.

En quinto lugar, se aborda el problema de la selección y asignación de recursos, así como de la planificación en el dominio *grid*, para conocer qué herramientas existen, qué algoritmos se utilizan y qué utilización se hace de las propiedades no funcionales de los recursos.

Por último, se hace un repaso a las distintas alternativas existentes para desplegar un entorno de pruebas *grid*.

2.1 Middleware grid: estándares de facto

La necesidad de disponer de e-infraestructuras ha favorecido el desarrollo de *middleware grid*, destacando *Globus Toolkit* [22] (ver figura 2.1), UNICORE [84] (ver figura 2.3) y *g-Lite* [19] (ver figura 2.2), que constituyen un núcleo muy sólido sobre el cual se soportan la mayor parte de los sistemas *grid* más importantes de la actualidad, siendo *Globus Toolkit* un estándar *de facto* en la mayoría de los casos.

Torres (2010) realiza un excelente resumen sobre las características de *Globus Toolkit* y *g-Lite*. Por su interés para contextualizar la presente tesis y dada su excelente redacción, se recomienda su lectura.

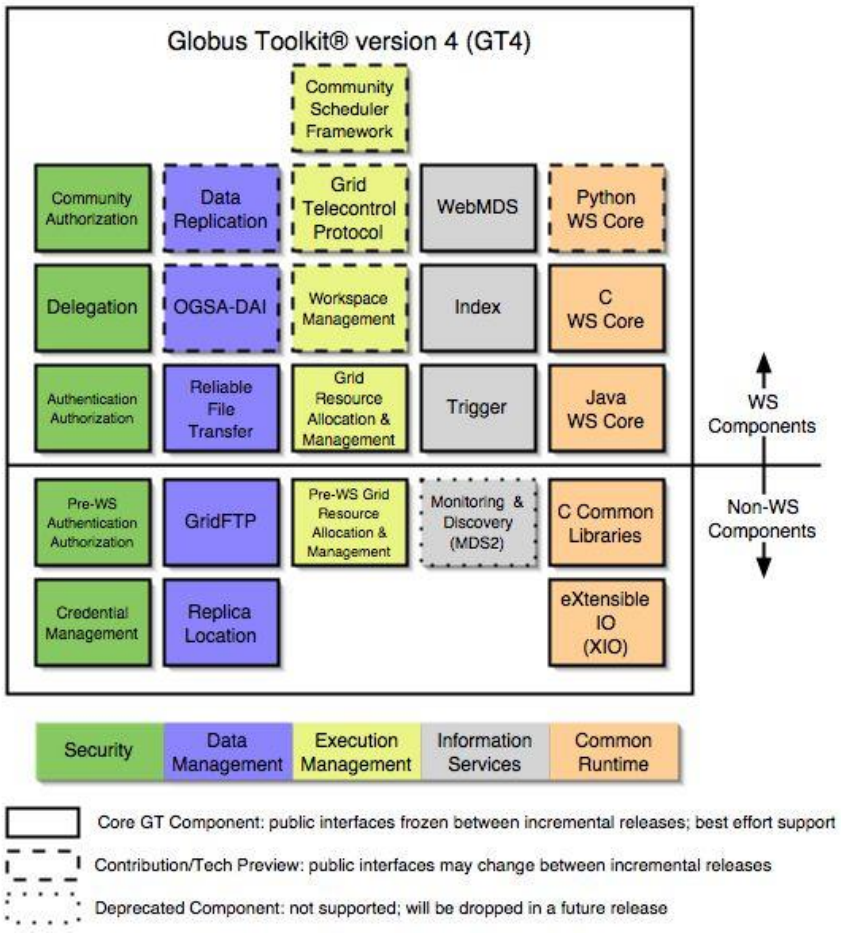

Figura 2.1 Componentes del middleware grid Globus Toolkit (recogido de [22])

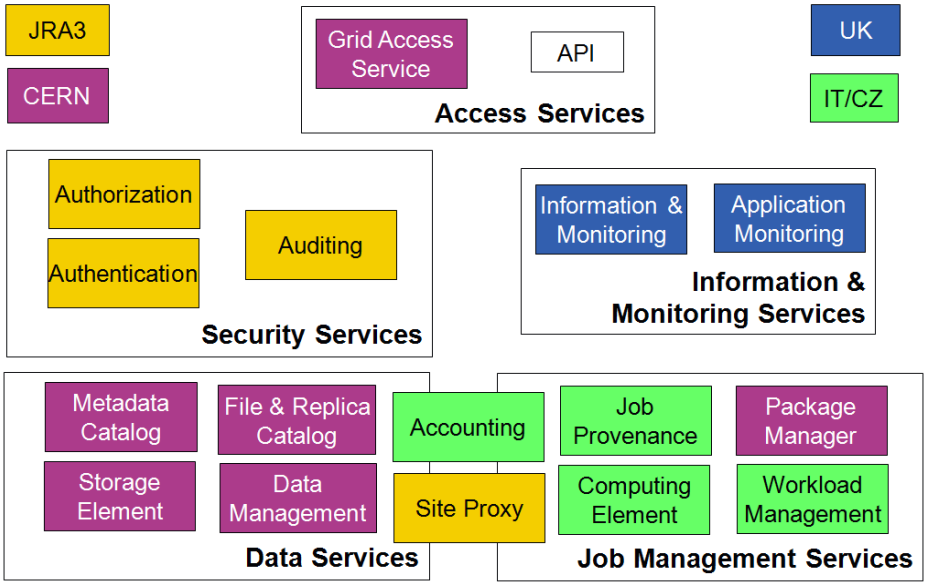

Figura 2.2 Componentes del middleware grid g-Lite (recogido de [19])

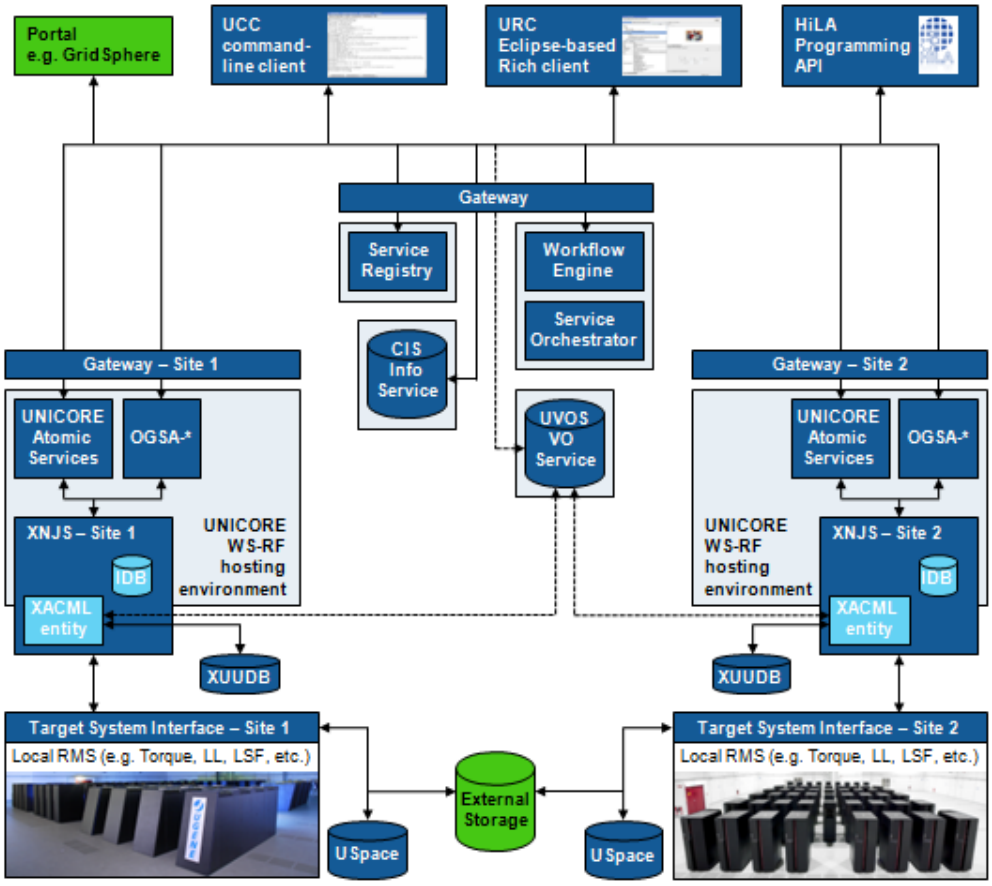

Figura 2.3 Componentes del middleware grid UNICORE (recogido de [84])

A lo largo de los años, *Globus Toolkit* se ha ido convirtiendo en un estándar de facto para el acceso a la *grid*, haciendo que una buena parte del *middleware grid* actual (incluido *g-Lite*) se base en su pila de componentes.

En relación al mismo, Torres (2010) resume perfectamente qué mecanismos ofrece para acceder a las propiedades de los recursos *grid* representados por su servicio *grid* correspondiente.

> «La información de estado del servicio es almacenada en un recurso, específicamente en una clase WSResourceProperties, definida por WSRF. Además, *WSRF* define un conjunto de interfaces (*PortTypes*) para interactuar con las propiedades de los recursos de un servicio. En WSDL, cada "PortType" identifica a un conjunto de operaciones abstractas y de mensajes, también abstractos, involucrados en estas operaciones. En general, los "PortTypes" definidos por WSRF permiten recuperar y modificar las propiedades de uno o varios recursos a la vez, ya sea accediendo por el nombre completo de los recursos, que no es más que una combinación de un nombre local y un identificador uniforme de recurso (Uniform Resource Identifier o URI), o bien interrogando a los recursos, utilizando el lenguaje XPath.» (Torres 2010: 16)

Así mismo, *Globus Toolkit* proporciona diferentes servicios de información mediante el sistema de monitorización y descubrimiento (*Monitoring and Discovery System* o MDS), cuyo funcionalidad está bien resumida por Torres (2010).

> «Cada uno de los servicios de información puede ser accedido de forma independiente, pero MDS proporciona un mecanismo estándar que los integra, y que permite publicar y recuperar información de configuración y de estado de los recursos computacionales en general (no sólo de los recursos de los servicios Web). Adicionalmente, *MDS* permite suscribir temas nuevos. Para ello, MDS proporciona varios mecanismos de suscripción y recogida de información que pueden ser utilizados por los servicios Web para suscribir sus recursos con el sistema de información. El mecanismo más simple consiste en utilizar encuestas programadas, a través de un colector que utiliza los "PortTypes" definidos por WSRF para recuperar, periódicamente, las propiedades de un recurso de un servicio. Otro mecanismo más complejo consiste en utilizar notificaciones, a través de un componente que se suscribe a un tema que contiene las propiedades de un recurso de un servicio, y que notifica cuando ocurren cambios en las mismas.» (Torres 2010: 19)

Los servicios para la asignación y gestión de recursos *grid* (*Grid Resource Allocation and Management* o GRAM) de *Globus Toolkit* proporcionan los medios para lanzar y monitorizar trabajos en la *grid* sobre varios tipos de planificadores locales (*Condor* [8], PBS [70], etc.) a partir de los requisitos especificados en un documento XML de descripción de trabajo, creando un gestor vinculado al trabajo (*JobManager*) que monitoriza el mismo durante todo su tiempo de vida y envía esta información al MDS. Respecto a estos servicios, Torres (2010) comenta que:

> »*GRAM* únicamente proporciona un acceso estandarizado a uno a más recursos locales gestionados a partir de los JobManager. La elección del recurso *GRAM* constituye una tarea de alto nivel que se aborda desde la perspectiva de la meta-planificación.» (Torres 2010: 21)

Si entendemos que un recurso o infraestructura *grid* puede ser una máquina, un clúster, un nodo *grid* o incluso toda una *grid*, queda claro que existirán diferentes niveles de planificación de recursos en la *grid*, por lo que cabe diferenciar los siguientes componentes en el modelo de planificación:

- LRMS (*Local Resource Management System*, sistemas de gestión de recursos locales, planificador, planificador local, gestor de recursos locales, gestor de carga local): en la mayoría de los casos se trata de herramientas que provienen de la computación de alto rendimiento y distribuida y que han sido adaptadas para sistemas *grid*. Ejemplos: *GridWay* [39], *Condor-G* [9].
- *Meta-planificador*: uno o más componentes del *middleware grid*. También se puede tratar de una herramienta externa que utiliza componentes o servicios *middleware*. Ejemplos: Condor, PBS.
- RMS (*Resource Management Service*, servicio de gestión de recursos): por lo general se trata de una herramienta *middleware grid* que proporciona una interfaz para la petición y el uso remoto de sistemas para la ejecución de trabajos. Ejemplo: GRAM.

Por lo tanto, un metaplanificador es un supervisor de gestores de recursos locales que controlan la utilización de recursos individuales como clústeres o supercomputadoras. Por ello, la metaplanificación es un componente clave de la *grid*, ya que es el responsable de optimizar la utilización de los recursos.

Torres (2010) realiza un excelente resumen sobre las características de los metaplanificadores *GridWay* y *Condor-G*, así como su relación con varios componentes de planificación local como el del *Globus Toolkit*. Por su interés para contextualizar la presente tesis y dada su excelente redacción, se recomienda su lectura.

2.2 Grid semántica

En los últimos años, varios proyectos han intentado avanzar en la misión y visión de la *grid* semántica. Un resumen de estas iniciativas puede encontrarse en Corcho et al. (2006), que básicamente son intentos de proveer una arquitectura para el desarrollo de aplicaciones orientadas a la *grid* semántica o simplemente servicios *grid* sensibles a la semántica.

Una de las iniciativas más relevantes es la propuesta de una arquitectura de referencia o marco sistemático para el diseño de componentes de *grid* semántica o aplicaciones. Como ya se mencionó en el contexto de la presente

tesis, la especificación OGSA tiene como objetivo definir un conjunto básico de capacidades y comportamientos para los sistemas o infraestructuras *grid*. La iniciativa OGSA semántica (*Semantic* OGSA o S-OGSA) propone una arquitectura de referencia que extiende OGSA para facilitar el manejo explícito de la información semántica asociada a la *grid* y define los servicios de conocimiento necesarios para apoyar un amplio espectro de nuevas capacidades de servicio. Guiados por un conjunto de principios de diseño, S-OGSA define un modelo, unas capacidades y unos mecanismos para proveer una plataforma unificada de exposición y provisión de metadatos explícitos en aplicaciones *grid*, incluyendo un marco formal y un conjunto de guías para facilitar el desarrollo de aplicaciones *grid* semánticas.

El modelo está determinado por los elementos que lo componen y sus interrelaciones, las capacidades se refieren a los servicios necesarios para tratar con dichos componentes y los mecanismos se corresponden con los elementos que habilitarán el acceso a la semántica una vez desplegada la arquitectura en una aplicación y sobre una plataforma *grid*.

Tal y como se puede apreciar en la figura 2.4, S-OGSA extiende el conjuntos de capacidades que un *middleware grid* debiera proveer para incluir "*Semantic Provisioning Services*" (caja cuadrada de color rosa) y "*Semantically Aware Grid Services*" o SAGS (puntos cuadrados de color rosa dentro de los servicios típicos especificados en OGSA).

Los *Semantic Provisioning Services* son los responsables de proveer y gestionar la información semántica explícita y sus asociaciones con las diferentes entidades *grid*, por lo que juegan un importante papel en la exposición, entrega y generación de metadatos, incluyendo servicios de razonamiento y gestión de ontologías, servicios de metadatos y servicios de anotación.

Los SAGS son aquellos servicios *grid* con funcionalidad ampliada que ofrecen las capacidades enumeradas por OGSA, pero que difieren de otros por tener una afiliación con información semántica explícita u operar usando dicha información.

Con el modelo S-OGSA se introduce la noción de la semántica ("*semantics*") dentro del modelo de la *grid*, como se puede apreciar en la figura 2.5 con la versión reducida del modelo S-OGSA con las principales entidades del mismo y sus relaciones.

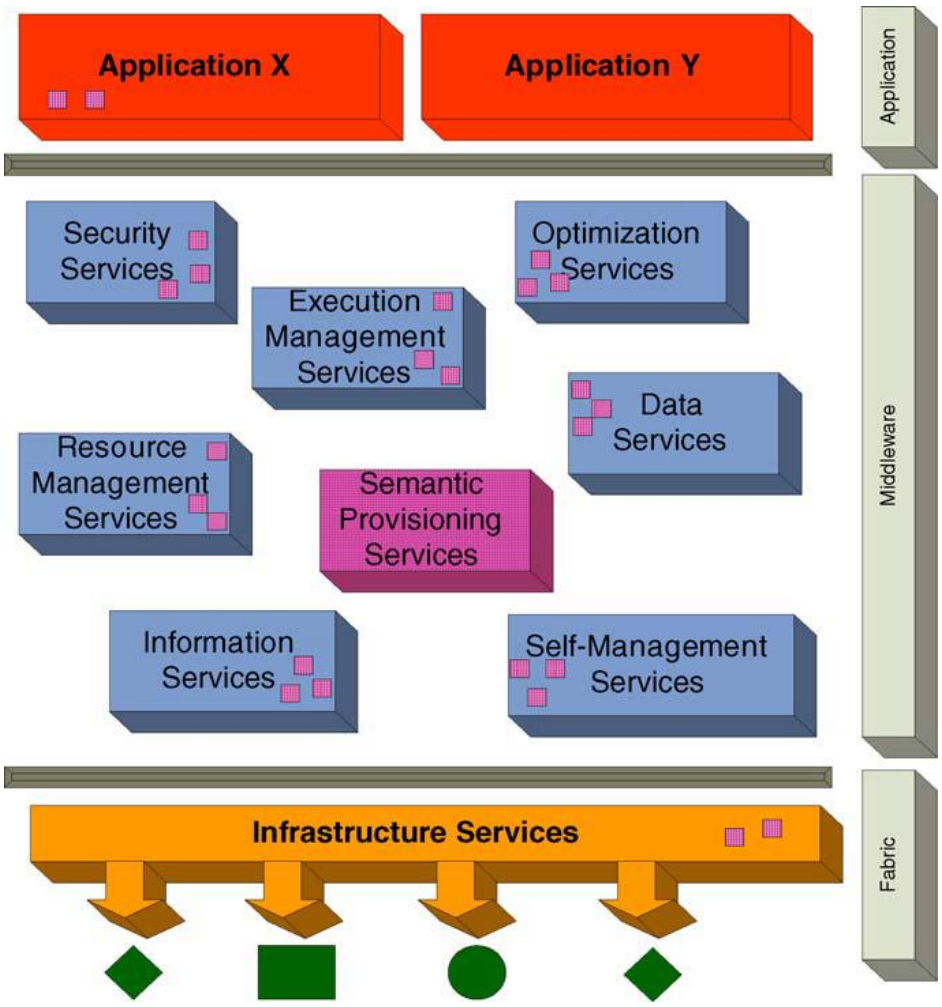

Figura 2.4 *Vista simplificada de la arquitectura propuesta por OGSA (azul) y los componentes adicionales de S-OGSA (rosa)*

S-OGSA propone extensiones a los actuales modelos *grid* para tratar con formas flexibles de metadatos explícitos. El componente central del modelo extendido es el "*Semantic Binding*", que relaciona entidades *grid* ("*Grid Entities*") y entidades de conocimiento ("*Knowledge Entities*") de diferentes maneras. Así mismo, también propone la reutilización de los mecanismos actuales de la *grid* para tratar los nuevos componentes del modelo como entidades *grid* y ofrecerlos como parte de las plataformas de servicios existentes en la *grid*.

En la siguiente figura 2.6 se muestra una versión extendida del modelo semántico de información *grid*. Los componentes en amarillo provienen de la *grid* tradicional, los componentes en gris aparecen en las plataformas de la web semántica y los componentes en rosa pertenecerían a la *grid* semántica.

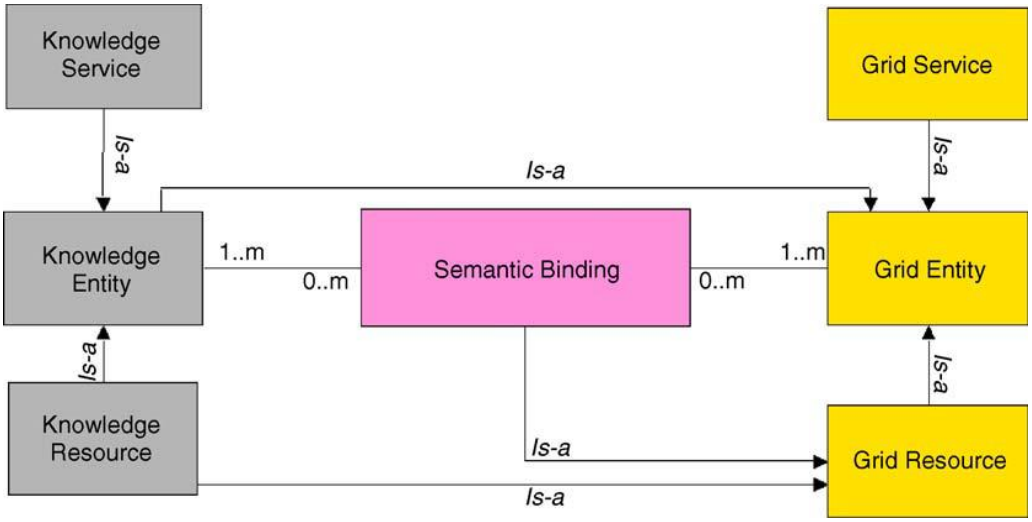

Figura 2.5 Entidades principales del modelo S-OGSA y sus relaciones

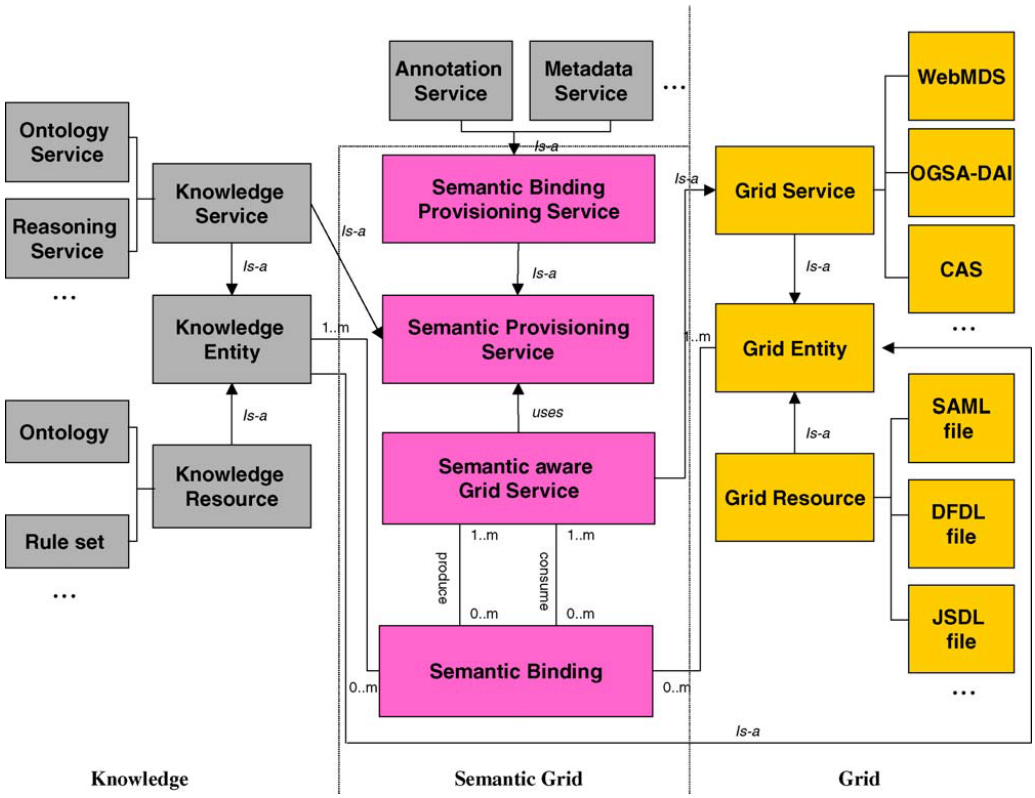

Figura 2.6 Modelo semántico de información grid (vista extendida)

Por último, S-OGSA propone habilitar el despliegue de su arquitectura en una aplicación *grid* semántica sobre una plataforma *grid* semántica mediante el

tratamiento de las entidades del modelo como recursos *grid* y el desarrollo de servicios *grid* semánticos para implementar las asociaciones semánticas o "*semantic bindings*" de todo tipo. Gracias al conjunto de especificaciones WSRF asociado a la implementación de los servicios *grid* que sirven para representar recursos *grid*, es posible utilizar las "*WS-ResourceProperties*" de los "*WS-Resources*" para asociar información sobre los metadatos codificados semánticamente de un recurso *grid*, de manera que esta pueda ser recuperada y consultada.

2.3 Modelos de representación de información grid y QoS

En esta sección se aborda el estado de la cuestión respecto a los modelos de representación de todo tipo de información relativa al dominio *grid*, prestando especial atención a los modelos relacionados con los recursos y las propiedades no funcionales o de calidad de servicio.

2.3.1 Modelos de representación de información sobre recursos grid

En los encuentros de la comunidad *grid* en el marco del OGF (*Open Grid Forum*) varios grupos relacionados con el modelado de datos e información de entornos *grid* plantean reiteradamente la necesidad de unificar criterios y desarrollos para intentar que los distintos modelos que propone cada grupo sean interoperables o incluso lleguen a converger en una misma especificación más genérica, dado que cada grupo enfoca el modelado desde sus propios casos de uso.

Existen iniciativas que, desde distintos enfoques y objetivos, han tratado aspectos relacionados con la representación de información sobre los recursos y el propio entorno de la *grid*: *GLUE Schema* (GLUE-WG) [25], *EGA Reference Model* (EGARM) [13], *OGSA* (OGSA-WG) [63], *Reference Model* (RM-WG) [72], *OGSA Basic Execution Services* (OGSA-BES-WG) [60], *Job Submission Description Language* (JSDL-WG) [46], *Information Modelling in OGSA* [45], *CIM based Grid Schema* (CGS-WG) [6], *GOM* [26], *Grid Ontology* (Parkin et al. 2006).

Del mismo modo, hay otros grupos de trabajo del OGF cuyos objetivos también se alinean con la planificación y/o gestión de recursos, el descubrimiento de los mismos, o el planteamiento de propiedades no funcionales para la *grid* y en algún caso aislado la medición de estas propiedades. En varios casos, la

actividad de estos grupos [59] es prácticamente nula o poco destacable: *OGSA Resource Selection Services* (OGSARSS-WG) [61], *Scheduling and Resource Management* (SRM) [74], *Grid Reliability and Robustness* (GRIDREL-RG) [31], *Trusted Computing* (TC-RG) [83], *Network Measurements Working Group* (NM-WG) [54], *Grid Benchmarking* (GB-RG) [29], *OGSA Resource Usage Service* (RUS-WG) [62], *Usage Record* (UR-WG) [85], *Grid Computing Environments* (GCERG) [30], *Semantic Grid* (SEM-RG) [76], GSMO [41].

GLUE *schema* 1.1 (OGF). El modelo utilizaba diagramas UML, XML Schema y Xpath 2.0 para representar estos recursos: *Computing Elements* (CE), *Storage Elements* (SE), *Clusters*, *Subclusters*, *Hosts*, *File Systems*, *Files*, *Main Memory*, *Processors*, *Operating System*, *Storage Devices*, *Network Adapter*, *Application Software*. No había referencia alguna a propiedades no funcionales como disponibilidad (*availability*), fiabilidad (*reliability*), rendimiento (*performance*), seguridad (exceptuando *SupportedSecurity* para SE) o confianza. Este modelo solamente representaba propiedades relacionables con calidad de servicio como las siguientes: *WorstResponseTime*, *EstimateResponseTime*, *TransferRate* y dos valores sobre sendas pruebas *benchmark* (*SI00*, *SF00*) para CE, además de *AccessTime*, *MaxIOCapacity* y *SupportedSecurity* para SE. El modelo no detallaba cómo medir estas propiedades y solamente especificaba para ellas si eran de tipo *integer* (*WorstResponseTime*, *EstimateResponseTime*, *TransferRate*, *AccessTime* y *MaxIOCapacity*), *float* (*SI00* y *SF00*) o *string* (*SupportedSecurity*). Este modelo se integraba en *Globus Toolkit 4.0.3* y había algunas herramientas de monitorización *grid* (*Ganglia* [17], *Hawkeye* [42]) que ofrecían la posibilidad de generar información de salida siguiendo este XML *Schema*.

GLUE *schema* 1.3 (OGF). Esta versión incluía una propiedad *UpTime* para la entidad *Host* y otra *Capability* para SE, pero seguía sin tener referencias a propiedades no funcionales como las mencionadas anteriormente. Posee una propiedad *Semantics* relacionada con la entidad *Servicio* que podría usarse de manera limitada para referenciar a propiedades no funcionales de dicha entidad. El modelo no detallaba cómo medir estas propiedades y solamente especificaba para ellas si eran de tipo *int32* (*UpTime*), *string* (*Capability*) o URI (*Semantics*). Este modelo se integraba en *Globus Toolkit 4* y algunas herramientas de monitorización *grid*.

GLUE *schema* 2.0 (OGF). Esta última versión incluía algunas propiedades (*OtherInfo*, *NetworkInfo*, *AggregatedBenchmarkInfo*, *SPECint2000*, *SPECfp2000*, *NetworkConnectivityIn*, *NetworkConnectivityOut*, *Authorization* y *Priority*) que podrían ser utilizadas para representar atributos

no funcionales, pero seguía sin ofrecer referencias a un esquema formal para las mismas. El modelo no detallaba cómo medir estas propiedades y solamente especificaba para ellas si eran de tipo *string* (*OtherInfo*, *NetworkInfo*, *AggregatedBenchmarkInfo* y *Authorization*), *int32* (*SPECint2000*, *SPECfp2000* y *Priority*) o booleanos (*NetworkConnectivityIn* y *NetworkConnectivityOut*). Este modelo ha tardado en integrarse en *Globus Toolkit 4* y algunas herramientas de monitorización *grid*.

OGSA-BES (OGF). Especificación para un *servicio básico de ejecución* (*Basic Execution Service* o BES) al que los clientes pueden enviar peticiones para iniciar, monitorizar y gestionar actividades de computación. El GLUE *schema* generalmente se considera una especialización del conjunto de atributos propuestos en el modelo BES. El modelo BES es extensible de tal manera que se podría conectar con una descripción de recurso usando el GLUE *schema*. El contenedor BES se corresponde con el servicio *GLUE Computing Service*, el *BES Contained Resource* se correspondería con el *Computing Element* y los *BES Basic Resources* se mapearían con el *GLUE Computing Resource*. El atributo BES *isAcceptingNewActivities* podría ser considerado como una propiedad no funcional relacionada con la disponibilidad o *availability*, pero no hay ninguna otra referencia a propiedades no funcionales como las mencionadas anteriormente. Este modelo incluye un atributo *BESextension* que se podría utilizar para representar algunas propiedades no funcionales, pero no existe referencia alguna a un esquema formal para ellas. El modelo no especifica cómo medir propiedades no funcionales.

JSDL (OGF). Este modelo utiliza XML Schema para describir requisitos relativos a recursos por parte de los trabajos *grid*, pero no hay referencia explícita a requisitos no funcionales o propiedades sobre los recursos que se piden, aparte de *IndividualNetworkBandwidth*, que podría ser utilizado para calcular algunas de ellas. Soporta una descripción simple de los requisitos del recurso: no utiliza un lenguaje exhaustivo de requisitos sobre recursos, evita descripciones explícitas jerárquicas o heterogéneas y puede ser extendido con otros elementos para realizar descripciones más ricas o más abstractas. El modelo no explica cómo medir propiedades no funcionales. Está influenciado por una serie de estándares o propuestas de estándar como DRMAA (*Distributed Resource Management Application API Specification 1.0*), CIM (*Common Information Model 2.9.0*) y POSIX (*Portable Operating System Interface 3*). JSDL 1.0 provee el vocabulario central para describir el envío de un trabajo a los entornos *grid* y está influenciado por una serie de sistemas como *Condor*, *Globus Toolkit*, LSF (*Load Sharing Facility*), PBS (*Portable*

Batch System), SGE (*SunGridEngine*) y UNICORE (*Uniform Interface to Computing Resources*).

DMTF*'s Common Information Model* (CIM). El modelo contenedor describe los objetos gestionados y sus relaciones para la definición de entornos de ejecución para actividades en una *grid*. El modelo BES define el servicio que puede actuar sobre este modelo contenedor. El esquema CIMv2.14 es el fundamento para el desarrollo de estos dos modelos. Ambos modelos se fusionan en la versión CIMv2.15. El modelo no representa propiedades no funcionales sobre recursos *grid* ni explica cómo medirlas.

EGA *Reference Model*. La GME o *Grid Management Entity* es aquella entidad lógica que gestiona los componentes *grid*, sus relaciones y sus ciclos de vida. El modelo menciona algunos típicos objetivos de nivel de servicio o SLO (*Service Level Objectives*) de negocio que formarían parte de un acuerdo de nivel de servicio o SLA (*Service Level Agreement*) para un determinado servicio, pero no explica nada más: requisitos de tiempo de respuesta, requisitos de capacidad, volumen o requisitos de rendimiento, requisitos de disponibilidad, requisitos de fiabilidad, requisitos de recuperación, requisitos de mantenimiento, requisitos de seguridad, requisitos de seguridad. La gestión de nivel de servicio o SLM puede incluir la gestión del rendimiento, el escalado, la disponibilidad, la seguridad, etcétera. El modelo solo da ligeramente algunas ideas acerca de cómo medir en general estas propiedades no funcionales. *Requisitos de disponibilidad*: disponibilidad total (incluyendo tanto apagón planificado y no planificado) expresada como porcentaje o como tiempo real por mes o año, por ejemplo. *Requisitos de fiabilidad*: alguna medida de la fiabilidad, como MTTF (tiempo medio hasta el fallo). *Requisitos de recuperación*: alguna medida de recuperación, como MTTR (tiempo medio de recuperación) o estado deseado post-recuperación. *Requisitos de mantenibilidad*: alguna medida de la capacidad de mantenimiento, como número máximo de apagones requerido para actualizaciones, etcétera. EGA RM referencia otros estándares o trabajos, como por ejemplo CIM o OGSA. EGA podría unificar sus modelos o al menos hacer que fueran más coherentes entre sí.

RM (OGF). Este modelo pretendía utilizar el modelo de referencia EGA y el glosario de OGSA como puntos de partida. Este enfoque trataba de definir las formas de representación y serialización del modelo, incluyendo el vocabulario RDF/RDFS/OWL, SML, XML Schema, Java, etcétera (previsto para el RM v3.0). No hay trabajo significativo realizado.

UR (OGF). A fin de que los recursos sean compartidos, los sitios deben ser capaces de intercambiar datos de uso y datos básicos de contabilidad en un

formato común. Este grupo de trabajo propone definir un registro de uso común. El modelo describe algunas propiedades, sobre la utilización de los recursos por parte de los trabajos, que podrían ser útiles para calcular algunas propiedades no funcionales: *WallDuration*, *Network*, *TimeDuration* o *Metric*. El modelo da algunas ideas acerca de cómo medir algunas propiedades que podrían servir para el cálculo de propiedades no funcionales, pero no hay ninguna propuesta sobre la forma de medir esas propiedades no funcionales. Este modelo describe las propiedades de uso y la representación XML *infoset* que permiten que la información de uso sea compartida en formato agregado. Un registro de uso agregado acumula información de uso a nivel de trabajo. Las propiedades agregadas especificadas pretenden dar cabida a la agregación flexible o la descomposición a otros niveles, tanto desde la perspectiva del consumidor y como desde la del proveedor de recursos. También se proporcionan extensiones para definiciones personalizadas. Este esfuerzo se concentra en la definición común de formato y no se ocupa de cómo se van a utilizar los datos de uso agregados, la interoperabilidad de datos con otro formato de almacenamiento, así como el mecanismo de comunicación y la seguridad.

GOM *(K-Wf Grid)*. Hay algunas referencias a QoS y SLA en este conjunto de ontologías que podrían ser útiles, pero no están bien definidas. Se pueden reutilizar algunas referencias sobre la clasificación de los recursos. El concepto "*Reliability*" aparece en la ontología de aplicación (*GOM application ontology*) como una subclase de "*ComparisonCriteria=QoSAttribute*", pero no está definida y no tiene métricas. Los proyectos *Askalon* y *K-WfGrid* dan soporte a métricas y conceptos comunes sobre el rendimiento. El conjunto de herramientas *Askalon* soporta un lenguaje de flujo de trabajo estructurado y flujos de trabajo de aplicaciones ejecutables, mientras que el proyecto *K-WfGrid* soporta un lenguaje de flujo de trabajo de red de Petri y los flujos de trabajo de los servicios web. Las herramientas de *Askalon* se pueden analizar para ver cómo miden el rendimiento.

GRO (IMG). Se trata de un modelo basado en una ontología con algunas referencias a conceptos de QoS y SLA que podrían ser útiles, pero no están bien definidas.

A continuación se muestran una tabla en dos partes (tabla 2.1 y tabla 2.2) que compara la capacidad de representación de propiedades no funcionales de los diferentes modelos de información sobre recursos *grid* analizados, atendiendo a criterios de exhaustividad, sistema de medida, uso de estándares/tecnologías y aplicaciones de soporte:

Modelo de información	Exhaustividad	Sistema de medida	Estándares / Tecnologías	Aplicaciones de soporte
GLUE 1.1	Buena taxonomía de recursos, pero no suficiente. No hay referencias a propiedades no funcionales, excepto para algunos valores de QoS aislados.	Ninguno. Solo define tipos de datos básicos (int, float, string) para algunas propiedades.	Diagramas UML, XML schema, Xpath 2.0	Integrado en GT 4.0.3. Ganglia y Hawkeye proveen una salida usando este XML schema.
GLUE 1.3	Taxonomía de recursos extendida. Hay nuevos atributos de QoS, pero nada más.	Ninguno. Solo define más tipos de datos básicos (int, float, string) para nuevas propiedades.	Diagramas UML, XML schema, Xpath 2.0	Se puede integrar en GT 4.0.x y algunas herramientas de monitorización grid.
GLUE 2.0	Algunos nuevos atributos podrían ser usados para representar algunas propiedades no funcionales, pero sigue sin haber referencia a un esquema formal.	Ninguno. Solo define más tipos de datos básicos (int, float, string) para nuevas propiedades.	Diagramas UML, XML schema, Xpath 2.0	Se puede integrar en GT 4.0.x y algunas herramientas de monitorización grid.
OGSA-BES	No hay taxonomía de recursos. Algunos nuevos atributos podrían ser usados para representar algunas propiedades no funcionales, pero sigue sin haber referencia a un esquema formal.	Ninguno	Diagramas UML, PortTypes. GLUE schema se considera una especialización de BES.	Ninguna
JSDL	No hay taxonomía de recursos. Describe requisitos sobre recursos, pero no hay referencia significativa a requisitos o propiedades no funcionales.	Ninguno	XML schema. Influenciado por DRMAA, CIM y POSIX.	Vocabulario central informado por Condor, GT4, LSF, PBS, SGE, y Unicore.
CIM	No hay taxonomía de recursos. No hay una descripción formal de propiedades no funcionales.	Ninguno	Diagramas UML	Ninguna

Tabla 2.1 Modelos de representación de información sobre recursos grid (parte 1)

Modelo de información	Exhaustividad	Sistema de medida	Estándares / Tecnologías	Aplicaciones de soporte
UR	No hay taxonomía de recursos. Algunos atributos sobre cómo se utilizan los recursos podrían ser útiles.	Ofrece ideas sobre cómo medir propiedades, pero no una propuesta sobre cómo hacerlo en la práctica.	Propiedades relativas a utilización de los recursos. Representación conjunto de información XML (XML infoset).	Ninguna
EGA	No hay taxonomía de recursos. No hay una descripción formal de propiedades no funcionales.	El modelo da ligeras ideas sobre cómo medir algunas propiedades no funcionales.	Referencias a CIM y OGSA. Podría unificar modelos o hacerlos más coherentes.	Ninguna
RM	No hay suficiente documentación. Usa el modelo de referencia EGA y el glosario de OGSA como puntos de partida.	No hay suficiente trabajo de referencia.	Este enfoque intenta utilizar OWL, SML y XML Schemas.	Ninguna.
QoSOnt2	No hay taxonomía de recursos. Sin referencias al dominio grid. Taxonomía útil de conceptos QoS y métricas.	Ninguno.	OWL.	Ninguna.
GOM	Útil taxonomía de recursos y servicios. Representación bastante completa de la propiedad rendimiento (performance).	Centrado básicamente en la propiedad rendimiento (performance).	OWL.	Herramientas sobre rendimiento del framework Askalon y el proyecto K-Wf Grid.
GRO	Útil ontología fundacional para la grid. Taxonomía de recursos básica. Algunas referencias a conceptos QoS.	Ninguno.	OWL.	Ninguna.

Tabla 2.2 Modelos de representación de información sobre recursos grid (parte 2)

2.3.2 Modelos de representación de información sobre propiedades no funcionales en el dominio grid

En el contexto de la representación y uso de la calidad de servicio en general, algunos de los trabajos encontrados describen aspectos relacionados con el descubrimiento de recursos a través de propiedades no funcionales. *MOQ* [52] propone un marco general y más bien filosófico para afrontar cualquier problema de concreción de aspectos cualitativos abstractos al mundo real; sin embargo, a pesar de que aporta información sobre el dominio de calidad de servicio o QoS en relación a cómo normalizar o concretar la evaluación de aspectos cualitativos, no se especifica un modelo. Sánchez-Macián, Bellido y Pastor (2005) tratan de relacionar QoS con *Quality of Experience* (*QoE*) y *Quality of Business* (*QoBiz*). Tosic, Pagurek y Patel (2003) enumeran algunos requisitos que se deben tener en cuenta a la hora de desarrollar una ontología para modelar aspectos de QoS. *FIPAQoS* [15] y *WS-QoS* [91] son ontologías sin implementación OWL, a diferencia de *OWL-QoS* [69] (anteriormente *DAML-QoS*), *OWL-QoS* de Maximilien y Singh (2004) y *QoSOnt* de Dobson, Lock y Sommerville (2005), que sí cumplen con la especificación OWL [68]. QoSOnt es una ontología OWL para QoS que está especialmente enfocada a los servicios web y que fue desarrollada por la Universidad de Lancaster como parte del proyecto DIRC [11] y actualizada en el proyecto SeCSE [75]. Esta última ontología ofrece una útil taxonomía de conceptos y métricas sobre QoS.

Torres (2010) realiza un excelente resumen sobre los estándares para la gestión de QoS en la *grid*. Por su interés para contextualizar la presente tesis y dada su excelente redacción, se recomienda su lectura:

> «En cuanto a la QoS, la actividad del OGF está orientada a estandarizar los resultados que han sido obtenidos en varios proyectos, como GRIDCC [35], G-QoSM (Al-Ali et al. 2002), BEinGRID [4], SmartLM [79], GridEcon [36], SORMA [80], BREIN [5], SLA@SOI [78], AssessGrid [3] y RESERVOIR [73], y a completar la especificación WS-Agreement, para reflejar las necesidades del Grid.
>
> »[…] WS-Agreement consiste en un lenguaje y un protocolo, que han sido diseñados con el objetivo de exponer la capacidad de los proveedores de servicio, y sobre la base de la oferta inicial, crear acuerdos entre proveedores y consumidores, que luego pueden ser monitorizados, en tiempo de ejecución, para verificar su cumplimiento.» (Torres 2010: 34)

La QoS en la *grid* resulta más compleja de evaluar que en otras arquitecturas orientadas a servicios porque hay que tener en cuenta la QoS de componentes de diferente nivel (red, *middleware*, aplicación, etcétera) y porque se agrupan recursos de varias organizaciones administrativas diferentes, haciendo todavía más difícil gestionar los SLA o *Service Level Agreements*.

2.3.3 Análisis comparativo de los modelos de representación de información

Por un lado, los modelos de información *grid* existentes no usan el mismo lenguaje para representar los mismos conceptos sobre recursos, aunque varios grupos de la comunidad *grid* han detectado la necesidad de integrar sus modelos y hacerlos interoperables. Además, existe una falta de representación de propiedades no funcionales o de calidad de servicio sobre los recursos. En el mejor de los casos, algunos modelos especifican un rango de valores para medir y comparar el rendimiento de los recursos *grid*, pero faltan muchos detalles en general. Algunos proveedores y consumidores de información *grid* desarrollados específicamente para algunos proyectos usan técnicas aisladas para tratar el rendimiento de los recursos y su disponibilidad en un momento dado.

Por otro lado, hasta la fecha, todas las iniciativas mencionadas anteriormente relacionadas con la calidad de servicio en general poseen ciertas limitaciones que hacen que sea difícil adoptarlas en entornos *grid* de producción. En muchos casos no son extensibles, por lo que no se pueden adaptar a la *grid*. En otros casos, no son completas, centrándose sólo en una propiedad no funcional y aportando una solución ad-hoc a un problema o dominio de aplicación concreto.

Además, en el caso concreto de la *grid*, las propuestas actuales de integración de información relativa a propiedades no funcionales o de calidad de servicio son ad-hoc para proyectos concretos y, en general, o no están esquematizadas, o no se ha llevado a cabo un análisis detallado, o no plantean un modelo común para los recursos y sus propiedades.

Es necesario señalar que el *middleware grid* permite especificar varios aspectos de la ejecución de un trabajo. Para ello, existen especificaciones estándares, como el Lenguaje de Descripción de Envío de Trabajos (*Job Submission Description Language* o JSDL), que se podrían extender con requisitos de QoS. Los esfuerzos hechos en el marco del OGF para estandarizar *WS-Agreements*, una especificación de servicios web para la gestión de SLA, siguen aún en desarrollo y no tenemos conocimiento de que exista un *middleware grid* que proporcione una implementación de *WS-Agreements*.

Por lo tanto, parece razonable la necesidad de nuevas propuestas de modelos unificados que facilite la labor de representar propiedades no funcionales en el dominio *grid*, sin olvidar la oportunidad de impulsar con estas iniciativas el paradigma de la *grid* semántica.

2.4 Proveedores y consumidores de información grid

El *middleware* de la *grid* es abundante y hay varias herramientas o aplicaciones que se pueden utilizar como proveedores de información *grid* sobre los recursos que se encuentran desplegados en un sistema *grid* y otras tantas aplicaciones potenciales consumidoras de dicha información.

Aunque los proveedores de información por excelencia son las herramientas de monitorización de recursos como *Ganglia* [17], *Nagios* [53], *GridICE* (Andreozzi et al. 2005), *Hawkeye* [42], *CluMon* [7], *INCA* [44] o *MonaLISA* [51], existen muchas otras aplicaciones que pueden ser usadas con éste mismo propósito: Globus *middleware* (GRAM, MDS, RFT, RLS, CAS), herramientas de monitorización ad-hoc de proyectos de investigación como *GEMLCA-GMT* (Juhasz, Kacsuk y Kranzlmüller 2004), *SCALEA-G* (Truong y Fahringer 2004) y *GRASP* [28], gestores locales de colas de trabajos como *Condor* [8], *PBS/Torque* [70], *SGE* [82], *LSF* [48] y *Maui* [49], así como cualquier servicio WSRF que publique propiedades de los recursos en los sistemas de información *grid*.

Las aplicaciones que podríamos considerar como consumidores de información *grid* son, por ejemplo, *GridWay meta-scheduler* [39], *GridARM Askalon's Grid Resource Management System* [33], *LCG/g-LITE broker* [47], *Nimrod-G Grid Resource Broker* [57], *EUROGRID/GRIP Resource Broker* [25], *Storage Resource Broker* [26], *NGS Resource Broker* [56], *Moab Grid Scheduler (aka Silver)* [50], *EGEE Workload Manager Service (WMS)* [14], *GRMS (Grid(Lab) Resource Management)* [40], *Gridbus Grid Service Broker* [34], *Advanced Resource Connector (NorduGrid's ARC)* [1], así como cualquier otro *middleware* de la *grid*: *WebMDS* [88], interfaces de usuario *grid* (portales *grid* como *P-GRADE* [71]), herramientas *grid* basadas en *CSF* [10], *WebCom-G* [87], *GPT* [27], *middleware HPC4U* [43] o cualquier cliente WSRF que busque y seleccione recursos.

Tradicionalmente, la mayoría de las herramientas o aplicaciones que realizan funciones de planificación, selección y/o gestión de recursos en la *grid* realizan descubrimiento de recursos basado en propiedades funcionales, como por ejemplo características del procesador, tamaño de la memoria, espacio de almacenamiento disponible, *software* que se encuentra instalado, tiempo de respuesta a las peticiones de servicio, etcétera, ya que esta es básicamente la información que ofrecen los proveedores de información *grid* comentados anteriormente. Algunas de estas iniciativas, como por ejemplo *GridWay* y *GridARM*, tratan de incorporar en sus algoritmos de selección criterios de rendimiento y disponibilidad, pero utilizan enfoques aislados y ad-hoc.

En la mayoría de los casos analizados, la utilización de propiedades no funcionales en la selección de recursos está relacionada con desarrollos circunscritos a las necesidades de un proyecto de investigación concreto o los experimentos de tesis doctorales, siendo muy escasa su adopción por parte del *middleware grid* que se utiliza como estándar *de facto*.

Por lo tanto, todos estos desarrollos a medida sugieren la necesidad de proponer nuevos enfoques y desarrollos tanto de modelos unificados como de metodologías que facilite la labor de integrar información sobre propiedades no funcionales de los recursos en el *middleware* estándar *de facto* de la *grid*, tanto el considerado proveedor como consumidor de información, de manera que se vayan dando avances significativos que impulsen la semantización formal de la *grid*.

2.5 Utilización de propiedades no funcionales en la selección de recursos grid

Esta sección presenta una revisión del estado de la cuestión respecto a la selección y asignación de recursos en entornos de computación *grid*, poniendo el foco en aquellas iniciativas que tratan de utilizar propiedades no funcionales o de calidad de servicio. Además, también se analizan los distintos niveles de planificación existentes en la *grid*.

Torres (2010) realiza un excelente resumen sobre el estado de la cuestión en esta área. Por su interés para contextualizar la presente tesis y dada su excelente redacción, se recomienda su lectura.

2.5.1 Asignación de recursos en entornos de computación distribuida

Desde Fernández-Baca (1989) se sabe que el problema de seleccionar el conjunto de recursos que minimice el coste computacional de ejecutar un trabajo determinado entre un grupo de recursos distribuidos es NP-completo, por lo que la mayor parte de las propuestas en esta área se han centrado en el desarrollo de algoritmos efectivos de selección de recursos.

Siguiendo el enfoque de la reserva anticipada nos encontramos la propuesta GARA de Foster et al. (1999b), apoyada posteriormente en Czajkowski, Foster y Kesselman (2005), así como en Singh, Kesselman y Deelman (2007).

Cabe mencionar que en Foster et al. (2004) también se exploró la posibilidad de utilizar información del estado del sistema en tiempo de ejecución para

mejorar la selección de recursos. Este trabajo logró el consenso en torno a que era necesario utilizar información real de monitorización de los sistemas.

Al-Ali et al. (2002), Al-Ali et al. (2004b), Lu, Subrata y Zomaya (2007), Subrata, Zomaya y Landfeldt (2008), Middleton et al. (2007), Hirschey (2005) son ejemplos de propuestas que abordan la selección de recursos en base a requisitos de QoS obtenidos mediante monitorización del sistema. Doulamis et al. (2007) presentaron un enfoque consistente en utilizar un modelo no lineal de la complejidad computacional de una aplicación para predecir la carga que supone cada trabajo antes de que sea enviado a la grid. Lu, Subrata y Zomaya (2007) se trazaron como objetivo encontrar el conjunto de recursos que minimizan el tiempo que un trabajo consume en el sistema. Subrata, Zomaya y Landfeldt (2008) propusieron un algoritmo basado en la teoría de juegos cooperativos para seleccionar recursos *grid* en base a requerimientos de QoS. Middleton et al. (2007) presentan un sistema que permite reservar recursos, de forma anticipada, para aplicaciones de simulación médica en una grid comercial. Zheng, Tham y Veeravalli (2008) presentan tres algoritmos para encontrar la distribución de la carga que minimice el tiempo de respuesta o el coste computacional en sistemas grid. Stuer, Vanmechelen y Broeckhove (2007) presentan un modelo económico que trata los recursos *grid* como bienes sustituibles. Hughes y Hillman (2006) presentan un enfoque centrado en el componente humano para predecir el comportamiento de servicios compuestos en base a la información de QoS de los componentes individuales. Por último, el proyecto BOINC (Anderson 2004) presenta una plataforma software que permite hacer computación intensiva utilizando recursos voluntarios. Una explicación más detallada de cada propuesta puede encontrarse en Torres (2010).

> «Un número considerable de estos proyectos de investigación han estado enfocados a resolver dos grandes retos: 1) encontrar el conjunto de recursos que minimizan el tiempo que un trabajo permanece en el sistema. En este sentido hay tres grandes objetivos: minimizar el tiempo de ejecución, minimizar la varianza del tiempo de ejecución (Completion-Time Variance o CTV), y minimizar la varianza del tiempo de espera (Waiting-Time Variance o WTV); y 2) proporcionar, tanto de forma anticipada como bajo demanda, los recursos que garanticen un nivel mínimo de disponibilidad para cumplir con los requerimientos del trabajo en cuestión.» (Torres 2010: 29)

2.5.2 Selección de recursos en un sistema distribuido

El trabajo de Fernández-Baca (1989) mencionado anteriormente demuestra que la complejidad computacional del problema general de selección de recursos para minimizar el coste total de ejecución y comunicación es NP-completa, a la vez que prepara las bases para la discusión de otros problemas

más específicos, sobre los que Torres (2010) se apoya para realizar su análisis sobre los problemas específicos del dominio grid.

> «Mientras que en la asignación de recursos tradicional en los sistemas multiprocesadores los indicadores estáticos tienen más peso que los indicadores dinámicos, en los sistemas Grid ocurre que los indicadores dinámicos predominan sobre los estáticos (Rajasekaran y Reif 2007). En general, los sistemas de información Grid, como WS-MDS, MDS y BDII (Berkeley Database Information Index), son servicios con un alto nivel de centralización, que proporcionan información acerca de los indicadores estáticos y dinámicos. A esto se contrapone el hecho de que para conseguir la escalabilidad necesaria, las organizaciones virtuales se ven obligadas a utilizar tiempos de refresco bastante altos. En consecuencia, es muy improbable que en el Grid se puedan conseguir niveles de calidad de servicio como los que se obtienen en un entorno local controlado, como un clúster.» (Torres 2010: 43)

Respecto a la problemática del alto nivel de centralización de los sistemas de información *grid*, Torres (2010) remarca que básicamente hay dos formas de afrontar esta dificultad: a) realizar una reserva anticipada de recursos para un trabajo, cuestión que tiene su complejidad en el dominio *grid*; b) predecir las prestaciones de las aplicaciones los recursos en la *grid*.

> «De este planteamiento se deducen dos conclusiones, por un lado, es imprescindible contar con información del estado real de los recursos para realizar la asignación. En consecuencia, cualquier algoritmo debe contar con información actualizada de los recursos, y para ello debe utilizar un sistema de monitorización que además de eficiente, sea efectivo.
>
> »Por otro lado, se hace necesario contar con un mecanismo preciso para evaluar los requerimientos de un trabajo antes de que éste sea enviado al Grid. De los mecanismos desarrollados en trabajos anteriores, la utilización de casos de uso significativos en combinación con resultados históricos parece ser la forma más factible para conseguir estas estimaciones.» (ibíd.)

2.5.3 Modelo computacional de asignación de recursos grid

El *middleware grid* actual está preparado para soportar, principalmente, aplicaciones de alta productividad (*High-Throughput Computing* o HTC) y aplicaciones de alto rendimiento (*High-Performance Computing* o HPC), por lo que los modelo de asignación de recursos en la *grid* tiene puntos en común con los modelos de los entornos de supercomputación o altas prestaciones, si bien su naturaleza y arquitectura distribuida generan ciertas diferencias.

Torres (2010) realiza un buen análisis de los requisitos necesarios para construir un modelo de asignación de recursos en la grid, haciendo referencia a la definición de un grupo de indicadores estáticos y dinámicos que caractericen al recurso en cuestión y que fueron mencionados en Al-Ali et al. (2004a).

> «[…] los indicadores estáticos no constituyen un problema para la monitorización del Grid, y que por tanto, los indicadores estáticos pueden ser colectados, evaluados y distribuidos utilizando los sistemas de información Grid combinados con unos pocos servicios diseñados para este fin. […]
>
> »En cambio, los indicadores dinámicos representan un reto para la monitorización del Grid, que no puede ser resuelto utilizando el middleware Grid actual. Estos indicadores cambian con demasiada frecuencia, por lo que no es posible recogerlos en los nodos de un sitio y propagarlos, de forma eficiente, al resto de los sitios. Más aún, el nivel de centralización de los sistemas de planificación Grid puede hacer que se conviertan en cuello de botella para otros servicios. La forma de evitar que esto suceda consiste en disminuir la carga de los sistemas de planificación, y para ello los administradores de las VO se ven obligados a aumentar los tiempos de refresco de la información.» (Torres 2010: 46)

2.5.4 Planificación de trabajos en el dominio grid

Prajapati y Shah (2014) realizan un buen análisis y resumen de la problematica relacionada con la planificación de trabajos en entornos de computación *grid*.

En la *grid*, los recursos suelen ser autónomos y el planificador *grid* de alto nivel no tiene control total de los recursos de cada nodo ni puede violar sus políticas locales de gestión de recursos, por lo que generalmente no se puede hacer una asignación de recursos a alto nivel.

Generalmente, cada nodo de la *grid* suele tener su propio planificador *grid* de bajo nivel y el mismo recurso local puede formar parte de más de una organización virtual o VO. El planificador *grid* de bajo nivel de cada nodo planifica y ejecuta cada trabajo individual independientemente de la aplicación *grid* a la que pertenece.

El planificador *grid* de alto nivel, también denominado planificador global, meta-planificador, super-planificador o *broker*, planifica los trabajos individuales de las aplicaciones *grid* sobre los nodos disponibles tratando de satisfacer los requisitos especificados para dichos trabajos, delegando la asignación de recursos final en el planificador local de cada nodo.

Las herramientas de monitorización de infraestructuras *grid* ofrecen valiosa información sobre el estado de sus nodos y recursos para que los planificadores *grid* de alto nivel puedan tomar sus decisiones. Asimismo, las herramientas de monitorización de trabajos permiten realizar un seguimiento del estado de ejecución de los mismos. Por ello, si se dispone de los componentes adecuados, la computación *grid* puede hacer uso de los recursos en función de su disponibilidad, capacidad, rendimiento, coste y demás requisitos de calidad de servicio de los usuarios.

En la computación *grid*, el problema de la planificación implica a una serie de tareas y un conjunto de recursos. El objetivo general sería generar una asignación óptima de las tareas sobre los recursos, pero este es un problema NP-completo, tal y como ya se ha mencionado anteriormente al hablar de los entornos distribuidos. Por lo tanto, es necesario utilizar la heurística para resolver este tipo de problemas en tiempo polinomial.

La planificación de grafos o flujos de trabajos es un caso especial de la planificación de trabajos individuales en el que existen dependencias entre los mismos.

Prajapati y Shah (2014) también mencionan que los algoritmos de planificación *grid* también pueden ser evaluados utilizando herramientas de simulación *grid*, teniendo en cuenta que la precisión de los resultados de la simulación dependerá de la precisión con la que el simulador imite las entidades del sistema *grid* real durante la simulación.

Por último, tal y como plantean Prajapati y Shah (2014), la decisión de realizar evaluaciones utlizando simuladores en lugar de sistemas reales suele estar justificada cuando no está disponible un entorno *grid* real con usuarios o investigadores, o cuando no es factible en la práctica o económicamente recomendable desplegar una *grid* real simplemente para evaluar el rendimiento de nuevas propuestas de algoritmos. De hecho, cuando los resultados de los experimentos son necesarios muy rápidamente y los investigadores no tienen tiempo suficiente para adaptarse a trabajar con un etorno *grid* real concreto, el estudio de experimentos basado en la simulación es la mejor forma de evaluar los conceptos propuestos. Igualmente, cuando cambiar la configuración de un entorno *grid* real es difícil o resulta casi imposible alcanzar la preparación de ciertos escenarios, la evaluación de algoritmos de planificación por medio de la simulación puede proporcionar resultados más rápidamente e incluso para escenarios prácticamente no factibles.

2.5.5 Conclusiones sobre la selección de recursos grid

Existen varios sistemas que permiten vincular trabajos con recursos distribuidos y planificar el orden de ejecución de los mismos en base a sus requerimientos de QoS. Las características de la *grid* hacen que este problema sea de difícil solución, de ahí que existan diversos enfoques en el estado de la cuestión.

En cuanto a los mecanismos de planificación de trabajos y de administración de recursos existentes en las infraestructuras *grid*, a pesar de los avances de los últimos años, el soporte para las propiedades no funcionales o de QoS en entornos de computación *grid* es aún muy limitado y, hasta el momento, no existe una solución definitiva para el problema.

Una vez más, todos los desarrollos a medida analizados sugieren la necesidad de proponer nuevos enfoques y desarrollos tanto de modelos unificados como de metodologías que facilite la labor de integrar información sobre propiedades no funcionales de los recursos en el *middleware* estándar *de facto* de la *grid*, tanto el considerado proveedor como consumidor de información, de manera que se vayan dando avances significativos que impulsen la semantización formal de la *grid*.

Los indicadores de QoS cuantitativos caracterizan a los recursos o servicios *grid* en cuanto a prestaciones y disponibilidad, pudiendo ser utilizados para diferenciarlos. Una posible solución consiste en evaluar asíncronamente los indicadores QoS y de carga del sistema para publicarlos posteriormente en los sistemas de información *grid* de manera que puedan ser consumidos fácilmente.

Tras el estudio realizado, también se detecta entre las propuestas analizadas un alto interés por los indicadores dinámicos y menos pruebas con indicadores estáticos o basados en datos históricos, como por ejemplo la fiabilidad.

Por último, cabe destacar también la imposibilidad de reproducir en la presente tesis los escenarios de experimentación de los trabajos mencionados, así como el acceso a los desarrollos e infraestructuras en los que se basan dichas investigaciones, por lo que en el siguiente apartado se hará un estudio de las alternativas existentes para crear un entorno de pruebas grid en la presente tesis.

En cualquier caso, una vez finalizada la tesis se plantea como línea de trabajo futuro contactar con los investigadores responsables de dichos desarrollos e infraestructuras para explorar la posibilidad de contribuir en sus desarrollos e investigaciones.

2.6 Entorno de pruebas grid

Tanto Leal (2010) como Prajapati y Shah (2015) realizan un excelente resumen sobre el estado de la cuestión en esta área, por lo que se recomienda su lectura. En primer lugar realizan un repaso a las herramientas para el estudio de sistemas grid, entre las que se incluyen la utilización de modelos matemáticos

teóricos, la simulación o la emulación de sistemas reales, desgranando las ventajas e inconvenientes de cada una de las alternativas mencionadas:

> «Como hemos visto, el modelo matemático tiende a idealizar la realidad, las condiciones de los sistemas reales no son repetibles ni controlables y los emuladores presentan más inconvenientes que ventajas frente a los simuladores. Por lo tanto, la solución pasaría por utilizar un simulador o directamente un sistema real.» (Leal 2010: 39)

En segundo lugar, justifican la necesidad y validez de la simulación como herramienta para el estudio de sistemas grid, enumerando los motivos más importantes enunciados en Murshed y Buyya (2002) por los que es preferible utilizar un simulador que una grid real.

> «Por lo tanto, emplear un simulador es bastante aconsejable, sobre todo si lo que se pretende es comparar el comportamiento de un algoritmo frente a otros algoritmos existentes para así poder mejorarlo. Además, podremos obtener resultados en un tiempo menor, con menos esfuerzo y por un coste económico inferior que empleando un Grid real.» (Leal 2010: 41)

Por un lado, Leal (2010) compara distintas herramientas para la simulación de entornos grid, como son Bricks (Aida et al. 2000), GridG [37], GangSim [18] derivado del sistema de monitorización Ganglia [17], OptorSim [67], GridSim [38] y SimGrid [77]. Por otro lado, Prajapati y Shah (2015), además de analizar detalladamente estas y otras herramientas que han ido surgiendo con los años, ofrecen una completa guía para seleccionar la herramienta de simulación que mejor se adapte a las necesidades de cada investigador. Este trabajo permite al investigador caracterizar su investigación mediante una serie de cuestionarios y utilizar varias tablas con hasta 31 criterios de comparación para realizar su selección. Además, ofrecen una serie de recomendaciones y reflexiones técnicas basadas en su experiencia con dos de las herramientas más utilizadas por los investigadores hasta la fecha, GridSim y SimGrid.

Además de tener en cuenta las premisas y restricciones planteadas al inicio de la tesis, así como la necesidad de utilizar la simulación tal y como plantean tanto Leal (2010) como Prajapati y Shah (2015), dado que el objetivo que se persigue en la evaluación de la presente tesis es realizar una prueba de concepto formal para evaluar qué efecto tiene la utilización de propiedades no funcionales en el subalgoritmo de selección de recursos de un algoritmo de planificación cualquiera a nivel de nodo *grid*, pudiendo considerar este último como un gestor de recursos local o LRM (*Local Resource Manager*), y que no se persigue abordar el problema de la planificación de trabajos en la *grid*, se ha decidido utilizar un entorno de simulación contrastado que posea un modelo de pruebas completo con al menos las siguientes características para minimizar los desarrollos necesarios y poder centrar los esfuerzos en el foco de la evaluación:

- Disponer de, al menos, un algoritmos de planificación funcional completamente implementado, independientemente de cuál sea.
- Disponer de un subalgoritmo de selección de recursos bien definido para poder realizar distintas versiones del mismo.
- Disponer de un modelo de datos para definir plataformas que representen un nodo *grid* en el que se puedan incluir fácilmente para cada recurso de computación del mismo:
    - Instantes de fallos y recuperación a lo largo del tiempo.
    - Propiedades no funcionales.
- Disponer de plataformas y trabajos de ejemplo que se puedan reutilizar para realizar los experimentos.
- Disponer de funcionalidades de traza de la ejecución.

Teniendo en cuenta las ventajas e inconvenientes de cada uno de los simuladores presentados y los estudios realizados tanto por Leal (2010) como por Prajapati y Shah (2015), finalmente se ha optado por la utilización de *SimGrid*. Este simulador presenta características muy importantes: ofrece la implementación de un modelo de pruebas completo (*SimDag*) y está orientado a la evaluación de algoritmos de planificación y tiene un gran soporte técnico. Además, resulta bastante significativo qué su página web esté actualizada, que disponga de una sección de contribuciones con herramientas muy interesantes y que *SimGrid* haya sido utilizado previamente por múltiples investigadores para presentar resultados.

Además, Casanova et al (2014) describen con gran detalle las características técnicas de SimGrid y sus ventajas respecto a otras herramientas, justificando aún más su utilización en la presente tesis. Incluso en la presentación previa de su trabajo en la conferencia SimuTools se hace referencia a cuestiones relativas a la dificultad de acceso a desarrollos de otros investigadores para reproducir y/o mejorar sus experimentos, tal y como se plantea en las restricciones de la presente tesis.

En la última sección del capítulo de desarrollo de la tesis se describe con detalle el funcionamiento y las partes más importantes de la arquitectura de *SimGrid*, así como las modificaciones realizadas sobre el modelo de pruebas *SimDag* que ofrece para poder realizar los experimentos asociados a la tesis.

2.7 Conclusiones

Por un lado, los modelos de información *grid* existentes no usan el mismo lenguaje para representar los mismos conceptos sobre recursos, aunque varios grupos de la comunidad *grid* han detectado la necesidad de integrar sus modelos y hacerlos interoperables. Además, existe una falta de representación de propiedades no funcionales o de calidad de servicio sobre los recursos. En el mejor de los casos, algunos modelos especifican un rango de valores para medir y comparar el rendimiento de los recursos *grid*, pero faltan muchos detalles en general. Algunos proveedores y consumidores de información *grid* desarrollados específicamente para algunos proyectos usan técnicas aisladas para tratar el rendimiento de los recursos y su disponibilidad en un momento dado.

Por lo tanto, el presente trabajo plantea abordar la creación de un modelo de representación de recursos para entornos *grid* en el que se puedan reflejar propiedades no funcionales de los recursos y propone una metodología para representar dichas propiedades que luego permita a las potenciales herramientas o aplicaciones *grid* consumidoras de este tipo de información ver mejorada su funcionalidad.

Por otro lado, en cuanto a los mecanismos de planificación de trabajos y de administración de recursos existentes en las infraestructuras *grid*, a pesar de los avances de los últimos años, el soporte para las propiedades no funcionales o de QoS en entornos de computación *grid* es aún muy limitado y, hasta el momento, no existe una solución definitiva para el problema.

Sobre la base de las generalidades y particularidades encontradas en el estudio del estado de la cuestión respecto a los modelos de representación de información sobre recursos *grid* y sus propiedades no funcionales, así como de los componentes responsables de su organización y gestión, la planificación de trabajos y la organización de los sistemas de información *grid*, se ha podido identificar el conjunto de tecnologías, herramientas y a utilizar en los desarrollos de la presente tesis que se exponen en el siguiente capítulo de este trabajo.

Por último, cabe mencionar que los simuladores grid son muy útiles para mostrar pruebas de concepto. Sin embargo, aunque se pueden utilizar para la evaluación de nuevos algoritmos propuestos, la verdadera validación de su utilidad se debe obtener mediante la evaluación de los mismos en sistemas grid reales. Una vez que los investigadores tengan acceso a un entorno grid real adecuado, deben evaluar los algoritmos propuestos en dicho entorno real.

Ten siempre en tu mente a Ítaca.
La llegada allí es tu destino.
Pero no apresures tu viaje en absoluto.
Mejor que dure muchos años,
y ya anciano recales en la isla,
rico con cuanto ganaste en el camino,
sin esperar que te dé riquezas Ítaca.

Constantino Cavafis

Capítulo 3

Desarrollo

En este capítulo se describen todos los desarrollos llevados a cabo para evaluar las hipótesis de la tesis planteada: el modelo de representación de información sobre recursos *grid* incluyendo propiedades no funcionales, la metodología para la representación de la información anterior, incluyendo la aplicación de la misma para el desarrollo de tres métodos sobre dos propiedades no funcionales a nivel de recurso de computación y uno a nivel de nodo *grid*, así como las modificaciones realizadas en el modelo de pruebas de la herramienta de simulación seleccionada para la evaluación de la tesis para comprobar el efecto de utilizar propiedades no funcionales en la selección de recursos. Los métodos desarrollados para representar las propiedades no funcionales comparten un mismo proceso de desarrollo en el que se incluyen actividades para la especificación, medición, validación y publicación de datos sobre las mismas que permitirían poblar el modelo de datos de la tesis.

3.1 Modelo de representación de propiedades no funcionales en el dominio grid

3.1.1 Requisitos del modelo

Dadas las limitaciones descritas en el estado de la cuestión respecto a los modelos de representación de información sobre recursos de un nodo *grid*, surge la necesidad de disponer de un nuevo modelo que permita unificar el vocabulario utilizado por los diferentes modelos existentes y representar

también información relativa a propiedades no funcionales sobre dichos recursos, requisitos que se pueden cubrir desarrollando una ontología para el dominio *grid* que reutilice y extienda algunas ontologías descritas en el estado de la cuestión de la presente tesis.

En el campo de la ingeniería del *software*, una *ontología* es una especificación de una conceptualización o una descripción de los conceptos y relaciones que pueden existir en un dominio de aplicación para una comunidad de agentes relacionados. Las ontologías permiten compartir y reusar conocimiento. Las aplicaciones pueden ser escritas de acuerdo a un compromiso de utilización del vocabulario definido por una ontología para un dominio concreto. De esa manera, los agentes que comparten la misma ontología pueden comunicarse entre ellos ya que todos entienden el mismo vocabulario.

Teniendo en cuenta las especificaciones y modelos presentados por otros grupos de trabajo, se propone una ontología para describir los tipos de recursos, servicios, roles y otros conceptos del dominio de la *grid*, así como para representar algunas de las propiedades no funcionales o de QoS más importantes asociadas a los mismos según la literatura del área: *rendimiento*, *disponibilidad* y *fiabilidad*. Cuando diferentes herramientas proveedoras de información *grid* soportan una ontología común, los usuarios *grid* y las aplicaciones consumidoras de información *grid* pueden beneficiarse de manejar una misma conceptualización sobre los recursos *grid* y su comportamiento en relación a la calidad de servicio.

Cabe destacar en este punto que se ha decidido utilizar este artefacto para implementar el modelo de información de la tesis porque, además de poder reutilizar desarrollos previos siguiendo esta tecnología, permitirá que sea integrado fácilmente como recurso de conocimiento para la *grid* tal y como propone la arquitectura S-OGSA que se utilizará en la metodología de la tesis.

Así mismo, el objetivo inicial de esta ontología en la presente tesis se reduce a generar una clasificación de términos *grid* y de QoS, así como de establecer relaciones entre los mismos, quedando fuera del alcance de la misma realizar desarrollos orientados a utilizarla para realizar inferencias u obtener alguna otra funcionalidad ofrecida por herramientas como los razonadores.

Por ello, dado que simplemente cumplirá una función de modelado de información, se propone explotar el modelo de información en formato RDFS. Además, dado que los lenguajes como OWL no pueden manejar directamente datos numéricos, se propone una modificación futura al modelo para que los datos puedan ser asociados con rangos de posibles valores.

Sin embargo, como se podrá ver en la metodología de la tesis, el enfoque de este modelo es híbrido, ya que se basa en una ontología y en otras técnicas a la hora de utilizar la información representada en la misma para la selección de recursos *grid*.

A continuación se presenta una visión general de la ontología desarrollada para representar información sobre recursos *grid* y sus propiedades no funcionales o de calidad de servicio.

3.1.2 Ontologías reutilizadas

A la hora de desarrollar nuestra ontología *NFP4Grid* hemos creado nuestras propias clases, pero también hemos intentado reusar y/o extender ciertas clases para integrar algunas ideas interesantes de las diferentes ontologías comentadas en el estado de la cuestión: GOM *Resource Ontology* [26], GOM *Service Ontology* [26], *Grid Ontology* de Parkin et al. (2006). Estas ontologías han servido de punto de partida para diseñar una clasificación de recursos *grid* más completa, teniendo en cuenta diferentes niveles de detalle, así como los diferentes roles que se pueden asociar a cada recurso en un entorno *grid*. Dado que estas ontologías de partida poseen una pobre definición de propiedades no funcionales o de calidad de servicio, también hemos integrado algunas ideas presentes en la ontología *QoSOnt* de Dobson, Lock y Sommerville (2005) en nuestro modelo.

Dado que el modelo S-OGSA aún no está adoptado por la comunidad *grid*, se propone crear una ontología propia para los intereses de la presente tesis reutilizando los elementos necesarios de las ontologías analizadas. En un futuro, lo ideal sería integrar la ontología que implementa el modelo de la presente tesis en la colección de ontologías que forman la *Grid Ontology*, estableciendo equivalencias entre clases y creando nuevas propiedades entre las mismas.

3.1.2.1 GOM Resource Ontology

A continuación se muestra una serie de reflexiones sobre las posibilidades de reutilización de esta ontología (*ResourceOntology.owl*):

- Las clases `Resource`, `ClusterNodeRole=GridNodeRole` son interesantes, aunque no describen subclases.
- La clase `NetworkConnection` no tiene propiedades ni está relacionada con la clase `Reliability`, cuando la red es un elemento clave

relacionado con la *reliability* de los recursos que se acceden a través de ella.

- La ontología importa otra (*GOM Application Ontology*) que a su vez hace uso de otra ontología de tiempo (*OWL- Time*).
- La clase `Reliability` aparece en la ontología importada (*GOM Application Ontology*) como una subclase de `ComparisonCriteria=QoSAttribute`, pero no está definida ni su subclasificación ni las métricas para medirla.
- La clase `Activity` y sus subclases son las que tienen relación con `ComparisonCriteria`, pero no con la clase `Reliability`.
- La ontología *OWL-Time* es más limitada que la de otra ontología (*QoSOnt2*) que se analizará posteriormente.

A continuación se muestran una serie de acciones que se han llevado a cabo en la ontología que implementará el modelo de la presente tesis:

- Se extenderá la clase `GridResource` con `Hardware=HardwareResource`, `Software=SoftwareResource` y `Data=StorageObject`.
- Se crearán las subclases `Token` y `Resource_Collection` dentro de `Resource`.
- Se extenderá la clase `SoftwareApplication=Program` con la subclase `Benchmark`.
- Se extenderá la clase `NetworkResource` con la subclase `NetworkHost`.
- Se extenderá la subclase `NetworkConnection` con las subclases `NetworkRoute>NetworkCommunication`.
- Se extenderá la clase `Hardware` con la subclase `Processing=ComputingResource`.
- Se extenderá la clase `StorageResource` con la subclase `Disk=HardDrive`.
- Se creará la clase `VO=VirtualOrganization`.
- Se creará la clase `QoSAttribute=ComparisonCriteria`.

3.1.2.2 GOM Service Ontology

A continuación se muestra una serie de reflexiones sobre las posibilidades de reutilización de esta ontología (*ServiceOntology.owl*):

- Importa las ontologías *GOM Resource Ontology*, *GOM Data Ontology*, *GOM Application Ontology* y OWL-S.

A continuación se muestran una serie de acciones que se han llevado a cabo en la ontología que implementará el modelo de la presente tesis:

- Se reutilizará la clase `StorageObject` como subclase de la clase `Resource`.
- Se reutilizará la clase `MonitoringService>` `PerformanceAnalysisService` como subclase de la clase `GridTool`.

3.1.2.3 Grid Ontology

A continuación se muestra una serie de reflexiones sobre las posibilidades de reutilización de la colección de ontologías que forman esta ontología fundacional (*Base.owl*, *Globus.owl*, *Unicore.owl*, *Ogsa-ontology.owl*, *S-OGSAOntology.owl*):

- Se trata de una ontología que trata de aglutinar conceptos genéricos para cualquier entorno *grid*, sea de la generación que sea.
- Dado que el modelo S-OGSA aún no está adoptado por la comunidad *grid* y que no existe un estándar o modelo de información *grid* global, se propone crear una ontología propia para los intereses de la presente tesis reutilizando los elementos necesarios de esta colección de ontologías, al igual que con el resto de ontologías reutilizadas. En un futuro, lo ideal sería integrar la ontología que implementa el modelo de la presente tesis en esta colección de ontologías, estableciendo equivalencias entre clases y creando nuevas propiedades entre ellas.

A continuación se muestran una serie de acciones que se han llevado a cabo en la ontología que implementará el modelo de la presente tesis, respecto a la subontología base (*Base.owl*):

- Se reutilizará la clase `Client` y sus subclases.
- Se reutilizará y extenderá la clase `Role`, creando la subclase `Server_Role` además de la ya existente `Client_Role`, así como nuevas subclases para identificar roles más específicos en un entorno *grid*, diferenciando entre `GridHost`, `GridNode`, `GridSystem` y otras.
- Se reutilizará la clase `Information_Service` y sus subclases, así como la clase `Execution_Service`, para incluirlas en la jerarquía de clases que se crearán para clasificar los tipos de `GridServices`.

- Se reutilizarán las subclases de la clase `Resource` creando equivalencias con clases ya existentes.
- Se reutilizará la clase `VO` y sus subclases.
- Se reutilizará la clase `Broker`.
- Se reutilizará las clases `QoS=QoSType` y `Agreement`, así como sus respectivas subclases, creando equivalencias con otras clases de la ontología *QoSOnt2* que se analizará en un posteriormente.
- Se reutilizará la clase `QoS=QoSType` como subclase de `QoSConcept`.

A continuación se muestran una serie de acciones que se han llevado a cabo en la ontología que implementará el modelo de la presente tesis, respecto a la subontología Globus (*Globus.owl*):

- Se reutilizarán las clases `Access_Point->Globus_Gatekeeper`.
- Se reutilizarán las subclases `Globus_GIIS`, `Globus_GRIS` y `Globus_MDS`.

3.1.2.4 QoSOnt2 Ontology

A continuación se muestra una serie de reflexiones sobre las posibilidades de reutilización de esta ontología (*QoSOnt2.owl*):

- En su versión anterior eran varias ontologías modulares.
- Describe los servicios web y algunos atributos relacionados con la calidad de servicio o QoS de los mismos, así como métricas para algunos de ellos.

A continuación se muestran una serie de acciones que se han llevado a cabo en la ontología que implementará el modelo de la presente tesis:

- Se reutilizarán las clases `ServiceProfile>Profile>QoSProfile` y `ServiceParameter>ServiceQoSParameter`.
- Se integrarán las clases `NetworkConcept>NetworkHost` y `NetworkConcept>NetworkRoute` como subclases de la clase `NetworkResource` existente.
- Se reutilizará y extenderá la clase `QoSConcept`.
    - Las instancias `Reliability`, `Performance` y `Availability` de la clase `QoSAttribute>MeasurableQoSAttribute` se convertirán en subclases para poder considerar distintos subtipos dentro de cada una de ellas.

- Se extenderá la clase `QoSMetric` con subclases que representen métricas de tamaño y velocidad que no están consideradas en esta ontología.
- Se incluirán nuevas subclases dentro de la clase `TimeMetric` para poder representar nuevas métricas sobre propiedades no funcionales.
- Se reutilizará y extenderá la clase `MeasurementConcept` con el objetivo de incluir medidas para conceptos como tamaño y velocidad que no están consideradas en esta ontología.

3.1.3 Modelo conceptual

En el presente apartado presentamos una visión general de la ontología *NFP4Grid* utilizando imágenes que representan las clases más importantes relacionadas con la clasificación de tipos de recursos *grid*, roles y propiedades de calidad de servicio. En las imágenes aparece el carácter "+" en algunas clases indicando que poseen subclases o superclases. La ontología NFP4Grid está disponible en [55] para poder consultar todo su contenido, así como para su pública revisión y comentario, aunque en el apéndice A de la presente tesis también se listan todas las clases y propiedades de la ontología.

En la figura 3.1 se muestra el diagrama parcial de clases relacionadas con los tipos de entidades *grid* que se representan en el modelo, atendiendo a la arquitectura tipo de una infraestructura *grid*. Como se puede observar en la figura, cualquier entidad *grid* es susceptible de tener un atributo de calidad de servicio o propiedad no funcional asociada.

Cabe recordar en este punto que los recursos *grid* pueden ser de diferente naturaleza y nivel, tal y como vimos en el estado de la cuestión al hablar de la arquitectura de la *grid*, su *middleware*, sus modelos de información e incluso en el modelo de la propuesta S-OGSA.

En la figura 3.2 se muestra el diagrama parcial de clases relacionadas con tipos de recursos *grid* que son gestionados a más bajo nivel, como son los recursos de computación, de almacenamiento, programas, datasets, etcétera.

En la figura 3.3 se muestra el diagrama parcial de clases relacionadas con tipos de recursos/servicios *grid* más generales y que están más relacionados con los servicios que provee el *middleware* de la *grid*. En este caso, los servicios también pueden ser de diferente naturaleza y nivel debido a los diferentes niveles de gestión y planificación de recursos/servicios existente en la *grid*.

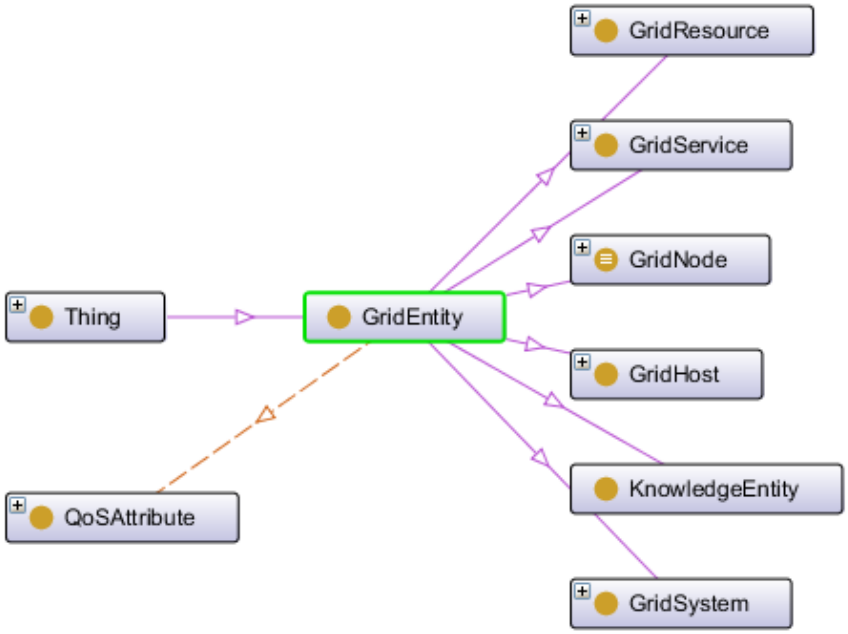

Figura 3.1 Diagrama parcial de clases relacionadas con tipos de entidades grid

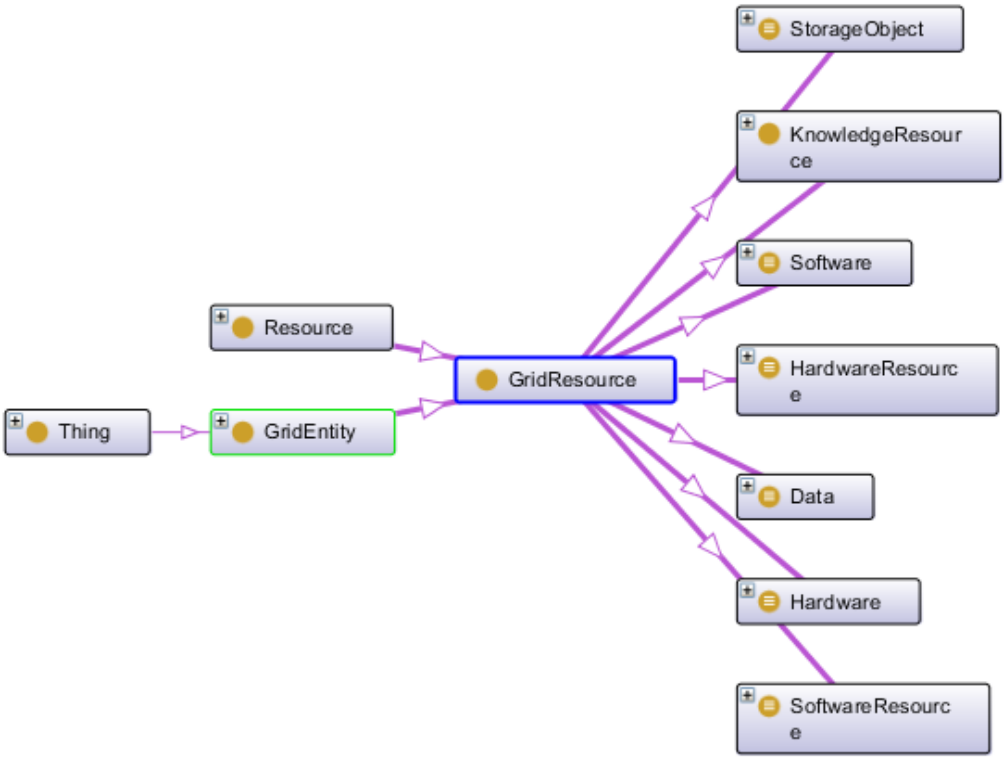

Figura 3.2 Diagrama parcial de clases relacionadas con tipos de recursos grid de más bajo nivel

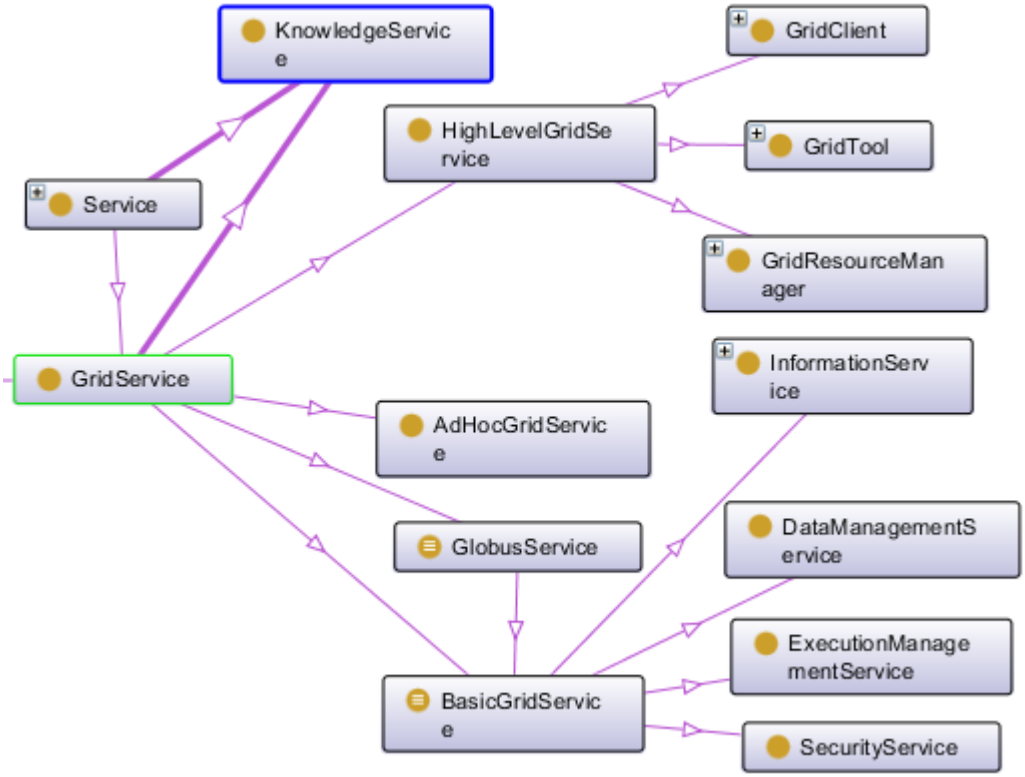

Figura 3.3 *Diagrama parcial de clases relacionadas con tipos de recursos/servicios grid más generales relacionados con middleware grid*

En la figura 3.4 se muestra el diagrama parcial de clases relacionadas con tipos de roles de las entidades *grid*. Las infraestructuras *grid*, incluyendo la jerarquía de recursos y servicios que proveen, pueden adoptar diferentes roles a lo largo de su vida en función del tipo de recursos y servicios que la configuran para especializarse en el alojamiento de experimentos de diferente naturaleza científica.

En la figura 3.5 se muestra el diagrama parcial de clases relacionadas con propiedades no funcionales o de calidad de servicio, incluyendo todo tipo de características sobre las mismas para poder representar propiedades medibles y no medibles, así como de distinta naturaleza. Cabe recordar en este punto que la ontología no es una lista cerrada de términos y que permite su ampliación y reutilización por parte de cualquier desarrollador, siendo esta una tarea fácil importándola en cualquier editor de ontologías.

En el siguiente apartado se muestran ejemplos de las clases y propiedades de la ontología relacionadas con los recursos *grid* y propiedades no funcionales o de calidad de servicio que se utilizan en el contexto de la tesis.

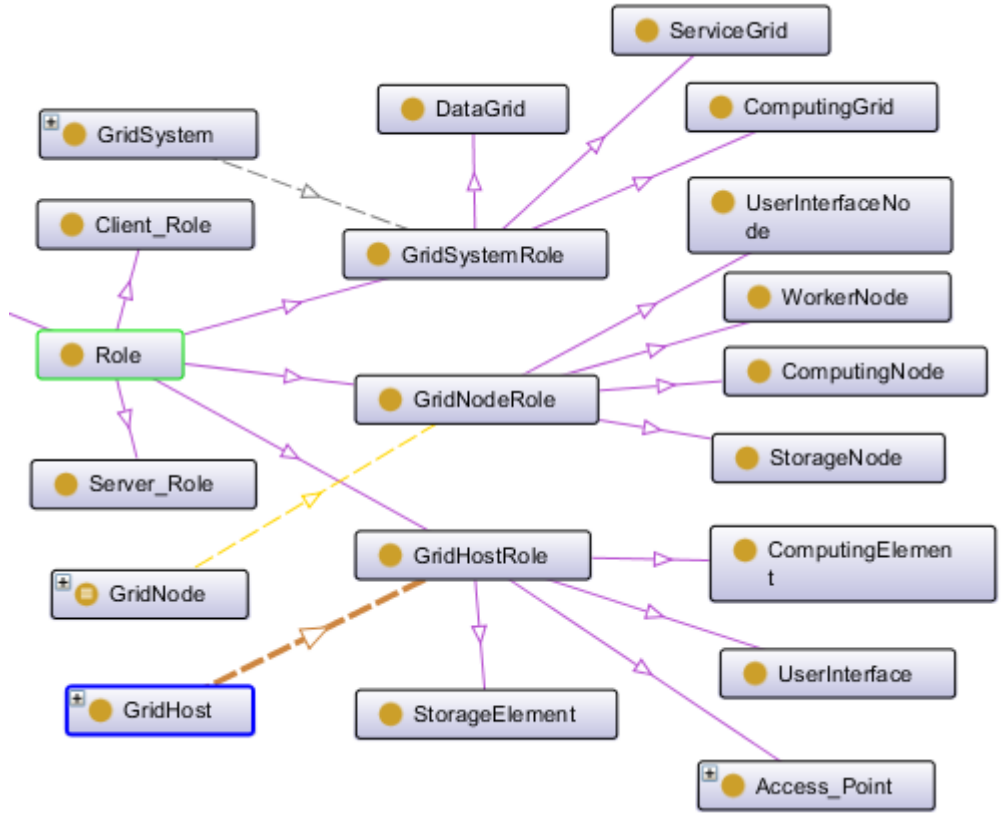

Figura 3.4 *Diagrama parcial de clases relacionadas con tipos de roles de las entidades grid*

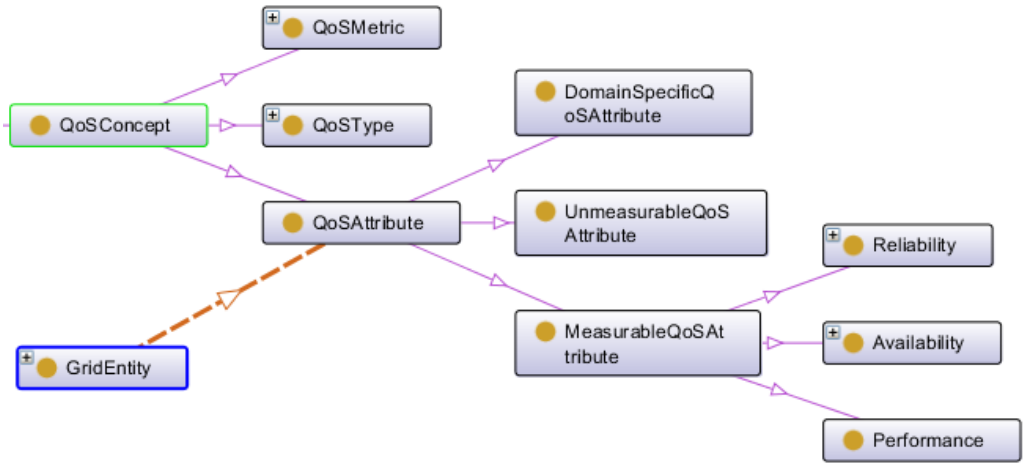

Figura 3.5 *Diagrama parcial de clases relacionadas con propiedades no funcionales o de calidad de servicio*

3.1.4 Clases y propiedades de la ontología utilizadas en la tesis

En la actual versión de la ontología solamente aparecen reflejadas las propiedades necesarias para los objetivos de la tesis, es decir, las necesarias para la representación de propiedades no funcionales fiabilidad (ver figura 3.6) y disponibilidad (ver figura 3.7) asociadas a entidades de la *grid*, y más concretamente a los recursos *grid*.

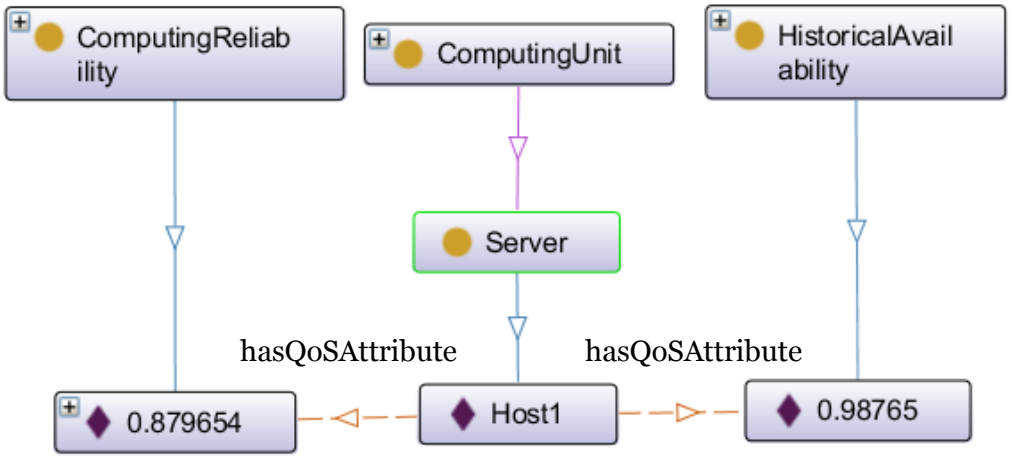

Figura 3.6 Clases y propiedades de la ontología relacionadas con la representación de la fiabilidad y la disponibilidad de un recurso de computación grid

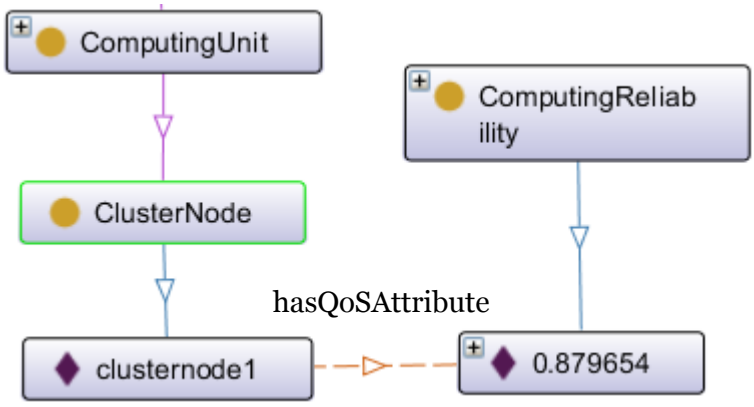

Figura 3.7 Clases y propiedades de la ontología relacionadas con la representación de la fiabilidad de un nodo grid de computación

Dado que en la mayoría de los casos los datos asociados a las propiedades no funcionales o de calidad de servicio son numéricos y que, en el ámbito de la presente tesis, la ontología simplemente cumplirá una función de modelado de información, dichos valores serán representados directamente en la ontología y se propone explotar el modelo de información en formato RDFS. Dado que los lenguajes como OWL no pueden manejarlos directamente, se propone una modificación futura al modelo para que los datos puedan ser asociados con rangos de posibles valores.

Por lo tanto, los consumidores de información podrán extraer los valores de las propiedades no funcionales que deseen sobre las diferentes entidades *grid* representadas en el modelo con el objetivo de utilizarlas para su beneficio, como sería el caso del *middleware* de la *grid* que se ocupa de la selección y asignación de recursos.

3.1.5 Integración del modelo con la arquitectura S-OGSA

Desde el enfoque más sencillo, la ontología NFP4Grid que implementa el modelo de la tesis sería considerada una instancia de la clase `K-Resource` (*Knowledge Resource*) dentro del modelo de información de S-OGSA. Obviamente, por las relaciones establecidas en dicho modelo, también pertenecería a las clases `G-Resource` (*Grid Resource*), `K-Entity` (*Knowledge Entity*) y `G-Entity` (*Grid Entity*).

Este primer enfoque permitiría utilizar la ontología en la creación de `S-Bindings` (*Semantic Bindings*) para representar asociaciones semánticas entre `G-Entities` y `K-Entities`, como podrían ser el conjunto de asertos, metadatos o tripletas de información sobre las propiedades no funcionales o de calidad de servicio que posee una determinada `G-Entity`, convirtiéndola así en una `SG-Entity` (*Semantic Grid Entity*).

Tal y como se indica en la propuesta S-OGSA, estos `S-Bindings` se modelarían también como `G-Resources` y se podrían gestionar mediante *Semantic Binding Provisioning Services* desarrollados por los responsables de cada `G-Entity`.

Concretamente, mediante un servicio de anotación para la creación de `S-Bindings` y un servicio de metadatos para exponerlos. Por las relaciones establecidas en el modelo S-OGSA, estos servicios también se consideran `G-Resources` que cualquier desarrollador de aplicaciones *grid* podría consumir directamente.

Sin embargo, en un escenario ideal, estos servicios deberían estar disponibles integrados como extensiones de los servicios *grid* ofrecidos por cualquier

middleware grid que gestionan las diferentes entidades *grid* a modo de SAGS (*Semantically Aware Grid Services*), que en la propuesta S-OGSA son los encargados de consumir los `S-Bindings` de dichas entidades *grid* a través de los *Semantic Binding Provisioning Services* disponibles para ello.

Cabe recordar que todos estos servicios se desarrollarían siguiendo la especificación WSRF utilizada en la *grid* para la implementación de servicios (*grid*) web con estado, utilizando el conjunto de operaciones típicas del protocolo HTTP (PUT, GET, POST y DELETE). Dado que los `WS-Resources` de la especificación tienen asociados `WS-ResourceProperties`, estos elementos podrían contener todo tipo de referencias a los `S-Bindings` de un determinado `G-Resource` (servicios de anotación, servicios de metadatos, etcétera) cuya información podría obtenerse a través de las operaciones `getProperties` y `queryProperties` existentes en la especificación WSRF.

Por último, un segundo enfoque más ambicioso consistiría en integrar completamente la ontología NFP4Grid que implementa el modelo de la presente tesis en la colección de ontologías de la *Grid Ontology*, estableciendo equivalencias entre clases y creando nuevas propiedades entre ellas.

3.1.6 Ontología fundacional para el modelo de información grid del OGF

Tal y como se mencionó en el estado de la cuestión, existen varias iniciativas específicas para proyectos de investigación o de grupos de trabajo del OGF que proponen modelos parciales para representar parte de la información del dominio *grid* necesaria para sus intereses.

A pesar del interés del OGF en unificar esfuerzos y modelos de información, hasta la fecha aún no existe un modelo estándar de información *grid* en general para representar todo el ecosistema de la *grid*.

Atendiendo a la posibilidad mencionada en el apartado anterior sobre la posibilidad de integrar la ontología NFP4Grid en la colección de ontologías de la *Grid Ontology*, dado que esta integra el modelo S-OGSA y contiene gran cantidad de clases para representar el ecosistema *grid*, la ontología fundacional resultante tras la integración de la ontología NFP4Grid podría ser considerada como un recurso o punto de partida para implementar el modelo global de información *grid*.

Una vez más, mediante la posibilidad de reutilización y extensión que ofrecen las ontologías con nuevas subclases y equivalencias, la creación de `S-Bindings` para asociar `G-Entities` con sus modelos de información específicos (GLUE,

WS-Agreement, JSDL, DFDL, etcétera), así como el desarrollo de *Semantic Binding Provisioning Services* apropiados, sería posible integrar los modelos de información de cada uno de los grupos de trabajo de estandarización del OGF.

Por último, los servicios *grid* ofrecidos por cualquier *middleware grid* que gestionan las diferentes entidades *grid* podrían extender su propia funcionalidad mediante SAGS (*Semantically Aware Grid Services*) desarrollados para consumir los `S-Bindings` de dichas entidades a través de los *Semantic Binding Provisioning Services* disponibles para ello.

Al igual que en el apartado anterior, cabe recordar que todos estos servicios se desarrollarían siguiendo la especificación WSRF, extendiendo las `WS-ResourceProperties` de los `WS-Resources` correspondientes.

3.2 Metodología para la representación de propiedades no funcionales en el dominio grid

El propósito de esta metodología para la representación de propiedades no funcionales en el dominio *grid* es el de dar soporte a aquellos desarrolladores/organizaciones que deseen publicar este tipo de información sobre los recursos *grid* que administran.

Dado que en el modelo se consideran propiedades no funcionales o de calidad de servicio para recursos *grid software* y *hardware* a varios niveles (sistema, servicio, aplicación, nodo, red, máquina, etcétera), cada una de ellas puede tener diferentes características (complejidad de cálculo, métricas, parámetros de relevancia, etcétera).

Este capítulo presenta una metodología para la población del modelo de representación de propiedades no funcionales para el dominio *grid*, indicando actividades a realizar para la definición, cálculo, medida, integración y publicación de propiedades no funcionales o de calidad de servicio en el modelo mencionado.

En primer lugar, se define lo que se entiende por metodología, así como sus conceptos asociados. En segundo lugar, se presentan los principios de diseño considerados para la definición de esta propuesta metodológica. En tercer lugar, se describen los diversos participantes implicados en el proceso de población de datos. En cuarto lugar, se describen los procesos seguidos para definir esta metodología y, además, se presenta una visión general de las actividades y tareas propuestas.

En siguientes subapartados, se aplica esta metodología a dos propiedades no funcionales a nivel de recurso de computación (fiabilidad y disponibilidad) y una a nivel de un nodo *grid* (fiabilidad).

Vilches (2011) realiza un excelente resumen sobre cómo formalizar una metodología. Por su interés para contextualizar la presente tesis y dada su excelente redacción, se recomienda su lectura para contextualizar y entender el enfoque utilizado para la metodología de la presente tesis.

3.2.1 Definición de la metodología

Hodge (2000) indica que los términos *metodología*, *método*, *técnica*, *proceso*, *actividad*, etcétera, son utilizados en la literatura de forma indistinta. En el contexto de la presente tesis se adoptarán las definiciones propuestas por el IEEE, tal y como propone Vilches (2011):

> «El IEEE define **metodología** como "*una serie integrada y exhaustiva de técnicas o métodos que crean una teoría general de sistemas de cómo una clase de trabajo intensivo pensado debería ser llevada a cabo*" (IEEE 1990). Las metodologías son ampliamente utilizadas en la Ingeniería del Software (Downs, Clare y Coe 1998, Pressmann 2000, Kruchten 2000, MÉTRICA v.3), Ingeniería del Conocimiento (Waterman 1986, Gómez-Pérez et al. 1997, Schreiber et al. 1999) o construcción de ontologías (Fernández-López et al. 1999, Staab et al. 2001, Gómez-Pérez, Fernández-López y Corcho 2003, Pinto, Tempich y Staab 2004, Suárez-Figueroa et al. 2008). La definición previa de metodología menciona que métodos y técnicas son partes de la metodología. Un **método** es un conjunto de "*procesos o procedimientos ordenados utilizados en ingeniería de un producto o transformación de un servicio*" (IEEE 1990a). Una **técnica** es "*un procedimiento directivo y técnico utilizado para lograr un objetivo dado*" (IEEE 1990b).» (Vilches 2011: 93-94)

Según De Hoog (1998) las metodologías y métodos no son lo mismo porque "las metodologías se refieren al conocimiento sobre los métodos". Greenwood (1973) identifica las diferencias entre métodos y técnicas concluyendo que un método es un procedimiento general, mientras que una técnica es una aplicación específica de un método y la forma en que el método es ejecutado. Normalmente, existen varias técnicas para aplicar un método dado.

> «De la definición de metodología, expuesta anteriormente, se deduce que métodos y técnicas están estrechamente relacionados, consecuencia de que ambos son utilizados para llevar a cabo las tareas presentes en los diferentes procesos de las que consta una metodología. El IEEE define un **proceso** (IEEE 1995) como "*una función que debe ser llevada a cabo en el ciclo de vida del software. Un proceso está compuesto de actividades*". Una **actividad** (IEEE 1995) es "*una tarea componente de un proceso*". Otra definición de actividad (IEEE 1997) afirma que es un cuerpo definido de trabajo para ser efectuado, incluyendo su información de entrada y salida requerida. En definitiva, una tarea es la unidad de trabajo más ínfima sujeta a la responsabilidad de gestión. Así, "*una tarea* (IEEE 1995) *es un trabajo bien definido asignado a uno o más miembros del proyecto. Las tareas relacionadas, normalmente, son agrupadas para formar actividades*".» (Vilches 2011: 94)

Tal y como se puede apreciar en la figura 3.8 (Gómez-Pérez, Fernández-López y Corcho 2003), una metodología está compuesta de métodos y técnicas. Los métodos están compuestos de procesos y son detallados con técnicas. Los procesos están compuestos de actividades, que a su vez están formadas por grupos de tareas.

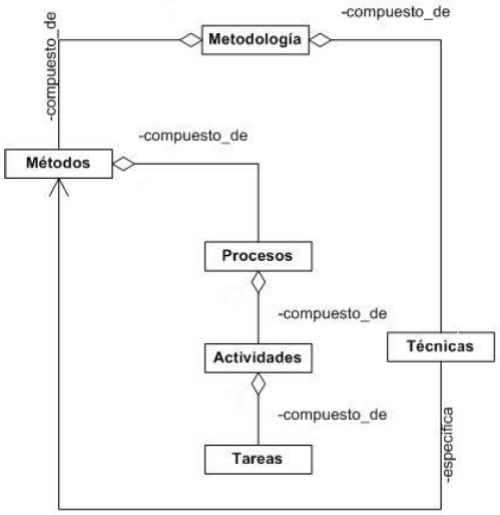

Figura 3.8 Representación gráfica de las relaciones terminológicas en las metodologías

En el caso que nos ocupa, por cuestiones ya mencionadas en las restricciones de la tesis, se desarrollarán únicamente tres métodos con distinta profundidad: dos para la representación de propiedades no funcionales a nivel de recurso de computación (*eficiencia* y *fiabilidad*) y otro para la representación de propiedades no funcionales a nivel de nodo *grid* (*fiabilidad*).

3.2.2 Principios de diseño

Los principios de diseño aplicados al desarrollo de esta metodología de representación de propiedades no funcionales en el dominio *grid* se han extraído de las condiciones necesarias y suficientes que, según Vilches (2011) citando a Paradela (2001), cada metodología debería satisfacer.

Las condiciones necesarias son independientes del dominio de aplicación de las metodologías y son comunes a cualquier metodología: completitud, eficiencia, efectividad, consistencia, cuestiones sobre la esencia, conjunto de reglas metodológicas generales, finitud, discernimiento, entorno, transparencia. Todas ellas definidas con más detalle en Paradela (2001).

Las condiciones suficientes son específicas del dominio al que se aplica la metodología. En el caso de la metodología de representación de propiedades no funcionales en el dominio *grid* de la presente tesis, las condiciones suficientes están fundamentadas en una serie de condiciones específicas propias de los diferentes componentes que conforman este acercamiento metodológico.

- *Basada en estándares o estándares de facto*. La metodología debe estar orientada a unificar criterios relacionados con la representación de propiedades no funcionales en el dominio *grid*, de manera que el concepto definido sea entendible por la comunidad *grid* y se ajuste en lo posible a definiciones y conceptos manejados por los organismos de estandarización correspondientes (OGF, W3C, etcétera). También debe perseguir la obtención de pautas y lecciones de buenas prácticas en su aplicación.
- *Dominio*. La metodología se propone para la representación de propiedades no funcionales en el dominio *grid*.
- *Usabilidad*. La metodología debe ser fácil de entender y aprender. Además, el esfuerzo necesario para utilizar la metodología debe ser mínimo con el fin de facilitar su éxito y generalizar su uso.
- *Inspirada en prácticas existentes*. La metodología debe estar fundada en diferentes propuestas metodológicas existentes.
- *Independiente de herramientas*. De forma general, una metodología debe ser independiente de las tecnologías existentes.
- *Flexibilidad*. Esta metodología debe ofrecer flexibilidad para ciertas tareas dependientes del propósito y/o del conocimiento de los expertos.
- *Integración*. La metodología debe ser consecuente con las directrices expuestas por el OGF a través de la especificación OGSA y las recomendaciones de la propuesta S-OGSA para la extensión de la *grid* con capacidades semánticas.

3.2.3 Responsables implicados: roles

Vilches (2011) distingue diversos participantes con distintos roles para llevar a cabo las tareas que componen las actividades del proceso de desarrollo de una metodología como la de esta tesis: jefe de proyecto, ingeniero de software, ingeniero de conocimiento, experto del dominio y desarrollador web.

Una descripción más detallada de los perfiles asociados a los roles mencionados puede encontrarse en Vilches (2011).

3.2.4 Proceso de desarrollo para los métodos

Esta sección describe, desde una perspectiva general, un único proceso de desarrollo común que se propone para cada uno de los métodos que se trabajarán en la metodología propuesta en la tesis, detallando el conjunto de actividades a realizar para la representación de propiedades no funcionales en el dominio *grid*. Este conjunto de actividades se basan en las principales actividades identificadas por el proceso de desarrollo de *software* en IEEE (1999), reemplazado por IEEE (2010a).

El proceso de desarrollo de la metodología se ha llevado a cabo con estos pasos:

- *Análisis* de alto nivel de las *características* que particularizan al dominio de trabajo.
- *Selección de procesos* relevantes pertenecientes a otras áreas de investigación.
- *Identificación* de las principales *tareas*, en los procesos relevantes elegidos, para seleccionar y analizar las tareas utilizadas en la mayoría de procesos de cada área.
- *Abstracción y adaptación* de las diferentes tareas para cubrir las necesidades del proceso de representación de propiedades no funcionales en el dominio *grid*.
- *Análisis* de las *dependencias* de las tareas para definir en consecuencia el orden de las mismas.

Un primer resultado de la aplicación de este proceso de desarrollo es la identificación del conjunto de actividades que componen el proceso de desarrollo de cada uno de los métodos que se han trabajado en la metodología. Estas actividades están formadas por:

- *Especificación de la propiedad no funcional del recurso*. Se engloban todas las tareas orientadas a acotar la definición y la forma de calcular la propiedad no funcional del recurso en cuestión, así como las restricciones para su utilización.
- *Medición de la propiedad no funcional del recurso*. Involucra todas las tareas dedicadas a la obtención de datos que permitan realizar el cálculo de la propiedad no funcional del recurso en cuestión, según lo especificado en la actividad anterior.
- *Validación de la propiedad no funcional del recurso*. Esta actividad se centra en asegurar que los datos sobre la propiedad no funcional del

recurso son válidos para el responsable de administrar el recurso y, opcionalmente, si se han obtenido correctamente.

- *Publicación y evolución de la propiedad no funcional del recurso*. Involucra a las actividades que permiten publicar y consumir los datos sobre la propiedad no funcional del recurso en los sistemas de información *grid* correspondientes.

3.2.4.1 Especificación de la propiedad

A continuación se enumeran las tareas orientadas a acotar la definición y la forma de calcular la propiedad no funcional del recurso en cuestión, así como las restricciones para su utilización.

1. Buscar definiciones y sistemas de medida de la propiedad no funcional en fuentes de información contrastadas y del dominio *grid*, como por ejemplo:
   1.1. Estándares y recomendaciones del OGF (*Open Grid Forum*).
   1.2. Revistas especializadas: *Journal of Grid Computing*, *Future Generation Computer Systems* (*The International Journal of Grid Computing and eScience*).
   1.3. Congresos de referencia: *Grid* (*International Conference Distributed Computing and Grid-technologies in Science and Education*), *IEEE/ACM CCGrid* (*IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*).
   1.4. Glosarios técnicos: IEEE, *Free On-line Dictionary of Computing*.
   1.5. Enciclopedias y diccionarios: Wikipedia, Real Academia Española (no existe versión inglesa), etcétera.
2. Construir una definición de la propiedad no funcional para adaptarla al dominio *grid*.
3. Identificar las referencias existentes tanto a la propiedad no funcional como al recurso *grid* en cuestión dentro de la ontología que implementa el modelo de información propuesto en la presente tesis.
4. Determinar si existen en el estado de la cuestión modelos matemáticos para la estimación de la propiedad no funcional en el dominio *grid* en el marco de la siguiente actividad.
5. Determinar las unidades de medida o rangos de valores que se pueden utilizar para la representación de datos sobre la propiedad no funcional.

3.2.4.2 Medición de la propiedad

A continuación se detallan las tareas dedicadas a la obtención de datos que permitan realizar la estimación de la propiedad no funcional del recurso en cuestión que gestionamos.

1. Determinar los datos sobre el comportamiento del recurso en cuestión de los que disponemos y comparándolos con las necesidades de los modelos matemáticos analizados en la actividad anterior.
2. Obtener los datos imprescindibles que no se posean a través de alguna de estas técnicas (por orden y para cada caso):
    2.1. Evaluar si se pueden obtener a partir de datos de otros recursos *grid* de inferior nivel, aplicando los métodos correspondientes para obtenerlos.
    2.2. Evaluar si se pueden utilizar técnicas de aprendizaje automático o inferencia que permitan correlacionar los datos de los que se dispone para intentar obtener un modelo matemático que permita estimar los datos que faltan.
    2.3. Evaluar si se pueden simular los datos generando valores realistas a partir de los modelos matemáticos más plausibles encontrados en el estado de la cuestión analizado.
3. Determinar un modelo matemático concreto para la estimación de la propiedad no funcional del recurso en cuestión en función de los datos disponibles tras realizar la tarea anterior.
    3.1. Seleccionar el modelo más apropiado del estado de la cuestión o evaluar la posibilidad de desarrollar un modelo particular en base a los datos disponibles utilizando, una vez más, técnicas de aprendizaje automático y de inferencia.
4. Realizar la estimación de la propiedad no funcional del recurso aplicando las fórmulas o modelos matemáticos determinados en la tarea anterior.
5. Determinar la unidad de medida o rango de valores que se utilizará para la representación del dato estimado sobre la propiedad no funcional del recurso en cuestión.

3.2.4.3 Validación de la propiedad

A continuación se detallan las tareas centradas en asegurar que los datos sobre la propiedad no funcional del recurso son válidos para el responsable de administrar el recurso y, opcionalmente, si se han obtenido correctamente.

1. Comprobar si son válidos los datos utilizados en la actividad anterior para estimar la propiedad no funcional:
   1.1. Comprobar si los datos reales de partida siguen siendo válidos.
   1.2. Aplicar la actividad de validación correspondiente a aquellos datos obtenidos a través de otros recursos *grid* de inferior nivel.
   1.3. Evaluar si los datos estimados cumplen el modelo matemático que se utilizó en su cálculo.
   1.4. Evaluar si los datos simulados cumplen el modelo matemático que se utilizó en su cálculo.
2. Comprobar si el dato de la propiedad no funcional obtenido en la actividad anterior cumple el modelo matemático que se utilizó para su medición (si cumple la función de distribución, si está dentro del rango de valores válido, etcétera).
3. Conseguir la aprobación del responsable del recurso en cuestión (elemento de computación o de almacenamiento, *gateway*, LRM, servicio *grid*, nodo *grid*, VO, etcétera) sobre el dato de la propiedad no funcional para poder realizar la siguiente actividad de publicación.

3.2.4.4 Publicación y evolución de la propiedad

A continuación se indican las actividades que permiten publicar y consumir los datos sobre la propiedad no funcional del recurso en los sistemas de información *grid* correspondientes.

Dado que la propuesta S-OGSA no está adoptada aún por la comunidad *grid*, las tareas que se describen a continuación se reducen al desarrollo de servicios *grid* semánticos para el recurso en cuestión, que serían los *Semantic Binding Provisioning Services* del modelo S-OGSA.

En la descripción de la presente actividad, se presupone que estos servicios tendrían acceso a otro servicio *grid*, desplegado en el sistema de información de la *grid* a la que se tiene acceso, que podríamos identificar como un *Semantic Provisioning Service* del modelo S-OGSA que da acceso a la ontología descrita en el modelo de representación de la tesis (un `K-Service` que accede a un `K-Resource`).

En la descripción de las tareas correspondientes se utilizará como referencia la especificación WSRF para el desarrollo de servicios *grid* así como el estándar *de facto* Globus Toolkit para el *middleware* de la *grid* y más concretamente la versión 4 (GT4).

En un escenario ideal, el acceso a todos estos servicios debería estar disponible a través de extensiones de los servicios *grid* ofrecidos por cualquier *middleware grid* que gestionan las diferentes entidades *grid* a modo de SAGS (*Semantically Aware Grid Services*), que en la propuesta S-OGSA son los encargados de consumir los `S-Bindings` de dichas entidades *grid* a través de los *Semantic Binding Provisioning Services* disponibles para ello.

En la situación actual, se propone que estos servicios se desarrollen siguiendo la especificación WSRF utilizada en la *grid* para la implementación de servicios (*grid*) web con estado, utilizando el conjunto de operaciones típicas del protocolo HTTP (PUT, GET, POST y DELETE).

Dado que los `WS-Resources` de la especificación tienen asociados `WS-ResourceProperties`, se propone ampliar las propiedades de un determinado recurso *grid* con nuevos elementos de este tipo que contendrán las referencias a los servicios *grid* semánticos (servicios de anotación y de metadatos) asociados al recurso para poder gestionar sus metadatos semánticos o `S-Bindings`, de manera que los programadores de aplicaciones *grid* o los desarrolladores de *middleware grid* puedan consultar y recuperar dichas referencias a través de las operaciones `getProperties` y `queryProperties` existentes en la especificación WSRF.

1. Consumir el servicio *grid* que da acceso a la ontología que implementa el modelo de representación de propiedades no funcionales (planteado en la presente tesis).
   1.1. Identificar las clases que se corresponderían tanto al recurso *grid* como a la propiedad no funcional en cuestión para poder crear más adelante instancias de dichas clases que los representen.
   1.2. Identificar las propiedades de la ontología (no confundir con las propiedades no funcionales) necesarias para crear los `S-Bindings` (Semantic Bindings) para representar asociaciones semánticas (predicados) entre las instancias que representarán tanto al recurso *grid* como a la propiedad no funcional en cuestión.
   1.3. En un escenario ideal, también habría que identificar las clases que se corresponderían a los servicios *grid* implementados para anotarlos semánticamente, ya que también se consideran `G-Resources` que cualquier desarrollador de aplicaciones *grid* podría consumir directamente.
2. Desarrollar dos servicios *grid* (Semantic Binding Provisioning Services):

2.1. Servicio *grid* de anotación para la creación de los `S-Bindings`.

2.1.1. Securizar el acceso a las operaciones del servicio para que solamente el administrador del recurso pueda invocarlas y crear los metadatos o tripletas de información haciendo referencia a las clases, propiedades e instancias identificadas anteriormente.

2.1.2. Seguir los pasos para la implementación del servicio *grid* en GT4 (ver indicaciones al final de esta actividad).*

2.2. Servicio *grid* de metadatos para exponer los `S-Bindings`.

2.2.1. Definir las operaciones que se ofertarán públicamente a través del servicio para devolver metadatos o tripletas de información del recurso.

2.2.2. Seguir los pasos para la implementación del servicio *grid* en GT4 (ver indicaciones al final de esta actividad).*

3. Añadir sendas `WS-ResourceProperties` al `WS-Resource` que representa el recurso *grid* en cuestión con las referencias a los servicios *grid* semánticos implementados.
4. Determinar una frecuencia de actualización de metadatos sobre el recurso *grid* en cuestión en función de las características de la medición realizada sobre la propiedad no funcional en cuestión (diaria, semanal, mensual, anual).

* Pasos para implementar un servicio *grid* en GT4:

1) Definir la interfaz de servicio, utilizando una versión extendida del lenguaje de especificaciones de servicio web (WSDL).
2) Implementar el servicio utilizando la API de Java GT4.
3) Implementar el recurso utilizando la API de Java GT4.
4) Definir los parámetros de despliegue del servicio utilizando un descriptor de despliegue de servicio web (*Web Service Deployment Descriptor* o WSDD) y la interfaz de nombrado y directorio de Java (*Java Naming and Directory Interface* o JNDI).
5) Compilar todo y generar un archivo GAR con lo necesario para realizar el despliegue del servicio en el contenedor de servicios web de GT4. La herramienta *software* "*ant*" permite automatizar los procesos de compilación y construcción de *software*.
6) Desplegar el servicio utilizando las herramientas de GT4.

3.3 Método para representar la fiabilidad de un recurso de computación grid

En este apartado se presenta el método para representar la *fiabilidad* como propiedad no funcional de un recurso de computación o indicador de calidad de servicio del mismo.

Dado que no se disponen de datos reales de partida sobre el recurso de computación en cuestión, la actividad de medición se desarrollará mediante tareas de simulación de datos realistas.

3.3.1 Especificación de la fiabilidad de un recurso de computación grid

Según la RAE [12], la fiabilidad es la *probabilidad de buen funcionamiento de algo*. Según la Wikipedia [89], se define como la probabilidad de que el comportamiento de un sistema o dispositivo desarrolle una determinada función, bajo ciertas condiciones y durante un período de tiempo determinado. Según IEEE (1990b), reemplazado por IEEE (2010b), la definición de fiabilidad (*reliability*) es la habilidad de un sistema o componente para realizar sus funciones bajo unas condiciones dadas durante un período de tiempo específico, es decir, para mejorar sus condiciones en un tiempo dado. Según el *Free On-line Dictionary of Computing* [16] se trata de un atributo de cualquier sistema que consistentemente produce los mismos resultados, preferiblemente alcanzando o superando sus especificaciones.

Como resumen de estas definiciones podemos concluir que *la fiabilidad es la probabilidad de que un elemento funcione de una manera correcta y consistente, entendiendo correcta como la forma esperada y consistente como determinista, en un tiempo determinado y en un entorno dado*, y se calcula así (ver figura 3.9):

$$\text{Fiabilidad} = e^{-\frac{Tiempo}{MTBF}}$$

Figura 3.9 Cálculo de la fiabilidad

Para obtener el factor MTBF (*Mean Time Between Failure*) o tiempo medio entre fallos, hay que determinar las variables necesarias para calcularlo: tamaño de la población, definir todos los posibles fallos, tiempo de diagnóstico y reparación de fallos, así como la tasa anual de fallos o AFR (*Anual Failure Rate*), cuya fórmula es (ver figura 3.10):

$$AFR = \frac{F \times \frac{52 semanas / año}{N}}{P}$$

Figura 3.10 Cálculo de la tasa anual de fallos

donde F es el número de fallos en un mismo periodo, N el número de semanas en dicho periodo y P el tamaño de la población de casos estudiados. Se puede convertir el AFR en el MTBF haciendo (ver figura 3.11):

$$MTBF = \frac{HorasAño}{AFR}$$

Figura 3.11 Cálculo del tiempo medio entre fallos

Los valores MTBF se podrían usar como medida de la fiabilidad del recurso, pero estarían en función del período de tiempo analizado y estos siempre serían números naturales positivos medido en la unidad de tiempo utilizada (segundos, minutos, horas, días).

Otra opción para representar la fiabilidad sería utilizar la inversa de ese valor, que se correspondería con la tasa de fallos del recurso por unidad de tiempo (la que se haya utilizado en la medición).

En cualquier caso, para poder comparar valores sería necesario especificar un rango de valores para poder clasificarlo o categorizarlo (por ejemplo, "muy malo", "malo", "normal", "bueno", "muy bueno").

3.3.2 Medición de la fiabilidad de un recurso de computación grid

Como se menciona en las premisas y restricciones de la presente tesis, los datos reales que se pueden obtener a través de portales públicos de monitorización de entornos *grid* en producción suelen ser de alto nivel y estar agregados ofreciendo solamente cierta información sobre su rendimiento global y sobre su estado actual, por lo que en la mayoría de los casos resulta muy complicado obtener datos de bajo nivel sobre el comportamiento de los recursos individuales de un nodo en concreto. Además, no se suelen publicar este tipo de datos de bajo nivel como tasas de fallos, tiempo medio entre fallos, tiempo medio de reparación, etcétera, ya sea por prohibición explícita del proveedor de su *hardware* o porque exige una dedicación de recogida y organización de los mismos que no suelen realizar los responsables de los nodos porque están

totalmente volcados en las tareas de administración de los mismos. Son contados los equipos de investigación que realizan este tipo de tareas y en la mayoría de los casos guardan con celo sus datasets para sus investigaciones y publicaciones.

Dado que no disponemos de datos reales sobre fallos de recursos que son necesarios para calcular el MTBF que se utiliza para estimar la fiabilidad y que mediante emulación o simulación en ningún caso se llegarán a reproducir las características de un entorno *grid* real, se utilizarán los modelos matemáticos obtenidos tras analizar el estado de la cuestión sobre el comportamiento de recursos de computación en entornos *grid* y similares, como los entornos de computación de altas prestaciones HPC (*High Performance Computing*) y HTC (*High Throughput Computing*).

En este punto se referencia al artículo más completo encontrado (Schroeder y Gibson 2010). Se trata de un estudio basado en datos reales recogidos de manera formal durante más de 10 años en el centro de supercomputación de Los Álamos (EE.UU.) durante dos períodos distintos de unos cinco años cada uno. Los resultados del segundo período se consideran más estables porque el sistema llevaba más tiempo en producción y se evitaban los fallos iniciales de puesta a punto del mismo, así como errores en la documentación de los tiempos de fallos y recuperación.

A continuación se muestran las conclusiones que se han obtenido del artículo de referencia seleccionado tras el estudio realizado:

- Las tasas de fallo varían ampliamente según el sistema y dependen mayoritariamente del tamaño del sistema y menos del *hardware*.
- Las tasas de fallo son aproximadamente proporcionales al número de procesadores en el sistema, aunque ese ratio no crece significativamente más rápido linealmente con el tamaño del sistema.
- Hay una correlación evidente entre la tasa de fallo y la carga de trabajo de un sistema.
- La curva de tasas de fallo para sistemas HPC no se ajusta mucho a ninguna de las funciones estadísticas más comunes, quizá a la log-normal.
- Los tiempos entre fallos no se modelan bien con la distribución exponencial, mientras que sí lo hacen con la gamma o la de Weibull con un *hazard rate* decreciente.

  - En el caso de la de Weibull, los parámetros que mejor se ajustan extraídos del artículo que se toma como referencia son *shape*, o forma, igual a 0.7, y *scale*, o escala, igual a 783203.

- Los tiempos medios de reparación de fallos varían mucho según los sistemas (desde una hora hasta más de un día), dependen del tipo del sistema y son relativamente insensibles al tamaño del sistema. Además, se modelan mejor con la distribución log-normal que con la exponencial.

  - A partir de los datos de la media y la desviación típica extraídos del artículo que se toma como referencia, los estimadores que más se aproximan a los parámetros μ y σ de la distribución log-normal son *meanlog* igual a 5.872066 y *sdlog* igual a 0.1909649.

Como el objetivo es simular datos realistas sobre tiempos entre fallos de un recurso de computación, se propone utilizar el modelo matemático expresado por la función de distribución de Weibull con los parámetros indicados. A continuación se muestra el código utilizado en la herramienta R para generar 200 000 valores simulados a partir del modelo matemático seleccionado.

```
library(fitdistrplus)
#Generate TBF data
set.seed(123)
forma1 <- 0.7
escala1 <- 783203
tbfhosts <- rweibull(n=200000, shape=forma1, scale=escala1)
write.table(tbfhosts, file = "tbfhosts.txt", quote = FALSE, sep
                           = " ", na = "NA", row.names = FALSE,
                           col.names = FALSE )
```

Código 3.1 Simulación de tiempos entre fallos para un recurso de computación

Dado que en el artículo que se toma como referencia los datos utilizados están en segundos, los siguientes (código 3.2) son valores de tiempos entre fallos medidos en segundos.

A partir de los datos simulados sobre tiempos entre fallos de un recurso de computación, se propone utilizar las fórmulas descritas en la actividad anterior para el cálculo de la fiabilidad y concretamente utilizar la inversa del MTBF de cada recurso como indicador, utilizando la escala de Weibull para evitar que los valores sean muy pequeños.

```
1072653.71342914
100677.196213371
667457.33910574
39885.2490894393
14536.7783503543
3922603.73716598
412565.733279376
35124.6751580074
373247.439994709
553133.850285057
...
```

Código 3.2 Resultados tras la simulación de tiempos entre fallos para un recurso de computación

A continuación se muestra el código utilizado en la herramienta R para realizar el cálculo del MTBF de cada recurso (61 en el ejemplo) a partir de los datos sobre tiempos de fallo, así como la fiabilidad de cada uno a partir de la inversa del MTBF escalada.

```
MTBFtbfexphosts <- vector(mode = "numeric", length = 61)
Reliabilitytbfhosts <- vector(mode = "numeric", length = 61)
MTBFtbfhosts[1] <- mean(tbfhosts[1:40])
Reliabilitytbfhosts[1] <- ((1/MTBFtbfhosts[1])*783203)
[...]
write.table(MTBFtbfhosts, file = "MTBFtbfhosts.txt", quote =
                          FALSE, sep = " ", na = "NA",
                          row.names = FALSE, col.names = FALSE
                          )
write.table(Reliabilitytbfhosts, file =
                          "Reliabilitytbfhosts.txt", quote =
                          FALSE, sep = " ", na = "NA",
                          row.names = FALSE, col.names = FALSE
                          )
```

Código 3.3 Simulación de MTBF y fiabilidad para varios recursos de computación

Dado que en el artículo que se toma como referencia los datos utilizados están en segundos, los siguientes (código 3.4) son valores de tiempos medios entre fallos (MTBF) de cada recurso están en segundos.

Después de obtener la inversa del MTBF para utilizar la tasa de fallo como indicador de fiabilidad, dado que en el artículo que se toma como referencia la escala para la función de distribución de Weibull es 783 203, los datos sobre fiabilidad se escalan con dicho valor para evitar que sean muy pequeños (código 3.5).

```
788054.38955634
1278776.65644465
842064.348182274
937039.558029037
635056.26879022
838859.797005687
906321.828810794
1559052.45961172
1249743.83829228
887118.884691636
...
```

Código 3.4 Resultados tras la simulación de MTBF para varios recursos de computación

```
0.993843839180858
0.612462697104215
0.930098752774256
0.835827039839582
1.23328126733084
0.93365184837281
0.864155507572469
0.502358336418683
0.626690827354037
0.882861376885515
...
```

Código 3.5 Resultados tras la simulación de la fiabilidad para varios recursos de computación

3.3.3 Validación de la fiabilidad de un recurso de computación grid

Dado que los datos sobre tiempos entre fallos han sido simulados y generados mediante un modelo matemático teórico, la validación de los mismos es trivial, ya que simplemente hay que comprobar si los mismos cumplen la función de distribución de Weibull, con los parámetros especificados anteriormente, cumplen dicha distribución o no.

Dado que se ha utilizado la herramienta R para generarlos, se puede hacer uso de las funciones que ofrecen para comprobar si un vector de datos se ajusta o no a una determinada distribución, sacando incluso un resumen y una gráfica (ver figura 3.12) que lo confirmen visualmente.

El cálculo de la variable tiempo medio entre fallo (MTBF) es trivial, dado que simplemente es la media de todos los valores utilizados.

```
library(fitdistrplus)
#Fit tbfhosts data with fitdist
fit.w <- fitdist(tbfhosts, "weibull")
summary(fit.w)
plot(fit.w)
```

Código 3.6 Comprobación de la validez de los datos simulados sobre tiempos entre fallos para recursos de computación

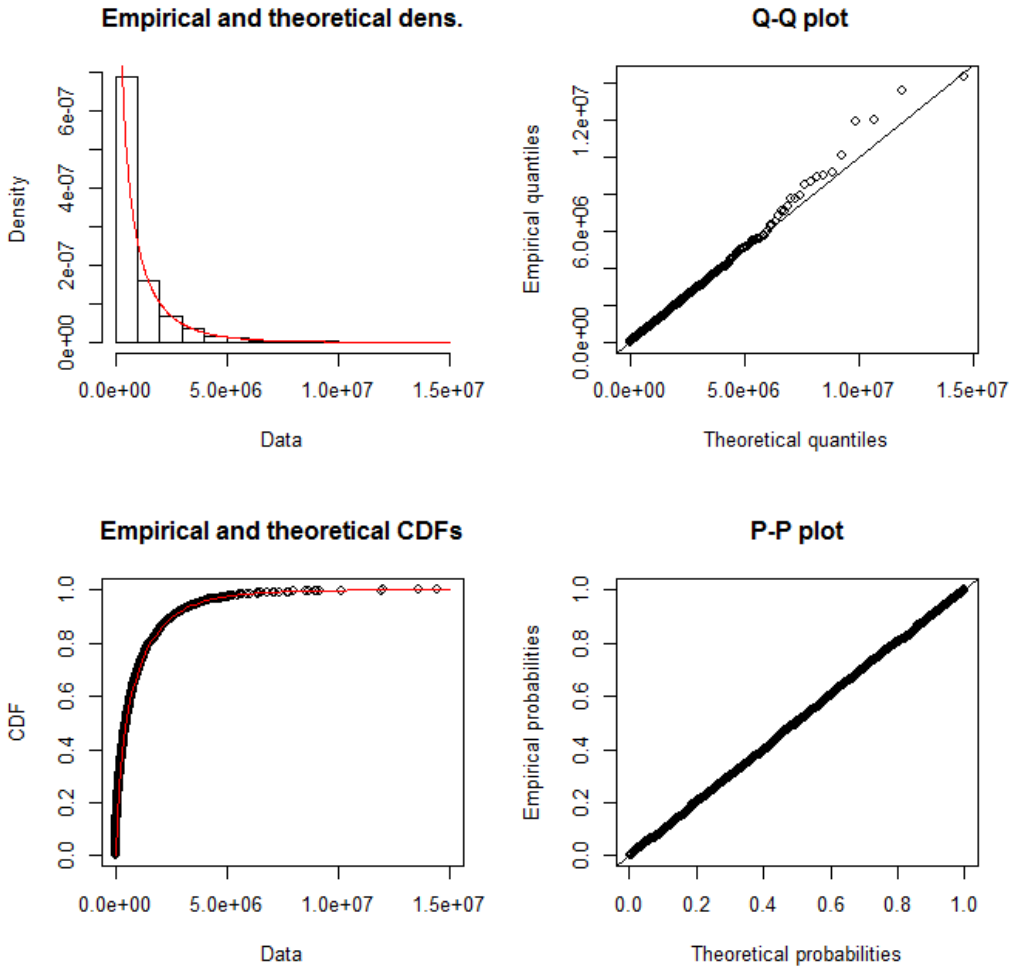

Figura 3.12 Visualización de la validez de los datos simulados sobre tiempos entre fallos para recursos de computación

Igualmente, dado que para calcular el indicador de fiabilidad se obtiene la inversa del MTBF y se multiplica por la escala 783 203, la validación de la operación también es trivial.

En este caso no es necesario solicitar aprobación de ningún responsable del recurso en cuestión, dado que estamos en un escenario de simulación.

3.3.4 Publicación y evolución de la fiabilidad de un recurso de computación grid

Dadas las premisas y restricciones de la presente tesis y la naturaleza de *SimGrid* como herramienta de simulación, no ha lugar a una utilización del modelo de representación de la tesis en esta actividad. En cualquier caso, las clases y propiedades que se utilizarían para generar los metadatos principales correspondientes serían las siguientes (ver tabla 3.1):

Sujeto	Predicado	Objeto
#Host1	rdf:type	nfp4grid:Server
#0.879654	rdf:type	nfp4grid:ComputingReliability
#Host1	nfp4grid:hasQoSAttribute	#0.879654

Tabla 3.1 Metadatos principales para la publicación de la fiabilidad de un recurso de computación

Las tareas de la presente actividad se adaptarán al contexto y la arquitectura de *SimGrid* de la siguiente manera:

- La propiedad no funcional *fiabilidad* se representará a nivel de cada recurso o *host* dentro del fichero que describe la plataforma, dado que el modelo de datos de *SimGrid* para las mismas permite incluir múltiples propiedades individuales sobre los recursos.
- En lugar de servicios *grid*, se implementarán aplicaciones en el lenguaje de programación Python para simular la publicación y consumo de dichas propiedades no funcionales, sin referencia al modelo de la tesis.
- La expresión con los criterios para la selección de recursos que desea el usuario para la ejecución de su trabajo (grafo de tareas) se especificará como parámetro del programa principal, dado que el fichero que describe el grafo de tareas según el modelo de datos de *SimGrid* no permite incluir ese tipo de información.

3.4 Método para representar la disponibilidad de un recurso de computación grid

En este apartado se presenta el método para representar la *disponibilidad* como propiedad no funcional de un recurso de computación o indicador de calidad de servicio del mismo.

Dado que no se disponen de datos reales de partida sobre el recurso de computación en cuestión, la actividad de medición se desarrollará mediante tareas de simulación de datos realistas.

3.4.1 Especificación de la disponibilidad de un recurso de computación grid

Según la RAE [12], la disponibilidad es *la cualidad o condición de disponible, que se puede disponer libremente de una cosa o que está lista para usarse o utilizarse*. Según la Wikipedia [89], el factor de disponibilidad de un equipo o sistema es una medida que nos indica cuánto tiempo está ese equipo o sistema operativo respecto de la duración total durante la que se hubiese deseado que funcionase. Típicamente se expresa en porcentaje. Según el *Free On-line Dictionary of Computing* [16] se trata del grado en el que un sistema sufre degradación o interrupción en su servicio al cliente como consecuencia de fallos de una o más de sus partes.

Barlow y Proschan (1975) definen la disponibilidad de un sistema reparable como *la probabilidad de que el sistema esté funcionando en un momento dado*. Lie, Hwang y Tillman (1977) desarrollan otras definiciones más elaboradas e incluso las clasifican en tipos de disponibilidad, pero todas asumen la anterior.

De todas ellas, en el ámbito de la ingeniería, se suele utilizar la disponibilidad inherente, ya que se basa en utilizar los tiempos medios entre fallos (MTBF) y los tiempos medios de reparación o recuperación (MTTR) para su cálculo.

Dado que en la presente tesis se va a utilizar la disponibilidad como indicador estático basado en datos de tiempos de fallos y tiempos de reparación, se elige esta definición y la fórmula que se utiliza para su cálculo (ver figura 3.13):

$$Disponibilidad = \frac{MTBF}{MTBF + MTTR}$$

Figura 3.13 Cálculo de la disponibilidad

En este caso, los valores MTTR también estarían en función del período de tiempo analizado y siempre serían números naturales positivos medidos en la unidad de tiempo utilizada (segundos, minutos, horas, días).

Sin embargo, por la forma de calcular los valores de disponibilidad, estos serían porcentajes o valores entre 0 y 1, por lo que son más fáciles de comparar y clasificar.

3.4.2 Medición de la disponibilidad de un recurso de computación grid

Dado que no disponemos de datos reales sobre fallos de recursos que son necesarios para calcular el MTBF y el MTTR que se utilizan para estimar la disponibilidad, y aplicando los mismos argumentos mencionados en el método de representación de la fiabilidad, se utilizarán los modelos matemáticos obtenidos del mismo artículo de referencia seleccionado tras el estudio realizado (Schroeder y Gibson 2006).

Las conclusiones obtenidas respecto a los tiempos medios de reparación son las siguientes:

- Los tiempos medios de reparación de fallos varían mucho según los sistemas (desde una hora hasta más de un día), dependen del tipo del sistema y son relativamente insensibles al tamaño del sistema. Además, se modelan mejor con la distribución log-normal que con la exponencial.
    - A partir de los datos de la media y la desviación típica extraídos del artículo que se toma como referencia, los estimadores que más se aproximan a los parámetros μ y σ de la distribución log-normal son *meanlog* igual a 5.872066 y *sdlog* igual a 0.1909649.

Igualmente, se aprovechan los desarrollos y cálculos realizados para la simulación de tiempos medios entre fallos o MTBF de los recursos de computación.

Dado que la tarea pendiente es simular datos realistas sobre tiempos de reparación de fallos de un recurso de computación, se propone utilizar el modelo matemático expresado por la función de distribución log-normal con los parámetros indicados. A continuación (código 3.7) se muestra el código utilizado en la herramienta R para generar 200 000 valores simulados a partir del modelo matemático seleccionado.

```
#Generate TTR data
meanlog <- 5.872066
sdlog <- 0.1909649
ttrhosts <- rlnorm(200000, meanlog , sdlog)
write.table(ttrexp4hosts, file = "ttrhosts.txt", quote = FALSE,
                           sep = " ", na = "NA", row.names =
                           FALSE, col.names = FALSE )
```

Código 3.7 Simulación de tiempos de reparación de fallos para un recurso de computación

Dado que en el artículo que se toma como referencia los datos utilizados están en minutos, los siguientes (código 3.8) son valores de tiempos de reparación de fallos están en minutos.

```
393.33259119866
302.337704788695
482.088705411154
360.976347462305
407.70369471092
471.445624160332
339.206383133082
377.907140438658
265.596089660917
418.461802501902
...
```

Código 3.8 Resultados tras la simulación de tiempos de reparación de fallos para un recurso de computación

A partir de los datos simulados sobre tiempos de reparación de fallos de un recurso de computación, se propone utilizar la fórmula descrita en la actividad anterior para el cálculo de la disponibilidad.

A continuación (código 3.9) se muestra el código utilizado en la herramienta R para realizar el cálculo del MTTR de cada recurso (61 en el ejemplo) a partir de los datos sobre tiempos de reparación de fallos, así como la disponibilidad de cada uno a partir de su MTBF (obtenido en la actividad de medida del método de medición de su fiabilidad) y su MTTR.

```
MTBFtbfexphosts <- vector(mode = "numeric", length = 61)
MTTRttrexphosts <- vector(mode = "numeric", length = 61)
Availabilitytbfhosts <- vector(mode = "numeric", length = 61)
MTBFtbfhosts[1] <- mean(tbfhosts[1:40])
MTTRttrhosts[1] <- mean(ttrhosts[1:40])
Availabilitytbfhosts[1] <- (MTBFtbfhosts[1]/(MTBFtbfhosts[1] +
                           (60*MTTRttrhosts[1])))
...
write.table(MTBFtbfhosts, file = "MTBFtbfhosts.txt", quote =
                           FALSE, sep = " ", na = "NA",
                           row.names = FALSE, col.names = FALSE
                           )
write.table(MTTRttrhosts, file = "MTTRttrhosts.txt", quote =
                           FALSE, sep = " ", na = "NA",
                           row.names = FALSE, col.names = FALSE
                           )
write.table(Availabilitytbfhosts, file =
                           "Availabilitytbfhosts.txt", quote =
                           FALSE, sep = " ", na = "NA",
                           row.names = FALSE, col.names = FALSE
                           )
```

Código 3.9 Simulación de MTTR, MTBF y disponibilidad para varios recursos de computación

Dado que en el artículo que se toma como referencia los datos utilizados están en minutos, los siguientes valores (código 3.10) de tiempos medios de reparación entre fallos (MTTR) de cada recurso están en minutos.

```
363.128085106174
363.10962759186
369.235675436744
354.667971796158
349.613325869334
360.72478488583
359.176629984422
364.098512995305
323.847039255338
361.192674755265
...
```

Código 3.10 Resultados tras la simulación de MTTR para varios recursos de computación

A continuación (código 3.11) se muestran los valores de disponibilidad (según la fórmula de la figura 3.13). Dado que los datos sobre disponibilidad ya están en un rango de valores de 0 a 1, no se considera realizar ninguna modificación de unidad ni escala.

```
0.973096377938322
0.983248351879957
0.974365116211165
0.977794383522065
0.968024783254962
0.974847873920881
0.976774180177113
0.986181331331053
0.984690190438087
0.976153401919739
...
```

Código 3.11 Resultados tras la simulación de la disponibilidad para varios recursos de computación

3.4.3 Validación de la disponibilidad de un recurso de computación grid

Los datos sobre tiempo medio entre fallos (MTBF) ya han sido validados en el método correspondiente. Dado que los datos sobre tiempos de recuperación de fallos (MTTR) han simulados y generados mediante un modelo matemático teórico, la validación de los mismos es trivial, ya que simplemente hay que comprobar si los mismos cumplen la función de distribución log-normal, con los parámetros especificados anteriormente, cumplen dicha distribución o no.

Dado que se ha utilizado la herramienta R para generarlos, se puede hacer uso (*código 3.12*) de las funciones que ofrecen para comprobar si un vector de datos se ajusta o no a una determinada distribución, sacando incluso un resumen y una gráfica que lo confirmen visualmente (ver *figura 3.14*).

El cálculo de la variable tiempo medio de recuperación de fallos (MTTR) es trivial, dado que simplemente es la media de todos los valores utilizados.

Igualmente, dado que para calcular el indicador de disponibilidad se obtiene como un ratio del MTBF dividido por la suma de MTBF y MTTR, la validación de la operación también es trivial.

En este caso no es necesario solicitar aprobación de ningún responsable del recurso en cuestión, dado que estamos en un escenario de simulación.

```
library(fitdistrplus)
#Fit ttrhosts data with fitdist
#fit.w <- fitdist(ttrhosts, "lnorm")
#summary(fit.w)
#plot(fit.w)
```

Código 3.12 Comprobación de la validez de los datos simulados sobre tiempos de reparación de fallos para recursos de computación

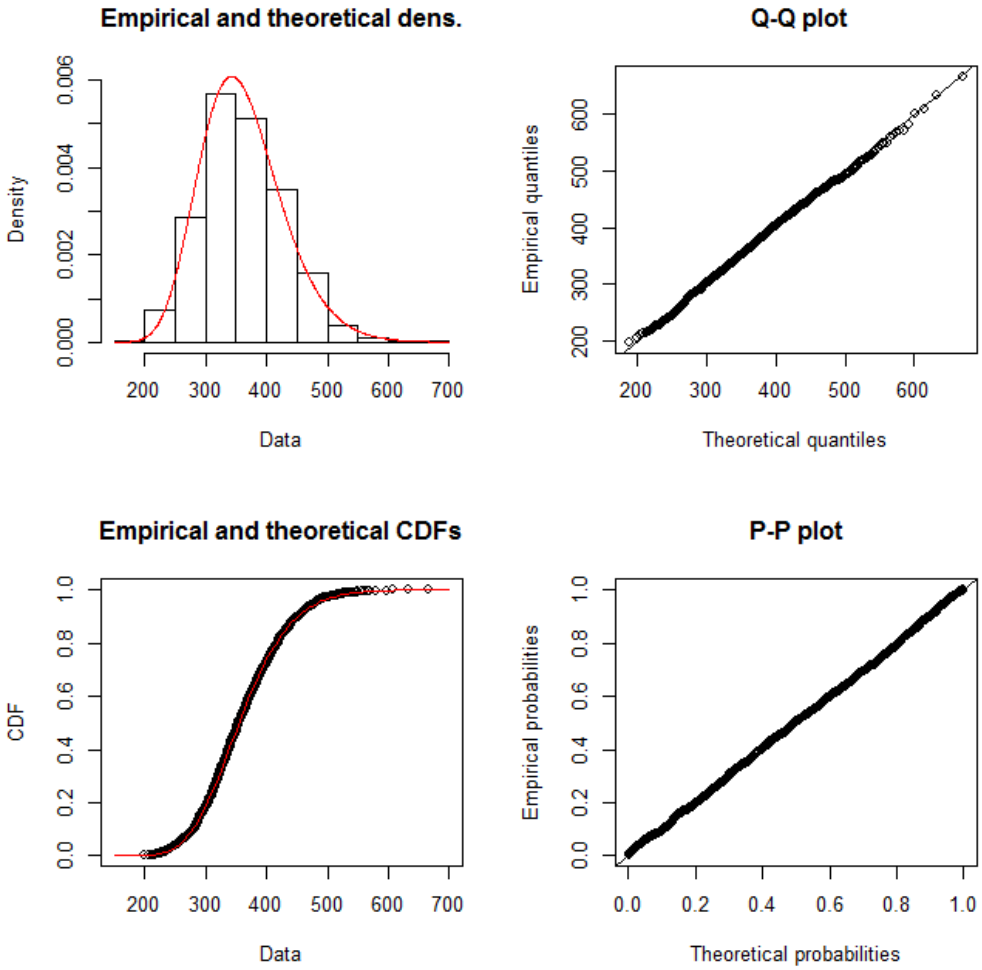

Figura 3.14 Visualización de la validez de los datos simulados sobre tiempos de recuperación de fallos para recursos de computación

3.4.4 Publicación y evolución de la disponibilidad de un recurso de computación grid

Dadas las premisas y restricciones de la presente tesis y la naturaleza de *SimGrid* como herramienta de simulación, no ha lugar a una utilización del modelo de representación de la tesis en esta actividad. En cualquier caso, las clases y propiedades que se utilizarían para generar los metadatos correspondientes serían las siguientes (ver tabla 3.2):

Sujeto	Predicado	Objeto
`#Host1`	`rdf:type`	`nfp4grid:Server`
`#0.98765`	`rdf:type`	`nfp4grid:HistoricalAvailability`
`#Host1`	`nfp4grid:hasQoSAttribute`	`#0.98765`

Tabla 3.2 Metadatos principales para la publicación de la disponibilidad histórica de un recurso de computación

Las tareas de la presente actividad se adaptarán al contexto y la arquitectura de *SimGrid* de la siguiente manera:

- La propiedad no funcional *disponibilidad* se representará a nivel de cada recurso o *host* dentro del fichero que describe la plataforma, dado que el modelo de datos de *SimGrid* para las mismas permite incluir múltiples propiedades individuales sobre los recursos.
- En lugar de servicios *grid*, se implementarán aplicaciones en el lenguaje de programación Python para simular la publicación y consumo de dichas propiedades no funcionales, sin referencia al modelo de la tesis.
- La expresión con los criterios para la selección de recursos que desea el usuario para la ejecución de su trabajo (grafo de tareas) se especificará como parámetro del programa principal, dado que el fichero que describe el grafo de tareas según el modelo de datos de *SimGrid* no permite incluir ese tipo de información.

3.5 Método para representar la fiabilidad de un nodo grid

En este apartado se introduce el método para representar la fiabilidad como propiedad no funcional o indicador de calidad de servicio de todo un nodo *grid*.

Dado que la fiabilidad de un nodo *grid* viene determinada por la fiabilidad de otros elementos con los que está conectado y de los que depende dicho nodo, tanto la definición como la medición de esta propiedad no funcional muestra una enorme complejidad y dificultad, por lo que se considera que excede el alcance de la presente tesis.

Como bien menciona Dabrowski (2009), desarrollar métodos para mejorar la fiabilidad en la *grid* sigue siendo un desafío a largo plazo. La mayoría de las iniciativas se basan en sistemas tolerantes a fallos y existen muy pocos intentos de medir distintas variables que pudieran servir para predecir o estimar la fiabilidad de la *grid*.

En cualquier caso, a modo de punto de partida, en esta sección se desarrollan parcialmente las dos primeras actividades del método para orientar cómo se podría abordar su desarrollo.

Uno de los trabajos teóricos más interesantes sobre fiabilidad en la *grid* es de Xie, Poh y Dai (2004), por lo que se propone utilizar dicha referencia para abordar el desarrollo de este método.

3.5.1 Especificación de la fiabilidad de un nodo grid

Adaptando la idea general de fiabilidad al entorno *grid* que nos ocupa y partiendo de las definiciones dadas en Xie, Poh y Dai (2004), podemos identificar y relacionar diferentes tipos de fiabilidad a distintos niveles y para distintos tipos de recursos o elementos de una *grid*:

- *Fiabilidad de programa grid*. Probabilidad de que un programa dado se ejecute satisfactoriamente sobre múltiples nodos virtuales, incluyendo el intercambio de información a través de enlaces virtuales con recursos remotos, en el ámbito de un sistema de computación *grid*.
- *Fiabilidad de sistema grid*. Probabilidad de que todos los programas involucrados en un sistema *grid* concreto se ejecuten correctamente.
- *Fiabilidad de servicio grid*. Probabilidad de que todos los programas relacionados con un determinado servicio se ejecuten correctamente para dar el servicio requerido. Un servicio *grid* puede implicar la ejecución de varios programas que utilicen recursos distribuidos en la *grid*. Esta fiabilidad es similar a la de un sistema *grid*, pero considerando únicamente los programas implicados en el servicio.
- *Fiabilidad grid*. Probabilidad obtenida al incorporar varios aspectos de la infraestructura *grid*, incluyendo el sistema de gestión de recursos, la

red de comunicación, así como los recursos *hardware* y *software* integrados en la misma. En primera instancia está determinada principalmente por la fiabilidad del sistema de gestión de recursos, aunque también hay una componente muy importante relacionada con la fiabilidad de la red de comunicación. Si se quiere mayor nivel de detalle, habrá que integrar también la fiabilidad de los recursos *software* y *hardware* involucrados.

3.5.2 Medición de la fiabilidad de un nodo grid

Se pueden reutilizar los modelos matemáticos que se proponen en Xie, Poh y Dai (2004) para estimar todos estos tipos de fiabilidad, pero también se puede diseñar un sistema de medida independiente de los métodos utilizados para calcular los valores concretos de fiabilidad para cada tipo de recurso *grid*. Dado que existen múltiples variables que se utilizan en el cálculo de cada tipo de fiabilidad y que puede suceder que los proveedores de información no ofrezcan algunas de ellas o que no utilicen las mismas fórmulas para obtenerlas, se sugiere la utilización de técnicas de aprendizaje automático o incluso redes bayesianas que permitan aprender cómo se relacionan las variables usadas en las definiciones anteriores.

Para ello se definen qué variables utilizar, qué rango de valores o fórmula (distribución) representa a cada una de ellas, las relaciones que se pueden establecer entre ellas a partir de las definiciones dadas en Xie, Poh y Dai (2004). Al ejecutar una red bayesiana con valores experimentales se puede intentar inferir qué variables están correlacionadas y cuáles podrían ser estimadas a partir de otras. Si no se dispone de proveedores de información que aporten valores de fiabilidad de un recurso, éste se podrá estimar a través de la red bayesiana, por ejemplo.

Una propuesta de relaciones entre conceptos de fiabilidad que se puede usar para diseñar la red bayesiana es la siguiente (ver figura 3.15): *resource management service failure ratio* (λ_{RMS}), *service time* (t_s), *size information exchanged / load network connection* (D), *network bandwidth / speed* (S), *communication time* (T_c), *link failure ratio* ($\lambda_{i,j}$), *host client failure ratio* (λ_i), *host server failure ratio* (λ_j), *run time program* (t_m), *usage time program* (t_u), *RMS reliability* (R_{RMS}), *link reliability* (R_L), *communication reliability* (R_c), *host program reliability* (R_{prog}), *host resource reliability* (R_{res}), *grid resource program reliability* (R_{GPR}), *grid node system reliability* (R_S), *grid resource program reliability with RMS reliability* ($R_{GPR\text{-}RMS}$), *grid node system reliability with RMS reliability* ($R_{S\text{-}RMS}$).

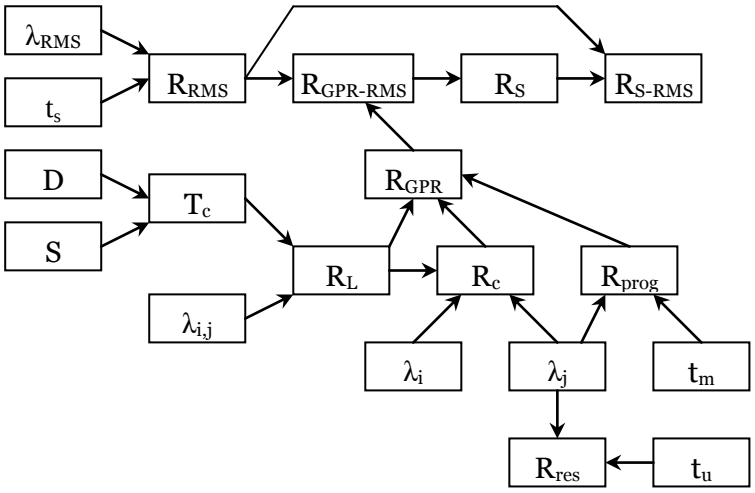

Figura 3.15 Relaciones entre variables que permiten calcular distintos tipos de fiabilidad asociadas a una grid

3.5.3 Validación de la fiabilidad de un nodo grid

No se desarrolla esta actividad en la presente tesis, si bien las tareas para llevar a cabo la misma están especificadas en el apartado correspondiente al proceso de desarrollo de los métodos de la metodología.

3.5.4 Publicación y evolución de la fiabilidad de un nodo grid

No se desarrolla esta actividad en la presente tesis, si bien las tareas para llevar a cabo la misma están especificadas en el apartado correspondiente al proceso de desarrollo de los métodos de la metodología.

3.6 Nuevas capacidades en el descubrimiento de recursos

A continuación se describen los desarrollos realizados sobre el modelo de pruebas *SimDag* existente en el entorno de simulación *SimGrid* para introducir nuevas capacidades en el subalgoritmo de selección de recursos que se implementa dentro del algoritmo min-min de planificación de trabajos existente. Se realizan aportaciones tanto en su modelo de datos como en su modelo de pruebas, que están más detallados en el apéndice B. El objetivo de estas modificaciones es poder evaluar el efecto de utilizar propiedades no funcionales en la selección de recursos respecto al comportamiento global del

algoritmo. Para ello, se analizarán los tiempos de finalización de cada experimento y los fallos que se producen.

Dadas las premisas y restricciones de la presente tesis y la naturaleza de *SimGrid* como herramienta de simulación, no ha lugar a una utilización del modelo de representación de la tesis en los desarrollos que se describen a continuación.

Sin embargo, sí se hará uso de la metodología de la tesis, dado que se utilizarán los resultados de implementar los métodos de representación de la fiabilidad y la disponibilidad utilizando datos realistas simulados. Las tareas de aquellas actividades que requieran el uso del modelo de representación de la tesis se adaptarán al contexto y la arquitectura de *SimGrid*:

- Las propiedades no funcionales (fiabilidad y disponibilidad) se representarán a nivel de cada recurso o *host* dentro del fichero que describe la plataforma (clúster, nodo o *grid*), dado que el modelo de datos de *SimGrid* para las mismas permite incluir múltiples propiedades individuales sobre los recursos.
- En lugar de servicios *grid*, se implementarán aplicaciones en el lenguaje de programación Python para simular la publicación y consumo de dichas propiedades no funcionales, sin referencia al modelo de la tesis.
- La expresión con los criterios para la selección de recursos que desea el usuario para la ejecución de su trabajo (grafo de tareas) se especificará como parámetro del programa principal, dado que el fichero que describe el grafo de tareas según el modelo de datos de *SimGrid* no permite incluir ese tipo de información.

3.6.1 Principales características del modelo de pruebas SimDag del entorno de simulación SimGrid

SimGrid tiene seis principales componentes (ver figura 3.16): *SimDag*, MSG, SMPI, XBT y SURF, de los cuales *SimDag*, MSG y SMPI se consideran API de programación o *toolkits* de *SimGrid*.

SimDag provee funcionalidades para simular la planificación y ejecución de tareas paralelas, dependientes o no, descritas siguiendo un modelo de grafos acíclicos dirigidos (*Direct Acyclic Graphs* o DAG) sobre una plataforma de recursos de computación descrita siguiendo un formato XML determinado y la implementación del popular algoritmo de planificación min-min en el lenguaje C.

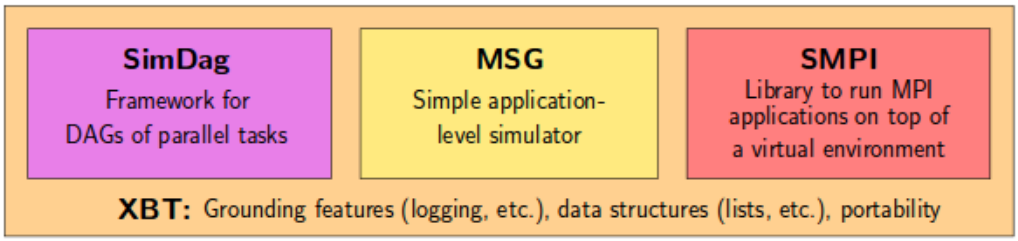

Figura 3.16 Componentes de SimGrid

Para ejecutar una simulación en *SimDag* se necesita:

- El código con el algoritmo de planificación que se desea ejecutar en la simulación.
- Una descripción de la plataforma en la que se ejecuta la aplicación (un archivo XML que describe principalmente la configuración y características de sus estaciones de trabajo y sus enlaces de comunicación).
- Un fichero con el grafo a ejecutar sobre la plataforma (un fichero XML con información sobre los trabajos y sus dependencias como, por ejemplo, tiempo de ejecución, subtareas de transferencia de datos entre tareas, etcétera).

SimDag posee un modelo de datos con diferentes tipos o estructuras de datos para cargar en memoria y manejar la plataforma con recursos de computación (*workstations*), el grafo de tareas (*tasks*) y sus dependencias (*task dependencies*), así como otras estructuras de datos intermedias necesarias para la simulación. Destacamos a continuación las más relevantes:

- `SD_workstation_t`: una *workstation* o estación de trabajo es el recurso de computación físico sobre el que una tarea puede ser ejecutada.
- `SD_task_t`: Una *task* o tarea es una cantidad de trabajo a computar que puede ser ejecutada en paralelo sobre varias *workstations*. Puede depender de otras tareas, por lo que en ese caso la tarea no puede comenzar hasta que las tareas de las que depende estén finalizadas. Además, cada tarea posee un estado para indicar si está planificada, en ejecución, finalizada, etcétera.

SimDag también dispone de un modelo funcional implementado a través de una API o conjunto de funciones para manejar las diferentes estructuras de datos, tal y como se puede ver en el apéndice B de la presente tesis.

Por último, *SimDag* también ofrece un modelo de pruebas basado en una plataforma formada por un clúster de 7 estaciones de trabajo, un grafo con 25 tareas y el algoritmo de planificación min-min implementado. Aunque durante el análisis de la herramienta se encontraron referencias a otros algoritmos, no se ha podido acceder a las implementaciones de los mismos.

El algoritmo de planificación min-min tiene el siguiente funcionamiento general respecto a la asignación de recursos a cada tarea y el orden de planificación de las mismas:

- Para cada tarea ejecutable selecciona el mejor recurso de computación disponible (aquel que minimiza su tiempo de finalización o *completion time*).
- Selecciona la tarea que tenga el menor tiempo de finalización de entre todas.
- Planifica la tarea seleccionada sobre el recurso de computación que se le asoció en el primer paso del algoritmo.

La implementación del algoritmo min-min en *SimDag* se apoya básicamente en las siguientes funciones:

- `double finish_on_at(SD_task_t task, SD_workstation_t workstation)`
    - Esta función estima el tiempo de finalización o EFT (*Earliest Finish Time*) de una determinada tarea ejecutable sobre una determinada estación de trabajo.
    - El cálculo tiene en cuenta cuándo quedaría libre la estación de trabajo tras ejecutar las tareas que tiene actualmente asignadas (incluido las subtareas de transferencia de datos, si las hubiera), el tiempo que falta para que terminen las tareas de las que depende la tarea que se está evaluando (si las hubiera) y el tiempo necesario para ejecutar la computación de la tarea en cuestión.
- `SD_workstation_t SD_task get_best_workstation (SD_task_t task)`
    - Encuentra la estación de trabajo en la que el tiempo de finalización o EFT para la tarea en cuestión sería mínimo.
    - Para ello, recorre todas las estaciones de trabajo e invoca a la función `finish_on_at`.

- `xbt_dynar_t SD_simulate (double how_long)`
    - Lanza la simulación ejecutando las tareas en estado ejecutable.
    - En el caso de la implementación de min-min en *SimDag*, la función recibe un parámetro negativo para indicar que la simulación no tiene un tiempo límite, por lo que esta parará cuando no haya más tareas ejecutables o cuando se vayan alcanzando los distintos puntos de control o *watchpoints* que se pueden indicar en el resto del código.
    - En el caso de min-min, se añade un punto de control o *watchpoint* cada vez que finaliza una tarea.
    - En cada parada, la función devuelve el conjunto de tareas cuyo estado ha cambiado.

El pseudocódigo del programa principal del algoritmo min-min aparece en el código 3.13.

```
- Carga del entorno
  * Asigna atributos para todas las estaciones de trabajo
- Carga el DAX o grafo de tareas
- Añade watchpoints sobre el estado SD_DONE para todos los
                          trabajos
- Planifica el nodo raíz del grafo en la primera estación de
                          trabajo
- Mientras no pare la simulación
  * Obtiene las tareas preparadas para ejecutarse (salta ciclo
                          si no hay)
  * Obtiene la tarea con el menor EFT de dichas tareas
    + Obtiene la mejor estación de trabajo para cada tarea
    + Calcula el EFT de cada tarea sobre su mejor estación de
                          trabajo
  * Planifica la tarea seleccionada sobre su mejor estación de
                          trabajo
  * Actualiza estados de estaciones de trabajo y dependencias
                          entre tareas
  * Reinicia estructuras de datos intermedias
- Obtiene el tiempo de finalización de la simulación
- Genera un fichero XML con datos sobre la ejecución de cada
                          tarea
- Vacía la memoria y finaliza el programa
```

Código 3.13 *Pseudocódigo del algoritmo min-min de SimDag en SimGrid*

3.6.2 Aportaciones al modelo de datos de SimGrid

A continuación se detallan las aportaciones realizadas sobre la descripción de las plataformas que se utilizarán en la evaluación de la tesis.

3.6.2.1 Generación de datos relativos a fallos y propiedades no funcionales sobre hosts o recursos de computación

A partir de las actividades de medición de las propiedades no funcionales fiabilidad y disponibilidad para recursos de computación descritas en los métodos de la metodología de la presente tesis, se obtienen datos sobre instantes de fallo y recuperación, así como sobre propiedades no funcionales fiabilidad y disponibilidad, para todos los recursos de computación de una plataforma o nodo concreto.

Como se puede ver en el código 3.14 en el lenguaje Python, a partir de los ficheros con tiempos entre fallos y tiempos de reparación de fallos, se crean múltiples ficheros de traza de comportamiento durante un año (31 536 000 segundos) para diferentes recursos. Se asume que se trata de una plataforma en producción y que todos los recursos están activos en el instante 0.

Además, durante la creación del fichero de traza para cada recurso, se guardan todos los tiempos entre fallos, el número de fallos y los tiempos de reparación de los mismos para cada recurso.

La información descrita anteriormente se va volcando a un fichero en forma de código para la herramienta R de manera que se vayan creando estructuras de datos para almacenar la información sobre cada recurso y utilizarla posteriormente en el cálculo de los siguientes indicadores de cada recurso: MTBF, MTTR, fiabilidad y disponibilidad (código 3.15).

El programa va leyendo datos de ambos ficheros (código 3.16), tiempos entre fallos y tiempos de reparación de fallos y los va combinando con estos criterios:

- Se asume que se trata de una plataforma en producción y que todos los recursos están activos en el instante 0.
- El tiempo leído del fichero de tiempos entre fallos se suma al último instante en el que estaba activo el recurso, generando con el resultado un instante de fallo o estado inactivo.
- El tiempo leído del fichero de tiempos de reparación de fallos se suma al último instante de fallo, generando con el resultado un instante de recuperación o de estado activo.
- Cuando se alcanza un instante de estado activo superior a un año, se reinician variables y se crea un nuevo fichero de traza para un nuevo recurso.

```
import sys
import os
import math
if len(sys.argv) != 3:
      print 'python %s tbfhosts.txt ttrhosts.txt' % sys.argv[0]
else:
      tbfFile = open(sys.argv[1])
      ttrFile = open(sys.argv[2])
      outputdir = os.path.join( os.getcwd(),
                           sys.argv[1].rstrip('.txt') )
      if not os.access(outputdir, os.F_OK):
            os.mkdir(outputdir)
      host = 1
      tFile = "host%s.trace" % host
      fFile = "fails%s" % sys.argv[1]
      traceFile = open( os.path.join(outputdir,tFile), "w" )
      failsFile = open( os.path.join(outputdir,fFile), "w" )
      MTBFdata = "MTBF%s" % sys.argv[1]
      MTTRdata = "MTTR%s" % sys.argv[2]
      ReliabilityData = "Reliability%s" % sys.argv[1]
      AvailabilityData = "Availability%s" % sys.argv[1]
```

Código 3.14 Código de la herramienta para generar ficheros para el modelo de pruebas de SimDag en SimGrid (parte 1)

```
      fTime = 0
      rTime = 0
      fails = 0
      firstFail = 1
      lastFail = firstFail+fails-1
      traceLine = "%s %s\n" % (fTime,1)
      traceFile.write( traceLine )
      failsLine = "# CAMBIAR EL TAM DE LOS VECTORES DENTRO DEL
                           ARCHIVO fails GENERADO PARA R"
      failsFile.write( failsLine )
      failsLine = "%s <- vector(mode = \"numeric\", length =
                           61)\n" % MTBFdata.rstrip('.txt')
      failsFile.write( failsLine )
      failsLine = "%s <- vector(mode = \"numeric\", length =
                           61)\n" % MTTRdata.rstrip('.txt')
      failsFile.write( failsLine )
      failsLine = "%s <- vector(mode = \"numeric\", length =
                           61)\n" %
                           ReliabilityData.rstrip('.txt')
      failsFile.write( failsLine )
      failsLine = "%s <- vector(mode = \"numeric\", length =
                           61)\n" %
                           AvailabilityData.rstrip('.txt')
      failsFile.write( failsLine )
```

Código 3.15 Código de la herramienta para generar ficheros para el modelo de pruebas de SimDag en SimGrid (parte 2)

```
      tbfData = tbfFile.readline()
      ttrData = ttrFile.readline()
      while tbfData != "":
            fTime = rTime+float(tbfData);
            traceLine = "%s %s\n" % (int(fTime),0)
            traceFile.write( traceLine )
            rTime = fTime+(float(ttrData)*60);
            traceLine = "%s %s\n" % (int(rTime),1)
            traceFile.write( traceLine )
            fails = fails + 1
            lastFail = firstFail+fails-1
            if rTime >= 31536000:
                  traceFile.close()
                  failsLine = "%s[%s] <- mean(%s[%s:%s])\n" %
                              (MTBFdata.rstrip('.txt'),host,sys.ar
                              gv[1].rstrip('.txt'),firstFail,lastF
                              ail)
                  failsFile.write( failsLine )
                  failsLine = "%s[%s] <- mean(%s[%s:%s])\n" %
                              (MTTRdata.rstrip('.txt'),host,sys.ar
                              gv[2].rstrip('.txt'),firstFail,lastF
                              ail)
                  failsFile.write( failsLine )
                  failsLine = "%s[%s] <- ((1/%s[%s])*783203)\n" %
                              (ReliabilityData.rstrip('.txt'),host
                              ,MTBFdata.rstrip('.txt'),host)
                  failsFile.write( failsLine )
                  failsLine = "%s[%s] <-
                              (%s[%s]/(%s[%s]+(60*%s[%s])))\n" %
                              (AvailabilityData.rstrip('.txt'),hos
                              t,MTBFdata.rstrip('.txt'),host,MTBFd
                              ata.rstrip('.txt'),host,MTTRdata.rst
                              rip('.txt'),host)
                  failsFile.write( failsLine )
                  host = host + 1
                  tFile = "host%s.trace" % host
                  traceFile = open(
                              os.path.join(outputdir,tFile), "w" )
                  fTime = 0
                  rTime = 0
                  fails = 0
                  firstFail = lastFail + 1
                  traceLine = "%s %s\n" % (fTime,1)
                  traceFile.write( traceLine )
            tbfData = tbfFile.readline()
            ttrData = ttrFile.readline()
```

Código 3.16 Código de la herramienta para generar ficheros para el modelo de pruebas de SimDag en SimGrid (parte 3)

Una vez generados los ficheros de traza con instantes de fallo y recuperación de cada recurso (código 3.17), la última información que se vuelca al fichero de salida con código para la herramienta R es la que permitirá calcular los

siguientes indicadores sobre cada recurso: MTBF, MTTR, fiabilidad y disponibilidad.

```
      failsLine = "%s[%s] <- mean(%s[%s:%s])\n" %
                           (MTBFdata.rstrip('.txt'),host,sys.ar
                           gv[1].rstrip('.txt'),firstFail,lastF
                           ail)
      failsFile.write( failsLine )
      failsLine = "%s[%s] <- mean(%s[%s:%s])\n" %
                           (MTTRdata.rstrip('.txt'),host,sys.ar
                           gv[2].rstrip('.txt'),firstFail,lastF
                           ail)
      failsFile.write( failsLine )
      failsLine = "%s[%s] <- ((1/%s[%s])*783203)\n" %
                           (ReliabilityData.rstrip('.txt'),host
                           ,MTBFdata.rstrip('.txt'),host)
      failsFile.write( failsLine )
      failsLine = "%s[%s] <- (%s[%s]/(%s[%s]+(60*%s[%s])))\n" %
                           (AvailabilityData.rstrip('.txt'),hos
                           t,MTBFdata.rstrip('.txt'),host,MTBFd
                           ata.rstrip('.txt'),host,MTTRdata.rst
                           rip('.txt'),host)
      failsFile.write( failsLine )

      failsLine = "write.table(%s, file = \"%s\", quote = FALSE,
                           sep = \" \", na = \"NA\", row.names
                           = FALSE, col.names = FALSE )\n" %
                           (MTBFdata.rstrip('.txt'),MTBFdata)
      failsFile.write( failsLine )
      failsLine = "write.table(%s, file = \"%s\", quote = FALSE,
                           sep = \" \", na = \"NA\", row.names
                           = FALSE, col.names = FALSE )\n" %
                           (MTTRdata.rstrip('.txt'),MTTRdata)
      failsFile.write( failsLine )
      failsLine = "write.table(%s, file = \"%s\", quote = FALSE,
                           sep = \" \", na = \"NA\", row.names
                           = FALSE, col.names = FALSE )\n" %
                           (ReliabilityData.rstrip('.txt'),Reli
                           abilityData)
      failsFile.write( failsLine )
      failsLine = "write.table(%s, file = \"%s\", quote = FALSE,
                           sep = \" \", na = \"NA\", row.names
                           = FALSE, col.names = FALSE )\n" %
                           (AvailabilityData.rstrip('.txt'),Ava
                           ilabilityData)
      failsFile.write( failsLine )
      tbfFile.close();
      ttrFile.close();
      traceFile.close()
      failsFile.close()
```

Código 3.17 Código de la herramienta para generar ficheros para el modelo de pruebas de SimDag en SimGrid (parte 4)

En el fichero de traza que se genera para cada recurso (código 3.18) se indica con un 1 el instante en el que está activo un recurso y con un 0 el instante en el que falla.

```
0 1
1072653 0
1096253 1
1196930 0
1215071 1
1882528 0
1911453 1
1951339 0
1972997 1
1987534 0
...
```

Código 3.18 Ejemplo de fichero de traza con instantes de fallo y recuperación de un recurso de computación dentro una plataforma para SimGrid

El otro fichero generado por la herramienta (código 3.19) es un conjunto de instrucciones para ejecutar en la herramienta R con el objetivo de obtener los indicadores mencionados para cada recurso: MTBF, MTTR, fiabilidad y disponibilidad.

```
MTBFtbfexphosts <- vector(mode = "numeric", length = 61)
MTTRttrexphosts <- vector(mode = "numeric", length = 61)
Availabilitytbfhosts <- vector(mode = "numeric", length = 61)
MTBFtbfhosts[1] <- mean(tbfhosts[1:40])
MTTRttrhosts[1] <- mean(ttrhosts[1:40])
Availabilitytbfhosts[1] <- (MTBFtbfhosts[1]/(MTBFtbfhosts[1] +
                            (60*MTTRttrhosts[1])))
...
write.table(MTBFtbfhosts, file = "MTBFtbfhosts.txt", quote =
                            FALSE, sep = " ", na = "NA",
                            row.names = FALSE, col.names = FALSE
                            )
write.table(MTTRttrhosts, file = "MTTRttrhosts.txt", quote =
                            FALSE, sep = " ", na = "NA",
                            row.names = FALSE, col.names = FALSE
                            )
write.table(Availabilitytbfhosts, file =
                            "Availabilitytbfhosts.txt", quote =
                            FALSE, sep = " ", na = "NA",
                            row.names = FALSE, col.names = FALSE
                            )
```

Código 3.19 Simulación de MTTR, MTBF y disponibilidad para varios recursos de computación

Tal y como se mencionaba en los métodos desarrollados en la metodología de la tesis para la estimar la fiabilidad y la disponibilidad, los valores de tiempos medios entre fallos (MTBF) de cada recurso están en segundos y los de tiempos medios de reparación (MTTR) en minutos, por lo que estos últimos se multiplican por 60 segundos.

Los datos de fiabilidad de todos los recursos están en el mismo fichero (código 3.20) y están escalados para evitar que sean valores muy pequeños, siguiendo la escala de la función de Weibull que se utilizó para generarlos, tal y como se justifica en la actividad de medición del método correspondiente.

```
0.993843839180858
0.612462697104215
0.930098752774256
0.835827039839582
1.23328126733084
0.93365184837281
0.864155507572469
0.502358336418683
0.626690827354037
0.882861376885515
...
```

Código 3.20 Ejemplo de fichero con datos de fiabilidad para los recursos de computación de una plataforma para SimGrid

Los datos de disponibilidad de todos los recursos están en el mismo fichero (código 3.21) y en un rango de valores de 0 a 1.

```
0.973096377938322
0.983248351879957
0.974365116211165
0.977794383522065
0.968024783254962
0.974847873920881
0.976774180177113
0.986181331331053
0.984690190438087
0.976153401919739
...
```

Código 3.21 Ejemplo de fichero con datos de disponibilidad para los recursos de computación de una plataforma para SimGrid

3.6.2.2 Representación de datos relativos a fallos y propiedades no funcionales sobre hosts o recursos de computación

Una vez obtenidos los ficheros de traza con instantes de fallo y recuperación, así como los datos sobre propiedades no funcionales fiabilidad y disponibilidad, para todos los recursos de computación de una plataforma o nodo concreto, se actualizan los ficheros de descripción de la misma añadiendo (código 3.22) a cada recurso la siguiente información:

- Un fichero de traza con los instantes de tiempo en los que falla cada recurso, aprovechando el atributo `state_file`.
- Los valores concretos de sus propiedades no funcionales, aprovechando la etiqueta `<prop>` existente en el vocabulario de descripción de plataformas de *SimGrid*.

```
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/
                          simgrid.dtd">
<platform version="3">
  <AS id="AS0" routing="Full">
    <host id="Host1" power="3.300140519709234E9"
                          state_file="host1.trace">
      <prop id="availability" value="0.973096377938322"/>
      <prop id="reliability" value="0.993843839180858"/>
   </host>
```

Código 3.22 Ejemplo de plataforma de SimGrid con propiedades no funcionales y fichero de fallos de cada recurso

De cara a automatizar esta tarea para todas las plataformas de prueba que se utilizan en los experimentos, se ha implementado un programa en Python (código 3.23) que genera un fichero de salida con la nueva versión de la plataforma a partir del fichero original descriptivo de la plataforma, los ficheros de texto con datos de fiabilidad y disponibilidad de sus recursos y un fichero con la lista de nombres de ficheros de traza de cada recurso con sus instantes de fallo y recuperación.

3.6.3 Aportaciones al modelo de pruebas de SimGrid

Las propiedades no funcionales pueden ser utilizadas como factores de discriminación o de corrección en los algoritmos de selección de recursos. Si las propiedades no funcionales o de calidad de servicio están calculadas solamente con datos históricos, se consideran *indicadores estáticos*. Si se corresponden

```
import sys
from lxml import etree
if len(sys.argv) != 5:
      print 'python %s platform.xml Availability.txt
                              Reliability.txt Traces.txt' %
                              sys.argv[0]
else:
      doc = etree.parse(sys.argv[1])
      availabilityFile = open(sys.argv[2])
      reliabilityFile = open(sys.argv[3])
      tracesFile = open(sys.argv[4])
      root = doc.getroot()
      for host in root.iter('host'):
            traceData = tracesFile.readline().rstrip('\r\n')
            availabilityData =
                              availabilityFile.readline().rstrip('
                              \r\n')
            reliabilityData =
                              reliabilityFile.readline().rstrip('\
                              r\n')
            host.set( "state_file", traceData )
            prop1 = etree.SubElement(host, 'prop')
            prop1.set( "id", "efficiency" )
            prop1.set( "value", "0.0" )
            prop2 = etree.SubElement(host, 'prop')
            prop2.set( "id", "availability" )
            prop2.set( "value", availabilityData )
            prop3 = etree.SubElement(host, 'prop')
            prop3.set( "id", "reliability" )
            prop3.set( "value", reliabilityData )
      outputfile = "NFP_%s" % sys.argv[1]
      doc.write(outputfile)
      availabilityFile.close()
      reliabilityFile.close()
```

Código 3.23 Código para automatizar la actualización de la plataforma de SimGrid con propiedades no funcionales y fichero de fallos de cada recurso

con una medición de la situación actual basada en la monitorización reciente del recurso, se considerarán *indicadores dinámicos*.

Dadas las premisas, restricciones y desarrollos de la presente tesis, se propone calcular un índice o *rating* de propiedades no funcionales de cada recurso como indicador de su idoneidad, basado en indicadores estáticos, dado que los valores de las propiedades no funcionales fiabilidad y disponibilidad se han obtenido simulando datos históricos del recurso.

Un ejemplo de indicador dinámico podría ser la propiedad no funcional correspondiente a la eficiencia de cada recurso, midiendo por ejemplo el

tiempo que tarda en terminar todos los trabajos que se le asignan de cara a calcular un ratio respecto al tiempo de ejecución ideal de dichos trabajos.

Dado que en la propia naturaleza del algoritmo min-min ya se realiza una evaluación dinámica sobre la situación y disponibilidad de los recursos (calcula el tiempo estimado de finalización o EFT de cada tarea en cada recurso), se ha abordado la implementación de variaciones del subalgoritmo de selección de recursos para combinar dicha información dinámica con un índice o *rating* estático basado en la utilización individual o combinada de las propiedades no funcionales fiabilidad y disponibilidad de cada recurso.

Con ello se pretende evaluar qué efecto tiene la utilización de propiedades no funcionales en la selección de recursos respecto a las prestaciones globales del sistema, así como si pueden resultar de ayuda en escenarios de fallos de los recursos, evitando tener que relanzar o migrar trabajos, con la consiguiente pérdida de trabajo y tiempo.

Las modificaciones realizadas sobre el código de partida del algoritmo de planificación min-min implementado en *SimDag* afectan a las siguientes funciones: programa principal (`main`), función de estimación de tiempo de finalización de una tarea en un recurso (`finish_on_at`) y función de selección del mejor recurso para una tarea (`SD_task_get_best_workstation`). Además, se ha implementado una nueva función para generar estadísticas en caso de fallo en la simulación de la ejecución del grafo de tareas sobre la plataforma con recursos que fallan.

A continuación, en el código 3.24 se muestra en pseudocódigo las modificaciones realizadas en el subalgoritmo de selección de recursos dado que es el componente principal sobre el que se reflejan las aportaciones de la presente tesis, si bien se ofrecen todos los detalles de implementación sobre todo el modelo de pruebas de SimGrid en los siguientes apartados.

3.6.3.1 Programa principal

Dado que el fichero que describe el grafo de tareas según el modelo de datos de *SimGrid* no permite incluir atributos o elementos a nivel de grafo, la expresión con los criterios para la selección de recursos que desea el usuario para la ejecución de su trabajo (grafo de tareas) se ha decidido especificarla como parámetro del programa principal.

Se ha definido un nuevo argumento de dos caracteres cuya combinación permite especificar una versión diferente del subalgoritmo de selección de recursos, tal como se muestra en el código 3.25 y el código 3.26.

```
- Declarar variables
- Declarar la estructura para contener los valores de cada una
                          de las propiedades no funcionales de
                          cada recurso
- Declarar la estructura para contener el índice o "rating" de
                          cada recurso
- Obtener la lista de recursos o estaciones de trabajo
- Para cada recurso o estación de trabajo
  * Obtener sus propiedades no funcionales
  * Cálcular el índice o "rating" de cada recurso a partir de la
                          combinación o no de sus propiedades
                          no funcionales (según el criterio 1-
                          5 especificado)
- Inicializar la referencia al mejor recurso con la primera
                          estación de trabajo de la lista
- Obtener el tiempo efectivo de finalización de la tarea en este
                          primer recurso o estación de trabajo
- Para cada recurso o estación de trabajo restante
  * Obtener el tiempo efectivo de finalización de la tarea en el
                          recurso o estación de trabajo en
                          cuestión
  * Si el tiempo es menor que el actual
  * Si el recurso o estación de trabajo no está apagado
    + Determinar la condición a utilizar para seleccionar el
                          mejor recurso (según el criterio A-D
                          especificado y el índice o "rating"
                          de cada recurso calculado antes)
    + Si se cumple la condición anterior
      ^ Actualiza la referencia del mejor recurso con este nuevo
                          recurso o estación de trabajo
- Liberar la memoria utilizada por las estructuras de datos
- Devolución de la refencia del mejor recurso o estación de
                          trabajo
```

Código 3.24 Pseudocódigo con las modificaciones realizadas al subalgoritmo de selección de recursos del modelo de pruebas de SimGrid

El primer carácter (O, A, B, C, D) se corresponde a un criterio de utilización del índice o *rating* de propiedades no funcionales, ya sea como factor de discriminación o como factor de corrección:

- O. Algoritmo min-min original (el recurso con mejor EFT): no utilizar el índice o *rating*.
- A. Algoritmo min-min original (el recurso con mejor EFT) con factor de discriminación en caso de igualdad: seleccionar el recurso con mejor *rating* de propiedades no funcionales.
- B. Algoritmo NFP (el recurso con mejor *rating* de propiedades no funcionales), con factor de discriminación en caso de igualdad: seleccionar el recurso con mejor EFT.

- C. Algoritmo min-min-NFP con factor de corrección basado en el mejor *rating* de propiedades no funcionales, con factor de discriminación en caso de igualdad: criterio A.
- D. Algoritmo min-min-NFP con factor de corrección basado en el mejor *rating* de propiedades no funcionales, con factor de discriminación en caso de igualdad: criterio B.

El segundo carácter se corresponde a una forma de calcular el índice o *rating* de propiedades no funcionales, de cara a evaluar el efecto de la utilización independiente o combinada de las mismas:

- 0: No calcular (para el algoritmo min-min original).
- 1: Usar la fiabilidad.
- 2: Usar la disponibilidad.
- 3: Combinar fiabilidad y disponibilidad mediante su suma.
- 4: Combinar fiabilidad y disponibilidad mediante su producto.
- 5: Combinar fiabilidad y disponibilidad mediante su media.

```
int main(int argc, char **argv)
{
 [...]
  workstation = SD_task_get_best_workstation(task,argv[3]);
[...]
  while ( !xbt_dynar_is_empty((changed = SD_simulate(-1.0))) &&
                            !stop ) {
    [...]
    xbt_dynar_foreach(ready_tasks, cursor, task) {
      [...]
      workstation = SD_task_get_best_workstation(task,argv[3]);
      [...]
    }
  [...]
```

Código 3.25 Modificación del programa principal del modelo de pruebas SimDag de SimGrid para incluir el subalgoritmo de selección de recursos

Cada uno de los criterios (A, B, C, D) se combina con cada uno de los diferentes índices de propiedades no funcionales o *ratings* (1, 2, 3, 4, 5), dando lugar a 20 versiones diferentes de subalgoritmos de selección de recursos.

Se ha decidido simplificar la combinación de las propiedades no funcionales mediante operaciones aritméticas básicas para simplificar igualmente las

condiciones del subalgoritmo de selección de recursos, siguiendo una lógica similar a la que utiliza el popular metaplanificador *GridWay* para la *grid* en su expresión RANK para la jerarquización de potenciales recursos seleccionables como parte de una de sus políticas de metaplanificación.

```
int main(int argc, char **argv)
{
  [...]
  double completion_time = 0.0;
  int stop = 0;
  FILE *stats = NULL;
  [...]
  if (argc < 4) {
    XBT_INFO("Usage: %s platform_file dax_file O0 [jedule_file]
                          [stats_file]", argv[0]);
    XBT_INFO
        ("example: %s simulacrum_7_hosts.xml Montage_25.xml O0
                          Montage_25.jed Montage_25.stats",
         argv[0]);
    exit(1);
  }
  [...]
  char *statsfilename;
  [...]
  if (argc == 4) {
    char *lastplatform = strrchr(argv[1], '.');
    char *lastdag = strrchr(argv[2], '.');
    tracefilename = bprintf("%.*s-%.*s-%s.jed",
                      (int) (lastdag == NULL ? strlen(argv[2]) :
                           lastdag - argv[2]),
                      argv[2],
                      (int) (lastplatform == NULL ?
                           strlen(argv[1]) : lastplatform -
                           argv[1]), argv[1],
                      argv[3]);
    statsfilename = bprintf("%.*s-%.*s-%s.stats",
                      (int) (lastdag == NULL ? strlen(argv[2]) :
                           lastdag - argv[2]),
                      argv[2],
                      (int) (lastplatform == NULL ?
                           strlen(argv[1]) : lastplatform -
                           argv[1]), argv[1],
                      argv[3]);
  } else {
    tracefilename = xbt_strdup(argv[4]);
    statsfilename = xbt_strdup(argv[5]);
  }
```

Código 3.26 Modificación del programa principal del modelo de pruebas SimDag de SimGrid para gestionar el nuevo argumento con el subalgoritmo de selección de recursos

También se han realizado modificaciones para poder generar un fichero de estadísticas en caso de fallo en la simulación de la ejecución del grafo de tareas sobre la plataforma con recursos que fallan, tal como se muestra en el código 3.27.

Se prepara el fichero, cuyo nombre puede seguir el patrón establecido para el otro fichero que genera el modelo de pruebas por defecto (ver código 3.26).

Además, se crean puntos de control o *watchpoints* con el objetivo de parar la simulación cuando falle una tarea para invocar a la función de estadísticas que se describirá más adelante.

```
[...]
  xbt_dynar_foreach(dax, cursor, task) {
    SD_task_watch(task, SD_FAILED);
    SD_task_watch(task, SD_DONE);
  }
[...]
  while ( !xbt_dynar_is_empty((changed = SD_simulate(-1.0))) &&
                              !stop ) {
    xbt_dynar_foreach(changed, changedcursor, changedtask) {
      if (SD_task_get_state(changedtask)==SD_FAILED) {
        stop = 1;
        continue;
      }
    }
    [...]
    }
  [...]
  completion_time = SD_get_clock(); // MODIFIED
  XBT_INFO("Simulation Time: %f", completion_time); // MODIFIED
  XBT_INFO("Producing the stats of the hosts into %s",
                             statsfilename);   // NEW
  stats = fopen(statsfilename, "w"); // NEW
  xbt_assert(stats, "Cannot write to %s", statsfilename); // NEW
  free(statsfilename); // NEW
  fprintf(stats, "%f\n", completion_time ); // NEW
  if ( !stop )
    output_xml(out, dax);
  fprintf(stats, "%f\n", completion_time );
  output_stats(stats, dax); // NEW
    fclose(stats); // NEW
  [...]
```

Código 3.27 Modificación del programa principal del modelo de pruebas SimDag de SimGrid (parte 3)

3.6.3.2 Cálculo del tiempo estimado de finalización o EFT

Respecto a esta función, las modificaciones han sido mínimas, ya que simplemente se accede al estado de cada recurso o *Workstation* (al atributo `state` del elemento `host` dentro del modelo de datos cargado a partir del fichero que representa la plataforma) con el objetivo de evitar realizar cálculos si la estación de trabajo está apagada.

```
static double finish_on_at(SD_task_t task, SD_workstation_t
                           workstation)
{
  [...]
  xbt_dict_t props;
  xbt_dict_cursor_t cursor2 = NULL;
  char *key, *data;
  char isTurnedOn = TRUE;

  props = SD_workstation_get_properties(workstation);

  xbt_dict_foreach(props, cursor2, key, data) {
    if ((strcmp(key, "state")==0) && (strcmp(data, "OFF")==0))
    {
      isTurnedOn = FALSE;
      XBT_INFO("%s is turned OFF",
                           SD_workstation_get_name(workstation)
                           );
    }
  }

  if ( isTurnedOn != FALSE )
  {
     [...]
  }
  else
  {
    return -1.0;
  }
}
```

Código 3.28 Modificaciones en la función de cálculo de tiempo estimado de finalización de una tarea en un recurso

3.6.3.3 Variantes del algoritmo de selección de recursos

Como ya se ha indicado anteriormente, a partir de un parámetro que recibe el programa principal, se especifica el criterio para la selección de recursos que desea el usuario para la ejecución de su trabajo (grafo de tareas): un nuevo argumento de dos caracteres cuya combinación permite especificar una versión diferente del subalgoritmo de selección de recursos.

Además del algoritmo min-min original (O0), cada uno de los criterios (A, B, C, D) se puede combinar con cada uno de los diferentes índices de propiedades no funcionales o *ratings* (1, 2, 3, 4, 5), dando lugar a las 20 versiones diferentes de subalgoritmos de selección de recursos que ya se han explicado anteriormente.

En lugar de crear diferentes versiones de la función que contiene el subalgoritmo de selección de recursos `SD_task_get_best_workstation`, se ha utilizado una variable booleana que recibe el resultado de evaluar la expresión condicional múltiple correspondiente al criterio especificado en el parámetro.

```
static SD_workstation_t SD_task_get_best_workstation(SD_task_t
                           task, char *algoritmo)
{
 [...]
  double efficiency, reliability, availability; // NFP
  double max_rating = -1.0;
  double* ratings;
  double sum_of_properties_values = 0.0;
  double mult_of_properties_values = 1.0;
  int count_properties = 0;
  int condition = 0;
  xbt_dict_t props;
  xbt_dict_cursor_t cursor2 = NULL;
  char *key, *data;
  ratings = malloc(SD_workstation_get_number()*sizeof(double));
 [...]
```

Código 3.29 Modificaciones en la función de selección del mejor recurso para una tarea (parte 1)

En primer lugar, para cada recurso o *workstation* se obtienen sus propiedades no funcionales. Acto seguido (código 3.30), se obtiene el índice de propiedades no funcionales o *rating* de cada recurso o *workstation* en base al modo especificado en el segundo carácter del parámetro (0, 1, 2, 3, 4, 5).

Por último (código 3.31), se utiliza el índice de propiedades no funcionales o *rating* de cada recurso o *workstation* como factor de discriminación o de corrección en base al criterio de selección especificado en el primer carácter del parámetro (O, A, B, C, D), evitando a los recursos o *workstations* apagados.

3.6.3.4 Generación de datos estadísticos sobre la simulación

Respecto a la información de salida que produce el modelo de pruebas, se ha implementado (código 3.32) una nueva función para generar estadísticas en caso de fallo en la simulación de la ejecución del grafo de tareas sobre la plataforma con recursos que fallan.

```
static SD_workstation_t SD_task_get_best_workstation(SD_task_t
                            task, char *algoritmo)
{
[...]
  for (i = 0; i < nworkstations; i++) {
    props = SD_workstation_get_properties(workstations[i]);
    xbt_dict_foreach(props, cursor2, key, data) {
//    if ( strcmp(key, "efficiency") == 0) {
//    efficiency = strtod(data, NULL);
//    count_properties = count_properties + 1;
//    sum_of_properties_values = sum_of_properties_values +
                            efficiency;
//    mult_of_properties_values = mult_of_properties_values *
                            efficiency;
//    }
      if ( strcmp(key, "reliability") == 0) {
        reliability = strtod(data, NULL);
        count_properties = count_properties + 1;
        sum_of_properties_values = sum_of_properties_values +
                            reliability;
        mult_of_properties_values = mult_of_properties_values *
                            reliability;
      }
      if ( strcmp(key, "availability") == 0) {
        availability = strtod(data, NULL);
        count_properties = count_properties + 1;
        sum_of_properties_values = sum_of_properties_values +
                            availability;
        mult_of_properties_values = mult_of_properties_values *
                            availability;
      }
    }
    if ( algoritmo[1] == '1' )
      ratings[i] = reliability;
    else if ( algoritmo[1] == '2' )
      ratings[i] = availability;
    else if ( algoritmo[1] == '3' )
      ratings[i] = sum_of_properties_values;
    else if ( algoritmo[1] == '4' )
      ratings[i] = mult_of_properties_values;
    else if ( algoritmo[1] == '5' )
      ratings[i] = sum_of_properties_values / count_properties;
    else
      ratings[i] = -1;
    XBT_INFO("%s ratings: %f",
                            SD_workstation_get_name(workstations
                            [i]), ratings[i]);
    sum_of_properties_values = 0.0;
    mult_of_properties_values = 1.0;
    count_properties = 0;
  }
[...]
```

Código 3.30 Modificaciones en la función de selección del mejor recurso para una tarea (parte 2)

```
static SD_workstation_t SD_task_get_best_workstation(SD_task_t
                            task, char *algoritmo)
{
[...]
  best_workstation = workstations[0];
  min_EFT = finish_on_at(task, workstations[0]);
  XBT_INFO("%s finishes on %s at %f", SD_task_get_name(task),
                            SD_workstation_get_name(workstations
                            [0]), min_EFT);
  max_rating = ratings[0];
  for (i = 1; i < nworkstations; i++) {
    EFT = finish_on_at(task, workstations[i]);
    XBT_INFO("%s finishes on %s at %f", SD_task_get_name(task),
                            SD_workstation_get_name(workstations
                            [i]), EFT);
    if (EFT != -1.0) {
      if ( algoritmo[0] == 'O' )
        condition = (EFT < min_EFT);
      else if ( algoritmo[0] == 'A' )
        condition = ( (EFT < min_EFT) || ( (EFT == min_EFT) &&
                            (ratings[i] > max_rating) ) );
      else if ( algoritmo[0] == 'B' )
        condition = ( (ratings[i] > max_rating) || ( (ratings[i]
                            == max_rating) && (EFT < min_EFT) )
                            );
      else if ( algoritmo[0] == 'C' )
        condition = ( ((EFT/ratings[i]) < (min_EFT/max_rating))
                            || ( ((EFT/ratings[i]) ==
                            (min_EFT/max_rating)) && ( (EFT <
                            min_EFT) || ((EFT == min_EFT) &&
                            (ratings[i] > max_rating)) ) ) );
      else if ( algoritmo[1] == 'D' )
        condition = ( ((EFT/ratings[i]) < (min_EFT/max_rating))
                            || ( ((EFT/ratings[i]) ==
                            (min_EFT/max_rating)) && (
                            (ratings[i] > max_rating) ||
                            ((ratings[i] == max_rating) && (EFT
                            < min_EFT)) ) ) );
      if ( (min_EFT == -1.0) || condition ) {
        min_EFT = EFT;
        max_rating = ratings[i];
        best_workstation = workstations[i];
      }
    }
  }
  free(ratings);
  return best_workstation;
}
```

Código 3.31 Modificaciones en la función de selección del mejor recurso para una tarea (parte 3)

Por un lado (código 3.32 y ss.), se preparan estructuras de datos para procesar todas las tareas de cualquier tipo (computación, comunicación y otras), tanto si se han completado como si han fallado, y obtener los siguientes datos:

```
static void output_stats(FILE * stats, xbt_dynar_t dax)
{
  unsigned int cursor, j, k;
  int current_nworkstations, failed_comp_tasks = 0,
                             failed_comm_tasks = 0,
                             done_comp_tasks = 0, done_comm_tasks
                             = 0, remaining_tasks = 0;
  const int nworkstations = SD_workstation_get_number();
  const SD_workstation_t *workstations =
                             SD_workstation_get_list();
  SD_task_t task;
  SD_workstation_t *wsl;
  unsigned int ws_idx;
  int task_count = 0, kindtask;
  double task_efficiency = 0.0, execution_time = 0.0, runtime =
                             0.0;
  double task_efficiency_sum = 0.0, execution_time_sum = 0.0,
                             runtime_sum = 0.0, remaining_flops =
                             0.0, remaining_bytes = 0.0;
  double workstation_total_runtime [nworkstations];
  double workstation_total_executiontime [nworkstations];
  double workstation_total_efficiency [nworkstations];
  int workstation_total_task_count [nworkstations];
  for(j = 0; j < nworkstations; j++)
  {
    workstation_total_runtime[j] = 0.0;
    workstation_total_executiontime[j] = 0.0;
    workstation_total_efficiency[j] = 0.0;
    workstation_total_task_count[j] = 0;
  }
  xbt_dynar_foreach(dax, cursor, task)
  {
     task_count++;
  }
[...]
```

Código 3.32 Función de generación de estadísticas en caso de fallo (parte 1)

- Número total de tareas de computación completadas (*Total COMP tasks done*)
- Número total de tareas de comunicación completadas (*Total COMM tasks done*)
- Número de tareas de computación que han provocado fallo (*Failed COMP tasks*)
- Número de tareas de comunicación que han provocado fallo (*Failed COMM tasks*)
- Número total de tareas sin completar (*Total remaining tasks*)
- Cantidad de trabajo sin completar (*Total remaining flops*)

```
static void output_stats(FILE * stats, xbt_dynar_t dax)
{
[...]
  xbt_dynar_foreach(dax, cursor, task) {
  //xbt_dynar_foreach Iterates over the whole dynar
    kindtask = SD_task_get_kind(task);
    current_nworkstations = SD_task_get_workstation_count(task);
    wsl = SD_task_get_workstation_list(task);
    switch (kindtask) {
      case SD_TASK_COMP_SEQ:
        if (SD_task_get_state(task)==SD_FAILED) {
          failed_comp_tasks = failed_comp_tasks + 1;
          remaining_flops = remaining_flops +
                            SD_task_get_remaining_amount(task)/4
                            .2e9;
          SD_task_dump(task);
          XBT_INFO("Task '%s' has failed on %d workstations.
                            %.6f flops remain to be done. [%f-
                            >%f] %s computes %f in %f time",
               SD_task_get_name(task), current_nworkstations,
                            SD_task_get_remaining_amount(task)/4
                            .2e9,
               SD_task_get_start_time(task),
                            SD_task_get_finish_time(task),
               SD_workstation_get_name(wsl[0]), runtime,
                            execution_time);
          fprintf(stats, "Task '%s' has failed on %d
                            workstations. %.6f flops remain to
                            be done. [%f->%f] %s computes %f in
                            %f time\n",
               SD_task_get_name(task), current_nworkstations,
                            SD_task_get_remaining_amount(task)/4
                            .2e9,
               SD_task_get_start_time(task),
                            SD_task_get_finish_time(task),
               SD_workstation_get_name(wsl[0]), runtime,
                            execution_time );
        }
[...]
```

Código 3.33 Función de generación de estadísticas en caso de fallo (parte 2)

- Cantidad de datos sin transferir (*Total remaining bytes*)
- Cantidad de trabajo asignado (*Total runtime*)
- Cantidad de trabajo ejecutado (*Total execution time*)
- Ratio de eficiencia de la computación (*Total COMP tasks efficiency*)

```
static void output_stats(FILE * stats, xbt_dynar_t dax)
{
[...]
        else
          if ( (SD_task_get_state(task) == SD_DONE) &&
               (SD_task_get_name(task) != "root") ) {
            done_comp_tasks = done_comp_tasks + 1;
            runtime = SD_task_get_amount(task) / 4.2e9;
            execution_time = SD_task_get_finish_time(task) -
                             SD_task_get_start_time(task);
            task_efficiency = runtime / execution_time;
            task_efficiency_sum += task_efficiency;
            execution_time_sum += execution_time;
            runtime_sum += runtime;
            for (j = 0; j < current_nworkstations; j++) {
              for (k = 0; k < nworkstations; k++) {
                if ( !strcmp(
                             SD_workstation_get_name(workstations
                             [k]),
                             SD_workstation_get_name(wsl[j])) ) {
                  workstation_total_runtime[k] += runtime;
                  workstation_total_executiontime[k] +=
                             execution_time;
                  workstation_total_efficiency[k] +=
                             task_efficiency;
                  workstation_total_task_count[k] =
                             workstation_total_task_count[k] + 1;
                }
              }
            }
            fprintf(stats, "[%f->%f] %s computes %f in %f time #
                             %s\n", SD_task_get_start_time(task),
                             SD_task_get_finish_time(task),
                    SD_workstation_get_name(wsl[0]), runtime,
                             execution_time,
                             SD_task_get_name(task) );
          }
        break;
[...]
```

Código 3.34 Función de generación de estadísticas en caso de fallo (parte 3)

Por otro lado, se preparan estructuras de datos para procesar todos los recursos o *workstations* y obtener los siguientes datos de cada uno de ellos:

- Tiempo total de trabajo asignado (suma de tiempos ideales de ejecución de cada tarea de computación)
- Tiempo total de trabajo ejecutado (suma de tiempos reales de ejecución de cada tarea de computación)
- Ratio de eficiencia (ratio del tiempo total de trabajo asignado sobre el tiempo total ejecutado)

```
static void output_stats(FILE * stats, xbt_dynar_t dax)
{
[...]
      case SD_TASK_COMM_E2E:
        if ( SD_task_get_state(task) == SD_FAILED ) {
          failed_comm_tasks = failed_comm_tasks + 1;
          remaining_bytes = remaining_bytes +
                            SD_task_get_remaining_amount(task);
          SD_task_dump(task);
          XBT_INFO("Task '%s' has failed on %d workstations.
                            %.6f bytes remain to be transfer.
                            [%f -> %f] %s -> %s transfers %.6f
                            bytes",
                SD_task_get_name(task), current_nworkstations,
                            SD_task_get_remaining_amount(task),
                SD_task_get_start_time(task),
                            SD_task_get_finish_time(task),
                SD_workstation_get_name(wsl[0]),
                            SD_workstation_get_name(wsl[1]),
                            SD_task_get_amount(task) );
          fprintf(stats, "Task '%s' has failed on %d
                            workstations. %.6f bytes remain to
                            be transfer. [%f -> %f] %s -> %s
                            transfers %.6f bytes",
                SD_task_get_name(task), current_nworkstations,
                            SD_task_get_remaining_amount(task),
                SD_task_get_start_time(task),
                            SD_task_get_finish_time(task),
                SD_workstation_get_name(wsl[0]),
                            SD_workstation_get_name(wsl[1]),
                            SD_task_get_amount(task) );
        }
        else {
          done_comm_tasks = done_comm_tasks + 1;
          fprintf(stats, "[%f -> %f] %s -> %s transfers %.6f
                            bytes # %s\n",
                            SD_task_get_start_time(task),
                            SD_task_get_finish_time(task),
                SD_workstation_get_name(wsl[0]),
                            SD_workstation_get_name(wsl[1]),
                            SD_task_get_amount(task),
                            SD_task_get_name(task));
        }
        break;
      default:
        fprintf(stats, "Task %s is of unknown kind %d",
                            SD_task_get_name(task),
                            SD_task_get_kind(task));
        remaining_tasks = remaining_tasks + 1;
    }
    runtime = 0.0;
    execution_time = 0.0;
    task_efficiency = 0.0;
  }
[...]
```

Código 3.35 Función de generación de estadísticas en caso de fallo (parte 4)

```
static void output_stats(FILE * stats, xbt_dynar_t dax)
{
[...]
  for (j = 0; j < nworkstations; j++) {
    if ( workstation_total_runtime[j] != 0 )
      fprintf(stats, "Host %s computes %.6f in %.6f with %.6f
                          (%.6f) efficiency.\n",
          SD_workstation_get_name(workstations[j]),
          workstation_total_runtime[j],
          workstation_total_executiontime[j],
          workstation_total_efficiency[j] /
                          workstation_total_task_count[j],
          workstation_total_runtime[j] /
                          workstation_total_executiontime[j]
                          );
  }

  fprintf(stats, "Total COMP tasks done: %d\n",
                          done_comp_tasks);
  fprintf(stats, "Total COMM tasks done: %d\n",
                          done_comm_tasks);
  fprintf(stats, "Failed COMP tasks: %d\n", failed_comp_tasks);
  fprintf(stats, "Failed COMM tasks: %d\n", failed_comm_tasks);
  fprintf(stats, "Total remaining tasks: %d\n",
                          remaining_tasks);
  fprintf(stats, "Total remaining flops: %.6f\n",
                          remaining_flops);
  fprintf(stats, "Total remaining bytes: %.6f\n",
                          remaining_bytes);
  fprintf(stats, "Total runtime (without failure): %.6f\n",
                          runtime_sum);
  fprintf(stats, "Total execution time: %.6f\n",
                          execution_time_sum);
  fprintf(stats, "Total COMP tasks efficiency: %.6f (%.6f)\n",
                          task_efficiency_sum/task_count,
                          runtime_sum/execution_time_sum);
}
```

Código 3.36 Función de generación de estadísticas en caso de fallo (parte 5)

3.6.3.5 Herramienta de automatización de experimentos

Dado que en la evaluación de la tesis se proponen diferentes experimentos para evaluar todos y cada uno de los subalgoritmos de selección de recursos en varios escenarios, se han desarrollado una serie de sencillos programas o *scripts* para el intérprete de comandos `bash` de cualquier sistema operativo de tipo UNIX, como es el caso de los sistemas GNU/Linux, que faciliten las tareas asociadas a dicha evaluación.

Todos estos programas reciben como parámetros tres archivos de texto:

- Una lista de nombres de fichero correspondientes a las diferentes plataformas con las que se desea trabajar (código 3.37).

- Una lista de nombres de ficheros correspondientes a los distintos grafos de tareas con los que se desea trabajar (código 3.38).
- Una lista de códigos correspondientes a los subalgoritmos de selección de recursos con los que se desea trabajar (código 3.39).

```
cluster_1-7.xml
NFP_cluster_1-7_v1.xml
NFP_cluster_1-7_v2.xml
NFP_cluster_1-7_v3.xml
NFP_cluster_1-7_v4.xml
NFP_cluster_1-7_v5.xml
node_1-49.xml
NFP_node_1-49_v1.xml
NFP_node_1-49_v2.xml
NFP_node_1-49_v3.xml
NFP_node_1-49_v4.xml
NFP_node_1-49_v5.xml
grid_1-1035.xml
NFP_grid_1-1035_v1.xml
NFP_grid_1-1035_v2.xml
NFP_grid_1-1035_v3.xml
NFP_grid_1-1035_v4.xml
NFP_grid_1-1035_v5.xml
```

Código 3.37 Ejemplo de archivo con nombres de ficheros sobre plataformas para la automatización de experimentos

```
Montage_25.xml
DAG-corto.xml
DAG-medio.xml
DAG-largo.xml
ndgf_atlas_dax_500.xml
```

Código 3.38 Ejemplo de archivo con nombres de ficheros sobre grafos de tareas para la automatización de experimentos

El *script* del código 3.40 permite automatizar la ejecución del programa del modelo de pruebas con todas las combinaciones posibles de plataforma, grafo de tareas y subalgoritmo de selección de recursos a partir de las listas con sus respectivos nombres de ficheros (archivos que se reciben como parámetro).

```
00
A1
A2
A3
A4
A5
B1
B2
B3
B4
B5
C1
C2
C3
C4
C5
D1
D2
D3
D4
D5
```

Código 3.39 Ejemplo de archivo con códigos sobre subalgoritmos de selección de recursos para la automatización de experimentos

```
for platform in $(cat $1)
do
    for dag in $(cat $2)
    do
      for algorithm in $(cat $3)
      do
        ./minmin_test $platform $dag $algorithm
      done
    done
done
```

Código 3.40 Código para la automatización de ejecuciones del programa del modelo de pruebas

Por su parte, el *script* del código 3.41 permite procesar los ficheros de estadísticas (generados tras cada ejecución del programa del modelo de pruebas) correspondientes a las combinaciones posibles de plataforma, grafo de tareas y subalgoritmo de selección de recursos a partir de las listas con sus respectivos nombres de ficheros (archivos que se reciben como parámetro).

El código anterior obtiene los tiempos de finalización de la simulación en cada caso y genera un fichero de resultados en forma de valores separados por coma o CSV que contiene (código 3.42) una línea por cada caso con la siguiente información: tiempo, plataforma, grafo, algoritmo.

```
for platform in $(cat $1)
do
  for dag in $(cat $2)
  do
    for algorithm in $(cat $3)
    do
      if [ -e TESIS/results/${platform%.*}/${dag%.*} ]
      then
        tiempo=$(head -1
                          TESIS/results/${platform%.*}/${dag%.
                          *}/${dag%.*}-${platform%.*}-
                          ${algorithm%.*}.stats)
        echo $tiempo,${platform%.*},${dag%.*},${algorithm%.*} >>
                          results.csv
      fi
    done
  done
done
```

Código 3.41 Código para procesar los ficheros de estadísticas y generar ficheros de resultados sobre las ejecuciones del programa del modelo de pruebas

```
98.184618,cluster_1-7,Montage_25,O0
98.184618,cluster_1-7,Montage_25,A1
98.184618,cluster_1-7,Montage_25,A2
98.184618,cluster_1-7,Montage_25,A3
98.184618,cluster_1-7,Montage_25,A4
98.184618,cluster_1-7,Montage_25,A5
[...]
```

Código 3.42 Ejemplo de fichero de resultados sobre las ejecuciones del programa del modelo de pruebas

Finalmente, el *script* del código 3.43 permite organizar en carpetas y subcarpetas los ficheros generados (fichero `jed` y fichero de estadísticas) tras cada ejecución del programa del modelo de pruebas correspondientes a las combinaciones posibles de plataforma, grafo de tareas y subalgoritmo de selección de recursos a partir de las listas con sus respectivos nombres de ficheros (archivos que se reciben como parámetro).

3.7 Conclusiones

En este capítulo se han presentado los desarrollos principales que sustentan la presente tesis: un modelo y una metodología de representación de propiedades no funcionales que respetan los estándares de la *grid* y se apoyan en la arquitectura S-OGSA para la semantización formal de la misma, así como un

```
for platform in $(cat $1)
do
  for dag in $(cat $2)
  do
    for algorithm in $(cat $3)
    do
      if [ ! -e TESIS/results/${platform%.*} ]
      then
        mkdir TESIS/results/${platform%.*}
      fi
      if [ ! -e TESIS/results/${platform%.*}/${dag%.*} ]
      then
        mkdir TESIS/results/${platform%.*}/${dag%.*}
      fi
      cp ${dag%.*}-${platform%.*}-${algorithm%.*}.*
                            TESIS/results/${platform%.*}/${dag%.
                            *}
    done
  done
done
```

Código 3.43 Código para organizar los ficheros de salida sobre las ejecuciones del programa del modelo de pruebas

framework tecnológico de pruebas en el entorno de simulación *SimDag* de *SimGrid* para realizar una prueba de concepto que permita evaluar los efectos de utilizar propiedades no funcionales en la selección de recursos.

Tanto el modelo como la metodología pretenden ser una propuesta poco intrusiva, dado que sugiere desplegar servicios *grid* semánticos que cumplan las restricciones de cualquier servicio *grid* de manera que puedan ser consumidos fácilmente por el resto de servicios y aplicaciones de la *grid*.

En cuanto al modelo de información, se ha desarrollado una ontología para representar los diferentes niveles o categorías de recursos y servicios de la *grid*, algunas propiedades no funcionales o de calidad de servicio, así como las asociaciones que se pueden plantear entre dichos elementos.

Respecto a la metodología, se ha definido y formalizado un proceso de desarrollo común para formalizar cualquier método de representación de la propiedad no funcional que se quiera abordar, indicando una serie de actividades y tareas a realizar.

Además, se han presentado sendos métodos de representación de propiedades no funcionales o indicadores de calidad de servicio sobre un recurso de computación: la fiabilidad y la disponibilidad.

Como se ha podido comprobar, dichos métodos son sencillos de implementar en un lenguaje de programación, dado que sólo tenemos que obtener información sobre el modelo matemático que representa el comportamiento errático y de recuperación de cada recurso durante un determinado período de tiempo significativo.

Al no disponer de datos reales, los métodos se basan en los modelos matemáticos encontrados en la literatura relacionada con la medición de la fiabilidad y la disponibilidad a nivel de recurso de computación, así como en otros modelos matemáticos hallados para la generación de valores realistas simulados sobre tiempos entre fallos y tiempos de recuperación sobre dicho tipo de recursos.

En caso de disponer de datos reales sobre tiempos entre fallos y tiempos de reparación del recurso en cuestión, se propone procesar dichos datos con técnicas de aprendizaje automático para obtener la ecuación lineal que represente más apropiadamente el comportamiento del recurso en cuestión, ya que dicho modelo siempre será mucho más preciso que los modelos teóricos generales descritos en la literatura.

Además, se ha iniciado el proceso de desarrollo del método para representar la fiabilidad de un nodo *grid*, punto de partida para abordar posteriormente incluso el asociado para representar la fiabilidad de todo un sistema *grid*, aunque no se han completado todas las actividades por exceder el alcance de la presente tesis.

También se han implementado los métodos de representación de la fiabilidad y la disponibilidad a nivel de recurso de computación para poder utilizar sus resultados en los experimentos asociados a la evaluación de la tesis sobre el entorno de simulación *SimDag* de *SimGrid*.

Por último, se ha presentado el conjunto de desarrollos realizados para crear un framework tecnológico de pruebas para el entorno de simulación *SimDag* de *SimGrid*, formado por una serie de modificaciones e implementaciones de nuevas funcionalidades sobre su modelo de pruebas, así como un conjunto de programas para automatizar las tareas asociadas a los experimentos asociados a la evaluación de la tesis.

Ítaca te dio el bello viaje.
Sin ella no habrías emprendido el camino.
Pero no tiene más que darte.

Constantino Cavafis

Capítulo 4

Evaluación

El objetivo de este apartado es la validación de las hipótesis de la presente tesis mediante la configuración y ejecución de una serie de experimentos que permita obtener unos resultados que puedan ser discutidos en relación a las hipótesis planteadas en este trabajo de investigación.

En primer lugar, se aborda la configuración de los experimentos llevados a cabo para, en segundo lugar, describir los resultados obtenidos tras la ejecución de los mismos. Por último, se discuten los resultados para la validación de las hipótesis de la presente tesis.

Cabe recordar en este punto que, dadas las premisas y restricciones planteadas en la presente tesis, el objetivo que se persigue con esta evaluación es realizar una prueba de concepto formal para comprobar qué efecto tiene la utilización de propiedades no funcionales en el subalgoritmo de selección de recursos de un algoritmo de planificación cualquiera a nivel de nodo *grid*, pudiendo considerar este último como un gestor de recursos local o LRM (*Local Resource Manager*), por lo que no se persigue abordar el problema de la planificación de trabajos en la *grid*.

En los diferentes escenarios que se detallarán en la configuración de los experimentos, según las necesidades, se combinan los siguientes elementos:

- Plataformas (18): un clúster con 7 recursos (`cluster_1-7.xml`) con idénticas propiedades funcionales, un nodo con 49 recursos (`node_1-49.xml`) con idénticas propiedades funcionales y una *grid* con 1035 recursos (`grid_1-1035.xml`) con diferentes propiedades

funcionales en general al estar repartidos en 9 clústeres diferentes conectados entre sí, que representa la topología de la infraestructura francesa *Grid'5000* para la experimentación en áreas de la ciencia de la computación en general y la computación distribuida y paralela en particular.

  - Además de las versiones originales de estas plataformas de pequeña, mediana y gran escala con recursos que idealmente no fallan, se han creado 5 versiones diferentes de cada una de ellas con diferentes ficheros de traza para cada uno de sus recursos de manera que tengan sus propios instantes de fallo y recuperación, ofreciendo diferentes escenarios de condiciones habituales:
    - `NFP_cluster_1-7_v1.xml`, `NFP_cluster_1-7_v2.xml`, `NFP_cluster_1-7_v3.xml`, `NFP_cluster_1-7_v4.xml`, `NFP_cluster_1-7_v5.xml`, `NFP_node_1-49_v1.xml`, `NFP_node_1-49_v2.xml`, `NFP_node_1-49_v3.xml`, `NFP_node_1-49_v4.xml`, `NFP_node_1-49_v5.xml`, `NFP_grid_1-1035_v1.xml`, `NFP_grid_1-1035_v2.xml`, `NFP_grid_1-1035_v3.xml`, `NFP_grid_1-1035_v4.xml`, `NFP_grid_1-1035_v5.xml`.

- Grafos (5): el grafo del modelo de pruebas de *SimDag* de la herramienta de simulación *SimGrid* (`Montage_25.xml`) con 25 trabajos dependientes entre sí que generan 25 tareas de computación de diferente duración y que dan también lugar a diferentes tareas de comunicación para la transferencia de datos entre dos recursos, que se corresponde con la aplicación *Montage* creada por el centro IPAC de la NASA para crear mosaicos personalizados del cielo a partir de múltiples imágenes de entrada; y un grafo (`ndgf_atlas_dax_500.xml`) con 500 trabajos independientes entre sí que representa una carga de trabajo habitual en un experimento *grid* de tamaño medio como es el caso de la fracción

del experimento ATLAS que se lleva a cabo en el LHC del CERN que se procesa en los recursos de la infraestructura Nordic DataGrid.

    - Además de la versión original del grafo de *SimDag*, se han generado otras 3 versiones del mismo (`DAG-corto.xml`, `DAG-medio.xml` y `DAG-largo.xml`) con diferente duración en cada una de las tareas de computación.

- Algoritmos (21): algoritmo min-min original, algoritmo min-min con factor de discriminación (cinco factores diferentes) y algoritmo min-min con factor de corrección (cinco factores diferentes). Todo ello para evaluar enfoques diferentes (O0, A1, A2, A3, A4, A5, B1, B2, B3, B4, B5, C1, C2, C3, C4, C5, D1, D2, D3, D4, D5) en la selección de los recursos.

Algunos de estos recursos han sido generados con las herramientas disponibles en la sección de contribuciones de *SimGrid* y representan escenarios realistas, tal y como se puede deducir leyendo los contenidos de dicha sección y las publicaciones científicas relacionadas con *SimGrid*.

Se plantean distintas versiones del comportamiento errático de las plataformas para evitar que los resultados estén condicionados excesivamente por las características funcionales y configuración de dichas plataformas, disponiendo así de más casuística.

Por lo tanto, se considera que esta configuración es suficientemente representativa para que los resultados obtenidos de la evaluación se puedan extrapolar a dominios más generales y validar las hipótesis de partida.

Además de esto, con la intención de realizar una equiparación objetiva, rigurosa y fiable, se llevarán a cabo varias pruebas estadísticas sobre los resultados de los experimentos, como el test de Shapiro–Wilk para contrastar la normalidad del conjunto de datos y determinar el estadístico a utilizar para comparar los resultados relacionados con los tiempos de finalización asociados a cada algoritmo, el test de Kruskal-Wallis como método no paramétrico para probar si el conjunto de datos proviene de la misma población en caso de no poder contrastar la normalidad del conjunto de datos y corroborar la influencia de los algoritmos utilizados sobre los resultados obtenidos, así como el test de de los rangos con signo de Wilcoxon para comprobar en este último caso si los resultados de un conjunto de algoritmos seleccionados son significativamente mejores que los del resto de algoritmos en cuestión.

4.1 Configuración de los experimentos

Los experimentos se enfocan a comprobar la eficacia y la efectividad respecto a la utilización de propiedades no funcionales en la selección de recursos de computación.

Se plantea la configuración de plataformas y grafos de tareas de diferente tamaño para comprobar cómo influye el número de recursos y el volumen de trabajo en los resultados de los experimentos.

Dadas las características del modelo de pruebas *SimDag* de la herramienta *SimGrid*, no se simulan diferentes cargas de trabajo de los recursos, aunque de alguna manera se realiza muy simplistamente mediante la configuración de grafos de tareas con diferente duración (muy corta, corta, media y larga).

En cualquier caso, dicha componente de carga de trabajo está presente en los experimentos dado que el propio algoritmo min-min evalúa dinámicamente el tiempo estimado de finalización para una determinada tarea en todos los recursos disponibles, para lo que tiene en cuenta la cantidad y duración de las tareas que ya tiene asignadas cada uno de ellos.

Por último, se ha de mencionar que toda la experimentación ha sido ejecutada sobre un ordenador Dell XPS L421X con un procesador dual-core Ivi Bridge de la marca Intel(R), modelo i7-3517U CPU @ 1.90 GHz 2.40 GHz, y una memoria RAM (DDR3 a 1600 MHz) instalada de 8 GB. Se ha utilizado el sistema operativo GNU/Linux Ubuntu 12.04 de 64 bits y todos los algoritmos de selección de recursos de la tesis han sido implementados utilizando el lenguaje de programación C en el que está implementado también el algoritmo min-min de planificación de trabajos existente en el modelo de pruebas SimDag dentro del entorno de simulación SimGrid.

4.1.1 Configuración del experimento E1 sobre la eficacia

La *eficacia* hace referencia al impacto de la utilización de propiedades no funcionales en la selección de recursos llevada a cabo en las mejores condiciones posibles o experimentales, es decir, sin presencia de fallo.

El *experimento E1* consiste en comparar el tiempo de finalización de la simulación, tanto si se utilizan o no propiedades no funcionales en la selección de recursos.

En este experimento se han utilizado las plataformas (pequeña, mediana y gran escala) sin fallos en los recursos, todos los grafos de tareas y todos los subalgoritmos de selección de recursos:

- Plataformas (3): `cluster_1-7.xml`, `node_1-49.xml` y `grid_1-1035.xml`.
- Grafos (5): `Montage_25.xml`, `DAG-corto.xml`, `DAG-medio.xml`, `DAG-largo.xml` y `ndgf_atlas_dax_500.xml`.
- Algoritmos (21): O0, A1, A2, A3, A4, A5, B1, B2, B3, B4, B5, C1, C2, C3, C4, C5, D1, D2, D3, D4, D5.

Dado que este primer experimento se pretende evaluar el rendimiento de las modificaciones introducidas en el subalgoritmo de selección de recursos en condiciones ideales respecto a su versión original, solamente se utilizarán las plataformas sin fallos.

Como ya se ha justificado anteriormente, estos recursos han sido generados con las herramientas disponibles en la sección de contribuciones de *SimGrid* y representan escenarios realistas, por lo que se consideran suficientemente representativos para que los resultados obtenidos de la evaluación se puedan extrapolar a dominios más generales.

4.1.2 Configuración del experimento E2 sobre la efectividad

La *efectividad* hace referencia al impacto de la utilización de propiedades no funcionales en la selección de recursos llevada a cabo en condiciones habituales de presencia de fallos, utilizando como factores de medida el tiempo de finalización de los trabajos y el número de fallos que se producen.

En este *experimento E2* se han utilizado cinco versiones de cada una de las plataformas (para ofrecer diferentes escenarios de condiciones habituales y sobre infraestructuras de pequeña, mediana y gran escala) con diferentes ficheros de traza con fallos y recuperaciones para cada recurso, todos los grafos de tareas menos el grafo más largo (`ndgf_atlas_dax_500.xml`) y todos los subalgoritmos de selección de recursos:

- Plataformas (15): `NFP_cluster_1-7_v1.xml`, `NFP_cluster_1-7_v2.xml`, `NFP_cluster_1-7_v3.xml`, `NFP_cluster_1-7_v4.xml`, `NFP_cluster_1-7_v5.xml`, `NFP_node_1-49_v1.xml`, `NFP_node_1-49_v2.xml`, `NFP_node_1-49_v3.xml`, `NFP_node_1-49_v4.xml`, `NFP_node_1-49_v5.xml`, `NFP_grid_1-1035_v1.xml`, `NFP_grid_1-1035_v2.xml`, `NFP_grid_1-1035_v3.xml`, `NFP_grid_1-1035_v4.xml`, `NFP_grid_1-1035_v5.xml`.

- Grafos (5): `Montage_25.xml`, `DAG-corto.xml`, `DAG-medio.xml` y `DAG-largo.xml`.
- Algoritmos (21): O0, A1, A2, A3, A4, A5, B1, B2, B3, B4, B5, C1, C2, C3, C4, C5, D1, D2, D3, D4, D5.

Se utilizan distintas versiones de las plataformas con diferente patrón de comportamiento errático para evitar que los resultados estén condicionados excesivamente por las características funcionales y configuración de dichas plataformas, disponiendo así de más casuística.

Como ya se ha justificado anteriormente, estos recursos han sido generados con las herramientas disponibles en la sección de contribuciones de *SimGrid* y los patrones de fallo y recuperación han sido generados mediante modelos matemáticos presentes en la literatura sobre el área, por lo que se considera que representan escenarios realistas.

Por lo tanto, estas configuraciones se consideran suficientemente representativas para que los resultados obtenidos de la evaluación se puedan extrapolar a dominios más generales.

4.2 Ejecución de los experimentos y discusión de los resultados

En este apartado se muestran los resultados obtenidos en cada experimento indicando, según el caso, tiempo de finalización, número de fallos, así como estadísticas de trabajo realizado y pendiente de realizar en caso de fallo. Además, se calculan medias, medianas y desviaciones típicas para el tiempo de finalización en cada escenario del experimento E2.

Los datos sobre tiempos de finalización están expresados en la misma unidad que el parámetro `runtime` de cada tarea de un grafo, es decir, en instantes de tiempo o ciclos de computación, sin hacer referencia a ninguna unidad de tiempo concreta, ya que esta dependerá de la velocidad del procesador en cuestión.

4.2.1 Ejecución del experimento E1 sobre la eficacia

A continuación (tabla 4.1 y ss.) se muestran los resultados obtenidos tras lanzar a ejecución todos los grafos de tareas, de diferente duración y características, sobre cada una de las plataformas sin posibilidad de fallo en sus recursos, y utilizando todos los algoritmos de selección de recursos disponibles.

Montage_25			
Algoritmo	**Cluster_1-7**	**Node_1-49**	**Grid_1-1035**
O0	98,184618	52,552652	3,270385
A1	98,184618	52,552652	3,270385
A2	98,184618	52,552652	3,270385
A3	98,184618	52,552652	3,270385
A4	98,184618	52,552652	3,270385
A5	98,184618	52,552652	3,270385
B1	98,184618	52,552652	3,270385
B2	98,184618	52,552652	3,270385
B3	98,184618	52,552652	3,270385
B4	98,184618	52,552652	3,270385
B5	289,975042	579,055753	118,289282
C1	98,184618	52,552652	3,270385
C2	98,184618	52,552652	3,270385
C3	98,184618	52,552652	3,270385
C4	98,184618	52,552652	3,270385
C5	289,975042	579,055753	118,289282
D1	289,975042	579,055753	118,289282
D2	289,975042	579,055753	118,289282
D3	289,975042	579,055753	118,289282
D4	289,975042	579,055753	118,289282
D5	289,975042	579,055753	118,289282

Tabla 4.1 Resultados del experimento E1 para el grafo Montage_25

DAG-corto			
Algoritmo	**Cluster_1-7**	**Node_1-49**	**Grid_1-1035**
O0	2107,692721	1562,226209	84,060619
A1	2107,692721	1562,226209	84,060619
A2	2107,692721	1562,226209	84,060619
A3	2107,692721	1562,226209	84,060619
A4	2107,692721	1562,226209	84,060619
A5	2107,692721	1562,226209	84,060619
B1	2107,692721	1562,226209	84,060619
B2	2107,692721	1562,226209	84,060619
B3	2107,692721	1562,226209	84,060619
B4	2107,692721	1562,226209	84,060619
B5	4117,348429	8223,627286	1678,619115
C1	2107,692721	1562,226209	84,060619
C2	2107,692721	1562,226209	84,060619
C3	2107,692721	1562,226209	84,060619
C4	2107,692721	1562,226209	84,060619
C5	4117,348429	8223,627286	1678,619115
D1	4117,348429	8223,627286	1678,619115
D2	4117,348429	8223,627286	1678,619115
D3	4117,348429	8223,627286	1678,619115
D4	4117,348429	8223,627286	1678,619115
D5	4117,348429	8223,627286	1678,619115

Tabla 4.2 *Resultados del experimento E1 para el grafo DAG-corto*

DAG-medio			
Algoritmo	**Cluster_1-7**	**Node_1-49**	**Grid_1-1035**
O0	9774,551252	5254,102495	274,147658
A1	9774,551252	5254,102495	274,147658
A2	9774,551252	5254,102495	274,147658
A3	9774,551252	5254,102495	274,147658
A4	9774,551252	5254,102495	274,147658
A5	9774,551252	5254,102495	274,147658
B1	9774,551252	5254,102495	274,147658
B2	9774,551252	5254,102495	274,147658
B3	9774,551252	5254,102495	274,147658
B4	9774,551252	5254,102495	274,147658
B5	28985,25315	57893,3243	11816,67718
C1	9774,551252	5254,102495	274,147658
C2	9774,551252	5254,102495	274,147658
C3	9774,551252	5254,102495	274,147658
C4	9774,551252	5254,102495	274,147658
C5	28985,25315	57893,3243	11816,67718
D1	28985,25315	57893,3243	11816,67718
D2	28985,25315	57893,3243	11816,67718
D3	28985,25315	57893,3243	11816,67718
D4	28985,25315	57893,3243	11816,67718
D5	28985,25315	57893,3243	11816,67718

Tabla 4.3 Resultados del experimento E1 para el grafo DAG-medio

DAG-largo			
Algoritmo	**Cluster_1-7**	**Node_1-49**	**Grid_1-1035**
O0	97741,52065	52540,91925	2736,648708
A1	97741,52065	52540,91925	2736,648708
A2	97741,52065	52540,91925	2736,648708
A3	97741,52065	52540,91925	2736,648708
A4	97741,52065	52540,91925	2736,648708
A5	97741,52065	52540,91925	2736,648708
B1	97741,52065	52540,91925	2736,648708
B2	97741,52065	52540,91925	2736,648708
B3	97741,52065	52540,91925	2736,648708
B4	97741,52065	52540,91925	2736,648708
B5	289851,4177	578932,1292	118165,658
C1	97741,52065	52540,91925	2736,648708
C2	97741,52065	52540,91925	2736,648708
C3	97741,52065	52540,91925	2736,648708
C4	97741,52065	52540,91925	2736,648708
C5	289851,4177	578932,1292	118165,658
D1	289851,4177	578932,1292	118165,658
D2	289851,4177	578932,1292	118165,658
D3	289851,4177	578932,1292	118165,658
D4	289851,4177	578932,1292	118165,658
D5	289851,4177	578932,1292	118165,658

Tabla 4.4 *Resultados del experimento E1 para el grafo DAG-largo*

ndgf_atlas_dax_500

Algoritmo	Cluster_1-7	Node_1-49	Grid_1-1035
O0	2264425,122	416032,2793	11325,3184
A1	2264425,122	416032,2793	11325,3184
A2	2264425,122	416032,2793	11325,3184
A3	2264425,122	416032,2793	11325,3184
A4	2264425,122	416032,2793	11325,3184
A5	2264425,122	416032,2793	11325,3184
B1	2264425,122	416032,2793	11325,3184
B2	2264425,122	416032,2793	11325,3184
B3	2264425,122	416032,2793	11325,3184
B4	2264425,122	416032,2793	11325,3184
B5	10555534,27	21082997,01	573925,7886
C1	2264425,122	416032,2793	11325,3184
C2	2264425,122	416032,2793	11325,3184
C3	2264425,122	416032,2793	11325,3184
C4	2264425,122	416032,2793	11325,3184
C5	10555534,27	21082997,01	573925,7886
D1	10555534,27	21082997,01	573925,7886
D2	10555534,27	21082997,01	573925,7886
D3	10555534,27	21082997,01	573925,7886
D4	10555534,27	21082997,01	573925,7886
D5	10555534,27	21082997,01	573925,7886

Tabla 4.5 Resultados del experimento E1 para el grafo ndgf_atlas_dax_500

Dada la naturaleza del experimento, en este caso simplemente se pretende comparar la diferencia de tiempos de finalización de la versión original del algoritmo y todas las demás versiones que utilizan propiedades no funcionales, para cada uno de los grafos de tareas.

Como puede observarse, en todos los escenarios, exceptuando las versiones B5, C5, D1, D2, D3, D4 y D5, todas las demás versiones del subalgoritmo de selección de recursos tienen el mismo tiempo de finalización que el algoritmo original sin utilización de propiedades no funcionales.

4.2.2 Discusión del experimento E1 sobre la eficacia

Como puede observarse, en todos los escenarios ideales sin fallos en los recursos e independientemente del grafo y de la plataforma, exceptuando las versiones B5, C5, D1, D2, D3, D4 y D5, todas las demás versiones del subalgoritmo de selección de recursos tienen el mismo tiempo de finalización que el algoritmo original sin utilización de propiedades no funcionales, independientemente de la forma de calcular el índice o *rating* de propiedades no funcionales (utilización independiente o combinada de la fiabilidad y la disponibilidad).

De todas las versiones que sí penalizan el tiempo de finalización de la simulación, cabe destacar que, también independientemente de la forma de calcular el índice o *rating* de propiedades no funcionales (utilización independiente o combinada de la fiabilidad y la disponibilidad), todas las versiones del criterio D ofrecen peores resultados. Se trata del algoritmo con factor de corrección basado en el mejor *rating*, con factor de discriminación en caso de igualdad basado en priorizar el recurso con mejor rating frente a aquel que ofrece el mejor tiempo estimado de finalización o EFT.

Además, la utilización del índice o *rating* de propiedades no funcionales basado en combinar fiabilidad y disponibilidad mediante su media, también ofrece peores resultados en los criterios de utilización del mismo como factor de corrección. En el caso de utilizarlo como factor de discriminación, se puede deducir que funciona bien en el algoritmo A5 porque este solamente lo utiliza en caso de igual tiempo de finalización estimado o EFT y no así en el algoritmo B porque este lo utiliza además para priorizar los recursos con mejor índice o *rating*.

Por lo tanto, respecto a la eficacia, se puede concluir que, evitando las combinaciones mencionadas, la utilización de las propiedades no funcionales

fiabilidad y disponibilidad en la selección de recursos no penaliza respecto al tiempo de simulación.

Ello permite continuar con el siguiente experimento con presencia de fallos en los recursos de computación con el objetivo de validar las hipótesis de la presente tesis.

4.2.3 Ejecución del experimento E2 sobre la efectividad

A continuación (tabla 4.6 a tabla 4.17) se muestran los datos estadísticos sobre los resultados obtenidos tras lanzar a ejecución todos los grafos de tareas de diferente duración (excepto el `ndgf_atlas_dax_500`) sobre cada una de las 5 versiones con fallo de cada una de las plataformas (para ofrecer diferentes escenarios de condiciones habituales e infraestructuras de pequeña, mediana y gran escala), utilizando todos los algoritmos de selección de recursos disponibles.

Por motivos de espacio, solamente se muestran tres columnas en cada tabla para evidenciar el número de fallos que se han producido y las diferencias entre medias y medianas de cada algoritmo respecto al algoritmo original, cuyo valor positivo el porcentaje de mejora en el tiempo de finalización medio del trabajo en cuestión sobre una determinada plataforma.

El resto de datos estadísticos concretos sobre cada algoritmo se documentan en el apéndice C de la presente tesis.

Montage_25 sobre NFP_cluster_1-7

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	0		
A1	0	0,023843%	0,034552%
A2	0	0,018779%	0,034379%
A3	0	0,044463%	0,034552%
A4	0	0,044463%	0,034552%
A5	0	0,044463%	0,034552%
B1	0	-467,92%	-446,11%
B2	0	-394,11%	-446,11%
B3	0	-459,84%	-446,11%
B4	0	-459,84%	-446,11%
B5	0	-459,84%	-446,11%
C1	0	-49,79%	-49,28%
C2	0	-0,53%	-0,07%
C3	0	-16,39%	-13,41%
C4	0	-45,30%	-38,88%
C5	0	-16,39%	-13,41%
D1	0	-195,34%	-195,34%
D2	0	-195,34%	-195,34%
D3	0	-195,34%	-195,34%
D4	0	-195,34%	-195,34%
D5	0	-195,34%	-195,34%

Tabla 4.6 Resultados del experimento E2 para el grafo Montage_25 sobre la plataforma NFP_cluster_1-7

Montage_25 sobre NFP_node_1-49

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	0		
A1	0	-0,020630%	0,004824%
A2	0	-0,065967%	0,004824%
A3	0	-0,020630%	0,004824%
A4	0	-0,020630%	0,004824%
A5	0	-0,020630%	0,004824%
B1	0	-793,84%	-633,63%
B2	0	-675,69%	-633,63%
B3	0	-793,84%	-633,63%
B4	0	-793,84%	-633,63%
B5	0	-793,84%	-633,63%
C1	0	-66,04%	-66,96%
C2	0	-0,38%	-0,19%
C3	0	-21,03%	-27,02%
C4	0	-66,16%	-66,96%
C5	0	-21,03%	-27,02%
D1	0	-1001,86%	-1001,86%
D2	0	-1001,86%	-1001,86%
D3	0	-1001,86%	-1001,86%
D4	0	-1001,86%	-1001,86%
D5	0	-1001,86%	-1001,86%

Tabla 4.7 Resultados del experimento E2 para el grafo Montage_25 sobre la plataforma NFP_node_1-49

Montage_25 sobre NFP_grid_1-1035

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	0		
A1	0	0,252374%	0,000000%
A2	0	0,593918%	0,000000%
A3	0	0,252374%	0,000000%
A4	0	0,252374%	0,000000%
A5	0	0,252374%	0,000000%
B1	0	-1469,63%	0,00%
B2	0	-800,17%	0,00%
B3	0	-1469,63%	0,00%
B4	0	-1469,63%	0,00%
B5	0	-4283,21%	-3516,98%
C1	0	-0,78%	0,00%
C2	0	0,59%	0,00%
C3	0	0,15%	0,00%
C4	0	-0,78%	0,00%
C5	0	-2813,44%	-3516,98%
D1	0	-3516,98%	-3516,98%
D2	0	-3516,98%	-3516,98%
D3	0	-3516,98%	-3516,98%
D4	0	-3516,98%	-3516,98%
D5	0	-3516,98%	-3516,98%

Tabla 4.8 Resultados del experimento E2 para el grafo Montage_25 sobre la plataforma NFP_grid_1-1035

DAG-corto sobre NFP_cluster_1-7

Alg.	*Fallos*	*Dif. Media*	*Dif. Mediana*
O0	0		
A1	0	0,001111%	0,001610%
A2	0	0,000875%	0,001601%
A3	0	0,002071%	0,001610%
A4	0	0,002071%	0,001610%
A5	0	0,002071%	0,001610%
B1	0	-275,72%	-261,29%
B2	0	-226,88%	-261,29%
B3	0	-270,37%	-261,29%
B4	0	-270,37%	-261,29%
B5	0	-270,37%	-261,29%
C1	0	-15,62%	-11,70%
C2	0	-0,70%	-0,72%
C3	0	-5,00%	-5,12%
C4	0	-14,44%	-11,45%
C5	0	-5,00%	-5,12%
D1	0	-95,35%	-95,35%
D2	0	-95,35%	-95,35%
D3	0	-95,35%	-95,35%
D4	0	-95,35%	-95,35%
D5	0	-95,35%	-95,35%

Tabla 4.9 Resultados del experimento E2 para el grafo DAG-corto sobre la plataforma NFP_cluster_1-7

DAG-corto sobre NFP_node_1-49

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	0		
A1	0	-0,000909%	-0,000375%
A2	0	-0,002649%	-0,000375%
A3	0	-0,000909%	-0,000375%
A4	0	-0,000909%	-0,000375%
A5	0	-0,000909%	-0,000375%
B1	0	-327,01%	-250,45%
B2	0	-270,55%	-250,45%
B3	0	-327,01%	-250,45%
B4	0	-327,01%	-250,45%
B5	0	-327,01%	-250,45%
C1	0	-21,66%	-23,51%
C2	0	-0,40%	-0,18%
C3	0	-13,97%	-16,78%
C4	0	-21,67%	-23,51%
C5	0	-13,97%	-16,78%
D1	0	-426,40%	-426,40%
D2	0	-426,40%	-426,40%
D3	0	-426,40%	-426,40%
D4	0	-426,40%	-426,40%
D5	0	-426,40%	-426,40%

Tabla 4.10 Resultados del experimento E2 para el grafo DAG-corto sobre la plataforma NFP_node_1-49

DAG-corto sobre NFP_grid_1-1035

Alg.	*Fallos*	*Dif. Media*	*Dif. Mediana*
O0	0		
A1	0	0,009538%	0,000000%
A2	0	0,023369%	0,000000%
A3	0	0,009538%	0,000000%
A4	0	0,009538%	0,000000%
A5	0	0,009538%	0,000000%
B1	0	-802,83%	0,00%
B2	0	-432,86%	0,00%
B3	0	-802,83%	0,00%
B4	0	-802,83%	0,00%
B5	0	-2320,36%	-1896,92%
C1	0	-2,95%	0,00%
C2	0	0,05%	0,00%
C3	0	-0,51%	0,00%
C4	0	-2,95%	0,00%
C5	0	-1518,04%	-1896,92%
D1	0	-1896,92%	-1896,92%
D2	0	-1896,92%	-1896,92%
D3	0	-1896,92%	-1896,92%
D4	0	-1896,92%	-1896,92%
D5	0	-1896,92%	-1896,92%

Tabla 4.11 Resultados del experimento E2 para el grafo DAG-corto sobre la plataforma NFP_grid_1-1035

DAG-medio sobre NFP_cluster_1-7

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	0		
A1	0	0,000240%	0,000347%
A2	0	0,000189%	0,000345%
A3	0	0,000447%	0,000347%
A4	0	0,000447%	0,000347%
A5	0	0,000447%	0,000347%
B1	1	-475,83%	-470,36%
B2	1	-372,19%	-371,38%
B3	1	-465,68%	-450,06%
B4	1	-465,68%	-450,06%
B5	1	-465,68%	-450,06%
C1	0	-47,84%	-38,23%
C2	0	-0,60%	-0,35%
C3	0	-15,55%	-13,46%
C4	0	-45,31%	-38,23%
C5	0	-15,55%	-13,46%
D1	1	-196,54%	-196,54%
D2	1	-196,54%	-196,54%
D3	1	-196,54%	-196,54%
D4	1	-196,54%	-196,54%
D5	1	-196,54%	-196,54%

Tabla 4.12 Resultados del experimento E2 para el grafo DAG-medio sobre la plataforma NFP_cluster_1-7

DAG-medio sobre NFP_node_1-49

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	2		
A1	0	-0,000206%	0,000048%
A2	1	-0,000877%	-0,000588%
A3	0	-0,000206%	0,000048%
A4	0	-0,000206%	0,000048%
A5	0	-0,000206%	0,000048%
B1	0	-793,80%	-633,56%
B2	2	-695,70%	-633,56%
B3	0	-793,80%	-633,56%
B4	0	-793,80%	-633,56%
B5	0	-793,80%	-633,56%
C1	0	-65,84%	-66,79%
C2	1	-0,21%	-0,21%
C3	1	-25,91%	-29,97%
C4	0	-65,96%	-66,79%
C5	1	-25,91%	-29,97%
D1	0	-1001,87%	-1001,87%
D2	0	-1001,87%	-1001,87%
D3	0	-1001,87%	-1001,87%
D4	0	-1001,87%	-1001,87%
D5	0	-1001,87%	-1001,87%

Tabla 4.13 Resultados del experimento E2 para el grafo DAG-medio sobre la plataforma NFP_node_1-49

DAG-medio sobre NFP_grid_1-1035

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	0		
A1	0	0,002924%	0,000000%
A2	0	0,007166%	0,000000%
A3	0	0,002924%	0,000000%
A4	0	0,002924%	0,000000%
A5	0	0,002924%	0,000000%
B1	0	-1756,12%	0,00%
B2	0	-957,51%	0,00%
B3	0	-1756,12%	0,00%
B4	0	-1756,12%	0,00%
B5	0	-5124,39%	-4210,33%
C1	0	-0,97%	0,00%
C2	0	0,01%	0,00%
C3	0	0,01%	0,00%
C4	0	-0,97%	0,00%
C5	0	-3368,26%	-4210,33%
D1	0	-4210,33%	-4210,33%
D2	0	-4210,33%	-4210,33%
D3	0	-4210,33%	-4210,33%
D4	0	-4210,33%	-4210,33%
D5	0	-4210,33%	-4210,33%

Tabla 4.14 Resultados del experimento E2 para el grafo DAG-medio sobre la plataforma NFP_grid_1-1035

DAG-largo sobre NFP_cluster_1-7

Alg.	Fallos	Dif. Media	Dif. Mediana
O0	3		
A1	3	0,000047%	0,000047%
A2	3	0,000035%	0,000035%
A3	3	0,000047%	0,000047%
A4	3	0,000047%	0,000047%
A5	3	0,000047%	0,000047%
B1	2	-470,36%	-448,46%
B2	2	-398,16%	-448,46%
B3	1	-465,70%	-450,08%
B4	1	-465,70%	-450,08%
B5	1	-465,70%	-450,08%
C1	3	-62,83%	-62,83%
C2	3	-1,23%	-1,23%
C3	2	-12,41%	-13,46%
C4	3	-62,56%	-62,56%
C5	2	-12,41%	-13,46%
D1	3	-196,55%	-196,55%
D2	3	-196,55%	-196,55%
D3	3	-196,55%	-196,55%
D4	3	-196,55%	-196,55%
D5	3	-196,55%	-196,55%

Tabla 4.15 Resultados del experimento E2 para el grafo DAG-largo sobre la plataforma NFP_cluster_1-7

DAG-largo sobre NFP_node_1-49			
Alg.	**Fallos**	**Dif. Media**	**Dif. Mediana**
O0	4		
A1	4	0,000005%	0,000005%
A2	5		
A3	4	0,000005%	0,000005%
A4	4	0,000005%	0,000005%
A5	4	0,000005%	0,000005%
B1	2	-458,35%	-370,75%
B2	2	-695,70%	-633,56%
B3	2	-458,35%	-370,75%
B4	2	-458,35%	-370,75%
B5	2	-458,35%	-370,75%
C1	4	-38,94%	-38,94%
C2	5		
C3	5		
C4	4	-38,94%	-38,94%
C5	5		
D1	1	-1001,87%	-1001,87%
D2	1	-1001,87%	-1001,87%
D3	1	-1001,87%	-1001,87%
D4	1	-1001,87%	-1001,87%
D5	1	-1001,87%	-1001,87%

Tabla 4.16 Resultados del experimento E2 para el grafo DAG-largo sobre la plataforma NFP_node_1-49

DAG-largo sobre NFP_grid_1-1035

Alg.	*Fallos*	*Dif. Media*	*Dif. Mediana*
O0	0		
A1	0	0,000293%	0,000000%
A2	0	0,000718%	0,000000%
A3	0	0,000293%	0,000000%
A4	0	0,000293%	0,000000%
A5	0	0,000293%	0,000000%
B1	0	-1759,25%	0,00%
B2	0	-959,22%	0,00%
B3	0	-1759,25%	0,00%
B4	0	-1759,25%	0,00%
B5	0	-5133,57%	-4217,90%
C1	0	-0,98%	0,00%
C2	0	0,00%	0,00%
C3	0	0,00%	0,00%
C4	0	-0,98%	0,00%
C5	0	-3374,32%	-4217,90%
D1	0	-4217,90%	-4217,90%
D2	0	-4217,90%	-4217,90%
D3	0	-4217,90%	-4217,90%
D4	0	-4217,90%	-4217,90%
D5	0	-4217,90%	-4217,90%

Tabla 4.17 Resultados del experimento E2 para el grafo DAG-largo sobre la plataforma NFP_grid_1-1035

4.2.4 Discusión del experimento E2 sobre la efectividad

Como puede observarse en la siguientes gráficas (figura 4.1 y figura 4.2), independientemente del grafo y de la plataforma, las versiones A1, A2 y C2 del subalgoritmo de selección de recursos ofrecen tiempos medios de finalización de la simulación menores que en el caso del algoritmo original O0 sin utilización de propiedades no funcionales, sobre todo calculando el índice o *rating* de propiedades no funcionales mediante la utilización independiente de la fiabilidad y disponibilidad. Incluso la mediana de dichos tiempos de finalización es menor o igual para dichos subalgoritmos basados en el algoritmo min-min original (el recurso con mejor tiempo estimado de finalización o EFT para la tarea), pero utilizando el mejor índice o *rating* como factor de discriminación en caso de igualdad.

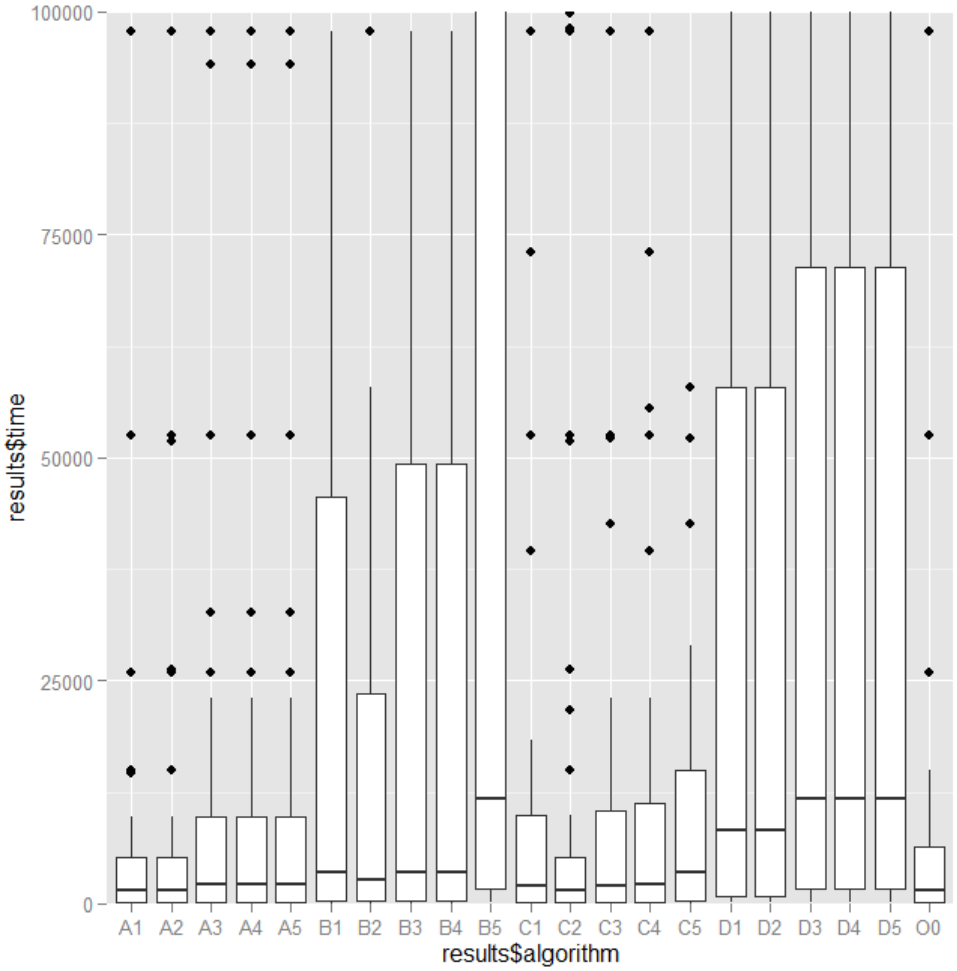

Figura 4.1 Comparativa general de los tiempos de finalización de todos los algoritmos

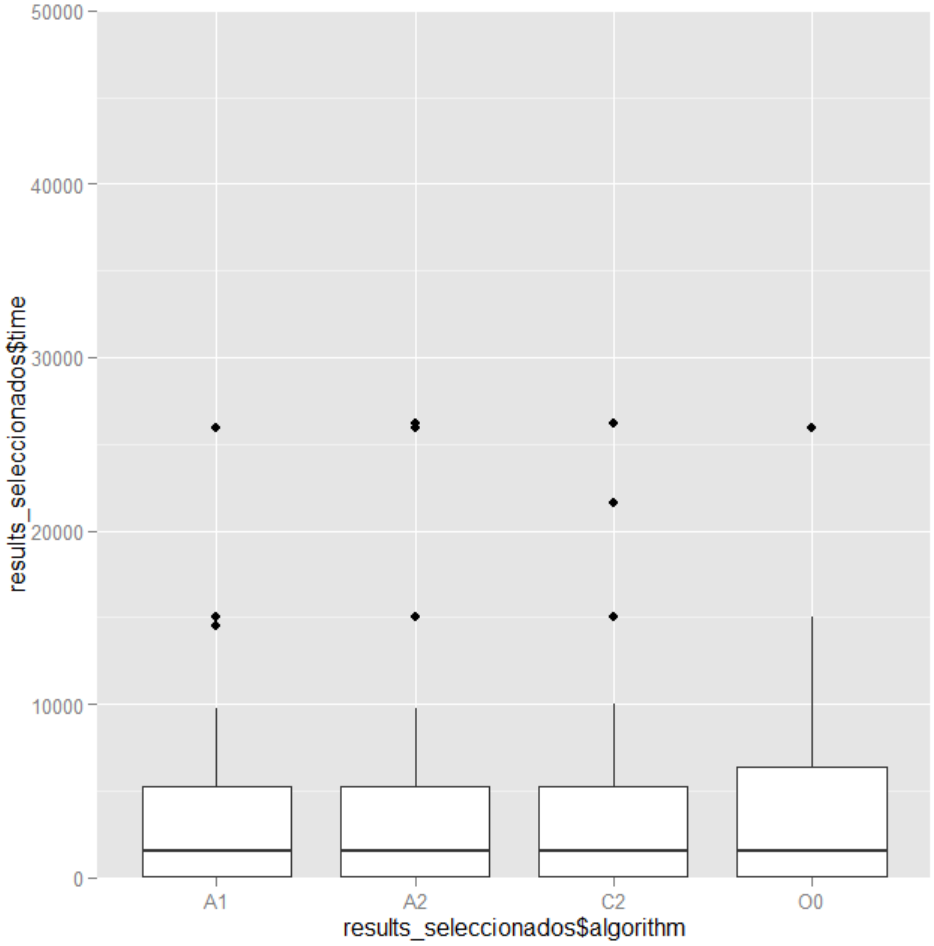

Figura 4.2 Comparativa de tiempos de finalización de los algoritmos seleccionados respecto al original

Desde el punto de vista de los fallos, dados los tiempos medios entre fallos (MTBF) y tiempos de recuperación (MTTR) generados realistamente para la simulación, cabe destacar que se producen en los escenarios con el grafo de mayor duración (como era de esperar) y las plataformas de pequeño y mediano tamaño, dado que en la de gran tamaño la disponibilidad de mayor número de recursos minimiza la posibilidad de fallo (como también era de esperar).

Por un lado, fijándonos en este segundo indicador y en los escenarios mencionados (grafo más largo y plataformas de pequeño y mediano tamaño), se puede apreciar que en todos los casos el número de fallos es menor (aproximadamente la mitad) para la familia de versiones B (B1, B2, B3, B4 y B5) del subalgoritmo de selección de recursos respecto a los que se producen con el algoritmo original sin utilización de propiedades no funcionales, independientemente de la forma de calcular el índice o *rating* de propiedades

no funcionales (utilización independiente o combinada de la fiabilidad y disponibilidad). Se trata del algoritmo que prioriza el recurso con mejor *rating* y que utiliza como factor de discriminación en caso de igualdad el recurso que ofrece un mejor tiempo estimado de finalización o EFT.

Por otro lado, esta reducción en cuanto al número de fallos lleva aparejada un mayor tiempo medio de finalización de la simulación en todos los casos (del orden de cinco veces mayor) respecto al algoritmo original sin utilización de propiedades no funcionales (caso base), por lo que solamente sería interesante este enfoque en casos muy puntuales en los que no estén disponibles o fallen los mecanismos de recuperación que suele haber implantados en este tipo de infraestructuras de computación.

Respecto a esta última circunstancia, se puede incluso especificar más en función del tamaño de la plataforma:

- En el caso de la pequeña plataforma, resultaría interesante seguir evaluando los subalgoritmos C3 y C5, dado que su tiempo medio de finalización apenas es un 12% mayor que en el caso base y han arrojado un significativo menor número de fallos. Se trata del algoritmo min-min con factor de corrección basado en el mejor *rating* (combinando fiabilidad y disponibilidad mediante su suma o media), que en caso de igualdad prioriza el recurso con mejor tiempo estimado de finalización o EFT, dejando dicho mejor índice o *rating* como factor de discriminación en caso de una nueva igualdad.
- En el caso de la plataforma de tamaño medio, la situación es todavía más extrema, ya que en el caso de la familia de subalgoritmos D (D1, D2, D3, D4 y D5) han arrojado un número mucho menor de fallos que el caso base, pero a cambio su tiempo medio de finalización es mucho mayor (del orden de 11 veces mayor). Se trata del algoritmo min-min con factor de corrección basado en el mejor *rating*, que en caso de igualdad prioriza el recurso con mejor *rating*, dejando el mejor tiempo estimado de finalización o EFT como factor de discriminación en caso de nueva igualdad.

A la luz de los resultados obtenidos, se puede concluir la validación de la hipótesis H1 de la presente tesis, como se remarcará en el siguiente apartado de conclusiones sobre la evaluación.

Dado que los resultados obtenidos están basados en la predicción y simulación de las propiedades no funcionales fiabilidad y disponibilidad sobre recursos de computación a partir de datos realistas sobre tiempos entre fallos y tiempos de

recuperación de fallos generados mediante modelos matemáticos presentes en la literatura del área, se puede concluir la validación de la hipótesis H2 de la presente tesis, como se remarcará en el apartado de conclusiones sobre la evaluación.

4.3 Pruebas estadísticas sobre los datos de los resultados

Se utiliza el test de Shapiro–Wilk para contrastar la normalidad del conjunto de datos obtenidos tras la experimentación por ser considerado uno de los test más potentes para el contraste de normalidad. Siendo la hipótesis nula que la población está distribuida normalmente, si el p-valor es menor a alfa (nivel de confianza) entonces la hipótesis nula es rechazada (se concluye que los datos no vienen de una distribución normal). Como se puede comprobar a continuación (código 4.1), en nuestro caso, el p-valor obtenido es menor que 0,05 (nivel de confianza) para todos los subconjunto de datos de una mismo algoritmo, por lo que rechazamos la hipótesis nula.

```
> # Test de normalidad
>
> for(algorithmType in levels(results$algorithm)){
+   grouptimes <- results$time[results$algorithm==algorithmType]
+   st <- shapiro.test(grouptimes)
+   print(paste0("Test de normalidad [ Group = ",algorithmType,"
                            ]-> ", st$p.value >= 0.05))
+ }
[1] "Test de normalidad [ Group = A1 ]-> FALSE"
[1] "Test de normalidad [ Group = A2 ]-> FALSE"
[1] "Test de normalidad [ Group = A3 ]-> FALSE"
[1] "Test de normalidad [ Group = A4 ]-> FALSE"
[1] "Test de normalidad [ Group = A5 ]-> FALSE"
[1] "Test de normalidad [ Group = B1 ]-> FALSE"
[1] "Test de normalidad [ Group = B2 ]-> FALSE"
[1] "Test de normalidad [ Group = B3 ]-> FALSE"
[1] "Test de normalidad [ Group = B4 ]-> FALSE"
[1] "Test de normalidad [ Group = B5 ]-> FALSE"
[1] "Test de normalidad [ Group = C1 ]-> FALSE"
[1] "Test de normalidad [ Group = C2 ]-> FALSE"
[1] "Test de normalidad [ Group = C3 ]-> FALSE"
[1] "Test de normalidad [ Group = C4 ]-> FALSE"
[1] "Test de normalidad [ Group = C5 ]-> FALSE"
[1] "Test de normalidad [ Group = D1 ]-> FALSE"
[1] "Test de normalidad [ Group = D2 ]-> FALSE"
[1] "Test de normalidad [ Group = D3 ]-> FALSE"
[1] "Test de normalidad [ Group = D4 ]-> FALSE"
[1] "Test de normalidad [ Group = D5 ]-> FALSE"
[1] "Test de normalidad [ Group = O0 ]-> FALSE"
```

Código 4.1 Test de Shapiro-Wilk para contrastar la normalidad de los resultados de los experimentos

Dado que no se puede contrastar la normalidad del conjunto de datos, se utiliza la prueba de Kruskal-Wallis como método no paramétrico para comprobar si el conjunto de datos proviene de la misma población. Intuitivamente, es idéntico al ANOVA con los datos reemplazados por categorías. Es una extensión de la prueba de la U de Mann-Whitney para 3 o más grupos. Ya que es una prueba no paramétrica, la prueba de Kruskal-Wallis no asume normalidad en los datos, en oposición al tradicional ANOVA. Sí asume, bajo la hipótesis nula, que los datos vienen de la misma distribución.

Dado que los datos de tiempos de finalización deberían variar en función del algoritmo utilizado, lo que interesa comprobar es que según ese criterio el conjunto de datos no debe pertenecer a la misma población.

```
> # Test de Kruskal-Wallis
>
> kruskal.test(results$time ~ results$algorithm)

        Kruskal-Wallis rank sum test

data:  results$time by results$algorithm
Kruskal-Wallis chi-squared = 159.72, df = 20, p-value < 2.2e-16
```

Código 4.2 Test de Kruskal-Wallis para contrastar la influencia de los algoritmos sobre los resultados de los experimentos

Como el p-valor obtenido (código 4.2) es menor que 0,05 (nivel de confianza), se puede inferir que el tiempo de finalización depende del algoritmo utilizado.

Por último, la prueba de los rangos con signo de Wilcoxon es una prueba no paramétrica para comparar la mediana de dos muestras relacionadas y determinar si existen diferencias entre ellas, en el caso de no poder suponer la normalidad de dichas muestras. Además, se deben cumplir las siguientes características: a) es libre de curva, no necesita una distribución específica, b) nivel ordinal de la variable dependiente, c) se utiliza para comparar dos mediciones de rangos (medianas) y determinar que la diferencia no se deba al azar (que la diferencia sea estadísticamente significativa). Se utiliza cuando la variable subyacente es continua pero no se presupone ningún tipo de distribución particular.

En este caso se utilizará para comprobar que efectivamente las medianas de tiempos de finalización del resto de subalgoritmos son significativamente mayores que las de los subalgoritmos seleccionados.

Dado que p-value<0.05 (código 4.3) se puede inferir que las medianas de tiempos de finalización del resto de subalgoritmos son significativamente mayores que las de los subalgoritmos seleccionados, tal y como se puede apreciar en la siguiente gráfica (figura 4.3). La diferencia entre medianas estimada es de 480.9297 y su intervalo de confianza estimado dicta que como mínimo hay una diferencia de 1380.74.

```
> selectedAlgorithms <- c("O0","A1","A2","C2")
> results$selectedAlgorithm <- c("RESTO",
                            "SELECCIONADOS")[as.numeric(results$
                            algorithm %in%
                            selectedAlgorithms)+1]
> wilcox.test(results$time ~ results$selectedAlgorithm, conf.int
                            = T, paired=F, alternative = "g")
        Wilcoxon rank sum test with continuity correction
data:  results$time by results$selectedAlgorithm
W = 221660, p-value = 1.181e-11
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 480.9297      Inf
sample estimates: difference in location 1380.74
```

Código 4.3 Test de Wilcoxon para comparar medianas de tiempos de finalización de los subalgoritmos

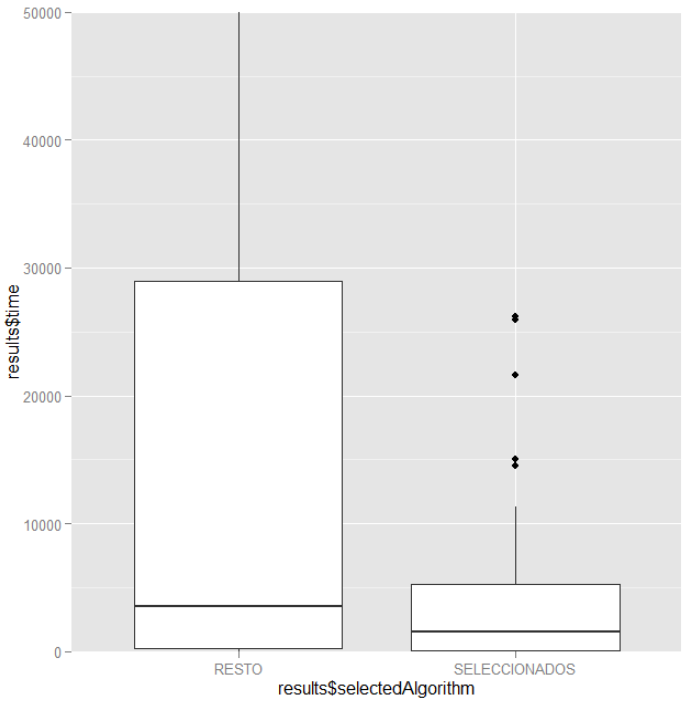

Figura 4.3 Comparativa de medianas de tiempos de finalización de los algoritmos

4.4 Conclusiones sobre la evaluación

En este capítulo se han presentado las descripciones y los resultados de los experimentos realizados sobre diferentes escenarios con el modelo de pruebas *SimDag* del simulador *SimGrid* para evaluar y validar a continuación cómo influyen varias políticas de utilización de propiedades no funcionales en el subalgoritmo de selección de recursos dentro del algoritmo min-min de planificación de trabajos.

Los **resultados obtenidos en los experimentos** llevados a cabo **respaldan, en general, que las estrategias de selección de recursos** para el algoritmo min-min **basadas en la utilización de las propiedades no funcionales** fiabilidad y disponibilidad como factor de discriminación en el caso de recursos con igual tiempo estimado de finalización o EFT **pueden ayudar a mejorar los tiempos de finalización** de los grafos de tareas, independientemente de la duración de las mismas y del tamaño de la plataforma sobre la que se ejecute, **dado que tampoco incrementan el número de fallos que se producen**.

También cabe mencionar que, ya sea como factor de discriminación o corrección, la priorización de indicadores estáticos relativos a propiedades no funcionales sobre los recursos respecto a la evaluación dinámica de la situación de un recurso no parece un enfoque adecuado. Si bien es algo que se podía intuir en un principio, ha quedado demostrado en los experimentos.

La mejora en los tiempos de finalización podría parecer insignificante, pero hay que tener en cuenta que se iría acumulando por cada trabajo lanzado a ejecución en cada recurso o nodo grid de computación. La evaluación de este extremo debería hacerse mediante un escenario de metaplanificación sobre una grid formada por varios recursos o nodos grid de computación que lamentablemente queda fuera del alcance de la presente tesis por las restricciones planteadas en la misma.

Desde el punto de vista de la reducción del número de fallos, solamente merece la pena seguir evaluando en pequeñas plataformas el algoritmo min-min con factor de corrección basado en el mejor *rating* (combinando fiabilidad y disponibilidad mediante su suma o media) y que en caso de igualdad prioriza el recurso con mejor tiempo estimado de finalización o EFT, dejando dicho mejor índice o *rating* como factor de discriminación en caso de una nueva igualdad. Ello es debido a que ofrece **un significativo menor número de fallos** respecto al caso base **a cambio de un incremento en el tiempo medio de finalización**, que sería **asumible siempre y cuando los mecanismos de recuperación** que suele haber implantados en este tipo de infraestructuras de

computación **no generen mayor beneficio,** o incluso **no estén disponibles o fallen.**

Los resultados de los experimentos de evaluación de la presente tesis están condicionados a las restricciones planteadas al inicio de la misma y que se vuelven a citar a continuación.

Restricción R3. Dada la dificultad de reproducir un entorno *grid* real y de conseguir datos reales de rendimiento sobre toda una infraestructura *grid*, los experimentos de evaluación de la tesis se realizarán con una herramienta de simulación implementando planificación y selección de recursos a nivel de nodo *grid*, quedando fuera del alcance de la tesis la realización de experimentos a nivel de metaplanificación.

Restricción R4. Dada la naturaleza de la evaluación de la tesis, no ha lugar a la utilización del modelo de representación propuesto en la misma. Así mismo, se adaptarán al contexto y la arquitectura del modelo de pruebas *SimDag* de *SimGrid* aquellas actividades de los métodos de representación de propiedades no funcionales desarrollados a partir de la metodología de la tesis.

Restricción R5. Los experimentos de evaluación de la tesis se realizarán con la herramienta *SimGrid*, modificando su modelo de pruebas *SimDag* que provee funcionalidades para simular la planificación y ejecución de tareas paralelas, dependientes o no, descritas siguiendo un modelo de grafos acíclicos dirigidos (*Direct Acyclic Graphs* o DAG) sobre una plataforma de recursos de computación descrita siguiendo un formato XML determinado y la implementación del popular algoritmo de planificación min-min en el que para cada tarea ejecutable se selecciona su mejor recurso de computación (aquel que minimiza su tiempo de finalización o completion time) para posteriormente seleccionar la tarea que tenga el menor tiempo de finalización de entre todas y planificarla sobre dicho mejor recurso de computación seleccionado al principio para la tarea.

Restricción R6. Los resultados de los experimentos de evaluación de la tesis se circunscriben al algoritmo de planificación mencionado anteriormente y las diferentes modificaciones realizadas sobre el mismo para la selección de recursos de computación mediante la utilización independiente o combinada de propiedades no funcionales fiabilidad y disponibilidad de los recursos de computación.

Restricción R7. Las propiedades no funcionales que se utilizarán en las modificaciones del algoritmo de planificación mencionado anteriormente y que se probarán en los diferentes escenarios que se planteen en los experimentos de evaluación de la tesis serán la fiabilidad y la disponibilidad de los recursos de computación como indicadores estáticos, dada la restricción planteada anteriormente respecto al modelo de pruebas.

Restricción R8. Los datos sobre fallos de los recursos que serán utilizados en la evaluación de la tesis para simular escenarios realistas serán generados siguiendo los modelos matemáticos más plausibles encontrados en el estado de la cuestión analizado.

En definitiva, los resultados de la evaluación de la presente tesis validan las dos hipótesis de partida enunciadas en la misma.

Hipótesis H1. Es posible utilizar propiedades no funcionales en la selección de recursos de computación *grid* para, en general, mejorar el tiempo de finalización medio o la tasa de fallos de un mismo trabajo lanzado a ejecución sobre un mismo nodo *grid*.

Hipótesis H2. Es posible utilizar herramientas matemáticas para la predicción y simulación de propiedades no funcionales sobre recursos de computación *grid*.

Y si pobre la encuentras, Ítaca no te engañó.
Así sabio como te hiciste, con tanta experiencia,
comprenderás ya qué significan las Ítacas.

Constantino Cavafis

Capítulo 5

Conclusiones y líneas futuras

Durante los capítulos precedentes se ha ido describiendo de forma detallada la investigación realizada en esta tesis doctoral. Se ha expuesto el análisis general del entorno de trabajo y las motivaciones para la realización de la tesis; las hipótesis y objetivos perseguidos con la realización de la misma; el análisis del estado de la cuestión junto con las necesidades y oportunidades de mejora identificadas en el mismo. Seguidamente se han descrito las características generales de las soluciones desarrolladas en la tesis, tanto el modelo como la metodología de representación de propiedades no funcionales para el dominio *grid*, así como el conjunto de algoritmos y el *framework* tecnológico desarrollados para la evaluación de la tesis. A continuación se han detallado los experimentos y pruebas realizadas con el fin de comprobar empíricamente las bondades de las soluciones propuestas. En este último capítulo se presentan las conclusiones que pueden extraerse de estas experiencias y, en definitiva, las aportaciones originales del trabajo. Asimismo se plantean algunas posibles acciones de mejora que podrían realizarse, a modo de líneas abiertas de investigación.

5.1 Conclusiones

En esta tesis se ha presentado un modelo híbrido basado en una ontología para representar recursos *grid* y propiedades no funcionales o de calidad de servicio asociados a los mismos y una metodología para especificar, calcular, medir y publicar datos sobre dichas propiedades. Se trata de un modelo que pretende promover el consenso en la representación de propiedades no funcionales de recursos en la *grid*, que sea interoperable con otros modelos de representación

existentes, que sea extensible para poder representar nuevas propiedades no funcionales y que sea genérico, de manera que no se circunscriba a un problema o dominio de aplicación concreto y pueda ser reutilizado, por ejemplo, junto al modelo de información S-OGSA para crear un modelo de información global del dominio *grid* que converja con los estándares de la *grid* propuestos desde el OGF y se apoye en la arquitectura S-OGSA para la semantización formal de la misma.

Tanto el modelo como la metodología pretenden ser una propuesta poco intrusiva, dado que sugiere desplegar servicios *grid* semánticos que cumplan las restricciones de cualquier servicio *grid* de manera que puedan ser consumidos fácilmente por el resto de servicios y aplicaciones de la *grid* y mejorar la funcionalidad de los proveedores y consumidores de información sobre recursos *grid*.

Respecto a la metodología, se ha definido y formalizado un proceso de desarrollo común para formalizar cualquier método de representación de la propiedad no funcional que se quiera abordar, indicando una serie de actividades y tareas a realizar.

Además, se han presentado sendos métodos de representación de propiedades no funcionales o indicadores de calidad de servicio sobre un recurso de computación: la fiabilidad y la disponibilidad. El proceso de desarrollo del método para representar la fiabilidad de un nodo *grid*, punto de partida para abordar posteriormente incluso el asociado para representar la fiabilidad de todo un sistema *grid*, no se ha completado por exceder el alcance de la presente tesis.

Los métodos de representación de la fiabilidad y la disponibilidad a nivel de recurso de computación desarrollados a partir de la metodología propuesta en la presente tesis han sido implementados usando simulación de datos para poder utilizar sus resultados en los experimentos asociados a la evaluación de la tesis.

En cuanto a los mecanismos de planificación de trabajos y de administración de recursos existentes en las infraestructuras *grid*, a pesar de los avances de los últimos años, el soporte para las propiedades no funcionales o de QoS en entornos de computación *grid* es aún muy limitado y, hasta el momento, no existe una solución definitiva para el problema.

Por ello y dadas las premisas y restricciones de la presente tesis, se ha desarrollado un *framework* tecnológico de pruebas sobre el entorno de simulación *SimDag* de *SimGrid* para realizar una prueba de concepto que

permita evaluar los efectos de utilizar propiedades no funcionales en la selección de recursos. El conjunto de desarrollos realizados, formado por una serie de modificaciones e implementaciones de nuevas funcionalidades sobre su modelo de pruebas y por un conjunto de programas para automatizar las tareas asociadas a los experimentos, han servido para demostrar que, en algunos casos, los tiempos de finalización de los trabajos y los errores en la selección de recursos son menores con un algoritmo de descubrimiento y selección de recursos basado en propiedades no funcionales respecto al caso de que no se tengan en cuenta dichas propiedades, concretamente la fiabilidad y la disponibilidad como indicadores estáticos de los recursos.

Los resultados de los experimentos de evaluación de la tesis se circunscriben al algoritmo de planificación min-min y las diferentes modificaciones realizadas sobre el mismo para la selección de recursos de computación mediante la utilización independiente o combinada de las propiedades no funcionales fiabilidad y disponibilidad de los recursos de computación.

Estos resultados respaldan, en general, que las estrategias de selección de recursos para el algoritmo min-min basadas en la utilización de las propiedades no funcionales fiabilidad y disponibilidad como factor de discriminación en el caso de recursos con igual tiempo estimado de finalización o EFT pueden ayudar a mejorar los tiempos de finalización de los grafos de tareas, independientemente de la duración de las mismas y del tamaño de la plataforma sobre la que se ejecute, dado que tampoco incrementan el número de fallos que se producen.

Si lo que se persigue es un significativo menor número de fallos asumiendo un incremento en el tiempo medio de finalización, siempre y cuando los mecanismos de recuperación que suele haber implantados en las infraestructuras de computación no generen mayor beneficio o no estén disponibles, merece la pena seguir evaluando el algoritmo min-min con factor de corrección basado en el mejor *rating* (combinando fiabilidad y disponibilidad mediante su suma o media) y que en caso de igualdad prioriza el recurso con mejor tiempo estimado de finalización o EFT, dejando dicho mejor índice o *rating* como factor de discriminación en caso de una nueva igualdad.

La principal conclusión a destacar es que la experimentación realizada en la evaluación ratifica las hipótesis de partida de la presente tesis.

A continuación, a modo de resumen, se cotejan los logros alcanzados con el cumplimiento tanto de los requisitos operacionales y arquitectónicos establecidos en un inicio, como de los objetivos operacionales y específicos, y por lo tanto, del objetivo general.

5.1.1 Cumplimiento de los requisitos

En el capítulo uno se establecieron cuatro requisitos operacionales y cuatro requisitos arquitectónicos. Los requisitos operacionales se han tratado de manera horizontal, es decir, en un desarrollo concreto de la tesis. En cambio, los requisitos arquitectónicos se han tratado de manera vertical, es decir, se han tenido en cuenta en todos los desarrollos. En la tabla 5.1 se muestran en qué desarrollos han sido tenidos en cuenta los requisitos operacionales.

Desarrollo	Requisitos operacionales
Modelo de representación de propiedades no funcionales para recursos grid	RO1, RO2
Metodología de representación de propiedades no funcionales para recursos grid	RO1, RO2
Métodos de representación de las propiedades no funcionales fiabilidad y disponibilidad para recursos de computación	RO1
Conjunto de algoritmos de selección de recursos	RO3
Framework tecnológico de pruebas sobre el entorno de simulación SimDag de SimGrid	RO4

Tabla 5.1 Requisitos operacionales respecto a los desarrollos de la tesis

5.1.2 Consecución de los objetivos operacionales

En el capítulo uno se establecieron diez objetivos operacionales. En la tabla 5.2 se muestra en qué apartados han sido abordados cada uno de ellos.

A la vista de la consecución de los diez objetivos operacionales planteados, los objetivos específicos pueden darse igualmente por alcanzados con los desarrollos de la tesis, así como el objetivo general, tal y como se muestra en la tabla 5.3.

	Objetivos operacionales	Apartado
OO1	Estudiar modelos de representación de información del dominio grid	2.3.1
OO2	Estudiar modelos de representación de propiedades no funcionales	2.3.2
OO3	Estudiar el middleware grid y las infraestructuras de computación grid	2.1, 2.2, 2.4
OO4	Diseñar un modelo común para la representación de propiedades no funcionales en la grid	3.1
OO5	Estudiar las soluciones que abordan el problema de la selección de recursos en la grid	2.5
OO6	Definir una metodología para la representación de propiedades no funcionales en la grid	3.2
OO7	Determinar las propiedades no funcionales sobre las que se aplicará la metodología de la tesis	2.5.4
OO8	Aplicar la metodología con propiedades no funcionales sobre recursos de distinto nivel	3.3, 3.4, 3.5
OO9	Seleccionar y configurar el entorno de pruebas adecuado para la evaluación de la tesis	2.6
OO10	Probar el uso de propiedades no funcionales en la selección de recursos grid	3.6, 4

Tabla 5.2 Objetivos operacionales más relevantes y el apartado donde se abordan

	Objetivos específicos	Objetivos operacionales
OE1	Definir la filosofía y el enfoque deseado para el modelo y la metodología de la tesis, así como para el entorno de pruebas	OO1, OO2, OO3
OE2	Desarrollar un sistema basado en un modelo y una metodología para la representación de propiedades no funcionales en el dominio grid	OO1, OO2, OO3, OO4, OO5, OO6
OE3	Seleccionar las propiedades no funcionales que van a emplearse en la experimentación	OO5, OO7, OO8
OE4	Diseñar e implementar la técnica de evaluación	OO7, OO8
OE5	Configurar un entorno de pruebas	OO9, OO10
OE6	Analizar y evaluar los resultados logrados en la experimentación	OO10

Tabla 5.3 Relación entre los objetivos específicos y los operacionales

Si bien todas las contribuciones científicas y tecnológicas de la presente tesis están descritas en el capítulo uno, en resumen, las principales aportaciones del presente trabajo de investigación son las siguientes:

Una propuesta de modelo unificado para la representación de propiedades no funcionales sobre recursos *grid* implementado mediante una ontología.

Una propuesta de metodología para la representación de propiedades no funcionales sobre recursos *grid* basada en un proceso de desarrollo que pretende ser una guía de ayuda para la elaboración de métodos de representación de propiedades no funcionales por parte de la comunidad *grid* en general y por los responsables de la administración de recursos *grid* en particular.

Conjunto de variantes del subalgoritmo de selección de recursos para el algoritmo de planificación min-min basadas en las propiedades no funcionales fiabilidad y disponibilidad de un recurso de computación para comprobar los resultados que se obtienen mediante su utilización combinada o no en el descubrimiento y selección de los recursos. Así como un conjunto de herramientas tecnológicas para los investigadores que necesiten utilizar entornos de simulación *grid* en general y que vayan a utilizar la herramienta *SimGrid* en particular.

5.1.3 Proyectos de investigación relacionados y publicaciones

El área de trabajo abordada en esta tesis ha dado lugar a la aprobación de dos proyectos de investigación con subvención externa, que corresponden a la convocatoria SAIOTEK del Departamento de Industria Comercio y Turismo del Gobierno Vasco. Estos datan de los periodos 2004/2005 y 2005/2006, tienen un carácter de investigación básica, fueron desarrollados en el contexto del grupo de investigación DELi de la Universidad de Deusto y el papel desempeñado por el autor de la tesis fue el de proporcionar el conocimiento científico sobre el paradigma de computación *grid*, los servicios web y las tecnologías semánticas.

Los dos proyectos fueron finalizados de una forma satisfactoria y sirvieron para avanzar en el conocimiento del estado de la cuestión sobre el área de trabajo de la presente tesis, detectar las limitaciones de las soluciones existentes, realizar propuestas iniciales y algunas publicaciones relacionadas con el contexto de la

tesis, contactar con otros investigadores relevantes en el área de trabajo e ir sentando las bases de la presente tesis doctoral.

De hecho, a través de dichos contactos el autor de la tesis ha podido realizar una estancia en el grupo de investigación IMG de la Universidad de Manchester para colaborar en el proyecto europeo OntoGrid, así como una serie de visitas a varios grupos de investigación y centros de supercomputación que han permitido centrar aún más el tema de la tesis: OEG de la Universidad Politécnica de Madrid, DSA de la Universidad Complutense de Madrid, I2Basque de la Universidad del País Vasco, CESGA de la Universidad de Santiago de Compostela, GRyCAP de la Universidad Politécnica de Valencia, CPC de la University of Westminster y FEUP de la Universidade do Porto.

En este sentido, los resultados de estos proyectos, las colaboraciones con otros investigadores y la evolución de la tesis a lo largo de los años han dado lugar a un total de cuatro publicaciones en congresos nacionales e internacionales, así como un capítulo de libro, vinculadas directamente con el tema de la tesis:

David Bujan Carballal, JosuKa Díaz Labrador, Joseba Koldobika Abaitua Odriozola, Inés Magdalena Jacob Taquet (2005) "XemGrid: Mejora en el descubrimiento de Grid services utilizando información semántica", *Jornadas Científico-Técnicas en Servicios Web (JSWEB 2005), I Congreso Español de Informática (CEDI 2005)*, Granada, ISBN: 84-9732-455-2, pp. 85-90.

David Bujan Carballal (2007) "Utilización de propiedades no funcionales para el descubrimiento de recursos Grid", Simposio de Doctorado en Web Semántica, XII Conferencia de la Asociación Española Para la Inteligencia Artificial (CAEPIA 2007). Salamanca.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2008) "Propiedades de Calidad de Servicio en el Descubrimiento de Recursos Grid", *IV Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB 2008)*, Sevilla.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2009) "A Model of Non Functional Properties for Grid Resources", *3rd CoreGRID Workshop on Grid Middleware (OGF23)*, Barcelona.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2009) "A Model of Non Functional Properties for Grid Resources", en Norbert Meyer, Domenico Talia, Ramin Yahyapour (eds.) *Grid and Services Evolution*, Springer, New York, ISBN: 978-0-387-85965-1, pp. 27-39.

5.2 Líneas de investigación futuras

Nadie discute la utilidad de las infraestructuras *grid* para llevar a cabo experimentos científicos de toda índole en el marco de la denominada e-Ciencia. Es seguro que el número de usuarios continuará creciendo, así como la diversidad de los recursos y servicios ofertados. Pero debido a las limitaciones de los sistemas de información, así como del *middleware grid* que provee y consume información sobre los recursos *grid*, la importancia de desarrollar e implantar soluciones que aborden de manera inteligente y unificada estas limitaciones serán fundamentales para mejorar el aprovechamiento de los recursos *grid* en particular y las prestaciones de las infraestructuras *grid* en general.

Con esta tesis se pretende realizar una pequeña aportación que contribuya a fomentar el desarrollo de nuevos servicios y funcionalidades en este sentido, pero son necesarias nuevas líneas de investigación que afronten diversos aspectos que no han sido abordados por esta tesis.

A continuación se mencionan algunas de ellas en relación a los diferentes desarrollos llevados a cabo en la presente tesis.

5.2.1 Modelo de representación para el dominio grid

Por extraño que pueda parecer, después de casi una década, se podría decir que aún no se ha integrado la visión de la *grid* semántica en las *grids* en producción ni existe un modelo unificado de información *grid*. Desde luego parece razonable pensar, a tenor de la falta de actividad en los últimos años de varios grupos de trabajo del OGF respecto al desarrollo de nuevos estándares para la *grid*, que no habrá muchos cambios respecto a los distintos modelos de información parcial existentes ya consolidados, por lo que persiste la necesidad de un modelo *grid* de información global que los unifique o al menos los relacione formal y metodológicamente.

A pesar del interés del OGF en unificar esfuerzos y modelos de información, hasta la fecha aún no existe un modelo estándar de información *grid* en general para representar todo el ecosistema de la *grid*.

Sería interesante desarrollar la posibilidad de integrar la ontología implementada para el modelo de representación de la presente tesis en la colección de ontologías de la Grid Ontology, dado que esta integra el modelo S-OGSA y contiene gran cantidad de clases para representar el ecosistema *grid*.

La ontología fundacional resultante podría ser el punto de partida para implementar el modelo global de información *grid*, ya que se podrían crear nuevas subclases, equivalencias y propiedades para asociar diferentes entidades *grid* con sus modelos de información estándar correspondientes (GLUE, WS-Agreement, JSDL, DFDL, etcétera).

5.2.2 Metodología de representación de propiedades no funcionales para el dominio grid

Dado que la propuesta S-OGSA no está adoptada aún por la comunidad *grid*, sería interesante contactar con investigadores que dispongan de infraestructuras *grid* de investigación (emuladas o reales) para abordar el despliegue y desarrollo de la arquitectura S-OGSA sobre alguna de ellas mediante la implementación de servicios *grid* que extiendan su funcionalidad (*Semantic Provisioning Service*, *Semantic Binding Provisioning Services*, SAGS o *Semantically Aware Grid Services*).

Mientras se trabaja en esa línea a largo plazo, a medio plazo sería interesante abordar el segundo enfoque más factible planteado en la actividad de publicación de la metodología de la presente tesis, consistente en el desarrollo de servicios *grid* de anotación y de consumo de metadatos sobre un determinado recurso y ampliar las propiedades del mismo en su entrada del sistema de información con referencias a los mismos.

5.2.3 Métodos de representación de propiedades no funcionales sobre recursos grid

Al no disponer de datos reales, las actividades de medición de los métodos desarrollados en la presente tesis se basan en los modelos matemáticos encontrados en la literatura relacionada con la medición de la fiabilidad y la disponibilidad a nivel de recurso de computación, así como en otros modelos matemáticos hallados para la generación de valores realistas simulados sobre tiempos entre fallos y tiempos de recuperación sobre dicho tipo de recursos.

Sería interesante contactar con investigadores que dispongan de infraestructuras *grid* reales y solicitarles acceso a datos reales sobre tiempos entre fallos y tiempos de reparación de los recursos en cuestión, de manera que

se puedan procesar dichos datos con técnicas de aprendizaje automático para obtener la ecuación lineal que represente más apropiadamente el comportamiento de los recursos en cuestión y comprobar si el modelo matemático que se obtiene ofrece mejores resultados en la selección de recursos que los modelos teóricos generales descritos en la literatura.

Otra línea de trabajo interesante sería desarrollar métodos para nuevas propiedades no funcionales sobre diferentes tipos de recursos *grid*, ya sean indicadores estáticos o dinámicos.

5.2.4 Nuevas capacidades en el descubrimiento de recursos

Respecto a las líneas futuras sobre este tema, se hace referencia a nuevos algoritmos de selección de recursos relacionados con el entorno de pruebas utilizado en la evaluación de la presente tesis, así como a la posibilidad de trabajar sobre entornos *grid* reales o emulados.

5.2.4.1 Algoritmos de selección de recursos para SimGrid

Con respecto a los subalgoritmos de selección de recursos para el algoritmo min-min de planificación y asignación de recursos a modo de gestor local o LRM de un nodo *grid*, se podrían enriquecer los experimentos con más variedad de plataformas y grafos ahondar en las conclusiones obtenidas en la evaluación de la presente tesis. También se podrían utilizar nuevos criterios en el algoritmo min-min incorporando otras propiedades no funcionales, diferentes combinaciones de las mismas, etcétera. Así mismo, sería interesante abordar la implementación de nuevos algoritmos de planificación para *SimDag* o incluso para SMPI, de manera que se puedan evaluar otros escenarios con aplicaciones basadas en paso de mensajes y no en grafos de tareas.

5.2.4.2 Algoritmos de selección de recursos para un entorno grid real o emulado

Como se ha mencionado en la presente tesis, a pesar de los avances de los últimos años, el soporte sobre propiedades no funcionales o de QoS en entornos de computación *grid* para los mecanismos de planificación de trabajos y de administración de recursos existentes en las infraestructuras *grid* es aún muy limitado y, hasta el momento, no existe una solución definitiva para el problema.

Una vez más, a través del contacto con investigadores que dispongan de infraestructuras *grid* de investigación (emuladas o reales) y con investigadores

en el área de la utilización de propiedades no funcionales en la *grid*, sería interesante abordar la integración de sus desarrollos con las propuestas de la presente tesis en el *middleware* estándar *de facto* de la *grid*, tanto el considerado proveedor como consumidor de información, de manera que se vayan dando avances significativos que impulsen la semantización formal de la *grid*.

Como complemento a la línea de trabajo futura mencionada anteriormente en relación a la actividad de publicación de la metodología de la presente tesis, a partir de propiedades no funcionales sobre los recursos *grid* (indicadores estáticos o dinámicos, cuantitativos o cualitativos) publicados en los sistemas de información *grid*, sería interesante abordar, por ejemplo, la implementación de nuevas funcionalidades relacionadas con los modelos de planificación del metaplanificador *GridWay*. En concreto, se podrían ampliar las políticas de selección de recursos basadas en la expresión RANK de priorización de recursos para poder realizar diferentes experimentos que evalúen su efectividad y eficacia.

Capítulo 6

Conclusions and future work

During the previous chapters the research done in this dissertation has been detailed. It has been discussed the general analysis of the work environment and the motivations for developing the thesis; hypothesis and objectives aimed at developing it; the analysis of the state of the art with the needs and opportunities for improvement identified on it. Then general characteristics of the solutions developed in the thesis have been described, both the model and the methodology for representing non-functional properties on the grid domain, as well as the set of algorithms and technological framework for the evaluation of the thesis. Next the experiments carried out to empirically test the benefits of the proposed solutions have been detailed. In this last chapter, the conclusions extracted from these experiences and, ultimately, the original contributions of this work are presented. It also raises some possible improvements to do in the future work.

6.1 Conclusions

This thesis has presented a hybrid model based on an ontology to represent grid resources and non-functional (quality of service) properties associated to them. This thesis has also presented a methodology to specify, calculate, assess and publish data on such properties. Such model aims at promoting a standard representation of non-functional properties on grid resources, which can be compatible with other existing representation models. This model is extensible and allows the definition of new non-functional properties, so it aims to be generic and not focused on a specific domain or environment. It could be used,

for example, in conjunction with the S-OGSA information model so that a global information model of the grid domain could be created integrating the grid standards proposed by OGF in order to promote a formal semantization for grid-based environments relying on the S-OGSA architecture.

Both, the proposed model and the proposed methodology aim at being non-intrusive: they suggest the creation of new semantic grid services which must operate within the constraints of existing grid services so that they can be consumed easily by other grid services or grid-based applications and they can also improve the functionality of producers and consumers of information on grid resources.

As far as the methodology is concerned, a common development process has been defined and formalized in order to guide the development of methods for the representation of non-functional properties which might be taken into consideration. Specifically, various activities (subdivided in several tasks) to be carried out have been listed.

Two methods for the representation of non-functional properties and indicators of quality of service on computational resources have also been presented: reliability and availability. The method for the representation of the reliability on a grid node, which is the starting point to eventually be able to estimate the reliability of a whole grid system, has not been completed because considered out of the scope of this thesis.

The methods for representing reliability and availability at computational resource level, developed according to the methodology proposed by this thesis, have been implemented using data simulation in order to use the related results in the experiments associated with the evaluation of the thesis.

As far as techniques for scheduling tasks and for managing resources available within the grid infrastructure are concerned, despite the advances and progress achieved in the last few years, there is very limited support on non-functional or QoS properties within grid environments. So far, no definitive solution has been identified.

Therefore, within the boundaries of the scope and objectives of this thesis, a technological testing framework based on the simulation environment *SimDag* for *SimGrid* tool has been developed. This provides a proof of concept for the evaluation of the usage of non-functional properties during the selection of resources. Various artifacts have been developed to prove that in some cases the job completion times and the number of errors during the selection of resources is lower when using an algorithm of resource discovery and selection based on non-functional properties (particularly reliability and availability as

static indicators of resources) than when not using them. Specifically, a set of amended and brand-new testing models have been implemented, and a set of tools for automating the execution of experiments has been put in place.

The results from the experiments for the evaluation of the thesis are limited to the scheduling algorithm min-min and around the various modifications made on it for the selection of computational resources using non-functional properties (reliability and availability) on computational resources in a standalone or combined fashion. These results generally confirm that the strategies for the selection of resources for the min-min algorithm, based on the use of non-functional properties (reliability and availability) as a selecting factor in case of resources with the same estimated completion time or EFT, can help to improve the completion time of DAG-based jobs, independently on their individual durations or the size of the platform upon the jobs are executed, mainly because the number of errors generated does not increase. If the main objective is to obtain a much lower number of errors, assuming an increase in the average completion time, provided that the recovery strategies which have deployed in the computational infrastructure do not produce a higher benefit or are not available, it is worth to keep evaluating the min-min algorithm with a correction factor based on the best "rating" (which combines reliability and availability via sum or average). In case of a tie, the algorithm priorities the resource with best estimated completion time or EFT, and then it uses such "rating" (or index) as a further ranking criterion.

The most remarkable conclusion is that the experiments performed in the evaluation phase of the thesis confirm the initial hypothesis of the thesis.

The following sections summarize how both the initial operational and architectural requirements, such as the operational and the specific goals of this thesis, have been fulfilled.

6.1.1 Fulfillment of the requirements

In the first chapter four operational requirements and four architectural requirements were defined. Operational requirements have been processed horizontally, whereby a specific requirement has been taken into consideration within a specific point developed by the thesis. On the other hand, architectural requirements have been processed vertically, which means they have been taken into account in each point developed by the thesis. Table 6.1 shows which points developed in the thesis have taken into account which operational requirements.

Points developed	Operational requirements
Model for representing non-functional properties on grid resources	RO1, RO2
Methodology for representing non-functional properties on grid resources	RO1, RO2
Methods for representing non-functional properties (reliability and availability) on grid resources	RO1
Set of resource selection algorithms	RO3
Technological testing framework on the simulation environment SimDag of SimGrid	RO4

Table 6.1 Operational requirements with respect to points developed in the thesis

	Operational objectives	Section
OO1	Review of models for representing information of the grid domain	2.3.1
OO2	Review of models for representing non-functional properties	2.3.2
OO3	Review of grid middleware and computational grid infrastructures	2.1, 2.2, 2.4
OO4	Design a unified model for representing non-functional properties in the grid	3.1
OO5	Review the solutions concerned with the selection of resources in the grid	2.5
OO6	Define a methodology for representing non-functional properties in the grid	3.2
OO7	Establish the non-functional properties on which the methodology of the thesis will be applied	2.5.4
OO8	Apply the methodology with non-functional properties on resources of different levels	3.3, 3.4, 3.5
OO9	Choose and configure a testing environment fit for the evaluation of the thesis	2.6
OO10	Validate the use of non-functional properties in the selection of grid resources	3.6, 4

Table 6.2 Most relevant operational objectives and paragraphs dealing with them

6.1.2 Fulfillment of the operational objectives

In the first chapter ten operational objectives were defined. Table 6.2 shows which sections have covered which operational objectives.

After fulfilling the ten operational objectives mentioned above, the specific objectives have been fulfilled implicitly by the points developed in the thesis, and also the main general objective has been fulfilled implicitly, as table 6.3 shows.

	Specific objectives	**Operational objectives**
OE1	*Define the philosophy and the appropriate scope of the model and methodology of the thesis*	OO1, OO2, OO3
OE2	*Develop a system based on a model and on a methodology for the representation of non-functional properties in the grid domain*	OO1, OO2, OO3, OO4, OO5, OO6
OE3	*Select non-functional properties which are going to be used in the experiments*	OO5, OO7, OO8
OE4	*Design and implement the technique to be used for the evaluation*	OO7, OO8
OE5	*Configure a testing environment*	OO9, OO10
OE6	*Analyse and evaluate the results obtained during the experiments*	OO10

Table 6.3 *Relationships among specific objectives and operational objectives*

All scientific and technological contributions of this thesis have been described in the first chapter. In summary, the main contributions of this research work are the following:

A proposed unified model for representing non-functional properties on grid resources, implemented using an ontology.

A proposed methodology for representing non-functional properties on grid resources based on a development process which aims at being a template for the implementation of methods for representing non-functional properties by the grid community, specifically by the administrators of grid resources.

A set of variants on the sub-algorithm of selecting resources for the scheduling algorithm min-min, based on non-functional properties (reliability and availability) of a computational resource; the results coming from using such sub-algorithm variants for the discovery of resources, have been analysed to determine their effectiveness in a standalone or combination fashion. A set of tools aimed at researchers who need to use grid simulation environments, specifically *SimGrid*, has also been developed.

6.1.3 Related research projects and publications

Some of the questions raised by this thesis were the inspiration behind two research projects, funded externally within the SAIOTEK programme of the Department of Industry, Commerce and Tourism of the Basque Government. Such projects took place respectively in 2004/2005 and 2005/2006. Their focus was basic research and they were developed behind DELi research group at the University of Deusto. The author of this thesis focused mainly on the dissemination of scientific knowledge on grid-based technologies, web services and semantic technologies.

Both projects were completed successfully and they were instrumental for gaining knowledge on the state of the art in the field covered by this thesis, for identifying the limitations of the existing solutions, for making initial contributions in the area, mainly in the form of publications, for establishing connections with other researchers in the field and for laying the foundations for this doctoral thesis.

Thanks to the academic connections established during these projects, the author of this thesis became a temporary member of the IMG research group at the University of Manchester and he worked on the European project OntoGrid. He also had the opportunity of visiting various supercomputing research centres. Visits to the following research groups and centres, particularly, were instrumental for defining the scope of this thesis: OEG at the Technical University of Madrid, DSA at the Complutense University of Madrid, I2Basque at the University of the Basque Country, CESGA at the University of Santiago de Compostela, GRyCAP at the Technical University of Valencia, CPC at the University of Westminster and FEUP at the Universidade do Porto.

In summary, the outcome of the two research projects mentioned before enriched with the experiences and collaborations matured with other researchers in the field, have crystallized in four publications presented in

national and international congresses and also a chapter in a book on the subject matter:

David Bujan Carballal, JosuKa Díaz Labrador, Joseba Koldobika Abaitua Odriozola, Inés Magdalena Jacob Taquet (2005) "XemGrid: Mejora en el descubrimiento de Grid services utilizando información semántica", *Jornadas Científico-Técnicas en Servicios Web (JSWEB 2005), I Congreso Español de Informática (CEDI 2005)*, Granada, ISBN: 84-9732-455-2, pp. 85-90.

David Bujan Carballal (2007) "Utilización de propiedades no funcionales para el descubrimiento de recursos Grid", Simposio de Doctorado en Web Semántica, XII Conferencia de la Asociación Española Para la Inteligencia Artificial (CAEPIA 2007). Salamanca.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2008) "Propiedades de Calidad de Servicio en el Descubrimiento de Recursos Grid", *IV Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB 2008)*, Sevilla.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2009) "A Model of Non Functional Properties for Grid Resources", *3rd CoreGRID Workshop on Grid Middleware (OGF23)*, Barcelona.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2009) "A Model of Non Functional Properties for Grid Resources", en Norbert Meyer, Domenico Talia, Ramin Yahyapour (eds.) *Grid and Services Evolution*, Springer, New York, ISBN: 978-0-387-85965-1, pp. 27-39.

6.2 Future work

Nobody argues against the effectiveness of grid infrastructures when it comes to carry out scientific experiments within the so called e-Science. It is clear the number of users, the number and diversity of services offered will keep increasing. However, in order to improve the utilization of grid resources and to improve the performance of the grid infrastructure in general, it is essential to develop and deploy solutions which tackle in an intelligent and consistent

way the limitations of the current information systems and of the current grid middleware which provides and consumes information on grid resources.

The current thesis tries to give a contribution by encouraging the development of new solutions and services designed to take in consideration and solve these problems. However new lines of investigation are necessary in order to tackle various aspects which have not been covered by this thesis.

The following sections suggest and present some of them.

6.2.1 Information model for the grid domain

It might sounds hardly believable but, after almost a decade, the vision of the semantic grid has not progressed yet into production. Also, a standard or unified grid information model does not exist yet. Moreover, it seems likely that, given the lack of activity on new grid standards by the various working groups of OGF, there will not be much change with respect to the various existing and established models of partial information. Therefore a new global grid information model, which might be able to unify them or at least to link them from a methodological and formal point of view, is still very much needed.

Despite the good intentions of the OGF forum to consolidate and unify efforts and models of information, no standard model of grid information which is able to represent the whole grid ecosystem exists yet.

An interesting idea worth pursuing would be the integration of the ontology implemented for the method of representation of this thesis with the collection of ontologies present in the Grid ontology, given that this integrates the S-OGSA model and already contains numerous classes which can represent the grid ecosystem.

The combined ontology could very well be the foundation for implementing a global grid information model, since it would be possible to create new subclasses, equivalences and properties to associate various grid entities with their corresponding standard information models (GLUE, WS-Agreement, JSDL, DFDL, etc.).

6.2.2 Methodology for representing non-functional properties in the grid domain

Since the S-OGSA proposal has not been adopted yet by the grid community, it would be interesting to contact researchers who are working on research or

development grid infrastructures (real or simulated ones) and try to develop with them grid services based on a S-OGSA architecture, whose implementation actually extend the functionality offered by the architecture itself (*Semantic Provisioning Service*, *Semantic Binding Provisioning Services*, SAGS or *Semantically Aware Grid Services*).

While the objective described above could require a long term effort, an objective still related to the methodology for representing non-functional properties, but more achievable in the short term, could be the development of grid services for the annotation and consumption of metadata on a specific resource and the extension of its properties at its entry point in the information system to reference these semantic grid services.

6.2.3 Methods for representing non-functional properties in the grid domain

Since no real data was available, the measurements while developing the methods of this thesis are based on mathematical models found in the literature on measuring reliability and availability at the computational resource level and on mathematical models designed to generate realistic simulated times between faults and realistic simulated times to repair.

It would be interesting to contact researchers working on real grid infrastructures and to ask them for data on time between faults and time to repair of their resources. Such data could be then analysed with machine learning techniques in order to obtain the linear equation which could represent the behaviour of these resources and therefore validate whether the mathematical model obtained offers better results when it comes to selecting resources than the general theoretical models described in the literature.

Another interesting future work would be to develop methods for new non-functional properties on several types of grid resources, regardless of whether they are static or dynamic indicators.

6.2.4 New capabilities in the discovery of resources

Possible future work on this topic could be the development of new algorithms for the selection of resources, associated with the testing environment used to evaluate this thesis, or the possibility of working on real or simulated grid environments.

6.2.4.1 Algorithms for the selection of resources on SimGrid

As far as the sub-algorithms for the selection of resources to be used within the min-min algorithm for the allocation and scheduling of resources by a local resource manager or LRM in a grid node, experiments could be enriched by extending the variety and number of platforms and DAG-based jobs covered. Deeper conclusions could be probably obtained in this way than those derived in the current thesis. Also, new criteria could be injected in the min-min algorithm by incorporating a different set of non-functional properties, and by combining them in different ways. Furthermore, it would be interesting to implement new scheduling algorithms for *SimDag* or even for SMPI, in such a way that different use cases could be evaluated using messaged-based applications rather than DAG-based jobs.

6.2.4.2 Algorithms for the selection of resources on a real or emulated grid environment

As already mentioned earlier, despite the advances which have taken place in the last few years, there is very limited support for non-functional and QoS properties in grid environment.

Once more, it would be interesting to contact researchers who are working on research or development grid infrastructures (real or simulated ones) and researchers in the field of non-functional properties for grid environments, in order to address the integration of their developments with the proposals of this thesis in the de facto standard grid middleware, both consumers and providers of information, so that they will take significant advances that encourage formal semantization of the grid.

Complementing the future work mentioned above regarding the activity of publishing defined in the methodology of this thesis, from non-functional properties on grid resources (static or dynamic, quantitative or qualitative indicators) published in the grid information systems, would be interesting to address, for example, the implementation of new functionalities related to scheduling models for *GridWay* metascheduler. Specifically, it could expand the resource selection policy based on the expression of RANK prioritization on resources to perform different experiments to assess their effectiveness and efficiency.

Referencias bibliográficas

K. Aida, A. Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi, U. Nagashima (2000) "Performance evaluation model for scheduling in global computing systems", *International Journal of High Performance Computing Applications*, 14(3), pp. 268-279.

R.J. Al-Ali, O.F. Rana, D.W. Walker, S. Jha, S. Sohail (2002) "G-QoSM: Grid service discovery using QoS properties", *Comput. Inform. J.*, 21(4), pp. 363-382.

R. Al-Ali, K. Amin, G. von Laszewski, O. Rana, D. Walker, M. Hategan, N. Zaluzec (2004a) "Analysis and Provision of QoS for Distributed Grid Applications", *J. Grid Comput.*, 2(2), pp. 163-182.

R. Al-Ali, A. Hafid, O. Rana, D. Walker (2004b) "An Approach for QoS Adaptation in Service-Oriented Grids", *Concurrency Comput. Pract. Exper.*, 16(5), pp. 401-412.

J. Almond, D. Snelling (1999) "UNICORE: Uniform Access to Supercomputing as an Element of Electronic Commerce", *Future Generation Comp. Syst.* 15(5-6), pp. 539-548.

D.P. Anderson (2004) "BOINC: A System for Public-Resource Computing and Storage", *Proc. of the 5th IEEE/ACM Int. Workshop on Grid Computing*.

Alain Andrieux, Dave Berry, Jon Garibaldi, Stephen Jarvis, Jon MacLaren, Djamila Ouelhadj, Dave Snelling (2003) "Open Issues in Grid Scheduling", *UK e-Science Institute Technical Report*, ISSN 1751-5971.

S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G.L. Rubini, G. Tortone, M.C. Vistoli (2005) "GridICE: a monitoring service for Grid systems", *Future Generation Computer Systems*, 21(4), pp. 559-571.

Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, Ming Xu (2007)

"Web Services Agreement Specification (WSAgreement)", *Global Grid Forum Technical Report GFD-R-P.107*.

R.E. Barlow, F. Proschan (1975) *Statistical theory of reliability and life testing: probability models*, Florida State Univ. Tallahassee.

William H. Bell, David G. Cameron, Luigi Capozza, A. Paul Millar, Kurt Stockingerand Floriano Zini (2009) "Optorsim - A Grid Simulator For Studying Dynamic Data Replication Strategies", *International Journal of High Performance Computing Applications*, 17(4), pp. 403–416.

Francine Berman (1998) "High-Performance Schedulers", en Ian Foster, Carl Kesselman (eds.) *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann.

L. Bitonti, T. Kiss, G. Terstyanszky, T. Delaitre, S. Winter, P. Kacsuk (2006) "Dynamic Testing of Legacy Code Resources on the Grid", *ACM International Conference on Computing Frontiers*, Ischia, Italy, May 2-5, ISBN 1-59593-302-6, pp. 261- 268.

M.C. Boudreau, D. Gefen, D. Straub (2001) "Validation in IS Research: A State-of-the-Art Assessment", *MIS Quarterly*, 25(1), pp. 1-16.

J.M. Brooke, D. Fellows, K. Garwood, C. Goble (2004a) "Semantic matching of Grid resource in Grid Computing", *Second European AcrossGrids Conference*, LNCS 3165, pp. 240-249.

John Brooke, Donal Fellows, Jon MacLaren (2004b) "Resource brokering: the EUROGRID/GRIP Approach", *Proceedings of the UK e-Science All Hands Meeting*.

David Bujan Carballal, JosuKa Díaz Labrador, Joseba Koldobika Abaitua Odriozola, Inés Magdalena Jacob Taquet (2005) "XemGrid: Mejora en el descubrimiento de Grid services utilizando información semántica", *Jornadas Científico-Técnicas en Servicios Web (JSWEB 2005), I Congreso Español de Informática (CEDI 2005)*, Granada, ISBN: 84-9732-455-2, pp. 85-90.

David Bujan Carballal (2007) "Utilización de propiedades no funcionales para el descubrimiento de recursos Grid", *Simposio de Doctorado en Web Semántica, XII Conferencia de la Asociación Española Para la Inteligencia Artificial (CAEPIA 2007)*, Salamanca.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2008) "Propiedades de Calidad de Servicio en el Descubrimiento de Recursos Grid", *IV Jornadas Cientifico-Técnicas en Servicios Web y SOA (JSWEB 2008)*, Sevilla.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2009a) “A Model of Non Functional Properties for Grid Resources", *3rd CoreGRID Workshop on Grid Middleware (OGF23)*, Barcelona.

David Bujan Carballal, Oscar Corcho García, JosuKa Díaz Labrador (2009b) “A Model of Non Functional Properties for Grid Resources”, en Norbert Meyer, Domenico Talia, Ramin Yahyapour (eds.) *Grid and Services Evolution*, Springer, New York, ISBN: 978-0-387-85965-1, pp. 27-39.

Rajkumar Buyya, Manzur Murshed (2002) “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing”, *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 14, pp. 1175–1220.

Henri Casanova (2001) “Simgrid: A Toolkit for the Simulation of Application Scheduling”, *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, IEEE Computer Society, pp. 430–438.

Henri Casanova, Arnaud Legrand, Loris Marchal (2003) “Scheduling Distributed Applications: the SimGrid Simulation Framework”, *Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid’03)*, IEEE Computer Society.

H. Casanova, A. Giersch, A. Legrand, M. Quinson, F. Suter (2014) “Versatile, scalable, and accurate simulation of distributed applications and platforms”, *Journal of Parallel and Distributed Computing*, vol. 74, nº 10, pp. 2899-2917.

Lawrence Chung (1993) *Representing and Using Non-Functional Requirements: A Process-Oriented Approach*, University of Toronto.

O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, C. Goble (2006) “An overview of S-OGSA: a Reference Semantic Grid Architecture”, *Journal of Web Semantics* 4(2), pp. 102-115.

K. Czajkowski, I. Foster, C. Kesselman (2005) “Agreement-based resource management”, Proc. of the IEEE, 93(3), pp. 631-643.

C. Dabrowski (2009) “Reliability in grid computing systems”, *Concurrency and Computation: Practice and Experience*, 21(8), pp. 927-959.

A. Dix, J. Finlay, G. Abowd, R. Beale (1998) *Human-Computer Interaction, second edition*, Prentice Hall, 1998.

G. Dobson, R. Lock, I. Sommerville (2005) "QoSOnt: a QoS ontology for service-centric systems", *31st EUROMICRO Conference on Software*

Engineering and Advanced Applications, 30 Aug.-3 Sept., pp. 80-87, http://digs.sourceforge.net/publications.htm

Fangpeng Dong, Selim G. Akl (2006) "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", *Ontario Queen's University Technical Report 2006-504*.

N. Doulamis, A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou, E. Varvarigos (2007) "Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing", *Computer Communications*, 30(3), pp. 499-515.

E. Downs, P. Clare, I. Coe (1998) *Structured Analysis and Design Method (SSADM)*, Prentice Hall.

Catalin L. Dumitrescu, Ian Foster (2005) "GangSim: a simulator for grid scheduling studies", *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)*, IEEE Computer Society, Volume 2, pp. 1151–1158.

D. Fernandez-Baca (1989) "Allocating modules to processors in a distributed system", *IEEE Trans. Software Eng.*, 15(11), pp. 1427-1436.

M. Fernández-López, A. Gómez-Pérez, J. Pazos, A. Pazos (1999) "Building a chemical ontology using methontology and the ontology design environment", *IEEE intelligent Systems*, 14(1), pp. 37-46.

I. Foster, C. Kesselman (1999a) "Globus: a toolkit-based grid architecture", *The grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann, Los Altos, CA, pp. 259-278.

I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, A. Roy (1999b) "A distributed resource management architecture that supports advance reservations and co-allocation", *Proc. of 7th Int. Workshop on QoS (IWQoS'99)*.

Ian Foster, Carl Kesselman, Steven Tuecke (2001) "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International Journal of Supercomputer Applications*, 15(3).

Ian Foster (2002a) "What is the Grid? A Three Point Checklist". *GRIDtoday*, 1(6).

Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke (2002b) "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", *Global Grid Forum Technical report*.

I. Foster, M. Fidler, A. Roy, V. Sander, L. Winkler (2004) "End-to-end quality of service for high-end applications", *Computer Communications*, 27(14), pp. 1375-1388.

Ian Foster (2006) "Globus Toolkit Version 4: Software for Service-Oriented Systems", *IFIP International Conference on Network and Parallel Computing*, LNCS 3779, pp. 2–13.

Foundation for Intelligent Physical Agents (2002) "FIPA Quality of Service Ontology Specification", http://www.fipa.org/specs/fipa00094

W.L. French, C.H. Bell (1996) *Organisational Development: Behavioral Science Interventions for Organization Improvement*, Prentice-Hall.

C. Goble, C. Kesselman, Y. Sure [eds.] (2005) *Semantic Grid: The Convergence of Technologies*, Dagsthul Seminar No 05271.

A. Gómez-Pérez, N. Juristo, C. Montes, J. Pazos (1997) *Ingeniería del conocimiento*, Centro de Estudios Ramón Areces, ISBN 84-8004-269-9.

A. Gómez-Pérez, M. Fernández-López, O. Corcho (2004) *Ontological Engineering*, Springer, ISBN: 978-1-85233-551-9.

E. Greenwood (1973) *Metodología de la investigación social*, Paidós.

Mark Hirschey (2005) *Fundamentals of Managerial Economics*, South-Western Pub., ISBN: 1490324288891.

G. Hodge (2000) *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files*, Digital Library Federation/Council on Library and Information Resources, Washington, DC.

R. de Hoog (1998) "Methodologies for building knowledge based systems: Achievements and prospects", *The Handbook of Applied Expert Systems*, pp. 1_11-1_14.

Fred Howell, Ross McNab (1998) "SimJava: a discrete event simulation package for Java with applications in computer systems modelling", *Proceedings of the First International Conference on Web-based Modelling and Simulation*, Society for Computer Simulation.

C. Hughes, J. Hillman (2006) "QoS Explorer: A Tool for Exploring QoS in Composed Services", *Proc. of the IEEE Int. Conf. on Web Services (ICWS)*, pp. 797-806.

IEEE (1990a) *IEEE Standard Glossary of Data Management Terminology*, IEEE Std. 610.5-1990, doi: 10.1109/IEEESTD.1990.94601.

IEEE (1990b) *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std. 610.12-1990 [reemplazado por IEEE (2010b)].

IEEE (1995) *IEEE Guide for Developing Software Life Cycle Processes*, IEEE Std. 1074.1-1995.

IEEE (1997) *Guide for Information Technology - Software Life Cycle Processes - Life Cycle Data*, IEEE Std. 12207.1-1997 [reemplazado por IEEE (2015)].

IEEE (1999) *IEEE Standard for Information Technology - Software Life Cycle Processes - Reuse Processes*, IEEE Std. 1517-1999 [reemplazado por IEEE (2010a)].

IEEE (2010a) *IEEE Standard for Information Technology--System and Software Life Cycle Processes--Reuse Processes*, IEEE Std. 1517-2010.

IEEE (2010b) *Systems and software engineering -- Vocabulary*, IEEE Std. 24765-2010.

IEEE (2015) *ISO/IEC/IEEE International Standard Systems and software engineering -- Content of life-cycle information items (documentation)*, IEEE Std. 15289-2015.

Z. Juhasz, P. Kacsuk, D. Kranzlmüller (2004) *Distributed and Parallel Systems: cluster and grid computing*, Springer.

Peter Kacsuk, Tamas Kiss, Gergely Sipos (2008) "Solving the grid interoperability problem by P-GRADE portal at workflow level", *Future Generation Computer Systems*, 24(7), pp. 744–751.

P. Kruchten (2000) *The Rational Unified Process: An Introduction*, Addison Wesley.

Katia Leal (2010) *Estrategias de planificación en infraestructuras grid federadas*, PhD thesis.

C.H. Lie, C.L. Hwang, F.A. Tillman (1977) "Availability of maintained systems: a state-of-the-art survey", *AIIE Transactions*, 9(3), pp. 247-259.

Dong Lu, Peter A. Dinda (2003) "GridG: Generating Realistic Computational Grids", *ACM SIGMETRICS Performance Evaluation Review*, 30(4).

K. Lu, R. Subrata, A.Y. Zomaya (2007) "On the performance-driven load distribution for heterogeneous computational grids", *J. Comp. System Sci.*, 73(8), pp. 1191-1206.

E.M. Maximilien, M.P. Singh (2004) "Toward Autonomic Web Services Trust and Selection", *Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC)*, New York, ACM Press, pp. 212–221, http://maximilien.org/publications/papers/2004/Maximilien+Singh04b.pdf

Métrica v.3 (s/f) *MÉTRICA versión 3. Metodología de planificación, desarrollo y mantenimiento de sistemas de información*, Ministerio para las Administraciones Públicas, España, http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html.

S.E. Middleton, M. Surridge, S. Benkner, G. Engelbrecht (2007) "Quality of Service Negotiation for Commercial Medical Grid Services", *J. Grid Comput.*, 5(4), pp. 429-447.

Manzur Murshed, Rajkumar Buyya (2002) "Using the GridSim Toolkit for Enabling Grid Computing Education", *Proceedings of the International Conference on Communication Networks and Distributed Systems Modeling and Simulation (CNDS 2002)*, SCS.

L. Paradela (2001) *Una metodología para la gestión del conocimiento*, tesis doctoral.

M. Parkin, S. van der Berghe, O. Corcho, D. Snelling, J. Brooke (2006) "Knowledge of the Grid: a Grid resource ontology", *Cracow Grid Workshop 2006*, Cracow, Poland.

H.S. Pinto, S. Staab, C. Tempich (2004) "DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolvInG", *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pp. 393-397.

H. B. Prajapati, V. A. Shah (2014) "Scheduling in Grid Computing Environment", *Fourth International Conference on Advanced Computing Communication Technologies (ACCT)*, pp. 315-324.

H. B. Prajapati, V. A. Shah (2015) "Analysis Perspective Views of Grid Simulation Tools", *Journal of Grid Computing*, vol. 13, nº 2, pp. 177-213.

R.S. Pressmann (2000) *Software Engineering, A practitioner's Approach, 5th Edition*, McGraw-Hill.

S. Rajasekaran, J. Reif [eds.] (2007) *Handbook of Parallel Computing: Models, Algorithms and Applications*, Chapman & Hall/CRC Computer & Information Science Series, ISBN: 9781584886235.

Ivan Rodero, Francesc Guim, Julita Corbalan, L.L. Fong, Y.G. Liu, S.M. Sadjadi (2008) "Looking for an Evolution of Grid Scheduling: Meta-Brokering", in *Grid Middleware and Services*, LNCS, pp. 105–119.

Mathilde Romberg (2002) "The UNICORE Grid infrastructure", *Scientific Programming*, 10(2), pp. 149–157.

Nelson S. Rosa, Paulo R. F. Cunha, George R. R. Justo (2002) *Process-NFL: A Language for Describing Non-Functional Properties*, IEEE Computer Society Press.

Alfonso Sánchez-Macián, Luis Bellido, Encarna Pastor (2005) "Ontologías para la Medida de la Calidad de Servicio Percibida", *V Jornadas de Ingeniería Telemática (Jitel 2005)*, ISBN 84-8408-346-2, pp. 693-700.

G. Schreiber (2000) *Knowledge engineering and management: the CommonKADS methodology*, MIT Press.

B. Schroeder, G. Gibson (2010) "A large-scale study of failures in high-performance computing systems", *IEEE Transactions on Dependable and Secure Computing*, 7(4), pp. 337-350.

G. Singh, C. Kesselman, E. Deelman (2007) "A provisioning model and its comparison with best-effort for performance-cost optimization in grids", *Proc. of the 16th Int. Symposium on High Performance Distributed Computing (HPDC '07)*, pp. 117–126.

V. Sipkova, V. Tran (2006) "Glite Lightweight Middleware for Grid Computing", *The 2nd International Workshop on Grid Computing for Complex Problems (GCCP'2006)*.

S. Sotiriadis, N. Bessis, F. Xhafa, N. Antonopoulos (2012) "From meta-computing to interoperable infrastructures: A review of meta-schedulers for HPC, grid and cloud", *2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 874-883.

Borja Sotomayor, Lisa Childers (2005) *Globus® Toolkit 4: Programming Java Services*, Morgan Kaufmann.

S. Staab, H.P. Schnurr, R. Studer, Y. Sure (2001) "Knowledge Processes and Ontologies", *IEEE Intelligent Systems*, 16(1), pp. 26–34.

D. Straub, D. Gefen, M.C. Boudreau (2004) "The ISWorld Quantitative", *Positivist Research Methods Website*, http://dstraub.cis.gsu.edu:88/quant/

G. Stuer, K. Vanmechelen, J. Broeckhove (2007) "A commodity market algorithm for pricing substitutable Grid resources", *Future Generation Comp. Syst.*, 23(5), pp. 688-701.

M. C. Suárez-Figueroa, K. Dellschaft, E. Montiel-Ponsoda, B. Villazón-Terrazas, Z. Yufei, G. Aguado de Cea, A. García, M. Fernández-López, A. Gómez-Pérez, M. Espinoza, M. Sabou (2008) *NeOn Deliverable D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks*, NeOn Project.

R. Subrata, A.Y. Zomaya, B. Landfeldt (2008) "A Cooperative Game Framework for QoS Guided Job Allocation Schemes in Grids", *IEEE Transactions of Computers*, 57(10), pp. 1413-1422.

Erik Torres (2010) *Técnicas de monitorización y diferenciación de servicios para la asignación de recursos en entornos de computación Grid, en base a indicadores de nivel de servicio*, PhD thesis.

V. Tosic, B. Pagurek, K. Patel (2003) "WSOL — A Language for the Formal Specification of Classes of Service for Web Services", *ICWS03*, Las Vegas, June 23–26, CSREA Press, pp. 375–381.

H.L. Truong, T. Fahringer (2004) "SCALEA-G: A unified monitoring and performance analysis system for the grid", *Scientific Programming*, 12(4), pp. 225-237.

Hong-Linh Truong, Thomas Fahringer, Francesco Nerieri, Schahram Dustdar (2005) "Performance Metrics and Ontology for Describing Performance Data of Grid Workflows", *IEEE International Symposium on Cluster Computing and Grid 2005 (CCGrid2005), 1st International Workshop on Grid Performability*, Cardiff, UK, May 9 - 12, IEEE Computer Society Press.

Jeffrey D. Ullman (1975) "NP-Complete Scheduling Problems", *Journal of Computer and System Sciences*, 10(3), pp. 384–393.

Luis Manuel Vilches Blázquez (2011) *Metodología para la integración basada en ontologías de información de bases de datos heterogéneas en el dominio hidrográfico*, tesis doctoral.

Y. Wadsworth (1998) "What is participatory Action Research?", *Action Research International*, Paper 2, http://www.aral.com.au/ari/p-ywadsworth98.html.

D.A. Waterman (1986) *A Guide to Expert Systems*, Addison-Wesley.

Min Xie, Kim-Leng Poh, Yuan-Shun Dai (2004) “Reliability of Grid Computing Systems”, *Computing System Reliability*, Springer, ISBN: 978-0-306-48636-4, pp. 179-205.

Q. Zheng, C.-K. Tham, B. Veeravalli (2008) “Dynamic Load Balancing and Pricing in Grid Computing with Communication Delay”, *J. Grid Comput.*, 6(3), pp. 239-253.

Referencias web

[1] Advanced Resource Connector - NorduGrid's ARC web site (*online*) http://www.nordugrid.org/middleware

[2] Askalon web site (*online*) http://www.askalon.org

[3] AssessGrid: Advanced Risk Assessment & Management for Trustable Grids (*online*) http://www.assessgrid.eu/

[4] BEinGRID: Business Experiments in Grid (*online*) http://www.beingrid.eu/

[5] BREIN: Business objective driven reliable and intelligent Grids for real business (*online*) http://www.eubrein.com/

[6] CIM based Grid Schema WG (*online*) http://forge.ogf.org/sf/projects/cgs-wg

[7] CluMon (*online*) http://clumon.ncsa.illinois.edu/

[8] Condor (*online*) https://research.cs.wisc.edu/htcondor/

[9] Condor-G (*online*) http://research.cs.wisc.edu/htcondor/tutorials/miron-condor-g-dagman-tutorial.html

[10] CSF (*online*) http://toolkit.globus.org/toolkit/docs/4.0/contributions/csf/

[11] DIRC project (*online*) http://www.dirc.org.uk

[12] DRAE (*online*) http://buscon.rae.es/drae/

[13] EGA Reference Model (*online*) http://forge.ogf.org/sf/projects/rm-wg

[14] EGEE Workload Manager Service - WMS project web site (*online*) http://glite.web.cern.ch/glite/packages/R3.0/deployment/glite-WMS/glite-WMS.asp

[15] FIPA-QoS specification (*online*) http:// www.fipa.org/specs/fipa00094

[16] Free On-line Dictionary of Computing (*online*) http://foldoc.org

[17] Ganglia web site (*online*) http://ganglia.sourceforge.net

[18] GangSim - A Simulator for Grid Scheduling Studies with support for uSLAs (*online*) http://people.cs.uchicago.edu/~cldumitr/GangSim

[19] gLite EGEE middleware for Grid computing (*online*) http://glite.web.cern.ch/glite/

[20] Global Grid Forum (*online*) http://www.ggf.org

[21] Globus Alliance (*online*) http://www.globus.org

[22] Globus Toolkit (*online*) http://www.globus.org/toolkit

[23] Globus Toolkit 4.0 WS_GRAM (*online*) http://www.globus.org/toolkit/docs/4.0/execution/wsgram

[24] GlobusToolkit Monitoring & Discovery System (MDS) (*online*) http://www.globus.org/toolkit/mds

[25] GLUE Schema Working Group (*online*) http://forge.ogf.org/sf/projects/glue-wg

[26] GOM Grid ontologies K-Wf Grid project (*online*) http://gom.kwfgrid.net/web/space/Grid+Ontologies

[27] GPT (*online*) http://toolkit.globus.org/grid_software/packaging/gpt.php

[28] GRASP web site (*online*) http://grail.sdsc.edu/projects/grasp

[29] Grid Benchmarking RG (*online*) http://forge.ogf.org/sf/projects/gb-rg

[30] Grid Computing Environments RG (*online*) http://forge.ogf.org/sf/projects/gce-rg

[31] Grid Reliability and Robustness RG (*online*) http://forge.ogf.org/sf/projects/gridrel-rg

[32] Grid Scheduling Architecture Research Group (*online*) http://forge.gridforum.org/projects/gsa-rg

[33] GridARM Askalon's Grid Resource Management System web site (*online*) http://www.dps.uibk.ac.at/projects/brokerage

[34] Gridbus Grid Service Broker web site (*online*) http://www.gridbus.org/broker

[35] GRIDCC: Grid Enabled Remote Instrumentation with Distributed Control and Computation (*online*) http://www.gridcc.org/cms/

[36] GridEcon: Grid Economics and Business Models (*online*) http://www.gridecon.eu/

[37] GridG (*online*) http://www.cs.northwestern.edu/ donglu/GridG.htm

[38] GridSim Toolkit (*online*) http://www.gridbus.org/gridsim

[39] GridWay meta-scheduler web site (*online*) http://www.gridway.org

[40] GRMS Grid (Lab) Resource Management web site (*online*) http://www.gridlab.org/WorkPackages/wp-9

[41] GSMO web site (*online*) http://www.gsmo.org

[42] Hawkeye - A Monitoring and Management Tool for Distributed Systems (*online*) http://www.cs.wisc.edu/condor/hawkeye

[43] HPC4U (*online*) https://www.cetic.be/HPC4U,497

[44] INCA web site (*online*) http://inca.sdsc.edu

[45] Information Modelling in OGSA (*online*) http://forge.ogf.org/sf/projects/rm-wg

[46] Job Submission Description Language WG (*online*) http://forge.ogf.org/sf/projects/jsdl-wg

[47] LCG/g-LITE broker web site (*online*) http://glite.web.cern.ch/glite/packages/R3.0/deployment/lcg-RB/lcg-RB.asp

[48] LSF (*online*) http://www.platform.com/Products/platform-lsf

[49] Maui (*online*) http://docs.adaptivecomputing.com/maui/index.php

[50] Moab Grid Scheduler web site (*online*) http://wb.clusterresources.com/pages/products/moab-grid-suite.php

[51] MonaLISA web site (*online*) http://monalisa.caltech.edu

[52] MOQ web site (*online*) http://www.moq.org/

[53] Nagios (*online*) https://www.nagios.org/

[54] Network Measurements Working Group (*online*) http://forge.ogf.org/sf/projects/nm-wg

[55] NFP4Grid owl file (*online*) http://morelab.deusto.es/ontologies/NFP4Grid.owl

[56] NGS Resource Broker web site (*online*) http://www.grid-support.ac.uk/content/view/205/196

[57] Nimrod-G Grid Resource Broker web site (*online*)
http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm

[58] OGF - Open Grid Forum (*online*) https://www.ogf.org

[59] OGF Work Groups web page (*online*)
https://forge.ogf.org/sf/sfmain/do/listProjects

[60] OGSA Basic Execution Services WG (*online*)
http://forge.ogf.org/sf/projects/ogsa-bes-wg

[61] OGSA Resource Selection Services WG (*online*)
http://forge.ogf.org/sf/projects/ogsa-rss-wg

[62] OGSA Resource Usage Service WG (*online*)
http://forge.ogf.org/sf/projects/rus-wg

[63] OGSA Working Group (*online*) http://forge.ogf.org/sf/projects/ogsa-wg

[64] OntoGrid web site (*online*) http://www.ontogrid.net

[65] Open Grid Services Architecture (OGSA), Version 1.5 (*online*)
http://www.ogf.org/documents/GFD.80.pdf

[66] Open Grid Services Architecture Glossary of Terms Version 1.6 (*online*)
http://www.ogf.org/documents/GFD.120.pdf

[67] OptorSim (*online*) http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html

[68] OWL (*online*) http://www.w3.org/TR/owl-ref/

[69] OWL-QoS web site (*online*)
http://www.ntu.edu.sg/home5/pg04878518/OWLQoSOntology.html

[70] Portable Barch System (*online*) http://www.pbsgridworks.com

[71] P-GRADE (*online*) http://portal.p-grade.hu/

[72] Reference Model Working Group (*online*)
http://forge.ogf.org/sf/projects/rm-wg

[73] RESERVOIR: Resources and Services Virtualization without Barriers (*online*) http://www.reservoirfp7.eu/

[74] Scheduling and Resource Management (*online*)
http://forge.ogf.org/sf/projects/srm

[75] SeCSE project (*online*) https://www.city.ac.uk/centre-for-human-computer-interaction-design/research/projects-list/secse

[76] Semantic Grid RG (*online*) http://forge.ogf.org/sf/projects/sem-rg

[77] SimGrid (*online*) http://simgrid.gforge.inria.fr

[78] SLA@SOI: Empowering the service industry with SLA-aware infrastructures (*online*) http://sla-at-soi.eu/

[79] SmartLM: Grid-friendly software licensing for location independent application execution (*online*) http://www.smartlm.eu/

[80] SORMA: Self-Organizing ICT Resource Management (*online*) http://sorma-project.org/

[81] Storage Resource Broker web site (*online*) http://www.sdsc.edu/srb/index.php/Main_Page

[82] Sun Grid Engine (*online*) http://www.sun.com/software/gridware

[83] Trusted Computing Research Group (*online*) http://forge.ogf.org/sf/projects/tc-rg

[84] UNICORE - Uniform Interface to Computing Resources (*online*) http://www.unicore.eu

[85] Usage Record WG (*online*) http://forge.ogf.org/sf/projects/ur-wg

[86] Web Services Agreement Specification (*online*) http://www.ogf.org/documents/GFD.107.pdf

[87] Webcom-G (*online*) https://www.scss.tcd.ie/Brian.Coghlan/meta/webcomg/index.html

[88] WebMDS (*online*) http://toolkit.globus.org/toolkit/docs/4.0/info/webmds/

[89] Wikipedia (*online*) https://es.wikipedia.org

[90] Worldwide LHC Computing Grid (*online*) http://lcg.web.cern.ch/LCG

[91] WS-QoS web site (*online*) http://www.csse.monash.edu.au/~shonali/ws-qos/index.html

Apéndice A

En este apéndice se listan las clases y propiedades de la ontología que implementa el modelo de información de representación de propiedades no funcionales sobre recursos *grid* de la presente tesis.

Namespace URI: `http://morelab.deusto.es/ontologies/NFP4Grid.owl#`

Classes: Access_Point | AdHocGridService | Advertised_QoS | Agreed_QoS | Agreement | Alpha | Autonomous_Client | Availability | BSD | BasicGridService | Benchmark | BenchmarkReliability | CPU | Castor | Client | Client_Role | Cluster | ClusterNode | CommandLine_Client | CommunicationReliability | CommunicationTime | ComputingElement | ComputingGrid | ComputingNode | ComputingReliability | ComputingResource | ComputingUnit | Data | DataBase | DataGrid | DataManagementService | DataReliability | DataSource | Dimension | Directory_Service | Disk | DomainSpecificQoSAttribute | EthernetAdapter | ExecutionManagementService | File | FrequencyMetric | GIIS | GNU_Linux | GRIS | GUI_Client | GlobusService | Globus_Gatekeeper | Globus_MDS_Service | GridBroker | GridClient | GridEntity | GridHost | GridHostRole | GridNode | GridNodeRole | GridPortal | GridResource | GridResourceManager | GridScheduler | GridService | GridSystem | GridSystemRole | GridTool | HSM | HardDrive | Hardware | HardwareReliability | HardwareResource | HighLevelGridService | HistoricalAvailability | IA-32 | IA-64 | InformationService | InstantAvailability | KnowledgeEntity | KnowledgeResource | KnowledgeService | LRMS | LocalFile | LocalService | MSWindows | MacOS | MeasurableQoSAttribute | Measurement | MeasurementConcept |

MeasurementOfSize | MonitoringResource | MonitoringService | MonitoringTool | MyrinetAdapter | NAS | NetworkAdapter | NetworkBandwidth | NetworkCommunication | NetworkConnection | NetworkHost | NetworkHostReliability | NetworkReliability | NetworkResource | NetworkRoute | NetworkRouteReliability | ODB | Observed_QoS | OperatingSystem | POSIXCompliantOS | Performance | PerformanceAnalysisService | PowerPC | Processing | Profile | Program | ProgramReliability | QoSAttribute | QoSConcept | QoSMetric | QoSProfile | QoSType | RDB | RLSFile | RateOfOccurrenceOfFailures | RealWorldResource | Reliability | RemoteFile | Requested_QoS | Requester_VO | Resource | ResourceReliability | Resource_Collection | Resource_VO | Role | RoundTripTime | RunTime | SLA | SecurityService | Sensor | Server | Server_Role | Service | ServiceGrid | ServiceParameter | ServiceProfile | ServiceQoSParameter | ServiceReliability | ServiceTime | Service_Agreement | SizeInfoExchanged | SizeMetric | Software | SoftwareApplication | SoftwareLibrary | SoftwareReliability | SoftwareResource | Speed | SpeedMetric | StorageElement | StorageNode | StorageObject | StorageReliability | StorageResource | System | SystemReliability | Throughput | TimeMetric | Token | UnitOfMeasurement | UnitOfSize | UnmeasurableQoSAttribute | UsageTime | UserInterface | UserInterfaceNode | VO | VirtualLink | VirtualNode | VirtualOrganization | WorkerNode | XMLDB | x86 | x86_80486 | x86_80586 | x86_80686 | x86_80786 | x86_80886

Properties: hasGridHostRole | hasGridNodeRole | hasGridSystemRole | hasQoSAttribute

Apéndice B

En este apéndice se documentan los detalles de implementación del entorno de simulación *SimGrid* y su modelo de pruebas *SimDag*.

Ejemplo de fichero de plataforma:

```
<?xml version='1.0'?>
 <!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
 <platform version="3">
 <AS  id="AS0"  routing="Full">
   <host id="Host1" power="3.300140519709234E9"/>
   <host id="Host2" power="3.867398877553016E9"/>
   <host id="Host3" power="1.6522665718098645E9"/>
   <host id="Host4" power="1.0759376792481766E9"/>
   <host id="Host5" power="2.4818410475340424E9"/>
   <host id="Host6" power="1.773869555571436E9"/>
   <host id="Host7" power="1.7843609176927505E9"/>
   <link id="l152" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l153" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l154" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l155" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l156" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l157" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l159" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l160" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l161" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l162" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l163" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l164" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l165" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l166" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l167" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l168" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l169" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l170" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l171" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l172" bandwidth="1.25E8" latency="1.0E-4" />
   <link id="l173" bandwidth="1.25E8" latency="1.0E-4" />
    <route src="Host1" dst="Host2">
     <link_ctn id="l155"/>
   </route>
    <route src="Host1" dst="Host3">
     <link_ctn id="l155"/>
     <link_ctn id="l154"/>
     <link_ctn id="l156"/>
   </route>
```

```
    <route src="Host1" dst="Host4">
      <link_ctn id="l152"/>
      <link_ctn id="l157"/>
    </route>
    <route src="Host1" dst="Host5">
      <link_ctn id="l152"/>
      <link_ctn id="l161"/>
    </route>
    <route src="Host1" dst="Host6">
      <link_ctn id="l166"/>
    </route>
    <route src="Host1" dst="Host7">
      <link_ctn id="l152"/>
      <link_ctn id="l169"/>
    </route>
    <route src="Host2" dst="Host3">
      <link_ctn id="l154"/>
      <link_ctn id="l156"/>
    </route>
    <route src="Host2" dst="Host4">
      <link_ctn id="l159"/>
    </route>
    <route src="Host2" dst="Host5">
      <link_ctn id="l162"/>
    </route>
    <route src="Host2" dst="Host6">
      <link_ctn id="l167"/>
    </route>
    <route src="Host2" dst="Host7">
      <link_ctn id="l154"/>
      <link_ctn id="l170"/>
    </route>
    <route src="Host3" dst="Host4">
      <link_ctn id="l160"/>
    </route>
    <route src="Host3" dst="Host5">
      <link_ctn id="l163"/>
    </route>
    <route src="Host3" dst="Host6">
      <link_ctn id="l163"/>
      <link_ctn id="l168"/>
    </route>
    <route src="Host3" dst="Host7">
      <link_ctn id="l156"/>
      <link_ctn id="l170"/>
    </route>
    <route src="Host4" dst="Host5">
      <link_ctn id="l164"/>
    </route>
    <route src="Host4" dst="Host6">
      <link_ctn id="l159"/>
      <link_ctn id="l167"/>
    </route>
    <route src="Host4" dst="Host7">
      <link_ctn id="l171"/>
    </route>
    <route src="Host5" dst="Host6">
      <link_ctn id="l168"/>
    </route>
    <route src="Host5" dst="Host7">
      <link_ctn id="l172"/>
    </route>
    <route src="Host6" dst="Host7">
      <link_ctn id="l173"/>
    </route>
</AS></platform>
```

Ejemplo de fichero con un grafo de tareas:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- generated: 2008-09-24T14:28:09-07:00 -->
<!-- generated by: shishir [??] -->
<adag xmlns="http://pegasus.isi.edu/schema/DAX"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://pegasus.isi.edu/schema/DAX
         http://pegasus.isi.edu/schema/dax-2.1.xsd" version="2.1" count="1"
         index="0" name="test" jobCount="25" fileCount="0" childCount="20">
<!-- part 1: list of all referenced files (may be empty) -->
<!-- part 2: definition of all jobs (at least one) -->
  <job id="ID00000" namespace="Montage" name="mProjectPP" version="1.0"
         runtime="13390">
    <uses file="region.hdr" link="input" register="true" transfer="true"
         optional="false" type="data" size="304"/>
    <uses file="2mass-atlas-ID00000s-jID00000.fits" link="input"
         register="true" transfer="true" optional="false" type="data"
         size="4222080"/>
    <uses file="p2mass-atlas-ID00000s-jID00000.fits" link="output"
         register="true" transfer="true" optional="false" type="data"
         size="4167312"/>
    <uses file="p2mass-atlas-ID00000s-jID00000_area.fits" link="output"
         register="true" transfer="true" optional="false" type="data"
         size="4167312"/>
  </job>
...
<!-- part 3: list of control-flow dependencies (may be empty) -->
  <child ref="ID00005">
    <parent ref="ID00001"/>
    <parent ref="ID00000"/>
  </child>
...
  <child ref="ID00024">
    <parent ref="ID00023"/>
  </child>
</adag>
```

Funciones para el manejo de estaciones de trabajo o *workstations*:

```
SD_workstation_t      SD_workstation_get_by_name (const char *name)
         Returns a workstation given its name.
const SD_workstation_t *    SD_workstation_get_list (void)
         Returns the workstation list.
int      SD_workstation_get_number (void)
         Returns the number of workstations.
void     SD_workstation_set_data (SD_workstation_t workstation, void *data)
         Sets the user data of a workstation.
void *   SD_workstation_get_data (SD_workstation_t workstation)
         Returns the user data of a workstation.
const char *    SD_workstation_get_name (SD_workstation_t workstation)
         Returns the name of a workstation.
xbt_dict_t      SD_workstation_get_properties (SD_workstation_t workstation)
         Returns a xbt_dict_t consisting of the list of properties assigned to
         this workstation.
const char *    SD_workstation_get_property_value (SD_workstation_t
         workstation, const char *name)
         Returns the value of a given workstation property.
void     SD_workstation_dump (SD_workstation_t ws)
         Displays debugging informations about a workstation.
```

```
const SD_link_t *     SD_route_get_list (SD_workstation_t src, SD_workstation_t
          dst)
          Returns the route between two workstations.
int       SD_route_get_size (SD_workstation_t src, SD_workstation_t dst)
          Returns the number of links on the route between two workstations.
double    SD_workstation_get_power (SD_workstation_t workstation)
          Returns the total power of a workstation.
double    SD_workstation_get_available_power (SD_workstation_t workstation)
          Returns the proportion of available power in a workstation.
e_SD_workstation_access_mode_t     SD_workstation_get_access_mode
          (SD_workstation_t workstation)
          Returns the access mode of this workstation.
void      SD_workstation_set_access_mode (SD_workstation_t workstation,
          e_SD_workstation_access_mode_t access_mode)
          Sets the access mode for the tasks that will be executed on a
          workstation.
double    SD_workstation_get_computation_time (SD_workstation_t workstation,
          double computation_amount)
          Returns an approximative estimated time for the given computation
          amount on a workstation.
double    SD_route_get_current_latency (SD_workstation_t src, SD_workstation_t
          dst)
          Returns the latency of the route between two workstations, i.e. the
          sum of all link latencies between the workstations.
double    SD_route_get_current_bandwidth (SD_workstation_t src,
          SD_workstation_t dst)
          Returns the bandwidth of the route between two workstations, i.e. the
          minimum link bandwidth of all between the workstations.
double    SD_route_get_communication_time (SD_workstation_t src,
          SD_workstation_t dst, double communication_amount)
          Returns an approximative estimated time for the given communication
          amount between two workstations.
SD_task_t       SD_workstation_get_current_task (SD_workstation_t workstation)
          Returns the kind of the task currently running on a workstation Only
          call this with sequential access mode set.
```

Funciones para el manejo de las tareas o *tasks*:

```
SD_task_t       SD_task_create (const char *name, void *data, double amount)
          Creates a new task.
void *    SD_task_get_data (SD_task_t task)
          Returns the user data of a task.
void      SD_task_set_data (SD_task_t task, void *data)
          Sets the user data of a task.
e_SD_task_state_t     SD_task_get_state (SD_task_t task)
          Returns the state of a task.
const char *    SD_task_get_name (SD_task_t task)
          Returns the name of a task.
void      SD_task_set_name (SD_task_t task, const char *name)
          Allows to change the name of a task.
void      SD_task_set_rate (SD_task_t task, double rate)
          Sets the rate of a task.
void      SD_task_watch (SD_task_t task, e_SD_task_state_t state)
          Adds a watch point to a task.
void      SD_task_unwatch (SD_task_t task, e_SD_task_state_t state)
          Removes a watch point from a task.
double    SD_task_get_amount (SD_task_t task)
          Returns the total amount of work contained in a task.
double    SD_task_get_alpha (SD_task_t task)
          Returns the alpha parameter of a SD_TASK_COMP_PAR_AMDAH task.
double    SD_task_get_remaining_amount (SD_task_t task)
          Returns the remaining amount work to do till the completion of a
          task.
```

```
double   SD_task_get_execution_time (SD_task_t task, int workstation_nb, const
         SD_workstation_t *workstation_list, const double *computation_amount,
         const double *communication_amount)
         Returns an approximative estimation of the execution time of a task.
void     SD_task_schedule (SD_task_t task, int workstation_nb, const
         SD_workstation_t *workstation_list, const double *computation_amount,
         const double *communication_amount, double rate)
         Schedules a task.
void     SD_task_unschedule (SD_task_t task)
         Unschedules a task.
double   SD_task_get_start_time (SD_task_t task)
         Returns the start time of a task.
double   SD_task_get_finish_time (SD_task_t task)
         Returns the finish time of a task.
xbt_dynar_t     SD_task_get_parents (SD_task_t task)
         Returns the dynar of the parents of a task.
xbt_dynar_t     SD_task_get_children (SD_task_t task)
         Returns the dynar of the parents of a task.
int      SD_task_get_workstation_count (SD_task_t task)
         Returns the amount of workstations involved in a task.
SD_workstation_t *    SD_task_get_workstation_list (SD_task_t task)
         Returns the list of workstations involved in a task.
void     SD_task_destroy (SD_task_t task)
         Destroys a task.
void     SD_task_dump (SD_task_t task)
         Displays debugging informations about a task.
void     SD_task_dotty (SD_task_t task, void *out_FILE)
         Dumps the task in dotty formalism into the FILE* passed as second
         argument.
SD_task_t       SD_task_create_comp_seq (const char *name, void *data, double
         amount)
         create a sequential computation task that can then be auto-scheduled
SD_task_t       SD_task_create_comp_par_amdahl (const char *name, void *data,
         double amount, double alpha)
         create a parallel computation task that can then be auto-scheduled
SD_task_t       SD_task_create_comm_e2e (const char *name, void *data, double
         amount)
         create a end-to-end communication task that can then be auto-
         scheduled
SD_task_t       SD_task_create_comm_par_mxn_1d_block (const char *name, void
         *data, double amount)
         create a complex data redistribution task that can then be auto-
         scheduled
void     SD_task_distribute_comp_amdhal (SD_task_t task, int ws_count)
         Blah.
void     SD_task_schedulev (SD_task_t task, int count, const SD_workstation_t
         *list)
         Auto-schedules a task.
void     SD_task_schedulel (SD_task_t task, int count,...)
         autoschedule a task on a list of workstations
void     SD_task_set_category (SD_task_t task, const char *category)
         Sets the tracing category of a task.
const char *    SD_task_get_category (SD_task_t task)
         Gets the current tracing category of a task.
```

Funciones para el manejo de las dependencias entre tareas:

```
void     SD_task_dependency_add (const char *name, void *data, SD_task_t src,
         SD_task_t dst)
         Adds a dependency between two tasks.
void     SD_task_dependency_remove (SD_task_t src, SD_task_t dst)
         Remove a dependency between two tasks.
const char *    SD_task_dependency_get_name (SD_task_t src, SD_task_t dst)
         Returns the name given as input when dependency has been created..
void *   SD_task_dependency_get_data (SD_task_t src, SD_task_t dst)
         Returns the user data associated with a dependency between two tasks.
int      SD_task_dependency_exists (SD_task_t src, SD_task_t dst)
         Indicates whether there is a dependency between two tasks.
```

Funciones para la inicialización de *SimDag*, la creación del entorno de ejecución, el lanzamiento de la simulación y la finalización de *SimDag*.

```
void     SD_init (int *argc, char **argv)
         Initializes SD internal data.
void     SD_application_reinit (void)
         Reinits the application part of the simulation (experimental feature)
void     SD_create_environment (const char *platform_file)
         Creates the environment. The workstations are created when you call
         the function SD_create_environment.
xbt_dynar_t     SD_simulate (double how_long)
         Launches the simulation.
double   SD_get_clock (void)
         Returns the current clock.
void     SD_exit (void)
         Destroys all SD internal data.
xbt_dynar_t     SD_daxload (const char *filename)
         loads a DAX file describing a DAG
xbt_dynar_t     SD_dotload (const char *filename)
         loads a DOT file describing a DAG
```

Código original del algoritmo min-min del modelo de pruebas de *SimDag*:

```
/* simple test to schedule a DAX file with the Min-Min algorithm.           */

/* Copyright (c) 2009-2015. The SimGrid Team.
 * All rights reserved.                                                     */

/* This program is free software; you can redistribute it and/or modify it
 * under the terms of the license (GNU LGPL) which comes with this package. */

#include <stdlib.h>
#include <stdio.h>
#include "simgrid/simdag.h"
#include "xbt/log.h"
#include "xbt/ex.h"
#include <string.h>

XBT_LOG_NEW_DEFAULT_CATEGORY(test,
                             "Logging specific to this SimDag example");
```

```
typedef struct _WorkstationAttribute *WorkstationAttribute;
struct _WorkstationAttribute {
  /* Earliest time at which a workstation is ready to execute a task */
  double available_at;
  SD_task_t last_scheduled_task;
};

static void SD_workstation_allocate_attribute(SD_workstation_t workstation)
{
  void *data;
  data = calloc(1, sizeof(struct _WorkstationAttribute));
  SD_workstation_set_data(workstation, data);
}

static void SD_workstation_free_attribute(SD_workstation_t workstation)
{
  free(SD_workstation_get_data(workstation));
  SD_workstation_set_data(workstation, NULL);
}

static double SD_workstation_get_available_at(SD_workstation_t workstation)
{
  WorkstationAttribute attr =
      (WorkstationAttribute) SD_workstation_get_data(workstation);
  return attr->available_at;
}

static void SD_workstation_set_available_at(SD_workstation_t workstation,
                                            double time)
{
  WorkstationAttribute attr =
      (WorkstationAttribute) SD_workstation_get_data(workstation);
  attr->available_at = time;
  SD_workstation_set_data(workstation, attr);
}

static SD_task_t SD_workstation_get_last_scheduled_task( SD_workstation_t
         workstation){
  WorkstationAttribute attr =
      (WorkstationAttribute) SD_workstation_get_data(workstation);
  return attr->last_scheduled_task;
}

static void SD_workstation_set_last_scheduled_task(SD_workstation_t
         workstation,
    SD_task_t task){
  WorkstationAttribute attr =
      (WorkstationAttribute) SD_workstation_get_data(workstation);
  attr->last_scheduled_task=task;
  SD_workstation_set_data(workstation, attr);
}

static xbt_dynar_t get_ready_tasks(xbt_dynar_t dax)
{
  unsigned int i;
  xbt_dynar_t ready_tasks;
  SD_task_t task;

  ready_tasks = xbt_dynar_new(sizeof(SD_task_t), NULL);
  xbt_dynar_foreach(dax, i, task) {
    if (SD_task_get_kind(task) == SD_TASK_COMP_SEQ &&
        SD_task_get_state(task) == SD_SCHEDULABLE) {
      xbt_dynar_push(ready_tasks, &task);
    }
  }
  XBT_DEBUG("There are %lu ready tasks", xbt_dynar_length(ready_tasks));
```

```
  return ready_tasks;
}

static double finish_on_at(SD_task_t task, SD_workstation_t workstation)
{
  volatile double result;
  unsigned int i;
  double data_available = 0.;
  double redist_time = 0;
  double last_data_available;
  SD_task_t parent, grand_parent;
  xbt_dynar_t parents, grand_parents;

  SD_workstation_t *grand_parent_workstation_list;

  parents = SD_task_get_parents(task);

  if (!xbt_dynar_is_empty(parents)) {
    /* compute last_data_available */
    last_data_available = -1.0;
    xbt_dynar_foreach(parents, i, parent) {

      /* normal case */
      if (SD_task_get_kind(parent) == SD_TASK_COMM_E2E) {
        grand_parents = SD_task_get_parents(parent);

        if (xbt_dynar_length(grand_parents) > 1) {
          XBT_ERROR("Warning: transfer %s has 2 parents",
                  SD_task_get_name(parent));
        }
        xbt_dynar_get_cpy(grand_parents, 0, &grand_parent);

        grand_parent_workstation_list =
            SD_task_get_workstation_list(grand_parent);
        /* Estimate the redistribution time from this parent */
        redist_time =
            SD_route_get_communication_time(grand_parent_workstation_list
                                            [0], workstation,
                                            SD_task_get_amount(parent));
        data_available =
            SD_task_get_finish_time(grand_parent) + redist_time;

        xbt_dynar_free_container(&grand_parents);
      }

      /* no transfer, control dependency */
      if (SD_task_get_kind(parent) == SD_TASK_COMP_SEQ) {
        data_available = SD_task_get_finish_time(parent);
      }

      if (last_data_available < data_available)
        last_data_available = data_available;

    }

    xbt_dynar_free_container(&parents);

    result = MAX(SD_workstation_get_available_at(workstation),
                 last_data_available) +
        SD_workstation_get_computation_time(workstation,
                                            SD_task_get_amount(task));
  } else {
    xbt_dynar_free_container(&parents);

    result = SD_workstation_get_available_at(workstation) +
        SD_workstation_get_computation_time(workstation,
                                            SD_task_get_amount(task));
```

```
  }
  return result;
}

static SD_workstation_t SD_task_get_best_workstation(SD_task_t task)
{
  int i;
  double EFT, min_EFT = -1.0;
  const SD_workstation_t *workstations = SD_workstation_get_list();
  int nworkstations = SD_workstation_get_number();
  SD_workstation_t best_workstation;

  best_workstation = workstations[0];
  min_EFT = finish_on_at(task, workstations[0]);

  for (i = 1; i < nworkstations; i++) {
    EFT = finish_on_at(task, workstations[i]);
    XBT_DEBUG("%s finishes on %s at %f",
           SD_task_get_name(task),
           SD_workstation_get_name(workstations[i]), EFT);

    if (EFT < min_EFT) {
      min_EFT = EFT;
      best_workstation = workstations[i];
    }
  }

  return best_workstation;
}

static void output_xml(FILE * out, xbt_dynar_t dax)
{
  unsigned int i, j, k;
  int current_nworkstations;
  const int nworkstations = SD_workstation_get_number();
  const SD_workstation_t *workstations = SD_workstation_get_list();
  SD_task_t task;
  SD_workstation_t *list;

  fprintf(out, "<?xml version=\"1.0\"?>\n");
  fprintf(out, "<grid_schedule>\n");
  fprintf(out, "   <grid_info>\n");
  fprintf(out, "      <info name=\"nb_clusters\" value=\"1\"/>\n");
  fprintf(out, "         <clusters>\n");
  fprintf(out,
          "            <cluster id=\"1\" hosts=\"%d\" first_host=\"0\"/>\n",
          nworkstations);
  fprintf(out, "         </clusters>\n");
  fprintf(out, "      </grid_info>\n");
  fprintf(out, "   <node_infos>\n");

  xbt_dynar_foreach(dax, i, task) {
    fprintf(out, "      <node_statistics>\n");
    fprintf(out, "         <node_property name=\"id\" value=\"%s\"/>\n",
            SD_task_get_name(task));
    fprintf(out, "         <node_property name=\"type\" value=\"");
    if (SD_task_get_kind(task) == SD_TASK_COMP_SEQ)
      fprintf(out, "computation\"/>\n");
    if (SD_task_get_kind(task) == SD_TASK_COMM_E2E)
      fprintf(out, "transfer\"/>\n");

    fprintf(out,
            "         <node_property name=\"start_time\" value=\"%.3f\"/>\n",
            SD_task_get_start_time(task));
    fprintf(out,
            "         <node_property name=\"end_time\" value=\"%.3f\"/>\n",
            SD_task_get_finish_time(task));
```

```
    fprintf(out, "         <configuration>\n");

    current_nworkstations = SD_task_get_workstation_count(task);

    fprintf(out,
            "            <conf_property name=\"host_nb\" value=\"%d\"/>\n",
            current_nworkstations);

    fprintf(out, "            <host_lists>\n");
    list = SD_task_get_workstation_list(task);
    for (j = 0; j < current_nworkstations; j++) {
      for (k = 0; k < nworkstations; k++) {
        if (!strcmp(SD_workstation_get_name(workstations[k]),
                    SD_workstation_get_name(list[j]))) {
          fprintf(out, "               <hosts start=\"%u\" nb=\"1\"/>\n",
                  k);
          fprintf(out,
                  "            <conf_property name=\"cluster_id\"
         value=\"0\"/>\n");
          break;
        }
      }
    }
    fprintf(out, "            </host_lists>\n");
    fprintf(out, "         </configuration>\n");
    fprintf(out, "      </node_statistics>\n");
  }
  fprintf(out, "   </node_infos>\n");
  fprintf(out, "</grid_schedule>\n");
}

int main(int argc, char **argv)
{
  unsigned int cursor;
  double finish_time, min_finish_time = -1.0;
  SD_task_t task, selected_task = NULL, last_scheduled_task;
  xbt_dynar_t ready_tasks;
  SD_workstation_t workstation, selected_workstation = NULL;
  int total_nworkstations = 0;
  const SD_workstation_t *workstations = NULL;
  xbt_dynar_t dax, changed;
  FILE *out = NULL;

  /* initialization of SD */
  SD_init(&argc, argv);

  /* Check our arguments */
  if (argc < 3) {
    XBT_INFO("Usage: %s platform_file dax_file [jedule_file]", argv[0]);
    XBT_INFO
        ("example: %s simulacrum_7_hosts.xml Montage_25.xml Montage_25.jed",
         argv[0]);
    exit(1);
  }
  char *tracefilename;
  if (argc == 3) {
    char *last = strrchr(argv[2], '.');

    tracefilename = bprintf("%.*s.jed",
                            (int) (last ==
                                   NULL ? strlen(argv[2]) : last -
                                   argv[2]), argv[2]);
  } else {
    tracefilename = xbt_strdup(argv[3]);
  }

  /* creation of the environment */
```

```
SD_create_environment(argv[1]);

/*  Allocating the workstation attribute */
total_nworkstations = SD_workstation_get_number();
workstations = SD_workstation_get_list();

for (cursor = 0; cursor < total_nworkstations; cursor++)
  SD_workstation_allocate_attribute(workstations[cursor]);


/* load the DAX file */
dax = SD_daxload(argv[2]);

xbt_dynar_foreach(dax, cursor, task) {
  SD_task_watch(task, SD_DONE);
}

/* Schedule the root first */
xbt_dynar_get_cpy(dax, 0, &task);
workstation = SD_task_get_best_workstation(task);
SD_task_schedulel(task, 1, workstation);

while (!xbt_dynar_is_empty((changed = SD_simulate(-1.0)))) {
  /* Get the set of ready tasks */
  ready_tasks = get_ready_tasks(dax);
  if (xbt_dynar_is_empty(ready_tasks)) {
    xbt_dynar_free_container(&ready_tasks);
    xbt_dynar_free_container(&changed);
    /* there is no ready task, let advance the simulation */
    continue;
  }
  /* For each ready task:
   * get the workstation that minimizes the completion time.
   * select the task that has the minimum completion time on
   * its best workstation.
   */
  xbt_dynar_foreach(ready_tasks, cursor, task) {
    XBT_DEBUG("%s is ready", SD_task_get_name(task));
    workstation = SD_task_get_best_workstation(task);
    finish_time = finish_on_at(task, workstation);
    if (min_finish_time == -1. || finish_time < min_finish_time) {
      min_finish_time = finish_time;
      selected_task = task;
      selected_workstation = workstation;
    }
  }

  XBT_INFO("Schedule %s on %s", SD_task_get_name(selected_task),
       SD_workstation_get_name(selected_workstation));

  SD_task_schedulel(selected_task, 1, selected_workstation);

  /*
   * SimDag allows tasks to be executed concurrently when they can by
       default.
   * Yet schedulers take decisions assuming that tasks wait for resource
   * availability to start.
   * The solution (well crude hack is to keep track of the last task
       scheduled
   * on a workstation and add a special type of dependency if needed to
   * force the sequential execution meant by the scheduler.
   * If the last scheduled task is already done, has failed or is a
   * predecessor of the current task, no need for a new dependency
  */

  last_scheduled_task =
    SD_workstation_get_last_scheduled_task(selected_workstation);
```

```
    if (last_scheduled_task &&
  (SD_task_get_state(last_scheduled_task) != SD_DONE) &&
  (SD_task_get_state(last_scheduled_task) != SD_FAILED) &&
  !SD_task_dependency_exists(
     SD_workstation_get_last_scheduled_task(selected_workstation),
     selected_task))
      SD_task_dependency_add("resource", NULL,
           last_scheduled_task, selected_task);

    SD_workstation_set_last_scheduled_task(selected_workstation,
         selected_task);

    SD_workstation_set_available_at(selected_workstation, min_finish_time);

    xbt_dynar_free_container(&ready_tasks);
    /* reset the min_finish_time for the next set of ready tasks */
    min_finish_time = -1.;
    xbt_dynar_free_container(&changed);
  }

  XBT_INFO("Simulation Time: %f", SD_get_clock());
  XBT_INFO
      ("------------------- Produce the trace file--------------------------
          ");
  XBT_INFO("Producing the trace of the run into %s", tracefilename);
  out = fopen(tracefilename, "w");
  xbt_assert(out, "Cannot write to %s", tracefilename);
  free(tracefilename);

  output_xml(out, dax);

  fclose(out);

  xbt_dynar_free_container(&ready_tasks);
  xbt_dynar_free_container(&changed);

  xbt_dynar_foreach(dax, cursor, task) {
    SD_task_destroy(task);
  }
  xbt_dynar_free_container(&dax);

  for (cursor = 0; cursor < total_nworkstations; cursor++)
    SD_workstation_free_attribute(workstations[cursor]);

  /* exit */
  SD_exit();
  return 0;
}
```

Apéndice C

En este apéndice se documentan el resto de datos estadísticos relativos a los resultados de los experimentos asociados a la evaluación de la presente tesis.

Montage_25 sobre NFP_cluster_1-7

Alg.	Media	Mediana	Desviación
O0	98,184618	98,184618	0
A1	98,161208	98,150693	0,010957194
A2	98,166180	98,150863	0,009704338
A3	98,140962	98,150693	0,000739382
A4	98,140962	98,150693	0,000739382
A5	98,140962	98,150693	0,000739382
B1	557,614915	536,198031	207496,4381
B2	485,141783	536,198031	34206,84627
B3	549,677482	536,198031	207054,6429
B4	549,677482	536,198031	207054,6429
B5	549,677482	536,198031	207054,6429
C1	147,073398	146,569010	3376,625274
C2	98,708406	98,255131	3,852245625
C3	114,278858	111,346852	319,6629949
C4	142,662887	136,355270	3053,664569
C5	114,278858	111,346852	319,6629949
D1	289,975042	289,975042	0
D2	289,975042	289,975042	0
D3	289,975042	289,975042	0
D4	289,975042	289,975042	0
D5	289,975042	289,975042	0

Tabla C.1 Resultados del experimento E2 para el grafo Montage_25 sobre la plataforma NFP_cluster_1-7

Montage_25 sobre NFP_node_1-49

Alg.	Media	Mediana	Desviación
O0	52,552652	52,552652	0
A1	52,563494	52,550117	0,003578776
A2	52,587319	52,550117	0,015437503
A3	52,563494	52,550117	0,003578776
A4	52,563494	52,550117	0,003578776
A5	52,563494	52,550117	0,003578776
B1	469,736464	385,543274	293770,3524
B2	407,646526	385,543274	85901,29711
B3	469,736464	385,543274	293770,3524
B4	469,736464	385,543274	293770,3524
B5	469,736464	385,543274	293770,3524
C1	87,260321	87,743228	845,4322561
C2	52,750041	52,649953	0,110548379
C3	63,604137	66,751463	353,2524763
C4	87,321248	87,743228	852,8878005
C5	63,604137	66,751463	353,2524763
D1	579,055753	579,055753	0
D2	579,055753	579,055753	0
D3	579,055753	579,055753	0
D4	579,055753	579,055753	0
D5	579,055753	579,055753	0

Tabla C.2 Resultados del experimento E2 para el grafo Montage_25 sobre la plataforma NFP_node_1-49

Montage_25 sobre NFP_grid_1-1035

Alg.	Media	Mediana	Desviación
O0	3,270385	3,270385	0
A1	3,262131	3,270385	0,001362438
A2	3,250962	3,270385	0,007545369
A3	3,262131	3,270385	0,001362438
A4	3,262131	3,270385	0,001362438
A5	3,262131	3,270385	0,001362438
B1	51,332837	3,270385	46199,98508
B2	29,438928	3,270385	13695,85327
B3	51,332837	3,270385	46199,98508
B4	51,332837	3,270385	46199,98508
B5	143,347954	118,289282	12558,74105
C1	3,296042	3,270385	0,013165838
C2	3,250962	3,270385	0,007545369
C3	3,265508	3,270385	0,000475703
C4	3,296042	3,270385	0,013165838
C5	95,280626	118,289282	10587,96539
D1	118,289282	118,289282	0
D2	118,289282	118,289282	0
D3	118,289282	118,289282	0
D4	118,289282	118,289282	0
D5	118,289282	118,289282	0

Tabla C.3 *Resultados del experimento E2 para el grafo Montage_25 sobre la plataforma NFP_grid_1-1035*

DAG-corto sobre NFP_cluster_1-7

Alg.	Media	Mediana	Desviación
O0	2107,692721	2107,692721	0
A1	2107,669306	2107,658795	0,010956946
A2	2107,674283	2107,658972	0,009703358
A3	2107,649061	2107,658795	0,000739739
A4	2107,649061	2107,658795	0,000739739
A5	2107,649061	2107,658795	0,000739739
B1	7919,068536	7614,850043	41866754,35
B2	6889,615941	7614,850043	6901948,049
B3	7806,320384	7614,850043	41777612,9
B4	7806,320384	7614,850043	41777612,9
B5	7806,320384	7614,850043	41777612,9
C1	2436,969308	2354,203220	194241,3821
C2	2122,498526	2122,804594	379,3131189
C3	2213,092839	2215,616763	44591,66363
C4	2412,066048	2349,024896	131783,769
C5	2213,092839	2215,616763	44591,66363
D1	4117,348429	4117,348429	0
D2	4117,348429	4117,348429	0
D3	4117,348429	4117,348429	0
D4	4117,348429	4117,348429	0
D5	4117,348429	4117,348429	0

Tabla C.4 Resultados del experimento E2 para el grafo DAG-corto sobre la plataforma NFP_cluster_1-7

DAG-corto sobre NFP_node_1-49

Alg.	Media	Mediana	Desviación
O0	1562,226209	1562,226209	0
A1	1562,240408	1562,232067	0,003214221
A2	1562,267590	1562,232067	0,013880241
A3	1562,240408	1562,232067	0,003214221
A4	1562,240408	1562,232067	0,003214221
A5	1562,240408	1562,232067	0,003214221
B1	6670,789379	5474,857923	59274324,41
B2	5788,826012	5474,857923	17332386,75
B3	6670,789379	5474,857923	59274324,41
B4	6670,789379	5474,857923	59274324,41
B5	6670,789379	5474,857923	59274324,41
C1	1900,549296	1929,464853	71600,17245
C2	1568,528716	1565,056459	239,5371789
C3	1780,498524	1824,315979	40339,28478
C4	1900,766543	1929,464853	71941,57235
C5	1780,498524	1824,315979	40339,28478
D1	8223,627286	8223,627286	0
D2	8223,627286	8223,627286	0
D3	8223,627286	8223,627286	0
D4	8223,627286	8223,627286	0
D5	8223,627286	8223,627286	0

Tabla C.5 Resultados del experimento E2 para el grafo DAG-corto sobre la plataforma NFP_node_1-49

DAG-corto sobre NFP_grid_1-1035

Alg.	*Media*	*Mediana*	*Desviación*
O0	84,060619	84,060619	0
A1	84,052602	84,060619	0,001285574
A2	84,040975	84,060619	0,007717892
A3	84,052602	84,060619	0,001285574
A4	84,052602	84,060619	0,001285574
A5	84,052602	84,060619	0,001285574
B1	758,920992	84,060619	9108730,466
B2	447,926558	84,060619	2647968,425
B3	758,920992	84,060619	9108730,466
B4	758,920992	84,060619	9108730,466
B5	2034,567789	1678,619115	2533989,17
C1	86,540272	84,060619	122,9735998
C2	84,016392	84,060619	0,039120904
C3	84,489729	84,060619	3,682700976
C4	86,540272	84,060619	122,9735998
C5	1360,136525	1678,619115	2028623,198
D1	1678,619115	1678,619115	0
D2	1678,619115	1678,619115	0
D3	1678,619115	1678,619115	0
D4	1678,619115	1678,619115	0
D5	1678,619115	1678,619115	0

Tabla C.6 Resultados del experimento E2 para el grafo DAG-corto sobre la plataforma NFP_grid_1-1035

DAG-medio sobre NFP_cluster_1-7

Alg.	Media	Mediana	Desviación
O0	9774,551252	9774,551252	0
A1	9774,527842	9774,517327	0,010957194
A2	9774,532814	9774,517497	0,009704338
A3	9774,507596	9774,517327	0,000739382
A4	9774,507596	9774,517327	0,000739382
A5	9774,507596	9774,517327	0,000739382
B1	56284,662586	55750,438184	2069230846
B2	46154,078060	46074,814207	231820542,7
B3	55292,483383	53766,079778	2068275235
B4	55292,483383	53766,079778	2068275235
B5	55292,483383	53766,079778	2068275235
C1	14450,708094	13511,815164	34991902,91
C2	9833,255485	9808,988075	28083,65538
C3	11294,049899	11089,994800	3766704,114
C4	14203,373742	13511,815164	30252832,99
C5	11294,049899	11089,994800	3766704,114
D1	28985,253147	28985,253147	0
D2	28985,253147	28985,253147	0
D3	28985,253147	28985,253147	0
D4	28985,253147	28985,253147	0
D5	28985,253147	28985,253147	0

Tabla C.7 Resultados del experimento E2 para el grafo DAG-medio sobre la plataforma NFP_cluster_1-7

DAG-medio sobre NFP_node_1-49

Alg.	Media	Mediana	Desviación
O0	5254,102495	5254,102495	0
A1	5254,113338	5254,099961	0,003578776
A2	5254,148557	5254,133403	0,012840955
A3	5254,113338	5254,099961	0,003578776
A4	5254,113338	5254,099961	0,003578776
A5	5254,113338	5254,099961	0,003578776
B1	46961,395348	38542,076342	2937703527
B2	41806,884595	38542,076342	433826748,2
B3	46961,395348	38542,076342	2937703527
B4	46961,395348	38542,076342	2937703527
B5	46961,395348	38542,076342	2937703527
C1	8713,404196	8763,060119	8437749,73
C2	5265,159336	5265,135098	479,086541
C3	6615,595143	6828,989955	2065919,736
C4	8719,504826	8763,060119	8512563,663
C5	6615,595143	6828,989955	2065919,736
D1	57893,324296	57893,324296	2,64698E-22
D2	57893,324296	57893,324296	2,64698E-22
D3	57893,324296	57893,324296	2,64698E-22
D4	57893,324296	57893,324296	2,64698E-22
D5	57893,324296	57893,324296	2,64698E-22

Tabla C.8 Resultados del experimento E2 para el grafo DAG-medio sobre la plataforma NFP_node_1-49

DAG-medio sobre NFP_grid_1-1035

Alg.	Media	Mediana	Desviación
O0	274,147658	274,147658	0
A1	274,139641	274,147658	0,001285574
A2	274,128014	274,147658	0,007717892
A3	274,139641	274,147658	0,001285574
A4	274,139641	274,147658	0,001285574
A5	274,139641	274,147658	0,001285574
B1	5088,520784	274,147658	463563771,9
B2	2899,129961	274,147658	137810641,8
B3	5088,520784	274,147658	463563771,9
B4	5088,520784	274,147658	463563771,9
B5	14322,544398	11816,677176	125587410,7
C1	276,806002	274,147658	141,3358564
C2	274,106570	274,147658	0,033764804
C3	274,120190	274,147658	0,015089601
C4	276,806002	274,147658	141,3358564
C5	9508,143805	11816,677176	106586526,5
D1	11816,677176	11816,677176	0
D2	11816,677176	11816,677176	0
D3	11816,677176	11816,677176	0
D4	11816,677176	11816,677176	0
D5	11816,677176	11816,677176	0

Tabla C.9 Resultados del experimento E2 para el grafo DAG-medio sobre la plataforma NFP_grid_1-1035

DAG-largo sobre NFP_cluster_1-7

Alg.	Media	Mediana	Desviación
O0	97741,520654	97741,520654	0
A1	97741,474417	97741,474417	0,000303171
A2	97741,486898	97741,486898	0
A3	97741,474417	97741,474417	0,000303171
A4	97741,474417	97741,474417	0,000303171
A5	97741,474417	97741,474417	0,000303171
B1	557483,306422	536074,406991	2,06578E+11
B2	486913,005930	536074,406991	15456378249
B3	552923,720095	537659,684052	2,06828E+11
B4	552923,720095	537659,684052	2,06828E+11
B5	552923,720095	537659,684052	2,06828E+11
C1	159151,111521	159151,111521	1156247724
C2	98944,210988	98944,210988	1472442,669
C3	109869,629168	110895,885238	19849209,28
C4	158892,195292	158892,195292	1131480058
C5	109869,629168	110895,885238	19849209,28
D1	289851,417742	289851,417742	0
D2	289851,417742	289851,417742	0
D3	289851,417742	289851,417742	0
D4	289851,417742	289851,417742	0
D5	289851,417742	289851,417742	0

Tabla C.10 Resultados del experimento E2 para el grafo DAG-largo sobre la plataforma NFP_cluster_1-7

DAG-largo sobre NFP_node_1-49

Alg.	Media	Mediana	Desviación
O0	52540,919252	52540,919252	0
A1	52540,916718	52540,916718	0
A2			
A3	52540,916718	52540,916718	0
A4	52540,916718	52540,916718	0
A5	52540,916718	52540,916718	0
B1	293364,488253	247336,907536	12711229120
B2	418067,732218	385419,649688	43382674815
B3	293364,488253	247336,907536	12711229120
B4	293364,488253	247336,907536	12711229120
B5	293364,488253	247336,907536	12711229120
C1	73001,118610	73001,118610	0
C2			
C3			
C4	73001,118610	73001,118610	0
C5			
D1	578932,129225	578932,129225	0
D2	578932,129225	578932,129225	0
D3	578932,129225	578932,129225	0
D4	578932,129225	578932,129225	0
D5	578932,129225	578932,129225	0

Tabla C.11 Resultados del experimento E2 para el grafo DAG-largo sobre la plataforma NFP_node_1-49

DAG-largo sobre NFP_grid_1-1035

Alg.	Media	Mediana	Desviación
O0	2736,648708	2736,648708	0
A1	2736,640691	2736,648708	0,001285574
A2	2736,629064	2736,648708	0,007717735
A3	2736,640691	2736,648708	0,001285574
A4	2736,640691	2736,648708	0,001285574
A5	2736,640691	2736,648708	0,001285574
B1	50881,122795	2736,648708	46357807702
B2	28987,214566	2736,648708	13781844158
B3	50881,122795	2736,648708	46357807702
B4	50881,122795	2736,648708	46357807702
B5	143224,330251	118165,658028	12558741071
C1	2763,445939	2736,648708	14361,83179
C2	2736,607620	2736,648708	0,033764804
C3	2736,621240	2736,648708	0,015089381
C4	2763,445939	2736,648708	14361,83179
C5	95079,828696	118165,658028	10659110319
D1	118165,658028	118165,658028	0
D2	118165,658028	118165,658028	0
D3	118165,658028	118165,658028	0
D4	118165,658028	118165,658028	0
D5	118165,658028	118165,658028	0

Tabla C.12 Resultados del experimento E2 para el grafo DAG-largo sobre la plataforma NFP_grid_1-1035