UNIVERSIDAD DE CÓRDOBA

Departamento de Informática y Análisis Numérico

Programa de doctorado en computación avanzada, energía y plasmas

Aplicación de flujos de trabajo científicos en dominios con procesamiento intensivo de datos

Application of scientific workflows in data-intensive computing domains

Memoria de Tesis presentada por

Rubén Salado Cid

como requisito para optar al grado

de Doctor en Informática

Director

Dr. José Raúl Romero Salguero

Córdoba, febrero de 2024

TITULO: *Aplicación de flujos de trabajo científicos en dominios con procesamiento intensivo de datos*

AUTOR: *Rubén Salado Cid*

Campus de Rabanales
Ctra. Nacional IV, Km. 396 A
14071 Córdoba

https://www.uco.es/ucopress/index.php/es/
ucopress@uco.es

INFORME RAZONADO DE LAS/LOS DIRECTORAS/ES DE LA TESIS

Este documento se presentará junto con el depósito de la tesis en https://moodle.uco.es/ctp3/

DOCTORANDA/O

Rubén Salado Cid

TÍTULO DE LA TESIS:

Aplicación de flujos de trabajo científicos en dominios con procesamiento intensivo de datos

INFORME RAZONADO DE LAS/LOS DIRECTORAS/ES DE LA TESIS
(se hará mención a la evolución y desarrollo de la tesis, así como a trabajos y publicaciones derivados de la misma)

En su tesis doctoral, D. Rubén Salado Cid ha abordado el estudio, desarrollo y evaluación experimental de la representación, transformación y gestión de flujos de trabajo científicos en dominios de procesamiento intensivo de datos. Inicialmente, se realizó un estudio comparativo y funcional de diversas herramientas de flujos de trabajo intensivos en datos (también conocidos como flujos de trabajo científicos). Posteriormente, se ha procedido a un análisis extenso de la literatura sobre las plataformas de desarrollo low- y no-code, identificando los fundamentos para permitir la creación de aplicaciones por parte de usuarios no expertos en desarrollo de software ("citizen developers").

La propuesta de esta tesis doctoral se ha centrado en el enfoque de la ingeniería del software dirigida por modelos (MSDE, model-driven software engineering), presentando una idea original e innovadora al trabajar con técnicas de modelado para facilitar la democratización de dominios intensivos en datos. La investigación llevada a cabo ha dado lugar a la creación de un lenguaje de modelado específico para definir estos flujos de trabajo en diferentes niveles. Usando los principios de MSDE, mediante transformaciones de modelos complejas se permite la reutilización y adaptación a las necesidades particulares de cada dominio. A su vez, esto ha permitido el desarrollo una herramienta low-code que permite a los científicos de datos crear herramientas específicas del dominio basadas en flujos de trabajo, según las necesidades del experto. Como ejemplo práctico, se presenta un caso de uso enfocado en el ámbito de la minería de datos educativos, donde los profesores y los administradores de centros educativos se convierten en científicos de datos. Esta herramienta demuestra el nivel de democratización en ciencia de datos alcanzado mediante estas técnicas, logrando una eficacia en la generación automática de herramientas mucho mayor que las propuestas existentes.

Los resultados obtenidos en la tesis han derivado en varias publicaciones, siendo la siguiente publicación la que asegura la calidad de la investigación realizada:

R. Salado-Cid, A. Vallecillo, K. Munir, J.R. Romero. "SWEL: A Domain-Specific Language for Modelling Data-Intensive Workflows". Business & Information Systems Engineering, August 2023. Springer. ISSN 1867-0202. DOI: 10.1007/s12599-023-00826-7

Esta publicación se indexa en el primer decil (Q1) de la categoría COMPUTER SCIENCE > INFORMATION SYSTEMS con un índice de impacto (en JCR) de 7,9 en 2022. Se trata de una revista extremadamente prestigiosa en su área, exigente y que reúne todos los estándares de calidad para avalar el trabajo realizado. Además, un capítulo de libro y varias publicaciones en conferencias internacionales y nacionales avalan la calidad de esta tesis doctoral. También cabe destacar las nuevas líneas de trabajo que deja abiertas a futuro, y que prometen contribuciones científicas de relevancia.

Por todo ello, la tesis doctoral de D. Rubén Salado Cid cumple con las condiciones y méritos necesarios para su defensa y presentación, destacando por su calidad científica y su potencial para influir en la evolución de las aplicaciones intensivas en datos.

Por todo ello, se autoriza la presentación de la tesis doctoral.

Córdoba, a 25 de febrero de 2024

El director,

Firmado por ROMERO SALGUERO JOSE RAUL - ***4544** el día 25/02/2024 con un

Fdo.:Dr. José Raúl Romero Salguero

La memoria titulada *"Aplicación de flujos de trabajo científicos en dominios con procesamiento intensivo de datos"*, que presenta Rubén Salado Cid para optar al grado de Doctor en el marco del programa de doctorado "Computación Avanzada, Energía y Plasmas", recopila un trabajo original de investigación realizado en el Departamento de Informática y Análisis Numérico de la Escuela Politécnica Superior de la Universidad de Córdoba.

Córdoba, febrero de 2024

El candidato:

Firmado por SALADO CID RUBEN - ***4487**
el día 28/02/2024 con un certificado
emitido por ACCVCA-120

Fdo.: Rubén Salado Cid

Esta tesis doctoral ha sido desarrollada en el marco del proyecto “EMERalD: EMERging trends in Data analysis” (**TIN2017-83445-P**) de la Convocatoria Proyectos de Excelencia 2017 de la Agencia Estatal de Investigación, subvencionado por el Ministerio de Economía, Industria y Competitividad, y fondos FEDER. También ha sido desarrollada en el marco del proyecto “INTENSE: ImproviNg DaTa SciENce USErs Experience” (**PID2020-115832GB-I00**) de la convocatoria Proyectos de I+D+i 2020 de la Agencia Estatal de Investigación, y de la Red de Investigación AI4Software en “Inteligencia Artificial Aplicada al Proceso de Desarrollo Software” (**RED2022-134647-T**) de la convocatoria de Redes de Investigación 2022, ambos proyectos subvencionados por el Ministerio de Ciencia e Innovación (MICIN/AEI/10.13039/501100011033).

Agradecimientos

"Ningún deber es más urgente que el de dar las gracias"
James Allen

Tras tanto esfuerzo y entrega, por fin llega el momento de cerrar este capítulo tan significativo en mi vida. Ha sido un largo camino que no solo ha marcado un crecimiento en mi carrera profesional, sino también un desarrollo a nivel personal. Mirar atrás me llena de orgullo al ver lo afortunado que he sido de contar con el apoyo de todas las personas que, de una manera u otra, han estado a mi lado y han hecho posible esta tesis. No puedo, ni quiero, dejar pasar la oportunidad de expresarles mi más sincero agradecimiento, desde lo más profundo de mi corazón.

A mi director José Raúl, ya que su incansable dedicación, liderazgo y motivación han sido pilares fundamentales en este viaje lleno de altibajos. Su cariño y paciencia conmigo han sido verdaderamente determinantes. Si tuviera que destacar una sola cosa de esta experiencia, sin duda alguna sería la gran amistad y complicidad que he encontrado en él. Para siempre quedarán en mi recuerdo todas esas sesiones de trabajo, intercambios de ideas y los proyectos que hemos compartido en todo este tiempo. No tengo la menor duda de que esto no es más que un mero punto y seguido en nuestro camino.

A todas esas personas con las que he tenido el placer de trabajar y aprender de ellas. A Aurora, compañera ejemplar que siempre está dispuesta a ayudar con la mejor de las actitudes. A Antonio Vallecillo, cuya ilusión y experiencia nos ayudó a seguir adelante con entusiasmo y mayor motivación. A todos mis compañeros del grupo de investigación, si bien no he pasado tanto tiempo con ellos como me hubiera gustado, el tiempo que lo hice me hicieron sentir como uno más.

Por último, gracias a mi familia por toda su confianza y apoyo incondicional. Especialmente a la persona más importante para mí, Rosa. Tu amor incondicional, apoyo constante y paciencia infinita han sido mi mayor fortaleza durante este proceso. Cada día, tu presencia ha sido mi refugio, tu aliento mi impulso y tu compañía mi mayor inspiración. Sin tu apoyo inquebrantable y tu comprensión, esta tesis hubiera sido imposible. Gracias por ser mi pilar, mi motivación y mi compañera de vida. Te amo más de lo que las palabras pueden expresar.

Gracias.
Rubén

Resumen

En la actualidad estamos inmersos en un entorno de continua generación de datos provenientes de diversas fuentes, como dispositivos IoT, redes sociales, transacciones comerciales y más. Esta explosión de datos ha dado lugar a un escenario donde se produce una vasta cantidad de información cada segundo. Sin embargo, el simple hecho de recolectar datos no es suficiente. La verdadera riqueza se encuentra en el análisis y comprensión de estos datos.

Las aplicaciones intensivas en datos se han convertido en una necesidad para empresas y organizaciones. Estas aplicaciones permiten no solo gestionar la vasta cantidad de información generada, sino también extraer conocimientos valiosos que impulsan la toma de decisiones informadas y la innovación en diversos sectores. Desde el ámbito de la medicina hasta el marketing, las aplicaciones intensivas en datos ofrecen una ventaja competitiva al desentrañar patrones complejos, prever tendencias futuras y optimizar procesos. En este contexto, la capacidad de desarrollar y desplegar rápidamente aplicaciones que sean capaces de procesar el gran volumen y variedad de datos de manera eficiente se convierte en un objetivo primordial. El reto aquí es encontrar métodos y herramientas que simplifiquen el desarrollo y la implementación de estas aplicaciones, eliminando las barreras de entrada y permitiendo que un amplio espectro de profesionales pueda aprovechar al máximo el potencial de los datos en sus respectivos campos de trabajo.

La creación de aplicaciones intensivas en datos no está exenta de desafíos. Uno de los principales problemas radica en la complejidad inherente de los procesos de desarrollo, que a menudo requieren de conocimientos profundos en diversas áreas de la informática, como la minería de datos, el aprendizaje automático y la computación distribuida. Esta complejidad puede resultar determinante para profesionales no especializados en tecnología, limitando así la adopción y el uso de estas aplicaciones en un amplio espectro de dominios. Además, la configuración avanzada de herramientas intensivas en datos para adaptarlas a problemas específicos y la necesidad de utilizar un gran número de herramientas dependiendo de las necesidades de cada dominio añaden una capa adicional de dificultad. En consecuencia, muchas organizaciones se enfrentan a la barrera de la curva de aprendizaje, los altos costos de implementación y la falta de recursos especializados, lo que limita su capacidad para aprovechar al máximo el potencial de sus datos.

Esta tesis busca la democratización del proceso de creación de este tipo de aplicaciones al adoptar un enfoque basado en flujos de trabajo, con el que los profesionales dispongan de herramientas con las que puedan diseñar, visualizar y ejecutar procesos de análisis de datos de manera más sencilla y efectiva. Estos flujos de trabajo

permiten la definición de una secuencia lógica de tareas y operaciones, desde la adquisición y limpieza de datos hasta la generación de informes y visualizaciones. Así, los flujos de trabajo intensivos en datos se presentan como una herramienta clave para simplificar el desarrollo de aplicaciones basadas en datos, permitiendo que un amplio espectro de profesionales pueda crear soluciones innovadoras y eficientes en sus respectivos dominios.

Durante el desarrollo de esta tesis doctoral se ha explorado el uso de una gran variedad de enfoques y técnicas, como el desarrollo de software dirigido por modelos y el desarrollo de plataformas de desarrollo *low-code y no-code*. En primer lugar, se ha propuesto un lenguaje de modelado específico del dominio de los flujos de trabajo intensivos en datos. Este lenguaje, agnóstico a cualquier herramienta, proporciona flexibilidad y practicidad a los expertos del dominio a la hora de definir aplicaciones intensivas en datos con un alto nivel de abstracción. Además, siguiendo los preceptos de la ingeniería del software dirigido por modelos, su sintaxis abstracta ha sido formalizada en términos de metamodelos, haciendo uso de transformaciones de modelos para alcanzar la interoperabilidad entre distintas herramientas basadas en flujos de trabajo. A continuación, se ha trabajado en una herramienta generadora de herramientas intensivas en datos específicas de dominio. Esta herramienta permite a los científicos de datos generar semi-automáticamente herramientas específicas de dominio orientadas a los requisitos y necesidades de los expertos de dominio, sin necesidad de conocimientos en programación o en la configuración y adaptación de métodos de procesamiento de datos. Esta herramienta generadora se basa en el principio de separación de responsabilidades y hace uso de técnicas de la ingeniería de software dirigida por modelos. Finalmente, se demuestra la aplicabilidad de la propuesta con la generación de una herramienta para el análisis del rendimiento académico en el dominio de la minería de datos educacional. Esta herramienta permite, a profesores y gestores académicos, el análisis de los datos que disponen sobre sus estudiantes, ya sea procedentes de plataformas de gestión del aprendizaje (como Moodle o Canva), como de cursos masivos en línea (MOOC por sus siglas en inglés). La practicidad de esta herramienta es evaluada sobre una serie de casos de uso específicos, como la predicción de aprobados y suspensos, o la predicción de la tasa de abandono en fases tempranas.

Abstract

Currently, we are in an environment of continuous data generation from various sources, such as IoT devices, social networks, commercial transactions, and more. This explosion of data has led to a scenario where a vast amount of information is produced every second. However, simply collecting data is not enough. The challenge relies in the analysis and understanding of this data.

Data-intensive applications are now essential for businesses and organizations. These applications not only allow managing the vast amount of generated information, but also extracting valuable insights that drive informed decision-making and innovation in various sectors. From the field of medicine to marketing, data-intensive applications offer a competitive advantage by unraveling complex patterns, predicting future trends, and optimizing processes. In this context, the primary goal is to develop and deploy applications quickly that can efficiently manage the large volume and variety of data becomes. Therefore, it is imperative to find methods and tools that streamline the development and implementation of these applications. This involves eliminating entry barriers and empowering a diverse range of professionals to fully leverage the potential of data in their respective fields of work.

The creation of data-intensive applications is not without challenges. One of the main problems is in the inherent complexity of development processes, which often require deep knowledge in various areas of computer science like data mining, machine learning, and distributed computing. This complexity can be decisive for non-technical professionals, thus limiting the adoption and use of these applications across a wide range of domains. Moreover, the complexity increases with the advanced configuration of data-intensive tools to address specific problems, along with the necessity of using numerous tools tailored to the requirements of each domain. Consequently, many organizations face the barrier of the learning curve, high implementation costs, and a lack of specialized resources, limiting their ability to take advantage of their data.

The aim of this Ph.D. Thesis is to democratize the process of creating such applications by adopting a workflow-based approach, providing professionals with tools to design, visualize, and execute data analysis processes more simply and effectively. These data-intensive workflows allow the definition of a logical sequence of tasks and operations, from data acquisition and cleaning to report generation and visualizations. Thus, data-intensive workflows emerge as a key tool to simplify the development of data-driven applications, enabling a wide range of professionals to create innovative and efficient solutions in their respective domains.

In this Ph.D. Thesis, a wide variety of approaches and techniques have been explored, such as model-driven software development, and low-code and no-code development platforms. Firstly, a domain-specific modeling language for data-intensive workflows has been proposed. This tool-agnostic language provides flexibility and ease of use to domain experts when defining data-intensive applications at a high level of abstraction. Furthermore, its abstract syntax has been formalized following the principles of model-driven software engineering, using model transformations to achieve interoperability between different workflow-based tools. Next, efforts have been focused on developing a tool to generate domain-specific data-intensive applications. This tool allows data scientists to semi-automatically generate domain-specific tools oriented to domain experts' requirements and needs, without the need for programming knowledge, or configuring and adapting data processing methods. This generator tool is based on the principle of separation of concerns, and makes use of model-driven software engineering techniques. Finally, the proposal's applicability is demonstrated by generating a tool for analyzing academic performance in the educational data mining domain. This tool enables teachers and academic managers to analyze their students' data, whether from learning management platforms (such as Moodle or Canvas) or from massive open online courses (MOOCs). The practicality of this tool is evaluated on a series of specific use cases, such as predicting passes and failures, or predicting dropout rates in early stages.

Prefacio

La legislación española para estudios de doctorado, RD 99/2011, publicada el 28 de enero de 2011 (BOE-A-2011-2541), otorga a cada universidad española competencias para establecer los procedimientos de supervisión y evaluación necesarios para garantizar la calidad de las tesis doctorales. Como requisito único para la defensa, esta regulación nacional indica que el manuscrito debe ir acompañado de un documento que detalle las actividades de aprendizaje complementarias realizadas por el estudiante.

En consecuencia, la Universidad de Córdoba tiene una regulación específica para estudios de doctorado, aprobada por su junta de gobierno el 21 de diciembre de 2011. Esta regulación establece un control de calidad con carácter general en el que se exige que durante el proceso de elaboración de la Tesis Doctoral, se haya generado, al menos, una aportación de calidad directamente relacionada con el trabajo de tesis. Según dicha regulación, se entiende como aportación de calidad la publicación (o aceptado para publicación) en revistas de investigación de alta calidad, es decir, que aparezcan en los primeros tres cuartiles de JCR (*Journal Citation Reports*).

Durante el trabajo de investigación de esta tesis doctoral se han generado diversas publicaciones, siendo una de ellas una aportación de calidad ya que se trata de una publicación en la revista *Business & Information Systems Engineering* (BISE), perteneciente al primer cuartil de JCR. También se ha publicado un capítulo para el libro *Digital Marketplaces Unleashed* de la editorial Springer, así como en diferentes conferencias internacionales y nacionales, como en *Intelligent Systems Design and Applications* (ISDA), en la Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA), en las Jornadas de Ingeniería del Software y Bases de Datos (JISBD) o en las Jornadas de Ciencia e Ingeniería de Servicios (JCIS). Además, un trabajo se encuentra en revisión para su publicación en una revista de alta calidad.

Con respecto al manuscrito, este debe incluir, como mínimo: la introducción al tema de investigación y un resumen, en español, del estado de la cuestión; los objetivos que se pretenden alcanzar; la metodología de investigación; una exposición de la investigación llevaba a cabo; los resultados y discusión, si procede; así como las conclusiones y la bibliografía referenciada. También se deberá incluir un informe de la persona encargada de su dirección.

Siguiendo estas pautas, esta memoria se compone de un total de siete capítulos, además de una serie de apéndices con información complementaria a la investigación:

En el Capítulo 1 se introduce el problema central de esta tesis doctoral, así como las motivación y objetivos que se pretenden conseguir. Además, se explica en detalle

la metodología seguida que consiste principalmente en la realización de un análisis bibliográfico del estado del arte y de la utilización de un marco experimental basado en el método científico y adaptado para el campo de la Informática. Por último, se adelantan las distintas contribuciones que han sido realizadas para la consecución de los objetivos planteados.

En el Capítulo 2 se hace una revisión de los fundamentos teóricos y del estado del arte de los distintos campos de investigación involucrados en esta tesis doctoral. En primer lugar, se hace una introducción detallada del estado del arte de las aplicaciones intensivas en datos y los flujos de trabajo científicos, destacando las fortalezas y debilidades de las soluciones existentes para su adopción en un mayor rango de dominios intensivos en datos. A continuación, se profundiza en los lenguajes de modelado específicos de dominio y su relación con la ingeniería de software dirigida por modelos. En este contexto, se tratan aspectos fundamentales como el metamodelado de lenguajes y la transformación de modelos. Finalmente, el estado actual de las plataformas de desarrollo *low-code* y *no-code* es presentado como una solución para facilitar la definición de aplicaciones intensivas en datos por parte de profesionales que no poseen conocimiento en informática.

Los Capítulos 3, 4 y 5 son los capítulos que desarrollan el trabajo de investigación que presentan las distintas contribuciones. En el Capítulo 3 se detalla el diseño e implementación de SWEL, un lenguaje de modelado específico de dominio para la definición a alto nivel de aplicaciones intensivas en datos. En el Capítulo 4 se presenta la arquitectura de WORKGENESIS y el proceso para la generación de herramientas intensivas en datos adaptadas a dominios específicos. Finalmente, se muestra la aplicabilidad de WORKGENESIS sobre el dominio de la minería de datos educacionales. Para ello, en el Capítulo 5 se genera una herramienta, LOCOAPE, que facilita la predicción del rendimiento académico de los estudiantes. Esta herramienta está orientada tanto a educadores sin conocimientos en análisis de datos como para profesionales expertos que pueden ampliar y adaptar su funcionalidades a sus propias necesidades.

En el Capítulo 6 se presentan las publicaciones derivadas del desarrollo de esta tesis doctoral, organizadas en publicaciones de revistas y libros, y en publicaciones de conferencias tanto nacionales como internacionales. Finalmente, en el Capítulo 7 se discuten las conclusiones finales y se destacan las líneas de trabajo abiertas.

También, se adjuntan una serie de apéndices que permiten ampliar la información de esta tesis doctoral. El Apéndice A incluye el metamodelo completo de SWEL, donde se realiza un informe resumido de cada metaclase junto con sus propiedades y asociaciones. En el Apéndice B se detalla el proceso de definición de las transformaciones de modelos que se han implementado, tanto las transformaciones unidireccionales

como las transformaciones bidireccionales. Finalmente, en el Apéndice C se incluye el formulario de evaluación utilizados por los profesionales para la evaluación de SWEL, así como todos los resultados anonimizados que se han recopilado.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

CASE *computer-aided software engineering*

CLI *command-line interface*

CIM *computation independent model*

DAG *directed acyclic graph*

DCG *directed cyclic graph*

DIA *data-intensive application*

DIW *data-intensive workflow*

DSML *domain-specific modelling language*

DSR *Design science research*

EDM *educational data mining*

GPL *general-purpose programming languages*

LCDP *low-code development platform*

LMS *learning management system*

M2M *model-to-model*

M2T *model-to-text*

ML *machine learning*

MDSE *model-driven software engineering*

MTL *model transformation language*

NCDP *no-code development platform*

OMG *Object Management Group*

PIM *platform independent model*

PSM *platform specific model*

RQ *research question*

SWEL *scientific workflow execution language*

SWfMS *scientific workflow management system*

T2M *text-to-model*

WfMS *workflow management system*

Capítulo 1

Introducción

"Estamos ahogados en información y
hambrientos de conocimiento"
Rutherford D. Rogers

En este capítulo se define el problema y su contexto, así como la motivación que impulsa esta investigación y los objetivos que se persiguen, presentando tanto el propósito general como los objetivos específicos. A continuación, la metodología de investigación empleada es detallada, exponiendo el enfoque metodológico adoptado, los procedimientos utilizados y la justificación de su elección. Finalmente, se presentan las contribuciones originales de este trabajo, resaltando su aporte al conocimiento existente.

1.1. Definición del problema

Muchas organizaciones en todo el mundo están invirtiendo en mejores soluciones que les permitan obtener conocimiento útil a partir de enormes cantidades de datos para lograr una posición líder en el mercado. Según una encuesta sobre tecnologías de la información y líderes empresariales realizada por Gartner [59], el 72 % de las empresas líderes en datos están apostando firmemente en iniciativas dirigidas por los datos y de transformación digital.

De la misma manera, un informe presentado por Accenture [139] ya destacaba que las empresas de alto rendimiento estaban incorporando análisis de datos masivos para respaldar su toma de decisiones y sus procesos de decisión. En este contexto, actores importantes en el mercado global coinciden en que las soluciones intensivas en datos son ya una ventaja competitiva para las organizaciones, y son clave para aumentar tanto la productividad como la innovación.

Para satisfacer la gran demanda de aplicaciones intensivas en datos en la industria, han surgido una gran cantidad de tecnologías y técnicas, creando un amplio y heterogéneo panorama de herramientas asociadas al concepto de *big data* [1]. *Big data* es un enfoque intensivo en datos que surge como respuesta a la creciente generación, análisis y almacenamiento de datos, tanto estructurados como no estructurados. Sin embargo, este enfoque va más allá de las capacidades de la tecnología tradicional debido a su gran magnitud y complejidad. La esencia del *big data* es la creación de nuevas herramientas de procesamiento de grandes volúmenes de datos que permitan el desarrollo de soluciones para generar nuevo conocimiento y mejorar la toma de decisiones.

FirstMark Capital publicó una revisión general [148] de las tecnologías intensivas en datos más importantes, clasificadas en diferentes categorías como infraestructura, análisis de datos, inteligencia artificial y aplicaciones. Todo este abanico de tecnologías hacen posible el desarrollo de soluciones *big data* en una amplia gama de dominios, como la sanidad, la industria o el marketing, donde el análisis de grandes cantidades de datos es esencial para descubrir nuevo conocimiento.

Sin embargo, este tipo de herramientas intensivas en datos suelen requerir una gran curva de aprendizaje debido fundamentalmente al profundo conocimiento necesario en diversas áreas del campo de la informática [91], como la minería de datos, el aprendizaje automático, la ingeniería de software o la computación distribuida, entre otras. Así, las soluciones intensivas en datos se adoptan más ampliamente en aquellos dominios cuyas empresas pueden realizar grandes inversiones económicas para emplear a profesionales bien capacitados en estos campos. En otros casos, el desarrollo de este tipo de soluciones acaban siendo externalizadas. Por todo ello, es esencial encontrar mecanismos y herramientas adecuadas para simplificar, popularizar y promover la adopción de soluciones intensivas en datos por parte de las

empresas y organizaciones en aquellos sectores donde, aunque el procesamiento de datos es muy necesario, el esfuerzo económico es limitado.

Así, existe la necesidad tanto de democratizar la definición y ejecución de flujos de trabajo consistentes en procesos de análisis de datos como de considerar los desafíos que ello supone. Para ello, debemos disponer de mecanismos que permitan simplificar la especificación de estos procesos, además de permitir la integración de tecnologías de procesamiento de datos con el fin de ejecutar automáticamente la secuencia de acciones necesarias. Todo ello debería ser posible sin requerir conocimientos en computación.

1.2. Motivación y objetivos

La adopción de soluciones intensivas en datos puede ser un desafío para muchas empresas y organizaciones debido a la curva de aprendizaje y al conocimiento técnico requerido, lo que resulta en una limitación de su aplicabilidad en un rango más amplio de dominios donde se necesita generar conocimiento a partir de los datos generados y almacenados. Por tanto, la búsqueda de mecanismos para facilitar la creación de aplicaciones que favorezcan análisis de datos radica principalmente en eliminar las distintas barreras que actualmente existan para su adopción, como la necesidad de poseer un conocimiento técnico profundo, la configuración avanzada de herramientas para adaptarlas a problemas específicos o la obligación de utilizar un gran número de herramientas dependiendo de las necesidades de cada dominio.

El análisis del estado del arte de las aplicaciones intensivas en datos ha permitido identificar la necesidad de facilitar la adopción de este tipo de soluciones por un público más amplio, además de considerar los distintos desafíos a los que los se deben enfrentar los profesionales no expertos en computación. En este contexto, la democratización de las aplicaciones intensivas en datos no solo implica hacer que haya un mayor número de herramientas disponibles, sino proporcionar mecanismos que permitan crear soluciones listas para usar, interoperables y que requieran muy poca o ninguna configuración. Esta democratización se logrará a través del aumento del nivel de abstracción, ocultando los detalles de bajo nivel relacionados con redes, computación distribuida o lenguajes de programación, de manera que se puedan

crear procesos complejos de análisis de datos en múltiples dominios con un esfuerzo limitado.

1.2.1. Objetivo principal de la tesis doctoral

El objetivo general de esta tesis doctoral es:

> *Encontrar mecanismos que faciliten la creación de herramientas intensivas en datos adaptadas a dominios específicos, reduciendo la necesidad de configuración y de poseer conocimientos específicos en computación.*

Se pretende dar respuesta a la complejidad existente en el desarrollo de aplicaciones intensivas en datos específicas, en lo que la definición, configuración y los conocimientos técnicos necesarios representan barreras significativas para su adopción en un mayor rango de dominios. Con la simplificación del proceso de creación de este tipo de aplicaciones, se favorece que los profesionales de diversos ámbitos y con diferentes niveles de habilidades tecnológicas dispongan de herramientas intensivas en datos adaptadas a sus necesidades con un menor esfuerzo y costes reducidos.

1.2.2. Subobjetivos de la tesis doctoral

Este objetivo principal se puede dividir en varios subobjetivos. Esta descomposición facilita una aproximación sistemática y detallada, abordando de manera específica cada aspecto de la investigación.

- **O_1:** *Diseño e implementación de un lenguaje de modelado específico de dominio, independiente de cualquier herramienta.*

  Al ser agnóstico de herramientas particulares, este lenguaje deberá proporcionar flexibilidad y practicidad, permitiendo a los expertos del dominio definir aplicaciones intensivas en datos con un alto nivel de abstracción. Se permite así ocultar los detalles de bajo nivel, facilitando la definición de conceptos complejos y favoreciendo la reutilización del conocimiento en múltiples dominios.

- **O_2:** *Diseño e implementación de mecanismos que permitan la interoperabilidad entre las distintas herramientas intensivas en datos actuales.*

  Estos mecanismos deben facilitar la integración entre diversas herramientas existentes en el ámbito de las aplicaciones intensivas en datos, promoviendo un entorno donde estas herramientas puedan trabajar de manera conjunta. Además se estudiará su aplicabilidad de estos mecanismos con algunas de las principales herramientas intensivas en datos.

- **O_3:** *Diseño e implementación de una herramienta que permita generar herramientas visuales intensivas en datos ya adaptadas a dominios específicos.*

  Al desarrollar una herramienta especializada en la generación de estas soluciones, se pretende facilitar a los profesionales de distintos dominios la creación de herramientas adaptadas a sus necesidades específicas sin requerir conocimientos profundos en desarrollo de software. Además se estudiará la aplicabilidad de esta herramienta en un dominio intensivo en datos particular: el campo de la educación. En concreto, se trabajará con el problema de la predicción automática del rendimiento académico de los estudiantes. Este caso de estudio proporcionará información valiosa sobre su funcionalidad y utilidad práctica, respaldando así su implementación en entornos reales.

1.3. Metodología de investigación

A continuación se detalla la metodología seguida en esta tesis doctoral, que se centra principalmente en el desarrollo y validación experimental de artefactos concretos y aplicables en contextos reales. Además, se han adoptado las mejores prácticas de investigación en Ingeniería de Software en lo que respecta a la búsqueda bibliográfica y la discusión de amenazas a la validez detectadas.

Siguiendo el método científico, se formula al menos una pregunta de investigación o *research question* (RQ), que es similar a una hipótesis ya que ambas representan una declaración formal del fenómeno o concepto en estudio [29]. La diferencia radica en la intención de tal declaración, ya que las hipótesis hacen predicciones sobre los resultados esperados basados en la relación entre variables dependientes e independientes, mientras que las RQs se centran en la exploración de los factores que rodean

al fenómeno con el objetivo de derivar nuevos conocimientos o mejorar soluciones previas. Por lo tanto, las RQs son más apropiadas cuando la naturaleza de la investigación no permite hacer predicciones, lo cual suele ser el caso de la investigación cualitativa orientada a soluciones [29]. La identificación de RQs es una práctica bien establecida en la investigación en ingeniería de software [43].

1.3.1. Análisis bibliográfico

A lo largo del desarrollo de esta tesis doctoral, se ha realizado la revisión exhaustiva y periódica de la literatura con el objetivo de recopilar y evaluar las investigaciones previas relacionadas tanto con el objetivo general (véase la Sección 1.2.1). De esta manera se contextualizan y fundamentan las decisiones tomadas a lo largo del estudio, así como se identifican posibles vacíos en el conocimiento existente.

Para llevar a cabo el análisis bibliográfico se han explorado bases de datos generales como ACM Library, DBLP, IEEE Xplore, ISI Web of Knowledge, ScienceDirect, SpringerLink y Scopus. Estas bases de datos proporcionan un panorama amplio de las investigaciones relevantes en el ámbito de estudio, permitiendo una visión completa de los avances.

Las fuentes consultadas son investigaciones originales, revisiones sistemáticas, estudios empíricos y teóricos publicados en revistas científicas, libros académicos y actas de conferencias relevantes. Se han excluido fuentes no verificables y material de divulgación no académica.

Una vez recopiladas las fuentes bibliográficas pertinentes, se llevó a cabo un proceso de análisis estructurado. Las fuentes son organizadas en categorías temáticas y cronológicas para facilitar la identificación de tendencias y evoluciones en el conocimiento del campo. Se ha realizado una lectura crítica de cada fuente, extrayendo conceptos clave, metodologías utilizadas, avances destacados y áreas de debate.

1.3.2. Marco experimental

La metodología seguida para cumplir con los objetivos definidos en la Sección 1.2 se basa en las fases habituales del método científico, tal y como se adopta a menudo en experimentos en el campo de la Ingeniería de Software [41]:

- Observación: Estudio de las necesidades para promover la adopción de soluciones intensivas en datos en un mayor rango de dominios y análisis de las herramientas actuales para la definición de aplicaciones intensivas en datos.
- Inducción: Extracción de los requisitos para la conceptualización de soluciones innovadoras para resolver los problemas del mundo real identificados previamente.
- Hipótesis: Planteamiento de la hipótesis de partida y formulación de las preguntas de investigación [28] adecuadas para el diseño de las soluciones innovadoras que dan respuesta a las mismas.
- Experimentación: Implementación de las soluciones diseñadas mediante artefactos software y su evaluación a través de casos de uso reales para determinar su utilidad y adecuación a los problemas identificados.
- Análisis crítico: Discusión de los resultados obtenidos con respecto a las preguntas de investigación correspondientes, considerando enfoques alternativos y posibles amenazas a la validez [156].

En el contexto de esta tesis doctoral, se han adoptado metodologías y prácticas más específicas que guían de manera efectiva este proceso experimental. El principal objetivo es obtener un marco experimental más sólido para la generación y evaluación de soluciones innovadoras mediante artefactos software.

Para ello, en el Capítulo 3 se ha utilizado una metodología denominada "*design science research*", conocida por su enfoque iterativo y centrado en la creación de artefactos software. Por otro lado, se han abordado prácticas de análisis de amenazas a la validez cuando es aplicable, reconociendo así la importancia de evaluar y mitigar posibles influencias que podrían afectar la validez interna y externa de nuestro estudio. De esta manera se busca no solo desarrollar soluciones efectivas, sino también garantizar su robustez y validez.

Design science research

Los problemas que nos encontramos en cualquier contexto tienden a ser resueltos mediante la creación de artefactos. Un artefacto es un objeto hecho por humanos

con el propósito de abordar un problema particular. Algunos artefactos pueden ser objetos físicos, como martillos, prótesis o vehículos. Sin embargo, otros artefactos pueden ser dibujos o planos, como el plano de una vivienda. Así, los algoritmos y procesos en tecnología también pueden ser considerados artefactos como, por ejemplo, un proceso para diseñar bases de datos o un algoritmo para predecir la tasa de abandono en un aula. Lo común a todos estos artefactos es que ayudan a las personas cuando encuentran problemas en un dominio específico.

En el contexto de la investigación científica, existe un paradigma denominado "*design science research*" [151] para la resolución de problemas a través de la creación de artefactos innovadores que permiten alcanzar soluciones novedosas. Esta metodología tiene como objetivo el generar conocimiento sobre cómo las cosas pueden y deben ser construidas o dispuestas (es decir, diseñadas) para conseguir un conjunto determinado de objetivos de investigación. Por tanto, es compatible con muchas áreas de investigación, como la ingeniería, la arquitectura, los negocios, la economía y disciplinas relacionadas con la tecnología para la creación de soluciones novedosas.

> *Design science research* (DSR) es el paradigma de investigación que implica el diseño y desarrollo de artefactos en un contexto determinado. Estos artefactos están diseñados para interactuar con el contexto del problema con el fin de mejorar y promover el avance del conocimiento.

Johannesson and Perjon [107] propusieron un *framework* para DSR compuesto por cinco fases interconectadas que son fundamentales para la generación de soluciones innovadoras y de conocimiento útil. Estas fases abarcan desde la investigación del problema y la definición de requisitos, el diseño y desarrollo de artefactos, y finalmente, la demostración y evaluación (Figura 1.1):

- **Explicación del problema**: En esta fase inicial, se realiza la tarea de comprender y explicar de manera completa el problema práctico que se pretende abordar. Esto implica la exploración de investigaciones relacionadas y determinar las posibles mejoras en las soluciones actuales. Por tanto, esta fase tiene como objetivo determinar la necesidad de abordar el problema y el impacto potencial de la solución propuesta.

- **Definición de requisitos**: En esta fase se establecen los requisitos funcionales y no funcionales que la solución propuesta debe cumplir. Se realiza un diseño detallado del artefacto que abordará el problema, considerando diferentes alternativas y enfoques posibles. Se pueden crear modelos conceptuales, diagramas y especificaciones técnicas que guíen el proceso de construcción.
- **Diseño y desarrollo del artefacto**: Durante esta fase se procede a la creación y desarrollo del artefacto según el diseño definido en la fase anterior. Esto puede incluir la implementación de software, la creación de prototipos físicos o la generación de materiales educativos, dependiendo del tipo de solución que se vaya a llevar a cabo. Es importante garantizar que el artefacto cumpla con los requisitos establecidos y se comporte de acuerdo a las expectativas.
- **Demostración del artefacto**: Una vez creado el artefacto, se procede a demostrar cómo se comporta con respecto al problema identificado en la fase inicial. Se llevan a cabo pruebas y experimentos para evaluar la eficacia y viabilidad del artefacto en el contexto práctico.
- **Evaluación del artefacto**: En esta fase se realiza una evaluación exhaustiva del artefacto desarrollado. Se analizan los resultados obtenidos de la demostración y las pruebas para determinar la efectividad de la solución en la resolución del problema. Se comparan los resultados con los objetivos y requisitos iniciales para validar la utilidad y el impacto del artefacto.

Análisis de la amenaza a la validez

El análisis de amenazas a la validez o *validity threats* es una práctica común dentro de la Ingeniería de Software [156] que se centra en los posibles factores que pueden influir en la precisión, la confiabilidad y la generalización de los resultados obtenidos en un estudio de investigación. Estas amenazas pueden surgir en diversas etapas del proceso de investigación y pueden comprometer la integridad y la validez de las conclusiones extraídas.

Al identificar y considerar posibles fuentes de sesgo o limitaciones en el diseño, la ejecución o la interpretación, se garantiza que las conclusiones sean más sólidas y aplicables en contextos más amplios. Además, la transparencia en la discusión de

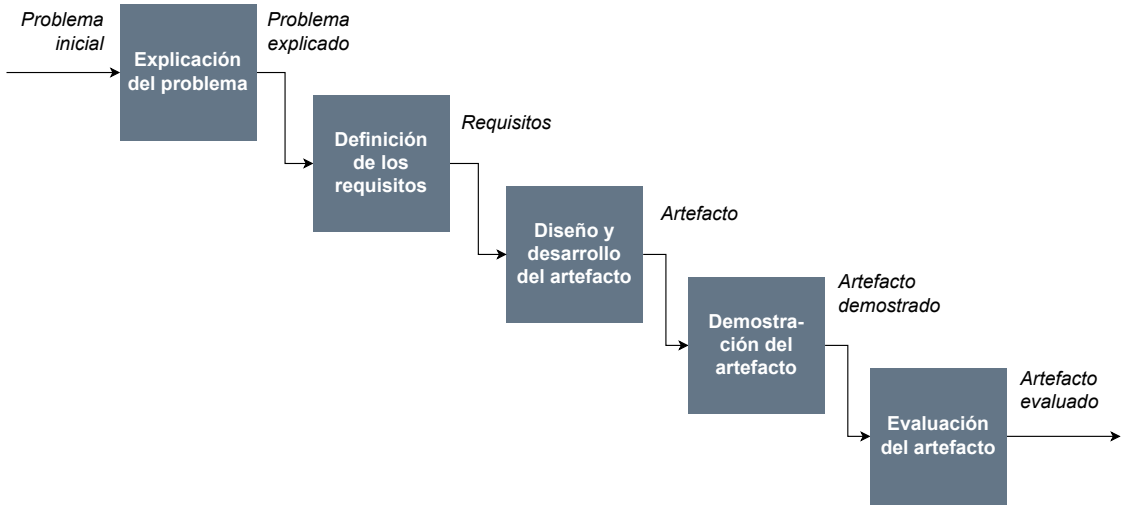


Figura 1.1: Fases del *framework* DSR (adaptada de [107]).

las amenazas a la validez permite a los investigadores y a la comunidad científica en general comprender mejor los alcances y las limitaciones de los resultados, promoviendo una evaluación crítica.

Cuando es aplicable, los aspectos relacionados con los cuatro tipos de amenazas a la validez se han discutido para los artefactos generados:

- **Validez de construcción**: Se centra en la relación entre la teoría y la observación. Es esencial asegurarse de que el diseño y desarrollo del artefacto reflejen con precisión las necesidades y requisitos identificados en la fase de definición de requisitos. Esto garantiza que el artefacto sea relevante y adecuado para abordar el problema específico.

- **Validez interna**: Está relacionada con la relación causal entre la hipótesis y los resultados. Es importante considerar los parámetros y las variables que pueden influir en los resultados y asegurarse de que los cambios observados en el artefacto se puedan atribuir con confianza a las modificaciones realizadas.

- **Validez de conclusión**: Se refiere a la relación entre el tratamiento y el resultado. Esto implica evaluar si los resultados observados en las pruebas y demostraciones realmente respaldan las afirmaciones y objetivos planteados en la fase de diseño. Es importante evitar conclusiones exageradas o infundadas basadas en interpretaciones erróneas de los datos.

- **Validez externa**: Se preocupa por la generalización de los resultados. Hay que considerar si el artefacto es aplicable en diversas condiciones y si los resultados obtenidos pueden ser extrapolados a casos más amplios. También hay que identificar las limitaciones en la aplicabilidad del artefacto y describir claramente en qué contextos se espera que sea efectivo.

1.4. Contribuciones

En esta sección se motivan las contribuciones derivadas del trabajo de investigación de esta tesis doctoral. Se destacan los avances alcanzados a través de la publicación de artículos científicos tanto en revistas como en conferencias, los cuales representan una contribución al estado del arte. Además, se describe el conjunto de artefactos software, en términos de DSR, que han sido desarrollados como resultados de investigación con el objetivo de facilitar la adopción de soluciones intensivas en datos en múltiples dominios.

Estas contribuciones están vinculadas a los objetivos $\mathbf{O}_1$–$\mathbf{O}_3$ de esta tesis doctoral (véase la Sección 1.2.2) y pueden ser divididas en dos categorías, detalladas a continuación:

1.4.1. Sobre la definición y reutilización del conocimiento en dominios intensivos en datos

Para permitir la definición y ejecución de aplicaciones intensivas en datos sin necesidad de poseer conocimientos en computación e independiente de cualquier herramienta subyacente, así como para favorecer la interoperabilidad del conocimiento, se han realizado las siguientes contribuciones:

- Un lenguaje de modelado específico de dominio que permite la definición del conocimiento del dominio a un alto nivel de abstracción ($\mathbf{O}_1$): Scientific Workflow Execution Language (SWEL). SWEL ha sido diseñado para capturar el conocimiento de las aplicaciones intensivas en datos de manera que pueda ser reutilizado en cualquier dominio y no dependa de ninguna herramienta específica para su ejecución. Una primera versión de este lenguaje ha sido presentada

en las Jornadas en Ingeniería de Software y Bases de Datos (JISBD) [130]. Finalmente, la versión final de SWEL ha sido publicada en la revista Business & Information Systems Engineering (BISE) [132].

- Una serie de mecanismos que permiten la reutilización del conocimiento y la interoperabilidad entre herramientas intensivas en datos mediante transformaciones de modelos ($\mathbf{O}_2$). Estas transformaciones utilizan SWEL como pieza central en el proceso de interoperabilidad para permitir que el conocimiento ligado a una herramienta específica pueda ser convertido y compatible con otras herramientas similares. La aplicación de estas transformaciones para favorecer la interoperabilidad entre herramientas en Industria 4.0 ha sido presentada en la Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA) [127]. Finalmente, el proceso de interoperabilidad propuesto, junto con una serie de casos de estudio aplicados en dominios intensivos en datos, han sido publicados junto a SWEL en la revista BISE [132].

1.4.2. Sobre la generación de herramientas intensivas en datos específicas de dominio

Para democratizar la definición de aplicaciones intensivas en datos en un mayor rango de dominios mediante la generación automática de herramientas intensivas en datos orientadas a profesionales no expertos en computación, se han realizado las siguientes contribuciones:

- Una herramienta orientada a expertos en análisis de datos para la generación automática de herramientas intensivas en datos específicas de dominio ($\mathbf{O}_3$): Workgenesis. Workgenesis es una solución *low-code* que permite a los científicos de datos definir los flujos de trabajo necesarios para definir procesos complejos de análisis de datos en cualquier dominio. También, Workgenesis permite generar herramientas específicas de dominio para que puedan ser utilizadas por los expertos del dominio sin necesidad de poseer conocimientos en desarrollo de software ni en computación. La necesidad de disponer de este tipo de herramientas para favorecer la adopción de soluciones intensivas en datos ha sido publicada como capítulo en el libro Digital Marketplaces

Unleashed (DMU) [128]. El prototipo de WORKGENESIS ha sido presentado tanto en las Jornadas de Ciencia e Ingeniería de Servicios (JCIS) [133] como en la conferencia internacional *Intelligent Systems Design and Applications* (ISDA) [131]. Además, una herramienta ha sido generada para facilitar la definición de aplicaciones basadas en algoritmos evolutivos, la cual ha sido presentada en CAEPIA [126].

- Aplicación de WORKGENESIS a un dominio intensivo en datos específico ($\mathbf{O}_3$), como es la minería de datos educacional [120]. Para ello, se ha generado una herramienta específica de dominio: LOCOAPE. LOCOAPE es una herramienta intensiva en datos para facilitar la predicción del rendimiento académico de los estudiantes. Esta herramienta está orientada a dos perfiles de usuario, al profesor sin conocimiento en análisis de datos y al experto en datos que pueden modificar la funcionalidad ofrecida para ampliarla y adaptarla a nuevos requisitos. Esta contribución se encuentra en proceso de revisión en revista indexada JCR.

1.4.3. Relación entre publicaciones y capítulos de la tesis

En la Tabla 1.1 se establece la relación entre los artículos científicos publicados y los capítulos vinculados a dichas publicaciones en el documento de esta tesis doctoral.

Publicación y año	**Capítulo 2**	**Capítulo 3**	**Capítulo 4**	**Capítulo 5**
JCIS 2014 [133]	X		X	
CAEPIA 2015 [126]			X	
JISBD 2016 [130]	X	X		
ISDA 2017 [131]	X	X	X	
DMU 2018 [128]	X		X	
CAEPIA 2018 [127]		X	X	
BISE 2023 [132]	X	X		
En revisión			X	X

Tabla 1.1: Relación de las publicaciones científicas y los capítulos de esta tesis.

Capítulo 2

Fundamentos y estado del arte

"Investigar es ver lo que todo el mundo ha visto,
y pensar lo que nadie más ha pensado"
Albert Szent-Györgyi

Este capítulo introduce los fundamentos y el estado del arte de las áreas de investigación en las que se basa esta tesis doctoral. Más concretamente, se definen, en primer lugar, las aplicaciones intensivas en datos junto con los dominios intensivos en datos, así como la tecnología de los flujos de trabajo científicos y los sistemas de gestión de flujos de trabajo científicos. A continuación, se introducen los lenguajes específicos de dominio, enmarcados dentro del ámbito de la ingeniería de software dirigida por modelos. En este contexto, los conceptos de metamodelado y transformaciones de modelos son también explicados. Finalmente, se presentan las plataformas de desarrollo *low-code* y *no-code*, destacando sus principales características, beneficios y diferencias, además de introducir algunas de las plataformas de desarrollo utilizadas hoy en día.

2.1. Aplicaciones intensivas en datos y flujos de trabajo científicos

El auge de los dominios intensivos en datos ha propiciado el desarrollo de aplicaciones específicas para la gestión y manipulación de datos que presentan tanto un gran

volumen como una gran variedad en su estructura. Estas aplicaciones intensivas en datos capturan el conocimiento específico del dominio con el fin de procesar y transformar los datos disponibles en conocimiento útil, válido y significativo para empresas y organizaciones. Es aquí donde los flujos de trabajo científicos se presentan como un mecanismo efectivo para definir este tipo de aplicaciones a un alto nivel de abstracción, ocultando los detalles de bajo nivel de implementación y permitiendo la automatización de su ejecución por sistemas especializados.

2.1.1. Aplicaciones intensivas en datos

La cantidad de datos generados cada día ha experimentado un aumento significativo en las últimas décadas y aún se espera que siga aumentando de manera considerable en los próximos años [143]. El fuerte crecimiento del volumen de datos hace que su gestión se haya convertido en uno de los principales retos tecnológicos debido, principalmente, a que el tamaño de los conjuntos de datos excede la capacidad de las herramientas que disponemos para gestionar, capturar y procesarlos en un tiempo razonable.

Este paradigma es el ampliamente conocido como *big data* [96], y ha convertido el procesamiento masivo de datos en el principal problema al que deben enfrentarse muchas de las aplicaciones actuales para extraer conocimiento novedoso, útil y valioso para las empresas y organizaciones [143]. Las aplicaciones que tienen que hacer frente a este tipo de necesidades se denominan aplicaciones intensivas en datos o *data-intensive application* (DIA), las cuales están orientadas principalmente al procesamiento de grandes cantidades de datos, tal y como se define en [17]:

> La **computación intensiva en datos** se ocupa de la producción, manipulación y análisis de datos a gran escala.

El procesamiento de un alto volumen de datos es uno de los grandes retos de este tipo de aplicaciones ya que requieren infraestructuras y técnicas especializadas para poder gestionar la enorme cantidad de información de manera eficiente y escalable. Aunque existen otros aspectos como la complejidad, velocidad o el almacenamiento de los mismos que también deben ser gestionados [20]. La complejidad hace referencia a que los datos se pueden encontrar tanto de forma estructurada, es decir, los datos

se encuentran bien organizados y su formato es conocido, pudiéndose almacenar en columnas y filas (normalmente en bases de datos relacionales o en hojas de cálculo tipo Excel), como de forma semi-estructurada o no estructurada, es decir, los datos no se ajustan a un modelo o esquema predefinido (algunos ejemplos son los ficheros de texto, documentos PDF o archivos de audio). La velocidad es la rapidez con la que se generan, se acceden y se consumen los datos. Finalmente, el almacenamiento es otro aspecto a tener en cuenta, considerando también el periodo de tiempo en el que los datos son útiles y deben ser persistidos.

Para poder gestionar y procesar esta gran cantidad de datos heterogéneos, las DIA siguen una serie de pasos donde en cada uno de ellos se realiza un tipo de operación concreta sobre los datos. Este tipo de procesos secuenciales, organizados en pasos o etapas, son conocidos como "*data pipelines*" [62]:

> Las aplicaciones intensivas en datos comprenden una serie de pasos de procesamiento para transformar conjuntos de datos masivos y complejos en información útil para los seres humanos y otros sistemas computacionales. Los científicos e ingenieros suelen denominar a estos sistemas de procesamiento de datos de múltiples etapas como ***data pipelines***.

En un *data pipeline* los datos en bruto procedentes de distintas fuentes de datos se capturan y almacenan. En la primera etapa del procesamiento se suelen aplicar técnicas para reducir el tamaño de los datos eliminando datos inválidos o no relevantes para puedan ser manipulados de forma más eficiente por los pasos posteriores.

Una vez realizada la captura y el procesamiento inicial, se aplican distintos tipos de algoritmos para analizar los datos y extraer información para que pueda ser consumida por humanos o por otros procesos computacionales. Finalmente, los resultados del análisis se presentan de forma que puedan ser utilizados para tomar decisiones. En esta última etapa es habitual utilizar herramientas avanzadas de visualización de datos. En la Figura 2.1 se ilustran las distintas fases que conforman un *data pipeline*.

2.1.2. Dominios intensivos en datos

Las DIA han sido tradicionalmente impulsadas por la computación científica. En esta ciencia intensiva en datos [68], los datos se presentan en escalas y formas muy

Figura 2.1: Ejemplo de las fases de un *data pipeline*.

diversas, abarcando grandes experimentos de cooperación internacional e incluyendo observaciones procedentes de distintos laboratorios.

Sin embargo, este escenario ha cambiado radicalmente ya que cada vez existen más empresas (buscadores web, empresas de publicidad en línea, redes sociales o comercio electrónico) y organismos públicos que están produciendo, extrayendo y procesando cantidades masivas de datos. Para estas entidades es fundamental analizar eficazmente estos enormes conjuntos de datos, puesto que constituyen una valiosa fuente de información sobre sus clientes y usuarios.

Este proceso de análisis de datos es cada vez más utilizado en ámbitos muy diversos, e incluye tareas que van tanto desde la adquisición y preparación de datos hasta la visualización y la extracción de conocimiento. Los dominios que tienen como principal objetivo el análisis de datos y extracción de conocimiento a partir de estos son conocidos como "dominios intensivos en datos" [47]:

> Los **dominios intensivos en datos** son aquellos en los que la perspectiva de los datos es la más relevante, y donde las aplicaciones deben tratar con grandes cantidades de datos (más o menos) estructurados, demandando operaciones básicas para extraer información valiosa.

Algunos de los ya numerosos dominios intensivos en datos son la producción industrial, la publicidad, la educación, la bioinformática o la sanidad, entre otros. Algunos de ellos son descritos a continuación e ilustran la importancia e impacto que tienen este tipo de aplicaciones [129]:

Sanidad: Los sistemas de información en el ámbito sanitario se han vuelto especialmente relevantes debido a la enorme cantidad de información gestionada en diferentes formas. Algunos ejemplos son los datos procedentes de sistemas de historial clínico electrónico o de los sistemas de diagnóstico y monitorización móviles[1].

El desarrollo de DIA ha supuesto beneficios que están ayudando significativamente a los pacientes y al personal sanitario en la mejora del diagnóstico y los tratamientos de enfermedades graves, en el descubrimiento de los efectos secundarios de determinados fármacos y compuestos, o en la detección de información potencialmente relevante. Así, este tipo de aplicaciones están transformando todo el proceso sanitario en un proceso basado en los datos más eficiente y menos costoso [11].

Gobierno: Los gobiernos están favoreciendo la digitalización masiva de datos [25] y la aplicación de enfoques de gobierno abierto [73] para mejorar los servicios públicos. El gobierno de Estados Unidos es uno de los principales promotores de los datos gubernamentales abiertos para fomentar la innovación y los descubrimientos científicos[2].

En este contexto, las DIA [77] permiten a los organismos públicos la extracción y descubrimiento de información relevante que sirva para mejorar los servicios básicos de los ciudadanos, reducir las tasas de desempleo, prevenir las amenazas de ciberseguridad o controlar los niveles de congestión del tráfico.

Producción industrial: Recientemente han surgido nuevos paradigmas en el campo de la producción industrial con el objetivo de mejorar los enfoques basados en los almacenes de datos tradicionales y las herramientas de inteligencia empresarial [48]. Un ejemplo es la fabricación inteligente [82], también conocida como fabricación predictiva, que gracias a las DIA es capaz de integrar grandes colecciones de datos, analizarlos y tomar decisiones para mejorar el rendimiento de los sistemas de fabricación existentes.

[1] IBM Software White Paper. *Data-driven healthcare organizations use big data analytics for big gains.* https://silo.tips/download/ibm-software-white-paper-data-driven-healthcare-organizations-use-big-data-analy (consultado: 25/10/2023)

[2] *Transparency and Open Government.* https://obamawhitehouse.archives.gov/the-press-office/transparency-and-open-government (consultado: 31/02/2023)

Del mismo modo, también se está adoptando el paradigma de la computación en la nube para permitir la prestación de servicios de fabricación con el fin de acceder a recursos distribuidos y mejorar el rendimiento del ciclo de vida del producto para reducir los costes [48].

Ciencia: La ciencia requiere cada vez más de nuevas herramientas computacionales para gestionar cantidades masivas de datos heterogéneos y distribuidos que deben ser almacenados, procesados, analizados y visualizados de manera eficiente. Estos métodos y herramientas computacionales permiten la producción de nuevo conocimiento en muchas áreas diferentes de la ciencia. Este tipo de ciencia intensiva en datos es denominada *e-Science* [69].

La *e-Science* incluye áreas como la física de partículas, la bioinformática, las ciencias de la tierra o las simulaciones sociales [68]. Las DIA permiten, por ejemplo, el análisis y almacenamiento de datos procedentes del Gran Colisionador de Hadrones de la Organización Europea para la Investigación Nuclear, que comenzó a tomar datos de forma masiva desde 2009.

Educación: La digitalización masiva de los datos procedentes del ámbito educacional ha sido impulsada por el uso de las plataformas de gestión del aprendizaje [84]. Estas plataformas, como Moodle, Blackboard o Canvas, sirven como entornos virtuales que no solo albergan contenido educativo, sino que también recopilan una amplia variedad de datos relacionados con la participación del estudiante, el rendimiento en evaluaciones y la interacción con el material de aprendizaje. Además, con la llegada de los cursos *on-line* abiertos y masivos (MOOC, por sus siglas en inglés), se ha producido una generación masiva de datos en este dominio, ofreciendo una oportunidad sin precedentes en el análisis y la comprensión de los patrones de aprendizaje a gran escala.

En este contexto, el análisis de datos educacionales se extiende más allá de la mera evaluación de calificaciones. Las instituciones educativas utilizan estos sistemas para identificar patrones de comportamiento del estudiante, anticipar posibles dificultades y personalizar la experiencia de aprendizaje [88].

Otros dominios: Otros muchos ámbitos están adoptando paulatinamente DIA para hacer frente a nuevos retos. Por mencionar algunos ejemplos, la banca [66] hace uso de DIA para cubrir algunos retos como la detección de fraudes con tarjetas. En el ámbito de las comunicaciones, los medios de comunicación y el entretenimiento necesitan recopilar y analizar datos del rendimiento de sus anunciantes, aprovechar el contenido de las redes sociales [108] o descubrir patrones de uso de la audiencia. Las compañías de seguros hacen uso de DIA para mejorar su rendimiento general optimizando la precisión de sus precios, las relaciones con sus clientes o la prevención de pérdidas, entre otras cuestiones. Además, otros ámbitos en los que se están aplicando enfoques intensivos en datos son el transporte [70] o el deporte [161], entre otros.

2.1.3. Flujos de trabajo científicos

Los *data pipelines* en las DIA pueden formularse en términos de flujos de trabajo. La tecnología de los flujos de trabajo surgió y se desarrolló, en su mayoría, en el contexto de la gestión de procesos de negocio [154] con el propósito de gestionar el conjunto de actividades que se realizan de forma coordinada en una empresa para conseguir un objetivo de negocio determinado. Cada proceso de negocio es ejecutado por una sola organización, pero puede interactuar con procesos de negocio realizados por otras organizaciones.

La gestión de estos procesos de negocio abarca conceptos, métodos y técnicas destinados a apoyar su diseño, administración, configuración, ejecución y análisis. Es en este contexto donde la *Workflow Management Coalition*[3] definió el concepto de flujo de trabajo [86]:

> Un **flujo de trabajo** es la automatización, total o parcial, de un proceso de negocio en el que los documentos, información o tareas se transmiten de un participante a otro acorde a un conjunto de reglas de negocio.

Los flujos de trabajo son un candidato ideal para representar y gestionar los *data pipelines* a mayor nivel de abstracción. Por este motivo, también son utilizados

[3] *Workflow Management Coalition*. https://wfmc.org/ (consultado: 15/11/2023)

por la comunidad científica para facilitar la creación de experimentos a gran escala, proporcionando documentación detallada sobre los mismos, describiendo el diseño de procesos computacionales, y permitiendo la gestión y el intercambio de conocimiento entre científicos [100].

> Un **flujo de trabajo científico** [57] se considera un mecanismo de alto nivel para definir y automatizar procesos computacionales intensivos en datos con el objetivo de realizar su ejecución en un periodo de tiempo razonable.

Los flujos de trabajo científicos permiten diseñar y razonar sobre la secuencia de operaciones que deben realizarse sobre los datos disponibles y de cómo se relacionan entre sí. Por tanto, los flujos de trabajo científicos no se centran en la implementación de cada componente de procesamiento de datos, sino que los tratan como "cajas negras" con entradas, parámetros y salidas específicas.

Este mayor nivel de abstracción favorece el intercambio de estos componentes computacionales con el fin de definir la secuencia de pasos adecuada en cada caso en función de los datos que se requieren y se producen. Así, los flujos de trabajo permiten el desarrollo de sofisticados análisis de datos y complejas simulaciones, fomentando la reproducibilidad y el intercambio de conocimiento [35].

La Figura 2.2 representa un flujo de trabajo científico sencillo en un dominio científico intensivo en datos como la bioinformática. Este flujo de trabajo, que se utiliza para encontrar posibles enfermedades a partir de una serie de términos introducidos por los usuarios, los cuadrados representan tanto las entradas como las salidas de datos (R representa los datos existentes en la memoria y F los datos almacenados en un archivo), los elementos rectangulares redondeados representan las actividades específicas del dominio y, por último, las flechas indican las dependencias entre los datos.

Otro dominio importante de aplicación de este tipo de flujos de trabajo es la ciencia de datos [37]. Los científicos de datos pueden sacar provecho de esta tecnología para facilitar la definición de sus procesos de análisis y descubrimiento de conocimiento relevante procedente de fuentes de datos heterogéneas y de gran tamaño.

En general, cuando un flujo de trabajo se utiliza en un dominio intensivo en datos puede ser conocido como flujo de trabajo intensivo en datos [152]:

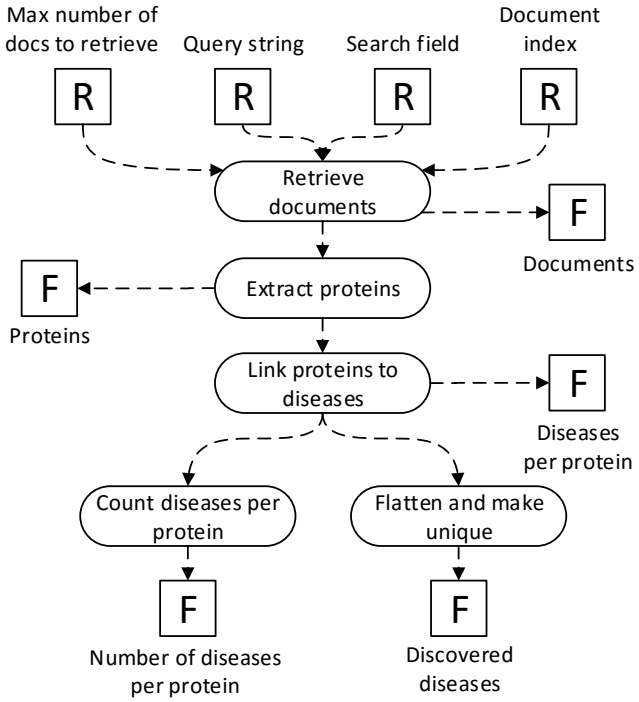


Figura 2.2: Ejemplo de un flujo de trabajo en bioinformática (extraído de [121]).

Un **flujo de trabajo intensivo en datos** o DIW en un mecanismo para el modelado, gestión y procesamiento de aplicaciones intensivas en datos. Técnicamente, está compuesto por un conjunto de tareas intensivas en cómputo o datos, así como las dependencias entre ellas. Durante la ejecución, cada tarea debe ser instanciada para su ejecución.

Diferencias entre flujos de trabajo de negocio e intensivos en datos

Un flujo de trabajo de negocio y un DIW presentan importantes diferencias debido a los requisitos de los dominios donde se utilizan, como la importancia del procesamiento y análisis de datos o la necesidad de la automatización de procesos. Esencialmente, esta diferencia en los requisitos hace que utilicen paradigmas de ejecución sustancialmente distintos, lo que resulta en diferencias fundamentales entre ellos [93].

Los flujos de trabajo de negocio modelan un entendimiento común del proceso de negocio dentro de las organizaciones, coordinando las actividades humanas y las tareas informáticas triviales (por ejemplo, enviar un correo electrónico o descargar un archivo) de acuerdo con la lógica de negocio. Por lo tanto, la orquestación eficaz de los procesos de negocio es un elemento clave en un entorno empresarial competitivo para lograr el objetivo de negocio en un esfuerzo colectivo de muchos participantes.

Sin embargo, los flujos de trabajo científicos definen el conjunto de tareas que deben ejecutarse automáticamente para realizar un experimento intensivo en datos a gran escala. Dichos experimentos suelen estar compuestos por una variedad de tareas de manipulación de datos, como el movimiento de datos, la transformación de datos, el análisis de datos y la visualización de datos. Por ello, la gestión eficiente de la ejecución es un aspecto clave en este tipo de flujos de trabajo.

Por otro lado, los DIW se iteran y evolucionan analizando el rendimiento de muchas ejecuciones. La naturaleza exploratoria de la definición y ejecución de procesos intensivos en datos a gran escala hace que la estructura de este tipo de flujos de trabajo sufra modificaciones con frecuencia debido al uso del método de prueba y error. Por el contrario, los flujos de trabajo de negocio se ejecutan de forma rutinaria, debido a una lógica de negocio más permanente. Esta lógica de negocio define el orden de ejecución de las tareas, lo que se conoce como flujo de trabajo dirigido por la lógica de negocio (*control-driven workflow*). Por su parte, el orden de ejecución en un flujo de trabajo científico se deriva de las dependencias de datos existentes entre los mismos, lo que es conocido como flujo de trabajo dirigido por los datos (*data-driven workflow*).

Además, la definición de los flujos de trabajo de negocio suele correr a cargo de informáticos e ingenieros, que utilizan aplicaciones informáticas sencillas en su trabajo. Sin embargo, los flujos de trabajo científicos son definidos por científicos de datos y expertos en computación, expertos en sus correspondientes dominios intensivos en datos. Además, se requieren conocimientos especializados en conceptos informáticos avanzados, como en creación de redes o en computación distribuida, que son necesarios para llevar a cabo procesos intensivos en datos.

Los distintos requisitos que tratan de modelar, los conocimientos requeridos a los profesionales típicamente encargados de su definición, así como los distintos paradigmas de ejecución subyacentes suponen algunas de las diferencias más relevantes

entre ambos tipos de flujos de trabajo (véase resumen en Tabla 2.1). Tradicionalmente el desarrollo de la tecnología de los flujos de trabajo ha estado orientado a los requisitos de los procesos de negocio, lo que resulta en que la mayoría de herramientas y mecanismos no sean aplicables a los DIW. Por ello, es deseable disponer de soluciones específicas para dar soporte a los requisitos propios de los DIW [93].

	Flujos de trabajo de negocio	Flujos de trabajo intensivos en datos
Objetivo	Procesos de negocio	*Data pipelines*
Modelado	Actividades manuales e informáticas triviales	Tareas computacionales intensivas en datos
Expertos	Ingenieros y especialistas en informática	Científicos de datos
Tipo de ejecución	Dirigida por la lógica de negocio	Dirigida por los datos
Naturaleza	Estable	Cambiante

Tabla 2.1: Resumen de las principales diferencias entre tipos de flujos de trabajo.

Representación de los flujos de trabajo

Cualquier definición de un flujo de trabajo se hace acorde a un lenguaje de flujo de trabajo [155]. Un lenguaje de flujo de trabajo está formado por un conjunto de constructores, estructuras de control, conectores y reglas semánticas que permiten la especificación los diferentes requisitos en cualquier dominio en particular:

> Un **lenguaje de flujos de trabajo** es un lenguaje específico de dominio que permite definir la información relevante de las aplicaciones basadas en el flujo de trabajo.

La mayoría de los lenguajes de flujos de trabajo son representaciones de alto nivel, lo que significa que ofrecen una visión abstracta de los procesos y tareas que deben ejecutarse. En algunos casos, también existen lenguajes que ofrecen representaciones diferentes para la versión abstracta del flujo de trabajo, y su versión concreta y ejecutable. Uno de los lenguajes más conocidos es BPMN (*Business Process Model*

and Notation)[4], ampliamente utilizado en la gestión de procesos de negocio. BPMN utiliza elementos gráficos estandarizados, como tareas y eventos, para representar de manera clara y comprensible los distintos aspectos de un proceso de negocio. Esto facilita la comunicación entre los equipos de negocios y de tecnología, mejorando la alineación de los procesos con los objetivos organizacionales. En dominios intensivos en datos, la complejidad de las operaciones requiere de lenguajes especializados para describir flujos de trabajo adaptados a la gestión y procesamiento de estos datos, así como a la ejecución en entornos de computación de alto rendimiento, como SCUFL (*Simple Conceptual Unified Flow Language*) [104] o DAX (*Directed Acyclic graph in XML*) [33].

2.1.4. Sistemas de gestión de flujos de trabajo científicos

Desde una perspectiva general, cualquier flujo de trabajo puede ser automatizado y gestionado por un sistema de gestión de flujos de trabajo o *workflow management system* (WfMS). Un WfMS es un sistema para diseñar, ejecutar y gestionar flujos de trabajo mediante la coordinación de los recursos disponibles (tanto computacionales como humanos) dentro de un entorno común [86]:

> Un **sistema de gestión de flujos de trabajo** es un sistema que define, crea y gestiona la ejecución de flujos de trabajo mediante el uso de software, que se ejecuta en uno o más motores de ejecución de flujo de trabajo, y que es capaz de interpretar la definición del proceso, interactuar con los participantes en el flujo de trabajo y, en caso necesario, invocar el uso de herramientas y aplicaciones informáticas.

Un WfMS da soporte al diseño de flujos de trabajo e, idealmente, no requiere que el creador de flujos de trabajo sea experto en computación. La gestión de los recursos necesarios, así como el acceso a los datos o la planificación y coordinación de las tareas, es delegada en un motor de ejecución. Este motor interpreta la lógica definida para automáticamente crear instancias ejecutables y optimizadas de los flujos de trabajo abstrayendo cualquier detalle de bajo nivel, como el paradigma de

[4]Object Management Group: Business Process Model and Notation (BPMN) 2.0.2. https://www.omg.org/spec/BPMN/2.0.2 (consultado: 14/11/2023)

ejecución (secuencial, paralelo o distribuido) o el tipo de plataforma computacional (computación en *grid* o *cloud*, entre otros).

Este tipo de sistemas son cada vez más utilizados en la comunidad científica para la creación y gestión de los flujos de trabajo científicos gracias a la facilidad para crear aplicaciones que utilizan y coordinan distintos tipos de recursos distribuidos, reducen los costes de ejecución y favorecen la colaboración [157]:

> Los **sistemas de gestión de flujos de trabajo científicos** o *scientific workflow management system* (SWfMS) han surgido como un paradigma para ayudar a los científicos a prototipar y ejecutar tanto experimentos como simulaciones computacionales, ayudándoles a representar y gestionar miles de tareas, docenas de repositorios de datos y recursos distribuidos.

Por tanto, un SWfMS es una herramienta eficiente para ejecutar DIW y gestionar grandes cantidades de datos. Además, brindan un fácil acceso a numerosas tecnologías de cómputo complejas sin necesidad de conocer el esfuerzo computacional relacionado con redes, sistemas de alto rendimiento o computación distribuida. Este tipo de herramientas normalmente permiten la representación visual y proporcionan mecanismos para reutilizar, parcial o totalmente, un flujo de trabajo con poco esfuerzo.

La utilidad de este tipo de sistemas depende en gran medida de las funciones disponibles para un dominio específico y la potencial capacidad de integración con herramientas de ese mismo dominio o dominios similares. La naturaleza diversa de las tareas realizadas en diferentes dominios intensivos en datos, como el procesamiento de imágenes, el análisis de datos geoespaciales o el acceso a fuentes de datos distribuidas, ha llevado al desarrollo de distintas soluciones adaptadas a necesidades específicas [30]. Como resultado, hasta la fecha existe una gran variedad de herramientas con distintas funcionalidades.

Ejemplos de sistemas de gestión de flujos de trabajo científicos

Algunos ejemplos destacados de SWfMS han sido desarrollados y utilizados en diversos contextos científicos, desde la bioinformática y la física, hasta la astronomía y la

meteorología. Cada uno de estos sistemas proporciona una serie de funcionalidades orientadas a abordar los desafíos específicos de sus respectivos dominios.

Kepler [92]: Este SWfMS está diseñado para crear, ejecutar y compartir modelos y análisis en un amplio rango de dominios científicos e ingenieriles. Un flujo de trabajo en Kepler está compuesto por un conjunto de componentes configurables, conocidos como *actores*, cuya ejecución es controlada por un componente especial llamado *director*. Se incluyen diversos tipos de actores para realizar tareas genéricas relacionadas con la conversión de tipos de datos, manipulación de matrices y cadenas de texto, operaciones con archivos, conexiones a bases de datos y visualización de datos, entre otras. Además, se incluyen actores para ejecutar fragmentos de código escritos en diferentes lenguajes de propósito general o *general-purpose programming languages* (GPL), como Java, R, Matlab, Groovy, JavaScript y Python. También permite invocar tanto flujos de trabajo anidados como servicios web (REST y SOAP/WSDL) y programas de línea de comandos.

Taverna [105]: Es un SWfMS independiente del dominio que proporciona un conjunto de herramientas orientadas a diseñar y ejecutar flujos de trabajo científicos y facilitar la experimentación *in silico*. Un flujo de trabajo en Taverna está compuesto por un conjunto de componentes configurables (procesadores) y conexiones que permiten la integración de diferentes servicios y procesos. Los procesadores de Taverna admiten la definición y ejecución de fragmentos de código escritos en Java y R (mediante un editor de texto integrado), así como la invocación de programas de interfaz de línea de comandos o *command-line interface* (CLI), otros flujos de trabajo de Taverna (es decir, flujos de trabajo anidados) y servicios web (REST y SOAP/WSDL). Además, Taverna ofrece un conjunto de procesadores preconfigurados que se pueden utilizar para realizar tareas genéricas, como la manipulación de cadenas de texto, operaciones de archivos, consultas SQL o manipulación de XML.

Pegasus/WINGS [58]: Este SWfMS combina dos herramientas para la creación, ejecución y gestión de flujos de trabajo científicos. Pegasus y WINGS. Pegasus proporciona la capacidad de ejecutar flujos de trabajo complejos que involucran múltiples tareas y dependencias entre ellas, mientras que WINGS ofrece una interfaz

gráfica intuitiva para diseñar flujos de trabajo mediante la selección de componentes y la definición de relaciones entre ellos. La creación de flujos de trabajo con Pegasus y WINGS se realiza en tres etapas: la primera define la estructura abstracta del flujo de trabajo y crea una plantilla; la segunda especifica los datos que serán utilizados en el flujo de trabajo y crea una instancia del mismo; finalmente, la tercera etapa se encarga de especificar las réplicas de datos y sus ubicaciones para formar un flujo de trabajo ejecutable.

VisTrails [52]: Es un SWfMS que está diseñado para gestionar flujos de trabajo en constante evolución tales como simulaciones, análisis de datos y visualización. A medida que un ingeniero o científico genera y evalúa hipótesis sobre los datos en estudio, se crean una serie de flujos de trabajo diferentes, aunque relacionados, mientras se ajusta el flujo de trabajo en un proceso interactivo. VisTrails admite la creación y ejecución de flujos de trabajo que contienen distintos recursos como bibliotecas, servicios en *grid* y servicios web. Estos flujos de trabajo se pueden ejecutar de forma interactiva a través de la interfaz gráfica o por lotes utilizando un servidor VisTrails.

LONI Pipeline [114]: Este SWfMS está orientado principalmente para científicos computacionales, y les permite crear rápidamente flujos de trabajo aprovechando las herramientas disponibles en neuroimagen, genómica o bioinformática, entre otros campos. Un flujo de trabajo en LONI Pipeline está compuesto por un conjunto de módulos que permiten la ejecución tanto de programas CLI como de *scripts* en Bash definidos en un editor propio. Además, permite la invocación de flujos de trabajo anidados y servicios web SOAP/WSDL. Para ampliar las capacidades del sistema, los usuarios pueden utilizar módulos y flujos de trabajo de LONI Pipeline.

Triana [21]: Es un SWfMS que proporciona una interfaz visual para facilitar la composición de aplicaciones científicas. Un flujo de trabajo en Triana consiste en un conjunto de unidades de ejecución (o componentes) y un conjunto de flujos que determinan el orden de ejecución. Dado que un componente en Triana es una clase Java que realiza una tarea específica, y su definición está codificada en XML (especificando el nombre, las entradas y salidas, y los parámetros de configuración),

los desarrolladores pueden ampliar el sistema integrando sus propios programas en Java. Además, Triana proporciona una amplia variedad de componentes listos para usar (cajas de herramientas) para diferentes propósitos, que van desde el análisis de señales hasta la manipulación de imágenes, cuyo número puede aumentar mediante la reutilización de flujos de trabajo de Triana.

ASKALON [50]: Es un SWfMS que proporciona un entorno de desarrollo y ejecución de flujos de trabajo científicos que se enfoca en aplicaciones intensivas en recursos y cálculos en sistemas de computación de alto rendimiento, como clústeres y supercomputadoras. Proporciona una plataforma flexible y escalable que permite a los científicos crear, planificar y ejecutar flujos de trabajo complejos y distribuidos en infraestructuras computacionales heterogéneas. Además, se encarga de la planificación y optimización automatizada de flujos de trabajo, la monitorización y gestión de tareas, así como de la generación de informes detallados sobre el rendimiento y los resultados de la propia ejecución. Un flujo de trabajo en ASKALON representa un proceso científico o analítico que consta de un conjunto de tareas interconectadas y dependientes entre sí. Cada tarea representa una unidad de cómputo que realiza una operación o análisis específico. Estas tareas se organizan en una secuencia lógica que define el flujo de ejecución del proceso.

Galaxy [60]: Este SWfMS permite la gestión, análisis y visualización de datos en el campo de la bioinformática y la genómica. Proporciona una interfaz web intuitiva y fácil de usar que permite a los científicos, investigadores y analistas realizar análisis complejos de datos genómicos sin la necesidad de conocimientos avanzados de programación. En Galaxy, los usuarios pueden construir flujos de trabajo personalizados que combinan diferentes herramientas y algoritmos para realizar análisis específicos en sus datos. Estos flujos de trabajo pueden ser compartidos con otros investigadores, lo que fomenta la colaboración y el intercambio de conocimientos en la comunidad científica. Además, Galaxy proporciona herramientas bioinformáticas preconfiguradas que cubren una amplia gama de análisis genómicos, como alineamiento de secuencias, identificación de variantes genéticas y anotación de genes, entre otros.

Nextflow [40]**:** Este SWfMS es una plataforma para el desarrollo y ejecución de flujos de trabajo científicos. Está diseñada para abordar los desafíos de la computación distribuida y el análisis de datos a gran escala, permitiendo a los científicos de datos crear flujos de trabajo complejos que pueden ejecutarse en entornos de infraestructura heterogéneos, como clústeres, *grids* o *cloud*. Su principal característica es el enfoque centrado en la portabilidad y reproducibilidad de los flujos de trabajo. Los flujos de trabajo se definen mediante un lenguaje declarativo que facilita la especificación de las tareas y sus dependencias, lo que permite a los usuarios describir el proceso científico de manera clara y concisa. Además, se encarga de gestionar automáticamente la distribución y ejecución de las tareas en los recursos disponibles, lo que garantiza una ejecución eficiente y escalable. También se integra con una amplia gama de herramientas y servicios externos, como bibliotecas especializadas, contenedores de Docker o servicios web, lo que amplía la funcionalidad y flexibilidad de los flujos de trabajo.

KNIME[5]**:** Es un SWfMS para la creación, integración y ejecución de flujos de trabajo de análisis de datos. Permite a los científicos de datos, analistas y desarrolladores diseñar flujos de trabajo complejos para el procesamiento, transformación, análisis y visualización de datos de manera eficiente y escalable. Para ello, proporciona una interfaz gráfica intuitiva basada en nodos y conexiones, lo que permite a los usuarios crear flujos de trabajo arrastrando y soltando los nodos que representan las operaciones y tareas específicas. Estos nodos pueden ser desde simples filtros y transformaciones de datos hasta algoritmos de aprendizaje automático y visualizaciones avanzadas. Una vez configurado el flujo de trabajo, se puede ejecutar en tiempo real para analizar los datos y obtener información detallada sobre la ejecución de cada nodo. Además, KNIME permite integrar una amplia gama de herramientas y algoritmos externos, lo que brinda a los usuarios acceso a una gran cantidad de funcionalidades y bibliotecas especializadas.

ClowdFlows [80]**:** Es un SWfMS basado en *cloud* para la creación, ejecución y compartición de flujos de trabajo analíticos que permite a los científicos de datos, analistas y profesionales de diferentes disciplinas diseñar y automatizar tareas

[5]KNIME (*Konstanz Information Miner*). https://www.knime.com/ (consultado: 01/10/2023).

complejas de procesamiento y análisis de datos. ClowdFlows está orientado a la colaboración haciendo que los usuarios puedan acceder y compartir sus flujos de trabajo, lo que fomenta la colaboración y el intercambio de conocimientos entre diferentes equipos. Además, proporciona un conjunto de flujos de trabajo predefinidos y de módulos totalmente reutilizables, lo que reduce y simplifica los tiempos de creación de flujos de trabajo complejos. Estos flujos de trabajo están destinados a ser ejecutados en la nube, sacando ventaja de la potencia de procesamiento y almacenamiento de los servidores distribuidos, permitiendo el análisis de grandes volúmenes de datos y la colaboración en proyectos de investigación a gran escala.

Dataiku[6]**:** Este SWfMS es una plataforma de ciencia de datos y análisis avanzado que ofrece un entorno colaborativo y escalable para la preparación, análisis y visualización de datos. Es una herramienta que permite a científicos de datos, analistas y equipos de negocio trabajar juntos en proyectos de datos complejos. Proporciona una interfaz gráfica que permite a los usuarios explorar y preparar datos de manera eficiente sin necesidad de saber programación. También ofrece una amplia gama de herramientas y algoritmos de aprendizaje automático que facilitan la creación de modelos predictivos y descriptivos, así como permite a los usuarios utilizar lenguajes de programación como Python o R para realizar análisis más avanzados.

RapidMiner[7]**:** Es un SWfMS para la ciencia de datos, el análisis de datos y el aprendizaje automático que permite a los científicos de datos, analistas y profesionales de diferentes sectores explorar, preprocesar, modelar y visualizar datos de manera eficiente. Proporciona una interfaz gráfica donde los usuarios pueden arrastrar y soltar nodos para construir flujos de trabajo de análisis de datos sin la necesidad de programación. RapidMiner ofrece una amplia variedad de herramientas y algoritmos de aprendizaje automático para realizar tareas como clasificación, regresión, agrupamiento y análisis de texto, entre otras. También permite la integración con lenguajes de programación como R y Python, lo que amplía la funcionalidad y flexibilidad de la plataforma. Además, proporciona una amplio rango de opciones de visualización para representar los resultados del análisis de datos, lo que facilita la interpretación de los resultados y la toma de decisiones informadas basadas en el análisis de datos.

[6]Dataiku. https://www.dataiku.com/ (consultado: 01/10/2023).
[7]RapidMiner. https://rapidminer.com/ (consultado: 01/10/2023).

Apache Airflow[8]**:** Este SWfMS es una herramienta para la programación, monitorización y gestión de flujos de trabajo y tareas automatizadas que permite a los desarrolladores y analistas crear flujos de trabajo complejos que involucran una secuencia de tareas y dependencias entre ellas. Apache Airflow permite definir flujos de trabajo mediante una interfaz de línea de comandos y mediante una interfaz de web utilizando un lenguaje declarativo basado en Python. Proporciona una amplia gama de operadores predefinidos que representan diferentes tipos de tareas, como ejecutar comandos CLI, enviar correos electrónicos, extraer y cargar datos, ejecutar consultas SQL, entre otras. Además, permite a los usuarios crear sus propios operadores personalizados para adaptar la herramienta a sus necesidades específicas. El motor de ejecución de Apache Airflow es altamente escalable y puede ejecutarse en un clúster distribuido, lo que permite gestionar flujos de trabajo complejos y de alto rendimiento.

Principal características de los sistemas de flujos de trabajo científicos

En términos generales, cada SWfMS ofrece un conjunto específico de características diseñadas para abordar los requisitos particulares de los dominios intensivos en datos y los casos de uso específicos que los desarrolladores consideran más relevantes para sus respectivas problemáticas. Estas herramientas han sido especialmente concebidas para aplicaciones en campos como la ciencia de datos y la *e-Science*, abordando desafíos asociados con la manipulación y análisis de grandes conjuntos de datos.

Las funcionalidades clave que ofrecen estos SWfMS están vinculadas al tipo de entorno de desarrollo que proporcionan, ya sea un editor de tipo textual o gráfico. Esto hace que la curva de aprendizaje sea más o menos pronunciada al requerir conocimientos específicos. Además, su capacidad de integración con plataformas de computación de alto rendimiento varía, permitiendo ejecuciones tanto en entornos locales como en entornos distribuidos como *grid*, *cluster* o *cloud*. Asimismo, la utilización de lenguajes específicos de flujos de trabajo es común, facilitando la representación y ejecución de las tareas dentro del sistema.

Otra característica importante es la capacidad de extensión de estas herramientas, que posibilita a los usuarios personalizar y adaptar las funcionalidades existentes

[8]Apache Airflow. `https://airflow.apache.org/` (consultado: 01/10/2023).

	Año de creación	**¿Es *open source*?**	**¿Proyecto activo?**	**Dominio**	**¿Es *low-code* o *no-code*?**
Kepler	2002	Sí	No	*e-Science*	*Low-code*
Taverna	2004	Sí	No	*e-Science*	*Low-code*
Pegasus/ WINGS	2001	Sí	Sí	*e-Science*	*Low-code*
VisTrails	2005	Sí	No	*e-Science*	*Low-code*
LONI Pipeline	2003	No	No	*e-Science*	*Low-code*
Triana	2003	Sí	No	*e-Science*	*Low-code*
ASKALON	2003	No	No	*e-Science*	*Low-code*
Galaxy	2005	Sí	Sí	*e-Science*	*Low-code*
Nextflow	2013	Sí	Sí	*e-Science*	*Low-code*
KNIME	2006	Sí	Sí	Ciencia de datos	*Low-code*
ClowdFlows	2010	Sí	No	Ciencia de datos	*No-code*
Dataiku	2013	Sí	Sí	Ciencia de datos	*Low-code*
RapidMiner	2006	No	Sí	Ciencia de datos	*Low-code*
Apache Airflow	2014	Sí	Sí	Ciencia de datos	*Low-code*

Tabla 2.2: Resumen de características de sistemas de flujos de trabajo intensivos en datos existentes.

para abordar nuevos casos de uso. Sin embargo, es importante destacar que esta flexibilidad a menudo requiere conocimientos en lenguajes de programación, lo que puede limitar su accesibilidad y aplicabilidad en dominios donde los profesionales carecen de experiencia en desarrollo de software.

En la Tabla 2.2 se muestra un resumen de las características relacionadas con su desarrollo y dominio de aplicación. Por su parte, la Tabla 2.3 proporciona un resumen de estas características de los SWfMS estudiados en cuanto a sus funcionalidades.

	Tipo de editor	**Tipo de motor**	**Lenguaje de flujos de trabajo**	**Extensibilidad**
Kepler	Gráfico	Local, *grid*	MoML	Java, R, Python Groovy, JavaScript
Taverna	Gráfico	Local, *cloud*	SCUFL	Java, R, CLI
Pegasus/ WINGS	Gráfico	Local, *grid*	DAX	CLI
VisTrails	Gráfico	Local	Basado en XML	Python
LONI Pipeline	Gráfico	Local, *grid*	Basado en XML	CLI
Triana	Gráfico	Local, *grid*	Basado en XML	Java
ASKALON	Textual	Local, *cluster*, *grid*, *cloud*	AGWL	CLI
Galaxy	Gráfico	Local, *cloud*	Basado en XML	CLI
Nextflow	Textual	Local, *cluster*, *grid* *cloud*	Propio	Groovy, Python Perl, CLI
KNIME	Gráfico	Local, *grid*, *cloud*	Basado en XML	Java
ClowdFlows	Gráfico	*Cloud*	Desconocido	No
Dataiku	Gráfico	*Cloud*	Basado en XML	Python, R
RapidMiner	Gráfico	Local, *grid*, *cloud*	Desconocido	CLI
Apache Airflow	Textual	Local, *cluster*, *cloud*	Python	Python

Tabla 2.3: Resumen de características de sistemas de flujos de trabajo intensivos en datos existentes.

2.2. Lenguajes de modelado específicos de dominio

Los lenguajes de modelado específicos de dominio permiten la creación de modelos que se adaptan a los requisitos y restricciones de un dominio particular. Estos lenguajes, que presentan un enfoque centrado en el dominio de aplicación, son diseñados bajo los preceptos de la ingeniería de software dirigida por modelos.

2.2.1. Ingeniería de software dirigida por modelos

La ingeniería de software dirigida por modelos o *model-driven software engineering* (MDSE) es un paradigma de desarrollo de software útil para la implementación integral de sistemas informáticos. El objetivo principal de este paradigma es promover el uso de modelos y sus transformaciones [135] para generar software mantenible que pueda incorporar fácilmente nuevos requisitos y evolucionar al mismo tiempo que surgen nuevas necesidades. Por tanto:

> La **ingeniería de software dirigida por modelos** es un paradigma que fomenta el uso sistemático de modelos como elemento central a lo largo de todo el ciclo de vida del desarrollo de software.

En este contexto, un modelo es considerado una representación simplificada de una determinada realidad con el objetivo de comprenderla mejor. Para lograr esta abstracción, en el modelo se prescinden de los detalles irrelevantes, independientemente de que este se represente mediante una notación textual o gráfica. Cuando se utiliza una representación gráfica de un modelo se hace en forma de diagrama. Según [147], un modelo es definido como:

> Un **modelo** es una especificación formal de la función, estructura y comportamiento de un sistema en un contexto determinado y desde un punto de vista específico (o punto de referencia).

De esta manera, es posible aumentar la productividad y reducir el *time-to-market* al permitir el desarrollo de sistemas complejos por medio de conceptos que están mucho

más cerca del dominio del problema en lugar de estar ligados a la tecnología concreta en la que se implementará. Esto hace que los modelos sean más fáciles de definir, comprender y mantener por parte del experto en el dominio de aplicación [136], lo que ayuda a la compresión de problemas complejos y sus posibles soluciones a través de abstracciones.

Object Management Group (OMG)[9] aboga por la separación entre la especificación de la funcionalidad de un sistema y su implementación en una plataforma tecnológica específica. Para lograr esta separación efectiva, se definen cuatro niveles de abstracción que representan distintos puntos de vista en relación con el sistema o sus componentes. Estos niveles proporcionan una jerarquía de modelos que facilita la comprensión y el diseño del sistema en distintos grados de detalle:

- Modelo independiente de la computación o *computation independent model* (CIM): Un modelo CIM no define ningún aspecto de la estructura del sistema, ocultando todas las especificaciones relacionadas con la tecnología con el fin de ser independiente de su implementación. Este tipo de modelo suele denominarse modelo de negocio o de dominio porque utiliza un vocabulario que resulta familiar a los expertos en la materia. Por tanto, resulta muy útil para ayudar a la comprensión del problema y para definir un vocabulario común para el resto de modelos. El modelo CIM desempeña un papel importante para reducir la brecha de conocimiento que suele existir entre los expertos del dominio del problema y los expertos responsables de implementar el sistema.

- Modelo independiente de la plataforma o *platform independent model* (PIM): Un modelo PIM presenta un nivel de independencia suficiente para poder ser usado en diferentes plataformas tecnológicas similares. Con este modelo se define la lógica del sistema, y sus interacciones, de forma independiente al tipo de tecnología que implementará cada parte y cómo se usará sobre una plataforma concreta.

- Modelo específico de plataforma o *platform specific model* (PSM): Un modelo PSM es una vista del sistema pensada para una plataforma tecnológica específica. Este modelo combina la especificación del modelo PIM con los detalles que definen cómo dicho sistema utiliza un determinado tipo de plataforma.

[9]Object Management Group (OMG). https://www.omg.org/ (consultado: 05/09/2023).

Así, OMG define el proceso del desarrollo dirigido por modelos como una serie de actividades que, de manera simplificada, se pueden describir de la siguiente manera:

- Construcción del PIM: En esta etapa, se crean modelos PIM que representan la funcionalidad del sistema de manera independiente de la plataforma. Estos modelos reflejan la lógica del negocio descrita por los modelos CIM que capturan los aspectos esenciales del dominio del problema.
- Selección del PSM: Se elige el modelo PSM que se adapta mejor a los requisitos del sistema según los detalles tecnológicos específicos necesarios para la implementación del mismo.
- Transformación del PIM a PSM: En esta fase, se realiza la transformación del modelo PIM al modelo PSM teniendo en cuenta los detalles de implementación y el diseño a la plataforma específica.
- Transformación del PSM a código ejecutable: La etapa final del proceso implica convertir los modelos PSM en código ejecutable. Este paso conlleva la generación de código que puede ejecutarse en la plataforma tecnológica seleccionada. La automatización de esta transformación garantiza la coherencia entre el diseño y la implementación, reduciendo posibles errores y mejorando la eficiencia del desarrollo.

En este proceso, las transformaciones de modelos juegan un papel fundamental, actuando como el puente que conecta los diferentes niveles de abstracción y permitiendo la evolución del modelo desde una fase a otra.

2.2.2. Transformaciones de modelos

Los modelos no son meros contenedores de documentación, sino que son artefactos que contienen información precisa, por lo que pueden ser utilizados y gestionados automáticamente para modificar su definición y/o crear nuevos modelos (e incluso código) [14]. De esta manera, la transformación de modelos es el mecanismo que permite ahorrar esfuerzo y reducir los errores automatizando la creación y la modificación de software en la medida de lo posible. Una transformación de modelos es entendida por [78] como:

Una **transformación de modelos** es la generación automática de un modelo de destino a partir de un modelo de origen. Para ello, la definición de transformación consiste en un conjunto de reglas de transformación que describen cómo un modelo en el lenguaje de origen puede transformarse en un modelo en el lenguaje de destino. Una regla de transformación es la declaración de cómo uno o más elementos en el lenguaje origen se transforman en uno o más elementos en el lenguaje de destino.

Por tanto, la transformación de modelos se traduce en el proceso de convertir uno o varios modelos en otros artefactos de modelado o implementación. En la Figura 2.3 se muestra un ejemplo de una transformación de modelos, donde un modelo entidad/relación origen se convierte en un modelo de clases equivalente manteniendo la semántica del modelo origen [85].

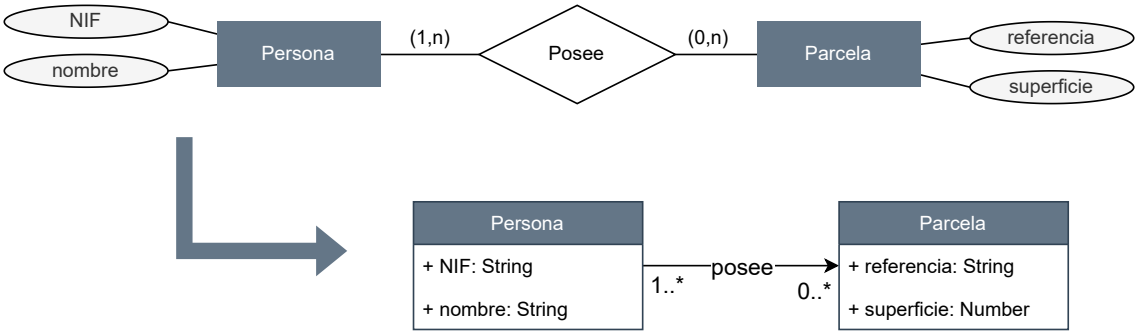


Figura 2.3: Transformación de un modelo entidad/relación a un modelo de clases.

Aunque las transformaciones entre modelos son las más habituales, hay que considerar otros tipos [38] que involucran artefactos basados en texto, y que resultan útiles para determinados problemas. Así, en función de los tipos de artefactos de entrada y salida, las transformación de modelos pueden clasificarse en tres categorías principales:

- Modelo a modelo o *model-to-model* (M2M): Este tipo de transformaciones es el más utilizado. Un ejemplo típico es el ilustrado en la Figura 2.3, donde el artefacto de entrada es un modelo que tienen que ser convertido en un modelo de salida. Por tanto, en las transformaciones M2M se producen nuevos modelos a partir de otros modelos.

- Modelo a texto o *model-to-text* (M2T): Este tipo de transformaciones se utiliza típicamente para tender puentes entre los lenguajes de modelado y los lenguajes de programación gracias a la generación automática de código. Sin embargo, se pueden utilizar para otros propósitos, como la generación de informes, documentación técnica, o incluso diagramas. En general, este tipo de transformaciones produce ficheros de texto, como código, reportes o documentación, a partir de modelos.

- Texto a modelo o *text-to-model* (T2M): Este tipo de transformaciones se emplea habitualmente para realizar tareas de ingeniería inversa, como la modernización de aplicaciones heredadas (también conocidas como *legacy*), las cuales se transforman en modelos a partir de los cuales poder generar otros modelos o código equivalente escrito en lenguajes de programación modernos. Por tanto, este tipo de transformaciones produce modelos a partir de ficheros de texto.

Las reglas de las transformaciones de modelos pueden ser definidas por GPL como Java o C. Sin embargo, los requisitos específicos de este tipo de transformaciones hacen que sea más idóneo el uso de lenguajes específicos para transformaciones de modelos, denominados "lenguajes de transformación de modelos":

> Un **lenguaje de transformación de modelos** o *model transformation language* (MTL) es un lenguaje diseñado con el único objetivo de permitir la especificación de las reglas de transformación que son necesarias para realizar una transformación de modelos.

En este sentido, los MTL ofrecen constructores y abstracciones específicas que simplifican la definición de reglas de transformación. Estos elementos se centran en la declaración de relaciones entre distintos elementos de los modelos. La capacidad de estos lenguajes para modelar de manera precisa las reglas de transformación contribuye directamente a la legibilidad y mantenibilidad de las transformaciones.

De la misma manera que ocurre con los GPL, los MTL pueden diferir unos de otros en varios aspectos [72], como el paradigma del lenguaje (declarativo o imperativo) o el tipado, hasta aspectos específicos de la propia transformación, como la direccionalidad o el nivel de abstracción. A continuación se detallan las principales

características de las transformaciones según su nivel de abstracción y su direccionalidad:

- Transformaciones de modelos según el nivel de abstracción: Las transformaciones de modelos se pueden clasificar en verticales y horizontales. En las transformaciones verticales los modelos origen y los de destino se encuentran en diferentes niveles de abstracción. Por ejemplo, la transformación de objetos del metamodelo a objetos del modelo de dominio. Por otro lado, en las transformaciones horizontales los modelos de origen y los de destino pertenecen al mismo nivel de abstracción. Un ejemplo es en la transformación de la descripción del modelo de una notación a otra, como se puede observar en la Figura 2.3.
- Transformaciones de modelos según su direccionalidad: Desde el punto de vista de la direccionalidad, las transformaciones de modelos se clasifican en dos grupos principales: unidireccionales y bidireccionales [101]. Una transformación unidireccional se realiza en una única dirección, desde el modelo origen al modelo destino. Sin embargo, en una transformación bidireccional puede llevarse a cabo desde el modelo origen al modelo destino o desde el destino hacia el origen. Uno de los grandes retos de las transformaciones bidireccionales es mantener la consistencia de dos (o más) modelos y permitir la sincronización entre ellos [142].

Tomando todas las características descritas, actualmente existen numerosos MTL dirigidos a abordar estos escenarios de transformación.

QVT (Query/View/Transformation) [76]: Es un MTL estándar definido por OMG para transformaciones M2M que se basa en tres tipos de elementos: la consulta (una expresión evaluada sobre un modelo que devuelve un conjunto de objetos que cumplen con la restricción impuesta por la propia consulta), las vistas (modelos que se derivan de otros modelos) y las transformaciones (especificaciones cuya ejecución genera un modelo de destino a partir de un modelo de origen). QVT proporciona un conjunto de reglas y constructores que permiten definir operaciones de transformación, tanto unidireccionales como bidireccionales, entre modelos de manera declarativa.

ATL (ATLAS Transformation Language) [4]: Es un MTL para transformaciones unidireccionales M2M desarrollado por el grupo ATLAS (INRIA & LINE) para Eclipse[10]. Este lenguaje proporciona un entorno de desarrollo integrado con otras funcionalidades como editores dedicados, depuradores de sintaxis, etc. ATL es principalmente un lenguaje declarativo, aunque se permiten constructores imperativos y orientados a objetos. Este enfoque imperativo facilita la expresión de lógica de transformación compleja y la manipulación de modelos a través de patrones y reglas definidas por el usuario.

JTL (Janus Transformation Language) [22]: Es un MTL para transformaciones M2M específicamente diseñado para soportar bidireccionalidad y propagación de cambios. JTL es un lenguaje declarativo y orientado a resolver problemas de búsqueda difíciles, principalmente aquellos que presenta un nivel de complejidad *NP-hard*.

MOFM2T (MOF Model to Text)[11]: Es un MTL de OMG para transformaciones M2T que permite convertir información contenida en modelos a representaciones textuales específicas, facilitando la generación automatizada de código, documentación u otros tipos de salida basados en modelos. Su especificación ha sido implementada por Acceleo[12], una herramienta desarrollada en Eclipse para la generación automática de código.

2.2.3. Metamodelado de lenguajes específicos de dominio

Todo modelo se expresa mediante un lenguaje de modelado, tal y como se hace con el lenguaje unificado de modelado (UML) para crear modelos de diseño, o con los lenguajes de programación para desarrollar código fuente. Un metamodelo es definido como [23]:

[10] Eclipse Modeling Project. https://www.eclipse.org/modeling/ (consultado: 09/09/2023)
[11] OMG. MOF Model to Text Transformation Language (MOFM2T), v1.0. https://www.omg.org/spec/MOFM2T/1.0/PDF (consultado: 09/09/2023)
[12] Acceleo. https://www.eclipse.org/acceleo/ (consultado: 09/09/2023)

Un **metamodelo** es el modelo de un lenguaje que recoge sus propiedades y características esenciales. Entre ellas están los conceptos del lenguaje soportados, su sintaxis textual y/o gráfica, y su semántica.

OMG define cuatro niveles de abstracción para el desarrollo de modelos, donde cada nivel es una instancia del nivel anterior (a excepción del nivel superior). En la Figura 2.4 se ilustran estos niveles y su relación.

Datos (M0): Es el nivel inferior y declara las entidades que maneja el sistema. Se refiere a los datos concretos o instancias de un modelo específico. Los datos siguen la estructura y las reglas definidas por el modelo de nivel superior (M1).

Modelo (M1): En este nivel se crean los modelos específicos de un dominio de aplicación. Este modelo sigue las reglas y estructuras definidas por el modelo de nivel superior, es decir, su metamodelo (M2).

Metamodelo (M2): En este nivel se definen los modelos que contienen la especificación de las clases y relaciones que se pueden utilizar para construir modelos en el nivel M1. UML[13] o SPEM[14] son ejemplos de metamodelo. Un metamodelo es conforme a un meta-metamodelo (M3).

Meta-metamodelo (M3): Es el nivel superior donde se define cómo deben estructurarse los metamodelos y qué elementos pueden contener. Por tanto, estas descripciones especifican cómo deben estructurarse los metamodelos, proporcionando reglas y conceptos para su creación. MOF[15] o ECORE[16] son ejemplos.

Por lo tanto, un metamodelo define un lenguaje de modelado y las instancias de un metamodelo son las representaciones concretas dentro de ese lenguaje de modelado.

[13]Unified Modeling Language (UML), 2.5.1: https://www.omg.org/spec/UML/2.5.1/ (consultado: 19/11/2023)
[14]Software & Systems Process Engineering Metamodel (SPEM), 2.0: https://www.omg.org/spec/SPEM/2.0/ (consultado: 19/11/2023)
[15]MetaObject Facility (MOF), 2.5.1: https://www.omg.org/spec/MOF/2.5.1/ (consultado: 19/11/2023)
[16]Eclipse Ecore Tools: https://projects.eclipse.org/projects/modeling.emft.ecoretools (consultado: 19/11/2023)

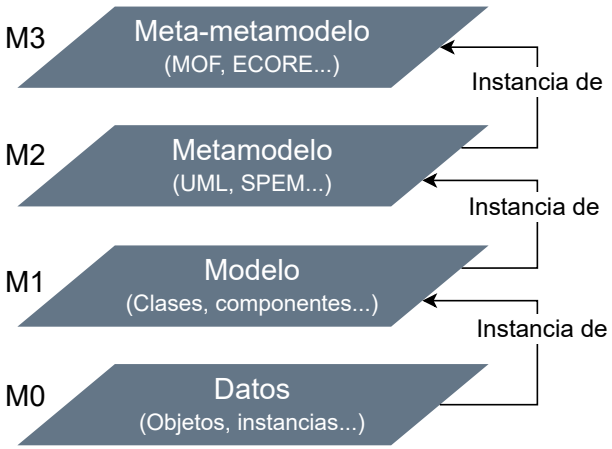


Figura 2.4: Niveles de abstracción para el metamodelado propuesto por OMG.

En este contexto, un lenguaje de modelado específico de dominio puede ser definido como:

> Un **lenguaje de modelado específico de dominio** o *domain-specific modelling language* (DSML) [109] es un lenguaje que está diseñado para ser utilizado en un determinado dominio de aplicación. Un DSML ofrece un conjunto de elementos y abstracciones que reflejan de manera precisa los conceptos y las reglas utilizadas en el dominio objetivo.

Los DSML formalizan la estructura, comportamiento y requisitos dentro de determinados dominios. La definición de un DSML implica al menos tres aspectos: los conceptos y reglas del dominio (sintaxis abstracta), la notación utilizada para representar estos conceptos (sintaxis concreta) y la semántica del lenguaje [23]:

- Sintaxis abstracta: Describe los elementos del lenguaje y cómo se relacionan entre ellos. Además, define el conjunto de reglas que restringen sus posibles combinaciones para respetar las restricciones del dominio. La sintaxis abstracta de un lenguaje de modelado se describe habitualmente mediante un metamodelo.
- Sintaxis concreta: Describe cómo los conceptos del lenguaje tienen que ser representados. En el caso de los lenguajes gráficos, se establecen los vínculos entre los elementos del metamodelo y los símbolos gráficos que los representan.

De la misma manera, es necesario establecer los vínculos entre los elementos del metamodelo y las estructuras sintácticas en el caso de los lenguajes textuales. En este caso, también es conocida como notación.

- Semántica: Describe el significado de los elementos del metamodelo asociándolos a conceptos del dominio del lenguaje y cómo se comportan cuando se ejecuta el modelo. Aunque la semántica de un DSML se da normalmente con lenguaje natural, también existen enfoques formales para definirla. Estos enfoques que utilizan formalismos matemáticos y lógicos para describir la semántica pueden incluir notaciones como lógica de primer orden, álgebra de procesos, o incluso teoría de categorías [24, 56].

2.3. Plataformas de desarrollo *low-code* y *no-code*

La necesidad de soluciones tecnológicas más accesibles y fáciles de usar ha dado lugar al desarrollo de plataformas de desarrollo *low-code* y *no-code*. Estas plataformas están diseñadas para que usuarios con diversos niveles de conocimiento en programación puedan crear aplicaciones sin necesidad de escribir código tradicional, lo que permite reducir significativamente sus costes y tiempos de desarrollo.

2.3.1. Origen y características

El desarrollo *low-code* es un enfoque de desarrollo de software que tiene como objetivo simplificar y acelerar el proceso de creación de aplicaciones. Este enfoque tiene su origen en diversas tendencias que se han ido desarrollando a lo largo de la historia, como el uso de lenguajes de cuarta generación y herramientas *computer-aided software engineering* (CASE) [97], el *rapid application development* [98], el *end-user development* [9] y el MDSE.

Sin embargo, fue en 2014 cuando se acuñó el término plataformas de desarrollo *low-code* [115] para referirse a aquellas plataformas que permiten la entrega rápida de aplicaciones con un mínimo de codificación manual y una inversión inicial mínima en términos de configuración e implementación. Con el tiempo, la definición evolucionó y en 2017 se amplió su definición para destacar el uso de interfaces de usuario visuales

y lenguajes declarativos, con especial énfasis en el desarrollo visual y el desarrollo basado en modelos con el fin de reducir el *time-to-market* y costes de desarrollo [123].

Según Robert Waszkowski [153] y Di Ruscio *et al.* [39], este tipo de plataformas de desarrollo pueden ser definidas como:

> Las **plataformas de desarrollo *low-code*** o *low-code development platform* (LCDP) son un conjunto de herramientas orientadas a programadores y a no programadores. Permiten generar y entregar rápidamente aplicaciones con un conocimiento mínimo en lenguajes de programación, y con el menor esfuerzo posible tanto para su instalación como para su configuración.

Por tanto, las LCDP surgen con el objetivo de favorecer el uso de entornos visuales e intuitivos donde usuarios no técnicos (conocidos como *citizen developers*) pueden diseñar, construir y desplegar aplicaciones sin la necesidad de poseer conocimientos en programación [124]. La idea fundamental es abstraer y automatizar las tareas repetitivas y de bajo nivel que consumen tiempo en el desarrollo tradicional de software, permitiendo que los equipos se centren en los aspectos esenciales de la lógica del dominio de la aplicación.

Las LCDP han generado un cambio significativo en la forma en que se construyen aplicaciones [115], ofreciendo un enfoque más ágil y accesible para el desarrollo de software. A medida que la tecnología continúa avanzando, estas plataformas son cada vez más usadas en la industria debido a los beneficios distintivos que aportan:

- Reducción de costes: Debido a la disminución de tiempo de desarrollo y a la posibilidad de que perfiles no expertos en desarrollo de software puedan crear aplicaciones, el coste se reduce.
- Rapidez: Dado que los usuarios solo tienen que reutilizar componentes y piezas de código para configurar las aplicaciones, en lugar de crearlas manualmente desde cero, el tiempo de desarrollo se reduce considerablemente y favorece la aparición de nuevas aplicaciones de manera más rápida.
- Reducción de la complejidad: Dado que las aplicaciones no se crean desde cero, el desarrollo de aplicaciones se simplifica. Este hecho permite centrarse más

en personalizar y adaptar el software para cumplir con los requisitos de los usuarios y sus dominios de aplicación.

- Mantenimiento sencillo: Estas plataformas permiten a los desarrolladores implementar cambios y actualizaciones en las aplicaciones de manera rápida y eficiente, sin necesidad de modificar grandes cantidades de código manualmente. Gracias a la necesidad de crear poco código, se reduce la cantidad de código para testear y mantener.

- Participación de perfiles no técnicos: Estas plataformas proporcionan interfaces de usuario simples e intuitivas que funcionan como entorno desde el desarrollo hasta el despliegue de las aplicaciones. En este contexto, no se requiere conocimiento técnico y los usuarios finales de estas aplicaciones se convierten en los desarrolladores de las mismas. Así que se aprovecha que ellos son los que tienen un profundo conocimiento sobre las necesidades del dominio.

- Validación temprana: El uso de LCDP implica la posibilidad de construir rápidamente productos mínimos viables para validar ideas y requisitos antes de invertir una gran cantidad de recursos en características y funcionalidades que podrían resultar innecesarias.

Desde el punto de vista de su arquitectura, las LCDP se componen de cuatro capas principales [39], tal y como se muestra en la Figura 2.5. La capa superior (Capa de Aplicación) está compuesta por el entorno gráfico con el que los usuarios interactúan directamente para crear y configurar sus aplicaciones. Esta capa proporciona una serie de elementos fundamentales en forma de "caja de herramientas" y *widgets* que se utilizan para definir la interfaz de usuario de la aplicación. Típicamente, las interfaces de usuario de las LCDP proporcionan funcionalidades como la de arrastrar y soltar elementos, la generación automática de artefactos o la definición de reglas para modelar la lógica de negocio.

Además, es posible definir y adaptar el comportamiento de la aplicación para un dominio específico. Para ello, es posible determinar cómo se obtienen los datos de fuentes externas (como bases de datos, ficheros en local o almacenados en servicios en la nube), cómo procesarlos mediante la reutilización de componentes ofrecidos por la propia LCDP o mediante el uso de servicios externos, así como definir cómo

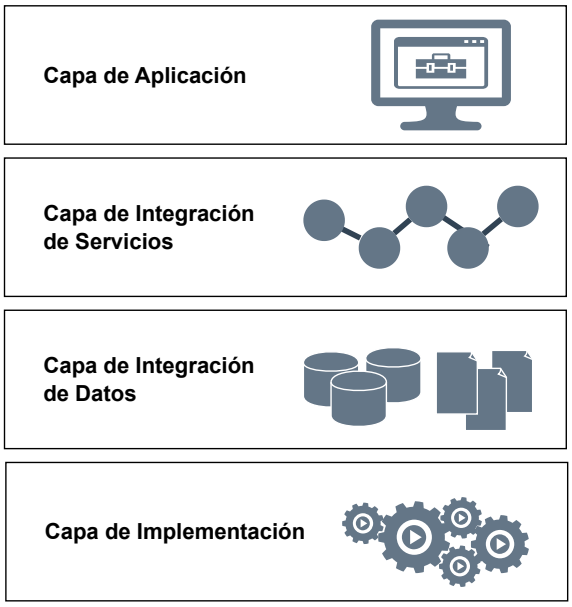


Figura 2.5: Separación en capas de una LCDP.

visualizarlos. Para esto, la siguiente capa (Capa de Integración de Servicios) permite a los usuarios reutilizar y adaptar funcionalidades ya existentes sin tener que desarrollarlas desde cero para operar y manipular los datos, incluso si estos proceden de fuentes de datos heterogéneas.

En este sentido, la tercera capa (Capa de Integración de Datos) se encarga de consolidar los datos provenientes de diversas fuentes de datos. Dependiendo de la LCDP utilizada, la aplicación desarrollada puede ser implementada en infraestructuras *cloud* dedicadas o en entornos locales (Capa de Implementación). El uso de contenedores y la orquestación de las aplicaciones se manejan en esta capa, junto con otras funcionalidades relacionadas con la integración y despliegue continuo.

Las LCDP comparten muchos de los principios de MDSE [18, 39]. Así, MDSE proporciona un *framework* de desarrollo especialmente útil para la creación de LCDP. El uso de modelos y metamodelos facilita la automatización, análisis y abstracción de herramientas que permiten definir y capturar de manera más eficiente el conocimiento del dominio donde operan. Además, favorece que el conocimiento del dominio sea modelado independientemente de las plataformas o tecnologías subyacentes donde acabarán operando. Gracias a esto, se permite reducir la complejidad accidental que introduce el uso de lenguajes de programación y el *time-to-market* de nuevas aplicaciones.

Ya se puede encontrar ejemplos de LCDP en numerosos dominios de aplicación [124]. Como Watson Assistant[17] de IBM para desarrollar bots conversacionales o Power Apps[18] de Microsoft para crear aplicaciones empresariales. También existen LCDP basadas en MDSE en dominios como los gemelos digitales [31] o el control de aerogeneradores [61].

Junto con el concepto *low-code*, el concepto *no-code* se empezó a utilizar de manera intercambiable para referirse a este tipo de plataformas que eliminan la necesidad de escribir código y que también agilizan el desarrollo de aplicaciones [117]:

> Las **plataformas de desarrollo *no-code*** o *no-code development platform* (NCDP) son entornos de desarrollo que permiten la creación de aplicaciones sin la necesidad de escribir código, utilizando interfaces visuales e intuitivas.

Algunos ejemplos de este tipo de herramientas son Shopify[19] para la creación de *e-commerce*. También existen NCDP especializadas en el desarrollo de aplicaciones web y móviles, como Bubble[20], Adalo[21] o AppGyver.[22]

2.3.2. Diferencias entre plataformas de desarrollo *low-code* y *no-code*

	Low-code	No-code
Tipo de usuario	Desarrolladores profesionales	*Citizen developers*
Código manual	Para adaptar y extender	No es necesario
Velocidad de desarrollo	Media	Alta
Flexibilidad	Alta	Baja

Tabla 2.4: Resumen de las principales diferencias entre *low-code* y *no-code*.

[17] IBM Watson Assistant. https://www.ibm.com/cloud/watson-assistant/ (consultado: 06/09/2023)
[18] Microsoft Power Apps. https://powerapps.microsoft.com/ (consultado: 06/09/2023)
[19] Shopify. https://www.shopify.com/ (consultado: 14/09/2023)
[20] Bubble. https://bubble.io/ (consultado: 14/09/2023)
[21] Adalo. https://adalo.com/ (consultado: 14/09/2023)
[22] AppGyver. https://www.appgyver.com/ (consultado: 14/09/2023)

Aunque las LCDP presentan muchas similitudes con las plataformas de desarrollo *no-code*, existen algunas diferencias que han de ser consideradas [117]. Estas diferencias son resumidas en la Tabla 2.4.

Así, las LCDP permiten la posibilidad de integrar líneas de código utilizando programación tradicional. De hecho, requieren un mínimo uso de codificación manual para crear y adaptar la aplicación, por lo que demanda ciertas habilidades de programación. Por este motivo, el *low-code* se orienta principalmente a desarrolladores que quieren reducir el tiempo de desarrollo y evitar la repetición de código. Por contra, la eliminación de la necesidad de escribir código hace que las NCDP se orienten a los denominados *citizen developers* [87], usuarios que poseen un gran conocimiento en sus dominios pero que, aunque puedan tener ciertas habilidades técnicas, carecen de la capacidad para escribir código manualmente. Esto hace que las NCDP sean ideales para la creación de aplicaciones mediante interfaces de usuario de arrastrar y soltar.

En términos de velocidad de desarrollo, las LCDP requieren más tiempo de desarrollo y puesta en marcha que las NCDP debido al mayor grado de personalización que ofrecen. Sin embargo, sigue siendo considerablemente más rápido que el desarrollo tradicional. Por otro lado, las NCDP sacrifican la capacidad de adaptación o extensión, reduciendo así el riesgo de introducir potenciales errores que normalmente se introducen mediante la codificación manual.

A pesar de esta falta de flexibilidad, las NCDP resultan ideales y más accesibles para un mayor número de usuarios ya que proporcionan funcionalidades específicas que no necesitan la creación de módulos o piezas de software desde cero.

2.3.3. Plataformas de desarrollo para dominios intensivos en datos

Las LCDP y NCDP facilitan el desarrollo de aplicaciones en muchos dominios como el desarrollo de aplicaciones web o móviles, o en *e-commerce*. En los dominios intensivos en datos, este tipo de plataformas de desarrollo resultan especialmente útiles en dominios intensivos en datos al permitir a los científicos de datos el rápido desarrollo de DIA reduciendo la necesidad de conocimientos en desarrollo de software, computación de alto rendimiento o redes de comunicación. En lugar de desarrollar

aplicaciones mediante el uso de algún lenguaje de programación, las interfaces visuales de arrastrar y soltar facilitan la definición de todas las tareas involucradas como la integración con fuentes de datos, la preparación y análisis de datos, así como la orquestación y conexión de servicios externos, o la visualización de datos.

Actualmente ya existen LCDP y NCDP para ciencia de datos especializadas en distintas tareas.

Deep Talk[23]**:** Permite configurar modelos de aprendizaje automático o *machine learning* (ML) supervisados y no supervisados sin necesidad de código. Por ejemplo, se utiliza para clasificar temas, agrupar conversaciones y analizar datos generales de texto, encuestas, correos electrónicos o chats.

Monkey Learn[24]**:** Está orientada a permitir realizar, sin necesidad de código, análisis en tiempo real de datos precedentes de redes sociales, correos electrónicos, documentos o reseñas en línea.

Obviously AI[25]**:** Tiene como objetivo gestionar todo el proceso de ciencia de datos, desde la recopilación de datos hasta el análisis de datos y ML con una mínima cantidad de código.

BigML[26]**:** Es una plataforma de ML que facilita la automatización de tareas de clasificación, regresión, predicción de series temporales, análisis de clústeres, detección de anomalías y descubrimiento de asociaciones.

DataRobot[27]**:** Es una plataforma que utiliza inteligencia artificial para permitir a los usuarios, incluso sin experiencia en ciencia de datos, construir modelos predictivos de manera eficiente utilizando técnicas avanzadas de ML.

Además, los SWfMS se posicionan como herramientas LCDP y NCDP para dominios intensivos en datos. Al igual que este tipo de plataformas de desarrollo, los SWfMS

[23]Deep Talk. https://www.deep-talk.ai (consultado: 09/10/2023).
[24]Monkey Learn. https://monkeylearn.com (consultado: 09/10/2023).
[25]Obviously AI. https://www.obviously.ai (consultado: 09/10/2023).
[26]BigML. https://bigml.com (consultado: 09/10/2023).
[27]DataRobot. https://www.datarobot.com (consultado: 09/10/2023).

como Nextflow [122], KNIME [150], RapidMiner [116] o Taverna [64] (analizados en la Sección 2.1.4), proporcionan los mecanismos necesarios para facilitar a los científicos y profesionales en ciencia de datos el diseño y ejecución de DIA sin la necesidad de conocimientos especializados en programación.

Capítulo 3

Lenguaje específico de dominio para el modelado de flujos de trabajo intensivos en datos

"El diseño de lenguajes de programación es como pasear por el parque. Bueno, por Parque Jurásico"
Larry Wall

3.1. Introducción

Las DIA se están volviendo cada vez más habituales en numerosos dominios que van desde, por ejemplo, el comercio electrónico, los mercados financieros, la manufacturación, el marketing, la educación o las ciencias sociales [146]. Además, el sector científico también muestra un interés particular en las DIA [36, 149] ya que muchas áreas de investigación científicas se han vuelto fuertemente impulsadas por los datos, como la bioinformática, la astronomía o la atención médica [20].

Independientemente del campo de aplicación, los procesos de gestión de datos y descubrimiento de conocimientos de cualquier DIA pueden ser definidas como DIW, permitiendo una definición a alto nivel de este tipo de procesos. Así, se mejora la comprensión de las tareas intensivas en datos por parte de profesionales que no están especializados en ciencia de datos, pero son expertos en sus respectivos campos [129].

En esencia, un DIW sirve como puente entre científicos de datos que son especialistas en el análisis de datos y extracción de conocimiento, usuarios no especializados en tecnologías de la información pero que son expertos en el dominio, y especialistas en computación e infraestructura [138].

Los GPL como Python o lenguajes estadísticos como R han sido utilizados tradicionalmente para desarrollar DIW en el ámbito de la ciencia de datos. Sin embargo, este tipo de lenguajes requieren conocimientos de computación avanzada que dificulta su adopción en un amplio rango de dominios donde sus usuarios no son necesariamente expertos en computación. Para solventar este problema, los SWfMS definen lenguajes específicos para la definición de DIW, tal y como se mencionó en la Sección 2.1.4. A menudo, el conocimiento modelado por un DIW está fuertemente acoplado a un SWfMS específico [158], lo que dificulta su interoperabilidad con otros lenguajes de DIW y motores de ejecución de flujos de trabajo. Esto limita la reutilización del conocimiento fuera de las plataformas para las que fueron definidos, obstaculizando así la colaboración entre profesionales de distintos dominios [26].

Para reducir estas limitaciones, SWEL (Scientific Workflow Execution Language) ha sido diseñado y desarrollado como un DSML para la especificación abstracta de *pipelines* intensivos en datos, ejecución y experimentación. SWEL es independiente de cualquier herramienta o plataforma, fomenta la colaboración entre profesionales al permitir la reutilización del conocimiento entre distintos dominios y facilita la interoperabilidad entre herramientas. SWEL cubre toda la especificación de un DIW, desde la definición de alto nivel del problema en términos de la secuencia de tareas intensivas en datos a realizar, las fuentes de información involucradas y los requisitos de la plataforma donde se va a ejecutar.

El enfoque metodológico utilizado en este capítulo es el de DSR. SWEL es el principal artefacto desarrollado tras un proceso iterativo e incremental de identificación y extracción de requisitos para resolver el problema planteado anteriormente, el cual se desarrolla de acuerdo con los preceptos de la MDSE (véase la Sección 2.2). Los conceptos y componentes se modelan formalmente mediante *metamodelos* que describen los elementos del lenguaje, las relaciones entre ellos, y sus restricciones y reglas de comportamiento. Estos metamodelos no están vinculados a ninguna sintaxis o plataforma tecnológica concreta, pero se pueden definir e implementar fácilmente puentes

hacia ellas mediante el uso de transformaciones de modelos y de herramientas MDSE existentes.

3.2. Metodología

El objetivo principal que se persigue el modelado de un lenguaje de especificación de DIW (véase el Capítulo 1), que es el artefacto primario según el paradigma metodológico DSR. Aquí, un artefacto es un objeto creado con el fin de resolver un problema práctico y real, que puede variar desde una definición, un modelo, un método o una instanciación en forma de un sistema completo. DSR permite realizar un proceso de investigación utilizando el diseño como un método de investigación en sí mismo [67]. En este contexto, SWEL es el artefacto que se va a diseñar y modelar y, por lo tanto, DSR permite asegurarse de que se está construyendo correctamente [42]. Una propiedad clave de DSR es que permite construir una solución a partir de problemas específicos, por ejemplo, analizando de manera iterativa los lenguajes específicos de dominio o las estrategias de representación específicas.

La primera actividad de DSR implica *explicar el problema* mediante el análisis del estado del arte, lo que permite motivar y justificar el objetivo principal: crear un lenguaje de modelado abstracto para la especificación a alto nivel de DIW. Este lenguaje tiene que ser independiente de cualquier herramienta o plataforma, lo que permitiría la colaboración entre profesionales mediante la reutilización de fragmentos de conocimiento [138], incluso de distintos dominios, y la interoperabilidad entre herramientas. Posteriormente, una segunda actividad de DSR consiste en esbozar una solución a estos problemas al *definir los requisitos* para el artefacto.

La tercera actividad consiste en el *diseño y desarrollo del artefacto*. En este caso se diseña y desarrolla SWEL como el artefacto primario. A continuación, se realiza la *demostración del artefacto desarrollado*, es decir, la actividad en la que SWEL se utiliza en un estudio de caso para demostrar cómo puede resolver una instancia del problema real.

La última actividad es *evaluar el artefacto*, esto es, validar en qué medida este resuelve el problema y satisface los requisitos. Considerando las características del artefacto de SWEL, se ha desarrollado un modelo de evaluación que se fundamen-

ta en un enfoque de evaluación continua. Este enfoque se ha aplicado de manera consistente a lo largo del proceso de refinamiento iterativo, utilizando el *feedback* solicitado a profesionales y expertos en el área. Durante la última iteración, se lleva a cabo una evaluación basada en métricas para verificar la idoneidad y adaptabilidad de SWEL, utilizando el marco de evaluación diseñado por Guizzardi *et al.* [63]. Para validar las conclusiones de los especialistas involucrados en la evaluación de SWEL, se ha realizado una evaluación a través de encuestas con expertos externos.

3.3. Explicación del problema

Las herramientas actuales permiten diseñar e implementar soluciones que no son suficientemente generalizables y reutilizables. Esto se debe, entre otros motivos, a la rigidez de los lenguajes que utilizan para el desarrollo de aplicaciones basadas en DIW, su falta de interoperabilidad, el acoplamiento del conocimiento generado, y la baja tolerancia al cambio en un contexto altamente cambiante como es la ciencia de datos.

3.3.1. Análisis del estado del arte

Lenguajes de programación de propósito general

Las DIA han sido tradicionalmente implementadas mediante GPL como Cobol, C, Fortran, Perl, C++, Java o Python. Sin embargo, la ausencia de funcionalidades específicas para la computación intensiva en datos [68] ha provocado que hayan surgido lenguajes de programación científica y de alto rendimiento. R [79] es un lenguaje para cálculos y gráficos estadísticos que proporciona una amplia variedad de técnicas estadísticas (modelos lineales y no lineales, pruebas estadísticas clásicas, análisis de series temporales, clasificación o agrupación) y herramientas gráficas, además de ser altamente extensible. Julia [12] es un lenguaje de programación de alto nivel diseñado para satisfacer los requisitos de cálculo numérico y científico de alto rendimiento, aprovechando las capacidades de concurrencia y paralelización de plataformas *multicore*. Swift [160] ofrece un enfoque de programación funcional y

se centra en escribir programas que se ejecutan en recursos de cómputo distribuido, como procesadores *multicore*, *clusters*, *cloud*, *grids* y supercomputadores.

Para hacer uso de estos lenguajes de programación, se requiere poseer un amplio conocimiento de numerosos conceptos de computación avanzada, como redes, computación de alto rendimiento y programación de *pipeline* de datos. Por esta razón, no son ampliamente utilizados en dominios cuyos usuarios no son expertos en informática.

Lenguajes de flujos de trabajo intensivos en datos

Los SWfMS intentan mitigar la brecha existente entre los GPL y los expertos del dominio al proporcionar elementos funcionales, generalmente a través de interfaces gráficas de usuario, que ocultan la dificultad de operar, optimizar y gestionar los recursos computacionales necesarios en este tipo de computación (véase la Sección 2.1.4). Tradicionalmente, los SWfMS han desarrollado y utilizan sus propios lenguajes de definición de DIW, como Triana[1] o LONI Pipeline[2], orientados a dominios científicos, y KNIME[3] para definir aplicaciones de minería de datos. Sin embargo, estos lenguajes están estrechamente acoplados a sus respectivos SWfMS y sus especificaciones no son públicamente accesibles.

En un intento de hacer públicas sus especificaciones, otros SWfMS han definido sus lenguajes de DIW mediante esquemas XML. SCUFL (*Simple Conceptual Unified Flow Language*) [104], que fue propuesto por Taverna[4], MoML (*Modelling Markup Language*) diseñado específicamente para Kepler[5] y DAX (*Directed Acyclic Graph in XML*) desarrollado para Pegasus[6], son ejemplos representativos. Si bien sus especificaciones están disponibles públicamente en forma de esquemas XML, estos lenguajes aún están destinados a sacar provecho de las características particulares proporcionadas por sus SWfMS respectivos. Algunos otros lenguajes de DIW específicos de la

[1]Triana: http://github.com/CSCSI/Triana (última actualización: 2014; consultado: 19/10/2023)
[2]LONI Pipeline: https://pipeline.loni.usc.edu (última actualización: 2020; consultado: 19/10/2023)
[3]KNIME. https://www.knime.com/ (última actualización: 2023; consultado: 19/10/2023)
[4]Taverna: https://incubator.apache.org/projects/taverna.html (última actualización: 2020; consultado: 19/10/2023)
[5]Kepler 2.5: https://kepler-project.org (última actualización: 2015; consultado: 19/10/2023)
[6]Pegasus 5.0: https://pegasus.isi.edu (última actualización: 2020; consultado: 19/10/2023)

plataforma fueron concebidos para ejecutarse en infraestructuras específicas. Este es el caso de GridAnt [6], que se propuso para describir *pipelines* para la computación en *grid*. Otro ejemplo es el motor de ejecución de DIW GridBus [159], que se basa en xWFL, un lenguaje basado en XML para la representación de requisitos de calidad de servicio.

Dos limitaciones clave de estos lenguajes de DIW específicos de la plataforma son: (1) su limitada interoperabilidad con otros lenguajes de flujo de trabajo y motores de ejecución, y (2) su limitada reutilización fuera de las plataformas para las que fueron definidos [32, 34]. Para abordar estos problemas, se definieron varios lenguajes de DIW de manera más abstracta e independiente de la herramienta. AGWL, IWIR y CWL son los ejemplos más conocidos de estos tipos de lenguajes.

AGWL (*Abstract Grid Workflow Language*) [49] fue el primer intento de crear un lenguaje de DIW independiente de la plataforma, aunque originalmente se desarrolló en el contexto de la herramienta ASKALON [50]. Más tarde, IWIR (*Interoperable Workflow Intermediate Representation*) [110] se diseñó para facilitar la portabilidad y la interoperabilidad entre lenguajes específicos de DIW, separando la lógica del flujo de trabajo de los datos y de procesamiento. Más recientemente, CWL (*Common Workflow Language*) [8] es un lenguaje que puede ejecutarse en diferentes entornos de software y hardware. CWL se basa en JSON-LD (*JSON for Linked Data*)[7]. Sin embargo, estos enfoques aún dependen de la tecnología, ya que dependen de estructuras predefinidas y analizables que se ajustan a una sintaxis concreta basada en JSON o XML. Esta dependencia dificulta la captura de la semántica del dominio, acoplando el conocimiento del experto a estas tecnologías y dificultando su cambio según las tendencias o las especificaciones de dichas tecnologías [14]. La tabla 3.1 muestra un resumen de las características y limitaciones de cada lenguaje específico de DIW.

Las notaciones de modelado de procesos empresariales de propósito general, como SPEM[8], BPMN, CMMN[9], o los diagramas de actividad de UML[10], podrían parecer

[7]JSON-LD - JSON for Linked Data: https://json-ld.org/ (consultado: 05/10/2023)

[8]Software & Systems Process Engineering Metamodel (SPEM), 2.0: https://www.omg.org/spec/SPEM/ (consultado: 19/09/2023)

[9]Case Management Model and Notation (CMMN), 1.1: https://www.omg.org/spec/CMMN/ (consultado: 19/09/2023)

[10]Unified Modeling Language (UML), 2.5.1: https://www.omg.org/spec/UML/2.5.1/ (consultado: 19/09/2023)

Lenguaje	Acoplado a herramienta	Especificación abierta	Ejecución sobre plataforma específica	Dependiente de notación
Triana	Sí	No	No	Sí
LONI Pipeline	Sí	No	No	Sí
KNIME	Sí	No	No	Sí
SCUFL	Sí	Sí	No	Sí
MoML	Sí	Sí	No	Sí
DAX	Sí	Sí	No	Sí
GridAnt	No	No	Sí	Sí
GridBus	No	No	Sí	Sí
xWFL	No	No	Sí	Sí
AGWL	No	Sí	No	Sí
IWIR	No	Sí	No	Sí
CWL	No	Sí	No	Sí
SWEL	No	Sí	No	No

Tabla 3.1: Resumen de características y limitaciones de lenguajes de DIW.

ser candidatos adecuados para especificar DIW a priori. Tienen la ventaja de ser notaciones estándar respaldadas por numerosos editores y otras herramientas genéricas de modelado como MetaEdit+ [75] o JetBrains MPS [15]. Sin embargo, requerirían un esfuerzo considerable para desarrollar extensiones que describan los requisitos y conceptos específicos de los tipos de procesos definidos por los DIW y las tareas utilizadas en el dominio de interés. Además, se necesitaría un gran esfuerzo para cubrir otros aspectos clave, como los requisitos de las plataformas computacionales o los motores de ejecución basados en datos que no son soportados por herramientas de procesos empresariales [93]. Por este motivo, SWEL adopta un enfoque diferente al definir metamodelos genéricos, independientes de la sintaxis y que podrían utilizarse tanto a nivel de modelado para implementar notaciones concretas de DIW como para intercambiar información entre las herramientas ya existentes. A nivel de ejecución, SWEL puede usarse para especificar los requisitos de las plataformas computacionales y los motores de ejecución de DIW, así como para alcanzar la interoperabilidad entre los existentes mediante transformaciones de modelos.

3.3.2. Motivación y objetivos

La reutilización del conocimiento representado por los DIW entre distintas herramientas supone todo un reto para los expertos de diversos dominios [55, 138]. Esta

problemática es experimentada de primera mano por muchas compañías y organizaciones que acaban utilizando diferentes SWfMS, dependiendo del problema en cuestión y de la familiaridad de cada profesional con respecto a una herramienta concreta. Sin embargo, precisamente cada DIW solo puede ejecutarse en la herramienta sobre la cual fue creado, y reutilizar fragmentos de DIW actualmente requiere duplicar el trabajo y reescribir manualmente todos los DIW para la nueva herramienta, lo que no suele ser viable en términos de tiempo y esfuerzo.

Además, concentrar el conocimiento en una sola tecnología y requerir que todos los profesionales involucrados aprendan una nueva herramienta no es deseable en la mayoría de situaciones. Se necesita una solución que permita la reutilización y adaptación del conocimiento capturado por los DIW existentes, independientemente de la herramienta que lo generó. No obstante, la idea de tener un DSML que permita esta definición es solo una parte de la solución al problema. La otra parte implica la reutilización del conocimiento de DIW en diferentes herramientas.

Identificación de elementos fundamentales

El DSML a desarrollar debe facilitar la creación de DIW de alto nivel, independientemente de la plataforma, con el fin de compartir el conocimiento. Con este objetivo, se identifican los elementos básicos comunes a la mayoría de los DIW, es decir, los elementos fundamentales que componen SWEL.

Inicialmente se llevan a cabo reuniones con diferentes profesionales. Estas reuniones permiten identificar las herramientas más comúnmente utilizadas para el diseño de DIW, las cuales se complementan con otras herramientas similares encontradas en la revisión de la literatura. Durante esta búsqueda, se descubre que algunas de estas herramientas son propietarias, y que la información sobre los detalles de implementación de sus DIW no está disponible públicamente. Estas soluciones no se consideran en el proceso, aunque se consultan sus manuales de referencia, en los casos en los que estos están disponibles. A continuación, se realiza una revisión de la literatura sobre los lenguajes de DIW utilizados por estas herramientas. Finalmente, se consultan una serie de estudios existentes para conocer en detalle más sobre las características comunes, los modelos de ejecución frecuentemente utilizados y las tareas típicas necesarias para crear un DIW. Como resultado, se identifican

los elementos principales de SWEL, que permitirán la definición de una amplia gama de DIW. Estos elementos se refieren tanto a la estructura del DIW como a su especificación.

Estructura del flujo de trabajo. Un flujo de trabajo se compone de tareas interconectadas que definen sus dependencias. Su estructura determina cómo se representan y ejecutan estas tareas y sus relaciones. En general, hay dos tipos de representaciones: aquellas basadas en grafos acíclicos dirigidos o *directed acyclic graph* (DAG), y aquellas basadas en grafos cíclicos dirigidos o *directed cyclic graph* (DCG). Ambas han sido tradicionalmente utilizadas para definir DIW, aunque las representaciones basadas en DAG son comúnmente utilizadas en dominios orientados a datos [158]. Sin embargo, los DCG podrían ser más apropiados para algunos dominios dirigidos por lógica de negocio. Por lo tanto, SWEL debe admitir ambos tipos para representar la mayor variedad posible de DIW y facilitar la reutilización en un mayor número de herramientas.

Especificación del flujo de trabajo. La definición de distintos tipos de tareas depende del tipo de lenguaje de DIW utilizado. Los lenguajes abstractos describen las tareas a un alto nivel de abstracción sin hacer referencia a los requisitos de plataformas específicas, recursos computacionales o lenguajes de programación. Estos flujos de trabajo no son directamente ejecutables, sino que deben pasar por una etapa de transofmración que los asocia a recursos computacionales específicos. La responsabilidad de esta relación suele recaer en el motor de ejecución o en la propia herramienta. Por contra, los lenguajes concretos contienen tareas específicas que tienen en cuenta detalles de implementación a bajo nivel, lo que los hace directamente ejecutables. Actualmente existe un mayor número de lenguajes concretos que de lenguajes abstractos, ya que la mayoría de los lenguajes se han diseñado durante el desarrollo de un SWfMS específico, como SCUFL de Taverna o MoML de Kepler. Sin embargo, también existe alguna propuesta de lenguaje abstracto independiente de cualquier plataforma o herramienta concreta, como es el caso de CWL.

SWEL debería admitir tanto tareas a un nivel alto de abstracción como tareas más específicas asociadas a aspectos de bajo nivel, como el lenguaje de programación, la plataforma de ejecución (*cloud* o *grid*, entre otros), así como los diferentes modelos de ejecución (secuencial o paralelo). Estos tipos de servicios de bajo nivel implican

la invocación a servicios web o la ejecución de programas escritos en un lenguaje de programación específico.

Identificación de tareas de alto nivel

El siguiente paso es identificar las tareas de alto nivel involucradas en un DIW. Para lograrlo, se realiza una búsqueda y revisión de varios lenguajes de DIW públicamente disponibles. Estos lenguajes sirven para identificar las principales tareas computacionales y de visualización de datos, así como las estructuras de control y los proveedores de entrada y salida de datos comúnmente utilizados. No todos los lenguajes tienen una especificación basada en modelos que proporcione una información precisa de todos los elementos. Por ello, se debe realizar un proceso de ingeniería inversa analizando su serialización textual, habitualmente basada en XML y JSON. Así, se crean varios DIW en diferentes herramientas y se analizan un gran número de DIW procedentes del repositorio público de DIW, *myExperiment.org*.

Durante la fase de identificación de tareas de alto nivel, también se valida que los elementos fundamentales identificados en la fase anterior son aplicables a los DIW analizados. Además, al descubrir nuevos elementos que no se habían identificado previamente, estos se incorporan en SWEL también como elementos principales.

Identificación de puntos de extensión

Se observa una amplia variedad de tareas dentro de los diferentes tipos de DIW que se analizan y definen. En muchos casos, las tareas dependen directamente de las características de las herramientas para las cuales se diseña el DIW. Un ejemplo típico son las tareas que invocan programas escritos en Java o Python y acceden a datos almacenados en bases de datos externas como MySQL o MongoDB. No es factible cubrir tal diversidad de tareas en un único lenguaje de DIW. Sin embargo, al permitir extender elementos más genéricos, se puede dar soporte a la inclusión de elementos más específicos o a aquellos que dependen de la plataforma en la que se ejecutan. Estos elementos genéricos se identifican como puntos de extensión y permiten modelar conocimiento más específico, así como para facilitar su reutilización y utilizar términos cercanos al dominio del experto.

Identificación de elementos que describen experimentos científicos

Gracias a las entrevistas con distintos profesionales, se detecta la importancia de incluir la información sobre el proyecto o experimento científico que servía de motivación para crear el DIW en particular. Esta información proporciona un contexto más amplio sobre el diseño y los objetivos de un DIW dado, facilitando tanto la comprensión de los resultados tras su ejecución, así como la transferencia de conocimiento entre profesionales.

Esta necesidad también se detecta al trabajar con DIW procedentes de repositorios como *myExperiment.org*. La mayoría de ellos incorporan esta información para proporcionar más contexto. Sin embargo, es importante destacar que esta información disponible en este tipo de repositorios públicos no está directamente vinculada al DIW, ni incluida en su definición, sino que simplemente es metainformación externa que ayuda a comprender el comportamiento que los autores esperaban conseguir al definir ese DIW. Con este fin, se investigan posibles lenguajes existentes para definir experimentos científicos intensivos en datos [106], con el objetivo de adaptar estas representaciones y que sean compatibles en mayor o menor medida con SWEL.

3.4. Definición de los requisitos

Para cumplir los objetivos definidos en la Sección 3.3.2, se definen a continuación los siguientes requisitos para el artefacto primario:

- **REQ1:** *Proporcionar un lenguaje abstracto de DIW adecuado para definir DIA a un alto nivel de abstracción.*

  El artefacto debe dar soporte a las principales características proporcionadas por los actuales SWfMS, como los modelos de ejecución cíclicos y acíclicos, los diferentes modelos de ejecución (secuencial, concurrente o iterativo), los distintos métodos de acceso a datos (en memoria, base de datos o almacenamiento externo) [51], los patrones de composición de datos [103], las numerosas estructuras de flujo de control [30], la composición de procesos y las tareas de preparación, procesamiento y visualización de datos [54].

- **REQ2:** *Dar soporte a la reutilización de fragmentos de conocimiento entre diferentes herramientas al proporcionar un lenguaje de DIW independiente de la plataforma.*

  Un lenguaje de DIW independiente de la plataforma facilita la colaboración en múltiples dominios [138] y desacopla la definición del DIW del procedimiento concreto de una herramienta en particular.

- **REQ3:** *Proporcionar un lenguaje de DIW independiente de cualquier notación en términos de su sintaxis concreta.*

  Este requisito de independencia tiene como objetivo permitir la mejor representación de la semántica del dominio [15]. Por lo tanto, los DIW definidos por el artefacto deben poder representarse en cualquier notación, tanto textual como gráfica, según los requisitos del dominio de aplicación.

3.5. Diseño de SWEL

Un DIW contiene un conjunto amplio y diverso de elementos de información, que van desde la especificación de la ejecución computacional hasta los conceptos del dominio, o el proyecto que impulsó su creación. La estructura del DIW determina la secuencia de actividades y las dependencias entre ellas, ya sea como un DCG o como un DAG. Este tipo de representación permite definir su ejecución de manera eficiente aplicando mecanismos como la paralelización o la composición de datos. Además, también oculta la complejidad inherente a la programación paralela o concurrente al proporcionar elementos de alto nivel para sacar provecho de dichas características.

3.5.1. Estructura general

La estructura del DIW se define de acuerdo a los conceptos del dominio y el conocimiento que los profesionales consideran relevantes para cumplir con los requisitos de un experimento científico intensivo en datos y con los de un proceso computacional intensivo en datos [30, 54, 103]. Así, la organización general de SWEL se distribuye en capas, tal y como se explica a continuación:

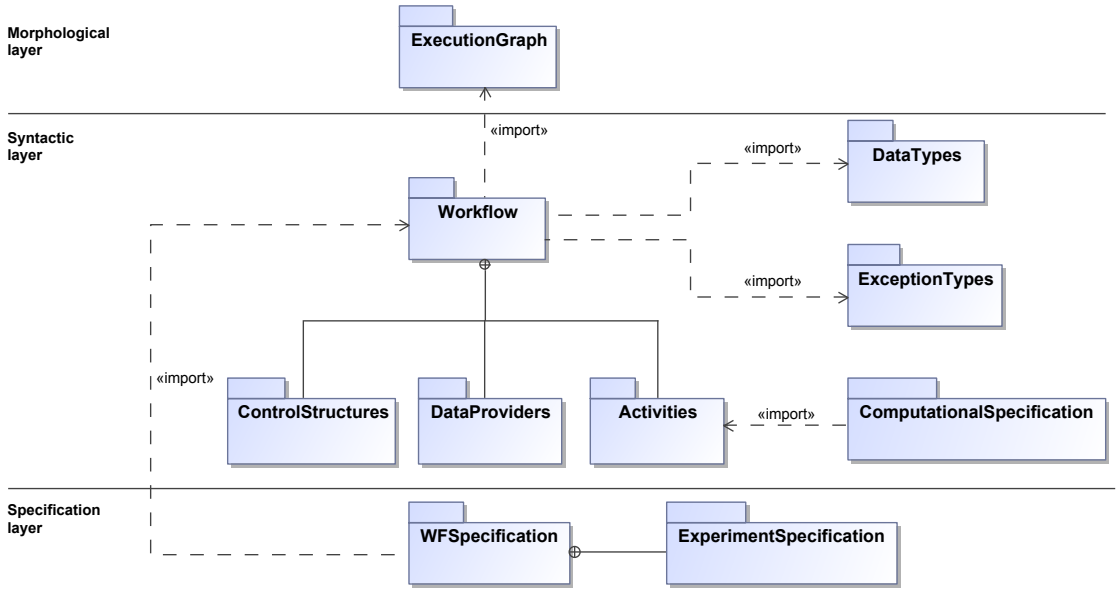


Figura 3.1: Capas del metamodelo de SWEL y dependencias entre paquetes de elementos.

- La *capa morfológica* contiene aquellos elementos que permiten la definición de bajo nivel de un DIW. Este se define como un DCG o un DAG, donde sus vértices representan elementos (como proveedores de datos o tareas computacionales) y sus arcos representan las distintas dependencias entre ellos.
- La *capa sintáctica* consiste en un conjunto de paquetes de elementos que permiten la representación de los requisitos específicos del dominio y de los recursos necesarios para el DIW. En este nivel, los elementos permiten la declaración de las estructuras de control, los tipos de datos, la gestión de errores, las tareas específicas del dominio y los recursos computacionales involucrados en su ejecución.
- La *capa de especificación* describe diferentes elementos de información contextual sobre el proyecto o experimento relacionado con la definición del propio DIW.

La especificación del metamodelo de SWEL puede consultarse en el Apéndice A. Además, el *technical report* completo de SWEL se encuentra disponible en el material suplementario [134].

En la Figura 3.1 se representa la estructura general del lenguaje como un diagrama de paquetes UML, donde cada uno de ellos contiene una parte específica (agrupación de elementos) del metamodelo de SWEL. En las siguientes secciones se describe el contenido de los distintos paquetes, especificados como diagramas de clases UML compuestos por metaclases y sus relaciones. Una *metaclase* define un elemento del lenguaje y puede tener atributos. Cabe reseñar que las metaclasses suelen ser concretas, pero también pueden ser abstractas (su nombre aparecerá en cursiva en el diagrama), lo que significa que no se pueden instanciar directamente en ningún elemento del lenguaje. Además, también se han definido *puntos de extensión* para indicar aquellos elementos del metamodelo que se espera que se extiendan en el futuro, lo que permitirá la escalabilidad de SWEL y su adaptación a diferentes contextos organizativos y tecnológicos.

3.5.2. Capa morfológica

La capa morfológica contiene los elementos que representan la estructura interna de un DIW, y consta de un único paquete, *ExecutionGraph*, representado en la Figura 3.2. En este nivel, SWEL define un DIW como un tipo particular de grafo, específicamente un grafo de ejecución (metaclase *ExecutionGraph*), que tiene como objetivo definir el conjunto de pasos necesarios para llevar a cabo un proceso computacional.

Un grafo de ejecución está compuesto por operaciones computacionales (metaclase *Node*) y las dependencias entre ellas, representadas como arcos dirigidos (metaclase *DirectedEdge*). Estas operaciones, o nodos, están identificadas de manera única por una etiqueta y proporcionan un conjunto de puntos de conexión (metaclase *Endpoint*) para determinar el tipo de dependencia que se establecerá con los otros nodos. Por un lado, una dependencia de datos requiere recibir o enviar datos desde/hacia otro nodo. Por tanto, se asocian puntos de conexión de datos (*DataEndpoint*) al tipo correspondiente de conexión, es decir, una conexión de datos (*DataLine*). Por otro lado, existe una dependencia de control cuando un nodo solo puede ser ejecutado después de que otro nodo haya sido ejecutado. Este tipo de dependencia está representado por puntos de conexión de control (*ControlEndpoint*), y los nodos están conectados a través de conexiones de control (*ControlLine*). Además, la aparición

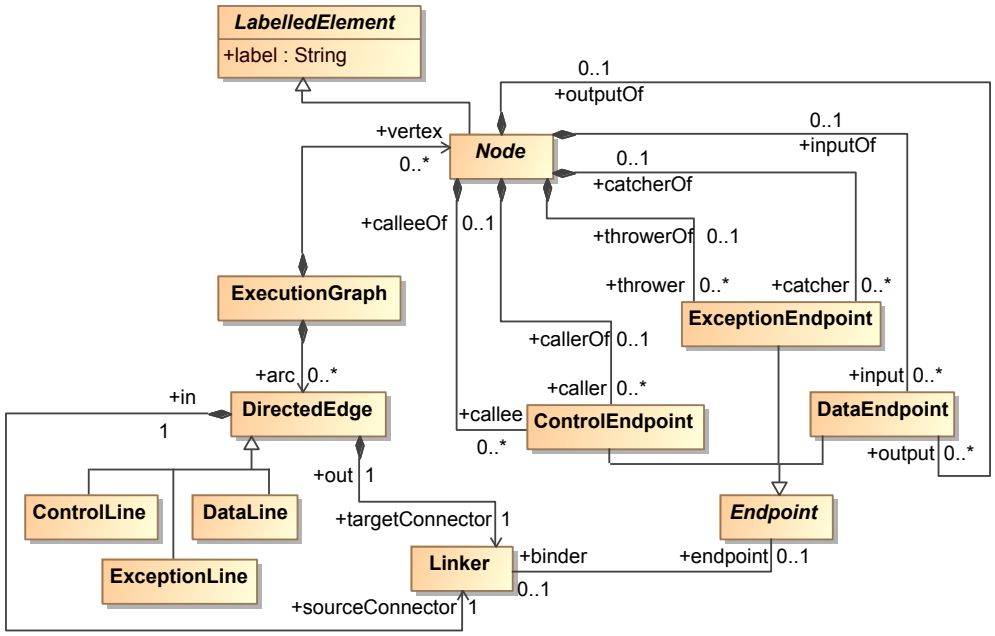


Figura 3.2: Elementos del metamodelo en la capa morfológica: paquete *ExecutionGraph*.

de problemas inesperados durante la ejecución del flujo de trabajo podría desencadenar una ruta de ejecución alternativa con un orden diferente entre las operaciones definidas. Aquí, los puntos de conexión de excepción (*ExceptionEndpoint*) están conectados a través de una conexión de excepción (*ExceptionLine*). La definición de qué punto de conexión específico se asocia a un enlace específico es realizada por un elemento específico (*Linker*).

3.5.3. Capa sintáctica

La capa sintáctica proporciona los elementos que capturan el conocimiento extraído de los expertos en el dominio. Como se mostraba en la Figura 3.1, esta capa está compuesta por siete paquetes. La declaración de los tipos de datos y las excepciones se representa en los paquetes *DataTypes* y *ExceptionTypes*. A partir de la definición de la estructura del grafo en la capa morfológica, el paquete *Workflow* proporciona un conjunto de elementos adaptables para definir *pipelines*. Estos consisten en tareas computacionales (*Activity*), proveedores de datos (*DataProvider*) y estructuras de control (*ControlStructure*), así como dependencias entre ellos. Finalmente, el

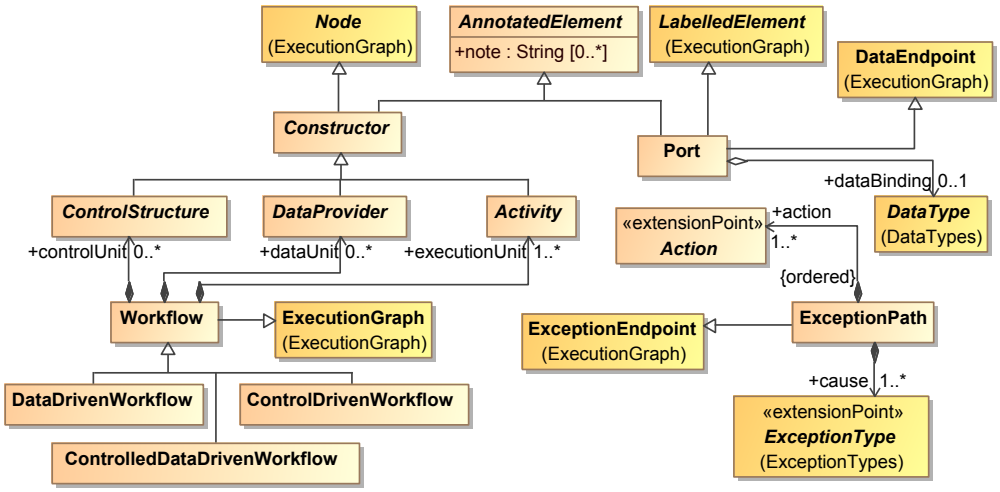


Figura 3.3: Elementos del metamodelo en la capa Sintáctica: paquete *Workflow*.

paquete *ComputationalSpecification* contiene los elementos que definen los recursos computacionales específicos que deben utilizarse en escenarios particulares, como en redes con servidores dedicados o plataformas basadas en la nube. Cabe destacar que esta organización no afecta al lenguaje, pero proporciona una forma legible de diferenciar aspectos relacionados entre sí, siguiendo el principio de diseño de separación de conceptos o *separation of concerns*. Por brevedad, nos centraremos solo en aquellos elementos que son más relevantes para la comprensión de SWEL.

En cuanto a los paquetes *DataTypes* y *ExceptionTypes*, se definen en SWEL una serie de tipos de datos (*DataType*) para clasificar los diferentes flujos de datos que atraviesan los *pipelines*. Los tipos de datos *básicos* están relacionados con los tipos primitivos de cualquier lenguaje de programación, como enteros, flotantes o cadenas. Los tipos de datos *complejos* definen el formato de un archivo específico, como una imagen, audio o vídeo. En cuanto a los elementos de excepción (*ExceptionType*), definen flujos alternativos de ejecución (*ExceptionPath*) cuando ocurre un error, por parámetros incorrectos, disco lleno o denegación de permisos. Hay que tener en cuenta que un camino de excepción es un punto de conexión de excepción, por lo que los elementos de excepción están vinculados a nodos concretos del DIW. Un error puede tener asociada una lista de acciones (*Action*) a realizar. Esto podría incluir la ejecución de una tarea específica si la actividad actual falla, repetir su ejecución varias veces o detener la ejecución del *pipeline*. Tanto los elementos de excepción como las acciones pueden ampliarse para cubrir una amplia gama de valores según

las necesidades particulares de cada caso de uso. Por ejemplo, cuando un servicio no está disponible temporalmente (*UnreachableCloudService*) en un escenario basado en la nube que implica notificar al proveedor antes de finalizar la ejecución del flujo de trabajo (*NotifyAndFinish*).

Tal y como se muestra en la Figura 3.3, el paquete *Workflow* declara un DIW (*Workflow*) como un conjunto de activos computacionales (*Constructor*) y flujos de datos. Los flujos de datos se centran en el manejo de datos cuya disponibilidad determina implícitamente la secuencia de ejecución (*DataDrivenWorkflow*). En ciertos casos, el flujo de ejecución debe definirse explícitamente, independientemente de la disponibilidad de datos (*ControlDrivenWorkflow*). Ambos enfoques también pueden combinarse (*ControlledDataDrivenWorkflow*). Tener tipos específicos permite la validación del flujo de trabajo para reducir errores humanos, como agregar una estructura de control a un flujo de trabajo dirigido por los datos. Sin embargo, dado que la metaclase *Workflow* no es abstracta, se puede instanciar directamente, dejando la verificación de su validez a la herramienta de modelado que implemente SWEL como lenguaje de DIW. Independientemente del tipo de flujo de trabajo, los flujos de datos se transmiten a través de puntos de conexión de datos (*Port*) que definen el tipo de datos aceptados por la operación a ejecutar.

En aquellos casos en los que se requiere un control explícito de la ejecución del *pipeline*, SWEL define algunas estructuras de control similares a las definidas por BPMN o UML. El paquete *ControlStructures* proporciona diferentes estructuras (*ControlStructure*) que permiten la definición tanto del punto de inicio (*Begin*) como del punto final (*End*) del DIW. Dependiendo de los requisitos del dominio, internamente el flujo de ejecución también se puede dividir en flujos paralelos y concurrentes (*Fork*), que permiten que algunas tareas computacionales se ejecuten simultáneamente. Además, diferentes flujos de ejecución pueden unificarse tanto en un flujo que se ejecuta cada vez que se completa uno de los flujos (*Merge*), o en un solo flujo que se ejecuta únicamente cuando todos o parte de los flujos han terminado (*Synchronizer*). Este paquete también proporciona estructuras condicionales (*Conditional*) que permiten la selección del próximo proceso a ejecutar en base al cumplimiento de una determinada condición. Estas condiciones se descomponen en uno o dos operandos (*Operand*) y un operador (*Operator*) que determina el tipo de

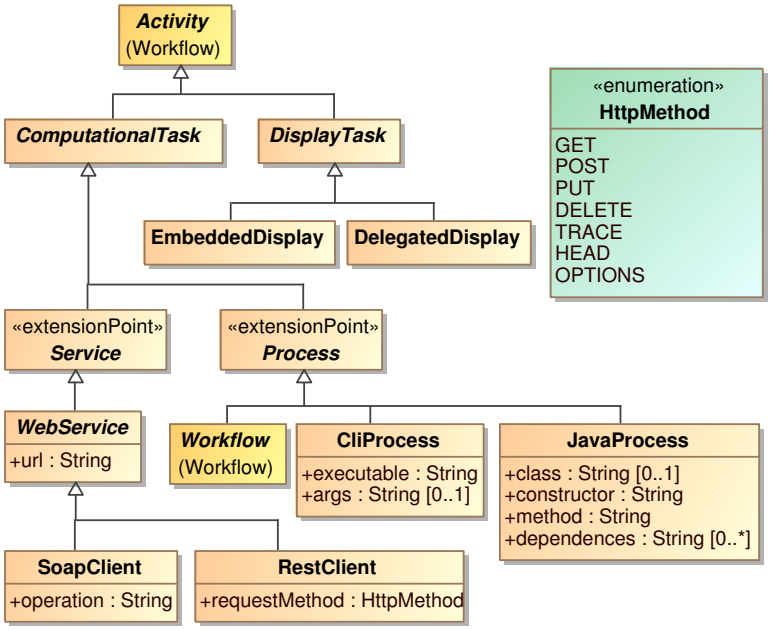


Figura 3.4: Elementos del metamodelo en la capa sintáctica: paquete *Activities*.

condición a evaluar, que puede ser una condición lógica (*LogicalOperator*), relacional (*RelationalOperator*) o matemática (*MathematicalOperator*).

La definición de *pipelines* dirigidos por datos es la práctica más habitual a la hora de definir un DIW. Así, tal y como se formaliza en el paquete *DataProviders*, los datos pueden originarse en diferentes fuentes, a las que el DIW accede a través de proveedores de datos (*DataProvider*). Estos proveedores permiten el acceso a datos según su localización, como a datos almacenados en memoria (*Record*), en un archivo (*File*), en una base de datos (*Database*), o en un *stream* de datos (*Stream*). Cabe destacar que el análisis de datos en *stream* en DIA, si bien es posible, no es tan común como el análisis de datos por lotes (en *batch*) [2, 81]. Por tanto, el elemento *Stream* permite la definición de *stream* de datos, cuyo procesamiento dependerá específicamente del ejecutor subyacente del flujo de trabajo [103].

La información extraída se transforma mediante tareas de diferentes naturalezas (véase el paquete *Activities* en la Figura 3.4). Las tareas computacionales (*ComputationalTask*) son tareas ejecutables realizadas sin interacción humana con el objetivo de transformar entradas de datos en nuevos datos de salida. Estas tareas consisten típicamente en invocar servicios (*Service*) —por ejemplo, servicios web (*WebService*)

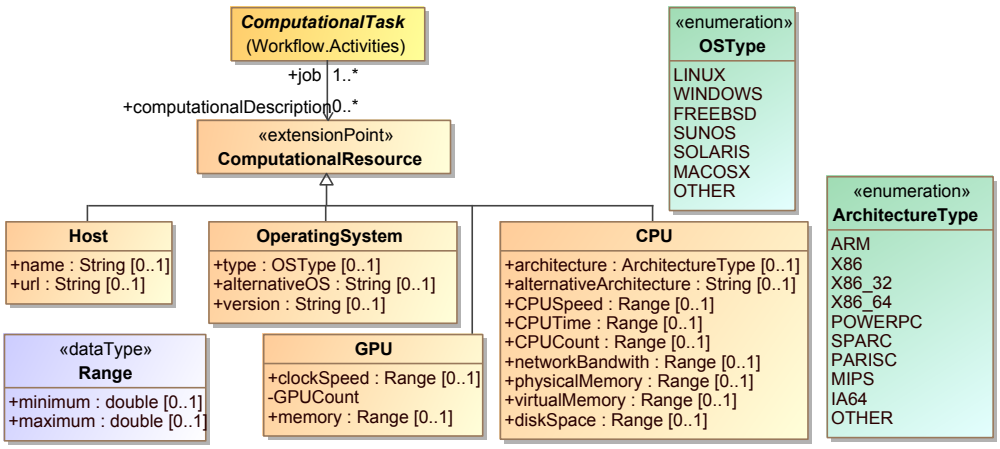


Figura 3.5: Elementos del metamodelo en la capa sintáctica: paquete *ComputationalSpecification*.

que incluyen REST[11] (*RestClient*) y SOAP[12] (*SoapClient*)— o procesos (*Process*), como procesos de Java (*JavaProcess*) o programas de CLI (*CliProcess*). En este caso, tanto *Service* como *Process* son puntos de extensión. Esto significa que se pueden agregar otros tipos de tareas a SWEL si fuese necesario. Por ejemplo, existen algunos dominios intensivos en datos donde puede ser necesario definir algunas tareas que involucren a humanos, aunque encontrar este tipo de tareas no es común en DIA. Además, los propios DIW también pueden ser considerados como procesos en sí mismos, y es habitual encontrar la definición de DIW anidados.

Otro tipo de actividad son las tareas de visualización de datos (*DisplayTask*). Este tipo de tareas están orientadas a facilitar la legibilidad y comprensión de los datos, así como de cualquier tipo de información, por parte del humano. Se han definido dos tipos principales de tareas de visualización: aquellas que están integradas y son configuradas dentro del *pipeline* (*EmbeddedDisplay*), y aquellas que muestran el resultado a través de una herramienta de visualización externa (*DelegatedDisplay*).

Finalmente, existen algunos requisitos específicos, como la gestión de la seguridad o ciertas limitaciones específicas de la plataforma, que pueden requerir una especificación explícita del entorno de ejecución y de los recursos computacionales. Por lo general, esta información se puede utilizar para que un SWfMS en particular pueda

[11] REST - REpresentational State Transfer: https://www.w3.org/2001/sw/wiki/REST (consultado: 23/12/2023)

[12] Especificación de SOAP: https://www.w3.org/TR/soap/ (consultado: 23/12/2023)

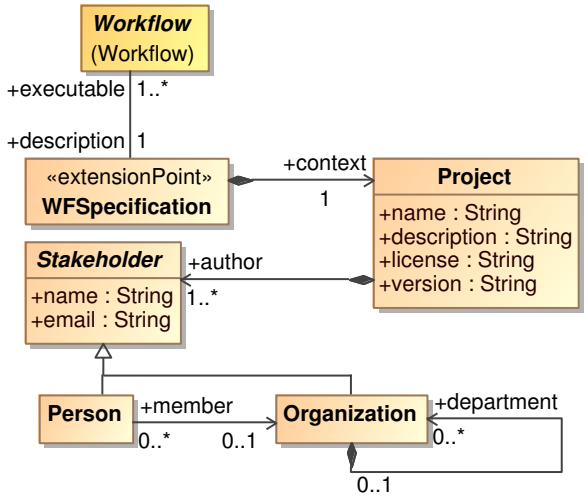
optimizar la ejecución de un DIW aprovechando todo el potencial de la plataforma de ejecución o especificando los requisitos computacionales necesarios. En el paquete *ComputationalSpecification* (véase la Figura 3.5), estos recursos computacionales (*ComputationalResource*) permiten definir tanto la ubicación del *host* (*Host*), como las restricciones en la plataforma y el sistema operativo (*OperatingSystem*), así como los requisitos de CPU y GPU (elementos *CPU* y *GPU*, respectivamente). Destacar que la definición de estos recursos computacionales es extensible para poder incluir recursos computacionales específicos de cada plataforma, siendo su definición actual compatible con JSDL (*Job Submission Description Language*).[13].

3.5.4. Capa de especificación

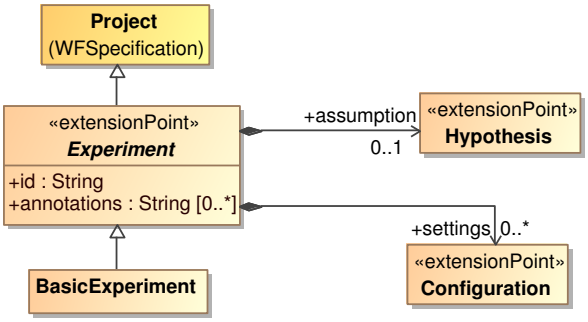
En el mayor nivel de abstracción, el paquete *WFSpecification* (véase la Figura 3.6a) define los elementos relacionados con la meta-información sobre un DIW, como una descripción general, sus términos y condiciones de uso, o el número de versión. Esta información no es ejecutable sino que está orientada a proporcionar una descripción general del proyecto (*Project*), como su nombre, licencia o versión. La información sobre los distintos participantes (*Stakeholder*) permite incluir la descripción de la organización (*Organization*) y las personas involucradas (*Person*).

Los DIW suelen diseñarse en el contexto de un experimento o proyecto en particular. Esta información puede ser modelada por los elementos proporcionados por el paquete *ExperimentSpecification* (véase la Figura 3.6b). Así, SWEL permite proporcionar meta-información sobre un experimento intensivo en datos (*Experiment*), que puede formularse para validar o rechazar una hipótesis (*Hypothesis*). En este contexto, un experimento intensivo en datos es un experimento computacional que se configura mediante un conjunto de propiedades (*Configuration*). Dado que hay diferentes tipos de experimentos científicos, y no está dentro del alcance de SWEL cubrir todos ellos, solo se proporciona un tipo esencial de experimento (*BasicExperiment*). Sin embargo, se permite la inclusión de nuevos tipos de experimentos a través de un punto de extensión. Cabe señalar que los conceptos de esta capa se basan los ya utilizados por otros lenguajes especializados en la definición de experimentos científicos, como SEDL (*Scientific Experiments Description Language*) [106].

[13]Especificación de JSDL, 1.0: http://www.ogf.org/documents/GFD.136.pdf (consultado: 18/10/2023)



(a) Paquete *WorkflowSpecification*.



(b) Paquete *ExperimentSpecification*.

Figura 3.6: Elementos del metamodelo en la capa de especificación.

3.6. Demostración de SWEL

Una vez creado el metamodelo de SWEL, se procede a demostrar cómo se comporta con respecto al problema identificado. En primer lugar, se muestra la independencia de cualquier plataforma mediante la definición de distintas notaciones. Además, también se demuestra que SWEL es compatible con la sintaxis concreta de lenguajes ya existentes. Finalmente, se explica cómo es posible alcanzar la interoperabilidad entre SWfMS gracias al uso de su metamodelo como elemento central del proceso de herradura de MDSE.

3.6.1. Sintaxis concreta gráfica y textual

Aunque SWEL es independiente de cualquier plataforma y de cualquier notación, sus elementos pueden ser asociados a una sintaxis concreta, ya sea gráfica o textual, para favorecer que la definición de DIW pueda ser utilizada por expertos en el dominio de la aplicación y ser gestionada por un SWfMS. En esta sección, presentamos dos ejemplos ilustrativos de dicha asociación a notaciones concretas, una gráfica y otra textual. Para la primera, la Tabla 3.2 muestra el subconjunto de SWEL que será representado gráficamente. La columna *Tipo de SWEL* indica el nombre de la metaclase abstracta que agrupa metaclases concretas relacionadas. La columna *Elemento de SWEL* muestra el nombre de la metaclase de lenguaje que se asociará a un *Elemento gráfico* particular.

Tipo de SWEL	**Elemento de SWEL**	**Elemento gráfico**
DirectedEdge	ControlLine	
	DataLine	
	ExceptionLine	
ControlStructure	Begin	
	End	
	Conditional	
	Fork	
	Synchronizer	
	Merge	
	Counter	
Activity	ComputationalTask	
	DisplayTask	
DataProvider	Record	
	File	
	Stream	
	Database	
Experiment	BasicExperiment	

Tabla 3.2: Ejemplo parcial de sintaxis concreta.

La Figura 3.7 ilustra el uso de esta notación gráfica para la definición de un DIW obtenido de un repositorio público de DIW [14]. Este es un fragmento de un *pipeline* dirigido por los datos que cuenta el número de publicaciones y citas por año para un autor, cuya información es consultada a través de un servicio de información biomédica en particular. Primero, busca publicaciones (tareas computacionales *searchPublications_input* y *searchPublications*) para recuperar registros bibliográficos de todos los artículos publicados por un solo autor (registro *author_name*). Luego, extrae las citas para obtener el año de todos los artículos citados (tarea computacional *extract_citation_pubYear*). El año de publicación se extrae (tarea computacional *extract_pubYear*) de los resultados en crudo del servicio web. Las citas y los años de publicación se envían tanto a una tarea de visualización para producir histogramas 2D como salida, como a una tarea que genera un informe que muestra el número de citas por año. En este caso de uso en particular, no es necesario utilizar estructuras de control, ya que es un *pipeline* dirigidos por los datos. Por lo tanto, únicamente las dependencias de datos existentes entre las tareas computacionales, junto con los puertos de entrada y salida correspondientes, serán utilizados por el motor de flujo de trabajo para determinar el orden de la ejecución y los patrones de composición subyacentes.

En el caso de la Figura 3.8, se utiliza la misma notación para la representación de un *pipeline* dirigido por flujos de control. Este *pipeline* tiene como objetivo realizar tanto una búsqueda a través de un servicio particular, como la validación posterior de los resultados que finalmente se exportan a archivos de diferentes formatos. A diferencia del ejemplo anterior, se muestra una DIA en un dominio con requisitos centrados en los flujos de control. Por lo tanto, se utilizan una serie de estructuras de control, como bifurcaciones o sincronizaciones, que definen el orden de ejecución según la lógica de negocio, en lugar de las dependencias de datos y su composición.

El Código 3.1 muestra un fragmento del DIW representado en la Figura 3.7, pero esta vez utilizando una notación textual de SWEL basada en JSON. Con esta sintaxis, en un primer nivel se define la meta-información del proyecto del paquete *Specification*. Un segundo nivel se utiliza para la definición de los distintos DIW que conforman este proyecto, incluyendo sus elementos: los nodos (tipo, atributos, configuración, notas, etc.) y las conexiones de control y de datos que existen entre ellos. Para el

[14] myExperiment. https://www.myexperiment.org (consultado: 10/01/2024)

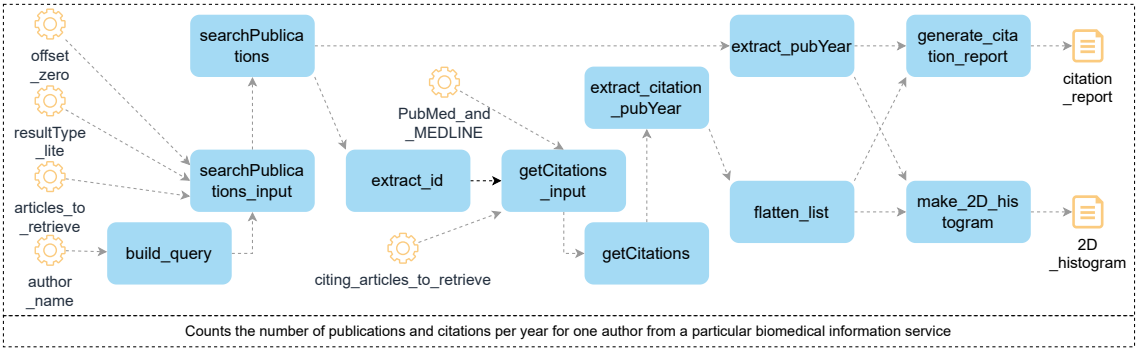


Figura 3.7: Representación en SWEL de un flujo de trabajo dirigido por los datos.

caso de este ejemplo, este fragmento de código JSON define un DIW (línea 1) con dos nodos (línea 2): un proveedor de datos de tipo *record* (línea 3) al que se le asigna un nombre (línea 4) y su respectivo valor (línea 5), y un servicio web (líneas 6–8) con información sobre la operación particular a ejecutar (línea 9). Ambos están vinculados por una conexión de datos (líneas 11–13).

Código 3.1: Fragmento de código de una notación de SWEL basada en JSON.

```
1  ... {"type": "dataDrivenWorkflow",
2      "nodes": [{
3          "type": "record",
4          "name": "citing_articles_to_retrieve",
5          "value": "100"
6      },{ "type": "soapClient",
7          "name": "searchPublications",
8          "url": "http://www.ebi.ac.uk/(..)/soap?wsdl",
9          "operation": "searchPublications" }, ...],
10     "links": [{
11         "type": "data",
12         "source": "extract_id",
13         "target": "getCitations_input" }, ...] }
```

La estructura de cualquier JSON puede ser descrita mediante otro JSON específico denominado *esquema* [15], lo que permite el desarrollo de un validador sintáctico. Este validador es una herramienta útil para verificar que el documento JSON está bien construido y satisface una estructura en particular. Con este fin, se ha diseñado y desarrollado un esquema JSON de acuerdo con la declaración de la notación concreta basada en JSON correspondiente al metamodelo de SWEL. La herramienta de validación se ha implementado en Java y proporciona una interfaz de línea de comandos cuyo único argumento es la ruta del archivo JSON a analizar. A partir

[15] JSON-Schema. http://json-schema.org (consultado: 23/10/2023)

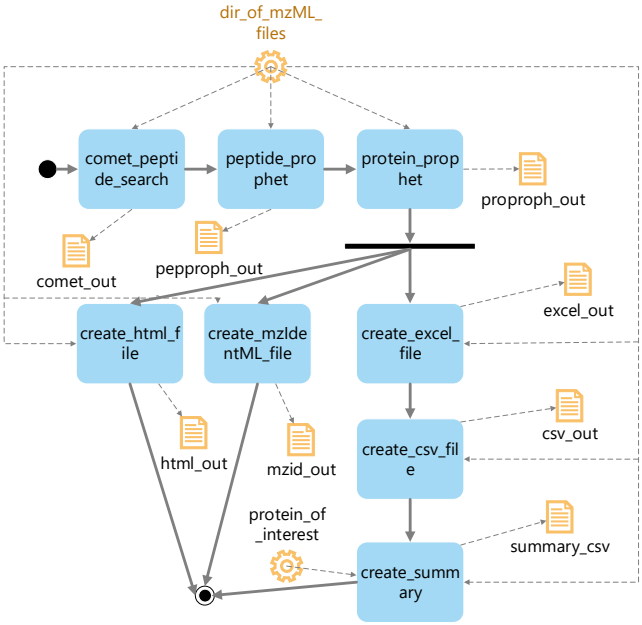


Figura 3.8: Representación en SWEL de un flujo de trabajo dirigido por la lógica de negocio.

de esta ruta, se lee y analiza el contenido del archivo para verificar si su contenido cumple con el esquema JSON. Si no cumple, la consola de ejecución informa sobre los errores detectados. Tanto el esquema JSON como el validador sintáctico están disponibles para su descarga en el material suplementario de esta tesis [134]

3.6.2. Compatibilidad con sintaxis concretas existentes

Cualquier DIW definido mediante SWEL no está acoplado a ninguna sintaxis concreta en particular. Por esto, dado que SWEL es independiente de cualquier plataforma y herramienta, se pueden utilizar múltiples sintaxis concretas para definir un DIW.

Para demostrar que SWEL puede ser asociado a múltiples sintaxis concretas se han utilizado dos SWfMS diferentes. La Tabla 3.3 muestra la relación correspondiente entre algunas de las metaclases de SWEL y la Tabla 3.4 muestra cómo se pueden asociar los principales elementos de LONI Pipeline con algunas de las metaclases de SWEL.

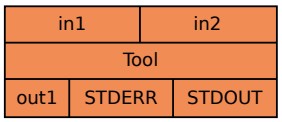
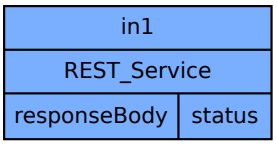
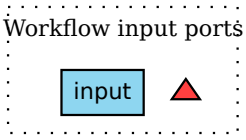
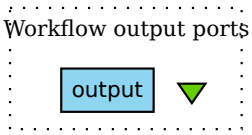
Metaclase	Notación
Activities::CliProcess	
Activities::RestClient	
DataProviders::Record (Input)	
DataProviders::Record (Output)	
ExecutionGraph::DataLine	

Tabla 3.3: Relación parcial de SWEL con la sintaxis concreta de Taverna.

A continuación se define un DIW en un dominio particular utilizando SWEL para su especificación a modo de estudio de caso. Para su representación se utilizan ambas sintaxis concretas.

Descubrimiento de enfermedades

Este DIW tiene como objetivo descubrir posibles enfermedades a partir de un conjunto de palabras clave introducidas por el usuario. En primer lugar, se obtiene un conjunto de documentos médicos, según una serie de parámetros específicos, y se procesan para extraer una lista de proteínas asociadas a diferentes enfermedades. A continuación, esta lista se utiliza para descubrir nuevas enfermedades relacionadas con las palabras clave dadas. La Figura 3.9 muestra la representación de este DIW utilizando la sintaxis concreta de Taverna presentada anteriormente, y la Figura 3.10 ilustra el mismo DIW utilizando la sintaxis concreta de LONI Pipeline.

Metaclass	Notación
Activities::CliProces	
ControlStructures::Conditional	
DataProviders::File (Input)	
DataProviders::Record (Input)	
DataProviders::File (Output)	
DataProviders::Record (Output)	
ExecutionGraph::DataLine	

Tabla 3.4: Relación parcial de SWEL con la sintaxis concreta de LONI Pipeline.

3.6.3. Interoperabilidad entre SWfMS

A menudo, es necesario reutilizar o adaptar el conocimiento de un DIW en diferentes dominios de aplicación distintos al original, o incluso reutilizar el conocimiento dentro del mismo dominio pero en herramientas diferentes. Por ejemplo, imagínese que un científico de datos genera un *pipeline* utilizando la herramienta Taverna. Si un fragmento de ese DIW necesita ser reutilizado o compartido para su uso en otro SWfMS como Kepler, debería ser diseñado nuevamente en la nueva herramienta. Aquí, además de los costes de recursos y tiempo que se necesitan, entran en juego factores como la interpretación de la persona que rediseñó el DIW y las posibles imprecisiones en la representación con respecto al *pipeline* original.

La interoperabilidad es uno de los beneficios que puede ser alcanzado mediante el uso de técnicas procedentes de la MDSE. Más específicamente, la aplicación del proceso de herradura [74] permite la reutilización de fragmentos de contenido generados con diferentes herramientas, tal y como se puede observar en la Figura 3.11. Como se observa, los DIW se definen en diferentes niveles de abstracción para obtener modelos

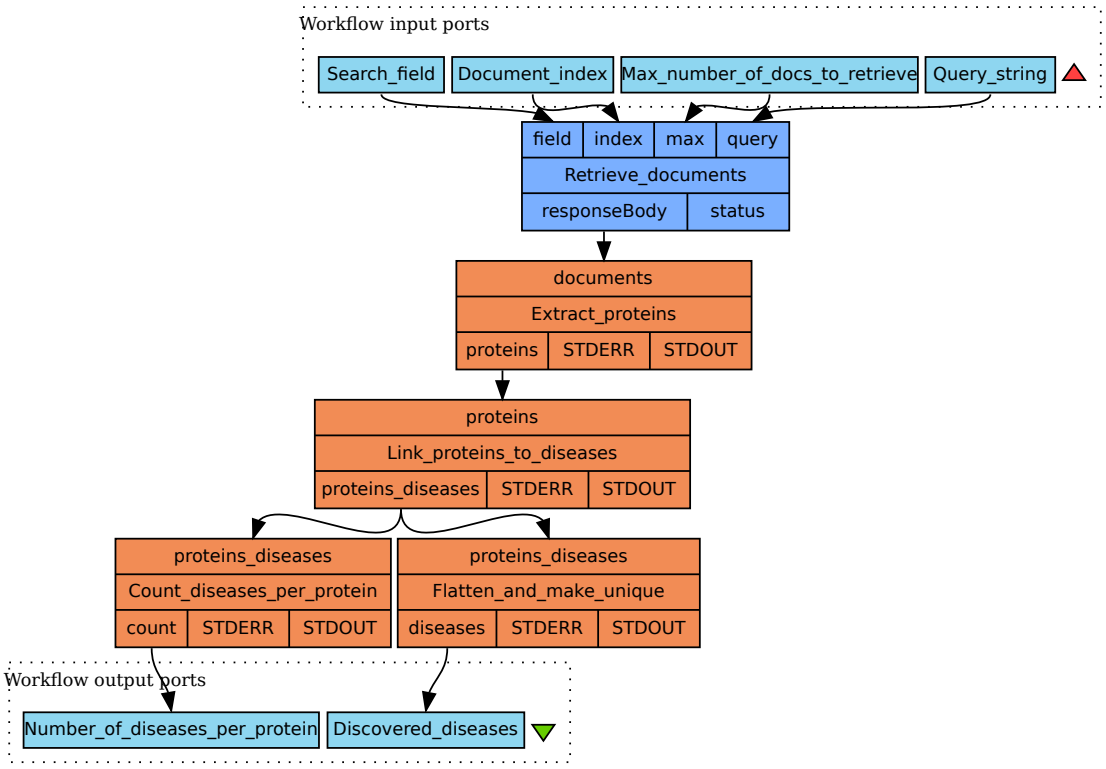


Figura 3.9: DIW utilizando la sintaxis concreta de Taverna.

a partir del código fuente, transformar esos modelos conforme a la plataforma de destino y, finalmente, generar el nuevo código fuente.

SWEL puede servir como el elemento central del proceso de herradura entre los lenguajes de DIW específicos de plataforma, SCUFL y MoML. La Figura 3.11 muestra el proceso implementado. Partiendo del código fuente generado por Taverna en el lenguaje de SCUFL, su representación se extrae como un modelo de SCUFL, en conformidad con el metamodelo de SCUFL. La representación abstracta del *pipeline* generado por el usuario se convierte en un modelo de SWEL mediante transformaciones de modelos unidireccionales. Esta especificación del modelo de SWEL, independiente de cualquier plataforma, se transforma en modelos de MoML (Kepler) y, posteriormente, en texto. Obsérvese que no existen metamodelos formalizados para SCUFL y MoML. Por ello, hemos definido sus respectivos metamodelos específicos de la plataforma mediante un proceso de reingeniería, recopilando parcialmente las principales características necesarias para este estudio de caso. El proceso de herra-

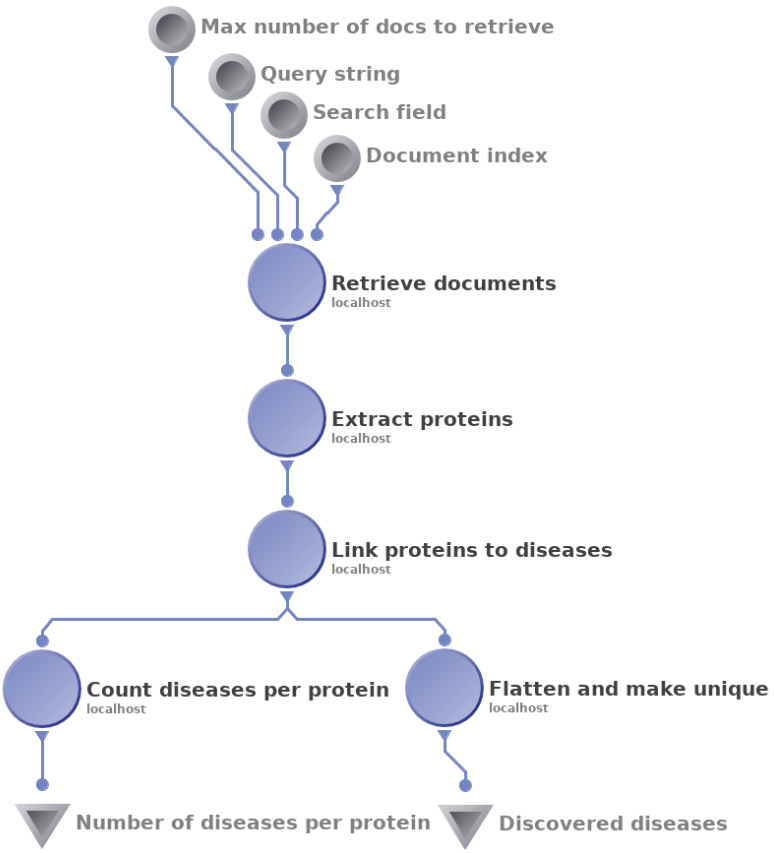


Figura 3.10: DIW utilizando la sintaxis concreta de LONI Pipeline.

dura mostrado en la Figura 3.11 podría ampliarse incorporando otros metamodelos (otras plataformas o SWfMS) según las necesidades de interoperabilidad específicas.

Para ilustrar este caso particular, la Figura 3.12 muestra un DIW en SCUFL, utilizado para extraer información sobre un gen particular de una base de datos de proteínas nucleares. Este DIW sería convertido en un DIW correspondiente en MoML, tal y como se ilustra en la Figura 3.13. Para ello, seguimos el proceso de herradura indicado en la Figura 3.11.

En primer lugar, se define una transformación T2M para convertir el código fuente de SCUFL en su modelo correspondiente. Dado que SCUFL es un lenguaje basado en XML, se utiliza el lenguaje de transformaciones estándar XSLT (*eXtensible Stylesheet Language for Transformations*). En el Código 3.2 se muestra la plantilla XSLT que transforma las conexiones de datos SCUFL (*Datalink*) en los elementos correspondientes del modelo. Además, se utilizan plantillas similares para transfor-

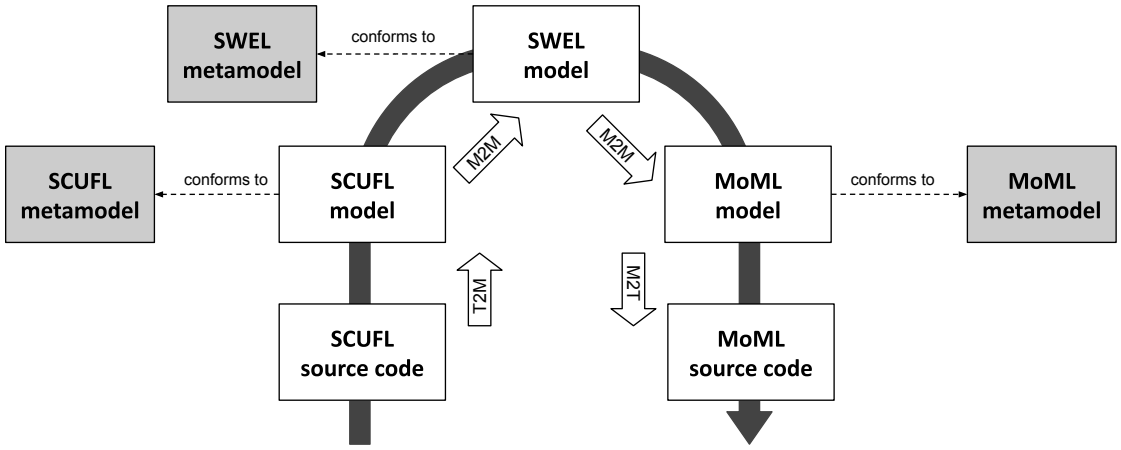


Figura 3.11: Proceso de herradura con SWEL como pivote para la interoperabilidad.

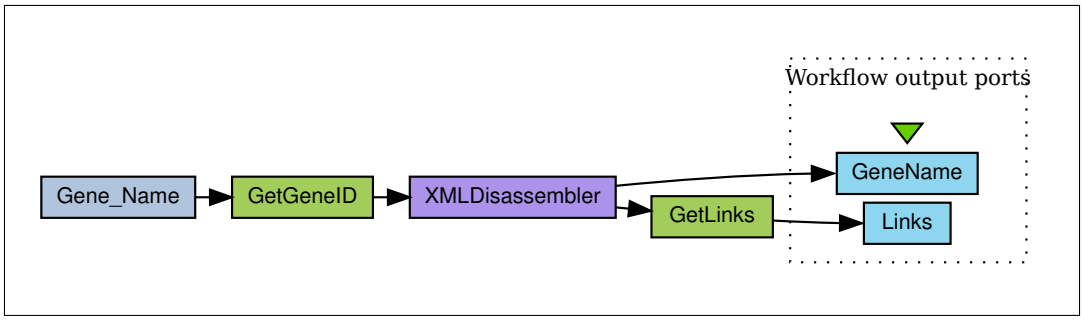


Figura 3.12: Representación del DIW origen de SCUFL.

mar el resto de los elementos y estructuras SCUFL, como las entradas y salidas de datos, o los elementos de procesamiento.

Código 3.2: Definición de la plantilla para Datalinks en XSLT

```
1  <xsl:template name="datalinks" match="datalinks/datalink">
2    <connection><source>
3      <xsl:attribute name="type"><xsl:value-of select="source/@type" /></
           xsl:attribute>
4      <xsl:attribute name="port"><xsl:value-of select="source/port" /></xsl:attribute
           >
5      <xsl:attribute name="processor"><xsl:value-of select="source/processor" /></
           xsl:attribute>
6    </source><target>
7      <xsl:attribute name="type"><xsl:value-of select="sink/@type" /></xsl:attribute>
8      <xsl:attribute name="port"><xsl:value-of select="sink/port" /></xsl:attribute>
9      <xsl:attribute name="processor"><xsl:value-of select="sink/processor" /></
           xsl:attribute>
10   </target></connection>
11 </xsl:template>
```

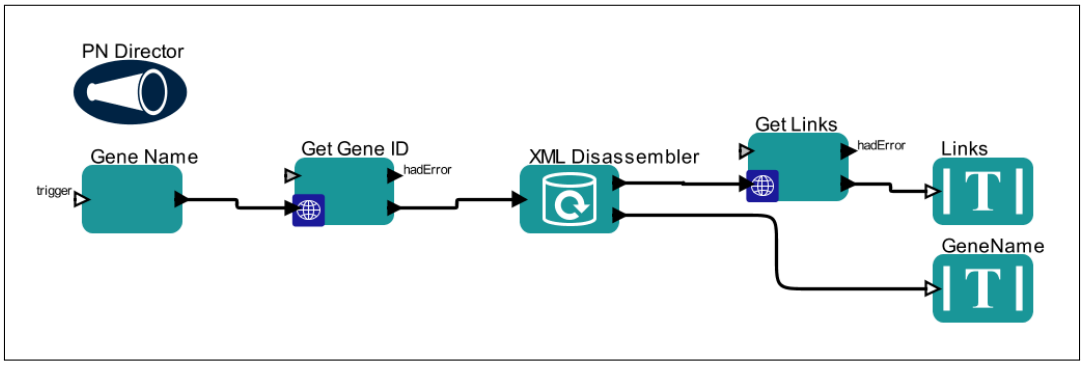


Figura 3.13: Representación específica del DIW generado para MoML de Kepler.

A continuación, las transformaciones de modelos M2M, definidas mediante QVT, declaran las relaciones y dependencias entre los modelos SCUFL y SWEL. Cada relación se ejecuta cuando se cumplen una serie de pre-condiciones específicas, y define un conjunto de post-condiciones para determinar el orden de ejecución de las relaciones posteriores. La primera relación inicia el proceso de conversión en un DIW de SWEL (*Workflow*). Así, cada elemento de SCUFL es transformado en el elemento correspondiente en SWEL como, por ejemplo, los procesadores (*Processor*) en nodos (*Node*), los enlaces de datos (*Datalink*) en líneas de datos (*Dataline*), o los puntos de entrada y salida (*Endpoint*) en puertos de entrada y salida (*Port*). A modo de ilustración, la relación QVT *MapRetryDispatchLayer* se muestra en el Código 3.3, asociando la capa de gestión de errores al volver a intentar una ejecución después de la detección de un error (*ConfigBeans::RetryConfig*) desde SCUFL (líneas 4–6) en la ruta de excepción correspondiente en SWEL (*Workflow::ExceptionPath*) (líneas 7–11).

Código 3.3: Declaración en QVT de SCUFL a SWEL de la relación *MapRetryDispatchLayer*.

```
1  relation MapRetryDispatchLayer {
2      ttimes : Integer;
3      tdelay : Integer;
4      checkonly domain source s : SCUFL::ConfigBeans::RetryConfig
5      {   maxRetries = ttimes,
6          initialDelay = tdelay
7      };
8      enforce domain target t : SWEL::Workflow::ExceptionPath
9      {   action = retry : SWEL::Workflow::Retry {
10         {   times = ttimes,
11             delay = tdelay
12         },
13         cause = unknown : SWEL::ExceptionTypes::UnknownException {} };
14 }
```

Nada impide que la definición actual del DIW en SWEL sea transformable a cualquier otra representación. Así, siguiendo el proceso de herradura, se definen nuevas transformaciones M2M en QVT para definir la correspondencia entre los elementos de SWEL y MoML. Una primera transformación convierte cada nodo y línea de flujo de trabajo de SWEL en la entidad MoML correspondiente (*Entity*), relación (*Relation*) y enlace (*Link*). La relación QVT *MapRecord* se implementa en el Código 3.4, que muestra la transformación de un proveedor de datos de tipo *record* en SWEL (*Workflow::DataProviders::Record*) (líneas 3–4) en el correspondiente elemento de MoML (*Entity*) (líneas 5–18).

Código 3.4: Declaración en QVT de SWEL a MoML de la relación *MapRecord*.

```
relation MapRecord {
    evalue : String;
    checkonly domain source r : SWEL::Workflow::DataProviders::Record
    { value = evalue };
    enforce domain target e : MoML::Entity
    {   class = 'ptolemy.actor.lib.StringConst',
        attribute = value : MoML::Property
        {   name = 'value',
            class = 'ptolemy.data.expr.Parameter',
            value = evalue },
        attribute = firingCountLimit : MoML::Property
        {   name = 'firingCountLimit',
            class = 'ptolemy.data.expr.Parameter',
            value = 'NONE' },
        attribute = none : MoML::Property
        {   name = 'NONE',
            class = 'ptolemy.data.expr.Parameter',
            value = '0' }
} };
```

A partir de este modelo MoML específico de la plataforma, se declaran transformaciones M2T en Acceleo[16]. Acceleo utiliza la especificación del lenguaje de transformación de modelos MOF2T (*MOF Model to Text Transformation Language*) [17]. El proceso de generación de todos los elementos se inicia mediante la invocación de la plantilla *GenerateEntity* (véase el Código 3.5). Aquí, se crean entidades MoML y sus subentidades, con una configuración dada y sus conexiones. El DIW resultante se representa en la Figura 3.13 siguiendo la notación concreta de Kepler.

[16]Generador de código basado en plantillas de código abierto Acceleo: http://www.eclipse.org/acceleo (consultado: 18/11/2023)

[17]MOF Model to Text Transformation Language (MOFM2T), 1.0: http://www.omg.org/spec/MOFM2T/1.0/ (consultado: 13/11/2023)

Código 3.5: Plantilla de Acceleo para la definición de *GenerateEntity*.

```
[template public generateEntity(entity : Entity)]
<entity name="[entity.name/]" class="[entity.class/]">
    [for (property : Property | entity.attribute)]
        [generateProperty(property)/]
    [/for]
    [for (subentity : Entity | entity.subentity)]
        [generateEntity(subentity)/]
    [/for]
    [for (port : Port | entity.endpoint)]
        ...
        <port name="[port.name/]" class="[port.class/]">
            ...
        </port>
        ...
    [for (relation : Relation | entity.connection)]
        <relation name="[relation.name/]" class="[relation.class/]"
            ...
        </relation>
    [/for]
    [for (link : Link | entity.binder)]
        [generateLink(link)/]
    [/for]
</entity>
[/template]
```

Es interesante destacar que la definición de las transformaciones de modelo ha servido para iterar y refinar el modelo de SWEL con el fin de cubrir las características principales de otros lenguajes existentes. Obsérvese que la falta de estandarización en la definición de DIW puede llevar a funcionalidades incompatibles en los SWfMS existentes, lo que puede dificultar la interoperabilidad entre ellos.

El total de las transformaciones definidas y utilizadas en esta sección pueden ser consultadas en detalle en el Apéndice B. Además, se ha desarrollado una herramienta de demostración para ilustrar el proceso completo de transformación de herradura y facilitar la interoperabilidad entre diferentes SWfMS. Actualmente, esta herramienta admite como entrada tanto DIW definidos tanto con SCUFL como con CWL, con el fin de generar DIW de MoML. Durante el proceso de transformación, se muestra la salida de cada transformación de modelos (de texto a modelo, de modelo a modelo y de modelo a texto), con la opción de guardarlas en un directorio local.

Se han recopilado la definición de los metamodelos, las transformaciones utilizadas en esta sección, junto con la herramienta que se ha desarrollado para ilustrar todo

el proceso de transformación. Estos, junto con algunos vídeos que ilustran su uso, están disponibles para su consulta y descarga en el material suplementario de esta tesis [134].

3.6.4. Definición colaborativa de flujos de trabajo intensivos en datos

En muchos casos, la definición de un DIW es un esfuerzo colectivo, donde profesionales de diferentes disciplinas y organizaciones trabajan para alcanzar un objetivo común, generalmente utilizando distintos SWfMS. En este contexto, es necesario disponer de mecanismos para propagar y actualizar los cambios sobre la definición de un DIW de manera consistente con el fin de mantener la sincronización entre todos los participantes y herramientas involucradas. Desafortunadamente, esta sincronización suele lograrse de manera manual, lo que es un proceso altamente propenso a errores y que no permite actualizaciones instantáneas o automáticas. Sin embargo, es posible definir transformaciones de modelos bidireccionales para propagar los cambios de los modelos origen a los modelos destino, y viceversa, manteniendo automáticamente la sincronización entre ellos. En lugar de desarrollar transformaciones independientes unidireccionales, las transformaciones bidireccionales permiten reducir el tiempo de desarrollo y evitar la necesidad de tener que diseñar transformaciones unidireccionales que sean mutuamente consistentes.

Lo más interesante es que las transformaciones bidireccionales permiten mantener sincronizada la misma definición de un DIW, aunque haya sido diseñada mediante dos lenguajes de DIW diferentes, como SWEL y CWL. Para demostrar cómo SWEL permite la definición colaborativa, se ha utilizado un DIW sencillo. Concretamente, el DIW representado en la Figura 3.14 calcula cuántas veces aparece una palabra en un documento en particular mediante la ejecución de dos tareas: el proceso *get-document* accede al documento a través de una URL (registro *url*) y almacena su contenido en un archivo de texto. Luego, el proceso *count-document-words* lee el archivo y extrae la lista de palabras junto con el número de sus ocurrencias. La lista resultante se almacena en un archivo de texto, llamado *count*.

Para este estudio de caso, se ha definido una serie de transformaciones bidireccionales con QVT entre SWEL y CWL. Dado que CWL no tiene un metamodelo

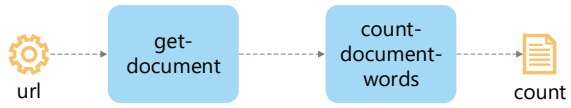


Figura 3.14: Representación de un DIW simple en SWEL.

formalizado, se ha definido manualmente un metamodelo CWL según los requisitos de este caso de uso en particular. El Código 3.6 ilustra la relación QVT *MapWorkflowCommandLineTool2CliProcess*, que define la correspondencia entre el proceso CLI definido por CWL (líneas 4–6) y el proceso CLI de SWEL (líneas 7–9).

Cabe destacar que en la relación tanto CWL (CommandLineTool) como SWEL (CliProcess) se han añadido como *enforce domain* para permitir a QVT realizar la transformación en ambas direcciones. En contraposición, el uso de *checkonly domain* denota que este modelo es de solo lectura y, en consecuencia, la relación es unidireccional.

Código 3.6: Relación bidireccional de QVT: *MapWorkflowCommandLineTool2CliProcess*.

```
1 relation MapWorkflowCommandLineTool2CliProcess {
2     varName : String;
3     varCommand : String;
4     enforce domain cwl clt : commonwl::CommandLineTool
5     {   name = varName,
6         baseCommand = varCommand };
7     enforce domain swel cp : SWEL::Workflow::Activities::CliProcess
8     {   name = varName,
9         executable = varCommand }; }
```

Nótese que alcanzar una consistencia completa entre DIW depende directamente de la tecnología utilizada para las transformaciones bidireccionales. Este es un campo de investigación en activo, por lo que muchas de las herramientas actuales son prototipos o aún están en fase de desarrollo.

El proceso de definición de las transformaciones utilizadas en esta sección es detallado en el Apéndice B. Además, estas transformaciones forman parte de la herramienta de demostración desarrollada para ilustrar el proceso completo de transformación de herradura, descrito en la Sección 3.6.3.

3.7. Evaluación de SWEL

A continuación, conforme a la metodología DSR, se evalúa SWEL para mostrar en qué medida resuelve el problema. Para ello, se realiza una validación de los requisitos, y se comprueba su idoneidad y adaptabilidad mediante un análisis cuantitativo, así como mediante la evaluación de expertos. Finalmente, se hace un análisis de amenazas a la validez.

3.7.1. Requisitos de validación

Para validar los requisitos descritos en la Sección 3.4, se desarrolló un artefacto, SWEL, y múltiples artefactos de apoyo o prototipos (véase la Sección 3.6). El *prototipado* es una técnica de validación común y bien conocida, que permite probar y experimentar con el lenguaje presentado para comprobar si cumple con los requisitos especificados. Otra técnica utilizada ha sido la *revisión continua de requisitos*, donde se revisan y contrastan los requisitos para verificar posibles errores y ambigüedades. La Tabla 3.5 representa los principales requisitos de investigación, así como los resultados generados que se han utilizado para la validación.

Tanto los artefactos primarios como los de apoyo generados (véase el material suplementario [134]) permiten cubrir todos los requisitos de investigación establecidos para su validación. Tanto el lenguaje como los artefactos asociados fueron evaluados continuamente por profesionales para asegurarse de que los resultados ayudan a resolver los problemas originales. Gracias a esto, se obtuvo una serie de comentarios muy valiosos durante todo el proceso, asegurando que los resultados fueran útiles y precisos según sus necesidades.

3.7.2. Evaluación cuantitativa

Para evaluar la idoneidad y adaptabilidad de SWEL en la definición de DIW, se ha llevado a cabo un análisis cuantitativo utilizando como referencia el *framework* propuesto por Guizzardi *at al.* [63]. Este *framework* evalúa lenguajes de modelado en base al análisis de cuatro métricas principales: *lucidity*, *soundness*, *laconicity* y *completeness*. SWEL ha sido comparado con los tres lenguajes de DIW más

***REQ1**: Lenguaje abstracto de DIW de alto nivel para la definición de DIA; **REQ2**: Reutilización del conocimiento entre distintos SWfMS con un lenguaje independiente de cualquier plataforma; **REQ3**: Lenguaje independiente de cualquier notación concreta en términos de sintaxis concreta.*

	REQ1	**REQ2**	**REQ3**
SWEL:Morphological layer (Sección 3.5.2)	X		
SWEL:Syntactic layer (Sección 3.5.3)	X		
SWEL:Specification layer (Sección 3.5.4)	X		
Sintaxis concreta de SWEL: notación gráfica (Sección 3.6.1)	X		X
Sintaxis concreta de SWEL: notación textual basada en JSON (Sección 3.6.1)	X		X
Validador sintáctico de notación basada en JSON (Sección 3.6.1)			X
Compatibilidad con notaciones concretas existentes (Sección 3.6.2)		X	X
Estudio de caso: Transformaciones unidireccionales T2M y M2T (Sección 3.6.3)		X	
Estudio de caso: Transformaciones unidireccionales M2M (Sección 3.6.3)		X	
Case de estudio: Transformaciones bidireccionales M2M (Sección 3.6.4)		X	
Herramienta de interoperabilidad: SCUFL, MoML, CWL (Sección 3.6.3)		X	

Tabla 3.5: Resultados de la validación de los requisitos de la investigación.

relevantes: SCUFL y MoML son los lenguajes definidos por dos SWfMS populares (Taverna y Kepler, respectivamente), CWL es un enfoque reciente destinado a establecer un estándar abierto. Por lo tanto, cabe destacar que los resultados de la evaluación deben interpretarse en este contexto, especialmente en lo que respecta a la completitud, expresividad y concisión del lenguaje. Para realizar una comparación justa, solo se han tenido en cuenta los elementos de SWEL que se utilizan en *pipelines* dirigidos por los datos. Además, dado que SWEL es un lenguaje extensible, se han definido algunos puntos de extensión en SWEL para poder incluir elementos específicos de los lenguajes comparados. En particular, se ha definido una lista de puntos de extensión en SWEL para cumplir con todas las tareas particulares requeridas por SCUFL: *Process$SCUFLBeanShell* define aque-

Métricas	SCUFL	MoML	CWL
Lucidity	13/18 (72.22 %)	11/12 (91.67 %)	3/13 (23.08 %)
Soundness	13/18 (72.22 %)	11/12 (91.67 %)	7/13 (53.85 %)
Laconicity	13/13 (100 %)	11/11 (100 %)	5/7 (71.43 %)
Completeness	13/13 (100 %)	11/11 (100 %)	7/7 (100 %)

Tabla 3.6: Evaluación cuantitativa de la sintaxis abstracta de SWEL.

llas actividades implementadas utilizando BeanShell[18] (un lenguaje de comandos compatible con Java); *Process$SCUFLRshell* define aquellas actividades implementadas utilizando el lenguaje de programación R; *Process$SCUFLInteraction* permite a los humanos a tomar ciertas decisiones mediante una interacción sencilla; *Process$SCUFLXPath* define expresiones para consultar o transformar documentos XML; *Process$SpreadsheetImport* permite la lectura de datos similares a una hoja de cálculo; y *Process$SCUFLLocalworker* que define la información sobre programas Java. Además, se ha implementado un punto de extensión para definir código JavaScript en CWL (*Process$CWLExpressionTool*).

Los resultados de estas métricas se muestran en la Tabla 3.6. *Lucidity* mide el grado de claridad de SWEL en términos de cuántos elementos del lenguaje tienen una representación única en los otros lenguajes. Los valores obtenidos muestran que SWEL es altamente expresivo, conciso y claro. En el caso de CWL, los bajos porcentajes se deben principalmente a que este lenguaje aún se encuentra en fase de definición y proporciona pocos elementos para especificar DIW. Este hecho también influye en el cálculo de *soundness*, que determina el grado de correspondencia de los elementos de SWEL con los elementos de otros lenguajes. *Laconicity* mide cuán conciso es SWEL considerando el número de elementos de otros idiomas que corresponden a cada elemento en SWEL, y sus valores resultantes están cerca del máximo en todos los casos. Por último, *completeness* indica el grado en que SWEL es compatible con los otros lenguajes. Cabe destacar que esta métrica es crucial para determinar la idoneidad de SWEL para lograr la interoperabilidad entre distintos SWfMS, y es la métrica en la que SWEL obtiene la puntuación más alta. Las asociaciones concretas entre los elementos de SWEL y el resto de lenguajes utilizados para calcular estas métricas se enumeran en la Tabla 3.7. Nótese que estas asociaciones pueden ser *uno a uno* (1:1) y *uno a muchos* (1:N) cuando un único elemento de SWEL se puede

[18] BeanShell: `https://beanshell.github.io/` (consultado: 24/11/2023)

asociar a múltiples elementos de otro lenguajes. Además, también pueden ser *muchos a uno* (N:1) cuando múltiples elementos de SWEL pueden ser asociados a un único elemento de otro lenguaje.

3.7.3. Evaluación por expertos

Como se detalla en la Sección 3.2, SWEL ha pasado por un proceso de refinamiento iterativo y de validación incremental, utilizando el *feedback* de profesionales y expertos en ciencia de datos y DIW. La evaluación de profesionales se ha realizado mediante encuestas para valorar la adecuación de SWEL como un modelo intermedio en el proceso de interoperabilidad entre las herramientas Taverna y Kepler. También analizamos la idoneidad y comprensibilidad de SWEL como lenguaje de representación de DIW. Para este fin, se ha seleccionado un equipo de once expertos, ninguno involucrado en el diseño, desarrollo y evaluación de SWEL. Cinco de los expertos trabajan en la academia en diversas áreas de ciencia de datos, mientras que seis son científicos de datos en la industria. Se han tenido en cuenta perfiles tanto *senior* como *junior* en ambos casos. Estos expertos tienen un amplio conocimiento en DIA y, por lo tanto, estaban bien preparados para proporcionar información relevante. Es importante tener en cuenta que estos especialistas no necesariamente son expertos en las herramientas intensivas en datos que son objeto de esta investigación, aunque sí que están familiarizados con el uso de SWfMS.

Para llevar a cabo el experimento, los expertos utilizaron la herramienta de transformación presentada en el estudio de caso en la Sección 3.6 y respondieron a un cuestionario sobre criterios comunes en la evaluación de herramientas [102]. Los criterios elegidos para formular las preguntas se referían a la eficacia, utilidad, precisión, efectividad, validez y completitud del proceso de conversión de DIW entre Taverna y Kepler. También analizaron la practicidad y comprensibilidad del modelo SWEL generado por la herramienta. El paquete que contiene las preguntas y respuestas del experimento está disponible en el material suplementario [134].

El experimento consta de tres ejercicios: (1) una primera transformación de entrenamiento de un DIW dado; (2) una transformación de un DIW elegido por el experto y descargado de un repositorio público; y (3) un análisis de la idoneidad y comprensibilidad de SWEL como modelo de representación para DIW, así como su precisión

SWEL	SCUFL	MoML	CWL
SoapClient	Activity [class=”net.sf.taverna.t2.activities.wsdl.WSDLActivity”]	Entity [class= .ºrg.sdm.spa.WSWithComplexTypes”]	-
RestClient	Activity [class=”net.sf.taverna.t2.activities.rest.RESTActivity”]	Entity [class= .ºrg.kepler.actor.rest.RESTService”’]	-
CliProcess	Activity[class=”net.sf.taverna.t2.activities.externaltool.ExternalToolActivity”]	Entity [class= ”ptolemy.actor.lib.Exec”]	CommandLineTool
JavaProcess	-	Entity	-
EmbeddedDisplay	-	Entity[class=”ptolemy.actor.lib.gui.Display”]	-
Workflow	Workflow		Workflow
Port	Port	Port	1) WorkflowStepInputParameter 2) WorkflowStepOutputParameter 3) InputToolParameter 4) OutputToolParameter
Record	Activity[class=”net.sf.taverna.t2.activities.stringconstant.StringConstantActivity”]	Entity[class=”ptolemy.actor.lib.StringConst”]	-
File	-	Entity[class=.ºrg.geon.BinaryFileReader”]	-
Stream	-	-	-
Database	-	Entity[class=.ºrg.geon.DatabaseQuery”]	-
Dataline	Datalink	Relation	1) WorkflowStepInputParameter 2) OutputToolParameter
Process$SCUFLBeanShell	Activity[class=”net.sf.taverna.t2.activities.beanshell.BeanshellActivity”]	N/A	N/A
Process$SCUFLRshell	Activity[class=”net.sf.taverna.t2.activities.rshell.RshellActivity”]	N/A	N/A
Process$SCUFLInteraction	Activity[class=”net.sf.taverna.t2.activities.interaction.InteractionActivity”]	N/A	N/A
Process$SCUFLXPath	Activity[class=”net.sf.taverna.t2.activities.xpath.XPathActivity”]	N/A	N/A
Process$SCUFLSpreadsheetImport	Activity[class=”net.sf.taverna.t2.activities.spreadsheet.SpreadsheetImportActivity”]	N/A	N/A
Process$SCUFLLocalworker	Activity[class=”net.sf.taverna.t2.activities.localworker.LocalworkerActivity”]	N/A	N/A
Process$CWLExpressionTool	N/A	N/A	ExpressionTool

Tabla 3.7: Relación entre los elementos de SWEL y los elementos de SCUFL, MoML y CWL.

Los elementos del lenguaje no contemplados por otros lenguajes se indican con un guión. ”N/Aïmplica que el elemento no debe considerarse en el cálculo de las diferentes métricas, ya que se basa en un elemento extendido.

al incorporar los conceptos de los modelos origen (Taverna) y destino (Kepler). Cada experto respondió a un total de 18 preguntas, calificadas en una escala del 1 (más baja) al 10 (más alta).

De acuerdo con los resultados de la encuesta, los expertos otorgaron un promedio de 9.38 a la valoración del ejercicio 2 (transformación de un DIW). Por lo tanto, consideramos que la herramienta resuelve el problema para el cual fue formulada. También podemos concluir que los modelos en los que se inspiró, incluidos los modelos parciales extraídos de SCUFL y MoML, así como los de SWEL, son adecuados en este escenario en particular. En cuanto al ejercicio 3, cabe destacar que no son expertos en MDSE. Por ello, les resultó difícil diferenciar la notación verbosa en XML necesaria para la serialización y lectura[19] de la abstracción del metamodelo. Sin embargo, los expertos dieron una valoración promedio de 7.86 a la comprensibilidad y practicidad del nivel morfológico, en comparación con 8.68 con el que se valoró el nivel sintáctico. Esta diferencia es consistente con el nivel de abstracción de la información representada. De hecho, era esperado que el nivel morfológico tuviera una puntuación más baja, ya que suele ser transparente al científico de datos y desde los SWfMS que lo gestionan. Finalmente, una valoración promedio de 8.73 afirma la importancia de incluir meta-información en el propio DIW.

Tanto el cuestionario como los datos desagregados obtenidos del experimento de la encuesta se pueden consultar en el Apéndice C.

3.7.4. Análisis de amenazas a la validez

De acuerdo con Wohlin *et al.* [156], existen cuatro tipos básicos de amenazas a la validez que pueden afectar la validez de nuestro estudio. A continuación abordamos cada una de ellas en detalle.

Validez externa. Estas amenazas están relacionadas con el grado en el que es posible generalizar los hallazgos y conclusiones de este estudio. En primer lugar, la evaluación comparativa se ha llevado a cabo con un conjunto seleccionado de lenguajes de DIW, que se ha considerado que representan los más importantes. Sin embargo, podría haber otros, o podrían aparecer nuevos que podrían desafiar los

[19]Los modelos intermedios de SWEL se generaron utilizando el formato estándar XMI (*XML Metadata Interchange*): `http://www.omg.org/spec/XMI/` (consultado: 18/11/2023)

resultados obtenidos, especialmente en lo que respecta a la concisión, completitud y expresividad de SWEL. Los puntos de extensión de SWEL se definieron precisamente para abordar este problema, pero no es posible prever todas las características que puedan surgir en el futuro. De todos modos, SWEL podría evolucionar a medida que aparezcan nuevos lenguajes o funcionalidades importantes. Para validar la practicabilidad de SWEL como herramienta de interoperabilidad, y la idoneidad y comprensibilidad del metamodelo propuesto, se llevó a cabo un experimento de encuesta con once expertos de hasta cinco empresas diferentes, así como de la academia.

Validez interna. Estas amenazas están relacionadas con los factores que podrían afectar los resultados de nuestra evaluación. Todas las herramientas desarrolladas y artefactos de software (metamodelos, transformaciones de modelos, etc.) han sido verificados por duplicado para corregir y mitigar estas amenazas. Sin embargo, tendríamos que llevar a cabo más experimentos para re-confirmar tales afirmaciones. Además, para asegurarnos de que las semánticas de los DIW se conservan tras realizar las transformaciones de modelos al convertirlos en modelos de distintos lenguajes, habría que realizar una validación formal. En la práctica, esto es algo complejo y requeriría de un extenso trabajo de investigación en una dirección distinta.

Validez de los fundamentos. Estas amenazas se refieren a la relación entre la teoría y lo observado, y están relacionadas con los problemas que podrían surgir durante el diseño de la investigación. Se ha utilizado un marco comparativo entre SWEL y otros enfoques de DIW. Sin embargo, hay dos aspectos que pueden representar una amenaza para la validez de los fundamentos. Por un lado, según nuestro conocimiento, SWEL es la única propuesta formalizada como un metamodelo, pero se está comparando con propuestas no metamodeladas. Para este propósito, hemos metamodelado parcialmente algunas tecnologías actuales (SCUFL, MoML, CWL), utilizando ingeniería inversa. Sin embargo, estos metamodelos podrían no ser precisos o completos. Hasta ahora, los experimentos llevados a cabo confirman que son adecuados y completos, pero se podrían realizar validaciones adicionales realizando nuevos experimentos de interoperabilidad con todos los tipos de aplicaciones de DIW. Además, SWEL es independiente de la plataforma, por lo que no se enfoca en las características específicamente ofrecidas por una herramienta en particular. Nuevamente, los mecanismos de extensión de SWEL se han diseñado precisamente

para abordar este problema. Estas extensiones serían suficientes para cubrir todas las características necesarias, pero puede ser el caso de que una nueva funcionalidad o una cierta propiedad de un lenguaje en particular no pueda ser expresada. Si fuera el caso, habría que evolucionar el lenguaje para tenerlas en cuenta.

Validez de la conclusión. Estas amenazas se refieren a los problemas que afectan a la capacidad de sacar conclusiones correctas y si los resultados pueden repetirse. En primer lugar, para lidiar con esta amenaza, se ha puesto a disposición pública todos los artefactos desarrollados y utilizados en este trabajo. En segundo lugar, se pueden realizar un mayor número de experimentos con diversos usuarios externos y especialistas de la industria para evaluar SWEL. Esto confirmaría sus propiedades en otros dominios, así como la validación de la idoneidad de SWEL en diferentes casos de uso.

Capítulo 4

Plataforma *low-code* para la generación automática de herramientas intensivas en datos

"La función de un buen software es
hacer que lo complejo aparente ser simple"
Grady Booch

4.1. Introducción

El desarrollo de herramientas intensivas en datos supone un gran desafío a la hora de crear soluciones que satisfagan las necesidades y requisitos particulares de un sector específico, ya que implica afrontar distintos retos técnicos, de diseño y de implementación [20]. Su desarrollo implica una inversión significativa en tiempo y en recursos, al requerir de un conocimiento profundo tanto del dominio intensivo en datos en cuestión, como del propio proceso de desarrollo de software, así como de las diversas tecnologías y técnicas de procesamiento de datos. Por este motivo, no es habitual encontrar soluciones específicas y, cuando existen, se tratan de soluciones *ad-hoc* que han sido implementadas desde cero, particularizadas a tipos de datos y casos de uso muy concretos, lo que dificulta su reutilización en otros dominios y casos de uso similares.

La mayor parte de la oferta de herramientas intensivas en datos se corresponde con herramientas de propósito general, como KNIME o RapidMiner, que los profesionales deben configurar y adaptar, en la medida de lo posible, a su problemática particular [129]. Este proceso de adaptación a requisitos más específicos sigue siendo un proceso costoso en muchos casos, sobre todo en aquellos dominios en los que sus profesionales no son expertos en desarrollo de software o en técnicas computacionales. Esto es debido a que estas herramientas genéricas normalmente están diseñadas para ser utilizadas, precisamente, por expertos en informática o científicos de datos, y no por expertos del dominio sin estos conocimientos.

El enfoque de desarrollo *low-code* ha supuesto un avance considerable en la democratización del desarrollo de software, ya que permite a profesionales no expertos en programación (también conocidos como *citizen developers*) diseñar y crear nuevas aplicaciones con una menor necesidad de poseer conocimientos en programación. Sin embargo, la adaptabilidad de las plataformas *low-code* a dominios específicos continúa siendo costosa [95], ya que, aunque es más ágil que partir desde cero, aún implica una carga considerable para los expertos en el dominio. Por tanto, se hace necesario una solución que combine la agilidad del desarrollo *low-code*, con la capacidad de poder generar aplicaciones ya adaptadas a dominios específicos. Esto permitiría disponer de herramientas con una baja necesidad de configuración, a un menor coste y orientadas a los expertos del dominio de aplicación.

En este capítulo se presenta WORKGENESIS, una plataforma diseñada para abordar los desafíos inherentes a la generación automática de SWfMS, adaptados a dominios específicos y orientados a los expertos del dominio. Tal y como se mencionó en la Sección 2.1, la tecnología de flujos de trabajo ofrece una serie de ventajas significativas en la definición y desarrollo de DIA. WORKGENESIS integra SWEL (véase el Capítulo 3) como parte fundamental de su arquitectura, aprovechando las ventajas relacionadas con la captura, representación y reutilización del conocimiento, y de su capacidad de adaptación a los requisitos de múltiples dominios intensivos en datos.

Así, WORKGENESIS reduce la complejidad de la adaptación de herramientas a los requisitos particulares de los dominios intensivos en datos, como pueden ser los tipos de datos particulares o los tipos de tareas de procesamiento específicas. Además, favorece la reutilización del conocimiento común empleado en distintos dominios

intensivos en datos con requisitos similares [46], como pueden ser tareas de limpieza, acceso o visualización de datos.

WORKGENESIS permite también la separación de las habilidades técnicas requeridas para gestionar y adaptar componentes agnósticos a cualquier dominio, de las habilidades necesarias para definir DIW específicos del dominio a un alto nivel de abstracción. Para ello, se hace uso de los preceptos de MDSE (véase la Sección 2.2.1) para implementar un arquitectura de muticapa, que abarca desde un SWfMS altamente configurable y extensible, hasta mecanismos para la reutilización de servicios y funcionalidades, así como la generación automática de herramientas orientadas a los expertos en el dominio.

4.2. Definición de los requisitos

La necesidad de disponer de SWfMS fáciles de usar, adaptados a dominio específicos e interoperables, ha llevado al diseño de una plataforma modular y adaptable. La idea principal es lograr una clara separación de responsabilidades entre el experto del dominio, o *citizen developer* (véase la Sección 2.3), y el experto en ciencia de datos. El experto del dominio es el usuario final de un SWfMS especializado, es decir, un SWfMS que proporciona las funcionalidades adaptadas a su dominio de aplicación para definir y ejecutar DIW de forma visual. Por otro lado, el experto en ciencia de datos es quien define, reutiliza y configura los distintos componentes comunes, que son independientes de cualquier dominio de aplicación, para generar automáticamente un SWfMS especializado.

Por ejemplo, el experto en ciencia de datos registra en WORKGENESIS una serie de servicios y DIW destinados a realizar procesamiento de datos generales, como la eliminación de datos duplicados o la normalización de datos, entre otros. Después, particulariza y configura estos servicios para adecuarlos a los tipos de datos específicos y las características del dominio de aplicación de destino. Finalmente, el experto del dominio genera automáticamente un SWfMS especializado mediante WORKGENESIS, el cual se le deberá proporcionar al experto del dominio para definir y ejecutar sus propios DIW, sin necesidad de realizar ninguna configuración avanzada.

Para lograr este objetivo, se definen varios requisitos habituales en las herramientas *low-code* [39], que se enumeran a continuación:

- **REQ1:** *Permitir representar los conceptos y relaciones fundamentales que permiten la definición de las aplicaciones que están desarrollando.*

  En los dominios intensivos en datos, existe una base común compartida entre todas las DIA que operan en ellos. Esta base común tiene que ver con los procesos computacionales, servicios y fuentes de datos involucrados en el procesamiento, análisis y visualización de datos. Desde el punto de vista del experto en ciencia de datos, es necesario poder integrar y utilizar nuevos métodos y algoritmos, así como permitir su configuración y adaptación a dominios específicos. Además, ya que potencialmente estos elementos van a ser compartidos entre múltiples dominios, su reutilización es esencial para poder facilitar la generación de nuevas herramientas adaptadas al dominio. Por otro lado, desde el punto de vista del experto del dominio, es necesario disponer de todos los elementos necesarios para la creación y ejecución de DIW ya adaptados a su respectivo dominio, reduciendo al máximo la necesidad de configuración. Aquí juega un papel fundamental el uso de modelos y metamodelos como mecanismos para capturar, adaptar, y reutilizar el conocimiento entre distintos dominios y herramientas (véase la Sección 3.6.3).

- **REQ2:** *Proporcionar formularios y páginas de datos para crear, editar y visualizar la información que la aplicación en desarrollo gestionará.*

  Este requisito resulta especialmente importante para que los expertos de dominio puedan disponer de herramientas útiles y fáciles de usar. Así, es responsabilidad del experto en ciencia de datos tomar la decisión sobre qué tipos de datos podrán ser gestionados por la nueva herramienta, qué tipos de visualizadores de datos resultan más útiles para el dominio en el que serán aplicados, o incluso cómo será el aspecto y distribución de algunos de los elementos de la interfaz de usuario. De esta manera, el experto del dominio podrá utilizar una herramienta cuya interfaz y funcionalidades están listas para ser utilizadas en su dominio.

- **REQ3:** *Permitir mecanismos para la definición de los flujos de control y datos de la aplicación en desarrollo a través de mecanismos intuitivos de especificación de la lógica de negocio.*

  Para reducir la necesidad de poseer conocimientos en desarrollo de software y facilitar la definición de DIA, es necesario proporcionar mecanismos para definir DIW a alto nivel de abstracción (véase el Capítulo 3). De esta manera, tanto el experto en ciencia de datos como el experto del dominio pueden disponer de una interfaz gráfica visual, así como de otros mecanismos típicos de la programación visual, que les permiten reducir el tiempo y el esfuerzo a la hora de definir una DIA.

- **REQ4:** *Admitir la integración de servicios y fuentes de datos externos para reutilizar funcionalidades o consumir datos proporcionados por sistemas de terceros, por ejemplo, mediante API dedicadas.*

  Este requisito hace posible el incremento de la capacidad de cómputo de las DIA, ya que permite disponer de un gran número de fuentes de datos y servicios con capacidades de cómputo especializadas. Estas integraciones permiten al científico de datos conectarse con herramientas *cloud* especializadas, disponiendo de un mayor número de algoritmos y procesos para lograr su objetivo. A su vez, los expertos del dominio también se ven favorecidos por este modelo de computación, ya que no necesitan servidores con grandes unidades de cómputo o gran capacidad de memoria para ejecutar y operar sus DIA.

- **REQ5:** *Generar e implementar las aplicaciones previamente definidas.*

  Todo el conocimiento modelado y adaptado al dominio por el científico de datos es utilizado para generar SWfMS listos para usar por los expertos del dominio, pudiendo elegir además el tipo de aplicación a generar dependiendo de los requisitos técnicos de cada caso. Así, se podría generar una interfaz de usuario web o de escritorio, o aprovechar la computación *cloud* o la ejecución en local. Estas herramientas generadas les permiten a los expertos del dominio, a su vez, desarrollar sus propias DIA, que podrán ser ejecutadas sobre el propio SWfMS o bien, ser potencialmente generadas como aplicaciones independientes.

4.3. Workgenesis: arquitectura e implementación

Workgenesis es una solución, basada en el principio de separación de responsabilidades, que proporciona funcionalidades adaptadas a las necesidades de cada perfil de usuario. Por un lado, el experto en ciencia de datos tiene el conocimiento acerca de los algoritmos e implementación de los métodos de análisis de datos y extracción de conocimiento. También tiene la experiencia necesaria para la adaptación de estos métodos a distintos dominios y casos de uso. Por otro lado, el experto del dominio tiene el conocimiento del dominio de aplicación, y es quien realmente necesita disponer de herramientas intensivas en datos para obtener una ventaja competitiva con respecto a la competencia.

Internamente, Workgenesis está compuesto, a su vez, por distintas herramientas que permiten cubrir las distintas necesidades de estos perfiles de usuario (véase la Figura 4.1). En primer lugar, una herramienta para la definición y generación de SWfMS adaptados a dominios específicos intensivos en datos. En segundo lugar, un SWfMS altamente configurable para ser adaptado tanto a distintos dominios de aplicación como a diferentes entornos de ejecución. Los SWfMS resultantes permiten a los expertos del dominio la creación visual y ejecución de DIA. Para ello, estas herramientas proporcionan los elementos típicos de la programación visual. Ambas herramientas, junto con algunos vídeos de demostración, se pueden encontrar en el material suplementario de esta tesis [134].

4.3.1. Herramienta generadora de SWfMS específicos de dominio

El experto en ciencia de datos desempeña un papel fundamental en la modelización de la parte del dominio relacionada con los métodos y algoritmos de procesamiento y visualización de datos. Además, también es el responsable de la personalización y generación de los SWfMS especializados que usarán los expertos del dominio.

Por ello, Workgenesis proporciona una herramienta generadora de SWfMS especializados, orientada a los expertos en ciencia de datos, que permite registrar, configurar y reutilizar los elementos necesarios para la definición y ejecución de DIW en sus respectivos dominios de aplicación (REQ1, REQ2, REQ3 y REQ4), como son

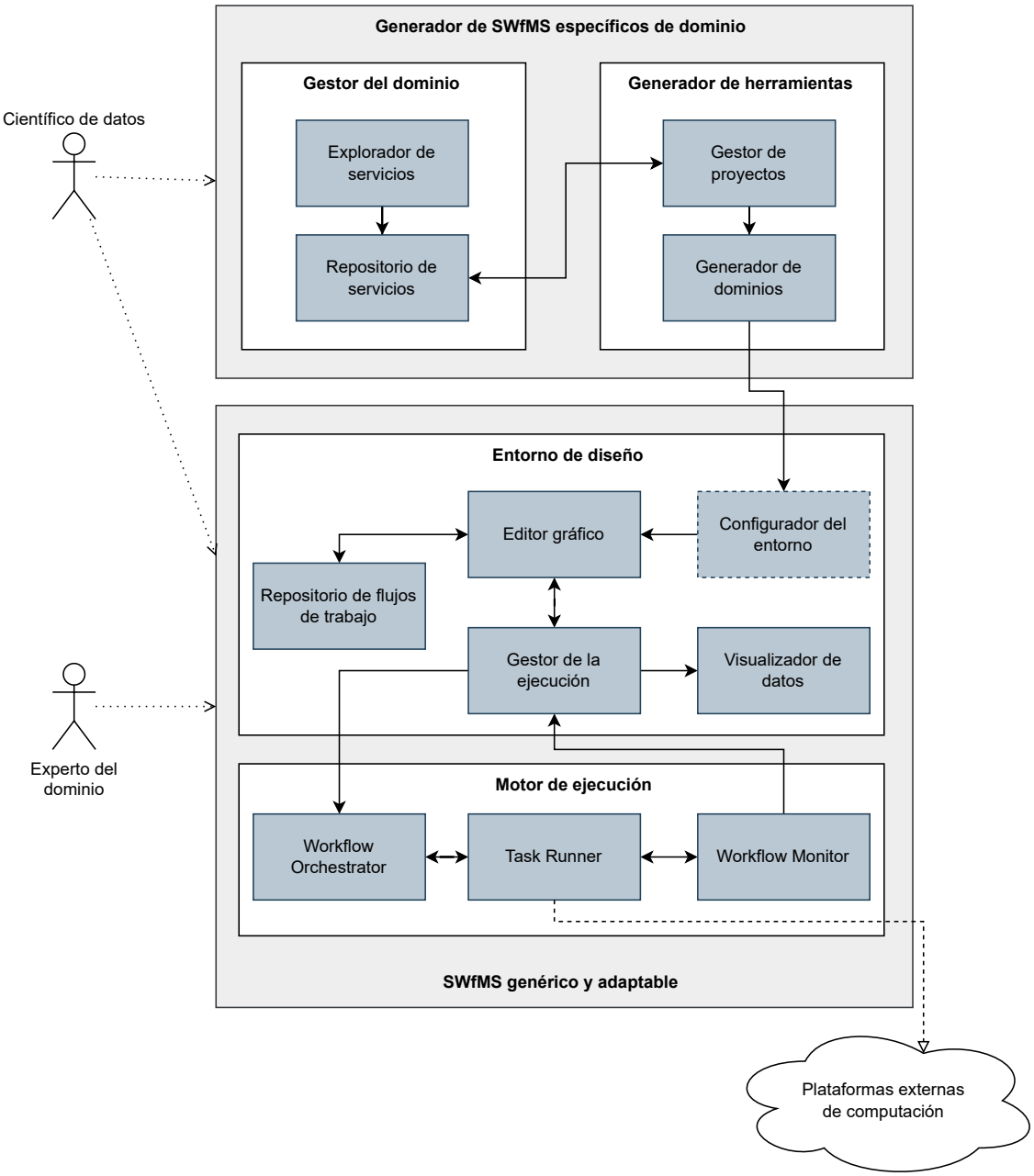


Figura 4.1: Arquitectura de WORKGENESIS.

las fuentes de datos (ficheros CSV, ARFF o bases de datos MySQL, entre otras) o las tareas computacionales (desde algoritmos de clasificación como J48 o SVM, o algoritmos de preprocesamiento, como la selección de atributos o la normalización de datos, en otros).

Para generar estos SWfMS específicos de dominio, el científico de datos no necesita poseer conocimientos en desarrollo de software, ya que dispone de funcionalidades

de alto nivel (REQ2) que permiten definir los aspectos relacionados con el apartado visual de la herramienta, con los elementos disponibles para la creación de DIW en el dominio de aplicación, y con el entorno de ejecución destino (REQ5). Así, además de reducir los costes y el *time-to-market* de nuevas herramientas intensivas en datos, también permite que el experto en ciencia de datos pueda centrarse en la lógica del dominio, sin preocuparse por los detalles técnicos propios del desarrollo de software.

Esta herramienta está formada, fundamentalmente, por dos componentes interrelacionados: un componente para capturar el conocimiento del dominio y gestión de la generación de SWfMS especializados, y por un SWfMS genérico y agnóstico a cualquier dominio, altamente configurable que permite definir y reutilizar la lógica del dominio mediante DIW y que es usado, a su vez, como base generar estos SWfMS especializados.

Gestión del conocimiento del dominio y generación de herramientas

Dado que existe una base de conocimiento común en los dominios intensivos en datos relacionado con el acceso, procesamiento y visualización de los datos, es esencial facilitar su registro, configuración y reutilización con el fin de generarSWfMS específicos de dominio. En la Figura 4.2, se muestra la interfaz de usuario de este componente junto con las distintas funcionalidades disponibles: la gestión de la lógica del dominio y la generación de herramientas.

Gestor del dominio: Para poder modelar y capturar todos los conceptos del dominio relacionados con la ciencia de datos, el gestor del dominio dispone de un *explorador de servicios* para la integración y configuración de los métodos y servicios, así como de un *repositorio de servicios* que permite su almacenamiento. Estos algoritmos deberán estar disponibles para ser registrados en la herramienta, conformando así la base fundamental de acciones de las DIA definidas en el SWfMS por el experto del dominio. Mediante una serie de formularios, el usuario puede configurar estos servicios para que puedan ser ejecutados según las necesidades del dominio donde serán utilizados. Workgenesis soporta algoritmos que pueden estar escritos en distintos lenguajes de programación o ser ofrecidos a través de servicios externos (véase la Sección 3.5.3). Esto es esencial tanto para implementar algoritmos propios (por ejemplo, un algoritmo C4.5 utilizado para generar árboles de decisión) como

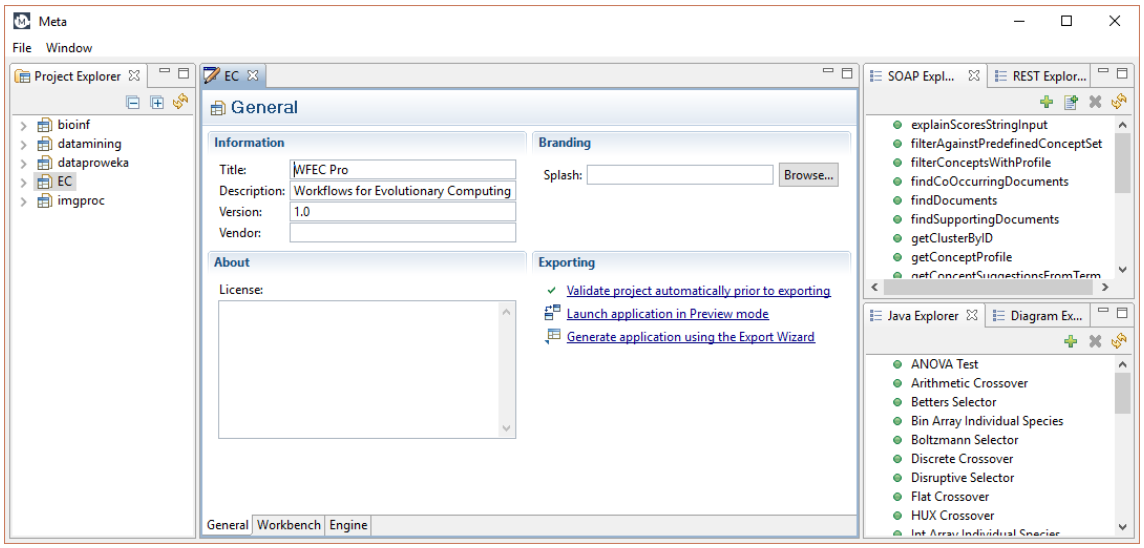


Figura 4.2: Configuración del dominio e integración de servicios.

para identificar e invocar servicios web de tipo REST o SOAP/WSDL. Otros tipos de conceptos que deben ser modelados están relacionados con la selección de los formatos de datos que serán permitidos gestionar, como CSV o ARFF.

Generador de herramientas: La reutilización del conocimiento es importante a la hora de reducir los tiempos de generación de nuevas herramientas, ya que es habitual que una serie de algoritmos y servicios básicos formen parte de distintos dominios de aplicación. A su vez, la capacidad de personalización visual y configuración del SWfMS dependiendo del entorno donde vaya a ser usado, también es un aspecto importante. Es por ello que este componente dispone de un *gestor de proyectos* con mecanismos para que el científico de datos pueda definir toda esta información. Cada proyecto contiene toda la configuración necesaria para que el *generador de dominios* pueda generar nuevos SWfMS especializados. En la Figura 4.2 se puede ver cómo el proyecto seleccionado se ha creado para la creación de un SWfMS para el dominio de la computación evolutiva.

Definición de la lógica del dominio con un SWfMS genérico y adaptable a cualquier dominio

Existe una gran cantidad de lógica de negocio que a menudo es común y que se puede utilizar en diversas DIA. Por ejemplo, un *data pipeline* que toma un conjunto

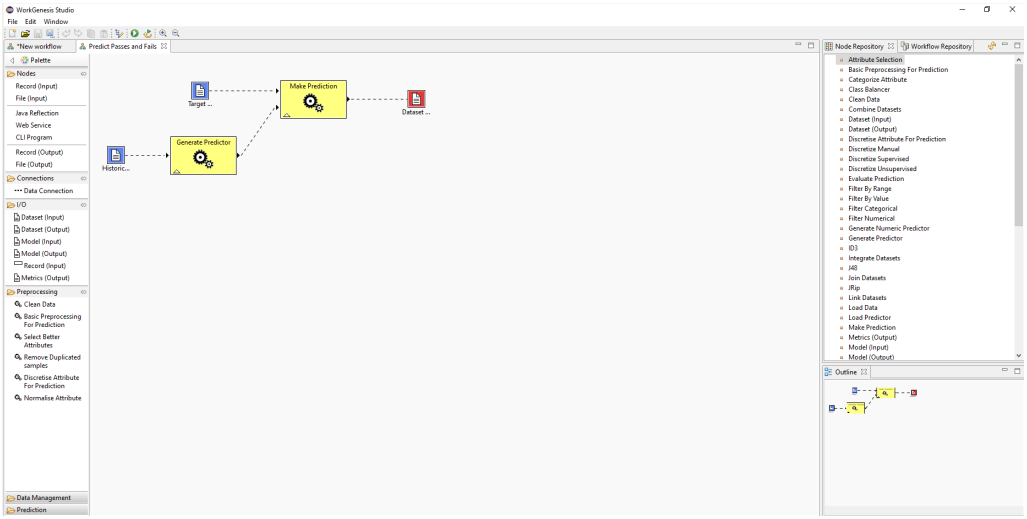


Figura 4.3: Definición de la lógica del dominio con un SWfMS genérico.

de datos de entrada, realiza una serie de procesos de limpieza para eliminar datos duplicados o nulos, y los normaliza en un determinado rango. O bien un *data pipeline* que toma una serie de conjuntos de datos, los unifica en uno solo y elimina los datos duplicados.

El experto en ciencia de datos tiene dos opciones para modelar esta lógica común, o bien puede hacerlo a través de un lenguaje de programación y crear el código que recoge los requisitos propios, o puede utilizar un SWfMS genérico y adaptable a cualquier dominio que le permita hacerlo mediante programación visual. Esta última opción implica el uso de un editor gráfico que incluya los algoritmos y servicios previamente definidos, facilitando así la definición de la lógica de negocio en forma de DIW (véase la Figura 4.3).

Este SWfMS contiene todos los algoritmos registrados con anterioridad por el propio experto en ciencia de datos y que está compuesta, a su vez, principalmente por dos componentes típicos de cualquier SWfMS: el entorno de diseño y el motor de ejecución.

Entorno de diseño: También conocido como *workbench*, permite tanto la definición visual del DIW, como la gestión y monitorización de su ejecución. Dentro del entorno de diseño se integran diversos módulos que proporcionan una variedad de funcionalidades: el editor gráfico, el repositorio de DIW y el gestor de la ejecución, entre otros.

- Editor gráfico: Es el elemento central que ofrece los elementos necesarios para representar visualmente un DIW. Para ello, se proporciona una paleta configurable que contiene los elementos que pueden ser usados para definir un DIW. Además, se dispone de un lienzo donde se pueden insertar y conectar los elementos añadidos desde la paleta, como tipos de datos, algoritmos y servicios previamente definidos, así como elementos genéricos que permiten añadir nuevos componentes. Además, la vista de propiedades proporciona información sobre las propiedades de cada elemento del DIW que se está definiendo, lo que proporciona una visión más precisa de su configuración.

- Repositorio de flujos de trabajo: Una vez diseñado el DIW, el repositorio permite que estos puedan ser almacenados y recuperados en cualquier momento. Los DIW contenidos en este repositorio pueden ser incluidos, a su vez, como elementos de otros nuevos DIW, permitiendo así la reutilización del conocimiento y la definición de nuevos DIW más complejos. Además, cualquier DIW disponible en el repositorio puede ser incluido como parte de los servicios definidos por el científico de datos. Por tanto, estos flujos de trabajo también podrán ser proporcionados en la herramienta generada posteriormente para su utilización por el experto del dominio.

- Gestor de la ejecución: Cuando el DIW está listo, el gestor de ejecución de flujos de trabajo permite la iniciar su ejecución y monitorizar el progreso sobre el propio editor gráfico. Las trazas generadas se recopilan y son mostradas visualmente para facilitar su entendimiento y conocer los posibles problemas que han podido surgir durante la ejecución de cada elemento del DIW. De esta manera, se puede consultar los resultados intermedios obtenidos, así como conocer las causas de los posibles problemas que hayan podido surgir.

- Visualizador de datos: Para mostrar gráficamente los datos procesados durante la ejecución, se proporciona un visualizador de datos que permite la representación de imágenes, vídeo, conjuntos de datos tabulares, así como cadenas de texto o números.

- Configurador del entorno: Ya que este SWfMS, agnóstico a cualquier dominio concreto y altamente personalizable, es utilizado como base para los SWfMS especializados a generar, este módulo es determinante para la generación poste-

rior de herramientas específicas de dominio. Este configurador es el encargado de la adaptación de un entorno de diseño genérico a un entorno de diseño adaptado al experto del dominio. Para ello, se ocultan todos los elementos genéricos que no son de interés para el experto del dominio, pero sí lo son para el científico de datos (como la invocación de servicios web o la ejecución de un programa Java), proporcionando únicamente los elementos específicos del dominio que el científico de datos haya decidido (como podría ser el proceso de limpieza de un conjunto de datos o la predicción del rendimiento de estudiantes), así como personalizando los elementos textuales y gráficos de la interfaz de usuario.

Motor de ejecución: La ejecución de cada elemento de un DIW es llevado a cabo por el motor de ejecución, que interpreta y ejecuta cada actividad computacional según el orden derivado de las conexiones existentes y del tipo de flujo de trabajo (ya sea dirigido por los datos o dirigido por la lógica de negocio). Para ello, este motor contiene todas las funcionalidades relacionadas con la invocación de servicios (tanto locales como remotos), la ejecución local de programas de computación y la gestión de datos. Para ello, se disponen de diferentes módulos específicos:

- Orquestador de flujos de trabajo: Un paso importante dentro del proceso de ejecución de un flujo de trabajo consiste en el análisis de la definición del DIW de nivel alto para generar la versión ejecutable de bajo nivel correspondiente (véase la Sección 3.5.3). Así, este módulo se encarga de coordinar y optimizar la ejecución teniendo en cuenta cada elemento del flujo de trabajo, así como las restricciones computacionales y la carga de trabajo del ejecutor de tareas.
- Ejecutor de tareas: Este módulo utiliza la versión ejecutable de una tarea para invocar los programas de computación o servicios correspondientes. Este ejecutor es flexible para integrar herramientas y plataformas externas (por ejemplo, para computación en *grid* o *cloud*), aumentando las capacidades de cómputo y reduciendo el tiempo de ejecución.
- Monitor de flujos de trabajo: Este módulo monitoriza la ejecución del flujo de trabajo y proporciona información relacionada con el tiempo, la memoria disponible y los resultados de cada ejecución.

Personalización y generación de herramientas específicas de dominio

Una vez que el dominio ha sido modelado y la lógica de negocio ha sido definida, un aspecto importante es la personalización, configuración y generación del SWfMS especializado. De esta manera, el científicos de datos puede proporcionar una experiencia de usuario mejorada al proporcionar una herramienta completamente adaptada a las necesidades del experto del dominio, y así también puede diferenciarse de otras herramientas generadas por WORKGENESIS. Para ello, existen distintos niveles de configuración que permiten personalizar por completo la herramienta generada a las necesidades de los expertos del dominio.

Metainformación: El experto en ciencia de datos puede incluir toda la información relacionada con la autoría de la herramienta a ser generada, como el título, la descripción, la versión y la licencia de uso. También es posible configurar aspectos como la imagen de marca y la pantalla inicial de carga. Esta información sirve para dar una identidad clara y reconocible a la herramienta resultante.

Configuración del entorno de diseño: Es posible configurar tanto con los elementos disponibles en la paleta, así como la configuración y organización de la misma del entorno de diseño. Los elementos incluidos en esta paleta serán los que estarán a disposición del experto del dominio para la creación de sus DIW. Por lo tanto, un mayor número de elementos amplía la capacidad de cubrir diversas necesidades del dominio en el que operará la herramienta. Esta capacidad de personalización no solo agiliza el proceso de diseño y definición de flujos de trabajo, sino que también optimiza la experiencia del usuario al proporcionar un entorno intuitivo y bien organizado.

Configuración del motor de ejecución: La capacidad de elegir el tipo de motor de ejecución más acorde a la plataforma de destino representa un aspecto importante en el proceso de generación de herramientas (véase las Sección 2.3). El experto en ciencia de datos podría elegir entre un motor local, diseñado para ejecutarse en la máquina del usuario, brindando un mayor control sobre el proceso de ejecución (*local engine*). Alternativamente, se puede elegir por un motor optimizado para

plataformas de ejecución basadas en Apache Hadoop[1], destinado a abordar cargas de trabajo de datos a gran escala (*big data engine*). Por último, existe la posibilidad de desplegar la herramienta en servicios en la nube, como Amazon Web Services[2] o Google Cloud Platform[3], aprovechando la escalabilidad y disponibilidad que ofrecen estos entornos (*cloud engine*). Esta capa de personalización está enfocada en las necesidades y preferencias del experto del dominio.

4.3.2. Herramienta generada para la creación y ejecución de DIW en dominios específicos

Las herramientas *no-code* generadas son SWfMS listos para operar en un dominio específico. De esta manera, los expertos de dominio no requieren ningún tipo de conocimiento en desarrollo de software para la creación y ejecución de DIA.

Definición de DIW específicos del dominio

El experto del dominio dispone de un entorno de trabajo completamente adaptado a su área de especialización, lo que simplifica y agiliza el proceso de definición de DIW. Este entorno proporciona una amplia variedad de elementos especializados, que abarcan desde elementos de entrada y salida, hasta algoritmos y DIW previamente configurados por el experto en ciencia de datos (véase el Capítulo 3). Esta variedad de recursos ya han sido previamente configurados para satisfacer los requisitos específicos del dominio en cuestión, permitiendo así a los expertos del dominio enfocarse únicamente en la definición de la lógica de negocio sin necesidad de conocer los detalles técnicos de bajo nivel.

Cada uno de estos elementos conforman los nodos del DIW (véase la Sección 3.5.2) que pueden ajustarse, en caso de que sea necesario, a través de una serie de puertos configurables. Estos puertos permiten refinar su configuración, garantizando así que la ejecución del DIW se alinee con los objetivos y requisitos del dominio. La posibilidad de anidar flujos de trabajo permite a los expertos del dominio aprovechar al máximo la reutilización y la combinación de componentes existentes.

[1] Apache Hadoop. https://hadoop.apache.org/ (consultado: 20/01/2024)
[2] Amazon Web Services. https://aws.amazon.com/ (consultado: 20/01/2024)
[3] Google Cloud Platform. https://cloud.google.com/ (consultado: 20/01/2024)

Ejecución y monitorización de DIW

Dado que la herramienta generada es un SWfMS completo (véase la Sección 2.1.4), el experto del dominio también tiene la capacidad de ejecutar sus aplicaciones intensivas en datos utilizando el propio entorno de diseño. Esto significa que los expertos del dominio pueden llevar a cabo la ejecución de sus DIA de manera directa. Las capacidades de ejecución están determinadas por el tipo de motor de ejecución elegido durante la generación de la herramienta.

Para conocer el estado de la ejecución, el motor de ejecución del SWfMS emite una serie de trazas que se integran de manera visual en el editor gráfico. Esto proporciona al experto del dominio una visión detallada del progreso y estado de cada nodo dentro del flujo de trabajo.

4.4. Integración de módulos basado en modelos

Workgenesis utiliza internamente SWEL (véase el Capítulo 3) como lenguaje para modelar el conocimiento específico del dominio y llevar a cabo su posterior ejecución. Los modelos de SWEL son transformados, además, a distintos PSM dependiendo del componente que los esté utilizando. Por un lado, es necesario serializar la información contenida en los modelos en formatos de intercambio de datos (ya sea XML o JSON) para poder almacenar y recuperar un DIW. Por otro lado, el motor de ejecución necesita crear un PSM que cumpla con los requisitos de la plataforma donde se realizará la ejecución.

4.4.1. Extensión de SWEL para representación visual

Debido a que SWEL es un lenguaje agnóstico a cualquier plataforma y compatible con distintas sintaxis concretas (véase la Sección 3.6.2), no define elementos específicos para la representación de DIW mediante notaciones gráficas. Por tanto, Workgenesis usa SWEL para que pueda ser utilizado como lenguaje de representación visual (Figura 4.4).

Este nuevo paquete permite recoger la información gráfica (*GraphicalElement*) relacionada con la posición y el tamaño de los elementos sobre un lienzo (*Bounds*)

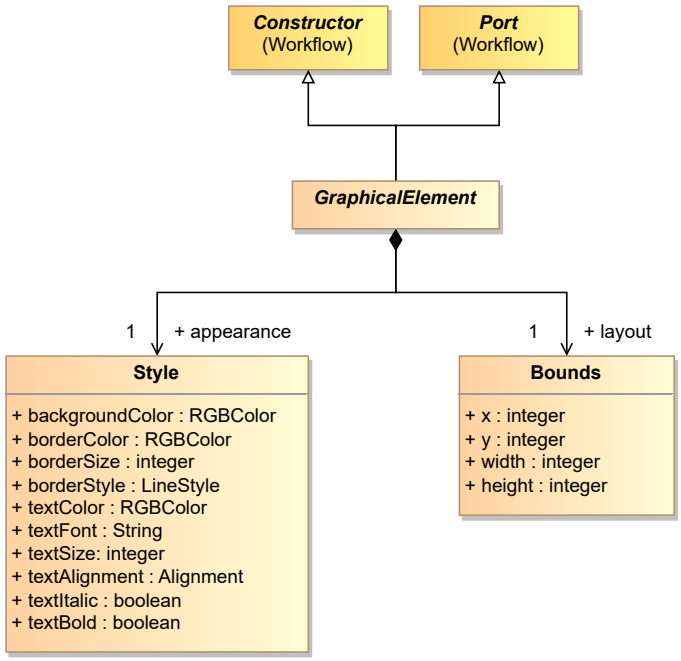


Figura 4.4: Metaclases para la representación gráfica de DIW con SWEL.

basada en su coordenadas x e y, así como en su ancho y en su alto. Además, es posible que cada elemento de un DIW pueda ser personalizado y representado visualmente de distintas maneras, dependiendo de las necesidades y preferencias del experto (*Style*). Los parámetros estéticos pueden ser personalizados son el texto del nodo (su color, tamaño, fuente...), así como el grosor y color del borde, o el color del fondo.

4.4.2. Serialización

Para poder almacenar y recuperar los DIW definidos con WORKGENESIS, se realiza un proceso de transformación de los modelos de SWEL hacia una sintaxis concreta particular (véase la Sección 2.2.1). Concretamente, esta serialización está basada en JSON y genera una serie de ficheros de configuración que contienen toda la información necesaria para su visualización gráfica, así como para su correspondiente ejecución.

Por un lado, se genera un fichero de configuración que es común a todos los elementos de un DIW y contiene toda la información gráfica del elemento, desde su posición y tamaño en el lienzo, hasta la definición de su aspecto visual en términos de colores,

tipo de letra o tamaño (véase el Código 4.1). Esta información es utilizada por WORKGENESIS de dos maneras distintas, dependiendo de si el DIW es el flujo de trabajo *padre*, o si se trata de un flujo de trabajo anidado dentro de otro. En el primer caso, esta información es parcialmente utilizada para configurar el editor de manera que el color de fondo de lienzo se corresponda con el color de fondo definido en este fichero. El resto de información es utilizada cuando el flujo de trabajo es un nodo de un flujo de trabajo superior. En este caso, toda la información es utilizada para representar visualmente el nodo y el texto que contiene. Además, los parámetros de configuración definidos por el usuario también son almacenados en este fichero. Dependiendo del tipo de todo, existirán unos parámetros u otros.

Código 4.1: Fragmento de un fichero de configuración común.

```
{ "name": "Categorize Attribute",
  "type": "DATA_DRIVEN_WORKFLOW",
  "nodes": [
    { "id": "695fdedb-7122-43e3-8f54-963ad7178bff" },
    { "id": "f9e2b99d-0713-40b2-93f2-19ae2ab72631" },
    { "id": "346614f4-7909-4e74-b660-214e97a184cd" },
    ...
  ], "connections": [
    {
      "type": "com.workgenesis.workbench.editor.model.DataConnection",
      "source": {
        "id": "695fdedb-7122-43e3-8f54-963ad7178bff",
        "port": "output"
      },
      "target": {
        "id": "dbaba2a5-c758-4320-821f-a69e4d1a33a1"
      },
      "bendpoints": [ ... ]
    }, ... ] }
```

Por otro lado, también se genera un fichero de configuración específico para los elementos *Workflow* (véase la Sección 3.5.3) y contiene la información sobre cómo cada elemento del DIW es conectado a otros elementos (véase el Código 4.2). Cada conexión (elemento *DirectedEdge*, Sección 3.5.2) contiene el identificador del elemento origen y del elemento de destino (elemento *Constructor*, Sección 3.5.3), así como su tipo, pudiendo ser de datos o de control (elementos *DataLine* y *ControlLine*, véase la Sección 3.5.2). Además, cierta información que utilizará WORKGENESIS para su representación en el editor gráfico también es incluida en este fichero, como por ejemplo los puntos de torsión que presentan las líneas de conexión.

Código 4.2: Fragmento de un fichero de configuración para elementos Workflow.

```
{ "id": "200ba508-2179-4134-8499-990e2da5e039",
  "name": "Dataset (Input)",
  ...,
  "type": "INPUT",
  "layout": { "x": 66, "y": 50, "width": 52, "height": 47 },
  "configuration": {
    "dataName": "Dataset",
    "allowedFormats": [ "arff", "csv" ]
  },
  "appearance": {
    "backgroundColor": { "red": 96, "green": 133, "blue": 238 },
   ...
    "textAlignment": "CENTER",
    "textItalic": false,
    "textBold": false
  },
  "ports": [ ... ] }
```

4.5. Estudio de caso: generación de un SWfMS para el procesamiento de imágenes

WORKGENESIS es una solución orientada a la generación automática de SWfMS con el objetivo de democratizar la definición y ejecución de DIA gracias a la disponibilidad de herramientas específicas de dominio a bajo coste y fáciles de utilizar (véase la Sección 1.1).

En esta sección, se plantea un estudio de caso en el dominio del procesamiento de imágenes para generar un SWfMS que proporcione las funcionalidades básicas para la manipulación y tratamiento de imágenes. En este dominio intensivo en datos, se gestiona y procesa una gran cantidad de información contenida en las imágenes, y se utilizan algoritmos intensivos en tareas de análisis y extracción de características. A continuación, se detalla cada una de las etapas que el científico de datos tiene que realizar:

1. **Registro de servicios y algoritmos:** La primera etapa implica el registro de los servicios y algoritmos que serán proporcionados a través de la herramienta generadora. En esta fase, el experto en ciencia de datos aporta su conocimiento sobre los recursos fundamentales para el dominio en cuestión.

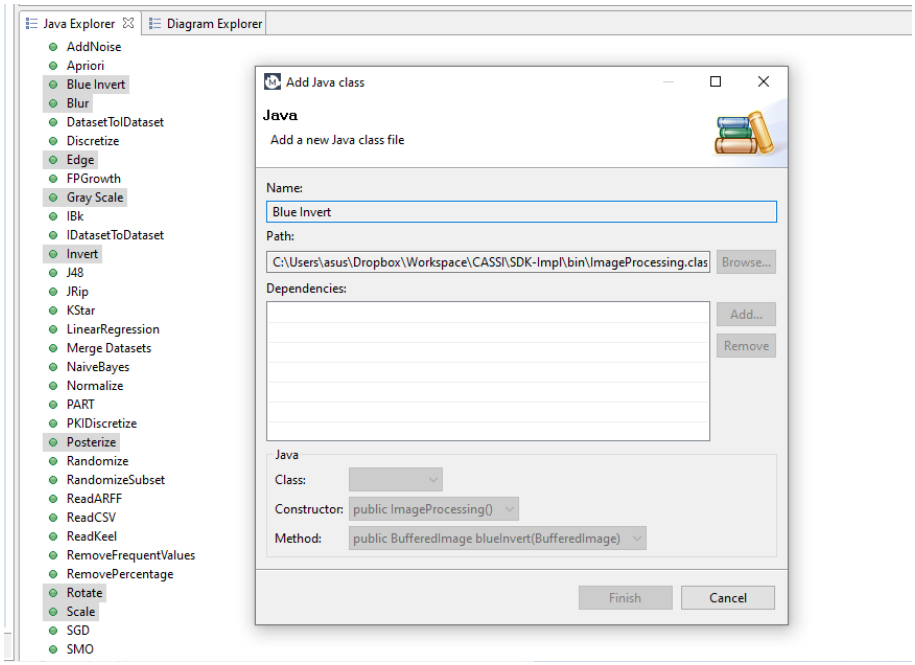


Figura 4.5: Registro de servicios en WORKGENESIS.

Estos elementos son la base sobre la cual se construirán las DIA por parte del experto del dominio, asegurando así que la herramienta resultante cuenta con las capacidades necesarias para abordar las tareas específicas del dominio.

En la Figura 4.5, se muestra el repositorio de servicios con todos los servicios que el científico de datos tiene registrados en su herramienta generadora (*AddNoise*, *FPGrowth*, *Merge Datasets*, entre otros muchos) donde se han resaltado los servicios relacionados con el dominio de estudio, como *Blue Invert*, *Blur* o *Edge*. Además, se muestra la ventana de registro del método *Blur*, que se trata de una tarea computacional en Java (elemento *JavaProcess*, Sección 3.5.3).

2. **Definición de la lógica del dominio:** La segunda fase se centra en la definición de la lógica del dominio utilizando el SWfMS agnóstico a cualquier dominio y altamente configurable (véase la Sección 4.3.1). Así, el experto en ciencia de datos puede modelar y capturar todos los aspectos relacionados con los procesos de análisis de datos del dominio en particular.

   En la Figura 4.6 se muestra un DIW, creado por el científico datos, que representa lógica del dominio, y que podrá ser utilizada directamente por el experto del dominio en la herramienta generada. En la parte izquierda se puede observar la paleta del SWfMS agnóstico con los elementos genéricos que utiliza

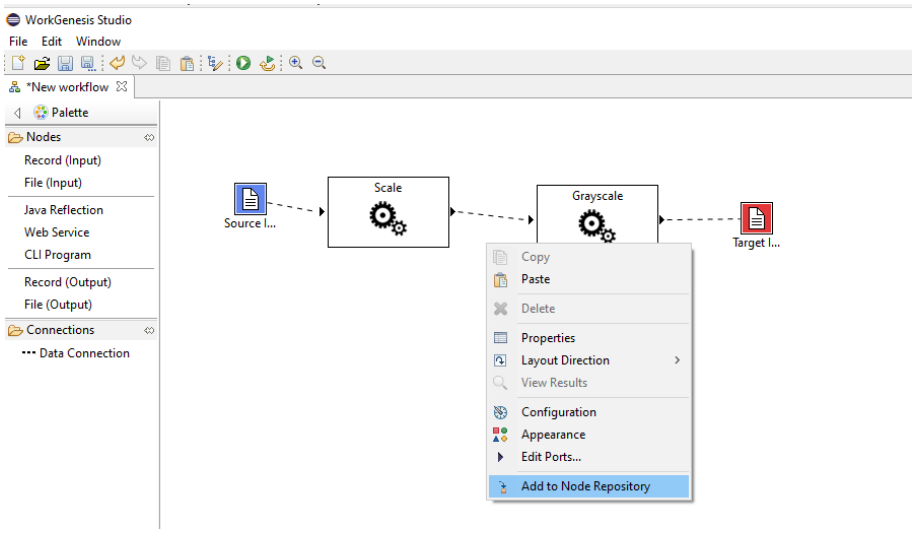


Figura 4.6: Definición de lógica de dominio en WORKGENESIS.

el científico de datos (elementos *ControlStructure*, *DataProvider* y *Activity*, Sección 3.5.3). Además, en el menú contextual se observan las distintas funcionalidades para su adaptación y personalización, donde se resalta la acción *Add to Node Repository* que se utiliza para incorporar el DIW diseñado al repositorio de servicios.

3. **Adaptación de servicios y algoritmos al dominio:** La tercera fase se enfoca en la adaptación de los servicios y algoritmos previamente registrados al contexto específico del dominio. Esta adaptación implica la configuración de parámetros, ajustes y personalizaciones que aseguran que estos recursos se integren en la lógica del dominio definida en la fase anterior. En la Figura 4.7 se muestra la ventana de configuración del aspecto gráfico de uno de los elementos, que se utilizará para su representación en el SWfMS destino.

4. **Configuración del proyecto:** En la cuarta etapa, se procede a la configuración detallada del proyecto que acabará generando la nueva herramienta, tal y como se explicó en la Sección 4.3.1. Cada proyecto tiene una relación directa con un SWfMS a generar, y cada servicio registrado puede formar parte de tantos proyectos como el científico de datos decida.

   En la Figura 4.8 se muestra, en la parte izquierda, el gestor de proyectos con todos los que el experto en ciencia de datos ha creado. En la parte central, se muestra la metainformación del proyecto seleccionado (en este caso, el del es-

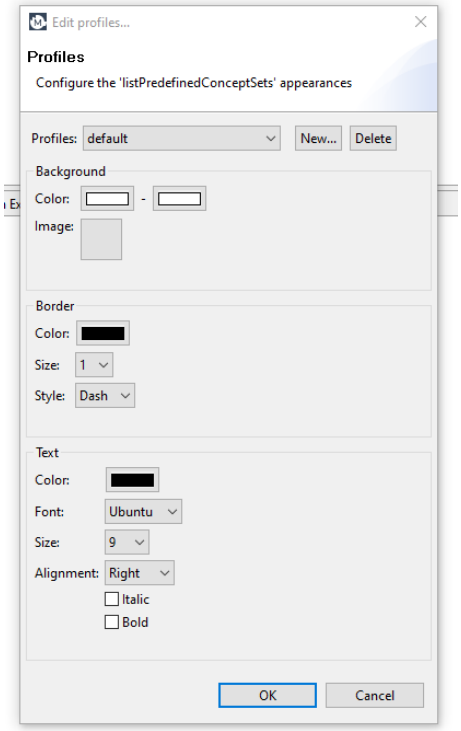


Figura 4.7: Configuración gráfica de un elemento en WORKGENESIS.

tudio de caso) donde se ha definido el nombre de la herramienta, su descripción o el tipo de licencia software (elemento *Project*, Sección 3.5.4).

5. **Generación automática de la herramienta:** Finalmente, en la quinta y última etapa, WORKGENESIS utiliza la configuración proporcionada para llevar a cabo la generación automática del SWfMS especializado. Esta fase representa la culminación de todo el proceso, donde la combinación de servicios, algoritmos, lógica del dominio y configuraciones se traduce en una herramienta funcional y adaptada al dominio en cuestión. La herramienta resultante es una solución adaptada y lista para ser utilizada por el experto del dominio, sin la necesidad de conocimientos en desarrollo de software. En la Figura 4.9 se muestra el SWfMS generado, donde se puede observar la paleta con los elementos proporcionados por el científicos de datos, los elementos de interfaz de usuario personalizados y el DIW que el experto del dominio ha definido, ejecutado y visualizado con WORKGENESIS.

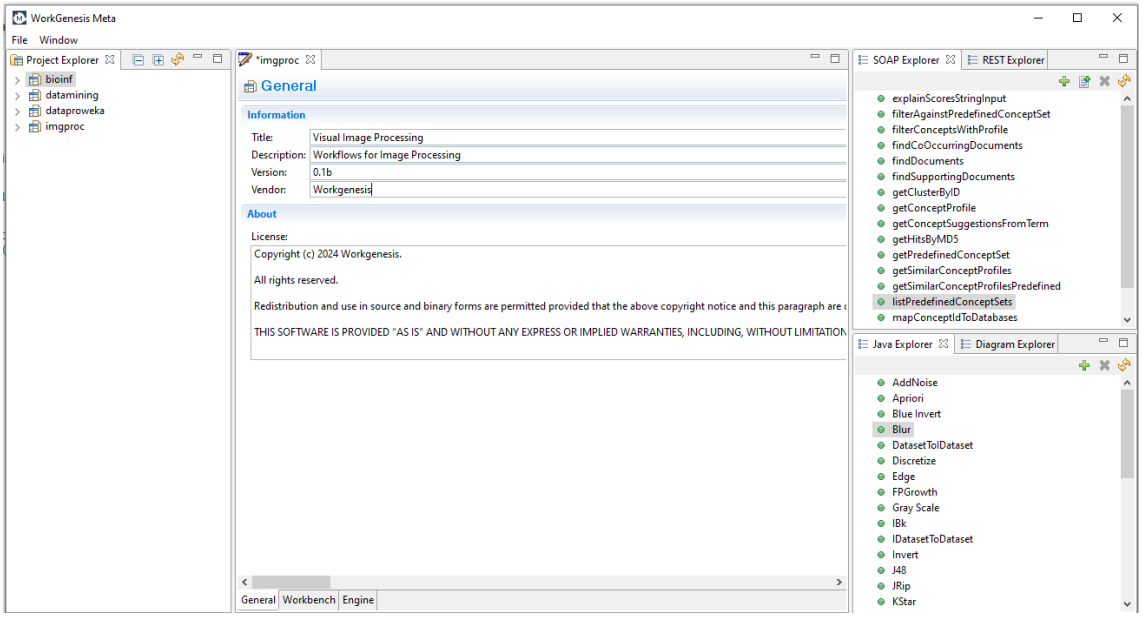


Figura 4.8: Configuración del proyecto en WORKGENESIS.

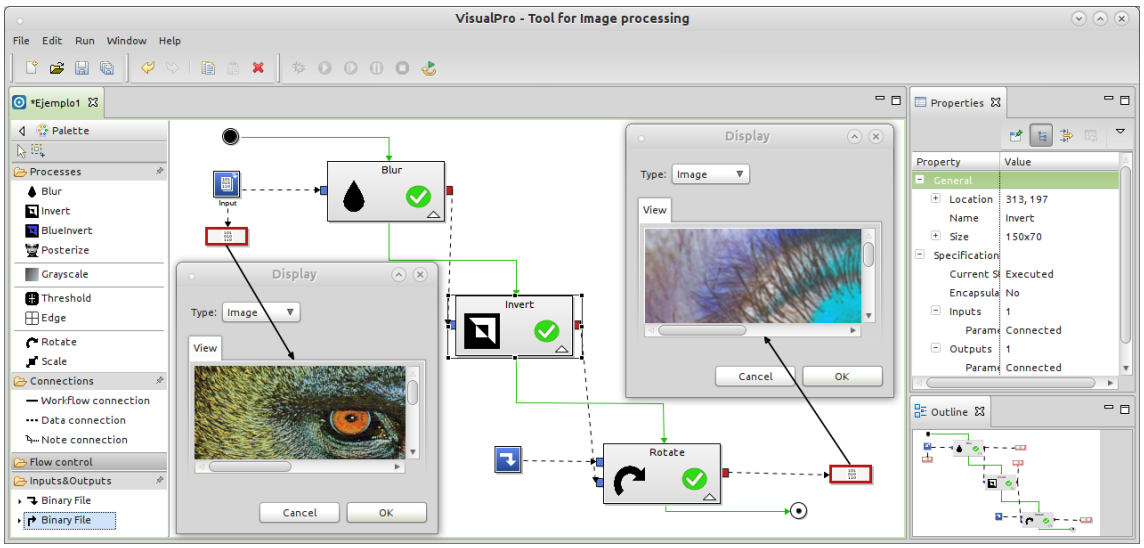


Figura 4.9: Herramienta generada con WORKGENESIS.

Capítulo 5

Herramienta *no-code* para la predicción automática del rendimiento académico de los estudiantes

"La información es el petróleo del siglo XXI,
y la analítica es el motor de combustión interna"
Peter Sondergaard

5.1. Introducción

El auge de nuevas tecnologías y la incorporación de entornos virtuales en las instituciones educativas ha generado cambios muy significativos tanto en la estructura de los sistemas educativos como en los paradigmas de aprendizaje [16]. Esta transformación ha generado una serie de beneficios, como un acceso más fácil al contenido de aprendizaje, una mayor flexibilidad para que los estudiantes planifiquen sus carreras, y nuevos enfoques novedosos para organizar los distintos recursos educativos a través de sistemas de gestión del aprendizaje o *learning management system* (LMS) [84].

Todas las interacciones que realizan los distintos actores educativos (estudiantes, profesores y gestores) dentro de estos sistemas de gestión de información generan

grandes cantidades de datos que, cuando se analizan adecuadamente, pueden proporcionar valioso conocimiento sobre el progreso de los estudiantes que permite facilitar y mejorar los planes educativos [119]. Como resultado, la minería de datos educativos o *educational data mining* (EDM) ha surgido como un campo activo de investigación enfocado en la extracción de conocimiento sobre el comportamiento de los estudiantes en todos los niveles educativos con el fin de mejorar tanto su experiencia de enseñanza como su rendimiento académico [141]. Aunque la EDM es aplicable en diversos dominios educativos, es en el ámbito de la ciencia y la ingeniería donde los educadores pueden obtener los máximos beneficios de estas tareas de análisis de datos, gracias a sus conocimientos previos sobre en informática y programación.

Uno de los grandes problemas que se aborda desde la perspectiva de la EDM es la predicción del rendimiento de los estudiantes [5]. A pesar de que la calidad de la educación no puede depender únicamente de indicadores de rendimiento [144], estos indicadores permiten obtener una visión general de los distintos problemas en el diseño y desarrollo de los cursos [3]. La predicción temprana del rendimiento académico es particularmente crucial, ya que afecta a todos los niveles educativos y tiene profundas consecuencias tanto para los estudiantes como para el propio sistema educativo [90]. Por tanto, identificar a estudiantes en situación de riesgo en etapas tempranas del aprendizaje puede prevenir el abandono, además de permitir a profesores y gestores diseñar planes efectivos de intervención de manera oportuna.

Tal y como señalan Tayebi *et al.* [145], el abandono académico es un problema especialmente crítico en programas de ingeniería en muchos países, con importantes consecuencias para las perspectivas de empleabilidad. Para abordar este tipo de desafíos, los investigadores de EDM han explorado distintos tipos de datos educativos que se pueden extraer de LMS y de las actividades de aprendizaje tradicionales, así como han investigado varias técnicas para preparar datos para el uso de aprendizaje automático o ML efectivo y diferentes algoritmos para el análisis del rendimiento académico [5].

Sin embargo, lo más común es que estas tareas queden fuera del área de conocimiento de profesores y gestores, a pesar de ser el público objetivo desde el punto de vista de EDM que debe desempeñar un papel clave en la interpretación de los resultados. En un esfuerzo por involucrar a todos los interesados, las herramientas de EDM deberían reducir la complejidad del propio proceso de EDM. No obstante, la mayoría

de las herramientas proporcionan un conjunto limitado y poco extensible de funcionalidades que solo permiten a profesores y gestores configurar y ejecutar algoritmos predefinidos. Esto supone que todavía se requiere la experiencia de un especialista en EDM para poder adaptar o crear procesos analíticos educativos personalizados y adaptados a distintos tipos de problemas.

En los últimos años, existe una tendencia creciente que consiste en facilitar la definición y aplicación efectiva de ML a *citizen developers* mediante la utilización de plataformas *low-code* (véase la Sección 2.3). En el campo de EDM, una plataforma *low-code* se puede concebir como una herramienta que permite realizar operaciones específicas a alto nivel sobre datos educacionales, mientras abstrae de todas la complejidad subyacente de las tareas de procesamiento y análisis de datos. Esto permite, por ejemplo, a los *citizen developers* manipular y configurar visualmente las operaciones que caen dentro de sus respectivas áreas de conocimiento.

En este contexto, se presenta LoCoAPe como una herramienta *no-code* que proporciona, a los profesores y gestores educativos, métodos accesibles de EDM centrados en la predicción del rendimiento de los estudiantes. LoCoAPe es una herramienta generada por Workgenesis (véase el Capítulo 4), que proporciona los elementos necesarios para el acceso y preprocesamiento de datos, métodos de ML y mecanismos para la visualización de información. Estos elementos pueden tanto combinarse y conectarse para crear nuevos DIW, lo que permite reutilización del conocimiento, como adaptarse para cubrir nuevos requisitos.

Gracias a la separación de responsabilidades que proporciona LoCoAPe (véase el Capítulo 4), permite a los educadores utilizar las vistas de la herramienta que mejor se alinee con sus conocimientos técnicos. Es importante tener en cuenta que se utiliza el término *educador* para identificar indistintamente a un profesor, gestor educativo o cualquier otro profesional de la educación que pueda utilizar la herramienta. En la vista básica, LoCoAPe proporciona procesos de EDM ya preparados para su funcionamiento sobre tres tareas de predicción del rendimiento académico: 1) para predecir si un estudiante aprobará o no; 2) para pronosticar la calificación anticipada del estudiante; y 3) para identificar a los estudiantes en riesgo a través de la predicción temprana del rendimiento. Además, LoCoAPe ofrece vistas avanzadas que permiten a los usuarios realizar tareas de preprocesamiento de datos y seleccionar entre una variedad de algoritmos de ML para personalizar y adaptar sus DIW

para situaciones específicas o ajustarlos para conjuntos de datos con características complejas. Cabe señalar que tras todas las operaciones de minería de datos proporcionadas por LoCoAPe se encuentra el uso de Weka [45] como proveedor de algoritmos, dada su popularidad en la comunidad de EDM [5].

Para validar las diversas aplicaciones de LoCoAPe, se han realizado distintos experimentos en forma de estudios de caso con datos de estudiantes del mundo real procedentes de varios niveles educativos. Además, se demuestra cómo LoCoAPe se puede configurar fácilmente utilizando las vistas avanzadas para llevar a cabo tareas de preprocesamiento y entrenar diferentes algoritmos predictivos.

5.2. Minería de datos educacionales

EDM es un campo activo de investigación que investiga el uso de técnicas estadísticas, de ML y de minería de datos para analizar datos educativos derivados de entornos de aprendizaje [120]. A medida que las tendencias educativas continúan evolucionando, EDM también ha avanzado para afrontar la creciente disponibilidad de datos, la aparición de modalidades de aprendizaje alternativas y la coexistencia de diversas metodologías de evaluación [120]. Dado el amplio abanico de roles involucrados en el sistema educativo, los métodos de EDM deben adaptarse a diversos objetivos y públicos [10].

La Figura 5.1 ilustra el proceso general de EDM basado en la minería de datos [119]. El primer paso implica la recopilación de datos sobre estudiantes, cursos o tareas del entorno educativo. El paso de preprocesamiento subsiguiente implica transformar los datos para que puedan ser procesados por el algoritmo de aprendizaje automático. Este paso suele incluir la anonimización de datos para descartar información sensible del alumnado, así como la integración de datos de múltiples fuentes. Una vez preparados los datos, se aplican uno o más algoritmos para extraer patrones interesantes que describan los datos o permitan predicciones futuras, por ejemplo, utilizando técnicas de ML. La clasificación y la regresión se emplean comúnmente para tareas predictivas. El paso final implica interpretar los resultados, para lo cual son necesarias métricas de rendimiento y la visualización de los modelos de decisión. A esto le sigue la toma de decisiones informada y la formulación de planes de intervención. El proceso de EDM es iterativo y puede requerir ajustes basados en

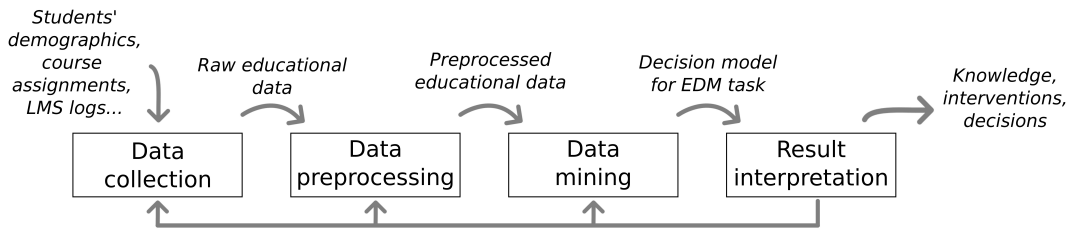


Figura 5.1: Secuencia de pasos y elementos de datos requeridos para resolver tareas de EDM, adaptado de [119].

resultados intermedios, como modificar los datos de entrada o ajustar los parámetros del algoritmo.

5.2.1. Predicción del rendimiento académico

Una tarea particular de EDM es la predicción del rendimiento académico de los estudiantes, que puede abordarse utilizando los pasos anteriores. Al considerar factores como las interacciones de los estudiantes con plataformas de aprendizaje en línea, información sociodemográfica y asignaciones de cursos, el objetivo es predecir si el estudiante aprobará o suspenderá un curso [5]. Con este fin, se emplean a menudo clasificadores como los árboles de decisión y las redes neuronales [5]. Alternativamente, si el objetivo es predecir la calificación específica que obtendrá un estudiante, la tarea predictiva puede formularse como un problema de regresión [90]. En ambos casos, el algoritmo de ML debe entrenarse utilizando datos históricos de estudiantes de quienes se conocen sus resultados académicos (conjunto de entrenamiento). Posteriormente, el modelo predictivo puede utilizarse para predecir el rendimiento académico de nuevos estudiantes, cuyos resultados no son conocidos por el algoritmo (conjunto de prueba). Siguiendo esta metodología, estudios recientes ya se han aplicado en MOOCs [113] y entornos de aprendizaje mixto [112].

Es importante destacar que la predicción temprana del rendimiento académico ha recibido una atención significativa en los últimos años [90]. Esto permite detectar a los estudiantes en riesgo lo antes posible, mitigar las tasas de abandono y proporcionarles planes de intervención personalizados. Desde la perspectiva de EDM, la predicción temprana implica trabajar con menos datos disponibles para fines de entrenamiento. Este enfoque también se ha adoptado para identificar a los estudian-

tes en riesgo en el contexto de la ingeniería en informática [65] y de electrónica de primer año [99].

5.2.2. Herramientas EDM

Las herramientas de EDM se han convertido en un recurso valioso para los educadores, proporcionándoles acceso a diversos métodos de investigación propuestos dentro del campo [140]. Estas herramientas facilitan diferentes tipos de análisis de datos educativos, abarcando uno o más pasos del flujo de trabajo de EDM descrito en la Sección 5.2.1, donde algunas de ellas se pueden personalizar para adaptarlas a sistemas de gestión del aprendizaje específicos. A continuación, se presentan varias herramientas de EDM junto con sus respectivas limitaciones.

DSS-PSP: Es un software de apoyo a la decisión diseñado para predecir si los estudiantes aprobarán o suspenderán el examen final del curso [89]. Esta herramienta ejecuta clasificadores de Weka, incluyendo una técnica multiclase especializada propuesta por los propios autores. Sin embargo, carece de soporte para la composición de flujos de trabajo.

MDM Tool: Es un *framework* de minería de datos que permite a los educadores llevar a cabo diversas tareas de EDM [94]. Integrada en Moodle, MDM Tool proporciona componentes para selección y visualización de datos, tareas de preprocesamiento básicas, así como tres tipos de análisis: agrupamiento, minería de reglas de asociación y clasificación.

UBUMonitor Tool: Es una aplicación de escritorio que se comunica con una plataforma Moodle a través de servicios web, facilitando la extracción y análisis de datos de interacción de los estudiantes [125]. La herramienta se enfoca principalmente en el análisis exploratorio de datos a través de técnicas de agrupamiento e incorpora un módulo con capacidades de monitorización para la detección temprana de abandono.

SPEET : Es una aplicación web orientada a gestores educativos desde una perspectiva de tutoría [111]. SPEET tiene como objetivo analizar los registros académicos de los estudiantes para tareas descriptivas y predicción de rendimiento en el contexto de grados de ingeniería.

Estas herramientas ofrecen un conjunto predefinido de funcionalidades, diferentes y no compatibles entre sí, que permiten a los educadores abordar tareas de EDM específicas. Sin embargo, aunque estas herramientas proporcionan una funcionalidad valiosa, cada una tiene sus propias limitaciones. Cabe destacar que, a menudo, carecen de flexibilidad en términos de composición de flujos de trabajo (por ejemplo, para reformular algunas tareas de EDM) y es posible que no sean completamente adaptables a las diversas necesidades de educadores y gestores en el campo de la educación.

5.2.3. Herramientas *low-code* y su aplicación en EDM

Uno de los problemas de las herramientas anteriores es que el rango y la diversidad de tareas, así como de algoritmos disponibles, solo pueden ampliarse con la participación de los desarrolladores de la herramienta. Esta limitación restringe el acceso fácil a los recursos de EDM y dificulta su aplicabilidad en diferentes contextos educativos. Es aquí donde las herramientas *low-code* contribuyen a la democratización de la ciencia de datos, en particular del ML, gracias al desarrollo rápido y fácil de aplicaciones personalizadas intensivas en datos (véase la Sección 2.3). Estas herramientas presentan varias características fundamentales [39] como una interfaz gráfica de usuario que simplifica la especificación y configuración de operaciones específicas del dominio, mecanismos de extensibilidad para la integración de fuentes de datos y servicios externos, o una amplia cantidad de operaciones que permiten definir la lógica de negocio mediante DIW personalizados.

Aunque algunas herramientas de EDM cumplen parcialmente con estas características, es posible que no estén diseñadas explícitamente para la predicción del rendimiento académico. Por ejemplo, E-learning Web Mining [53] es una aplicación web que emplea un sistema de plantillas para definir y configurar varias tareas de EDM. Cada plantilla especifica los datos de entrada y los algoritmos basados en agrupamiento para tareas como la creación de perfiles de estudiantes o la predicción del

rendimiento académico [53]. Aunque los educadores pueden registrar nuevas plantillas, este mecanismo de extensión es menos flexible en comparación con la definición visual de DIW donde cada operación se puede configurar en tiempo de ejecución.

Otra herramienta, desarrollada por Bouhineau *et al.* [13], se basa en la composición de flujos de trabajo, combinando tres tipos de bloques: datos, algoritmos y visualización. Sin embargo, su propósito principal es recopilar y analizar registros del sistema educativo, careciendo de capacidades predictivas relacionadas con el rendimiento académico.

En resumen, aunque algunas herramientas de EDM presentan características típicas de herramientas *low-code*, su funcionamiento puede no estar específicamente adaptado a la predicción del rendimiento académico, una tarea con múltiples variantes y factores condicionantes. Esto destaca la necesidad de una herramienta *low-code* que permita a los educadores crear flujos de trabajo personalizables y abordar de manera efectiva tareas predictivas dentro del ámbito de su entorno educativo.

5.3. LoCoAPe: arquitectura e implementación

La Figura 5.2 ilustra los elementos principales que componen la arquitectura general de LoCoAPe, junto con sus relaciones. Los dos componentes fundamentales se detallan a continuación: el entorno de diseño (Sección 5.3.1) y el motor de ejecución de flujos de trabajo (Sección 5.3.2). LoCoAPe recibe como entrada las fuentes de datos que contienen información sobre los estudiantes, pruebas de evaluación y los resultados registrados hasta el momento. Genera tablas con predicciones de estudiantes, métricas de rendimiento y visualizaciones de modelos de decisión como salida.

LoCoAPe ha sido generada mediante Workgenesis (véase el Capítulo 4) como una aplicación independiente de Eclipse para aprovechar su portabilidad, su amplia comunidad de soporte y su ecosistema. Más específicamente, se ha utilizado Eclipse GEF para implementar el editor basado en diagramas, Datapro4j para manipular datos de diversas fuentes heterogéneas y Weka para ejecutar los algoritmos subyacentes (preprocesamiento, clasificación y regresión) y calcular las métricas de rendimiento.

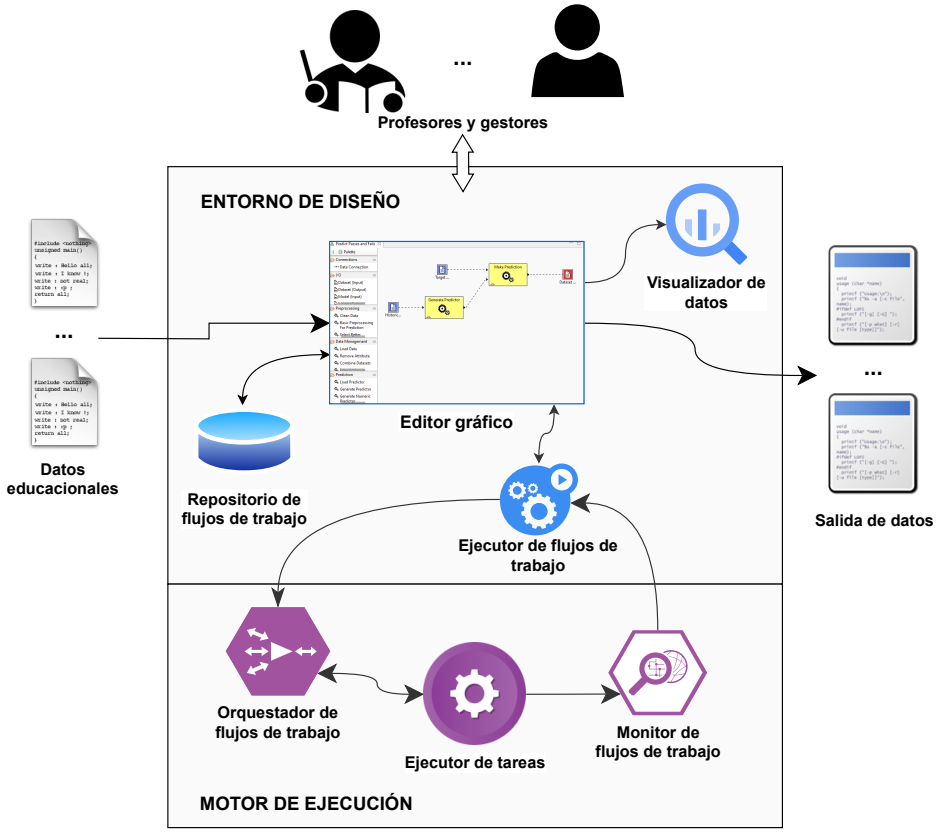


Figura 5.2: Visión general de la arquitectura de LoCoAPe.

5.3.1. Entorno de diseño de LoCoAPe

El entorno de diseño de LoCoAPe está orientado al diseño que permite definir visualmente DIW y monitorizar su ejecución (véase la Sección 4.3.1). Consta de cuatro componentes principales (véase la Figura 5.2): el editor gráfico, el repositorio de flujos de trabajo, el visualizador de datos y el ejecutor de flujos de trabajo.

Editor gráfico: Este componente ha sido diseñado para adaptarse a diferentes roles de usuario, como profesores con conocimiento en datos o gestores, con el fin de facilitar la definición y configuración de tareas EDM. Así, la paleta contiene proveedores de datos (véase la Sección 3.5.3), tareas EDM pre-configuradas, y métodos de preprocesamiento y aprendizaje automático configurables. Estos elementos se utilizan para definir los DIW. Cabe destacar que, si bien algunos elementos de la paleta son comunes a todas las vistas de usuario, otros solo aparecen en una vista particu-

lar según el conocimiento técnico del usuario. El término *vista de usuario* se utiliza para referirse a cada nivel de uso, como se detalla en la Sección 5.4.

Repositorio de flujos de trabajo: Este componente facilita el almacenamiento y recuperación de los nuevos DIW definidos. De esta manera, es posible ampliar las funcionalidades de LoCoAPe y cubrir nuevos requisitos.

Visualizador de datos: Este componente proporciona a los usuarios la capacidad de examinar la información producida después de la ejecución de un flujo de trabajo. Dependiendo de la naturaleza de los datos, el visualizador presenta información ya sea a través de una visualización textual (números o contenido basado en texto como reglas) o mediante un visor gráfico que admite la representación de datos tabulares y modelos de decisión basados en árboles.

Ejecutor de flujos de trabajo: Este componente es responsable de comunicarse con el motor de ejecución de flujos de trabajo para iniciar una nueva ejecución, así como para pausar o detener una ejecución en curso. También admite la capacidad de reiniciar una ejecución al eliminar todos los resultados existentes. Durante la ejecución, este ejecutor captura una serie de trazas que proporcionan información en tiempo real sobre el estado de cada operación. Esto permite a los usuarios monitorizar el progreso de cada tarea contenida e invocada por el flujo de trabajo en tiempo de ejecución. El estado *No ejecutado* se utiliza para indicar cuando una tarea está pendiente de ejecución, mientras que *Ejecutándose* se emplea cuando una tarea se está ejecutando actualmente por el motor. Si una tarea se ha ejecutado con éxito y su salida puede ser visualizada, se asigna el estado *Finalizado*. Finalmente, el estado *Error* denota que una tarea no se ha ejecutado y proporciona información sobre el error que ocurrió.

5.3.2. Motor de ejecución de LoCoAPe

Ya que LoCoAPe es una herramienta generada con Workgenesis, esta comparte algunos de sus componentes. En este caso, el motor de ejecución de Workgenesis permite la ejecución de DIW de cualquier dominio intensivo en datos. Esto significa

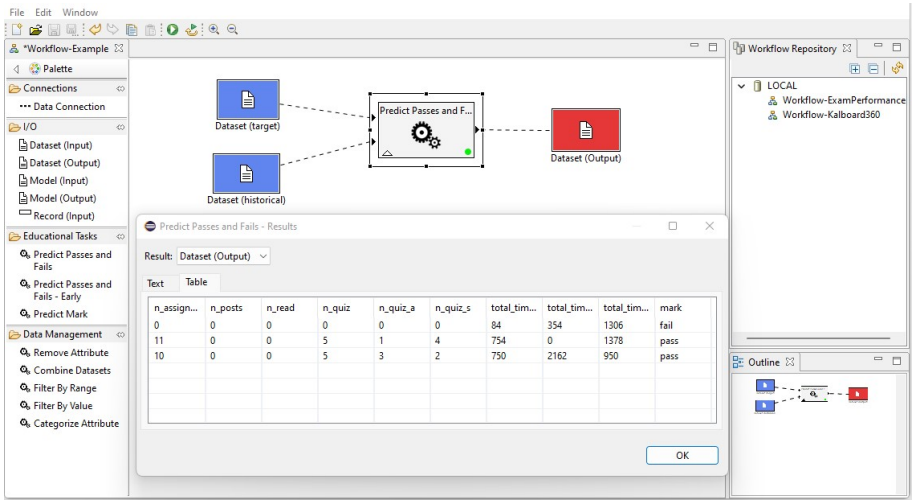


Figura 5.3: Captura de pantalla de la vista simplificada del educador con un DIW predefinido para predecir el rendimiento del estudiante.

que las funcionalidades de ejecución ofrecidas por LoCoAPe son idénticas a las ya explicadas en la Sección 4.3.1.

5.4. Descripción de las vistas de LoCoAPe

Esta sección describe el conocimiento definido y proporcionado por LoCoAPe (véase el Capítulo 3), categorizado en tres vistas de usuario: la *vista simplificada del educador* (Sección 5.4.1), la *vista avanzada del educador* (Sección 5.4.2), y la *vista del experto en datos* (Sección 5.4.3). Estas vistas están organizadas en capas, de manera que una capa superior oculta las complejidades de la capa inferior. De la misma manera, en las capas inferiores se puede modificar la definición y estructura de las operaciones en las capas superiores. Visualmente, cada vista tiene un color de fondo diferente que le sirve de referencia al usuario.

5.4.1. Vista simplificada del educador

En esta vista, las tareas de EDM de alto nivel están configuradas con valores predeterminados. El objetivo es que los educadores simplemente tengan que conectar y configurar los conjuntos de datos que contienen la información de los estudiantes

para acceder rápidamente a su correspondiente análisis de datos. Actualmente, se definen tres tareas de EDM que son definidas como DIW:

- Predecir aprobados y suspensos (*Predict Passes and Fails*): Esta tarea de EDM implica la utilización de dos conjuntos de datos de entrada. El primer conjunto de datos (conjunto de datos históricos) debe contener datos sobre el rendimiento de los estudiantes, incluidos sus atributos y resultado (aprobado o suspenso). Este conjunto de datos se utiliza para entrenar el clasificador, que en este caso es un árbol de decisión. El segundo conjunto de datos, conocido como conjunto de datos objetivo, debe contener los mismos atributos pero para un conjunto diferente de estudiantes. El predictor utilizará este conjunto de datos para determinar el resultado, que se generará como un nuevo conjunto de datos separado al conjunto de datos de entrada. Este conjunto de datos de salida también se puede visualizar dentro de la propia herramienta seleccionando la opción *View results*. La herramienta mostrará el conjunto de datos con el resultado de la predicción (aprobado o suspenso) en formato textual o tabular, tal como se ilustra en la Figura 5.3.

- Predecir aprobados y suspensos de manera temprana (*Predict Passes and Fails (early)*): Esta tarea realiza una operación similar a la anterior pero con la particularidad de que se establecen dos momentos de evaluación para la predicción, como puede ser al principio y a la mitad de un curso. Por tanto, requiere de cuatro conjuntos de datos de entrada: dos conjuntos de datos históricos que contienen la información de los estudiantes en los dos momentos de evaluación y su resultado final, y dos conjuntos de datos objetivo que contienen a otros estudiantes sin sus resultados. Al igual que en la tarea anterior, la salida de cada predictor estará disponible en su propio conjunto de datos de salida, cada uno mostrando el resultado de la predicción (aprobado o suspendido) para cada momento del curso.

- Predecir notas (*Predict Mark*): El objetivo es predecir la puntuación numérica esperada para cada estudiante. La tarea requiere dos conjuntos de datos de entrada, un conjunto de datos históricos y un conjunto de datos objetivo, que funcionan de manera similar a los de la tarea anterior. Sin embargo, ahora el resultado en el conjunto de datos históricos es un valor numérico. La herra-

mienta ajusta un modelo de regresión lineal con los datos históricos y lo aplica para calcular la puntuación de los estudiantes en el conjunto de datos objetivo.

Además de estas tareas de EDM, al educador se le permite realizar algunos procesos básicos de gestión de datos, como eliminar atributos, unir conjuntos de datos por filas o columnas, filtrar filas por un umbral dado o discretizar atributos numéricos. Estas acciones están disponibles directamente desde el propio editor gráfico de la herramienta.

5.4.2. Vista avanzada del educador

En la vista anterior, los educadores pueden ejecutar tareas de alto nivel de EDM, sin la posibilidad de modificar su estructura o configurar sus parámetros internos. Sin embargo, permitir estas acciones podría resultar útil para los educadores que tienen algún conocimiento de EDM y que deseen personalizar estas tareas para adaptarlas a sus datos y algoritmos específicos. En esta vista avanzada, los educadores pueden acceder a nuevas operaciones para expandir los DIW de EDM predeterminados con operaciones adicionales que son más o menos comunes. Incluso también pueden combinarse según las necesidades de cada caso. La Figura 5.4 muestra la composición interna del DIW de alto nivel *Predict passes and fails*. Las operaciones disponibles sobre los predictores en esta vista son:

- Cargar predictor (*Load Predictor*): Esta operación permite al educador cargar un modelo ya entrenado desde un archivo y utilizarlo como entrada para realizar las predicciones (*Make Prediction*). El modelo de decisión se proporciona como salida.
- Generar predictor (*Generate Predictor*): Este DIW utiliza un conjunto de datos históricos como entrada para generar un clasificador con el algoritmo J48 predeterminado de Weka. La salida es un modelo de decisión listo para hacer predicciones para un conjunto de datos objetivo. La opción *View results* proporciona acceso al árbol de decisiones tanto en formato de texto como gráfico.
- Generar predictor numérico (*Generate Numeric Predictor*): Este DIW utiliza un modelo de regresión en lugar de un clasificador. El modelo resultante puede

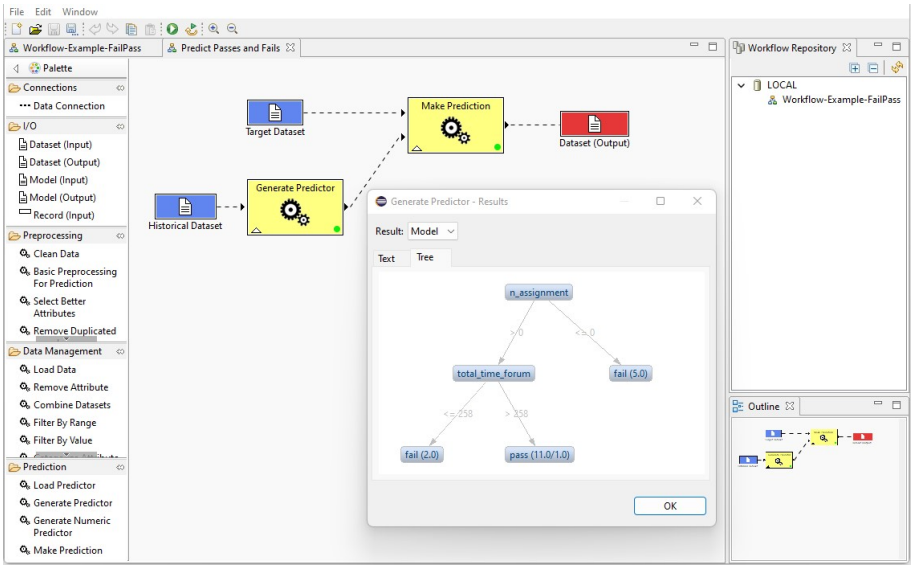


Figura 5.4: Captura de pantalla de la vista avanzada del educador con un árbol de decisiones.

predecir un valor numérico como una combinación lineal de un atributo del estudiante. El modelo ajustado se puede inspeccionar seleccionando la opción *View results*.

- Realizar predicción (*Make Prediction*): Este DIW utiliza un conjunto de datos objetivo y un predictor (modelo de clasificación o regresión) como entradas para estimar el atributo de resultado para cada estudiante. El resultado se añade en una nueva columna.

Esta vista también ofrece DIW adicionales para tareas de preprocesamiento típicas de EDM [118]:

- Eliminar datos duplicaos (*Remove Duplicated Samples*): Este DIW analiza el conjunto de datos de entrada y elimina cualquier fila que esté duplicada.
- Normalizar atributos (*Normalize Attribute*): Este DIW transforma todos los atributos numéricos de un conjunto de datos para que sus valores estén dentro de un rango predefinido (por defecto: [0,1]).
- Seleccionar los mejores atributos (*Select Better Attributes*): Este DIW ejecuta un algoritmo de selección de atributos para seleccionar únicamente aquellos

atributos que son más importantes para el predictor del conjunto de datos de entrada. Internamente, está configurado para usar el método de búsqueda *best-first* implementado por Weka.

- Discretizar un atributo (*Discretize Attribute*): Este DIW divide los valores de los atributos numéricos en el número especificado de *bins*, con el rango para cada *bin* determinado por el método de Fayyad & Irani disponible en Weka.
- Limpiar datos (*Clean Data*): Este DIW ejecuta dos acciones de manera secuencial tras cargar un conjunto de datos. Primero, se eliminan las filas duplicadas y, luego, se reemplazan los valores faltantes con un valor preconfigurado.
- Procesamiento básico para predicción (*Basic Preprocessing for Prediction*): Este DIW encapsula tres acciones en secuencia: 1) la discretización de atributos numéricos, 2) un algoritmo de selección de atributos para reducir el número de atributos, y 3) un equilibrador de clases para asignar pesos a las muestras en función de la distribución de clases. Estas acciones suelen ayudar a mejorar el rendimiento de los clasificadores en situaciones particulares, como datos de alta dimensión y que presenten desequilibrio.

Para modificar el comportamiento de un DIW, es decir, cambiar las operaciones o su configuración, los educadores deben abrirlo y modificarlo a través de la vista de experto en datos.

5.4.3. Vista de experto en datos

Los DIW creados en la vista del educador se pueden personalizar para cambiar su comportamiento predeterminado. Con este fin, la vista de experto en datos está diseñada para usuarios con experiencia en ML. En esta vista, las operaciones son atómicas y no se pueden descomponer más, pero se pueden ajustar sus parámetros seleccionando la opción *Properties*. La Figura 5.5 proporciona un ejemplo de los parámetros disponibles en la implementación que hace Weka del algoritmo J48.

Además, la paleta ofrece un mayor número de operaciones que permiten la composición de DIW de EDM complejos. Todas las operaciones utilizadas para predefinir DIW en las vistas del educador están incluidas en esta vista también, además de

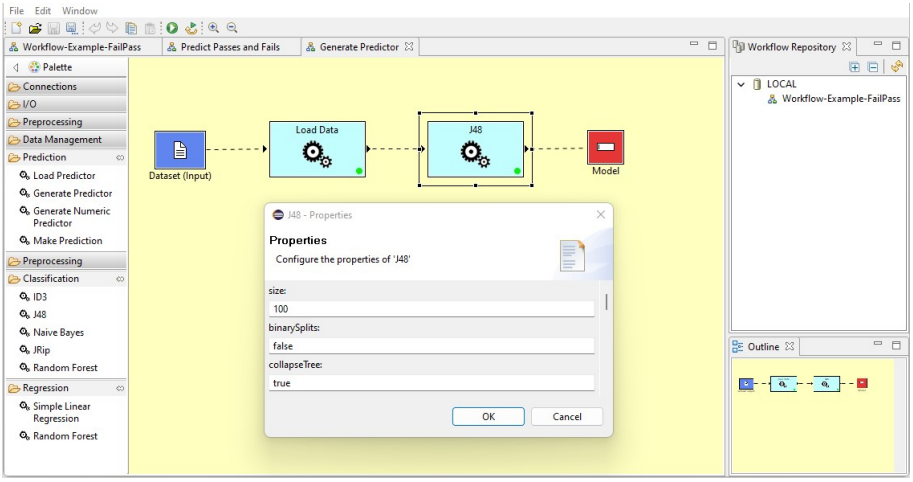


Figura 5.5: Captura de pantalla de la vista de experto en datos con la ventana de configuración de parámetros.

disponer de operaciones adicionales para que los usuarios las reemplacen por otras equivalentes. Por ejemplo, se puede cambiar el tipo de predictor de un árbol de decisión (J48) a un clasificador basado en reglas (JRip). Los usuarios también pueden agregar nuevas operaciones al DIW, como agregar un paso de preprocesamiento en uno de los flujos de preprocesamiento definidos en la Sección 5.4.2. Además, los usuarios también pueden crear sus propios DIW desde cero. A continuación, se describen los tres tipos diferentes de operaciones:

- Operaciones de preprocesamiento (*Preprocessing*): Son operaciones de preprocesamiento de bajo nivel, que corresponden al concepto de "filtro" en Weka. Entre otras cosas, estas operaciones ofrecen opciones para discretizar datos utilizando enfoques supervisados o no supervisados, eliminar atributos o muestras, filtrar muestras por atributos categóricos o numéricos, o manejar valores faltantes mediante su reemplazo o eliminación.

- Operaciones de clasificación (*Classification*): Son cinco algoritmos utilizados con frecuencia en EDM, que incluyen dos algoritmos (ID3 y J48) que generan árboles de decisión, JRip que construye un clasificador basado en reglas interpretable, el enfoque probabilístico (Naive Bayes), y un método de conjunto como Random Forest.

- Operaciones de regresión (*Regression*): Para predictores numéricos, se pueden aplicar varios métodos de regresión, incluyendo regresión lineal o regresión de

vector de soporte (SVR, por sus siglas en inglés), que es una adaptación de las máquinas de soporte vectorial a problemas de regresión.

La Figura 5.6 resume el contenido de cada capa, agrupado por funcionalidades. Esta estructura en capas facilita la localización de aquellas operaciones que son similares, como reemplazar un algoritmo de clasificación entre los disponibles, o que son compatibles, como concatenar todas las operaciones de preprocesamiento.

5.5. Experimentación

5.5.1. Metodología

LoCoAPe es una herramienta *no-code* que permite a educadores y científicos de datos predecir el rendimiento académico de los estudiantes sin requerir experiencia en técnicas de minería de datos ni ML. Esta herramienta también permite a los científicos de datos personalizar cómo se aplican los conjuntos de datos académicos y los algoritmos para obtener soluciones más precisas. Con este fin, LoCoAPe debe satisfacer tres *requisitos de diseño* que se validarán a través de sus respectivos estudios de caso. El desarrollo de estos estudios de caso (véase de las Secciones 5.5.2 a 5.5.4) no solo valida la precisión de las operaciones de LoCoAPe, sino que también nos permite extraer conclusiones sobre su practicidad, que se discuten en la Sección 5.5.5.

Nótese que estos estudios de caso demuestran cómo Workgenesis (véase el Capítulo 4) se aplica en un dominio donde sus usuarios son considerados *citizen developers*.

Requisitos de diseño

De acuerdo con los DIW de EDM descritos en la Sección 5.4.1, los requisitos de diseño se formulan sobre la base de las tareas de predicción de rendimiento de nivel superior que LoCoAPe pretende satisfacer:

REQ1. Predecir calificación: LoCoAPe debe permitir a los educadores predecir la calificación de un estudiante utilizando técnicas de regresión univariante y mul-

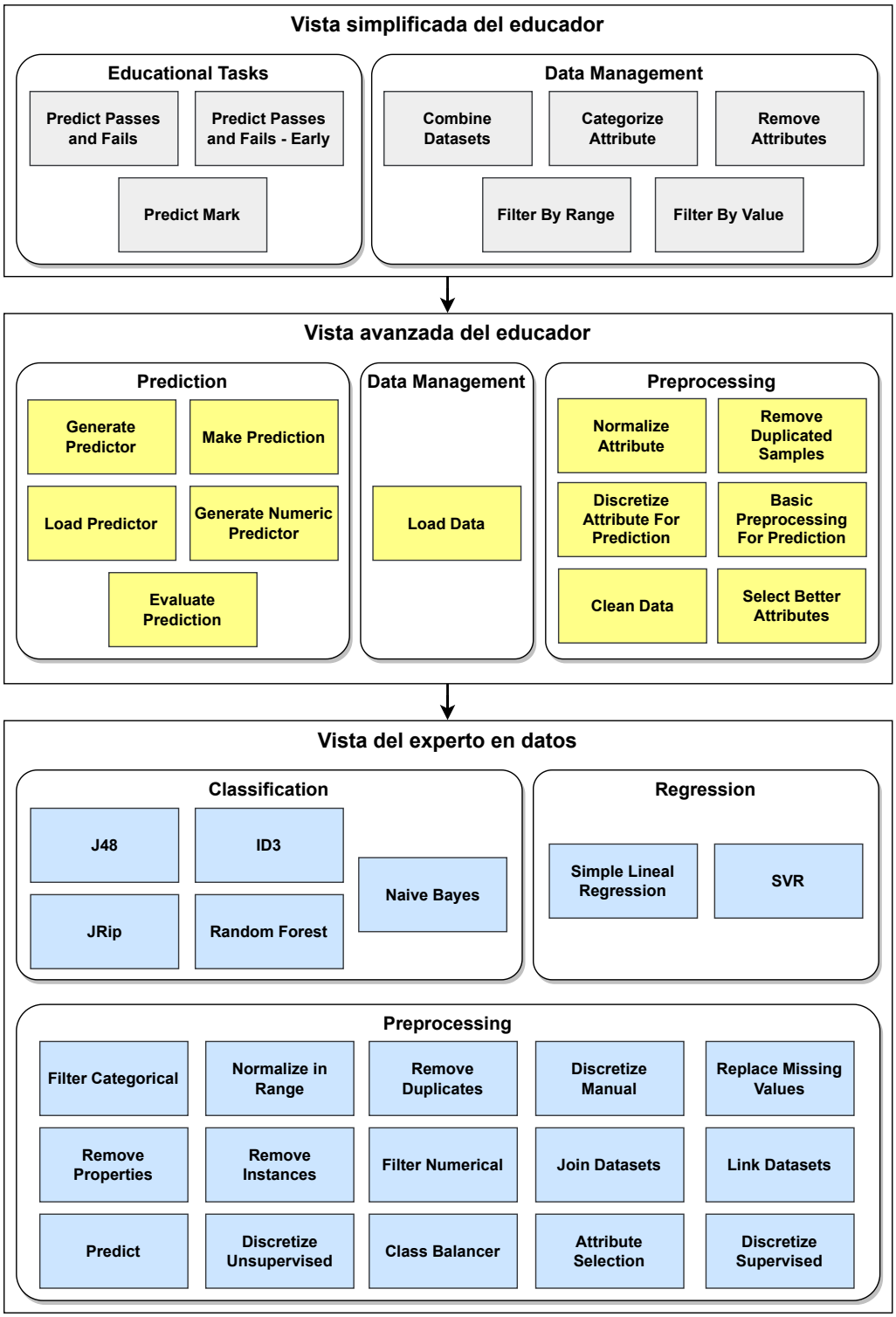


Figura 5.6: Capas de vistas de usuario de LoCoAPe.

tivariante. Los modelos de regresión deben depender únicamente de características numéricas que representen el historial académico e información personal del estu-

diante.

REQ2. Predecir aprobados y suspensos: LoCoAPe debe permitir a los educadores predecir si un estudiante aprobará o suspenderá utilizando técnicas de clasificación interpretables. Los clasificadores deben construirse seleccionando un conjunto específico de atributos numéricos y categóricos recopilados de diversas fuentes educativas.

REQ3. Predicción temprana de aprobados o suspensos: LoCoAPe debe permitir a los educadores predecir el rendimiento del estudiante en dos momentos distintos del curso, facilitando la identificación temprana de estudiantes en riesgo. Los modelos de clasificación deben entrenarse utilizando una combinación de atributos numéricos y categóricos, teniendo en cuenta un posible desequilibrio en el conjunto de datos.

Procedimiento experimental

Para cada requisito, se presenta un estudio de caso utilizando una de las vistas de LoCoAPe. Para la vista simplificada del educador, se muestra la sencillez de uso al no necesitar realizar ningún tipo de configuración. Cuando se utiliza la vista avanzada del educador, el procedimiento involucra al educador cargando el conjunto de datos, realizando un preprocesamiento básico (si es necesario) y entrenando el algoritmo predeterminado para construir un modelo de decisión.

Finalmente, en la vista del experto de datos, se destaca el potencial y flexibilidad de LoCoAPe al construir un DIW desde cero. Se aplican pasos de preprocesamiento específicos, y se cambia el algoritmo de entrenamiento en función de los requisitos de diseño y las características del conjunto de datos.

En el material suplementario [134] de esta tesis se puede encontrar esta experimentación con una mayor extensión. Para cada requisito, se puede observar el uso de cada una de las vistas, junto con una serie de vídeos demostrativos paso a paso.

Conjuntos de datos

LoCoAPe maneja los conjuntos de datos históricos y objetivos como ficheros independientes. Para garantizar mejores condiciones experimentales, se ha generado una partición con el 70 % de las muestras para el conjunto de datos histórico. El 30 % restante de las muestras representa el conjunto de datos objetivo, el que se ha eliminado el resultado del estudiante (clase o calificación). El proceso de partición también mezcla las muestras y garantiza que la proporción original de muestras de cada clase se mantenga en cada partición. Aunque estos son procedimientos experimentales comunes, no son obligatorios para ejecutar los DIW de EDM. Además, se ha requerido alguna conversión de formato para convertir los conjuntos de datos originales a CSV, aunque la mayoría de los conjuntos de datos seleccionados se proporcionan en dicho formato. Los conjuntos de datos cubren una amplia gama de niveles educativos y materias, demostrando la flexibilidad de LoCoAPe en el análisis de datos de distinta naturaleza. Además, también se incluye un conjunto de datos específico para una disciplina de ingeniería (Programación) y otro (Kalboard 360) que recoge las interacciones de estudiantes con diversos antecedentes (el 20 % etiquetado como *IT*) en un LMS. La Tabla 5.1 recopila los conjuntos de datos utilizados en los diferentes estudios de caso, como se describe a continuación:

- *Secundaria (Portugués)*: Este conjunto de datos contiene 33 atributos que representan las calificaciones de los estudiantes, datos demográficos y factores sociales. La información está disponible para dos grados diferentes: matemáticas (395 estudiantes) y lengua portuguesa (649). Consideramos el conjunto de datos para el curso de lengua portuguesa porque tiene más muestras. El conjunto de datos tiene atributos categóricos (4), binarios (13) y numéricos (16), sin valores faltantes.

- *Rendimiento en exámenes*: Este conjunto de datos proporciona información demográfica y académica sobre 1000 estudiantes. La información demográfica está representada por cinco atributos categóricos, como género o raza. Luego, tres atributos numéricos contienen las puntuaciones de habilidades matemáticas, de lectura y de escritura, todos ellos expresados en el rango [0,100].

- *Kalboard 360*: Este conjunto de datos recopila información de 480 estudiantes utilizando un LMS llamado Kalboard 360. Los 16 atributos contienen in-

formación demográfica (género, nacionalidad, etc.), datos académicos (etapa educativa, nivel de grado, etc.) y atributos de comportamiento (participación en actividades y uso de la plataforma en línea). El atributo objetivo representa el rendimiento del estudiante expresado en tres niveles: bajo (calificaciones de 0 a 69), medio (de 70 a 89) y alto (de 90 a 100).

- *Open University Learning Analytics Dataset (OULAD)*: Este conjunto de datos proporciona archivos separados con información sobre estudiantes, cursos y sus evaluaciones, e interacciones con el entorno de aprendizaje virtual. Con estos archivos, se pueden evaluar varios aspectos, incluida la predicción del rendimiento del estudiante dividido en cuatro niveles: abandono, suspenso, aprobado o distinción. El experimento se centrará en un curso en particular con 383 estudiantes, combinando su información personal con sus puntuaciones de evaluación (en total, 13 atributos).

- *Programación*: Este conjunto de datos contiene información sobre 6 tareas y 3 calificaciones de exámenes de 44 estudiantes durante un curso de programación. También incluye dos indicadores de rendimiento de los estudiantes: el resultado del examen de admisión (ACT) y el promedio de calificaciones (GPA) en la escuela secundaria. La puntuación final del curso varía de 14 a 100. Este conjunto de datos contiene algunos valores faltantes, siendo ACT y GPA los dos atributos más afectados.

- *Curso de ciencias*: Este conjunto de datos recopila las calificaciones de tareas, cuestionarios y exámenes de 486 estudiantes universitarios de un curso de ciencias. La calificación final se expresa en un rango [0-100], donde el rendimiento de los estudiantes con menos de 60 se considera "débil". Se les identifica como estudiantes en riesgo, que representan el 4,3 % de la muestra. El conjunto de datos se ha utilizado en un estudio de predicción temprana, realizando predicciones al 20 % y 50 % del curso [71].

	REQ1	**REQ2**	**REQ3**
Conjuntos de datos	Rendimiento en exámenes Secundaria (Portugués)	Kalboard 360 OULAD	Programación Curso de ciencias
Número de instancias	1000 649	480 383	44 486
Número de características	8 33	17 15	17 10
Tipos de características	CAT,NUM CAT,NUM	CAT,NUM,NOM CAT,NUM,NOM	CAT,NUM CAT,NUM
Referencias	[137] [27]	[7] [83]	[44] [71]

***CAT**: Categórico; **NUM**: Numérico; **NOM**: Nominal.*

Tabla 5.1: Conjuntos de datos utilizados en la experimentación con LoCoAPe.

Algoritmos

Para REQ1 (predecir la calificación), se aplican dos algoritmos de regresión: regresión lineal simple en la vista avanzada del educador, y SVR en la vista de experto en datos. Para REQ2, los algoritmos aplicados son J48, la opción predeterminada en la vista avanzada del educador, y JRip al utilizar la vista de experto en datos. Para REQ3, se mantiene J48 en la vista avanzada del educador, mientras que se utiliza Naive Bayes para la vista de experto en datos.

5.5.2. REQ1. Predecir la calificación

Para este requisito de diseño, se ha configurado un estudio de caso que utiliza la vista simplificada del educador. El objetivo de este experimento es predecir la calificación de los estudiantes sin necesidad de realizar ninguna configuración ni adaptación de los elementos proporcionados. En la Figura 5.3 se muestra que en el DIW definido solo se necesita utilizar y conectar directamente los elementos disponibles en la paleta del entorno de diseño.

Dos estudios de caso más, uno con la vista avanzada del educador y otro con la vista del experto en datos, se pueden encontrar en el material suplementario [134].

5.5.3. REQ2. Predecir aprobados y suspensos

Para validar este requisito, se formula un estudio de caso para la vista avanzada del educador. El conjunto de datos de *Kalboard 360* no tiene valores faltantes ni identificadores únicos que deban eliminarse. Por lo tanto, no se requiere preprocesamiento especial y se puede ejecutar el DIW predefinido (consulte la Figura 5.4). Cabe destacar que el atributo objetivo no es binario, como se espera (aprobado o suspenso). Sin embargo, este conjunto de datos se puede utilizar ya que el predictor interno (árbol de decisión J48) requiere un conjunto discreto de valores. Esto permite demostrar otro escenario de uso con una representación de datos ligeramente diferente, donde la decisión final se puede categorizar como rendimiento "bajo" (L), "medio" (M) o "alto" (H). Una vez completado el DIW, se puede inspeccionar el árbol de decisión por el cual se ha determinado el nivel de rendimiento para los estudiantes en el conjunto de datos objetivo. La Figura 5.7 muestra la ejecución exitosa del DIW, junto con los resultados. El árbol de decisiones consta de 36 nodos y 19 ramas, y los siguientes atributos desempeñan un papel crucial en el proceso de toma de decisiones: número de días de ausencia, respuestas a la encuesta de los padres, género del estudiante y frecuencia de uso de recursos en línea. Los nodos hoja del árbol muestran la decisión asignada (L, M o H) junto con el número de estudiantes que respaldan esa decisión. En caso de que se presenten dos números, el segundo número representa el recuento de estudiantes clasificados incorrectamente. De esta manera, el educador obtiene información sobre el rendimiento del clasificador. En este caso, el árbol clasifica correctamente a 273 estudiantes (el 81.25 % de las muestras en el conjunto de datos histórico) y clasifica erróneamente a 63 estudiantes (el 18.75 %).

Un estudio de caso con la vista del experto en datos se puede encontrar en el material suplementario [134].

5.5.4. REQ3. Predicción temprana de aprobados o suspensos

Para cumplir con este requisito, se desarrolla un estudio de caso utilizando el conjunto de datos que contiene las calificaciones de los estudiantes para tareas y exámenes a lo largo de un curso. Para trabajar con el conjunto de datos del *Curso de ciencias*

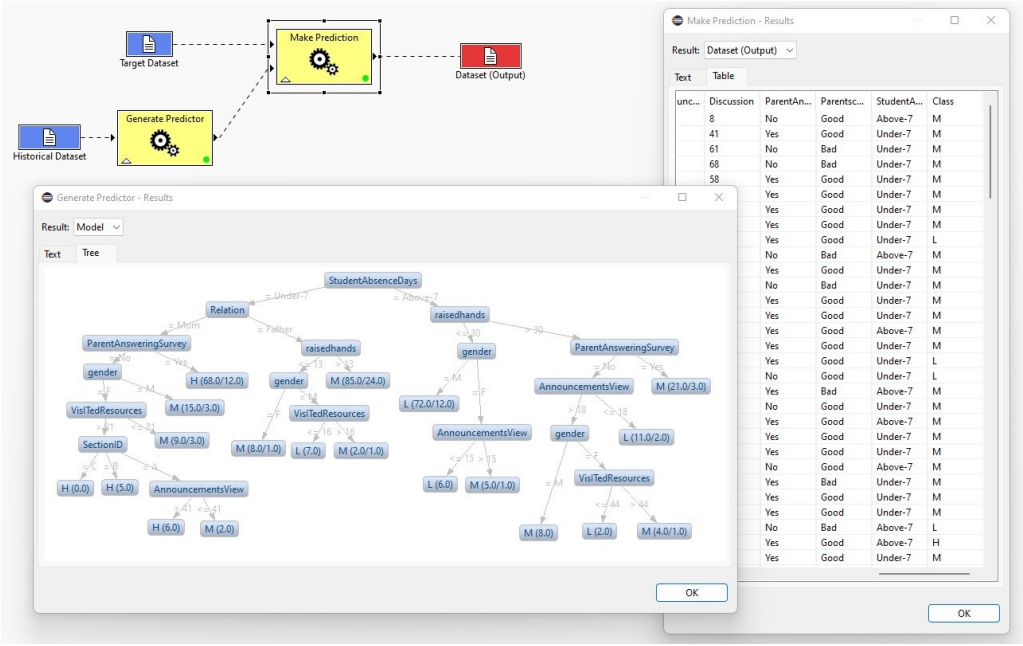


Figura 5.7: Flujo de trabajo, árbol de decisión y predicciones para el conjunto de datos de Kalboard 360.

para la predicción del rendimiento académico de los estudiantes, se sigue los procesos indicados por Injadat *et al.* [71]. Los autores definen dos momentos de predicción: la etapa del 20 %, basada en las calificaciones del primer cuestionario y la primera tarea, y la etapa del 50 %, que incluye una tarea adicional y el examen parcial. El conjunto de datos está desequilibrado, con solo un 4.3 % de estudiantes etiquetados como de "rendimiento débil". Para solventar este efecto en el entrenamiento, los autores utilizaron diversas estrategias de entrenamiento y evaluaron el rendimiento de varios clasificadores utilizando métricas de especificidad y sensibilidad. También se ha tenido en cuenta la naturaleza desbalanceada del conjunto de datos al diseñar el DIW.

La Figura 5.8 muestra los dos DIW creados para cada etapa de predicción. Los procesos de preprocesamiento y entrenamiento están definidos por tres operaciones de color azul. Primero, se elimina el atributo de ID del estudiante. Se puede usar la operación *Remove Property* en la vista experta de datos, que internamente contiene *Remove Attribute*. A continuación, la operación de *Class Balancer* asigna pesos a cada muestra de acuerdo con su distribución de clases. Esto compensa el desbalanceo de clases, asegurando que cada clase tenga el mismo peso total y que los errores de clasificación para la clase minoritaria se penalicen más durante el entrenamien-

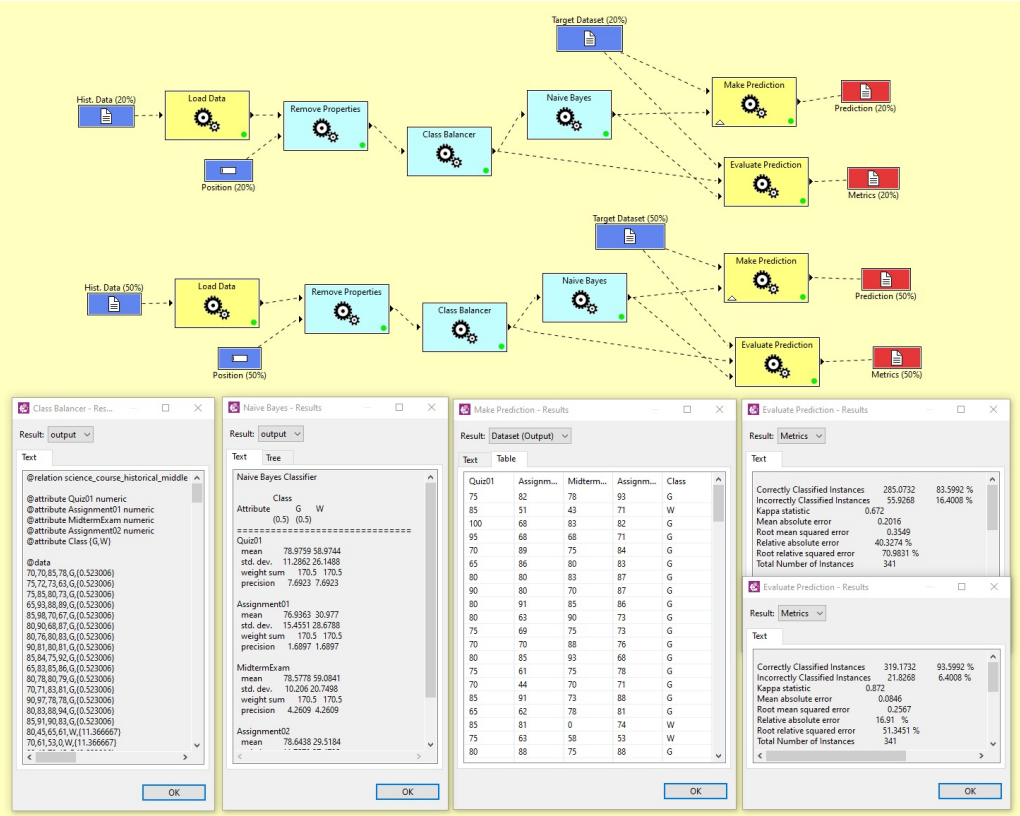


Figura 5.8: Flujo de trabajo, clasificador Naive Bayes y predicciones tempranas para el conjunto de datos del *Curso de ciencias*.

to. No es necesario realizar más preprocesamiento ya que el conjunto de datos no contiene valores faltantes, y las calificaciones están definidas en la misma escala. Para la clasificación, se cambia el algoritmo predeterminado a Naive Bayes, uno de los algoritmos comparados en el artículo original [71]. Para evaluar el rendimiento del algoritmo de Naive Bayes en los dos momentos del curso, se añade la operación *Evaluate Prediction* a ambos DIW.

Después de ejecutar ambos DIW con los conjuntos de datos correspondientes, se obtienen las predicciones y métricas de rendimiento. La Figura 5.8 muestra los resultados obtenidos en la etapa del 20 %. De los 145 estudiantes en el conjunto de datos objetivo, se predijo que 8 tenían un "rendimiento débil", es decir, se consideraron en riesgo. Las calificaciones finales reales confirmaron que 6 de estos estudiantes realmente tuvieron un rendimiento débil al final del curso. Sin embargo, otros dos estudiantes con calificaciones finales bajas no fueron detectados en la etapa del 20 %. Al observar las predicciones en la etapa del 50 %, se predijo que más estudiantes (9)

estaban en riesgo. El clasificador Naive Bayes confirmó la predicción de bajo rendimiento de los 6 estudiantes identificados en la etapa del 20 %. Sin embargo, los otros dos estudiantes no fueron detectados. Para identificar la razón del descenso en el rendimiento del estudiante, es probable que se requieran predicciones en una etapa posterior, ya que ambos estudiantes obtuvieron buenas calificaciones en tareas y exámenes en ambas etapas (calificaciones superiores a 50). Por lo tanto, la predicción del clasificador de "buen rendimiento" es razonable.

Otro hallazgo interesante es que el clasificador predijo un "rendimiento en riesgo" para estudiantes que obtuvieron buenas calificaciones al final del curso. Esto ocurrió para 2 estudiantes en la etapa del 20 %, 3 estudiantes en el 50 % y 2 estudiantes en ambas etapas. Al examinar los registros de los estudiantes, se descubrió que algunas tareas no se entregaron o las calificaciones fueron considerablemente más bajas que las de los otros estudiantes. Las predicciones tempranas alertan sobre tales anomalías, pero el riesgo de fracaso parece haberse mitigado en las tareas y exámenes finales. En términos de rendimiento de clasificación, el porcentaje de estudiantes clasificados correctamente en el conjunto de datos objetivo es del 85.6 % en la etapa del 20 % y del 93.6 % en la etapa del 50 %. Por lo tanto, incluso con datos limitados disponibles para el aprendizaje, se logra un buen rendimiento. A medida que se utilizan más datos del curso para el entrenamiento, el rendimiento mejora, lo que sugiere que las nuevas tareas y calificaciones de los exámenes ayudan a refinar el modelo de decisión.

Un estudio de caso con la vista avanzada del educador se puede encontrar en el material suplementario [134].

5.5.5. Discusión de los resultados

Los estudios de caso realizados han abarcado una amplia gama de casos de uso, demostrando de manera efectiva el potencial y la flexibilidad de LoCoAPe en el campo de EDM. Una de las principales fortalezas de LoCoAPe es su capacidad para dotar a los educadores de la capacidad de navegar fácilmente a través del *pipeline* de ML y definir sus DIW, lo que hace que todo el proceso de EDM sea más accesible y comprensible a través de una interfaz fácil de usar. Esto es gracias a la capacidad de generación de herramientas específicas de dominio de Workgenesis,

que junto a su integración con SWEL, permiten que los *citizen developers* puedan crear, o adaptar, y ejecutar sus propios DIW.

Además, la orientación a distintos perfiles de usuario (véase la Sección 2.3) permite una simplicidad y facilidad de uso. Las vistas proporcionadas de nivel superior ocultan detalles innecesarios, centrándose únicamente en aspectos esenciales como la definición del conjunto de datos y el tipo de tarea predictiva. Este enfoque de diseño es especialmente significativo para lograr una curva de aprendizaje baja y, por lo tanto, promover la adopción de la herramienta. Además, la estructura jerárquica de las vistas permite a los educadores interesados no solo acceder y editar flujos de trabajo predefinidos, sino también ajustar parámetros y extender operaciones según sea necesario. Esta flexibilidad ha permitido adaptar los DIW para acomodar características específicas del conjunto de datos, como múltiples fuentes de datos, la necesidad de limpieza de datos o la presencia de desbalanceo de clases.

La arquitectura en capas facilita la inspección de los componentes internos del DIW en busca de errores resultantes de un formato incorrecto o inconsistencias entre los requisitos de entrada y salida de operaciones individuales. Esta característica asegura que los DIW sean robustos y confiables, brindando a los educadores confianza en los resultados producidos por la herramienta. Otra ventaja notable de LoCoAPe es la capacidad de crear DIW desde cero utilizando una gran variedad de operaciones, los cuales pueden ser almacenados para su reutilización futura (véase el Capítulo 3). Esta característica es particularmente útil para los educadores con conocimientos en EDM ya que pueden aprovechar su experiencia para diseñar DIW personalizados adaptados a sus requisitos específicos.

Capítulo 6

Publicaciones asociadas a la tesis

En esta sección se detallan los trabajos científicos derivados de esta investigación.

6.1. Revistas indexadas

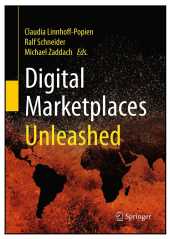
SWEL: A Domain-Specific Language for Modeling Data-Intensive Workflows

Título	SWEL: A Domain-Specific Language for Modeling Data-Intensive Workflows
Autores	R. Salado-Cid, A. Vallecillo, K. Munir, J.R. Romero
Revista	Business & Information Systems Engineering
Año	2023
Editorial	Springer
DOI	10.1007/s12599-023-00826-7

IF (JCR 2022)	7.9
Categoría	Computer Science, Information Systems
Posición	15/158 (Q1) - Decil 1

6.2. Capítulos de libro

On the Need of Opening the Big Data Landscape to Everyone: Challenges and New Trends



Título	On the Need of Opening the Big Data Landscape to Everyone: Challenges and New Trends
Autores	R. Salado-Cid, A. Ramírez, J.R. Romero
Libro	Digital Marketplaces Unleashed
Editores	C. Linnhoff-Popien, R. Schneider, M. Zaddach
Páginas	675–687
Año	2017
Editorial	Springer
DOI	10.1007/978-3-662-49275-8_60

6.3. Conferencias internacionales

1. R. Salado-Cid, J.R. Romero. *Enabling the Definition and Reuse of Multi-Domain Workflow-Based Data Analysis*. In Proceedings of the 16th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 687–696. Porto (Portugal). December 2016. DOI: 10.1007/978-3-319-53480-0_68.

6.4. Conferencias nacionales

1. R. Salado-Cid, J. Molino, J.R. Romero. *Interoperabilidad de flujos de trabajo intensivos en datos en Industria 4.0: caso de estudio*. En Actas de la XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA) - I Workshop en Aplicaciones de la Inteligencia Artificial para la Industria 4.0, pp. 1333–1338. Granada (España). Octubre 2018.

2. R. Salado-Cid, J.R. Romero. *Lenguaje específico para el modelado de flujos de trabajo aplicados a ciencia de datos*. En Actas de la XXI Jornadas de Ingeniería del Software y Bases de Datos (JISBD), pp. 387–396. Salamanca (España). Septiembre 2016. Disponible en: 11705/JISBD/2016/017

3. R. Salado-Cid, G. Luque, J.R. Romero. *Sistema de gestión de flujos de trabajo para la definición visual de aplicaciones basadas en algoritmos evolutivos*. En Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA) - II Jornadas de Algoritmos Evolutivos y Metaheurísticas (JAEM), pp. 261–270. Albacete (España). Noviembre 2015.

4. R. Salado-Cid, J.R. Romero, S. Ventura. *Metaherramienta para la generación de aplicaciones científicas basadas en workflows*. En Actas de la X Jornadas de Ciencia e Ingeniería de Servicios (JCIS), pp. 307–320. Cádiz (España). Septiembre 2014.

Capítulo 7

Conclusiones y trabajo futuro

"Si al principio la idea no es absurda,
entonces no hay esperanza para ella"
Albert Einstein

A lo largo del desarrollo de esta tesis doctoral, se han realizado diversas contribuciones en el ámbito de las DIA. La Sección 7.1 recopila las conclusiones. La aplicación de técnicas procedentes del campo de MDSE con un enfoque *low-code* y *no-code* sugiere que esta tesis doctoral podría servir como base para investigaciones futuras en el ámbito de la ciencia de datos. Por lo tanto, la en la Sección 7.2 se muestran algunas líneas de investigación futuras.

7.1. Conclusiones

En esta tesis doctoral se ha investigado el uso de mecanismos que permitan facilitar la creación de herramientas intensivas en datos adaptadas a dominios específicos, tal y como se detalló en la Sección 1.2.1. Para ello, se ha seguido la metodología de investigación centrada principalmente en el desarrollo y validación experimental de artefactos (Sección 1.3). Los artefactos diseñados e implementados se han orientado a dar solución a cada uno de los distintos subobjetivos, explicados en la Sección 1.2.2. Más específicamente, se pueden destacar las siguientes contribuciones:

Diseño e implementación de un lenguaje de modelado específico de dominio, independiente de cualquier herramienta

SWEL es un DSML que proporciona un metamodelo extensible para la especificación de DIW en diferentes niveles de abstracción, desde la especificación de alto nivel de problemas intensivos en datos hasta la representación detallada del *data pipeline* ejecutable. Además del metamodelo, sus restricciones y reglas de negocio, SWEL puede ser asociado a diferentes notaciones concretas, tanto textuales como gráficas, lo que permite su adaptabilidad a diversos contextos organizativos y distintas herramientas.

Hasta el momento, SWEL esta es la única propuesta formalmente metamodelada en este ámbito, lo que permite definir mecanismos para realizar transformaciones de modelos que permiten alcanzar la interoperabilidad y adaptabilidad del conocimiento modelado. Con este fin, se ha llevado a cabo una evaluación cuantitativa basada en métricas que permiten comparar SWEL con otras propuestas relacionadas, así como una evaluación mediante encuestas con expertos externos. Los resultados muestran que, en comparación con otros lenguajes, SWEL es un lenguaje adecuado para definir DIW y permitir la interoperabilidad entre herramientas. Además, los expertos externos han respaldado los beneficios en términos de comprensibilidad y practicabilidad que aporta el metamodelo en capas.

Diseño e implementación de mecanismos que permitan la interoperabilidad entre las distintas herramientas intensivas en datos actuales

A través del análisis y desarrollo de Workgenesis, se ha logrado establecer una solución basada en el principio de separación de responsabilidades y en los principios de MDSE. Estos principios han permitido proporcionar herramientas altamente especializadas y adaptadas a las necesidades particulares de los científicos de datos y de los expertos del dominio.

Uno de los principales logros ha sido la superación de la barrera del conocimiento técnico requerido para la creación y ejecución de DIA. Workgenesis se enfoca en ser una plataforma accesible y efectiva, permitiendo a expertos del dominio sin formación técnica el desarrollar soluciones software integrales de análisis de datos que les permitan obtener una ventaja competitiva.

La capacidad de registrar y adaptar servicios y algoritmos, así como definir la lógica del dominio a través de un SWfMS, es un aspecto fundamental para la creación y generación de herramientas altamente personalizadas. La flexibilidad y configurabilidad de Workgenesis permite la generación de soluciones específicas para cada dominio con un menor esfuerzo del que supondría el desarrollo a medida de una herramienta similar.

Por otro lado, la configuración de la metainformación, funcionalidades del *workbench* y capacidades del motor de ejecución proporciona un mayor nivel de adaptabilidad, permitiendo a los expertos del dominio disponer de cada herramienta según sus necesidades y preferencias. La posibilidad de elegir el tipo de motor de ejecución acorde a la plataforma de despliegue permite asegurar que las herramientas generadas se integren de manera óptima en diversos entornos de trabajo.

Workgenesis es una herramienta cuyo enfoque, centrado en el experto del dominio y su capacidad para aprovechar al máximo su conocimiento, permite la creación automática de soluciones intensivas en datos altamente especializadas, adaptables y efectivas.

Diseño e implementación de una herramienta que permitan generar herramientas visuales intensivas en datos ya adaptadas a dominios específicos

Como caso de aplicación práctica real de Workgenesis en la generación de herramientas visuales intensivas en datos adaptadas a un dominio de aplicación, se ha creado LoCoAPe. LoCoAPe es una herramienta altamente adaptable para aplicar técnicas de EDM en la resolución de tareas de predicción académica. Siguiendo los principios de diseño de herramientas *no-code*, LoCoAPe ofrece operaciones predefinidas que se pueden combinar de distintas maneras a través de una interfaz de usuario tipo arrastrar y soltar. De esta manera, LoCoAPe se adapta a educadores con distintos niveles de conocimiento en técnicas de EDM, desde aquellos que desean abordar tareas con DIW preconstruidos hasta aquellos que poseen la experiencia técnica, como para crear y configurar sus propios procesos de análisis de datos.

Los experimentos llevados a cabo muestran cómo LoCoAPe capacita a los educadores para predecir el rendimiento académico en diferentes etapas (predicción temprana o al final del curso), presentando las predicciones ya sea como calificaciones numéricas o como valores discretos (aprobado y suspenso). Además, se han utilizado varias operaciones de preprocesamiento en los experimentos que muestran la capacidad de LoCoAPe para llevar a cabo tareas de transformación de datos. La herramienta también admite la visualización de los modelos de decisión generados y sus predicciones, lo que permite a los educadores comprender el impacto de los factores educativos. En particular, LoCoAPe puede ser muy útil para los educadores de ingeniería, ya que las carreras de ingeniería sufren tasas de abandono más altas y el uso de LMS (para recopilar datos) está más consolidado.

7.2. Líneas de trabajo futuro

El reciente avance que está experimentando el campo de la inteligencia artificial, en especial el avance relacionado con los modelos de lenguaje de gran tamaño (LLM, por sus siglas en inglés) [19], supone una oportunidad de investigación para permitir la generación automática de DIW.

Por un lado, el metamodelo de SWEL es un candidato ideal para ser integrado en un LLM con el objetivo de permitir la definición de DIW a partir de una descripción en lenguaje natural proporcionada por el experto del dominio. Esto permitiría aumentar aún más el nivel de abstracción a la hora de crear DIA de forma más rápida y sencilla.

Por otro lado, es posible adaptar *scientific workflow execution language* (SWEL) para generar *data pipelines* de ML de forma automática. Para ello, se puede hacer uso de técnicas de AutoML que permitan, dado un conjunto de datos, seleccionar de forma automática el algoritmo de clasificación o regresión, la composición del flujo de trabajo y la hiperparametrización de los algoritmos de procesamiento, así como la generación de modelos que componen el *data pipeline*. Esto permitiría subir un nivel de abstracción la dificultad inherente a las técnicas de aprendizaje automático.

Finalmente, se ha de extender la generación de herramientas visuales adaptadas a un mayor rango de dominios intensivos en datos. Esto supone la evaluación y exposición

de SWEL a nuevos requisitos, lo que abre la posibilidad de diseñar extensiones del lenguaje para cubrir requisitos más específicos de cada dominio.

Parte I

Apéndices

Apéndice A

Metamodelo de SWEL

A.1. Introducción

En MDSE, un modelo es una representación simplificada de un sistema o concepto del mundo real, definido por un DSML. La sintaxis abstracta de un DSML es un metamodelo que especifica las relaciones entre los elementos del modelo.

SWEL es un DSML que proporciona a los expertos en dominio, incluidos los científicos de datos, una especificación de su sintaxis abstracta compuesta por todos los elementos, relaciones y restricciones necesarias para describir DIW, tanto a un alto nivel de abstracción como a un nivel bajo si se define una sintaxis concreta. La sintaxis abstracta de SWEL y su semántica se presenta en forma de metamodelo que es independiente de cualquier sintaxis concreta, y consecuentemente de cualquier implementación o herramienta particular.

Este lenguaje se basa en una arquitectura de tres capas, donde cada capa se centra en diferentes aspectos involucrados en la definición de un DIW. Una capa cuyos elementos permiten definir el grafo de ejecución subyacente, otra capa que proporciona elementos para capturar los conceptos del dominio, y una última capa para recoger información, legible por humanos, sobre el proyecto o experimentación para el que el DIW ha sido desarrollado.

Así, el principal objetivo de SWEL es permitir diseñar y ejecutar DIA a un alto nivel de abstracción en cualquier dominio impulsado por datos. Además, pretende

favorecer la reutilización y la portabilidad del conocimiento capturados por los DIW gracias a su independencia de cualquier plataforma.

El resto de este documento está organizado de la siguiente manera. Los fundamentos para el diseño y desarrollo de SWEL se explican en la Sección A.2. La Sección A.3 describe la estructura multicapa del lenguaje. Finalmente, las Secciones A.4, A.5 y A.6 describen en detalle los elementos del lenguaje que componen cada capa específica.

Una versión extendida de este Apéndice se puede consultar en el *Technical Report* [134].

A.2. Fundamentos

Estructura del flujo de trabajo intensivo en datos. Un flujo de trabajo consiste en una serie de tareas interconectadas que definen sus dependencias. Su estructura determina cómo se representan y ejecutan estas tareas y relaciones. Generalmente, hay dos tipos de representaciones: aquellas basadas en un grafo acíclico dirigido (DAG, por sus siglas en inglés) y aquellas basadas en un grafo cíclico dirigido (DCG, por sus siglas en inglés). Ambos tipos han sido utilizados para definir DIW, aunque las representaciones basadas en DAG son comúnmente utilizadas en dominios orientados a los datos. Sin embargo, los DCG podrían ser más apropiados para algunos dominios impulsados por la lógica empresarial. Por lo tanto, SWEL debe admitir ambos tipos para representar la gama más amplia posible de DIW y facilitar la reutilización entre un mayor número de herramientas.

Especificación del flujo de trabajo intensivo en datos. La definición de varios tipos de tareas depende del tipo de lenguaje utilizado. Los lenguajes abstractos describen tareas a un alto nivel de abstracción que no hacen referencia a plataformas específicas, recursos computacionales o lenguajes de programación. Estos flujos de trabajo no son directamente ejecutables, sino que deben pasar por una etapa de conversión que los asocia con recursos computacionales específicos. La responsabilidad de esta asociación generalmente recae en el motor de ejecución o en la propia herramienta. Por contra, los lenguajes concretos contienen tareas específicas que tie-

nen en cuenta detalles de implementación a nivel bajo, lo que los hace directamente ejecutables.

SWEL debería admitir tanto tareas de alto nivel de abstracción como tareas más específicas asociadas con aspectos a nivel bajo como el lenguaje de programación, la plataforma de ejecución (*cloud*, *grid*, etc.) o los modelos de ejecución (secuencial, paralelo, etc.). Contar con elementos de alto nivel de abstracción es esencial para la reutilización del conocimiento entre herramientas. Sin embargo, dado que la mayoría de las herramientas utilizan definiciones a nivel bajo, es necesario proporcionar elementos que permitan su definición, como la invocación de servicios web o la ejecución de programas escritos en un lenguaje de programación específico, entre otros.

A.3. Estructura del lenguaje

El lenguaje está especificado por un metamodelo, cuyos elementos permiten la definición de DIW. Los diferentes tipos de elementos que forman el metamodelo son:

- Metaclases: Una metaclase es la unidad básica que representa un concepto particular del lenguaje. Cada metaclase se define por:
  - Metapropiedades: Un conjunto de propiedades que determinan el comportamiento de la metaclase en el lenguaje. Las propiedades que definen una metaclase son: *isAbstract* para indicar que la metaclase no puede ser utilizada directamente en el lenguaje, *isFinal* para indicar que la metaclase no puede ser especializada y, finalmente, *isExtensible* para determinar si la metaclase puede ser extendida para ser utilizada en un dominio particular.
  - Generalización: La lista de metaclases de las cuales la metaclase hereda atributos y relaciones.
  - Especialización: La lista de metaclases que heredan los atributos y relaciones de la metaclase.
  - Atributos: El conjunto de valores que define la metaclase.

    - Asociaciones: La lista de relaciones que la metaclase tiene con otras metaclases.
    - Restricciones: El conjunto de reglas que restringen el comportamiento de la metaclase en el lenguaje.

- Paquetes: Un paquete es un grupo de metaclases que representan conceptos similares o relacionados. Un paquete puede importar las metaclases de otros paquetes para extender o modificar su definición.
- Capa: Una capa establece una división teórica entre paquetes, que se definen en un nivel de abstracción diferente.

SWEL presenta una arquitectura de tres capas (Figura A.1), donde cada capa se centra en diferentes aspectos de la definición del lenguaje.

- Capa morfológica: Esta capa contiene los elementos básicos de bajo nivel para definir un grafo ejecutable como un conjunto de nodos y conexiones. Estos elementos se extienden en la siguiente capa.
- Capa sintáctica: Esta capa extiende los elementos proporcionados por la capa morfológica para añadir nuevos elementos. Estos nuevos elementos definen un DIW como un conjunto de constructores interconectados que capturan los requisitos y recursos específicos del dominio.
- Capa de especificación: Esta capa define nuevos elementos para ampliar las capacidades de la capa sintáctica, proporcionando elementos que recopilan información legible por humanos relacionada con los experimentos computacionales intensivos en datos definidos por el DIW.

A.4. Capa morfológica

Esta capa agrupa los paquetes destinados a estructurar un DIW como un grafo ejecutable, o computacional, compuesto por nodos y conexiones. La capa morfológica está compuesta por un único paquete, *ExecutionGraph*.

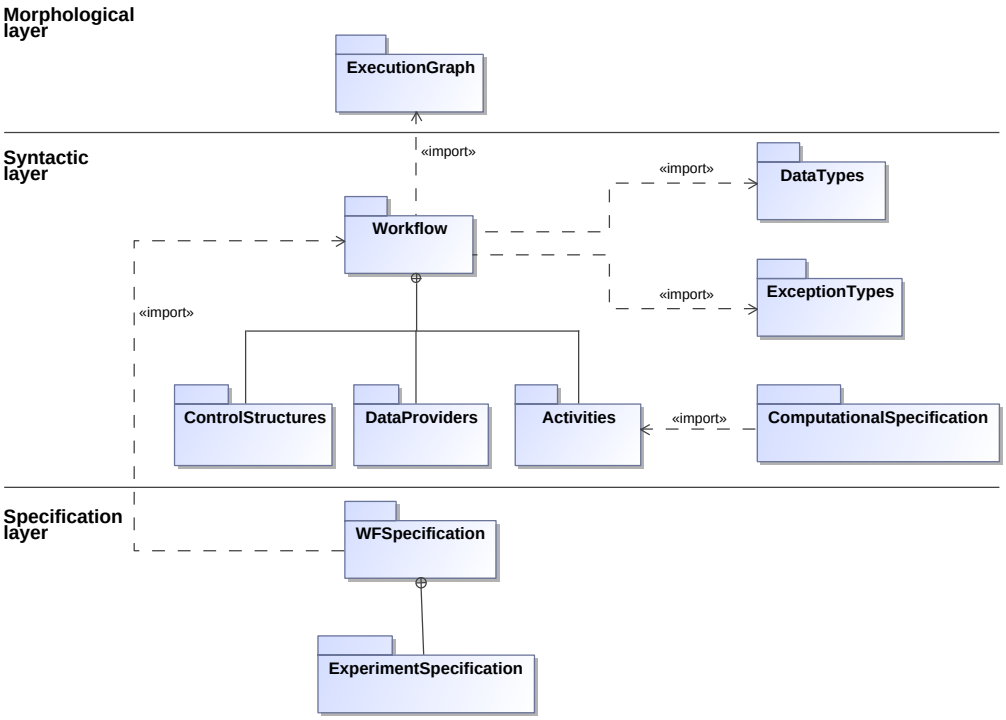


Figura A.1: Estructura multicapa de SWEL

A.4.1. Paquete *ExecutionGraph*

Este paquete proporciona metaclases para definir grafos ejecutables (Figura A.2). Cada nodo de este tipo de grafo representa una tarea computacional a ejecutar y las dependencias entre tareas se expresan mediante conexiones. Las metaclases que componen este paquete se enumeran en la Tabla A.1.

A.4.1.1. Metaclase *ExecutionGraph*

Un grafo de ejecución especifica el flujo de información en un proceso computacional de múltiples pasos. Este grafo está compuesto por un conjunto de vértices o nodos, los cuales pueden estar conectados por arcos o bordes.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

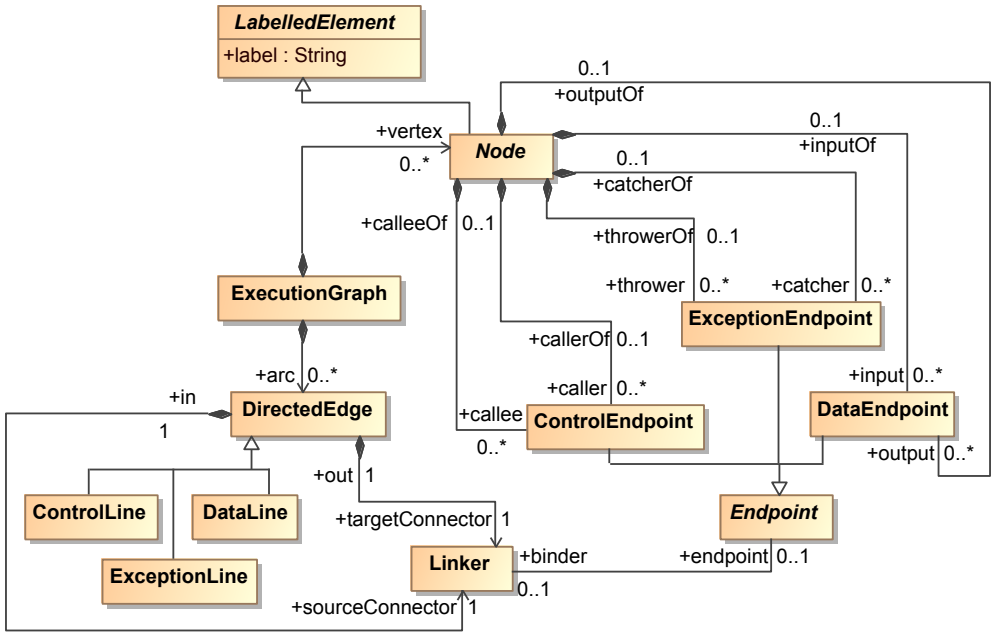


Figura A.2: Paquete *ExecutionGraph*

Asociaciones

- arc : DirectedEdge [0..*] – Conjunto de arcos utilizados para conectar nodos
- vertex : Node [0..*] – Conjunto de vértices o pasos del grafo de ejecución

A.4.1.2. Metaclase *Node*

Un nodo define la unidad fundamental de ejecución de la cual están compuestos los grafos de ejecución. Un nodo puede estar conectado a otros nodos mediante sus puntos de conexión, o puede existir sin estar conectado.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- LabelledElement (sección A.4.1.12)

Tabla A.1: Metaclases del paquete *ExecutionGraph*

Metaclase	**Descripción**	**Sección**
ExecutionGraph	Grafo compuesto por nodos ejecutables y aristas dirigidas	A.4.1.1
Node	Unidad de ejecución de un grafo de ejecución	A.4.1.2
DirectedEdge	Conexión entre nodos de un grafo de ejecución	A.4.1.3
ControlLine	Dependencia de control entre nodos	A.4.1.4
ExceptionLine	Dependencia entre nodos cuando se produce un error	A.4.1.5
DataLine	Dependencia de datos entre nodos	A.4.1.6
Endpoint	Punto de conexión de un nodo	A.4.1.7
ControlEndpoint	Punto de conexión para líneas de control	A.4.1.8
ExceptionEndpoint	Punto de conexión para líneas de excepción	A.4.1.9
DataEndpoint	Punto de conexión para líneas de datos	A.4.1.10
Linker	Relación entre una arista dirigida y un punto de conexión	A.4.1.11
LabelledElement	Elemento con nombre	A.4.1.12

Asociaciones

- output : DataEndpoint [0..*] – Conjunto de puntos de conexión para especificar la salida de datos del nodo
- caller : ControlEndpoint [0..*] – Conjunto de puntos de conexión para especificar el siguiente nodo a ejecutar
- callee : ControlEndpoint [0..*] – Conjunto de puntos de conexión para especificar que el nodo debe ser ejecutado después de un nodo previo
- input : DataEndpoint [0..*] – Conjunto de puntos de conexión para especificar la entrada de datos al nodo
- thrower : ExceptionEndpoint [0..*] – Conjunto de puntos de conexión para especificar una situación de error durante la ejecución del nodo
- catcher : ExceptionEndpoint [0..*] – Conjunto de puntos de conexión para gestionar una situación de error que ha sido propagada desde un nodo previo

A.4.1.3. Metaclase *DirectedEdge*

Un borde dirigido define una unidad básica de la cual están construidos los grafos de ejecución. Un borde dirigido especifica la conexión entre un nodo fuente y un nodo objetivo en el grafo de ejecución, que está definido por un enlazador.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- ControlLine (sección A.4.1.4)
- ExceptionLine (sección A.4.1.5)
- DataLine (sección A.4.1.6)

Asociaciones

- targetConnector : Linker [1] – El enlazador especifica el nodo destino
- sourceConnector : Linker [1] – El enlazador especifica el nodo origen

Restricciones

- "targetConnector" y "sourceConnector" deben ser diferentes

A.4.1.4. Metaclase *ControlLine*

Una línea de control es un tipo de borde dirigido que especifica una dependencia de control entre nodos, donde un nodo debe ser ejecutado después de otro nodo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- DirectedEdge (sección A.4.1.3)

Restricciones

- "sourceConnector" y "targetConnector" solo pueden asociarse con un enlazador cuyo punto de conexión sea "ControlEndpoint".

A.4.1.5. Metaclase *ExceptionLine*

Una línea de excepción es un tipo de borde dirigido que especifica la propagación de una situación de error producida durante la ejecución de un nodo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- DirectedEdge (sección A.4.1.3)

Restricciones

- "sourceConnector" y "targetConnector" solo pueden asociarse con un enlazador cuyo punto de conexión sea "ExceptionEndpoint".

A.4.1.6. Metaclase *DataLine*

Una línea de datos especifica el flujo de datos entre nodos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- DirectedEdge (sección A.4.1.3)

Restricciones

- "sourceConnector" y "targetConnector" solo pueden asociarse con un enlazador cuyo punto de conexión sea "DataEndpoint".

A.4.1.7. Metaclase *Endpoint*

Un punto de conexión es un punto de conexión de un nodo que permite la comunicación entre nodos mediante bordes dirigidos. El tipo de punto de conexión determina el tipo de comunicación que se puede establecer.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- ControlEndpoint (sección A.4.1.8)
- ExceptionEndpoint (sección A.4.1.9)
- DataEndpoint (sección A.4.1.10)

Asociaciones

- binder : Linker [0..1] – El enlazador se utiliza para especificar el borde dirigido que se va a enlazar con un punto de conexión de un nodo.

A.4.1.8. Metaclase *ControlEndpoint*

Un punto de conexión de control permite la conexión de líneas de control para especificar la dependencia de control entre nodos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- Endpoint (sección A.4.1.7)

Asociaciones

- callerOf : Node [0..1] – Nodo donde el punto de conexión permite llamar a un nodo para ejecutar.
- calleeOf : Node [0..1] – Nodo donde el punto de conexión se utiliza para ser llamado por un nodo.

Restricciones

- "callerOf" y "calleeOf" deben ser iguales cuando ambos existan.

A.4.1.9. Metaclase *ExceptionEndpoint*

Un punto de conexión de excepción permite la conexión de líneas de excepción para especificar situaciones de error durante la ejecución de un nodo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- Endpoint (sección A.4.1.7)

Asociaciones

- catcherOf : Node [0..1] – Nodo donde el punto de conexión permite capturar errores propagados.
- throwerOf : Node [0..1] – Nodo donde el punto de conexión permite propagar errores.

Restricciones

- "catcherOf" y "throwerOf" deben ser iguales cuando ambos existan.

A.4.1.10. Metaclase *DataEndpoint*

Un punto de conexión de datos permite la conexión de líneas de datos para especificar el flujo de datos entre nodos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- Endpoint (sección A.4.1.7)

Asociaciones

- inputOf : Node [0..1] – Nodo donde el punto de conexión permite entrada de datos.
- outputOf : Node [0..1] – Nodo donde el punto de conexión permite salida de datos.

Restricciones

- "inputOf" y "outputOf" deben ser iguales cuando ambos existan.

A.4.1.11. Metaclase *Linker*

Un enlazador especifica la conexión de un borde dirigido con un punto de conexión particular de un nodo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Asociaciones

- endpoint : Endpoint [0..1] – Punto de conexión utilizado para conectar el nodo y el borde dirigido.

A.4.1.12. Metaclase *LabelledElement*

Un elemento nombrado es un elemento que tiene un nombre. Esto permite que un elemento pueda ser referenciado utilizando su nombre como identificador.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- Node (sección A.4.1.2)

Atributos

- label : String [1] – La etiqueta del elemento

A.5. Capa sintáctica

Esta capa agrupa los paquetes que capturan los requisitos y recursos específicos del dominio en un flujo de trabajo. El paquete *Workflow* proporciona metaclases para

definir tareas específicas del dominio que se organizan como flujos de trabajo, el paquete *DataTypes* contiene metaclases que especifican los tipos de datos admitidos, el paquete *ExceptionTypes* permite la gestión y tolerancia a fallos y, finalmente, *ComputationalSpecification* proporciona metaclases para definir los recursos computacionales necesarios para ejecutar un DIW.

A.5.1. Package *Workflow*

Este paquete proporciona metaclases para definir diferentes tipos de DIW (Figura A.3). Un DIW está compuesto por un conjunto de elementos que intervienen en la ejecución de un proceso computacional, como estructuras de control, proveedores de datos y actividades, y el flujo de información entre ellos. Tales elementos se enumeran en la Tabla A.2.

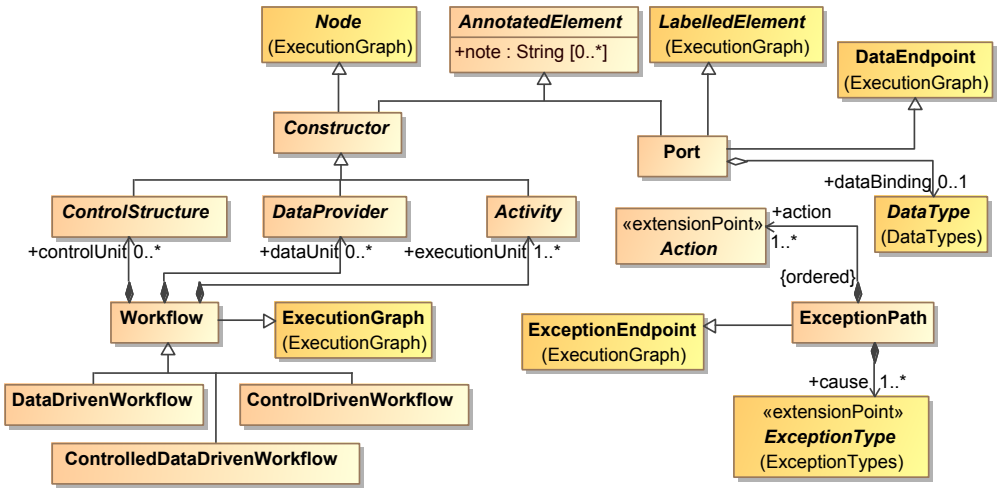


Figura A.3: Paquete *Workflow*

A.5.1.1. Metaclase *Workflow*

Un flujo de trabajo especifica la secuencia de pasos y recursos involucrados en un proceso computacional ejecutable, junto con el flujo de información necesario para coordinar y realizar dichos pasos. Se organiza como un grafo, donde los pasos están definidos por nodos y los flujos de información se especifican mediante bordes dirigidos. Dependiendo de los flujos de información y pasos admitidos, existen diferentes tipos de DIW.

Tabla A.2: Metaclases del paquete *Workflow*

Metaclase	Descripción	Sección
Workflow	Conjunto de tareas y recursos de un proceso computacional	A.5.1.1
DataDrivenWorkflow	DIW cuyo flujo de ejecución está determinado únicamente por dependencias de datos	A.5.1.2
ControlledDataDrivenWorkflow	DIW cuyo flujo de ejecución está determinado por dependencias de datos y dependencias de control	A.5.1.3
ControlDrivenWorkflow	DIW cuyo flujo de ejecución está determinado únicamente por dependencias de control	A.5.1.4
Constructor	Paso de un DIW	A.5.1.5
ControlStructure	Unidad de control de un DIW	A.5.1.6
DataProvider	Unidad de datos de un DIW	A.5.1.7
Activity	Unidad de ejecución de un DIW	A.5.1.8
Port	Tipos de datos expuestos por los constructores	A.5.1.9
ExceptionPath	Ruta alternativa en caso de error	A.5.1.10
Action	Operación a realizar en caso de error	A.5.1.11
JumpTo	Próximo paso a ejecutar en caso de error	A.5.1.12
Checkpoint	Guarda el estado actual de la ejecución	A.5.1.13
Replicate	Ejecución en un host diferente	A.5.1.14
Retry	Intenta la ejecución un número específico de veces	A.5.1.15
Abort	Detiene la ejecución del DIW	A.5.1.16
AnnotatedElement	Elementos con notas legibles por humanos	A.5.1.17

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ExecutionGraph (sección A.4.1.1)

Especialización

- DataDrivenWorkflow (sección A.5.1.2)
- ControlledDataDrivenWorkflow (sección A.5.1.3)
- ControlDrivenWorkflow (sección A.5.1.4)

Asociaciones

- executionUnit : Activity [1..*] – Conjunto de unidades de ejecución del DIW.
- controlUnit : ControlStructure [0..*] – Conjunto de unidades de control destinadas a gestionar el flujo de ejecución del DIW.
- dataUnit : DataProvider [0..*] – Conjunto de recursos de datos del DIW.
- description : WFSpecification::WFSpecification [0..*] – Información sobre el proyecto que condujo al diseño del DIW.

A.5.1.2. Metaclase *DataDrivenWorkflow*

Un flujo de trabajo dirigido por datos es un DIW cuyo flujo de ejecución está controlado únicamente por las dependencias de datos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Workflow (sección A.5.1.1)

Restricciones

- "controlUnit" debe ser cero.

A.5.1.3. Metaclase *ControlledDataDrivenWorkflow*

Un flujo de trabajo dirigido por datos controlado es un DIW cuyo flujo de ejecución es gestionado por las dependencias de datos. Sin embargo, tambíen puede definirse mediante dependencias de control si no hay dependencias de datos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Workflow (sección A.5.1.1)

A.5.1.4. Metaclase *ControlDrivenWorkflow*

Un flujo de trabajo controlado por control es un DIW cuyo flujo de ejecución es gestionado por las dependencias de control, que especifica el orden temporal en el que se ejecutan los nodos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Workflow (sección A.5.1.1)

A.5.1.5. Metaclase *Constructor*

Un constructor es la unidad básica que compone un DIW. El conjunto de constructores define la secuencia de pasos y recursos requeridos por la ejecución del DIW. Dependiendo del tipo de pasos o recursos, existen diferentes tipos.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Node (sección A.4.1.2)
- AnnotatedElement (sección A.5.1.17)

Especialización

- ControlStructure (sección A.5.1.6)
- DataProvider (sección A.5.1.7)
- Activity (sección A.5.1.8)

A.5.1.6. Metaclase *ControlStructure*

Una estructura de control es una unidad de control que define la gestión de la ejecución del DIW.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Constructor (sección A.5.1.5)

Especialización

- Fork (sección A.5.4.4)
- Begin (sección A.5.4.2)
- End (sección A.5.4.3)

- Conditional (sección A.5.4.1)
- Synchronizer (sección A.5.4.5)
- Merge (sección A.5.4.6)
- Counter (sección A.5.4.7)

A.5.1.7. Metaclase *DataProvider*

Un proveedor de datos es una unidad de datos que define los recursos de datos de un DIW. Un recurso de datos puede ser consumido y producido por los diferentes constructores del DIW.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Constructor (sección A.5.1.5)

Especialización

- Record (sección A.5.3.1)
- Resource (sección A.5.3.2)

A.5.1.8. Metaclase *Activity*

Una actividad es una unidad de ejecución que define un proceso computacional.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Constructor (sección A.5.1.5)

Especialización

- InteractionTask (sección A.5.2.1)
- ComputationalTask (sección A.5.2.2)
- DisplayTask (sección A.5.2.3)

A.5.1.9. Metaclase *Port*

Un puerto especifica el tipo de datos que pueden ser consumidos y producidos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- LabelledElement (sección A.4.1.12)
- DataEndpoint (sección A.4.1.10)
- AnnotatedElement (sección A.5.1.17)

Asociaciones

- dataBinding : DataTypes::DataType [0..1] – El tipo de dato que puede ser gestionado por el puerto.

A.5.1.10. Metaclase *ExceptionPath*

Una ruta de excepción especifica el conjunto de acciones a realizar en caso de error.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ExceptionEndpoint (sección A.4.1.9)

Asociaciones

- cause : ExceptionTypes::ExceptionType [1..*] – Lista de causas que producen un error.
- action : Action [1..*] – Lista ordenada de acciones a realizar en caso de error.

A.5.1.11. Metaclase *Action*

Una acción especifica cómo debe modificarse el flujo de ejecución en caso de error.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Especialización

- JumpTo (sección A.5.1.12)
- Checkpoint (sección A.5.1.13)
- Replicate (sección A.5.1.14)
- Retry (sección A.5.1.15)
- Abort (sección A.5.1.16)

A.5.1.12. Metaclase *JumpTo*

Esta metaclase define el constructor que debe ejecutarse en caso de error.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Action (sección A.5.1.11)

Asociaciones

- target : Constructor [1] – El constructor a ejecutar en caso de error.

A.5.1.13. Metaclase *Checkpoint*

Esta metaclase especifica que se debe crear un punto de control para guardar el estado de la ejecución del DIW.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Action (sección A.5.1.11)

A.5.1.14. Metaclase *Replicate*

Esta metaclase especifica que el constructor del DIW debe ejecutarse en una máquina diferente.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Action (sección A.5.1.11)

Atributos

- host : String [1] – El equipo donde volver a intentar la ejecución del constructor del DIW.

A.5.1.15. Metaclase *Retry*

Esta metaclase define que un constructor del DIW debe ejecutarse un número de veces en caso de error.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Action (sección A.5.1.11)

Atributos

- times : int [1] – El número de veces que se debe intentar la ejecución.
- delay : int [1] – El intervalo de tiempo, en milisegundos, que debe transcurrir entre cada intento.

A.5.1.16. Metaclase *Abort*

Esta metaclase determina que la ejecución del DIW se cancela en caso de error.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Action (sección A.5.1.11)

A.5.1.17. Metaclase *AnnotatedElement*

Un elemento anotado es un elemento que tiene un conjunto de información destinada a ser leída por humanos.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- Constructor (sección A.5.1.5)
- Port (sección A.5.1.9)

Atributos

- note : String [0..*] – Lista de información sobre el elemento.

A.5.2. Package *Activities*

Este paquete proporciona metaclases para definir tareas computacionales (Figura A.4) que reciben un conjunto de datos de entrada para producir nuevos datos de salida. En la Tabla A.3 se enumeran todas las metaclases.

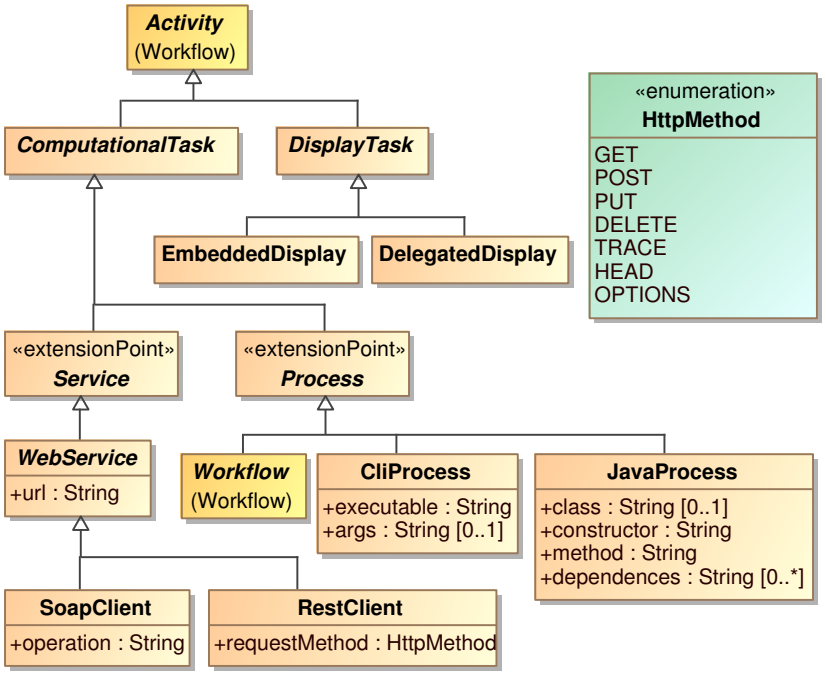


Figura A.4: Paquete *Activities*

A.5.2.1. Metaclase *InteractionTask*

Una tarea de interacción es una tarea que requiere la participación de un ser humano.

Tabla A.3: Metaclases del paquete *Activities*

Metaclase	Descripción	Sección
InteractionTask	Tarea que requiere intervención humana	A.5.2.1
ComputationalTask	Tarea ejecutada por computadoras	A.5.2.2
DisplayTask	Tarea para mostrar información de datos	A.5.2.3
Service	Servicio a ser consumido por clientes	A.5.2.4
WebService	Servicio servido usando el protocolo HTTP	A.5.2.5
RestClient	Cliente para consumir servicios REST	A.5.2.6
SoapClient	Cliente para consumir servicios SOAP	A.5.2.7
Process	Proceso de cómputo ejecutable	A.5.2.8
JavaProcess	Proceso a ser ejecutado usando Java	A.5.2.9
CliProcess	Proceso que expone una interfaz de línea de comandos	A.5.2.10
EmbeddedDisplay	Visualización a ser mostrada en el DIW	A.5.2.11
DelegatedDisplay	Visualización a ser mostrada en una herramienta externa	A.5.2.12
HttpMethod	Enumeración de métodos HTTP	A.5.3.7

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Generalización

- Activity (sección A.5.1.8)

A.5.2.2. Metaclase *ComputationalTask*

Una tarea computacional es una tarea que es ejecutada por una computadora.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Generalización

- Activity (sección A.5.1.8)

Especialización

- Service (sección A.5.2.4)
- Process (sección A.5.2.8)

Asociaciones

- computationalDescription : ComputationalSpecification::ComputationalResource [0..*] – Especificación de los recursos computacionales necesarios para su ejecución.

A.5.2.3. Metaclase *DisplayTask*

Una tarea de visualización es una tarea destinada únicamente a mostrar información. Esta información puede mostrarse utilizando tablas, gráficos, imágenes, etc.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Generalización

- Activity (sección A.5.1.8)

Especialización

- EmbeddedDisplay (sección A.5.2.11)
- DelegatedDisplay (sección A.5.2.12)

A.5.2.4. Metaclase *Service*

Un servicio es un tipo de tarea computacional que se ejecuta en cualquier ubicación. Normalmente, un servicio es consumido por clientes.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Generalización

- ComputationalTask (sección A.5.2.2)

Especialización

- WebService (sección A.5.2.5)

A.5.2.5. Metaclase *WebService*

Un servicio web es un tipo de servicio que utiliza el protocolo HTTP para ser expuesto y consumido por clientes web.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Service (sección A.5.2.4)

Especialización

- RestClient (sección A.5.2.6)
- SoapClient (sección A.5.2.7)

Atributos

- url : String [1] – La URL del servicio web

A.5.2.6. Metaclase *RestClient*

Un cliente REST define el consumo de un servicio web expuesto via HTTP.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- WebService (sección A.5.2.5)

Atributos

- requestMethod : HttpMethod [1] – El método HTTP a usar

A.5.2.7. Metaclase *SoapClient*

Un cliente SOAP define el consumo de un servicio web expuesto con SOAP/WSDL.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- WebService (sección A.5.2.5)

Atributos

- operation : String [1] – El método a ser consumido por el cliente.

A.5.2.8. Metaclase *Process*

Un proceso es una instancia de un programa que se ejecuta para generar nuevos datos de salida.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Generalización

- ComputationalTask (sección A.5.2.2)

Especialización

- JavaProcess (sección A.5.2.9)
- CliProcess (sección A.5.2.10)
- Workflow (sección A.5.1.1)

A.5.2.9. Metaclase *JavaProcess*

Un proceso Java es un tipo de proceso que define la ejecución de un programa en Java.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Process (sección A.5.2.8)

Atributos

- class : String [0..1] – La clase Java que contiene el método a ejecutar.
- constructor : String [1] – El constructor Java utilizado para crear una instancia.
- method : String [1] – El método a ejecutar.
- dependencies : String [0..*] – Las dependencias para ejecutar el programa Java.

A.5.2.10. Metaclase *CliProcess*

Un proceso de interfaz de línea de comandos es un tipo de proceso que define la ejecución de un programa que expone una interfaz de línea de comandos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Process (sección A.5.2.8)

Atributos

- executable : String [1] – La ruta al programa ejecutable.
- args : String [0..1] – Los argumentos proporcionados al programa al iniciar.

A.5.2.11. Metaclase *EmbeddedDisplay*

Una visualización incrustada define la configuración necesaria para que un visualizador gráfico pueda mostrar información sobre el DIW.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- DisplayTask (sección A.5.2.3)

A.5.2.12. Metaclase *DelegatedDisplay*

Una visualización delegada define la configuración necesaria para que una herramienta externa pueda mostrar información gestionada por el DIW.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- DisplayTask (sección A.5.2.3)

A.5.3. Package *DataProviders*

Este paquete proporciona metaclases para definir diferentes tipos de proveedores de datos (Figura A.5) destinados a gestionar los tipos de datos de un DIW según su origen y destino. En la Tabla A.4 se muestran las metaclases correspondientes.

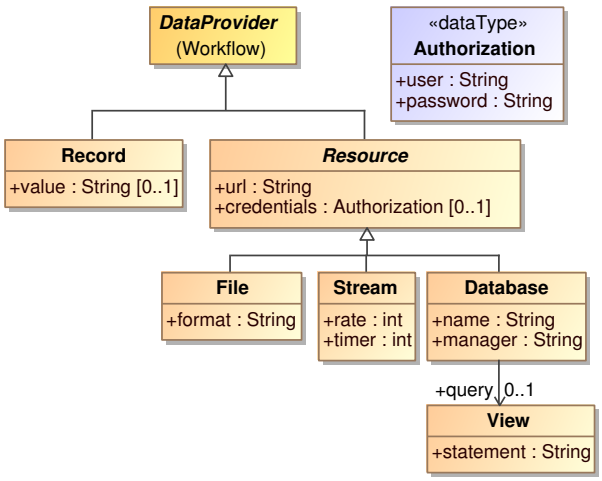


Figura A.5: Paquete *Data Providers*

Tabla A.4: Paquete *Data Providers*

Metaclase	Descripción	Sección
Record	Datos almacenados en la memoria principal	A.5.3.1
Resource	Datos almacenados en la memoria secundaria	A.5.3.2
File	Datos almacenados en un archivo	A.5.3.3
Stream	Datos para ser consumidos como un flujo de datos	A.5.3.4
Database	Datos almacenados en una base de datos	A.5.3.5
View	Subconjunto de datos obtenidos de una base de datos	A.5.3.6

A.5.3.1. Metaclase *Record*

Un registro define una estructura de datos básica que contiene información almacenada en memoria principal.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- DataProvider (sección A.5.1.7)

Atributos

- value : String [0..1] – La información almacenada en memoria principal.

A.5.3.2. Metaclase *Resource*

Un recurso define información que está almacenada y es accesible a través de una URL específica.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- DataProvider (sección A.5.1.7)

Especialización

- File (sección A.5.3.3)
- Stream (sección A.5.3.4)
- Database (sección A.5.3.5)

Atributos

- url : String [1] – La URL para acceder a la información.
- credentials : Authorization [0..1] – Datos de autorización para acceder a la información.

A.5.3.3. Metaclase *File*

Esta metaclase define el acceso a la información almacenada en un archivo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Resource (sección A.5.3.2)

Atributos

- format : String [1] – El formato de archivo que contiene la información.

A.5.3.4. Metaclase *Stream*

Esta metaclase define el acceso a la información que debe ser procesada mientras se transmite.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Resource (sección A.5.3.2)

Atributos

- rate : int [1] – La velocidad de lectura.
- timer : int [1] – La cantidad de tiempo para determinar que un flujo no está transmitiendo datos.

A.5.3.5. Metaclase *Database*

Esta metaclase define el acceso a la información almacenada en una base de datos y que debe ser accedida a través de un sistema de gestión de bases de datos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Resource (sección A.5.3.2)

Atributos

- manager : String [1] – La URL para acceder a la base de datos.

Asociaciones

- query : View [0..1] – Instrucción SQL para consultar datos.

A.5.3.6. Metaclase *View*

Una vista define una instrucción SQL para recuperar un conjunto de datos de una base de datos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Atributos

- statement : String [1] – La instrucción SQL para recuperar datos.

A.5.3.7. Enumeration *HttpMethod*

HttpMethod es un tipo de enumeración que especifica los literales para definir el método HTTP que se utilizará.

Values

- GET – Indica que el método HTTP method es GET.
- POST – Indica que el método HTTP method es POST.
- PUT – Indica que el método HTTP method es PUT.
- DELETE – Indica que el método HTTP method es DELETE.
- TRACE – Indica que el método HTTP method es TRACE.
- HEAD – Indica que el método HTTP method es HEAD.
- OPTIONS – Indica que el método HTTP method es OPTIONS.

A.5.4. Package *ControlStructures*

Este paquete proporciona metaclases para definir diferentes tipos de estructuras de control que gestionan el flujo de ejecución del DIW (Figura A.6). Todas las estructuras de control admitidas se enumeran en la Tabla A.5.

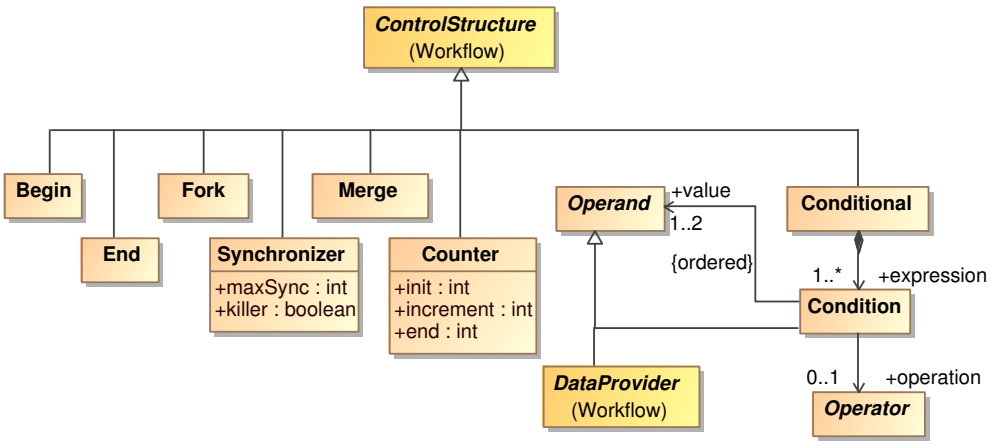


Figura A.6: Paquete *Control Structures*

Tabla A.5: Paquete *Control Structures*

Metaclase	**Descripción**	**Sección**
Conditional	Determina el siguiente paso a ejecutar según una condición	A.5.4.1
Begin	Determina el primero paso a ejecutar	A.5.4.2
End	Determina el último paso a ejecutar	A.5.4.3
Fork	Determina los pasos que pueden ejecutarse en paralelo	A.5.4.4
Synchronizer	Detiene la ejecución hasta que los pasos hayan terminado	A.5.4.5
Merge	Une los arcos dirigidos en uno solo	A.5.4.6
Counter	Determina el número de veces que el flujo de ejecución pasa por él	A.5.4.7
Condition	Expresión que puede ser verdadera o falsa	A.5.4.8
Operand	Objeto a evaluar en una condición	A.5.4.9
Operator	Tipo de operación de una condición	A.5.4.10
LogicalOperator	Operador basado en condiciones lógicas	A.5.4.11
RelationalOperator	Operador basado en relaciones	A.5.4.12
MathematicalOperator	Operador de operaciones matemáticas	A.5.4.13
ANDOperator	Operador lógico AND	A.5.4.14
OROperator	Operador lógico OR	A.5.4.15
NOTOperator	Operador lógico NOT	A.5.4.16
XOROperator	Operador lógico XOR	A.5.4.17
GreaterOperator	Operador lógico “mayor que”	A.5.4.18
GreaterEqualOperator	Operador lógico “mayor o igual que”	A.5.4.19
LessOperator	Operador lógico “menor que”	A.5.4.20
LessEqualOperator	Operador lógico “menor o igual que”	A.5.4.21
EqualOperator	Operador lógico de igualdad	A.5.4.22
NotEqualOperator	Operador lógico de desigualdad	A.5.4.23
SumOperator	Operador de suma	A.5.4.24
SubtractionOperator	Operador de resta	A.5.4.25
ProductOperator	Operador de producto	A.5.4.26
DivisionOperator	Operador de división	A.5.4.27
ModuleOperator	Operador de módulo	A.5.4.28

A.5.4.1. Metaclase *Conditional*

Un condicional define una estructura de control que evalúa una expresión como verdadera o falsa para determinar el siguiente paso a ejecutar.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ControlStructure (sección A.5.1.6)

Asociaciones

- evaluableExpression : Condition [1..*] – Una expresión booleana

A.5.4.2. Metaclase *Begin*

Esta metaclase define el inicio de la definición del DIW. Los constructores conectados a esta estructura de control serán los primeros en ejecutarse.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ControlStructure (sección A.5.1.6)

A.5.4.3. Metaclase *End*

Esta metaclase define el final de la definición del DIW. Los constructores conectados a esta estructura de control serán los últimos en ejecutarse.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ControlStructure (sección A.5.1.6)

A.5.4.4. Metaclase *Fork*

Esta metaclase define la separación del flujo de ejecución en múltiples ramas que pueden ejecutarse en paralelo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ControlStructure (sección A.5.1.6)

A.5.4.5. Metaclase *Synchronizer*

Esta metaclase define la unión del flujo de ejecución que previamente fue separado en múltiples ramas.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ControlStructure (sección A.5.1.6)

Atributos

- maxSync : int [1] – El número de ramas a ejecutar antes de continuar.
- killer : boolean [1] – Detener las ramas cuya ejecución no ha terminado.

A.5.4.6. Metaclase *Merge*

Este metaclase define la unión del flujo de ejecución que fue previamente separado en múltiples ramas por una estructura de control condicional.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ControlStructure (sección A.5.1.6)

A.5.4.7. Metaclase *Counter*

Este metaclase define un contador para registrar las veces que el flujo de ejecución pasa a través de él. Puede ser utilizado para crear bucles de forma explícita.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ControlStructure (sección A.5.1.6)

Atributos

- init : int [1] – El valor inicial.

- increment : int [1] – El valor que será sumado cada vez.
- end : int [1] – El valor máximo.

A.5.4.8. Metaclase *Condition*

Una condición define una expresión booleana que produce un valor positivo o negativo cuando se evalúa.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Operand (sección A.5.4.9)

Asociaciones

- value : Operand [1..2] – Los operadores de la expresión.
- operation : Operator [0..1] – El tipo de expresión.

A.5.4.9. Metaclase *Operand*

Un operando define el objeto de una expresión evaluable.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- Condition (sección A.5.4.8)
- DataProvider (sección A.5.1.7)

A.5.4.10. Metaclase *Operator*

Un operador define el tipo de una expresión evaluable.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- LogicalOperator (sección A.5.4.11)
- RelationalOperator (sección A.5.4.12)
- MathematicalOperator (sección A.5.4.13)

A.5.4.11. Metaclase *LogicalOperator*

Este metaclase define un tipo de operador basado en condiciones lógicas, tales como AND, OR o XOR.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Operator (sección A.5.4.10)

Especialización

- ANDOperator (sección A.5.4.14)
- OROperator (sección A.5.4.15)
- NOTOperator (sección A.5.4.16)
- XOROperator (sección A.5.4.17)

A.5.4.12. Metaclase *RelationalOperator*

Este metaclase define un tipo de operador basado en algún tipo de relación entre los operandos.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Operator (sección A.5.4.10)

Especialización

- GreaterOperator (sección A.5.4.18)
- GreaterEqualOperator (sección A.5.4.19)
- LessOperator (sección A.5.4.20)
- LessEqualOperator (sección A.5.4.21)
- EqualOperator (sección A.5.4.22)
- NotEqualOperator (sección A.5.4.23)

A.5.4.13. Metaclase *MathematicalOperator*

Este metaclase define un tipo de operador basado en operaciones matemáticas.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- Operator (sección A.5.4.10)

Especialización

- SumOperator (sección A.5.4.24)
- SubtractionOperator (sección A.5.4.25)
- ProductOperator (sección A.5.4.26)

- DivisionOperator (sección A.5.4.27)
- ModuleOperator (sección A.5.4.28)

A.5.4.14. Metaclase *ANDOperator*

Este metaclase define un operador AND.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- LogicalOperator (sección A.5.4.11)

A.5.4.15. Metaclase *OROperator*

Este metaclase define un operador OR.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- LogicalOperator (sección A.5.4.11)

A.5.4.16. Metaclase *NOTOperator*

Este metaclase define un operador NOT.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- LogicalOperator (sección A.5.4.11)

A.5.4.17. Metaclase *XOROperator*

Este metaclase define un operador XOR.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- LogicalOperator (sección A.5.4.11)

A.5.4.18. Metaclase *GreaterOperator*

Este metaclase define un operador que evalúa si el valor de un operando es mayor que otro.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- RelationalOperator (sección A.5.4.12)

A.5.4.19. Metaclase *GreaterEqualOperator*

Este metaclase define un operador que evalúa si el valor de un operando es igual o mayor que otro.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- RelationalOperator (sección A.5.4.12)

A.5.4.20. Metaclase *LessOperator*

Este metaclase define un operador que evalúa si el valor de un operando es menor que otro.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- RelationalOperator (sección A.5.4.12)

A.5.4.21. Metaclase *LessEqualOperator*

Este metaclase define un operador que evalúa si el valor de un operando es igual o menor que otro.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- RelationalOperator (sección A.5.4.12)

A.5.4.22. Metaclase *EqualOperator*

Este metaclase define un operador que evalúa si el valor de un operando es igual a otro.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- RelationalOperator (sección A.5.4.12)

A.5.4.23. Metaclase *NotEqualOperator*

Este metaclase define un operador que evalúa si el valor de un operando es diferente del otro.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- RelationalOperator (sección A.5.4.12)

A.5.4.24. Metaclase *SumOperator*

Este metaclase define un operador que suma el valor de los operandos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- MathematicalOperator (sección A.5.4.13)

A.5.4.25. Metaclase *SubtractionOperator*

Este metaclase define un operador que resta el valor de los operandos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- MathematicalOperator (sección A.5.4.13)

A.5.4.26. Metaclase *ProductOperator*

Este metaclase define un operador que multiplica el valor de los operandos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- MathematicalOperator (sección A.5.4.13)

A.5.4.27. Metaclase *DivisionOperator*

Este metaclase define un operador que divide el valor de los operandos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- MathematicalOperator (sección A.5.4.13)

A.5.4.28. Metaclase *ModuleOperator*

Este metaclase define un operador que encuentra el resto después de la división del valor de un operando por otro.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- MathematicalOperator (sección A.5.4.13)

A.5.5. Package *DataTypes*

Este paquete proporciona metaclases para definir los diferentes tipos de datos que son compatibles con un DIW (Figura A.7). La lista de tipos de datos compatibles se muestra en la Tabla A.6.

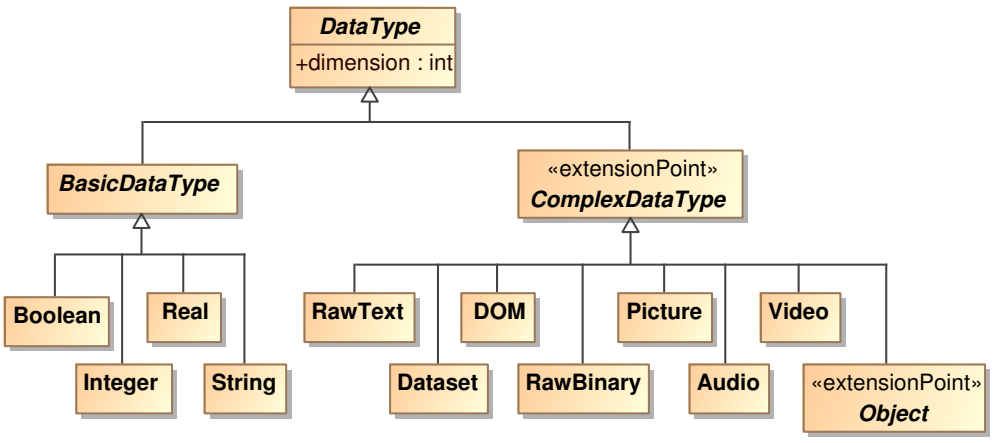


Figura A.7: Paquete *Data Types*

Tabla A.6: Paquete *Data Types*

Metaclase	Descripción	Sección
DataType	Tipo de dato a ser producido o consumido	A.5.5.1
BasicDataType	Tipos de datos primitivos básicos	A.5.5.2
ComplexDataType	Tipos de datos no primitivos	A.5.5.3
Boolean	Tipo de dato booleano	A.5.5.4
Integer	Tipo de dato entero	A.5.5.5
Real	Tipo de dato real	A.5.5.6
String	Tipo de dato "string"	A.5.5.7
RawText	Tipo de dato de texto plano	A.5.5.8
Dataset	Tipo de dato similar a una tabla	A.5.5.9
DOM	Tipo de dato de modelo de objeto de documento	A.5.5.10
RawBinary	Tipo de dato codificado en binario	A.5.5.11
Picture	Tipo de dato de imagen	A.5.5.12
Audio	Tipo de dato de audio	A.5.5.13
Video	Tipo de datos de vídeo	A.5.5.14
Object	Tipo de dato de instancia	A.5.5.15

A.5.5.1. Metaclase *DataType*

Un tipo de datos define el tipo de dato a ser consumido o producido a través de un puerto. Es equivalente a un tipo de datos primitivo en la mayoría de lenguajes de programación se denomina "básico", si no, se conoce como "complejo".

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- BasicDataType (sección A.5.5.2)
- ComplexDataType (sección A.5.5.3)

Atributos

- dimension : int [1] – Dimensión de los datos

A.5.5.2. Metaclase *BasicDataType*

Un tipo de datos básico es un tipo de datos primitivo que existe en la mayoría de lenguajes de programación (Java, C, JavaScript): número entero, booleano, texto...

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Generalización

- DataType (sección A.5.5.1)

Especialización

- Boolean (sección A.5.5.4)
- Integer (sección A.5.5.5)
- Real (sección A.5.5.6)
- String (sección A.5.5.7)

A.5.5.3. Metaclase *ComplexDataType*

Un tipo de datos complejo es cualquier tipo de dato que no se considere un tipo de dato básico. Por ejemplo, imágenes, videos, audios...

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Generalización

- DataType (sección A.5.5.1)

Especialización

- RawText (sección A.5.5.8)
- Dataset (sección A.5.5.9)
- DOM (sección A.5.5.10)
- RawBinary (sección A.5.5.11)
- Picture (sección A.5.5.12)
- Audio (sección A.5.5.13)
- Video (sección A.5.5.14)
- Object (sección A.5.5.15)

A.5.5.4. Metaclase *Boolean*

Esta metaclase define un tipo de dato booleano.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- BasicDataType (sección A.5.5.2)

A.5.5.5. Metaclase *Integer*

Esta metaclase define un tipo de dato de número entero.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- BasicDataType (sección A.5.5.2)

A.5.5.6. Metaclase *Real*

Esta metaclase define un tipo de dato de número flotante.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- BasicDataType (sección A.5.5.2)

A.5.5.7. Metaclase *String*

Esta metaclase define un tipo de dato de cadena de texto.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- BasicDataType (sección A.5.5.2)

A.5.5.8. Metaclase *RawText*

Esta metaclase define un tipo de dato cuyo contenido es texto plano.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.5.9. Metaclase *Dataset*

Esta metaclase define un tipo de dato cuyo contenido es estructurado como un conjunto de datos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.5.10. Metaclase *DOM*

Esta metaclase define un tipo de dato cuyo contenido se basa en un modelo de objeto de documento, como XML o HTML.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.5.11. Metaclase *RawBinary*

Esta metaclase define un tipo de dato cuyo contenido está codificado en binario.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.5.12. Metaclase *Picture*

Esta metaclase define un tipo de dato cuyo contenido es una imagen.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.5.13. Metaclase *Audio*

Esta metaclase define un tipo de dato cuyo contenido es un audio.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.5.14. Metaclase *Video*

Esta metaclase define un tipo de dato cuyo contenido es un vídeo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.5.15. Metaclase *Object*

Esta metaclase define un tipo de dato cuyo contenido es un objeto de un lenguaje de programación.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: Sí

Generalización

- ComplexDataType (sección A.5.5.3)

A.5.6. Package *ExceptionTypes*

Este paquete proporciona metaclases para definir los diferentes tipos de errores durante la ejecución del DIW (Figura A.8), que se enumeran en la Tabla A.7.

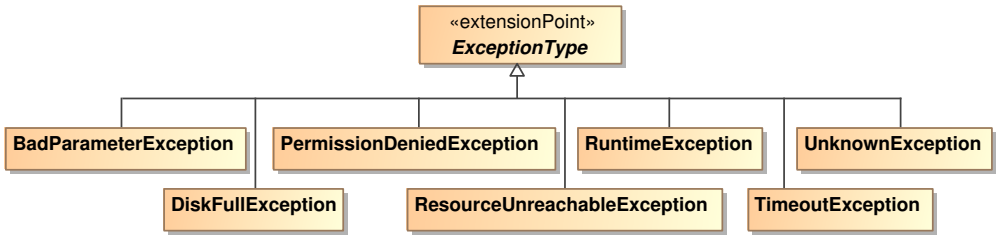


Figura A.8: Paquete *Exception Types*

Tabla A.7: Paquete *Exception Types*

ExceptionType	Error en la ejecución del DIW	A.5.6.1
BadParameterException	Error de parámetro incorrecto	A.5.6.2
DiskFullException	Error de disco lleno	A.5.6.3
PermissionDeniedException	Error de permiso denegado	A.5.6.4
ResourceUnreachableException	Error de recurso inaccesible	A.5.6.5
RuntimeException	Error en tiempo de ejecución	A.5.6.6
TimeoutException	Error de tiempo de espera	A.5.6.7
UnknownException	Error desconocido	A.5.6.8

A.5.6.1. Metaclase *ExceptionType*

Un tipo de excepción define un tipo de error que se produce durante la ejecución del DIW. Un error puede ocurrir durante la ejecución de un paso particular o antes de la ejecución del mismo.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Especialización

- BadParameter (sección A.5.6.2)
- PermissionDenied (sección A.5.6.4)
- DiskFullException (sección A.5.6.3)
- ResourceUnreachable (sección A.5.6.5)
- RuntimeException (sección A.5.6.6)
- TimeoutException (sección A.5.6.7)
- UnknownException (sección A.5.6.8)

A.5.6.2. Metaclase *BadParameterException*

Esta metaclase define que un error es causado por un parámetro de configuración incorrecto.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ExceptionType (sección A.5.6.1)

A.5.6.3. Metaclase *DiskFullException*

Esta metaclase define que un error es causado cuando no hay suficiente espacio de almacenamiento.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ExceptionType (sección A.5.6.1)

A.5.6.4. Metaclase *PermissionDeniedException*

Esta metaclase define que un error es causado cuando se deniega el acceso a un recurso.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ExceptionType (sección A.5.6.1)

A.5.6.5. Metaclase *ResourceUnreachableException*

Esta metaclase define que un error es causado cuando el acceso a un recurso no es alcanzable.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ExceptionType (sección A.5.6.1)

A.5.6.6. Metaclase *RuntimeException*

Esta metaclase define que un error ocurre durante la ejecución de un paso del DIW.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ExceptionType (sección A.5.6.1)

A.5.6.7. Metaclase *TimeoutException*

Esta metaclase define que un error se produce cuando ha expirado un tiempo de espera.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ExceptionType (sección A.5.6.1)

A.5.6.8. Metaclase *UnknownException*

Esta metaclase define que un error se produce por razones desconocidas.

Metapropiedades

- *isAbstract*: No
- *isFinal*: Sí
- *isExtensible*: No

Generalización

- ExceptionType (sección A.5.6.1)

A.5.7. Package *ComputationalSpecification*

Este paquete proporciona metaclases para definir los diferentes tipos de recursos computacionales necesarios para ejecutar una tarea computacional (Figura A.9). Esta información está destinada a optimizar la ejecución del DIW, por lo tanto, no es necesario incluirla en la definición del DIW. Todos los recursos computacionales se enumeran en la Tabla A.8.

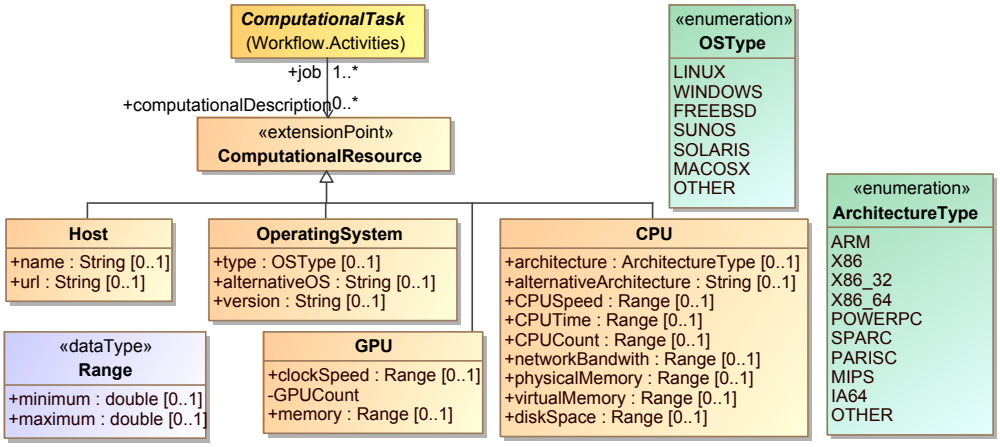


Figura A.9: Paquete *Computational Specification*

Tabla A.8: Paquete *Computational Specification*

Metaclase	Descripción	Sección
ComputationalResource	Tipo de recursos computacional	A.5.7.1
Host	Recurso accesible a través de la red	A.5.7.2
OperatingSystem	Tipo de sistema operativo	A.5.7.3
CPU	Tipo de CPU	A.5.7.4
GPU	Tipo de GPU	A.5.7.4
OSType	Enumeración de sistemas operativos	A.5.7.6
ArchitectureType	Enumeración de arquitecturas CPU	A.5.7.7

A.5.7.1. Metaclase *ComputationalResource*

Esta metaclase define el tipo de recursos computacional.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: Sí

Especialización

- Host (sección A.5.7.2)
- OperatingSystem (sección A.5.7.3)
- CPU (sección A.5.7.4)
- GPU (sección A.5.7.5)

A.5.7.2. Metaclase *Host*

Un *host* define la información necesaria para localizar una máquina específica en la red.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ComputationalResource (sección A.5.7.1)

Atributos

- name : String [0..1] – El nombre de la máquina en la red.
- url : String [0..1] – La URL para acceder a la máquina.

Restricciones

- "name" o "url" debe ser definido.

A.5.7.3. Metaclase *OperatingSystem*

Un sistema operativo define las características básicas de un sistema operativo.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ComputationalResource (sección A.5.7.1)

Atributos

- type : OSType [0..1] – El tipo de sistema operativo.
- alternativeOS : String [0..1] – Sistema operativo alternativo compatible.
- version : String [0..1] – Versión del sistema operativo.

A.5.7.4. Metaclase *CPU*

Esta metaclase define la configuración de una CPU.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ComputationalResource (sección A.5.7.1)

Atributos

- architecture : ArchitectureType [0..1] – El tipo de arquitectura.
- alternativeArchitecture : String [0..1] – Arquitectura alternativa compatible.
- CPUSpeed : Range [0..1] – Velocidad de reloj de la CPU.
- CPUTime : Range [0..1] – Número de segundos de CPU permitidas.

- CPUCount : Range [0..1] – Número de CPUs.
- networkBandwidth : Range [0..1] – Ancho de banda disponible en la red.
- physicalMemory : Range [0..1] – Cantidad de memoria física.
- virtualMemory : Range [0..1] – Cantidad de memoria virtual.
- diskSpace : Range [0..1] – Cantidad de espacio en disco.

A.5.7.5. Metaclase *GPU*

Esta metaclase define la configuración de la GPU.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- ComputationalResource (sección A.5.7.1)

Atributos

- clockSpeed : Range [0..1] – La velocidad de reloj de la GPU.
- GPUCount : Range [0..1] – Número de núcleos de la GPU.
- memory : Range [0..1] – Cantidad de memoria física.

A.5.7.6. Enumeración *OSType*

Tipo de enumeración que especifica los valores para definir el tipo de sistema operativo compatible.

Values

- LINUX – Indica que el sistema operativo está basado en Linux.
- WINDOWS – Indica que el sistema operativo está basado en Windows.
- FREEBSD – Indica que el sistema operativo está basado en FreeBSD.
- SUNOS – Indica que el sistema operativo está basado en SunOS.
- SOLARIS – Indica que el sistema operativo está basado en Solaris.
- MACOSX – Indica que el sistema operativo está basado en MacOsX.
- OTHER – Indica que el sistema operativo es desconocido.

A.5.7.7. Enumeración *ArchitectureType*

Especifica los valores para definir el tipo de arquitectura.

Values

- ARM – Indica que la arquitectura está basada en ARM.
- X86 – Indica que la arquitectura está basada en X86.
- X86_32 – Indica que la arquitectura está basada en X86_32.
- X86_64 – Indica que la arquitectura está basada en X86_64.
- POWERPC – Indica que la arquitectura está basada en POWERPC.
- SPARC – Indica que la arquitectura está basada en SPARC.
- PARISC – Indica que la arquitectura está basada en PARISC.
- MIPS – Indica que la arquitectura está basada en MIPS.
- IA64 – Indica que la arquitectura está basada en IA64.
- OTHER – Indica que la arquitectura es desconocida.

A.6. Capa de especificación

A.6.1. Paquete *WFSpecification*

Este paquete proporciona metaclases para definir metainformación relacionada con el DIW (Figura A.10). Esta información está relacionada con los recursos humanos (Tabla A.9) involucrados en el diseño y ejecución.

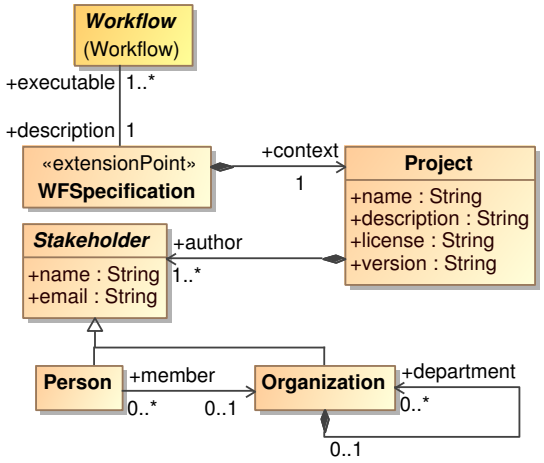


Figura A.10: Paquete *WFSpecification*

Tabla A.9: Paquete *WFSpecification*

Metaclase	Descripción	Sección
WFSpecification	Metainformación relacionada con el DIW	A.6.1.1
Project	Proyecto que propició el diseño del DIW	A.6.1.2
Stakeholder	Participante en el proyecto	A.6.1.3
Person	Entidad humana	A.6.1.4
Organization	Entidad compuesta por varias personas	A.6.1.5

A.6.1.1. Metaclase *WFSpecification*

Esta metaclase define los metadatos relacionados con el DIW. Esta información está destinada a ser leída por humanos, no a ser procesada por computadoras.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Asociaciones

- context : Project [1] – La información sobre el contexto que condujo al diseño del DIW.

A.6.1.2. Metaclase *Project*

Un proyecto define la información sobre el contexto en el que se ha diseñado el DIW. Esta información está destinada únicamente para ser consumida por humanos.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Atributos

- name : String [1] – Nombre del proyecto.
- description : String [1] – Descripción del proyecto.
- license : String [1] – Licencia del proyecto.
- version : String [1] – Versión del proyecto.

Asociaciones

- author : Stakeholder [1..*] – Responsable del proyecto.

A.6.1.3. Metaclase *Stakeholder*

Esta metaclase define la información principal sobre una entidad que está participando en el proyecto.

Metapropiedades

- *isAbstract*: Sí
- *isFinal*: No
- *isExtensible*: No

Especialización

- Person (sección A.6.1.4)
- Organization (sección A.6.1.5)

Atributos

- name : String [1] – El nombre de la entidad.
- email : String [1] – El correo electrónico de la entidad.

A.6.1.4. Metaclase *Person*

Esta metaclase define una entidad humana.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Stakeholder (sección A.6.1.3)

Asociaciones

- memberOf : Organization [0..1] – Organización a la que pertenece.

A.6.1.5. Metaclase *Organization*

Una organización especifica una entidad que comprende a varias personas, como una institución o una asociación.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Stakeholder (sección A.6.1.3)

Asociaciones

- department : Organization [0..*] – La lista de departamentos, estructurada como organizaciones, que componen la organización.

A.6.2. Paquete *ExperimentSpecification*

Este paquete proporciona metaclases para definir metainformación sobre un experimento científico (Figura A.11). Esta información no está destinada a ser utilizada durante la ejecución del DIW, sino que se utiliza como documentación para humanos sobre el problema a resolver.

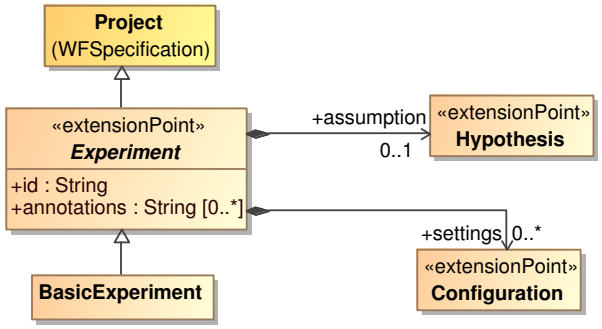


Figura A.11: Paquete *Experiment Specification*

A.6.2.1. Metaclase *Experiment*

Esta metaclase permite definir un experimento para apoyar, refutar o validar una hipótesis.

Tabla A.10: Paquete *Experiment Specification*

Metaclase	Descripción	Sección
Experiment	Tipo de experimento para aceptar o rechazar una hipótesis	A.6.2.1
BasicExperiment	Experimento con parametrización básica	A.6.2.2
Hypothesis	Problema a evaluar	A.6.2.3
Configuration	Configuración inicial	A.6.2.4

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: Sí

Generalización

- Project (sección A.6.1.2)

Especialización

- BasicExperiment (sección A.6.2.2)

Atributos

- id : String [1] – Identificador del experimento.
- annotations : String [0..*] – Lista de notas sobre el experimento.

Asociaciones

- assumption : Hypothesis [0..1] – La suposición que debe ser aceptada o rechazada por un experimento.
- settings : Configuration [0..*] – El conjunto de parámetros necesarios para realizar un experimento.

A.6.2.2. Metaclase *BasicExperiment*

Esta metaclase permite definir un experimento básico que no requiere una parametrización especial o información adicional.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: No

Generalización

- Experiment (sección A.6.2.1)

A.6.2.3. Metaclase *Hypothesis*

Esta metaclase permite definir una hipótesis que define el problema que debe ser probado por un experimento particular. La hipótesis puede ser aceptada o rechazada.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: Sí

A.6.2.4. Metaclase *Configuration*

Esta metaclase permite definir el conjunto de parámetros y ajustes iniciales de un experimento.

Metapropiedades

- *isAbstract*: No
- *isFinal*: No
- *isExtensible*: Sí

Apéndice B

Transformaciones de modelos de SWEL

B.1. Definición de los metamodelos

Un metamodelo es un modelo que describe la estructura y las reglas de construcción de otros modelos en un dominio específico, que es definido por un lenguaje de metamodelado. Uno de los lenguajes de metamodelado más destacados es ECORE, que forma parte del Eclipse Modeling Framework (EMF)[1] y se utiliza ampliamente en el desarrollo de aplicaciones en el entorno Eclipse.

ECORE se basa en el estándar de OMG conocido como MOF. Esto garantiza la interoperabilidad y la compatibilidad con otras herramientas y tecnologías de modelado que cumplen con el estándar MOF. ECORE permite definir metamodelos utilizando una sintaxis XML, proporcionando elementos para definir clases, atributos, relaciones, enumeraciones y otras construcciones de modelado. ECORE es altamente compatible con Eclipse y se integra con otras herramientas y tecnologías de modelado en el ecosistema Eclipse.

Ya que los lenguajes utilizados en la transformación de modelos no se encuentran metamodelados, la creación de los respectivos metamodelos con ECORE se ha realizado, mediante un proceso de ingeniería inversa, a partir del análisis de un gran número de instancias de flujos de trabajo de cada herramienta. Este proceso ha

[1] Eclipse Modeling Project. https://eclipse.dev/modeling/emf/ (consultado: 17/02/2024)

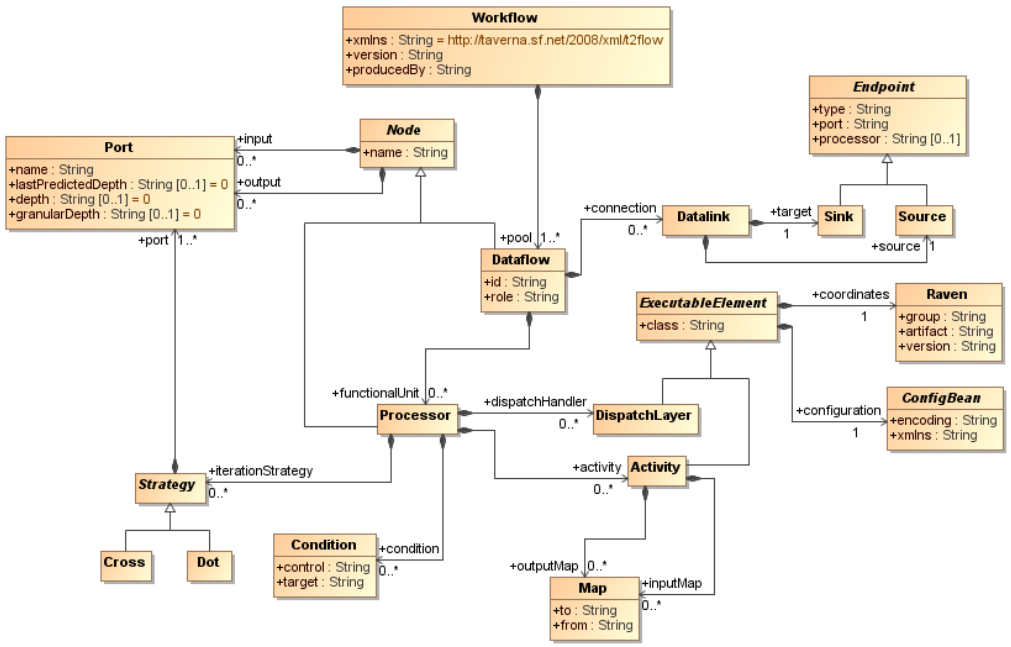


Figura B.1: Diagrama parcial del metamodelo de SCUFL definido.

permitido obtener una comprensión de la variedad de funcionalidades y elementos presentes en los flujos de trabajo, así como identificar patrones y estructuras comunes que se repiten en múltiples instancias.

Así se han podido inferir los elementos clave de los flujos de trabajo, como las tareas, los conexiones, condiciones, dependencias y los distintos tipos flujos de soportados. El resultado de este proceso de ingeniería inversa es la creación de los metamodelos en ECORE para SCUFL (Figura B.1), MoML (Figura B.2) y CWL que representan las estructuras y funcionalidades de los flujos de trabajo de cada herramienta.

Nótese que se ha definido un metamodelo parcial para cada lenguaje, considerando únicamente aquellos elementos relevantes para los ejemplos presentados. Además, estos metamodelos han sido validados con el propio uso, no por los respectivos creadores.

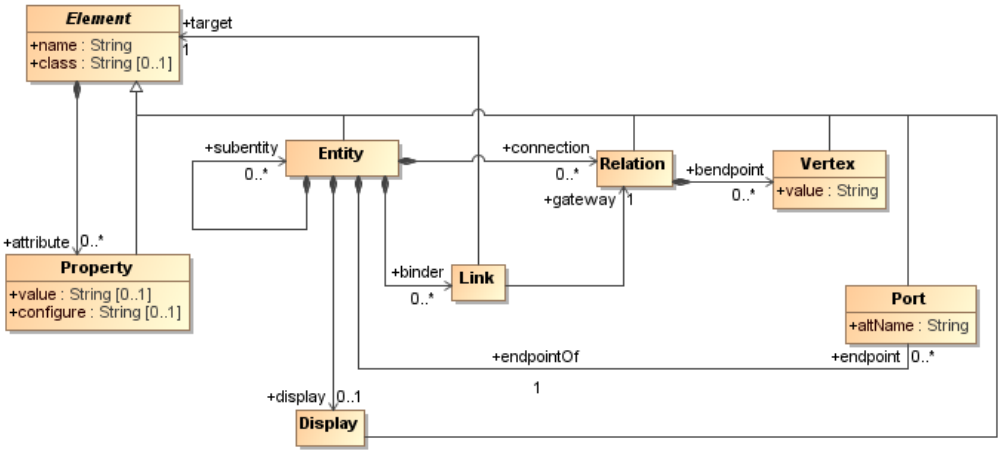


Figura B.2: Diagrama del metamodelo de MoML definido.

B.2. Transformaciones unidireccionales M2T

Para generar el código fuente a partir de un modelo se aplica una serie de transformaciones M2T. Estas transformaciones M2T se han definido utilizando Acceleo[2], un generador de código basado en plantillas de código abierto, desarrollado por Eclipse, basado en un estándar de la OMG. Este generador de código implementa un lenguaje de transformación M2T denominado MOFM2T, o Mof2Text[3], que combina fragmentos de código generativo con expresiones de consulta para acceder a los elementos del modelo de entrada (véase el Código B.1).

Las plantillas de Acceleo pueden presentar dependencias entre ellas durante el proceso de generación de código. Esto sucede cuando una plantilla invoca directamente a otra plantilla utilizando la sintaxis de llamada a plantilla de Acceleo, cuando una plantilla extiende o hereda el comportamiento de otra plantilla para reutilizar y especializar las plantillas existentes, o cuando una plantilla incluye o importa otra plantilla para utilizarla como parte de su propia definición, permitiendo así dividir una generación de código compleja en partes más pequeñas y manejables.

Código B.1: Plantilla de Acceleo.

```
1 [template public generateProperty(property : Property)]
2 [if (property.value = null and property.class = null)]
3 <property name="[property.name/]">
```

[2]Acceleo. http://www.eclipse.org/acceleo (consultado: 12/01/2024)

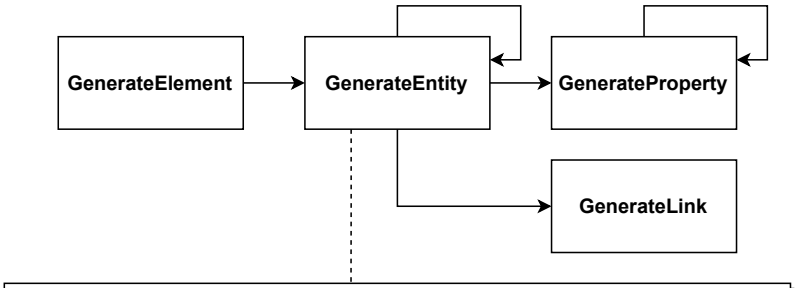
[3]MOF *model to text transformation language* (MOFM2T), 1.0. http://www.omg.org/spec/MOFM2T/1.0/PDF (consultado: 13/01/2024)

```
[elseif (property.value = null)]
<property name="[property.name/]" class="[property.class/]">
[else]
<property name="[property.name/]" class="[property.class/]"
    value="[property.value/]">
[/if]
[for (subproperty : Property | property.attribute)]
  [generateProperty(subproperty)/]
[/for]
</property>
[/template]
```

Los flujos de trabajo en Kepler se describen y representan utilizando un lenguaje de marcado específico llamado MoML. MoML permite representar la estructura del flujo de trabajo, incluidos los componentes del flujo de trabajo (llamados *actores* en Kepler), así como las conexiones y dependencias entre estos componentes. Además de la estructura del flujo de trabajo, MoML también permite especificar la configuración y los parámetros de los actores en el flujo de trabajo. Esto incluye información como valores predeterminados, configuraciones específicas del actor y cualquier otro parámetro relevante para la ejecución del flujo de trabajo. Toda esta información es serializada en formato XML utilizando etiquetas y atributos.

De esta manera, una serie de plantillas de Acceleo han sido implementadas para realizar estas transformaciones M2T. Todas las plantillas y sus dependencias se ilustran en la Figura B.3. Cada plantilla se utiliza para generar un elemento específico de MoML, de acuerdo con su metamodelo. *GenerateElement* es la plantilla principal destinada a iniciar la generación de todos los elementos utilizando *GenerateEntity*. La plantilla *GenerateEntity* genera la entidad MoML correspondiente y sus subentidades con la configuración específica (utilizando la plantilla *GenerateProperty*) y las conexiones entre entidades (mediante la plantilla *GenerateLink*). Un fragmento de una plantilla Acceleo se muestra en la Figura B.3. Esta plantilla genera el código fuente de cada entidad existente en el modelo MoML correspondiente (*Entity*).

Todas las transformaciones unidireccionales M2T implementadas se encuentran disponibles en el material suplementario [134].




```
[template public generateEntity(entity : Entity)]
<entity name="[entity.name/]" class="[entity.class/]">
  [for (property : Property | entity.attribute)]
    [generateProperty(property)/]
  [/for]
  [for (subentity : Entity | entity.subentity)]
    [generateEntity(subentity)/]
  [/for]
  [for (port : Port | entity.endpoint)]
    ...
    <port name="[port.name/]" class="[port.class/]">
    ...
    </port>
    ...
  [for (relation : Relation | entity.connection)]
    <relation name="[relation.name/]" class="[relation.class/]"
    ...
    </relation>
  [/for]
  [for (link : Link | entity.binder)]
    [generateLink(link)/]
  [/for]
</entity>
[/template]
```

Figura B.3: Gráfico de dependencias entre plantillas de Acceleo y vista parcial de la plantilla *GenerateEntity*.

B.3. Transformaciones unidireccionales M2M

QVT[4] es un lenguaje de transformación de modelos estándar, desarrollado por OMG, que se utiliza para especificar transformaciones M2M. QVT ofrece un conjunto de funcionalidades para definir reglas y operaciones que permiten transformar un modelo de origen (denominado modelo de entrada o modelo fuente) en uno o más modelos de destino (denominados modelos de salida o modelos objetivo).

Se denomina *relation* a un tipo específico de regla que define una relación entre conjuntos de elementos en modelos de entrada y modelos de salida (véase el Código B.2). Las relaciones en QVT son especialmente útiles cuando se necesita expresar

[4]Query/View/Transformation Specification 1.0. https://www.omg.org/spec/QVT/1.0/ (consultado: 12/01/2024)

correspondencias o asociaciones entre elementos de diferentes modelos que no se ajustan fácilmente a un patrón de transformación directa.

Las dependencias entre relaciones QVT pueden surgir cuando una relación depende del resultado de otra relación para su ejecución, o cuando una relación utiliza los resultados de otra relación como parte de su lógica interna. Estas dependencias pueden ser directas o indirectas, y pueden complejizar la secuencia de ejecución de las relaciones en una transformación.

Algunos escenarios comunes de dependencias entre relaciones en QVT son:

- Dependencias directas: Una relación puede depender directamente del resultado de otra relación. Por ejemplo, si la relación *R1* calcula un conjunto de elementos que luego se utiliza como entrada para la relación *R2*, entonces *R2* depende directamente de *R1*. En este caso, la relación *R2* no puede ejecutarse hasta que *R1* haya terminado de ejecutarse y haya generado su resultado.
- Dependencias indirectas: Una relación puede depender indirectamente del resultado de otra relación a través de una cadena de dependencias. Por ejemplo, si la relación *R1* depende de la relación *R2*, que a su vez depende de la relación *R3*, entonces *R1* tiene una dependencia indirecta de *R3*. En este caso, la ejecución de *R1* también está condicionada por la ejecución exitosa de *R2* y *R3*.
- Dependencias condicionales: Una relación puede depender de otra relación solo bajo ciertas condiciones. Por ejemplo, si la relación *R1* debe ejecutarse solo si ciertas condiciones son verdaderas, y estas condiciones se basan en los resultados de la relación *R2*, entonces hay una dependencia condicional entre *R1* y *R2*. En este caso, *R1* puede ejecutarse solo después de que *R2* haya producido su resultado y se hayan verificado las condiciones requeridas.
- Dependencias cíclicas: En algunos casos, puede haber dependencias cíclicas entre relaciones, donde una relación depende directa o indirectamente de sí misma. Estas dependencias deben manejarse cuidadosamente para evitar bucles infinitos o resultados inconsistentes en la transformación.

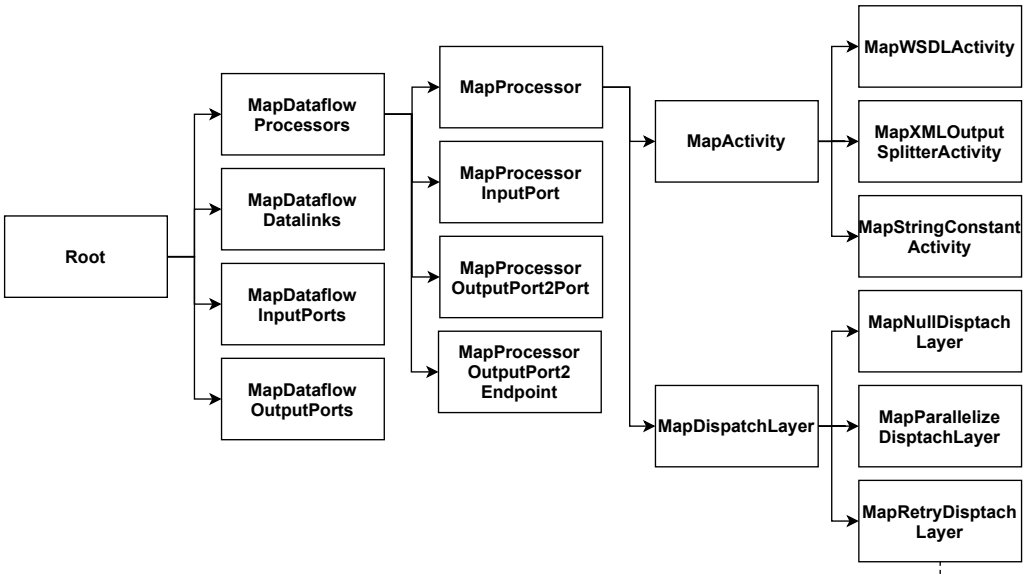
Código B.2: Relación QVT.

```
relation MapDataflowInputPorts {
  tname : String;

  checkonly domain source s : SCUFL::Dataflow {
    input = sourceInputPort : SCUFL::Port {
      name = tname
    }
  };

  enforce domain target t : SWEL::DataDrivenWorkflow {
    vertex = record : SWEL::Record {
      name = tname,
      output = output : SWEL::DataEndpoint {}
    }
  };

  when {
    MapDataflowProcessors(s, t);
  }
}
```

B.3.1. De SCUFL a SWEL

Una serie de transformación M2M se han definido para transformar el modelo de SCUFL en un modelo compatible de SWEL. Estas transformaciones están definidas utilizando QVT, que permite la definición de las relaciones para asociar los elementos del modelo fuente al modelo objetivo, conforme a los metamodelos correspondientes.

En la Figura B.4 se representan todas las relaciones definidas, así como las dependencias entre las mismas. Cada relación se ejecuta cuando se cumplen sus precondiciones, y define un conjunto de postcondiciones para determinar el orden de ejecución de las relaciones restantes. Así, la relación *Root* está destinada a iniciar la transformación de un flujo de trabajo SCUFL en un flujo de trabajo de SWEL, lo que permite la conversión de cada elemento de SCUFL en un elemento de SWEL, como procesadores de SCUFL en nodos de SWEL con la relación *MapDataflowProcessor* y todas sus relaciones directas e indirectas, enlaces de datos de SCUFL en líneas de datos de SWEL con la relación *MapDataflowDatalinks*, o puntos de conexión de entrada y salida de SCUFL en puertos de entrada y salida de SWEL con las relaciones *MapDataflowInputPorts* y *MapDataflowOutputPorts*.




```
relation MapRetryDispatchLayer {
    ttimes : Integer;
    tdelay : Integer;
    checkonly domain source s : SCUFL::ConfigBeans::RetryConfig
    { maxRetries = ttimes,
      initialDelay = tdela };
    enforce domain target t : SWEL::Workflow::ExceptionPath
    { action = retry : SWEL::Workflow::Retry {
      { times = ttimes,
        delay = tdelay },
      cause = unknown : SWEL::ExceptionTypes::UnknownException {}
    };
}
```

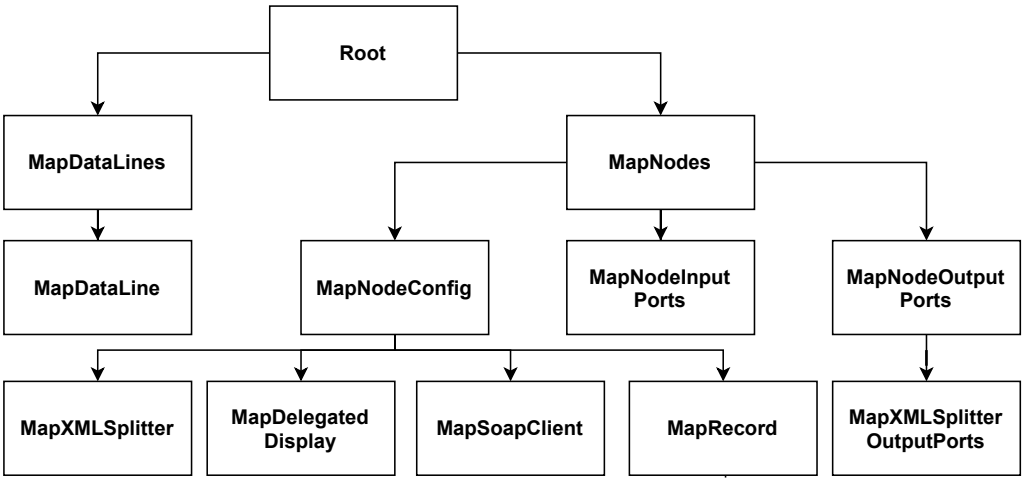
Figura B.4: Grafo de dependencias entre relaciones de QVT de SCUFL a SWEL y definición de la relación *MapRetryDispatchLayer*.

Además, una relación específica es ilustrada en la Figura B.4 que se encarga de asociar la capa de gestión de reintentos en la ejecución cuando se detecta un error en SCUFL, con el correspondiente en elemento de SWEL, de acuerdo al metamodelo de SWEL.

Todas las transformaciones unidireccionales M2M de SCUFL a SWEL implementadas se encuentran disponibles en el material suplementario [134].

B.3.2. De SWEL a MoML

Una serie de transformaciones M2M definen las relaciones entre los elementos del modelo SWEL y los elementos del modelo MoML. Estas transformaciones están definidas utilizando el lenguaje QVT y en la Figura B.5 se muestran las dependencias




```
relation MapRecord {
  evalue : String;
  checkonly domain source r : SWEL::Workflow::DataProviders::Record
  { value = evalue };
  enforce domain target e : MoML::Entity
  { class = 'ptolemy.actor.lib.StringConst',
    attribute = value : MoML::Property
    { name = 'value',
      class = 'ptolemy.data.expr.Parameter',
      value = evalue },
    attribute = firingCountLimit : MoML::Property
    { name = 'firingCountLimit',
      class = 'ptolemy.data.expr.Parameter',
      value = 'NONE' },
    attribute = none : MoML::Property
    { name = 'NONE',
      class = 'ptolemy.data.expr.Parameter',
      value = '0' } };
}
```

Figura B.5: Grafo de dependencias entre relaciones QVT de SWEL a MoML y definición de la relación *MapRecord*.

existentes entre todas las relaciones de QVT. La relación *Root* es la transformación inicial que depende de *MapDataLines* y *MapNodes* (y todas sus relaciones directas e indirectas) para convertir cada nodo y línea del flujo de trabajo de SWEL en una entidad, relación y enlace correspondientes de MoML. Una relación QVT concreta es mostrada en la Figura B.5 que es utilizada para transformar un proveedor de datos de tipo registro de SWEL en el elemento MoML correspondiente, de acuerdo con el metamodelo MoML.

Todas las transformaciones unidireccionales M2M de SWEL a MoML implementadas se encuentran disponibles en el material suplementario [134].

B.4. Transformaciones unidireccionales T2M

XSLT[5] es un lenguaje de transformación de texto basado en XML que se utiliza principalmente para transformar documentos XML en otros formatos, como HTML, XML o texto plano. Aunque XSLT se diseñó originalmente para la transformación de documentos XML a HTML para su presentación en la web, también se puede utilizar como herramienta de transformación T2M.

XSLT utiliza una sintaxis declarativa que especifica como debe ser el resultado de la transformación, en lugar de una sintaxis imperativa que describe los pasos para llegar al resultado (véase el Código B.3). Esto significa que en lugar de proporcionar instrucciones paso a paso sobre cómo realizar la transformación, se define un conjunto de reglas y patrones que se aplican a los elementos del documento de origen para generar el documento de salida.

En XSLT, se definen plantillas que especifican como se deben transformar los diferentes elementos del documento de origen en el documento de salida. Estas plantillas se aplican a los elementos que coinciden con ciertos patrones, que pueden basarse en el nombre del elemento, su estructura o cualquier otro criterio. Además, se permite la recursión y la anidación de plantillas, lo que permite realizar transformaciones complejas y estructuradas que implican la manipulación de múltiples elementos y niveles de anidamiento en el documento de origen.

Código B.3: Definición de una plantilla XSLT.

```
<xsl:template name="activities" match="activities">
      <xsl:for-each select="activity">
          <activity>
              <xsl:attribute name="class">
                  <xsl:value-of select="class" />
              </xsl:attribute>
          <xsl:apply-templates select="." />
          </activity>
      </xsl:for-each>
</xsl:template>
```

[5]XSL Transformations (XSLT) 2.0. https://www.w3.org/TR/xslt20/ (consultado: 15/01/2024)

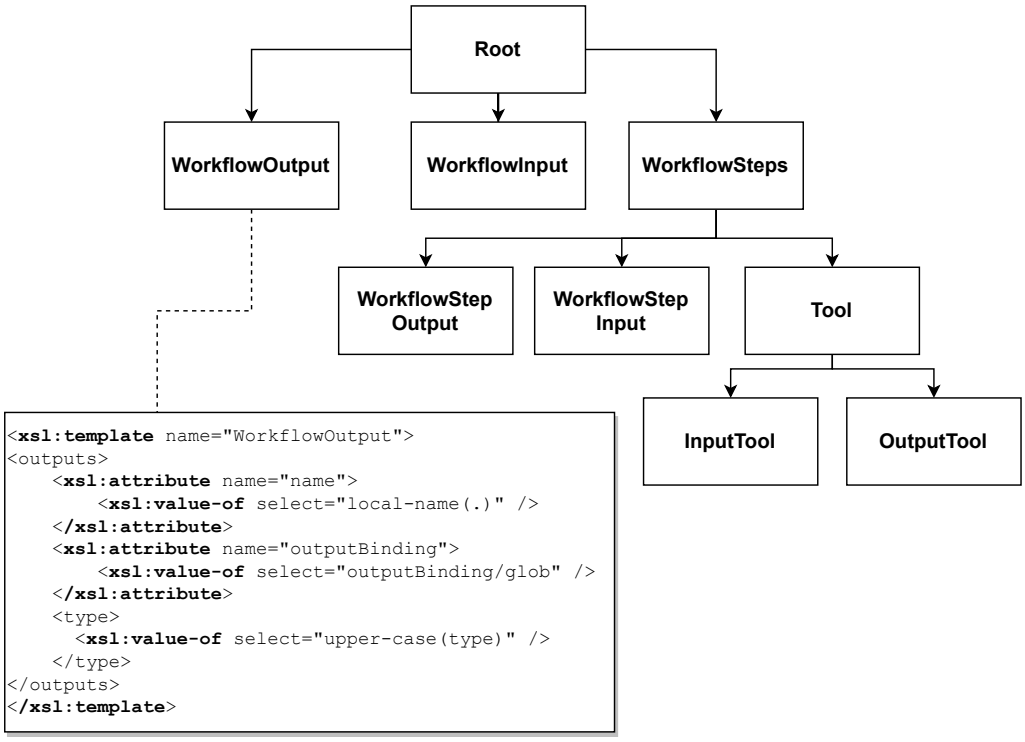


Figura B.6: Grafo de dependencias de plantillas XSLT para CWL y definición de la plantilla *Datalinks*.

B.4.1. CWL

Para transformar una serialización de un flujo de trabajo de CWL en un modelo CWL se ha definido un conjunto de plantillas XSLT. Las plantillas XSLT y el orden de ejecución se ilustran en la Figura B.6. Cada plantilla permite la conversión de un elemento específico de CWL, o parte de él, en su correspondiente elemento del modelo. La primera plantilla que se procesa es *Root*. La plantilla *Root* transforma el flujo de trabajo raíz de CWL y especifica el orden de las siguientes plantillas que se deben utilizar para procesar los elementos que componen un flujo de trabajo. Por lo tanto, las siguientes plantillas son *WorkflowOutput* y *WorkflowOutput*, que realiza la transformación de los elementos de entrada y salida de un flujo de trabajo de CWL, en su representación de modelo correspondiente. La definición de la plantilla XSLT para *WorkflowOutput* es mostrada en la Figura B.6. Los distintos pasos del flujo de trabajo de CWL, como la herramienta de ejecuión de CLI, así como sus entradas y salidas, son transformadas por la plantilla *WorkflowSteps*. La configuración de estos pasos son gestionadas por las dependencias de esta plantilla.

Todas las transformaciones unidireccionales T2M de CWL implementadas se encuentran disponibles en el material suplementario [134].

B.4.2. SCUFL

Para transformar una serialización de un flujo de trabajo de SCUFL en un modelo de SCUFL, se ha definido un conjunto de plantillas XSLT. Las plantillas XSLT y el orden de ejecución se ilustran en la Figura B.7. Cada plantilla permite la conversión de un elemento específico de SCUFL, o parte de él, en su correspondiente elemento del modelo. La primera plantilla que se procesa es *Root*. La plantilla *Root* transforma el flujo de trabajo raíz de SCUFL y especifica el orden de las siguientes plantillas que se deben utilizar para procesar los elementos que componen un flujo de trabajo. Por lo tanto, la siguiente plantilla es *Datalinks*, que realiza la transformación de los enlaces de datos de SCUFL en su representación de modelo correspondiente, donde los enlaces de datos y sus propiedades se asignan de acuerdo con el metamodelo de SCUFL. Un fragmento de código XSLT mostrado en la Figura B.7 muestra esta plantilla. A continuación, las entradas y salidas de datos, y los servicios web y procesos SOAP correspondientes del flujo de trabajo raíz se transforman mediante las plantillas *Port* y *Processor*, respectivamente. Para transformar un procesador, se utiliza nuevamente la plantilla *Port*, además de *Activities* y *Configuration*. Estas últimas plantillas asignan diferentes propiedades utilizadas para ejecutar un procesador en el WfMS correspondiente.

Todas las transformaciones unidireccionales T2M de SCUFL implementadas se encuentran disponibles en el material suplementario [134].

B.5. Transformaciones bidireccionales M2M

Aunque inicialmente se concibió principalmente para transformaciones unidireccionales, permitiendo la conversión de modelos de una forma a otra de manera efectiva y eficiente, QVT también se destaca por su capacidad para manejar transformaciones bidireccionales. Estas transformaciones bidireccionales M2M permiten la sincronización y la propagación de cambios entre dos modelos, asegurando que cualquier

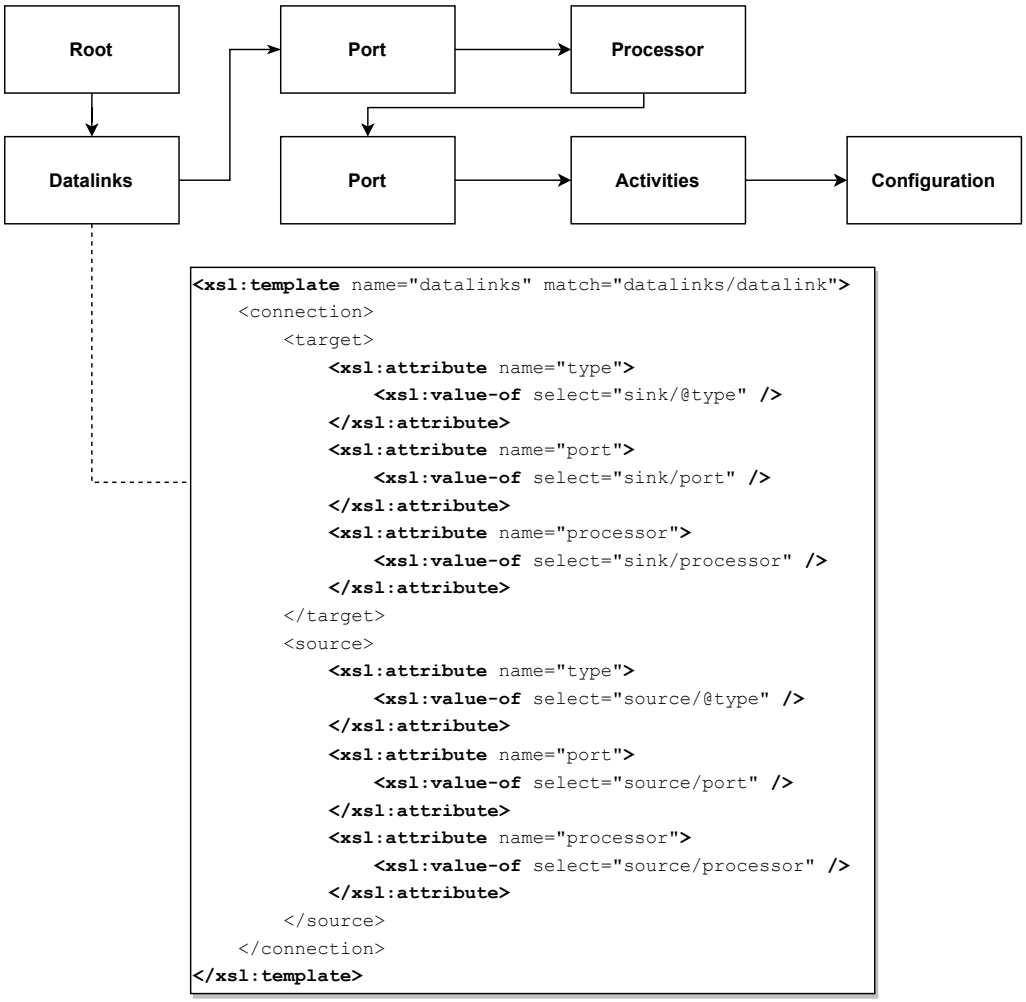


Figura B.7: Grafo de dependencias de plantillas XSLT para SCUFL y definición de la plantilla *Datalinks*.

modificación realizada en uno de los modelos se refleje de manera coherente en el otro y viceversa.

QVT ofrece una amplia gama de constructores y operaciones que permiten expresar transformaciones bidireccionales complejas entre modelos. Esto incluye la capacidad de especificar reglas de correspondencia entre elementos de los modelos de entrada y salida, así como reglas de transformación inversa para revertir los cambios.

En las transformaciones bidireccionales, es fundamental garantizar que los cambios realizados en un modelo se reflejen de manera consistente en el otro modelo. QVT proporciona mecanismos para sincronizar automáticamente los cambios entre los modelos y mantener su coherencia durante la transformación. En situaciones don-

de se producen conflictos entre los cambios realizados en los dos modelos, QVT proporciona mecanismos para resolver estos conflictos de manera controlada y predecible. Esto puede incluir la especificación de reglas de resolución de conflictos o la intervención manual del usuario para tomar decisiones sobre cómo manejar los conflictos.

Además, QVT puede mantener un historial de cambios realizado en los modelos durante la transformación bidireccional. Esto permite realizar un seguimiento de los cambios realizados en cada modelo, y facilita la auditoría y la reversión de cambios en caso necesario.

En la Figura B.8 se muestran todas las relaciones definidas entre los elementos de SWEL y CWL, así como las dependencias existentes entre las mismas. Así, la relación *Root* iniciar la transformación de un flujo de trabajo de CWL en un flujo de trabajo de SWEL, y viceversa. Las relaciones *MapWorkflowInputs*, *MapWorkflowInputs*, *MapOutputToolParametersReferences* y *MapWorkflowStepInputParameterReferences*, así como todas sus relaciones directas e indirectas, establecen la relación entre los puntos de entrada y salida de un flujo de trabajo de SWEL y CWL. Finalmente, la relación *MapWorkflowSteps* es la que define la relación entre los distintos pasos de un flujo de trabajo de SWEL y CWL, esta relación tiene una serie de relaciones directas dependiendo del tipo de paso a relacionar. La definición de la relación *MapOutputToolParameter2Record* es mostrada en la Figura B.8.

Todas las transformaciones bidireccionales M2M de CWL a SWEL, y viceversa, implementadas se encuentran disponibles en el material suplementario [134].

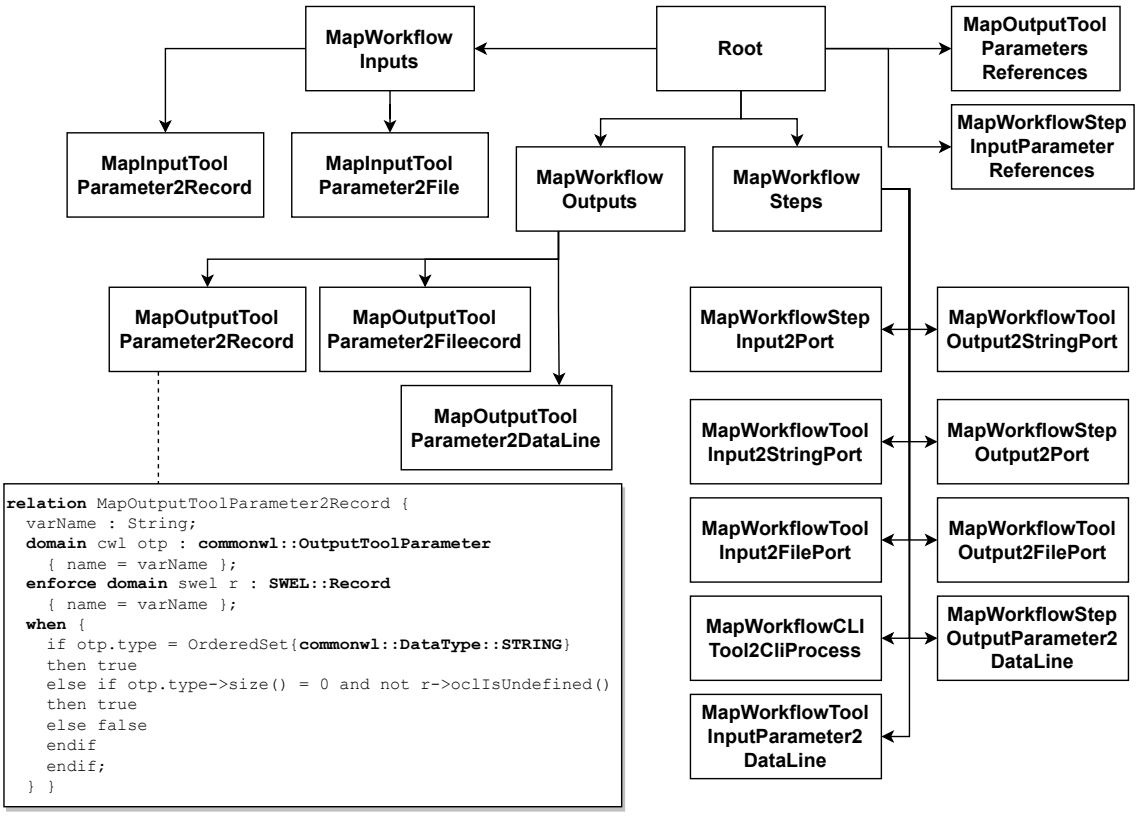


Figura B.8: Grafo de dependencias entre relaciones de QVT de SWEL a CWL y definición de la relación *MapOutputToolParameter2Record*.

Apéndice C

Evaluación de expertos

C.1. Formulario de evaluación

EXPERT EVALUATION
"SWEL: A Domain-Specific Language for Modelling Data-Intensive Workflows"

PID:
Occupation:
Industry/Academy:

INSTRUCTIONS:
The experiment consists of using a tool designed to transform an input workflow carried out in Taverna Workbench Core 2.5 into an equivalent workflow in Kepler 2.5. To this end, this tool generates intermediate models conformant to the SWEL metamodel from the source workflow (Taverna's SCUFL). All models are displayed in the user interface so that, if necessary, more details of the internal transformation process can be known until the target workflow (Kepler's MoML) is reached. The resulting workflow is displayed in textual format in the transformation tool interface itself and can also be saved and loaded with the Kepler tool.

Notice that the scope of this experiment is limited to the following Taverna elements: *Workflow Input Port*, *Workflow Output Port*, *Text Constant*, *WSDL Service* and *XML Splitter*. Any other processing elements will be ignored by the tool as they are outside the scope of this experiment.

The experiment consists of three exercises:

1. The transformation of an *already existing workflow*: Firstly, an illustrative workflow created using Taverna is presented for your examination within the Taverna platform. You may review all its details and execute it to observe its behaviour. Subsequently, this workflow needs to be entered into the transformation tool to generate an equivalent workflow in Kepler. Once generated, the resulting workflow must be opened within Kepler to obtain further details and enable its execution. This is a training exercise.
2. To create a *new workflow*: The user is required to use the supported elements provided in this experiment within Taverna. Alternatively, if preferred, an existing workflow can be downloaded from a public repository such as myExperiment.org.
3. To analyse *the generated SWEL model*.

OBJECTIVE:
The experiment determines whether a data scientist considers the resulting Kepler workflows to be equivalent to the Taverna input workflows. This will determine the significance and practicability of SWEL as an intermediate language for interoperability between WfMSs.

CONTEXT:
The data management and knowledge discovery processes of any data intensive (DI) application are normally formulated as a pipeline. The pipeline contains a sequence of individual tasks that must be completed to obtain meaningful and comprehensive results. These tasks typically include, among others, data acquisition, cleansing and preparation, information analysis, and data visualisation. Such pipelines are known as either data-intensive workflows (DIW) or scientific workflows. One of the problems of existing DIW technologies is the inability to reuse knowledge between different workflow tools. Changing from one tool to another means re-creating the same

workflows, which may introduce mistakes and errors in the translation. Furthermore, segments of workflows created with different tools cannot be seamlessly combined.

To address these issues, we have developed SWEL, a metamodelled language for workflow definition and execution. This language is designed to be platform-agnostic and enables knowledge to be reused across different tools. As a result, a series of model transformations have been defined that use SWEL as an intermediate language. This facilitates the conversion of knowledge from one tool to another. For instance, a workflow defined in Taverna's SCUFL can be transformed into SWEL and subsequently used to generate an identical workflow in any other target tool, such as Kepler's MoML.

ACTIONS TO BE UNDERTAKEN:

IMPORTANT NOTE: Please answer all questions in the score table at the end of the survey.

Now, please *respond to questions 1 to 7*.

Exercise 1

ABOUT THE WORKFLOW: The workflow provided for this exercise extracts information about a particular gene from a nuclear protein database. The given *GeneName* is used by the SOAP/WSDL web service *GetGeneID* to obtain the corresponding gene identifier. This identifier is extracted from the XML returned by the web service thanks to the local process *XMLDisassembler*, which is used by the SOAP/WSDL web service *GetLinks* to obtain the links to the specific information. Both the gene name and information links are the outputs *GeneName* and *Links*.

Please complete the following steps:

1. Open the provided workflow (/source/Example.t2flow) with Taverna Workbench Core 2.5 provided.
2. Execute the transformation tool.
3. Select SCUFL as the input language (tool: Taverna) in the transformation tool and select MoML as the target (tool: Kepler).
4. Open the provided Taverna workflow in a plain text editor and copy & paste its content in the window "Source" of the transformation tool.
5. Select an output directory where all intermediate and resulting files of the transformation will be generated.
6. Start the transformation process with the "Transform" button.
7. Go to the output directory by clicking the "Open directory" button, browse the subfolder "target", and open the file in Kepler format.
8. Open the output workflow (/target/Example.xml) in Kepler 2.5 to determine whether you, as a data scientist, consider it to be equivalent to that of Taverna Workbench Core 2.5.

Exercise 2

Please complete the following steps:

1. Open the Taverna Workbench Core 2.5 tool to create a workflow with the elements supported in the experiment, or download it from a public repository like myExperiment.org.
2. Execute the steps explained in Exercise 1 to generate the resulting workflow compatible with Kepler 2.5.

3. *Respond to questions 8 to 13*.

Exercise 3
Please complete the following steps:

1. Open the resulting workflows from Exercise 1 and Exercise 2 in the Taverna Workbench Core 2.5 tool.
2. Open the intermediate SWEL model generated by the transformation tool, and analyse its elements considering the SWEL metamodel.
3. *Respond to questions 14 to 18*.

SURVEY:
Please answer the following questions on a scale of 1 (do not agree at all) to 10 (completely agree).

Respondent profile:

1. On a scale of 1 to 10, how much experience do you have in creating and running data-intensive applications?
2. On a scale of 1 to 10, what is your level of knowledge of the terminology and concepts used in data-intensive applications?
3. On a scale of 1 to 10, how much knowledge do you have of different WfMSs that can be used in data-intensive applications?

About the source workflow (Taverna):

4. On a scale of 1 to 10, how long have you used Taverna in the past (1 - Never used it; 10 - Used it for more than 1 year)?
5. On a scale of 1 to 10, how familiar are you with Taverna's basic concepts, components and services?

About the target workflow (Kepler):

6. On a scale of 1 to 10, how long have you used Kepler in the past (1 - Never used it; 10 - Used it for more than 1 year)?
7. On a scale of 1 to 10, how familiar are you with the basic concepts (e.g. workflows, nodes, connections and ports), different types of nodes and modules in Kepler?

About the transformation process:

8. *Efficacy*: The degree to which the artefact achieves its goal considered narrowly, without addressing situational concerns.
   On a scale of 1 to 10, how effective did you find the transformation of workflows from Taverna to Kepler?
9. *Usefulness*: The degree to which the artefact positively impacts the task performance of individuals.
   On a scale of 1 to 10, how much did the transformation improve your ability to create workflows in Kepler based on a Taverna workflow?
10. *Accuracy*: The degree of agreement between outputs of the artefact and the expected outputs.
    On a scale of 1 to 10, how much did you find that the transformation produced accurate results in converting workflows from Taverna to Kepler?

11. *Effectiveness*: The degree to which the artefact achieves its goal in a real situation
    On a scale of 1 to 10, how well did you find that the transformation achieved the objective in a real-world situation, such as using the Kepler-converted workflow?

12. *Validity*: Refers to how well the artefact works, i.e., that it correctly achieves its objective.
    On a scale of 1 to 10, how much do you think the transformation produced valid results when converting workflows from Taverna to Kepler?
13. *Completeness*: The degree to which the activity of the artefact contains all necessary elements and relationships between elements.
    On a scale of 1 to 10, how well did you find that the transformation included all the necessary elements and relationships between elements during the conversion of workflows from Taverna to Kepler?

About the generated SWEL model:

The *morphological layer* contains those elements that enable the low-level definition of a workflow. It is represented as a directed cyclic or acyclic graph, where its vertexes represent elements, and their arcs stand for dependencies between them.

14. On a scale of 1 to 10, to what extent can you identify the nodes and connections defined in the SWEL model?
15. On a scale of 1 to 10, to what extent do you consider that the SWEL model corresponds to the expected structure of a directed graph?

    The *syntactic layer* consists of a set of packages enabling the representation of domain-specific requirements and workflow resources. At this level, elements allow the declaration of control structures, data types, fault-tolerant handlers, domain-specific tasks, and computational resources.
16. On a scale of 1 to 10, to what extent can you identify the domain-specific elements such as data providers and processes in the generated SWEL model?
17. On a scale of 1 to 10, to what extent do you consider that the SWEL model captures the relevant knowledge of the data-intensive domain?

18. On a scale of 1 to 10, to what extent do you consider it relevant for the workflow representation to include descriptive information (meta-information) on the DI project, experiment, or application?

QUESTION	0	1	2	3	4	5	6	7	8	9
0	■									
1										■

(Row “1”, column “3” stands for the response to question “13”, scored with an integer between 1 and 10)

Any other comments? (Free text)

C.2. Resultados

	PID	Occupation	Industry/Academy	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Free text
	012AF	Student (data scier	Academy	3	5	3	1	3	3	5	8	10	8	8	10	10	5	7	10	8	10	
	012CG	Associate professo	Academy	7	8	3	1	2	2	3	10	10	10	10	9	10	10	7	9	9	8	I could read the SWEL's code much more easily than Taverna's code. Although I finally found sufficient information in the SWEL's code to get an idea of the graph, it seems to me that there is some pieces of information, redundant pieces of information, missing. For instance, there was an arc whose target connector was missing and a vertex (the right one from the Taverna's model) indicatint to be the target of previous arc. As said, even though there is enough information to replicate the graph with the code, including the names of the sources and target vertexes in the connectors would have helped a lot in reading the SWEL's code.
	060JT	Data scientist	Industry (Ministry of I	8	7	3	6	8	1	4	10	10	10	10	10	10	8	9	10	10	7	It is highly effective as the result is equivalent to the origin workflow and the tool automatizes the transformation process completely and accurately.
	060PP	Student (data scier	Academy	4	6	2	1	4	1	1	10	8	10	10	10	10	3	5	8	9	10	
	599JM	Data scientist	Industry	6	7	5	1	5	1	5	10	10	10	9	10	8	7	7	5	8	9	Having in mind my low-level knowledge about this domain, I find that the Kepler equivalent of the Taverna workflow generated by this tool, within the constraints of the target software that are not present in the source one, or viceversa, is consistently efficient and sufficiently representative. SWEL is also well structured and fulfills its functions competently.
	599RB	Data scientist	Industry (Ministry of I	8	9	6	3	4	6	9	9	7	9	8	9	9	6	8	7	7	9	
	719AD	Junior data scientis	Industry	5	8	3	1	4	1	3	10	10	10	10	10	10	7	7	7	8	8	The graphical representation of the transformation is very clear, and it is easy to understand with the naked eye which elements of the input correspond to those of the output. Similarly, the steps to be taken to carry out the transformation are very small and immediate.
	719JR	Associate professo	Academy	9	9	9	4	7	7	8	10	9	10	6	10	9	8	10	10	10	10	The only element that I could not find in Kepler after the transformation is the label "Workflow output ports". The rest of elements are perfectly transformed. I was specially surprised by those elements in Taverna that are not visible but transformed and shown in Kepler, such as "hadError" in the process "listPredefinedConceptSets". My response to q11 is because the transformation is limited to a set of elements, not all. But I presume that extending the conversion application might reach fully applicability.
	734FH	Data scientist	Industry	7	7	7	1	1	1	1	10	10	8	8	8	8	9	10	10	10	6	My experience with WfMSs is focused on setting up pipelines for continuous integration and deployment, as well as studying the application of MLOps for machine learning projects. In answers to 10, 11, 12, and 13, I do not have enough knowledge about these specific tools to provide a strong answer, although at first glance, it seems that the transformation contains all the necessary elements and relationships for configuring the workflow.
	734AR	Lecturer (Data scie	Academy	7	8	7	4	7	2	3	10	9	10	9	10	10	10	10	9	9	10	I found the idea of automatic workflow translation very relevant to data scientists. I think it is especially useful in situations where a data scientist collaborates with other data scientists and each usually works with a different tool. The tool is easy to use, it greatly simplifies an otherwise cumbersome process if one does not have knowledge of some tool. Since it is a demo, it is reasonable that the layout of the transformed workflow is not yet perfect. I would recommend adding some logging about the transformation process for expert data scientists who want to know how the process is going. That would help in case a complex workflow has some elements that cannot be transformed. In general, I think the SWEL language is easier to understand than the WfMS specific languages.

Bibliografía

[1] B. Abu-Salih, P. Wongthongtham, D. Zhu, K. Y. Chan, and A. Rudra. *Introduction to Big Data Technology*, pages 15–59. Springer Singapore, 2021.

[2] A. B. A. Alaasam, G. I. Radchenko, and A. N. Tchernykh. Micro-workflows data stream processing model for industrial internet of things. *Supercomputing Front. Inn.*, 8(1):82–98, 2021.

[3] B. Albreiki, N. Zaki, and H. Alashwal. A Systematic Literature Review of Student' Performance Prediction Using Machine Learning Techniques. *Education Sciences*, 11(9):552, 2021.

[4] B. Alkhazi, C. Abid, M. Kessentini, and M. Wimmer. On the value of quality attributes for refactoring atl model transformations: A multi-objective approach. *Information and Software Technology*, 120:106243, 2020.

[5] S. Alturki, I. Hulpuş, and H. Stuckenschmidt. Predicting Academic Outcomes: A Survey from 2007 Till 2018. *Technology, Knowledge and Learning*, 27(1):275–307, 2022.

[6] K. Amin, G. von Laszewski, M. Hategan, N. Zaluzec, S. Hampton, and A. Rossi. GridAnt: a client-controllable grid workflow system. In *Proc. of HICSS'04*, 2004.

[7] E. A. Amrieh, T. Hamtini, and I. Aljarah. Mining educational data to predict student's academic performance using ensemble methods. *International Journal of Database Theory and Application*, 9(8):119–136, 2016.

[8] P. Amstutz et al. Common Workflow Language Description, v1.2. https://w3id.org/cwl/v1.2/, 2020.

[9] C. Ardito, P. Buono, M. F. Costabile, R. Lanzilotti, and A. Piccinno. End users as co-designers of their own tools and products. *Journal of Visual Languages & Computing*, 23(2):78–90, 2012.

[10] B. Bakhshinategh, O. R. Zaiane, S. ElAtia, and D. Ipperciel. Educational data mining applications and tasks: A survey of the last 10 years. *Education and Information Technologies*, 23(1):537–553, 2018.

[11] L. J. Basile, N. Carbonara, R. Pellegrino, and U. Panniello. Business intelligence in the healthcare industry: The utilization of a data-driven approach to support clinical decision making. *Technovation*, page 102482, 2022.

[12] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

[13] D. Bouhineau, S. Lalle, V. Luengo, N. Mandran, M. Ortega, and C. Wajeman. Share data treatment and analysis processes in technology enhanced learning. In *Workshop Data Analysis and Interpretation for Learning Environments*, 2013.

[14] M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice, Second Edition*. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2017.

[15] A. Bucchiarone, A. Cicchetti, F. Ciccozzi, and A. Pierantonio. *Domain-Specific Languages in Practice: with JetBrains MPS*. Springer International Publishing, 2021.

[16] C. I. Büyükbaykal. Communication Technologies and Education in the Information Age. *Procedia - Social and Behavioral Sciences*, 174:636–640, 2015.

[17] R. Buyya, C. Vecchiola, and S. Thamarai Selvi. Chapter 8 - data-intensive computing: Mapreduce programming. In R. Buyya, C. Vecchiola, and S. T. Selvi, editors, *Mastering Cloud Computing*, pages 253–311. Morgan Kaufmann, 2013.

[18] J. Cabot. *Positioning of the Low-Code Movement within the Field of Model-Driven Engineering*. Association for Computing Machinery, New York, NY, USA, 2020.

[19] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 2024.

[20] C. P. Chen and C.-Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275(Supplement C):314–347, 2014.

[21] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang. Programming scientific and distributed workflow with Triana services. *Concurrency and Computation: Practice and Experience*, 18(10):1021–1037, 2006.

[22] A. Cicchetti, D. D. Ruscio, R. Eramo, and A. Pierantonio. Jtl: A bidirectional and change propagating transformation language. In *SLE*, 2010.

[23] T. Clark, A. Evans, P. Sammut, and J. Willans. Applied metamodelling: A foundation for language driven development (third edition). 2015.

[24] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. 01 2001.

[25] A. Coe, G. Paquet, and J. Roy. E-governance and smart communities: A social learning challenge. *Social Science Computer Review*, 19:80–93, 02 2001.

[26] T. Coleman, H. Casanova, L. Pottier, M. Kaushik, E. Deelman, and R. Ferreira da Silva. WfCommons: A framework for enabling scientific workflow research and development. *Future Generation Computer Systems*, 128:16–27, 2022.

[27] P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In *5th FUture BUsiness TEChnology Conference (FUBUTEC)*, pages 5–12. EUROSIS, 2008. https://archive.ics.uci.edu/ml/datasets/student+performance.

[28] J. R. Covvey, C. McClendon, and M. R. Gionfriddo. Back to the basics: Guidance for formulating good research questions. *Research in Social and Administrative Pharmacy*, 2023.

[29] J. W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications Ltd., 3 edition, 2008.

[30] V. Curcin and M. Ghanem. Scientific workflow systems - can one size fit all? In *2008 Cairo International Biomedical Engineering Conference*, pages 1–9, 2008.

[31] M. Dalibor, M. Heithoff, J. Michael, L. Netz, J. Pfeiffer, B. Rumpe, S. Varga, and A. Wortmann. Generating customized low-code development platforms for digital twins. *Journal of Computer Languages*, page 101117, 2022.

[32] L. de la Garza et al. From the desktop to the grid: scalable bioinformatics via workflow conversion. *BMC Bioinform.*, 17:127, 2016.

[33] E. Deelman et al. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015.

[34] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: an overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[35] E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. F. da Silva, G. Papadimitriou, and M. Livny. The evolution of the pegasus workflow management software. *Computing in Science & Engineering*, 21(4):22–36, 2019.

[36] Y. Demchenko, P. Grosso, C. de Laat, and P. Membrey. Addressing big data issues in scientific data infrastructure. In *Proc. of CTS'13*, pages 48–55. IEEE, 2013.

[37] V. Dhar. Data science and prediction. *Communications of the ACM*, 56(12):64–73, 2013.

[38] D. Di Ruscio, R. Eramo, and A. Pierantonio. *Model Transformations*, pages 91–136. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[39] D. Di Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer. Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 21(2):437–446, Apr 2022.

[40] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4):316–319, 2017.

[41] G. Dodig Crnkovic. Scientific methods in computer science. In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, pages 126–130, 2002.

[42] A. Dresch, D. Lacerda, and J. A. Valle Antunes Jr. *Design Science Research: A Method for Science and Technology Advancement*. Springer Cham, 2015.

[43] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. *Selecting Empirical Methods for Software Engineering Research*, pages 285–311. 01 2008.

[44] J. Edwards. Dataset: 2021 CS1 Keystroke Data. https://doi.org/10.7910/DVN/BVOF7S, 2022.

[45] Eibe Frank, Mark A. Hall, and Ian H. Witten. The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, Fourth Edition. https://www.cs.waikato.ac.nz/ml/weka/, 2016.

[46] E. Elmroth, F. Hernández, and J. Tordsson. Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. *Future Generation Computer Systems*, 26(2):245–256, 2010.

[47] F. Emanuel, V. Burriel, and O. Pastor. Data and conceptual model synchronization in data-intensive domains: The human genome case. In S. Cherfi, A. Perini, and S. Nurcan, editors, *Research Challenges in Information Science*, pages 644–650, Cham, 2021. Springer International Publishing.

[48] B. Esmaeilian, S. Behdad, and B. Wang. The evolution and future of manufacturing: A review. *Journal of Manufacturing Systems*, 39:79 – 100, 2016.

[49] T. Fahringer, S. Pllana, and A. Villazon. AGWL: abstract grid workflow language. In *International Conference on Computational Science*, pages 42–49. Springer-Verlag, 2004.

[50] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazon, and M. Wieczorek. *ASKALON: A Development and Grid Computing Environment for Scientific Workflows*, pages 450–471. Springer London, London, 2007.

[51] R. Ferreira da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman. A characterization of workflow management systems for extreme-scale applications. *Future Generation Computer Systems*, 75:228–238, 2017.

[52] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[53] D. García-Saiz, C. Palazuelos, and M. Zorrilla. Data mining and social network analysis in the educational field: An application for non-expert users. In *Educational data mining*, pages 411–439. Springer, 2014.

[54] D. Garijo, P. Alper, K. Belhajjame, O. Corcho, Y. Gil, and C. Goble. Common motifs in scientific workflows: An empirical analysis. *Future Generation Computer Systems*, 36:338–351, 2014.

[55] D. Garijo, Y. Gil, and O. Corcho. Abstract, link, publish, exploit: An end to end framework for workflow sharing. *Future Generation Computer Systems*, 75:271–283, 2017.

[56] S. Getir, M. Challenger, and G. Kardas. The formal semantics of a domain-specific modeling language for semantic web enabled multi-agent systems. *International Journal of Cooperative Information Systems*, 23:1–53, 09 2014.

[57] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, 2007.

[58] Y. Gil, J. Kim, V. Ratnakar, and E. Deelman. Wings for pegasus: A semantic approach to creating very large scientific workflows. 01 2006.

[59] L. Goasduff. Gartner survey finds 72 % of data & analytics leaders are leading or heavily involved in digital transformation initiatives.

2022. https://www.gartner.com/en/newsroom/press-releases/2021-05-05-gartner-finds-72-percent-of-data-and-analytics-leaders-are-leading -or-heavily-involved-in-digital-transformation-initiatives.

[60] J. Goecks, A. Nekrutenko, J. Taylor, and T. G. Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, Aug 2010.

[61] A. Gómez, X. Mendialdua, K. Barmpis, G. Bergmann, J. Cabot, X. de Carlos, C. Debreceni, A. Garmendia, D. S. Kolovos, and J. de Lara. Scalable modeling technologies in the wild: an experience report on wind turbines control applications development. *Software and Systems Modeling*, 19(5):1229–1261, Sep 2020.

[62] I. Gorton and D. K. Gracio. *Data-Intensive Computing: Architectures, Algorithms, and Applications*. Cambridge University Press, 2012.

[63] G. Guizzardi, L. Ferreira Pires, and M. van Sinderen. An ontology-based approach for evaluating the domain appropriateness and comprehensibility appropriateness of modeling languages. In *Proc. of MODELS'05*, pages 691–705. Springer, 2005.

[64] A. T. Guler, C. J. Waaijer, Y. Mohammed, and M. Palmblad. Automating bibliometric analyses using taverna scientific workflows: A tutorial on integrating web services. *Journal of Informetrics*, 10(3):830–841, 2016.

[65] A. Gupta, D. Garg, and P. Kumar. An ensembling model for early identification of at-risk students in higher education. *Computer Applications in Engineering Education*, 30(2):589–608, 2022.

[66] H. Hassani, X. Huang, and E. Silva. Digitalisation and big data mining in banking. *Big Data and Cognitive Computing*, 2(3), 2018.

[67] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28:75–105, 2004.

[68] T. Hey, S. Tansley, K. Tolle, and J. Gray. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, October 2009.

[69] T. Hey and A. E. Trefethen. The UK e-science core programme and the grid. *Future Generation Computer Systems*, 18(8):1017–1031, 2002.

[70] O. Iliashenko, V. Iliashenko, and E. Lukyanchenko. Big data in transport modelling and planning. *Transportation Research Procedia*, 54:900–908, 2021.

[71] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami. Systematic ensemble model selection approach for educational data mining. *Knowledge-based Systems*, 200:105992, 2020.

[72] N. Kahani, M. Bagherzadeh, J. R. Cordy, J. Dingel, and D. Varró. Survey and classification of model transformation tools. *Software & Systems Modeling*, 18(4):2361–2397, Aug 2019.

[73] E. Kalampokis, E. Tambouris, and K. Tarabanis. A classification scheme for open government data: Towards linking decentralised data. *International Journal of Web Engineering and Technology*, 6(3):266–285, 2011.

[74] R. Kazman, S. S. Woods, and S. J. Carrière. Requirements for integrating software architecture and reengineering models: CORUM II. In *Proc. of WCRE'98*, pages 154–163, 1998.

[75] S. Kelly and J. Tolvanen. Collaborative modelling and metamodelling with MetaEdit+. In *Proc. of MODELS'21 Companion*, pages 27–34. IEEE, 2021.

[76] D.-K. Kim, L. Lu, and B. Lee. Design pattern-based model transformation supported by qvt. *Journal of Systems and Software*, 125:289–308, 2017.

[77] G.-H. Kim, S. Trimi, and J.-H. Chung. Big-data applications in the government sector. *Communications of the ACM*, 57(3):78–85, 2014.

[78] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture - Practice and Promise*. Addison Wesley, 2003.

[79] M. Kohl. *Introduction to statistical data analysis with R*. Ventus Publishing ApS, 11 2015.

[80] J. Kranjc, R. Orač, V. Podpečan, N. Lavrač, and M. Robnik-Šikonja. Clowdflows: Online workflows for distributed big data mining. *Future Generation Computer Systems*, 68:38–58, 2017.

[81] J. Kranjc, J. Smailovic, V. Podpecan, M. Grcar, M. Znidarsic, and N. Lavrac. Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the clowdflows platform. *Inf. Process. Manag.*, 51(2):187–203, 2015.

[82] A. Kusiak. Smart manufacturing must embrace big data. *Nature*, 544(7648):23–25, 2017.

[83] J. Kuzilek, M. Hlosta, and Z. Zdrahal. Open university learning analytics dataset. *Scientific Data*, 4:170171, 2017.

[84] K.-W. Lai. Digital technology and the culture of teaching and learning in higher education. *Australasian Journal of Educational Technology*, 27(8):1263–1275, 2011.

[85] P. Langer, M. Wimmer, and G. Kappel. Model-to-model transformations by demonstration. In L. Tratt and M. Gogolla, editors, *Theory and Practice of Model Transformations*, pages 153–167, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[86] P. Lawrence, editor. *Workflow Handbook 1997*. John Wiley & Sons, 1997. With participation of the Workflow Management Coalition.

[87] M. Lebens, R. J. Finnegan, S. C. Sorsen, and J. Shah. Rise of the citizen developer. *Muma Business Review*, 5:101–111, 2022.

[88] D. J. Lemay, C. Baek, and T. Doleck. Comparison of learning analytics and educational data mining: A topic modeling approach. *Computers and Education: Artificial Intelligence*, 2:100016, 2021.

[89] I. E. Livieris, K. Drakopoulou, T. Kotsilieris, V. Tampakas, and P. Pintelas. DSS-PSP - A Decision Support Software for Evaluating Students' Performance. In G. Boracchi, L. Iliadis, C. Jayne, and A. Likas, editors, *International Conference on Engineering Applications of Neural Networks*, pages 63–74. Springer, 2017.

[90] J. López Zambrano, J. A. Lara Torralbo, and C. Romero Morales. Early prediction of student learning performance through data mining: a systematic review. *Psicothema*, 33:456–465, 2021.

[91] D. Loshin. Chapter 3 - Achieving organizational alignment for Big Data analytics. In D. Loshin, editor, *Big Data Analytics*, pages 21–28. Morgan Kaufmann, 2013.

[92] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[93] B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers. *Scientific Workflows: Business as Usual?*, pages 31–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[94] J. M. Luna, C. Castro, and C. Romero. MDM tool: A data mining framework integrated into Moodle. *Computer Applications in Engineering Education*, 25(1):90–102, 2017.

[95] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan. Characteristics and challenges of low-code development: the practitioners' perspective. In *Proceedings of the 15th ACM/IEEE international symposium on empirical software engineering and measurement (ESEM)*, pages 1–11, 2021.

[96] C. Lynch. How do your data grow? *Nature*, 455(7209):28–29, 2008.

[97] J. Martin. *Application Development Without Programmers*. A James Martin book. Prentice-Hall, 1982.

[98] J. Martin. *Rapid Application Development*. Macmillan Publishing Co., Inc., USA, 1991.

[99] J. A. Martínez-Carrascal, D. Márquez Cebrián, T. Sancho-Vinuesa, and E. Valderrama. Impact of early activity on flipped classroom performance prediction: A case study for a first-year engineering course. *Computer Applications in Engineering Education*, 28(3):590–605, 2020.

[100] T. McPhillips, S. Bowers, D. Zinn, and B. Ludäscher. Scientific workflow design for mere mortals. *Future Generation Computer Systems*, 25(5):541–551, 2009.

[101] T. Mens and P. Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142, 2006. Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005).

[102] M. Mijac. Evaluation of design science instantiation artifacts in software engineering research. In *Proc. of CECIIS'19*, pages 313–321. Springer, 2019.

[103] J. Montagnat, T. Glatard, and D. Lingrand. Data composition patterns in service-based workflows. In *2006 Workshop on Workflows in Support of Large-Scale Science (WORKS'06)*, pages 1–10, 2006.

[104] T. Oinn et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[105] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[106] J. A. Parejo. *Moses: A metaheuristic optimization software ecosystem. Applications to the automated analysis of software product lines and service-based applications.* PhD thesis, University of Sevilla, Sevilla, 2013.

[107] E. P. Paul Johannesson. pages XII, 197. Springer, Cham, 2016.

[108] S. Peng, S. Yu, and P. Mueller. Social networking big data: Opportunities, solutions, and challenges. *Future Generation Computer Systems*, 86:1456–1458, 2018.

[109] A. Pescador, A. Garmendia, E. Guerra, J. Sánchez Cuadrado, and J. de Lara. Pattern-based development of domain-specific modelling languages. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 166–175, 2015.

[110] K. Plankensteiner et al. Fine-grain interoperability of scientific workflows in distributed computing infrastructures. *Journal of Grid Computing*, 11(3):429–455, 2013.

[111] M. Prada, M. Domínguez, J. L. Vicario, P. A. V. Alves, M. Barbu, M. Podpora, U. Spagnolini, M. J. V. Pereira, and R. Vilanova. Educational data mining for tutoring support in higher education: A web-based tool case study in engineering degrees. *IEEE Access*, 8:212818–212836, 2020.

[112] B. Predić, G. Dimić, D. Rančić, P. Štrbac, N. Maček, and P. Spalević. Improving final grade prediction accuracy in blended learning environment using voting ensembles. *Computer Applications in Engineering Education*, 26(6):2294–2306, 2018.

[113] Y. Qian, C.-X. Li, X.-G. Zou, X.-B. Feng, M.-H. Xiao, and Y.-Q. Ding. Research on predicting learning achievement in a flipped classroom based on MOOCs by big data analysis. *Computer Applications in Engineering Education*, 30(1):222–234, 2022.

[114] D. E. Rex, J. Q. Ma, and A. W. Toga. The LONI Pipeline processing environment. *NeuroImage*, 19(3):1033–1048, 2003.

[115] C. Richardson and J. Rymer. New development platforms emerge for customer-facing applications. *Forrester Research, Cambridge*, 2014.

[116] P. Ristoski, C. Bizer, and H. Paulheim. Mining the web of linked data with rapidminer. *Journal of Web Semantics*, 35:142–151, 2015.

[117] K. Rokis and M. Kirikova. Challenges of low-code/no-code software development: A literature review. In Ē. Nazaruka, K. Sandkuhl, and U. Seigerroth, editors, *Perspectives in Business Informatics Research*, pages 3–17, Cham, 2022. Springer International Publishing.

[118] C. Romero, J. R. Romero, and S. Ventura. *A Survey on Pre-Processing Educational Data*, pages 29–64. Springer International Publishing, 2014.

[119] C. Romero and S. Ventura. Data mining in education. *WIREs Data Mining and Knowledge Discovery*, 3(1):12–27, 2013.

[120] C. Romero and S. Ventura. Educational data mining and learning analytics: An updated survey. *WIREs Data Mining and Knowledge Discovery*, 10(3):e1355, 2020.

[121] D. D. Roure, C. Goble, J. Bhagat, D. Cruickshank, A. Goderis, D. Michaelides, and D. Newman. myExperiment: Defining the social virtual research environment. In *4th IEEE International Conference on e-Science*, pages 182–189. IEEE Press, 2008.

[122] T. Rubio, M. Chernigovskaya, S. Marquez, C. Marti, P. Izquierdo-Altarejos, A. Urios, C. Montoliu, V. Felipo, A. Conesa, V. Greiff, and S. Tarazona. A nextflow pipeline for t-cell receptor repertoire reconstruction and analysis from rna sequencing data. *ImmunoInformatics*, 6:100012, 2022.

[123] J. Rymer. The Forrester Wave: Low-Code Development Platforms For AD&D Pros. *Forrester Research, Cambridge*, 2017.

[124] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 171–178, 2020.

[125] M. C. Sáiz-Manzanares, J. J. Rodríguez-Díez, J. F. Díez-Pastor, S. Rodríguez-Arribas, R. Marticorena-Sánchez, and Y. P. Ji. Monitoring of Student Learning in Learning Management Systems: An Application of Educational Data Mining Techniques. *Applied Sciences*, 11(6):2677, 2021.

[126] R. Salado-Cid, G. Luque, and J. R. Romero. Sistema de gestión de flujos de trabajo para la definición visual de aplicaciones basadas en algoritmos evolutivos. In *XVI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'15)*, pages 261–270, 2015.

[127] R. Salado-Cid, J. Molino, and J. R. Romero. Interoperabilidad de flujos de trabajo intensivos en datos en industria 4.0: caso de estudio. In *XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'18)*, pages 1333–1338, 2018.

[128] R. Salado-Cid, A. Ramírez, and J. R. Romero. *On the Need of Opening the Big Data Landscape to Everyone: Challenges and New Trends*, pages 675–687. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[129] R. Salado-Cid, A. Ramírez, and J. R. Romero. *On the Need of Opening the Big Data Landscape to Everyone: Challenges and New Trends*, pages 675–687. Springer, 2018.

[130] R. Salado-Cid and J. R. Romero. Lenguaje específico para el modelado de flujos de trabajo aplicados a ciencia de datos. In *XXI Jornadas en Ingeniería del Software y Bases de Datos (JISBD'16)*, pages 227–240, 2016.

[131] R. Salado-Cid and J. R. Romero. Enabling the definition and reuse of multi-domain workflow-based data analysis. In *Proceedings of the 16th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 261–270, 2017.

[132] R. Salado-Cid, A. Vallecillo, K. Munir, and J. R. Romero. SWEL: A domain-specific language for modeling data-intensive workflows. *Business & Information Systems Engineering*, 2023.

[133] R. Salado-Cid, S. Ventura, and J. R. Romero. Metaherramienta para la generación de aplicaciones científicas basadas en workflows. In *X Jornadas de Ciencia e Ingeniería de Servicios (JCIS'14)*, pages 96–105, 2014.

[134] Salado-Cid, R. and Romero, J. R. Material suplementario de esta tesis doctoral. https://doi.org/10.5281/zenodo.10713350, 2024.

[135] D. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, 2006.

[136] B. Selic. The pragmatics of model-driven development. *IEEE Software*, 20(5):19–25, 2003.

[137] J. Seshapanpu. Dataset: Students Performance in Exams. https://www.kaggle.com/datasets/spscientist/students-performance-in-exams, 2018.

[138] R. J. Sethi and Y. Gil. Scientific workflows in data analysis: Bridging expertise across multiple domains. *Future Generation Computer Systems*, 75:256–270, 2017.

[139] D. Simchi-Levi, J. Gadewadikar, B. McCarthy, and L. LaFiandra. Winning with analytics. Technical report, Accenture, 2015. https://www.accenture.com/_acnmedia/Accenture/next-gen/hp-analytics/pdf/Accenture-Linking-Analytics-to-High-Performance-Executive-Summary.pdf.

[140] S. Slater, S. Joksimović, V. Kovanovic, R. S. Baker, and D. Gasevic. Tools for Educational Data Mining: A Review. *Journal of Educational and Behavioral Statistics*, 42(1):85–106, 2017.

[141] S. Smith, D. Cobham, and K. Jacques. The Use of Data Mining and Automated Social Networking Tools in Virtual Learning Environments to Improve Student Engagement in Higher Education. *International Journal of Information and Education Technology*, 12(4), 2022.

[142] P. Stevens. *A Landscape of Bidirectional Model Transformations*, pages 408–424. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[143] A. Szalay and J. Gray. Science in an exponential world. *Nature*, 440(2020 Computing):413–414, 2006.

[144] M. Tam. Measuring quality and performance in higher education. *Quality in Higher Education*, 7(1):47–54, 2001.

[145] A. Tayebi, J. Gómez, and C. Delgado. Analysis on the Lack of Motivation and Dropout in Engineering Students in Spain. *IEEE Access*, 9:66253–66265, 2021.

[146] J. B. Tera Allas and, M. Chui, P. Dahlström, E. Hazan, N. Henke, S. Ramaswamy, and M. Trench. Artificial intelligence is getting ready for business, but are businesses ready for AI? In *Analytics comes of age*, pages 18–34. McKinsey Analytics, 2018.

[147] F. Truyen. *The Fast Guide to Model Driven Architecture*. Cephas Consulting Corp., 2006. This document is based on the OMG's MDA Guide, omg/2003-06-01.

[148] M. Turck. Red Hot: The 2021 Machine Learning, AI and Data (MAD) Landscape. 2021. https://mattturck.com/data2021/.

[149] W. v. d. Aalst and E. Damiani. Processes meet big data: Connecting data science with process science. *IEEE Transactions on Services Computing*, 8(6):810–819, 2015.

[150] F. Villarroel Ordenes and R. Silipo. Machine learning for marketing on the knime hub: The development of a live repository for marketing applications. *Journal of Business Research*, 137:393–410, 2021.

[151] J. vom Brocke, A. Hevner, and A. Maedche. *Introduction to Design Science Research*, pages 1–13. Springer International Publishing, Cham, 2020.

[152] M. Wang, L. Zhu, and Z. Zhang. Risk-aware intermediate dataset backup strategy in cloud-based data intensive workflows. *Future Generation Computer Systems*, 55:524–533, 2016.

[153] R. Waszkowski. Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, 52(10):376–381, 2019. 13th IFAC Workshop on Intelligent Manufacturing Systems IMS 2019.

[154] M. Weske, W. M. P. van der Aalst, and H. M. W. Verbeek. Advances in business process management. *Data Knowledge Engineering*, 50(1):1–8, July 2004.

[155] M. Weske and G. Vossen. Workflow languages. In P. Bernus, K. Mertins, and G. Schmidt, editors, *Handbook on Architectures of Information Systems*, International Handbooks on Information Systems, pages 359–379. Springer Berlin Heidelberg, 1998.

[156] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer, 2012.

[157] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3):171–200, Sep 2005.

[158] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3):171–200, 2006.

[159] J. Yu and R. Buyya. *Gridbus Workflow Enactment Engine*, pages 119–146. CRC Press, 2009.

[160] Y. Zhao et al. Swift: Fast, Reliable, Loosely Coupled Parallel Computation. In *Proc. of SCW'07*, pages 199–206. IEEE Computer Society, 2007.

[161] Y. Zhou and X. Chen. Simulation of sports big data system based on markov model and iot system. *Microprocessors and Microsystems*, 80:103525, 2021.