UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Programa de Doctorado en Tecnologías de la Información y la Comunicación

Servicios de minería de datos en Cloud Computing

Tesis doctoral

Manuel Jesús Parra Royón

Dirigida por:

D. José Manuel Benítez Sánchez

Granada, Junio de 2019

Editor: Universidad de Granada. Tesis Doctorales
Autor: Manuel Jesús Parra Royón
ISBN: 978-84-1306-273-0
URI: http://hdl.handle.net/10481/56592

UNIVERSIDAD
DE GRANADA

Servicios de minería de datos en Cloud Computing

Tesis Doctoral

presentada para obtener el

GRADO DE DOCTOR

en el

PROGRAMA DE DOCTORADO DE TECNOLOGÍAS
DE LA INFORMACIÓN Y LA COMUNICACIÓN

UNIVERSIDAD DE GRANADA

por

Manuel Jesús Parra Royón

Director:

José Manuel Benítez Sánchez

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD
DE GRANADA

Data Mining Services in Cloud Computing

DOCTORAL DISSERTATION

presented to obtain the

DOCTOR OF PHILOSOPHY DEGREE

in the

INFORMATION AND COMMUNICATION
TECHNOLOGY PROGRAM

UNIVERSITY OF GRANADA

by

Manuel Jesús Parra Royón

Ph.D. Advisor:

José Manuel Benítez Sánchez

Department of Computer Science and Artificial Intelligence

Esta tesis doctoral ha sido desarrollada con la financiación de la beca de «Excelencia» asociada al proyecto de la Junta de Andalucía "Minería de Datos en Cloud Computing" *P12-TIC-2958*, también desde fondos asociados al proyecto *TIN 2016-81113-R* del Ministerio de Economía, Industria y Competitividad del Gobierno de España. De forma adicional ha recibido soporte financiero desde el proyecto europeo *LIFE+: Environment Policy and Governance LIFE12 ENV/ES/000686*.

En relación al proyecto "Minería de Datos en Cloud Computing" y dentro del contexto de este trabajo de tesis, se hecho creado un sitio web[1] donde está incluido todo el material complementario.

[1]DMServices: `https://dicits.ugr.es/linkeddata/dmservices/`

Resumen

Esta tesis aborda el problema de la definición y descripción de servicios de minería de datos en Cloud Computing. Cada proveedor de servicios de minería de datos en Cloud Computing tiene su particular forma de definir servicios, la cual es incompatible con los demás proveedores, de modo que obliga a rediseñar las aplicaciones o las herramientas que utilicen estos servicios, cuando queremos por ejemplo, moverlas a otro proveedor de Cloud Computing.

Actualmente no existe una estandarización consolidada para la definición de servicios de Cloud Computing entre proveedores que permita describir la experimentación de flujos de trabajo con minería de datos de un modo eficaz y eficiente, y que además considere todos los aspectos clave que un proveedor necesita gestionar. Para resolver el problema de la descripción de servicios de este tipo se ha diseñado una propuesta basada en tecnología semántica y *Linked Data* llamada «dmcc-schema».

Con esta propuesta de servicios semánticos se persiguen dos objetivos básicos: a) definir y describir el modelado de trabajo con minería de datos como servicio en Cloud Computing y b) definir y describir el modelado de la gestión de los servicios en Cloud Computing para el gobierno de los mismos por parte de los proveedores.

Con la finalidad de llevar a cabo estos objetivos por un lado se ha diseñado un vocabulario semántico completo que aborda tanto la definición de la experimentación con minería de datos como la gestión del servicio en los proveedores Cloud Computing, permitiendo una definición directa y eficiente, además de añadir capacidades de descubrimiento, integrabilidad y portabilidad, solucionando el problema del anclaje al proveedor. Para completar la definición, de forma adicional, se ha diseñado una propuesta de arquitectura de despliegue de servicios de minería de datos llamada OC^{2}DM, que permite, a partir de las definiciones de los servicios creados con el esquema, construir

y desplegar el servicio de forma que sea completamente funcional. Por último se ha desarrollado una herramienta de intermediación («Brokering») de servicios Cloud Computing, para poder capturar todas las características de los servicios de minería de datos, habilitando un único punto de gestión para la intermediación de servicios de diferentes proveedores. Con el propósito de confirmar la aplicabilidad y la validez del trabajo en entornos reales, tres casos de uso han sido desarrollados en diferentes áreas, tales como la definición de servicios en Física de Alta Energías, el estudio de las series temporales o los sistemas difusos como servicios en Cloud Computing.

Summary

This thesis addresses the problem of the definition and description of data mining services in Cloud Computing. Each provider of data mining services in Cloud Computing has its own particular way of defining services, which is incompatible with other providers, so it forces to redesign the applications or tools that use these services, when we want, for example, to move them to another provider of Cloud Computing.

Currently, there is no consolidated standardization for the definition of Cloud Computing services between providers that would describe the experimentation of workflows with data mining in an effective and efficient way, and that would also consider all the key aspects that a provider needs to manage. To solve the problem of describing services of this type, a proposal based on semantic technology and *Linked Data* called «dmcc-schema» has been designed.

With this semantic services proposal, two basic objectives are aimed: a) defining and describing the modeling of the work with data mining as a service in Cloud Computing and b) defining and describing the modeling of the management of the services in Cloud Computing for the governance of the same by the providers.

In order to carry out these objectives on the one hand, a complete semantic vocabulary has been designed that addresses both the definition of experimentation with data mining and service management in Cloud Computing providers, allowing a direct and efficient definition, in addition to adding discovery capabilities, integrability and portability, solving the provider lock-in problem. To complete the definition, in an additional way, a proposal has been designed for the deployment architecture of data mining services called OC^{2}DM, which allows, from the definitions of the services created with the scheme, to build and deploy the service so that it is fully functional. Finally, a Cloud Computing services intermediation tool («Brokering») has been deve-

loped in order to capture all the features of the data mining services, enabling a single management point for the intermediation of services from different providers.

To confirm the applicability of the work in real environments, three use cases have been developed in different areas, such as the definition of services in High Energy Physics, the study of time series or fuzzy systems as services in Cloud Computing.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Desde hace ya mas de 10 años el modelo tradicional de consumo y trabajo con aplicaciones, herramientas y plataformas «on-premise» ha ido evolucionando de equipos locales a un nuevo modelo de servicios basado en Cloud Computing. El modelo Cloud Computing se ha introducido casi sin darnos cuenta en nuestro día a día, así como en nuestra relación con las Tecnologías de la Información, sin prácticamente ninguna fricción. Usamos teléfonos móviles, tabletas, televisores y muchos otros dispositivos que están conectados a Internet, consumiendo diversos servicios de Cloud Computing de una manera totalmente transparente, habilitando el consumo, el uso y el acceso mucho más uniforme de servicios tales como aplicaciones para el almacenamiento de fotografías, herramientas ofimáticas, plataformas de desarrollo o servicios de computación, infraestructura o redes bajo demanda. Cloud Computing se basa en el modelo de «utility computing» o servicios bajo demanda, de tal modo que los usuarios de dichos servicios pagan por el uso de los recursos de computación del proveedor, con lo que el modelo supone una reducción de los costes en infraestructura para los usuarios.

Son muchos los proveedores de Cloud Computing que recientemente están añadiendo un nuevo conjunto de servicios a su catálogo. Estos nuevos servicios están relacionados con la minería de datos y el aprendizaje automático. Tradicionalmente el procesamiento de datos en áreas como Ciencia de Datos o la Inteligencia Artificial se realizaba desde aplicaciones o plataformas de computación relegadas a entornos «on-premise» tales como equipos indivi-

duales o plataformas de computación distribuida. Desde hace relativamente pocos años esta tendencia ha ido cambiando, desplazando este tipo de herramientas y plataformas de computación a entornos basados en Cloud Computing ofreciendo servicios directamente consumibles. Dado el potencial de Cloud Computing en aspectos tales como su aparentemente infinita escalabilidad de recursos, el rendimiento, la flexibilidad y la rápida integración, el modelo de Cloud Computing puede considerarse el entorno natural para el procesamiento datos a gran escala.

Gran parte de los servicios ofrecidos por los proveedores carecen de una definición y descripción homogénea que permita la integración de servicios desde otros proveedores y que, por tanto, agrava el problema del anclaje al proveedor, suponiendo la imposibilidad de la migración de aplicaciones y servicios desarrollados en diferentes proveedores. Este problema, de igual modo, también afecta a la definición de servicios de minería de datos en Cloud Computing, no existiendo una propuesta de estandarización contundente para su modelado. Cada uno de los proveedores tiene su descripción de servicios de minería de datos que, en gran medida, es incompatible con el resto de los demás proveedores. El problema de la definición de servicios ha sido abordado desde diferentes perspectivas, todas ellas sobre la base de propuestas de descripción sintáctica y semántica, siendo esta última una de las opciones mejor consideradas para solucionar la definición de servicios.

Con el objetivo de abordar el problema de la definición de servicios, el trabajo desarrollado en esta tesis doctoral se centra en dos ejes fundamentales, por un lado realizar una propuesta de modelado para la descripción y definición de servicios de minería de datos en Cloud Computing (incluyendo algoritmos, funciones, parámetros, flujos de datos, entradas, salidas, modelos, etc.), y por otro lado diseñar el modelado para la gestión del servicio de minería de datos ofrecido por el proveedor (autenticación, interfaces, instancias, regiones, acuerdos del nivel servicio, etc.), ambos desde la perspectiva del diseño de servicios basados en tecnología semántica. Para desarrollar la propuesta se ha desarrollado un vocabulario y esquema llamado `dmcc-schema` con el que se capturan todos los elementos clave para el modelado de servicios de minería de datos en Cloud Computing, integrando los aspectos de la experimentación y de la gestión del servicio por parte de los proveedores.

No sólo es necesario tener una definición y descripción de servicios de minería de datos para entornos de Cloud Computing, también sería útil implementar una arquitectura de despliegue que permita dar funcionalidad a estos servicios, así como ofrecer la capacidad de desplegarlos sobre infraestructuras de computación de cualquier tipo. Para ello, en el contexto del trabajo de te-

sis se ha diseñado una plataforma llamada OC^{2}DM (*Open Cloud Computing Data Mining*) que persigue esos objetivos y da soporte funcional a las definiciones de servicios desarrollados con `dmcc-schema`. De forma adicional, para completar la eficacia del modelado semántico de servicios, se implementará un *Broker* para minería de datos en Cloud llamado «BrokerMD» con el objetivo de permitir la consulta, selección e intermediación de la experimentación de flujos de trabajo desde distintos proveedores.

1.2. Objetivos

De este modo, los objetivos de este trabajo de tesis son los siguientes:

1. Diseñar una propuesta de modelado para la definición y descripción de servicios de minería de datos para entornos de Cloud Computing, que incluya tanto el flujo de trabajo con el procesamiento de datos, como los demás aspectos relacionados con la gestión del servicio por parte de los proveedores y la interacción con los usuarios consumidores.
2. Implementar una arquitectura de despliegue de servicios de minería de datos en Cloud, que tenga capacidad para transformar las definiciones de los servicios de la propuesta semántica del primer objetivo, en servicios funcionales implementados como algoritmos siguiendo el modelo de Cloud Computing.
3. Desarrollar una herramienta de intermediación (*Brokering*) de servicios de minería de datos en Cloud Computing, que permita agregar, seleccionar, componer e intermediar servicios de este tipo desde diferentes proveedores, por ejemplo, permitiendo una optimización en la composición de flujos de trabajo en estos entornos Cloud.

La validación de estos tres objetivos se realizará a través de tres casos de uso que pondrán en valor todo el trabajo desarrollado y su aplicación práctica en diferentes áreas del conocimiento.

1.3. Estructura

Esta memoria se estructura según se detalla a continuación. En el capítulo 2, se realiza un estudio del contexto de la tesis, donde se tratan todos

los aspectos relacionados con Cloud Computing, arquitectura de servicios o modelado semántico entre otros. La semántica del modelado de servicios de minería de datos para Cloud Computing es descrita en el capítulo 3, donde se hace un estudio detallado de todos los componentes del esquema propuesto para solucionar el problema de la homogeneización de servicios de minería de datos entre proveedores. El capítulo 4 describe el diseño y la implementación de la arquitectura de despliegue desarrollada para dar soporte funcional a los servicios definidos en el esquema `dmcc-schema`. Posteriormente, en el capítulo 5 se realiza la implementación de una herramienta de *Brokering* en Cloud llamada «BrokerMD» para la consulta, selección e intermediación de servicios de minería de datos en Cloud utilizando como base la tecnología semántica desplegada en los capítulos anteriores.

Posteriormente, se ha trabajado sobre tres casos de uso reales donde se han implantado y desplegado los resultados de la investigación realizada. Gracias a estos casos de uso se puede confirmar la efectividad y la eficacia del trabajo desempeñado para la definición de servicios de minería de datos en Cloud. De este modo en el capitulo 6, que se desarrolla en el contexto del CERN, se realiza el modelado de servicios de computación para Física de Altas Energías basado en el esquema propuesto `dmcc-schema` para el procesamiento de datos en la etapa de monitorización de la calidad de los datos. En el capítulo 7, se realiza el modelado de servicios para la experimentación con flujos de trabajo de series temporales en Cloud Computing. En el capítulo 8 se implementa una propuesta de modelado semántico para sistemas difusos como servicio (*Fuzzy Systems-as-a-service – FSaaS*). Para finalizar, en el capítulo 9 se exponen las conclusiones generales del trabajo de tesis, así como las líneas futuras de investigación.

Capítulo 2

Contexto de general del trabajo

Motivación

El trabajo de investigación realizado durante el desarrollo de esta Tesis, ha implicado el estudio de múltiples tecnologías, herramientas y plataformas. Conviene introducir una serie de conceptos importantes, relacionados con esos elementos ya que permitirán al lector conocer de un modo más amplio todos los aspectos que se tratan a lo largo de los siguientes capítulos.

2.1. Cloud Computing

El uso tradicional de computadoras y aplicaciones de escritorio con el que habitualmente trabajamos en entornos empresariales o personales, supone una utilización de los recursos de computación que oscilan de media entre aproximadamente el 10 % de procesamiento por parte de la CPU, el *60 %* el uso de memoria (RAM y almacenamiento masivo) y el resto se dedica a la gestión de la conexión de datos y redes de comunicación. Está claro que estos niveles de uso corresponden a un uso moderado/bajo de los recursos de un computador y no se corresponden con el uso general de un computador personal donde el trabajo con un procesador de textos, o aplicaciones de mensajería electrónica suponen un coste computacional excepcionalmente más bajo que los porcentajes indicados.

Esto nos da una idea sobre la cuestión relativa a que hemos pagado el

total del coste de computación cuando solamente usamos una parte reducida de esta capacidad de cómputo y sobre todo de recursos e infraestructura.

Si nos centramos en los entornos empresariales, ocurre algo similar; las compañías y organizaciones disponen de una infraestructura que puede oscilar entre las decenas hasta miles de computadoras que los empleados utilizan para realizar las tareas relacionadas con su trabajo. Este fondo de recursos de computación formado por equipos de escritorio y grupos de servidores «on-premise» están infrautilizados, puesto que las tareas que se desarrollan sobre los mismos no suponen un excesivo coste computacional que requiera utilizar el *100 %* el potencial disponible. Además, esto claramente indica que al igual que en otros entornos más sencillos, a nivel empresarial o institucional, supone un uso poco eficaz de recursos mucho más importante.

	Computación tradicional	**Cloud Computing**
Consumo	Dedicado	Compartido
Facilidad uso	Hardware	Auto-servicio
Escalabilidad	Manual	Escalado bajo demanda
Disponibilidad	Manual	Recuperación automática
Aprovisionamiento	Semanas	Segundos-Minutos
Costes	Pago completo	Pago por uso

Tabla 2.1: Diferencias generales entre la computación tradicional y el modelo de Cloud Computing.

Trasladando este problema a entornos de computación tales como centros de datos o infraestructuras de cómputo de tipo *clúster*, por ejemplo, se observa que gran parte de las compañías e instituciones poseen equipos de cómputo completos dedicados a tareas fijas generales, tales como servidores web, servidores de almacenamiento o servicios de autenticación, entre muchos otros. Estos equipos son utilizados de manera similar, con tasas de uso mínimas que hacen que la inversión realizada en ellos no repercuta en el rendimiento de computación general. Por tanto, teniendo en cuesta estos factores, nos podemos hacer la siguiente pregunta, *¿por qué no se realiza una mejor gestión de los recursos de cómputo que mejore la ocupación más eficiente de los mismos?*. En este tipo de entornos se pueden aprovechar las capacidades de cómputo e infraestructura que no están siendo usadas para ofrecer o destinar esos recursos a otras cosas.

De este modo, la idea que supone solamente pagar por los recursos de computación que se usen en cada momento supone una ventaja a varios ni-

veles, tales como la inversión en infraestructuras completas o los costes de mantenimiento, entre otros. Este concepto se denomina *pago-por-uso* o también se define con el nombre de modelo «utility», donde servicios, por ejemplo, habituales en nuestro hogar como la energía o el agua son consumidos bajo demanda y se facturan sólo por lo que se consume. Este modelo es el mismo tras el cual se rige el concepto de Cloud Computing.

2.1.1. El concepto de Cloud Computing

Actualmente estamos inmersos en la era de las Tecnologías de la Información (TI), donde nuestra vida digital se ha solapado a nuestra vida real; Internet, los dispositivos móviles y los datos están presentes en todos los ámbitos de nuestra vida cotidiana.

No sólo dentro de nuestro ámbito personal el desarrollo de las TI ha sido adoptado de forma transparente, también en los entornos empresariales e institucionales, desde hace ya más de 10 años se está produciendo un movimiento muy importante en la integración de las TI dentro de estos entornos.

La entrega de los recursos informáticos a través una red de comunicaciones tiene sus raíces en los años sesenta. El concepto científico de computación en la nube se le puede atribuir a John McCarthy, quien propuso la idea de la computación como un servicio público, de forma similar a las empresas de servicios que se remontan a los años sesenta. A partir de los años setenta, la computación en la nube ha ido evolucionando. La Web 2.0 y la irrupción de la "web semántica" (ver sección 2.3.4) es la evolución más reciente. Uno de los primeros pasos para la integración de software como servicio en Cloud Computing fue la llegada de `salesforce.com` en 1999, que fue la empresa pionera en el concepto de la entrega de aplicaciones de tipo «Enterprise Resource Planning» (ERP) a través de una página web. El siguiente actor destacado fue *Amazon Web Services* en 2002, que comienza a proveer de un conjunto de servicios basados en la nube, incluyendo almacenamiento, computación e incluso herramientas de Inteligencia Artificial. Posteriormente en 2006, *Amazon* lanzó su *Elastic Compute Cloud* (EC2) como un servicio que permite a empresas y usuarios utilizar servicios de computación en los que se ejecuten sus aplicaciones informáticas.

Hoy en día tanto para el entorno personal como para el profesional la tecnología se ha convertido en ubicua e impregna prácticamente todo nuestro entorno; tanto es así que el concepto de Cloud Computing está tan consolidado en nuestro día a día que no somos conscientes de lo que simboliza o

las implicaciones que tiene a todos los niveles. Estamos simplemente acostumbrados a que con sólo tocar una botón en una aplicación desde nuestro terminal móvil, un documento, una fotografía o unos datos *se suban a la nube*.

Los usuarios consumidores de Cloud Computing ven la punta del iceberg de todo lo que supone el modelo de Cloud Computing; en general no conocen bien el alcance de mismo, ni del cambio de paradigma que está suponiendo nuestra forma de interactuar con las TI en la actualidad.

El cambio de modelo ofrecido por Cloud Computing, permite aumentar el número de servicios basados en Internet. Esto genera beneficios tanto para los proveedores que pueden ofrecer de forma más rápida y eficiente, un mayor número de servicios, como para los usuarios que tienen la posibilidad de acceder a ellos, disfrutando de la "transparencia" e inmediatez del sistema y de un modelo de pago por consumo. Este modelo de pago por uso es fundamental y es la base del modelo de «utility», gracias al cual sólo se paga por lo que se usa y nada más, permitiendo prescindir del servicio en Cloud Computing cuando ya no sea necesario y por tanto gestionar de una forma eficiente los gastos en infraestructura o software, entre otros.

La definición del concepto de Cloud Computing más completa y ampliamente aceptada es la proporcionada por el National Institute of Standars and Technology (NIST)[1]:

> *"La computación en nube es un modelo que permite el acceso ubicuo, conveniente y bajo demanda a un conjunto compartido de recursos informáticos de cómputo configurables (por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios) que pueden aprovisionarse y liberarse rápidamente con un mínimo esfuerzo de gestión o interacción con el proveedor de servicios. Este modelo de nube promueve la disponibilidad y se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de implementación".*

En esta definición de Cloud Computing se describen claramente tres aspectos clave relativos al modelo en cuestión. Cloud Computing, no sólo se refiere a aspectos relacionados con la perspectiva tecnológica y de infraestructura (recursos de cómputo), también se puede ver desde una perspectiva conceptual, esto es, la de los servicios de Cloud Computing.

[1] www.nist.gov

En este mismo documento del NIST donde se define el concepto, también se añaden varias de las características fundamentales que debe poseer Cloud Computing y que son:

1. **Servicios bajo demanda.** Un consumidor de servicios puede aprovisionar recursos de cómputo sin intervención de un humano para cada servicio-proveedor existente.
2. **Acceso a través de banda ancha.** La red debe tener suficiente ancho de banda para soportar el uso de Cloud Computing.
3. **Conjunto de recursos.** El proveedor proporciona al usuario consumidor una serie de recursos de cómputo que se encuentran distribuidos en la infraestructura del proveedor, usando un modelo llamado «multi-tenant». Este modelo consiste en asignar y re-asignar recursos físicos y virtuales de forma dinámica de acuerdo con la demanda del usuario consumidor.
4. **Elasticidad rápida.** El proveedor debe soportar el uso, aprovisionamiento y finalización de recursos de una forma elástica, permitiendo que los recursos se asignen de forma dinámica atendiendo a criterios de demanda y se liberen cuando esta demanda lo requiera.
5. **Medición del servicio.** Los sistemas Cloud Computing controlan y optimizan automáticamente el uso de recursos aprovechando las capacidades de medición para cada uno de los servicios que están incorporados. Los recursos pueden ser monitorizados, auditados y controlados de modo que tanto el consumidor como el proveedor de los mismos puedan conocer los datos del servicio, y por tanto los costes.

Cloud Computing por tanto puede ser considerado como un nuevo modelo de prestación de servicios de negocio y tecnología, que da la oportunidad al usuario consumidor de acceder a un catálogo de servicios estandarizados y responder con ellos a las necesidades de su negocio, de forma flexible y adaptativa, que además, en caso de una demanda de recursos no previsibles o de picos de trabajo se realiza el pago o la tarificación únicamente por el consumo efectuado.

2.1.2. Modelo de servicios: niveles de abstracción

En 2002 el NIST estableció una serie de criterios para la organización, estructura y composición del modelo de Cloud Computing. En esta prime-

ra aproximación se definió un modelo basado en tres capas de abstracción de servicios que constituyen la base de la nube que conocemos hoy: Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS). Recientemente el modelo de capas de abstracción ha ido ampliándose con nuevos modelos de servicios, tal y como se describe con algunos ejemplos en la sección 2.1.2.4. La base desarrollada al inicio del Cloud Computing, sigue actualmente siendo la referencia en cuando al modelo de servicio y la abstracción. En la Figura 2.1 se pueden ver las capas de abstracción en el modelo de Cloud Computing.

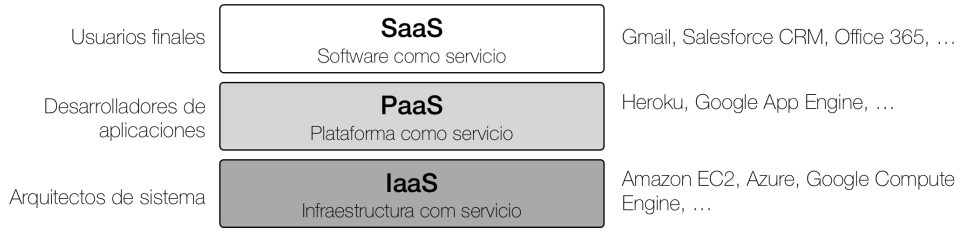

Figura 2.1: Niveles de abstracción del modelo de Cloud Computing, usuarios objetivo (a la izquierda) y herramientas disponibles (derecha).

2.1.2.1. Infrastructure as a Service

Infraestructura como Servicio (IaaS) es el paradigma de servicio de Cloud Computing de más bajo nivel y posiblemente uno de los más destacados. Con IaaS, los recursos de hardware pre-configurados son proporcionados a los usuarios a través de una interfaz virtual. A diferencia de PaaS y SaaS, IaaS no incluye aplicaciones ni siquiera un sistema operativo (la implementación de todo eso queda a criterio del cliente consumidor), simplemente permite el acceso a la infraestructura necesaria para alimentar o dar soporte a ese software como base de la pila.

IaaS puede proporcionar almacenamiento adicional para copias de seguridad de datos corporativos, ancho de banda de red para el servidor de un sitio web de una empresa, o incluso puede permitir el acceso a la computación de altas prestaciones que antes sólo era accesible para corporaciones.

Según el paradigma de Cloud Computing para el modelo de servicio proporcionado por IaaS, los proveedores de servicios alojan la infraestructura como servidores, unidades de almacenamiento, hardware de red, capa de virtualización o hipervisores para computación, entre muchos otros. Este modelo

rompe completamente el esquema de negocio heredado de antaño donde se invierte en infraestructura «on-premise» y equipamiento informático de centros de datos. Los modernos proveedores de servicios de IaaS también ofrecen otros servicios de gestión de la infraestructura de Cloud Computing. Estos servicios incluyen la monitorización del rendimiento del servicio, la facturación detallada de los servicios al cliente, el acceso a los registros, la seguridad digital, el balanceado de carga de trabajo, las copias de seguridad, la replicación, la recuperación, etc.

El paradigma de infraestructura como servicio ofrece un modelo de computación rentable para usuarios y clientes consumidores. Los clientes de IaaS pagan a los proveedores de servicios por el uso de la infraestructura utilizada, pero también pueden negociar tarifas por hora, semana, mes o incluso hacer una reserva de infraestructura por años. El modelo de pago por uso elimina importantes inversiones de capital y los costos iniciales que son soportados por las entidades cuando despliegan hardware y software en sus instalaciones.

Algunos de los servicios más populares de IaaS son *Amazon EC2*[2], *IBM SoftLayer*[3] y *Google Compute Engine* (GCE)[4]. Los proveedores de IaaS comparten las siguientes características principales:

1. En lugar de comprar el hardware directamente, los usuarios pagan por IaaS bajo demanda.
2. La infraestructura es escalable en función de las necesidades (procesamiento, almacenamiento y comunicaciones).
3. Permite ahorrar a las empresas los costes de compra y mantenimiento de su propio hardware.
4. Debido a que los datos están en la nube, no existe un único punto de fallo.

2.1.2.2. Platform as a Service

Gartner[5] define PaaS como un modelo de servicio en Cloud Computing que se utiliza para poner disponible una plataforma a los usuarios desde la cual puedan desarrollar, inicializar y gestionar aplicaciones. Las ofertas de

[2] `https://aws.amazon.com/es/ec2/`
[3] `www.ibm.com/Cloud/IaaS`
[4] `https://cloud.google.com/compute/`
[5] `https://www.gartner.com/it-glossary/platform-as-a-service-paas/`

PaaS suelen incluir una base de operaciones y un conjunto de aplicaciones y herramientas de desarrollo. PaaS elimina la necesidad de que las organizaciones construyan y mantengan la infraestructura tradicionalmente usada para desarrollar aplicaciones. A veces, a PaaS se le denomina «middleware», refiriéndose a la forma en que conceptualmente se sitúa en algún punto entre SaaS e IaaS. Las plataformas como *App Engine de Google*, *IBM BlueMix*, y *Apache Stratos* son productos populares de PaaS que están ayudando a racionalizar y desarrollar software sobre plataformas en Cloud Computing.

PaaS es una categoría de servicios Cloud que proporciona una plataforma gracias a la cual los usuarios desarrollan, ejecutan y gestionan aplicaciones sin la complejidad de construir y mantener la infraestructura típicamente asociada al desarrollo y lanzamiento de una aplicación. Esto es, toda la pila de elementos necesarios para poner en marcha un proyecto de desarrollo, visto desde la perspectiva de la infraestructura necesaria, los *frameworks* utilizados y las capacidades de escalado, entre otras. Los proveedores de PaaS alojan el hardware y el software en su infraestructura, liberando así a los clientes de cualquier obligación de instalar hardware y software propio para desarrollar o ejecutar una nueva aplicación.

La escalabilidad sigue siendo una de las palabras clave de PaaS. Los servicios de PaaS son típicamente escalables y producen ahorros sustanciales de costos para los desarrolladores, puesto que por una fracción del coste de los paquetes de desarrollo, no sólo tienen la plataforma, también tienen una serie de ventajas extra como la flexibilidad del despliegue de forma transparente.

1. PaaS proporciona una plataforma con herramientas para probar, desarrollar y alojar aplicaciones en un mismo entorno.
2. Permite a las organizaciones centrarse en el desarrollo sin tener que preocuparse por la infraestructura subyacente. Es el propio proveedor del PaaS el que suministra todos los recursos de forma transparente al consumidor.
3. Los proveedores gestionan la seguridad, los sistemas operativos, el software del servidor y las copias de seguridad. El usuario consumidor no tiene que lidiar con la complejidad de validar la seguridad del sistema donde está desarrollando su PaaS.
4. Facilita el trabajo colaborativo incluso si los equipos trabajan a distancia. Al ser un entorno Cloud Computing, el modelo distribuido está implícito en su naturaleza, permitiendo trabajar con diferentes servicios de forma concurrente.

La tecnología PaaS plantea problemas y riesgos particulares para los proveedores de servicios. Estos desafíos incluyen el equilibrio entre el control, el coste y la capacidad de un servicio basado en PaaS, la provisión de soporte completo para múltiples usuarios (y cómo se deben compartir los recursos), el diseño de controles de acceso basados en funciones, la creación de registros de auditoría y la integración de servicios de terceros. Pueden surgir desafíos adicionales en forma de gestión de la virtualización, el ajuste de la arquitectura computacional de PaaS, el diseño de la interoperabilidad con otros servicios en la nube y la tolerancia a fallos.

Google App Engine[6], CloudFoundry[7], Heroku[8], AWS (Beanstalk)[9] son algunos ejemplos de PaaS.

2.1.2.3. Software as a Service

A veces denominado "software bajo demanda", el SaaS es un modelo de concesión de licencias o de uso de servicios, donde los usuarios pueden trabajar con producto de software completo y funcional a través de la web mediante suscripción. Los usuarios finales suelen acceder a este tipo de servicios a través de un navegador web (lo que hace que el sistema operativo del usuario sea en gran medida irrelevante y pueda usarse desde virtualmente cualquier dispositivo) y se puede facturar en función del consumo del mismo.

Los servicios ofrecidos desde el modelo de SaaS son los más visibles de la arquitectura de Cloud Computing. De hecho, muchos usuarios podrían estar usando productos SaaS sin darse cuenta y no saber distinguir cuando están usando una aplicación para escritorio y cuando están usando una aplicación como servicio web. Productos populares como *Office365*[10] y ERPs como el de *Salesforce.com*[11] han llevado los servicios de SaaS a la vanguardia del lugar de trabajo y son utilizados por miles de empresas. Surge como una alternativa enormemente rentable a las instalaciones y paquetes de software local; el modelo SaaS ofrece una gran variedad de aplicaciones como programas de planificación de recursos empresariales, software de oficina y comunicaciones, paquetes de nómina y contabilidad, gestión de recursos humanos, aplicaciones

[6] `https://cloud.google.com/appengine/`
[7] `https://www.cloudfoundry.org/`
[8] `https://www.heroku.com/`
[9] `https://aws.amazon.com/es/elasticbeanstalk/`
[10] `https://www.office.com/`
[11] `https://www.salesforce.com/es/`

móviles, ... y prácticamente cualquier software susceptible de ser ejecutado bajo Cloud Computing.

Cuando hablamos de SaaS, tenemos que tener en cuenta que permite a los usuarios una serie de ventajas y beneficios significativos en comparación con el modelo tradicional de instalación de paquetes de software. Una de las ventajas principales es que los costes iniciales son más bajos y las tasas de licencia según el producto, servicio o el modo de tarificación pueden ser nulas. Todos los SaaS consideran al menos las siguientes características principales:

1. Los proveedores de SaaS son los que proporcionan a los usuarios software y aplicaciones a través de un modelo de suscripción.
2. Los usuarios no tienen que administrar, instalar o actualizar el software; los proveedores de SaaS lo hacen.
3. Los datos están seguros en la nube; los fallos en los equipos no provocan la pérdida de datos.
4. El uso de los recursos se pueden dimensionar en función de las necesidades del servicio.
5. Las aplicaciones son accesibles desde casi cualquier dispositivo conectado a Internet y desde prácticamente cualquier parte del mundo.

Google Drive[12], Dropbox[13], Gmail[14] o Salesforce[15], son algunos ejemplos de los SaaS más conocidos.

2.1.2.4. XaaS

De forma general todos los modelos pueden ser agrupados dentro de alguna de las categorías de IaaS, PaaS y SaaS, pero desde hace unos años han surgido nuevos modelos de servicios que combinan diferentes características y que ofrecen nuevas funcionalidades a los usuarios consumidores. Este tipo de nuevos modelos de servicio es muy variado, y cubre servicios de contenedores de aplicaciones, funciones o almacenamiento entre muchos otros. En los siguientes párrafos se revisarán de forma breve los que hemos considerado más significativos dentro de Cloud Computing:

[12] `https://www.google.es/drive/`
[13] `https://www.dropbox.com/`
[14] `https://www.google.com/gmail/`
[15] `https://www.salesforce.com/es/`

FaaS. Función como servicio (*Function-as-a-Service*), es un nuevo modelo de servicio que permite a los usuarios consumidores desarrollar, ejecutar y gestionar funcionalidades de aplicaciones, sin tener que lidiar con toda la complejidad que supone preparar y mantener toda una infraestructura completa para el soporte, desarrollo y ejecución de una aplicación. Construir una aplicación siguiendo este modelo es una forma de conseguir una arquitectura que se ha denominado "sin servidor" o "serverless". Este modelo se utiliza principalmente para la construcción de micro-servicios.

FaaS es todavía relativamente joven, y se introdujo en `hook.io` a finales de 2014; posteriormente el modelo fue seguido por *AWS Lambda*[16], *Google Cloud Functions*[17], *Microsoft Azure Functions*[18] y muchos otros. Un servicio de FaaS significa que se puede cargar en la nube funcionalidades modulares que se ejecutan de forma independiente. Antes de esto, se necesitaba escalar un servidor monolítico para manejar la carga de trabajo, pero con FaaS se puede dividir el servidor en varias funciones que se pueden escalar automáticamente y por separado.

StaaS. Almacenamiento como servicio (*Storage-as-a-Service*), es otro de los grandes modelos de servicio de Cloud Computing y relativamente cercano a lo que sería un modelo de SaaS e IaaS, ya que al fin y al cabo lo que permite es utilizar unos servicios de almacenamiento virtual a través de un software para la gestión del almacenamiento como servicio.

Como alternativa al almacenamiento en cintas magnéticas, discos duros u otros sistemas de almacenamiento, los administradores de sistemas satisfacen sus necesidades de almacenamiento y copia de seguridad mediante acuerdos de nivel de servicio con un proveedor de StaaS, normalmente sobre la base del coste por GigaByte almacenado y del coste por transferencia de datos. El cliente transfiere los datos destinados al almacenamiento al proveedor de servicios según una programación establecida a través de la red del proveedor de StaaS o a través de Internet. El proveedor de almacenamiento proporciona al cliente el software necesario para acceder a sus datos almacenados. Los clientes utilizan el software para realizar tareas estándar asociadas con el almacenamiento, incluyendo transferencias y copias de seguridad de datos. Este tipo de almacenamiento está en auge entre las pequeñas y medianas empresas, ya que no se requiere un presupuesto inicial para configurar los discos duros, los servidores o el propio personal de TI.

[16] `https://aws.amazon.com/es/lambda/`
[17] `https://cloud.google.com/functions/?hl=es`
[18] `https://azure.microsoft.com/es-es/services/functions/`

StaaS también se comercializa como una excelente técnica para mitigar los riesgos en la recuperación de desastres al proporcionar almacenamiento de datos a largo plazo y mejorar la estabilidad en cuanto al acceso a los mismos. Sin embargo, en este modelo de servicio nos encontramos con problemas relativos a la localización de los datos y la regulación de los mismos.

CaaS. Contenedores como servicio (*Container-as-a-Service*). En lugar de virtualizar la pila de hardware como en el caso de las máquinas virtuales en el IaaS, los contenedores se virtualizan a nivel de sistema operativo, con múltiples contenedores ejecutándose directamente sobre el núcleo del Sistema Operativo (SO). Esto significa que los contenedores son mucho más ligeros: comparten el núcleo del sistema operativo, comienzan mucho más rápido y utilizan una fracción de la memoria en comparación con el arranque de todo un sistema operativo sistema operativo. Hay muchas opciones para contenedores, pero *Docker*[19], *Kubernetes*[20] y *Moby*[21] son los más extendidos.

Los contenedores pueden ejecutarse prácticamente en cualquier lugar, lo que facilita enormemente el desarrollo y la implementación: a) en sistemas operativos *Linux*, *Windows* y *MacOS*; b) en máquinas virtuales o en sistemas de computación físicos; c) en máquinas de desarrolladores o en centros de datos locales; y, d) por supuesto, en la nube pública. La gran popularidad del formato de imagen *Docker* para contenedores ayuda aún más a la portabilidad. *Google Container Engine* (GKE), *AWS* (ECS), *Azure* (ACS) y *Pivotal* (PKS) son algunos ejemplos de proveedores de CaaS.

MLaaS/DMaaS. Machine Learning como servicio (MLaaS) y minería de datos como servicio (DMaaS). Al adoptar software y servicios de Inteligencia Artificial, las empresas pueden mejorar las capacidades de sus productos, interactuar mejor con los clientes, agilizar las operaciones comerciales y crear estrategias comerciales predictivas.

Los desarrolladores pueden construir de forma rápida y eficiente con MLaaS ya que tienen acceso a algoritmos y modelos pre-entrenados que de otro modo les llevarían muchos recursos poner en marcha. Los desarrolladores que tienen el conocimiento y las habilidades para construir modelos de aprendizaje máquina son difícilmente accesibles y son caros, por lo que la facilidad y la velocidad de configuración junto con

[19] `https://www.docker.com/`
[20] `https://kubernetes.io/es/`
[21] `https://mobyproject.org/`

Propiedad/Tipo	**Nube pública**	**Nube privada**	**Nube híbrida**	**Nube comunitaria**
Escalabilidad	Muy alta	Limitada	Muy alta	Limitada
Fiabilidad	Moderada	Muy alta	Media/Alta	Muy alta
Seguridad	Depende del proveedor	Seguridad muy alta	Segura	Segura
Rendimiento	Bajo/Medio/Alto	Bueno	Bueno	Muy bueno
Coste	Económico	Coste alto	Coste alto	Coste alto

Tabla 2.2: Propiedades de los diferentes tipos de nubes.

los beneficios monetarios son un gran atractivo para las empresas que integran estos servicios.

Hay una serie de proveedores de servicios que están añadiendo a su catálogo, servicios MLaaS para que las empresas puedan elegir paquetes de tipo muy variado, incluyendo el procesamiento de lenguaje natural (NLP), visión artificial o plataformas y APIs de aprendizaje automático.

Amazon Machine Learning[22], *Google ML-Engine*[23], *Microsoft ML-Studio*[24] e *IBM Watson*[25] ofrecen en su catálogo de servicios diferentes funcionalidades para el desarrollo de productos basados en minería de datos y aprendizaje máquina.

2.1.3. Modelos de despliegue de Cloud Computing

El modelo de despliegue hace referencia a las diferentes variantes existentes para la puesta en marcha de Cloud Computing. NIST propone 4 modelos de despliegue para Cloud Computing: Público, Privado, Comunidad e Híbrido. En la Tabla 2.2 se pueden ver algunas de las características principales de estos tipo de modelos de despliegue.

[22] `https://aws.amazon.com/machine-learning/`
[23] `https://cloud.google.com/ml-engine/`
[24] `https://azure.microsoft.com/es-es/services/machine-learning-studio/`
[25] `https://www.ibm.com/watson`

2.1.3.1. Cloud pública

Las nubes públicas son la forma más común de desplegar entornos de Cloud Computing. Los recursos de la nube (como los servidores, el almacenamiento, las redes o las bases de datos, entre otros) se entregan a través de Internet. Con una nube pública, todo el hardware, software e infraestructura de soporte es propiedad del proveedor de la nube y está gestionada por él. En una nube pública, los usuarios consumidores comparten el mismo hardware, almacenamiento y red con otros usuarios consumidores de la nube.

Algunas de las ventajas de las nubes públicas:

- Costes más bajos: no es necesario comprar hardware ni software, y sólo se paga por el volumen de uso de los servicios que se utilizan.
- Sin mantenimiento: es el propio proveedor de servicios el encargado del mantenimiento de la infraestructura.
- Recursos casi ilimitados: es posible utilizar una gran variedad de recursos, de modo que estos puedan ser usados según la demanda y las necesidades de la empresa.
- Alta fiabilidad: una vasta red de servidores proporciona seguridad contra fallos de cualquier tipo en la infraestructura.

2.1.3.2. Cloud privada

Una nube privada consiste en recursos informáticos utilizados exclusivamente por una empresa u organización. La nube privada puede estar ubicada físicamente en el centro de datos de la organización, o puede ser alojada por un proveedor de servicios externo. En una nube privada, los servicios y la infraestructura siempre se mantienen en una red privada y el hardware y el software se dedican exclusivamente a su organización. De esta manera, una nube privada puede facilitar a una organización la personalización de sus recursos para cumplir con los requisitos específicos de TI. Las nubes privadas son a menudo utilizadas por gobiernos, instituciones financieras y cualquier otra organización de tamaño mediano o grande con operaciones críticas para el negocio que buscan un mayor control sobre su entorno. Algunas de sus características más destacadas son:

- Mayor flexibilidad: la organización puede personalizar su entorno de Cloud Computing para satisfacer necesidades empresariales específicas.

- La mejora de la seguridad: los recursos no se comparten con otros, por lo que es posible alcanzar niveles más altos de control y seguridad.
- Alta escalabilidad: las nubes privadas permiten escalabilidad y eficiencia, pero no al nivel que los proveedores ofrecen en la nube pública.

2.1.3.3. Cloud híbrido

Las nubes híbridas combinan infraestructura local, o nubes privadas, con nubes públicas para que las organizaciones puedan incluir las ventajas de ambas. En una nube híbrida, los datos y las aplicaciones pueden moverse entre nubes privadas y públicas para obtener mayor flexibilidad y más opciones de implementación. Por ejemplo, puede utilizar la nube pública para necesidades de alto volumen y menor seguridad, como el correo electrónico basado en web, y la nube privada (u otra infraestructura local) para operaciones confidenciales y críticas para el negocio, como el trabajo con datos internos.

En una nube híbrida, es posible modelar una provisión de recursos denominada *inter-cloud*. Esto ocurre cuando una aplicación o recurso se ejecuta en la nube privada hasta que hay un pico en la demanda, momento en el que la organización puede "moverse" a la nube pública para aprovechar los recursos informáticos adicionales para computación, servicios, etc.

Algunas de las ventajas de las nubes híbridas son las siguientes:

- Control: la organización o empresa puede mantener una infraestructura privada para activos sensibles.
- Flexibilidad: es posible aprovechar los recursos adicionales en la nube pública cuando los necesite bajo demanda.
- Rentabilidad: con la capacidad de escalar a la nube pública, el usuario paga por los recursos de computación adicional sólo cuando lo necesita.
- La facilidad de transición a la nube no tiene por qué ser problemática, ya que se puede migrar de forma gradual.

2.1.3.4. Cloud comunitaria o Cloud para comunidades

Una comunidad en nube es una plataforma «multi-tenancy» que permite que varias empresas trabajen en la misma plataforma, ya que tienen necesidades y preocupaciones similares. Un ejemplo de uso de una nube comunitaria

sería probar algunos productos de seguridad de gama alta o incluso probar algunas características de un entorno de nube pública.

El gobierno, la sanidad y algunas industrias privadas reguladas están aprovechando las características de seguridad añadidas dentro de un entorno de nube comunitario. En lugar de limitarse a aprovisionar espacio en una nube pública, las organizaciones pueden probar y trabajar en una plataforma de nube segura, "dedicada" e incluso conforme a determinadas normativas. La parte realmente interesante es que con una nube de comunidad, la presencia puede ser *on-site* u *off-site*.

Otro ejemplo significativo podría ser cuando varias organizaciones pueden requerir una aplicación específica que reside en un conjunto de servidores en la nube. En lugar de dar a cada organización su propio servidor en la nube para esta aplicación, el proveedor de *hosting* permite a varios clientes conectarse a su entorno y segmentar sus sesiones. El cliente, sin embargo, sigue usando las mismas piezas de hardware que otras personas. Sin embargo, todo el mundo está utilizando estos servidores con el mismo propósito: acceder a esa única aplicación, que es lo que la convierte en una nube de comunidad.

2.1.4. Actores en Cloud Computing

La arquitectura de referencia propuesta por NIST, define cinco actores principales en términos de funciones y responsabilidades. La siguiente lista detalla cada uno de los actores que intervienen en Cloud Computing:

- Consumidor (*Cloud Consumer*). Una persona u organización que mantiene una relación comercial y utiliza el servicio de los proveedores de Cloud.
- Proveedor de Cloud Computing (*Cloud Provider*). Una persona, organización o entidad responsable de poner un servicio a disposición de las partes interesadas.
- Auditor de nube (*Cloud Auditor*). Una parte que puede llevar a cabo una evaluación independiente de los servicios en la nube, las operaciones del sistema de información, el rendimiento y la seguridad de la implementación de la nube.
- Intermediador (*Broker*). Una entidad que gestiona el uso, el rendimiento y la entrega de servicios en la nube y negocia las relaciones entre los proveedores y los consumidores de Cloud.

- Portador (*Carrier*). Un intermediario que proporciona conectividad y transporte de servicios en la nube desde los proveedores de Cloud a los consumidores de la nube.

2.2. Servicios Web y arquitecturas orientadas a servicios

Es más probable que los servicios que están bien diseñados sean reutilizables. Las empresas pueden beneficiarse de la reutilización de al menos dos maneras: a) evitando los gastos de desarrollo de nuevo software, y b) aumentando la fiabilidad del catálogo de software a lo largo del tiempo. Se pueden realizar pruebas menos extensas si un servicio existente se coloca en una nueva aplicación, en comparación con las pruebas necesarias para implementar software que se escribió desde cero.

El concepto de servicios dentro de una aplicación existe desde hace tiempo. Los servicios están pensados para ser bloques de construcción independientes que de forma conjunta representan un entorno de aplicación. Sin embargo, a diferencia de los componentes tradicionales, los servicios tienen una serie de características únicas que les permiten participar como parte de una arquitectura orientada a servicios.

Una de estas cualidades es la completa autonomía de otros servicios. Esto significa que cada servicio es responsable de su propio dominio, lo que normalmente se traduce en limitar su alcance a una función empresarial específica (o a un grupo de funciones relacionadas).

Este enfoque de diseño da como resultado la creación de unidades aisladas de funcionalidad de negocio unidas entre sí. Debido a la independencia de que gozan los servicios dentro de este entorno, la lógica de programación que encapsulan no necesita cumplir con ninguna plataforma o conjunto de tecnologías, lo que añade un grado de abstracción importante.

2.2.1. Service Oriented Architecture

La *Arquitectura Orientada a Servicios* (*SOA*) implica el despliegue de servicios con las siguientes características:

1. Maneja un proceso de negocio como el cálculo de una cotización de bolsa o la distribución de correo electrónico; maneja una tarea técnica como el acceso a una base de datos; o proporciona datos de negocio y los detalles técnicos para construir una interfaz gráfica.
2. Puede acceder a otro servicio. Con la tecnología de tiempo de ejecución adecuada, puede acceder a un programa tradicional y responder a diferentes tipos de solicitudes, como aplicaciones web.
3. Es relativamente independiente de otros programas.
4. Un servicio puede gestionar la interacción con otras entidades, tanto externas como internas.

La utilización de una arquitectura orientada a servicios implica un estilo de desarrollo que se centra en el negocio en su conjunto, en la modularidad y la reutilización. En la Figura 2.2 se pueden ver algunos elementos fundamentales sobres los que se asienta el diseño en SOA.

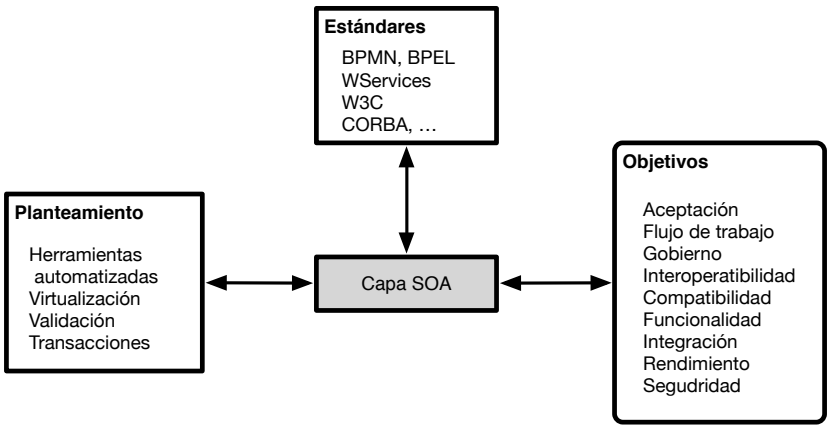

Figura 2.2: Arquitectura de la "web semántica".

Una aplicación orientada a servicios es una aplicación que se compone principalmente de servicios, que a menudo se encuentran en una jerarquía. El nivel superior contiene uno o más servicios de integración, cada uno de los cuales controla un flujo de actividades.

El segundo nivel se compone de servicios que cumplen con una tarea empresarial de nivel relativamente bajo. Por ejemplo, un servicio de integración puede invocar una serie de servicios comerciales para verificar los detalles que fueron proporcionados por un cliente. Posteriormente otro servicio, ejecuta un cálculo a partir de esos datos y ofrecer los resultados.

El tercer nivel consiste en servicios de acceso a los datos, cada uno de los cuales se encarga de la tarea relativamente técnica de lectura/escritura desde y hacia las áreas de almacenamiento de datos, como las bases de datos y colas de mensajes.

El punto central de SOA es la flexibilidad. Algunos servicios de integración proporcionan diferentes operaciones a diferentes consumidores de servicios, y otros invocan otros servicios de integración. Además, un consumidor puede acceder a diferentes tipos de servicios desde una aplicación orientada a servicios.

2.2.2. Web Services

Los servicios web (*Web Services*) son aplicaciones autónomas modulares que se pueden describir, publicar, localizar e invocar a través de Internet.

El servidor de aplicaciones da soporte a los servicios web que se desarrollan e implementan de acuerdo con la especificación para el entorno donde se han desarrollado. El servidor de aplicaciones da soporte a los modelos de servicios.

Un escenario típico de servicios web es una aplicación de empresa que solicita un servicio de otra aplicación existente. La solicitud se procesa con una dirección web determinada, utilizando mensajes *SOAP*[26] a través de un protocolo de transporte HTTP. El servicio recibe la solicitud, la procesa y devuelve una respuesta. Ejemplos de un servicio web pueden ser los informes meteorológicos o la obtención de información bursátil. La llamada al método es síncrona, esto es, espera a que el resultado esté disponible. Los servicios web de transacciones, que dan soporte a operaciones de cotizaciones, de empresa a empresa (*B2B*[27]) o de empresa a cliente (*B2C*[28]), son, por ejemplo, la reserva de vuelos o los pedidos de compra *on-line*. Los servicios web pueden incluir el propio servicio o el cliente que accede al servicio. Los servicios permiten aumentar la flexibilidad de los procesos empresariales al integrarse con aplicaciones que de otra forma no se comunicarían.

Los servicios web reflejan el enfoque SOA en la programación. Este enfoque está basado en la creación de aplicaciones detectando e implementando los servicios de red disponibles o invocando las aplicaciones disponibles para

[26] `SimpleObjectAccessProtocol`
[27] Business to Business
[28] Business to Costumer

que realicen una tarea. Los servicios web proporcionan interoperatividad, por ejemplo, las aplicaciones de servicios web proporcionan componentes creados en distintos lenguajes de programación para que funcionen juntos como si se hubieran creado utilizando el mismo lenguaje. Los servicios web dependen de las tecnologías de transporte existentes (como HTTP) y las técnicas de codificación de datos estándar (como *XML*, *Extensible Markup Language*) para invocar la implementación. Los componentes fundamentales de los servicios web son:

- *Web Services Description Language* (*WSDL*): es la estructura basada en *XML* que describe el servicio web. La solicitud de servicio web utiliza este archivo para enlazarse con el servicio.
- *SOAP*: es el protocolo basado en *XML* que utiliza la solicitud de servicio web para invocar el servicio.
- *Universal Description, Discovery and Integration - UDDI*: es el registro que alberga el intermediario del servicio, define un modo de publicar y encontrar información sobre servicios web, con dos funciones: definir cómo se comunican los clientes con los registros *UDDI* y albergar el conjunto de registros (para consultar y publicar información) en los directorios de servicios.

2.3. Web semántica

Cuando utilizamos un navegador web desde nuestro terminal móvil, tableta o PC, accedemos a la información de un modo sencillo, siguiendo la información y los documentos enlazados que ofrecen los sitios en cuestión. Este modo de gestionar la información, e interconectar contenido y documentos en Internet es uno de los pilares fundamentales de la *World Wide Web* (*W3C*). La web que conocemos, en términos generales podríamos definirla como una colección de documentos e información enlazada. Además de esta "Web de documentos e información" tradicional, la *World Wide Web* desde hace ya varios años está convirtiéndose en lo que se ha denominado "web de datos".

El objetivo final de la "web de datos" es permitir a los ordenadores hacer un trabajo más útil y desarrollar plataformas que puedan soportar interacciones a través de la red. El término "web semántica" se refiere a la visión del

	WWW	**"web semántica"**
Contenido	Sin estructturar	Formal
Público	Personas	Aplicaciones
Enlaces	Destino/Lugar	Destino/Lugar/Significado
Vocabulario	Formato	Semántico y lógico
Logica	No estándar	Descripciones lógicas

Tabla 2.3: Diferencias entre la web clásica y la *"web semántica"*

W3C[29] de la web de datos enlazados. En esa sección revisaremos las tecnologías de la "web semántica" que permiten a las personas crear almacenes de datos en la web, crear vocabularios y escribir reglas para el manejo de datos, o conocer los principios de diseño de *Linked Data*. Por ultimo se pondrán en valor varias de las tecnologías potenciadoras de la "web semántica" como son *RDF*, *OWL* o SPARQL, entre otras.

2.3.1. Definición y fundamentos

La llamada *"web semántica"* es una extensión del *World Wide Web*, con la ventaja añadida de que posibilita la creación de estructura y significado para los datos e información mostrada. Esto permite que la información, los metadatos y los documentos puedan ser comprensibles por las máquinas de modo que todas las entidades expuestas tengan un significado gracias al cual conocer qué es cada cosa y las relaciones que posee. Con ello los agentes software pueden integrar datos desde múltiples fuentes, que podrían parecen inconexas pero que al estar basadas en tecnología semántica, es posible inferir hechos o responder a preguntas complejas sobre esos datos.

El término *"web semántica"* fue propuesto por primera vez por *Tim Berners-Lee* [Berners-Lee et al., 2001] en 2001 para definir una web de datos, una evolución de la web clásica que conocemos, y que en su artículo describe esta evolución esperada hacia una web de conocimiento basada en tecnologías semánticas. Para definir más precisamente la "web semántica", es necesario conocer algunos conceptos relacionados con la semántica.

La semántica se refiere a los aspectos del significado, sentido o interpretación de signos lingüísticos como símbolos, palabras, expresiones o representaciones formales. Conocer el significado de conceptos o entidades permite un

[29]W3C: *World Wide Web Consortium*

uso más efectivo de los datos. Cuando nos referimos la Web, el significado de los elementos que se muestran está ausente generalmente en la mayoría de las fuentes de información, por lo que no se dota a esa documentación de un significado conceptual. Son los usuarios los que a través de su conocimiento son capaces de distinguir conceptos y su significado.

Si usamos como ejemplo una web, al codificar la página de contenido con *HTML*, tenemos un código y unas etiquetas para definir aspectos tales como el formato de la información o la estructura de la plantilla del aspecto que queremos que tenga el documento web. Con `<p>La web semántica fue propuesta ...</p>` estamos indicando que el contenido *"La web semántica fue propuesta ..."* corresponde simplemente con un párrafo, y que este será procesado por el motor de renderizado del navegador dentro del documento con los aspectos de formato correspondientes de color, posición, tamaño, etc. Con esto estamos expresando que semánticamente sabemos que lo que hay contenido entre las dos etiquetas `<p>...</p>` es un párrafo, y que el contenido podría ser relevante, pero no se puede extraer más información. Estas semánticas, por ejemplo, son bastante débiles y limitan en gran medida una búsqueda o extracción de conocimiento de estos datos con precisión. Además, no considera que los elementos incluidos en el documento (imágenes, términos, frases, etc.) puedan estar enlazados o relacionados con otros, de modo que se enriquecería de forma automática el contenido de la web.

La semántica permite dar un significado útil a conceptos a través de relaciones y enlaces con otros conceptos e ideas. De modo que si una palabra clave se relaciona con otras, en relaciones bien definidas o en forma de red de relaciones de palabras, se consigue dar un significado general a la palabra o concepto específico. Conceptos como por ejemplo "agua", puede ser relacionada con otras palabras claves como "ríos", "mares", "océanos", "canalización", "bebida", "vida", etc. de modo que las relaciones entre ellas exponen la semántica. Cuando una estructura formal captura el conjunto de términos, estos términos se adhieren a reglas gramaticales concretas, de modo que si los términos forman un estándar o lenguaje asociado conjuntamente a una gramática, estos ayudan a incorporar aspectos semánticos; además cada vez que se incrementa el tamaño de la red contextual de reglas gramaticales y términos del lenguaje, es la semántica la que se enriquece más. De este modo la "web semántica" se puede definir como una web de datos descritos y enlazados de modo que se establezca unos contextos y semánticas que se adhieren a construcciones gramaticales y lingüísticas bien definidas.

La "web semántica" se enfoca en la semántica a través de conexiones estandarizadas a conjuntos de información relacionada. Para poder hacer

efectivo este modelo, es vital que los datos puedan ser etiquetados de forma unívoca y que estos mismos puedan ser direccionables, de modo que si se define una entidad, por ejemplo conceptual "coche", es posible relacionar un coche con otro por su atributos e identificadores, y además es posible encontrarlo a efectos de búsqueda. Cada elemento único, como por ejemplo, "coche", se conecta a un contexto mucho más amplio. En la "web semántica" se ofrecen caminos potenciales para su definición, para relaciones jerárquicas conceptuales, para relaciones hacia la información relacionada o para relaciones hacia la instancias específicas de "coche" (un modelo de coche en concreto, propiedad de una persona concreta, etc. por ejemplo). Para concretar la base fundamental de la "web semántica" se han establecido diferentes componentes clave a la hora de trabajar con este modelo siendo los siguientes elementos básicos los más característicos:

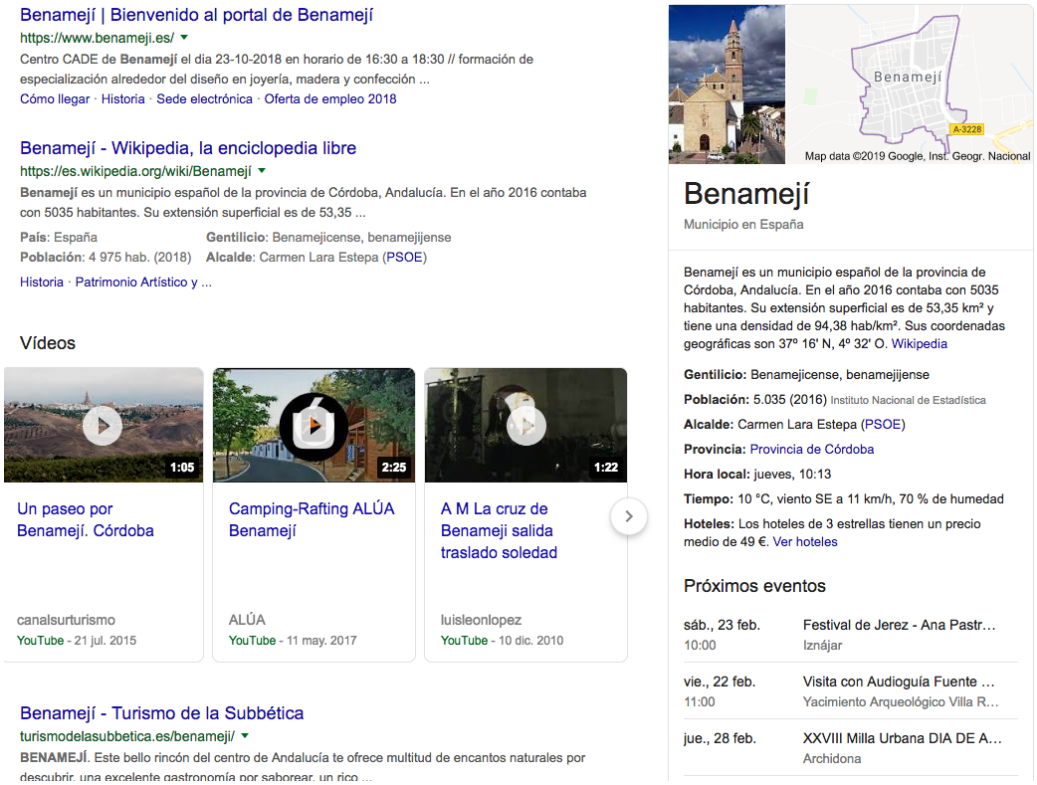

Figura 2.3: Contenido semántico enlazado y descubierto por un buscador (a la derecha).

Sentencias. Una sentencia está formada por varios elementos que típicamente forman una tripleta o triple. Una tripleta consiste en un sujeto, un objeto y un objeto; por ejemplo *Manuel* `esTipo` *Persona* (`Manuel`

`isType Person`), contiene *Manuel* que es un sujeto, `esTipo` (`isType`) que es el predicado y finalmente *Persona*, que es el objeto. De este modo se pueden definir desde una sola a cientos de miles o incluso millones de estas combinaciones de tripletas. Estas sentencias definen la estructura de la información, las instancias y las restricciones sobre los componentes del vocabulario (ver Figuras 2.5 y 2.4).

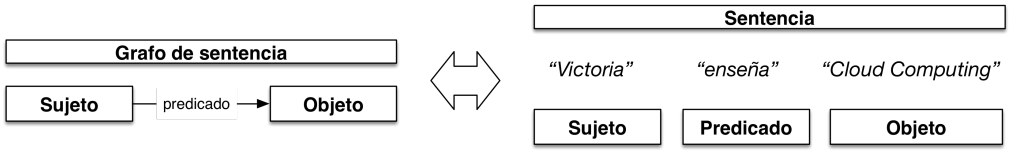

Figura 2.4: Grafo de sentencia y composición de una sentencia

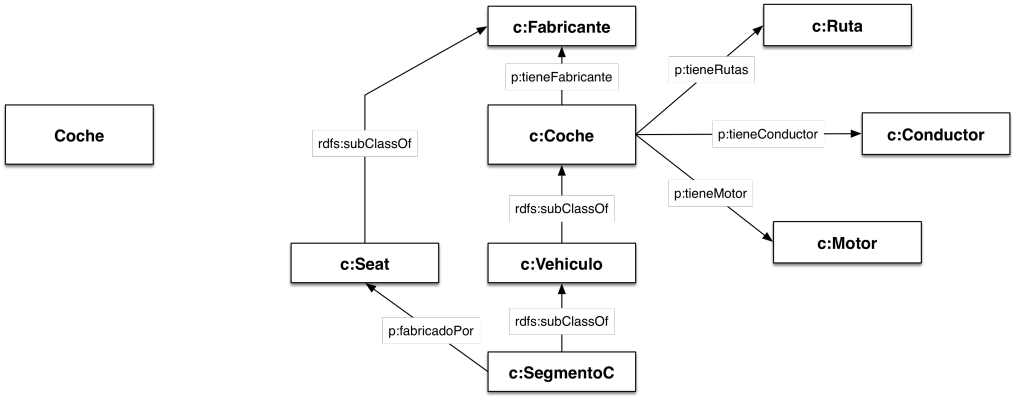

Figura 2.5: Diferencia entre la definición de un concepto con texto normal (izquierda) y el mismo utilizando tecnología semántica (derecha).

URI. El *Unified Resource Identifier* permite identificar elementos de forma única a lo largo y ancho de toda Internet. Cada componente que forma una sentencia, contiene una URI que afirma la identidad de la misma en todo el entorno web. El uso de URI, permite eliminar los conflictos en cuanto al modo de nominar los elementos y asegura que dos ítems son iguales o son diferentes según su URI. Dentro de la "web semántica" juegan un papel muy importante puesto que además permiten enlazar con otros elementos de información adicional, permitiendo descubrir más conocimiento sobre los datos.

Lenguaje. Las sentencias o tripletas tienen que ser construidas en un lenguaje acorde con la "web semántica".

Ontología. Una ontología consiste en un grupo de sentencias que definen conceptos, relaciones y restricciones. De modo que con las ontología podemos capturar y definir en profundidad áreas de conocimiento de cualquier tipo.

Datos de instancias. Una instancia es una sentencia que contiene información sobre instancias específicas más que sobre un concepto genérico. Por ejemplo, cuando en la Figura 2.5, se indica el concepto coche, estamos hablando de un concepto genérico que puede tener muchos atributos, etc., pero si hacemos una instanciación del concepto coche, podríamos entonces decir que una instancia de coche sería *MiCoche* (*Seat Altea, Matrícula X, Conductor Y,* etc.) De modo que el concepto *Coche* podría ser utilizado por millones de instancia de tipo `coche`. El término instancia está en la misma línea que su definición para el ámbito de la programación dirigida a objetos.

Herramientas de consulta. Para poder consultar toda la información contenida dentro de los conjuntos de datos de ontologías y sentencias, es necesario disponer de herramientas para hacer consultas. Estas herramientas permiten acceder de una manera eficiente la información contenida en las ontologías, tanto a nivel de exploración de los grafos como para realizar búsquedas completas.

Razonadores. Los razonadores añaden capacidad de inferencia a la "web semántica". Esto permite añadir aspectos de lógica tanto en las consultas como el procesamiento, de modo que podríamos hacer consultas de tipo "Es Manuel una persona", o más complejas como "Es el estado civil casado es un tipo de relación simétrica".

Una prueba de estos conceptos se pueden verificar, realizando una búsqueda en alguno de los de buscadores más conocidos como *Google*, *Yandex*, o *Bing*. Cuando realizamos una consulta para buscar un concepto o término, por ejemplo, el nombre de un pueblo llamado "Benamejí"; a la derecha de la pantalla de resultados obtenemos una información exacta de lo que estamos buscando. Además de esta información aparecen otros datos y documentos relacionados tanto de forma directa como indirecta al término que buscamos. Esa información que muestra indica que realmente está entendiendo el contenido y que de forma precisa conoce que el término "Benamejí" es un "municipio" que está dentro de la provincia de "Córdoba" que forma parte de "España", y con estos atributos relaciona otros muchos más tales como pueden ser el tamaño de la población, quién es el alcalde, el tiempo que hace actualmente o fotografías destacadas de sus monumentos (todos ellos,

elementos enlazados y relacionados con el término a través de tecnología semántica).

En 2012, *Google*, añadió a su motor de búsqueda una serie de funcionalidades tales como en grafos de conocimiento («Knowledge Graphs») para su tecnología semántica. Con estás herramientas, el buscador recopila todos los datos sobre personas, lugares y cosas para crear resultados de búsqueda interconectados y presentarlos en forma de respuestas útiles y no sólo de simples enlaces. Con esto se persigue una mejora de la organización de la información global y por otro lado hacerla accesible a los usuarios en cualquier momento y desde cualquier lugar para mejorar la experiencia con la información. En la "web semántica" la semántica se dirige por medio de enlaces estandarizados a información relacionada. Todos estos elementos correspondientes a términos, enlaces y relaciones forman un vocabulario y ontología. Para la definición y descripción de conceptos se utilizan diferentes conjuntos de ontologías para cada domino de problema que a su vez pueden estar enlazadas con otros dominios y así sucesivamente. Todos los aspectos clave de la "web semántica" son desarrollados en detalle en la siguientes subsecciones.

2.3.2. Arquitectura de la "web semántica"

El uso, despliegue e implantación de la "web semántica" frente a la web tradicional supone un cambio de paradigma, ya que es necesario pasar de una web basada lenguaje natural no estructurado a una web estructurada y organizada, donde el contenido está semánticamente etiquetado. En la figura 2.6 se puede ver la estructura de la "web semántica". Cada uno de los componentes de la Figura 2.6 se detallan a continuación:

Unicode. Se trata de una codificación del texto que permite utilizar los símbolos de diferentes idiomas sin que aparezcan caracteres extraños. De esta forma, se puede expresar información en la "web semántica" en cualquier idioma.

URI. Es el acrónimo de "Uniform Resource Identifier" o Identificador Uniforme de Recursos, identificador único que permite la localización de un recurso que puede ser accedido vía Internet. Se trata de la URL (descripción de la ubicación) más el URN (descripción del espacio de nombre).

RDF + rdfschema. Basada en la capa anterior, esta capa define el lenguaje universal con el cual podemos expresar diferentes ideas en la "web

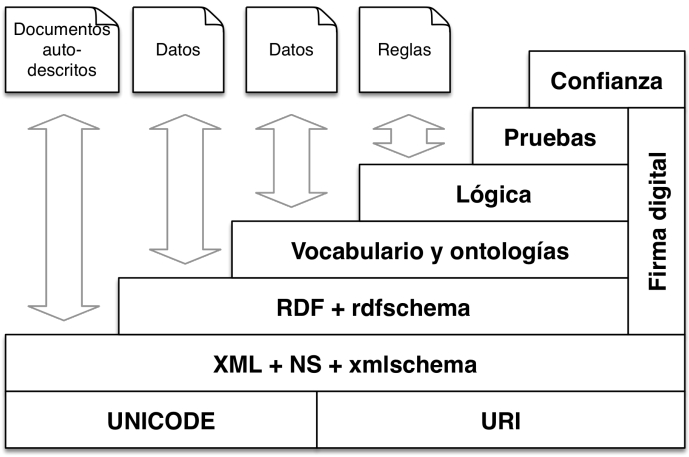

Figura 2.6: Arquitectura de la "web semántica".

semántica". *RDF*[Beckett et al., 2014] es un lenguaje simple mediante el cual definimos sentencias en el formato de triple o tripleta (sujeto: el recurso al que nos referimos; predicado: el recurso que indica qué es lo que estamos definiendo; y objeto: puede ser el recurso o un literal que podría considerarse el valor de lo que acabamos de definir). El modelo *RDF* o *Resource Description Framework* es un modelo común (*Framework*) que permite hacer afirmaciones sobre los recursos (*Description*) y que hace posible que estos recursos pueden ser nombrados por URIs (*Resource*). Por su parte *RDF Schema* provee un vocabulario definido sobre *RDF* que permite el modelo de objetos con una semántica claramente definida. Esta capa no sólo ofrece descripción de los datos, sino también cierta información semántica. Tanto esta capa como la anterior corresponden a las anotaciones de la información (metadatos).

Lenguaje de Ontologías. Ofrece un criterio para catalogar y clasificar la información. El uso de ontologías permite describir objetos y sus relaciones con otros objetos ya que una ontología es la especificación formal de una conceptualización de un dominio concreto del conocimiento. Esta capa permite extender la funcionalidad de la "web semántica", agregando nuevas clases y propiedades para describir los recursos.

Lógica. Además de ontologías se precisan también reglas de inferencia. Una ontología puede expresar la regla *"Si un código de ciudad está asociado a un código de estado, y si una dirección es el código de ciudad, entonces esa dirección tiene el código de estado asociado"*. De esta forma, un programa podría deducir que una dirección de la Universidad

Complutense, al estar en la ciudad de Madrid, debe estar situada en España, y debería por lo tanto estar formateado según los estándares españoles.

Pruebas. Será necesario el intercambio de "pruebas" escritas en el lenguaje unificador (se trata del lenguaje que hace posible las inferencias lógicas hecha posibles a través del uso de reglas de inferencia tal como es especificado por las ontologías) de la "web semántica".

Confianza. Los agentes deberían ser muy escépticos acerca de lo que leen en la "web semántica" hasta que hayan podido comprobar de forma exhaustiva las fuentes de información. (*Web Of Trust RDF Ontology - WOT* [30]).

Firma digital. Módulo de encriptación de los datos que serán utilizados por los ordenadores y los agentes para verificar que la información adjunta ha sido ofrecida por una fuente específica confiable (*XML Signature WG*[31]).

2.3.3. Ontologías y Vocabularios

En la "web semántica" un vocabulario define los conceptos y las relaciones, usados para describir y representar un área de conocimiento. Un vocabulario es usado para agrupar o clasificar términos específicos para un dominio particular de aplicación, entorno, para caracterizar relaciones, o para definir restricciones entre todos los términos. Dependiendo del domino que estemos tratando y del detalle que se quiera conseguir, estos vocabularios puede ser tremendamente complejos (miles de términos) o vocabularios sencillos que definen cosas a alto nivel, donde con varios términos es posible definir conceptos de este tipo. No hay un distinción clara entre ontología y vocabulario, pero en cierto modo cuando nos referimos a ontologías, entonces estamos hablando de una colección de términos más formales y estrictos en contraposición a un vocabulario donde probablemente no existe está firmeza. Por tanto vocabularios y ontologías forman una parte fundamental de la "web semántica".

Un vocabulario dentro de la "web semántica" tiene como objetivo permitir la integración de los datos, teniendo en cuenta que pueden existir ambigüedades sobre los términos del vocabulario usados en diferentes documentos, y que

[30] `http://xmlns.com/wot/0.1/` and FOAF:`http://xmlns.com/foaf/0.1/`
[31] `http://www.w3.org/Signature/`

probablemente estos hacen referencia al mismo término. Además también es importante destacar sobre los vocabularios, que si se incluyen algunas relaciones adicionales para un término es posible que se consiga un conocimiento y un descubrimiento de información extra que mejorará la eficacia del vocabulario.

Para entender de forma clara el concepto de ontología, si trasladamos el domino de aplicación al campo de la salud, tenemos que, por ejemplo, en medicina, los doctores utilizan ontologías sin apenas darse cuenta para representar el conocimiento sobre enfermedades, síntomas, problemas, medicamentos, o tratamientos entre otros. Igualmente si cambiamos al dominio de las compañías farmacéuticas, igualmente utilizan un grafo de conocimiento tal que pueden relacionar un medicamento con sus recomendaciones, composición, efectos secundarios, posología, etc. De modo que si combinamos estos dos dominios de conocimiento con los datos de pacientes, tendremos una red muy extensa de elementos interconectados relacionados con el paciente, y mediante el cual será posible descubrir conocimiento, explorar relaciones entre conceptos, servir de herramienta de toma de decisiones para la búsqueda de tratamientos, o realizar un estudio sobre los efectos de un medicamento en una población. Toda esa información puede ser recogida mediante ontologías.

Una ontología es un concepto filosófico que desde hace bastante tiempo se está considerando dentro del área de la Ciencias de la Computación. Dentro de este área trata de establecer un efectivo y eficiente mapa conceptual dentro de uno o múltiples dominios interconectados (relacionados) con la finalidad de facilitar la inter-operatibilidad (intercambio y comunicación) de información entre diferentes entidades o actores. Las ontologías se definen de forma estándar un vocabulario de conceptos y unas relaciones entre ellos dentro de un dominio de aplicación o conocimiento concreto. Tratan de recoger reglas lógicas y restricciones de modo que una entidad computacional sea capaz de comprender los términos y las relaciones que existen dentro un dominio de conocimiento. Desde mediados de los años 70, esta nueva área de conocimiento ha ido evolucionando y está en continuo crecimiento, extendiéndose al campo de la Inteligencia Artificial, donde se trata de capturar el conocimiento como una clave para construir entornos más solidos y fiables para sistemas de Inteligencia Artificial. Además, el término ha sido adoptado por los investigadores en Inteligencia Artificial y se reconoce la aplicabilidad del trabajo desde la lógica matemática de modo que es posible crear nuevas ontologías como modelos computacionales que les habiliten para ciertas clases de razonamiento automático de los algoritmos como se estudia en [Gruber, 1993]. Posteriormente en la década de los 90 hubo un fuerte impulso

para crear una base que serviría como componente estándar para los sistemas de conocimiento [Gruber, 1995] además de delimitar cual sería el espacio específico para la definición de ontologías en el área de la informática. Los aspectos fundamentales de una ontología son los siguientes:

- Una ontología define, describe y especifica los conceptos, relaciones y otros aspectos clave para el modelado del dominio concreto.
- Una ontología es una especificación que toma la forma de la definición un vocabulario.

Los elementos que componen una ontología son:

- **Conceptos.** Cuando se realiza el proceso de definir o describir un domino de conocimiento, cada elemento individual susceptible de ser especificado se conoce como concepto. Se refiere a una clase o una idea básica que se quiere formalizar.
- **Instancias.** Cuando queremos definir un concepto concreto y preciso usamos una instanciación del mismo.
- **Relaciones.** Los conceptos pueden estar relacionados entre sí, y representa la interacción y el enlace entre varios conceptos.
- **Funciones.** Operaciones que permiten definir el tipo concreto de relación que se aplica entre dos o más conceptos.
- **Axiomas.** Para cada grupo o subgrupo de relaciones, se pueden establecer una serie de funciones lógicas entre ellas, tales como pertenencia, relación, parte de, subgrupo de etc.

2.3.4. Linked Data

Los datos enlazados o *Linked Data* es un método para publicar datos estructurados utilizando vocabularios variados y complementarios como *schema.org*, *skos*[32] o *dcterms*[33] que pueden ser conectados entre sí e interpretados por máquinas.

[32]Simple Knowledge Organization System: `https://skos.um.es/`

[33]Dublin Core terms: `http://www.dublincore.org/specifications/dublin-core/dcmi-terms/`

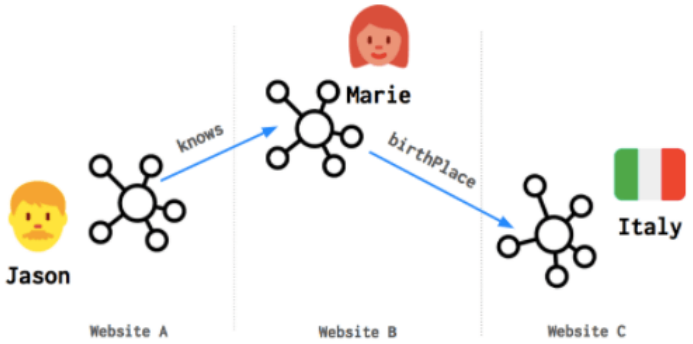

Figura 2.7: Datos estructurados y conectados con *Linked Data*.

Utilizando datos enlazados, las sentencias codificadas en tres partes pueden ser distribuidas a través de diferentes sitios web. En el sitio web *A* podemos presentar la entidad *Jason* y el hecho de que conoce a *Marie*. En el sitio web *B* podemos proporcionar toda la información sobre *Marie* y en el sitio web *C* podemos encontrar información sobre el lugar de nacimiento de *Marie* (ver Figura 2.7). Cada página contiene los datos estructurados para describir una entidad y el enlace a la entidad que podría ser descrito en un sitio web diferente. En 2006 *Tim Berners-Lee* [Bizer et al., 2011] describió los datos enlazados de la siguiente manera:

> *" La "web semántica" no se trata sólo de poner datos en la web. Se trata de hacer enlaces, para que una persona o máquina pueda explorar la red de datos. Con los datos enlazados, cuando tienes algunos de ellos, puedes encontrar otros datos relacionados."*

La computación de datos enlazados describe un método de publicación y enlace de datos provenientes de fuentes heterogéneas que pueden ser interconectadas y compartidas entre sí, además de descubiertas de forma programática.

Linked Data se basa en tecnologías web estándar como HTTP y URI, pero en lugar de utilizarlas para servir páginas web para lectores humanos, las amplía para compartir información de forma que puedan ser leídas automáticamente por los ordenadores. Esto permite conectar y consultar datos de diferentes fuentes.

Las consultas sobre datos enlazados se realizan utilizando un lenguaje de consulta semántico llamado SPARQL que permite recuperar y manipu-

lar datos almacenados en formato *Resource Description Framework* (*RDF*). *Linked Data* sigue cuatro principios de diseño:

- Uso de URIs (*Uniform Resource Identifiers*) como nombre para los términos definidos.
- Uso de HTTP URIs para que las entidades puedan enlazar con esos nombres.
- Cuando alguien consulta una URI, esta provee de mucha información útil utilizando los estándares propuestos para la "web semántica" (*RDF*, *OWL*, SPAQL, etc.).
- Incluir enlaces a otras URIs de modo que se puedan descubrir más cosas o vocabularios de otros dominios.

Además de los principios, tiene también un valor añadido identificado por los siguientes aspectos tales como:

- **Integración de datos flexible.** Facilita la integración de datos y permite la interconexión de diferentes conjuntos de datos, vocabularios y dominios de conocimiento.
- **Incremento en la calidad de los datos.** Es necesario que estos tengan calidad. Con *Linked Data* se persigue que los datos que se tengan estén sujetos a un conjunto de relaciones e interconexión que mejoran la calidad del significado y la exploración de los mismos.
- **Nuevos servicios.** Permite crear nuevas definiciones de todo tipo de servicios. Con ello es posible convertir cualquier dominio de conocimiento en una descripción de datos semánticos.
- **Reducción de costes.** La reutilización de los vocabularios es un aspecto fundamental. Esto hace que no se tenga que re-diseñar algo que ya existe de modo que esto supone una mejora en la reducción de tiempo y coste de desarrollo de estas tecnologías semánticas.

Capítulo 3

Semántica para la definición de servicios de minería de datos en Cloud Computing

Motivación

En los últimos años, con el auge del Cloud Computing, muchas empresas que ofrecen servicios en la nube, están potenciando una nueva serie de servicios en su catálogo, como la minería de datos y el aprendizaje automático, aprovechando los enormes recursos informáticos de los que disponen. Se han presentado diferentes propuestas de definición de servicios para abordar el problema de describir los servicios genéricos y específicos en Cloud Computing de forma exhaustiva. Teniendo en cuenta que cada proveedor tiene su propia definición de la lógica de sus servicios, cabe señalar que la posibilidad de describir los servicios de forma flexible entre proveedores es fundamental para mantener la usabilidad y portabilidad de este tipo de servicios.

En este capítulo se desarrolla una propuesta de esquema basada en tecnologías semánticas para la definición de servicios de minería de datos en Cloud Computing llamada `dmcc-schema`. Contempla todos los aspectos clave de Cloud Computing como precios, autenticación, SLA, recursos de computación o flujo de trabajo en minería de datos, entre otros. Con el fin de validar la efectividad del esquema desarrollado, se han creado varios casos de uso con definiciones de servicios de minería de datos de proveedores reales de Cloud Computing y se han definido una serie de servicios que incluyen algoritmos

como *Random Forest* o *KMeans*, entre otros.

3.1. Introducción

El incremento de la capacidad de cómputo disponible, las necesidades de los usuarios y la demanda de servicios de Cloud Computing está haciendo que gran parte de procesamiento de información, aplicaciones, servicios y en general todo lo relativo a nuestra vida digital se esté migrando hacía este tipo de entornos en la nube. La integración de Cloud Computing en nuestra relación con las Tecnologías de Información y las Comunicaciones ha sido completamente transparente, y además ha supuesto una reducción considerable de la problemática que otros modelos de computación tienen [Liu, 2016].

Utilizamos dispositivos, tales como tabletas, teléfonos móviles, ordenadores personales, etc. conectados a Internet de forma casi continua, consumiendo servicios, aplicaciones, recursos y datos que físicamente ni siquiera están en nuestros dispositivos, permitiendo que gran parte de la computación se realice en los proveedores de Cloud Computing. El incremento en el número de dispositivos conectados a Internet crece casi de forma exponencial cada año suponiendo un aumento del número de usuarios que consumen servicios de Cloud Computing [Marr, 2016].

La adopción del fenómeno de Cloud Computing ha supuesto un nuevo paradigma en el modo en que las Tecnologías de la Información y los servicios son consumidos, explorados y desplegados. Tal y como se indica en las sección 2.1, Cloud Computing es un modelo de proveer servicios a empresas, instituciones o usuarios finales siguiendo el concepto de «utility», tal como ocurre con servicios como el gas o la electricidad [JoSEP et al., 2010]. Este modelo se basa en el pago por volumen de uso, de modo que el cliente está sujeto a un determinado pago en función del volumen del servicio que use en cada momento. Además, el modelo no solo se basa en la gestión de costes, también considera diferentes aspectos relacionados las licencias de uso, acuerdos en base al contrato, limitaciones, etc. que el cliente debe considerar a la hora de contratar el servicio. Si nos centramos en el modelo de «utility», podemos confirmar que Cloud Computing supone una forma de provisión de servicios tal que los recursos de cómputo son contratados por los clientes o usuarios a través de servicios en la Web. Los proveedores de Cloud Computing poseen una enorme capacidad de cómputo distribuida a lo largo de todo

el mundo, mediante la cual son capaces de ofrecer múltiples tipos de servicios a empresas, organizaciones y usuarios [Bhardwaj et al., 2010].

Hoy en día nos encontramos en la era de la información, donde ingentes cantidades de datos son generados cada segundo desde dispositivos móviles, aplicaciones, equipamientos industriales, transportes, o redes sociales entre muchos otros. Este volumen de información crece de forma extremadamente grande y seguirá haciéndolo durante los próximos años. Según se estima en la revista Forbes [Marr, 2016], para 2020 el crecimiento en los datos generados por empresas y organizaciones seguirá en auge y la cantidad de datos generados se incrementará en un 4.300 %. Parte de este incremento está motivado por el incremento de elementos de sensorización (*Internet de las cosas* o IoT [Doukas and Maglogiannis, 2012]), dispositivos móviles y sobre todo la facilidad de conexión e intercambio de información y datos que existe hoy en día gracias a Internet. En relación a los dispositivos móviles, según el estudio presentado por *Gartner* [Gartner Inc., 2016], estima que para después de 2020 el volumen de dispositivos móviles conectados a internet superará los 25.000 millones, y desbordará la cifra de los 44.000 millones de GigaBytes de datos generados para cada año.

Con este volumen de información generado desde cientos de miles de fuentes de datos, es lícito hacerse las siguientes preguntas: *¿Es posible realizar un procesamiento de los datos, para obtener o extraer conocimiento de los mismos?*, *¿existen herramientas capaces de procesar tales cantidades de datos?* y sobre todo, *¿cómo los servicios de Cloud Computing pueden aportar valor añadido a este procesamiento?*, esto en realidad supone un reto importante, pues toda esa información generada desde dispositivos, aplicaciones o servicios necesita ser procesada de forma eficiente. En este escenario, el modelo de Cloud Computing puede ser considerado un actor fundamental, debido a sus características inherentes tales como el escalado, la flexibilidad y capacidad de cómputo prácticamente ilimitada, por lo que ofrece todos los elementos deseables para llevar a cabo estas tareas.

Desde hace más de 5 años los proveedores de Cloud Computing han venido integrando dentro de su catálogo de servicios, un subgrupo destinados al análisis de datos y al procesamiento masivo de los mismos [Talia, 2013]. Este subgrupo de servicios ofrece a los usuarios un conjunto de diferentes herramientas, plataformas y APIs listas para ser consumidas como servicios de Cloud Computing para la resolución de problemas de minería de datos. La minería de datos es una herramienta muy efectiva para el análisis de datos desde múltiples enfoques y para la extracción de información de los datos. Aspectos tales como clasificación de datos, categorización, predicción

o búsqueda de patrones, son parte de los problemas que puede resolver la minería de datos.

La tendencia actual de migrar todos los entornos de trabajo a ambientes de Cloud Computing no ha dejado atrás a las herramientas y plataformas de minería de datos en Cloud Computing. Es un área muy competitiva para el análisis de datos, en gran medida dictado por la gran variedad, capacidad y cantidad de recursos que ofrece para el trabajo con los datos a costes relativamente bajos comparados con otras soluciones [Hashem et al., 2015]. Estas herramientas incluyen modelos estadísticos, algoritmos matemáticos, técnicas y métodos de aprendizaje máquina (tales como redes neuronales o arboles de decisión, entre otros), técnicas de pre-procesamiento y otros elementos clave del análisis y la predicción de datos [Assunção et al., 2015]. El interés suscitado por el uso de este tipo de técnicas en sectores muy diversos como banca, administración, farmacia, medicina o aseguradoras, entre muchos otros, indica la necesidad que tiene la industria en mejorar su investigación, optimizar sus procesos o predecir mejores resultados sobre sus datos, por ejemplo.

La prestación de los servicios de minería de datos por parte de los proveedores de Cloud Computing se realiza de diferentes modos. Proveedores y servicios tales como *Amazon Sage Maker* [1], o *Microsoft Azure Studio*[2], ofrecen un conjunto de algoritmos de minería de datos, listos para ser utilizados desde una API web. Este modelo, permite automatizar e integrar el proceso de análisis de los datos a través de llamadas al servicio de modo que se ejecute el algoritmo en cuestión y se obtengan los resultados en cada llamada. Además, proveedores de Cloud Computing y servicios concretos de minería de datos como *Amazon ML*[3], *Google Cloud ML*[4] o *IBM Machine Learning*[5], ofrecen herramientas y plataformas completas para:

a) realizar el prototipo del análisis de los datos,
b) entrenar el modelo, y por último,
c) desplegar el modelo en un servicio web directamente consumible por los usuarios.

Además, existe otro grupo de servicios de minería de datos de más alto ni-

[1] `https://aws.amazon.com/sagemaker/`
[2] `https://azure.microsoft.com/en-us/services/machine-learning-studio/`
[3] `https://aws.amazon.com/machine-learning/`
[4] `https://cloud.google.com/products/machine-learning/`
[5] `https://www.ibm.com/analytics/machine-learning`

vel que incluyen paquetes completos que resuelven problemas muy concretos, como podrían ser, la clasificación de imágenes (visión por computador), el análisis de sentimientos en textos, el análisis de vídeo, la detección de objetos o el procesamiento de lenguaje natural. Estos servicios trabajan con «bundles» (paquetes de análisis de datos completos) y pueden encontrarse en diferentes proveedores de Cloud Computing como *Algoritmia*[6] [Tafti et al., 2017] o *Amazon Recognition*[7], por ejemplo.

Los proveedores de servicios de Cloud Computing son muy reticentes a la hora de exponer sus servicios de un modo más o menos estándar. Si tratamos la cuestión dentro de ámbito de los servicios de minería de datos para Cloud Computing, ocurre algo similar a los demás servicios entre proveedores; cada servicio y proveedor de servicios, probablemente tenga una descripción y definición de sus servicios ajustada a sus necesidades y además esta definición sea incompatible con la de los demás proveedores de Cloud. Esto hace que, por ejemplo, una definición de servicio para un algoritmo *Random Forest* [Liaw et al., 2002] en un proveedor de Cloud Computing como *Amazon Sage Maker*, no sea compatible o no se ajuste al servicio propuesto por otra plataforma como la de *Microsoft Azure Studio*. Elementos como el nombre de los parámetros del algoritmo, los valores por defecto o las características del algoritmo pueden ser definidas de un modo dentro de un proveedor de servicios, y otro proveedor puede definir esos elementos de otra modo, aunque realmente ambas serían a efectos prácticos la misma definición de algoritmo o funcionalidad que se espera. Esto realmente supone un problema considerable, pues impide definir servicios o modelos de servicios independientes del proveedor, y disminuye la capacidad de poder comparar servicios entre proveedores a todos los niveles, no únicamente centrados en la funcionalidad o la prestación de servicios [Lin et al., 2018].

Una propuesta para la estandarización en la definición de los servicios de minería de datos entre proveedores de Cloud Computing mejoraría la competitividad del mercado en cuanto al consumo de estos servicios, permitiendo a terceras entidades operar con los servicios de un modo transparente, abstrayendo las peculiaridades de las definiciones que ofrecen cada uno de los proveedores [Ghazouani and Slimani, 2017]. Este aspecto es fundamental cuando se considera, por ejemplo, la figura del *Broker* [Grivas et al., 2010] de servicios de Cloud Computing, de modo que una homogeneización en la descripción de los servicios, permitiría a este tipo de sistemas una mejor gestión de la cartera de proveedores y servicios a través de los cuales ofrece una

[6] `https://algorithmia.com/`

[7] `https://aws.amazon.com/rekognition/`

intermediación entre proveedor y consumidor de servicios.

La definición de servicios no es una tarea sencilla. Existen una gran cantidad de propuestas para la definición de servicios genéricos de diferente tipo, tanto a nivel sintáctico como a nivel semántico. La definición de servicios de minería de datos en Cloud Computing está poco cubierta, puesto que es un área bastante nueva en su aplicación para Cloud Computing y no existe una propuesta final que permita representar un servicio de minería de datos al completo siguiendo todos los aspectos clave de un servicio en Cloud y además del proceso concreto de análisis de datos. En los últimos años la definición semántica de servicios web ha ido ganando terreno a otro tipo de definiciones, como las sintácticas, en gran parte debido a la flexibilidad que tiene la tecnología semántica para adaptarse a múltiples dominios. Una propuesta semántica, junto con las directrices de *Linked Data* [Bizer et al., 2009] (ver sección 2.3.4), permite una mejora en la integración de la definición de los servicios, tomando los modelos y estructuras de la web semántica [Berners-Lee et al., 2001] y habilitando enlaces a datos y descripciones provenientes de otros muchos dominios mediante el uso de lenguajes semánticos como *RDF* [Klyne and Carroll, 2004], *JSON-LD* [Lanthaler and Gütl, 2012] o *Turtle* [Beckett et al., 2014]. En este capítulo se tratará en profundidad la definición de servicios de minería de datos utilizando tecnología semántica y siguiendo las recomendaciones de *Linked Data*.

Cuando se define un servicio de Cloud Computing, éste se puede separar en dos partes. Una parte relativa a todo lo que representa la funcionalidad del servicio, por tanto, el flujo de trabajo con algoritmos, y el procesado de datos. La segunda parte en la que se compone la definición del servicio es la relativa a toda la gestión del servicio en sí; es decir, cuando utilizamos un servicio, el proveedor considera aspectos adicionales para dar completitud al servicio ofertado, estando más alineados con lo que es el mantenimiento o el modelo de negocio del propio servicio. La definición de servicios de minería de datos que mostramos en este trabajo no solo se centra en la funcionalidad del servicio como tal: minería de datos (algoritmos, parámetros, o modelos, entre otros), sino que también permite definir y modelar elementos fundamentales como la tarificación del servicio, la autenticación, el SLA, los recursos de computación o el catálogo de servicios ofertado, entre otros.

Para poder llevar a cabo esta definición de servicios que integre todos los aspectos clave de un servicio de Cloud Computing para minería de datos hemos desarrollado una propuesta llamada `dmcc-schema`, que utiliza tecnología semántica y aprovecha la propuesta de *Linked Data* para permitir el intercambio, portabilidad, descubrimiento e integración de este tipo de servicios.

El trabajo realizado con `dmcc-schema` ha supuesto un estudio pormenorizado de las propiedades de los servicios de minería de datos de varios de los proveedores de Cloud Computing más conocidos, tales como *Amazon*, *Microsoft* o *Google*. De este estudio se ha extraído toda la información necesaria para cubrir las peculiaridades de este tipo de servicios entre proveedores, junto con su oferta de servicios de minería de datos. Con esta información se ha conseguido definir un esquema que facilita la definición de servicios de minería de datos de forma genérica y ofrece una homogeneidad para el trabajo con las diferentes características de la definición de los servicios por parte de los proveedores. De modo que con una única definición del servicio es posible aglutinar gran parte de los detalles de especificación de los servicios de esta tipología, siendo por ejemplo una herramienta muy adecuada para la comparación de servicios de minería de datos desde múltiples puntos de vista, tales como los costes de ejecución del servicio, el catálogo de algoritmos disponibles como servicio, las restricciones asociadas al SLA o las características de las instancias, por ejemplo.

3.2. Trabajos relacionados previos

En la actualidad, dentro de los ámbitos académicos e industriales existe un interés creciente en la minería de datos y más concretamente en todo lo relacionado con la aplicación de técnicas de Inteligencia Artificial a prácticamente cualquier tipo de problema. La aplicación de esta tecnología en empresas e instituciones se realiza a través de herramientas, algoritmos y plataformas para el análisis y el procesado de la información, con el objetivo de extraer conocimiento de los datos que disponen. La idea subyacente es llevar las capacidades "inteligentes" al siguiente nivel, y permitir, por ejemplo, la habilidad de predecir cual será el mejor cliente para un producto determinado, como optimizar procesos industriales, o como clasificar y detectar elementos en vídeo o imágenes. Las empresas quieren capitalizar este valioso recurso utilizando análisis predictivos para obtener una información extra que pueden aplicar para optimizar su inteligencia de negocio (*Business Intelligence*).

La expansión reciente de Cloud Computing y esta tendencia en auge sobre la necesidad de analizar datos y extraer conocimiento en los últimos años, ha hecho que los proveedores de Cloud adopten en su catálogo servicios de minería de datos. La disponibilidad de análisis predictivos en la nube ya es significativa [Guazzelli et al., 2009a], y ha permitido que esta tecnología, que

normalmente ha sido difícil de poner en marcha debido a su complejidad, coste, requisitos de computación, etc., sea accesible a cada vez más organizaciones y usuarios finales que, de otro modo, no estarían disponibles. Estos servicios de minería de datos en Cloud están sirviendo para desplegar elementos de análisis de datos dentro de sus entornos de trabajo ofreciendo un alto grado de integración.

Los servicios de minería de datos ofertados por los proveedores de Cloud Computing requieren una especificación completa, que no sólo se limite a aspectos técnicos del propio servicio, también debe considerar aspectos clave en la funcionalidad y la gestión de Cloud Computing junto otros elementos derivados de la relación *cliente−servicio−proveedor* a todos los niveles. La definición de servicios y modelos de servicios ha sido tratada desde diferentes perspectivas. En los siguientes párrafos se van a detallar parte de las propuestas más destacadas para la definición y descripción de servicios.

En el plano de definición sintáctica de servicios hay que considerar como referencia clave, los modelos basados en *Service Oriented Architecture* (*SoA*) [Newcomer and Lomow, 2005] y *XML* (*eXtended Markup Language*) como los más extendidos y ampliamente utilizados en los últimos años. Derivados de estos modelos y de otros lenguajes de referencia, han surgido múltiples propuestas adicionales para la definición de servicios web. Lenguajes como *WSDL* [Juric, 2007], *WADL* [Hadley, 2006], *SoAML* [Elvesæter et al., 2011] o *UUID* [Leach et al., 2005] (ambos relacionados con el lenguaje unificado de modelado *UML* [D'souza and Wills, 1998]) permiten especificar a nivel técnico la descripción servicios como un conjunto de «EndPoints»[8] a través de los cuales se operan mensajes que contienen información orientada a documentos o a procedimientos (funciones).

Por otro lado la incorporación de tecnología semántica para la definición de servicios, ha permitido que se pueda llevar a cabo un modelado de servicios mucho más flexible que el desarrollado con los modelos de descripción sintácticos [Martin et al., 2007]. Esto es debido a la propia naturaleza de cada uno de los modelos, donde el modelado sintáctico está más limitado que el modelado semántico, ya que este último puede expresar con más riqueza todo lo que envuelve a un servicio y no sólo limitarse a la parte técnica o sintáctica. La tecnología semántica permite capturar todas las características funcionales y no funcionales, por lo que aumenta el abanico de posibilidades de definición y descripción que ofrecen los modelos sintácticos.

[8]Un «EndPoint», es una interfaz de interconexión entre servicio y usuario, a través de la cual es posible interactuar con una o varias funcionalidades del servicio, intercambiando mensajes.

Dentro de las propuestas de lenguajes semánticos podemos destacar como base para modelar servicios, *OWL-S* [Martin et al., 2004] que permite integrar aspectos clave como descubrimiento de servicios, modelado de procesos o parte de los detalles de interacción con los servicios. Con *WSMO* [Domingue et al., 2005] es posible modelar servicios web y describir de forma semántica elementos fundamentales como el descubrimiento, invocación, y otros aspectos lógicos que forman parte del servicio y su descripción. Una comparación conceptual que identifica las superposiciones entre *OWL-S* y *WSMO* para evaluar su aplicabilidad en un entorno real, además de su potencial para convertirse en posibles estándares ha sido estudiada en [Lara et al., 2004]. Otros esquemas de definición de servicios más completos, como *SA-WSDL* [Fredj et al., 2008], ofrecen todo lo que *WSDL* es capaz de describir a nivel técnico y además admite anotaciones semánticas sobre los elementos sintácticos que expresa *WDSL*. Con *SA-REST* [Sheth et al., 2007] se demuestra que el uso de micro-formatos[9] para la anotación semántica de servicios *RESTful* y el uso de dichos servicios semánticamente ofrecen un mejor soporte en cuanto a interoperabilidad en los servicios. La propuesta *WSMO-lite* [Vitvar et al., 2008] permite mejorar las descripciones de servicio existentes, añadiendo las capacidades de *SoA* con una integración inteligente y automatizada. Otra propuesta como *MSM* [Taheriyan et al., 2012] (*Minimal Sevice Model*) tiene la capacidad de expresar intercambio, automatización y composición de servicios a nivel semántico con el enfoque en el servicio final.

Otros esquemas o lenguajes para la definición de servicios basados en tecnología semántica como *USDL* [Kona et al., 2009] consideran una visión *holística* de la descripción de servicios, pero con una importante parte dedicada al modelo de negocio que envuelve al servicio y que sin esta parte no estaría completo un servicio en Cloud Computing. *USDL* considera aspectos como precios, entidades, relaciones comerciales, acuerdos legales y consideraciones técnicas. Posteriormente como una mejora de *USDL* se desarrolla *Linked-USDL* [Pedrinaci et al., 2014], el cual permite modelar servicios genéricos de Cloud Computing, siguiendo gran parte del modelo de *USDL* pero añadiendo una extenso conjunto de componentes para modelar el servicio en Cloud, tales como gestión de los datos del servicio o negocio, las interfaces, la interacción, o aspectos como *SLA*, además la ventaja de esta nueva aproximación a la definición de servicios es que sigue las recomendaciones de la propuesta de *Linked Data*. Gracias a *Linked Data* este esquema para la definición de servicios, puede ser extendido con otras propuestas semánticas de modo que

[9] Anotaciones semánticas que utilizan etiquetas HTML/XHTML para incluir metadatos adicionales y otros atributos en las webs.

se tiene una definición de servicio mucho más abierta y flexible.

Gran parte de estas propuestas basadas en lenguajes semánticos no consideran todos los aspectos clave de la definición de los servicios de Cloud Computing para minería de datos, ya que se centran mas en los aspectos genéricos de definición de servicios, por lo que no tienen en cuenta la descripción de los componentes de un modelado de trabajo con minería de datos.

La práctica totalidad de las herramientas más utilizadas hoy en día para el análisis de datos tales como *KNIME* [et al., 2007],
Weka [Hall et al., 2009a] y Orange [Demšar et al., 2013], así como otras plataformas y bibliotecas para el procesamiento de datos con los lenguajes más extendidos actualmente como *C/C++*, *Java*, *Python*, *R*, o *Scala*, carecen de las capacidades básicas para la definición de servicios de minería de datos para Cloud Computing. Este tipo de aplicaciones y plataformas no están diseñadas para ofrecer servicios para Cloud Computing con todos los elementos mínimos que son requeridos, tanto para la parte de trabajo con minería de datos, como también en la parte de gestión del servicio de Cloud Computing.

Para solventar el problema de los flujos de trabajo con minería de datos, se han desarrollado diferentes propuestas desde la perspectiva semántica. Por ejemplo, *Exposè* [Vanschoren and Soldatova, 2010] permite describir el flujo de trabajo de experimentación [Marozzo et al., 2018]. Con *OntoDM* [Panov et al., 2008] y utilizando *BFO* (Basic Formal Ontology), se pretende crear un marco para la experimentación y replicación de resultados en minería de datos (también llamado, ciencia o experimentación reproducible). En *MEXcore* [Esteves et al., 2015b] y *MEXalgo* [Esteves et al., 2015a] se realiza una especificación completa del proceso de descripción de la experimentación en problemas de minería de datos, además, junto con *MEXperf* [Esteves et al., 2015b] que añade medidas de rendimiento a la experimentación, se completa la definición de este tipo de esquema. Otro enfoque similar al proporcionado por *MEX* es el que podemos encontrar en *ML-Schema* [Esteves et al., 2016], que intenta establecer un esquema estandarizado para algoritmos, datos y experimentación. *ML-Schema* considera términos como *Tarea*, *Algoritmo*, *Implementación*, *Hiper-parámetros*, *Datos*, *Características* o *Modelos*, entre muchos otros elementos adicionales. Las propuestas más recientes intentan unificar diferentes esquemas y vocabularios semánticos desarrollados previamente siguiendo las directivas de *Linked Data*.

Con *Linked Data* es posible reutilizar vocabularios, esquemas y otros conceptos. Esto enriquece significativamente la definición del esquema, permitiendo crear una definición completa del modelo basada en otros esquemas y

vocabularios existentes. Con *Linked-USDL* [Berners-Lee et al., 2001], *Mex**, *ML-Schema*, *OntoDM* y *Exposé*, es posible crear flujos de trabajo completos para minería de datos y además incluir otros vocabularios adicionales que completarían una descripción con *Linked Data*, para conseguir crear un definición integral de servicio que contenga tanto la parte de gestión de Cloud Computing como de composición de la funcionalidad específica del servicio.

En esta revisión de los trabajos relacionados previos se ha estudiado gran parte de la investigación relacionada con los aspectos de la definición de servicios desde diferentes perspectivas y con un enfoque mucho más especifico hacía el ámbito de la minería de datos. El estudio de estos trabajos nos permite extraer cuales son los nichos clave a la hora de desarrollar una nueva propuesta para la definición y descripción de servicios minería de datos para Cloud Computing.

3.3. Propuesta semántica para la definición de servicios de minería de datos con `dmcc-schema`

La web semántica aplicada a la definición de servicios de Cloud Computing, permite realizar tareas de negociación, composición e invocación con un alto grado de automatización. Esta automatización, basada en diferentes conceptos de la tecnología semántica, es fundamental en Cloud Computing ya que permite descubrir y explorar servicios para su consumo por parte de otras entidades utilizando todo el potencial que ofrecen lenguajes semánticos como *RDF* y lenguajes de consulta semántica como SPARQL [Prud et al., 2006]. Adicionalmente, si consideramos *Linked Data* como parte integradora de la definición de servicios, obtendremos un conjunto creciente de esquemas y vocabularios reutilizables para la definición de servicios de Cloud Computing, en la cual no sólo considera los aspectos base de una definición sino que también es capaz de extender su dominio, a prácticamente cualquier otro domino, sin más que incluir o enlazar con otros esquemas o vocabularios.

En este capítulo detallamos la propuesta `dmcc-schema`, un esquema y un conjunto de vocabularios que ha sido diseñado para abordar el problema de describir y definir los servicios de minería de datos en Cloud Computing de una manera integral. No sólo se centra en resolver el problema específico del modelado, con la definición de flujo de trabajo y algoritmos (es decir,

toda la parte relacionada con la minería de datos), sino que también incluye los aspectos principales de un servicio de la gestión de Cloud Computing. Un conjunto de vocabularios existentes se han integrado en el esquema de `dmcc-schema` y de forma adicional se han creado nuevos vocabularios *ad-hoc* para cubrir ciertos aspectos que no son implementados por otros esquemas externos. Los vocabularios han sido reutilizados siguiendo las recomendaciones propuestas por *Linked Data* y considerando todas las partes importantes como la definición de experimentos y algoritmos, así como la interacción o autenticación que han sido definidas en otros vocabularios específicamente.

3.3.1. Esquema `dmcc-schema`

Cuando nos referimos a un servicio de Cloud Computing, NIST[10] establece una serie aspectos que debe tener para su especificación completa:

1. **Autenticación**. El servicio o servicios requieren acceso autenticado.
2. **Catálogo**. El proveedor tiene un catálogo de servicios listos para ser descubiertos y utilizados.
3. **Entidades**. Los servicios interactúan entre entidades que ofrecen o consumen servicios.
4. **Interacción**. Puntos de acceso e interfaces para el consumo de servicios para usuarios y entidades.
5. **Precios**. Los servicios ofrecidos tienen un costo y están sujetos a una determinada tarificación.
6. **SLA/SLO.** Los servicios tienen SLA y objetivo de nivel de servicio (*SLO*).

No existe una estandarización sobre qué elementos debe tener un servicio en Cloud Computing para su definición completa, pero de acuerdo con NIST, debe cumplir con aspectos como el autoservicio (descubrimiento) o la medición (precios, *SLA*), entre otros. Para el desarrollo del `dmcc-schema`, se ha llevado a cabo un estudio exhaustivo de las características y servicios disponibles en las plataformas de servicios de minería de datos.

La tabla 3.1 contiene información sobre los servicios de minería de datos analizados de un subconjunto de proveedores de Cloud Computing; proveedores líderes como *Amazon*, *Microsoft*, *Google*, *IBM* y *Algorithmia*, han sido

[10]National Institute of Standards and Technology, `www.nist.gov`

estudiados en profundidad en aspectos como la gestión del *SLA*, el precio de las diferentes variantes de uso/consumo, el catálogo de servicios (y algoritmos que implementan), los métodos de interacción con sus servicios y la autenticación.

Proveedor	**Nombre del servicio para minería de datos**
Google	Cloud Machine Learning Engine
Amazon	Amazon SageMaker, Amazon Machine Learning
IBM	Watson Machine Learning, Data Science
Microsoft Azure	Machine Learning Studio
Algorithmia	Algorithms bundles

Tabla 3.1: Plataformas y servicios de minería de datos para Cloud Computing analizados para `dmcc-schema`.

Nuestro esquema se ha complementado reutilizando vocabularios externos: para la interacción con el servicio, utilizando *schema.org* [Guha et al., 2015], autenticación, con *Web API Authentication* (*waa*) [Maleshkova et al., 2010] diseño de precios, con *GoodRelations* [Hepp, 2008] y experimentación y algoritmos de minería de datos, utilizando el vocabulario concreto de `dmcc-schema`. En la tabla 3.2 se muestran los vocabularios reutilizados por `dmcc-schema` para definir cada módulo del servicio completo. Utilizando `dmcc-schema` en la descripción de servicios es posible definir tanto la parte del flujo de trabajo de los servicios de minería de datos, como todo lo relacionado con la gestión de un servicio genérico de Cloud Computing, en una única definición más compacta, directa y sencilla que las otras propuestas.

Módulo	**Vocabularios**
SLA	`gr`, `schema`, `ccsla`
Regiones	`ccregion`
Instancias	`ccinstances`
Precios	`gr`, `schema`, `ccpricing`, `ccinstance`, `ccregion`
Autenticación	`waa`
Interacción	`schema`
minería de datos	`mls`, `ccdm`
Proveedor de servicios	`gr`, `schema`

Tabla 3.2: Vocabularios reutilizados por `dmcc-schema`.

La propuesta está diseñada para servir como un punto de unión entre la gestión de servicios y la experimentación en minería de datos, elementos que

	dmcc	mls	Exposè	MEX	Linked-USDL
Tareas	●	●	●	●	-
Implementación	●	●	●	-	-
Datos	●	●	-	●	-
Modelo	●	●	●	●	●
Autenticación	●	-	-	-	●
Precios	●	-	-	-	●
SLA	●	-	-	-	●
Instancias	●	-	-	-	-
Regiones	●	-	-	-	-

Tabla 3.3: Características incluidas en el esquema `dmcc-schema` comparadas con las otras propuestas.

las demás propuestas citadas en la sección 3.2 no tienen plenamente en cuenta. Como se muestra en la tabla 3.3, todas las características proporcionadas por `dmcc-schema` son comparadas con parte de las propuestas (ver sección 3.2) que están en la misma línea sobre la definición de minería de datos y/o el modelado de servicios genéricos para Cloud Computing. Como se ha señalado, la propuesta considera elementos como *SLA*, los recursos informáticos (instancias), las regiones o la autenticación, junto con la experimentación en minería de datos, mientras que las otras propuestas no lo contemplan.

El esquema `dmcc-schema` es ideal para estructurar los datos e información de los servicios de minería de datos de forma coherente, normalizando las propiedades y los principales conceptos y entidades que forman parte de Cloud Computing para los diferentes proveedores de este tipo de servicios.

Con respecto a los beneficios de usar `dmcc-schema` sobre otros esquemas, podemos destacar lo siguiente:

1. ofrece una estructura de datos flexible que puede integrar las propiedades de los servicios de minería de datos existentes,
2. unifica diferentes esquemas en uno solo, ofrece menos fricción y más integración de los diferentes esquemas incluidos, y por último
3. comprende una solución "todo en uno" para la definición de servicios de minería de datos en Cloud Computing.

Dentro del contexto que ofrece `dmcc-schema`, se ha tratado de enfatizar la independencia de la plataforma de Cloud, es decir, `dmcc-schema` no define

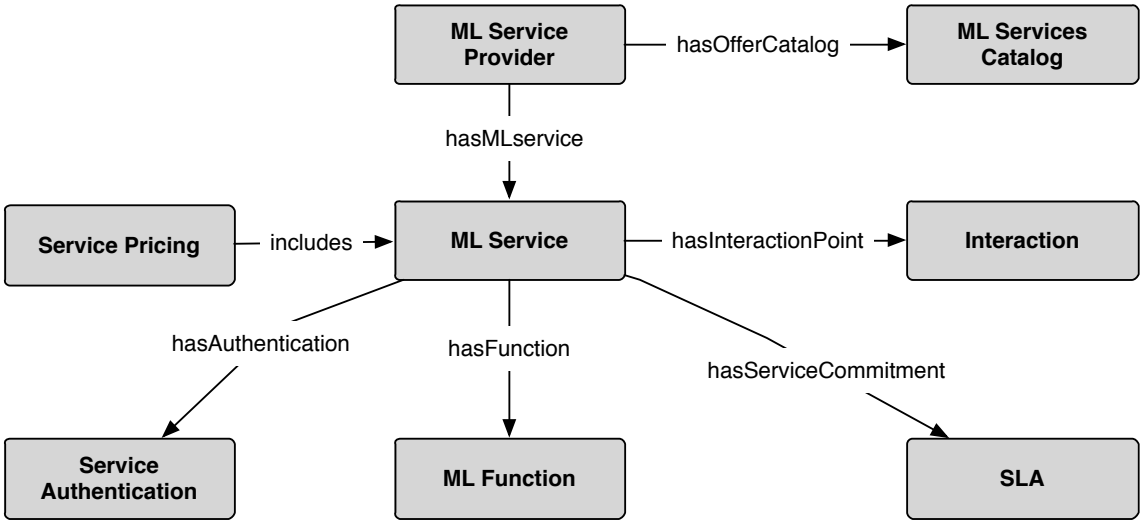

Figura 3.1: Clases principales y relaciones entre los módulos de `dmcc-schema`.

aspectos de despliegue de servicios o elementos más cercanos a las herramientas de implementación o la plataforma física o virtual donde se implantará el servicio. La definición y descripción que ofrece `dmcc-schema` está al más alto nivel y sólo aborda los aspectos de definición y modelado del servicio sin entrar en los detalles del despliegue de infraestructura, por ejemplo.

El diagrama de alto nivel de las entidades del `dmcc-schema` se puede ver en la Figura 3.1. En las siguientes subsecciones se detallarán cada uno de los componentes de `dmcc-schema`.

3.3.2. Autenticación

Hoy en día, la seguridad en los sistemas informáticos y en particular en las plataformas de Cloud Computing es un aspecto que debe tomarse muy en serio a la hora de desarrollar servicios y aplicaciones de Cloud Computing. De este modo, un servicio fiable y robusto también debe estar protegido contra posibles accesos no autorizados. En la capa exterior de seguridad dentro del modelo de consumo de servicios de Cloud Computing, la autenticación debe considerarse como parte fundamental de una definición de servicio Cloud Computing, sin la cual no es posible utilizar, acceder y consumir esos servicios.

La autenticación en plataformas Cloud Computing cubre una amplia gama de posibilidades. Cabe señalar que para la gran mayoría de los servicios,

la opción más utilizada para gestionar el acceso de los usuarios a los servicios es mediante una `API Key` o bien `OAuth`, junto con otros mecanismos adicionales.

En nuestro caso se ha optado para el modelado de la autenticación, la reutilización del vocabulario
waa (`waa:WebApiAuthentication`) [Maleshkova et al., 2010]. Este esquema permite modelar muchos de los sistemas de autenticación disponibles de una forma sencilla. En la Figura 3.2 se pueden ver todos los componentes del esquema de autenticación `waa:`.

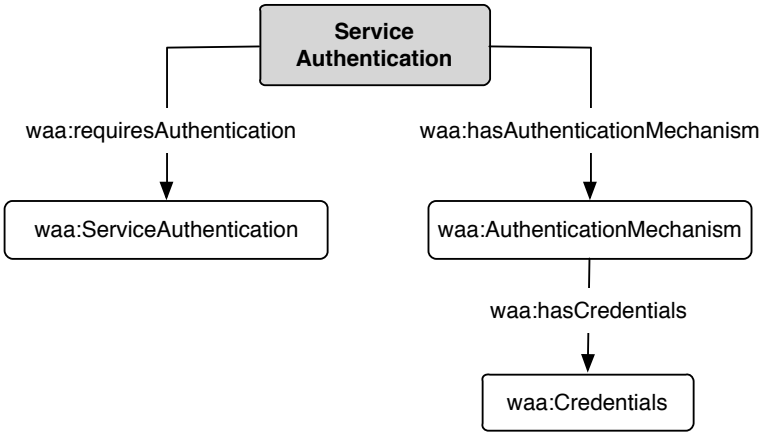

Figura 3.2: Esquemas de autenticación ofrecidos por `waa:`.

3.3.3. Servicio de minería de datos y experimentación

Para la parte principal del servicio, donde se especifica y modela la experimentación y ejecución de algoritmos, se han reutilizado algunas partes esenciales de *ML-Schema*. Otros esquemas que se han considerado a la hora de modelar los servicios específicos de minería de datos como *MEXcore*, *OntoDM*, *DMOP* o *Exposé* también proporcionan una abstracción adecuada para modelar el servicio, pero son más complejos y su vocabulario es mucho más extenso. Sin embargo *ML-Schema* se ha diseñado para simplificar el modelado de experimentos de minería de datos, en concordancia con lo que ofrecen los proveedores de Cloud Computing. Hemos ampliado *ML-Schema* adaptando su modelo a uno específico y heredando gran parte de sus características.

En la Figura 3.3 destacan los siguientes componentes de vocabulario ofrecido por `dmcc-scchema` (para acortar el nombre de `dmcc-scchema` en la ins-

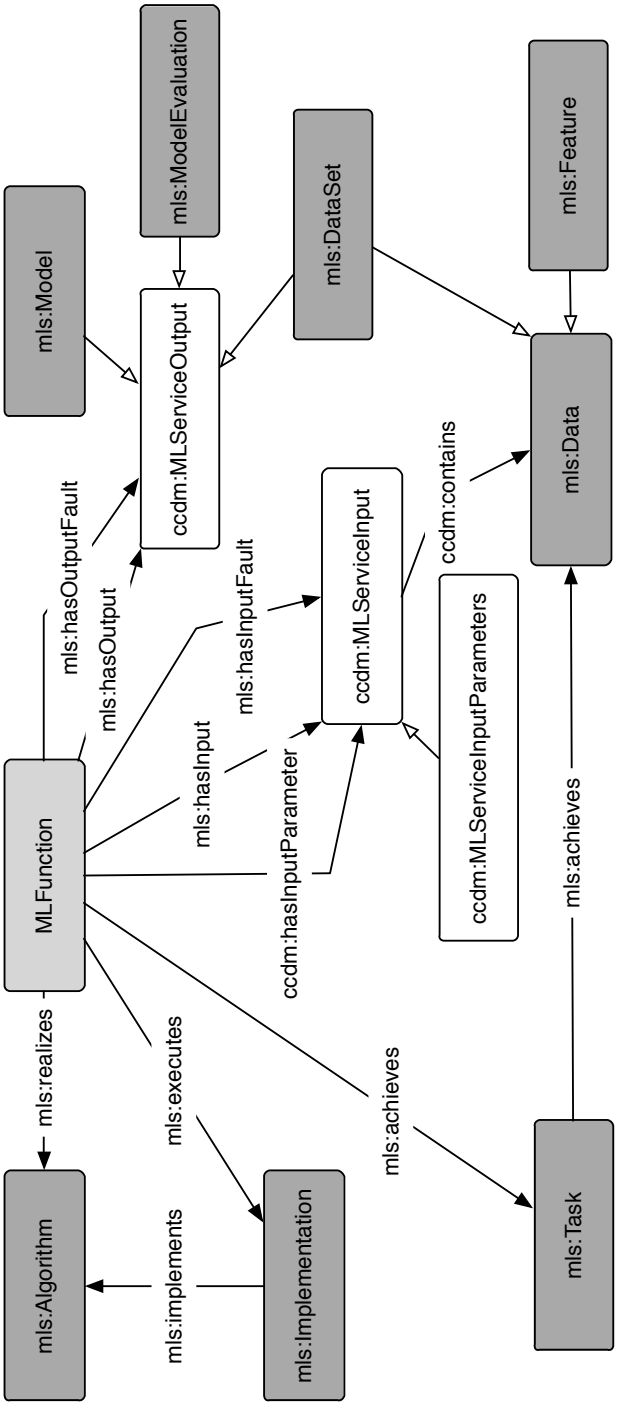

Figura 3.3: Experimentación, algoritmos y flujos de trabajo de minería de datos incluidos en la definición de `dmcc-scchema`.

tanciacion de los ejemplos, hemos usado `ccdm` como el nombre del esquema para la definición):

`ccdm:MLFunction` Configura las operaciones, función o algoritmo a ejecutar. Por ejemplo, *Random Forest* o *K-Means*.

`ccdm:MLServiceOutput` La salida del algoritmo o la ejecución. Aquí la salida del experimento se modela como *Modelo*, *Evaluación de modelo* o *Datos*.

`ccdm:MLServiceInput` La entrada del algoritmo, que corresponde a la configuración de la implementación del algoritmo. Aquí, puede describir el modelo de la entrada de datos del experimento, como el conjunto de datos y los parámetros (`ccdm:MLServiceInputParameters`) del algoritmo ejecutado.

`mls:Modelo` Contiene información específica del modelo que se ha generado a partir de la ejecución del método de minería de datos.

`mls:ModelEvaluation` Proporciona las medidas de rendimiento del modelo.

`mls:Datos` Contienen la información de tablas compuestas o sólo atributos (columna de la tabla), sólo instancias (fila), o un único valor.

`mls:Tarea` Es una parte del experimento que necesita ser realizado en el proceso de minería de datos y su experimentación, de modo que podemos tener múltiples tareas de experimentación.

3.3.4. Interacción

La interacción con los servicios en Cloud Computing se realiza generalmente a través de una API *RESTful*. Esta API proporciona la funcionalidad básica de la interacción con el consumidor de servicios, que debe estar previamente autenticado para utilizar los servicios identificados de esta manera. Para la interacción se ha utilizado la entidad `Action` del esquema proporcionado por el vocabulario `schema.org`[11]. Con esta definición, los puntos de entrada de servicios, métodos y variables de interacción se especifican completamente para todos los servicios.

[11] Action: https://schema.org/Action

3.3.5. Acuerdos de nivel de servicio – SLA

El comercio de servicios de Cloud Computing se realiza a través una serie de acuerdos contractuales entre las partes interesadas que participan en los servicios. Tanto el proveedor como el consumidor del servicio deben acordar las condiciones del servicio antes y durante la prestación del mismo. El SLA define criterios técnicos, relacionados con la disponibilidad, el tiempo de respuesta o la recuperación de errores. Los objetivos del nivel de servicio o SLO son características específicas medibles del SLA, tales como disponibilidad, rendimiento, frecuencia, tiempo de respuesta o calidad de servicio. Además del SLA, debe contemplar la adopción de medidas cuando no se puedan alcanzar tales acuerdos, en cuyo caso se ofrecerá una indemnización a los consumidores o bien algún tipo de beneficio sobre el coste de la interrupción de los servicios.

Las SLAs estudiadas dentro el entorno de servicio de minería de datos se establecen mediante el uso los términos y definiciones los acuerdos [Ambulkar and Borkar, 2012], [Li and Du, 2013], [Zheng et al., 2013]. Estos términos pueden tener ciertas condiciones asociadas que, en caso de violación, pueden suponer una compensación de tipo muy variado. Algunos proveedores pueden integrar una compensación tal como créditos de utilización de recursos de cómputo o bien directamente aplicar una bonificación del coste sobre los servicios que se están utilizando.

En términos generales, el SLA para los servicios de Cloud Computing viene dado por un término de los acuerdos que contiene una o más definiciones, como por ejemplo el término "*Porcentaje Mensual de Tiempo de Actividad*" (en inglés "Monthly Uptime Percentage" *MUP*), que especifica el "*máximo de minutos disponibles menos el tiempo de inactividad dividido por el máximo de minutos disponibles en un mes de facturación*". Para este término se establece una métrica y un intervalo sobre el cual se aplica una compensación cuando no se cumpla el término del contrato. En este contexto se ha creado un esquema denominado `ccsla` [12] para la definición de todos los componentes que se han considerado para un SLA y este tipo de servicios de minería de datos. La Figura 3.4 contiene el esquema del modelado de SLA que se ha diseñado. En la tabla 3.4 se muestra un ejemplo con un par de intervalos *MUP* y su compensación asociada.

[12]Cloud Computing SLA schema: `https://lov.okfn.org/dataset/lov/vocabs/ccsla`

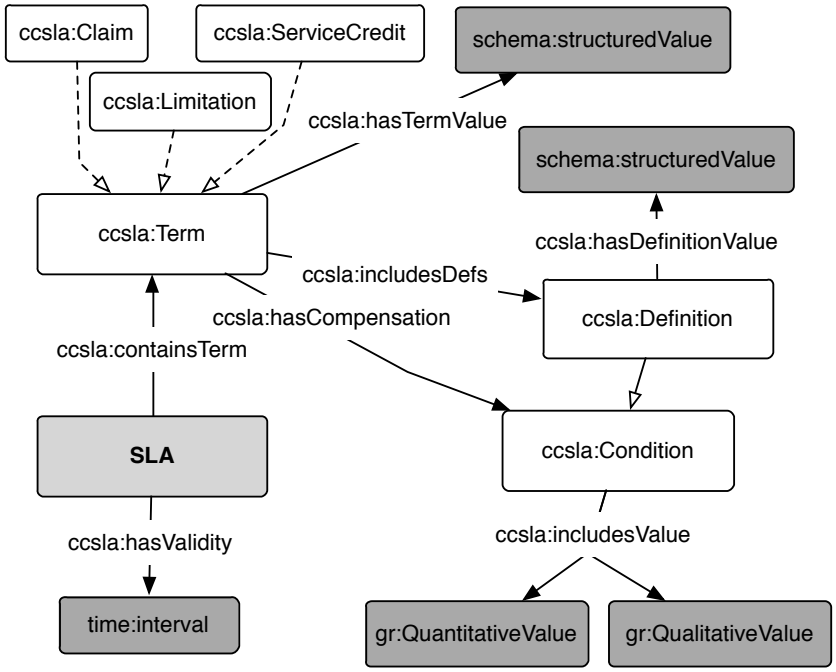

Figura 3.4: Esquema para la gestión de un SLA ofrecido por `ccsla`.

3.3.6. Precios y tarificación de los servicios

Los servicios de Cloud Computing se ven afectados por los costes, tarifas y precios que varían según las propiedades del servicio ofrecido o su consumo. Siguiendo el modelo de «utility» de pago por uso, los costes de utilización del servicio están directamente relacionados con las características del servicio y la explotación que se hace de él. El modelado de precios de los servicios en Cloud Computing es una tarea compleja debido a la gran variedad de planes, condicionantes y modelos de negocio disponibles entre los diversos proveedores de Cloud Computing. No sólo existe un único plan de precios para el uso de los recursos, sino que también ese uso se ve afectado por aspectos técnicos, de configuración (coste de las instancias) o de localización del servicio (regiones). En la tabla 3.5 se recogen algunos de los aspectos modificadores

Porcentaje de actividad mensual (MUP)	**Compensación**
[0.00 %, 99.00 %]	25 %
[99.00 %, 99.99 %]	10 %

Tabla 3.4: Ejemplo de un término *MUP* y compensaciones asociadas.

de precios para el uso de este tipo de servicios para un subconjunto de los proveedores de servicios de minería de datos más conocidos.

Los siguientes componentes del vocabulario se pueden ver en diagrama de la Figura 3.5:

ccpricing:PricingPlan Permite definir los atributos del plan de tarifas y sus detalles. Por ejemplo, un plan de precios gratuito o similar. Debe coincidir con los atributos de `MinPrice` y `MaxPrice`.

gr:Especificación de precios Proporciona todas las herramientas para definir los precios de los servicios con el mayor nivel de detalle. Viene del esquema de `GoodRelations`[13] (`gr`).

ccpricing:Compuesto Componentes del precio del servicio. Los servicios se cobran de acuerdo a valores que están relacionados con ciertos atributos de la configuración del servicio. En el compuesto se pueden añadir elementos como el tipo de instancias, así como el coste/precio de la región.

ccinstances:Instancia Proporciona el vocabulario específico para la definición de los atributos de precio relacionados con la instancia o instancias en las que se ejecuta el servicio de minería de datos.

ccregiones: Region Permite utilizar el esquema de localizaciones y regiones, ofrecido por los proveedores. Este es un valor adicional a los costos del servicio, como parte del precio.

ccregiones: Region Permite utilizar el esquema de localizaciones y regiones, ofrecido por los proveedores. Este es un valor adicional a los costos del servicio, como parte del precio.

Proveedor / Servicio	Region	Instancia	Almacenamiento
Amazon SageMaker	sí	sí	sí
Amazon Lambda	sí	no	no
Azure Machine Learning	sí	sí	sí
Azure Functions	sí	no	no
IBM BlueMix	sí	sí	no
Google Machine Learning	sí	sí	no
Algorithmia	no	no	no

Tabla 3.5: Componentes del precio considerados para cada proveedor y servicio analizado.

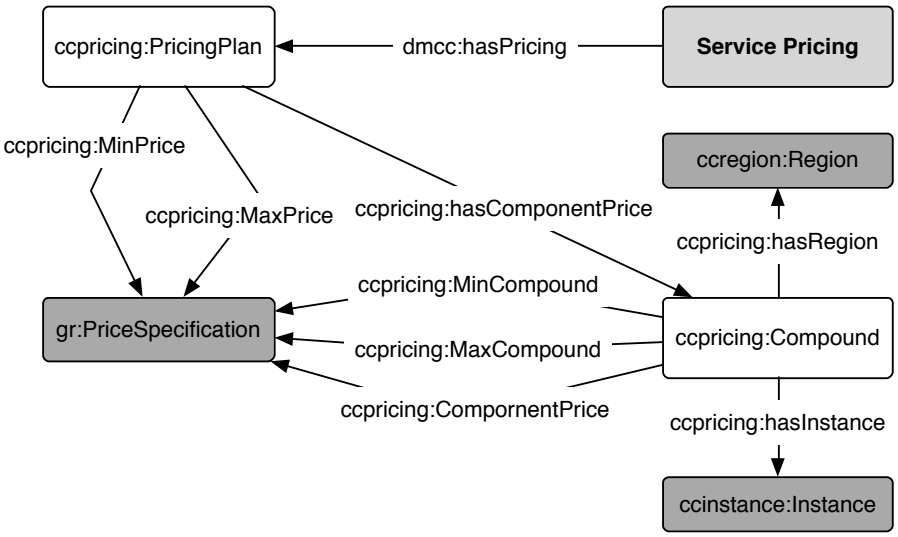

Figura 3.5: Esquema de precios soportado por `dmcc-schema`, donde se incluyen las instancias, las regiones y la gestión de la tarificación.

Para esta parte del modelado, se han desarrollado tres vocabularios que complementan la definición de precios y que no estaban disponibles. Por un lado `ccinstances`[14], que contiene todo lo necesario para la definición de instancias (CPU, número de núcleos de CPU, modelo, RAM o HD), por otro lado `ccregions`[15], que proporciona el modelado general de las regiones de servicio y ubicación en Cloud Computing y por último el esquema para el modelado de precios `ccpricing`[16].

En el sitio web de información de la Tesis está toda la información adicional sobre los vocabularios diseñados, ejemplos, diagramas y conjuntos de datos relacionados con la propuesta de servicios de minería de datos en Cloud Computing («data mining services» – *dmservices*). El diagrama completo de entidades, clases, relaciones y atributos de la propuesta `dmcc-schema` está disponible en el enlace: `http://dicits.ugr.es/linkeddata/dmservices/#dmcc-schema`.

[13] `https://www.w3.org/wiki/GoodRelations`
[14] Cloud Computing instances schema: https://lov.linkeddata.es/dataset/lov/vocabs/cci
[15] Cloud Computing regions schema: https://lov.linkeddata.es/dataset/lov/vocabs/ccr
[16] Cloud Computing pricing schema: https://lov.linkeddata.es/dataset/lov/vocabs/ccp

3.4. Casos de uso

En esta sección ilustraremos cómo el esquema `dmcc-schema` puede ser usado de forma efectiva para definir un conjunto de servicios específicos de minería de datos, así como varios aspectos adicionales relacionados con Cloud Computing tales como son el SLA, la interacción, los precios o el tipo de autenticación. Para ello crearemos una instancia de servicio para los algoritmos *K-Means* y *Random Forest*. Tanto en la implementación del servicio de *K-Means*, como de *Random Forest*, utilizaremos el modelo de función por defecto y los parámetros de estos algoritmo siguiendo la especificación sintáctica que provee el lenguaje *R* para estas funciones. Para instanciar los servicios utilizaremos diversos vocabularios como `dcterms` (`dc:` Dublin Core [Weibel et al., 1998]), `GoodRelations` [Hepp, 2008] (`gr:`), RDFS (`rdfs:`) o `schema.org` (`s:`). A lo largo de todos los casos de uso, nombraremos `dmcc-schema` con el espacio de nombres `dmcc` para simplificar.

Para comenzar con la descripción del servicio es necesario tener en cuenta el diagrama que aparece en la Figura 3.1. Este diagrama contiene todas las partes funcionales que serán necesarias para definir el servicio al completo y a partir del cual comenzar a describir cada componente del servicio de forma individual.

3.4.1. Definición de los datos de proveedor de servicios

Para comenzar con la descripción del servicio, la primera parte que se va a desarrollar corresponde con instanciación de la entidad `dmcc:ServiceProvider`, incluyendo algunos de sus datos, utilizando la clase `dmcc:MLServiceProvider`. Con esta clase se pueden definir diferentes aspectos básicos de la gestión y de los servicios incluidos por el proveedor.

```
1  _:MLProvider a dmcc:MLServiceProvider;
2   rdfs:label "ML Provider"@en ;
3   dc:description
4     "DICITS ML SP"@en ;
5   gr:name "DITICS ML Provider";
6   gr:legalName "U. of Granada";
7   gr:hasNAICS "541519";
8   s:url <http://www.dicits.ugr.es>;
9   s:serviceLocation
10   [ a s:PostalAddress;
11     s:addressCountry "ES";
```

```
12      s:addressLocality "Granada";
13    ] ;
14    s:contactPoint
15    [
16     a s:ContactPoint;
17     s:contactType "Costumer Service";
18     s:availableLanguage [
19       a s:Language;
20       s:name "English";];
21     s:email "ml@dicits.ugr.es";
22    ];
23   dmcc:hasMLService
24             _:MLServiceDicitsRF,
25             _:MLServiceDicitsKMeans;
26   dmcc:hasOfferCatalog
27             _:MLServiceDicitsCatalog;
28  .
```

Listado 3.1: Definición de los datos del proveedor de servicio de minería de datos.

En el listado de código 3.1 se realiza la primera definición de los datos del servicio. Esta especificación por un lado expone los datos administrativos básicos del servicio (nombre, datos legales del proveedor, contacto y localización, etc.) y por otro lado describe los aspectos del servicio de minería de datos (datos del catálogo de servicios, los servicios específicos incluidos en el proveedor, etc.). En el siguiente listado detallamos cada uno de los aspectos que se incluyen para la definición de los datos de un proveedor con `dmcc:MLServiceProvider` :

Línea 2 `rdfs:label` Define el nombre sencillo del proveedor del servicio.

Líneas 3-4 `dc:description` Permite la descripción ampliada del nombre del proveedor.

Líneas 5-7 `gr: ...` Incluye los datos relativos a varios aspectos legales del proveedor como son el nombre `gr:name`, nombre oficial/legal del proveedor `gr:legalName`, o su identificador de negocio `gr:hasNAICS`.

Líneas 9-22 `s: ...` Gracias a `schema.org` (`s:`) es posible realizar una descripción de los datos de la localización del proveedor y el servicio de una forma totalmente detallada. Con `s:serviceLocation` se definen aspectos de la dirección postal o física del proveedor y con `s:contactPoint`, se describen los puntos de contacto con el proveedor, por ejemplo desde

la persona de contacto hasta los medios de contacto electrónicos como por ejemplo el e-mail.

Líneas 23-25 `dmcc:hasMLService` Utilizando `dmcc-schema` definimos los servicios que el proveedor de minería de datos es capaz de ofrecer. En este caso con `_:MLServiceDicitsRF` y `_:MLServiceDicitsKMeans` estamos indicando que el proveedor ofrece dos servicios, uno para un algoritmo *Random Forest* y otro para un algoritmo *KMeans*. Ambos serán definidos mas adelante con los detalles del servicio específico de minería de datos.

Líneas 26-27 `dmcc:hasOfferCatalog` Definimos los datos de nuestro catálogo de servicios con `_:MLServiceDicitsCatalog`.

3.4.2. Definición del servicio

La definición del servicio aglutina todos los aspectos clave relacionados con la funcionalidad de minería de datos. En el listado de código 3.2 se muestra la instanciación de cada una de las entidades para `_:MLServiceDicitsRF`. Todas las características del servicio, tales como los puntos de interacción `_:MLServiceInteraction`, acuerdos del nivel de servicio `_:MLServiceSLA`, la parte central de minería de datos `_:MLServiceFunction`, la autenticación y el plan de precios `_:MLServicePricing` son definidos dentro de `_:MLServiceDicitsRF`.

```
1  _:MLServiceDicitsRF a dmcc:MLService;
2   rdfs:label
3      "ML Service dicits.ugr.es"@en ;
4   dc:description
5      "DICITS ML Service"@en ;
6   dmcc:hasInteractionPoint
7      _:MLServiceInteraction;
8   dmcc:hasServiceCommitment
9      _:MLServiceSLA;
10  dmcc:hasFunction
11     _:MLServiceFunction;
12  dmcc:hasAuthentication
13     _:MLServiceAuth;
14  dmcc:hasPricingPlan
15    _:MLServicePricing;
16  .
```

Listado 3.2: Componentes del modelo de minería de datos.

Cada una de las definiciones del listado 3.2 se detallan en las siguientes subsecciones.

3.4.3. Definición de la interacción

La interacción con el servicio se realiza utilizando la clase `dmcc:Interaction` que incluye la propiedad `dmcc:hasEntryPoint`, esta permite definir una entidad `Action` sobre un recurso u objeto (ver listado de código 3.3). En este caso utilizamos el vocabulario `schema.org` para establecer el método de acceso al servicio *Random Forest*, mediante el cual especificamos que el servicio se consumirá a través de una API de tipo *RESTful*, utilizando el método de acceso `HTTP POST`. Además indicamos la plantilla de *URL* a través de la cual será consumido el servicio `http://dicits.ugr.es/ml/rf/` y sus parámetros.

```
1  _:MLServiceInteraction
2       a dmcc:Interaction;
3   rdfs:label "Service Interaction"@en;
4   dc:description
5        "Service EntryPoint"@en ;
6   dmcc:hasEntryPoint [ a s:Action;
7   s:target [
8    a s:EntryPoint;
9    s:httpMethod "POST";
10   s:contentType
11    "x-www-form-urlencoded";
12   s:urlTemplate
13    "http://dicits.ugr.es/ml/rf/;";
14   s:formula-input [
15    a s:PropertyValueSpecification;
16    s:valueRequired true;
17    s:valueName "formula"; ];
18   s:dataset-input [
19    a s:PropertyValueSpecification;
20    s:valueRequired true;
21    s:valueName "data";];
22   ...
```

Listado 3.3: Interacción con el servicio.

Los elementos que se describen en el listado de código 3.3 son los siguientes:

Lineas 1-2 `_:MLServiceInteraction` Instancia un objeto de tipo `dmcc:Interaction` con el que se permite describir toda la interacción de los servicios con los consumidores.

Lineas 3-5 `rdfs: ..` En esta parte se definen varias características de la interacción de los servicios con los usuarios consumidores tales como el nombre o la descripción de la entidad.

Lineas 6- `dmcc:hasEntryPoint ...` Este grupo de sentencias establece los puntos de entrada o acceso a los servicios que ofrece el proveedor. La especificación de un punto de acceso puede resultar una tarea compleja, para simplificarlo, en este ejemplo únicamente se definen los elementos sobre el tipo de método HTTP usado `s:httpMethod "POST";` y el tipo de plantilla de URL que se usa para desplegar los servicios `s:urlTemplate`. En este caso algo como `http://dicits.ugr.es/ml/rf/` ha sido especificado para indicar como se construye la URL del servicio de minería de datos para *Random Forest*, con `s:formula-input` y `s:dataset-input` como parámetros que llevaría asociado el servicio. En este caso con esos dos parámetros le indicamos al servicio que en la URL se codifican el `dataset` y la `formula`.

3.4.4. Definición de la autenticación

Para este ejemplo, se ha optado por una autenticación basada en `API KEY`. Una autenticación basada en `API KEY` consiste en que el proveedor ofrece a los consumidores un identificador (`KEY`) que puede utilizar para consumir los servicios. Este tipo de autenticación basada en este tipo de «Tokens», destaca por la sencillez en cuanto al consumo de los servicios, pues solo es necesario tener el identificador para poder acceder al servicio y por otro parte la comodidad de gestión por parte del proveedor para asignar y controlar «Tokens» para el consumo por parte de los clientes de un modo muy eficiente. La forma de instanciar este tipo de autenticación para nuestros servicios se realiza mediante `waa:requiresAuthentication`. La preparación del mecanismo de autenticación se muestra en el listado de código 3.4.

```
1 _:MLServiceAuth
2       a dmcc:ServiceAuthentication;
3  rdfs:label
4    "Service Authentication"@en ;
5  dc:description "Service Auth"@en ;
6  waa:requiresAuthentication waa:All;
7  waa:hasAuthenticationMechanism
```

```
 8     [ a waa:Direct ;
 9       waa:hasInputCredentials
10      [ a waa:APIkey;
11       waa:isGroundedIn "key";
12      ];
13    waa:wayOfSendingInformation
14        waa:ViaURI;
15  ]
16  .
```

Listado 3.4: Descripción de la autentiación de los servicios.

La descripción de cada uno de los elementos que aparecen en el listado 3.4 corresponden con:

Línea 1-2 `_:MLServiceAuth` Instancia un objeto de tipo `dmcc:ServiceAuthentication` con el que se permite describir toda la autenticación disponible para los servicios de cara a los usuarios consumidores.

Líneas 3-4 `rdfs:label ...` Define todos los aspectos relacionados con la identificación de la autenticación.

Línea 6 `waa:requiresAuthentication` Especifica si el servicio necesita autenticación, en este caso afirmativo, la usamos y la especificamos con `waa:All`.

Líneas 7-12 `waa:hasAuthenticationMechanism` Se define el mecanismo de autenticación que el servicio tendrá, de modo que se establece que será una autenticación directa contra la plataforma del proveedor `waa:Direct`, además se indicará el tipo de entrada de las credenciales de acceso que será por medio de una `APIKey waa:hasInputCredentials`.

Líneas 13-15 `waa:wayOfSendingInformation` Identifica como se enviará la información de la autenticación. En nuestro caso al usar una API KEY, los datos de la KEY estarán disponibles a través de la URL `waa:ViaURI`.

3.4.5. Definición de los acuerdos de nivel de servicio (SLA)

Para la definición de un acuerdo de nivel de servicio, hemos tomado como ejemplo los acuerdos que tienen establecidos algunos proveedores como

Amazon o *Microsoft Azure*. En estos proveedores, y para todos los servicios en general, identifican un término correspondiente al porcentaje de tiempo que la infraestructura de computación está disponible (está funcionando), también llamado *MUP*. En este ejemplo, para el término *MUP* hemos definido dos intervalos: menos de 99.00 % se compensa con 30 créditos (en el uso del servicio) y el intervalo de 99.00 % a 99.99 % se compensa con 10 créditos. Para modelar estos términos en el acuerdo de nivel de servicio usamos `ccsla:SLA` junto con la propiedad `ccsla:containsTerm`, en el que definimos los términos específicos para el SLA. En el listado de código 3.5 un ejemplo concreto de SLA es definido.

```
1   _:SLATermMUP_A a ccsla:Term;
2   rdfs:label "SLA MUP"@en ;
3    dc:description
4      "SLA Term: Monthly Uptime %"@en ;
5    dc:comment "Calculated by
6      subtracting from 100% the
7      percentage of minutes
8      during ..."@en ;
9    ccsla:hasCompensation
10     _:SLACompensationMUP_A,
11     _:SLACompensationMUP_B;
12   ccsla:includeDefs
13     _:SLADefinition_A,
14     _:SLADefinition_B;
15  .
```

Listado 3.5: Términos y compensación con SLA.

El modelado de un SLA siempre es complejo puesto que prácticamente da cabida a cualquier tipo de términos y descripciones que contienen los acuerdos acerca de la calidad del servicio entre usuario consumidor y el proveedor de servicios. Para modelar este ejemplo se usará el mismo conjunto de términos del acuerdo de nivel de servicio planteados en la tabla 3.4. Cada una de las sentencias usadas para definir el SLA en el listado de código 3.5 se detallan a continuación:

Línea 1 `_:SLATermMUP_A a ccsla:Term;` Con `_:SLATermMUP_A` se instancia una clase de tipo `ccsla:Term;`, desde la cual definir y describir términos del SLA.

Líneas 2-8 `rdfs:label ...` Contiene la descripción detallada del SLA. Se especifica en que consiste el SLA `dc:comment Calculated ...` y las descripciones básicas con `dc:description SLA Term: Monthly...`.

Líneas 9-11 `ccsla:hasCompensation ...` Para definir cada uno de los elementos de compensación en cuanto a si se viola alguna o algunas de las condiciones del acuerdo o del término del nivel del servicio establecido por el propio proveedor, se establecen con `ccsla:hasCompensation` las compensaciones específicas. En este ejemplo contamos con dos compensaciones `_:SLACompensationMUP_A`, y `_:SLACompensationMUP_B`. Ambos términos se especifican en listado de codigo 3.5.

Líneas 12-14 `ccsla:includeDefs ...` Con la definición de `_:SLADefinition_A` y `_:SLADefinition_B` conseguimos identificar y dar significado a cada una de las compensaciones que se aplican para `_:SLACompensationMUP_A`, y `_:SLACompensationMUP_B` respectivamente.

Los elementos pendientes de definir dentro del listado 3.5 corresponden con `_:SLACompensationMUP_A` y `_:SLADefinition_A` (de igual modo sería para la compensación para el término `B`). En el listado 3.6 se formaliza `_:SLADefinition_A`.

```
1  _:SLADefinition_A a ccsla:Definition;
2   ccsla:hasDefinitionValue [
3   a s:structuredValue;
4   s:value [
5    a s:QuantitativeValue;
6    s:maxValue 99.99;
7    s:minValue 99.00;
8    s:unitText "Percentage";
9    ];
10  ];
11  .
```

Listado 3.6: Definición de un término del SLA.

De este modo, para establecer la definición de la primera parte del intervalo de compensación para este término del SLA, que oscila de 99.00 % a 99.99 %, hay que tener en cuenta lo siguiente:

Línea 2 `ccsla:hasDefinitionValue` Indica que la definición del término del SLA tiene un valor determinado.

Líneas 3-10 `a s:structuredValue;` Establece que es un dato estructurado de tipo rango o intervalo. Este rango se define mediante `s:maxValue`

`99.99;` y `s:minValue 99.00`. Para el tipo de dato rango se especifica que es un valor de cantidad con `a s:QuantitativeValue;` y la unidad en la que se mide ese rango es un porcentaje `s:unitText "Percentage";`.

3.4.6. Definición del plan de precios del servicio

En cuanto a la definición del coste económico del servicio hemos considerado para este ejemplo de caso de uso un servicio gratuito de minería de datos, pero limitado a *250* horas de computación de sus servicios dentro de una instancia con un modelo de *CPU Intel i7*, 64 GB de RAM y una única región geográfica concreta.

```
1 _:MLServicePricing a
2         ccpricing:ServicePricing;
3  rdfs:label "Service Pricing"@en ;
4  dc:description
5    "Service Pricing"@en ;
6  ccpricing:hasPricing
7   _:PricingPlanFree,
8   _:PricingPlanRegular;
9 .
```

Listado 3.7: Tarificación del servicio: plan gratuito y plan normal.

En el listado 3.7 se definen los tipos de planes de datos que soporta el servicio con `ccpricing:hasPricing`. En este caso se han especificado dos: `_:PricingPlanFree` y `_:PricingPlanRegular;`. De este modo, se pueden crear varios planes de precios y tarifas para cada uno de los servicios que ofrezca el proveedor. Para este ejemplo se especifica un plan gratuito con `_:ComponentsPricePlanFree` donde el modelado de la tarificación se realiza utilizando la definición de precios proporcionada por `ccpricing`. El plan de precios tendrá en cuenta un conjunto de otras entidades que afectan al precio, como la región geográfica o el tipo de instancia. Por ejemplo, para definir las características del tipo de instancia usada en el plan "Free", usamos `ccinstance:Instance;` y algunos atributos como RAM o CPU como se ve en el listado 3.8. En este último listado se definen los tres elementos modificadores del coste del servicio `ccpricing:hasRegion`, `ccpricing:hasInstance`, y `ccpricing:MaxCompound` para el plan de precios "Free" `_:ComponentsPricePlanFree`.

```
1 _:ComponentsPricePlanFree
```

```
2      a ccpricing:Compound;
3   ccpricing:hasRegion _:RegionFree;
4   ccpricing:hasInstance _:InstanceFree;
5   ccpricing:MaxCompound _:MaxUsageFree;
6  .
```

Listado 3.8: Componentes del modelado de precios.

Del listado 3.8, hay que definir todos los elementos. Para no extender cada definición, sólo haremos la definición de uno de los elementos que componen el precio, por ejemplo la instancia que estarían dentro del plan de precios "Free". La definición de las instancias del plan "Free" se pueden ver en el listado 3.9.

```
1  _:InstanceFree
2      a ccinstances:Instance;
3     ccinstances:hasRAM [
4       a ccinstances:ram;
5          s:value "64"
6          s:unitCode "E34";
7     ] ;
8     ccinstances:hasCPU [
9       a ccinstances:cpu;
10       ccinstances:cpu_model
11         "Intel i7";
12    ] ;
13 .
```

Listado 3.9: Precios y selección de características de instancias.

La definición específica de las características de las instancias se hace mediante el uso de `ccinstances:hasRAM` y `ccinstances:hasCPU`. Para definir las características de la RAM indicamos la cantidad para la instancia, en este caso 64 (GigaBytes) y la unidad en que se mide con el código internacional E34 que corresponde a GBytes. En cuanto al tipo de CPU y características, debido a su amplitud se ha establecido el identificador de modelo de CPU como el más adecuado para este fin (adicionalmente otros elementos pueden ser definidos para caracterizar y definir las CPUs de las instancias).

Para definir los límites del uso libre (`MaxUsageFree`) determinamos el plan de acceso libre al servicio y con una limitación de las horas de cálculo a 250. Para ello utilizamos las clases `gr:PriceSpecification` y `gr:Offering` tal y como se muestra en el listado 3.10. Ejemplos adicionales para el esquema de precios y un conjunto de datos de los planes de precios de *Amazon*

SageMaker están disponibles en la web del trabajo de investigación.

```
_:MaxUsageFree a
   gr:PriceSpecification,
   gr:Offering;
 gr:max 0.00;
 gr:priceCurrency "USD";
 gr:includesObject [
  a gr:TypeAndQualityNode;
  gr:amountOfThisGood "250";
  gr:hasUnitOfMeasurement "HRS";
 ];
.
```

Listado 3.10: Especificación de precios para el plan "Free".

También se ha incluido un plan de precios normales, para cuando se quiera usar instancias bajo demanda, para ello usamos `_:PricingPlanRegular` definido en el listado 3.7. Para establecer precios condicionados al tiempo de utilización, se realiza de modo similar, pero se ha incluido otro tipo de instancia y la misma región como se muestra en el listado 3.11. En este caso, el plan de tarifas normales ofrece el consumo del tipo de instancia `_:instance02` a un coste de $0.0464 por hora en la región de "North Virgnia" `_:RegionNVirginia`.

```
_:PricingPlanRegular a ccpricing:Compound;
    rdfs:label "Price Components";
    rdfs:comment
     "List of components and limits";
    ccpricing:hasRegion
     _:RegionNVirginia;
    ccpricing:hasInstances
     _:instance02;
    ccpricing:withMaxCompound [
     a gr:Offering;
       gr:hasPriceSpecification [
         a gr:PriceSpecification;
           gr:hasCurrency
            "USD"^^xsd:string;
           gr:hasCurrencyValue
            "0.0464 USD"^^xsd:float ;
          ];
         gr:includeObject [
           a gr:TypeAndQuantityNode;
           gr:amountOfThisGood
```

```
21                "1"^^xsd:integer ;
22              gr:hasUnitOfMeasurement
23                "HRS"^^xsd:string ;
24              ];
25            ];
26           .
```

Listado 3.11: Precios para la región y la instancia.

En el listado 3.11, se define completamente el plan de precios de un modelo regular de consumo de instancias. La definición de los datos tarificación es más compleja que para el plan de precios "Free". Para poder definir las características de la tarificación se usa `gr:hasPriceSpecification`, desde donde se indica el tipo de moneda usada y el valor del servicio por unidad de medida (`gr:amountOfThisGood` y `gr:hasUnitOfMeasurement`).

3.4.7. Definición de la experimentación en minería de datos

Finalmente se define una de las partes importantes del esquema `dmcc-schema`. Para ello inicialmente es necesario utilizar `ccdm:MLFunction` con la que se definen varios aspectos relativos al algoritmo o funcionalidad que se desea ejecutar. Además se especifican los parámetros de entrada y la salida del algoritmo y todo el conjunto de operaciones básicas del trabajo con conjuntos y el procesamiento de datos.

En el listado 3.1, anteriormente desarrollado se puede ver como en las líneas $23 - 25$ se identifican los dos algoritmos que se van a desarrollar como servicios: `_:MLServiceDicitsRF`, y `_:MLServiceDicitsKMeans`. Cada uno de los servicios está instanciado como una clase `dmcc:MLService;`. En el listado 3.2 se instancia el servicio correspondiente a la función que se realizar con `Random Forest _:MLServiceDicitsRF a dmcc:MLService;` es ahí donde se define toda la funcionalidad del servicio y en concreto donde se especifica todo el trabajo con el servicio de minería de datos `dmcc:hasFunction _:MLServiceFunction;` que se detalla en el listado 3.12.

```
1  _:MLServiceFunction
2    a ccdm:MLFunction ;
3     ccdm:hasInputParameters
4       _:RF_InputParameters;
5     mls:hasInput
```

```
6       _:RF_Input;
7     mls:hasOutput
8       _:RF_Output
9  .
```

Listado 3.12: Componentes de la función/algoritmo.

En el listado 3.12, se indican todos los elementos relacionados con la experimentación con minería de datos, como son los parámetros de entrada (parámetros del algoritmo/función `_:RF_InputParameters`, y ficheros de datos `_:RF_Input;`) y por otro la salida (un modelo u otros ficheros de datos `_:RF_Output`). Los datos de entrada y salida de los algoritmos deben estar incluidos en la definición de la operación a realizar. Los parámetros de entrada del algoritmo se definen con `dmc:MLServiceInputParameters` y la lista de parámetros incluye cada uno de los parámetros `parameter_01, [....]`, como se puede ver en el listado 3.13.

```
1  _:RF_InputParameters
2    a ccdm:MLServiceInputParameters ;
3      ccdm:Parameters
4         _:parameter_01,
5         [...]
6      dc:description
7          "Input Parameters" ;
8      dc:title "Input"
9      .
```

Listado 3.13: Definición de los parámetros de entrada.

La definición de `ccdm:hasInputParameters _:RF_InputParameters` permite especificar los parámetros generales de entrada del algoritmo. Por ejemplo, para el algoritmo *Random Forest* es posible declarar parámetros como `dc:title "ntrees"`. (número de árboles generados), si un parámetro es obligatorio (`ccdm:mandatory "false"`) o si tiene un valor por defecto. El listado 3.14 muestra la definición de uno de los parámetros `parámetro_01`.

```
1  _:parameter_01
2    a ccdm:MLServiceInputParameter ;
3      ccdm:defaultvalue "100" ;
4      ccdm:mandatory "false" ;
5      dc:description
6        "Number of trees" ;
7      dc:title "ntrees" .
```

Listado 3.14: Parámetros y atributos.

Para concluir la definición de este caso de uso, se ha considerado un modelo de salida con `mls:Model` y una evaluación del modelo `mls:ModelEvaluation` para especificar los resultados de la ejecución del servicio en `mls:hasOutput _:RF_Output`. La evaluación del modelo es el resultado específico si el algoritmo devuelve un valor o conjunto de valores. Cuando el algoritmo del servicio realiza por ejemplo un pre-procesado, el resultado es un conjunto de datos. Para el modelo hay que definir, por ejemplo, si el modelo resultante es PMML [Guazzelli et al., 2009b] con un `_:RF_Model a dmc:PMML_Model`, como se muestra en el listado 3.15.

```
1  _:KMeans_Model
2    a ccdm:PMML_Model ;
3      ccdm:storagebucket
4        <dicits://models/> ;
5      dc:description
6        "PMML model" ;
7      dc:title "PMML Model" .
```

Listado 3.15: Modelo y almacenamiento del modelo en PMML.

El segundo algoritmo como servicio ofrecido por el proveedor es un *KMeans*, como se especifica en el listado 3.1 en la línea 25 `_:MLServiceDicitsKMeans`. Este nuevo servicio sigue el mismo patrón que la definición de servicio que se ha realizado para *Random Forest* pero modificando la parametrización propia del algoritmo. El listado 3.16 contiene parte de la definición del algoritmo con un pequeño subconjunto de parámetros de entrada como ejemplo, donde se indica para cada uno de los parámetros, el nombre, la obligatoriedad o el valor por defecto.

```
1  _:parameter_01
2       a ccdm:MLServiceInputParameter ;
3          ccdm:defaultvalue "" ;
4      ccdm:mandatory "true" ;
5      dc:description
6       "numeric matrix of data" ;
7      dc:title "x"
8          .
9  _:parameter_02
10      a ccdm:MLServiceInputParameter ;
11         dmc:defaultvalue "3" ;
12     ccdm:mandatory "true" ;
13     dc:description
14     "either the number of clusters." ;
15     dc:title "centres"
```

```
16                .
17  ...
```

Listado 3.16: Ejemplo de un par de parámetros para *K-Means*.

Otros servicios implementados con `dmmc-schema` tales como *NaiveBayes*, *Regresión lineal*, *SVM* u *Optics*, entre muchos otros, están disponibles en el sitio web de `dmcc-schema`[17]

3.5. Validación

Con el fin de validar `dmcc-schema`, se ha llevado a cabo una transcripción al esquema `dmcc-schema` de parte de los servicios que se han estudiado en la tabla 3.1. Como resultado de esta transcripción, se ha generado un conjunto de datos que contiene la definición de los servicios de minería de datos con todos los componentes relacionados con la gestión y experimentación de Cloud Computing, tales como instancias de ejecución, regiones, SLA, precios, algoritmos, almacenamiento o parámetros de entrada y salida; este conjunto de datos se ha desplegado en la plataforma *Apache Jena-Fuseki*[18] y permite disponer de un total de aproximadamente 6.000 tripletas en *RDF/Turtle*, listas para poder ser consultadas.

Posteriormente, como elemento adicional de validación, se han realizado una serie de consultas utilizando SPARQL dentro de *Apache Jena-Fuseki*. El objetivo de esta parte es confirmar que con `dmcc-schema` es posible capturar todas las características de los servicios de minería de datos y que se pueden integrar en una única definición, a través de la cual es posible realizar consultas (y gestionar los servicios siempre se integre dentro de un *Broker* de Cloud Computing) sobre este tipo de servicios para múltiples proveedores. Consultas como,

"Lista todos los proveedores que ofrecen un servicio con un algoritmo Random Forest",

u otras más elaboradas como:

"Muéstrame el coste de ejecutar un algoritmo K-Means para un dataset de 100MB para 2 horas de uso de instancia de tipo T1",

[17] http://dicits.ugr.es/linkeddata/dmservices/
[18] `http://dicits.ugr.es/linkeddata/jena/`

ofrecen resultados correctos y permiten mostrar todos los detalles de los servicios y proveedores. Las consultas, resultados y los conjunto de datos están disponibles en el enlace de material extendido de `dmcc-schema`[19].

3.6. Conclusiones

En este capítulo hemos presentado `dmcc-schema`, un esquema compacto y directo para la descripción y definición de los servicios en Cloud Computing. La propuesta desarrollada pretende recoger, por un lado, todo lo relacionado con la definición de los algoritmos como servicio para minería de datos y por otro lado, todos los demás aspectos que componen la gestión de un servicio en Cloud, que es descuidado y en muchos casos no considerado por otras propuestas de forma general. Las otras propuestas de definición de servicios carecen de esta integración, sin embargo `dmcc-schema` permite resolver este escalón en términos de creación de servicios "todo en uno" para entornos de Cloud Computing. `dmcc-schema` se presenta como una herramienta ligera para la definición y modelado de servicios de minería de datos con el objetivo de ofrecer una descripción portable entre los diferentes proveedores de este tipo de servicios. Por lo tanto, con `dmcc-schema` es posible capturar todas las principales características y detalles (gestión y experimentación) de los proveedores más conocidos como *Amazon*, *Azure* o *Google*, entre otros.

El esquema `dmcc-schema` se ha construido a partir de tecnología semántica, utilizando para implementarlo un lenguaje basado en ontologías y siguiendo la propuesta de *Linked Data*, relativas a la reutilización e integración de otros esquemas externos, lo que enriquece el modelado de servicios que se ha diseñado. Además, también asegura que la definición de servicios pueda ser ampliada y mejorada en el futuro, con el objetivo de ofrecer una definición mucho más portable de los servicios y poder adaptarse a los cambios en la gestión futura de los servicios de Cloud Computing.

Otra característica importante a la hora de usar `dmcc-schema` es que abstrae las definiciones heterogéneas de los distintos proveedores de modo que consigue que los servicios de minería de datos tengan una especificación única que pueda aglutinar diferentes especificaciones de servicios. Por esta razón la portabilidad para este tipo de servicios entre diferentes proveedores está asegurada, equilibrando las diferencias entre las definiciones, y permitiendo utilizar `dmcc-schema`, por ejemplo, como parte integral de una

[19] `http://dicits.ugr.es/linkeddata/dmservices/`

herramienta de intermediación de servicios de Cloud Computing, almacenando y gestionando los servicios de otros proveedores de minería de datos en Cloud Computing.

Se confirma la eficacia del esquema, ya que permite definir un servicio de minería de datos con todos sus elementos (algoritmos, costes/precios, ...), y con respecto a los términos de eficiencia, el esquema ha sido validado mediante la transcripción de servicios reales existentes en *Amazon SageMaker*, verificándose que el esquema `dmcc-schema` puede incluir todas las características clave de estos servicios.

Finalmente, es importante destacar que `dmcc-schema` está siendo utilizado dentro de una plataforma de computación y flujo de trabajo para minería de datos en Cloud, llamada OC^{2}DM. Como parte de la plataforma, el esquema `dmcc-schema` es un elemento clave, ya que es el que permite definir y describir los servicios completos que se integran, permitiendo un alto grado de flexibilidad, portabilidad en el diseño y despliegue de los servicios.

Capítulo 4

Implementación de una plataforma de servicios de minería de datos para Cloud Computing

Motivación

Desde hace más de cinco años los proveedores de Cloud Computing incluyen servicios de minería de datos y procesamiento de información. Servicios como *Amazon SageMaker*, *Google Machine Learning*, *IBM Watson* o *Microsoft Azure ML*, aprovechan los recursos de la infraestructura de computación de los proveedores para desplegar algoritmos como servicios listos para ser consumidos bajo demanda. Estos servicios de minería de datos carecen de una descripción y una definición estandarizada. Con una estandarización de la definición de servicios entre los diferentes proveedores de Cloud Computing sería posible, por ejemplo, trasladar un servicio completo a otro proveedor tanto en el aspecto funcional como en cuanto a la definición del servicio. Para solucionar esto, hemos desarrollado un estudio con el objetivo de diseñar e implementar una arquitectura completa para el despliegue de servicios de minería de datos, que abarque desde la definición y descripción de los servicios, hasta la implementación de una plataforma completa para el soporte, computación y despliegue de dichos servicios en Cloud Computing. El resultado es una plataforma basada en el esquema `dmcc-schema`, presentado en el capítulo anterior, para el despliegue de servicios de minería de datos

conocida como OC^{2}DM: *Open Cloud Computing Data Mining*. OC^{2}DM permite aprovechar la infraestructura informática disponible (ya sea en entorno local o en Cloud) para desplegar servicios de minería de datos utilizando las herramientas de procesamiento de datos a gran escala. Además, la definición de servicios semánticos en las capas superiores supone una considerable mejora en diferentes aspectos tales como la portabilidad, la flexibilidad y la interoperatibilidad de los servicios, factores no considerados en su totalidad por las actuales plataformas de minería de datos.

4.1. Introducción

El número de dispositivos conectados a Internet ha ido creciendo desde hace varios años y se estima que para 2021 la cifra aumentará hasta 27.100 millones. Dispositivos, sistemas o aplicaciones están conectados a Internet consumiendo servicios de Cloud Computing desde una amplia variedad de fuentes y plataformas, tales como teléfonos móviles, sensores de IoT, coches autónomos o *weareables*, entre muchos otros. En nuestro día a día podemos ver que prácticamente cualquier dispositivo o aplicación que usamos de forma habitual está conectada a Cloud Computing transmitiendo datos e información casi de manera ininterrumpida. El auge de Internet y el abaratamiento de los costes de fabricación de dispositivos, diseñados ya de forma casi nativa para el acceso a Internet, ha motivado que prácticamente "cualquier" cosa tenga posibilidades de estar interconectada y sea capaz de consumir servicios de Cloud Computing de modo totalmente transparente. Esta facilidad de conexión implica que el volumen de datos que se transfieren a través de las redes de comunicación sea extremadamente elevado, siendo un factor importante a considerar en un futuro, puesto que la conexión de nuevos dispositivos seguirá creciendo de forma lineal en los próximos años.

Según diferentes estudios, el crecimiento del tráfico en los centros de datos se estima que se triplicará en los próximos años, por lo que el volumen de operaciones gestionadas desde plataformas de Cloud Computing superará el 90 % hasta 2021, como se describe en [Networking, 2016], suponiendo un total estimado de más de 20 ZettaBytes de tráfico provenientes únicamente de servicios de Cloud Computing. Estas cifras nos dan una idea de dos aspectos muy importantes: a) el incremento superlativo en el volumen de información que se generará en los próximos años, y b) la demanda incremental de consumo de servicios de diversos tipos desde Cloud Computing.

El catálogo de servicios ofrecido por los proveedores de Cloud Compu-

ting es una muestra de la capacidad operativa de un proveedor de servicios o de una empresa en el contexto del cliente final, de un espacio de mercado o de una unidad de negocio interna. Dentro del contexto de la computación en nube, el catálogo de servicios es un elemento integral de la arquitectura del Cloud Computing. Un catálogo de servicio en Cloud Computing se caracteriza por lo siguiente:

- Contiene un conjunto de servicios que un usuario final puede consumir desde diferentes fuentes (dispositivos móviles, APIs, desde otros servicios, etc.).
- Actúa como índice para los usuarios finales, incluyendo los compromisos de precios y nivel de servicio, y los términos y condiciones para el aprovisionamiento de servicios.
- También se puede utilizar como mecanismo de gestión de la demanda, dirigiendo o incitando a los clientes hacia determinados servicios o configuraciones de servicio o alejándolos de los servicios heredados o en declive, así como asegurándose de la alineación con la gestión y los estándares a través de configuraciones y opciones de servicio predeterminadas.
- Tiene un aspecto de autoservicio, es decir, permite seleccionar las ofertas de servicios del catálogo.
- Sirve como la interfaz de aprovisionamiento para la automatización de servicios.

Además, el catálogo de servicios que ofrece un proveedor debe considerar los siguientes elementos:

- Debe ser medible. El catálogo de servicios debe ser medible para poder ser facturado, así como gestionado para disponibilidad y rendimiento.
- Debe ser completo. Un catálogo debe tener todos los elementos necesarios para poder descubrir, parametrizar y explotar cada uno de los servicios que alberga.
- Debe permitir el escalado de recursos. Los servicios prestados pueden ampliarse o reducirse de acuerdo con los requisitos usuario consumidor o la demanda de los servicios. Debe permitir la escalabilidad horizontal y vertical de los servicios prestados a través de una automatización integrada y transparente.

- Debe ser flexible. Para permitir adaptarse a los nuevos y cambiantes requisitos de servicio para los consumidores finales, integrando otros elementos del propio proveedor o externos.

Considerando los aspectos mencionados anteriormente, los proveedores de Cloud Computing son capaces de ofrecer en su catálogo un amplio conjunto de servicios listos para usar. Estos servicios se enmarcan dentro de alguna de tres capas del modelo de Cloud Computing como se ha presentado en la sección 2.1.2; SaaS, PaaS y IaaS, como por ejemplo almacenamiento de ficheros y datos (*DropBox*, *Google Drive*, *Mega*, etc.), bases de datos (*Amazon DynamoDB*[1], *MySQL*,...), redes, o recursos informáticos entre otros [Yang et al., 2017].

Además de este tipo de servicios básicos, en el catálogo del proveedor, se incluyen una serie de servicios que aprovechan al máximo las capacidades de Cloud Computing. Estos, son los servicios de procesamiento de datos [García et al., 2016b]. En los últimos años los proveedores de Cloud Computing han añadido una nueva serie de servicios de procesamiento de grandes datos conocidos como *Big Data-as-a-Service*, *Data Mining-as-a-Service* o *Machine Learning-as-a-Service* [Purohit et al., 2017]. Estos servicios se han convertido en una herramienta esencial para el procesamiento de datos y la extracción de conocimiento, dentro de este tipo de entornos. Los principales proveedores aprovechan su vasta infraestructura informática para ofrecer servicios de minería de datos a consumidores siguiendo el modelo propuesto por *NIST* [Mell, Peter et al., 2011] para el consumo y explotación de servicios en Cloud Computing. En esta misma línea para el procesamiento de datos a gran escala, los proveedores de Cloud Computing son una opción muy a tener en cuenta, permitiendo el escalado aparentemente ilimitado de la infraestructura [Talia, 2013], entre otras muchas ventajas.

El crecimiento de los servicios de minería de datos en Cloud Computing ha ido en paralelo con la explosión de nuevos lenguajes, herramientas y plataformas de procesamiento para *Big Data*. Gran parte de estas herramientas, permiten flujos de trabajo de minería de datos y aprendizaje automático sin tener en cuenta aspectos clave de un servicio de Cloud Computing tales como precios, interfaces, SLA, catálogo de algoritmos, etc. Sin embargo, aspectos como la transparencia, la abstracción y la portabilidad de los servicios entre diferentes plataformas son a menudo difíciles de implementar tanto en entornos de Cloud Computing como en *on-premise*. Teniendo esto en cuenta, es legítimo pensar que sería necesario proponer una arquitectura para el

[1] `https://aws.amazon.com/es/dynamodb/`

despliegue de servicios de minería de datos que permitiera abstraer la plataforma informática subyacente, dejando fuera de consideración la tecnología o la arquitectura de soporte y que se centrase en el servicio en sí mismo.

En este capítulo proponemos una arquitectura de despliegue escalable sobre plataformas de procesamiento de datos para soportar servicios de minería de datos en Cloud Computing. Para llevar a cabo la propuesta se ha utilizado, por un lado, un esquema y lenguaje para la descripción de los servicios de minería de datos con `dmcc-schema` (descrito en el capítulo 3) y por otro, el desarrollo de una plataforma para el despliegue para estos servicios. Esta arquitectura de despliegue se llama OC^{2}DM.

Una de las características fundamentales de OC^{2}DM es que aprovecha la infraestructura informática disponible para habilitar los servicios de minería de datos de forma que consigue la abstracción de la arquitectura y las plataformas informáticas sobre las que está construida, logrando un alto grado de flexibilidad en el desarrollo de servicios. La arquitectura de despliegue OC^{2}DM permite definir elementos clave de Cloud Computing como son los algoritmos como servicios, costes/precios, instancias, interfaces, autenticación o SLA, entre otros, utilizando todas las capacidades que ofrecen las tecnología semántica. Posee la capacidad de entregar servicios de minería de datos utilizando implementaciones de algoritmos desde diferentes plataformas de software y procesamiento de datos bien conocidas como *R* [Team, 2000], *Hadoop* [Shvachko et al., 2010] o *Spark* [Zaharia et al., 2010], incluyendo prácticamente cualquier tipo de implementación de algoritmos o funciones como servicios consumibles en Cloud Computing.

El capítulo se estructura de la siguiente manera: en la siguiente sección (4.2) presentamos el estudio del estado del arte relativo a minería de datos, aspectos de Cloud Computing, herramientas y servicios relacionados. En la sección 4.3 describimos nuestra propuesta de arquitectura de despliegue para servicios de minería de datos OC^{2}DM. La implementación y despliegue de los servicios de minería de datos para OC^{2}DM se han llevado a cabo dentro en la sección 4.4. Por último, las conclusiones se describen en la última sección 4.5.

4.2. Trabajo relacionado y estado del arte

Durante más de 20 años, se han desarrollado un conjunto de software para el procesamiento y la minería de datos. Plataformas comerciales como

MatLab[2], *IBM SPSS*[3], *SAS*[4] o *RapidMiner*[5] por ejemplo, ofrecen un entorno completo para este tipo de tareas. Sin embargo, en la últimos años están ganando terreno a las soluciones privativas otras plataformas de código abierto que ofrecen unas herramientas prácticamente igual de completas para la minería de datos y el aprendizaje automático con un amplio número de algoritmos listos para usar y continuamente actualizados.

Software desarrollado para ordenadores personales, tales como *KNIME* [Berthold, Michael R and Cebron, Nicolas and Dill, Fabian and Gabriel, Thomas R and Köt *Weka* [Hall et al., 2009b] y *Orange* [Demšar et al., 2013], son algunos de los mejores ejemplos de estas aplicaciones para minería de datos, ofreciendo una amplia gama de algoritmos y una interfaz gráfica de usuario a través de la cual componer un flujo de trabajo, crear modelos, o visualizar resultados y datos [Jovic et al., 2014]. Aprovechando el motor *R*, *Rattle* [Cortez, 2010a] o *rminer* [Cortez, 2010b] ofrecen una interfaz gráfica y un conjunto de herramientas de minería de datos similar a lo que ofrecen otros entornos como *KNIME* para el flujo de trabajo con datos. Todas estas herramientas permiten abordar problemas de clasificación, regresión, pre-procesamiento, series temporales, aprendizaje no supervisado y muchos otros más, a través de una interfaz visual que facilita el diseño de todas las etapas de trabajo con datos y reduce el tiempo necesario para el aprendizaje del proceso de minería de datos. Este tipo de herramientas facilitan el acceso al análisis de datos para entornos científicos y académicos que de otro modo necesitarían tener conocimientos avanzados de programación. En la Figura 4.1 se pueden ver algunos de los interfaces de estas aplicaciones.

Además de este tipo de plataformas, recientemente se han desarrollado diversas bibliotecas y entornos de trabajo para minería de datos [Cetinsoy et al., 2016]. Alrededor del lenguaje de programación *Python* se han desarrollado un conjunto de bibliotecas para el análisis y el procesamiento de datos, como `scikit-learn` [Pedregosa et al., 2011] o `Pandas` [McKinney et al., 2010] como las más destacadas en este ámbito. `scikit-learn` es parte de un entorno de trabajo que contiene herramientas para la minería y el análisis de datos y *Python Pandas* es una biblioteca de código abierto que proporciona análisis para minería de datos de alto rendimiento. El lenguaje *R* y su entorno de desarrollo *RStudio/RStudio-Server* ha supuesto una revolución en los últimos años dentro de la Ciencia de Datos. Permite importar un gran número de paquetes adi-

[2]MatLab: https://www.mathworks.com/products/matlab.html
[3]IBM SPSS: https://www.ibm.com/es-es/products/spss-statistics
[4]SAS: https://www.sas.com/es_es/software/platform.html
[5]RapidMiner: https://rapidminer.com/

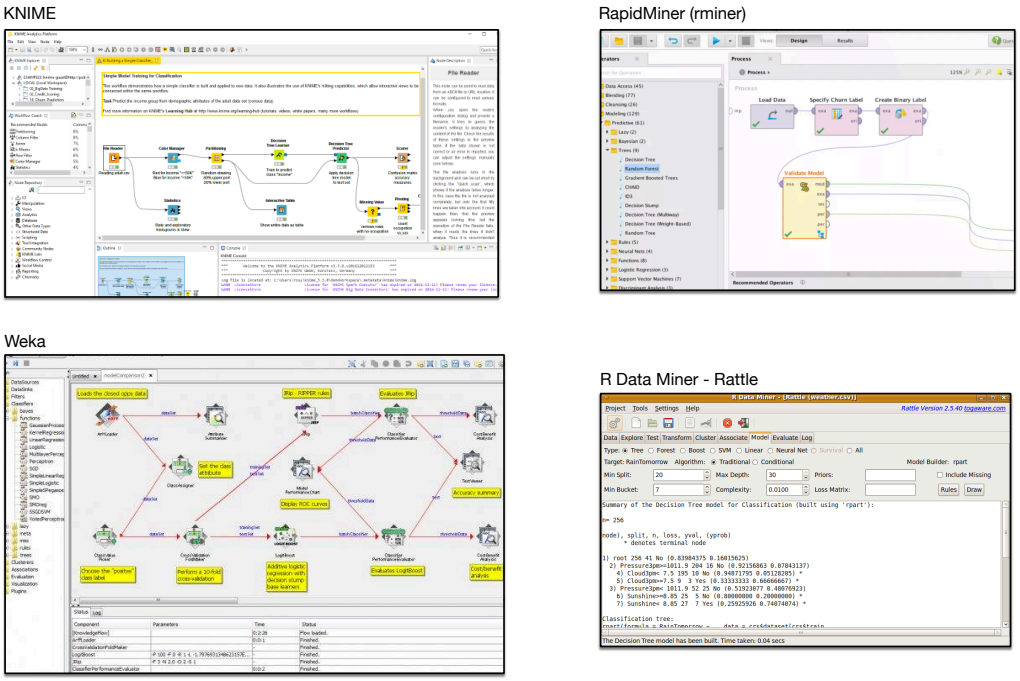

Figura 4.1: Herramientas visuales para el trabajo con minería de datos.

cionales[6] y cubre prácticamente todas las áreas del conocimiento, y de modo especial una gran parte de esos paquetes están destinados a resolver problemas de minería de datos. También en una posición discreta se encuentra *Julia* [Bezanson et al., 2017], un lenguaje de programación dinámico de alto nivel y con buen rendimiento para la computación técnica que compite con *R* y *Python*.

La mayoría de ellos se limitan a las capacidades de procesamiento de un único ordenador o de un conjunto limitado de ordenadores, que no soportan una escalabilidad flexible. Un problema importante con el análisis de datos es el volumen de información que se maneja, lo que hace imposible utilizar herramientas o plataformas regulares para procesar dichos volúmenes de datos. Para solventar este reto tecnológico una coordinación entre hardware y software es fundamental.

El procesamiento masivo de datos y la extracción de conocimiento está liderado por varias plataformas, entornos de trabajo distribuidos y altamente

[6]R tiene una vista de tareas («task view») con el objetivo de proporcionar una orientación sobre qué paquetes son relevantes para las tareas relacionadas con un tema determinado: `https://cran.r-project.org/web/views/`

escalables como *Hadoop* [Vavilapalli et al., 2013] y *Spark* [Zaharia et al., 2010]. Todas ellas son de código abierto y tienen un gran ecosistema de aplicaciones sobre el paradigma *Map-Reduce* [Fernández et al., 2014], que cubren gran parte de los problemas para grandes volúmenes de datos. Para el almacenamiento de datos altamente distribuido y tolerante a fallos utilizan el sistema de ficheros distribuido *HDFS*. Bibliotecas como *Mahout* [Owen et al., 2011] o *Machine Learning Lib* (*MLLib*) [Meng et al.,] forman parte de estas plataformas, y permiten añadir nuevas funcionalidades para la ejecución de algoritmos y aprendizaje automático. Dentro de la plataforma *Spark*, es posible trabajar con lenguajes como *Python*, *Scala*, *Java* o *R* (*SparkR*), lo cual permite integrar de forma sencilla cualquier desarrollo de minería de datos en otras plataformas o en otros servicios. Además, herramientas como *KNIME* o *Weka* han sido adaptadas a versiones con capacidades de computación distribuida [Kholod et al., 2016].

Cloud Computing se ha convertido en una plataforma poderosa para el procesamiento a gran escala de datos, eliminando la necesidad de mantener costosos centros de cálculo e infraestructuras. Un porcentaje significativo de empresas e instituciones están migrando sus datos y aplicaciones a servicios en Cloud Computing para poder gestionar la demanda de procesamiento de sus datos y extraer conocimiento de los mismos, dentro de los entornos de análisis que los proveedores ofrecen. La mayoría de los proveedores de Cloud Computing como *Amazon* (*AWS*), *Google* (*Google Cloud*), *IBM* (*Watson*) o *Microsoft* (*Azure*) [Marozzo et al., 2016], [Shadroo and Rahmani, 2018], ponen a disposición de los clientes su infraestructura y sus recursos informáticos con el objetivo de ofrecer servicios de minería de datos. Estos proveedores y servicios contienen una amplia gama de algoritmos disponibles para ser utilizados desde una API en la web o a través del despliegue de infraestructuras completas con *Hadoop* o *Spark* donde realizar entrenamiento de datos, o creación y explotación de modelos para integrarlos en otras aplicaciones.

4.3. Arquitectura de despliegue para servicios de minería de datos en Cloud Computing: OC²DM

En este capítulo presentamos y describimos una arquitectura de despliegue, denominada *OC²DM: Open Cloud Computing Data Mining platform.*

OC²DM es una plataforma de código abierto[7] que permite ejecutar algoritmos de minería de datos como servicios en Cloud Computing, proporcionando una interfaz común para el flujo de trabajo de la minería de datos para todo tipo problemas y volumen de datos. *OC²DM* permite definir algoritmos y funciones como servicios que pueden ser consumidos desde una API de tipo *RESTful*. Permite además, incluir en la definición todos los aspectos clave a tener en cuenta dentro de un servicio de este tipo sobre Cloud Computing, tal y como se especifica en *NIST*[8] [Mell, Peter et al., 2011]. En el núcleo de *OC²DM* se consideran aspectos como la gestión de negocio, el establecimiento de precios (consumo, uso de regiones o instancias), los acuerdos de nivel de servicio, la compensación u otros elementos tales como la composición del servicio, el modelado del flujo de trabajo, la autenticación, la interacción con el servicio o el catálogo de servicios, entre otros.

La arquitectura de despliegue proporcionada por *OC²DM* ha sido diseñada para un alto grado de escalado de la infraestructura computacional subyacente, integrando una variedad de *frameworks* y plataformas de procesamiento masivo de datos, como *R* o *SparkR*, *Hadoop*, o *Spark*, entre otras. Además, permite la incorporación de prácticamente cualquier implementación de algoritmo adicional (en *R*, *Scala* o *Python*). *OC²DM* ofrece por un lado una herramienta para la descripción de los servicios de minería de datos y por otro lado un motor de procesamiento de los algoritmos de minería de datos como servicios de forma totalmente transparente, sin tener que preocuparse por el escalado de los mismos, dejando a *OC²DM* la responsabilidad del escalado de acuerdo con la demanda del procesamiento en cada momento. En las siguientes subsecciones se detallará la arquitectura y todos los componentes básicos de OC²DM.

4.3.1. Servicios de minería de datos en Cloud Computing

Los proveedores de Cloud Computing, gracias a su vasta infraestructura de recursos de computación, almacenamiento y redes, son los entornos naturales para el despliegue de plataformas de procesamiento de datos para *Big Data*. Desde hace algunos años, el auge de los servicios de minería de datos en Cloud Computing han ido ganando fuerza. Cloud Computing suscita gran interés para las empresas, ya que no tienen que soportar el mantenimiento de

[7]Repositorio OC²DM: https://github.com/dicits/occml
[8]National Institute of Standards and Technology

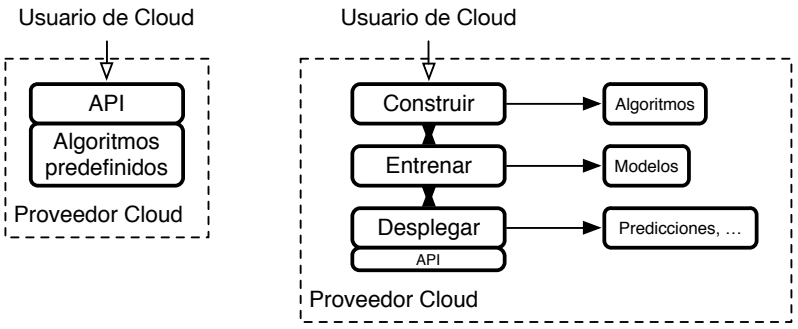

Figura 4.2: Modelo de trabajo con los servicios de minería de datos en Cloud Computing.

dicha infraestructura, y dejan que sea el proveedor de servicios el que realice dichas tareas. Los servicios de minería de datos que ofrecen los proveedores de Cloud Computing cubren un amplio espectro, y albergan una gama de servicios muy amplia que va desde una API completa para la ejecución de algoritmos de minería de datos (*Software-as-a-Service* o *Function-as-a-Service*), plataformas completas de análisis de datos (*Platform-as-a-service*) e incluso infraestructura informática (*Infrastructure-as-a-service*) como servicio.

En este capítulo nos centraremos en los dos primeros (ver Tabla 4.2; por un lado, los servicios de computación en nube que permiten al usuario consumir los servicios para la ejecución de algoritmos y funciones de minería de datos (de modo que, por ejemplo, puedan integrarse desde otras aplicaciones) y por otro lado, los servicios de Cloud Computing en los que el flujo de trabajo se separa en dos etapas: la primera etapa, en la que se realizan las transformaciones, el pre-procesamiento y el modelado, y la segunda etapa, en la que se habilita una API para utilizar este modelo con el fin de realizar predicciones sobre datos. Ambos tipos de servicios pueden utilizarse para trabajar en análisis de grandes volúmenes de datos, gracias al uso de plataformas como *Spark* o *Hadoop* con un soporte flexible de infraestructura como servicio. En la tabla 4.1 se muestran el conjunto de proveedores de Cloud Computing analizados, y detallando parte de las características del servicio de minería de datos ofrecido por cada uno de ellos, así como las propiedades de *OC²DM*.

Proveedor Servicio	Config. Alg.	Consume model	Coste Modelo	Fuentes	IaaS	Escalado	SLA
Amazon Sage Maker	Sí	EndPoint	T,I,R,S	CSV	CPU,GPU	M, D	MTU
	Incluidos: Blazing Text, DeepAR Forecasting, Factorization Machines, Image Clasification, Kmeans, LDA, LL, Neural Topic Model, PCA, Random Cut Forest, Seq2Seq, XGBoost						
Microsoft Azure ML Studio	–	EndPoint / Web Services	T,I,R,S	CSV	CPU	M	MTU
	Incluidos: Averaged prerceptron, Bayesian Linear Regression, Boost Decision Trees, Decision forest, FFQ regression, KMeans, Linear regression, Logistic regression, NN regression, One vs All, Ordinal regression, Poisson regression, PCA, SVM						
GooleML Engine	Sí	EndPoint / Web Services	T,I	-	CPU,GPU	M	MTU
	Incluidos: Algorithms from SciKitLearn, TensorFlow, Keras						
IBM Watson	Y	EndPoints / Web Services	T,I,R,S	CSV	CPU	M	MTU
	Incluidos: MLLib algorithms and WML (Watson Model Library)						
Algorithmia	Sí	EndPoint	-	CSV	CPU	D	MTU
	Incluidos: Algorithms bundles; image recognition, sentiment analysis, etc						
DataBricks	Sí	Enpoint / WS	-	CSV	GPU	D	MTU
	Incluidos: MLLib						
OC²DM	Y	EndPoints	-	CSV	CPU	D	Custom
	Incluidos: J48, Cor, DBScan, Hierarchical Clustering, KMeans, Logistic Regression, Linear,Regression, GLM, NaiveBayes, Optics, RandomForest, Spec, SVM, Time Series Algorithms (ETS, DTW, ARIMA, FastShapelet, Tests: [Shaphiro, Jarque,...]), Multilayer Perceptron, and MLLib						

Tabla 4.1: Tabla comparativa de las características de los diferentes proveedores de servicios de minería de datos en Cloud Computing.

4.3.2. Definición semántica de servicios de minería de datos en OC²DM con `dmcc-schema`

Uno de los elementos centrales del OC²DM es la definición y descripción de los servicios de minería de datos. La definición de servicios se ha abordado desde diferentes perspectivas, tanto sintácticas como semánticas. A nivel sintáctico mediante el uso de SOA [Newcomer and Lomow, 2005] y *XML* [Bray et al., 1997] y otros lenguajes de definición de servicios como *WSDL* [Christensen et al., 2001], *WADL* [Hadley, 2006] o *UUID* [Bellwood et al., 2002] entre otros. El uso de lenguajes de nivel semántico para la definición de servicios, permite hacer el modelado de forma versátil, graduando la escala de detalle que se necesite en cada momento, y dando una enorme flexibilidad para capturar características funcionales y no funcionales, que los lenguajes sintácticos no alcanzan. Dentro de los lenguajes semánticos para la definición de servicios, hay bastantes propuestas, tales como *OWL-S* [Martin et al., 2004], *WSMO* [Domingue et al., 2005], o *SAWSDL* [Kopeckỳ et al., 2007]. Propuestas más completas como *MSM* [Taheriyan et al., 2012], *USDL* [Kona et al., 2009] y otras propuestas generales como *Linked-USDL* que permiten abordar aspectos específicos del Cloud Computing como la gestión del negocio, los precios, los acuerdos de servicio, etc. utilizando tecnologías semánticas y las directrices de *Linked Data*. En la sección 3.2 se hace un estudio mucho más amplio de las diferentes propuestas para la descripción de servicios.

En cuando a la definición y descripción de los servicios de minería de datos que albergará el catálogo de la arquitectura de despliegue propuesta por *OC²DM*, se ha optado por utilizar tecnología semántica, adoptando la propuesta `dmcc-schema` detallada en el capítulo 3. `dmcc-schema` es una propuesta semántica que utiliza *Linked Data* para combinar diferentes esquemas y vocabularios con el objetivo de componer el servicio. Una de las ventajas de utilizar el lenguaje semántico es la riqueza con las que los elementos, entidades y relaciones se definen, y que además, junto con los datos enlazados, mejoran los esquemas ya desarrollados. La definición de los servicios utilizando el esquema `dmcc-schema` describe de forma integral un servicio de minería de datos en Cloud Computing, que contiene todos los componentes con los que un servicio de Cloud Computing se identifica, tales como acuerdos del nivel de servicio, tarifas y precios, experimentación de minería de datos, flujo de trabajo, interacción, autenticación o catálogo, entre otros.

En la Figura 3.1 se representa el esquema general de definición del servicio utilizando `dmcc-schema`. Los servicios de minería de datos en OC²DM se

pueden definir con dos lenguajes: *RDF/Turtle* [Beckett et al., 2014] o *JSON-LD* [Lanthaler and Gütl, 2012]. Con `dmcc-schema` se define una abstracción importante en aspectos de despliegue de servicios o infraestructura, que no se desarrollan en la definición del servicio, dejando toda la lógica y funcionalidad del servicio a otros módulos de la plataforma que se encargan de transformar la definición del servicio en un despliegue totalmente funcional de operaciones de minería de datos sobre Cloud Computing.

Cada una de las descripciones de los servicios de minería de datos, ya sea en formato *RDF/Turtle* o *JSON-LD*, deben estar disponibles para ser descubiertas, de modo que puedan ser integradas en el catálogo de servicios de OC²DM. Cada vez que la arquitectura de despliegue se pone en funcionamiento y también de forma automática, se realiza un comprobación de las descripciones de servicios instaladas, de modo que si existen nuevas las añade al catálogo de servicios de OC²DM, creando todas las estructuras y "EndPoints" necesarios para poder empezar a trabajar con el servicio. En las siguientes secciones se detalla como se realiza el proceso de captura y descubrimiento de los servicios, así como la asimilación de los mismos dentro de la plataforma.

4.3.3. Plataforma para arquitectura de despliegue de servicios

La arquitectura de despliegue de servicios que se ha desarrollado a nivel general consta de dos elementos clave: el sistema encargado de la gestión del catálogo de servicios de minería de datos y el sistema que gestiona la ejecución de los servicios a todos los niveles. En la Figura 4.3 se puede ver el diagrama general donde aparece ambos sistemas. Tanto la gestión del catálogo como la gestión de la ejecución de los servicios funcionan debajo de un servidor web que es el que provee del acceso a las APIs de cada uno de estos componentes desde el exterior y es el que comunica con los demás módulos de OC²DM.

El sistema de gestión del catálogo de servicios, permite descubrir nuevos servicios de minería de datos que estén disponibles, publicar el catálogo de servicios y habilitar los puntos de acceso al servicio desde las interfaces externas que se proveen al usuario. Dentro del sistema de gestión de la ejecución de los servicios, existen otros subsistemas (serán detallados en las siguientes párrafos) que son responsables de tareas relacionadas con la ejecución final de los servicios, y con la gestión de la infraestructura de computación a varios niveles (desde el aprovisionamiento y el almacenamiento masivo distribuido,

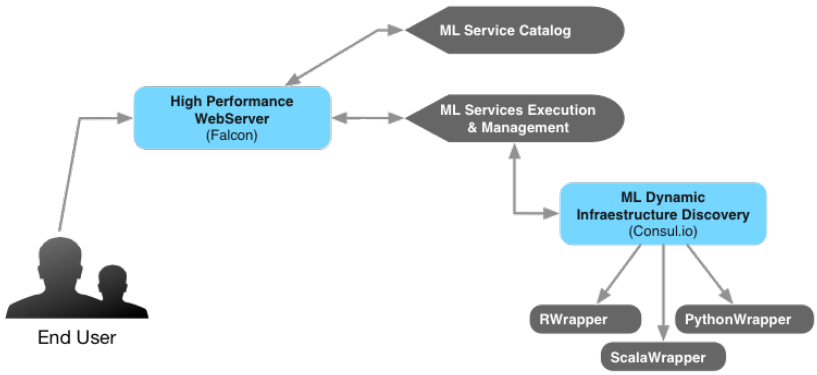

Figura 4.3: Esquema general de la arquitectura de OC²DM.

al despliegue de los servicios).

El esquema mostrado en la Figura 4.4 recoge con más detalle los dos elementos clave de OC²DM junto con todos sus módulos principales. El esquema de módulos funcionales incluye por un lado el servicio de definición y descubrimiento de servicios de minería de datos (*Service definition and discovery*) y por otro lado el servicio de ejecución y almacenamiento (*Service execution and Storage*).

4.3.3.1. Service Catalog

Esta unidad funcional llamada *Service Catalog*, corresponde al servicio de descubrimiento e interacción con los consumidores de servicios de minería de datos ofrecidos por OC²DM. Esta herramienta cuenta con varias utilidades adicionales, pero la más importante recae en la capacidad de ofrecer un catálogo de servicios de minería de datos que puede ser descubierto para el entorno en el que se ejecuta OC²DM. Las funciones clave de esta unidad funcional son las siguientes:

Descubrimiento de servicios de minería de datos. Permite que el catálogo de servicios ofrecidos por la plataforma donde se instala OC²DM pueda ser explorado de forma programática siguiendo el esquema `dmcc-schema` y se obtenga el conjunto de servicios de minería de datos disponibles en la plataforma. Esto ofrece una ventaja importante, ya que permite la posibilidad de encontrar servicios simplemente consultando un catálogo y utilizando el esquema de definición de servicios, garantizando la completa portabilidad y migración.

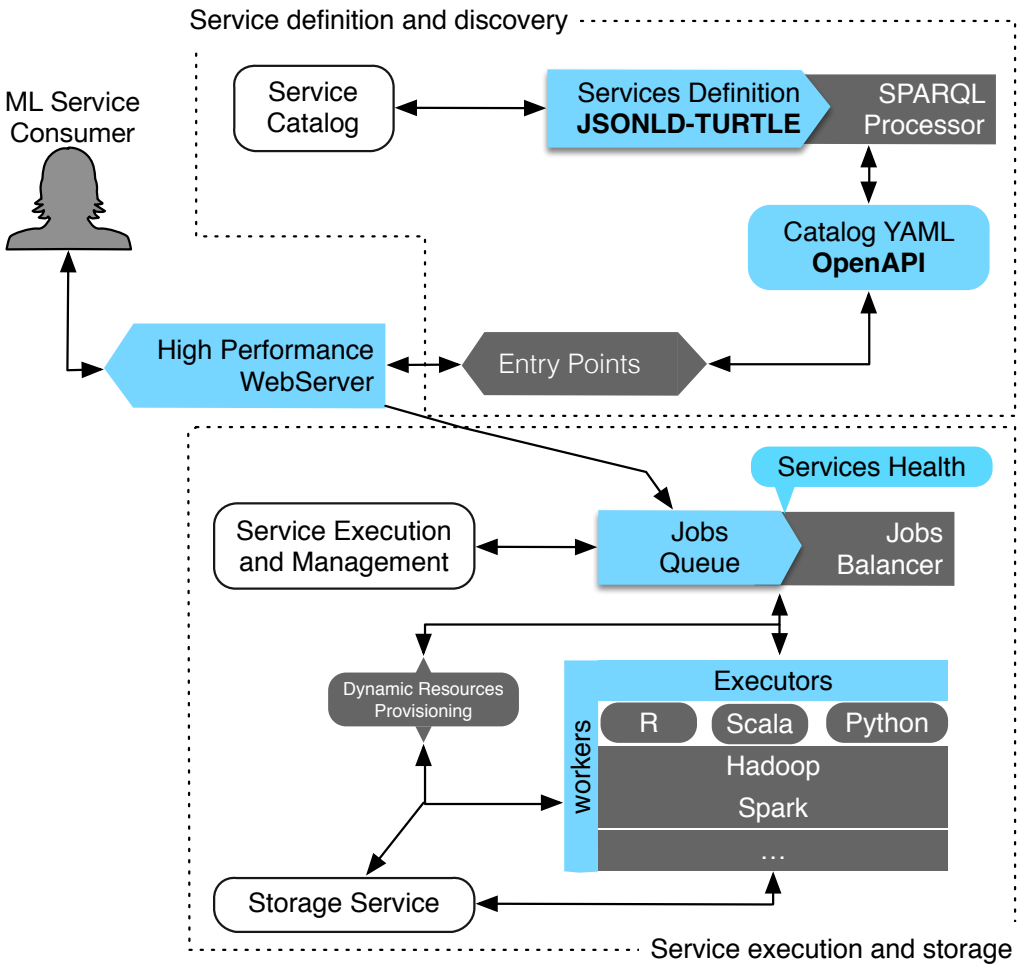

Figura 4.4: Esquema de los módulos funcionales de OC²DM.

Publicación de servicios de minería de datos. Los servicios ofrecidos por la plataforma se publican de forma dinámica. Cuando se crea una nueva definición en *RDF/Turte* o *JSON-LD* de servicio de minería de datos utilizando `dmcc-schema`, esta definición se procesa y se convierte en una nueva estructura a través de la cual es posible consumir el servicio desde una API (*RESTful*, *WSDL*, etc.). El procesamiento de la descripción de los servicios se hace con herramientas de consulta semánticas, como SPARQL. Gracias a este procesador de elementos semánticos se extraen todas las propiedades, entidades y relaciones de las definiciones de los servicios. En la Tabla 4.2 se pueden ver algunos de los elementos que se extraen del servicio.

Habilitación de puntos finales para el catálogo de servicios. Cuando se extraen mediante SPARQL todos los elementos que forman un nuevo servicio de minería de datos, hay unas clases en la definición del servicio que se traducen a "EndPoints". Esto es, elementos de la descripción del servicio que automáticamente generan una API de tipo *RESTful* con URLs que conectan la especificación del servicio con puntos de acceso e interacción con la plataforma. Por ejemplo, la parametrización de un algoritmo o función (entradas de la función: conjuntos de datos, parámetros de invocación, etc.), supone la creación de diferentes APIs de tipo *RESTful* para su gestión. El servicio de catálogo se encarga de realizar esta tarea de forma automática.

Provisión del esquema de definición del servicio. Además de ofrecer el catálogo de servicios que se encuentran disponibles en la plataforma. El servicio de catálogo ofrece las herramientas necesarias para que las nuevas instancias de servicios que se integran en OC²DM puedan ser validadas contra el esquema `dmcc-schema`, de modo que otros proveedores, usuarios o incluso intermediadores de servicios (*Brokers*) puedan construir servicios a partir de esta definición.

Elementos de definición del servicio	**Entidades en OC²DM**
Datos del proveedor	Configuración del servicio
Salida del algoritmo (modelo)	EndPoint API RESTful
Parámetros del algoritmo (input)	EndPoint API RESTful
Precios	Estructuras de tarificación
SLA	Contratos asociados al servicio
Entradas de datos del algoritmo	EndPoints de almacenamiento
Autenticación	Esquema de acceso al servicio

Tabla 4.2: Equivalencia de varios de los elementos de definición de servicios con entidades que se generan en OC²DM.

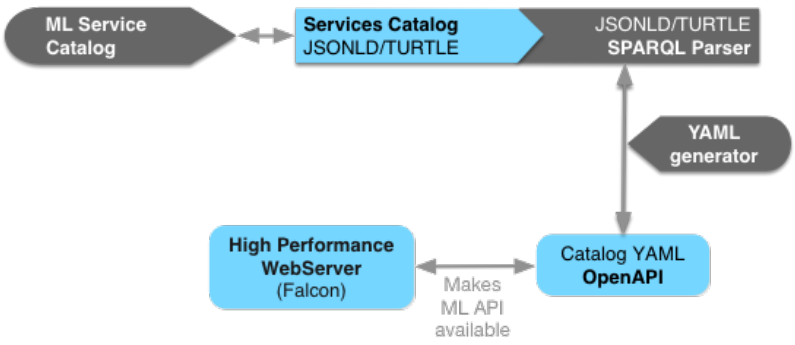

Figura 4.5: Esquema de creación de catálogo a partir de una definición de servicio de minería de datos.

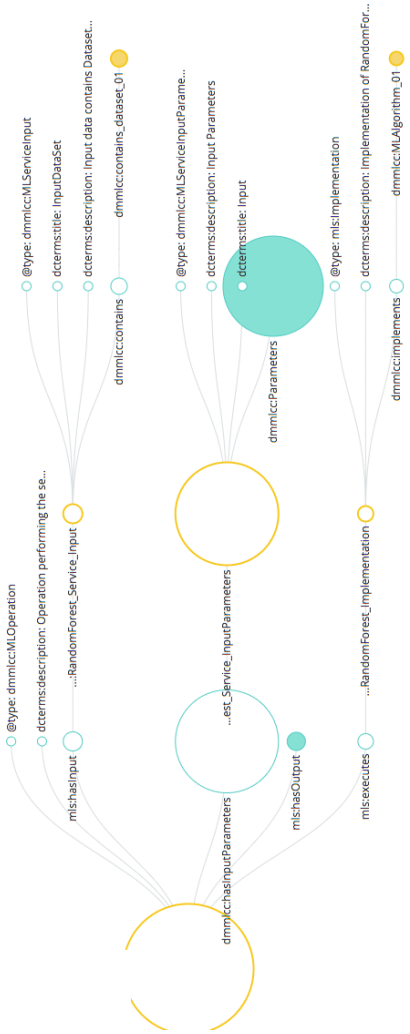

Figura 4.6: Diagrama con el detalle de parte de los nodos del modelado semántico con *JSON-LD* para el código del ejemplo OC²DM 4.2.

En el diagrama de la Figura 4.5 se detallan las operaciones y entidades que intervienen en la captura de un nuevo servicio de minería de datos para OC²DM. Como se observa el proceso consta de varias etapas que se detallan a continuación:

1. Se crea un nuevo servicio a partir de una plantilla de modelo de servicio, por ejemplo con JSON-LD. Esta plantilla de modelo, contiene el esqueleto mínimo para poder realizar un modelado de un servicio, que consta del contexto `@context` y del grafo con las definiciones y relaciones semánticas `@graph` (ver Listado 4.1). Una vez creada la estructura, la plantilla del Listado 4.2 contiene todos los elementos que se van a describir del servicio conforme con `dmcc-schema`. En este ejemplo se definen los datos básicos del servicio y del algoritmo *Random Forest* que se creará como servicio (línea 1-9). Además, se incluyen algunos elementos para definir como el tipo de autenticación, y la operativa `dmmlcc:hasOperation:` {...} del servicio en materia de interacción: a) entrada de datos `"mls:hasInput"`:{...}, b) entrada de parámetros al algoritmo `"dmmlcc:hasInputParameters"`:{...} y c) salida `"mls:hasOutput"`:{...}. En la figura 4.6 se puede ver un extracto de los nodos del modelado.

2. Una vez creado el servicio, el procesador del catálogo se encarga de leer la definición del mismo. Para ello conecta con el motor de SPARQL que realiza todas las consultas pertinentes para extraer la información necesaria que permite componer algunos de los elementos que conectarán el servicio con el exterior y que habilitarán todos los "EndPoints" necesarios para explotar el servicio. Por ejemplo, para extraer los datos básicos de los parámetros de entrada del algoritmo, el procesador de SPARQL utiliza una consulta como la que aparece en el Listado 4.3.

3. Cuando el procesador de SPARQL obtiene toda la información de cada elemento que se ha definido, construye dos estructuras:

   a) Genera una API compatible con OpenAPI v2, con todos los puntos de interacción del servicio a través de URLs, parámetros HTTP, etc. que conectan con cada uno de los módulos de ejecución que más adelante han de ser definidos en cuanto a funcionalidad para cada implementación específica.

   b) Almacena todos los elementos de gestión del servicio en el catálogo de OC²DM. De este modo es posible conocer el tipo de autenticación que ser usará para el servicio, la tarificación, etc.

4. Por último, el servicio web es el que utiliza los modelos de OpenAPI generados en el paso anterior para habilitar todos los interfaces de explotación del servicio.

```
1  "@context": {
2   "dmmlcc": "http://dicits.ugr.es/dmmlcc#",
3         "dcterms": "http://purl.org/dc/terms/#",
4     "mls": "http://www.w3.org/ns/mls#",
5     ...
6   },
7   "@graph": {
8   ...
9   }
```

Listado 4.1: Esqueleto mínimo de la definición de un servicio con JSON-LD

```
1  "@graph":  [
2   {
3    "@id": "dmmlcc:RandomForest_Service",
4    "@type": "dmmlcc:MLService",
5    "dcterms:description": "Implements Breiman's RF",
6    "dcterms:creator":"Manuel Parra",
7    "dcterms:created":"2017-05-06",
8    "dcterms:publisher":"DICITS_ML",
9    "dcterms:title":"RandomForest",
10   "dmmlcc:hasAuthentication": {},
11   "dmmlcc:hasOperation": {
12   "mls:hasInput":{...},
13   "dmmlcc:hasInputParameters":{...},
14   "mls:hasOutput":{...}
15  },
16  "mls:executes":{...},
17   ...
18        }
19 ]
```

Listado 4.2: Plantilla de modelado de servicio en OC²DM con JSON-LD

```
1     PREFIX dmmlcc: <http://dicits.ugr.es/dmmlcc#>
2     PREFIX waa: <http://purl.oclc.org/NET/WebAuthentication>
3     PREFIX mls: <http://www.w3.org/ns/mls>
4     SELECT  ?params  ?description ?mandatory ?defaultvalue
5     WHERE {
6       ?mlservice dmmlcc:hasInputParameters
```

```
7         ?mlserviceinputparameters .
8         ?mlserviceinputparameters dmmlcc:Parameters ?params .
9         ?params dmmlcc:mandatory ?mandatory .
10        ?params dcterms:description ?description .
11        ?params dmmlcc:defaultvalue ?defaultvalue .
12      }
13  }
```

Listado 4.3: Consulta sobre la definición de servicios de los elementos de entrada del servicio con SPARQL

4.3.3.2. Service Execution and Management

Una vez definidos los servicios del catálogo, la siguiente parte a desarrollar es el sistema que se encarga de la gestión y la ejecución de los trabajos de procesamiento de los algoritmos y funciones que se ejecutarán finalmente en la infraestructura de computación. Este sistema tiene varias funcionalidades adicionales, como el control y la gestión de la cola de trabajos enviados a ejecutar a la infraestructura y por otro lado el balanceador de trabajos, que se encarga de repartir la carga de trabajo en la infraestructura donde está desplegado OC²DM. Las funciones principales que lleva a cabo el servicio son las siguientes:

Ejecución y gestión de servicios. Un elemento que queda por conectar desde la definición de los servicios, es el que indica al servicio que implementación de algoritmo debe ser ejecutada para dar cobertura al servicio en cuestión. En la definición semántica de los servicios de minería de datos no se indica cuál será la implementación del algoritmo o la función. Cada servicio y cada algoritmo está identificado por un nombre único, una descripción, etc. Existen muchas implementaciones de algoritmos en diferentes lenguajes, bibliotecas, o plataformas de computación, de modo que de un algoritmo *Random Forest*, existen diversas implementaciones en *R*, en *Python* con `scikit-learn` y en *Scala* dentro de la *MLLib* de *Spark*. Por consiguiente la definición de servicio es completamente agnóstica, no conociendo a priori que implementación será la que se utilizará para dar soporte a un algoritmo de *Random Forest*.

En OC²DM es necesario realizar la conexión entre el "Entry Point" del servicio que ha sido generado y la lógica que permite realizar la ejecución final en una implementación concreta de un algoritmo dentro

de un lenguaje o plataforma soportada por OC²DM. Esto quiere decir que si hemos definido un servicio de *Random Forest*, podemos tener en OC²DM dos (o más) implementaciones diferentes, una desde *R* y otra desde *Scala*, por ejemplo. Es el usuario administrador de la plataforma de OC²DM es el encargado crear el código necesario para seleccionar una u otra implementación. Para facilitar esta tarea, se dispone de una biblioteca a través de la cual se puede:

- Obtener la parametrización del servicio. Esto es, todos los parámetros de entrada del servicio, parámetros de los algoritmos, características de los datos de entrada, salida de datos, etc.
- Componer cada una de las implementaciones. Para ello, la biblioteca incluye funciones de «wrappering» para adaptarse a prácticamente a cualquier lenguaje o plataforma para la implementación. Permite por ejemplo directamente implementar la funcionalidad del servicio con el algoritmo programado directamente *Python* (lenguaje nativo de OC²DM). También es posible utilizar los "wrappers" para realizar una implementación en *Scala*, *R*, *C++*, o virtualmente en cualquier lenguaje que esté disponible en la plataforma OC²DM.
- Seleccionar la implementación. Si el administrador solo tiene una implementación disponible de un algoritmo, este terminará ejecutándose siempre en esa implementación concreta. Sin embargo, si la plataforma OC²DM está funcionando con soporte para *Spark*, entonces valdría la pena crear una nueva implementación que usase la biblioteca *MLLib* para programar la llamada al algoritmo del servicio, con lo que se tendrían dos implementaciones, que programáticamente podrían ser seleccionados según las características del problema a resolver, tamaño de los datos de entrada, etc., con lo que para determinadas condiciones usaría una u otra.

La Figura 4.7 indica un flujo de trabajo desde la descripción de servicio, a la creación de la OpenAPI, y luego la implementación requerida para la ejecución del servicio en particular. El diagrama detalla las operaciones que se tienen en cuenta para la ejecución, por ejemplo, de un algoritmo de servicio cuya implementación final está en *R* o *SparkR*.

Planificación de trabajos. La arquitectura de despliegue OC²DM es un entorno basado en Cloud Computing, donde múltiples usuarios pueden interactuar con la plataforma de forma directa, consumiendo servicios de minería de datos. Debido a esto es necesario realizar una planificación de los trabajos. Cada trabajo que entra en OC²DM se pone en un

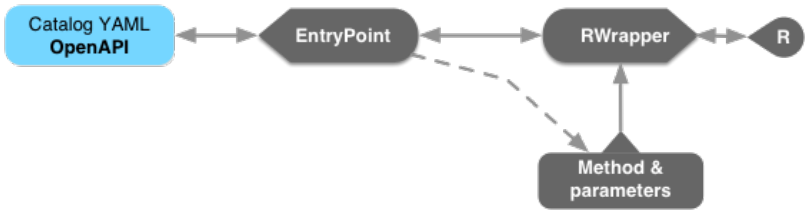

Figura 4.7: Diagrama de la API compatible con OpenAPI y punto de entrada para un servicio donde se tiene una implementación de un algoritmo en R.

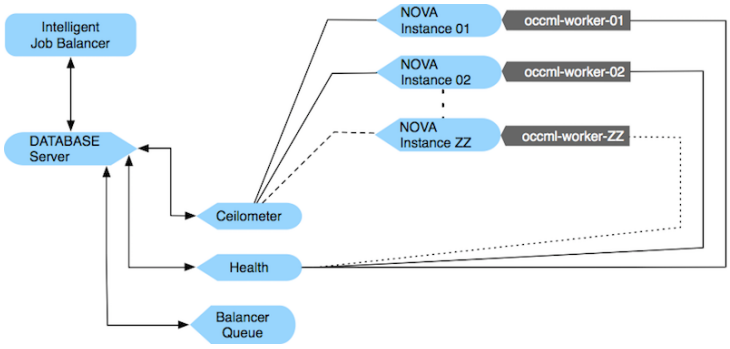

Figura 4.8: Componentes de decisión del balanceador de trabajos.

cola de trabajos y se valida para su ejecución en función de las propiedades del trabajo y de las condiciones de carga de la infraestructura que se está usando. La cola de trabajos contiene todas las solicitudes procedentes de las entidades consumidoras de servicios (los usuarios) y almacena toda la información relativa a las funciones a ejecutar, parametrización, datos, etc.

El subsistema llamado *JobBalancer* se encarga de tomar los trabajos de la cola y planificar una ejecución coherente, basada en diferentes parámetros de rendimiento de la infraestructura (ver Figura 4.8). Estos parámetros de desempeño son proporcionados por cada uno de los nodos de los "workers" que son consultados continuamente por un servicio que recupera las estadísticas de rendimiento de cada una de las unidades de computación que forman la infraestructura de soporte de OC²DM. Entre los parámetros de salud que se recuperan se encuentran datos simples como procesos en ejecución de OC²DM, estadísticas de CPU, acceso al disco y almacenamiento de datos, entre otros. El *JobBalancer* puede configurarse para diferentes tipos de repartos de trabajo; por defecto se utiliza un algoritmo sencillo de *Round-Robin* para colocar los trabajos que están listos para ejecutarse en las unidades de

computación de forma circular.

Cuando se trata de trabajos que necesitan una computación distribuida, la asignación de los recursos de computo se hace de forma diferente, pues es la plataforma subyacente de computación (*Spark*, *Hadoop*, etc.) la que soporta la ejecución distribuida del servicio. De modo que se pasa el control a cada gestor de recursos concreto de la plataforma distribuida de computación.

Algunos de los mecanismos de balanceo de trabajos disponibles en OC²DM son los siguientes (ver Figura 4.9):

- Al azar. Selecciona aleatoriamente el nodo de destino del trabajo. No se comprueba la carga de trabajo de los nodos.
- Round-robin. Los trabajos se colocan en los nodos siguiendo una distribución de asignación circular. No se comprueba la carga de trabajo de los nodos.
- Round-robin con prioridad de carga de trabajo. Asigna el trabajo al siguiente nodo con menos carga de trabajo. La carga de trabajo de los nodos es comprobada antes.
- Round-robin basado en la historia. Verifica el estado del nodo y el historial de carga del nodo. Si el nodo tiene un historial de carga importante, se pasa al siguiente nodo con carga igual o menor.

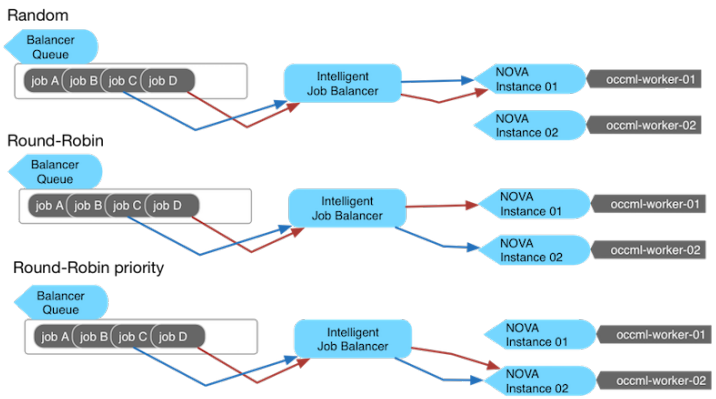

Figura 4.9: Ejemplo de funcionamiento de varios de los métodos de balanceo y distribución de trabajos del *JobBalancer*

Aprovisionamiento dinámico de recursos. Esta función es responsable del escalado dinámico de recursos de computación. Aumenta o disminuye dinámicamente el número de unidades de computación utilizadas

en la infraestructura. Esta función es muy útil para optimizar el uso de los recursos. No es fácil establecer un criterio único para definir la función de incremento o decremento de unidades de computación, por lo que hemos optado por una propuesta híbrida basada en la demanda actual e histórica.

La provisión de recursos de computación no es una tarea trivial. Más si cabe cuando cada unidad de computación debe tener todo el software cargado para poder ser utilizado por los algoritmos que estén implementados, y además si OC²DM tiene soporte para plataformas como *Hadoop* y *Spark*, y por consiguiente *HDFS* para el almacenamiento, la adición de varias unidades de computación requiere de una integración más compleja. Para solventar estos problemas, la creación de nuevas unidades de computo de forma dinámica se hace mediante la utilización de imágenes base que contienen todo el software ya pre-instalado y se inyecta en la configuración (para almacenamiento *HDFS*, *Hadoop*, *Spark*, etc.) en tiempo de instanciación con la parametrización adecuada. De este modo se mantiene un «pool» de instancias pre-configuradas y listas para ser iniciadas o paradas bajo demanda.

Para realizar las pruebas de instalación y despliegue de los servicios de OC²DM se ha usado como infraestructura de recursos de computación, *OpenStack*. La configuración de las pruebas realizadas para el despliegue inicial de la infraestructura para *OpenStack* y sus servicios se realiza desde los ficheros de configuración del aprovisionamiento, algunos de los parámetros son los siguientes:

- OpenStack *Nova*:
  - `start_instances` = 5. Será el número de instancias iniciales de las que se desplegarán como nodos esclavos o ejecutores.
  - `max_scalable_instances` = 20. Instanciaciones máximas hasta donde se puede escalar el sistema. Si el valor es 0, entonces no hay límite.
  - `start_flavour` = `s1.small`. Será el sabor por defecto que se utilizará en las imágenes de inicio.
  - `flavours_available` = `flavours`. Directorio que contiene una lista (*JSON*) de "sabores" (ordenados, de la gama baja a la alta).
  - `allow_dynamic_flavour` = `no`. Permite crear nuevas instancias basadas en sabores con mayor rendimiento. Por defecto es no, por lo que se creará una nueva instalación con el mismo sabor. En caso afirmativo, se crearán nuevas instancias

utilizando los sabores de forma incremental. Por ejemplo: las primeras 5 instancias son *S1.small*, las siguientes se crearán usando un sabor incremental, *S2.small*, *M1.medium*, etc.

- OpenStack *Glance*:
  - `image_base_name` = `fedora−occml` Define la plantilla base desde la que se instanciarán las nuevas máquinas virtuales.
  - `volume_size` = 40 Define el tamaño de las instancias por defecto.

Executors y workers. Estas son las funciones que se encargan de ejecutar el código final correspondiente a la implementación de los servicios de minería de datos en Cloud Computing dentro de OC²DM. Estas funciones transfieren las ejecuciones a los nodos "workers" de la plataforma OC²DM y controlan con qué "executor" se inicia cada servicio, dependiendo de la implementación que corresponda: *R*, *Scala* o *Python*, junto con el *framework* utilizado, por ejemplo *Spark* o *Hadoop*. El "executor" es por tanto el servicio encargado de tomar la implementación escogida desde el «wrapper» del servicio y proveer de la ejecución en el nodo o unidad de computación destino o "worker".

4.3.3.3. Almacenamiento distribuido de los datos

Los servicios y algoritmos que se ejecutan necesitan espacio para almacenar tanto la entrada de datos, la salida (datos, modelos) y los resultados intermedios para el procesamiento de datos. Para el almacenamiento de estos datos, se ha decidido utilizar *HDFS*, principalmente por su atributos de fiabilidad, redundancia y elasticidad. Inicialmente se consideró la opción de integrar el propio sistema de almacenamiento de *OpenStack Swift* [Arnold, Joe, 2014]. *Swift* ofrece servicios de almacenamiento en nube para almacenar y recuperar datos desde una API y además comparte muchas características con *Hadoop*, tales como que está construido para escalar, está optimizado para durabilidad, disponibilidad y concurrencia en todo el conjunto de datos. Pero se descartó puesto que era necesario utilizar un adaptador para convertir los objetos de almacenamiento *Swift* a *HDFS*.

Otro elemento a la hora de considerar *HDFS* es que para que se pueda trabajar con los datos aprovechando la ubicación del procesamiento de datos de nodos, vital para trabajar con algoritmos que se ejecutan bajo el paradigma *Map-Reduce*. OC²DM puede escalar el servicio de almacenamiento *HDFS* disponible simplemente añadiendo más nodos a la plataforma (por ejemplo a

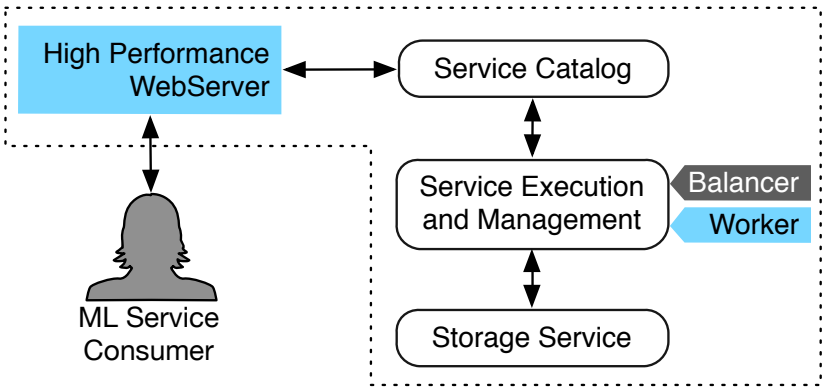

Figura 4.10: OC²DM en configuración para un único nodo.

través del escalado dinámico). Además de los servicios de minería de datos, la plataforma permite este servicio de almacenamiento de datos, necesario para la entrada y salida de información de los algoritmos.

4.3.4. Arquitectura mono-nodo y multi-nodo

La plataforma OC²DM ha sido diseñada para ser altamente escalable, soportando varias configuraciones posibles de replicado y distribución de los sistemas que forman la arquitectura de despliegue. Hay dos modos de trabajo, que son: a) una configuración mínima de un solo nodo y b) una configuración de varios nodos o multi-nodo. La configuración de un solo nodo permite probar la plataforma en nodos simples o máquinas virtuales, desplegando el número mínimo de sistemas básicos de OC²DM para el soporte de servicios de minería de datos diseñados para esta configuración. Con este modelo de despliegue es posible conocer con más detalle la definición de los servicios y las implementaciones de los algoritmos en Cloud Computing, así como crear nuevos servicios. En la Figura 4.10 se puede ver el esquema de la arquitectura de un solo nodo que ofrece OC²DM, donde todos los componentes del diagrama 4.4 están en el mismo nodo de cálculo y almacenamiento. Para esta sencilla configuración, sólo es necesario la plataforma R.

El enfoque multi-nodo, como se muestra en la Figura 4.11, ofrece un sistema escalable y preparado para la alta demanda en entornos de Cloud Computing para servicios de computación intensiva de minería de datos. El diagrama de la Figura 4.11 muestra cómo se replican los servicios en diferentes nodos para ofrecer redundancia y alta disponibilidad dentro de la plataforma.

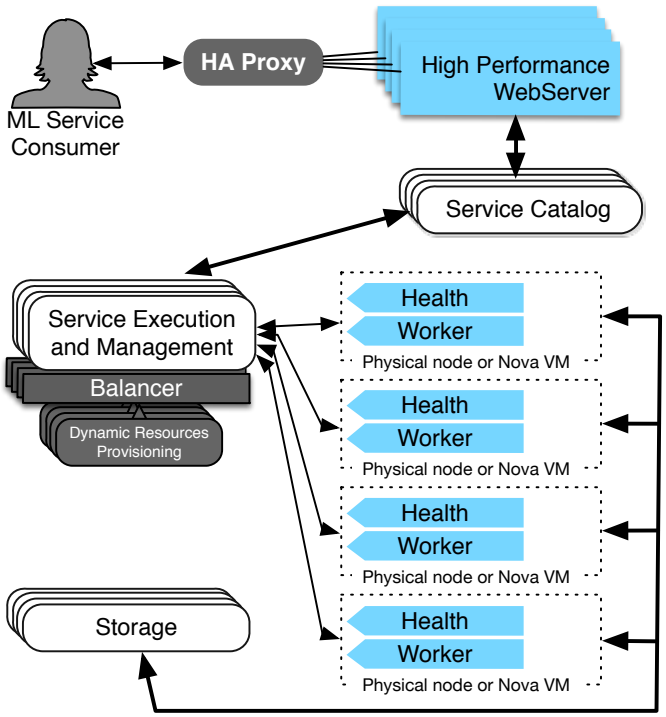

Figura 4.11: OC^{2}DM en una configuración donde se despliega en múltiples nodos.

Como se ha visto, en esta configuración la plataforma dispone de un «pool» de nodos o máquinas virtuales que se inician o paran dinámicamente en función del rendimiento de la plataforma. Cada uno de los nodos contiene servicios específicos para extraer parámetros de salud que serán utilizados para equilibrar la gestión de los trabajos.

4.4. Caso de uso

Para ilustrar el correcto funcionamiento del *OC^{2}DM*, mostraremos cómo se realiza todo el proceso de trabajo con la plataforma, desde la creación de la definición de un servicio de minería de datos hasta su ejecución, pasando por la implementación de la interfaz que conecta la definición con la implementación específica. En este caso de uso se especifica un servicio de minería de datos donde se describe un algoritmo *Random Forest* [Ho, 1998] como servicio. Esta definición de servicio incluye sólo una pequeña parte de la descripción creada con el lenguaje *RDF/Turtle*. Para un ejemplo completo de la descripción del servicio *Random Forest* en OC^{2}DM, puede consultar el

sitio web de la plataforma[9].

La ventaja de utilizar una descripción de servicio utilizando `dmcc-schema` es que separa completamente la definición del servicio, con todo lo que implica (definición de costes, precios, SLA, interfaces de autenticación, regiones, etc), de la implementación y gestión del propio servicio de ejecución del algoritmo.

4.4.1. *Random Forest* como una descripción de servicio en Cloud Computing en OC^{2}DM

Con *RDF/Turtle*, definimos el esqueleto principal de la descripción del servicio según el esquema `dmcc-schema`. Reduciendo el tamaño del ejemplo sólo trataremos los aspectos relacionados con el algoritmo a ejecutar y sus parámetros. Los otros aspectos de la descripción del servicio como autenticación, *SLA*, costes o tarifas, pueden ser estudiados a partir del ejemplo desplegado en capítulo 3.

Primero definimos cada uno de los aspectos del servicio (ver capítulo 3 para una descripción completa) en el Listado 4.4.

```
1  _:MLServiceDicitsRF a dmc:MLService;
2   rdfs:label
3      "ML Service dicits.ugr.es"@en ;
4   dc:description
5      "DICITS ML Service"@en ;
6   dmc:hasInteractionPoint
7      _:MLServiceInteraction;
8   dmc:hasServiceCommitment
9      _:MLServiceSLA;
10  dmc:hasFunction
11     _:MLServiceFunction;
12  dmc:hasAuthentication
13     _:MLServiceAuth;
14  dmc:hasPricingPlan
15    _:MLServicePricing;
16  .
```

Listado 4.4: Ejemplo de servicio para *Random Forest* con *Turtle/RDF*

Ahora nos centramos en los datos relativos a la definición de la función o

[9]http://dicits.ugr.es/linkeddata/dmservices/#dmcc-schema

funciones que ejecutará el servicio, donde hay que describir `dmc:hasFunction _:MLServiceFunction;`, tal y como indica el Listado 4.5.

```
dmc:RandomForest_Function
  a dmc:MLFunction ;
   dmc:hasInputParameters
     dmc:RF_InputParameters;
   dmc:description
    "Operation performing
       the service";
   mls:hasInput
    dmc:RF_Input;
   mls:hasOutput
    dmc:RF_Output .
```

Listado 4.5: Parametrización de los datos de entrada y salida del servicio que luego serán procesados para obtener una OpenAPI.

La definición de `dmc:hasFunction _:MLServiceFunction;` contiene todos los elementos del servicio relativos al algoritmo, en este caso *Random Forest*; la definición de los parámetros de entrada, la entrada de datos y las salidas de datos son los elementos necesarios para la definición del algoritmo. Por ejemplo la parametrización de entrada se definiría del siguiente modo para dos de los parámetros obligatorios (ver Listado 4.6).

```
_:parameter_01
     a ccdm:MLServiceInputParameter ;
        ccdm:defaultvalue "" ;
    ccdm:mandatory "true" ;
    dc:description
     "numeric matrix of data" ;
    dc:title "x"
        .

_:parameter_02
     a ccdm:MLServiceInputParameter ;
        ccdm:defaultvalue "3" ;
    ccdm:mandatory "true" ;
    dc:description
    "either the number of clusters." ;
    dc:title "centres"
        .
...
```

Listado 4.6: Ejemplo de parámetros y características.

Además de definir todos los aspectos relacionados con el algoritmo, es posible también describir cual será la implementación que se utilizará para el servicio. No es necesaria esta especificación, pero si es posible indicarlo de cara a seleccionar cual será el *framework* que se usará de forma especifica para la ejecución interna del servicio. Sin embargo, en el caso que no se seleccione una implementación, será el propio *OC²DM* el encargado de seleccionar la implementación adecuada para cada caso, como se ha indicado en la subsección 4.3.3.2.

```
1 _:implementation01
2       a ccdm:MLImplementation ;
3          ccdm:value "core.R.rf" ;
4          .
5 ...
```

Listado 4.7: Ejemplo de la definición de la implementación.

4.4.2. Implementación y despliegue del servicio de *Random Forest*

En los pasos anteriores hemos definido los servicios mismos con toda su parametrización, pero para implementar la funcionalidad del servicio *Random Forest*, se deben definir una serie de interfaces dentro del *Execution & Management*. Estas interfaces permitirán conectar la descripción del servicio con el código de ejecución en la plataforma OC²DM. En este ejemplo existe una implementación disponible en *R* y *SparkR*, por lo que la interfaz servirá de "wrapper" para el código de ejecución del algoritmo en la plataforma de despliegue destino.

Un nuevo archivo de servicio, por ejemplo llamado `RandomForest.py`, debe definirse dentro de la función, que contendrá la lógica de la función, algoritmo o código que se ejecutará para ese servicio y se desplegará sobre la infraestructura de OC²DM. El framework disponible con OC²DM permite utilizar las funciones de la biblioteca a través de las cuales es posible capturar todos los elementos de la definición semántica del servicio creado (ver Listado 4.8).

```
1 def rf (parameterCore):
2   rf = ro.r("""
3     library("randomForest")
4     dataset = read.csv(file="{0}",
5                 header = TRUE, sep=',')
```

```
6       fit = randomForest({1},
7                          data=dataset)
8       saveRDS(fit, "{2}")
9       """.format(
10        parameterCore.dataset['ruta'],
11        parameterCore.parameters,
12        parameterCore.outputPMML))
```

Listado 4.8: Implementación del algoritmo Random Forest a partir de un wrapper para R.

Sólo los parámetros (nombres, valores por defecto, etc.) se definen una vez dentro de la definición del servicio en *Turtle*. El mecanismo de conversión de la especificación del servicio a la implementación del servicio transforma estos parámetros en el objeto `parameterCore` que contiene la estructura de parámetros del algoritmo definido por el `dmcc-schema`. Para la implementación de este ejemplo, se utiliza una versión de un solo nodo del algoritmo con lenguaje *R*. Esta implementación puede ser modificada para usar la versión distribuida del algoritmo desde la plataforma *Spark* usando el mismo código de listado 4.8, para *R*, *Python* o *Scala* e importando la función correspondiente de *MLlib* en *Spark*.

Con este sencillo ejemplo se puede ver el potencial de la plataforma como una herramienta para el desarrollo de algoritmos y flujos de trabajo más complejos de minería de datos. La plataforma OC^{2}DM ofrece un amplio conjunto de primitivas y algoritmos ya implementados, así como la posibilidad de implementar cualquier algoritmo como servicio.

4.5. Conclusiones

En este capítulo se ha desarrollado una arquitectura para el despliegue de servicios de minería de datos en entornos de Cloud Computing llamada OC^{2}DM. Elementos como la definición y descripción de un servicio completo de Cloud Computing utilizando tecnología semántica y el modelando de las relaciones entre conjuntos semánticos mediante el uso de *Linked Data*, ofrece una importante ventaja competitiva con respecto a las demás propuestas que existen en el mercado. Estas ventajas se centran en la capacidad de permitir la migración y la portabilidad de servicios de un proveedor a otro sin modificar la definición del servicio ni su implementación sobre la plataforma.

Cuando se despliega OC^{2}DM en una plataforma, las definiciones que se

importen desde otro servicio utilizando `dmmc-schema` serán totalmente válidas tanto a nivel conceptual como a nivel funcional para trabajar sobre la nueva plataforma. Esta prueba de concepto desarrollada con *OC²DM* permite por un lado validar la definición abstracta de servicios de minería de datos sobre Cloud Computing, con todos sus componentes clave como precios, SLA, interfaces, regiones o instancias, o experimentación entre muchos otros; y por otro lado permite desplegar esos servicios de forma funcional, sobre infraestructuras heterogéneas de computación, ofreciendo flexibilidad y escalado en el trabajo con minería de datos.

OC²DM ofrece un catálogo de algoritmos y funciones de minería de datos comparable al ofrecido por otros proveedores de Cloud Computing, pero con la excepción de que esta plataforma puede ser utilizada para aprovechar los recursos informáticos e infraestructura disponibles en organizaciones o empresas, para trabajar con servicios de este tipo del mismo modo que se hace en esas plataformas. Otro aspecto clave a destacar es la posibilidad de transformar cualquier algoritmo o funcionalidad existente en un servicio, de forma que pueda integrarse fácilmente en otras plataformas o dentro de otros servicios, a modo de «Building blocks». Así, los servicios pueden estar interconectados entre sí, con lo que lo convierte en una plataforma altamente escalable para crear flujos de trabajo complejos de minería de datos. Otro de los puntos fuertes del OC²DM es que representa un instrumento base muy conveniente para definir flujos de trabajo con datos sin tener conocimientos de programación, puesto que habilita una capa superior o "Front End" de un modo similar a lo que *KNIME* o *Weka* ofrecen, pero con la ventaja que OC²DM está diseñado para trabajar con procesamiento distribuido y abstrayendo la plataforma de computación subyacente.

Capítulo 5

Servicios de intermediación en Cloud Computing para minería de datos

Motivación

En los dos capítulos anteriores se ha desarrollado un vocabulario para la definición de servicios además de una arquitectura de despliegue para servicios de minería de datos y que utiliza como base para la creación de los servicios funcionales de Cloud Computing la descripción propuesta con `dmcc-schema`. De este modo tenemos una herramienta basada en tecnología semántica para la definición de servicios específicos de minería de datos y por otro lado tenemos una plataforma donde esas definiciones cobran sentido creando servicios completos y funcionales en Cloud Computing.

La estandarización en la definición de servicios es un elemento clave para la interoperatibilidad en Cloud Computing; de hecho, una estandarización global en la definición de servicios mejoraría la portabilidad, la migración y la competitividad del mercado de servicios de Cloud Computing, permitiendo a los consumidores una selección mejor y una gestión más eficaz y eficiente de los servicios utilizados. Estas funcionalidades se pueden obtener a través de un servicio de *Brokering* de Cloud Computing. Preguntas como, *¿cuál es el proveedor de servicios de almacenamiento más económico para mis requerimientos?*, o *¿cuál es el servicio con el coste más bajo para hacer una computación de datos sobre varias instancias?*, pueden ser respondidas

por un *Broker* de Cloud Computing.

En esta misma línea, nos preguntamos acerca de la importancia de desarrollar un *Broker* en Cloud Computing para servicios de flujos de trabajo en minería de datos dado el auge y la proliferación de este tipo de servicios en los últimos años por parte de los grandes proveedores. En este nuevo escenario, un *Broker* para servicios de minería de datos puede ser un actor decisivo para la integración de servicios heterogéneos de este tipo ofrecidos por los proveedores de Cloud Computing, sirviendo de punto de acceso único para la gestión, uso y entrega de este tipo de servicios. En este capítulo desarrollaremos el concepto de *Broker* y de modo específico un *Broker* de minería de datos para la agregación e intermediación de este tipo de servicios.

5.1. Introducción

Cloud Computing referencia un modo de proveer servicios bajo demanda siguiendo el modelo de pago por uso, donde Internet y la tecnologías de virtualización de la computación son explotadas para proveer de servicios de computación de un modo muy flexible y totalmente escalable. El modo de suministrar los servicios es configurable y accesible de forma ubicua, donde el modelo de coste de gestión y la administración es contrapuesto al modelo tradicional.

Los servicios ofrecidos en Cloud Computing son de tipos muy variados. La propuesta de consumo de servicios desde Cloud Computing empodera al usuario u organización a utilizar directamente todo el potencial y ventajas que ofrecen el modelo de consumo de pago por uso, en lugar de la inversión en infraestructura, software y mantenimiento del modelo tradicional que se tenía (ver Figura 5.1).

Cloud Computing soluciona problemas como el mantenimiento de la infraestructura, la gestión, la fiabilidad, la disponibilidad y la reducción de costes, ya que están implícitos en la forma de consumir los servicios, dejando al proveedor de los servicios encargado del cumplimento de todas esas cuestiones y dando el control total al consumidor sobre los servicios. El usuario consumidor puede centrarse en el uso y explotación de servicios, y no en toda la problemática de la gestión de la infraestructura del servicio, con lo que se apuesta por una abstracción importante a la hora del despliegue de recursos de computación tanto hardware como software.

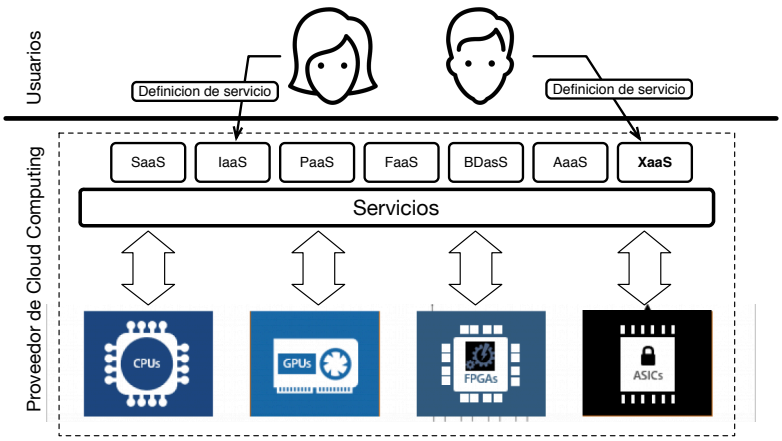

Figura 5.1: Modelo de Cloud Computing para la abstracción de recursos de computación.

El catálogo de servicios ofrecidos por los diferentes proveedores de Cloud Computing sigue creciendo cada día. Sin embargo, cada proveedor alberga su catálogo de servicios en gran medida incompatibles con los demás servicios del mismo tipo o similares, ofrecidos por otros proveedores. Por ejemplo, los servicios de almacenamiento de archivos, como `DropBox`[1], `OneDrive`[2] o `Google Drive`[3], permiten a los usuarios almacenar ficheros y datos de una forma sencilla, cómoda y totalmente escalable. Tal es así que el usuario del servicio final, sólo se preocupa de la gestión de los ficheros en Cloud Computing y no de todo lo que involucra la gestión de esa infraestructura que soporta el almacenamiento, como nodos de computación, procesamiento o almacenamiento distribuido, entre otros. Los tres productos son básicamente lo mismo, pero cada uno de ellos tiene un modelo propietario de acceso a las interfaces y métodos para la gestión de servicio (ver Figura 5.2). Esto no sólo ocurre con este tipo de servicios sino que también se puede extrapolar a gran parte del conjunto de servicios de los proveedores.

A pesar del tremendo desarrollo y adopción de Cloud Computing, aún existe una falta de estandarización y homogeneización generalizada sobre la definición y descripción de servicios ofrecidos desde las plataformas de Cloud Computing. Es esta falta de estandarización la que hace que los proveedores ofrezcan sus servicios a través de sus interfaces cerrados, sin considerar un modelo más abierto que a fin de cuentas mejoraría en gran medida el consumo y la intermediación de los de los servicios, conjuntamente con la mejora de

[1]https://www.dropbox.com/
[2]https://onedrive.live.com/about/es-es/
[3]https://drive.google.com/

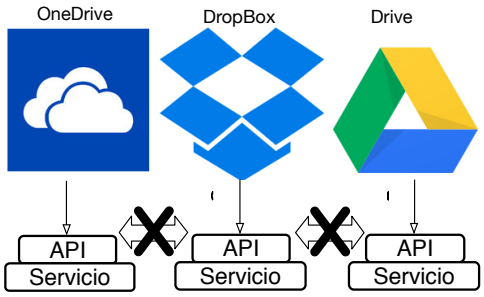

Figura 5.2: Incompatibilidad entre diferentes servicios, proveedores y APIs

la competencia en cuanto a la contratación de los mismos. Es comprensible que cada proveedor de Cloud Computing sea reacio a abrir a un estándar su infraestructura o servicios puesto que eso supondría que el cliente podría cambiar de proveedor de servicio.

Cuando consideramos la migración o una portabilidad de servicios desde un proveedor a otro es necesario que exista una compatibilidad entre las definiciones entre ambos proveedores. Es necesario considerar que la definición y descripción entre cada una de las partes sea normalizada. Pero además es necesario que el acceso y gestión de los recursos de computación necesarios para soportar el servicio estén también estandarizados o al menos sea posible tener un interfaz que abstraiga la gestión de esos recursos para eliminar parte de la heterogeneidad de las infraestructuras que soportan los servicios. Si esto no se llega a alcanzar, entonces el traspaso de los servicios a otros proveedores no está garantizado, con el consecuente coste que supondría el despliegue de un servicio desde cero en un nuevo proveedor.

La evolución imparable y el dinamismo proporcionados por Cloud Computing está produciendo un incremento en los servicios ofertados por los proveedores. La selección de un proveedor de Cloud Computing y de unos servicios no es una tarea sencilla. El volumen de proveedores de Cloud Computing y su catálogo de servicios cada vez es mayor. Los proveedores se enfrentan a importantes cambios en las tendencias de mercado y deben estar preparados para adaptar sus servicios a las necesidades de los usuarios.

La selección de los servicios por parte de los usuarios consumidores está basada en varios criterios, como el coste global del servicio, las limitaciones de los acuerdos de nivel de servicio (donde se incluyen aspectos como la fiabilidad, los tiempos inactividad, las bonificaciones, etc.), la localización de la infraestructura, el almacenamiento, o la diversidad de servicios que es

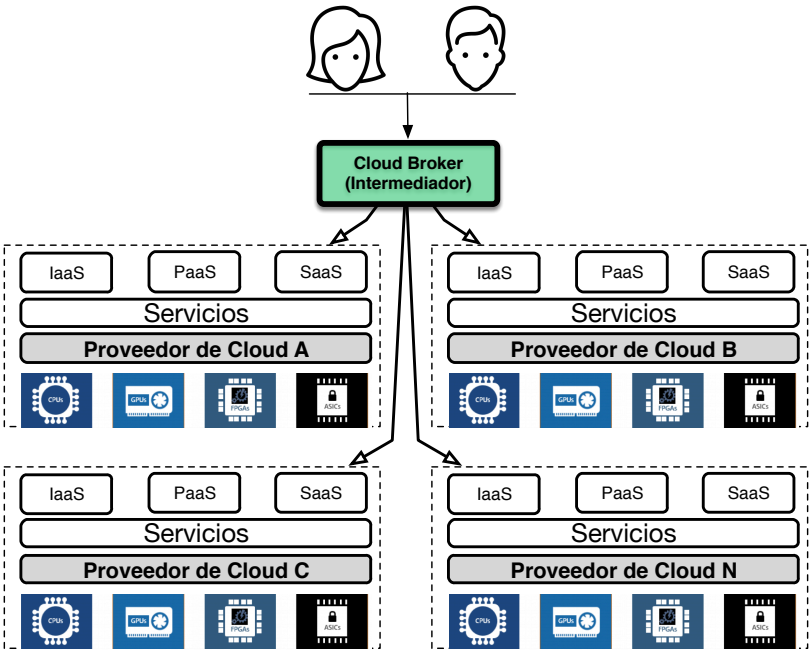

Figura 5.3: Broker: Intermediador entre diferentes proveedores y servicios en Cloud Computing

capaz de integrar el proveedor. Si incluimos todos estos criterios, la elección de un proveedor de Cloud para albergar una plataforma, aplicación o sistema, resulta una operación compleja donde la elección del proveedor de Cloud Computing que satisfaga todos o parte de nuestros requisitos supone un problema adicional.

En este escenario, surge el *Broker* de Cloud Computing también llamado el "intermediador" de servicios. Un *Broker* de Cloud Computing puede actuar como mediador tanto en la selección y compra de servicios, como en la administración directa de los mismos, desde una única entidad que "entiende" a un conjunto de servicios y proveedores, tal y como se observa en la Figura 5.3. Cuando se hace referencia a "entiende" se indica que un *Broker* de Cloud Computing, puede actuar como un intermediario entre dos partes, esto es, entre el proveedor de los servicios y el consumidor de los servicios o cliente y que es capaz de conocer todos los detalles relativos al servicio de los proveedores que incluye (interfaces, costes, integración, etc.). NIST[4] define con las siguientes palabras lo que es un *Broker* de Cloud Computing:

[4]National Institute of Standars and Technology

> *"Un Cloud Broker es una entidad que gestionar el uso rendimiento y entrega de servicios y negocia las relaciones entre proveedores y consumidores de Cloud."*

En un *Broker* de Cloud, el consumidor de los servicios puede consultar y seleccionar cuales son los servicios de Cloud Computing mejores y más efectivos en coste, y el proveedor de los servicios puede tener mas oportunidades para mejorar sus servicios y maximizar el beneficio. El *Broker* por tanto tiene capacidad para poner en contacto ambas partes, proveyendo de elementos como el arbitraje, la agregación y la intermediación de cara al usuario de los servicios ofrecidos por el *Broker*.

La arbitración de servicios consiste en la capacidad de mejorar las características generales de los servicios, ofreciendo más flexibilidad en la selección de servicios, por otro lado la agregación de servicios permite incluir nuevos servicios desde diferentes proveedores y que se gestionen como un único servicio de cara al usuario, aunque también es posible que se añadan nuevos servicios para construir otros servicios y mejorar la oferta combinada del *Broker*. Por último la intermediación, consiste en capacidad de gestionar de forma autónoma cada una de las entidades involucradas: servicios y proveedores, con respecto al usuario consumidor que sólo vería una una entidad de *Broker* a través de la cual realizar la intermediación.

En términos generales los *Broker* de Cloud son utilizados como herramientas de toma de decisiones, para la predicción, selección y provisión de recursos de computación de forma inteligente. Gran parte de los trabajos se centran en como se debe realizar la planificación de los recursos de computación para satisfacer la demanda de los usuarios a través del *Broker* y en concreto sobre aspectos tales como la optimización de los costes o la mejora del rendimiento de la computación en entornos de Cloud Computing.

Nuestra propuesta va un paso más allá y se centra en el modelo de servicio que el usuario quiere utilizar desde Cloud Computing y no sólo en los aspectos de la provisión de las unidades de computación o la simple optimización de los recursos para un servicio genérico. La propuesta que presentamos se basa en el servicio en sí, es decir, en los servicios de minería de datos sobre Cloud Computing. Desde hace ya varios años los proveedores de Cloud han ido incluyendo en su catálogo una parte especifica de servicios cubriendo múltiples aplicaciones; este nuevo catálogo de servicios son los denominados *DAaaS* o *BDaa* (*Data Analisys as a Service* y *BigData as a Service*), que permiten la ejecución de algoritmos de minería de datos y Machine Learning como servicios en Cloud Computing. Este tipo de servicios permiten a

los usuarios la utilización de una infraestructura de computación y almacenamiento de los proveedores de Cloud Computing desde un punto de vista mucho más abstracto, centrándose en el servicio que se ofrece al usuario para resolver su problema, función o algoritmo concreto. Con esto el proveedor proporciona a los usuario un conjunto de herramientas y APIs gestionadas como servicio con todos los aspectos que condicionan un servicio de Cloud Computing, tales como la tarificación por el uso, el coste por la localización, la complejidad del servicio o la cuestiones relativas al nivel de servicio que debe ser proporcionado por el proveedor de los servicios, entre muchos otros aspectos.

Si nos enfocamos en este tipo de servicios, un *Broker* para servicios de minería de datos en Cloud Computing tiene mucho sentido puesto que permitirá al usuario:

- consultar y seleccionar dentro de un *MarketPlace* los proveedores y servicios existentes para resolver un problema relacionado con la minería de datos,
- intermediar entre el usuario y los proveedores para la negociación de los servicios y recursos ofrecidos por los proveedores como servicios de minería de datos,
- componer flujos de trabajo complejos de minería de datos sobre diferentes proveedores de Cloud, optimizando el rendimiento (por ejemplo), y
- centralizar la gestión de servicios de procesamiento de minería de datos desde un único servicio de *Brokering* que abstraiga las plataformas propias de los proveedores de Cloud.

Un *Broker* de minería de datos en Cloud Computing se encarga de conocer las interfaces y detalles de cada uno de los servicios de minería de datos proporcionados por los proveedores, sirviendo de plataforma para la agregación, selección e intermediación de este tipo de servicios. Nuestra propuesta de investigación se centra en el desarrollo de un *Broker* para la consulta, agregación e intermediación de servicios de minería de datos en Cloud Computing, que permita al usuario tener un único punto de interacción para todo el espectro de proveedores disponibles que ofrecen minería de datos como servicio, siendo este un elemento vital para la abstracción de los servicios sin importar a priori donde o como se implementen tras las infraestructura de

cada proveedor. La propuesta de *Broker* se ha llamado *BrokerMD* y busca tres objetivos básicos:

- Despliegue de un catálogo de servicios de minería de datos desde *BrokerMD* a modo de *MarketPlace* de algoritmos, que permita la búsqueda de servicios a partir de unos criterios de selección y unas restricciones dadas. Además, como sub-objetivo, consideramos importante la capacidad que debe tener todo *Broker* de agregar proveedores y servicios al *MarketPlace*, de modo que sea posible que un proveedor o usuario del *Broker* añada sus algoritmos o funciones, como servicios en Cloud Computing.
- Selección e intermediación de servicios. *BrokerMD* conoce las interfaces de interacción de los proveedores y los servicios de minería de datos que ofrecen, de modo que puede gestionar cada una de estas entidades de una forma transparente y flexible para el cliente consumidor de los servicios de Cloud Computing.
- *Brokering* para flujos de trabajo de minería de datos sobre Cloud Computing. Consiste en trabajar desde *BrokerMD* con un flujo complejo de procesamiento de datos relacionado con la minería de datos y Machine Learning, de modo sea capaz de buscar, seleccionar y ejecutar todo el flujo de operaciones y algoritmos desde diferentes proveedores a través de la plataforma de *Brokering* desplegada, tratando de optimizar el procesamiento con las restricciones que el usuario consumidor establezca.

Consideramos que *BrokerMD* es una propuesta sólida como modelo para la intermediación de servicios de minería de datos en Cloud Computing proporcionando una serie de herramientas de gestión útiles para la integración de este tipo de servicios en entornos de prácticamente cualquier tipo, donde se requiera, desde la ejecución de algoritmos simples a flujos de trabajo completos, con una escalabilidad garantizada y aprovechando los recursos heterogéneos ofrecidos por los diversos proveedores de Cloud Computing.

El desarrollo del capítulo está estructurado del siguiente modo: en la sección siguiente 5.2 se analiza gran parte de los trabajos relacionados junto con las técnicas de *Brokering* y algoritmos más utilizados para la optimización de recursos, además de selección de los servicios más adecuados para las necesidades y restricciones de los usuarios consumidores. Propuestas, plataformas y *frameworks* para el *Brokering* de servicios de minería de datos son

también estudiadas en esa sección correspondiente a la revisión del estado de arte y trabajos relacionados. La sección 5.3 desglosa todo el diseño del *BrokerMD* considerando cada uno de los objetivos clave planteados; en esta sección se trabaja sobre los aspectos de la normalización de los servicios utilizando tecnología semántica, también se incluyen cuestiones relativas a los la integración y agregación de servicios y elementos adicionales relacionados con la intermediación y selección de servicios. Finalmente en la sección 5.8 las conclusiones y el trabajo futuro derivado de nuestra propuesta son discutidos.

5.2. Trabajos previos relacionados

La adopción de *Brokers* para servicios de Cloud Computing sigue siendo aún escasa y limitada a entornos muy cerrados. Como se ha descrito en la introducción, el mercado de servicios de Cloud está sobre saturado de servicios de diferente índole, lo cual hace que la selección de un servicio y un proveedor de servicios sea una tarea compleja. Esto es debido a la enorme oferta de servicios donde cada una los proveedores posee interfaces heterogéneos, APIs exclusivas y en general particularidades que suponen un reto importante a la hora de decantarse por un proveedor de servicios o servicio en concreto. Esta heterogeneidad de interfaces y descripciones entre proveedores incrementa aún más el problema de la portabilidad en el diseño y desarrollo de aplicaciones en este tipo de entornos. Como respuesta a esta problemática, los *Broker* de Cloud Computing pueden servir de agentes para intermediar servicios entre usuarios y proveedores de Cloud, reduciendo al mínimo los riesgos de selección de servicios y proveedores. De este modo el uso y la implantación de *Brokers* aplicados a Cloud Computing, ayudan a eliminar las diferencias de definición entre los diferentes proveedores de Cloud y a encontrar servicios en la nube más afines a sus preferencias [Binz et al., 2012].

En esta sección realizaremos un estudio de las propuestas de *Brokering* desde diferentes puntos de vista. Además se enfocará parte del estudio en los aspectos como la migración y portabilidad, toma de decisiones, interoperatibilidad o algoritmos utilizados para la selección y toma de decisiones.

Gran parte de trabajos relativos a *Brokering* en Cloud consideran inhabilidad de un único proveedor para satisfacer de forma completa los requisitos de los usuarios a la hora de usar un *Broker*. De este modo, en artículos como [Jrad et al., 2013, Kim et al., 2015] se indica la necesidad de

usar un *Broker* para realizar flujos de trabajo entre diferentes proveedores [Ferrer et al., 2012]. Otros estudios como [Kertész et al., 2014] realizan un estudio de los acuerdos de nivel de servicio, a través de los cuales, una correcta selección de un proveedor estaría asociado a una selección adecuada de la SLA para los servicios que convengan mejor a los usuarios que utilizan los recursos computaciones con restricciones a lo largo de tiempo. La necesidad de utilizar un *Broker* para la provisión de recursos computacionales con el foco en la disponibilidad y la fiabilidad ha sido trabajada en [Javadi et al., 2012] [Apduhan et al., 2015]. Elementos como la sobre utilización de los recursos de un proveedor Cloud, ocasiona un desbordamiento en la selección de recursos de un único proveedor que puede ser eludida mediante la utilización de un *Broker* a través del cual es capaz de añadir recursos al sistema procedentes de otros Clouds o proveedores. Este escenario ha sido estudiado en [Copil et al., 2014] y [Nair et al., 2010].

En cuanto a la funcionalidad que los *Brokers* son capaces de desempeñar, su uso en la toma de decisiones, la monitorización de recursos o la migración, son los elementos clave a considerar. Diferentes enfoques se han tratado para implementar la funcionalidad de toma de decisiones en *Brokers*, gracias a la cual es posible ayudar a los usuarios para a seleccionar los recursos de un modo más óptimo [Jrad et al., 2013]. En algunos de estos enfoques se trata con requisitos funcionales y no funcionales para que los sistemas puedan recomendar a los proveedores y servicios que mejor se adaptan a las necesidades de los usuarios. Los enfoques que implementan esta funcionalidad aconsejan a los usuarios de Cloud en la toma de decisiones relativa a servicios [Javed et al., 2016, Chichin et al., 2014]. Los *Brokers* también están siendo muy tenidos en cuenta en la negociación de SLA [Nair et al., 2010], gracias a los cuales es posible capturar los requisitos del usuario y permiten ofrecer servicios en la nube que satisfagan esas condiciones. El resultado del proceso de negociación es un SLA que es capaz de gestionar los compromisos entre el proveedor y el cliente. Otras aproximaciones trabajan la intermediación a partir de elementos basados en tecnología semántica [Dastjerdi et al., 2010].

Para la solución de la planificación y asignación de recursos en servicios de Cloud Computing, diferentes marcos de trabajo basados en Cloud *Brokers* han sido desarrollados. En [Kertész et al., 2014] se combinan elementos de negociación, *Brokering* y despliegue. *STRATOS* provee de herramientas para la toma de decisiones automática en la adquisición de recursos entre diferentes proveedores, teniendo en cuenta el numero de recursos y la localización de los mismos. Otras herramientas para *Brokering* como *OPTIMIS* [Ferrer et al., 2012] permiten agregar una variedad de arquitecturas

de diferentes Clouds. Para el *Brokering* de servicios de ejecución de aplicaciones científicas, en [Anastasi et al., 2017] se propone una herramienta que asiste a los usuarios para seleccionar los recursos del modo más eficiente en cuando al coste. Estudios como [Wang et al., 2013, Simarro et al., 2011] permiten optimizar la reserva de infraestructuras de computación desde diferentes proveedores y abstraer el despliegue y gestión de los componentes de la infraestructura. La minimización de los costes económicos usando un *Broker* para la instanciación de unidades de cómputo entre diferentes proveedores usando programación dinámica ha sido tratado en [Wang et al., 2015]. Otras propuestas de *Broker* para la selección de recursos, utilizan algoritmos genéticos para la evaluación de las *SLA* desde diferentes proveedores [Amato et al., 2013]. De igual modo para la selección de servicios donde se prima el *QoS* en la elección de las instancias, han sido utilizados algoritmos genéticos [SureshKumar and Varalakshmi, 2017]. En [Chamorro et al., 2016] se estudia la reserva de máquinas virtuales orientadas a entornos dinámicos, considerando ofertas de recursos variables, precios y los requisitos dinámicos de los usuarios, para ello compara los beneficios de usar una aproximación basada en programación lineal y algoritmos genéticos.

Algunas soluciones de *Brokering* como "Middleware" en las que se consigue una intermediación entre diferentes proveedores de Clouds, son *CompatibleOne* [Yangui et al., 2014], que permite combinar diferentes servicios Cloud desde diferentes proveedores. *SoCloud* [Paraiso et al., 2016] por ejemplo extiende la propuesta basada en *OASIS SCA* [Binz et al., 2013], para construir aplicaciones de tipo *PaaS* en entornos multi-cloud, con políticas para asegurar el balanceo y la alta disponibilidad de los servicios. Otra herramienta para controlador una aplicación desplegada a través de múltiples nubes es desarrollado en [Copil et al., 2014].

Para lidiar con la heterogeneidad de la definición de los recursos de Cloud, diferentes propuestas semánticas han sido estudiadas. De este modo surge la idea de unificar la configuración de los recursos y el acceso a las APIs a partir del uso de tecnología semántica. Una comparación entre *OWL-S* y *WSMO* para evaluar su aplicabilidad en un entorno real de servicios ha sido estudiada en [Lara et al., 2004]. La propuesta *WSMO-lite* [Vitvar et al., 2008] permite mejorar las descripciones de servicio existentes, añadiendo las capacidades de *SoA* con una integración inteligente y automatizada. Otra propuesta como *MSM* [Taheriyan et al., 2012] (*Minimal Sevice Model*) tiene la capacidad de expresar intercambio, automatización y composición de servicios a nivel semántico con el enfoque en el servicio final. Además propuestas más completas como *Linked-USDL* [García et al., 2016a, Pedrinaci et al., 2014],

permiten modelar cualquier recurso en Cloud e incluyen partes como modelado de precios, SLA, interacción, o roles entre otros. En la sección 3.2 gran parte de las propuestas de tipo semántico han sido estudiadas.

Con este estudio de las diferentes propuestas para *Brokering* y desde múltiples puntos de vista, es posible tener una idea general de la enorme aplicabilidad de los *Brokers* de servicios en Cloud, considerando sus capacidades, herramientas, marcos de trabajo y algoritmos utilizados para la toma de decisiones.

5.3. Propuesta de diseño de un *Broker* para servicios de minería de datos en Cloud Computing

El desarrollo de un *Broker* para servicios de minería de datos, es una cuestión compleja. De igual modo que si tratamos con un servicio de *Broker* genérico para negociación de recursos/reserva de cómputo, *BrokerMD* supone además de una abstracción de la infraestructura de cómputo subyacente, añadiendo una capa extra sobre la cual se montan los servicios de minería de datos, que son los que finalmente utilizan esos recursos desde varios proveedores. En nuestra propuesta de *Broker*, *BrokerMD* está enfocado en los servicios en sí y no tanto la arquitectura de despliegue subyacente.

Durante la fase de análisis de las características de los servicios de minería de datos se han estudiado algunos de los proveedores de Cloud Computing más conocidos (*Amazon*, *Azure*, *Google*, o *IBM*). Dentro de este estudio se ha extraído la información relativa a cómo estos proveedores ofertan servicios de minería de datos. El conjunto de modelos de servicio que se ofrece de modo general consiste en una serie de fases que van desde la creación y manipulación de los datos, aplicación de algoritmos, fases de entrenamiento, modelado, hasta el despliegue de los modelos para la predicción y explotación. El objetivo de este estudio es conocer por un lado cuales son los aspectos que debemos considerar de cara al diseño del *Broker*, pensando en como los proveedores de Cloud Computing resuelven el problema del flujo de trabajo y operaciones en minería de datos con todas sus fases. De forma paralela, se ha verificado cuales son las partes del proceso de trabajo con estos servicios que son afectadas por costes de diferente tipo, como almacenamiento, almacenamiento de resultados intermedios, transferencia de datos, almacenamiento de

modelo, llamadas a predicción con el modelo creado o los costes relacionados con el uso de los recursos de computación y procesamiento para cada una de las operaciones de minería de datos.

En los siguientes subsecciones analizamos los elementos más destacados que se han podido extraer del análisis del modelo de servicios de minería de datos en Cloud Computing desde los proveedores más conocidos de esos servicios.

Algoritmo o función de minería de datos como servicio. Este modelo está bastante extendido entre los proveedores y consiste en publicar o exponer un servicio de minería de datos para que lo usuarios puedan utilizarlo e integrarlo desde otras aplicaciones, desde un interfaz web, o bien desde la propia aplicación de gestión del proveedor. El modelo de trabajo permite ejecutar y usar una serie de algoritmos predefinidos desde un catálogo de algoritmos listos para ser consumidos por los usuarios. No sólo se controla la función a ejecutar, también se gestionar el acceso a los datos del problema y los resultados en forma de nuevos datos o bien de modelos que son directamente explotables.

Al igual que los servicios generales ofertados por los proveedores, este tipo de servicios específicos de minería de datos, están afectados por diversos costes que van en función de varios aspectos clave en la gestión de un servicio de Cloud Computing, como es el tiempo de ejecución del servicio y los recursos involucrados, el tamaño de los datos de entrada y salida, el uso del modelo o las restricciones del acuerdo del nivel de servicio entre otras.

Por ejemplo, el servicio de *Amazon SageMaker*, está compuesto por tres módulos: construcción, capacitación e implementación. El módulo de construcción ofrece una herramienta para trabajar con los datos, ejecutar algoritmos y visualizar las salidas del procesamiento. El módulo de capacitación permite la creación de un modelo de minería de datos asociado. El módulo de implementación habilita un entorno para desplegar los modelos y que estos sean consumidos (predicción, inferencia, etc.).

Si nos centramos en la fase de creación en el servicio de *Amazon SageMaker* o *Microsoft Azure Machine Learning Studio*, este pone a disposición de los usuario una serie de algoritmos listos para usar (ver tabla 5.1 para *Amazon SageMaker* y tabla 5.2 Microsoft Azure MLS). Estos algoritmos están diseñados para que el consumidor de los servicios no tenga que preocuparse por el tamaño de los datos o por la escalabilidad

BlazingText Algorithm
DeepAR Forecasting Algorithm
Factorization Machines Algorithm
Image Classification Algorithm
IP Insights Algorithm
K-Means Algorithm
K-Nearest Neighbors (k-NN) Algorithm
Latent Dirichlet Allocation (LDA) Algorithm
Linear Learner Algorithm
Neural Topic Model (NTM) Algorithm
Object2Vec Algorithm
Object Detection Algorithm
Principal Component Analysis (PCA) Algorithm
Random Cut Forest (RCF) Algorithm
Semantic Segmentation Algorithm
Sequence-to-Sequence Algorithm
XGBoost Algorithm

Tabla 5.1: Algoritmos y funciones disponibles en *Amazon SageMaker*.

que pudiesen requerir. Se centran en el algoritmo, que posee una entrada de datos, parametrización y una salida de datos. Además otra de las ventajas de este tipo de servicios es la capacidad para incluir nuevos algoritmos a la plataforma, de modo que le otorga una gran flexibilidad a la hora de añadir nuevos desarrollos que pueden posteriormente convertirse en servicios listos para consumir.

Modelo de minería de datos como servicio. Además de incluir un conjunto de algoritmos listos para ser utilizados, los proveedores incluyen opciones adicionales que integran la generación del modelo como parte de las fases de planificación de un trabajo con minería de datos. Tras entrenar con los datos en una fase previa (es decir, tras detectar los patrones en los datos), se crea un modelo que servirá para hacer las predicciones. Podemos asociar el significado de un modelo a un filtro en el que entran datos nuevos y cuya salida es la clasificación de ese dato según los patrones que se han detectado en el entrenamiento. Por ejemplo, si se entrena un modelo con datos históricos de clientes para detectar el riesgo de baja de una tarjeta de crédito, el modelo clasificará a los nuevos clientes en función de su comportamiento para predecir el riesgo de baja. Este modelo se puede ver en la Figura 5.4. El flujo de

PCA
Multiclass/Two-Class Decision Forest
Multiclass/Two-Class Decision Jungle
Multiclass/Two-Class Logistic Regression
Multiclass/Two-Class Neural Network
Two-Class Support Vector Machine
K-means
K-means
Bayesian Linear Regression
Boosted Decision Tree Regression
Decision Forest Regression
Fast Forest Quantile Regression
Linear Regression
Neural Network Regression

Tabla 5.2: Algoritmos disponibles en *Microsoft Azure ML Studio*.

trabajo según la figura para el aprendizaje máquina incluye estas fases:

1. Elección de un algoritmo adecuado y configuración de las opciones iniciales.
2. Entrenamiento del modelo en datos compatibles.
3. Creación de predicciones mediante el uso de nuevos datos, basados en los patrones del modelo.
4. Evaluar el modelo para determinar si las predicciones son exactas, cuánto error hay y si hay algún ajuste excesivo.

Las etapas 2 y 3 son las que permiten trabajar con el modelo, entrenarlo y posteriormente crear las predicciones u obtener resultados extra en base a esas salidas.

Cada una de estas etapas tiene un coste asociado (económico, rendimiento, tiempo, etc.). El de mayor peso recae en el entrenamiento y la generación del modelo, pues es una parte importante de las operaciones y supone una mayor complejidad al involucrar diferentes recursos de computación.

Explotación y evaluación como servicio. Cuando ya se ha creado un modelo para unos datos de entrenamiento, lo más probable es que se

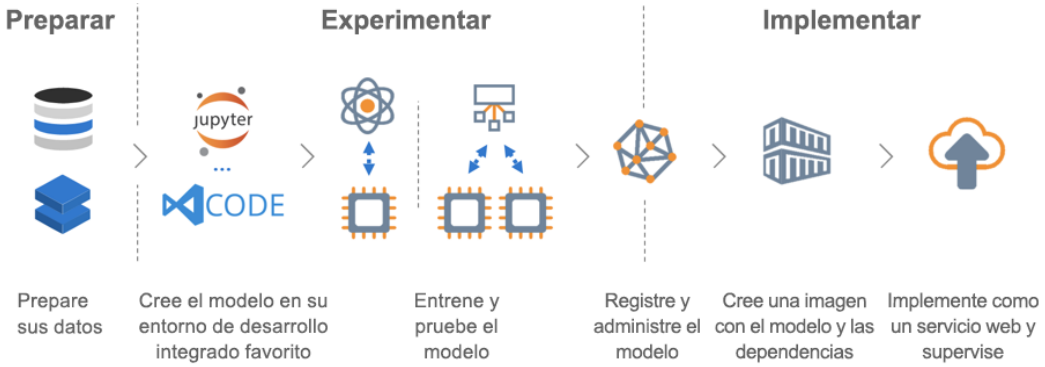

Figura 5.4: Modelo de trabajo con *Microsoft Azure Machine Learning Studio*

quiera utilizar ese modelo para realizar predicciones, de modo que desde una aplicación o desde otro entorno, la predicción contra el modelo creado permite obtener resultados sobre los datos para lo usuarios. Esta fase del trabajo con minería de datos al igual que las demás partes se encuentra soportada por unos costes en función de diferentes parámetros de acceso, tales como el tamaño del modelo usado (almacenamiento), o el número de llamadas que se realizan para predicciones. Cuando se trabaja con este tipo de modelado de flujos de trabajo en minería de datos, esta etapa final suele ser la que se integra en otros sistemas y herramientas de cada cliente en forma de una API Web (de tipo *RESTful* o *WSDL*, por ejemplo), de modo que puede ir realizando las llamadas con nuevos datos sobre el modelo entrenado y obteniendo los resultados. Este tipo de modelado impone una serie de restricciones que el cliente debe valorar, como son los costes relacionados con el volumen de la explotación, tamaño del modelado o los ciclos de actualización del modelo, por ejemplo, que repercutirán en los costes finales de utilización del servicio.

5.3.1. Componentes del *Broker* diseñado

La propuesta *BrokerMD* que se ha diseñado consta de las siguientes partes (ver Figura 5.5):

- Catálogo de servicios. El catálogo de servicios es el componente que se encarga de gestionar el conjunto de todos los servicios que forman parte de la cartera del *Broker*. En esta parte, se almacenan los servicios desde las diferentes descripciones de los proveedores de minería de datos, de

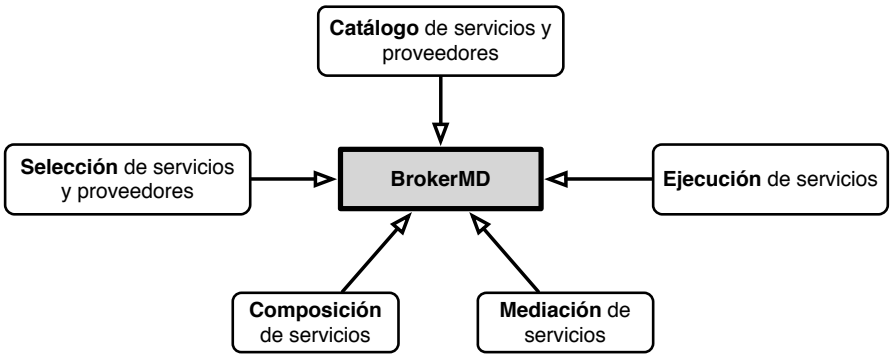

Figura 5.5: Componentes que se han diseñado para *BrokerMD*.

modo que se ofrece una interfaz común para la consulta y descubrimiento de servicios de este tipo. El catálogo trabaja directamente con las descripciones de servicios que `dmcc-schema` puede capturar desde los proveedores de minería de datos. Desde el catalogo por ejemplo es posible realizar búsquedas y consultar información con cualquier nivel de detalle sobre los servicios que *BrokerMD* almacena. En la sección 5.4 se detalla el diseño de este componente.

- Selección de los servicios. El `BrokerMD` almacena el catálogo de los servicios que ha capturado o se han agregado desde otros proveedores utilizando la definición `dmcc-schema`. Con esta base de datos proporcionada por el catalogo, el *Broker*, puede realizar una selección de los servicios que darán soporte a un flujo de trabajo de minería de datos. El usuario solicita ejecutar un flujo de trabajo al *BrokerMD* y el componente de selección de servicios es el encargado de realizar esta operación de planificación de tareas. Para solucionar la planificación del flujo de trabajo el componente de selección del servicio utiliza algoritmos de optimización para encontrar el modo de ejecutar el trabajo de minería de datos utilizando toda la información proporcionada por el catálogo. Todo el proceso de desarrollo del componente de selección de los servicios se describe en la sección 5.5.

- Mediación de servicios. Es el componente del *Broker* que se encarga de conocer la lógica de la plataforma ofrecida por cada uno de los proveedores de minería de datos de modo que sea capaz, por ejemplo, de ejecutar un algoritmo, una función o un servicio concreto en un proveedor dado. De este modo, cuando un usuario solicita ejecutar un algoritmo *Random Forest* al *Broker*, es este quién conoce la forma de conectar, autenticar, invocar y gestionar el servicio en cada uno de los

proveedores que lo tengan, con lo que el usuario al final solo gestiona un único punto de acceso al servicio *Random Forest*, mientras que el *Broker* hace el esfuerzo por el usuario de abstraer ese servicio con todos los proveedores. En la sección 5.6 se realiza la propuesta para la mediación de servicios.

- Composición y ejecución. Otros de los componentes vitales del *Broker* es la composición y la ejecución final de los servicios. Por un lado la composición de servicios, permite al *Broker* encadenar servicios desde otros proveedores a través de los cuales, se conforma un flujo de trabajo remitido por un usuario al *Broker*. Por otro lado la ejecución de los servicios se encarga del gestión de la ejecución de cada una de las tareas que componen un flujo de trabajo de minería de datos. Este componente utiliza parte de la funcionalidad contenida en la mediación de servicios, pues es responsable de realizar la ejecución final de cada una de las operaciones (funciones, algoritmos, etc.) en cada proveedor, además de recuperar toda la información de esas operaciones, los datos, las transferencias, etc. En la sección 5.7 se detallan ambos componentes.

5.4. Catálogo de servicios

Otra de las principales características de los *Broker* es que son capaces albergar un catálogo de servicios. Nuestra propuesta de *BrokerMD* igualmente posee esta cualidad, mediante la cual es capaz de almacenar toda la información relativa a los servicios de minería de datos y disponibles desde diferentes plataformas o bien que han sido integrados en el mismo (es lo que se refiere a la agregación de servicios) desde diferentes fuentes de agregación, tales como *Market Place*, descubrimiento de servicios automático o sistemas de extracción de información para servicios.

Un catálogo de servicios de minería de datos es una lista completa de servicios relacionados con el procesamiento de datos, que el *BrokerMD* ofrece a sus clientes. Este catálogo de minería de datos es una parte de la cartera de servicios que se publicitan, de modo que proporciona a los clientes el apoyo necesario para la prestación y búsqueda de los servicios que se requeridos por parte del consumidor.

Un catálogo de servicios en Cloud Computing incluye:

- El nombre del servicio u operación y su descripción.

- Todos los servicios listados por categorías, tales como procesamiento, algoritmos, etc.
- Todos los servicios de apoyo a los servicios principales.
- Acuerdos de nivel de servicio y plazos de cumplimiento para los servicios (*SLA*).
- Contactos y puntos de acceso.
- Costes de los servicios.
- Estimaciones del tiempo de ejecución.
- Datos de interacción con el servicio: almacenamiento, requisitos de salida, etc.

El diseño del catálogo de servicios ofrecido por el *BrokerMD*, considera un conjunto de prácticas y directrices definidas para la gestión de servicios de las Tecnologías de la Información con el fin de alinear los servicios de TI con las necesidades del negocio. El catálogo de servicios ofrecido por el *BrokerMD* permite a un sencillo consumidor o a un científico de datos que busca utilizar un servicio, encontrar de forma rápida detalles sobre los servicios que están buscando.

Desde el punto de vista del usuario, el catálogo ofrecido por el *BrokerMD* hará que sea más rápido averiguar qué servicios de minería de datos se ofrecen, sus descripciones y cómo contratarlos o trabajar sobre ellos; de este modo el único punto de encuentro entre el usuario consumidor y el servicio concreto será el *BrokerMD* que intermediará con el proveedor en concreto del servicio en cada momento sin que el usuario tenga que tratar directamente con ese proveedor.

De igual modo, tal y como se verá en en las siguiente secciones, no sólo será posible obtener en el catálogo servicios individuales, también habilitará para la composición de servicios. Con esto se consigue que el usuario no sólo se limite a la contratación e intermediación de los servicios con el *Broker*, también debe permitir componer el flujo de trabajo de minería de datos y que el *BrokerMD* actúe como herramienta de toma de decisiones para la selección de este tipo de servicios.

5.4.1. Consulta de servicios en el *BrokerMD*

Tanto el catálogo como la consulta de servicios están muy relacionados y parte de la funcionalidad de ambos puede ser difusa. En cuanto al catálogo de servicios, es posible conocer por parte de las entidades consumidoras del *Broker*, la lista de operaciones o tareas que soporta el *Broker* y que son directamente gestionables por él; de este modo el catálogo podría ser visto como una entidad que permite al usuario conocer que métodos u algoritmos están disponibles para el proceso de experimentación, además de otros datos citados en la subsección anterior. La consulta de servicios, además de ofrecer al usuarios la lista de servicios que puede desplegar el *BrokerMD* a modo de catálogo individual de operaciones, también permite realizar consultas al motor del *BrokerMD*. Este tipo de consultas no sólo son para conocer o listar operaciones simples, también consideran otras mucho más complejas como por ejemplo las de tipo "*Random Forest más económico*" o "*Coste total de un flujo de trabajo de minería de datos*". De este modo la parte correspondiente al sistema de consulta de servicios del *BrokerMD*, implica una complejidad adicional con respecto a lo que supone un simple catálogo de servicios de minería de datos.

La consulta de los servicios se puede realizar de dos modos, por un lado desde SPARQL y por otro lado desde la definición de experimentación de flujos de trabajo con `dmcc-schema`. En los siguientes apartados se detallan cada uno de ellos.

5.4.1.1. Consulta desde SPARQL

Cuando se trabaja con datos, la cuestión vital del flujo de datos -es decir, el movimiento de datos, desde la creación hasta el almacenamiento, la revisión y la reorientación- se ha visto tradicionalmente obstaculizada por la disparidad en varios aspectos, entre ellos:

- Identificadores de elementos de datos (es decir, sujetos o entidades).
- Protocolos de acceso a datos (estructuras de servicios web).
- Estructuras de datos (formatos).
- Ubicación de datos (acceso y disponibilidad).
- Idiomas de consulta de datos (localización si existe).

El lenguaje de consulta SPARQL es un lenguaje de consulta declarativo (como SQL) para realizar operaciones de manipulación y definición de datos en datos representados como una colección de sentencias/expresiones del lenguaje *Turtle/RDF*.

Cada consulta SPARQL tiene un modificador de solución (o cabecera) y un cuerpo de consulta. El modificador de soluciones proporciona la base para categorizar diferentes tipos de soluciones de consulta en SPARQL. El cuerpo de consulta comprende una colección de patrones de declaración RDF (representados usando el lenguaje *Turtle*, por ejemplo) que representan las relaciones de las entidades a las que se extiende una consulta. Para llevar a cabo una consulta simple al motor de SPARQL que despliega el *Broker* se usa la sentencia `SELECT`, que es muy parecida a un *SQL SELECT*, que proyecta una solución de consulta en forma de tabla. Este tipo de consulta es muy versátil y permite realizar prácticamente cualquier tipo de búsqueda de entre los datos proporcionados por el *Broker* de minería de datos.

Para poder poner en marcha este servicio de consulta dentro del *BrokerMD* se ha desplegado dentro del *Broker* un servicio de consulta SPARQL. Este tipo de servicio está basado en el protocolo HTTP (también conocido como servicio web), ofrece una API para realizar operaciones declarativas de definición y manipulación de datos sobre datos representados en la Base de Datos de servicios que alberga el *Broker*, como colecciones de tripletas semánticas en *RDF*. Naturalmente, este tipo de servicio es proporcionado por una aplicación de Sistema de Gestión de Bases de Datos (*DBMS*) o *Triple/Quad Store* asociada a una URL que identifica su punto de inicio HTTP, es decir, la dirección a la que se envían los mensajes (consultas). Para nuestro despliegue de *BrokerMD*, hemos optado por la integración de un servicio semántico llamado *Apache Jena-Fuseki*[5], que es a su vez es un *framework* de código abierto para el trabajo con tecnología semántica. Este *framework* y servicio proporciona una API para extraer y consutar datos, además de gestionar grafos de conocimiento en *RDF*. Los grafos del servicio de *Apache Jena-Fuseki* se representan como un modelo abstracto disponible para poder explotarlo desde cualquier aplicación o servicio. Los modelos puede obtenerse desde datos de archivos, bases de datos, URLs o una combinación de estos.

Un "EndPoint" en SPARQL ofrece varios beneficios únicos a través de una API para el *BrokerMD*:

- La interacción de datos declarativos (manipulación y definición) puede

[5]Apache Jena-Fuseki: https://jena.apache.org/

ser integrada a través de HTTP - dirigiendo los datos representados como colecciones de sentencias/declaraciones RDF de grano fino.

- Las URLs de documentos HTTP aportan una gran flexibilidad a las consultas de datos, es decir, cada componente de una URL es una ranura para la alteración parametrizada de los detalles de una consulta que se extiende desde su punto final de destino hasta la naturaleza de la solución de consulta.

- Se admite una amplia variedad de tipos de contenido para documentos de soluciones de consulta: HTML, JSON, CSV, RDF-Turtle, RDF-N-Triples, RDF-XML y otros.

- Los tipos de contenido de todos los documentos de solución de consulta son negociables - cortesía de la negociación de contenido HTTP ("con-negativo").

- Se puede acceder a los puntos finales con cualquier agente o servicio de usuario compatible con HTTP.

Con ello, utilizando la capacidad ofrecida por los lenguajes semánticos, podemos mirar efectivamente a los endpoints de SPARQL como herramientas muy flexibles para la consulta y descubriento de servicios descritos semánticamente. Este hecho queda claramente demostrado por el número cada vez mayor de Endpoints en SPARQL asociados a los nodos que componen la ya masiva - y aún en crecimiento - Nube de *Linked Open Data*, con lo que cualquier consulta realizada a un motor de SPARQL, puede ser extendida de forma que admita otros conjuntos de datos y no se quede limitada al dominio del servicio de minería de datos proporcionada por el *Broker*.

El acceso al motor semántico donde están albergados los servicios se encuentra en el sitio web del proyecto[6].

Desde este EndPoint de SPARQL se tiene acceso a la base de datos de servicios que está integrada dentro del motor del *Broker* de minería de datos. Actualmente posee más de 24000 tripletas en *Turtle/RDF*, a través de las cuales se pueden obtener datos de proveedores (los integrados por defecto en nuestro *Broker* ML y están disponibles son: *Amazon SageMaker* y *Azure ML Studio*), algoritmos proporcionados, costes, instancias, localizaciónes, APIs o acuerdos de SLA, entre muchos otros.

[6]Motor SPARQL en *Apache Jena-Fuseki*: https://dicits.ugr.es/linkeddata/*BrokerMD*

Para demostrar el funcionamiento de este tipo de consultas al motor del *BrokerMD*, se ha seleccionado un grupo de consultas que permiten demostrar la flexibilidad y las capacidades básicas de este tipo de sentencias de búsqueda con SPARQL. Gracias a SPARQL se pueden desarrollar infinidad de consultas, desde simples listados a operaciones de búsqueda con agregaciones, operaciones calculadas, divisiones, etc. La complejidad de las mismas se deja al usuario a la hora se realizar una consulta con las restricciones que considere. Algunos ejemplos de consultas se exponen en los siguientes párrafos:

Consulta de los proveedores almacenados en el *BrokerMD*. La consulta siguiente realizada en SPARQL, permite consultar el listado de proveedores que están contenidos en el *Broker* para minería de datos. Este es el ejemplo más simple que se puede proveer para una consulta sobre los proveedores que ofrecen sus servicios sobre el *BrokerMD*. En el listado5.1 se obtienen los nombres de los proveedores de Cloud que están dentro de la Base de Datos que maneja el *Broker*.

```
1   PREFIX
2    s:     <http://www.w3.org/2000/01/rdf-schema#>
3   PREFIX
4    dmc: <http://cookingbigdata.com/linkeddata/dmcc-schema>
5   PREFIX
6    dc:    <http://purl.org/dc/elements/1.1/>
7   SELECT ?s ?name
8    WHERE {
9     ?s ?p dmc:MLProvider .
10    ?s dc:title ?name .
11   }
```

Listado 5.1: Consulta en SPARQL para obtener los nombres de los proveedores que están en el sistema de almacenamiento del *Broker*.

Consulta los servicios / algoritmos disponibles de cada proveedor. Cada proveedor almacenado en el motor de la Base de Datos, alberga el cojunto de servicios o algoritmos que provee dentro de su catálogo y que son directamente gestionables por el *Broker*. El listado 5.2 corresponde a una consulta sobre los datos del algoritmo, y el proveedor que lo tiene en su catálogo.

```
1   PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2   PREFIX s:     <http://www.w3.org/2000/01/rdf-schema#>
3   PREFIX dmc: <http://cookingbigdata.com/linkeddata/dmcc-schema>
4   PREFIX dc:    <http://purl.org/dc/elements/1.1/>
5
```

```
6    SELECT ?name ?label ?algor
7     WHERE {
8      ?s ?p dmc:MLProvider ;
9       dc:title ?name ;
10       s:label ?label ;
11       dmc:hasMLService [
12        a dmc:MLService;
13        s:label ?algor
14       ];
15  .
16 }
```

Listado 5.2: Consulta en SPARQL donde se recuperan los datos básicos de los algoritmos como servicio almacenados.

Consulta de los proveedores que ofrecen un algoritmo *Random Forest*. Utilizando la tecnología semántica es posible explorar todos los servicios ofrecidos por los proveedores almacenados. De este modo por ejemplo utilizando la consulta mostrada en el siguiente listado, es posible conocer cuales son los proveedores que ofrecen un servicio con un algoritmo *Random Forest*, además de poder añadir datos concretos de costes o localización asociada, para poder calcular de forma más precisa los costes de experiminentación desde la misma consulta.

```
1  PREFIX
2    rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX
4    s:     <http://www.w3.org/2000/01/rdf-schema#>
5  PREFIX
6    dmc: <http://dicits.ugr.es/linkeddata/dmcc-schema>
7  PREFIX
8    dc:    <http://purl.org/dc/elements/1.1/>
9
10 SELECT ?name ?label ?algor
11  WHERE {
12   ?s ?p dmc:MLProvider ;
13    dc:title ?name ;
14    s:label ?label ;
15    dmc:hasMLService [
16     a dmc:MLService;
17      s:label ?algor
18     ];
19 .
20 FILTER (?algor = "RandomForest")
```

```
21          }
```

Listado 5.3: Consulta en SPARQL donde se recuperan los datos detallados de los proveedores que proporcionan un servicio de Random Forest.

Consulta los 20 mejores proveedores-servicios de *Random Forest* a nivel de costes económicos más bajos. Esta consulta, supone una primera aproximación a un tipo de consultas complejas que permiten conocer los datos de los costes económicos de la ejecución de un algoritmo de *Random Forest*, por ejemplo. Para el listado 5.4 no se ha incluido ningún aspecto aspecto adicional como localización de la experimentación o necesidad de uso de una instancia de tipo concreto. Todos esos factores adicionales pueden incluirse en la consulta, de modo que es posible obtener un listado muy completo sobre los servicios que ofrece el *Broker*.

```
1
2  PREFIX adms: <http://www.w3.org/ns/adms#>
3  PREFIX pr: <http://purl.org/ontology/prv/core#>
4  PREFIX gr: <http://purl.org/goodrelations/v1#>
5  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
6  PREFIX s:     <http://www.w3.org/2000/01/rdf-schema#>
7  PREFIX dmc: <http://cookingbigdata.com/linkeddata/dmcc-schema>
8  PREFIX dc:    <http://purl.org/dc/elements/1.1/>
9  PREFIX price:    <http://cookingbigdata.com/linkeddata/ccpricing/>
10
11 SELECT ?name ?label
12        ?algor ?priceplan
13        ?priceplancompound
14        ?aaaa ?cost
15 WHERE {
16   ?s ?p dmc:MLProvider ;
17         dc:title ?name ;
18         s:label ?label ;
19         dmc:hasMLService [
20          a dmc:MLService;
21                 s:label ?algor;
22                 ];
23                 dmc:hasPricingPlan [
24          a price:PricesPlan;
25            s:label ?priceplan;
26            price:hasComponentPrice ?priceplancompound ;
27             ];
28         .
29         ?priceplancompound s:label ?aaaa.
```

```
30      ?priceplancompound price:withMaxCompound [
31       a gr:Offering;
32                gr:hasPriceSpecification [
33                a gr:PriceSpecification;
34                gr:hasCurrencyValue ?cost;
35          ];
36        ].
37
38    FILTER (?algor = "RandomForest")
39  }
40
41  ORDER BY ASC(?cost) LIMIT 20
```

Listado 5.4: Consulta con los costes de ejecución de cada algoritmo como servicio disponible con Random Forest. Extrae todos los datos de la tarificación.

5.4.1.2. Consulta con `dmcc-schema`

Otra de las modalidades de consulta al *BrokerMD* es más compleja y consiste en indicar el flujo de trabajo de minería de datos con todas las operaciones y la ejecución de algoritmos. Un flujo de datos o procesamiento de ejemplo se puede observar en la figura 5.6.

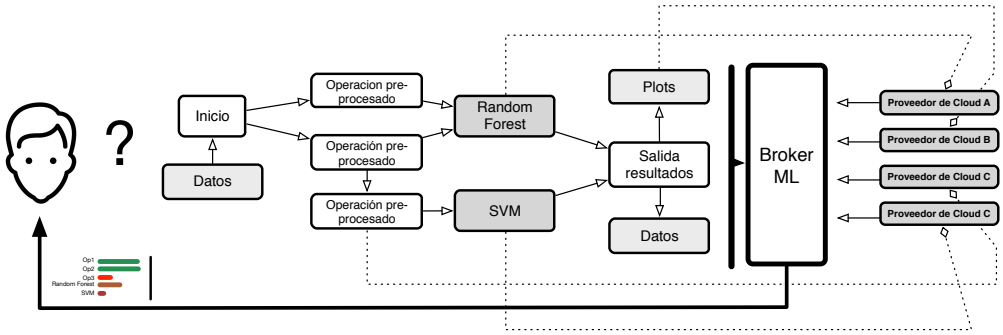

Figura 5.6: Flujo de trabajo de minería de datos que un usuario puede consultar al *BrokerMD*.

En este tipo de consultas el usuario lo que busca es ejecutar el flujo de trabajo de minería de datos y conocer una serie de valores sobre la ejecución de ese flujo de trabajo antes de realizar la ejecución sobre el *BrokerMD*. Como se ha indicado en la sección 5.5.5 sobre el diseño y modelado de la herramienta de toma de decisión basada en algoritmos evolutivos, se han delimitado unos objetivos que el usuario persigue a la hora de trabajar con

minería de datos. Estos objetivos son: a) el coste económico, b) la estimación del tiempo de ejecución, y c) la calidad del servicio. De este modo la solución que el *Broker* devueve al usuarios, por un lado será:

- **Cómo se va a ejecutar la experimentación**. Es decir el modo en el que se ejecutarán las operaciones, teniendo en cuenta e indicando cual será el proveedor, las operaciones a realizar, los recursos utilizados o las transferencias de datos que se llevarán a cabo. Hay que tener en cuenta que el diseño del *BrokerMD* alberga las definiciones de los servicios provenientes de distintos proveedores que ofrecen servicios de minería de datos, por lo que el plan de ejecución de la experimentación tendrá en cuenta todos los aspectos relacionados con una ejecución de servicios en *inter-cloud*.
- **Cúal es el valor de los tres objetivos**. El usuario es el responsable de decidir que plan de ejecución se ejecutará finalmente, basándose en la herramienta de toma de decisiones que le proporciona el servicio de *Broker*. Esta herramienta le permitirá al usuario seleccionar los criterios a través de los cuáles guiarse y decidir cual de los planes de ejecución se realizará. Para cada una de las soluciones mostradas al usuario una serie de objetivos se presentan desglosados en diferentes valores de detalle; estos valores permiten conocer los datos específicos sobre sub-objetivos que pueden determinar el plan de la solución. Sin embargo el usuario tendrá que elegir el plan de ejecución determinando cual es la solución que mejor se adapta a sus restricciones, de tal modo que si elige una solución con un valor determinado valor de los objetivos, es probable que exista otra solución que empeore algún otro objetivo. El *BrokerMD* propone al usuario un frente de Pareto mediante el cual tomar una decisión mucho mas eficaz en la selección de una solución al flujo de trabajo.

El proceso de consulta sobre la experimentación en el *BrokerMD* aglutina varias partes que trabajan de forma conjunta para ofrecer al usuario la selección de los planes de ejecución de un experimento. Estos módulos y el flujo de trabajo completo del *BrokerMD* se puede observar en la figura 5.7. En esta figura se puede ver como todo el conjunto de tareas que se realizan desde que el usuario realiza una consulta al *BrokerMD*, desde la utilización del catálogo, la extracción de los datos a través de SPARQL, la creación de un flujo de trabajo de experimentación, la ejecución del algoritmo multi-objetivo para la selección de los planes de ejecución y finalmente la obtención de las soluciones de ejecución. Como última etapa también se ha tenido en cuenta que

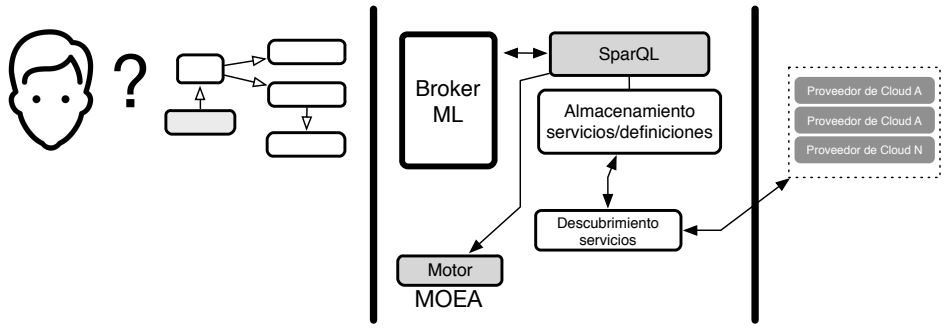

Figura 5.7: Operaciones realizadas por el *BrokerMD*, desde que el usuario envía un flujo hasta que consigue obtener los resultados la selección de servicios.

una vez generadas las soluciones, el usuario consumidor indica al *BrokerMD* que solución será implementada directamente sobre Cloud Computing.

Para el modelado de la consulta de experimentación se utiliza la definición de los servicios de experimentación en minería de datos con `dmcc-schema`. Con este esquema de definición se cubre todo el flujo de trabajo para minería de datos, permitiendo definir operaciones y sus dependencias, entrada y salida de datos, modelos o parametrización entre muchas otras características más. El uso de tecnología semántica ofrece al usuario una enorme flexibilidad a la hora de la definición de flujos de trabajo de este tipo, de modo que los elementos que se integran en la definición pueden estar detallados al nivel que se considere adecuado. En esta misma línea la definición de estos elementos, puede verse como una metodología "Building Blocks" como se puede observar en la figura 5.9. Este tipo de modelado ofrece al usuario la capacidad de combinar distintos bloques o cajas que tienen unas propiedades de entrada y salida, además pueden estar interconectados entre si para obtener al final un modelado que resuelve un problema de minería de datos basado en operaciones más pequeñas.

5.4.2. Agregación de servicios al *BrokerMD*

Un "agregador" de servicios de Cloud Computing es un tipo de agente de Cloud Computing que empaqueta e integra múltiples servicios de computación en uno o más servicios compuestos. En nuestro caso, en el *BrokerMD*, el "agregador" es considerado una entidad que permite la inclusión de servicios desde diferentes proveedores. Esta integración, que incluye servicios de múltiples proveedores, suele ser más eficiente y rentable para el cliente consu-

midor que comprar cada servicio por separado. Los términos de "agregador" y *Broker*, a veces se utilizan indistintamente para hacer referncia al mismo término, pero hay una sutil diferencia. El papel de un *Broker* en Cloud es más similar al de un consultor e intermediario entre compradores y vendedores de servicios, mientras que un "agregador" de la nube da ese paso adelante y es más análogo al de un integrador de sistemas: agrupar y ensamblar servicios de nube de terceros y/o internos en ofertas propias desde un único punto de acceso.

El modelo de la agregación de servicios está en continua evolución. Para la agregación se pueden ofrecer sólo un número finito de paquetes fijos, mientras que otros pueden crear gradualmente nuevos paquetes o grupos de servicios mezclando y haciendo coincidir las características de los servicios existentes. El éxito de un "agregador" de Cloud Computing depende en parte de las capacidades de federación e integración de nubes por parte de los proveedores cuyos servicios son adoptados por el "agregador". Idealmente, la federación de estos servicios y proveedores es totalmente transparente para el cliente que consume los servicios agregados desde una interfaz de usuario lógica que abstrae todas las diferencias existentes. Si los proveedores de Cloud están mal federados o no se ha conseguido una integración adecuada, el "agregador" puede tener que trabajar más a fondo para integrar los servicios para los clientes.

La agregación de los servicios en el *BrokerMD* sigue la misma línea de trabajo que en gran parte del trabajo de investigación desarrollado, esto es, el uso de un esquema de definición de servicios con `dmcc-schema`. Gracias a este esquema de servicios de minería de datos, es posible homogeneizar la definición y la composición de servicios, permitiendo al *BrokerMD* una agregación de los mismos mucho más efectiva, intentando reducir de manera considerable la fricción que supone la integración de servicios y proveedores desde diferentes fuentes. La agregación de servicios se basa en la capacidad que tiene el *BrokerMD* de admitir diferentes definiciones de servicios desde distintos proveedores o descripciones heterogéneas, para integrarlas y poder intermediar entre todos los servicios que el *BrokerMD* incluye. En nuestra propuesta de *BrokerMD* se ha tratado de establecer un único esquema para la definición de esos servicios usando `dmcc-schema`; esto tiene varias ventajas, tales como la utilización de un único esquema de definición de datos simplifica la gestión de todas las partes que se integran en el *Broker* con lo que no hay que lidiar con diferentes definiciones de datos para cada uno de los componentes como el catálogo, la consulta, el almacenamiento de la base de datos o el modelado de la experimentación.

La propuesta de *Broker* que se ha desarrollado admite dos modos de agregación de servicios. Estos modos de agregación son las siguientes:

- **Por parte de los proveedores de Cloud Computing**. La agregación de servicios corresponde al proveedor y es este quien adapta sus servicios al esquema que gobierna la definición de los mismos en *BrokerMD* o bien añade una capa adicional para el consumo de servicios mediante descripciones semánticas concretas. De modo que se consigue que el proveedor pueda utilizar un esquema común para la definición de los servicios de minería de datos que ofrece, suponiendo una mejora importante en la portabilidad de los mismos hacia/desde otros proveedores; pero lo que más destaca de esta parte dentro del *Broker* es la capacidad de integrar un esquema de definición de servicios usando **dmcc-schema** para homogeneizar las particularidades de los mismos.
- **Mediante descubrimiento de servicios**. Esta parte denominada en inglés *Service Discovery* permite al modelo ontológico tener en cuenta las propiedades, atributos y relaciones funcionales y no funcionales, la plataforma y los servicios de software para facilitar el descubrimiento y la selección de los servicios o proveedor adecuado. Cada proveedor que transforme su servicios dentro del esquema de `dmcc-schema`, será fácilmente encontrable por entidades como el *BrokerMD* que conoce perfectamente como trabajar con el esquema. El descubrimiento de servicios no solo se centra en encontrar servicios de mismo tipo que se publiquen desde los proveedores, también es responsable de incluir su descripción y definición de servicios dentro del *BrokerMD* de modo que el nuevo servicio pueda ser integrado sin ningún tipo de fricción ya que ambos "hablan el mismo lenguage". Gracias a la tecnología semántica, es posible publicar y descubrir servicios web dentro de un ecosistema orientado a los servicios. Descubrir servicios es una tarea fundamental que implica el proceso de búsqueda de servicios web que satisfagan requisitos específicos. El descubrimiento de servicios sin considerar un esquema de modelado intermedio como `dmcc-schema` u otros basados en tecnología semántica, es una tarea no trivial, ya que la búsqueda de texto y el descubrimiento de entidades, atributos y relaciones en los sistemas de información y documentación de los servicios no llegará demasiado lejos, ya que el texto en lenguaje natural es a menudo informal y, en consecuencia, los nombres de los servicios y las documentaciones pueden ser ambiguos. Se requieren representaciones ricas y formales de los servicios e interacciones para capturar la semántica de las solicitudes y descripciones de los servicios, así como el contexto de

una interacción propuesta con los servicios. De este modo la utilización de `dmcc-schema` puede eliminar la problemática de un descubrimiento complejo basado en la extracción de información sobre texto natural.

BrokerMD puede almacenar descripciones y definiciones de servicios desde diferentes proveedores, siempre que publiquen sus servicios siguiendo la propuesta de definición de servicios `dmcc-schema`. De este modo, los servicios estarán disponibles para ser consultados, agregados e integrados dentro del catálogo del *Broker* como si fuesen un único servicio desde el que interactuar con los demás proveedores y servicios.

5.5. Selección de servicios de minería de datos

El modelado del problema para la selección e intermediación de servicios para Cloud Computing es especialmente complejo si nos centramos en el problema específico de la selección de servicios de minería de datos desde diferentes proveedores de Cloud Computing.

5.5.1. Definición del problema de la selección de servicios de minería de datos en entornos multi-proveedor

Después del estudio preliminar sobre el modelado de este tipo de servicios en los proveedores más conocidos de Cloud Computing, el modelo de servicio dirigido a los usuarios se establece en diferentes modos tales que los servicios son entregados a los usuarios o aplicaciones finales de la forma más efectiva para el usuario. Hay que tener en cuenta que cualquier uso/consumo de recursos del proveedor de Cloud Computing puede estar sujeto a costes de muy tipo muy variado; la idea de este tipo de servicios de alto nivel es intentar reducir al mínimo el número de recursos involucrados en la entrega y despliegue de los mismos, de modo que se facilite al usuario final la selección del servicio desde el proveedor final.

Esto supone una simplificación desde el punto de vista del usuario, al cual probablemente únicamente le pueden interesar aspectos como el precio

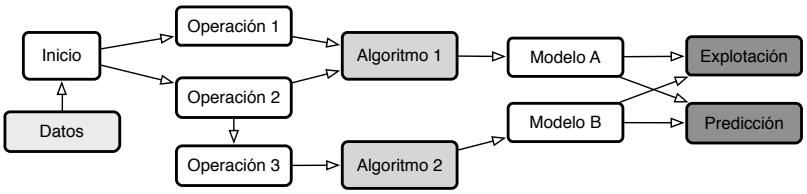

Figura 5.8: Ejemplo de grafo de operaciones en un flujo de trabajo tradicional con minería de datos.

total del uso del servicio a lo largo de tiempo o la calidad de los servicios ofertados medidos de diferente forma (QoS). En los servicios de minería de datos este tipo de simplificación sólo se da en determinados escenarios, tales como el trabajo con funciones donde prima la utilización de operaciones de procesamiento como servicio o también algoritmos como servicios, dentro de un entorno controlado. El proceso de trabajo con minería de datos en Cloud Computing es mucho más complejo tal y como se explica en la sección 3. No sólo consiste en aplicar funciones, operaciones o algoritmos, también es necesario realizar un flujo de trabajo, evaluar resultados, trabajar con modelos o desplegar entornos para la explotación de esos modelos. Además de todos estos elementos, diferentes partes del proceso de trabajo con minería de datos están condicionadas al uso de diferentes estructuras de cómputo de tipos muy variados, ofreciendo distintos rendimientos en su ejecución, y por tanto ofreciendo mejores o peores resultados en cuanto a rendimiento en el trabajo con este tipo de problemas. De forma adicional, otros aspectos como los procesos de entrenamiento de los datos y creación de los modelos son tenidos muy en cuenta como elementos de peso en el trabajo de minería de datos como servicio.

Un problema de minería de datos a nivel abstracto, está compuesto de una serie de operaciones de diferente tipo que se realizan siguiendo una estructura como se puede ver, por ejemplo, en la figura 5.8. Ese procesamiento de los datos desde el servicio de minería de datos puede ser visto como un subconjunto de operaciones o tareas con unas características concretas (ver Figura 5.9). Estas características hacen referencia a aspectos relacionados directa o indirectamente con la operación a realizar, de modo que tal y como se observa en la figura 5.10, cada elemento que forma el servicio está sujeto a costes y restricciones de diferentes tipos impuestos por el modelo de negocio del proveedor de Cloud. Este modelo de negocio de un proveedor de servicios de Cloud Computing tiene en cuenta una elevada cantidad de condicionantes que afectan a diferentes tipos de costes (asociados al consumo de recursos, tiempo, movimientos de datos, etc.); el proveedor debe rentabilizar toda la

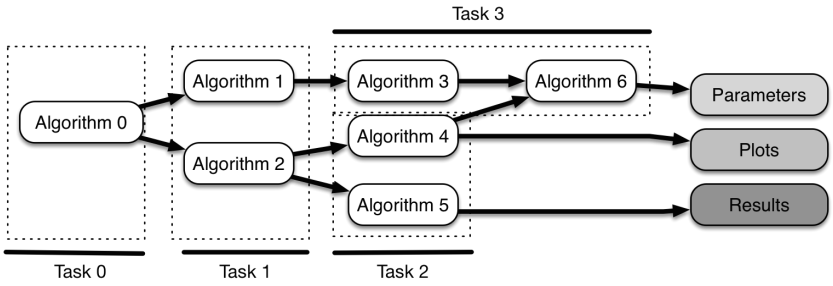

Figura 5.9: Ejemplo de un flujo de trabajo con minería de datos donde se realizan diferentes operaciones y al final del proceso podemos tener resultados, modelos, gráficas, o datos, entre otros.

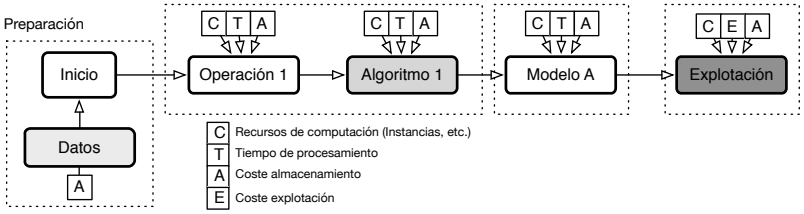

Figura 5.10: Algunos de los elementos modificadores de coste o tiempo en el proceso de trabajo con minería de datos en Cloud Computing.

inversión realizada para proveer de la capacidad de ofrecer infraestructura, comunicaciones, o almacenamiento por ejemplo, como elementos base de gran parte de los servicios de las capas superiores.

Un subconjunto de los modificadores de los costes y restricciones considerados por los proveedores para los servicios de minería de datos son detallados en la tabla 5.3. En esta tabla se puede ver como prácticamente para cualquier operación que se lleve a cabo está sujeta a costes y restricciones que finalmente afectan al coste económico (uso de instancias, transferencias, etc.), el tiempo de ejecución (tipo de instancia, volumen de datos, etc.) o la calidad del servicio esperada según la configuración (localización, movimiento de datos, tipo de instancias, etc.) utilizada en cada caso.

Sin embargo, la selección de un servicio de minería de datos para resolver un problema de procesamiento de cualquier tipo, se complica cuando en la selección aparecen más de un proveedor de Cloud Computing que estén habilitados con este tipo de servicios. Este escenario hace que la selección de los servicios se vuelva una tarea mucho más compleja puesto que el número de elementos a considerar para conseguir una selección de servicios lo más ajustada posible a los criterios y preferencias propuestos por parte del usuario,

Modificador	**Unidad**	**Incidencia**
Carga de datos	$/GB	Baja/Media/Alta
Salida de datos	$/GB	Media
Uso de instancias	$/h.	Media/Alta
Explotación modelos	$/paquetes	Baja
Algoritmos	$/h y $/llamadas	Media
Funciones como servicios	$/h y $/llamadas	Media/Alta
Almacenamiento modelo	$/GB	Baja/Media
Localización de instancias	$/h.	Media
Movimiento datos	$/GB	Baja

Tabla 5.3: Modificadores del coste a varios niveles cuando se trabaja con minería de datos en Cloud Computing. La incidencia refleja el grado de peso de los modificadores, de modo que una incidencia Media/Alta, indica que los modificadores pueden variar desde un tipo de coste modelado por uso de por ejemplo instancias sencillas, a un coste alto, por ejemplo al usar instancias de despliegue con rendimiento mucho mayor, como *GPUs* o *FPGAs*.

se ve incrementado de forma considerable, haciendo inviable una selección eficaz y eficiente de los mismos.

En este contexto multi-proveedor, la utilización de un *Broker* para Cloud Computing surge como una de las mejores y más consistentes opciones para la selección y mediación de servicios de minería de datos para múltiples proveedores, ofreciendo un servicio de intermediación que unifica de forma inteligente la gestión de recursos en este tipo entornos. En nuestra propuesta de intermediación de servicios de minería de datos el *Broker* se encarga de diferentes aspectos tales como la agregación, la consulta, la selección o la intermediación, entre otros. Los aspectos relacionados con la selección de servicios son uno de los componentes angulares en la construcción de un *Broker*; es de tal importancia, que una mala planificación de la gestión de los servicios del *Broker*, puede redundar en un incremento de costes económicos, una perdida de calidad del servicio o un bajo rendimiento para los servicios de minería de datos.

5.5.2. Herramienta multi-objetivo para la toma de decisiones en Cloud Computing

La selección de los servicios y los proveedores que realizarán el procesamiento en este tipo de problemas es una tarea muy costosa: supone el análisis

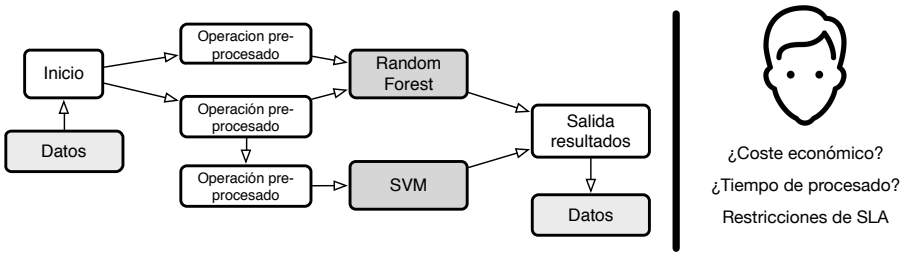

Figura 5.11: Flujo de operaciones de minería de datos y preguntas que el *Broker* puede responder al usuario.

de parámetros y decenas combinaciones para cada servicio y para cada proveedor con el que el *Broker* interactúe. Por consiguiente, un *Broker* debe estar diseñado para ofrecer al usuario una selección de los servicios que mejor se ajusten a las preferencias y restricciones que le indiquen de los usuarios consumidores sobre el problema a desarrollar. Por ejemplo, en la figura 5.11 podemos ver un conjunto de muestra con un flujo de trabajo, que un usuario podría querer solucionar como parte de un problema de minería de datos, y donde en la columna de la derecha, se establecen algunos criterios de selección y restricciones sobre el problema que podrían plantearse al *Broker*. Para poder realizar una selección efectiva y eficiente de servicios/proveedores, el *Broker* de Cloud Computing a través de diferentes algoritmos de optimización y planificación tiene la capacidad de proveer al usuario final de una herramienta completa de toma de decisiones a través de la cual presentarle la solución o conjunto de soluciones acordes a los criterios que se hayan establecido. Finalmente el *Broker* tendrá la competencia de intermediar para que el procesado de minería de datos que el usuario quiere resolver se convierta en real combinando diferentes servicios desde uno o varios proveedores.

La complejidad de este tipo de *Brokers* radica en el hecho de que trabajan con servicios y proveedores dentro un entorno *inter-cloud*. La idea detrás *inter-cloud* es que en una única funcionalidad común combinaría distintas nubes individuales en un conjunto sin fricciones en términos de operaciones y servicios bajo demanda.

El modelo *inter-cloud* simplemente se aseguraría de que una nube pueda utilizar recursos fuera de su alcance aprovechando los servicios de agregación preexistentes con otros proveedores de Cloud Computing. Dicho de otro modo permitiría utilizar recursos desdes otros proveedores, de forma exclusiva o combinada, para finalmente solucionar un problema de entrega de servicios global. El concepto de *inter-cloud* está íntimamente relacionado con el *Bro-*

ker, ya que este último es clave como elemento "unificador" de las diferentes nubes de los proveedores, y más concretamente en los servicios de minería de datos, donde será el encargado de seleccionar "¿qué?", "¿cuándo?", "¿cómo?" y "¿dónde?" se debe realizar el flujo de trabajo:

- **¿Qué?.** Hace referencia al flujo de trabajo o al conjunto de operaciones de minería de datos que hay que desarrollar. El usuario es el responsable de proporcionar al servicio del *Broker* el grafo con las operaciones que formarán parte del flujo de trabajo. Este flujo de trabajo puede estar comprendido desde simples operaciones, como *"aplicar una transformación de dominio a los datos"*, a operaciones más complejas que pueden ser *"predecir la temperatura con un horizonte de 12 horas"*.

- **¿Cómo?.** Cuando tenemos un flujo de procesamiento de operaciones de minería de datos, es necesario tener una planificación adecuada de las tareas u operaciones que se realizarán. Esto quiere decir que el *Broker* será capaz de estructurar el proceso de ejecución de las tareas como servicios dentro de los diferentes proveedores de servicios de minería de datos, identificando que funciones realizar y como hacerlas, que resultados se obtienen, como se utilizan, etc.

- **¿Cuándo?.** La planificación de las tareas del flujo de trabajo en minería de datos realizada desde diferentes proveedores está sujeta a una serie de restricciones sobre la secuenciación y el paralelismo. Esto quiere decir que a la hora de saber "cómo" se harán las tareas es necesario estudiar "cuándo" se realizarán de modo que la planificación pueda ser más eficaz a la hora de por ejemplo seleccionar tareas que podrán ejecutarse en paralelo u otras que tendrán que esperar resultados, además de considerar los movimientos temporales de datos entre proveedores, siempre que existan.

- **¿Dónde?.** Finalmente, otro aspecto clave es donde se ejecutan los servicios. Teniendo en cuenta que estamos trabajando en entornos *intercloud*, los servicios pueden ser en teoría ejecutados en diferentes proveedores de Cloud y por ende sobre arquitecturas y redes de comunicación con diferentes latencias. No sólo existe esta problemática, también consideramos que incluso dentro de cada proveedor, la localización de los centros de computación y almacenamiento de datos tiene que ser considerado a la hora de ofrecer los servicios de intermediación a través del *Broker* hacia los usuarios finales o la explotación de los servicios por los usuarios.

5.5.3. Formulación del problema multi-objetivo

En un entorno gestionado por un *Broker*, el usuario debe proveer los criterios y restricciones específicas para el desarrollo del procesamiento del flujo de trabajo en minería de datos. De este modo el *Broker* servirá como una herramienta para la toma de decisiones, ofreciendo los resultados de la selección de servicios y proveedores junto con el plan de ejecución de las operaciones de minería de datos, siempre ajustados a los criterios y restricciones del usuario. El *Broker* además de considerar estos aspectos sobre las restricciones de la ejecución, tiene en cuenta una serie de objetivos, que permiten al usuario conocer aspectos cruciales en la toma de decisión a la hora se la selección en la intermediación, como el coste económico de la ejecución, la estimación del tiempo de ejecución, u otros aspectos que el usuario podría contemplar como la calidad del servicio (QoS) medida desde diferentes puntos de vista.

Por tanto, por un lado tenemos que solucionar una planificación de operaciones (distribuidas entre diferentes servicios y proveedores, con diferentes localización y coste), y por otro lado se busca una optimización de todo el proceso de modo que el usuario pueda tomar una decisión en la selección de la intermediación a través del *Broker*, teniendo en cuenta diversos objetivos de optimización general. Este tipo de problemas se denominan problemas de optimización multi-objetivo. La optimización multi-objetivo implica la minimización o maximización de múltiples funciones sujetas a un conjunto de restricciones. Los enfoques comunes para la optimización multi-objetivo incluyen: a) **logro de objetivos**: reduce los valores de una función vectorial lineal o no lineal para alcanzar los valores de objetivo dados en un vector de objetivo. La importancia relativa de los objetivos se indica mediante un vector de pesos. Los problemas de logro de ojetivos también pueden estar sujetos a restricciones lineales y no lineales, b) **minimax**: minimiza los valores en el peor de los casos de un conjunto de funciones multivariadas, posiblemente sujetas a restricciones lineales y no lineales, c) **frente de Pareto**: encuentra soluciones no inferiores, es decir, soluciones en las que una mejora en un objetivo requiere una degradación en otro.

Para un problema de optimización multi-objetivo no trivial, no existe una solución única que optimice simultáneamente todos los objetivos. En ese caso, se dice que las funciones objetivo entran en conflicto, y existe un número de soluciones denominadas soluciones óptimas de Pareto. Una solución se llama no dominada, u óptima de Pareto, si ninguna de las funciones objetivo puede ser mejorada en valor sin degradar algunos de los otros valores objetivos. Sin información adicional, todas las soluciones óptimas de Pareto

se consideran igualmente buenas (ya que los vectores no se pueden ordenar por completo). El objetivo en este tipo de problemas multi-objetivo es encontrar un conjunto representativo de soluciones óptimas de Pareto, y/o cuantificar las ventajas y desventajas de satisfacer los diferentes objetivos, y/o encontrar una solución única que satisfaga los criterios un usuario para la toma de decisiones. El problema de optimización multi-objetivo se puede definir matemáticamente como, encontrar un vector $x^* = [x_1^*, x_2^*, \cdots, x_n^*]^T$, que satisfaga las m restricciones:

$$g_i(x) \geq 0 \qquad i = 1, 2, \cdots, m \qquad (1)$$

y las p restricciones:

$$h_i(x) = 0 \qquad i = 1, 2, \cdots, p \qquad (2)$$

y optimice la función vectorial:

$f(x) = [f_1(x), f_2(x), \cdots, f_k(x)]^T$

de donde $x = [x_1, x_2, \cdots, x_n]^T$ es el vector de variables de decisión.

El modelado multi-objetivo del problema de optimización llevado a acabo por el *Broker* se detalla en las siguientes subsecciones.

5.5.4. Objetivos de la selección de servicios

Los objetivos que se han tenido en cuenta para el diseño de la selección de los servicios de intermediación de *Broker* son los siguientes:

1. **Coste económico** global flujo de trabajo de minería de datos. Uno de los factores a tener en cuenta a la hora de desplegar servicios de minería de datos en Cloud es el coste económico de ejecutar procesos de minería de datos a gran escala. En las infraestructuras «on-premise» que disponen las organizaciones y empresas, los costes económicos del procesamiento podrían ser prorrateados y se calcularían en base a la depreciación de la infraestructura y al consumo de energía eléctrica. En este tipo de flujos de trabajo el proveedor de servicios puede imputar costes económicos muy variados, tales como la localización de los datos, el tipo de servicio, el almacenamiento usado, o la tipología de la infraestructura, por ejemplo. De modo que prácticamente cualquier operación o servicio que haga uso de recursos es susceptible de ser tasada económicamente.

2. **Tiempo estimado** de cómputo flujo de trabajo de minería de datos. El usuario probablemente necesite conocer una estimación del tiempo total y un seguimiento individualizado del coste de tiempo de cada una de las tareas para la ejecución de todo el flujo de trabajo completo de minería de datos.

3. **Calidad del Servicio**. Este objetivo, también llamado *Quality of Service* (QoS) es la descripción o medición del rendimiento global de un servicio. En nuestro caso, por ejemplo, un servicio de computación en nube, correspondería en particular al rendimiento visto desde diferentes puntos de vista por parte de usuarios consumidores. Para medir cuantitativamente la calidad de servicio, se consideran varios aspectos relacionados con el servicio de red, tales como la pérdida de conexión, la velocidad, el retardo de transmisión, la disponibilidad del servicio, etc. En determinados escenarios los clientes buscan obtener unos parámetros de calidad de servicio mínimos que le garanticen ciertos aspectos que pueden ser decisivos para su trabajo. Estos aspectos abarcan un espectro muy amplio de posibilidades lo que hace que el estudio de los mismo sea complejo.

Para cada de uno de estos objetivos se ha detallado un conjunto adicional de medidas por los que se ve afectado.

1. **Coste económico**: Objetivo a *minimizar*. Para el coste económico se han teniendo en cuenta los siguiente sub-indicadores:

   C_{ih} identifica el coste económico de cada una de las instancias utilizadas en flujo de trabajo de minería de datos. Es dependiente del tiempo en horas que se use la instancia y del tipo de instancia sobre la cual se ha ejecutado alguna operación.

   C_{srio} identifica el coste económico por el uso del almacenamiento desde diferentes localizaciones. El coste de almacenamiento se realiza por GBytes de datos almacenado o transferido desde o hacia el proveedor. Depende de la región o área donde se encuentren los datos, de modo que un mismo conjunto de datos cargado en dos localizaciones diferentes pueden tener costes distintos. Estos costes de almacenamiento están supeditados a operaciones que incluyan tanto entrada o salida de información, que puede ser susceptible de carga o almacenamiento y donde también existen escenarios que requieran movimiento de datos entre diferentes localizaciones.

C_{sla} con este modificador de coste tenemos en cuenta todos los aspectos que subyacen de los acuerdos de nivel de servicio. Es decir, cuando utilizamos unos servicios de un proveedor, el proveedor nos hace firmar unos "mínimos" de servicio, o acuerdos, desde donde el proveedor ajusta los precios de los servicios cuando se producen por ejemplo caídas de sus sistemas e infraestructuras, o acceso a sus servicios. Cuando esto ocurre el usuario es bonificado según los acuerdos del nivel de servicio, de modo que por ejemplo para una SLA donde se indica que si ocurre un porcentaje de inactividad del 1 % al mes (7,2 horas de inactividad en todo el mes), usuario es bonificado (con créditos o con reembolso de dinero).

C_{api} con operaciones que forman parte del flujo de trabajo de minería de datos, se considera que existen unos costes debidos a la explotación de resultados almacenamos en el proveedor. Estos resultados pueden ser por ejemplo los modelos desarrollados para la predicción de datos, por ejemplo. El proveedor puede incluir un coste adicional por ese uso del modelo y por su explotación del cara al usuario final. El ejemplo más claro es cuando una vez desarrollado un modelo, comenzamos a hacer peticiones al modelo para obtener una predicción de los datos; cada una de esas llamadas o bloques de llamadas tienen un coste asociado que varía en función del volumen de las mismas.

C_{ir} identifica el coste de uso y explotación de una instancia en una región determinada. Las instancias pueden estar distribuidas en diferentes localizaciones, de modo que según donde esté geográficamente tendrá un coste asociado.

C_{sio} corresponde al coste de entrada y salida de datos desde las diferentes operaciones.

C_{tra} identifica el coste que supone la transferencia de datos de un proveedor a otro para las operaciones en las que es necesario transportar los datos tanto desde la salida de un proveedor como la entrada de datos hacia otro.

2. **Coste temporal**: Objetivo a *minimizar*. Para el coste en tiempo estimado de la ejecución de una experimentación en minería de datos, se han considerado los siguientes sub-objetivos:

   T_e identifica el tiempo estimado de ejecución de cada una de las tareas u operaciones del flujo de trabajo.

T_l corresponde a la latencia entre el fin de una tarea y el inicio de otra tarea. Este tiempo se considera tanto dentro del proveedor (es decir de las operaciones que realizan en un único proveedor), como en las tareas que se desarrollan entre diferentes proveedores.

T_t se refiere al tiempo de transferencia de los datos entre diferentes proveedores.

3. **Calidad del servicio**: Objetivo a *maximizar*. Para la calidad del servicio se han tenido en cuenta los siguientes elementos:

Q_{sla} corresponde a la métrica usada para evaluar el porcentaje de tiempo que los servicios del proveedor no han estado disponibles para los usuarios. Generalmente en los acuerdos de nivel de servicio se especifican los mínimos a partir de los cuales, el proveedor debe bonificar al usuario por el servicio que no ha podido prestar y por los problemas que eso implicase para el usuario.

Q_{ip} se refiere a la incidencia en la calidad del servicio que supone la utilización de diferentes proveedores como parte del flujo de trabajo en minería de datos. De este hecho se deduce que un incremento en el número de proveedores `inter-cloud` que intervengan, ocasionaría un descenso en la calidad del servicio, desde el punto de vista del rendimiento, tiempo de ejecución, etc.

Q_{ld} identifica aspectos relacionados con la localización tanto de las instancias como de los datos en el mismo proveedor y en proveedores `inter-cloud`.

5.5.5. Propuesta de selección basada en algoritmos MOEA

Una solución exacta para el problema de la planificación de tareas requiere un número de pasos computacionales que crecen muy rápido en función de la complejidad y del tamaño del problema. Como no se puede encontrar de manera eficiente una solución óptima para este tipo de problemas considerados NP-completos, se ha dedicado una gran cantidad de tiempo al estudio de algoritmos que proporcionen soluciones aproximadas cuyo tiempo de ejecución es manejable. La solución de problemas NP-completos ha sido estudiada desde diferentes puntos de vista, como búsqueda local, métodos heurísticos o algoritmos evolutivos, entre otros.

Es necesario utilizar algoritmos que tengan un buen equilibrio entre rendimiento y resultados prometedores. Al tratarse de problemas de optimización

donde existen varios objetivos que a menudo son contrapuestos, no es posible centrarse en uno solo de los objetivos, sin que los demás no se vean afectados. Una solución que es óptima puede no ser óptima en comparación con otras. Diferentes soluciones pueden producir múltiples opciones que suponen un compromiso entre unos y otros objetivos.

Para esta propuesta de optimización se ha seleccionado una solución basada en algoritmos evolutivos, más concretamente en Algoritmos Evolutivos Multi-Objetivo (MOEA). El problema de la selección de servicios desde distintos proveedores de minería de datos, con restricciones tales como el tiempo de inicio de las tareas, la precedencia de operaciones, la reserva de recursos, la disponibilidad de los mismos o la localización, entre otros, supone la resolución de un problema de planificación de recursos, donde hay que satisfacer una serie de restricciones. Este tipo de problemas se denominan "Resource Constrained Scheduling Problems" (RCSP) y han sido ampliamente estudiados [Reklaitis, 1996]. Una de las aproximaciones más utilizadas para dar solución a este tipo de problemas se basa en la aplicación de algoritmos evolutivos gracias a su capacidad para abordar espacios de soluciones de tamaño considerable y obtener resultados relativamente buenos en un tiempo limitado [Tian and Murata, 2016]. Los algoritmos evolutivos son métodos adaptativos, basados en procesos genéticos que ocurren en la naturaleza. Así, a través de las evolución de las generaciones a lo largo del tiempo, los individuos y las poblaciones consiguen evolucionar de acuerdo con la selección natural y la supervivencia de los más fuertes. La evolución del algoritmo y su rendimiento en la obtención de soluciones prometedoras depende en gran medida de una codificación adecuada, que veremos más adelante.

Los principios básicos de estos algoritmos se definen en diferentes trabajos [Holland, 1975, Goldberg, 1989, Reeves, 2003, Michalewicz, 1996]; el algoritmo genético subyacente (ver figura 5.12) mantiene una población de soluciones o individuos, donde cada uno de ellos es una solución al problema. Cada individuo tiene asociado un valor (objetivo único) o valores (multi-objetivo) que establecen la bondad de la solución con respecto a los objetivos. Un subconjunto de individuos de la población en cada generación se cruzan y mutan, de modo que se crean nuevos individuos, diferentes de los originales, y se añaden a la población. Para cada generación se seleccionan los individuos que pasarán a la siguiente generación y así sucesivamente hasta alcanzar un número máximo de soluciones o un objetivo específico.

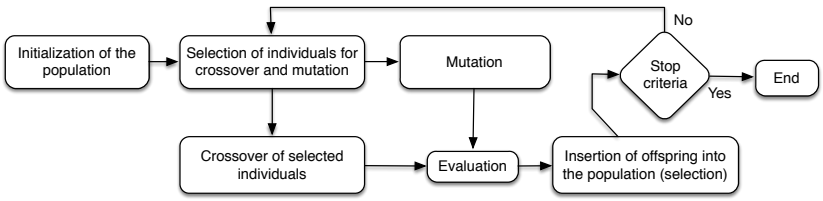

Figura 5.12: Diagrama de funcionamiento de un algoritmo evolutivo estándar.

5.5.5.1. Representación de las soluciones

Como se ha indicado en el apartado anterior, la representación de los individuos (cromosoma o solución) de la población que gestiona el algoritmo evolutivo es crítica para asegurar el éxito.

Para nuestro problema, la codificación seleccionada se basa en la organización y planificación de tareas que deben ejecutarse a modo de flujo de trabajo. Cada una de las operaciones necesita considerar cuando se inicia y de cuáles tareas depende su ejecución. Además de esta parte, que se encarga de la gestión en el tiempo y las precedencia de operaciones, es necesario tener en cuenta parte de las características de las operaciones, por ejemplo relativas a localización de la tarea, parametrización o modelo de despliegue, entre otras. De este modo, la forma de componer una tarea se subdivide en tiempo t y modo de ejecución m. El tiempo controla cuando se debe iniciar la tarea con respecto a su tareas padre u origen y el modo controla todos los elementos adicionales de ejecución de la operación. En la figura 5.13 se representa el modelado de las soluciones, donde cada operación está dividida en dos partes: tiempo y modo de operación. La parte correspondiente al modo de operación se desglosa del siguiente modo:

En la Figura 5.16 se puede ver el traspaso de un flujo de trabajo de operaciones de minería de datos a la representación de la solución o cromosoma que usará el algoritmo evolutivo para trabajar. En esta representación se convierte cada una de las operaciones en una estructura tipo vector con cada uno de os componentes de tiempo y modo de operación desglosados.

Una tarea u operación se ha definido del siguiente modo:

1. Inicio de la tareas. El inicio de la tarea corresponde al valor temporal de cuando una tarea comienza con respecto a su tarea padre. Este valor puede ser positivo o cero de modo que si es positivo indica que una tarea empieza tiempo después que su predecesora; si es cero indica que

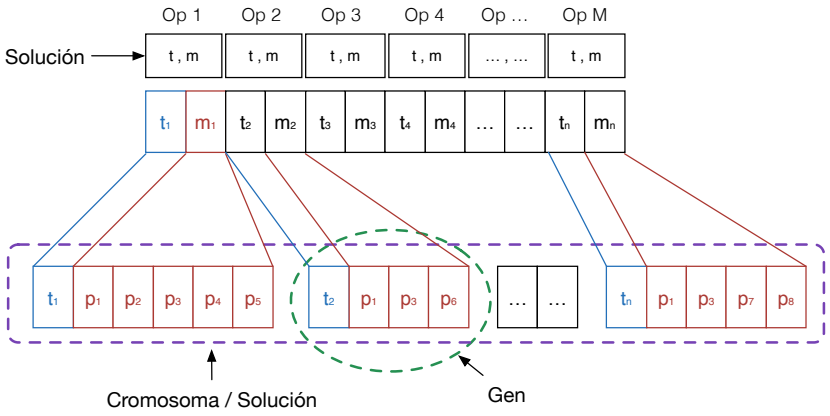

Figura 5.13: Diagrama detallado con todos los elementos de la representación de las soluciones (cromosoma) y el desglose de cada uno de los genes (operaciones/tareas) en componentes de tiempo y modo de ejecución.

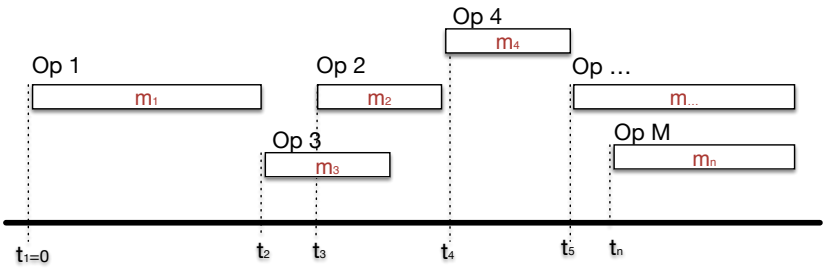

Figura 5.14: Ejemplo de una planificación de operaciones

la tareas comienza inmediatamente después de su tarea predecesora. El tiempo de inicio de la tarea se define en unidades de tiempo. Además para poder gestionar este valor es necesario mantener meta-información sobre la jerarquía de operaciones según la entrada que el usuario provee sobre como se debe ejecutar el flujo de trabajo. La Figura 5.14 representa como se modelan las tareas y la precedencia de las mismas.

2. Proveedor de Cloud Computing donde se ejecutan operaciones del proceso de minería de datos. Una tarea puede estar disponible o no en uno u otro proveedor, de modo que las tareas, pueden ser ejecutadas desde diferentes proveedores. Este valor contiene la información del identificador donde la tarea se va a ejecutar.

3. Localización física de la ejecución. Hace referencia al centro de datos donde las instancias se están ejecutando, siempre que esto sea conocido.

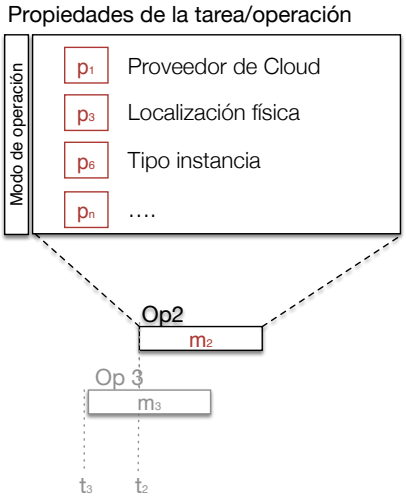

Figura 5.15: Detalle del desglose de parte de los componentes del modo de ejecución de una operación. En este caso el modo indica el proveedor, la localización o la instancia entre otros.

4. Tipo de operación realizada. Las operaciones que se pueden desarrollar en minería de datos son de tipo muy variado y depende de la categoría concreta. Cada una de las operaciones está sujeta a unas determinadas restricciones, de modo que por ejemplo no tienen las mismas implicaciones la explotación del modelo que el entrenamiento del modelo, en cuanto a costes económicos, de tiempo o utilización de recursos de almacenamiento y computación. En la Figura 5.15 se puede ver el detalle de algunos de los parámetros de ejemplo utilizados para desplegar el modo de ejecución de la tarea. En nuestro trabajo con minería de datos se han seleccionado los siguientes bloques:

   - Operaciones generales (incluidas operaciones de pre-procesamiento).
   - Obtención del modelo.
   - Explotación del modelo.
   - Fase de entrenamiento.
   - Aplicación de algoritmos de minería de datos.
   - Adquisición y salida de datos.

5. Soporte de almacenamiento. Cuando las operaciones de minería de datos son ejecutadas pueden requerir almacenamiento. Por un lado la entrada de datos, pero también es posible que sea necesario almacenar los datos temporales o intermedios en el espacio de almacenamiento

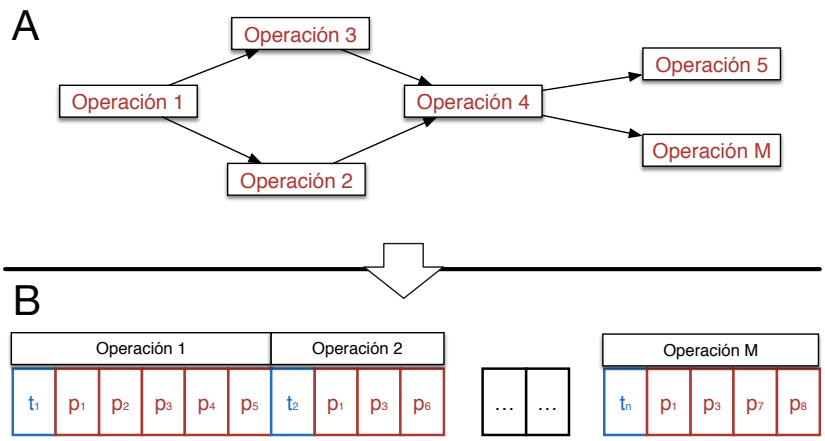

Figura 5.16: Ejemplo de la representación de un modelado de flujo de trabajo en minería de datos a una codificación válida para usar en el algoritmo evolutivo.

masivo del proveedor o los proveedores en cuestión. En estos casos se indica que la operación está sujeta a la utilización de almacenamiento.

6. Utilización de instancias. Determinadas operaciones como entrenamiento, modelado, o pre-procesamiento de datos requieren de instancias de computación. Según el número de instancias utilizadas y el tiempo transcurrido de uso de las mismas, se establece un coste económico y temporal. Sin embargo para otras operaciones la selección de instancias no está disponible; esto ocurre cuando se utilizan funciones como servicio o *FaaS*, donde en este caso no es posible conocer cuantas instancias son utilizadas para la ejecución de una tarea, ya que *FaaS* está enfocada en los resultados y no en la infraestructura que se despliega.

7. Parametrización de la operación. Cuando se aplica un algoritmo dentro de una operación a un conjunto de datos, este algoritmo puede ejecutarse con una serie de parámetros de configuración que permiten modificar el comportamiento del algoritmo. Estos parámetros permiten afinar el modo de ejecución del algoritmo y por tanto obtener mejor rendimiento, precisión o calidad de los resultados; además la parametrización redunda directamente en los costes económicos y temporales de esas operaciones. En nuestra propuesta no hemos desarrollado este aspecto en su totalidad, ya que el simple hecho de seleccionar unos parámetros de ejecución de un algoritmo para obtener unos resultados u otros, es en sí mismo es un problema de optimización.

5.5.5.2. Formulación de los objetivos

Los objetivos de la selección de servicio para los flujos de trabajo en minería de datos son tres: a) cálculo del coste económico, b) cálculo del tiempo de ejecución y c) valor de QoS (calidad del servicio). En los siguientes párrafos se detallan cada uno de los objetivos, el desglose de los mismos y cómo se calculan sus valores.

Cálculo del coste económico

El cálculo del **coste económico** C_e de una solución corresponde con la suma de los costes de cada una de las operaciones o tareas que realizan en el flujo de trabajo de minería de datos:

$$C_e = \sum_{i=1}^{n} C_{op_i} \tag{5.1}$$

de donde C_{op_i} se desglosa como la suma de diferentes costes asociados a la tarea que se realiza. Estos costes están condicionados a las propiedades de la tarea:

$$C_{op_i} = C_{ih} + C_{srio} + C_{sla} + C_{api} + C_{ir} + C_{sio} + C_{tra} \tag{5.2}$$

de donde los costes se reparten del siguiente modo: C_{ih} es el coste de la instancia del servicio, C_{srio} es el coste del almacenamiento por localización, C_{sla} es el coste del SLA, C_{api} incluye los costes asociados si la operación usa la API, C_{ir} es el coste de explotación en una localización, C_{sio} es el coste de transferencia de datos entre operaciones y C_{tra} es el coste imputado del traspaso de datos entre proveedores (ver detalles de componentes en la sección: 5.5.4).

Cálculo del coste de tiempo

El cálculo del objetivo correspondiente al tiempo se realiza sumando cada coste de tiempo de las operaciones que se llevan a cabo. La formula para ello es la siguiente:

$$T_g = \sum_{i=1}^{n} T_{op_i} \tag{5.3}$$

de donde T_{op_i} se desglosa como la suma de diferentes costes de tiempo asociados. Estos costes temporales están condicionados a la tipología de la tarea:

$$T_{op_i} = T_e + T_l + T_t \tag{5.4}$$

De donde T_e es el tiempo de ejecución estimado de la tarea, T_l es el tiempo de latencia, T_t es el coste de tiempo que supone mover o transferir los datos de las operaciones con salida de un proveedor a otro (ver detalles de componentes en la sección: 5.5.4).

Cálculo del QoS

El cálculo del QoS se hace sumando cada uno de los elementos que afectan a la calidad de los servicios. La formula para ello es la siguiente:

$$Q_s = \sum_{i=1}^{n} Q_{op_i} \tag{5.5}$$

de donde Q_{op_i} se desglosa como la suma de diferentes medidas de calidad que se han considerado. Estas medidas están condicionadas a la tipología de la tarea:

$$Q_{op_i} = Q_{sla} + Q_{ip} + Q_{ld} \tag{5.6}$$

Q_{sla} es la métrica que penaliza la violación del SLA, Q_{ip} es el valor utilizado para penalizar la ejecución en *inter-cloud* de los servicios y Q_{ld} que penaliza la localización de los algoritmos en diferentes regiones (ver detalles de componentes en la sección: 5.5.4).

5.5.5.3. Operadores

La implementación de algoritmos evolutivos requiere de la definición de operadores de inicialización, cruce, mutación y selección para explorar y explotar el conjunto de soluciones en busca de resultados prometedores. Debido a la naturaleza del problema y a la aplicación crítica de los operadores, es necesario diseñar de forma coherente cada uno de los operadores, de modo que puedan actuar tanto en la parte del tiempo como en la del modo de ejecución de forma independiente.

5.5.5.3.1. Inicialización La población inicial se genera de forma aleatoria. De este modo, durante la evolución de las generaciones del algoritmo, las soluciones factibles pueden coexistir con soluciones no factibles. Esto permite explorar soluciones que no son factibles a priori, pero que pueden estar cerca de ser buenas si las soluciones continúan la evolución. De esta manera, se produce diversidad dentro de la ejecución del algoritmo genético. Además, debido a esta condición, es necesario el uso de un operador de reparación (ver subsección 5.5.5.3.4) que permite hacer viable un porcentaje de la población que ha sido modificado tras el cruce o la mutación, de forma que la calidad de la población que evoluciona no se deteriore en gran medida.

La creación de la población inicial necesita conocer la estructura del gen individual (u operación) y del cromosoma para generarlo (una solución). Cada gen está compuesto de dos partes t, m. La primera parte t corresponde a la hora de inicio de la operación o tarea. La hora de inicio se define en términos de tiempo en relación con su operación padre o dependiente en jerarquía. Así que, si por ejemplo, el tiempo de una tarea es 0, esto significa que comienza inmediatamente después de su tarea padre. Para inicializar estos valores, se toma un valor aleatorio en un rango de tiempo estimado en función del tipo de operación.

La segunda parte de m de gen u operación es más compleja y requiere información adicional sobre la configuración y características del flujo de trabajo. Los valores que describen el modo de operación m_r son un conjunto de variables de cada modo de operación m. La inicialización de cada uno de estos recursos está determinada por la naturaleza del recurso, por lo que es necesario implementar un base de datos de meta-información donde cada recurso o propiedades r incluye información sobre el rango de valores disponibles para cada recurso, además de cuándo, dónde y cómo se aplica. Cada valor del recurso o variable asociado con el modo de tarea y operación m se rellena aleatoriamente dentro de un rango definido y válido para el recurso del proveedor.

5.5.5.3.2. Operador de cruce La aplicación del cruce de dos cromosomas o soluciones produce nuevos descendientes. Los métodos de mezcla y emparejamiento se aplican a ambos padres. El operador de cruce consta de dos partes, una para la parte correspondiente al tiempo de inicio y otra para la del modo de ejecución como se indica en la figura 5.17. Al igual que ocurre el operador de mutación, el cruce de soluciones puede devolver soluciones no factibles, por lo que es necesario aplicar un operador de reparación a parte de las soluciones de la población que violen alguna de las restricciones que

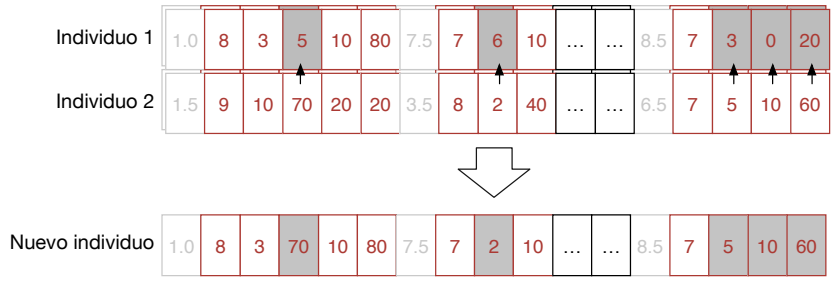

Figura 5.17: Cruce de dos individuos (cromosomas) para producir una nueva solución con parte de las características de ambos padres.

afectan a la factibilidad. Se ha elegido un enfoque de cruce uniforme porque es el más simple y rápido de los métodos de cruce. En la fase de cruce los genes de cruzan de forma completa de modo que no se alteran las partes internas del gen, por ejemplo una parte del modo de ejecución y otra parte no; si esto fuese de ese modo el número de soluciones no factibles después de los cruces sería mucho mayor y por tanto es probable que no se alcancen soluciones adecuadas en un tiempo aceptable.

5.5.5.3.3. Operador de mutación El operador de mutación es el responsable de cambiar el contenido del cromosoma produciendo otro nuevo cromosoma con algunos cambios en ciertos genes u operaciones, activados según una probabilidad de mutación. La mutación es una política para reforzar la diversidad en el conjunto de la población. Introduce cambios en los cromosomas permitiendo explorar la posibilidad de crear y transmitir a la descendencia a través de nuevas características que no existen en ambos padres.

Siguiendo la misma línea que el cruce, la mutación se aplica de diferentes maneras a cada uno de los componentes dentro del gen, basándose en el tipo de elemento t o m. La mutación para el tiempo de inicio t y la mutación para la parte de propiedades (modo de operación) $p_1, p_2, ..., p_n$ se muestra en la figura 5.18. La mutación de los recursos se realiza cambiando los valores del recurso aleatoriamente dentro de un rango definido y válido para el recurso. Una función "gaussiana" se utiliza para la parte de tiempo, sin embargo para la parte del modo de operación se ha usado una operador uniforme.

5.5.5.3.4. Reparación de soluciones Debido a los cambios realizados en las soluciones después de aplicar modificaciones como el cruce y la mu-

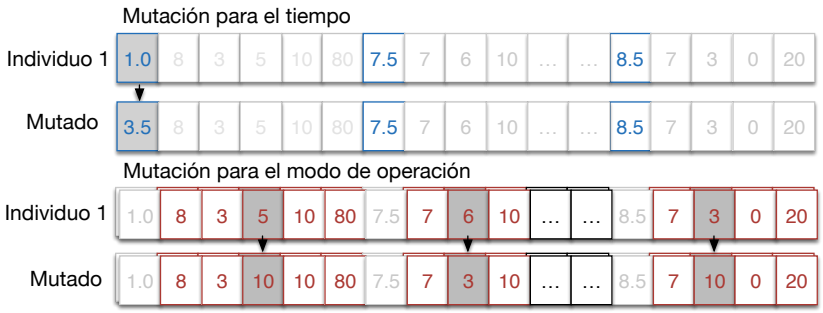

Figura 5.18: Mutación de una solución para la parte correspondiente al modo de operación (propiedades de la operación).

tación, las soluciones obtenidas probablemente serán no factibles. Esto puede ser un gran problema a largo plazo, porque si todas las nuevas soluciones que se generan después de que se aplican los operadores genéticos no son viables, la evolución tendrá problemas para encontrar soluciones buenas y viables a lo largo de generaciones. En los artículos [Salcedo-Sanz, 2009, Mitchell et al., 2003], por ejemplo, el tema de la reparación de soluciones ha sido ampliamente estudiado. La reparación es responsable de corregir las inconsistencias que se producen al cruzar y mutar soluciones. Para la reparación se trabaja sobre todo el cromosoma y se valida para que no haya inconsistencias en los valores asignados para cada componente del modo de ejecución. La reparación se aplica a un porcentaje de la población en cada generación, este parámetro puede ser seleccionado al inicio de la ejecución.

5.5.5.3.5. Selección En un problema multi-objetivo, el operador de selección es una parte fundamental para permitir la selección del conjunto de población que pasará a la siguiente generación. Para la selección se ha usado el algoritmo de selección de ordenamiento no dominado para algoritmos genéticos (NSGA-II) [Deb, 2014, Deb et al., 2002] (ver Figura 5.19).

5.6. Mediación de los servicios de minería de datos

Cada proveedor de Cloud Computing tiene sus propias *API* para el uso de los servicios tanto de minería de datos como de despliegue de infraestructura, siendo diferente el mecanismo para cada uno de los proveedores. Si por un lado contamos con que `dmcc-schema` permite a los proveedores publicar

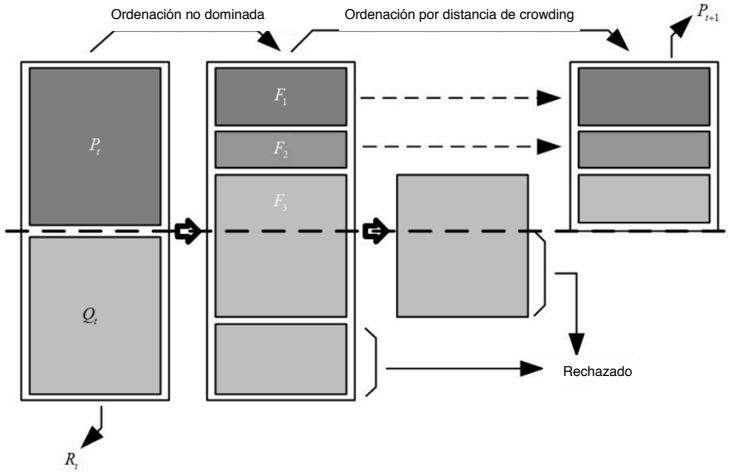

Figura 5.19: Esquema de funcionamiento para la selección de la población basado en ordenación no dominada con NSGA−II.

sus servicios y todas las características de los mismos de un modo flexible y altamente portable, también necesitamos por otro lado determinados elementos que servirán de pegamento entre el esquema homogéneo propuesto de por `dmcc-schema` y las particularidades del proveedor. Estos elementos están determinados por las *APIs* del propio proveedor que son las que el *BrokerMD* debe utilizar para intermediar con los diferentes proveedores de servicios. Estas *APIs* por un lado se centran en algoritmos como servicio y por otro lado en reserva y despliegue de las infraestructuras de computación. De este modo el *BrokerMD* debe conectar la definición de los servicios con la funcionalidad existente en el proveedor. No todos los proveedores han adoptado una política en cuando a la estandarización de sus servicios y sobre todo en los aspectos clave de la mediación entre un servicio de *BrokerMD* y el propio proveedor desde el que proporcionar los servicios, haciendo complicada una tarea tan simple como por ejemplo ejecutar desde una *APIs* un mismo algoritmo a nivel de servicio en dos proveedores distintos.

Cuando el proveedor, por ejemplo, ha asumido el uso de `dmcc-schema`, la mediación de los servicios es directa, pues no sería necesario utilizar una capa adicional que haga de intermediador entre la especificación y el uso/ejecución de las funciones/APIs del servicio del proveedor. Este proceso de mediación consiste en una serie de funcionalidades que debe ofrecer el proveedor para mapear cada entidad del esquema `dmcc-schema` en servicios o funciones que puede desarrollar. Los componentes de la mediación para el *BrokerMD* con un mayor peso corresponden con los siguientes:

- Servicios de autenticación. Es el encargado de dar acceso a todos los servicios del proveedor desde un único punto de entrada. Con una autenticación centralizada es posible que todos los servicios puedan compartir credenciales.
- Servicios de almacenamiento. La transferencia de datos tanto dentro del proveedor como hacia o desde otro proveedor debe ser gestionada, de modo que los datos puedan moverse entre diferentes proveedores.
- Minería de datos como servicio. Proveedores de Cloud Computing como *Amazon* o *Microsoft Azure* ofrecen servicios exclusivos de minería de datos, tales como algoritmos u operaciones como servicios.
- Reserva y despliegue de servicios de computación. Todos los proveedores de Cloud Computing tienen la capacidad de utilizar servicios de computación para la experimentación en minería de datos.
- Explotación de APIs. Operaciones como la explotación de los modelos que han sido generados a partir de algoritmos y datos, con el fin de realizar por ejemplo predicción de datos, por ejemplo.

5.7. Composición y ejecución de servicios de minería de datos

Otros elementos a considerar dentro de las capacidades que incluye el *BrokerMD*, es la composición y la ejecución de los servicios finales para los usuarios. Cuando un usuario envía o compone una flujo de trabajo al *Broker*, este tiene que lidiar con todos los elementos relacionados con la composición de los servicios desde diferentes proveedores, puesto que almacena la información y detalles de los mismos. Además de la composición, una vez compuestos todos los servicios, se lleva a cabo la ejecución del flujo de trabajo propuesto por la selección realizada por parte de los algoritmos evolutivos, según los criterios del usuario en referencia a los objetivos sobre los que se quiere decidir.

5.7.1. Composición basada en tecnología semántica

La composición de servicios web, pueden ser construidos de manera local pero tienen como objetivo ajustarse a las restricciones globales de la com-

posición a través del uso de reglas de orquestación. Esta base de reglas está muy implicada en las formas estándar de comunicación y comprensión del dominio.

La composición de los servicios se puede realizar de forma estática y dinámica. Tanto las composiciones estáticas como las dinámicas de los servicios web pueden describirse colectivamente como:

> *"Una composición de servicios web consiste en servicios web orquestados a través de un proceso local, que en sí mismo es potencialmente un servicio. Las composiciones estáticas de servicios web son conocidas en tiempo de diseño y están vinculadas a una composición en tiempo de diseño. Las composiciones de servicios web dinámicos son una o varias composiciones en las que los servicios web no se conocen en el momento del diseño y que se descubren o cuyas propiedades se resuelven en base a un proceso de criterios establecido en el momento del diseño."*

Las composiciones estáticas de servicios web aparecen actualmente como el modo de composición de servicios web más utilizado tanto en la industria como en el mundo académico. Están formados por la identificación manual (es decir, por evaluación humana) de la aplicabilidad de un servicio web a un dominio en particular. Por lo tanto, la composición se limita a los servicios web incluidos en el diseño. Las composiciones estáticas están representadas por rutas conocidas, representaciones de datos conocidas y resultados esperados como parte de un enlace formal y técnico con el servicio web. Las composiciones dinámicas de servicios web constituyen la base para el descubrimiento y la flexibilidad en las invocaciones de servicios web.

Por ejemplo el lenguaje de definición de servicios web del tipo *WSDL* (sintácticos) se esfuerzan por detallar la interfaz técnica y localizar un servicio determinado, pero no identifica qué hace ese servicio, qué función o funciones desempeña para satisfacer la solicitud, ni tampoco sugiere qué nivel de servicio prestará. Gracias a las limitaciones ofrecidas por este tipo de lenguajes, es necesario ir un paso más allá en cuanto a la definición de servicios, de tal modo que se puedan considerar otros aspectos como la descripción del servicio, las funciones, aspectos de contratación, interfaces entre las entidades, o capacidades de descubrimiento. Todas estas características son tenidas en cuenta por los lenguajes semánticos, como ya se ha comentado en secciones anteriores. Un aspecto fundamental para la composición de servicios es la habilidad que tiene la tecnología semántica de definición de

servicios, de ofrecer una especificación muy flexible que responde a prácticamente cualquier tipo de dominio de conocimiento.

El objetivo es poder construir un mecanismo semiautomático de descubrimiento y ejecución de servicios, de modo que la composición de un servicio web incluya el descubrimiento y la invocación dinámica de los mismos. En estos dos apartados como se ha indicado en las secciones anteriores, la definición de los servicios de forma semántica es fundamental para este cometido y es el *BrokerMD* el encargado de orquestar todos los servicios almacenados junto con las propiedades de su proveedor dentro de las definiciones propuestas por `dmcc-schema`.

Por ejemplo al definir un tipo de función o algoritmo como servicio con `dmcc-schema` es posible indicar en la definición del mismo la entrada que posee, como un conjunto de datos o la parametrización que soportará; además también es posible definir cual será la salida del algoritmo, por ejemplo, otro conjunto de datos (si suponemos que la función es una operación de filtrado). El descubrimiento de los atributos y entidades de los servicios que están soportados bajo *BrokerMD* utilizando `dmcc-schema` permite que la composición de los servicios se realice de un modo eficaz y muy compacto, puesto que es posible saber que servicios o funciones concretas pueden ser enlazadas, combinadas o intercambiadas. Esto se debe a la robusta definición de servicios proporcionada por los lenguajes y tecnologías semánticas, y más concretamente gracias a esquema diseñado para el trabajo con minería de datos como servicio.

5.7.2. Ejecución de los servicios

Cuando el usuario ya ha seleccionado como se va a ejecutar el flujo de trabajo con datos que han sido propuestos por el *BrokerMD*, este debe encargarse igualmente de realizar toda la gestión y la ejecución de los servicios y cada una de las operaciones dentro de cada proveedor que se han indicado dentro de la planificación de ejecución propuesta. El *BrokerMD* debe conocer todos los detalles de la mediación de los servicios, tal y como se ha indicado en la sección 5.6 de mediación de servicios. En este tipo de problemas donde el procesamiento de datos se realiza de forma distribuida y no sólo dentro del mismo proveedor de Cloud Computing, también en proveedores diferentes, con la consiguiente problemática que ello implica a la hora de lidiar con las diferencias de las definiciones de los servicios de los proveedores.

Para la intermediación de servicios es necesario introducir una herramien-

ta, en este caso un lenguaje de definición de servicios de minería de datos como `dmcc-schema`, que sirva como primera capa para dotar de uniformidad a los servicios ofrecidos por el proveedor, siendo el *Broker* el que conoce la lógica de gestión de los servicios de cada proveedor; posteriormente es necesario que el proveedor sea capaz de manejar esa definición y por tanto poder ejecutar las operaciones o funciones desde su infraestructura. Esta parte interconecta la definición de servicios con las herramientas y *APIs* que el proveedor tiene para poder llevar a cabo la ejecución de un algoritmo o servicio concreto en el propio proveedor.

Sin embargo no es posible lidiar con las definiciones de servicios en los diferentes proveedores y además con el conocimiento de la ejecución de esos mismo métodos sobre la infraestructura, de modo que lo habitual es que los proveedores de Cloud no implementen una conversión directa de *dmcc-schema* a una implementación completamente funcional y ejecutable, elementos que debe poseer el propio *Broker*.

Por consiguiente se han contemplado dos opciones para la implementación de la funcionalidad y ejecución del esquema de *dmcc-schema* dentro del *BrokerMD* desde la perspectiva multi-proveedor:

- **Ejecución sobre API del proveedor**. Cuando el proveedor de Cloud posee una lista de servicios de minería de datos en el catálogo, tales como algoritmos, funciones, herramientas de entrenamiento, modelado, etc., el traspaso del esquema general que ofrece `dmcc-schema` al modelo de API del proveedor se hace de forma muy flexible y transparente. Con esto lo que se trata de hacer es una correspondencia entre los servicios definidos por el proveedor con `dmcc-schema` y la *API* de minería de datos que se provee de forma genérica para los usuarios de la plataforma.

  Además, también es posible que parte de los servicios no estén implementados como algoritmos o dentro de la *API* del proveedor; debido a la complejidad del trabajo con minería de datos no todas las operaciones que se realizan en un flujo de trabajo pueden ser implementadas como llamadas únicas a funciones que reciben parámetros y producen resultados. Parte de estas operaciones siguen la filosofía indicada en la sección 5.3.

- **Ejecución sobre la plataforma de despliegue OC^{2}DM**. En el capítulo 4 se ha desarrollado una arquitectura de despliegue de servicios de minería de datos llamada OC^{2}DM, con dos componentes fun-

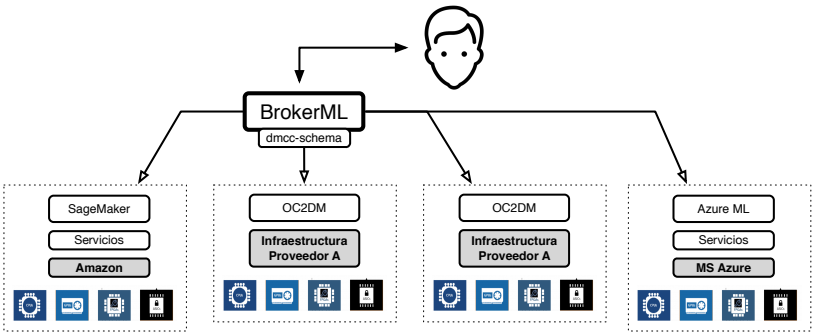

Figura 5.20: Esquema general de abstracción del *BrokerMD* sobre los diferentes servicios de minería de datos en otros proveedores. Desde el *BrokerMD* es posible unificar la interfaz de intermediación para los usuarios que usan el sistema, sin tener que manejar 4 proveedores diferentes. También se puede ver como OC^{2}DM puede actuar como una entidad proveedora de servicios de igual modo que ocurre con proveedores de uso general.

damentales: a) la descripción de servicios de minería de datos en Cloud Computing con el esquema `dmcc-schema` y b) una plataforma para el despliegue funcional de esas definiciones de servicios de minería de datos. De este modo, un usuario o administrador de la plataforma, puede crear una nueva definición de servicio de minería de de datos (por ejemplo, un algoritmo para clasificar series temporales en Big Data), añadirlo al catálogo de servicios y desplegar la funcionalidad del servicio sobre la plataforma OC^{2}DM en Cloud Computing.

La plataforma OC^{2}DM puede ser instalada en cualquier sistema *on-premise* o en cualquier proveedor de servicios de Cloud, como una herramienta más, pero con la ventaja que que permite habilitar servicios de minería de datos en entornos Cloud, facilitando la abstracción de la arquitectura subyacente (ver Figura 5.20). Dadas las características del *BrokerMD*, la plataforma OC^{2}DM supone un complemento perfecto para el *Broker*, ya que por un lado homogeneiza todas las definiciones de los servicios y además sirve de plataforma para la ejecución final de los servicios diseñados, garantizando la portabilidad, composición y descubrimiento de este tipo servicios entre proveedores de Cloud Computing que integren OC^{2}DM.

5.8. Conclusiones

En este capítulo se ha desarrollado una propuesta de *Broker* para servicios de minería de datos en Cloud Computing llamada *BrokerMD*. *BrokerMD* es una propuesta formada por un conjunto de herramientas que permiten la selección, la agregación, la composición y la intermediación de servicios de minería de datos entre diferentes proveedores de Cloud Computing. Con el auge de la minería de datos, el entorno natural dadas las características de flexibilidad, escalabilidad y rendimiento, es Cloud Computing. De este modo el empuje de estos servicios en entornos Cloud está más que justificado. Una herramienta que ofrezca a los usuarios la capacidad de realizar propuestas de selección de servicios de minería de datos que considere diferentes proveedores de Cloud Computing, hace que el proceso de toma de decisiones sea mucho más eficiente en cuanto a la optimización de flujos de trabajo con minería de datos entre proveedores. Con *BrokerMD* es posible tener una herramienta de toma de decisiones completa para seleccionar la mejor planificación de la ejecución posible, por ejemplo.

Para la construcción de *BrokerMD*, ha sido necesario utilizar tecnología semántica para homogeneizar la definición de los servicios de minería de datos de los diferentes proveedores. Para ello se ha usado el vocabulario desarrollado con `dmcc-schema`, gracias al cual el *Broker* tiene la capacidad de almacenar de una forma estandarizada todos los elementos que componen los servicios de minería de datos para cada uno de los proveedores. Con esto se han conseguido capturar todas las propiedades de los servicios de este tipo de modo que los datos sean utilizados por los componentes del *BrokerMD* para la composición e intermediación.

Otro de los elementos fundamentales a la hora de usar *BrokerMD* para la selección de servicios de minería de datos, es la capacidad de abstracción que ofrece al usuario en cuanto a todo el espectro de proveedores existente. El usuario no tienen que lidiar con todas las definiciones diferentes de servicios de los proveedores, sino que desde una única herramienta, abstrae servicios y proveedores de minería de datos. Esto supone una ventaja muy importante ya que es el *BrokerMD* el que se encarga de ofrecerle las propuestas de ejecución de su flujo de trabajo en *inter-cloud*, con datos de costes, tiempo o calidad del servicio, por ejemplo, dejando al usuario la decisión de seleccionar una u otra solución, sin preocuparse que partes u operaciones son ejecutadas en uno u otro proveedor de Cloud Computing.

Finalmente con *BrokerMD* se ha conseguido consolidar el diseño de un

Broker de servicios de minería de datos, confirmando que su aplicación a entornos reales está más que justificada y que en los próximos años, los *Brokers* para Cloud Computing en general serán unas herramientas muy a tener en cuenta como elementos integradores de servicios de proveedores de Cloud Computing con definiciones de servicios no estandarizadas. Los *Brokers* darán al usuario de servicios de minería de datos la capacidad de enfocar el problema en la solución, en los costes o el rendimiento, y no en otros aspectos como la migración de servicios, la incompatibilidad de definiciones de servicios o la complejidad en la selección de los mismos.

Casos de uso

Capítulo 6

Servicios de minería de datos CERN

6.1. Motivación

El periodo de estancia de investigación internacional, se llevó a cabo en la *"Organización Europea para la Investigación en Energía Nuclear"*, CERN, en Ginebra, Suiza. Esta estancia fue motivada por dos aspectos: a) la importancia del CERN como institución y referente en todos los campos relacionados con las tecnologías web, servicios y Cloud Computing y b) la posibilidad de trabajar sobre la problemática de la definición de servicios estandarizados en Cloud Computing para flujos de procesamiento de datos en este tipo de entornos, así como la aplicación práctica de la propuesta de `dmcc-schema`. El CERN es hoy en día una de las instituciones de investigación científica más importantes de Europa y del mundo en muchos aspectos. No sólo es conocido por el principal foco de trabajo que es la Física de partículas, también es uno de los sitios donde la computación se lleva al extremo, siendo un referente global en cuanto a infraestructura, plataformas o servicios. La colaboración con CERN se ha centrado en la definición y descripción de servicios de computación para la experimentación en Física de Altas Energías - «High Energy Physics» (HEP) - dentro de una pequeña etapa de monitorización de la calidad de los datos - «Data Quality Monitoring» (DQM) -. Para ello una propuesta llamada `dqmwf-schema` basada en `dmcc-schema` ha sido desarrollada con el objetivo de proporcionar una definición de servicios para el procesamiento de datos en esta etapa de la experimentación con HEP.

6.2. Introducción

Sería imposible para cualquiera concebir la realización de un experimento de física de partículas hoy en día sin el uso de ordenadores y software. Desde los años 60, los físicos de Alta Energías han sido pioneros en el uso de computadoras para la adquisición, simulación y análisis de datos. Esto no sólo ha acelerado el progreso en este campo, sino que ha impulsado la tecnología informática en general, desde el desarrollo de la *World Wide Web* en el CERN hasta la implantación de recursos masivos distribuidos de la *Worldwide LHC Computing Grid* (WLCG) que soporta toda la experimentación que se lleva a cabo en el *LHC*[1]. Durante muchos años, estos desarrollos y la creciente complejidad del análisis de datos se han visto acompañados de una serie de mejoras tanto en hardware como en software.

Todo experimento de física de partículas debe basarse en nuestros conocimientos actuales de física, lo que significa que la generación de eventos físicos simulados (a través de, por ejemplo, simulaciones de *Monte Carlo*) es esencial. Para gran parte del programa actual de experimentación en HEP, se sirve de la generación de eventos basados en el modelo de la simulación, una tarea relativamente modesta en términos de necesidades requerimientos computacionales. Sin embargo, en experimentos como *LHC*[2] existe una demanda creciente de eventos de orden superior para permitir comparaciones mucho más precisas entre los experimentos reales y las predicciones del *Modelo Estándar*. Estos cálculos son especialmente difíciles tanto en términos de software, como de complejidad matemática, lo que aumenta enormemente la carga computacional para poder realizar tanto las simulaciones como las comparaciones.

Los detectores de partículas son cada vez más grandes y sofisticados, esto significa que los físicos de partículas necesitan una simulación del detector cada vez mejor. Los modelos que describen el paso de partículas a través del detector necesitan ser mejorados en muchas áreas, por ejemplo, para el funcionamiento de alta precisión en general para todos los experimentos realizados en el CERN. Dado que la simulación supone un gran consumo de recursos para los experimentos actuales (más o menos representa más de la mitad de toda la computación realizada), es un área clave que necesita adaptarse a las nuevas arquitecturas de computación, junto al procesamiento y análisis de los datos obtenidos de la detección real que representa un porcen-

[1]Large Hadron Collider `https://home.cern/science/accelerators/large-hadron-collider`

[2]https://home.cern/science/accelerators/large-hadron-collider

taje muy alto en las necesidades de computación de estos tipos de sistemas y plataformas.

Para el análisis de los datos se necesita tener gran capacidad de cálculo y velocidad para comprobar nuevos modelos con los datos existentes en tiempo real o con datos de histórico. Mantener esa agilidad es un gran desafío dado el número de eventos[3] que los físicos tienen que procesar y la necesidad de mantener el volumen de datos y eventos total de datos bajo control.

Para el soporte de todos estos experimentos que se realizan en los centros de investigación de HEP, es necesario aplicar flujos de trabajo de procesamiento de datos muy complejos que requieren una coordinación entre hardware y software a todos los niveles. Este procesamiento de datos, que afecta a diferentes etapas, trabaja desde la extracción de los datos del detector hasta el análisis de datos para, por ejemplo, detectar una partícula u obtener parámetros de estado de la detección, o como en este caso de uso para facilitar la monitorización de la calidad. El modelado de flujo de trabajo para el procesamiento de los datos es diferente según el experimento, módulo o etapa del mismo. Existe un problema importante a la hora de la definición de flujos de trabajo para la experimentación a lo largo de los diferentes experimentos científicos que se desarrollan en HEP. La mayoría de esta experimentación trabaja de forma tradicional a través de lenguajes y plataformas que incluyen herramientas para modelar, de una forma muy estricta y cercana a la infraestructura, como se desplegará un flujo de trabajo. Estás herramientas no consideran ningún aspecto que, por ejemplo, permita definir esos flujos de trabajo en entornos de Cloud Computing de una forma más estandarizada y que de forma adicional provea propiedades como la portabilidad, la flexibilidad o la interoperatibilidad.

El objetivo principal de este caso de uso el demostrar la validez de `dmcc-schema` como base para dar soporte a un vocabulario nuevo llamado `dqmwd-schema` que permita definir servicios de flujo de procesamiento de datos en la etapa de DQM para experimentación en HEP, mediante el uso de tecnología semántica. Además de esto, también es capaz de integrar varios aspectos adicionales de la gestión de servicios por parte de los proveedores, añadiendo capacidades de autenticación, interacción o acuerdos de calidad, entre otros. En las siguientes subsecciones se realizará una introducción al problema y experimento concreto con el que se trabaja en el CERN, y las propiedades y estructura de la etapa de procesamiento que ha sido modelada como servicio.

[3]Un evento, corresponde la detección de una colisión de partículas. Un evento permite a los físicos trabajar con esa colisión para investigar lo que ocurrió o sí se produjeron nuevas partículas.

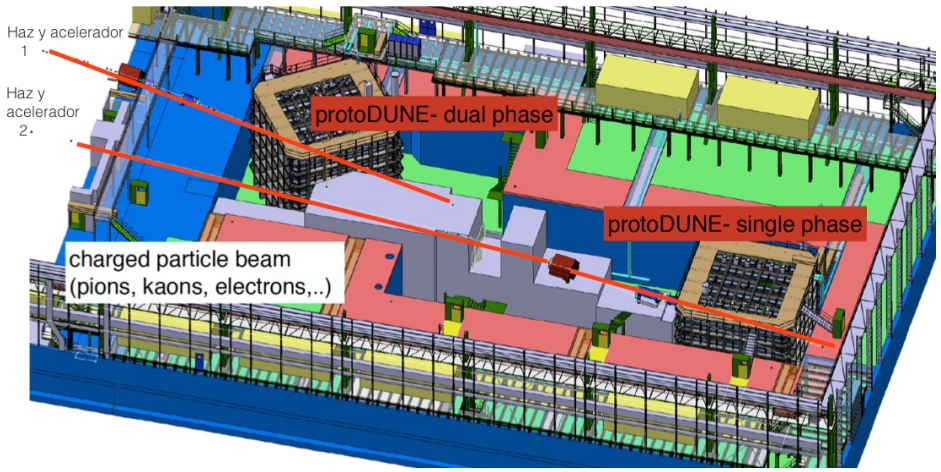

Figura 6.1: Detectores *protoDUNE* (*Single Phase* y *Dual Phase*) y los dos aceleradores de partículas para cada detector. ©CERN.

Posteriormente en la sección 6.3 se realiza una breve descripción de trabajos previos relacionados con los flujos de trabajo en el área de HEP. La sección 6.4 se centra en el objetivo de caso de uso y es realizar una propuesta de modelado semántico para el módulo de DQM en el experimento*protoDUNE*. Finalmente en la sección 6.6 se especifican las conclusiones de la propuesta.

6.2.1. Experimento *protoDUNE* - neutrinos

El trabajo que se ha desarrollado en el CERN se centra en un experimento concreto que se llama *protoDUNE*. Este experimento tiene como objetivo el estudio de los neutrinos. El detector consta de dos grandes detectores de neutrinos llamados *Single Phase* (SP) y *Double Phase* (DP). Como el nombre del experimento indica, *protoDune*, es un experimento formado por dos prototipos del futuro detector *Deep Underground neutrino Experiment* (DUNE), cuya construcción ha comenzado recientemente en el *FermiLab*, Estados Unidos (ver Figura 6.2)[4].

Cada uno de estos detectores (SP y DP) es una cámara de tipo "Time Projection" (*Liquid Argon Time Projection Chamber* LArTPC) llena de argón líquido de un tamaño $10 \times 10 \times 10$ metros, que contiene unas 800 toneladas de argón en estado líquido a -184^{o}C. El detector registra los rastros de partículas que atraviesan el argón, tanto procedentes de los rayos cósmicos

[4]DUNE: `https://www.fnal.gov/pub/science/lbnf-dune/`

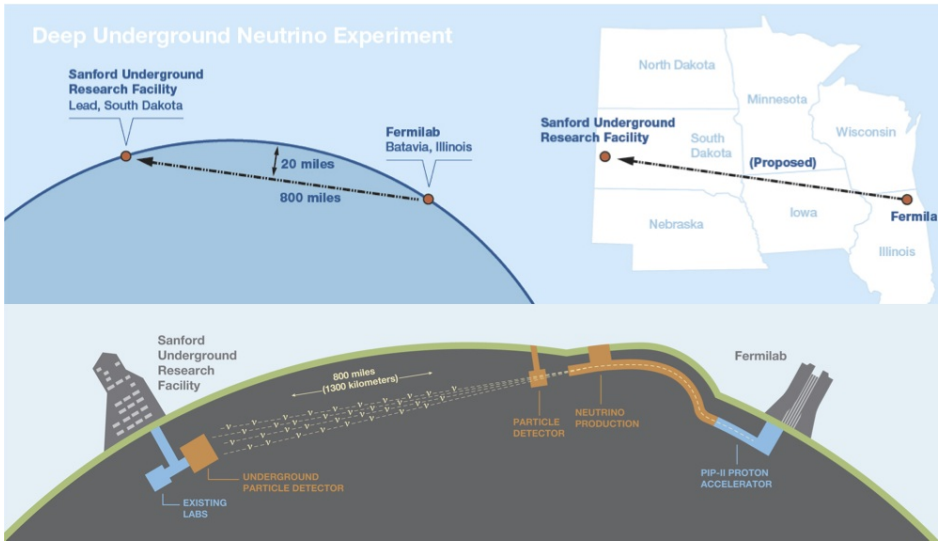

Figura 6.2: El experimento DUNE cruza tres estados en EE.UU. y atraviesa una sección bajo tierra que une dos puntos separados más de 1300 kilómetros, para la detección de neutrinos. Consta de un acelerador de neutrinos en FermiLab (Illinois) y un detector en SandFord (South Dakota). ©FemiLab ©CERN.

como de un haz creado en el acelerador del CERN. En la Figura 6.3 se puede ver ejemplo de la primera detección de la trayectoria de neutrinos que se tuvo en *protoDune*.

Cuando los neutrinos atraviesan los detectores e interaccionan con los núcleos de argón, producen partículas cargadas. Esas partículas dejan rastros de ionización en el líquido, que pueden ser vistos por los sistemas de seguimiento («tracking») capaces de crear imágenes tridimensionales de procesos subatómicos, que de otro modo serían invisibles (ver Figura 6.4).

Este experimento no sólo estudia los neutrinos, sino también su contrapartida de antimateria. Tiene como otros sub-objetivos adicionales buscar diferencias en el comportamiento entre los neutrinos y los anti-neutrinos, lo que podría darnos pistas sobre por qué el universo visible está dominado por la materia y no por anti-materia. También es capaz de caracterizar los neutrinos producidos cuando una estrella colapsa, lo que revelaría la formación de estrellas de neutrones y agujeros negros, e investigará si los protones viven para siempre o eventualmente se desintegran. Además el estudio de los neutrinos ayudaría a cumplir el sueño de "Einstein" de una tener una teoría unificada.

No sólo son necesarios los detectores y los aceleradores, también son im-

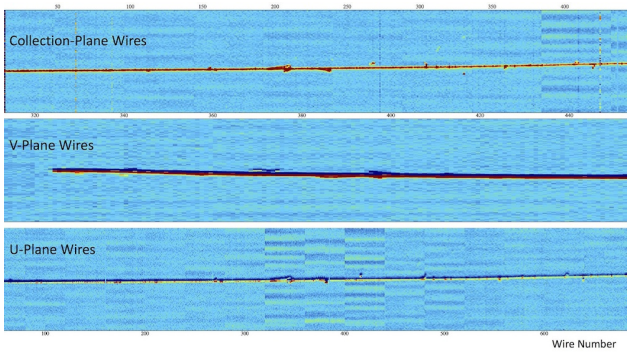

Figura 6.3: Detección de las interacciones de neutrinos con el argón. En la figura se puede ver el rastro que deja la partícula a través del cubo del detector. ©CERN.

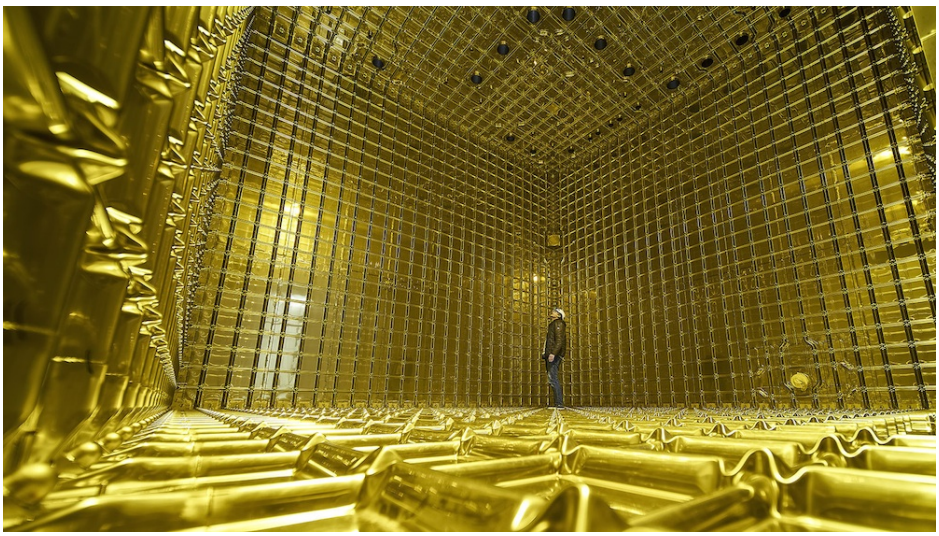

Figura 6.4: Interior de uno de los cubos detectores de neutrinos antes del llenado con argón líquido. ©CERN.

prescindibles otros elementos de los experimentos de HEP en general y en *protoDune* en particular, como los módulos de adquisición de los datos (DAQ) o sistemas para la monitorización de la calidad de los datos, entre muchos otros. En este caso de uso nos centraremos principalmente DQM que es el área de trabajo sobre el que se ha trabajado durante la estancia de investigación. En la siguiente subsección se detallan los componentes iniciales de la extracción y análisis de datos (DAQ y DQM respectivamente).

6.2.2. DAQ y DQM

Para que los detectores puedan capturar los datos de las colisiones o las interacciones entre las partículas se necesita una conjunto de sensores hardware que permiten la adquisición de datos «Data AcQuisition» (DAQ). Gracias a estos dispositivos se miden fenómenos eléctricos o físicos como voltaje, corriente, temperatura, etc. procedentes de los experimentos. Luego estos valores son los que se usarán para realizar un estudio de los datos del experimento y analizar los resultados para buscar o detectar neutrinos, por ejemplo, en el experimento *protoDUNE*. Además dentro de la infraestructura hardware y software que tiene el experimento, existe un módulo (vital en todos los experimentos de HEP), mediante el cual se realiza la monitorización de la calidad (DQM) del proceso de captura de datos y del posterior análisis.

6.2.2.1. DAQ

En la Figura 6.5 se puede observar de forma general todo el flujo de datos que ocurre desde el detector hasta el almacenamiento de los experimentos en las diferentes infraestructuras de computación. El detector, posee varios cientos de miles de dispositivos DAQ que caputan los datos de voltaje, temperatura, etc. del experimento. Estos dispositivos DAQ son extremadamente precisos, además, todos ellos están sincronizados de forma muy estricta, para que la captura de datos sea todo lo fiable que la tecnología actual puede ofrecer. Por tanto, el DAQ canaliza los datos desde los detectores hasta el almacenamiento.

En *protoDUNE* los sistemas DAQ, la gestión de datos y los sistemas DQM están diseñados para ser capaces de hacer frente a volúmenes de datos de la escala de PetaBytes y, al mismo tiempo, realizar toda la selección y movimiento de datos entre las diferentes infraestructuras.

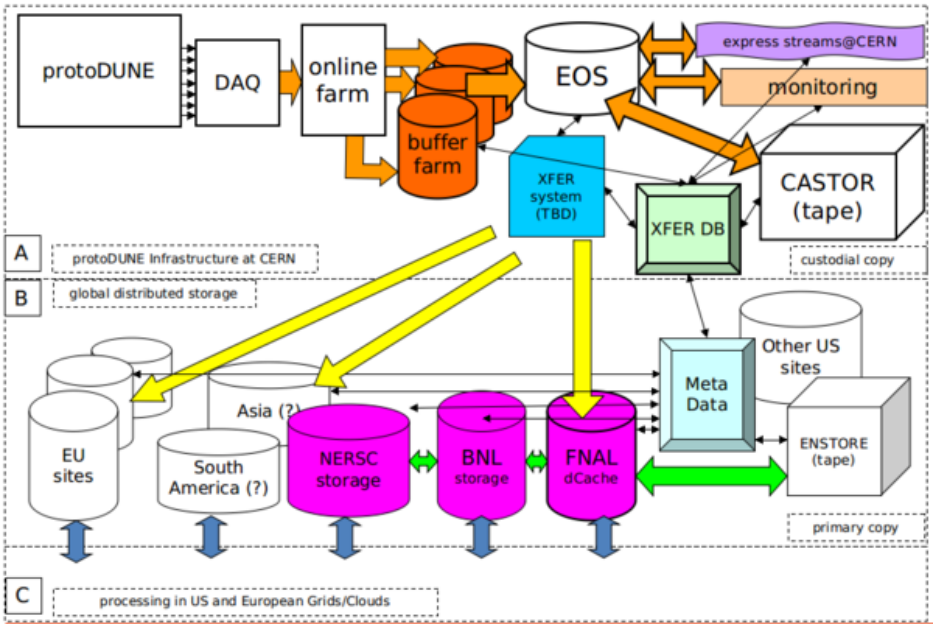

Figura 6.5: Diagrama general conceptual del flujo de datos en el experimento protoDUNE. ©CERN.

Característica	**Valor**
Canales DAQ	15360
Velocidad de datos instantánea	1.44 GB/s
Velocidad media de datos	576 MB/s
Almacén de datos para 3 días	300 TB

Tabla 6.1: Datos del DAQ del detector *protoDUNE* SP.

En la Tabla 6.1 se muestran algunas de las características que los sistemas de DAQ realizan en el experimento. Esta tabla da una idea de las necesidades de computación que requiere el experimento *protoDUNE*. Si nos vamos a otros experimentos mucho más complejos estos números se quedan mucho más atrás, por ejemplo, con el caso del DAQ del experimento más conocido, el LHC, donde uno de sus detectores, llamado ATLAS, está diseñado para capturar hasta 1.700 millones de colisiones por segundo, con un volumen de datos de más de 60 millones de MegaBytes por segundo y posee 100 millones de canales DAQ.

El DAQ tiene algunas capas intermedias de procesos que se ejecutan en computadoras dedicadas. La "capa exterior" del sistema en la que se unifican los datos en un formato adecuado para su escritura en archivos está formada

por los «Event Builders». Los «Event Builders» están alojados en unas pocos nodos con conexión de alta capacidad. Su función es ensamblar los elementos de lectura recibidos lectura a partir de fragmentos de datos obtenidos de la capa interna del DAQ. Los «Event Builders» crean la información necesaria a nivel de archivo y escriben los datos resultantes en el almacenamiento *on-line* que posee el DAQ.

6.2.2.2. DQM

La monitorización de los datos de calidad (DQM) es una parte integral del proceso de captura de datos de los experimentos de HEP. DQM implica un análisis automatizado de los datos de monitorización a través de algoritmos (definidos por el usuario) y la generación de un resumen de los resultados del análisis (a los científicos) mientras se procesan los datos.

Hay dos entornos en DQM. Por un lado, el entorno *on-line*, donde DQM proporciona al administrador la información de la ejecución actual que se puede utilizar para detectar los problemas desde el principio. Por otro lado, durante la reconstrucción *off-line*, DQM realiza un análisis más complejo de los datos capturados desde los sensores, y los resultados se utilizan para evaluar la calidad de los datos reconstruidos.

Con más de 15.000 canales de datos y una toma de datos en el rango de 10^5 Hz es esencial controlar el estado del hardware de DAQ y determinar la calidad de los datos producidos de forma eficiente. Como se ha indicado antes, en el entorno *on-line*, esta información puede alertar al administrador para que tome medidas a fin de evitar la toma de datos erróneos, y por ejemplo, parar la ejecución del experimento en busca de errores hardware en los detectores.

En el entorno *off-line*, se pueden realizar comprobaciones más complejas de los datos para determinar la calidad de la Física. La monitorización de los datos en ambos entornos es muy importante en la puesta en marcha de un experimento cuando no se sabe completamente el rendimiento del detector en condiciones experimentales reales (ruido, interferencias, fallos eléctricos, magnetismo, etc.). Por lo tanto, la disponibilidad de una herramienta para la rápida detección y resolución de problemas relacionados con el detector y el hardware es fundamental.

Además, DQM es una parte integral del modelo de calibración rápida, tal que permite una ejecución de flujo de trabajo especializado antes de que los

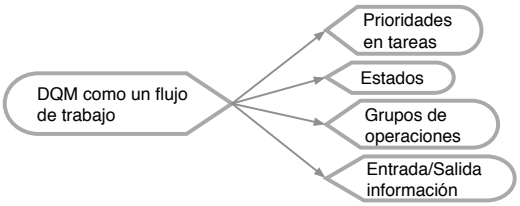

Figura 6.6: DQM como un flujo de trabajo de minería de datos donde se consideran prioridades, estados, operaciones y entrada/salida de información.

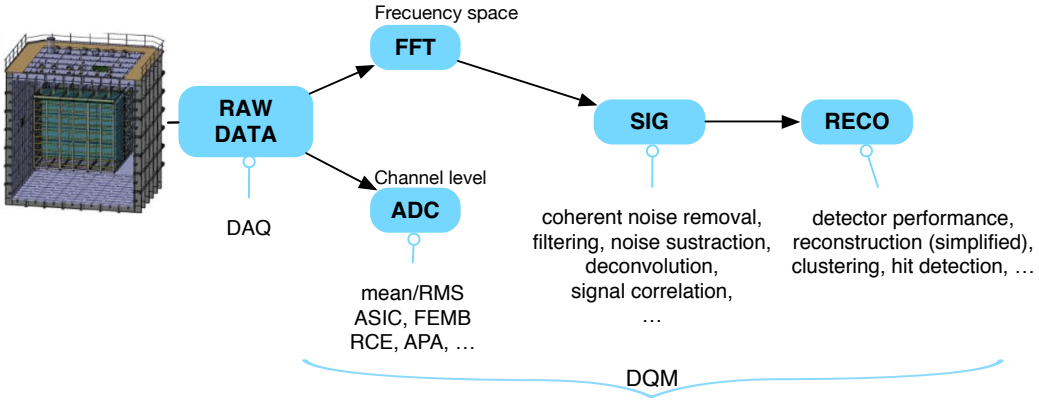

Figura 6.7: Diagrama general de flujo de trabajo en *DQM* con las operaciones principales. Conecta los datos de salida del detector (DAQ) con los resultados de monitorización de la calidad para los experimentos, a través de un flujo de trabajo formado por diferentes etapas.

datos se reconstruyan para calcular y validar el conjunto más actualizado de condiciones y calibraciones utilizadas posteriormente durante la reconstrucción de los eventos. También DQM se utiliza en la simulaciones de *Monte Carlo* para monitorizar la calidad de las muestras que se utilizan finalmente en los análisis de datos para la comparación los resultados reales.

La fase DQM, aplica algoritmos de análisis a varios tipos de datos de monitorización *on-line* (histogramas, valores, resultados, tablas, etc.) de acuerdo con una configuración particular para el experimento. Los resultados de este análisis pueden generar alarmas o avisos de diferente tipo cuando se encuentran desviaciones de los valores esperado. En la Figura 6.6 se indica como DQM puede verse como un flujo de operaciones donde existe prioridad, grupos de operaciones, salidas y entradas reutilizables o estados, entre otras. Un resumen de estos resultados se muestra al administrador y a los usuarios del experimento. En la Figura 6.7 se puede ver parte de las operaciones que se realizan en DQM.

Los algoritmos y operaciones se organizan en etapas y se priorizarán para garantizar la producción de los resultados visuales y de valores de estado en DQM.

6.3. Trabajo relacionado previo

El modelado flujos de trabajo ha sido explorado desde diferentes perspectivas. Al ser un modelado muy cercano a las implementaciones de «frameworks», lenguajes y plataformas, la mayoría de los modelos de definición de este tipo de problemas tradicionalmente han usado estructuras basadas en ficheros de configuración de tipos muy variados donde se especifica cada operación, los parámetros, los algoritmos y los resultados a obtener, además de permitir composición de diferentes operaciones de modo similar a lo que se hace con "pipes" en los sistemas UNIX. Aunque este modelo se sigue usando para parte de la experimentación científica, desde hace varios años con aparición de nuevos experimentos de HEP y el incremento de la complejidad de esta experimentación, nuevas propuestas para el trabajo con flujos de datos han sido desarrolladas.

En [Collaboration et al., 2010] se describe el flujo de datos desde el detector a través de los diversos sistemas informáticos *on-line* y *off-line*, así como los flujos de trabajo utilizados para registrar los datos, para alinear y calibrar el detector y para todos los análisis. Otros como [Wakefield et al., 2008], o [Klimentov et al., 2015] tratan la problemática del flujo de trabajo en HEP como una gestión de operaciones y carga de trabajo para grandes volúmenes de datos.

Otras propuestas como [Gutsche et al., 2017] tratan el tema del procesado de datos y el flujo de trabajo en HEP para plataformas basadas en herramientas de Big Data. En [De et al., 2014, Golubkov et al., 2012] se detalla el gestor de flujo de trabajo de alto nivel que traduce las necesidades de los físicos para el procesamiento y análisis del nivel de producción en flujos de trabajo reales ejecutados en varios GRIDSs para el detector ATLAS[5] del experimento LHC.

Lenguajes específicos de dominio para modelado de flujos de trabajo para experimentos de tipo *LArTPC* (*protoDUNE*) llamado FHiCL[6], permite manejar la representación de configuraciones jerárquicas [Green et al., 2012] y

[5]ATLAS: `https://atlas.cern/`

[6]FCL: `https://cdcvs.fnal.gov/redmine/projects/fhicl`

establecer de un modo simple las dependencias de operaciones. En [Potekhin, 2017] una definición de trabajos de computación para DQM ha sido desarrolla, permitiendo agrupar los trabajos y operaciones, y trata las dependencias como grafos, permitiendo a usuarios crear su propios flujos de trabajo mediante GraphML[Brandes et al., 2001].

La gran mayoría de estas propuestas tratan el problema de la definición de flujos de procesamiento desde una perspectiva sintáctica, de modo que están limitadas en muchos aspectos, en comparación con riqueza de la propuesta semánticas que hacemos en este caso de uso. Son múltiples las propuestas semánticas que abordan el problema de la definición de servicios de flujos de trabajo y operaciones. El estudio relativo a la definición de este tipo de problemas se ha descrito ampliamente en la sección 3.2.

6.4. Modelado se servicios para HEP y DQM

La mayor parte del trabajo de computación que se realiza en CERN proviene de dos fuentes: las simulaciones de los experimentos y, el procesamiento y análisis de los datos de los detectores. Este tipo de procesamiento se realiza desde plataformas *ad-hoc* para computación de datos procedentes de experimentación con física de partículas. Las plataformas de análisis y procesamiento de datos están diseñadas para ser altamente paralelizables y totalmente distribuidas, puesto que el volumen de datos que es necesario que trabajen es extremadamente elevado y requiere de una gran capacidad de proceso.

Cuando se realiza una experimentación tanto desde la fase de extracción de datos hasta la fase de pre-procesado o pos-procesado, el trabajo con estos flujos de operaciones, se realiza con aplicaciones parametrizables y diferentes «frameworks» que permiten ser integrados en *C/C++, Python o Shell-Script*, entre otros. Estos paquetes de aplicaciones ($ROOT$[7]) y funciones albergan bibliotecas de operaciones para todo el espectro de procesamiento de datos y el uso algoritmos dentro del dominio de la física de partículas, Matemáticas o Estadística, entre muchos otros campos. Estas operaciones y algoritmos son ejecutados como trabajos en plataformas de tipo clúster y GRID.

La experimentación con un flujos de trabajo en HEP para el procesado de datos en DQM en concreto, está formado por un conjunto de operaciones que reciben parámetros de entrada, obtienen unas salidas, y estas son admitidas

[7]Data Analysis Framework: `https://root.cern.ch/`

por otras funciones, y así sucesivamente. En la Figura 6.7 se puede ver un esquema genérico del modo de operar a través de flujo de procesado con "pipes" en DQM.

Este tipo de flujos de trabajo están muy acoplados a la infraestructura y sobre todo a las herramientas que se utilizan para la realización de la computación, de modo que no hay mucha flexibilidad para trasladar parcial o totalmente los flujos de trabajo a otras arquitecturas, plataformas u otros entornos (como podría ser, por ejemplo, Cloud Computing) de una forma sencilla.

6.4.1. Operaciones en el flujo de trabajo con DQM

El procesamiento de los datos en DQM está formado por una serie de operaciones que componen un flujo de trabajo completo. Este flujo de trabajo tiene como objetivo producir una serie de resultados y realizar varias actuaciones:

- Resumen y gráficos de datos a nivel de ADC[8], por ejemplo, valores medios/RMS[9] a nivel de canal y estadísticas sobre varios grupos de medidas.
- Resumen de los datos a nivel de ADC en el espacio de frecuencias (FFT*FFT. Fast Fourier Transform*) sobre el canal. Esto proporciona en gran medida medidas del ruido y su evolución.
- Resumen de los datos después del procesamiento de la señal con FFT.
- Mitigación de anomalías.
- Eliminación de los ruidos (importante para la captura más precisa) sustracción de los ruidos y operaciones de filtrado.
- Operaciones de de-convolución.
- Cálculo de correlaciones de señales para diagnóstico.
- Visualización de histogramas, gráficos o una visualización básica de eventos 2D antes y después del procesamiento de la señal.

El flujo de trabajo a alto nivel se puede ver en la siguiente Figura 6.8. En la figura se distinguen las siguiente etapas de DQM:

[8]ADC. Conversor Analógico Digital
[9]RMS: Error cuadrático medio

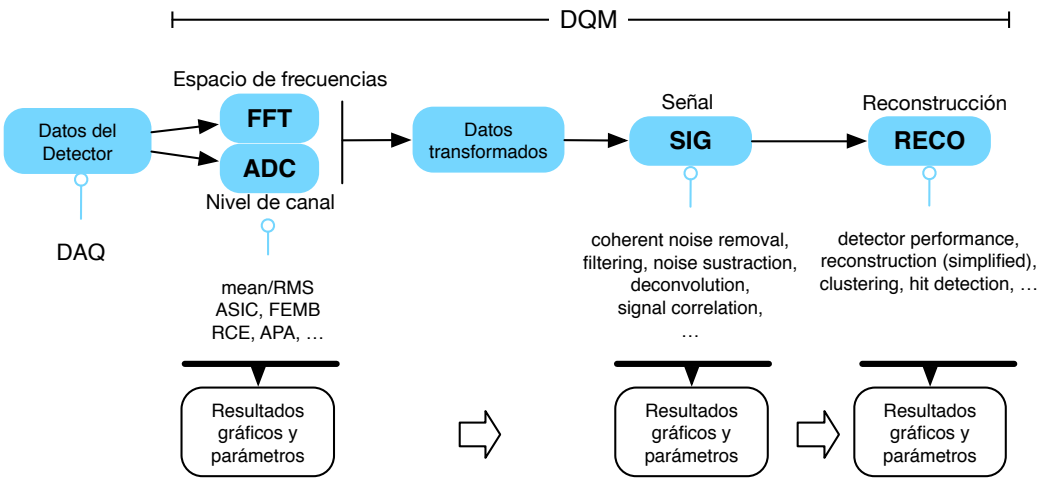

Figura 6.8: Componentes del procesado de datos con DQM, desde la entrada de datos en modo «raw» hasta la obtención de diferentes resultados de tipo gráfico, valores o parámetros, entre otros.

- ADC/FFT. Corresponde con el conjunto de operaciones que se realizan a nivel de conversores analógicos-digitales, y otras operaciones como FFT. Desde esta etapa se realizan operaciones como medias, varianzas, cálculos de errores, etc, produciendo dos tipos de resultados: gráficos y datos transformados (listos para ser utilizados por las siguientes etapas).

- SIG. Esta etapa se corresponde con todas las operaciones que se realizan de las señales obtenidas en la fase de transformación de los datos procedentes de los ADCs. Operaciones como la eliminación coherente del ruido, filtrado, sustracción de ruido, de-convolución, correlación de señales, etc. son ejecutadas en esta etapa. Al igual que en la etapa anterior, se obtienen dos resultados, por un lado elementos de visualización gráfica, y por otro lado resultados en forma de datos que se usarán en la ultima etapa de reconstrucción del evento.

- RECO. Esta es la última etapa en DQM, en la que se realiza la reconstrucción de los eventos y en ella se realizan múltiples operaciones de alto coste computacional como, operaciones de rendimiento de reconstrucción, *clustering*, detección de impactos (partículas), seguimiento de partículas (*tracking*), entre muchos otros. También genera gráficos y resultados en forma de parámetros que son utilizados por los expertos en el experimento para tomar decisiones sobre los eventos que se están produciendo.

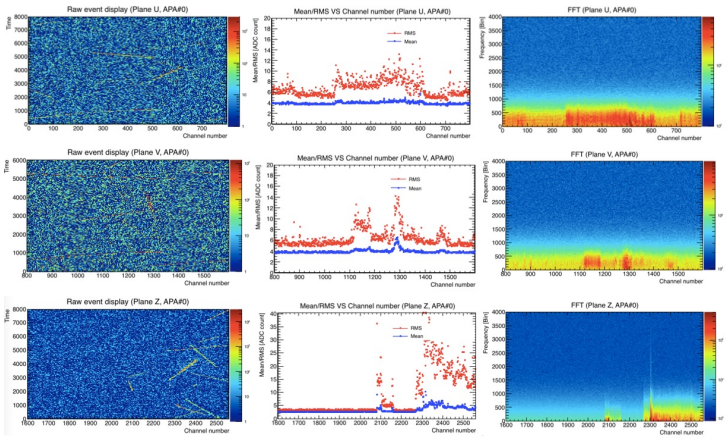

Figura 6.9: Ejemplo de parte de los resultados que se obtienen en las fases de RECO, SIG y ADC/FFT.

Actualmente en CERN este flujo de operaciones se diseña desde herramientas simples que ejecutan cada una de las operaciones tomando entradas y generando salidas hacia otras operaciones que toman esos resultados y a su vez generan diferentes tipos de datos de forma continua, hasta conformar un flujo de procesamiento de operaciones completos para la etapa de DQM. Este tipo de herramientas están bajo el soporte de diferentes *frameworks* y entornos como *ROOT*[Brun and Rademakers, 1997] o *LArSoft*[Church, 2013], entre otros, que proveen de parte de los algoritmos y funciones para desarrollar un flujo de datos para DQM (o para otras etapas o módulos de trabajo de experimentación en HEP), el resto de algoritmos pueden ser integrados posteriormente como parte de las especificaciones concretas del experimento. Cada una de estas herramientas posee definiciones de algoritmos y funciones (de *clustering*, de clasificación, de *tracking*, etc.) con un formato específico, una parametrización y una estructura lógica de entrada y salida. Este tipo de formato de representación de flujos de trabajo no es muy flexible, ya que está bastante anclado a una especificación mediante ficheros de configuración y parametrización. Esto reduce la capacidad de exportar un flujo de trabajo de DQM a otros entornos o experimentos de una forma estandarizada, considerando aspectos claves como la capacidad de migración y la reproducibilidad de flujos de trabajo, además de la composición de operaciones para formar flujos de procesamiento generales en los experimentos y etapas de HEP.

La idea de esta parte del trabajo ha sido mejorar la definición de los flujos de trabajo con una propuesta de descripción y definición de servicios de minería de datos aplicado a problemas en HEP y más concretamente para la etapa de procesamiento DQM. Para este modelado de servicios se

utilizará como base el esquema `dmcc-schema` desarrollado en el capítulo 3, que junto con las directrices de *Linked Data*, permite añadir elementos para el procesamiento de algoritmos y operaciones como servicios en DQM.

6.4.2. Transformación semántica de los flujos de trabajo FCL

El modo tradicional de crear un flujo de procesamiento para la etapa de DQM está definido en ficheros FCL[Green et al., 2012] que contienen cada una de las operaciones a realizar junto con su parametrización, dependencias, prioridad y jerarquía. La dificultad para componer este tipo de flujos de trabajo se incrementa cuando la etapa de DQM requiere más validaciones o comprobaciones o se quiere obtener un mayor volumen de información sobre la calidad de los eventos que se están produciendo en el experimento. Las definiciones en FLC (o FHiCL) están en un lenguaje específico para describir documentos de configuración jerárquicos, permitiendo grupos anidados, precondiciones, además de referencias para reutilización de componentes. Este lenguaje fue creado como resultado de las aportaciones de las comunidades HEP y puede ser integrado en *Python* y *C++*.

La definición de los servicios de procesamiento de los flujos de datos con DQM se ha realizado teniendo en cuenta la base que ofrece el lenguaje FCL para trabajar con las operaciones del experimento *protoDUNE*. El modelado para la experimentación en DQM consta de los siguientes elementos generales:

- *Service*: este bloque contiene configuraciones para servicios específicos del experimento.
- *Input*: este bloque indica al flujo de trabajo qué tipo de fuente de datos se espera (este es de tipo "EmptyEvent" en el caso de la generación de *Monte Carlo*, "RootInput" en el caso de cualquier cosa después de un generador o reconstrucción de *Monte Carlo*, e incluye otros módulos específicos para los datos del detector), el nombre de archivo para la fuente de entrada, y cuántos eventos (el número) se va a procesar.
- *Output*: este bloque le dice al trabajo qué tipo de salida tiene que hacer, es decir, "RootOutput", y cuál debe ser el nombre del archivo de salida. Es posible definir más de un archivo de salida si se desea ejecutar un trabajo que es capaz de producir diferentes archivos de salida basados en cuanto, por ejemplo, criterios de filtrado.

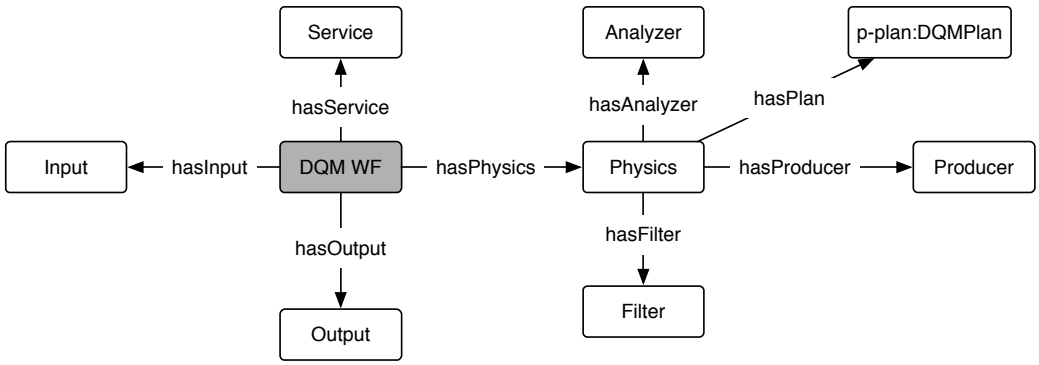

Figura 6.10: Diagrama de componentes del modelo semántico para la definición de servicios de DQM.

- *Física*: este bloque es donde se configuran todos los módulos de productor (*producer*), analizador (*analyzer*) y filtros (*filters*).

Todos estos componentes pueden verse en la Figura 6.10, donde muestra el diagrama correspondiente al modelado semántico general de un servicio de flujo de trabajo para operaciones con DQM. Este esquema se ha llamado `dqmwf-schema`. El esquema considera todos los elementos indicados en la lista anterior y añade un componente adicional que permite realizar planes de ejecución (`p-plan:DQMPlan`) sobre los elementos definidos en el esquema.

Ahora pasamos a definir cada uno de estos elementos que aparecen en el diagrama de modelado semántico del servicio para DQM.

- *DQM WF*, es la entidad primaria que alberga las propiedades, y datos relativos al experimento. Es la clase central del modelado `dqmwf-schema`. Para su instanciación por ejemplo usaríamos `DQMprotoDUNE_exp1 a:DQMWF`, donde indicamos que `DQMprotoDUNE_exp1` es el identificador del flujo de trabajo que se va a crear. Un ejemplo de la definición de todos los elementos del primer nivel que cuelgan de `DQMWF` sería el que se indica en el listado de código 6.1.

```
1 <DQMprotoDUNE_exp1> a:DQMWF;
2   DQMWF:hasInput DataSource1;
3   DQMWF:hasOutput Output1;
4   DQMWF:hasService ServiceData1;
5   DQMWF:hasPhysics Phy_1;
6 .
```

Listado 6.1: .

- *Salida de datos de la experimentación* `:Output`. Para que `DQMprotoDUNE_exp1` tenga una o varias salidas, utilizamos la propiedad `DQMWF:hasOutput` e indicamos los elementos de salida que tiene, por ejemplo en el listado 6.1 indicamos sólo uno `DQMWF:hasOutput Output1;`
- *Entrada de datos* `:Input`. La entrada de datos se define del mismo que que ocurre con las salidas. En este caso usamos `DQMWF:hasInput DataSource1;` para indicar que sólo una entrada de datos será la que tendrá en cuenta en el procesamiento de los datos del flujo de trabajo del experimento.
- *Elementos de análisis* (`:Physics`). Los elementos de análisis de los eventos se han denominado `Physics`. Para instanciar que un modelado tiene uno o varios análisis se usará `DQMWF:hasPhysics Phy_1;`. En este caso se ha indicado sólo un estudio del tipo `Physics` y que ha identificado como `Phy_1`. El modelado de las operaciones físicas, se puede subdividir en tres:

  - Operaciones de análisis (`Analyzer`). Para indicar que existe una o varias operaciones de análisis de datos, se usará: `DQMWF:hasAnalysis An1, An2;`. En este ejemplo se definen dos análisis `An1` y `An2`.
  - Operaciones de filtrado (`Filter`). Con `:hasFilter` es posible indicar los filtros que se aplicarán a los datos para el procesado de los mismos.
  - Productores (`Producer`). Los productores, son aquellas operaciones que se encargan de aplicar operaciones en busca de elementos de detección. Estos elementos son algoritmos, funciones, operaciones complejas, etc., que tienen como objetivo detectar el paso de partículas, trazar la trayectoria, agrupar (*clustering*) áreas del detector, y muchas otras más. `:hasProducer P1;` permite indicar desde el módulo de `Physics` que es posible incluir en la cadena de trabajo operaciones de este tipo. Este grupo de productores está formado por un número elevado de funciones de diferente tipo y que abarcan prácticamente todos los tipos de experimentos, tanto para aceleradores como para astro-partículas

El diagrama de la Figura 6.10 muestra todos los componentes del modelado general, sin embargo, la complejidad del diagrama crece de manera considerable cuando incluimos, por ejemplo, parte de las operaciones relativas a la definición `Physics` que son descritas. La Figura 6.11 incluye parte del modelo de descripción de experimentos de DQM con cierto grado de de-

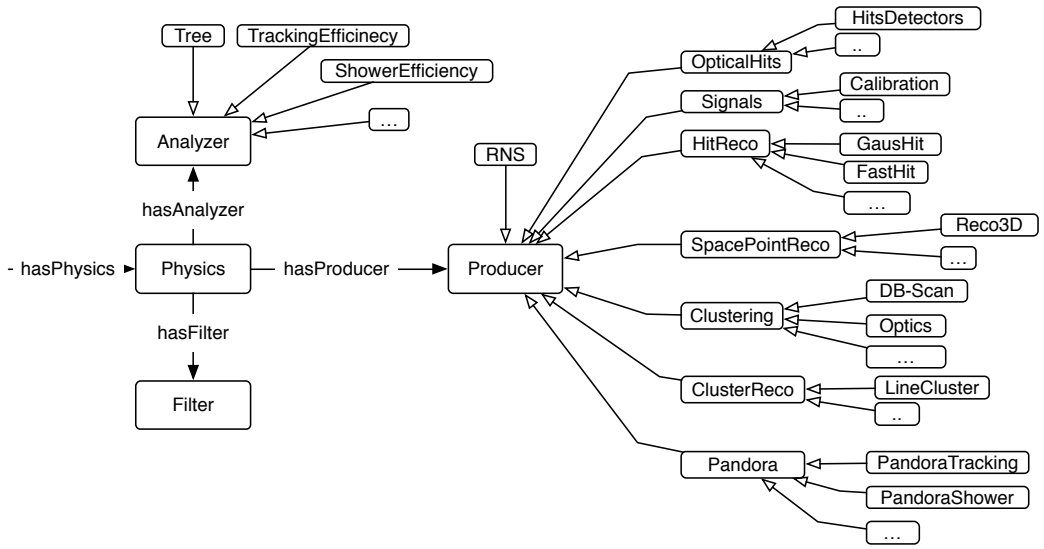

Figura 6.11: Diagrama del modelado correspondiente a las operaciones dentro la clase `Physics`: `Analysis`, `Filter` y `Producer`

talle para la parte correspondiente a los algoritmos de procesamiento de los eventos.

El modelado de servicios realizado con `dqmwf-schema` no considera otros aspectos relacionados con Cloud Computing. De hecho estos aspectos ya han sido estudiados en el capítulo 3, donde con el modelo de definición de servicios de minería de datos proporcionado por el esquema `dmcc-schema` es posible no sólo realizar la integración con este tipo de problemas, sino que también es posible incluir dentro de la descripción del servicio todos elementos de la gestión de Cloud Computing que podría requerir cualquier proveedor. La Figura 6.12 muestra la inclusión de varias partes del vocabulario que `dmcc-schema` provee, dentro de la propuesta para el modelado de trabajo en DQM. El esquema de la Figura 6.12 añade varios elementos adicionales que permiten completar el flujo de trabajo con toda la gestión necesaria de un servicio en Cloud. Todos estos elementos han sido descritos en el capítulo 3.

6.5. Ventajas del modelado semántico en HEP

En esta sección expondremos algunas de las ventajas más destacadas del modelado de servicios semánticos para la experimentación flujos de trabajo en DQM y en general con la experimentación en HEP.

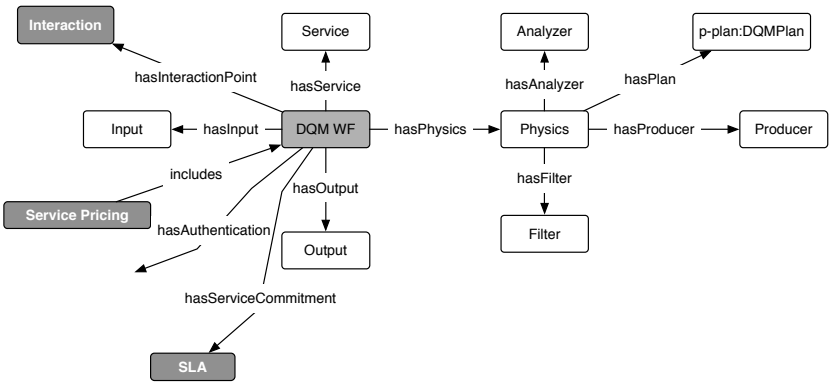

Figura 6.12: Modelado del servicio de `dqmwf-schema` con `dmcc-schema` que permite la inclusión en el flujo de trabajo de otros aspectos relacionados con la gestión del servicio, tales como la autenticación, las interfaces que se exponen, los costes (si se requieren) o elementos de control de la calidad del servicio (SLA), entre otros.

- Estandarización de la definición de la experimentación. Uno de los problemas más importantes a la hora de realizar este tipo de flujos de trabajo con procesamiento de datos, es que cada uno de los enfoques usados por «frameworks» y plataformas para HEP son diferentes y poseen formatos para la composición de experimentos que la mayoría de veces hay que adaptar para que funcionen de un entorno de experimentación a otro.

  Con la propuesta `dqmwf-schema` se ofrece un modelado uniforme para la composición directa de servicios de flujos de procesamiento de datos para experimentación en HEP, y más concretamente en la fase de análisis de la calidad en DQM. Este tipo de estandarización permite capturar gran parte de los detalles de los flujos de trabajo en la experimentación en HEP, de modo que cualquier científico podría crear una definición de servicio que será portable a otros experimentos, sin tener que adaptar la definición. Esto mejoraría por ejemplo la abstracción sobre la arquitectura en la que se desarrolla la experimentación, dando mayor importancia al modelado del servicio que se tiene que ejecutar que a otros elementos relativos a las plataformas de computación.

  Aspectos como la portabilidad, la flexibilidad y reproducibilidad son claves para la estandarización de este tipo de servicios. Gracias a las herramientas semánticas que despliega `dqmwf-schema` es posible realizar este tipo de modelados.

- Modelo basado en servicios. Al igual que ocurre con Cloud Computing, el modelo tradicional de computación está cambiando y está moviéndo-

se a entornos basados en servicios en Cloud Computing. De este modo, la idea que subyace de `dqmwf-schema` es ofrecer una definición y descripción de servicios que pueda ser integrada en plataformas y entornos de Cloud Computing. El modelo de definición semántica de servicios es considerado el más apropiado para este tipo de descripción de servicios dado que no sólo se limita a aspectos funcionales, también es capaz de representar otros elementos no funcionales.

El modelo de servicios de HEP planteado en este caso de uso, permite que cada una de las operaciones del flujo de trabajo tenga las propiedades de un servicio que está preparado para ser utilizado en entornos de Cloud Computing. La ventaja de usar servicios descritos con tecnología semántica es que permite el descubrimiento, la composición de servicios efectiva y un catalogo completo para la experimentación con HEP.

- Cloud Computing. La descripción de este tipo de servicios, al igual que se detalló en el capítulo 3, permite que se puedan incluir todas las propiedades que un entorno de Cloud Computing requiere, es decir, por un lado la definición de los servicios de experimentación para HEP y por otro lado la gestión de los servicios del propio proveedor, tales como autenticación del servicio de experimentación, puntos de interacción de los servicios, etc.

- Soporte para servicios de *Brokering*. La definición de los servicios utilizando tecnología semántica permite que herramientas como los *Brokers* de servicios en Cloud puedan albergar un catálogo, realizar una selección e intermediar servicios desde diferentes proveedores, con un único punto de gestión que realiza todas esas operaciones por los usuarios. Este tipo de herramientas permite abstraer los detalles de cada proveedor o también de cada plataforma de computación que exponga sus servicios, por ejemplo, utilizando `dqmwf-schema`.

  La utilización de un *Broker* en este contexto, permitiría una mejora importante en cuanto a la gestión de los recursos y plataformas de computación, ofreciendo una mejor planificación, por ejemplo, de la ejecución de flujos de trabajo. Esto implicaría una mejora del rendimiento de estas etapas de procesamiento en HEP y redundaría en una optimización general del procesado de de datos. Actualmente este procesamiento se hace sobre entornos clásicos (HT-Condor, Slurm), que actúan como gestores de trabajos y recursos. Un *Broker*, por ejemplo, no está limitado a ese tipo de ambientes, también es capaz de trabajar con múltiples entornos, plataformas o servicios, tratando de intermediar todos los recursos.

En el capítulo 5 se detallan todos los conceptos fundamentales de un *Broker* para servicios en Cloud Computing.

6.6. Conclusiones

La propuesta desarrollada permite la definición de flujos de trabajo para experimentación en HEP y más en concreto permite modelar servicios de la etapa DQM de procesamiento de datos. Se ha utilizado tecnología semántica para la definición y descripción de cada uno de los componentes de la experimentación incluyendo los aspectos específicos de este tipo de servicios científicos tales como las entradas, las salidas, los análisis, los productores de operaciones y el filtrado. Además de ello se ha considerado el flujo de trabajo asociado a cada una de las operaciones de modo que es posible componer como se desarrollará todo el trabajo de monitorización de la calidad, para describir de un modo muy flexible y estandarizado toda esa información.

Gracias al modelado semántico utilizando en el esquema `dqmwf-schema` es posible abstraer elementos como la plataforma, el lenguaje, o los detalles de la implementación de las operaciones por ejemplo, siendo una herramienta fundamental para la estandarización de este tipo de servicios de HEP con una ventaja importante como la portabilidad entre plataformas y entornos de Cloud Computing. Este aspecto, por ejemplo, sería muy interesante para las diferentes comunidades científicas que trabajan en HEP o astro-partículas, ya que permitiría la migración e intercambio de definiciones de flujos de trabajo para experimentación, con lo que se aseguraría la reproducibilidad de los flujos de trabajo de este tipo, si tener que preocuparse de la infraestructura que soportará la computación de las operaciones.

Capítulo 7

Modelado de series temporales en Cloud Computing

Motivación

El modelado de series temporales es la base para estudios del comportamiento de procesos que ocurren durante a lo largo del tiempo. Las aplicaciones de los modelos de series temporales son variadas, tales como ventas, meteorología, predicción de precios, etc. En las decisiones que implican un factor de conocimiento a futuro, los modelos ofrecidos por las series temporales son unos de los métodos más eficaces en cuanto a predicción. Para realizar este tipo de análisis, prácticamente la totalidad de aplicaciones para el modelado utilizan lenguajes de programación (*Python, R, MatLab o Scala*), o entornos de modelado de flujo de trabajo para minería de datos (*WEKA* o *KNIME*, por ejemplo) que integran funciones y algoritmos específicos para el trabajo con series temporales. Este tipo de herramientas trabajan sobre plataformas *on-premise*, de modo que no tiene en cuenta su despliegue o su implantación en Cloud Computing. La herramientas de modelado de series temporales tradicionales no han sido portadas a modelos basados en Cloud Computing, suponiendo un nicho muy interesante para cubrir con este tipo de herramientas.

La idea principal de este caso de uso se centra en el estudio de la necesidad de poder definir servicios de modelado y análisis de series temporales dentro de plataformas de Cloud Computing. Con ello se persigue disponer de servicios de análisis y modelado de series temporales que sean escalables, además

de que se puedan combinar servicios de distintos proveedores o incluso migrar entre ellos. El flujo de trabajo con series temporales puede ser considerado como un problema específico dentro de la minería de datos, dada su naturaleza. Esto implica, por tanto, que se requiere un estudio específico para modelado de este tipo de problemas. El estudio de series temporales considera diferentes metodologías de trabajo, análisis y flujos de trabajo que pueden ser modelados como parte del esquema desarrollado con `dmcc-schema`. La motivación principal de este caso de uso es poner en valor por un lado la capacidad de la tecnología semántica para abordar un problema de modelado de series temporales, reutilizando los componentes base del esquema `dmcc-schema`, y por otro lado verificar que es posible crear servicios de flujo de trabajo con series temporales en entornos de Cloud Computing.

Para llevar a cabo la propuesta, se ha diseñado una definición semántica para el modelado completo del flujo de trabajo con series temporales llamado `tswf-schema` que se expone en detalle en las siguientes secciones y gracias a la cual es posible modelar cualquier problema de este tipo como un servicio en Cloud Computing.

7.1. Introducción

La predicción del precio de las acciones en la bolsa, la estimación de la producción y el consumo de productos o predicción de la temperatura y precipitaciones de un lugar, son varias de las actividades de análisis de datos donde prima el estudio de series temporales como componentes fundamentales para el éxito del modelado en predicción. En el momento actual donde la computación en la nube prácticamente se ha integrado de una manera totalmente transparente en nuestra relación con las tecnologías de la información, las actividades relacionadas con el análisis de datos en general, avanza cada vez más dentro del espectro de servicios ofertados por las plataformas de servicios en Cloud Computing.

Las series temporales están formadas por secuencias de datos espaciados en el tiempo; eventos, actividades o dispositivos que generan continuamente información que se marca temporalmente y se almacena para estudio en tiempo real o de modo diferido. De forma creciente en interés, el uso de dispositivos, las actividades de usuarios, vehículos, agricultura, sistemas de control, o los dispositivos en el IoT [Gubbi et al., 2013] entre otros, produce una ingente cantidad de información en los próximos años y un importante

porcentaje de esa información es en forma de series temporales. No sólo en volumen de datos, si no en cantidad de negocio, en concreto según *Ericcsson* en su informe [Ericsson, 2018], el flujo de información que se moverá supondrá en torno a los 69.000 millones de dólares para los próximos años. Estas cifras dan una idea sobrada de la necesidad de procesamiento de este tipo de datos. Las series temporales, necesitarán ser procesados en forma de servicios de computación en la nube siguiendo las características de flexibilidad, escalabilidad y seguridad definidas por el NIST[1] para los servicios de Cloud Computing en general.

Los proveedores de Cloud Computing ofrecen herramientas muy potentes para el análisis y la extracción de conocimiento de los datos. Dentro del área de la minería de datos, estos proveedores de Cloud Computing, no ofrecen apenas funcionalidades para el análisis de series temporales dentro de su catalogo de servicios. Esto supone tener que implementar en cada una de las plataformas de Cloud los servicios, los métodos y algoritmos específicos para el trabajo con series temporales. Además, el modelado de series temporales puede ser una tarea de análisis no lineal y de elevada complejidad que implica un flujo de trabajo muy variado. En el estudio del modelado de series temporales intervienen diferentes metodologías, compendios de aplicación de técnicas de minería de datos, modelos, o distintas evaluaciones de medidas de rendimiento y precisión en los resultados, entre otros.

La experimentación con flujos de trabajo de series temporales cuando es vista como un servicio entre diferentes proveedores de Cloud Computing, conlleva un problema importante en cuanto a la integración de las definiciones no estandarizadas de este tipo de servicios. Cuando se migran servicios de un proveedor a otro, la solución más adecuada sería armonizar la descripción de los servicios para la mayoría o la totalidad de los proveedores. Esto mejoraría la interoperatibilidad de esos servicios entre diferentes proveedores. En el capítulo 3 (`dmcc-schema`) se realiza un estudio completo sobre la estandarización de los servicios de minería de datos para Cloud Computing. Para el flujo de trabajo con series temporales `dmcc-schema` sirve como base para este tipo de modelado.

En este caso de uso real, se propone una definición de servicios para flujos de trabajo con series temporales para entornos de Cloud Computing utilizando tecnología semántica. Para la descripción de los diferentes componentes, se han usado vocabularios y esquemas semánticos ya existentes, completando

[1]Cloud Computing Services: https://csrc.nist.gov/publications/detail/sp/800-145/final

la descripción global de este tipo de definición de servicios. El caso de uso presenta la propuesta `tswf-schema`, para el trabajo con series temporales, que junto con `dmmc-schema`, cubren la definición completa de este tipo de servicios en entornos de Cloud Computing. Además, `tswf-schema` se integra perfectamente dentro de la arquitectura de despliegue OC^{2}DM (ver capítulo 4) transformando la definición de servicios para el modelado de flujos de series temporales en implementaciones funcionales para entornos de Cloud Computing que pueden ser consumidos como servicios.

7.2. Trabajos previos relacionados

El análisis de series temporales es un área muy estudiada en la literatura científica que se ha abordado desde distintas perspectivas y se ha aplicado a múltiples campos del conocimiento. En el modelado de series temporales se busca estudiar los datos de las observaciones pasadas de la serie temporal para crear un modelo que describa las series temporales al completo. Parte importante del trabajo realizado con series temporales se ha desarrollado dentro de temáticas muy variadas tales como negocios, economía, acciones, ciencias ambientales o ingeniería entre otros [Dominici et al., 2002, Aljawarneh et al., 2016, Xu et al., 2014].

El análisis y modelado de series temporales es una tarea compleja que incluye la aplicación de diversas operaciones, técnicas y algoritmos. Este procedimiento puede verse como un flujo de trabajo que tiene como objetivo la resolución de problemas con datos de series temporales [Montgomery et al., 2015].

Las series temporales han sido estudiadas en profundidad y no hay un criterio único que establezca cual es el procedimiento a realizar para el flujo de trabajo en este tipo de problemas. Existen diversas metodologías para abortar el problema del modelado y la aproximación a la resolución de series temporales. La propuesta más ampliamente usada es la metodología *Box-Jenkins* [Chatfield, 2016]. Esta metodología puede ser considerada como un flujo de trabajo no lineal. *Box-Jenkins* es utilizada en el proceso de construcción del modelo *ARIMA* para las series temporales abarcando aspectos como identificación, estimación, test y aplicación de métodos [Makridakis and Hibon, 1997]. Si nos centramos en el modelado, técnicas emparentadas con ARIMA, tales como *AR*, *MA* o *ARMA* [Übeylı and Güler, 2004], son también utilizados como parte del proceso de flujo de trabajo con series temporales. El modelado de series temporales desde una perspectiva no lineal ha sido abordado con el uso de *ANN* [Zhang, 2003], múltiples modelos

lineales, *TAR* o *ARCH* [Engle, 2001]. Otras técnicas de modelado basadas en minería de datos y Machine Learning han sido propuestas incluyendo *Random Forest* [Liaw et al., 2002], *SVM* [Zhang, 2003], *NN*, o las más modernas como *Deep Learning* [Yang et al., 2015]. En el análisis de series temporales se incluyen además, un amplio conjunto de tareas de procesamiento de datos (extracción, transformación y carga - *ETL*) [Hoey and Dassi, 2014].

Para la resolución de este tipo de análisis comúnmente se han usado lenguajes y herramientas de programación, además de algunas plataformas de minería de datos [Jovic et al., 2014]. Estas utilidades incluyen todo lo necesario en cuanto a componentes y funciones para el flujo de trabajo relativo a series temporales. Lenguajes como *R* (con su «Task View»[2] para series temporales) o *Python* (con bibliotecas especificas para series temporales) [Hothorn, 2018] [VanderPlas, 2016], paquetes de software como *SAS* o *MatLab* y entornos de minería de datos como *KNIME* [Berthold, Michael R and Cebron, Nicolas and I *WEKA* [Hall et al., 2009b], ofrecen las herramientas para el trabajo con series temporales. En todas ellas es posible diseñar un flujo de trabajo con series temporales donde se puede realizar la aplicación de operaciones, visualizar los resultados, validar errores y aplicar múltiples algoritmos para el modelado y posterior predicción [Rangra and Bansal, 2014].

La mayor parte de las herramientas tradicionales de análisis de series temporales están diseñadas para trabajar sobre ordenadores personales o estaciones de trabajo y no están preparadas para ser usadas en entornos de computación en Cloud Computing. Aprovechando las capacidades de cómputo de organizaciones y proveedores de servicios, parte de ese conjunto de recursos e infraestructura están siendo destinados a minería de datos como servicios de Cloud Computing [Hashem et al., 2015] bajo demanda. El análisis de series temporales no es una excepción y cada vez más los proveedores de Cloud están incluyendo dentro del catalogo de servicios funciones y algoritmos específicos para el trabajo con series temporales [Chen et al., 2015a]. Actualmente hay una demanda creciente de servicios que permitan desplegar un flujo de trabajo completo sobre un conjunto de datos tales como series temporales [Marozzo et al., 2016, Chen et al., 2014]. Estos flujos de trabajo sobre problemas de minería de datos sobre Cloud Computing son muy interesantes debido a su carácter escalable. Existe una clara necesidad de mover gran parte del procesamiento de datos a plataformas en la nube, abstrayendo la infraestructura de computación subyacente y las necesidades de escalado, ambas aseguradas en el modelo de Cloud Computing [Kranjc et al., 2015, Kranjc et al., 2012].

[2]CRAN Task View: https://cran.r-project.org/web/views/

Uno de los principales problemas de los servicios de Cloud Computing es la falta de una definición estandarizada de estos servicios. Esto ocurre de igual modo con la descripción de flujos de trabajo y experimentación en Cloud Computing para el análisis de series temporales [Dillon et al., 2010, Barry, 2003]. La portabilidad de servicios y la capacidad de abstraer la infraestructura subyacente hace necesario el estudio de este tipo de problemas y su viabilidad sobre Cloud Computing.

El modelado de series temporales para la experimentación es considerado en los trabajos de investigación [Talia, 2013] o [Data et al., 2013], donde se realiza un estudio del problema como un modelado en Cloud Computing de servicios, dando la posibilidad de desplegar parte de la experimentación de un modo completo e integral. Otras propuestas han utilizado tecnologías semánticas (ver capítulo 3), para solventar el modelado en general de minería de datos, siendo unas propuestas efectivas para la definición y descripción de servicios en Cloud Computing. En el caso de uso que se estudia en este capítulo la definición de servicios ha sido abordada desde una perspectiva semántica siguiendo las recomendaciones de *Linked-Data*, de modo que es posible extender las funcionalidades y los dominios de abstracción para una mejor descripción de servicios de minería de datos. Este tipo de propuestas de modelado de servicios permite una conexión mucho más directa con el desarrollo natural de Cloud Computing.

7.3. Modelado de series temporales como servicio para Cloud Computing

El modelado de series temporales es una tarea desafiante que integra diferentes componente, estructuras y algoritmos. En este trabajo se ha hecho una propuesta completa para el modelado de flujos de trabajo con series temporales en Cloud Computing. El esquema propuesto se denomina `tswf-schema` y ha sido diseñado utilizando tecnología semántica, siguiendo las directrices de *Linked Data* para la definición y descripción de conceptos, entidades y relaciones con otros esquemas. Se ha definido un diagrama completo con el que es posible modelar cualquier flujo de trabajo con series temporales, como se puede ver en el apartado 7.4, en el que se desarrollan y modelan varios ejemplos de trabajo con series temporales utilizando `tswf-schema`.

El esquema se divide en varias partes que se corresponden con las principales tareas en el procesamiento de series temporales: pre-procesamiento de

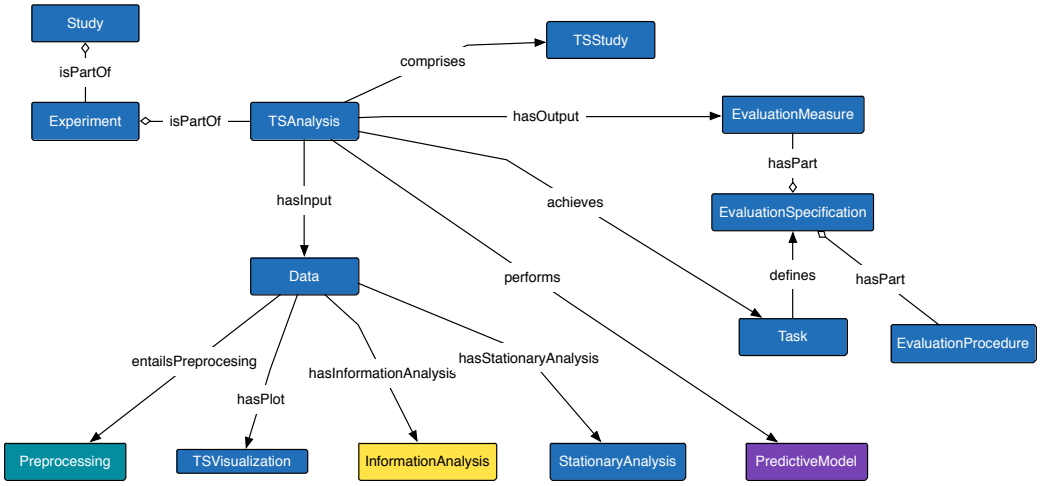

Figura 7.1: Componentes generales del esquema `tswf-schema`.

datos, visualización de datos, funciones para el análisis de información, estacionalidad, selección de modelos predictivos, problemas de aprendizaje, entrada de datos, y medidas de evaluación del rendimiento, entre otras. Cada una de estas partes se desarrolla en detalle a lo largo de esta sección. En la Figura 7.1 se puede ver el diagrama general del modelo de flujo con series temporales que se ha desarrollado. Por razones de espacio, no se ha mostrado el esquema completo, dado su tamaño, por lo que se ha mostrado cada una de las principales clases generales y luego en cada clase ha sido desglosado el contenido detallado de cada uno. Siguiendo la especificación de *Linked Data* (ver sección 2.3.4), se han utilizado vocabularios de otros esquemas, completando y ampliando la definición del esquema (ver sección 3.2). El esquema proporcionado por [Correa Publio, 2017] y [Parra-Royon et al., 2018] ha sido utilizado como base para el flujo de trabajo de experimentación con series temporales, además de otros vocabularios como *SKOS* [Miles and Bechhofer, 2009] o *schema.org*[3], entre otros.

Para la definición del modelado de flujos de trabajo se han tenido en cuenta diferentes trabajos de investigación relacionados con el modelado y análisis de series temporales. Para las clases generales y otras partes del programa, se ha revisado la investigación relacionada con el área, junto con el trabajo real de análisis de series temporales de varias fuentes, tales como [De Gooijer and Kumar, 1992, Harris and Sollis, 2003]. Esto ha permitido extraer gran parte de las operaciones y métodos utilizados para modelar el series temporales también de múltiples dominios de conocimiento. En la

[3]Schema: https://schema.org/

propuesta desarrollada en `tswf-schema` se ha intentado integrar el mayor número de funcionalidades, junto con las más comunes durante el proceso de flujo de trabajo con series temporales. También incluye modelización genérica establecida para la metodología *Box-Jenkins* así como experimentación basada en flujos de trabajo para procesar problemas de minería de datos aplicada a series temporales [Bontempi et al., 2012].

Otro elemento destacado que se ha integrado con la definición propuesta con `tswf-schema`, es que contiene todos los elementos clave en la descripción de este tipo de servicios en Cloud Computing, tales como puntos de interacción, precios, instancias o SLAs, entre otros, y sirve además como herramienta para la integración de un flujo de trabajo con series temporales como servicio en plataformas Cloud Computing. Esto significa que permite tener la descripción completa de servicios de Cloud Computing, tanto desde un punto de vista funcional relativo al objeto de análisis de las series temporales, como a todos los elementos que un proveedor de Cloud Computing necesita gestionar para ofertar ese servicio.

En las siguientes subsecciones se detallan los componentes principales de la definición del esquema `tswf-schema`, y además se realiza una breve instanciación de algunos de los componentes:

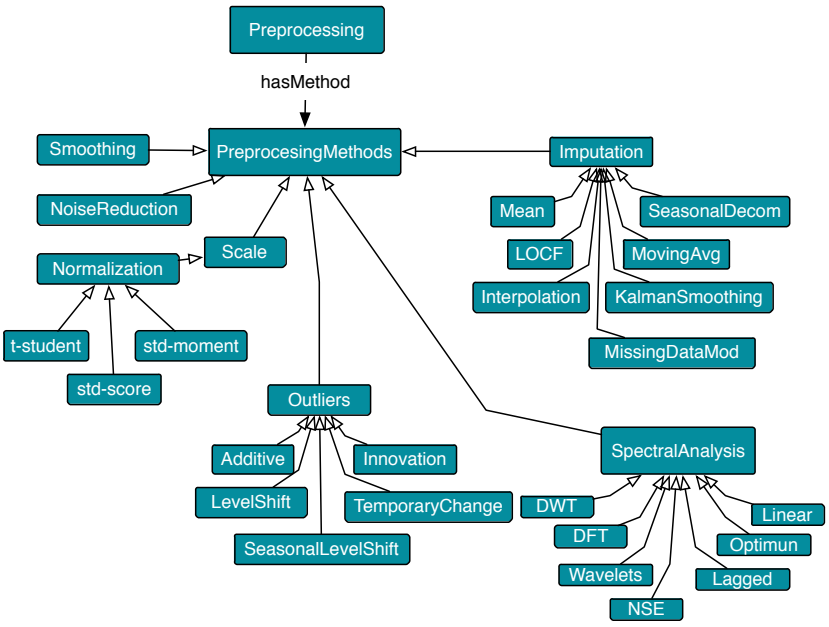

Figura 7.2: Detalle de la composición de los módulos de pre-procesamiento en `tswf-schema`.

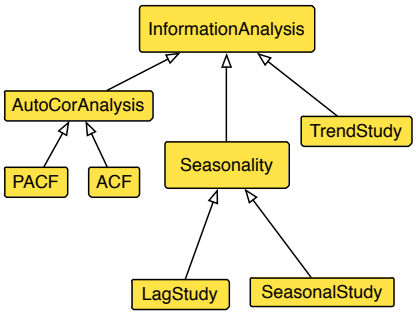

Figura 7.3: Elementos incluidos en el análisis de la información con `tswf-schema`.

Pre-procesamiento. Parte del análisis y trabajo con series temporales requiere operaciones sobre los datos, donde se aplican transformaciones, reducciones, limpiezas o imputaciones entre otras. En el diagrama de la figura 7.2 se pueden ver los elementos más utilizados en el procesamiento de series temporales a nivel de datos. Básicamente se ha realizado una agrupación en diferentes categorías, tales como imputación, valores atípicos, análisis espectral, escalado, reducción de ruido o suavizado. Cada una de las partes en las que se desglosa el componente, contiene varias de las funcionalidades que se han considerado las más utilizadas en el tratamiento de este tipo de datos, según el trabajo realizado en [García et al., 2015].

Para aplicar un algoritmo de pre-procesamiento se realiza mediante la propiedad de objeto `tswf-schema:hasMethod`, y donde posteriormente se definen cada uno de los métodos que se aplicarán a los datos de entrada. Para incluir en el modelado una fase de pre-procesamiento en la que se realiza una normalización de los datos, basada en la *T-student*, se usaría `a tswf-schema:Scale` y `a tswf-schema:t-student`. No sólo sería necesario instanciar el método o los métodos, también es necesario incluir la parametrización de los mismos.

Análisis de la información y la estacionalidad. La mayoría de los estudios estadísticos con series temporales se basan en la metodología *Box-Jenkins*, en la que se busca y modelar propiedades como la tendencia, o la estacionalidad. Como se muestra en las Figuras 7.3 y 7.4, se han dividido en dos partes, por un lado, el análisis de la información, que incluye pruebas de correlación, estacionalidad y tendencias y por otro lado, también se incluyen pruebas estadísticas, integrando estacionariedad, normalidad, aleatoriedad o no linealidad entre otras.

Con el análisis de la información y la estacionalidad, se permite que

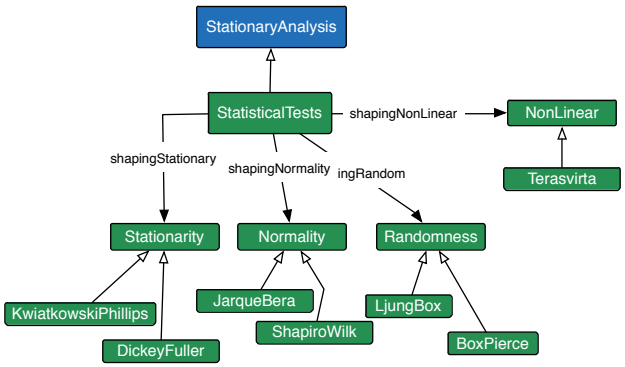

Figura 7.4: Componentes detallados de la estacionalidad con `tswf-schema`.

el modelado contemple un estudio completo de las características de la serie temporal, que nos permitirán mejorar el conocimiento sobre los datos con los que se trabaja. Para incluir un análisis de la auto-correlación (PACF[4] y ACF[5]) es necesario instanciar `tswf-schema:hasInformationAnalysis` con el conjunto de análisis que se modelarán. Para un estudio de auto-correlación se usa `a tswf-schema:PACF` y `a tswf-schema:ACF`. En cuanto al estudio de la estacionalidad, por ejemplo si se quiere incluir un test de normalidad de *Jarque-Bera*, se usaría `tswf-schema:shapingNormality JarqueBera`.

Medidas de evaluación. Se han considerado cuatro grupos principales de medidas, relacionados con los más utilizados en problemas de series temporales, tales como la clasificación, las medidas de similitud [Serra and Arcos, 2014], la precisión de la previsión [Weigend, 2018], el rendimiento de la agrupación [Montero et al., 2014], o el análisis de los residuos. Se han tenido en cuenta medidas de clasificación como *F1-Score*, *ROC* o matriz de confusión. Para medidas de similitud se han implementado otras como DTW [Mueen and Keogh, 2016], Edit Distance [Chen et al., 2015b] o Jaccard [Aghabozorgi et al., 2015], ya que son comunes en problemas de este tipo. En cuanto a las medidas de error, que identifican la calidad del ajuste del pronóstico, se han incluido todas las medidas consideradas en el trabajo [Armstrong and Collopy, 1992]. Si el problema que se aborda está relacionado con la agrupación de series temporales, se han tenido en cuenta algunas medidas como *APN*, *AD/ADM* o *Silhouette*. La Figura 7.5 detalla el conjunto de medidas

[4]Partial autocorrelation function
[5]Autocorrelation

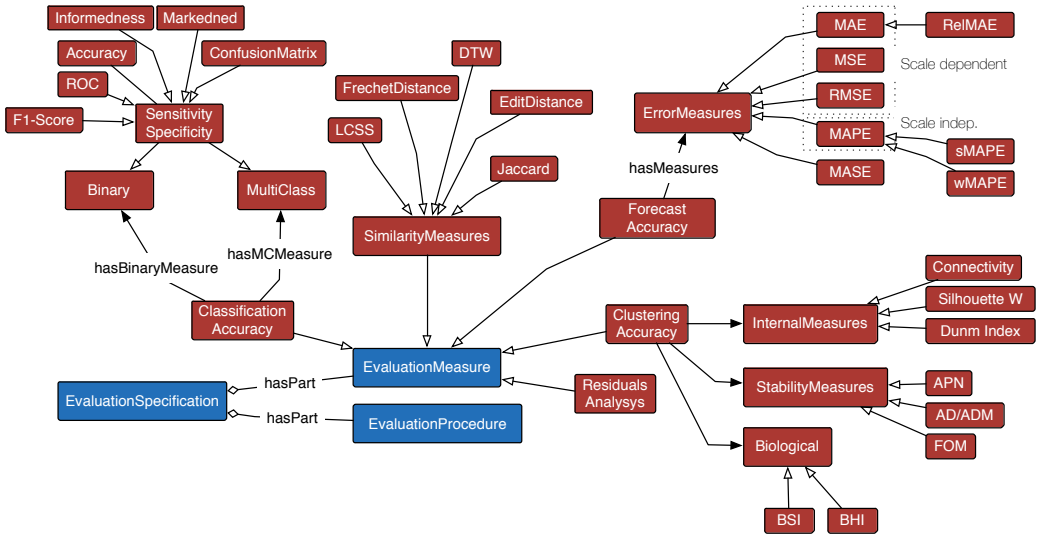

Figura 7.5: Medidas de evaluación integradas en `tswf-schema`.

de evaluación que se han incluido en el esquema `tswf-schema`.

El conjunto de medidas de evaluación es el más extenso en cuanto a clases que se pueden definir. En el modelado de la precisión de la predicción, para las medidas de error *MSE* y *RMSE* se realizaría la definición con `tswf-schema:hasMeasures MSE,RMSE`. De igual modo sería posible incluir en el modelado otras medidas relativas a precisión en la clasificación de series temporales o de calidad de agrupamiento (clustering) de series temporales.

Modelo predictivo. Al instanciar un modelo que trata de ajustar los datos del series temporales, se pueden aplicar una multitud de algoritmos y métodos estadísticos. En nuestro trabajo se han integrado los algoritmos y métodos más comúnmente utilizados en este área del análisis de datos de series temporales. A partir de métodos estadísticos muy extendidos como *ARIMA* o sus variantes (*ARIMA*, *AR*, *SARIMA*, *ARIMAX*, etc.). El análisis de regresión también se ha implementado con métodos como *LASSO* o *MARS* entre otros. Finalmente, se ha considerado una parte importante de los métodos de Machine Learning [Ahmed et al., 2010]. Estos algoritmos incluyen la aplicación de técnicas basadas en redes neuronales, *Random Forest* o *SVM* entre las más destacadas. El diagrama completo de los métodos y algoritmos incluidos en el `tswf-esquema` se puede ver en la figura 7.6.

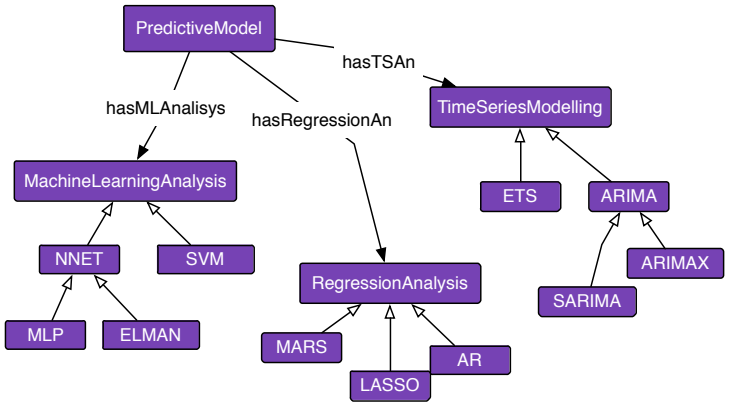

Figura 7.6: Componentes del módulo de análisis predictivo con `tswf-schema`.

Por ejemplo para incluir un modelo predictivo de tipo *ARIMA*, sería necesario instanciar `tswf-schema:hasTSAn` con el tipo `TimeSeriesModeling`, de modo que para utilizar un modelado con *ARIMA*, tendíamos `a tswf_schema:ARIMA`, con sus correspondientes parámetros de configuración.

Visualización de los datos. Una parte importante de la fase de análisis del trabajo en una serie temporal es la visualización de los datos en una serie de gráficos. Esto permite obtener la información visual, desde diferentes tipos de trazos como descomposición, o la diferenciación de las series temporales. En la figura 7.7 se pueden ver las implementadas en el esquema.

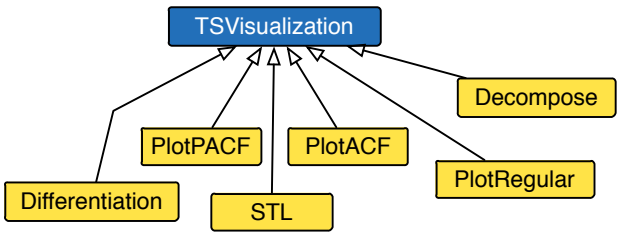

Figura 7.7: Modelado de la visualización de series temporales con `tswf-schema`.

Por ejemplo para realizar la visualización basada en diferenciación de una serie temporal, utilizaríamos primero `tswf-schema:hasPlot Diff`, y posteriormente definimos `Diff` como el tipo de visualización que deseamos con `_:Diff a TSVisualization, Differentiation`.

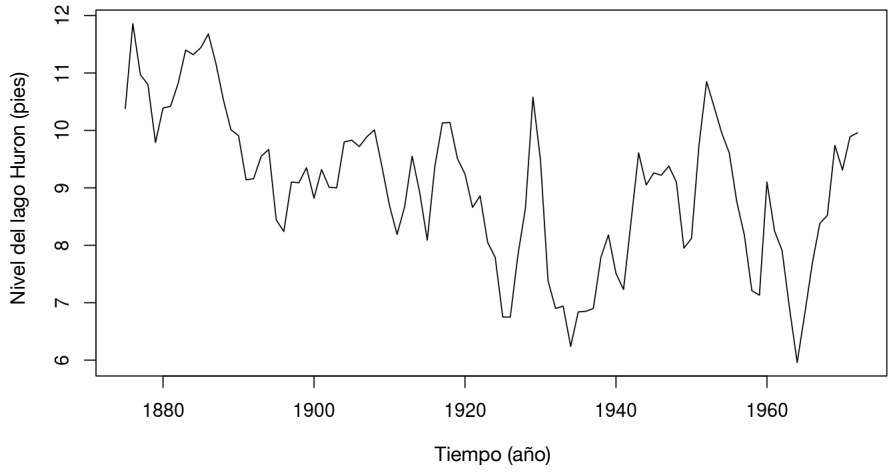

Figura 7.8: Visualización de los datos de la serie temporal de ejemplo del lago «Huron».

Se han descrito todos los componentes de `tswf-schema` de los elementos para el modelado de la experimentación con series temporales; para validar que la definición del servicio permite abordar cualquier tipo de problema o análisis con series temporales, diferentes ejemplos del libro de Hyndman [Hyndman and Athanasopoulos, 2018] han sido trasportados a nuestro esquema semántico `tswf-schema`. En la siguiente subsección se aborda un ejemplo completo de modelado de servicio de series temporales con el esquema propuesto.

7.4. Uso del esquema `tswf-schema`

Para mostrar la validez del esquema desarrollado con `tswf-schema` se ha descrito un problema clásico de análisis de series temporales. En este problema se realizan una serie de análisis sobre un conjunto de datos de lago «Huron»[6] (ver Figura 7.8) que contiene datos del nivel de agua del lago (en pies) desde 1875 hasta 1972. Para la definición de este ejemplo en `tswf-schema` se hará una descripción completa del servicio tomando todos los elementos que se implementan en *R*, pero en este caso haremos la transformación necesaria al esquema.

[6] `https://pkg.robjhyndman.com/fma/reference/huron.html`

La primera parte para la definición del servicio corresponde con los datos generales de los módulos o clases que se van a incluir en el estudio o análisis. Esto sería equivalente a preparar una etapa inicial en la que se identifican las operaciones que se quieren realizar sobre los datos a nivel global. Si observamos el diagrama de la Figura 7.1, la parte inicial desde podemos empezar a definir el análisis de la serie temporal corresponde con `tswf-schema:TSAnalysis`, desde donde se definen parte de los elementos que se pueden modelar con el esquema.

En el listado de código 7.4 se instancian cada uno de los componentes que corresponden a la capa de abstracción más alta que se va a modelar en el estudio de las serie temporal de ejemplo. Lo primero que indicamos es que es vamos a hacer un análisis de series temporales con `:TS_1`, que es de tipo `tswf:TSAnalysis`, con lo que a partir de aquí todo lo que definimos debe estar en concordancia con el esquema de la Figura 7.1 en cuanto a nombre de las propiedades de las clases y las relaciones entre ellas. De las líneas 4 a 8, se identifican todos los elementos que formarán parte del estudio:

- Entrada de datos. Se describe la entrada de datos que se usará para el modelado de la serie temporal. Para ello es necesario instanciar, `tswf:hasInput :TSinput_LakeHuron ;`, de modo que más adelante definiremos la información exacta de los datos de entrada de `:TSinput_LakeHuron`. Es posible añadir más entradas de datos o conjuntos de datos, si fuese necesario añadiendo una instancia de `tswf:hasInput` cada uno de los conjuntos de datos, como por ejemplo `tswf:hasInput :DataSet1,:DataSet2 ;`. La definición aún no está completa, más adelante se definirán los detalles del conjunto de datos en particular.

- Salida del estudio. La salida del estudio puede ser múltiples: un valor o conjunto de valores, una tabla, otro conjunto de datos o unas gráficas. Para este ejemplo de serie temporal utilizaremos una medida de evaluación la cual se instanciará con `tswf:hasOutput :TSEvaluationMeasure ;`, desde donde `:TSEvaluationMeasure` debe ser definida más adelante. De igual modo que en la entrada de datos, es posible incluir diferentes medidas de evaluación del análisis de la serie temporal.

- Modelo predictivo. En este ejemplo se realizará un modelado predictivo utilizando *ARIMA*, de modo que para ello, primero se instancia `tswf:perform :TSPredictiveModel ;` y luego en `:TSPredictiveModel` se describen los métodos que se aplicarán a los datos conforme al diagrama 7.6.

- Estudio del problema. Cada uno de los análisis puede estar dentro de una serie de estudios diferentes, con un mismo objetivo de analizar los datos de la serie temporal. Con esto se permite tener un mismo conjunto de datos y distintos estudios de modelado con el esquema `tswf-schema`.

- Tarea. La tarea indica la posibilidad de que el estudio esté dentro de un conjunto mayor de operaciones o tareas que se tengan que hacer a modo de flujo de trabajo. De este modo el modelado del análisis puede pertenecer a otro grupo de tareas por ejemplo relacionadas con la Minería o el procesamiento de Datos.

```
1 _:TS_1 rdf:type owl:NamedIndividual,
2          tswf:TSAnalysis ;
3
4   tswf:hasInput  :TSinput_LakeHuron ;
5   tswf:hasOutput :TSEvaluationMeasure ;
6   tswf:perform   :TSPredictiveModel ;
7   tswf:compris   :TSProblemStudy ;
8   tswf:achieves  :TSTask1 ;
9 .
```

Una vez definidos los elementos vertebrales del análisis, ahora pasamos a describir parte de los elementos que lo componen. Para el elemento `:TSinput_LakeHuron` que corresponde con la entrada de datos, es posible definir una serie de elementos para el modelado, tales como el tipo de dato correspondiente a la serie temporal, los gráficos que se generarán, o los análisis de información o estacionalidad. En el listado de código 7.4 por ejemplo en las líneas 5-8 se definen los graficos que se crearán a partir de la entrada de datos `tswf:hasPlot` para los siguientes ACF `:TSplot_ACF_LakeHuron`, series temporalesL `:TSplot_series temporalesL_LakeHuron`, o un gráfico normal de visualización de la serie con `:TSplot_Regular_LakeHuron`. Posteriormente en este mismo listado de código desde la línea 7-15, una serie de estudios sobre análisis de la información, son realizados. En esta parte es posible combinar diferentes análisis, de modo que para un mismo conjunto de datos de series temporales, podamos tener todas las opciones de estudio sobre la información de la serie temporal, como *PACF* `tswf-schema:PACF`, *ACF* `tswf-schema:ACF` o *Lag* `tswf-schema:Lag` entre otros. Finalmente este estudio se complementa con un análisis de la estacionalidad `tswf:hasStationaryAnalysis :TSStatAnalysis;` que más adelante se detallará que métodos se van a describir.

```
:TSinput_LakeHuron a tswf:Data, tswf:Dataset ;

  tswf:hasTimeInterval :TSdataInterval;
  tswf:hasQuality :TSdataClass;
  tswf:hasPlot :TSplot_PACF_LakeHuron,
    :TSplot_ACF_LakeHuron,
    :TSplot_series temporalesL_LakeHuron,
    :TSplot_Regular_LakeHuron;

  tswf:hasInformationAnalysis
    :TSInfoAnalysis_PACF,
    :TSInfoAnalysis_ACF,
    :TSInfoAnalysis_Lag,
    :TSInfoAnalysis_Seas,
    :TSInfoAnalysis_TrendStd;

  tswf:hasStationaryAnalysis
    :TSStatAnalysis;
 .
```

Para detallar por ejemplo el estudio de la estacionalidad `:TSStatAnalysis` se pueden utilizar varios test como *DickeyFuller* `tswf:shapingStationary :TSDickeyFuller;`, *Jarque-Bera* `tswf:shapingNormality :TSJarqueBera;` o *Jung-Box* `tswf:shapingRandomness :TSLjungBox;`, tal y como se ven en el listado 7.4.

```
:TSStatAnalysis a tswf:StatisticalTest  ;

 tswf:shapingStationary :TSDickeyFuller;
 tswf:shapingNormality :TSJarqueBera;
 tswf:shapingRandomness :TSLjungBox;
 .
```

Para este ejemplo además se han aplicado varios modelos predictivos para hacer estimaciones, de modo que ahora pasamos a definir en detalle como se especifica cada uno de ellos, pero nos centraremos en la descripción del modelo *ARIMA*. En el listado 7.4 hemos especificado que nuestro modelado de series temporales va a tener un modelo predictivo (uno o varios, según el flujo de trabajo que queramos), llamado `:TSPredictiveModel`. En el listado 7.4 recoge cada uno de los grupos de modelos predictivos que se aplicarán al estudio; para ello, por un lado dos tipos de análisis basados en técnicas de Machine Learning (`:hasMLAnalysis` y `:hasMLRegressionAnalysis`) de tipo análisis (*SVM*) y regresión (*AR*), y por otro lado el correspondiente a

ARIMA `tswf:hasTSAnalysis`.

```
:TSPredictiveModel a tswf:PredictiveModel;
  tswf:hasMLAnalysis     :TSMLAnalysis;
  tswf:hasMLRegressionAnalysis  :TSRegressionAnalysis;
  tswf:hasTSAnalysis     :TSAnalysis;
.
```

Para definir la parametrización de *ARIMA* `tswf:ARIMA` desde de instanciación `:TSAnalysis`, en el listado 7.4 se indican los parámetros básicos (no todos son mostrados) del modelo de aprendizaje, tales como `tswf:order`: donde se realiza la especificación de la parte no estacional del modelo *ARIMA*: los tres componentes (p, d, q) orden AR, el grado de diferenciación y el orden MA. También se especifica el parámetro `tswf:seasonal`, gracias al cual se hace una especificación de la parte estacional del modelo *ARIMA*, más el período `tswf:period`.

```
:TSAnalysis a
    tswf:TimeSeriesModelling, tswf:ARIMA;
    tswf:params [
     [ a tswf:order;
       tswf:p "0";
       tswf:d "1";
       tswf:q "0"; ],
    [ a tswf:seasonal;
       tswf:period  "12";]
    ]
.
```

A partir de aquí sería necesario definir cada uno de los elementos adicionales que hemos ido indicando para el modelado de la serie temporal. No todos son mostrados aquí con detalle por problemas de espacio. El ejemplo completo detallado donde se realiza el modelado completo y estudio de la serie temporal del lago «Huron» mediante la tecnología semántica del esquema que proporciona `tswf-schema`, junto con otros ejemplos procedentes del libro de *Hyndman*, han sido detallados en el sitio web de contenido adicional del trabajo de tesis. Con este ejemplo sencillo, se constata como con el esquema `tswf-schema` es posible realizar un modelado completo de un problema con series temporales, listo para ser desplegado como servicio en Cloud Computing.

7.5. Ventajas del modelado de series temporales como servicio en Cloud Computing

Cuando se realiza una experimentación con series temporales, se está muy ligado al lenguaje, la plataforma o el sistema donde se realiza el modelado del análisis. Debido a esto, al realizar este tipo experimentación se está sujeto a una implementación concreta o a unos algoritmos que probablemente sólo están disponibles en unos lenguajes o bibliotecas específicas. Cuando se desarrolla una propuesta basada en tecnología semántica, es posible reducir esta fricción al máximo. Conjuntamente con el despliegue de este tipo de experimentación en Cloud Computing se obtendría un alto grado de flexibilidad, reproducibilidad y escalado, elementos clave en el modelo Cloud Computing, del cual heredaría todos sus atributos. En esta sección vamos a describir algunas de las ventajas más destacadas del modelado servicios de series temporales en Cloud Computing:

Estandarización de la definición de servicios. Lenguajes como *R*, *Python* o *MatLab*, ofrecen un amplio conjunto de bibliotecas para el trabajo con series temporales. Con esas herramientas es posible realizar el análisis de las series temporales de forma integral. Prácticamente la totalidad de las herramientas que están disponibles para el análisis de series temporales, son dependientes del lenguaje, de la plataforma, o de la arquitectura subyacente. Esto hace que la posibilidad de migrar o portar diseños de modelos de series temporales sea complicado.

Además, con esto reduce la capacidad de hacer ciencia reproducible. Cuando se trata de servicios en Cloud Computing, un elemento fundamental es la abstracción que supone el uso del modelo de Cloud. Gracias al uso de la tecnología semántica para la definición de servicios, se consigue un alto grado de homogeneización en cuanto a la descripción de los elementos que componen el modelado de series temporales. Disponer de un modelo semántico que capture todos los detalles de estudio completo de series temporales permitiría un grado de portabilidad y migración completamente garantizado. Con ello, el modelo desarrollado con `tswf-schema` sería desplegable como servicio en cualquier plataforma de Cloud Computing.

Escalabilidad del modelado. Cuando tradicionalmente desarrollamos un análisis de series temporales, hemos de tener en cuenta las características del problema y adaptar nuestros recursos al tamaño del mismo.

Será necesario aumentar la capacidad de cómputo si el problema lo requiriese en cada caso. El modelado de series temporales en Cloud Computing ofrece una ventaja excepcional y es que una de las características innatas del Cloud es la virtualmente ilimitada capacidad de escalado de los recursos computacionales.

Las demás herramientas de modelado de series temporales están muy sujetas a la arquitectura donde se desarrollan y las posibilidades de escalar los recursos de computación resultan complejas de implementar. El modelo de servicios de Cloud Computing permite un escalado de recursos computaciones bajo demanda, de modo que es posible adaptar la infraestructura a la carga de trabajo o las propiedades del problema en concreto. La definición de un modelado de servicio de series temporales, facilita en gran medida el despliegue de este tipo de servicios y sobre todo la abstracción sobre la infraestructura subyacente, dando más relevancia a la definición del servicio de series temporales que al soporte computacional requerido.

Flexibilidad. El esquema diseñado `tswf-schema` con tecnología semántica, está basada en las recomendaciones de la propuesta de *Linked Data*; esto supone que el propio esquema `tswf-schema` puede ser ampliado, modificado y mejorado si es necesario en el futuro. Uno de los valores fundamentales de *Linked Data* es la capacidad de extender el esquema a otros nuevos esquemas adicionales, de modo que se puedan complementar.

Como ejemplo aplicado a nuestro esquema `tswf-schema` la flexibilidad que ofrece a través de *Linked Data*, se fundamenta en el hecho de que es posible añadir referencias a otros esquemas, como por ejemplo `dmcc-schema`, mediante el cual un estudio con series temporales hecho en `tswf-schema` puede incluir partes de otros vocabularios que complementan la definición de servicios de minería de datos, y por tanto heredar todas las propiedades de ese nuevo esquema. Con esto se impide que la definición quede obsoleta con el tiempo y habilita a los desarrolladores a incluir nuevos esquemas que son siempre compatibles con lo ya desarrollado.

7.6. Conclusiones

En este caso de uso se ha desarrollado una propuesta de definición semántica para servicios de análisis y modelado de series temporales para Cloud

Computing. Gracias a la capacidad de la tecnología semántica para la definición de servicios, es posible componer un análisis de series temporales del mismo modo que se haría con las herramientas tradicionales como *R, Python o Scala*, ofreciendo toda funcionalidad requerida para un estudio completo, pero con ventajas clave como la homogeneización, la portabilidad y la independencia de la arquitectura o plataforma donde se desplieguen.

La utilización de la definición propuesta por `tswf-schema` en entornos de Cloud Computing, además supondría una ventaja con la que desmarcarse de las herramientas de modelado tradicionales, ya que este tipo de servicios se beneficiaría de todas las características de Cloud Computing como el escalado de recursos, la capacidad de procesamiento o la flexibilidad de la implementación.

En la propuesta se desarrolla un ejemplo básico donde se transforma un sencillo estudio de una serie temporal en el esquema `tswf-schema`, permitiendo modelar de forma completa todo el proceso de análisis. Este sencillo ejemplo de estudio, podría extrapolarse para cualquier problema o estudio con series temporales, con lo que la propuesta desarrollada por `tswf-schema` serviría para capturar todos los requerimientos del análisis de series temporales.

Capítulo 8

Modelado de sistemas difusos en Cloud Computing

8.1. Motivación

Los sistemas difusos han llegado a ser ampliamente aceptados y aplicados en una gran cantidad dominios tales como el control, la electrónica o la mecánica. El software para la construcción de estos sistemas ha sido tradicionalmente explotada a partir de herramientas, plataformas y lenguajes que funcionan en la infraestructura informática local u «on-premise». Por otro lado, el auge y la ubicuidad del modelo de computación en la nube ha traído una forma revolucionaria para el despliegue de servicios de computación. El impulso de los servicios en la nube está conduciendo hacia servicios cada vez más específicos como minería de datos y Machine Learning. Desafortunadamente, hasta ahora, no hay una definición disponible de sistema difuso como servicio en Cloud Computing. Este caso de uso trata de identificar esta oportunidad y se centra en el desarrollo de una propuesta para la definición de un servicio en Cloud Computing para el modelado de sistemas difusos. Para conseguirlo, la propuesta persigue tres objetivos: la descripción de los servicios para sistemas difusos mediante el uso de tecnología semántica, la composición de los servicios y la explotación de los mismos en plataformas de Cloud Computing para su integración con otros servicios. Como un ejemplo ilustrativo de este caso de uso, se aborda un problema del mundo real con la propuesta de especificación que se ha desarrollado.

8.2. Introducción

Durante más de 30 años, el software de sistemas difusos se han desarrollado como la aplicación más directa y práctica de la teoría de los conjuntos difusos [Zadeh, 1965]. Estos sistemas relacionados con el software han hecho posible que científicos e ingenieros se ocupen de la representación de conocimiento y razonamiento sujetos a imprecisión e incertidumbres para la solución de problemas de muy diversa temática. Los sistemas difusos son extremadamente efectivos en el modelado de sistemas complejos y tienen la capacidad de incorporar conocimiento experto humano incluso afectado por la incertidumbre.

En este sentido, los sistemas difusos se han implementado y aplicado en diferentes campos de aplicación como industria [Precup and Hellendoorn, 2011], electrónica [Kannan et al., 2014], sistemas de control [Alpay and Erdem, 2018], informática o mecánica [Pandeeswari and Kumar, 2016], entre muchos otros.

A lo largo de los años, se han desarrollado varias herramientas de software con el objetivo de facilitar el modelado y la rápida adopción del software de sistemas difusos dentro de un amplio conjunto de dominios y problemas. Tales aplicaciones permiten a los investigadores con conocimientos básicos de lógica difusa modelar un sistema difuso completo con el mínimo esfuerzo y de una manera totalmente flexible. El software de propósito general para sistemas difusos, para aplicaciones y para lenguajes específicos, está disponible como herramientas (XFuzzy [Moreno Velo et al., 2001], FuzzyStudio [de Souza et al., 2014], ...), plataformas (KNIME [Berthold, Michael R and Cebron, Nicolas a WEKA [Hall et al., 2009b],...) y bibliotecas (FRBS [Riza et al., 2015], pyFuzzy [McCulloch, 2017], ...).

En los últimos diez años y con una progresión creciente, Coud Computing está empujando el modelo tradicional de prestación de servicios bajo demanda a través de Internet [Zhang et al., 2010]. De hecho, Coud Computing se está convirtiendo rápidamente en una alternativa generalizada a las costosas infraestructuras locales con aspectos clave como la escalabilidad, la disponibilidad bajo demanda o el pago por uso. Software as a Service (SaaS), Platform as a Service (PaaS), e Infrastructure as a Service (IaaS) son las tres capas abstractas principales de servicios en Coud Computing, donde los recursos informáticos de los proveedores se despliegan como servicios (almacenamiento, computación y comunicaciones principalmente) listos para ser consumidos por los usuarios tal y como se muestra en la Figura 8.1.

Las soluciones tradicionales para la construcción de sistemas difusos care-

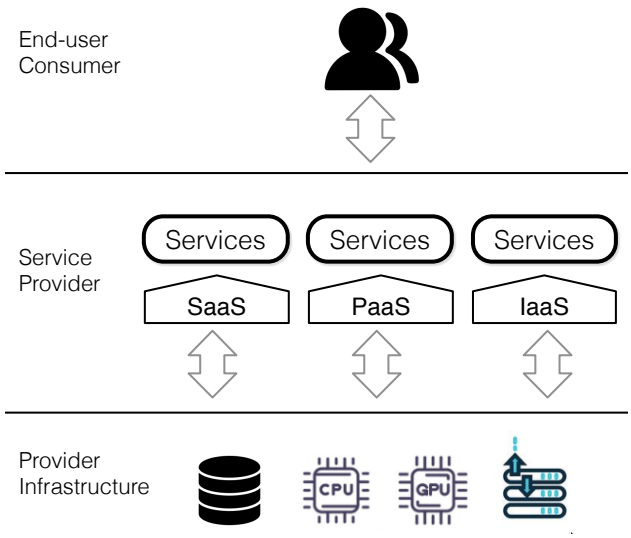

Figura 8.1: Arquitectura y componentes generales para el modelo Coud Computing.

cen de soporte para Cloud Computing. Por lo tanto, no se consideran aspectos como la flexibilidad, la portabilidad, el escalado de los recursos bajo demanda o el descubrimiento de servicios.

De esta manera, una extensión natural del software para sistemas difusos sería la implementación, construcción y despliegue de modelos de lógica difusa como servicio en plataformas de Cloud Computing, es decir, «Fuzzy Systems-as-a-Service» (FSaaS), con los beneficios [Marston et al., 2011] de escalabilidad, velocidad de desarrollo, portabilidad o interoperabilidad, así como otras características.

El objetivo de este caso de uso es hacer una propuesta efectiva para la construcción de modelos difusos sobre plataformas de Cloud Computing como servicio, considerando aspectos clave como el diseño, descripción (construcción), despliegue, composición y, finalmente, explotación. Para ello, se ha utilizado tecnología semántica permitiendo la creación y despliegue de sistemas difusos como servicios de computación en Cloud.

Este trabajo también considera aspectos como la composición e integración de los servicios de lógica difusa con otros servicios en Cloud Computing. La propuesta es lo suficientemente general como para acoger nuevos modelos de despliegue de servicios como los servicios «serverless» o la función como un servicio (FaaS). Esta propuesta pretende abordar el problema de la integración de herramientas y plataformas de modelado de sistemas difusos en entornos nativos de Cloud Computing, de forma que incluya todas las

características inherentes al modelo Cloud como la escalabilidad, la flexibilidad y la integración, siendo un paso adelante en la transformación de las herramientas tradicionales en servicios de Cloud Computing.

El documento está estructurado de la siguiente manera: la sección 8.3 recopila el estado del arte del software de sistemas difusos desde diferentes perspectivas. Luego, en la sección 8.4 se detalla la definición de sistemas difusos como un servicio basado en tecnología semántica. En la sección 8.5 se describen todas las ventajas de implementar este software en Cloud Computing en comparación con las herramientas de software tradicionales. Las conclusiones y observaciones finales del caso de uso desplegado se describen en la última sección.

8.3. Trabajos previos relacionados

El gran éxito del uso de sistemas difusos en una gran variedad de campos como la toma de decisiones, los sistemas de control, el reconocimiento de imágenes, ha motivado la llegada de muchas herramientas de desarrollo de software para la construcción de este tipo de sistemas. El desarrollo de sistemas difusos se ha realizado utilizando lenguajes, plataformas, librerías o entornos de desarrollo, tanto en versiones comerciales como de código abierto. La implementación para el uso posterior de los sistemas son siempre dispositivos de hardware específicos u «on-premise». Cloud Computing, sin embargo, permite una nueva forma de despliegue de servicios para tareas informáticas que también resultará muy beneficiosa para los sistemas difusos. En los siguientes párrafos se realiza un estudio del software para sistemas difusos.

El desarrollo de los modelos difusos se ha abordado desde diferentes puntos de vista. En [Alcalá-Fdez and Alonso, 2016] se hace una taxonomía de software para sistemas difusos distinguiendo entre propósito general, y propósito específico y detallando los lenguajes propuestos para el modelado de sistemas difusos.

Un problema importante a la hora de construir sistemas difusos es que no existe un estándar para modelar este tipo de sistemas. Aspectos como la terminología, la metodología o la compatibilidad se resolverían si se dispusiera de una homogeneización en el modelado. Para tratar de mitigar esto, se han desarrollado diferentes propuestas de lenguaje en los últimos años. La mayoría de las propuestas lingüísticas se centran en proporcionar entornos para modelar sistemas difusos tanto para dominios de propósito gene-

ral como para áreas específicas. Por ejemplo, el lenguaje de control difuso (`FCL` [Tiegelkamp and John, 1995]) ofrece una base común para el desarrollo de lógica difusa, a través del cual integrar los controladores. Lenguajes como Fuzzy Markup Language (*FML* [Acampora et al., 2013]) ofrece un lenguaje basado en *XML* para definir la especificación de sistemas difusos o *XFL* [López et al., 1997], una especificación de lenguaje para el modelado de sistemas difusos de propósito general utilizados dentro del sistema *Xfuzzy* [Moreno Velo et al., 2001] como plataforma CAD. Por otro lado, *XFSML* [Moreno-Velo et al., 2012], basado en *XML* y que sirve como punto de partida para la definición de un lenguaje de modelado estándar, junto con otras opciones más robustas como *IEEE P1855/D2.0*, y representan una clara apuesta por la estandarización en el modelado de control difuso.

Otro enfoque frecuente es el uso de lenguajes de programación de propósito general, complementado con librerías específicas que aportan funcionalidad de sistemas difusos. Se han desarrollado bibliotecas y paquetes muy diversos para lenguajes como *Matlab* [Johanyák et al., 2006], C/C++ [Rada-vilela, 2013], Java [Wagner, 2013], o *R* [Riza et al., 2015], entre otros. Usando estas librerías es posible construir modelos de sistemas difusos dentro del programa general que de otra manera se construirían. Esta elección permite fácilmente la integración con funcionalidades soportadas por librerías adicionales, por ejemplo, visualización, análisis de datos, monitorización, etc.

Por otro lado, el modelado de software para sistemas difusos ha sido ampliamente integrado en aplicaciones visuales para el modelado experimental y el procesamiento de datos, tales como *KNIME* [Berthold, Michael R and Cebron, Nicolas and Dill, Fa
o *WEKA* [Hall et al., 2009b]. Estas herramientas además de proporcionar un motor para el trabajo con minería de datos, también integran módulos para el modelado de sistemas difusos de una manera totalmente visual, permitiendo al usuario construir el sistema a través de la interconexión de bloques, etapas u operaciones.

La mayor parte de este software tradicional para el modelado de sistemas difusos está diseñado para funcionar en «on-premise». Hay una minúscula porción de software para sistemas difusos desplegado en la web. Con propuestas como *CAVUS* [Cavus, 2010], o *FuzzyStudio* [de Souza et al., 2014] es posible modelar un sistema difuso completo usando un navegador web. Sin embargo, estas propuestas no son nativas para la nube, por lo que no pueden aprovechar las características que ofrecerían las soluciones basadas en estos entornos. Para que este tipo de sistemas difusos se despliegue en la nube, es necesario disponer de las herramientas adecuadas para poder definir y describir los servicios para Cloud Computing de forma eficaz, con el fin de

aprovechar todo su potencial.

Se han desarrollado diferentes propuestas para la definición de servicios tanto en el campo sintáctico como en el semántico. A nivel sintáctico, parte de las propuestas se basan en la arquitectura orientada a servicios (*SOA*) y *XML*, además de otros lenguajes derivados como *WSDL* [Christensen et al., 2001], *WADL* [Hadley, 2006] o *SoAML* [Elvesæter et al., 2010] que tratan a nivel sintáctico los aspectos técnicos de los servicios en general.

Las propuestas con tecnología semántica tienen un valor añadido creciente, ya que ofrecen una flexibilidad mucho mayor que los lenguajes sintácticos, permitiendo capturar las características funcionales y no funcionales del servicio. *OWL-S* [Martin et al., 2004] o *WSMO* [Domingue et al., 2005], son algunas de las propuestas básicas para el desarrollo de definiciones semánticas en general. Otros, como *USDL* [Kona et al., 2009], incluyen muchos más aspectos relacionados con el servicio en Cloud Computing, como entidades, precios, composición o aspectos legales. Por otro lado *Linked-USDL* [Pedrinaci et al., 2014] ofrece un vocabulario y un esquema completo para la definición de servicios genéricos en Cloud Computing e incluye elementos de modelado como catálogo, interacción o Acuerdo de Nivel de Servicio (SLA) entre muchos otros.

En esta línea, nuestra propuesta pretende llenar el vacío existente en la descripción de los servicios para sistemas difusos en Cloud Computing, a través de los cuales es posible diseñar, desplegar y explotar software para sistemas difusos en plataformas Cloud, con todas las ventajas que ello conlleva.

8.4. Definición y descripción de un servicio de sistema difuso en Cloud Computing

8.4.1. Sistemas difusos

Introducida por L. Zadeh en 1965, la teoría de conjuntos difusos surgió como una extensión de la teoría clásica de conjuntos, para el modelado de conjuntos donde sus elementos pueden pertenecer con un grado entre 0 y 1, y no necesariamente igual a los valores extremos. Este grado de membresía indica cuán cerca está de ser miembro (1) o no ser miembro (0), mientras que los valores intermedios indican una membresía parcial. El grado de membresía se indica mediante una función llamada función de membresía.

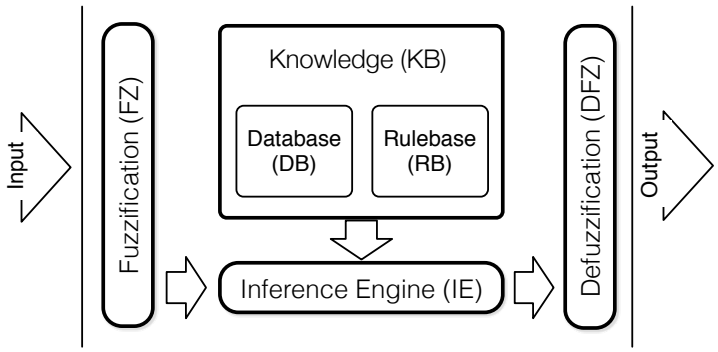

Figura 8.2: Componentes del modelo Mamdani.

Además, muchos conceptos «difusos» derivan del lenguaje humano, que es eminentemente vago. La lógica derivada de los conjuntos difusos, la lógica difusa, se convierte en una herramienta muy poderosa para representar conceptos, variables y reglas lingüísticas, necesarias para representar el conocimiento experto humano. Un concepto clave es la variable lingüística que toma como valores los términos lingüísticos de un conjunto que se expresa como conjuntos difusos.

El concepto tradicional de sistema basado en reglas se extiende naturalmente al sistema basado en reglas difusas (FRBS), cuando algunos o todos sus componentes son de naturaleza "difusa". El caso más frecuente es cuando las reglas son lingüísticas. En relación a la estructura de la regla, hay dos modelos muy utilizados: Mamdani y TakagiSugenoKang (TSK). La figura 8.2 representa la arquitectura del modelo Mamdani, que consta de 4 componentes clave a) «Fuzzification» (Conversor de nítido-crisp a difuso): transforma las entradas crisp en valores difusos, b) base de conocimientos: almacena una base de datos y una base de datos de reglas, c) motor de inferencias: realiza operaciones de razonamiento sobre reglas difusas y datos de entrada, d) «Defuzzification» (Conversor de difuso a nítido-crisp): produce valores nítidos a partir de valores lingüísticos con el fin de producir un resultado. En el modelo Mamdani, tanto el antecedente como el consecuente se componen de variables lingüísticas que siguen este modelo:

$$\text{IF } X_1 \text{ is } A_1 \text{ and } ... \, X_n \text{ is } A_n \text{ THEN } Y \text{ is } B \tag{8.1}$$

Las reglas TSK difieren de las de Mamdani en que su consecuencia es una función de las variables de entrada y por lo tanto no se necesita ningún conversor de difuso a nítido. De esta forma, la construcción de un FRBS requiere

la definición de cada uno de los elementos que lo componen: variables, reglas, motor de inferencia, módulos de conversión de nítido a difuso y viceversa. El componente más sensible de un FRBS es la base de conocimientos. Existen dos enfoques generales para producirlo: a) derivados del conocimiento experto a través de algunos procesos de extracción de conocimiento, b) calculados a través de métodos de aprendizaje. Además, a lo largo de los años se han desarrollado varias propuestas híbridas que integran las mejores propiedades de los sistemas difusos con otras técnicas de inteligencia computacional. Ejemplos claros de esta hibridación son los sistemas neurodifusos y los sistemas genéticos difusos. Ambos tipos de FRBS (es decir, Mamdani y TSK), así como los sistemas híbridos más utilizados, deben tenerse en cuenta a la hora de definir una plataforma para el desarrollo y la explotación del FRBS.

8.4.2. Definición y descripción de un servicio de sistema difuso en Cloud Computing

8.4.2.1. Semántica del modelado de sistemas difusos

La parte principal del diseño de un servicio para lógica difusa se centra en la construcción del sistema difuso y sus componentes. Después de un cuidadoso examen de las monografías ampliamente aceptadas y de consultar algunas de las propuestas mencionadas en la sección 8.3, se ha definido un sistema semántico difuso. Los componentes y la estructura de la propuesta se muestran en la figura 8.3. Los componentes más relevantes se detallan a continuación:

- Entradas y salidas. Corresponden a las variables de entrada y salida del sistema difuso. Ambos tipos de variables necesitan ser definidas indicando su nombre y características específicas a través de su Tipo. Para la definición semántica de ambos tipos de variables es necesario utilizar `fsschema:hasInput` y `fsschema:hasOuput`, ambos apuntan respectivamente a `Input` y `Out` como clases de definición para los tipos de variables. Ambos requieren un elemento `fsschema:type` y `fsschema:Membership`. (no se muestra en la Figura 8.3).
- Definición de tipos de variables lingüísticas. El tipo de una variable de los sistemas difusos se define por el nombre del tipo y universo del discurso, en conjunción con las funciones de membresía. El universo del discurso se identifica con los valores máximo, mínimo y cardinalidad.

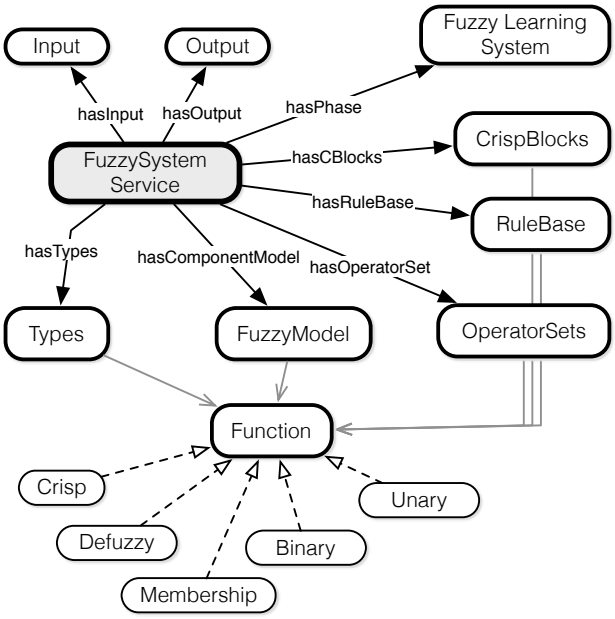

Figura 8.3: Estructura general del esquema de modelado de sistemas difusos con `fsschema`.

Las funciones de membresía están contenidas dentro de una clase más general que aglutina todas las funciones que el servicio es capaz de desplegar para la construcción del sistema difuso y que se utilizan para describir etiquetas de idiomas. Para definir cada uno de los tipos de variables de modelo difuso es necesario instanciar `fsschema:hasType` y `fsschema:Type`. Para especificar completamente el «Tipo» utilizamos las propiedades `min`, `max`, `cardinality` y `nombre`, e incluimos también todas las funciones de membresía necesarias con `fsschema:hasMembershipF`. En el diagrama 8.3 se muestra el conjunto de funciones que incluye el `fsschema` dentro de la clase `fsschema:Function`. Por ejemplo, para instanciar una función de membresía triangular, utilice `fsschema:MembershipF` y las siguientes propiedades: para definir el nombre de la etiqueta lingüística `label` y `fsschema:hasProperties` de la clase `Function` para `fsschema:Membresía` de la clase `triangular` con las propiedades `a`, `b`, `c`, correspondientes a los valores de la función triangular, como se ilustra en la figura 8.4. De esta manera con `fsschema` es posible recoger todos los conceptos habituales que se manejan en el área de sistemas difusos.

Se definen seis tipos de funciones con `fsschema` (ver Tabla 8.1): funciones binarias que pueden ser utilizadas como funciones T-norms, S-norms, y funciones de implicación; funciones unarias que están relacio-

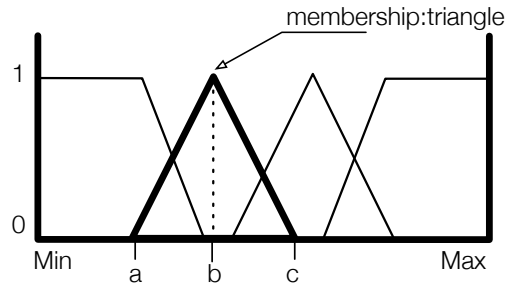

Figura 8.4: Modelado difuso para una función de membresía con `fsschema`.

Unary	not,sqrt,square, cubic, sugeno, pow
Binary	max,min,prod,sum,zadeh,godel,...
Crisp	add,diff,prod,div,sample
Membership	triangle,trapezoid,slope,...
Defuzzification	mean,Gamma,TakagiSugeno,..

Tabla 8.1: Grupos de funciones y funciones incluidas en `fsschema` (no todas son mostradas).

nadas con coberturas lingüísticas; funciones nítidas que implementan bloques nítidos; funciones de membresía que se utilizan para definir la semántica de las etiquetas lingüísticas; familias de funciones de membresía; y métodos de conversión de nítido a difuso y viceversa.

- Bloques Crisp. Las funciones Crisp se utilizan para describir operaciones matemáticas entre variables. Estas funciones pueden asignarse a otros módulos crujientes e incluirse en la composición de sistemas difusos. La tabla 8.1 muestra algunas de las funciones incluidas por `fsschema`.

- Conjuntos de operadores. Un conjunto de operadores contiene las funciones matemáticas que se asignan a cada operador difuso. Por tanto, `fsschema:hasOperatorSet` y `fsschema:OperatorsSet` son necesarios para describir los operadores que se utilizarán en todo el conjunto.

- Regla y base de reglas. Las reglas se construyen con un antecedente y un consecuente. El antecedente describe las relaciones entre las variables de entrada del sistema difuso y el consecuente describe la asignación de una variable lingüística a una variable de salida, por ejemplo en el modelo Mamdani. En el modelo TSK, la consecuencia está representada por una función de las variables de entrada.

Para cada regla, se puede establecer un peso específico o un valor de confianza. La instanciación se realiza con `fsschema:hasRuleBase` de la clase `fsschema:RuleBase` y datos como nombre de las reglas base, variables de entrada, salida, la construcción de las reglas se hace con `fsschema:Rule` y las propiedades `fsschema:hasAntecent` (tiene antecedente) y `fsschema:hasConclusion` (tiene consecuente o conclusión).

- «Fuzzification» y «Defuzzification». El módulo «fuzzification» se encarga de convertir una entrada nítida en una variable lingüística utilizando las funciones de membresía. Este módulo también puede definirse como un módulo de mapeo desde una entrada observada a un conjunto difuso de etiquetas en un universo de entradas específicas para el universo del discurso. Por otro lado, el módulo de «defuzzification» convierte la salida difusa del motor de inferencia en valores Crisp usando funciones de membresía similares a las utilizadas por el módulo «fuzzification». Un ejemplo natural y simple de una función «fuzzification», es convertir un valor de entrada nítido en un «singleton» difuso, dentro del universo específico del discurso. Para ello, el tipo `fsschema:Type` de la entrada se especifica como una función de pertenencia que se declara con `fsschema:MembershipF`, el conjunto de etiquetas lingüísticas `label` y su pertenencia `fsschema:Membership`, en este caso `singleton`, se asigna con el valor de definición `a`.

- Métodos de aprendizaje. Los métodos de aprendizaje constan de dos partes: a) la identificación de la estructura, donde se genera una base de reglas y b) la optimización de la estimación de los parámetros de las funciones de los miembros. La estructura que se ha diseñado para describir el modelado de la parte del sistema difuso se puede ver en la figura 8.5. Para la instanciación de una fase de aprendizaje se utiliza `fsschema:hasPhase LearningM`. A continuación, para la descripción de `LearningM` es necesario indicar algunas entidades, como el algoritmo de aprendizaje que se utilizará, `fsschema:hasAlg`. También es necesario definir el conjunto de datos de formación con `fsschema:hasTData` y el tipo de entrada y salida (`fsschema:hasInput` `fsschema:hasOutput`) que utilizarán los métodos de aprendizaje para construir una estructura basada en reglas. El sistema de aprendizaje soporta procedimientos independientes así como varios modelos híbridos como los sistemas Neuro-difusos.

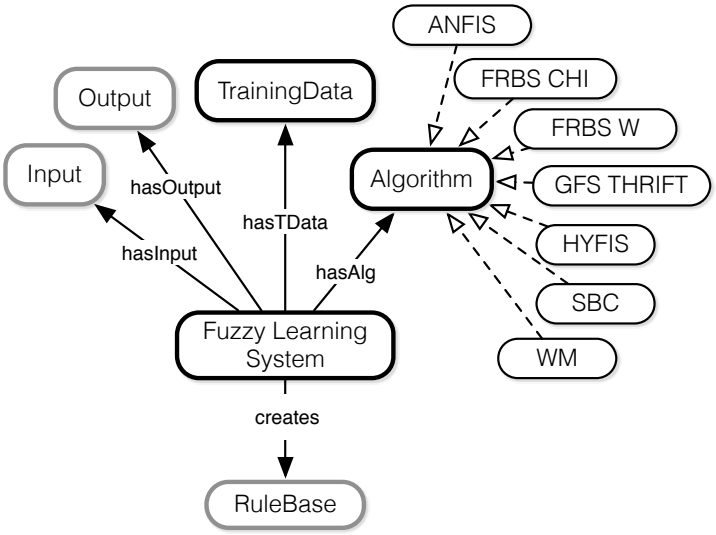

Figura 8.5: Descomposición de la clase de aprendizaje difuso.

8.4.2.2. Semántica de gestión de servicios en Cloud Computing para sistemas difusos.

La definición de un servicio en Cloud Computing no está completa sin tener en cuenta los elementos de gestión que el proveedor de la nube necesita para el intercambio de información con el consumidor usuario de servicios u otras entidades externas. Estos elementos son, por ejemplo, el precio de los servicios, la autenticación, la SLA o las interfaces de interacción, entre otros.

La definición semántica de los servicios de Cloud Computing tiene las ventajas de negociación, composición e invocación, con un alto grado de automatización. Esta automatización es fundamental porque permite que los servicios sean explorados y descubiertos para su explotación por otras entidades. En esta línea, por ejemplo, otras entidades, como usuarios o corredores de Cloud Computing o Cloud, podrían comprobar los costes asociados al uso de los sistemas difusos en Cloud o saber qué métodos son necesarios para autenticarse al servicio de los sistemas difusos si están definidos. La figura 8.6 muestra el diagrama de modelado para la definición completa del servicio de sistema difuso, incluyendo todos los elementos adicionales de la gestión de servicios en Cloud Computing tales como autenticación, catálogo, entidades, interacción, precios y SLA.

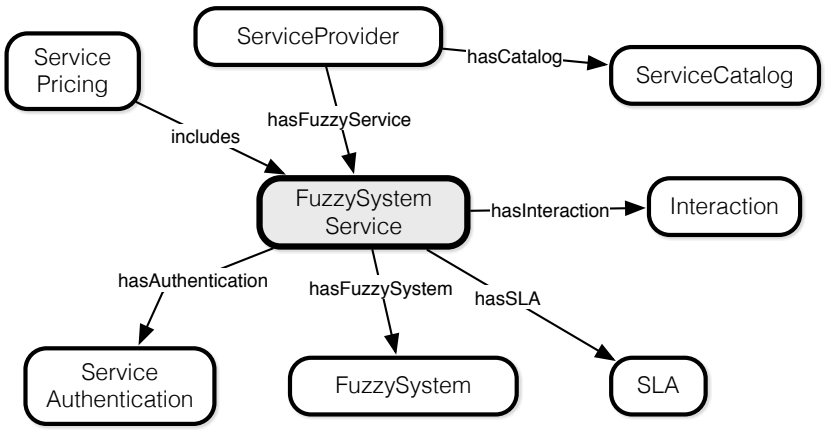

Figura 8.6: Modelo semántico general para la gestión de Cloud Computing incluyendo el sistema difuso.

8.5. Ventajas de la propuesta de sistema difuso como servicio

A continuación se presenta un desglose de las principales ventajas de la adopción de sistemas difusos dentro de la nube:

- Estandarización del modelado. En la sección 8.3 se ha realizado una compilación de las distintas opciones para trabajar con lógica difusa a través de aplicaciones de propósito general, aplicaciones específicas, lenguajes, toolbox y librerías. Este conjunto de software es muy diverso y contiene muchos tipos diferentes de implementaciones para el modelado de sistemas difusos, y en la mayoría de los casos es incompatible en términos de definición y descripción del modelado con respecto al resto del software. Una de las características más destacadas del uso de la tecnología semántica para la definición de servicios en Cloud Computing es que permite capturar en detalle todos los componentes de un servicio (tanto desde el punto de vista del modelo de Cloud, como desde la descripción del modelado). La idea con los servicios Cloud es que la definición del servicio puede ser uniforme entre los diferentes proveedores de Cloud, logrando un alto grado de portabilidad en la construcción y modelado de sistemas difusos entre entidades en Cloud Computing. Esto significa que desde una única descripción de servicio, el despliegue de un sistema completo de lógica difusa en Cloud no depende de la arquitectura subyacente, el lenguaje implementado u otras variables que afecten a los sistemas locales.

- Síntesis y explotación del servicio. Una vez realizado tanto el modelado como la validación del sistema difuso la última parte del trabajo con el sistema corresponde a la integración en otras aplicaciones, entornos o sistemas físicos, por ejemplo. Como resultado, lo que finalmente se consigue es la explotación del sistema que se ha diseñado, ya sea en un entorno de prueba o en un entorno real que permita el despliegue del software implementado. Las herramientas «on-premise» disponen de módulos para la síntesis y explotación del modelo construido a lenguajes como *C/C++*, *Python*, *Java* o *VHDL* (para *FPGA* [Vuong et al., 2006]), que permiten su integración en otras aplicaciones y sistemas. Sin embargo, cuando se trata de sistemas difusos en Cloud Computing, este modo de explotación no tiene sentido debido a la naturaleza propia de la nube. El Cloud Computing aprovecha los nuevos modelos de explotación de servicios ofreciendo la capacidad de interoperatibilidad entre servicios. Con esto, por ejemplo, la salida de un servicio en Cloud puede ser utilizada para ofrecer inteligencia computacional a otro servicio que la demande. Esto implica que el modelo construido como servicio puede integrarse completamente con otros servicios en Cloud Computing.

- Modelo Serverless. La computación serverless es un modelo de ejecución de Cloud Computing en el que el proveedor de Cloud gestiona dinámicamente la asignación de recursos de computación, el modelo serverless es el compañero ideal para el despliegue de sistemas difusos. Cuando se diseña el sistema difuso, se puede integrar en un servicio serverless, lo que significa que gran parte del problema de escalado o distribución de los recursos informáticos reside en el proveedor. Esto, por ejemplo, permite que la fijación de precios se base en la cantidad real de recursos consumidos por el servicio, en lugar de en unidades computacionales pre-compradas. Esta funcionalidad de servicio se puede integrar en otras herramientas que se implementan en la nube, facilitando la implementación y el uso de modelos de cualquier otro servicio.

- Escalado de modelos difusos en BigData. Una de las ventajas fundamentales de Cloud Computing es la capacidad de aprovisionar recursos informáticos de forma dinámica y bajo demanda [Mell, Peter et al., 2011]. Cuando nos ocupamos de los problemas de modelado en sistemas difusos en Cloud, necesitan una infraestructura informática que soporte todas sus partes, como la construcción, validación y explotación del sistema diseñado. Para cada una de estas etapas, el modelo de nube puede proporcionar recursos de computación. Del mismo modo, gra-

cias a esta característica, por ejemplo cuando existe una base de reglas de un modelo de sistema difuso, con un número masivo de reglas, es necesario utilizar las capacidades innatas de la computación distribuida tanto en el procesamiento como en el almacenamiento que Cloud Computing ofrece de forma transparente.

- Portabilidad y reproducibilidad de modelos difusos. Otra de las grandes ventajas de utilizar Cloud Computing para el modelado de sistemas difusos es que gracias al uso de tecnología semántica para la definición de servicios, los modelos que se construyen son completamente portátiles y agnósticos respecto a los proveedores de Cloud Computing. De esta manera es posible trabajar con la misma definición de modelado de diferentes proveedores, de forma que se asegura la reproducibilidad [Peng, 2011] de la experimentación con sistemas difusos en Cloud Computing. También mejoraría la competitividad entre los proveedores que aceptan este tipo de servicios en su catálogo para aprovechar su infraestructura.

- Transformación digital. La nube es el elemento clave de los proyectos de transformación digital y proporciona la escala y la velocidad necesarias para que las empresas se centren en la transformación. En este contexto, el desarrollo de sistemas difusos en Cloud Computing permitiría avanzar en la transformación digital de procesos y sistemas que hasta ahora sólo se centraban en las herramientas tradicionales de modelización. Teniendo este modelo de construcción de servicios basados en sistemas difusos en Cloud, ofrece la posibilidad de trasladar los modelos difusos existentes a modelos basados en Cloud como «Fuzzy Systems-as-a-Service».

8.6. Conclusiones

El Cloud Computing está desplazando el modelo tradicional de consumo de servicios de Tecnologías de la Información y la Comunicación por parte de los usuarios hacia un modelo basado en servicios de Internet, donde no sólo existen servicios de almacenamiento genéricos, o recursos informáticos, sino también servicios más específicos como la minería de datos en Cloud Computing. Aprovechando esta tendencia creciente, la idea de cubrir la brecha existente para construir un esquema de definición y modelado de sistemas difusos en Cloud Computing, nos ha llevado a proponer en este artículo un

esquema y vocabulario completo basado en tecnología semántica llamado `fsschema`.

Este esquema permite de forma compacta y directa describir modelos de sistemas difusos como servicios en Cloud Computing. Por un lado, permite definir todos los componentes de un FRBS (entradas, salidas, base de reglas, funciones de pertenencia, operadores, elementos de aprendizaje,...) y, por otro, también contempla todos los aspectos de la gestión de Cloud Computing, que un proveedor en Cloud debe tener en cuenta para el despliegue del servicio y su posterior explotación por parte de los usuarios consumidores.

De esta manera, `fss-cc` recoge en una única definición todos los elementos que permiten describir un servicio completo en Cloud Computing, lo que es negado por otras propuestas para el modelado de estos sistemas.

El esquema se basa en la web semántica y el uso de ontologías y vocabularios, lo que a nivel práctico, por un lado, asegura que el modelado de sistemas difusos pueda ser fácilmente extendido y completado, y por otro lado confirma las capacidades relacionadas con la portabilidad de los servicios de sistemas difusos entre diferentes proveedores de Cloud.

Posibilidades como el escalamiento virtualmente infinito de la infraestructura, la integración de servicios de sistemas difusos dentro del ecosistema de un proveedor de Cloud, las capacidades para entregar y explotar el servicio difuso desde APIs o como funciones «serverless» independientes de los recursos, así como la inminente explotación de IoT en la que los sistemas difusos para los dispositivos serán más que relevantes, y ponen de manifiesto la importancia de la adopción del modelado de sistemas difusos en Cloud Computing

Finalmente, la aplicabilidad de `fsschema` a problemas de lógica difusa del mundo real ha sido ilustrada mediante el desarrollo de este caso de uso.

Capítulo 9

Conclusiones y líneas de trabajo futuras

El objetivo principal de esta tesis era abordar el problema de la definición y descripción de servicios de minería de datos para entornos de Cloud Computing, tanto desde la perspectiva del flujo de trabajo con algoritmos, como de la gestión del servicio a todos los niveles. Ambos son elementos claves para la estandarización de este tipo de definición de servicios y suponen una solución robusta para una descripción completa y unificada.

La principal aportación de este trabajo consiste en el diseño de un vocabulario y un esquema llamado `dmcc-schema` para la definición de servicios de minería de datos que unifica los elementos propios del proceso de trabajo con datos tales como los algoritmos, la parametrización o el flujo de trabajo, con los elementos de la gestión que un proveedor de Cloud Computing necesita implementar para poder dar soporte a todos los niveles a esos servicios que demandan los usuarios consumidores. Estos elementos de gestión incluyen la definición de los tipos de autenticación, los acuerdos de uso (SLA), las interfaces de explotación del servicio o la tarificación que se aplica, entre otros. Se ha escogido una propuesta de definición basada en tecnología semántica, ya que como se ha puesto de manifiesto a lo largo de la tesis, es una herramienta que permite capturar con el nivel de detalle deseado las características de los servicios, además de poder especificar tanto elementos funcionales como no funcionales de este tipo de problemas. Con eso se alcanza el primero objetivo marcado.

El segundo objetivo era el diseño y la implementación de una arquitectura de despliegue de servicios de minería de datos en Cloud Computing

(OC^{2}DM), que tiene la capacidad de transformar las definiciones de algoritmos y flujos de trabajo de la propuesta semántica `dmcc-schema` en servicios completamente funcionales y listos para consumir. La arquitectura puede incluir virtualmente cualquier algoritmo, y además permitir cualquier implementación a nivel de *BackEnd*. Con esta arquitectura se valida que el esquema `dmcc-schema` puede ser utilizado por proveedores o en entornos como este, para desplegar servicios de esta tipología.

En el tercer objetivo se ha desarrollado una propuesta de *Broker* llamado *BrokerMD* para la agregación, selección e intermediación de servicios de minería de datos en Cloud Computing. Se ha conseguido desplegar un servicio de *Brokering* a través del cual los usuarios pueden modelar un flujo de trabajo de minería de datos, siendo el *Broker* el encargado, por ejemplo, de seleccionar el modo de ejecutar el trabajo desde varios proveedores. Además, no soló sirve como herramienta de selección, también ofrece capacidades de agregación e intermediación. La puesta en marcha de todas las funcionalidades del *Broker* se ha alcanzado de forma satisfactoria.

Para confirmar que los tres objetivos perseguidos pueden ser trasladados a entornos y problema reales, tres casos de uso ponen en valor la eficacia y la eficiencia de la definición de servicios de minería de datos con tecnología semántica en plataformas de Cloud Computing. En todos los casos de uso se ha ofrecido una solución al problema de la definición de servicios en áreas como la lógica difusa, las series temporales o el procesamiento de datos en experimentos de HEP, aportando una solución que es portable, flexible e interoperable desde y hacia entornos basados en Cloud Computing. En el caso de uso de HEP, se ha conseguido diseñar un vocabulario gracias al cual describir de forma completa flujos de trabajo para la monitorización de la calidad de los datos en este tipo de entornos. El caso de uso de series temporales, ofrece un modelo para definición de análisis de este tipo de problemas que abarca prácticamente la totalidad del estudio de series temporales, con las ventajas de portabilidad o reproducibilidad en entornos de Cloud. El último caso de uso se centra en los sistemas difusos y ofrece un vocabulario y esquema para poder describir servicios difusos completos en entornos de Cloud Computing.

En el desarrollo del trabajo de la tesis, han ido apareciendo diferentes líneas de investigación y posibles trabajos futuros que se detallan en los siguientes párrafos.

Como posible línea de investigación futura sería interesante aprovechar la

definición de servicios de minería de datos realizada con `dmcc-schema`, para incluir el modelado de otras técnicas de Inteligencia Computacional, como la computación evolutiva o las redes neuronales. Con ello se conseguiría diseñar una aproximación semántica para la descripción de este tipo de técnicas de Inteligencia Artificial, para su despliegue como servicio dentro de entornos de Cloud Computing, con todas las ventajas que implica.

Otra acción futura consistiría en la creación de una plataforma de tipo «Market Place» que permitiese a los proveedores de servicios de Cloud Computing y usuarios individuales publicar sus servicios de minería de datos utilizando el esquema desarrollado `dmcc-schema`, e incluyendo todos los detalles del servicio, desde la experimentación hasta los costes, por ejemplo. Con ello se propone un mercado de algoritmos y paquetes de algoritmos de minería de datos listos para ser utilizados desde Cloud Computing.

De forma adicional, como trabajo futuro dentro de la arquitectura de despliegue desarrollada con OC^{2}DM, sería interesante adaptar parte del diseño e implementación para incluir capacidades de ejecución de funciones en Cloud siguiendo el modelo *serverless*, utilizando por ejemplo *OpenWhisk*[1], con lo que se daría un paso más en la abstracción de servicios. También dentro de este contexto, otra mejora sustancial en cuando a las posibilidades de escalado de la plataforma, sería la posibilidad del despliegue de un orquestador de contenedores (*Docker Swarm* o *Kubernetes*) en lugar de máquinas virtuales para el soporte de computación de los servicios, de modo que la gestión de los recursos de la infraestructura sea mucho más eficiente que con el modo actual.

Para finalizar y como línea de investigación futura sobre el desarrollo de *BrokerMD* y su aplicación al caso de uso de CERN, una propuesta muy interesante, y que se está poniendo en práctica, consistiría en integrar un servicio de Brokering en Cloud para los flujos de trabajo de procesamiento de datos, que permita a los científicos diseñar análisis de datos sin preocuparse por la infraestructura donde estos se ejecuten. Con ello, por ejemplo, sería posible integrar algoritmos que tengan implementaciones en diferentes arquitecturas (para GRIDs, Clústers, GPUs, FPGAs, etc) como servicios, de modo que el *Broker* pueda gestionar, componer e intermediar por el usuario de forma inteligente todos esos recursos. En este entorno, el caso de uso relativo a los sistemas difusos como servicio en Cloud Computing tendría una utilidad práctica importante en los sistemas DAQ de experimentos en HEP, dado que podría mejorar la fiabilidad, la monitorización y la detección de

[1] `https://openwhisk.apache.org/`

fallos (lo que se llama *Autonomous-DAQ*) de forma eficiente para millones de dispositivos conectados, ofreciendo un entorno basado en Cloud Computing para servicios difusos.

Bibliografía

[Acampora et al., 2013] Acampora, G., Loia, V., Lee, C.-S., and Wang, M.-H. (2013). *On the power of fuzzy markup language*. Springer.

[Aghabozorgi et al., 2015] Aghabozorgi, S., Shirkhorshidi, A. S., and Wah, T. Y. (2015). Time-series clustering – A decade review. *Information Systems*, 53:16–38.

[Ahmed et al., 2010] Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621.

[Alcalá-Fdez and Alonso, 2016] Alcalá-Fdez, J. and Alonso, J. M. (2016). A Survey of Fuzzy Systems Software: Taxonomy, Current Research Trends, and Prospects. *IEEE Transactions on Fuzzy Systems*, 24(1):40–56.

[Aljawarneh et al., 2016] Aljawarneh, S., Radhakrishna, V., Kumar, P. V., and Janaki, V. (2016). A similarity measure for temporal pattern discovery in time series data generated by IoT. In *Engineering & MIS (ICEMIS), International Conference on*, pages 1–4. IEEE.

[Alpay and Erdem, 2018] Alpay, Ö. and Erdem, E. (2018). The Control of Greenhouses Based on Fuzzy Logic Using Wireless Sensor Networks. *International Journal of Computational Intelligence Systems*, 12(1):190–203.

[Amato et al., 2013] Amato, A., Di Martino, B., and Venticinque, S. (2013). Multi-objective genetic algorithm for multi-cloud brokering. In *European Conference on Parallel Processing*, pages 55–64. Springer.

[Ambulkar and Borkar, 2012] Ambulkar, B. and Borkar, V. (2012). Data mining in cloud computing. In *MPGI National Multi Conference*, volume 2012.

[Anastasi et al., 2017] Anastasi, G. F., Carlini, E., Coppola, M., and Dazzi, P. (2017). Qos-aware genetic cloud brokering. *Future Generation Computer Systems*, 75:1–13.

[Apduhan et al., 2015] Apduhan, B., Younas, M., and Uchibayashi, T. (2015). Improving Reliability and Availability of IaaS Services in Hybrid Clouds. In *International Conference on Computational Science and Its Applications*, pages 557–568. Springer.

[Armstrong and Collopy, 1992] Armstrong, J. S. and Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. *International journal of forecasting*, 8(1):69–80.

[Arnold, Joe, 2014] Arnold, Joe (2014). *Openstack swift: Using, administering, and developing for swift object storage*. OReilly Media, Inc.

[Assunção et al., 2015] Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., and Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79:3–15.

[Barry, 2003] Barry, D. K. (2003). *Web services, service-oriented architectures, and cloud computing*. Elsevier.

[Beckett et al., 2014] Beckett, D., Berners-Lee, T., Prud'hommeaux, E., and Carothers, G. (2014). RDF 1.1 Turtle. *World Wide Web Consortium*.

[Bellwood et al., 2002] Bellwood, T., Clément, L., Ehnebuske, D., Hately, A., Hondo, M., Husband, Y. L., Januszewski, K., Lee, S., McKee, B., Munter, J., et al. (2002). UDDI Version 3.0. *Published specification, Oasis*, 5:16–18.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific American*, 284(5):28–37.

[Berthold, Michael R and Cebron, Nicolas and Dill, Fabian and Gabriel, Thomas R and Köt
Berthold, Michael R and Cebron, Nicolas and Dill, Fabian and Gabriel, Thomas R and Kötter, Tobias and Meinl, Thorsten and Ohl, Peter and Thiel, Kilian and Wiswedel, Bernd (2009). KNIME-the Konstanz information miner: version 2.0 and beyond. *AcM SIGKDD explorations Newsletter*, 11(1):26–31.

[Bezanson et al., 2017] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

[Bhardwaj et al., 2010] Bhardwaj, S., Jain, L., and Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology*, 2(1):60–63.

[Binz et al., 2013] Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., and Wagner, S. (2013). OpenTOSCA – a runtime for TOSCA-based cloud applications. In *International Conference on Service-Oriented Computing*, pages 692–695. Springer.

[Binz et al., 2012] Binz, T., Breiter, G., Leyman, F., and Spatzier, T. (2012). Portable cloud services using tosca. *IEEE Internet Computing*, 16(3):80–85.

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22.

[Bizer et al., 2011] Bizer, C., Heath, T., and Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic Services, Interoperability and Web A pplications: emerging concepts*, pages 205–227. IGI Global.

[Bontempi et al., 2012] Bontempi, G., Taieb, S. B., and Le Borgne, Y.-A. (2012). Machine learning strategies for time series forecasting. In *European Business Intelligence Summer School*, pages 62–77. Springer.

[Brandes et al., 2001] Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., and Marshall, M. S. (2001). GraphML progress report structural layer proposal. In *International Symposium on Graph Drawing*, pages 501–512. Springer.

[Bray et al., 1997] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (1997). Extensible markup language (XML). *World Wide Web Journal*, 2(4):27–66.

[Brun and Rademakers, 1997] Brun, R. and Rademakers, F. (1997). ROOT — an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1-2):81–86.

[Cavus, 2010] Cavus, N. (2010). The evaluation of learning management systems using an artificial intelligence fuzzy logic algorithm. *Advances in Engineering Software*, 41(2):248 – 254.

[Cetinsoy et al., 2016] Cetinsoy, A., Martin, F. J., Ortega, J. A., and Petersen, P. (2016). The past, present, and future of machine learning apis. In *Conference on Predictive APIs and Apps*, pages 43–49.

[Chamorro et al., 2016] Chamorro, L., López-Pires, F., and Barán, B. (2016). A genetic algorithm for dynamic cloud application brokerage. In *Cloud Engineering (IC2E), 2016 IEEE International Conference on*, pages 131–134. IEEE.

[Chatfield, 2016] Chatfield, C. (2016). *The analysis of time series: an introduction*. CRC press.

[Chen et al., 2015a] Chen, F., Deng, P., Wan, J., Zhang, D., Vasilakos, A. V., and Rong, X. (2015a). Data mining for the internet of things: literature review and challenges. *International Journal of Distributed Sensor Networks*, 11(8):431047.

[Chen et al., 2014] Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile networks and applications*, 19(2):171–209.

[Chen et al., 2015b] Chen, Z., Zuo, W., Hu, Q., and Lin, L. (2015b). Kernel sparse representation for time series classification. *Information Sciences*, 292:15–26.

[Chichin et al., 2014] Chichin, S., Chhetri, M. B., Vo, Q. B., Kowalczyk, R., and Stepniak, M. (2014). Smart cloud marketplace-agent-based platform for trading cloud services. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 03*, pages 388–395. IEEE Computer Society.

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web Service Definition Language (WSDL). Technical report, World Wide Web Consortium.

[Church, 2013] Church, E. D. (2013). LArSoft: a software package for liquid argon time projection drift chambers. *arXiv preprint arXiv:1311.6774*.

[Collaboration et al., 2010] Collaboration, C. et al. (2010). CMS data processing workflows during an extended cosmic ray run.

[Copil et al., 2014] Copil, G., Moldovan, D., Truong, H.-L., and Dustdar, S. (2014). On controlling cloud services elasticity in heterogeneous clouds. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 573–578. IEEE Computer Society.

[Correa Publio, 2017] Correa Publio, D. E. (2017). MLSchema: machine learning and data mining ontologies. http://ml-schema.github.io/documentation/ML%20Schema.html. Accessed: 2018-05-20.

[Cortez, 2010a] Cortez, P. (2010a). Data mining with neural networks and support vector machines using the R/rminer tool. In *Industrial Conference on Data Mining*, pages 572–583. Springer.

[Cortez, 2010b] Cortez, P. (2010b). Data mining with neural networks and support vector machines using the r/rminer tool. In *Industrial Conference on Data Mining*, pages 572–583. Springer.

[Dastjerdi et al., 2010] Dastjerdi, A. V., Tabatabaei, S. G. H., and Buyya, R. (2010). An effective architecture for automated appliance management system applying ontology-based cloud discovery. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 104–112. IEEE Computer Society.

[Data et al., 2013] Data, B., Catlett, C., et al. (2013). A cloud framework for Big Data analytics workflows on Azure. *Cloud Computing and Big Data*, 23:182.

[De et al., 2014] De, K., Golubkov, D., Klimentov, A., Potekhin, M., Vaniachine, A., Collaboration, A., et al. (2014). Task management in the new ATLAS production system. In *Journal of Physics: Conference Series*, volume 513, page 032078. IOP Publishing.

[De Gooijer and Kumar, 1992] De Gooijer, J. G. and Kumar, K. (1992). Some recent developments in non-linear time series modelling, testing, and forecasting. *International Journal of Forecasting*, 8(2):135–156.

[de Souza et al., 2014] de Souza, M., dos Santos, F., de Soto, A. R., and Vahldick, A. (2014). Fuzzystudio: A web tool for modeling and simulation of fuzzy systems. In *2014 Brazilian Conference on Intelligent Systems*, pages 306–311. IEEE.

[Deb, 2014] Deb, K. (2014). Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[Demšar et al., 2013] Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., et al. (2013). Orange: data mining toolbox in Python. *The Journal of Machine Learning Research*, 14(1):2349–2353.

[Demšar et al., 2013] Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., and Zupan, B. (2013). Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353.

[Dillon et al., 2010] Dillon, T., Wu, C., and Chang, E. (2010). Cloud computing: issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 27–33. Ieee.

[Domingue et al., 2005] Domingue, J., Roman, D., and Stollberg, M. (2005). Web service modeling ontology (wsmo)-an ontology for semantic web services.

[Dominici et al., 2002] Dominici, F., McDermott, A., Zeger, S. L., and Samet, J. M. (2002). On the use of generalized additive models in time-series studies of air pollution and health. *American journal of epidemiology*, 156(3):193–203.

[Doukas and Maglogiannis, 2012] Doukas, C. and Maglogiannis, I. (2012). Bringing IoT and cloud computing towards pervasive healthcare. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 922–926. IEEE.

[D'souza and Wills, 1998] D'souza, D. F. and Wills, A. C. (1998). *Objects, components, and frameworks with UML: the catalysis approach*, volume 1. Addison-Wesley Reading.

[Elvesæter et al., 2011] Elvesæter, B., Carrez, C., Mohagheghi, P., Berre, A.-J., Johnsen, S. G., and Solberg, A. (2011). Model-driven service engineering with SoaML. In *Service Engineering*, pages 25–54. Springer.

[Elvesæter et al., 2010] Elvesæter, B., Panfilenko, D., Jacobi, S., and Hahn, C. (2010). Aligning business and IT models in service-oriented architectures using BPMN and SoaML. In *Proceedings of the First International Workshop on Model-Driven Interoperability*, pages 61–68. ACM.

[Engle, 2001] Engle, R. (2001). GARCH 101: The use of ARCH/GARCH models in applied econometrics. *Journal of economic perspectives*, 15(4):157–168.

[Ericsson, 2018] Ericsson (2018). Capture the value in IoT - Ericsson. url=https://www.ericsson.com/en/internet-of-things/audience-page/capture-the-value-in-iot.

[Esteves et al., 2016] Esteves, D., Lawrynowicz, A., Panov, P., Soldatova, L., Soru, T., and Vanschoren, J. (2016). ML Schema Core Specification. Technical report, World Wide Web Consortium.

[Esteves et al., 2015a] Esteves, D., Moussallem, D., Neto, C. B., Lehmann, J., Cavalcanti, M. C. R., and Duarte, J. C. (2015a). Interoperable Machine Learning Metadata using MEX. In *International Semantic Web Conference (Posters & Demos)*.

[Esteves et al., 2015b] Esteves, D., Moussallem, D., Neto, C. B., Soru, T., Usbeck, R., Ackermann, M., and Lehmann, J. (2015b). MEX vocabulary: a lightweight interchange format for machine learning experiments. In *Proceedings of the 11th International Conference on Semantic Systems*, pages 169–176. ACM.

[et al., 2007] et al., M. R. B. (2007). KNIME: The Konstanz Information Miner. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer.

[Fernández et al., 2014] Fernández, A., del Río, S., López, V., Bawakid, A., del Jesus, M. J., Benítez, J. M., and Herrera, F. (2014). Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(5):380–409.

[Ferrer et al., 2012] Ferrer, A. J., HernáNdez, F., Tordsson, J., Elmroth, E., Ali-Eldin, A., Zsigri, C., Sirvent, R., Guitart, J., Badia, R. M., Djemame, K., et al. (2012). OPTIMIS: A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, 28(1):66–77.

[Fredj et al., 2008] Fredj, M., Georgantas, N., Issarny, V., and Zarras, A. (2008). Dynamic service substitution in service-oriented architectures. In *2008 IEEE Congress on Services-Part I*, pages 101–104. IEEE.

[García et al., 2016a] García, J. M., Fernandez, P., Pedrinaci, C., Resinas, M., Cardoso, J., and Ruiz-Cortés, A. (2016a). Modeling service level agreements with linked USDL agreement. *IEEE Transactions on Services Computing*, 10(1):52–65.

[García et al., 2015] García, S., Luengo, J., and Herrera, F. (2015). *Data preprocessing in data mining*. Springer.

[García et al., 2016b] García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., and Herrera, F. (2016b). Big data preprocessing: methods and prospects. *Big Data Analytics*, 1(1):9.

[Gartner Inc., 2016] Gartner Inc. (2016). Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016. Technical report, Gartner.

[Ghazouani and Slimani, 2017] Ghazouani, S. and Slimani, Y. (2017). A survey on cloud service description. *Journal of Network and Computer Applications*, 91:61–74.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[Golubkov et al., 2012] Golubkov, D., Kersevan, B., Klimentov, A., Minaenko, A., Nevski, P., Vaniachine, A., and Walker, R. (2012). ATLAS Grid Data Processing: system evolution and scalability. In *Journal of Physics: Conference Series*, volume 396, page 032049. IOP Publishing.

[Green et al., 2012] Green, C., Kowalkowski, J., Paterno, M., Fischler, M., Garren, L., and Lu, Q. (2012). The art framework. In *Journal of Physics: Conference Series*, volume 396, page 022020. IOP Publishing.

[Grivas et al., 2010] Grivas, S. G., Kumar, T. U., and Wache, H. (2010). Cloud broker: Bringing intelligence into the cloud. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 544–545. IEEE.

[Gruber, 1993] Gruber, T. (1993). What is an Ontology.

[Gruber, 1995] Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928.

[Guazzelli et al., 2009a] Guazzelli, A., Stathatos, K., and Zeller, M. (2009a). Efficient deployment of predictive analytics through open standards and cloud computing. *ACM SIGKDD Explorations Newsletter*, 11(1):32–38.

[Guazzelli et al., 2009b] Guazzelli, A., Zeller, M., Lin, W.-C., Williams, G., et al. (2009b). PMML: An open standard for sharing models. *The R Journal*, 1(1):60–65.

[Gubbi et al., 2013] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660.

[Guha et al., 2015] Guha, R. V., Brickley, D., and MacBeth, S. (2015). Schema.org: Evolution of structured data on the web. *Queue*, 13(9):10:10–10:37.

[Gutsche et al., 2017] Gutsche, O., Cremonesi, M., Elmer, P., Jayatilaka, B., Kowalkowski, J., Pivarski, J., Sehrish, S., Surez, C. M., Svyatkovskiy, A., and Tran, N. (2017). Big Data in HEP: A comprehensive use case study. *arXiv preprint arXiv:1703.04171*.

[Hadley, 2006] Hadley, M. J. (2006). Web application description language (wadl). *World Wide Web Consortium*.

[Hall et al., 2009a] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009a). The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

[Hall et al., 2009b] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009b). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

[Harris and Sollis, 2003] Harris, R. and Sollis, R. (2003). *Applied time series modelling and forecasting*. Wiley.

[Hashem et al., 2015] Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., and Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information systems*, 47:98–115.

[Hepp, 2008] Hepp, M. (2008). Goodrelations: An ontology for describing products and services offers on the web. *Knowledge Engineering: Practice and Patterns*, pages 329–346.

[Ho, 1998] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844.

[Hoey and Dassi, 2014] Hoey, B. and Dassi, N. (2014). System and methods for real-time detection, correction, and transformation of time series data. US Patent App. 13/916,513.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press.

[Hothorn, 2018] Hothorn, T. (2018). CRAN task view: Machine learning & statistical learning. url=https://cran.r-project.org/web/views/MachineLearning.html.

[Hyndman and Athanasopoulos, 2018] Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

[Javadi et al., 2012] Javadi, B., Abawajy, J., and Buyya, R. (2012). Failure-aware resource provisioning for hybrid cloud infrastructure. *Journal of parallel and distributed computing*, 72(10):1318–1331.

[Javed et al., 2016] Javed, B., Bloodsworth, P., Rasool, R. U., Munir, K., and Rana, O. (2016). Cloud market maker: An automated dynamic pricing marketplace for cloud users. *Future Generation Computer Systems*, 54:52–67.

[Johanyák et al., 2006] Johanyák, Z. C., Tikk, D., Kovács, S., and Wong, K. W. (2006). Fuzzy rule interpolation Matlab toolbox-FRI toolbox. In *2006 IEEE International Conference on Fuzzy Systems*, pages 351–357. IEEE.

[JoSEP et al., 2010] JoSEP, A. D., KAtz, R., KonWinSKi, A., Gunho, L., PAttERSon, D., and RABKin, A. (2010). A view of cloud computing. *Communications of the ACM*, 53(4).

[Jovic et al., 2014] Jovic, A., Brkic, K., and Bogunovic, N. (2014). An overview of free software tools for general data mining. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, pages 1112–1117. IEEE.

[Jrad et al., 2013] Jrad, F., Tao, J., and Streit, A. (2013). A broker-based framework for multi-cloud workflows. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 61–68. ACM.

[Juric, 2007] Juric, M. B. (2007). *SOA Approach to Integration: XML, Web services, ESB, and BPEL in real-world SOA projects*. Packt Publishing Ltd.

[Kannan et al., 2014] Kannan, D., de Sousa Jabbour, A. B. L., and Jabbour, C. J. C. (2014). Selecting green suppliers based on GSCM practices: Using fuzzy TOPSIS applied to a Brazilian electronics company. *European Journal of Operational Research*, 233(2):432–447.

[Kertész et al., 2014] Kertész, A., Kecskemeti, G., and Brandic, I. (2014). An interoperable and self-adaptive approach for sla-based service virtualization in heterogeneous cloud environments. *Future Generation Computer Systems*, 32:54–68.

[Kholod et al., 2016] Kholod, I., Kuprianov, M., and Petukhov, I. (2016). Parallel and distributed data mining in cloud. In *Industrial Conference on Data Mining*, pages 349–362. Springer.

[Kim et al., 2015] Kim, W.-J., Kang, D.-K., Kim, S.-H., and Youn, C.-H. (2015). Cost adaptive vm management for scientific workflow application in mobile cloud. *Mobile Networks and Applications*, 20(3):328–336.

[Klimentov et al., 2015] Klimentov, A., Buncic, P., De, K., Jha, S., Maeno, T., Mount, R., Nilsson, P., Oleynik, D., Panitkin, S., Petrosyan, A., et al. (2015). Next generation workload management system for big data on heterogeneous distributed computing. In *Journal of Physics: Conference Series*, volume 608, page 012040. IOP Publishing.

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, W3C.

[Kona et al., 2009] Kona, S., Bansal, A., Simon, L., Mallya, A., and Gupta, G. (2009). Usdl: a service-semantics description language for automatic service discovery and composition. *International Journal of Web Services Research (IJWSR)*, 6(1):20–48.

[Kopeckỳ et al., 2007] Kopeckỳ, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). SaWSDL: Semantic annotations for WSDL and XML schema. *IEEE Internet Computing*, 11(6).

[Kranjc et al., 2012] Kranjc, J., Podpečan, V., and Lavrač, N. (2012). Clowdflows: a cloud based scientific workflow platform. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 816–819. Springer.

[Kranjc et al., 2015] Kranjc, J., Smailović, J., Podpečan, V., Grčar, M., Žnidaršič, M., and Lavrač, N. (2015). Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform. *Information Processing & Management*, 51(2):187–203.

[Lanthaler and Gütl, 2012] Lanthaler, M. and Gütl, C. (2012). On using JSON-LD to create evolvable RESTful services. In *Proceedings of the Third International Workshop on RESTful Design*, pages 25–32. ACM.

[Lara et al., 2004] Lara, R., Roman, D., Polleres, A., and Fensel, D. (2004). A conceptual comparison of WSMO and OWL-S. In *European Conference on Web Services*, pages 254–269. Springer.

[Leach et al., 2005] Leach, P., Mealling, M., and Salz, R. (2005). A universally unique identifier (UUID) URN namespace. Technical report, Microsoft.

[Li and Du, 2013] Li, X. and Du, J. (2013). Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing. *IET Information Security*, 7(1):39–50.

[Liaw et al., 2002] Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

[Lin et al., 2018] Lin, D., Squicciarini, A. C., Dondapati, V. N., and Sundareswaran, S. (2018). A cloud brokerage architecture for efficient cloud service selection. *IEEE Transactions on Services Computing*, pages 1–1.

[Liu, 2016] Liu, L. (2016). Services computing: from cloud services, mobile services to internet of services. *IEEE Transactions on Services Computing*, 9(5):661–663.

[López et al., 1997] López, D., Moreno, F., Barriga, A., and Sánchez-Solano, S. (1997). XFL: a language for the definition of fuzzy systems. In *Proceedings of 6th International Fuzzy Systems Conference*, volume 3, pages 1585–1591. IEEE.

[Makridakis and Hibon, 1997] Makridakis, S. and Hibon, M. (1997). ARMA models and the Box–Jenkins methodology. *Journal of Forecasting*, 16(3):147–163.

[Maleshkova et al., 2010] Maleshkova, M., Pedrinaci, C., Domingue, J., Alvaro, G., and Martinez, I. (2010). Using semantics for automating the

authentication of web APIs. *The Semantic Web–ISWC 2010*, pages 534–549.

[Marozzo et al., 2016] Marozzo, F., Talia, D., and Trunfio, P. (2016). A workflow management system for scalable data mining on clouds. *IEEE Transactions on Services Computing*.

[Marozzo et al., 2018] Marozzo, F., Talia, D., and Trunfio, P. (2018). A workflow management system for scalable data mining on clouds. *IEEE Transactions on Services Computing*, 11(3):480–492.

[Marr, 2016] Marr, B. (2016). Big data overload: Why most companies can't deal with the data explosion. url=https://goo.gl/VZbe4R.

[Marston et al., 2011] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., and Ghalsasi, A. (2011). Cloud computing—The business perspective. *Decision support systems*, 51(1):176–189.

[Martin et al., 2004] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). OWL-S: Semantic markup for web services. *W3C member submission*, 22(4).

[Martin et al., 2007] Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D. L., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to web services with OWL-S. *World Wide Web*, 10(3):243–277.

[McCulloch, 2017] McCulloch, J. (2017). Fuzzycreator: A python-based toolkit for automatically generating and analysing data-driven fuzzy sets. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE.

[McKinney et al., 2010] McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

[Mell, Peter et al., 2011] Mell, Peter et al. (2011). The NIST definition of cloud computing.

[Meng et al.,] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. MLlib: Machine learning in apache spark.

[Michalewicz, 1996] Michalewicz, Z. (1996). Evolution strategies and other methods. In *Genetic Algorithms+ Data Structures= Evolution Programs*, pages 159–177. Springer.

[Miles and Bechhofer, 2009] Miles, A. and Bechhofer, S. (2009). SKOS simple knowledge organization system reference. *W3C recommendation*, 18:W3C.

[Mitchell et al., 2003] Mitchell, G. G., O'Donoghue, D., Barnes, D., and McCarville, M. (2003). Generepair-a repair operator for genetic algorithms. In *Proc. Genetic and Evolutionary Computation Conference (GECCO) Late Breaking Papers*, pages 235–239.

[Montero et al., 2014] Montero, P., Vilar, J. A., et al. (2014). Tsclust: An R package for time series clustering. *Journal of Statistical Software*, 62(1):1–43.

[Montgomery et al., 2015] Montgomery, D. C., Jennings, C. L., and Kulahci, M. (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons.

[Moreno-Velo et al., 2012] Moreno-Velo, F. J., Barriga, A., Sánchez-Solano, S., and Baturone, I. (2012). XFSML: An XML-based modeling language for fuzzy systems. In *2012 IEEE International Conference on Fuzzy Systems*, pages 1–8. IEEE.

[Moreno Velo et al., 2001] Moreno Velo, F. J., Baturone Castillo, M. I., Sánchez Solano, S., and Barriga Barros, Á. (2001). Xfuzzy 3.0: a development environment for fuzzy systems. *Proceeding of the International Conference in Fuzzy Logic and Technology*.

[Mueen and Keogh, 2016] Mueen, A. and Keogh, E. (2016). Extracting optimal performance from dynamic time warping. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2129–2130. ACM.

[Nair et al., 2010] Nair, S. K., Porwal, S., Dimitrakos, T., Ferrer, A. J., Tordsson, J., Sharif, T., Sheridan, C., Rajarajan, M., and Khan, A. U. (2010). Towards secure cloud bursting, brokerage and aggregation. In *2010 eighth IEEE European conference on web services*, pages 189–196. IEEE.

[Networking, 2016] Networking, C. V. (2016). Cisco global cloud index: Forecast and methodology, 2016-2021. *Cisco Public, San Jose*.

[Newcomer and Lomow, 2005] Newcomer, E. and Lomow, G. (2005). *Understanding SOA with Web services*. Addison-Wesley.

[Owen et al., 2011] Owen, S., Anil, R., Dunning, T., and Friedman, E. (2011). *Mahout in Action*. Manning Publications Co., Greenwich, CT, USA.

[Pandeeswari and Kumar, 2016] Pandeeswari, N. and Kumar, G. (2016). Anomaly detection system in cloud environment using fuzzy clustering based ANN. *Mobile Networks and Applications*, 21(3):494–505.

[Panov et al., 2008] Panov, P., Džeroski, S., and Soldatova, L. (2008). Ontodm: An ontology of data mining. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 752–760. IEEE.

[Paraiso et al., 2016] Paraiso, F., Merle, P., and Seinturier, L. (2016). soCloud: a service-oriented component-based PaaS for managing portability, provisioning, elasticity, and high availability across multiple clouds. *Computing*, 98(5):539–565.

[Parra-Royon et al., 2018] Parra-Royon, M., Atemezing, G., and Benítez, J. M. (2018). Data Mining definition services in Cloud Computing with Linked Data. *arXiv preprint arXiv:1806.06826*.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Pedrinaci et al., 2014] Pedrinaci, C., Cardoso, J., and Leidig, T. (2014). Linked USDL: a vocabulary for web-scale service trading. In *European Semantic Web Conference*, pages 68–82. Springer.

[Peng, 2011] Peng, R. D. (2011). Reproducible research in computational science. *Science*, 334(6060):1226–1227.

[Potekhin, 2017] Potekhin, M. (2017). The protoDUNE-SP experiment and its prompt processing system. *PoS*, EPS-HEP2017:513.

[Precup and Hellendoorn, 2011] Precup, R.-E. and Hellendoorn, H. (2011). A survey on industrial applications of fuzzy control. *Computers in industry*, 62(3):213–226.

[Prud et al., 2006] Prud, E., Seaborne, A., et al. (2006). Sparql query language for rdf. *W3C*.

[Purohit et al., 2017] Purohit, P., Apoorva, D., Lathashree, P., et al. (2017). Big data in cloud computing. *International Journal of Advance Research, Ideas and Innovations in Technology*, 3(3):1312–1318.

[Rada-vilela, 2013] Rada-vilela, J. (2013). Fuzzylite a fuzzy logic control library in C++.

[Rangra and Bansal, 2014] Rangra, K. and Bansal, K. (2014). Comparative study of data mining tools. *International journal of advanced research in computer science and software engineering*, 4(6).

[Reeves, 2003] Reeves, C. (2003). Genetic algorithms. In *Handbook of metaheuristics*, pages 55–82. Springer.

[Reklaitis, 1996] Reklaitis, G. V. (1996). Overview of scheduling and planning of batch process operations. In *Batch processing systems engineering*, pages 660–705. Springer.

[Riza et al., 2015] Riza, L., Bergmeir, C., Herrera, F., and Benítez, J. M. (2015). frbs: Fuzzy Rule-Based Systems for Classification and Regression in R. *Journal of Statistical Software, Articles*, 65(6):1–30.

[Salcedo-Sanz, 2009] Salcedo-Sanz, S. (2009). A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer science review*, 3(3):175–192.

[Serra and Arcos, 2014] Serra, J. and Arcos, J. L. (2014). An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314.

[Shadroo and Rahmani, 2018] Shadroo, S. and Rahmani, A. M. (2018). Systematic survey of big data and data mining in internet of things. *Computer Networks*, 139:19–47.

[Sheth et al., 2007] Sheth, A. P., Gomadam, K., and Lathem, J. (2007). Sarest: Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, 11(6):91–94.

[Shvachko et al., 2010] Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pages 1–10. Ieee.

[Simarro et al., 2011] Simarro, J. L. L., Moreno-Vozmediano, R., Montero, R. S., and Llorente, I. M. (2011). Dynamic placement of virtual machines

for cost optimization in multi-cloud environments. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 1–7. IEEE.

[SureshKumar and Varalakshmi, 2017] SureshKumar, M. and Varalakshmi, P. (2017). A heuristic approach for web service selection with cost optimization. In *Computing and Communications Technologies (ICCCT), 2017 2nd International Conference on*, pages 374–382. IEEE.

[Tafti et al., 2017] Tafti, A. P., LaRose, E., Badger, J. C., Kleiman, R., and Peissig, P. (2017). Machine learning-as-a-service and its application to medical informatics. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 206–219. Springer.

[Taheriyan et al., 2012] Taheriyan, M., Knoblock, C. A., Szekely, P., and Ambite, J. L. (2012). Rapidly integrating services into the Linked Data cloud. In *International Semantic Web Conference*, pages 559–574. Springer.

[Talia, 2013] Talia, D. (2013). Clouds for scalable big data analytics. *Computer*, 46(5):98–101.

[Team, 2000] Team, R. C. (2000). R language definition. *Vienna, Austria: R foundation for statistical computing*.

[Tian and Murata, 2016] Tian, J. and Murata, T. (2016). Robust Scheduling for Resource Constraint Scheduling Problem by Two-Stage GA and MMEDA. In *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 1042–1047.

[Tiegelkamp and John, 1995] Tiegelkamp, M. and John, K.-H. (1995). *IEC 61131-3: Programming industrial automation systems*. Springer.

[Übeylı and Güler, 2004] Übeylı, E. D. and Güler, I. (2004). Spectral analysis of internal carotid arterial Doppler signals using FFT, AR, MA, and ARMA methods. *Computers in biology and medicine*, 34(4):293–306.

[VanderPlas, 2016] VanderPlas, J. (2016). gatspy: General tools for Astronomical Time Series in Python. *Astrophysics Source Code Library*.

[Vanschoren and Soldatova, 2010] Vanschoren, J. and Soldatova, L. (2010). Exposé: An ontology for data mining experiments. In *International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010)*, pages 31–46.

[Vavilapalli et al., 2013] Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al. (2013). Apache Hadoop YARN: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM.

[Vitvar et al., 2008] Vitvar, T., Kopeckỳ, J., Viskova, J., and Fensel, D. (2008). WSMO-lite annotations for web services. In *European Semantic Web Conference*, pages 674–689. Springer.

[Vuong et al., 2006] Vuong, P. T., Madni, A. M., and Vuong, J. B. (2006). VHDL implementation for a fuzzy logic controller. In *2006 World Automation Congress*, pages 1–8. IEEE.

[Wagner, 2013] Wagner, C. (2013). Juzzy-a java based toolkit for type-2 fuzzy logic. In *2013 IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ)*, pages 45–52. IEEE.

[Wakefield et al., 2008] Wakefield, S., Gutsche, O., Evans, D., Hassan, A., Hufnagel, D., Mason, D., Metson, S., Miller, M., Mohapatra, A., and van Lingen, F. (2008). Large Scale Job Management and Experience in Recent Data Challenges within the LHC CMS experiment. In *Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research*.

[Wang et al., 2013] Wang, W., Niu, D., Li, B., and Liang, B. (2013). Dynamic cloud resource reservation via cloud brokerage. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 400–409. IEEE.

[Wang et al., 2015] Wang, W., Niu, D., Liang, B., and Li, B. (2015). Dynamic cloud instance acquisition via IaaS cloud brokerage. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1580–1593.

[Weibel et al., 1998] Weibel, S., Kunze, J., Lagoze, C., and Wolf, M. (1998). Dublin core metadata for resource discovery. Technical report, Dublin Core.

[Weigend, 2018] Weigend, A. S. (2018). *Time series prediction: forecasting the future and understanding the past*. Routledge.

[Xu et al., 2014] Xu, X., Huang, S., Chen, Y., Browny, K., Halilovicy, I., and Lu, W. (2014). TSAaaS: Time series analytics as a service on IoT. In *Web Services (ICWS), 2014 IEEE International Conference On*, pages 249–256. IEEE.

[Yang et al., 2017] Yang, C., Huang, Q., Li, Z., Liu, K., and Hu, F. (2017). Big data and cloud computing: innovation opportunities and challenges. *International Journal of Digitvargal Earth*, 10(1):13–53.

[Yang et al., 2015] Yang, J., Nguyen, M. N., San, P. P., Li, X., and Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pages 3995–4001.

[Yangui et al., 2014] Yangui, S., Marshall, I.-J., Laisne, J.-P., and Tata, S. (2014). Compatibleone: The open source cloud broker. *Journal of Grid Computing*, 12(1):93–109.

[Zadeh, 1965] Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3):338–353.

[Zaharia et al., 2010] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95.

[Zhang, 2003] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175.

[Zhang et al., 2010] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.

[Zheng et al., 2013] Zheng, Z., Zhu, J., and Lyu, M. R. (2013). Service-generated big data and big data-as-a-service: an overview. In *IEEE International Congress on Big Data (BigData Congress)*, pages 403–410. IEEE.

Índice de figuras

Índice de tablas

Índice alfabético