U.N.L.P.

Universidad Nacional de La Plata

Facultad de Informática

Una propuesta de transformación M2M para el análisis de la fase ASM de MoWebA

Tesis presentada para obtener el grado de

Magíster en Ingeniería de Software.

Autor:
Daniel Bonhaure

Director:
Ph.D. Claudia Pons
Asesor Científico:
Ing. Magalí González
Codirector:
M.Sc. Nathalie Aquino

Junio, 2017

Una propuesta de transformación M2M para el análisis de la fase ASM de MoWebA

Lic. Daniel Pierre Bonhaure Falcón

Junio, 2017

Dedicatorias

A mis amores!

A mis padres y hermanos!

A Pepé y todos mis abuelos!

A Javier y toda la familia!

Agradecimientos

- A mis tutores y co-tutores, por guiarme en este largo camino.
- A Luca Cernuzzi, por sus valiosos aportes.
- A mis padres, sin su apoyo y aliento esto no habría sido posible.
- A mis amores, Carmencita y Franchu, por ser mis motorcitos.
- A mis hermanos y toda la flia., por el amor y apoyo incondicional.
- A Jacob y Deicy, por recibirme siempre.
- A mis compañeros de "La Pecera", por los buenos momentos.

El presente trabajo ha sido desarrollado con el apoyo financiero del Consejo Nacional de Ciencia y Tecnología (CONACYT, Paraguay) en el marco del proyecto denominado "Mejorando el proceso de desarrollo de software: propuesta basada en MDD" (14-INV-056); el cual es llevado adelante en conjunto entre el Departamento de Electrónica e Informática (DEI) de la Universidad Católica "Nuestra Señora de la Asunción" y el Centro de Investigación en Métodos de Producción de Software (PROS) de la Universidad Politécnica de Valencia, España.

Resumen

En el escenario de las metodologías de ingeniería web [1] [2], existe una tendencia actual a seguir el enfoque MDD [3], adoptando MDA. En MDA, la transformación de modelos es esencial. Por lo tanto, en lugar de generar el código directamente desde el modelo conceptual (transformación modelo a texto), los modelos conceptuales, es decir, los Modelos Independientes de la Plataforma (PIM), se traducen a uno o más Modelos Específicos de Plataforma (PSM) mediante transformaciones modelo a modelo (M2M). Posteriormente, los PSM se traducen a código mediante técnicas de transformación modelo a texto (M2T).

Una de las estrategias que MDA promueve para facilitar el cambio y la portabilidad del modelo conceptual es la prescripción del PIM separado del PSM. Sin embargo, en la práctica, las metodologías web actuales suelen enfrentar las tendencias de evolución arquitectónica extendiendo sus notaciones de modelado directamente a nivel de PIM (e.g., WebML RIA [4], UWE para RIA [5], OOH4Ria [6], entre otros). Las propuestas MDD para el desarrollo de aplicaciones móviles también siguen esta línea (e.g., WebRatio [3], MD2 [7], MobiCloud [8], entre otros). Tal extensión generalmente resulta en un PIM enriquecido que incluye características y restricciones de una determinada arquitectura o plataforma específica. Los PIM enriquecidos se obtienen añadiendo marcas (pero sin considerarlos PIM marcados), o añadiendo nuevos modelos enteros, o incluso, añadiendo nuevos elementos a la notación del PIM. Aunque se podría argumentar que después de extender el PIM para dar soporte a una arquitectura específica, este ya no es un PIM, sino un PSM, en general, la literatura no analiza esta distinción.

El problema es que esta tendencia de adaptar el PIM a arquitecturas específicas va en contra del principio de portabilidad del PIM promovido por MDA. El PIM enriquecido pierde parte de su independencia de la arquitectura, y se hace cada vez más complejo de entender y manejar. La consecuencia, pérdida de portabilidad y de la capacidad de reutilización de los modelos conceptuales.

Un problema de portabilidad complementario surge cuando las metodologías web tienden a enriquecer el PSM con detalles arquitectónicos, ya que la misma arquitectura puede implementarse en diferentes plataformas (e.g., RIA puede implementarse en Backbase, Dojo, GWT, jQuery, entre otros). En este caso el problema no es del modelo conceptual, sino que, para la misma arquitectura, se requieren múltiples PSM de acuerdo a las diferentes plataformas de implementación.

Para preservar la independencia del PIM y la portabilidad de los modelos hacia diferentes arquitecturas, algunos autores ya han propuesto la introducción de un Modelo Específico de la Arquitectura (ASM) (e.g., [9], [10]). El ASM permite que los

detalles relacionados con la arquitectura se muevan del espacio de problemas (PIM) a un modelo intermedio en el espacio de solución (ASM), evitando que el PIM los contenga y contribuyendo así, a su tan ansiada portabilidad. Además, el ASM puede resultar en diferentes PSM, uno por cada plataforma de implementación.

En este trabajo presentamos cómo la adopción del ASM apoya la portabilidad del PIM y mantiene el modelo arquitectónico independiente de la plataforma de implementación. Para el efecto, adoptamos MoWebA, con sus reglas de transformación M2M que permiten la correcta generación de múltiples ASM partiendo del mismo PIM y múltiples PSM partiendo del mismo ASM.

Así, el alcance de este trabajo es presentar las reglas de transformación PIM-ASM definidas en MoWebA para dos arquitecturas diferentes (Aplicaciones Enriquecidas de Internet y móviles) y mostrar su relevancia en pos de la reducción del problema de la portabilidad del PIM. Hemos seleccionado RIA y arquitecturas móviles debido a la relevancia y el impacto actual de ambas.

Con este trabajo se logró comprobar que mediante transformaciones M2M es posible obtener modelos ASM para diferentes arquitecturas partiendo de un único PIM sin necesidad de modificarlo previamente. A pesar de que sólo fueron considerados dos modelos ASM (i.e., uno para RIA y otro para persistencia móvil), se concluyó que es completamente posible obtener modelos ASM para otras arquitecturas.

La principal ventaja de obtener varios modelos ASM partiendo de un único modelo PIM es que la portabilidad del PIM se conserva de manera considerable y significativa. Es importante recordar que este enfoque (es decir, la captura de los elementos arquitectónicos en el modelo ASM), es muy diferente al de la mayoría de las metodologías actuales que, a diferencia de MoWebA, tienden a agregar los elementos específicos de la arquitectura al PIM (mediante la adaptación/extensión del mismo).

Adicionalmente, este estudio muestra que mediante el uso de archivos de configuración externos es posible mejorar en gran medida el grado de automatización de las transformaciones PIM-ASM, más aún cuando es posible procesar estos archivos invocando código Java nativo.

Palabras clave: Desarrollo Dirigido por Modelos, Arquitectura Dirigida por Modelos, Modelo Específico de la Arquitectura, Transformación Modelo a Modelo, Aplicaciones de Internet Enriquecidas, MoWebA.

Abstract

In the scenery of web engineering methodologies [1] [2], there is a current trend on following the Model-Driven Development (MDD) approach [3], adopting the Model-Driven Architecture (MDA). In MDA, model transformations are essential. Thus, instead of directly generate the code from the conceptual model (model-to-text transformation), the conceptual models, i.e. Platform Independent Models (PIMs), are translated to one or more Platform Specific Models (PSMs) by means of model-to-model (M2M) transformations. Subsequently, PSMs are translated into code by means of model-to-text (M2T) transformation techniques.

One of the strategies that MDA promotes to facilitate changeability and portability of the conceptual model is the prescription of the PIM separated from the PSM. However, in practice, current web methodologies tend to cope with architectural evolution trends by extending their modeling notations directly at the PIM level (e.g., WebML RIA [4], UWE for RIA [5], OOH4Ria [6], among others). MDD proposals for mobile applications development also follow this landscape. Examples of this are WebRatio [3], MD2 [7], MobiCloud [8], among others. Such extension usually results in an enriched PIM that includes characteristics and constraints of a certain specific architecture or platform. The enriched PIMs are obtained either by adding marks (but without considering this as a marked PIM), or by adding new entire models, or even by adding new elements in the PIM notation. Although it could be argued that after extending a PIM to support a specific architecture it is no longer a PIM, but a PSM, in general, the literature does not analyze this distinction.

The problem is that this tendency of adapting the PIM to specific architectures goes against the principle of the PIM portability promoted by MDA. The enriched PIM loses part of its independence from the architecture/platform, and becomes increasingly more complex to understand and manage. The consequence is a loss of portability and reusability of the conceptual models.

A complementary portability problem arises when web methodologies tend to enrich the PSM with architectural details, since the same architecture can be implemented in different platforms (e.g., the RIA architecture can be implemented in Backbase, Dojo, GWT, jQuery, among others). In this case, the problem is not concerning the conceptual model, but it requires for the same architecture multiple PSMs according to the different implementation platforms.

To preserve the independence of the PIM and the portability of models towards different architectures, some authors have already proposed the introduction of an Architectural Specific Model (ASM) (e.g., [9], [10]). The ASM allows details related to the architecture to be moved out from the problem space (PIM) to an intermediate

model in the solution space (ASM), avoiding the PIM from containing such details and contributing to its so desired portability. Moreover, the ASM model can result in different PSM models, one per each implementation platform.

In this work we present how the adoption of ASM supports the PIM portability and maintains the architectural model independent from the implementation platform. For the effect, we adopted MoWebA with its M2M transformations rules that allow the correct generation of different ASMs from the same PIM model and different PSMs from the same ASM.

Thus, the scope of this work is to present the PIM-ASM transformation rules defined in MoWebA for two different architectures (Rich Internet Applications and mobile) and show its relevance to reduce the PIM portability problem. We selected RIA and mobile architectures due the relevance and current impact of both.

With this work, it was verified that M2M transformations allow ASM models for different architectures to be obtained from a single PIM model without modifying it. Despite only two ASM models have been considered (i.e., one for the RIA architecture and another for mobile persistence), it was concluded that it is completely possible to obtain ASM models for other different architectures.

The main advantage of obtaining several ASM models from a single PIM model is that the PIM portability is considerably and significantly preserved. It is important to remember that this approach (i.e., the capture of architectural elements in the ASM model) is very different from the approach of most current methodologies that, unlike MoWebA, tend to add specific elements of the architecture to the PIM (through the adaptation/extension of it).

Additionally, this study shows that by using external configuration files it is possible to greatly improve the degree of automation of PIM to ASM transformations, even more when it is possible to process these files by invoking Java native code.

Keywords: Model-Driven Development, Model-Driven Architecture, Architectural Specific Model, Model-to-Model Transformation, Rich Internet Applications, MoWebA.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El Desarrollo de Software Dirigido por Modelos (MDD - del inglés "Model-Driven Development") es una disciplina, dentro de la ingeniería del software, que de un tiempo a esta parte ha adquirido gran importancia y ha logrado significativos avances, al punto de haberse convertido en un paradigma de desarrollo de software [11].

MDD propone el uso de un mayor nivel de abstracción en la especificación tanto del problema como de la solución, en relación con los métodos tradicionales de desarrollo de software [11]. La idea central detrás de este paradigma es la de lograr el mayor grado posible de automatización en la generación de código final, utilizando para ello los modelos generados en la etapa de diseño.

Por otro lado, la Arquitectura Dirigida por Modelos (MDA - del inglés "Model-Driven Architecture"), propuesta de la OMG (Object Management Group), es la propuesta más extendida para MDD [11]. Enmarcado dentro de lo que sería este framework encontramos a MoWebA (Model Oriented Web Approach); Un enfoque metodológico MDD para el desarrollo de aplicaciones WEB que adopta el estándar MDA en sus diferentes fases [9].

Un aporte diferenciador de MoWebA es la inclusión de un nuevo nivel de abstracción a los ya definidos por el framework MDA. Este nuevo nivel lo constituye el Modelo Específico de la Arquitectura (ASM - del inglés "Architecture Specific Model") [12], que permite, entre otras cosas, omitir la definición de Modelos Independientes de la Plataforma (PIM - del inglés "Platform-Independent Model ") específicos para cada tipo de arquitectura, fortaleciendo de esta manera la portabilidad de los mismos.

Además, como la adopción de MoWebA; y por ende, de los modelos ASM; exige la definición de una arquitectura de trabajo, este proyecto se enfoca en la arquitectura RIA (Rich Internet Application), la que soporta la creación de sistemas web con características similares a las de las aplicaciones de escritorio [9].

1.1. Motivación

La importancia adquirida por MDD dentro del campo de la ingeniería del software se debe en parte a las promesas realizadas por este paradigma y en parte a la necesidad de la comunidad de encontrar nuevas formas de enfrentar los problemas cada vez más complejos que conlleva la producción de software.

Si bien los problemas per se no cambiaron mucho desde que iniciara la *crisis*

del software[1], la complejidad de estos no hizo más que ir en aumento ya que, por ejemplo, cada vez se acortan más los tiempos disponibles para producción; aumentan las exigencias en cuanto a confiabilidad, usabilidad, robustez y demás características del software; aumenta la complejidad de las funcionalidades demandas (hasta el punto de requerir complicadas interacciones con el medio ambiente, la realidad aumentada en teléfonos inteligentes es un claro ejemplo de esto).

El enfoque MDD es una de las propuestas actuales que buscan enfrentar esta, aún vigente, crisis del software.

Emulando a los lenguajes de alto nivel e inspirado en su éxito, MDD propone el uso de un mayor nivel de abstracción en la especificación tanto del problema como de la solución, en relación con los métodos tradicionales de desarrollo de software [11].

En un primer momento los programas se escribían usando lenguajes de bajo nivel como el Assembler. Más tarde se crearon lenguajes de alto nivel, incluso, actualmente existen lenguajes de tan alto nivel que tienen un parecido considerable con el lenguaje natural y el hecho de que éstos son traducidos a programas escritos en lenguajes de bajo nivel es algo que pasa totalmente desapercibido. Es más, actualmente se considera que los compiladores son tan buenos que difícilmente un programador logre código tan eficiente como el producido por estos.

La gran promesa de MDD es lograr lo mismo, pero basándose completamente en modelos, es decir, realizando un trabajo similar al compilador pero partiendo de modelos en lugar de código en algún lenguaje de alto nivel. Una meta ambiciosa!!, sin embargo, en su momento también se consideró ambiciosa la meta propuesta por los lenguajes de alto nivel.

El enfoque MDD es la evolución natural de la ingeniería de software basada en modelos, más específicamente, del Desarrollo Basado en Modelos (MBD - del inglés "Model-Based Development"), la cual es enriquecida mediante el agregado de transformaciones automáticas entre modelos. De hecho, el adjetivo "dirigido" (driven) en MDD, a diferencia de "basado" (based) en MBD, enfatiza el hecho que en este paradigma los modelos son sumamente importantes, tanto o más que el código fuente, ya que la idea central es lograr el mayor grado posible de automatización a la hora de generar código, partiendo de los modelos generados en la etapa de diseño.

Por lo tanto, MDD plantea una forma completamente nueva de entender el desarrollo y mantenimiento de sistemas de software donde los modelos son utilizados para dirigir las tareas de comprensión, diseño, construcción, pruebas, despliegue, operación, administración, mantenimiento y modificación de los sistemas [11].

Existen varias propuestas concretas para MDD. Una de las más extendidas actualmente es la propuesta MDA, propuesta para MDD desarrollada por el Object Management Group y promovida a partir del 2000. El framework MDA es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos, siguiendo el proceso MDD [11].

La idea clave detrás de MDA es la de separar la lógica de negocio, la funcionalidad de una aplicación, de su implementación en una plataforma determinada. De este modo, el desarrollador crea modelos de alto nivel e independientes de la plataforma (PIM), que posteriormente, mediante transformaciones modelo a modelo (M2M),

[1] término acuñado en la primera conferencia organizada por la OTAN sobre desarrollo de software en 1968

generan de manera automática modelos dependientes de la plataforma (PSM). Y finalmente, al igual que en MDD, el código de la aplicación es derivado de los modelos específicos de la plataforma (PSM) mediante transformaciones modelo a texto (M2T) [13].

Al igual que MDD, MDA también considera modelos CIM, modelos independientes de cualquier detalle relacionado a la computación y que se refieren a aspectos del negocio. Sin embargo, la automatización de la transformación CIM a PIM es un problema complicado debido al salto semántico que los separa [13].

Enmarcado dentro de lo que sería el framework MDA encontramos a MoWebA; un enfoque metodológico que se define en González et al. [14] como una aproximación basada en modelos para el desarrollo a aplicaciones Web que, además de definir elementos relacionados al proceso de desarrollo del software, contempla aspectos metodológicos (etapas, productos, dimensiones) y los contempla en un entorno que incluye herramientas de modelado y generación automática de código, uso de estándares, arquitectura robusta a través de la separación de conceptos, etc.

Una de las problemáticas abordadas por MoWebA se hace evidente al analizar la adopción, dentro de MDD, de las arquitecturas RIA. Resulta que un gran número de las metodologías MDD existentes en la actualidad adoptan las arquitecturas RIA extendiendo sus modelos PIM, más específicamente, la notación de sus modelos PIM, con primitivas adicionales o patrones [9]. Si bien estos cambios derivan en metodologías muy similares a las originales, estas son, en esencia, nuevas metodologías. Evidencia de esto es que, por ejemplo, WebML define WebML RIA [15]; UWE, UWE para RIA [5]; OO-H, OOH4RIA [6]; etc.

Esta necesidad de definir modelos PIM específicos para arquitecturas diferentes va en contra del principio de portabilidad del PIM pretendido tanto por MDD como por MDA, y es una de las problemáticas abordadas por MoWebA. En concreto, el problema de portabilidad de los modelos PIM es abordado por MoWebA mediante la inclusión de un nuevo nivel de abstracción a los ya establecidos por MDA.

Este nuevo nivel de abstracción lo constituye el modelo ASM. El ASM está conformado por un modelo intermedio entre el PIM y el PSM cuya finalidad es la de enriquecer los modelos previos con información adicional referente a la arquitectura del sistema [9]. Arquitecturas como RIA, SOA, REST son ejemplos claros de las arquitecturas referidas.

Al capturar toda la información referente a la arquitectura, el modelo ASM propuesto por MoWebA permite trasladar la definición de la arquitectura, desde el dominio del problema, donde se sitúa cuando es definida en el PIM, al dominio de la solución, donde se sitúa cuando es definida en el ASM y, lo más importante, libera a los modelos PIM de contener esta información. La separación planteada permite alcanzar la tan ansiada portabilidad del PIM, contribuyendo incluso a la evolución del software al permitir modelos PIM más legibles y por ende fáciles de mantener.

El problema con el enfoque adoptado por las metodologías que, a diferencia de MoWebA, ubican la definición de la arquitectura del sistema en el dominio del problema (i.e., en el PIM) en lugar de en el dominio de la solución (i.e., en el ASM), es que cuando se requiere un cambio de arquitectura, el mencionado enfoque obliga a re-escribir todos los modelos PIM del sistema, algo que no es obligatorio en el enfoque planteado por MoWebA. A fin de comprender mejor la situación, a continuación se

plantea un ejemplo práctico; si utilizando alguna de las metodologías tradicionales (WebML, UWE, etc), se pretende desarrollar aplicaciones bajo diferentes tipos de arquitectura (por ejemplo RIA, SOA y REST), será necesario utilizar tres definiciones diferentes de PIM, una orientada a RIA (WebML RIA [4], UWE para RIA [5]), otra orientada a SOA (WebML SOA, UWE para SOA -si existiesen-), y otra orientada a REST. Lo que, entre otras cosas, implica la necesidad de crear tres modelos PIM diferentes, triplicando trabajo.

Si bien la separación de conceptos es considerada como beneficiosa por la comunidad toda y, al menos en principio, la inclusión de la fase ASM definida por MoWebA conlleva enormes beneficios para el ciclo de desarrollo propuesto por MDD, no existen aún suficientes estudios que validen estas afirmaciones o permitan evaluar el grado de automatización M2M alcanzado luego de la inclusión de esta nueva fase.

Justamente esto es lo que se pretende con la realización del presente trabajo de tesis. Sin embargo, antes de llevar adelante un análisis de este tipo, es necesario establecer reglas de transformación que permitan traducir modelos PIM a modelos ASM para alguna(s) arquitectura(s) actualmente relevante(s), ya que MoWebA aún no dispone de reglas de transformación PIM-ASM para ninguna.

1.2. Objetivos

En el marco del enfoque MoWebA, el objetivo general del presente trabajo de tesis se centra en analizar algunos aspectos relevantes del nuevo nivel de abstracción planteado por esta propuesta, el modelo ASM.

En pos de la consecución de este objetivo será necesario avanzar sobre los siguientes objetivos específicos.

- Identificar una(s) arquitectura(s) relevante(s) para la especialización del ASM.
- Definir reglas de transformación que permitan transformar modelos PIM a modelos ASM para la(s) arquitectura(s) seleccionada(s).
- Validar el proceso de transformación PIM-ASM, garantizando que las reglas definidas se ajusten adecuadamente tanto al enfoque MoWebA como a la(s) arquitectura(s) seleccionada(s).
- Identificar un(unos) aspecto(s) relevante(s) del ASM para el análisis de este nuevo nivel de abstracción.
- Definir y llevar adelante experiencias que aporten datos relevantes sobre el(los) aspecto(s) del ASM a ser analizado(s).

Una vez finalizado el presente trabajo de tesis, MoWebA contará con reglas de transformación M2M para la transformación automática de modelos PIM a modelos ASM, lo que será un gran aporte tanto para MoWebA como para la comunidad toda. Además, se generarán datos y experiencias que permitirán validar los aportes de la inclusión del modelo ASM al ciclo de desarrollo MDD, permitiendo hacer luz sobre lo beneficiosa o no de esta inclusión.

1.3. Propuesta

Una vez finalizada la revisión bibliográfica llevada a cabo en el marco del presente trabajo de tesis y detallada en el capítulo 3, se observó cierta tendencia de la comunidad científica a dejar de lado algunos aspectos clave de MDD: i) las transformaciones M2M, ii) la portabilidad del PIM, iii) el desarrollo de herramientas que implementen transformaciones M2M, iv) las validaciones formales y, sobre todo, v) a otros enfoques MDD (puesto que es más común proponer un nuevo enfoque MDD desde cero, que mejorar o extender alguno pre-existente).

Una vez identificadas estas problemáticas, y a sabiendas de la existencia de MoWebA, un enfoque metodológico cuya adopción permite abordarlas; considerando además las conclusiones del SMS llevado a cabo como parte de la revisión bibliográfica del presente trabajo de tesis, fundamentalmente aquella que resalta la existencia de muchas propuestas inconclusas y/o pendientes de validación y la necesidad de seguirlas estudiando; resulta claro que la propuesta de solución más apropiada sería la adopción de MoWebA como base para extenderla/completarla.

Confrontando además, la nueva fase propuesta por MoWebA (el ASM), con la problemática identificada en la sección 1.1 (el problema de portabilidad del PIM), es posible notar que la utilización de esta etapa intermedia entre el PIM y el PSM parece devolver a los modelos PIM su tan ansiada portabilidad, ya que, por ejemplo, permite la definición de un solo modelo PIM para desarrollar las tres variantes del sistema planteado en el ejemplo presentado en la parte final de la sección 1.1. Además, al librar al PIM de la definición de aspectos relacionados con la arquitectura, mejora su legibilidad, y por ende, su mantenibilidad.

Sin embargo, MoWebA, uno de los enfoques existentes dentro de lo que sería MDD orientado a arquitecturas RIA, todavía no es capaz de automatizar las transformaciones M2M requeridas para transformar un PIM a un ASM y este a un PSM. Y por lo tanto, las ventajas y/o desventajas que pudiera arrojar la inclusión del ASM al ciclo de desarrollo, sobre todo las ventajas prometidas al plantearse su inclusión, aún no fueron debidamente estudiadas y validadas.

En resumen, la solución propuesta consiste, en primer lugar, en la adopción de MoWebA como metodología base. El objetivo perseguido con esto es contribuir con una metodología MDD existente e inconclusa, en vez de proponer una nueva.

En segundo lugar, y como algo imperativo para la consecución del objetivo final de esta tesis (analizar aspectos relevantes del ASM), la solución propuesta consiste en la definición de reglas de transformación que permitan automatizar la transformación PIM-ASM para la(s) arquitectura(s) seleccionada(s), y además, en la definición de un proceso de desarrollo que especifique las herramientas a ser utilizadas y los pasos a seguir para la ejecución de estas reglas. En este punto es importante remarcar que la correcta definición de estas reglas de transformación es crucial para evitar sesgos o desviaciones en los análisis posteriores.

De esta manera, y en tercer lugar, una vez estén completamente definidas y validadas las reglas de transformación, se llevarán a cabo distintas experiencias, recabando de ellas la mayor cantidad de datos posible. Datos que posteriormente serán analizados rigurosamente a la luz de el(los) aspecto(s) del ASM seleccionado(s) como objeto(s) de análisis.

Las arquitecturas abordadas en el análisis son RIA y persistencia móvil. Ambas fueron seleccionadas debido a su relevancia actual. Sin embargo, RIA es adoptada como arquitectura principal. Con ella se busca i) identificar el grado máximo de automatización posible, y ii) cubrir el ciclo de desarrollo completo, desde el PIM hasta el código. La persistencia móvil, por otro lado, es abordada únicamente con el objetivo de verificar si es posible o no obtener varios modelos ASM a partir de un único modelo PIM y, posteriormente, analizar las implicancias de esto.

Por otro lado, en base las conclusiones obtenidas en el SMS, sobre todo las relacionadas al problema de portabilidad del PIM, los aspectos del ASM seleccionados como objeto de análisis fueron los siguientes: i) la posibilidad de generar múltiples ASM a partir de un único PIM, y ii) el grado de portabilidad del PIM alcanzado como consecuencia de la utilización del ASM.

1.3.1. Escenario motivador

Supongamos el desarrollo de un sistema siguiendo alguna metodología MDD tradicional (e.g., WebML, UWE, etc), es decir, alguna que no contemple una etapa intermedia entre el PIM y el PSM que se encargue de capturar los detalles referentes a la arquitectura adoptada. Un buen ejemplo sería un sistema web tradicional, estático, punto de partida de las mayoría de los enfoques MDD orientados al desarrollo de aplicaciones WEB.

Asumiendo los supuestos del párrafo anterior, el proceso de desarrollo del sistema sería el siguiente: primero se crearían manualmente los modelos PIM, los cuales serían transformados a modelos PSM mediante transformaciones M2M automáticas. Posteriormente, estos PSM serían transformados a código mediante transformaciones M2T automáticas. El único proceso manual aquí sería la creación de los PIM.

En concreto, la problemática abordada se manifiesta cuando el sistema anterior requiere ser modificado. Específicamente, cuando es necesario cambiar la arquitectura del sistema, por el motivo que fuere.

En este punto se pueden dar dos situaciones bastante diferentes, dependiendo de la metodología MDD que haya sido adoptada para el desarrollo del sistema.

Si la metodología adoptada captura los detalles referentes a la arquitectura en el PIM, entonces para realizar el cambio requerido se tendría que seguir el siguiente ciclo: primero se tendría que volver a crear el PIM, esta vez considerando los conceptos referentes a la nueva arquitectura, es decir, adoptando el PIM específico para la arquitectura deseada. Es importante notar que este es un proceso manual exactamente igual al realizado al desarrollar el sistema por primera vez (a pesar que ahora solo lo estamos modificando). El siguiente paso consistiría en transformar estos PIM en PSM mediante transformaciones M2M automáticas. Para, finalmente, transformar estos PSM a código mediante transformaciones M2T, también automáticas.

Consideremos ahora el caso en el que la metodología adoptada captura los detalles referentes a la arquitectura en un modelo intermedio entre el PIM y el PSM. Consideremos, puntualmente, el modelo ASM de MoWebA. En este caso, el proceso necesario para realizar el cambio requerido sería completamente automático ya que se reutilizaría el PIM pre-existente y se lo transformaría, mediante transformaciones M2M automáticas, en modelos ASM. Luego, estos ASM serían transformados en modelos PSM, también mediante transformaciones M2M automáticas. Finalmente, estos

PSM serían transformados a código mediante transformaciones M2T automatizadas.

Comparando ambos procesos de modificación, es posible conjeturar que la existencia del modelo ASM mejora los tiempos de desarrollo debido a que el cambio de arquitectura, al utilizarlo, resultó ser un proceso más automatizado. Por el contrario, la modificación utilizando PIM específicos requirió que los modelos PIM sean re-escritos de manera completamente manual, lo que implica un trabajo similar al llevado a cabo cuando el sistema fue desarrollado por primera vez.

1.4. Publicaciones

Esta tesis fue desarrollada en el marco de un proyecto más amplio[2] que tiene entre otros objetivos, explorar los beneficios de la aplicación de MDD. A continuación se mencionan publicaciones relacionadas a esta tesis que fueron el resultado de investigaciones dentro de dicho proyecto.

Model-to-model transformations for RIA architectures: A systematic mapping study Autores: Daniel Bonhaure, Magalí González, Nathalie Aquino, Luca Cernuzzi y Claudia Pons. Publicado en: XLII Conferencia Latinoamercana de Informática (CLEI 2016) [16].

Dicho estudio, fue seleccionado posteriormente para la publicación de una versión extendida en idioma inglés en el CLEI electronic journal. *Exploring Model-to-Model Transformations for RIA Architectures by means of a Systematic Mapping Study* Autores: Daniel Bonhaure, Magalí González, Nathalie Aquino, Luca Cernuzzi y Claudia Pons. Publicado en: CLEI electronic journal (to appear) [17].

1.5. Organización

El Capítulo 2 presenta las bases teóricas del presente trabajo de tesis. En este capítulo se presentan las bases conceptuales del Desarrollo Dirigido por Modelos (MDD) y de su propuesta más extendida, la Arquitectura Dirigida por Modelos (MDA). Así también, se presenta a MoWebA, un enfoque metodológico MDD para el desarrollo de aplicaciones WEB que adopta el estándar MDA en sus diferentes fases, y al modelo ASM propuesto esta. Sobre el final del capítulo se presenta a las arquitecturas RIA y de persistencia móvil, junto con algunos trabajos relacionados con este estudio.

El Capítulo 3 presenta el Mapeo Sistemático de la Literatura (SMS) llevado a cabo como parte de la revisión bibliográfica del presente trabajo de tesis, y que resultó fundamental para la identificación de la problemática abordada y el planteamiento de la solución propuesta. La problemática abordada se presenta sobre el final del capítulo, mientras que la solución propuesta es planteada y desarrollada un capítulo más adelante.

El Capítulo 4 presenta la solución propuesta y la implementación de la misma. Luego de ser presentadas algunas consideraciones preliminares, en este capítulo se detalla el proceso de desarrollo propuesto junto con todos los aspectos relacionados a la definición y ejecución de las reglas de transformación M2M planteadas, aspectos como el lenguaje utilizado para la definición de las reglas, los metamodelos y perfiles

[2] `http://www.dei.uc.edu.py/proyectos/mddplus/`

utilizados, el mapeo entre elementos del PIM y del ASM, las reglas de transformación derivadas de este mapeo, los archivos de configuración propuestos, y finalmente, una breve descripción de los desafíos abordados.

El Capítulo 5 presenta la validación de la solución propuesta. La validación consiste en la aplicación de las reglas de transformación M2M a un caso práctico real. El caso práctico contempla la creación de un PIM para un sistema de marcación de empleados, su posterior transformación a ASM mediante las reglas de transformación M2M definidas en este trabajo, y finalmente, la posibilidad de obtener código mediante las reglas de transformación M2T desarrolladas en trabajos previos que abordan la generación de código desde el ASM.

El Capítulo 6 presenta las conclusiones del presente estudio, abordando entre otras cosas, los principales aportes del mismo, un análisis comparativo entre los objetivos perseguidos y los resultados obtenidos, y finalmente, algunas líneas de trabajo futuro sobre el tema.

Capítulo 2

Bases teóricas

Históricamente, el proceso de desarrollo de software ha experimentado una gran cantidad de problemas, muchas veces de gran magnitud, los cuales dieron origen al concepto de "crisis del software".

El término crisis del software apareció por primera vez en 1968, en la primera conferencia organizada por la OTAN sobre desarrollo de software, donde se habló por primera vez del conjunto de dificultades o errores ocurridos en la planificación, estimación de los costos, productividad y calidad de un software. Para dar solución a los problemas que se presentaban en esta conferencia se creó una nueva rama de ingeniería, la ingeniería de software.

La ingeniería del software establece que la construcción de software debe ser encarada de la misma forma en que otras ingenierías construyen sistemas complejos, como puentes, edificios, barcos y aviones [11]. La idea básica consiste en considerar al software a construir como un producto complejo y a su construcción como un trabajo ingenieril [11].

Hacia el final de los años 70, Demarco [18] introduce el concepto de desarrollo de software basado en modelos o MBD (Model Based Development). El principal fundamento detrás de este concepto es que el desarrollo de un sistema de software debe ser precedido por la definición de un modelo, tal como se realiza en otras ingenierías.

Bajo el enfoque MBD, el proceso de desarrollo de software implica, en las primera fases, la construcción de distintos modelos (modelos de requisitos, modelos de análisis y modelos de diseño), y posteriormente, la traducción de estos modelos a código. Esto, si bien representa un paso importante dentro de la ingeniería de software, aún carece de características que permitan asegurar la calidad y corrección del producto final.

Algunos de los problemas más importantes del proceso de desarrollo MBD son: i) el problema de la productividad, el mantenimiento y la documentación, y ii) el problema de la flexibilidad a los cambios tecnológicos [11].

El problema de la productividad, el mantenimiento y la documentación consiste en la perdida gradual de la sincronización entre los modelos y el código. Generalmente se da que al iniciar la codificación, el hecho de traducir a código lo especificado por los modelos, hace que modelos y código estén perfectamente sincronizados. Sin embargo, con el tiempo los cambios realizados en el código dejan de ser representados en los modelos, muchas veces por considerar esta actividad como una perdida de tiempo. Lo que aumenta cada vez más la brecha entre modelos y código. Finalmente, lo que ocurre

es que los modelos terminan siendo tan diferentes al código que pierden completamente su utilidad como herramientas de mantenimiento, lo que, dada la complejidad de los sistemas de software actuales, dificulta considerablemente esta importante actividad.

El problema de la flexibilidad a los cambios tecnológicos. La industria del software avanza a una velocidad vertiginosa, por lo que nuevas tecnologías surgen de manera muy frecuente y se hacen populares rápidamente. Además, generalmente estas nuevas tecnologías ofrecen beneficios que hacen que adoptarlas sea imperativo, ocasionando muchas veces la perdida de valor de las inversiones realizadas en tecnología.

2.1. MDD

El enfoque MDD es la evolución natural del Desarrollo Basado en Modelos (MBD), enriquecido mediante la automatización de las transformaciones entre modelos. De hecho, el adjetivo “dirigido"(driven) en MDD, a diferencia de “basado” (based) en MBD, enfatiza el hecho que en este paradigma los modelos son sumamente importantes, tanto o más que el código fuente. Esto debido a que, en MDD, la idea central es lograr el mayor grado posible de automatización a la hora de generar código a partir de los modelos generados en la etapa de diseño.

Una de las grandes promesas de MDD es la de mejorar el desarrollo de software mediante la aplicación de un proceso guiado por modelos y soportado por potentes herramientas. Es así que, en MDD, los modelos se generan desde los más abstractos hacia los más concretos, a través de sucesivas transformaciones y/o refinamientos, hasta llegar finalmente al código. Es decir, en MDD, los modelos pasan de ser entidades meramente contemplativas a convertirse en entidades productivas [11].

Los puntos claves de la iniciativa MDD fueron identificados en [19] de la siguiente manera:

1. El uso de un mayor nivel de abstracción en la especificación tanto del problema a resolver como de la solución correspondiente, en relación con los métodos tradicionales de desarrollo de software.
2. El aumento de confianza en la automatización asistida por computadora para soportar el análisis, el diseño y la ejecución.
3. El uso de estándares industriales como medio para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica.

Mientras que los pilares sobre los que se apoya la iniciativa son, a grandes rasgos, los siguientes [11]:

1. Modelos con diferentes niveles de abstracción, escritos en lenguajes bien definidos.
2. Definiciones de cómo un modelo se transforma en otro modelo más específico.
3. Herramientas de software que den soporte a la creación de modelos y su posterior transformación.

Por lo tanto, MDD plantea una forma completamente nueva de entender el desarrollo y mantenimiento de sistemas de software donde los modelos son utilizados para

dirigir las tareas de comprensión, diseño, construcción, pruebas, despliegue, operación, administración, mantenimiento y modificación de los sistemas [11].

2.1.1. Modelos definidos por MDD

Para lograr lo expuesto en la sección anterior, MDD identifica cuatro tipos de modelos diferentes:

- CIMs,
  Los CIMs (Computational Independent Model), son los modelos con más alto nivel de abstracción y son independientes de cualquier metodología computacional. Los CIMs representan el contexto, los requerimientos y el propósito de la solución pero todo esto sin ninguna implicación computacional. Usualmente se hace referencia a los CIMs como modelos del dominio ya que en su construcción se utiliza vocabulario que resulta familiar para los expertos en el dominio en cuestión. En principio, hay partes del CIM que ni siquiera pueden se mapeadas a la implementación final del software.
- PIMs,
  Los PIMs (Platform Independent Model), son modelos independientes de la plataforma, es decir, independientes de cualquier tecnología de implementación. Este es el nivel que describe el comportamiento y la estructura de la aplicación pero de una manera en la que resulta independiente de la plataforma de implementación [3]. Dentro del PIM el sistema se modela desde el punto de vista de cómo se soporta mejor al negocio, sin tener en cuenta como va a ser implementado [11]. Es importante notar que el PIM es solo para las partes del CIM que serán resueltas mediante una solución basada en software.
- PSMs,
  Los PSMs (Platform Specific Model), son modelos específicos para una plataforma dada. Cada PSM es la proyección del PIM a un plataforma específica [11]. Incluso si el PSM no será ejecutado directamente, lo que sí puede llegar a ocurrir por ejemplo al interpretar modelos [3], esta representación debe tener toda la información pertinente al comportamiento y la estructura de una aplicación con respecto a una plataforma específica de tal manera que los desarrolladores puedan traducirla linealmente a código.
- IMs,
  Los IMs (Implementation Model), son los modelos que representan el código fuente. Constituyen el paso final del desarrollo basado en modelos (MDD), son básicamente la traducción o transformación de cada PSM a código.

Cada uno de estos modelos representan o modelan al mismo sistema mostrando diferentes niveles de abstracción, cuando mayor es el nivel de abstracción, más general es la información que provee el modelo sobre el sistema modelado.

En general las diferencias entre el modelo de máximo nivel de abstracción y el de mínimo nivel de abstracción son bastante grandes, sin embargo, agregando los niveles intermedios obtenemos modelos con la diferencia justa como para que, a pesar de las diferencias, sea posible ir transformando uno en otro con relativa facilidad.

2.1.2. Beneficios de MDD

El desarrollo dirigido por modelos permite mejorar las prácticas más comunes dentro del desarrollo de software. Los beneficios de utilizar MDD son los siguientes [11]:

- **Incremento en la productividad:** Partiendo de los modelos definidos en la etapa de diseño, MDD permite generar código de manera automática. Lo que reduce el costo y el tiempo de desarrollo de software.

- **Adaptación a los cambios tecnológicos:** Gracias a que los modelos de alto nivel están libres de detalles de implementación, MDD resuelve el problema generado por el vertiginoso progreso tecnológico. Principal responsable de que los componentes de software se vuelvan obsoletos rápidamente.

- **Adaptación a los cambios en los requisitos:** MDD facilita el proceso de agregar o modificar funcionalidades ya que, cuando esto es necesario, solo se debe definir/modificar los modelos necesarios, reutilizando las transformaciones M2M definidas previamente.

- **Consistencia:** Gracias a la automatización, MDD permite la generación consistente de código (tarea propensa a errores cuando se realiza de manera manual).

- **Re-uso:** los modelos y las transformaciones definidas en un proceso MDD pueden ser fácilmente reutilizados. Los modelos para producir el mismo sistema para varias plataformas y las transformaciones para producir sistemas diferentes para una misma plataforma.

- **Mejoras en la comunicación con los usuarios:** En general, los modelos de más alto nivel, al omitir detalles específicos de la implementación, suelen ser fáciles de entender. Incluso para los usuarios finales.

- **Mejoras en la comunicación entre los desarrolladores:** Como en MDD los modelos son también parte del sistema y no solo de la documentación, se mantienen actualizados y confiables. Lo que contribuye con la comunicación entre los desarrolladores.

- **Captura de la experiencia:** Los modelos y las transformaciones capturan la experiencia de los desarrolladores más expertos. Permitiendo que los demás las aprovechen incluso si los primeros no están presentes.

- **Los modelos son productos de larga duración:** Esto se debe a que los modelos de más alto nivel son resistentes a los cambios tecnológicos y solo son afectados por los cambios en los requisitos.

- **Posibilidad de demorar las decisiones tecnológicas:** Esto se debe a que, al aplicar MDD, las decisiones tecnológicas se especifican en la etapas finales del desarrollo. Lo que permite demorar la toma de este tipo de desiciones.

2.2. MDA

Existen varias propuestas concretas para MDD. Una de las más extendidas actualmente es la propuesta MDA, propuesta para MDD desarrollada por el Object Management Group, y promovida a partir del 2000 con los objetivos de afrontar los desafíos de integración de las aplicaciones y los continuos cambios tecnológicos [11]. El framework MDA es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos, siguiendo el proceso MDD [11] [20].

La idea clave detrás de MDA es la de separar la lógica de negocio, la funcionalidad de una aplicación, de su implementación en una determinada plataforma. De este modo, el desarrollador crea modelos de alto nivel e independientes de la plataforma (PIM), que posteriormente, mediante transformaciones modelo a modelo (M2M), generan de manera automática modelos dependientes de la plataforma (PSM). Y finalmente, al igual que en MDD, el código de la aplicación es derivado de los modelos específicos de la plataforma (PSM) mediante transformaciones modelo a texto (M2T) [13].

Al igual que MDD, MDA también considera modelos CIM, modelos independientes de cualquier detalle relacionado a la computación y que se refieren a aspectos del negocio. Sin embargo, la automatización de la transformación CIM a PIM es un problema complicado debido al salto semántico que los separa [13].

Acorde con las directivas de la OMG, las dos motivaciones principales para MDA son la interoperabilidad (independencia de los fabricantes a través de la estandarización) y la portabilidad (independencia de plataforma) de los sistemas de software [11]. Así mismo, promete mejorar la mantenibilidad de los sistemas a través de la separación de conceptos [20].

La arquitectura dirigida por modelos (MDA) esta relacionada con varios estándares. Algunos de ellos son: el Unified Modeling Language (UML), el Meta-Object Facility (MOF), el XML Metadata Interchange (XMI), el Enterprise Distributed Oject Computing (EDOC), el Software Process Engineering Metamodel (SPEM) y el Common Warehouse Metamodel (CWM).

2.2.1. Principios sobre los que descansa MDA

El OMG ha definido MDA y los lenguajes asociados a MDA, en base los siguientes principios que subyacen a la visión particular del OMG sobre la Ingeniería Dirigida por Modelos (MDE) [3].

- Los modelos deben ser expresados en una notación bien definida, de manera a permitir una comunicación y comprensión eficaz de la descripción de sistemas de escala empresarial.

- Las especificaciones de los sistemas deben organizarse en torno a un conjunto de modelos asociados a transformaciones que implementen las asignaciones y las relaciones entre ellos. Todo esto permite un diseño organizado basado en un marco arquitectónico multicapa y multiperspectiva.

- Los modelos deben ser construidos en conformidad con un conjunto de metamodelos, lo que facilita la integración y transformación significativa entre los modelos, y la automatización a través de herramientas.

- Este marco de modelado debe aumentar la aceptación y la adopción del enfoque del MDE, y fomentar la competencia entre los proveedores de herramientas de modelado.

2.2.2. Mapeos

Un mapeo consiste en la definición de la correspondencia entre elementos de dos modelos diferentes [3]. Los mapeos pueden definirse entre elementos de toda clase de modelos.

El mapeo entre diferentes modelos (o niveles de modelado) puede ser implementado a través de transformaciones. Los mapeos pueden proporcionar una especificación conceptual que permite implementar la transformación entre los diferentes niveles de modelado propuestos por MDA (o incluso por MDD). El objetivo es, obviamente, automatizar la implementación del mapeo (es decir, de las transformaciones entre modelos) tanto como sea posible, evitando así costosas transformaciones manuales.

Muchas veces, a la hora de definir un mapeo es necesario tener en cuenta decisiones tomadas a lo largo del desarrollo en términos de algunos detalles en modelos de nivel inferior [3]. Por ejemplo, en un mapeo de PIM a PSM deben ser tenidas en cuenta decisiones que, si bien se aplican a nivel de PIM, dependen de opciones a nivel de PSM. En estos casos, los mapeos definen anotaciones (marcas, en la jerga MDA) que se utilizarán para guiar las transformaciones entre los dos niveles de modelado.

2.3. MoWebA

MoWebA es un enfoque navegacional, centrado en roles y basado en modelos para el desarrollo de aplicaciones web y móviles [9] [21]. Define aspectos metodológicos (i.e., procesos, etapas, productos, dimensiones) y los complementa con un entorno de desarrollo completo, incluyendo herramientas de modelado y transformación, generación automática de código, uso de estándares y arquitectura en capas, entre otros.

La figura 2.1 presenta las dimensiones de MoWebA, las cuales cubren los procesos de modelado y transformación de la misma. MoWebA adopta el enfoque MDA identificando tres fases diferentes, todas relacionadas con las actividades de modelado y transformación (ver el eje de las fases, etiquetado como “phases”): i) el espacio del problema, cubierto por el CIM (Computational Independent Model) y el PIM (Platform Independent Model); ii) el espacio de solución, cubierto por el ASM (Architectural Specific Model) y el PSM (Platform Specific Model); iii) y la definición del código fuente, cubierto por el ISM (Implementation Specific Model) y código manual. Por otro lado, el eje de los niveles (etiquetado “levels” en la figura 2.1), aborda perspectivas complementarias que deben ser consideradas en cada fase (contenido, lógica de negocio, navegación, presentación, usuarios). Y por último, el eje de los aspectos (etiquetado “aspects” en la figura 2.1) aborda las consideraciones estructurales y de comportamiento para cada perspectiva.

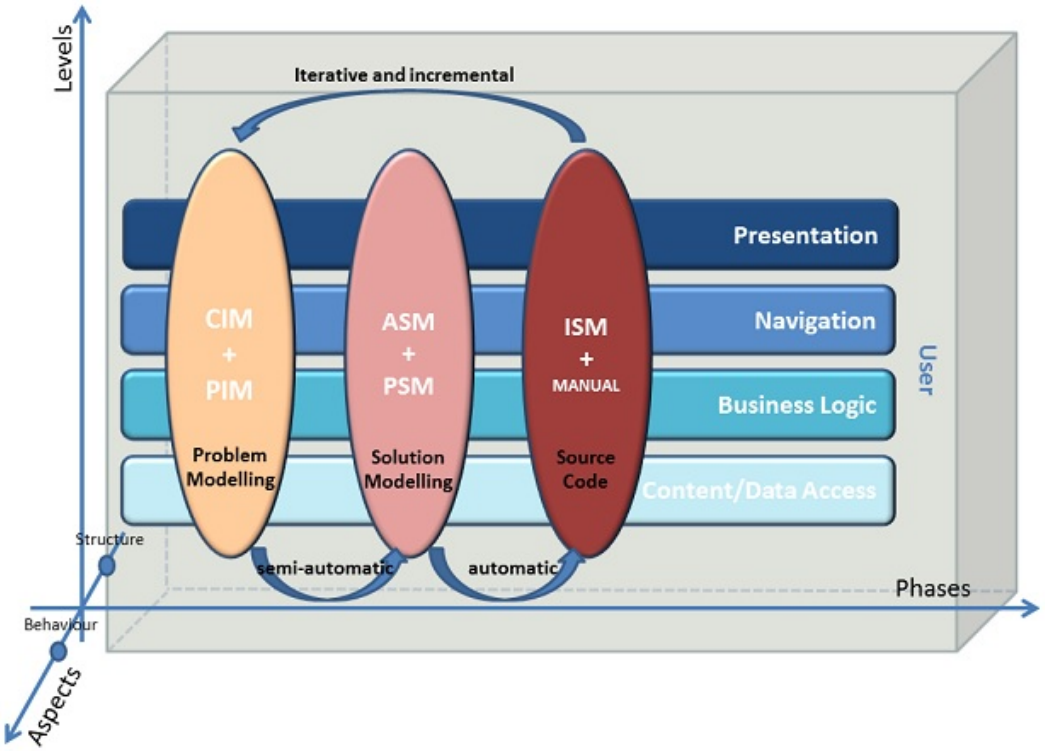


Figura 2.1: Dimensiones de MoWebA

2.3.1. Desarrollo de aplicaciones con MoWebA

Con el fin de desarrollar aplicaciones, MoWebA define dos procesos complementarios principales: el primero relacionado con las actividades de modelado y el segundo relacionado con las actividades de transformación. Para la formalización de los procesos de modelado y transformación, MoWebA adopta el lenguaje MOF en la definición de la sintaxis abstracta, y la extensión por medio los perfiles UML para la definición precisa del lenguaje de modelado (i.e., sintaxis concreta).

El proceso de modelado abarca siete etapas con sus correspondientes actividades, para la especificación del sistema en desarrollo (considerando el espacio del problema, la(s) arquitectura(s) y la(s) plataforma(s) objetivo). Estas etapas consideran el CIM, el PIM, el ASM y el PSM. La definición de la CIM previene la identificación tardía de los requisitos, centrándose en las especificaciones de requisitos funcionales. La especificación PIM se basa en cinco modelos, ofreciendo una fuerte separación de conceptos: Dominio, Lógica, Navegación, Presentación y Usuario. El ASM enriquece los modelos con información referente a una arquitectura específica (por ejemplo, RIA, SOA, REST, entre otros), y el PSM añade información relacionada con la plataforma de destino (e.g., lenguajes de programación específicos, frameworks, entre otros).

El proceso de transformación, por otro lado, está relacionado con los pasos, técnicas y herramientas que permiten la definición y ejecución de las transformaciones M2M (i.e., modelo a modelo) y/o M2T (i.e., modelo a texto/código). Este proceso va a través de cada fase de MoWebA (i.e., de CIM/PIM a ASM/PSM, y de ASM/PSM a

ISM/ajustes manuales). La transformación de CIM/PIM a ASM/PSM debería realizarse de manera automática (i.e., teniendo como entrada el modelo PIM y un archivo de configuración), precisamente lo que buscamos en este trabajo. Para lograr este objetivo, existe un proceso que permite la definición de metamodelos para arquitecturas y/o plataformas específicas, junto con las correspondientes reglas de mapeo que guíen y orienten la transformación de modelos PIM a modelos ASM/PSM. La transformación de modelos ASM/PSM a ISM se realiza de manera automática. Sin embargo, dado que las experiencias reales han demostrado que algunas veces es necesario realizar ajustes manuales, MoWebA considera una fase de "ajustes manuales", donde es posible agregar código adicional que permita introdicir mejoras en la aplicación producida. Por último, vale la pena mencionar que el proceso de transformación puede ser realizado iterativamente, lo que permite un desarrollo incremental de la aplicación.

2.3.2. Definición y aplicación de modelos ASM

El modelo ASM enriquece los modelos previos con información adicional relacionada con la arquitectura del sistema modelado (e.g., RIA, REST, entre otros). De manera complementaria, el PSM se orienta al refinamiento de los modelos previos por medio del agregado de información adicional relacionada con la plataforma y el lenguaje que fueron seleccionados para el sistema final (e.g., Java, .NET, PostgreSQL). En esta etapa, los ingenieros de software se trasladan de la definición conceptual del problema (modelos CIM/PIM) a la definición de la solución (modelos ASM/PSM).

Es necesario definir el modelo ASM antes de poder utilizarlo. Esta definición abarca la especificación del metamodelo correspondiente, entre otras etapas. Brambilla et al. [3] han recomendado un proceso que permite la definición de una sintaxis abstracta y MoWebA lo sigue con el objetivo de definir un metamodelo para el ASM. Además, MoWebA complementa el proceso sugerido con pasos adicionales que van desde la definición de la sintaxis concreta hasta la generación del código final de una aplicación [12]. Los pasos de este proceso se sintetizan a continuación:

1. Definir el metamodelo del ASM usando MOF.
2. Definir el perfil UML correspondiente.
3. Especificar las reglas de mapeo entre elementos del PIM y elementos del ASM.
4. Definir reglas de transformación M2M para la transformación de PIM a ASM utilizando lenguajes de transformación estándar (e.g., ATL o QVT).
5. Definir reglas de transformación de ASM a PSM y PSM a código, o bien de ASM a código, utilizando lenguajes de transformación M2M y M2T (e.g., Acceleo) respectivamente.

Este trabajo se enfoca en las etapas 3 y 4, para arquitecturas RIA y persistencia móvil, algo posible, únicamente, gracias a que las demás etapas (para ambas arquitecturas) ya fueron abordadas en trabajos previos, tes [22] y [23] respectivamente.

Como puede ser observado en las etapas del proceso anterior, el enfoque MoWebA requiere algún esfuerzo adicional como contrapartida al mejoramiento de la portabilidad del PIM y al hecho de facilitar la evolución arquitectónica de las aplicaciones. Esto incluye la necesidad de especificación de metamodelos ASM y la definición de las correspondientes reglas de transformación, con el fin de lograr transformaciones automáticas para la arquitectura propuesta. En cualquier caso, es importante notar que

los pasos 1, 2 y 3 del proceso, se ejecutan una sola vez para cada nueva arquitectura a ser abordada.

Una vez definido el ASM para una arquitectura específica, puede ser utilizado para el desarrollo de aplicaciones que implementen la arquitectura seleccionada, siguiendo estos pasos:

1. Definir los diagramas correspondientes a los modelos CIM/PIM, siguiendo el proceso de modelado definido previamente (ver apartado 2.3.1).
2. Aplicar las reglas de transformación M2M definidas, para así obtener una primera versión del modelo ASM.
3. Realizar los ajustes manuales necesarios para completar el modelos ASM (solo en caso de ser necesario).
4. Generar el(los) modelo(s) PSM y el código final, aplicando reglas de transformación M2M y M2T respectivamente.
5. Realizar unos últimos ajustes manuales (también, solo en caso de ser necesario).

2.4. RIA

En los inicios de internet la mayoría de las aplicaciones web implementaban una arquitectura cliente-servidor. Entre las principales características de este tipo de arquitecturas destacan: la navegación a través de links, la recarga entera de la página ante la más mínima acción del usuario, interfaz poco interactiva, almacenamiento de datos y lógica de negocio implementadas totalmente del lado del servidor y una comunicación síncrona entre las partes [24]. Todos factores que inciden directamente en la experiencia del usuario, sobre todo a causa de tiempos de respuesta excesivamente prolongados.

En respuesta a estos problemas aparecen las Aplicaciones Enriquecidas de Internet (RIA). Brambilla et al. [25] definen a las RIA como aplicaciones de internet que explotan el poder de los clientes web para incrementar la adaptabilidad y usabilidad de las interfaces de usuario, ofreciendo funcionalidades similares a las de aplicaciones de escritorio. A pesar de que las RIA siguen el paradigma cliente-servidor, son capaces de transferir al cliente: i) el procesamiento de la interfaz de usuario, ii) la lógica de negocio, y iii) el manejo de datos.

Las RIA son una combinación de la presentación e interactividad de las aplicaciones de escritorio con la arquitectura y forma de distribución de las aplicaciones web [26]. La capa de presentación con su interacción asociada es trasladada al cliente, la computación de la página y datos de la misma se distribuyen entre servidor y cliente, y la comunicación se realiza mayormente en forma asíncrona. Este esquema permite obtener elementos de interfaz más interactivos, evitar la recarga innecesaria de la página, soportar contenidos multimedia, evitar transferencia de datos redundantes y minimizar tiempos de respuesta mejorando notablemente la experiencia del usuario [24].

2.4.1. Características de las RIA

En base a lo referido en [26] y en [27], las cuatro principales características de las RIA son las siguientes:

- **Distribución de datos:** Las aplicaciones web que implementan RIA tienen la capacidad de distribuir el almacenamiento de datos entre el cliente y el servidor. El hecho de poder almacenar datos en el cliente (lo que no es posible en las aplicaciones web tradicionales), permite a las RIA reducir los tiempos de respuesta, validar datos localmente, e incluso, ser utilizadas en modo fuera de línea.

- **Distribución de lógica de negocios:** Otra capacidad interesante de las RIA es la posibilidad de realizar operaciones complejas del lado del cliente. Operaciones como el ordenado y filtrado de datos, validación local (gracias al almacenamiento local de datos), reordenamiento de la página, etc.

- **Comunicación asíncrona:** Las RIA permiten la utilización, tanto de comunicación síncrona, como de comunicación asíncrona. Además, al implementar RIA, tanto el cliente como el servidor son capaces de iniciar la comunicación (lo que tampoco es posible en las aplicaciones web tradicionales). La comunicación asíncrona es la responsable de la recarga parcial de las páginas y de servicios del tipo push y pull.

- **Mejora de interfaz de usuario:** La interfaz de usuario es un apartado en el que las RIA aventajan de manera considerable a las aplicaciones web tradicionales, sobre todo en cuanto a comportamiento y presentación. Entre otras cosa, las RIA permiten enriquecer la interfaz de usuario mediante el uso de widgets y contenedores sumamente personalizables, la división de una página en varias subpáginas independientes, la implementación de animaciones y multimedia, etc.

2.5. Persistencia móvil

La arquitectura que en este trabajo denominamos como persistencia móvil, se deriva del enfoque MoWebA Mobile propuesto en [23].

MoWebA Mobile parte de la extensión del metamodelo y los perfiles de MoWebA, y se enfoca en el desarrollo de aplicaciones móviles nativas centradas en la capa de datos. Abarcando fundamentalmente dos aspectos: i) la persistencia de datos, y ii) los proveedores de datos.

Luego de destacar la necesidad actual de especificaciones que permitan describir la persistencia en el modelado de aplicaciones móviles y, sobre todo, la poca adopción de MDA. En [23], se pretende cubrir conceptualmente el modelado de la persistencia en el desarrollo de aplicaciones móviles. Tanto mediante elementos específicos de la arquitectura que permitan tener en cuenta las distintas opciones y/o mecanismos de persistencia en el modelado (a fin de posibilitar el almacenamiento de datos incluso sin conexión de red). Así como también considerando la conexión con fuentes remotas, es decir, con los proveedores de datos.

Por todo lo anterior, en [23] deciden optar, en primer lugar, por el abordaje de la capa de datos. Capa que cubre conceptualmente todos los aspectos mencionados en el párrafo anterior. Y en segundo lugar, por MoWebA, puesto que la consideran una metodología capaz de generar un impacto positivo en cuanto al problema de

portabilidad existente en el desarrollo de aplicaciones móviles gracias a su capacidad de contener y aislar elementos arquitectónicos en el ASM.

El aporte más importante de MoWebA Mobile para con el presente trabajo de tesis, lo constituye, justamente, la definición de un ASM centrado en la capa de datos. La razón es tan simple como contundente. La existencia de metamodelos y perfiles UML orientados al modelado de nuevas arquitecturas siguiendo el enfoque propuesto por MoWebA, mejora considerablemente las posibilidades de análisis del ASM.

2.6. Trabajos relacionados

Como ya fue mencionado, algunos enfoques MDD incluyen aspectos arquitectónicos a nivel del modelado conceptual sin hacer una distinción clara entre el modelo independiente y el modelo arquitectónico. Por ejemplo, para generar RIA, algunos enfoques han extendido la notación de sus PIM con primitivas o patrones adicionales, específicos de la arquitectura (por ejemplo, WebML RIA [4], UWE para RIA [5], RUX-model [28], [29], entre otros). Otras propuestas decidieron añadir información específica sobre la arquitectura a nivel del PSM. En este caso, arquitectura y detalles de la plataforma se incluyen en el mismo nivel de modelado (el PSM), perdiendo éste portabilidad a nivel arquitectónico con respecto a las diferentes tecnologías donde se puede implementar (por ejemplo, UWA para RIA [30]).

Ya en 2004, en Mikkonen et al. [10] notaron que los estilos arquitectónicos no tienen un lugar claro en MDA. Ellos propusieron modificar MDA añadiendo una nueva capa denominada Modelo Específico de la Arquitectura. Esta nueva capa encapsula propiedades específicas de la arquitectura, y al mismo tiempo, hace explícitos en el modelo los estilos arquitectónicos, patrones de diseño y otras decisiones de diseño importantes. Esta interesante propuesta fue discontinuada, sin embargo, inspiró la propuesta de MoWebA, en la cual el mismo PIM puede ser utilizado como base para generar diferentes ASMs para diferentes arquitecturas (por ejemplo, RIA, REST, cliente-servidor, SOA, móvil).

Posteriormente, en Manset et al. [31] definieron un enfoque MDD centrado en la arquitectura para la generación automática de aplicaciones grid. Ellos combinaron el desarrollo impulsado por modelos con procesos centrados en la arquitectura, que consideran la arquitectura como el principal artefacto en el proceso de desarrollo de software. También en esta dirección, en Marcos et al. [32] extendieron MIDAS, un marco metodológico para el desarrollo de sistemas de información web, mediante la integración de aspectos de diseño arquitectónico. MIDAS considera tres puntos de vista diferentes de los sistemas de información web, a saber, el contenido, el hipertexto y el comportamiento, que son ortogonales a los niveles de abstracción MDA (CIM, PIM y PSM). Además, la arquitectura del software se concibe como una perspectiva transversal, que a su vez es ortogonal a los tres puntos de vista mencionados. Por lo tanto, se definen tanto los modelos de arquitectura independientes de la plataforma como los modelos de arquitectura específicos de la plataforma. Más recientemente, estos esfuerzos se convirtieron en ArchiMeDes [33], [34], un framework MDD para la especificación de arquitecturas orientadas a servicios. A nivel de PIM, ArchiMeDes define un lenguaje específico del dominio (DSL) que permite la definición de arquitecturas de servicios conceptuales. A nivel de PSM, diferentes DSL soportan el modelado

de plataformas de ejecución concretas o tecnologías de implementación.

ArchMDE [35] es otro enfoque MDE centrado en la arquitectura, y se enfoca en el desarrollo y validación de sistemas embebidos de tiempo real. En este caso, el PIM se descompone en dos modelos: i) un Modelo Independiente del Estilo Arquitectónico, que define la estructura y el comportamiento de la arquitectura funcional del sistema de tiempo real; Y ii) un Modelo Específico del Estilo Arquitectónico, que tiene en cuenta las características funcionales, no funcionales y arquitectónicas del sistema.

Todos estos trabajos constituyen esfuerzos centrados en la arquitectura, es decir, trabajos que consideran la arquitectura como el artefacto principal en el proceso de desarrollo de software y, por lo tanto, los conceptos arquitectónicos aparecen temprano en el proceso de desarrollo (por ejemplo, a nivel PIM). En el caso de MoWebA, aunque existe el interés de poder desarrollar una misma aplicación para diferentes arquitecturas, el proceso de desarrollo no se basa en la arquitectura. También se puede resaltar que algunas de estas obras son específicas para la arquitectura orientada a servicios ([33], [34]), mientras que en MoWebA se pueden definir diferentes ASM para diferentes arquitecturas. Por último, algunos de estos trabajos son específicos del dominio (aplicaciones grid [31], sistemas de tiempo real [35]), en contraposición a MoWebA, que permite el desarrollo de sistemas de información de gestión general.

2.7. Síntesis del capítulo

En este capítulo han sido presentadas las bases teóricas del presente trabajo de tesis, junto con algunos trabajos relacionados, ya sobre el final del mismo.

Dado que MDD es la evolución natural de MBD, en la introducción del capítulo se explica el origen y el objetivo de MBD. Para luego, ya en la primera sección, presentar a MDD, un paradigma de desarrollo de software que busca el mayor grado posible de automatización en la generación de código final a partir de los modelos generados en la etapa de diseño.

Posteriormente son presentados MDA, la propuesta MDD más extendida actualmente, y MoWebA, un enfoque metodológico MDD que adopta el estándar MDA en sus diferentes fases.

Al presentar MoWebA, se hace un especial énfasis en la nueva fase que esta propone, el modelo ASM, puesto que este es un concepto muy relevante para el presente trabajo de tesis. El ASM constituye un modelo intermedio entre el PIM y el PSM que permite enriquecer los modelos previos con información adicional referente a la arquitectura del sistema en desarrollo.

Finalmente, en las últimas secciones del capítulo se presentan dos arquitecturas de gran relevancia actualmente, que serán adoptadas en el presente trabajo de tesis como arquitecturas de referencia, estas son RIA y persistencia móvil.

Capítulo 3

Mapeo sistemático de la literatura

Resulta que la mayoría de las metodologías MDD existentes en la actualidad adoptan las arquitecturas RIA, ya sea marcando sus modelos PIM, o extendiéndolos con primitivas adicionales y/o patrones [9], o incluso, añadiendo nuevos modelos completos a nivel de PIM. Evidencia de esto es que, por ejemplo, WebML define WebML RIA [15]; UWE, UWE para RIA [5]; OO-H, OOH4RIA [6]; etc.

Esta tendencia a adaptar el PIM para permitirle capturar información relativa a la arquitectura, ya sea añadiendo marcas (pero sin considerar esto como un PIM marcado), o añadiendo nuevos modelos enteros, o incluso añadiendo nuevos elementos a la notación del PIM, no cumple estrictamente con el principio de portabilidad del PIM promovido tanto por MDD como por MDA.

En consecuencia, la idea principal del mapeo sistemático realizado en el marco del presente trabajo de tesis, es analizar propuestas y otros trabajos de investigación disponibles en los campos de MDD y MDA, en los que se llevan a cabo transformaciones M2M orientadas a arquitecturas RIA. Centrando el interés en identificar áreas poco estudiadas, trabajos pendientes, y en general, cualquier oportunidad de investigación que pueda o requiera ser abordada en el futuro.

La elección de un Mapeo Sistemático de la Literatura (SMS) por sobre una Revisión Sistemática de la Literatura (SLR) obedece a que los SMSs son más generales, tienen un alcance más amplio y su principal objetivo es descubrir cuáles son las tendencias de investigación, mientras que las SLRs son más adecuadas para aquellos casos en los que existe una necesidad de agregar evidencia y, en general, requieren de objetivos más claros y preguntas de investigación más específicas [36].

Si bien no fue posible encontrar en la literatura ningún SMS o SLR que abarque exactamente los mismos temas que los contemplados en este estudio, sí fueron hallados dos trabajos que siguen metodologías afines y que son bastante recientes, [37] y [38].

Casteleyn et al. [37] presenta un SMS que analiza los primeros diez años de las RIA, el período de estudio abarca desde el 2002, año en el que el término RIA apareció por primera vez en un documento de Macromedia [39]) hasta el 2011, el año anterior a la realización de su estudio. Así también, en Aragón et al. [38] se presenta un análisis dirigido, en primera instancia y según ellos mismos lo indican, por los principios generales establecidos para la realización de un SLR, pero que sin embargo, no fue

publicado como tal.

El principal objetivo de este SMS es identificar *"¿Qué se ha hecho?"* y *"¿Qué se debe hacer en el futuro?"* en el contexto de las metodologías MDD centradas en arquitecturas RIA. Esto contrasta con los otros dos trabajos presentados previamente ya que [38] cubre solo MDD mientras [37] cubre solo RIA. Además, este trabajo es el único que analiza MDD y RIA en conjunto, y es el único que analiza las metodologías MDD existentes, centrándose en RIA y en las transformaciones M2M implicadas .

3.1. Metodología

Aunque los SMS aplican la misma metodología básica que las SLR [40] [41], están diseñados para dar una visión más general y amplia sobre un área de investigación en particular. Mediante la identificación de la cantidad y el tipo de investigación existentes en el área, así como también de los resultados disponibles [42] [43], los SMS permiten la identificación de brechas de investigación en la literatura. La principal diferencia entre SMS y SLR radica en el alcance y en los procedimientos de análisis. El alcance es con frecuencia más amplio en SMSs que en SLRs, y en general, no se espera una síntesis demasiado profunda en los SMS [44].

El presente SMS ha sido llevado a cabo teniendo en cuenta las directrices previstas en [36, 42, 43] y fue realizado en dos etapas: la de planificación y la de realización. Las actividades concernientes a cada una de estas dos etapas se detallan a continuación.

3.1.1. Etapa de planificación

Durante la etapa de planificación fueron llevadas a cabo diferentes actividades con el objetivo de establecer un protocolo que guíe, oriente y avale la realización del presente SMS. De manera muy resumida, las actividades contempladas fueron las siguientes: (1) definición de las preguntas de investigación; (2) definición de la estrategia de búsqueda; (3) definición de los criterios de selección; (4) definición del procedimiento de selección; (5) definición de la estrategia de extracción de datos; Y (6) selección del método de síntesis. Cada una de estas actividades se explica detalladamente en los siguientes apartados.

3.1.1.1. Preguntas de investigación

Como ya fue señalado al inicio de este capítulo, la razón por la que se realiza el presente SMS es la necesidad de obtener una visión general sobre las áreas de investigación relacionadas con MDD y MDA, especialmente en lo que respecta a propuestas y trabajos de investigación que contemplen la definición de reglas de transformación M2M para arquitecturas RIA.

Teniendo en cuenta esto, junto con los intereses y objetivos identificados en la introducción del capítulo con respecto al alcance pretendido para el presente SMS, han sido planteadas las preguntas de investigación detallas en la tabla 3.1.

	Pregunta de investigación	Motivación
Preguntas de investigación generales	**RQ1:** ¿Cuáles son los objetivos perseguidos en la investigación sobre MDD y las transformaciones M2M?	Identificar cuál es el área de investigación más significativa y qué áreas han sido poco estudiadas, explorando conceptos básicos, recopilando conocimiento de las prácticas actuales, avanzando la práctica a través de la ciencia del diseño (Design Science).
	RQ2: ¿Qué tipo de métodos se utilizaron en la investigación relacionada con MDD y las transformaciones M2M?	Determinar si las propuestas en este campo de investigación son más prácticas o más de investigación básica y también identificar oportunidades para realizar futuras investigaciones.
	RQ3: ¿Cuál es el alcance de las propuestas en los trabajos seleccionados?	Determinar el tipo de resultado obtenido en cada uno de los trabajos analizados, e.g. nuevas propuestas, herramientas utilizadas, etc.
	RQ4: ¿Cuáles de las fases de modelado son contempladas?	Identificar qué fases de MDA adoptan cada una de las propuestas analizadas, ya que algunas cumplen más estrictamente que otras con lo definido por MDA, e incluso algunas proponen nuevas fases de modelado.
	RQ5: ¿La propuesta sigue o no el estándar MDA?	Determinar el grado de adopción del estándar MDA frente al de los principios MDD.
	RQ6: ¿La propuesta cuenta con herramientas que la soporten? ¿Cuán desarrolladas están?	Identificar si la propuesta llegó a ser algo más que eso, es decir, si existe alguna herramienta que permita utilizarla y/o probarla.
	RQ7: ¿Cuál es el IDE señalado para el uso de la herramienta propuesta?	Se pretende identificar el IDE para el cual fue desarrollada la herramienta propuesta.
Preguntas sobre transformaciones M2M	**RQ8:** ¿Cuál es el nivel de automatización de la transformación M2M?	Identificar el nivel de automatización de las transformaciones M2M planteadas en cada una de las propuestas analizadas.
	RQ9: ¿Pertenecen los modelos origen y destino, al mismo nivel de abstracción?	Identificar y analizar el nivel de abstracción del los modelos origen y destino involucrados en las transformaciones M2M.
	RQ10: ¿En qué lenguaje se modelan los modelos origen y destino?	Diferenciar si los modelos origen y destino son modelados con lenguajes diferentes o no.
	RQ11: ¿Cuál es la relación entre los modelos origen y destino?	Identificar si el modelo destino constituye un nuevo modelo o se obtiene mediante una actualización/modificación del modelo origen.
	RQ12: ¿Qué relación existe entre el número de modelos origen y el número de modelos destino?	Se pretende identificar si se permiten o no múltiples modelos origen y/o múltiples modelos destino.
	RQ13: ¿Qué lenguaje de transformación M2M utiliza la propuesta analizada?	Se busca identificar el lenguaje de transformación M2M utilizado en cada una de las propuestas analizadas.

Tabla 3.1: Preguntas de investigación

3.1.1.2. Estrategia de búsqueda

La estrategia de búsqueda ha sido definida teniendo en cuenta cuestiones relativas al problema de terminología existente en el contexto de MDD. Los términos a menudo varían bastante dependiendo de diferentes factores, tales como los estándares utilizados para definir cada uno de los elementos involucrados en el proceso, aspectos legales en cuanto a patentes existentes sobre algunos de los términos más comunes, entre otras cosas.

Como ejemplo, el OMG mantiene marca registrada sobre MDA, así como varios términos similares incluyendo MDD [11]. Por lo tanto, cuando la comunidad científica internacional se refiere a ideas relacionadas con la ingeniería de modelos sin centrarse exclusivamente en los estándares del OMG, acostumbra usar el acrónimo de MDE, que significa Model-Driven Software Engineering, y que aún no ha sido patentado por el OMG [11]. Y es en este sentido que utilizamos el acrónimo en el presente trabajo, a pesar de que estrictamente MDE es más general que MDD y MDA respectivamente e incluso los engloba, como bien se señala en [13].

Así también, otra corriente muy importante dentro de MDD, la comunidad científica española, utiliza principalmente el acrónimo DSDM, que significa Desarrollo de Software Dirigido por Modelos, en lugar de los tradicionales MDD, MDA o MDE (siempre considerándolos como sinónimos de MDD y no de acuerdo a su verdadero significado). Además, una importante corriente de habla inglesa utiliza el acrónimo MDSE (Model-Driven Software Engineering) como una sigla alternativa para MDE.

Otra situación particular se observa con el acrónimo de las transformaciones M2M. En ese sentido, algunos trabajos mencionan directamente el lenguaje de transformación involucrado en el proceso, ATL (del inglés Atlas Transformation Language) o QVT (del inglés Query/View/Transformation), sin hacer una referencia explícita a las transformaciones M2M. Debido a esto, a la hora de definir nuestra estrategia de búsqueda, en este SMS decidimos utilizar estos tres acrónimos como sinónimos.

3.1.1.2.1. Cadena de búsqueda La selección realizada sobre los términos principales, sinónimos, palabras alternativas o términos relacionados con los términos principales se presenta en la tabla 3.2 y se fundamenta, sobre todo, en los aspectos citados en la sección anterior.

Términos principales	Términos alternativos
MDD, MDE, MDA	DSDM, MDSD, MDSE, Model-Driven, Model Driven
M2M, ATL, QVT	Model-to-Model
RIA	Rich Internet Applications

Tabla 3.2: Términos de búsqueda

La cadena de búsqueda se muestra en la figura 3.1 y fue definida a partir de los términos presentes en la tabla 3.2, mediante el seguimiento/ejecución de las siguientes instrucciones:

1. Los términos principales se unen a sus correspondientes términos alternativos por medio del operador lógico OR.

2. Los términos principales se unen a otros términos principales por medio del operador lógico AND.

("MDD" OR "DSDM" OR "MDSD" OR "MDE" OR "MDSE" OR "MDA" OR "Model-Driven" OR "Model Driven")
AND
("M2M" OR "Model-to-Model" OR "ATL" OR "QVT")
AND
("RIA" OR "Rich Internet Applications")

Figura 3.1: Cadena de búsqueda

3.1.1.2.2. Fuentes de búsqueda Continuando con el desarrollo de la estrategia de búsqueda, es importante señalar que la búsqueda de obras publicadas abarca el periodo comprendido entre el 2002 (en marzo de ese año el término RIA apareció por primera vez en un documento de Macromedia [39]) y el 2015. Se considera el año de aparición del término RIA como límite inferior debido a que el presente estudio se centra en MDD para arquitecturas RIA.

Las bibliotecas digitales a ser utilizadas son cinco de las más relevantes para el contexto de este estudio. Han sido seleccionadas en base a las experiencias reportadas por Dyba et al. [45] y Petercen et al. [36], y se enumeran a continuación:

1. IEEE Xplore,
2. Springer Link,
3. ACM Digital Library,
4. ScienceDirect,
5. Scopus

La cadena de búsqueda se aplica a cada una de las fuentes de búsqueda mencionadas anteriormente, buscando, siempre que el buscador de las bibliotecas digitales lo permita, en títulos y resúmenes. Y cuando esto no es posible o se obtienen pocos resultados, buscando a texto completo.

3.1.1.3. Criterios de selección de estudios

Para la selección de los estudios primarios a ser analizados, se tienen en cuenta los criterios de inclusión y exclusión que se muestran en la tabla 3.3.

3.1.1.4. Procedimiento para la selección de estudios

El procedimiento de selección de los estudios primarios comprende tres etapas: La primera consiste, básicamente, en la inclusión de todos aquellos trabajos que cumplan los criterios de inclusión previamente establecidos y que hayan sido seleccionados siguiendo la estrategia de búsqueda definida.

La segunda etapa consiste en aplicar, a todos los artículos obtenidos en la etapa anterior, los criterios de exclusión definidos para la revisión del título y el resumen. En

Criterios de inclusión
1. Publicaciones relacionadas con MDD para RIA.
2. Artículos científicos en revistas académicas, conferencias y talleres.
3. Publicaciones en inglés, español y portugués.
4. Trabajos publicados entre el 2002 y el 2015.
Criterios de exclusión (para revisión de títulos y resúmenes)
1. Publicaciones no relacionadas con MDD para RIA.
2. Artículos de los cuales se disponga solo un resumen.
3. Publicaciones que no hayan sido revisadas por pares.
4. Publicaciones sobre el mismo estudio, duplicadas en diferentes fuentes.
Criterios de exclusión (para revisión del texto completo)
1. Publicaciones en las que se mencionen los criterios de búsqueda pero no constituyan sujetos de estudio (e.g., artículos que sólo mencionan los términos de búsqueda como ejemplos o en sentencias introductorias del resumen, pero sin profundizarlos posteriormente).
2. Publicaciones sobre el mismo estudio, duplicadas en diferentes fuentes.

Tabla 3.3: Criterios de inclusión y exclusión

caso de que surjan dudas, es decir, cuando el título y el resumen no permitan discernir con claridad la exclusión o no del artículo, este no será excluido en esta etapa.

Al igual que la segunda etapa, la tercera etapa toma como entrada los artículos no excluidos en la etapa anterior. Aquí se aplican los criterios de exclusión definidos para la revisión del texto completo del estudio primario. Luego de la lectura completa de los artículos, ya no deberían existir dudas sobre su inclusión o exclusión; Sin embargo, en caso que surja alguna duda, el artículo bajo análisis no será excluido. La razón esa que consideramos que una exclusión incorrecta será mucho más perjudicial que una inclusión incorrecta.

La aplicación de los criterios de inclusión y exclusión fue llevada a cabo por uno de los autores, mientras que los demás contribuyeron con la resolución de las ambigüedades que se fueron dando en el transcurso del proceso. Por lo tanto, cada artículo fue revisado por un sólo autor, lo que representa una amenaza para la confiabilidad del SMS (véase también la discusión acerca de las amenazas a la validez del SMS, Sección 3.4). Sin embargo, en Petticrew et al. [46] se sostiene que cuando en un SMS se obtiene un gran número de estudios y muchos de ellos constituyen ruido claramente identificable, el proceso puede llevarse a cabo de forma individual.

3.1.1.5. Estrategia para la extracción de datos

El formulario de extracción de datos tiene dos partes, la primera, con los metadatos de cada estudio primario: título, autores, tipo de publicación, nombre de la conferencia/revista, año, etc. Y la segunda, con las dimensiones y categorías del esquema definido para clasificar los estudios primarios seleccionados y respon-

der a las preguntas de investigación definidas. Este formulario se encuentra disponible en la siguiente url: `www.dei.uc.edu.py/proyectos/mddplus/documentos/exploring-m2m-ria-by-sms/`.

El proceso de extracción y categorización fue realizado por uno de los autores, mientras que los otros aportaron información para resolver ambigüedades durante el proceso y comprobaron la correctitud del proceso de extracción/categorización remontando la información expuesta en los formularios de extracción hasta las sentencias en algunos de los trabajos analizados, los cuales fueron seleccionados al azar. Tener otro autor revisando el proceso de extracción es una práctica común en las revisiones sistemáticas para las ciencias sociales [46].

El esquema definido para la clasificación de los estudios primarios, se establece tomando en consideración cada una de las preguntas de investigación (ver tabla 3.1) y la literatura relevante (ver tabla 3.4), para así restringir las respuestas posibles a cada una de las preguntas de investigación.

Al definir las posibles categorías para cada una de las preguntas de investigación se hizo evidente que algunas ellas no son mutuamente excluyentes, ya que, en algunos casos, las propuestas bajo análisis pueden ser clasificadas en más de una categoría al mismo tiempo. Un claro ejemplo de esto son los niveles de abstracción definidos por MDA. Es muy común, de hecho es la norma, que las propuestas contemplen más de una fase (no olvidemos que la definición y transformación de modelos con diferente nivel de abstracción es uno de los pilares de MDD, y por tanto, de MDA, la propuesta MDD más extendida [11]). Además, en algunas ocasiones la información proporcionada en los artículos es tan escasa que no permite la clasificación de la propuesta en ninguna de las categorías definidas para cada una de las preguntas de investigación.

Para reflejar este hecho en los formularios de extracción, y para facilitar su comprensión, han sido definidas categorías especiales cuya finalidad es la de capturar las intersecciones entre categorías, buscando de esta manera que las categorías establecidas sean lo más puras posible. Así, volviendo al ejemplo de las fases propuestas por MDA, en lugar de sólo las categorías CIM, PIM y PSM, definimos las siguientes: CIM; PIM; PSM; CIM y PIM en simultáneo; PIM y PSM en simultáneo; CIM, PIM y PSM en simultáneo; Otros, en caso de que se propusiera una nueva fase; Y ninguno, una categoría que nos permite agrupar propuestas que no pueden agruparse en ninguna de las otras categorías, por ejemplo, cuando la información en los artículos no permita identificar correctamente las fases abordadas.

3.1.1.6. Síntesis de los datos extraídos

En primer lugar, se ha realizado una síntesis cuantitativa, considerando el número y/o porcentaje de artículos en cada dimensión/categoría, e ilustrando estos datos mediante tablas y/o gráficos. La idea de esto es dar respuesta a cada pregunta de investigación a través de una correspondencia uno a uno entre pregunta y dimensión. Sin embargo, también se ha considerado el cruce de dimensiones cuando resultó ser oportuno.

Finalmente, se ha añadido una síntesis narrativa, la cual se encuentra enmarcada dentro lo que sería una síntesis de tipo cualitativa. Esta síntesis narrativa, como se menciona en [41], consiste en la elaboración de un resumen narrativo (en lugar de uno estadístico) de los resultados obtenidos en los estudios primarios. Es un marco general

Dimensión	Categoría
RQ1: Objetivo de la investigación	Evaluar, Mejorar, Proponer. Evaluar y mejorar, Evaluar y proponer.
RQ2: Tipo de evidencia / método de investigación	**Validados:** experimento, caso de estudio, encuesta, SLR y/o SMS. **No validados:** especulación, ejemplos y sus variantes, proyectos de clase.
RQ3: Tipo de resultado de la investigación	Método (técnica, metodología, proceso), Herramienta, Método y herramienta.
RQ4: Fases de MDA	CIM (**C**omputational **I**ndependent **M**odel), PIM, PSM, Combinaciones entre CIM, PIM y PSM, Ninguna.
RQ5: MDA puro vs enfoques MDD	MDA, MDD.
RQ6: Grado de desarrollo de la herramienta propuesta	Nivel 0: no se propone herramienta alguna, Nivel 1: definición teórica o trabajo futuro, Nivel 2: implementación parcial, Nivel 3: implementación completa, Nivel 4: herramienta disponible, Nivel 5: herr. y documentación disponibles.
RQ7: IDE propuesto	Eclipse, MagicDraw, Otro.
RQ8: Grado de automatización de la transformación M2M	Automática, Semi-automática, Manual, Automática and semi-automática, Semi-automática y manual.
RQ9: Clasificación según el nivel de abstracción de los modelos origen y destino	Transformación M2M horizontal, Transformación M2M vertical.
RQ10: Clasificación según el lenguaje de los modelos origen y destino	Transformación M2M endógena, Transformación M2M exógena.
RQ11: Relación entre los modelos origen y destino	Se puede dar que el destino sea un nuevo modelo, se actualice el modelo origen, o ambas cosas.
RQ12: Número de modelos origen y destino	Uno a Uno, Uno a Muchos, Muchos a Uno, Muchos a Muchos.
RQ13: Lenguaje de las reglas de transformación M2M	ATL, QVT o una combinación de ambos, Otro lenguaje de transformación.

Tabla 3.4: Dimensión/Categoría

de descripciones narrativas y ordenamiento de evidencia primaria con comentarios e interpretaciones.

3.1.2. Etapa de realización

La aplicación del protocolo de revisión arrojó los resultados preliminares que pueden ser observados en la tabla 3.5.

Fuente de búsqueda	Etapa 1	Etapa 2	Etapa 3	Artículos
IEEE Xplore	33	15	9	[6, 47-54]
Springer Link	58	28	19	[5, 55-72]
ACM Digital Library	0	0	0	
ScienceDirect	4	2	0	
Scopus	37	7	2	[73, 74]
Sugerencia de expertos	3	3	3	[15, 75, 76]

Tabla 3.5: Resultados por fuente de búsqueda

Entre los años 2002 y 2015, excluyendo los 3 trabajos recomendados por los expertos, se encontraron 132 artículos aplicando la estrategia de búsqueda previamente definida. La búsqueda se realizó en títulos y resúmenes, salvo cuando las fuentes de búsqueda no lo permitieron, o cuando los resultados obtenidos fueron muy escasos, en cuyo caso la búsqueda involucró al texto completo. De estos 132 artículos, después de comprobar que todos cumplen con los criterios de inclusión, fueron conservados los 132 artículos científicos y se procedió al cierre de la primera etapa del procedimiento de selección de los estudios primarios.

En la segunda etapa, el procedimiento de selección involucró la toma de los 132 artículos previamente seleccionados y la aplicación, sobre cada uno de ellos, de los criterios de exclusión definidos para la revisión del título y del resumen. Al finalizar la segunda etapa, se conservaron 52 de los artículos seleccionados en la etapa anterior (siempre excluyendo los 3 artículos sugeridos por los expertos).

En la tercera etapa del procedimiento de selección de estudios primarios fueron aplicados, sobre los 52 artículos seleccionados en la etapa dos, los criterios de exclusión definidos para la revisión del texto completo. Lo que resulta en la exclusión de 22 artículos más. Así, al finalizar la tercera etapa, de la lista original de 132 artículos, sólo quedaron 30 (nuevamente excluyendo de las cuentas a los 3 artículos sugeridos por los expertos).

Una situación particular se dio con 3 propuestas que no fueron detectadas en la búsqueda realizada, pero que más tarde fueron agregadas gracias a la sugerencia de expertos: WebML4RIA [15], OOHDM-RIA [75] y RUX-Method [76]. Al parecer, ninguna de estas 3 propuestas apareció en la búsqueda realizada debido a que ninguna propone o contempla transformaciones M2M. Sin embargo, y en vista de que son propuestas MDD de gran relevancia en la actualidad, se consideró prudente incluirlas de todos modos. Finalmente, luego de esta última inclusión manual fundamentada en

la sugerencia de expertos, el número total de trabajos seleccionados crece a 33 (ver tabla 3.5).

Si bien la distribución de artículos por fuentes de búsqueda es llamativa y denota una marcada tendencia a favor de IEEE Xplore y Springer Link, distribuciones similares fueron observadas en otros trabajos sobre temas afines, dos ejemplos son [77] y [78].

3.2. Resultados

Los resultados globales, basados en el número de estudios primarios clasificados en cada una de las posibles respuestas (dimensiones/categorías) para las preguntas de investigación definidas, se muestran en la tabla 3.6. Finalmente, un análisis de los resultados obtenidos, orientado de manera independiente a cada una de las preguntas de investigación planteadas, es presentado en ulteriores sub-secciones.

Tabla 3.6: Resultados por preg. de investigación y categoría

PI	Categoría		#	Artículos
PI–1	Evaluar		0	
	Mejorar		3	[50, 54, 71]
	Proponer		15	[49, 51, 53, 56, 59, 60, 65-69, 72, 73, 75, 76]
	Evaluar y mejorar		1	[58]
	Evaluar y proponer		14	[5, 6, 15, 47, 48, 52, 55, 57, 61-64, 70, 74]
PI–2	Validados	Experimento	1	[57]
		Caso de estudio	14	[6, 15, 47, 48, 52, 55, 58, 61-65, 70, 74]
		SMS y/o SLR	0	
	No validados	Ejemplo	7	[49-51, 54, 56, 59, 75]
		Running example	4	[53, 65, 67, 69]
		Example scenario	1	[68]
		Proyecto de clase	1	[66]
		Sin validación	5	[60, 71-73, 76]
PI–3	Método		9	[47-50, 52-54, 69, 76]
	Herramienta		4	[58, 60, 66, 71]
	Método y herramienta		19	[5, 6, 15, 51, 55-57, 59, 61-65, 67, 68, 70, 72-74]
	Sin clasificar		1	[75]

continúa en la siguiente página ...

Tabla 3.6: Continuación de la página anterior

PI	Categoría		#	Artículos
PI–4	CIM		0	
	PIM		2	[15, 75]
	PSM		3	[60, 64, 65]
	CIM y PIM		2	[57, 72]
	PIM y PSM		17	[6, 48, 49, 52-55, 58, 59, 61-63, 66-69, 71]
	CIM, PIM y PSM		7	[5, 47, 50, 51, 56, 73, 74]
	Otra		0	
	Ninguna		2	[70, 76]
PI–5	MDA		15	[6, 47, 48, 50, 51, 53, 54, 56, 59, 61, 66, 67, 72, 74, 75]
	MDD		18	[5, 15, 49, 52, 55, 57, 58, 60, 62-65, 68-71, 73, 76]
PI–6	Nivel 0	Sin herramienta	3	[50, 52, 75]
		Usa herr. existentes	5	[47-49, 69, 76]
	Nivel 1		4	[51, 57, 63, 72]
	Nivel 2		4	[56, 67, 70, 71]
	Nivel 3		14	[53-55, 58-62, 64-66, 68, 73, 74]
	Nivel 4		1	[6]
	Nivel 5		2	[5, 15]
PI–7	Eclipse		18	[6, 15, 48, 52-55, 57, 58, 60, 62, 66, 67, 70-73, 76]
	MagicDraw		2	[5, 56]
	Otro		4	[47, 64, 68, 74]
	Sin clasificar		9	[49-51, 59, 61, 63, 65, 69, 75]
PI–8	Automática		8	[48, 50, 55, 59, 60, 62, 63, 73]
	Semi-automática		17	[5, 6, 47, 49, 51-54, 61, 64-67, 69-71, 76]
	Manual		1	[74]

continúa en la siguiente página ...

Tabla 3.6: Continuación de la página anterior

PI	Categoría	#	Artículos
	Automática y semi-automática	2	[56, 58]
	Semi-automática y manual	1	[57]
	Sin clasificar	4	[15, 68, 72, 75]
PI–9	Horizontal	2	[64, 65]
	Vertical	22	[6, 47-49, 51-54, 56-61, 63, 66, 67, 69-71, 73, 74]
	Ambas	3	[50, 55, 62]
	Sin clasificar	6	[5, 15, 68, 72, 75, 76]
PI–10	Endógenas	24	[6, 47-54, 56-63, 66, 67, 69-71, 73, 74]
	Exógenas	3	[55, 64, 65]
	Sin clasificar	6	[5, 15, 68, 72, 75, 76]
PI–11	Nuevo	24	[6, 47-54, 57, 58, 60-67, 69-71, 73, 74]
	Actualización	0	
	Ambos	3	[55, 56, 59]
	Sin clasificar	6	[5, 15, 68, 72, 75, 76]
PI–12	1 a 1	4	[55, 58, 66, 69]
	1 a *	5	[52, 56, 60, 61, 74]
	* a 1	3	[48, 59, 63]
	* a *	13	[6, 47, 49-51, 53, 54, 57, 64, 65, 67, 71, 73]
	Sin clasificar	8	[5, 15, 62, 68, 70, 72, 75, 76]
PI–13	ATL	12	[5, 48, 52, 55, 57, 59, 62, 63, 65, 66, 71, 73]
	QVT	13	[6, 47, 49, 51, 53, 54, 56, 58, 60, 61, 67, 69, 70]
	Otro	2	[75, 76]
	ATL y QVT	1	[50]
	Sin clasificar	5	[15, 64, 68, 72, 74]

Fin de la Tabla 3.6

PI–1: ¿Cuáles son los objetivos perseguidos en la investigación sobre MDD y las transformaciones M2M?

El propósito perseguido al plantear los objetivos de investigación fue determinar dónde se sitúa el interés de investigación sobre MDD orientado a RIA y, en consecuencia, averiguar qué áreas siguen siendo menos investigadas.

Como se puede observar en la tabla 3.6 y/o en la figura 3.2, la mayor parte del esfuerzo de investigación se centra en la creación de nuevas propuestas. Esto es además sumamente visible, ya que 29 de 33 de los trabajos seleccionados (el 88 %) fueron clasificadas en una de las siguientes categorías: "Proponer (15)" y "Evaluar y proponer (14)", y proponen nuevos métodos, la mayoría de las veces extendiendo/adaptando la notación PIM de enfoques MDD pre-existentes. Sólo una de las propuestas es multi-enfoque, ya que propone una solución que puede ser adoptada para cualquier enfoque MDD.

De los cuatro trabajos que muestran mejoras (cuatro considerando las dos categorías relacionadas,"Mejorar (3)" y "Evaluar y mejorar (1)"), dos lo hacen creando herramientas que pretenden facilitar, y sobre todo incentivar, la adopción de las propuestas abordadas. El artículo restante, por el contrario, plantea la unión de dos enfoques MDD pre-existentes, aspirando a unificar las mejores características de ambos.

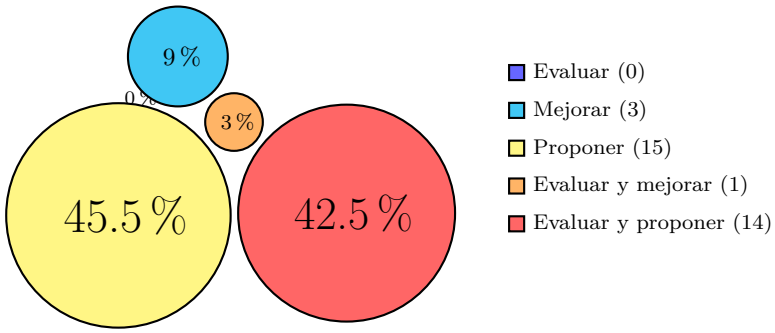


Figura 3.2: Porcentajes para PI–1

También es importante destacar los trabajos que se centran en la evaluación de otras propuestas. Hay 15 de ellos, considerando simultáneamente las categorías "Evaluar y mejorar (1)" y "Evaluar y proponer (14)", y han sido categorizados de esta manera porque presentan estudios de casos o experimentos como evidencia de la validez de las ideas propuestas. Y ambos métodos, tanto los estudios de caso como los experimentos, constituyen herramientas de validación con rigurosidad y solidez apropiadas.

PI–2: ¿Qué tipo de métodos se utilizaron en la investigación relacionada con MDD y las transformaciones M2M?

Los resultados según el método de investigación utilizado y la validación o no de las propuestas analizadas se muestran en la tabla 3.6 y en la figura 3.3. Básicamente los métodos de investigación contemplados como dimensión categoría para esta pregunta de investigación son considerados validados o no validados dependiendo de su

rigurosidad y solidez. Las propuestas validadas corresponden a 15 de los 33 trabajos, mientras que las no validadas representen 18 de los 33 trabajos, es decir, el 45.5 % y 54.5 % de los trabajos analizados, respectivamente.

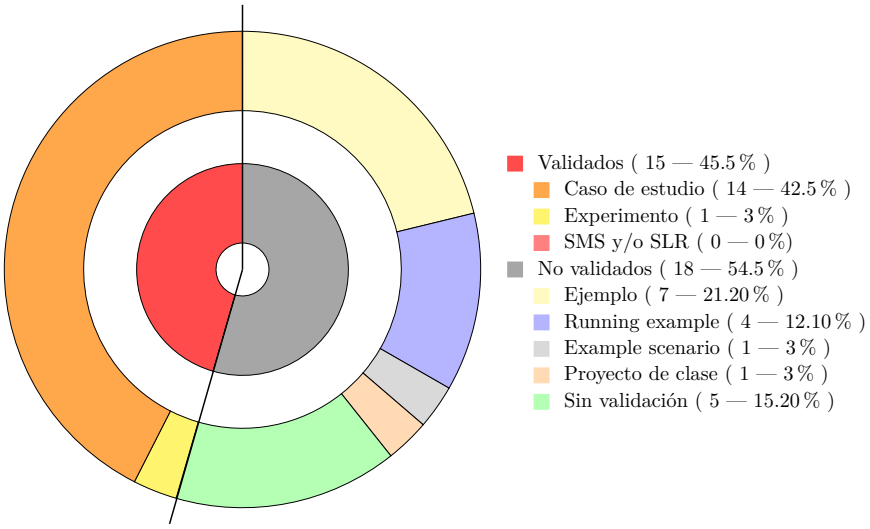


Figura 3.3: Porcentajes para PI–2

La categoría que agrupa los trabajos no validados incluye a aquellos que no muestran ningún tipo de evidencia experimental sobre las contribuciones de su propuesta, generalmente utilización de métodos tales como: ejemplos (y sus variantes), proyectos de clase, especulación, etc. Como bien se muestra en la figura 3.3, el ejemplo no validado es el método utilizado con mayor frecuencia, constituyendo 12 de los 18 trabajos no validados (12 considerando las tres categorías relacionadas con ejemplos, “Ejemplo”, “Running example” y “Example scenario”). De los cuales 7 (del total de 12) son ejemplos clásicos, 4 son ejemplos de ejecución (Running examples) y 1 es un escenario de ejemplo (Example scenario).

Así también, observando la figura 3.3, es posible notar que 5 artículos no presentan ningún tipo de evidencia que respalde sus aportes mientras que 1 artículo presenta como evidencia pruebas realizadas con alumnos (planteadas como un proyecto de clase).

La categoría que agrupa los trabajos validados, sin embargo, incluye a aquellos que sí muestran algún tipo de evidencia experimental sobre las contribuciones de su propuesta (ver la figura 3.3). En este caso el método experimental utilizado con mayor frecuencia es el estudio de caso, expuesto en 14 de los 15 artículos clasificados en la categoría/dimensión “Validados”. Mientras que, en sólo 1 de estos 15 trabajos clasificados como estudios validados, se observa un experimento como método de validación.

Como ya fue señalado en la introducción, al momento de realizar esta revisión, no fueron detectados ningún SMS ni ningún SLR en la literatura. Lo que de alguna manera refuerza la necesidad de realización de este estudio.

PI–3: ¿Cuál es el alcance de las propuestas en los trabajos seleccionados?

Después de verificar los datos presentados en la tabla 3.6 y/o en la figura 3.4, se puede observar que, efectivamente, el tipo de resultado más frecuente es la definición conjunta de un método y una herramienta que lo soporte, es decir, una herramienta que permita la utilización del método o metodología propuesto. Esto, así como ya fue mencionado previamente, constituye el planteamiento esperado, puesto que es la mejor manera de respaldar el aporte de las propuestas.

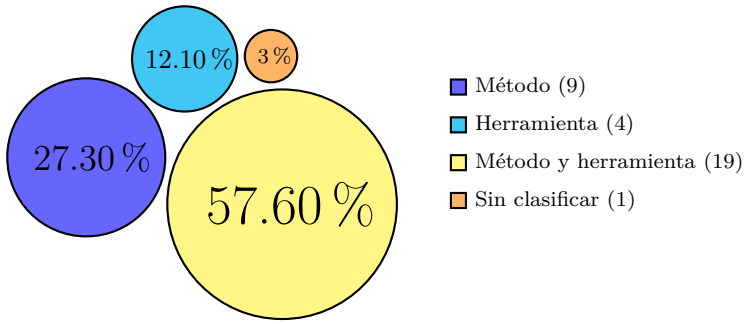


Figura 3.4: Porcentajes para PI–3

Nuevos métodos MDD para RIA son propuestos en 28 de los 33 artículos analizados (considerando de manera conjunta las categorías "Método (9)" y "Método y herramienta (19)"), es decir, el 85 % del total de estudios considerados en el SMS. De estos 28, 9 proponen únicamente un nuevo método mientras que en 19 son propuestos tanto método como herramienta de soporte. En contraste, los artículos que presentan únicamente nuevas herramientas son solo 4, y 1 pudo ser categorizado a partir de la información proporcionada en el trabajo analizado. Vea la figura 3.4 para conocer el análisis porcentual de esta pregunta de investigación.

PI–4: ¿Cuáles de las fases de modelado son contempladas?

Si bien el estándar MDA, la propuesta MDD más extendida actualmente, pregona la utilización simultánea de las fases CIM, PIM y PSM en una misma propuesta (con el objetivo de ir disminuyendo progresivamente el nivel de abstracción de los modelos hasta llegar finalmente al código), como podrá ser observado en breve, esto no se da con mucha frecuencia.

Procediendo al análisis de los datos obtenidos, presentados en la tabla 3.6 y en la figura 3.5, es fácil ver que las fases más comúnmente utilizadas son la PIM y la PSM, con 28 y 27 ocurrencias respectivamente (considerando todas las categorías relacionadas con alguna de ellas). Sin embargo, si se toma en consideración en cuántos trabajos ambas fases son utilizadas de manera conjunta, algo crucial para la aplicación de las transformaciones M2M según nuestras pretensiones, se puede observar que el número de ocurrencias desciende a 24 de 33, es decir, un 73 % solamente.

En contrapartida, solo 9 de las 33 propuestas consideran la fase CIM, nuevamente sumando todas las propuestas en las categorías/dimensiones relacionadas con esta fase. Además, solo en 2 de 33 propuestas, las fases CIM y PIM son utilizadas en

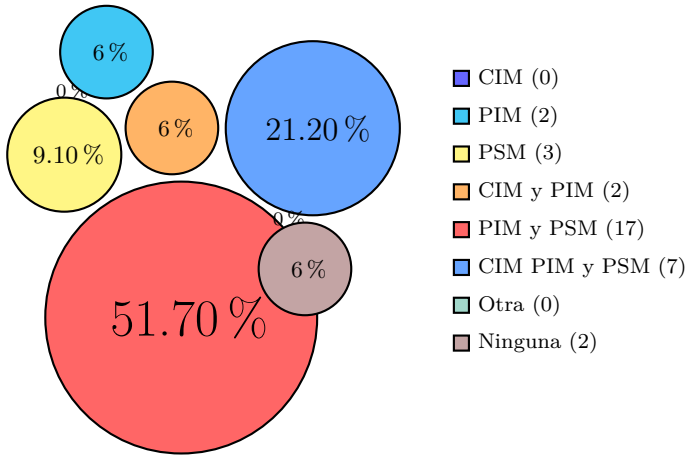


Figura 3.5: Porcentajes para PI–4

conjunto, mientras que solo en 7 más (tan solo un 22.20 %) son utilizadas de manera conjunta las tres fases propuestas por MDA, es decir, las fases CIM, PIM y PSM. Incluso, en 2 de las propuestas analizadas no se consideran ninguna de las fases propuestas por MDA, algo bastante llamativo y completamente inesperado. Por lo tanto, en base a todo lo mencionado previamente, es bastante razonable concluir que aún faltan desarrollar propuestas que sigan estrictamente las normas establecidas por el estándar MDA.

PI–5: ¿La propuesta sigue o no el estándar MDA?

Como bien puede observarse en la tabla 3.6 y/o en la figura 3.6, del total de 33 propuestas analizadas, 15 de ellas, el 45 %, pueden ser clasificadas en la categoría MDA, la cual agrupa todas aquellas propuestas que reportan, explícitamente, seguir este estándar. Aunque en principio esto parece un porcentaje bajo, la verdad es que constituye un muy buen porcentaje considerando que se trata de una sola de las propuestas MDD existentes.

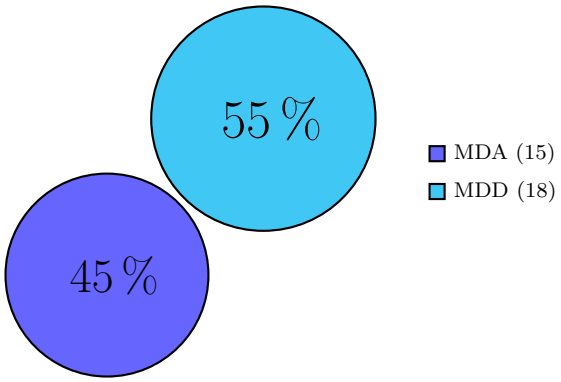


Figura 3.6: Porcentajes para PI–5

Además, dado que las propuestas clasificadas como MDA puro son solo aquellas

que mencionan explícitamente que siguen esta norma, y considerando las características de muchas de las propuestas clasificadas como MDD (18 de 33), concluimos que es sumamente probable que la tasa real de adopción del estándar MDA sea mucho mayor. Creemos que, simplemente, muchas de las propuestas se rigen por lo propuesto por MDA pero no hacen una mención explícita de ello.

PI–6: ¿La propuesta cuenta con herramientas que la soporten? ¿Cuán desarrolladas están?

Al analizar la cantidad de herramientas de soporte a MDD disponibles, el panorama no es muy alentador, ya que, como bien puede observarse en la tabla 3.6 y/o en la figura 3.7, el 24.20 % de los artículos ni siquiera menciona el desarrollo de una herramienta que permita aplicar la propuesta planteada. El 12.10 % solo presenta una definición teórica de la herramienta e igual porcentaje solo una implementación parcial de la misma, con lo mínimo necesario para demostrar la solución propuesta. Aunque, y es importante recalcar esto, estas dos últimas categorías siempre señalan la finalización de la herramienta como trabajo futuro.

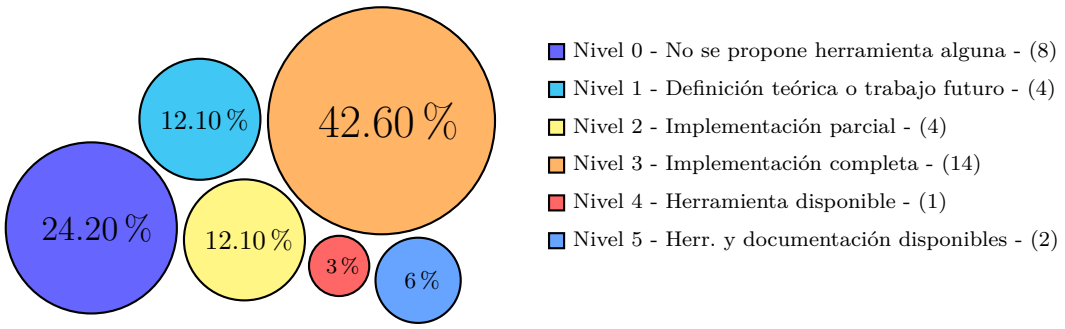


Figura 3.7: Porcentajes para PI–6

Con esto tenemos que el 48.40 % de las propuestas no cuenta con una herramienta que permita su aplicación/utilización (niveles 0, 1 y 2). Y la situación es todavía peor si consideramos el 42.60 % que presume herramientas completamente funcionales pero que, sin embargo, no facilitan el acceso a las mismas.

Finalmente, tan solo 3 de las 33 propuestas analizadas, apenas el 9 %, cuentan con herramientas de fácil acceso. Herramientas que pueden ser descargadas y utilizadas sin inconvenientes. Por otro lado, de estas tres herramientas accesibles, una no cuenta con tutoriales, ni guías, ni cualquier otro tipo de material instructivo o documentación (ver tabla 3.6). Las otras dos, sin embargo, sí son herramientas que pueden ser utilizadas sin problemas para experimentar el verdadero poder de MDD, y cuentan con suficientes guías y recursos como para que cualquier persona las utilice. Éstas son: WebRatio (desarrollada para dar soporte a “WebML4RIA” [15]) y MagicUWE (desarrollada para dar soporte a “UWE for RIA” [5]).

PI–7: ¿Cuál es el IDE señalado para el uso de la herramienta propuesta?

Esta pregunta de investigación busca descubrir cuál es el IDE más utilizado en el mundo MDD. Recordemos que la definición de potentes herramientas que soporten todo el proceso de desarrollo es uno de los pilares de MDD (y por tanto de MDA, la principal propuesta MDD) y un aspecto fundamental para su adopción por parte de la industria.

Observando los datos presentados en la tabla 3.6, podemos notar que la gran mayoría de las herramientas son desarrolladas a partir del IDE Eclipse. Concretamente, 18 de las 24 propuestas que detallan el IDE utilizado como base para el desarrollo de una herramienta, se basan en este popular entorno de desarrollo (ver figura 3.8).

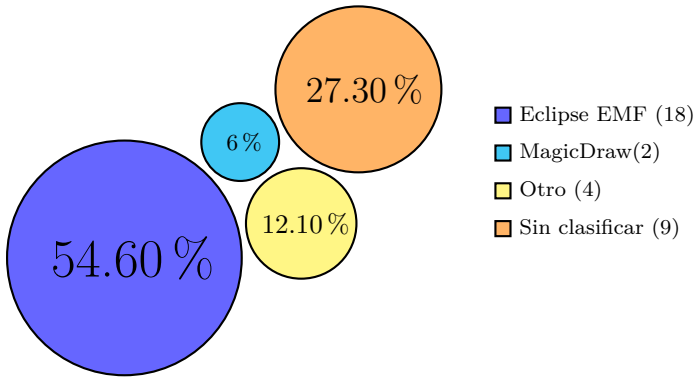


Figura 3.8: Porcentajes para PI–7

De las 6 propuestas restantes, 2 adoptaron el IDE llamado MagicDraw como punto de partida para el desarrollo de su herramienta y 4 adoptaron otros IDEs o no se basaron en ninguno pre-existente, desarrollando uno desde cero. Además de todo esto, 9 propuestas quedaron sin poder ser clasificadas debido a la escasa información presente en los artículos, información referente al IDE utilizado como punto de partida para el desarrollo de la herramienta propuesta.

Con el fin de facilitar la comprensión de la situación planteada en los párrafos precedentes, en la figura 3.8 se presenta un análisis porcentual de la misma.

PI–8: ¿Cuál es el nivel de automatización de la transformación M2M?

Al analizar los datos referentes al estado de automatización de las propuestas seleccionadas (datos presentados en la tabla 3.6) se puede apreciar que de las 33 propuestas, tan solo 8 presumen una automatización completa, lo que equivale apenas al 24.30 % de las mismas (ver figura 3.9).

El resto de los artículos se distribuyen de la siguiente manera. El 51.60 % de las propuestas analizadas, es decir, 17 de las 33, contemplan exclusivamente transformaciones M2M semi-automáticas, mientras que 1 sola propuesta (el 3 %) contempla una combinación de transformaciones manuales y semi-automáticas, 2 (el 6 %) una combinación de transformaciones semi-automáticas y automáticas, y 6 (el 12.10 %) no pudieron ser clasificadas en ninguna de las categorías señaladas a partir de la información proveída en sus correspondientes artículos.

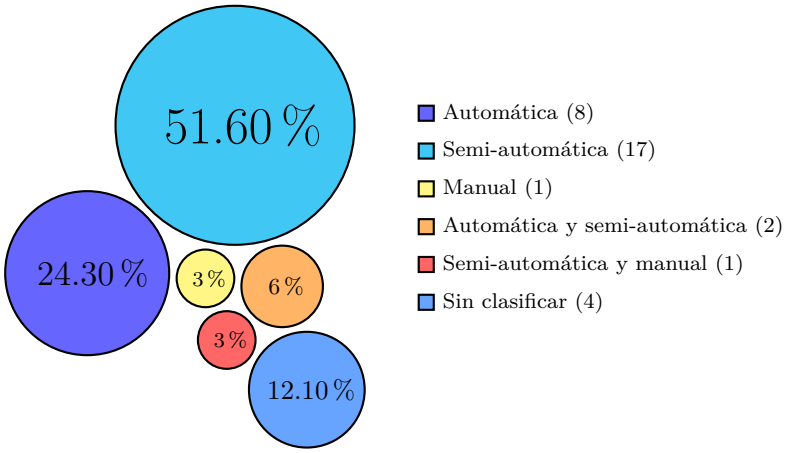


Figura 3.9: Porcentajes para PI–8

PI–9: ¿Pertenecen los modelos origen y destino, al mismo nivel de abstracción?

De acuerdo a la taxonomía propuesta en [79], las transformaciones pueden clasificarse, de acuerdo al nivel de abstracción de los modelos origen y destino, en verticales y horizontales. Se considera como horizontal una transformación M2M cuyos modelos origen y destino pertenecen al mismo nivel de abstracción. Y una transformación M2M vertical a aquella cuyos modelos origen y destino pertenecen a diferentes niveles de abstracción.

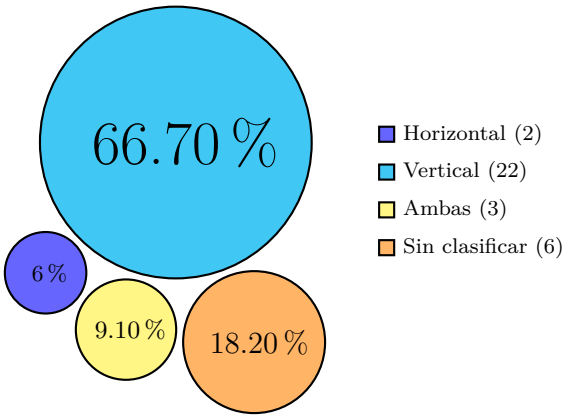


Figura 3.10: Porcentajes para PI–9

Siguiendo esta clasificación se pudo identificar que 5 propuestas presentan transformaciones M2M horizontales mientras que 25 propuestas utilizan transformaciones M2M verticales, esto considerando también, en cada caso, las propuestas que pueden ser clasificadas en ambas categorías. La figura 3.10, por otro lado, presenta la clasificación sin tener en cuenta este aspecto.

Como es posible observar en la figura 3.10, la combinación de las mencionadas categorías no se corresponde con el total de 33 propuestas debido a que algunas de ellas,

el 18.20 % para ser precisos, no permiten la identificación de tipo de transformación M2M abordado a partir de la información proporcionada en los respectivos trabajos.

El objetivo perseguido con el planteamiento de esta pregunta de investigación fue, simplemente, mejorar el conocimiento acerca del tipo de transformación M2M abordado en cada una de las propuestas analizadas. Y, como puede verse en la figura 3.10, la mayoría de las propuestas analizadas contemplan transformaciones M2M verticales.

PI–10: ¿En qué lenguaje se modelan los modelos origen y destino?

En [79], se propone una clasificación de las transformaciones M2M de acuerdo al lenguaje de modelado en el que se expresan los modelos origen y destino. Así, las transformaciones M2M cuyos modelos origen y destino son expresados en el mismo lenguaje de modelado se denominan endógenas, mientras que aquellas transformaciones M2M cuyos modelos origen y destino son expresados en lenguajes de modelado diferentes, se denominan exógenas.

Como es posible observar en la tabla 3.6 y en la figura 3.11, siguiendo esta clasificación encontramos que 24 propuestas plantean transformaciones endógenas, mientras que tan solo 3 propuestas proponen transformaciones exógenas, ninguna de las cuales es vertical.

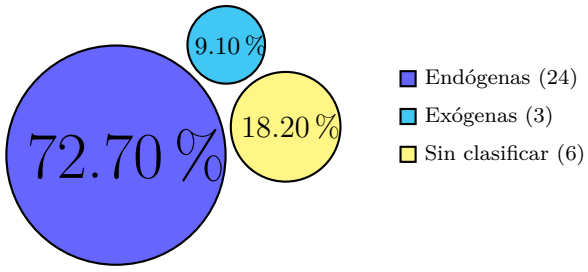


Figura 3.11: Porcentajes para PI–10

En este caso en particular, la sumatoria de ambas clasificaciones tampoco da el total de propuestas analizadas, es decir, 33. Esto se debe a que, simplemente, no fue posible deducir este dato con la información sonsacada de los artículos correspondientes.

PI–11: ¿Cuál es la relación entre los modelos origen y destino?

Según [80] es factible la clasificación de las transformaciones M2M de acuerdo a la relación existente entre los modelos origen y destino. Básicamente las transformaciones pueden ser de dos tipos, dependiendo de cómo se obtengan los modelos destino. Una opción es que el modelo destino sea uno totalmente nuevo y la otra es que el modelo destino sea creado a partir de la modificación del modelo origen.

Tanto en la tabla 3.6 como en la figura 3.12 se puede observar que 24 propuestas dan lugar a modelos destino totalmente nuevos, mientras que 3 obtienen los modelos destino mediante una combinación de ambas técnicas. Así también, al observar tanto la tabla como la figura mencionadas, se puede notar que 6 propuestas no pudieron ser clasificadas debido a la escasa información presente en los artículos (escasa información considerando este aspecto puntual de las transformaciones M2M).

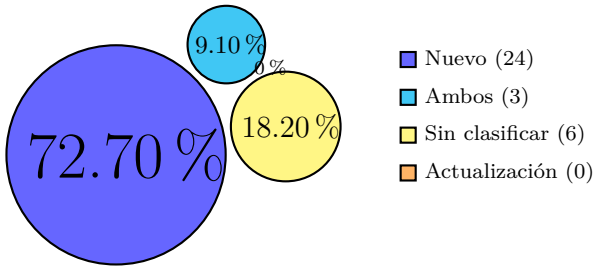


Figura 3.12: Porcentajes para PI–11

PI–12: ¿Qué relación existe entre el número de modelos origen y el número de modelos destino?

Otra posible clasificación para las transformaciones M2M, también definida en [79], permite clasificarlas según el número de modelos origen y modelos destino involucrados. En este caso tenemos cuatro opciones de clasificación: 1 a 1 (uno a uno), 1 a * (uno a muchos), * a 1 (muchos a uno), * a * (muchos a muchos).

Observando los datos obtenidos y dispuestos en la tabla 3.6, es fácil notar que la mayoría de las propuestas adoptan transformaciones M2M en las que están involucrados varios modelos origen y varios modelos destino. La proporción es la siguiente: de un total de 25 propuestas, 13 adoptan transformaciones M2M del tipo * a *, 3 adoptan transformaciones M2M del tipo * a 1, 5 adoptan transformaciones M2M del tipo 1 a * y 4 adoptan transformaciones M2M del tipo 1 a 1. Un análisis porcentual sobre esta situación se puede observar en la figura 3.13.

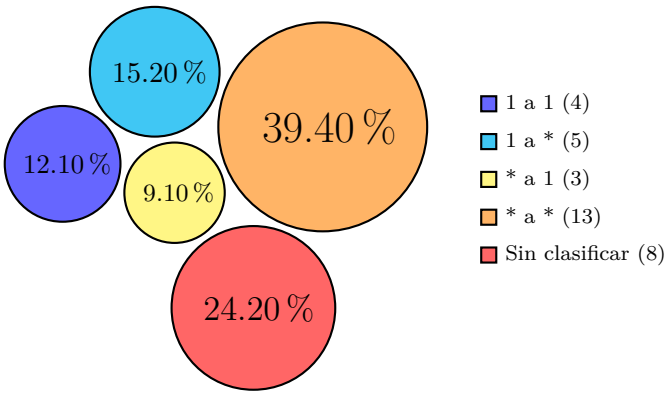


Figura 3.13: Porcentajes para PI–12

La razón por la que no fueron categorizadas las 33 propuestas analizadas es que, en muchos casos, partiendo únicamente de los datos proveídos por los artículos en los que fueron publicados, no fue posible deducir el tipo de transformación planteada por cada una de las propuestas.

PI–13: ¿Qué lenguaje de transformación M2M utiliza la propuesta analizada?

Orientando el análisis hacia la identificación del lenguaje de transformación M2M utilizado en cada una de las propuestas analizadas, se hace evidente el predominio de dos lenguajes de transformación M2M, ATL y QVT. Los datos muestran que ambos fueron adoptados, de manera simultánea, en 1 sola propuesta. Sin embargo, cuando fueron utilizados de manera separada, ATL fue adoptado por 12 propuestas y QVT por 13 propuestas.

Tan solo 2 propuestas utilizan lenguajes diferentes a ATL y QVT para definir sus transformaciones M2M (ver tabla 3.6 y/o figura 3.14). En ambos casos fue utilizado un lenguaje de transformación denominado XSLT (XSL Transformation), un estándar de la organización W3C (World Wide Web Consortium). XSLT fue utilizado, generalmente, para la transformación de documentos XML.

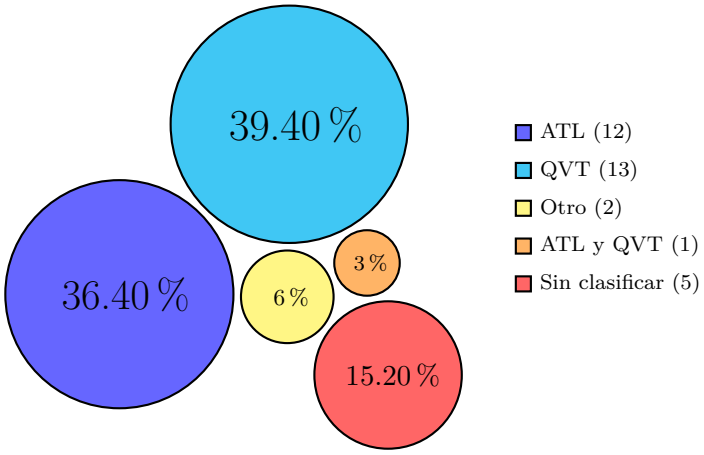


Figura 3.14: Porcentajes para PI–13

En este caso, al igual que al interpretar la pregunta anterior, fueron clasificadas tan solo 28 propuestas en lugar de 33. Es por esto que, al igual que en casos anteriores y para facilitar la lectura de los datos, fue creada una categoría especial con el objeto de agrupar aquellos artículos que, debido a la escasa información que proveen al respecto, no permitieron determinar el lenguaje de transformación utilizado en sus propuestas.

3.3. Interpretación de los resultados

Una vez realizada la síntesis cuantitativa, se procedió a realizar una interpretación de la misma con el fin de explicar con más detalle los resultados obtenidos. Los resultados cuantitativos de cada una de las preguntas de investigación son interpretados/analizados a continuación.

PI–1: ¿Cuáles son los objetivos perseguidos en la investigación sobre MDD y las transformaciones M2M?

Los resultados muestran una clara tendencia de la comunidad científica hacia la creación de nuevas propuestas en lugar de hacia la mejora o evaluación de propuestas pre-existentes.

Si bien la enorme diversidad en cuanto a propuestas denota la importancia de MDD para la comunidad científica, también conlleva aspectos negativos derivados de esta división del esfuerzo. Algunos de los aspectos negativos de mayor relevancia son:

- En primer lugar, la existencia de muchas propuestas inconclusas, propuestas que proponen ideas muy innovadoras, con gran proyección en cuanto a la utilidad que pudieran tener, pero que por falta de seguimiento por parte de la comunidad, no son desarrolladas completamente, y por ende, no llegan a la industria, perdiéndose de esta manera muchas buenas ideas.
- En segundo lugar, el correspondiente retardo en el desarrollo del campo, ya que al estar tan divida la comunidad científica se hacen complicadas hasta las decisiones más simples.

PI–2: ¿Qué tipo de métodos se utilizaron en la investigación relacionada con MDD y las transformaciones M2M?

El hallazgo de que sólo el 45.50 % de los artículos evalúa sus propuestas mediante validaciones rigurosas y formales es bastante llamativo. La mayoría de los artículos emplean uno o más ejemplos para ilustrar el problema investigado y la solución propuesta, pero estos ejemplos no pueden ser considerados como evaluaciones rigurosas de la contribución hecha.

Aunque es común que en los artículos extraídos de conferencias y talleres, los cuales suelen tener restricciones de longitud, se presente el problema investigado y se proponga una solución para abordarlo, dejando la validación formal como trabajo futuro, no tenemos forma de confirmar que este sea el motivo real de la escasa existencia de este tipo de evaluación.

Lo concreto es que esta situación se da y, por lo tanto, la validación formal de las propuestas MDD para arquitecturas RIA es claramente una oportunidad de investigación que podría ser abordada en un futuro.

PI–3: ¿Cuál es el alcance de las propuestas en los trabajos seleccionados?

El análisis de los datos referentes a esta pregunta de investigación no hace más que afianzar lo señalado al interpretar la pregunta anterior.

El hecho que el 85 % de los artículos propongan nuevas metodologías denota que aún no se ha creado una metodología que satisfaga todas las necesidades de MDD. Así también, el elevado número de metodologías propuestas confirma la brecha de investigación detectada al analizar la pregunta de investigación anterior.

Por lo tanto, sería un gran aporte, por ejemplo, tomar alguna de las metodologías existentes, alguna que aún esté inconclusa o que aún no haya sido validada adecuadamente, para luego completarla o dotarla de una herramienta que permita su aplicación

y posterior validación formal.

PI–4: ¿Cuáles de las fases de modelado son contempladas?

Los resultados muestran que el 51.70 % de las propuestas consideran las fases PIM y PSM simultáneamente, mientras que tan solo el 21.20 % de ellas contemplan todas las fases señaladas por el estándar MDA, es decir, las fases CIM, PIM y PSM.

Si bien el porcentaje de adopción de las tres fases señaladas por el estándar MDA es muy bajo, el hecho de considerar simultáneamente las fases PIM y PSM es positivo, no al nivel de lo que sería la aplicación simultánea de las fases CIM, PIM y PSM, pero positivo al fin. De todas formas, aún no se alcanza un porcentaje de adopción que permita concluir una fuerte adopción de MDA.

A pesar de que poco más de la mitad de las propuestas analizadas consideran tanto la fase PIM como la fase PSM, lo que de alguna manera induce a pensar que las transformaciones M2M son abordadas por la misma cantidad de propuestas, esto no es así, el porcentaje de propuestas que realmente utilizan transformaciones M2M es muy bajo. Luego de un análisis pormenorizado de la situación, se llegó a la conclusión de que este particular resultado se debe a que algunas propuestas se encuentran en fases muy tempranas de su desarrollo y aún no lidian con el complejo problema de automatizar las transformaciones M2M. Algunas, por ejemplo, proponen meta-modelos adaptados a RIA y plantean situaciones en las que las ideas plasmadas serían de utilidad, pero no avanzan más allá, e incluso, optando algunas veces por omitir las transformaciones M2M previamente planificadas, utilizando directamente transformaciones M2T en su lugar, y dejando las transformaciones M2M como trabajo futuro.

Otro factor influyente en este resultado son los artículos incluidos gracias a la sugerencia de expertos. Resulta que estos artículos no contemplan transformaciones M2M, sino solo transformaciones M2T o T2T. Se las incluyó, o bien por ser de las más extendidas actualmente, o bien por ser de las más completas y de las pocas en contar con herramientas comerciales.

PI–5: ¿La propuesta sigue o no el estándar MDA?

El porcentaje de propuestas que mencionan explícitamente seguir lo estipulado por el estándar MDA es de 45 % (15 de los 33 artículos analizados), lo cual es un muy buen resultado considerando que MDA constituye una sola las propuestas MDD existentes en la actualidad.

Además, al cruzar estos datos con los obtenidos en otras preguntas de investigación; por ejemplo, la pregunta de investigación número 4: ¿Cuáles de las fases de modelado son contempladas?, la pregunta de investigación número 9: ¿Pertenecen los modelos origen y destino, al mismo nivel de abstracción? y la pregunta de investigación número 10: ¿En qué lenguaje se modelan los modelos origen y destino?; es razonable concluir que el verdadero porcentaje de adopción del estándar MDA es mayor que el 45 % detectado en este SMS. Probablemente lo que está ocurriendo es que, simplemente, algunas de las propuestas analizadas no mencionan explícitamente la adopción del estándar MDA y, por lo tanto, probablemente muchas de las propuestas que han sido clasificadas como enfoques MDD, realmente se rigen por lo que estipula MDA pero no hacen mención de ello.

PI–6: ¿La propuesta cuenta con herramientas que la soporten? ¿Cuán desarrolladas están?

El objetivo perseguido con el planteamiento de esta pregunta de investigación es determinar el grado de desarrollo de cada una de las propuestas analizadas.

Es así que, considerando toda la información obtenida, se ha arribado a la conclusión de que, en general, las propuestas MDD para RIA se encuentran en etapas muy tempranas de su desarrollo, salvo contadas excepciones (e.g., OOH4RIA [6], WebML4RIA [15] y UWE para RIA [5]).

Esta escasa disponibilidad de herramientas puede deberse al hecho de que la publicación analizada haya sido realizada en una etapa muy prematura del desarrollo de la propuesta. Lo que es bastante factible ya que varias de las propuestas que sí cuentan con herramientas que permiten probarlas y/o utilizarlas, tuvieron un ciclo de vida similar, es decir, las primeras publicaciones no mencionan nada acerca de una herramienta pero, sin embargo, esta es presentada en publicaciones posteriores.

Dado que muchas de las propuestas llevan años de ser publicadas, es probable que el principal motivo para la marcada escasez de herramientas sea simplemente una falta de continuidad de la comunidad científica para con el desarrollo de las mismas. Esto, entre otras cosas, revela que una práctica habitual en el mundo académico consiste en el desarrollo de herramientas ad-hoc que permitan cubrir los análisis considerados en los diferentes estudios, sin dar mucha importancia al desarrollo de herramientas y su posterior distribución a la industria, hecho que, claramente, conllevaría grandes avances en cuanto a la adopción de MDD. Y esto abre una gran brecha de investigación que posibilita futuros trabajos en pos de mejorar y completar metodologías que aún no se encuentren completamente desarrolladas.

PI–7: ¿Cuál es el IDE señalado para el uso de la herramienta propuesta?

Sobre esto, casi la totalidad de las propuestas analizadas desarrollan herramientas creando plugins para Eclipse o modificando y redistribuyendo este IDE como si fuese uno diferente. La principal razón de esto podría ser la licencia adoptada por este IDE, la cual facilita considerablemente su modificación y adaptación. Aunque también es importante resaltar que, de todas formas, Eclipse es el IDE que más fuertemente apuesta por el enfoque MDD.

En segundo lugar en cuanto a la cantidad de propuestas que lo adoptan, se encuentra el IDE denominado MagicDraw, un IDE muy potente pero que, lastimosamente, solo cuenta con versiones de pago y además es cerrado.

Sin embargo, también existen propuestas que desarrollan herramientas completamente nuevas. Un resultado que llama mucho la atención, sobre todo al considerar la complejidad y el arduo trabajo que implica esta decisión.

PI–8: ¿Cuál es el nivel de automatización de la transformación M2M?

Si bien apenas el 24.30 % de las propuestas presumen una automatización completa de las transformaciones M2M, este no es un resultado malo. De hecho, es un resultado perfectamente razonable ya que la automatización completa de las transformaciones

en particular y del proceso de desarrollo completo en general, es una tarea sumamente compleja.

Probablemente la complejidad de automatizar la totalidad de las transformaciones M2M involucradas en un proceso MDD sea el motivo por el cual una gran cantidad de artículos señalan esta automatización como trabajo futuro, limitándose a automatizar aspectos puntuales con el objetivo de validar solo los aspectos más relevantes de la solución planteada.

Un resultado que sí llama mucha la atención es la existencia de una propuesta que contempla transformaciones manuales. Aunque considerando que es posible demostrar los aportes de una propuesta mediante los ejemplos apropiados y transformaciones manuales, es aceptable que alguna publicación, por limitaciones de tiempo o espacio, recurra a ellas para demostrar ciertos puntos.

PI–9: ¿Pertenecen los modelos origen y destino, al mismo nivel de abstracción?

La respuesta a esta pregunta de investigación permite identificar si las transformaciones realizadas modifican o no el nivel de abstracción de los modelos destino (con respecto a los modelos de origen). Esta información es relevante porque cuando los modelos origen y destino de la transformación M2M residen en el mismo nivel de abstracción, es decir, cuando la transformación M2M es horizontal y, además, la propuesta bajo análisis propone únicamente este tipo de transformaciones, se posible concluir que la propuesta ciertamente no sigue las directrices propuestas por MDA.

Aun cuando la presencia de transformaciones M2M verticales (aquellas que disminuyen progresivamente el nivel de abstracción de los modelos) no se puede utilizar para predecir el grado de adopción de MDA. Teniendo en cuenta que MDA sugiere la utilización de transformaciones M2M verticales entre capas con diferente nivel de abstracción y transformaciones M2M horizontales en el interior de cada una de ellas [81], el hecho de que la mayoría de las propuestas hayan adoptado este tipo de transformación M2M, sumado a los resultados obtenidos en la pregunta de investigación número 5, sobre las fases MDA adoptadas, permite concluir que los principios propuestos por MDA están siendo tenidos en cuenta por la comunidad científica, al menos parcialmente. Todo esto permite prever un alentador futuro para MDA, a pesar de que, por el momento, su grado de adopción no sea el adecuado.

PI–10: ¿En qué lenguaje se modelan los modelos origen y destino?

Ésta pregunta de investigación solo admite dos respuestas posibles. Las transformaciones M2M pueden ser, o endógenas o exógenas. Al analizar los datos obtenidos en el presente trabajo se observa una clara preferencia, por parte de la comunidad científica, para con las transformaciones M2M endógenas, es decir, aquellas en las que el modelo origen y el modelo destino están expresados en el mismo lenguaje de modelado.

Este resultado refuerza la hipótesis que presagia un prometedor futuro para el estándar MDA, ya que, aunque la adopción de este tipo de transformaciones no es obligatoria para la adopción de MDA, el análisis de los resultados obtenidos en este estudio permite conjeturar que contribuye a su adopción, sobre todo debido a que las

propuestas que utilizan transformaciones M2M exógenas tienden a alejarse bastante de lo estipulado por este estándar. Por ejemplo, las tres transformaciones que utilizan transformaciones M2M exógenas, también utilizan transformaciones M2M horizontales, además de lenguajes de modelado que no constituyen propuestas del OMG para el estándar MDA.

PI–11: ¿Cuál es la relación entre los modelos origen y destino?

En general, obtener modelos destino totalmente nuevos es más sencillo que modificar el modelo origen para ajustarlo a lo estipulado por las transformaciones M2M definidas y, efectivamente, según los datos obtenidos por el presente estudio, el tipo de transformación M2M más frecuente es justamente este.

A diferencia de las preguntas anteriores, esta no aporta mucho en pos de aproximar el grado de adopción de MDA, sin embargo, es un criterio importante a la hora analizar el tipo de transformaciones M2M más frecuentemente utilizado en el mundo MDD orientado a arquitecturas RIA.

Entre otra cosas, esta pregunta permite sentar bases para futuros análisis sobre transformaciones M2M para RIA. Los datos obtenidos serán de gran utilidad para futuros trabajos que pretendan, por ejemplo, evaluar la evolución de estas transformaciones M2M.

PI–12: ¿Qué relación existe entre el número de modelos origen y el número de modelos destino?

Esta pregunta de investigación proporciona un criterio de clasificación de las transformaciones M2M que permite anticipar el grado de complejidad que tendrán las mismas al momento de ser definidas durante la adopción/utilización de cada una de las propuestas analizadas.

Las transformaciones M2M que contemplan un solo modelo origen para la producción de un solo modelo destino, transformaciones “1 a 1”, son más sencillas de definir y comprender, por lo que cabría esperar que la adopción de las mismas se ciña a una curva de aprendizaje más baja.

Sin embargo, este criterio solo aplica a la complejidad de las transformaciones en sí, ya que por ejemplo, es muy posible que las propuestas que contemplen transformaciones “1 a 1”, logren la mencionada sencillez en base a modelos con una menor separación de conceptos, es decir, definiendo menos y más complejos modelos, trasladando la complejidad de las transformaciones M2M a los modelos mismos, lo que impacta negativamente en la mantenibilidad y comprensibilidad de los diseños.

En base a los hechos presentados previamente, y a la luz de los resultados obtenidos, es posible concluir que, en general, la comunidad científica tiende a disminuir la complejidad de los modelos en lugar de disminuir la complejidad de las transformaciones M2M, ya que la mayoría de las propuestas utilizan transformaciones “* a 1” o “* a *”. Lo que es más que razonable, considerando que una importante motivación de MDD es, justamente, mejorar la documentación de los sistemas.

PI–13: ¿Qué lenguaje de transformación M2M utiliza la propuesta analizada?

Esta pregunta de investigación fue formulada con el objetivo de identificar cuál es el lenguaje de transformación más utilizado actualmente para la definición de las transformaciones M2M.

Los resultados obtenidos ponen en muy buena posición al lenguaje QVT, propuesto por el OMG y estándar oficial para la definición de las transformaciones M2M. Básicamente, el 39.40 % de las propuestas utiliza QVT, mientras el 36.40 % utiliza ATL, tomando como referencia aquellas propuestas que indican cuál es el lenguaje de transformación M2M que utilizan, es decir, 28 de las 33 propuestas analizadas. Lo cual, por un lado demuestra el gran acogimiento de ATL como lenguaje de transformación M2M y avala a numerosos autores que consideran a ATL como el estándar de facto para la definición de transformaciones M2M, mientras por el otro, demuestra el gran esfuerzo de la comunidad científica por incentivar la utilización de QVT.

3.4. Amenazas a la validez del SMS

Esta sección describe las cuestiones que deben ser mejoradas en futuras replicaciones de este SMS, así como también otros aspectos que deben ser tenidos en cuenta para la generalización de los resultados obtenidos.

Petersen y Gencel [82] analizaron los diferentes esquemas de clasificación de la validez existentes en la actualidad y discutieron su aplicabilidad a la ingeniería del software. Concluyendo que deben tenerse en cuenta los siguientes tipos de validez: la validez descriptiva, la validez teórica, la validez generalizadora y la validez interpretativa. Aunque en este estudio también ha sido incluido un análisis pormenorizado de la repetibilidad del SMS.

3.4.1. Validez descriptiva

La validez descriptiva es la medida en que las observaciones se describen con precisión y objetividad, es decir, el rigor con el que se documenta y se presenta el conjunto de datos e informaciones utilizados. Las amenazas a la validez descriptiva son generalmente mayores en estudios cualitativos que en estudios cuantitativos.

Para reducir esta amenaza, se ha diseñado un formulario de recolección de datos, de manera a normalizar la extracción y el registro de los mismos. Este formulario tiene un gran impacto en la mejora de la objetividad del proceso de extracción de datos ya que puede ser revisado y evaluado en todo momento. Por lo tanto, esta amenaza a la validez del SMS se considera bajo control.

3.4.2. Validez teórica

La validez teórica es un análisis más abstracto que la validez descriptiva, básicamente mide la capacidad de capturar lo que se pretende capturar. Además, factores de confusión tales como sesgos y la selección de sujetos juegan un papel importante en su análisis.

3.4.2.1. Amenaza de sesgo en la selección

El sesgo en la selección se refiere a la distorsión que pudiera tener el análisis a causa de los criterios utilizados en la selección de las publicaciones. Se ha intentado aliviar esta amenaza, al menos en cierta medida, definiendo los criterios de inclusión y exclusión de manera que permitan reunir la mayor cantidad posible de documentos relacionados con las transformaciones M2M para arquitecturas RIA.

3.4.2.2. Amenaza de literatura faltante

Debido a que en [42] se menciona que, en algunos casos, una cadena de búsqueda sencilla y simple puede ser tan efectiva como una cadena de búsqueda compleja, en este SMS se optó por utilizar una cadena de búsqueda sencilla, conformada por los términos más frecuentes en la literatura.

La selección de los términos de búsqueda y de las bibliotecas digitales utilizadas podría impedir que algunos estudios relevantes aparezcan entre los resultados de las búsquedas. Así, por ejemplo, los estudios publicados en actas de conferencias que no son indexadas por las bibliotecas digitales seleccionadas no aparecerán entre los resultados de las búsquedas realizadas.

3.4.2.3. Amenaza de sesgo en la publicación

El sesgo en la publicación se refiere al problema de que los resultados positivos tienen más probabilidades de ser publicados que los resultados negativos. Esta baja probabilidad de publicación de resultados negativos se debe a que, en general, los resultados negativos tardan más en ser publicados, y además, son citados con menor frecuencia en otras publicaciones [42].

En lo que respecta a esta amenaza, al menos hasta cierto punto, el hecho de no haber considerado la literatura gris (es decir, informes industriales o tesis doctorales o resultados no publicados), pudo haber afectado la validez de los resultados obtenidos. Aunque, por otro lado, el haber considerado en el SMS únicamente estudios sometidos a revisión de pares, actúa como filtro de calidad, o mejor aún, como criterio para considerar artículos de cierta calidad.

3.4.2.4. Amenaza de inexactitud en la extracción y clasificación

El proceso de extracción y categorización fue realizado por uno solo de los autores, mientras que los demás se limitaron a aportar información que permita resolver ambigüedades durante el proceso y revisar al azar el proceso de extracción y categorización. Con esto, consideramos que el proceso de extracción y categorización ha sido parcialmente validado. Además, dado que la validación parcial ha sido realizada por los autores de este estudio, puede sufrir de sesgo.

Finalmente, también vale la pena mencionar que no se ha realizado una evaluación sobre la calidad de las obras seleccionadas. En Genero et al. [41], se explica que la inclusión de este punto no es esencial en un SMS.

3.4.3. Validez generalizadora

La búsqueda realizada en el marco del presente SMS contempla las bases de datos más comunes y frecuentemente sugeridas de la literatura sobre ingeniería del software (vea el párrafo *fuentes de búsqueda* en la sub-sección 3.1.1.2 para acceder a la lista completa de las fuentes de búsqueda utilizadas). Además, dada la amplitud del análisis realizado y el elevado número de publicaciones consideradas, es razonable asumir que ha sido creado un conjunto de resultados generalizable. Sin embargo, esta suposición debe ser confirmada por medio de estudios posteriores.

3.4.4. Validez interpretativa

La validez interpretativa se logra cuando las conclusiones obtenidas son razonables dados los datos analizados, y por lo tanto permite asumir la validez de la conclusión. Una amenaza en la interpretación de los datos es el sesgo del investigador.

El sesgo del investigador se caracteriza por el diseño e implementación de experimentos que tienden a producir resultados favorables o deseables para los investigadores. El sesgo del investigador ocurre en todas las etapas del proceso. Incluso si el diseño del estudio es imparcial, la selección, extracción de datos, y las etapas de síntesis pueden distorsionar los resultados [83].

Para reducir las amenazas de validez interpretativa de este trabajo, todas las interpretaciones y conclusiones alcanzadas fueron minuciosamente validadas por todos los autores.

3.4.5. Repetibilidad

La repetibilidad se refiere a la posibilidad de lograr los mismos resultados repitiendo los mismos procesos.

Como la repetibilidad requiere de informes detallados sobre el proceso de investigación, en el marco de este estudio fue realizado un informe detallado sobre la planificación (el protocolo del mapeo) y el proceso de realización de este SMS, que incluye además gran parte de los datos extraídos.

La idea detrás de esto es permitir que otros investigadores cuenten con las herramientas necesarias para replicar este estudio, incluso han sido publicados los formularios de extracción, los cuales pueden ser accedidos aquí: `www.dei.uc.edu.py/proyectos/mddplus/documentos/exploring-m2m-ria-by-sms/`.

Además, la repetibilidad también se vio favorecida por el uso de las diferentes guías existentes para la realización de SMSs.

3.5. Observaciones finales y oportunidades

En el presente estudio ha sido llevado a cabo un SMS con el fin de reunir, clasificar y analizar todas las investigaciones relacionadas con enfoques MDD para arquitecturas RIA que hayan sido llevadas a cabo por la comunidad científica internacional entre el 2002, año de aparición del término RIA [39], y el 2015 (ambos inclusive).

Los principales objetivos perseguidos con la realización de este SMS, han sido los siguientes: i) proporcionar una consolidada visión general del campo de investigación; ii) identificar temas de investigación bien establecidos, tendencias, temas de investigación abiertos, y oportunidades de investigación que puedan o requieran ser abordados en el futuro.

Si bien el SMS revela varios datos interesantes, los hechos de mayor relevancia y trascendencia son lo que se mencionan a continuación, uno por cada pregunta de investigación formulada.

1. La mayor parte del esfuerzo de investigación de la comunidad científica se centra en crear o proponer nuevas metodologías en lugar de estudiar mejor o colaborar con propuestas pre-existentes.

2. Tan solo el 45.5 % de los artículos analizados respaldan sus afirmaciones con validaciones rigurosas y formales.

3. Si bien la mayoría de las propuestas presentan tanto métodos como herramientas, estas herramientas no son de fácil acceso o únicamente son mencionadas como trabajo futuro.

4. La mayoría de las propuestas adoptan las fases PIM y/o PSM, mientras que ninguna propone nuevas fases a las ya definidas por MDA.

5. Al menos el 45 % de las propuestas analizadas siguen fielmente lo estipulado por el estándar MDA.

6. Si bien muchas de las propuestas estudiadas proponen herramientas, pocas las desarrollan y publican.

7. El IDE más utilizado para el desarrollo de herramientas que soporten a los métodos propuestos, es Eclipse, ya sea creando nuevos IDEs a partir de este, como creando plugins para el Eclipse Modeling Framework (EMF).

8. Son pocas las herramientas que automatizan completamente el proceso de desarrollo, son más comunes los procesos semi-automáticos.

9. Según la taxonomía propuesta en [79], el tipo de transformación M2M más frecuente es la transformación M2M vertical.

10. Según la taxonomía propuesta en [79], el tipo de transformación M2M más frecuente es la transformación M2M endógena.

11. Según la taxonomía propuesta en [80], el tipo de transformación M2M más frecuente es aquella cuyos modelos destino son creados desde cero y no a partir de la modificación de los modelos origen.

12. Según la taxonomía propuesta en [79], el tipo de transformación M2M más frecuente es la transformación de muchos a muchos.

13. En cuanto al lenguaje de transformación M2M, el grado de adopción de ATL y QVT es prácticamente el mismo.

Además del análisis cuantitativo resultante del mapeo sistemático realizado, también fue llevado a cabo un análisis cualitativo con el fin de inferir algunas conclusiones a partir de la interpretación y el análisis de los datos recopilados para cada una de las preguntas de investigación planteadas. Los resultados de este análisis cualitativo, uno por cada pregunta de investigación, se detallan en la sección 3.3. Sin embargo, a continuación se mencionan aquellos aspectos más relevantes para la prosecución del presente trabajo de tesis.

En este sentido, el primer hecho importante de resaltar, y sin duda uno de los más significativos, es que ninguna de las propuestas analizadas propone nuevas fases a las ya definidas por el estándar MDA. Más aún, ninguna de estas propuestas contempla la fase ASM introducida por MoWebA [12] [9] [14].

Otro aspecto relevante es que, en general, las propuestas que extienden enfoques MDD pre-existentes para permitirles modelar aspectos específicos de las arquitecturas RIA, lo hacen extendiendo/adaptando la notación del PIM. Y por lo tanto, corriendo el riesgo de afectar negativamente la portabilidad de estos modelos (cuya portabilidad es uno de los pilares del paradigma MDD).

Si nos centramos en las preguntas de investigación 1 a 7, podemos notar que la mayoría de los estudios sobre MDD proponen nuevos enfoques, invirtiendo la mayor parte del esfuerzo en el desarrollo del enfoque, y por consiguiente, dejando la definición de una herramienta que lo soporte, como trabajo futuro. Inclusive, pudimos notar que muchas veces se dejan de lado las transformaciones M2M y se recurre, exclusivamente, a las transformaciones M2T. Es decir, debido fundamentalmente a que la comunidad científica todavía centra sus esfuerzos en definir nuevas propuestas MDD para arquitecturas RIA, existe una gran cantidad de propuestas inconclusas y el número de herramientas desarrolladas y distribuidas es aún muy bajo. En este punto es importante recodar que otro de los pilares de MDD es, justamente, la creación de potentes herramientas que permitan la automatización del proceso de desarrollo.

Si bien ninguno de estos aspectos amenaza la validez de las propuestas analizadas en el presente SMS, es probable que, de alguna manera, la ausencia de propuestas que incluyan transformaciones M2M haya contribuido a la aparición del problema de portabilidad del PIM. Problema que será analizado en la siguiente sección.

Ocurre que al reducir la cantidad de tiempo invertido en el desarrollo de herramientas, y sobre todo, al no considerar las transformaciones M2M, se fue haciendo poco evidente que extender la notación del PIM con elementos propios de una arquitectura en particular, no es la mejor solución para adoptarla. La razón es que al extender la notación del PIM, este se vuelve exclusivo para la arquitectura destino, y por lo tanto, pierde portabilidad con respecto a otras arquitecturas.

Más aún, considerando que en el marco de este estudio se considera al PIM como un elemento del dominio del problema y a la arquitectura como un elemento del dominio de la solución, pareciera ser más acertado liberar al PIM de aquellos detalles referentes

a la arquitectura, trasladándolos del dominio del problema (modelos CIM/PIM) al dominio de la solución (modelos ASM/PSM).

Resumiendo, después de estudiar minuciosamente cada una de las preguntas de investigación planteadas en el SMS, se observó cierta tendencia de la comunidad científica a dejar de lado algunos aspectos relevantes de MDD: i) las transformaciones M2M, ii) la portabilidad del PIM, iii) el desarrollo de herramientas que implementen transformaciones M2M, iv) las validaciones formales y, sobre todo, v) a otros enfoques MDD (puesto que es más común proponer un nuevo enfoque MDD desde cero, que mejorar o extender alguno pre-existente).

En base a todo lo anterior, fueron detectadas varias oportunidades de investigación. Entre ellas merecen una mención especial algunas que consideramos particularmente importantes para MDD. Algunas de las cuales incluso fueron abordadas por las siguientes etapas del presente proyecto final de carrera. Estas son:

- La oportunidad de contribuir con alguna de las tantas propuestas inconclusas, en lugar proponer constantemente nuevas metodologías.

- La necesidad de abordar la definición y utilización de reglas de transformación M2M. Debido a que muchas de las propuestas estudiadas las contemplan pero no las definen ni utilización, optando en su lugar por la definición y utilización de transformaciones M2T.

- La posibilidad de contribuir con el desarrollo de herramientas que soporten cada una de la propuestas existentes, lo que aporta a la comunidad en dos formas: i) facilitando la realización de validaciones formales; y, ii) fomentando la adopción de MDD por parte de la industria.

- La existencia y la falta de abordaje por parte de la comunidad científica, del problema de portabilidad del PIM. Derivado de la tendencia actual de las metodologías MDD existentes, a extender la notación del PIM con elementos propios de cada nueva arquitectura a ser abordada.

3.6. El problema de portabilidad del PIM

Una de las principales conclusiones del presente SMS es que, en general, tanto las propuestas que extienden enfoques MDD pre-existentes como aquellas creadas desde cero con el objetivo de modelar arquitecturas específicas, lo hacen extendiendo la notación del PIM con elementos propios de la arquitectura objetivo, y por ende, afectando negativamente su portabilidad.

Esta extensión de la notación del PIM deriva en la existencia de PIMs específicos para diferentes arquitecturas, lo que, más que una simple adaptación/extensión de la metodología base, a fines prácticos, parece constituir la definición/utilización de una nueva metodología. Esto debido a que, una vez que los PIM son especializados mediante la extensión de su notación, al margen de las semejanzas que pudieran existir, los modelos generados en base a ellos constituyen modelos diferentes (por causa de los elementos modelados a partir de la notación extendida), y por lo tanto, un cambio de arquitectura implica, necesariamente, la creación de nuevos modelos PIM o, en el

mejor de los casos, la modificación de aquellos modelos PIM que hayan sido definidos previamente.

A fin de comprender mejor esta situación, a continuación se plantea un ejemplo práctico; si utilizando alguna de las metodologías tradicionales (WebML, UWE, etc), se pretende desarrollar aplicaciones bajo diferentes tipos de arquitectura (por ejemplo RIA, SOA y REST), será necesario utilizar tres definiciones diferentes de PIM, una orientada a RIA (WebML RIA [4], UWE para RIA [5]), otra orientada a SOA (WebML SOA, UWE para SOA -si existiesen-), y otra orientada a REST. Lo que, entre otras cosas, implica la necesidad de crear tres modelos PIM diferentes, triplicando trabajo. Algo que ocurre sin importar si las tres aplicaciones son creadas de manera independiente y desde cero, o si, por el contrario, se decide crear primero una, siguiendo alguna de las metodologías, para luego ir modificando los modelos generados con el objetivo de adaptarlos a las demás metodologías.

Siguiendo con los ejemplos, otro caso representativo consiste en imaginar un sistema completamente desarrollado que no haya adoptado RIA como arquitectura al momento de su desarrollo inicial (e.g., un sistema web estático o un servicio web RESTful). Este sistema tendría un PIM completamente modelado debido a que ya habría cumplido un ciclo completo de desarrollo a través de alguna metodología MDD. Si en algún momento surge la necesidad de que este sistema adopte RIA, el PIM existente, que fue creado siguiendo un enfoque MDD no orientado al modelado de arquitecturas RIA, no podría ser re-utilizado por un enfoque que sí esté orientado a hacerlo. La razón principal de esta incompatibilidad es que, el segundo enfoque abordado, contaría con notación exclusiva para modelar RIA, notación que por obvias razones no existiría en el primer enfoque.

También es importante considerar que la definición de PIMs específicos para una arquitectura en particular, viola el principio de portabilidad del PIM pretendido tanto por MDD como por MDA, así como también el principio de separación de conceptos promovido por esta última. Además, la complejidad añadida al modelado de diagramas correspondientes al PIM, por parte de los detalles relacionados con la arquitectura, dificulta el mantenimiento del sistema al aumentar innecesariamente la complejidad los modelos generados.

Se ha llegado a la conclusión de que la portabilidad del PIM se ve afectada, luego del análisis llevado a cabo en este SMS, tras notar que los PIM generados por cada una de las propuestas analizadas, lo modifican tanto, que hacen obsoleto cada PIM creado con la metodología MDD que fue extendida por la propuesta bajo estudio (en el caso de propuestas creadas a partir de la extensión de enfoques pre-existentes). Incluso cuando fueron analizados los modelos PIM generados por propuestas creadas desde cero, estos resultaron ser tan específicos para la arquitectura abordada por la propuesta, que resultaría imposible su re-utilización para el desarrollo de sistemas orientados a otras arquitecturas.

Claramente este problema con la portabilidad del PIM es un problema que merece ser abordado. Y es aquí donde entra en juego MoWebA, un enfoque MDD para desarrollar aplicaciones web que, entre otras cosas, propone la fase ASM considerada en este estudio [12] [9] [14].

La fase ASM de MoWebA es sumamente importante ya que contribuye a resolver el problema de la portabilidad del PIM. El ASM es un nuevo nivel de abstracción,

anexado a los ya definidos por el estándar MDA, que se ubica entre el PIM y el PSM, representando un nuevo nivel de abstracción intermedio entre ambos. La idea de esto es que el ASM concentre toda la información referente a la arquitectura del sistema a ser desarrollado, liberando al PIM de contenerla, y, en teoría, devolviéndole su tan ansiada portabilidad.

Para comprender mejor el aporte del ASM para con la resolución del problema de portabilidad del PIM. Imaginemos nuevamente la situación planteada unos párrafos más arriba, un sistema completamente desarrollado en el que no se adoptó RIA como arquitectura; Si en este punto consideramos adoptar RIA, entonces el PIM seguiría siendo de utilidad ya que toda la información referente a la arquitectura se encontraría encapsulada en el ASM.

3.7. PIM vs ASM

Con el fin de observar las diferencias entre el PIM y el ASM, a continuación se procederá a comparar, mediante el planteamiento de un ejemplo práctico, las dos opciones disponibles a la hora de modificar/extender una metodología MDD preexistente para adaptarla al modelado de sistemas con arquitectura RIA.

Para ilustrar la primera opción, será necesario imaginar las consecuencias de extender el PIM para el modelado de RIA, un enfoque adoptado por la mayoría de las propuestas MDD (e.g., WebML RIA [15], UWE para RIA [5], OOH4RIA [6]).

Mientras que para ilustrar la segunda opción, será necesario imaginar las consecuencias de encapsular en el ASM todos los elementos arquitectónicos específicos de RIA (la solución adoptada por MoWebA [9, 12]).

Con fines didácticos, en el ejemplo se asumirá lo siguiente:

- Inicialmente, el sistema será desarrollado utilizando una metodología MDD para RIA. Una vez finalizado el desarrollo del sistema, se intentará obtener una nueva versión del mismo para una arquitectura diferente, reciclando la mayor cantidad posible de modelos.
- Todas las hipotéticas metodologías MDD a ser utilizadas, considerarán transformaciones M2M.
- Las transformaciones M2M y M2T serán totalmente automáticas.
- Para facilitar la comprensión del ejemplo, no se considerará la fase CIM.

Opción 1, extender la notación del PIM

Si la solución adoptada es la primera, es decir, la que extiende la notación del PIM con el fin de modelar elementos pertenecientes a la arquitectura, entonces los pasos que serían necesarios para desarrollar el sistema con un enfoque RIA son los siguientes:

1. Definir, manualmente, el PIM.
2. Transformar PIM a PSM mediante transformaciones M2M.

3. Transformar PSM a código mediante transformaciones M2T.
4. Implementar el sistema.

Mientras que los pasos para desarrollar exactamente el mismo sistema, en una segunda etapa y para una arquitectura diferente, serían estos:

1. Definir, manualmente, el PIM.
2. Transformar PIM a PSM mediante transformaciones M2M.
3. Transformar PSM a código mediante transformaciones M2T.
4. Implementar el sistema.

Como podrá haber notado, en ambos casos fue necesario definir el PIM. En otras palabras, el PIM no fue reutilizado, algo que sí cabría esperar al momento de aplicar una metodología MDD.

Opción 2, modelar la arquitectura en el ASM

Si la solución adoptada es la segunda opción (aislar elementos de la arquitectura en el ASM), entonces los pasos que serían necesarios para desarrollar el sistema con un enfoque RIA son los siguientes:

1. Definir, manualmente, el PIM.
2. Transformar PIM a ASM mediante transformaciones M2M.
3. Transformar ASM a PSM mediante transformaciones M2M.
4. Transformar PSM a código mediante transformaciones M2T.
5. Implementar el sistema.

Mientras que los pasos que serían necesarios para desarrollar exactamente el mismo sistema, pero en un momento posterior y para una arquitectura diferente, serían estos:

1. Reutilizar el PIM.
2. Transformar PIM a ASM mediante transformaciones M2M.
3. Transformar ASM a PSM mediante transformaciones M2M.
4. Transformar PSM a código mediante transformaciones M2T.
5. Implementar el sistema.

Como podrá haber notado, gracias a la utilización del ASM, luego de haber cambiado la arquitectura del sistema, no fue necesario redefinir el PIM, sino que este fue reutilizado. Esto es lo que queremos decir cuando afirmamos que el ASM devuelve al PIM su portabilidad.

3.8. Síntesis del capítulo

En este capítulo ha sido presentada la revisión bibliográfica llevada a cabo en el marco del presente trabajo de tesis mediante la ejecución de un SMS. Luego, a partir del análisis de los resultados obtenidos, han sido detectadas y señaladas varias oportunidades de investigación y problemáticas que pueden o requieren ser abordadas en el futuro, algunas de las cuales, inclusive, serán abordadas por esta tesis.

Capítulo 4

Reglas y proceso de desarrollo propuestos

A la luz de las conclusiones del SMS llevado a cabo como parte de la revisión bibliográfica del presente trabajo de tesis, fundamentalmente aquella que resalta la existencia de muchas propuestas inconclusas y/o pendientes de validación y la necesidad de seguirlas estudiando; considerando además las oportunidades de investigación detectadas a partir de ellas (detalladas en la sección 3.5), y la problemática identificada (detallada en la sección 3.6); y por sobre todo, a sabiendas de la existencia de MoWebA, un enfoque metodológico cuya adopción permite encarar todos estos aspectos; resulta claro que la propuesta de solución más apropiada sería adoptar MoWebA como metodología base y contribuir con ella.

Confrontando la propuesta de MoWebA (el modelo ASM), con la problemática identificada en la sección 3.6 (el problema de portabilidad del PIM), es posible notar que la utilización de esta etapa intermedia entre el PIM y el PSM permite (o al menos parece hacerlo) devolver a los modelos PIM su tan ansiada portabilidad, ya que, por ejemplo, permite la definición de un solo modelo PIM, tanto para desarrollar las tres variantes del sistema planteado en el primer ejemplo de la sección 3.6, como para modificar la arquitectura de un sistema existente (ver el segundo ejemplo de la sección 3.6, y/o la sección 3.7). Además, al librar al PIM de la definición de aspectos relacionados con la arquitectura, mejora su legibilidad, y por ende, su mantenibilidad.

Sin embargo, MoWebA, uno de los enfoques existentes dentro de lo que sería MDD orientado a arquitecturas RIA, todavía no es capaz de automatizar las transformaciones M2M requeridas para transformar un PIM a un ASM y este a un PSM. Y por lo tanto, las ventajas y/o desventajas que pudiera arrojar la inclusión del ASM al ciclo de desarrollo, sobre todo las ventajas prometidas al plantearse su inclusión, aún no fueron debidamente estudiadas y validadas.

En resumen, la solución propuesta consiste, en primer lugar, en la adopción de MoWebA como metodología base. El objetivo perseguido con esto es contribuir con una metodología MDD existente e inconclusa, en vez de proponer una nueva.

En segundo lugar, y como algo imperativo para la consecución del objetivo final de esta tesis (analizar aspectos relevantes del ASM), la solución propuesta consiste en la definición de reglas de transformación que permitan automatizar la transformación PIM-ASM para la(s) arquitectura(s) seleccionada(s), y además, en la definición de un

proceso de desarrollo que especifique las herramientas a ser utilizadas y los pasos a seguir para la ejecución de estas reglas. En este punto es importante remarcar que la correcta definición de estas reglas de transformación es crucial para evitar sesgos o desviaciones en los análisis posteriores.

De esta manera, y en tercer lugar, una vez estén completamente definidas y validadas las reglas de transformación, se llevarán a cabo distintas experiencias, recabando de ellas la mayor cantidad de datos posible. Datos que posteriormente serán analizados rigurosamente a la luz de el(los) aspecto(s) del ASM seleccionado(s) como objeto(s) de análisis.

Las arquitecturas abordadas en el análisis son RIA y persistencia móvil. Ambas fueron seleccionadas debido a su relevancia actual. Sin embargo, RIA es adoptada como arquitectura principal. Con ella se busca i) identificar el grado máximo de automatización posible, y ii) cubrir el ciclo de desarrollo completo, desde el PIM hasta el código. La persistencia móvil, por otro lado, es abordada únicamente con el objetivo de verificar si es posible o no obtener varios modelos ASM a partir de un único modelo PIM y, posteriormente, analizar las implicancias de esto.

Por otro lado, en base las conclusiones obtenidas en el SMS, sobre todo las relacionadas al problema de portabilidad del PIM, los aspectos del ASM seleccionados como objeto de análisis fueron los siguientes: i) la posibilidad de generar múltiples ASM a partir de un único PIM, y ii) el grado de portabilidad del PIM alcanzado como consecuencia de la utilización del ASM.

4.1. Consideraciones preliminares

Con el fin de corroborar si el ASM realmente contribuye a la portabilidad PIM en la medida de lo esperado, y teniendo en cuenta que MoWebA aún no ha definido reglas de transformación M2M que permitan automatizar la transformación de modelos PIM a modelos ASM. En el marco del presente trabajo de tesis han sido desarrollados dos conjuntos diferentes de reglas de transformación M2M que permiten automatizar este proceso de transformación PIM-ASM siguiendo el enfoque MoWebA. Uno de estos dos conjuntos de reglas se ocupa de las arquitecturas RIA, mientras el otro, de los aspectos arquitectónicos de la persistencia móvil.

El paquete de reglas para RIA constituye uno de los principales aportes de este estudio y se invirtió tiempo considerable en su desarrollo, en todo momento el objetivo fue automatizar al máximo el proceso de transformación. Por otro lado, el paquete de reglas para persistencia móvil tuvo un enfoque diferente. También fue necesario desarrollarlas ya que estas no fueron definidas previamente, sin embargo, al hacerlo, no se buscó maximizar la automatización del proceso de transformación, sino verificar la posibilidad de obtener múltiples ASM a partir de un único PIM.

En la tabla 4.1 se detallan aquellas taxonomías dentro de las que se enmarcan las reglas de transformación M2M desarrolladas. Las taxonomías contempladas son aquellas abordadas en [16].

Para la definición de las reglas de transformación M2M, se optó por el lenguaje de transformación ATL (Atlas Transformation Language) por sobre QVT (Query/View/Transformation). Esta elección se fundamenta, sobre todo, en el hecho de que ATL es considerado uno de los lenguajes de transformación M2M más ampliamente utili-

Taxonomía	PIM-ASM	Motivo
Automatización	Completa	Se incluyen dos archivos de configuración como entrada de la transformación M2M.
Abstracción — propuesto en [79]	Vertical	Los modelos origen y destino residen en diferentes niveles de abstracción.
Lenguaje — propuesto en [79]	Endógeno	Los modelos origen y destino se expresan en el mismo lenguaje de modelado.
Destino — propuesto en [80]	Modelo nuevo	Al culminar el proceso de transformación tenemos dos modelos, el modelo origen, sin cambios, y un modelo destino totalmente nuevo.
Cardinalidad — propuesto en [79]	Uno a uno	Se obtiene un solo modelo destino por cada conjunto de reglas.

Tabla 4.1: Taxonomía de las reglas de transformación M2M

zados en la actualidad, tanto en la academia como en la industria, y además, existe y está disponible de manera gratuita, una madura herramienta[1] de código abierto que lo soporta [3].

Otra elección de suma importancia fue el modo de ejecución del motor de transformaciones M2M de ATL, el cual tiene dos modos de ejecución: el modo de ejecución predeterminado y el modo de ejecución por refinamiento [13].

El modo de ejecución predeterminado es el modo normal de ejecución, tiene la particularidad de que solo crea en el modelo destino, aquellos elementos que son nombrados o afectados por las reglas de transformación definidas, por lo tanto, para copiar un modelo es necesario definir reglas para cada uno de sus elementos.

El modo de ejecución por refinamiento difiere del modo de ejecución predeterminado, al solo requerir la definición de las reglas que generan aquellos elementos del modelo destino en los que se modifica alguna característica del modelo origen, puesto que los demás elementos los copia en lugar de ignorarlos, cosa que sí ocurre en el modo de ejecución por defecto.

Si bien, cuando los metamodelos de los modelos de origen y de destino son diferentes es obligatorio utilizar el modo de ejecución predeterminado, cuando son iguales, se puede optar por cualquiera de los dos modos de ejecución [13].

En este caso, ambos metamodelos, tanto el de los modelos de origen como el de los modelos de destino, son idénticos, más aún, son el mismo metamodelo[2]. La razón de esto es que MoWebA está implementado en base a perfiles [9], y por lo tanto, el PIM y el ASM comparten el mismo metamodelo y difieren solo en los perfiles aplicados. Considerando además que la transformación M2M de PIM a ASM, conceptualmente, se adapta mejor al modo de ejecución por refinamiento que al modo de ejecución predeterminado (puesto que el PIM es justamente refinado/modificado para la obtención del ASM). En este trabajo se optó por utilizar, para la transformación de los modelos

[1] `http://www.eclipse.org/atl/`

[2] `http://www.eclipse.org/modeling/mdt/?project=uml2`

CIM/PIM a modelos ASM/PSM, el modo de ejecución por refinamiento.

Si bien esta elección redujo considerablemente la cantidad de reglas definidas, también derivo en complicaciones. La primera y principal es que la aplicación de perfiles en ATL se realiza en el bloque imperativo y cuando se utiliza el modo de refinamiento, el compilador estándar, el motor de transformaciones incluido por defecto, no permite la utilización de este bloque.

El compilador por defecto en Eclipse ATL[1] es la Máquina Virtual Específica de EMF, o EMF-specific Virtual Machine (EMF-specific VM). Sin embargo, ATL dispone de otros compiladores. En este trabajo se optó por la Máquina Virtual de Transformación de EMF, o EMF Transformation Virtual Machine (EMFTVM).

Si bien la razón principal de esta elección fue que EMFTVM permite el uso del bloque imperativo en modo de refinamiento, posibilitando la aplicación de perfiles bajo este modo, este compilador cuenta con otras características interesantes. Por ejemplo, su rendimiento es aproximadamente 80 % mejor que el de la EMF-specific VM [84], y además, permite la invocación de métodos Java nativos [85].

4.2. Proceso de desarrollo propuesto

Un esquema general del proceso de desarrollo propuesto se presenta en la Figura 4.1. Al analizarla, es sencillo notar que en el proceso intervienen tres IDEs diferentes.

El primero es MagicDraw[3], en este IDE se importan los Perfiles UML de MoWebA y se generan los modelos CIM/PIM que luego serán transformados a modelos ASM/PSM mediante las reglas de transformación M2M definidas para el efecto. Una vez generados los modelos CIM/PIM, estos son exportados como documentos XMI (XML Metadata Interchange). En esta primer etapa se opta por MagiDraw porque los Perfiles UML para MoWebA fueron desarrollados con este IDE, y por lo tanto, la integración de los perfiles al IDE es inmediata, funcional y robusta, y además, contamos con bastante experiencia haciéndolo, lo que reduce considerablemente la posibilidad de errores e imprevistas es esta etapa inicial.

El segundo IDE utilizado es Eclipse, concretamente el paquete Eclipse Modeling Tools, versión Neon Release (4.6.0)[4], desarrollado y mantenido por el Eclipse Modeling Project y el Eclipse Modeling Framework (EMF).

Este IDE pasó por un proceso de configuración bastante complejo. El primer ajuste consistió en configurarlo para soportar el lenguaje de transformación M2M denominado ATL (Atlas Transformation Language). Luego fue necesario modificar el motor de transformación M2M debido a que el motor por defecto no se ajusta a los requerimientos del presente estudio, concretamente, no permite la utilización del bloque de las sentencias imperativas en modo refinamiento. Otra importante modificación consistió en configurar al IDE, y al nuevo motor de transformación, para que puedan ejecutar código java nativo, código desarrollado en el marco del presente trabajo y hecho a medida para procesar archivos de configuración externos. Archivos cuya finalidad es la personalización del proceso de transformación sin necesidad de modificar las reglas de transformación M2M. Una vez finalizada la configuración del IDE, se procedió a

[3] `https://www.nomagic.com/products/magicdraw`

[4] `http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/neon3`

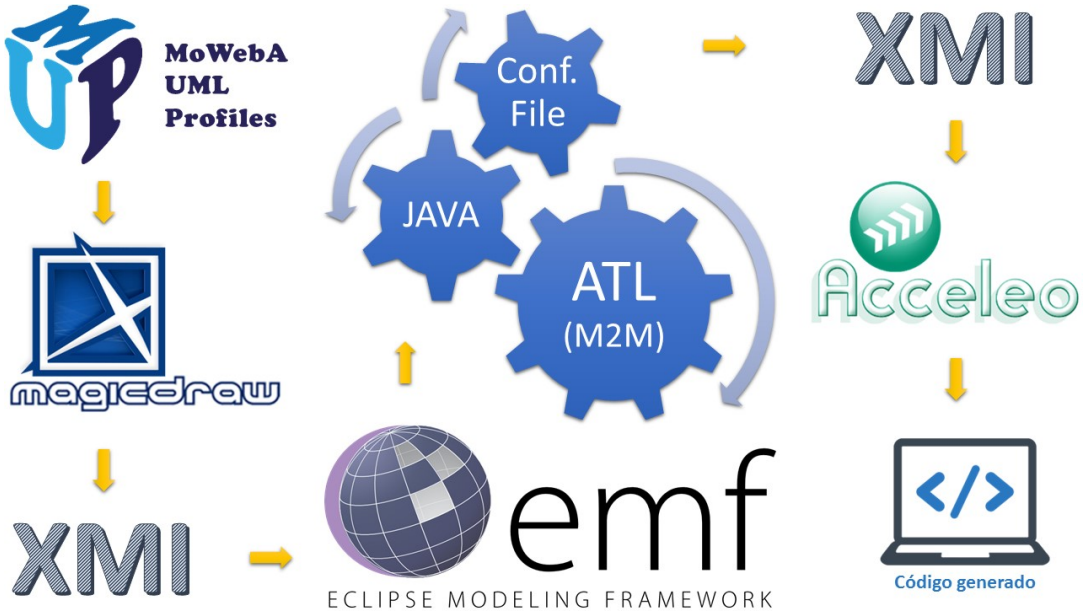


Figura 4.1: Esquema general de la solución propuesta

importar los modelos CIM/PIM creados con MagicDraw y exportados como documentos XMI, para finalmente, en base a las reglas de transformación M2M definidas y lo estipulado en los archivos de configuración, transformarlos a modelos ASM/PSM expresados en formato XMI.

El tercer IDE también es Eclipse y la versión base es similar al IDE anterior, es decir, la versión base también consiste en el paquete Eclipse Modeling Tools, versión Neon Release (4.6.0)[5], desarrollado y mantenido por el Eclipse Modeling Project y el Eclipse Modeling Framework (EMF). Sin embargo, la configuración del IDE es bastante diferente. Este IDE se ocupa de la transformación M2T y para ello hace uso de la herramienta Acceleo. El proceso de transformación consiste en importar los modelos ASM/PSM en formato XMI que fueron generados en la etapa anterior y transformarlos a código mediante la aplicación de reglas de transformación M2T definidas para el efecto, en trabajos afines desarrollados en el marco del proyecto de investigación que posibilitó la realización del presente trabajo de tesis, MDD+[6].

4.3. Metamodelo y perfiles

Como ya fue mencionado previamente, el metamodelo[2] utilizado para la definición de las reglas de transformación M2M es el mismo tanto para RIA como para persistencia móvil, y es el definido para UML2 por el proyecto MDT (Eclipse Model Development Tools) [86].

Sin embargo, en el marco del presente trabajo de tesis, también ha sido considerado el metamodelo de MoWebA (ver apéndice A), el cual fue utilizado para la definición

[5] http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/neon3
[6] http://www.dei.uc.edu.py/proyectos/mddplus/

de las sintaxis abstractas de las arquitecturas seleccionadas y ayudó a especificar los Perfiles UML correspondientes a las sintaxis concretas.

Este metamodelo específico para MoWebA ha sido publicado en la web del proyecto MDD+, concretamente en la sección de herramientas (`http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/`), bajo la opción "Metamodelo MoWebA". Aunque también puede ser descargado utilizando el siguiente link de descarga: `https://goo.gl/vWtYPY`.

Para evitar confusiones, es importante aclarar que el metamodelo específico para MoWebA que ha sido publicado en la web del proyecto de investigación que engloba al presente estudio y que, además, es detallado en el apéndice A, aún no contempla elemento alguno que permita el modelado de arquitecturas RIA o de elementos arquitectónicos propios de la persistencia móvil.

Los perfiles para RIA y persistencia móvil, ambos, fueron creados en trabajos previos enmarcados bajo el mismo proyecto de investigación que el presente trabajo de tesis. Estos trabajos previos son [22] y [23] respectivamente. Además, en ambos casos, los perfiles fueron creados a partir de la extensión/adaptación de la versión más actualizada existente del perfil de MoWebA (ver apéndice B), la cual puede ser descargada aquí: `https://goo.gl/j0vu9b`. Aunque también es posible acceder a ella a través de la web del proyecto MDD+, en la sección de herramientas (`http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/`), bajo la opción "Perfiles MoWebA" (es importante tener presente que esta versión del perfil de MoWebA es la que fue extendida/adaptada posteriormente, y por lo tanto, aún no contempla el modelado de elementos arquitectónicos propios de RIA y/o persistencia móvil).

En la definición de las reglas de transformación M2M se han utilizado partes de un perfil unificado de MoWebA, el cual fue desarrollado en el marco de este estudio. La unificación realizada consistió en agregar a los Perfiles de MoWebA previamente existentes (ver apéndice B), aquellos perfiles que fueron definidos para RIA en [22] (ver figura 4.2) y para persistencia móvil en [23] (ver figura 4.3). Así, al definir las reglas de transformación M2M para RIA se utilizó el paquete "RIA Profile", el cual contiene los elementos específicos para el modelado de arquitecturas RIA, mientras que al definir las reglas de transformación M2M para persistencia móvil se utilizó el paquete "Mobile Profile", el cual contiene los elementos específicos para el modelado de esta arquitectura.

Como los metamodelos y perfiles para RIA [22] y persistencia móvil [23] fueron desarrollados en trabajos diferentes, también existen, y han sido publicadas, definiciones de metamodelos que contemplan "MoWebA y RIA[7]" y "MoWeba y persistencia móvil[8]" por separado.

A la hora de analizar los metamodelos y perfiles definidos para el modelado de arquitecturas RIA es importante considerar que estos cubren solo las siguientes características: i) el almacenamiento de datos en el cliente, ii) la lógica de negocios en el cliente, y iii) la comunicación asíncrona entre cliente y servidor [22].

Así también, a la hora de analizar los metamodelos y perfiles definidos para el modelado de elementos arquitectónicos propios de la persistencia móvil es importante considerar que estos se enfocan, única y exclusivamente, en la persistencia [23].

[7] `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/mowebaria/`
[8] `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/mowebamobile/`

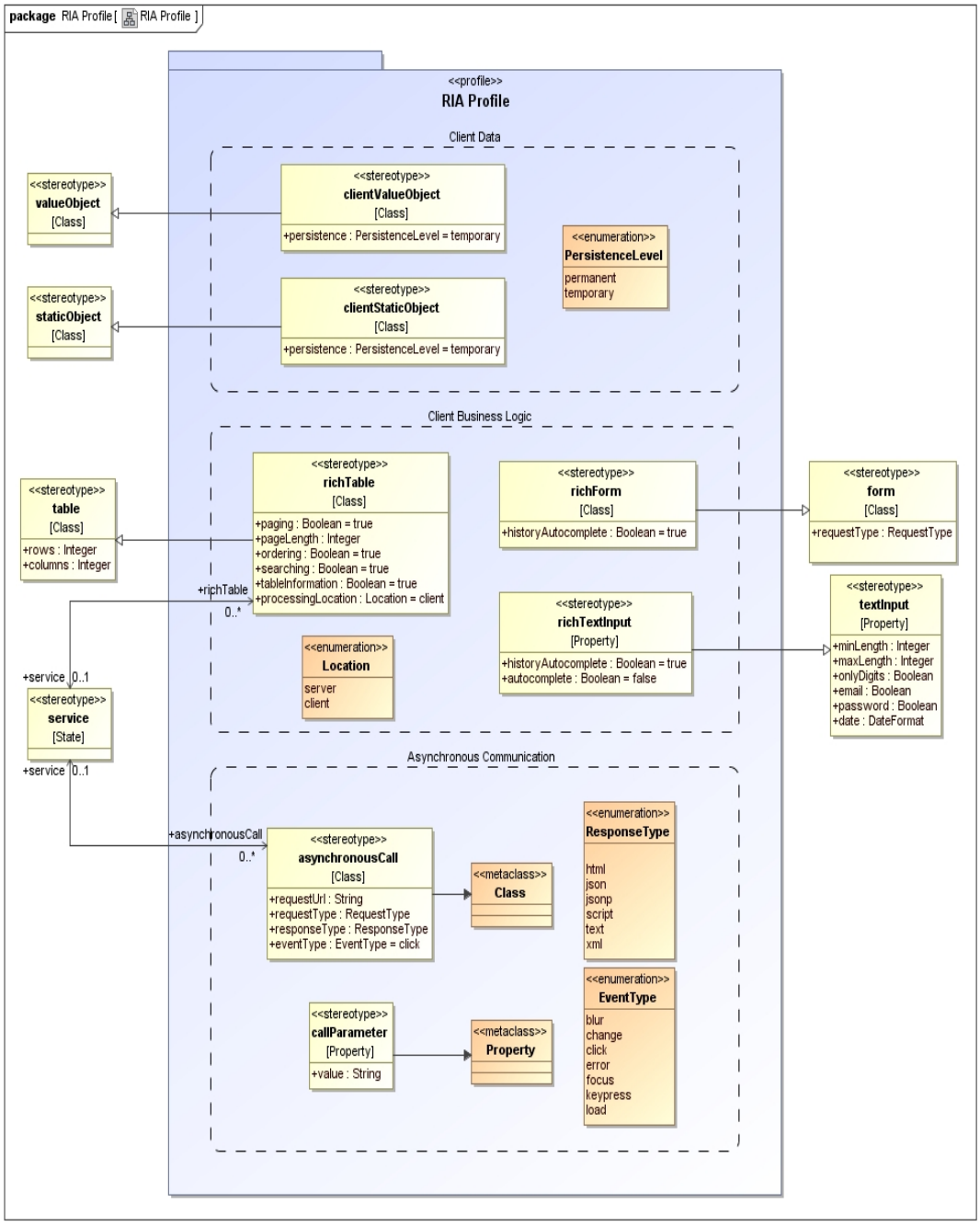


Figura 4.2: Perfil RIA para MoWebA

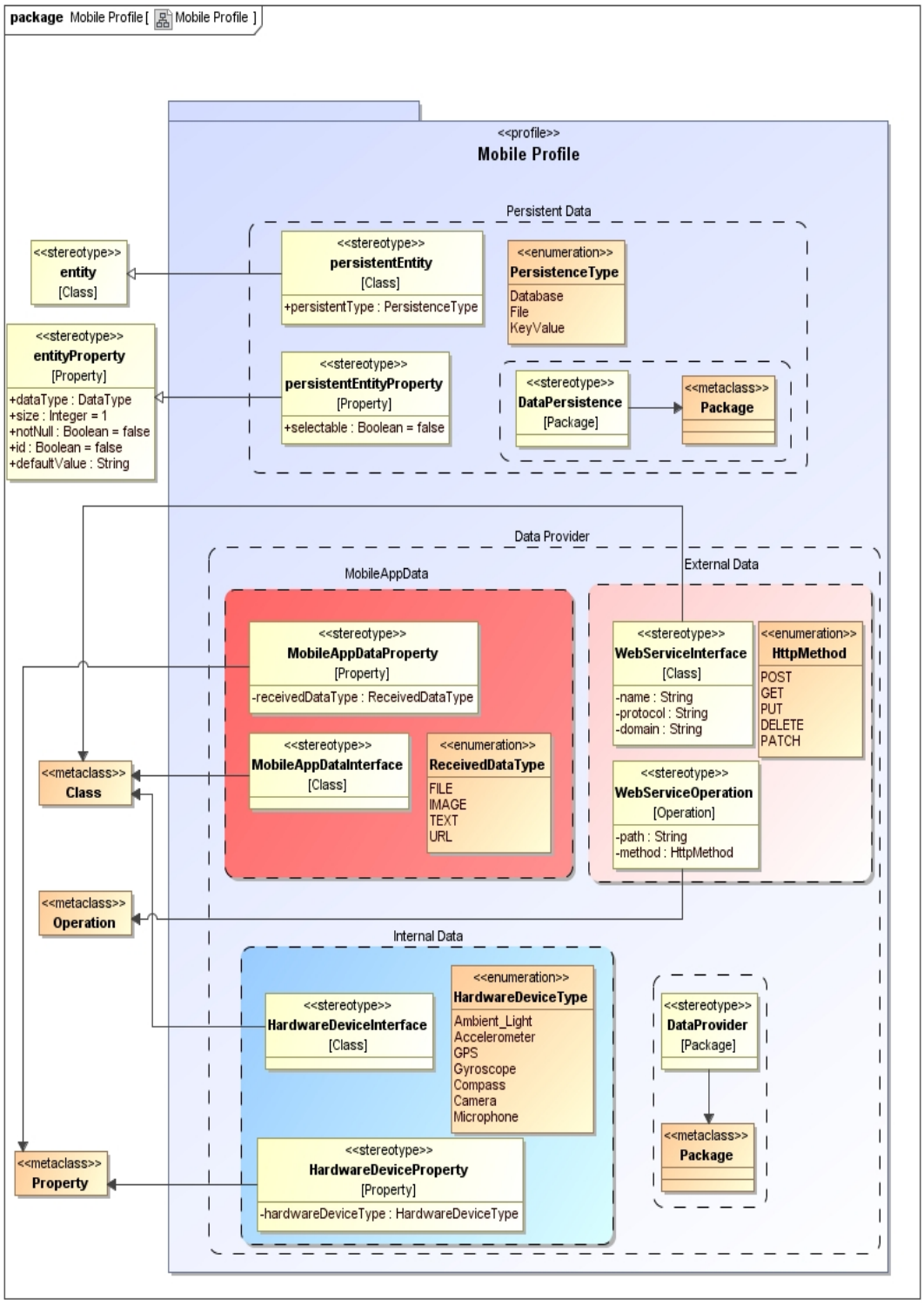


Figura 4.3: Perfil de persistencia móvil para MoWebA

4.4. El mapeo PIM-ASM

Antes de la definición de las reglas de transformación M2M ha sido realizado un mapeo entre los elementos presentes en cada uno de los perfiles seleccionados, de manera separada e independiente, es decir, un mapeo para cada arquitectura.

El propósito del mapeo realizado fue la identificación de aquellos elementos en los perfiles que posteriormente dan origen a los elementos modelados tanto en los modelos PIM como en los modelos ASM.

Una vez identificados estos elementos, el siguiente paso consistió en identificar, a partir de la relación entre ellos, i) cuales de los elementos en el modelo origen (en este caso el modelo PIM) deben ser transformados, y sobre todo, una vez descubierta la necesidad de transformación, ii) en cuales de los elementos del modelo destino (en este caso el modelo ASM) deben ser transformados estos elementos del modelo origen.

El mapeo fue producto de un análisis visual realizado sobre el perfil definido para cada una de las arquitecturas de referencia, es decir, el perfil para RIA[9]" desarrollado por Guido Nuñez et al. en [22] (ver Figura 4.2), y el perfil para persistencia móvil[10]" desarrollado por Manuel Nuñez en [23] (ver Figura 4.3).

4.4.1. Mapeo #1 (relaciones de tipo herencia)

A la hora de realizar el mapeo se observó que, en general, cuando la relación entre dos clases en cualquiera de los perfiles contemplados (tanto el perfil correspondiente a RIA como el perfil correspondiente a persistencia móvil) es una herencia, entonces se da que, cuando esa clase o estereotipo representado en el perfil aparece en el modelo origen (en este caso el modelo PIM), la transformación M2M que permite obtener el elemento correcto en el modelo destino (en este caso el modelo ASM), es muy simple y consiste en la aplicación del estereotipo correcto.

RIA

Al momento de analizar la aparición de relaciones de tipo herencia en el perfil correspondiente a las arquitecturas RIA, detallado en la figura 4.2, han sido detectados los mapeos desarrollados a continuación.

Objeto valuado En la figura 4.4 se puede observar una relación de tipo herencia detectada en el perfil RIA, la cual implica lo siguiente: cuando una clase de tipo ValueObject (o una clase con el estereotipo ValueObject aplicado) aparece el modelo origen (en este caso el PIM), debe ser transformada en una clase de tipo ClientValueObject en el modelo destino (en este caso el ASM), mediante la aplicación del estereotipo ClientValueObject.

[9] `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/mowebaria/`
[10] `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/mowebamobile/`

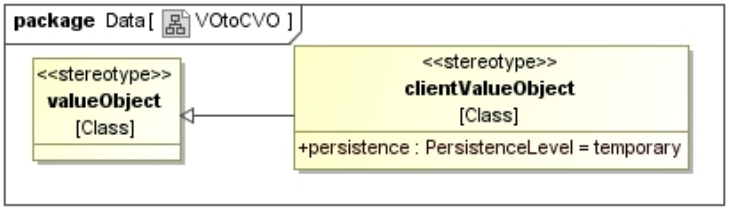


Figura 4.4: Mapeo–RIA–Herencia – ValueObject–ClientValueObject

Objeto estático En la figura 4.5 se puede observar una relación de tipo herencia detectada en el perfil RIA, la cual implica lo siguiente: cuando una clase de tipo StatiObject (o una clase con el estereotipo StatiObject aplicado) aparece en el modelo origen (en este caso el PIM), debe ser transformada en una clase de tipo ClientStatiObject en el modelo destino (en este caso el ASM), mediante la aplicación del estereotipo ClientStatiObject.

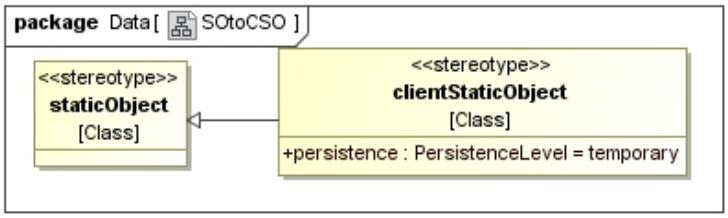


Figura 4.5: Mapeo–RIA–Herencia – StaticObject–ClientStaticObject

Tablas En la figura 4.6 se puede observar una relación de tipo herencia detectada en el perfil RIA, la cual implica lo siguiente: cuando una clase de tipo Table (o una clase con el estereotipo Table aplicado) aparece en el modelo origen (en este caso el PIM), debe ser transformada en una clase de tipo RichTable en el modelo destino (en este caso el ASM), mediante la aplicación del estereotipo RichTable.

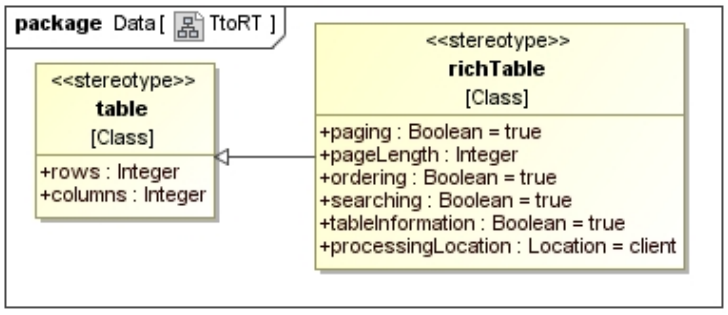


Figura 4.6: Mapeo–RIA–Herencia – Table–RichTable

Formularios En la figura 4.7 se puede observar una relación de tipo herencia detectada en el perfil RIA, la cual implica lo siguiente: cuando una clase de tipo Form (o una clase con el estereotipo Form aplicado) aparece en el modelo origen (en este caso el PIM), debe ser transformada en una clase de tipo RichForm en el modelo destino (en este caso el ASM), mediante la aplicación del estereotipo RichForm..

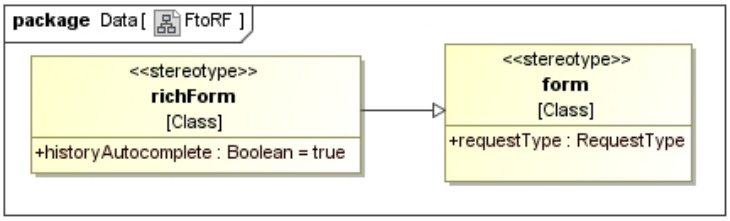


Figura 4.7: Mapeo–RIA–Herencia – Form–RichForm

Introducción de texto En la figura 4.8 se puede observar una relación de tipo herencia detectada en el perfil RIA, la cual implica lo siguiente: cuando una clase de tipo TextInput (o una clase con el estereotipo TextInput aplicado) aparece en el modelo origen (en este caso el PIM), debe ser transformada en una clase de tipo RichTextInput en el modelo destino (en este caso el ASM), mediante la aplicación del estereotipo RichTextInput.

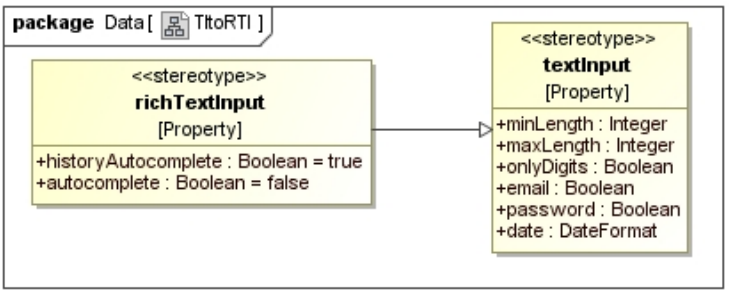


Figura 4.8: Mapeo–RIA–Herencia – TextInput–RichTextInput

Persistencia móvil

Por otro lado, al momento de analizar la aparición de relaciones de tipo herencia en el perfil correspondiente a persistencia móvil, el cual es detallado en la figura 4.3, el mapeo es bastante similar. Los mapeos que han sido detectados son los siguientes.

Entidades En la figura 4.9 se puede observar una relación de tipo herencia detectada en el perfil correspondiente a persistencia móvil, la cual implica lo siguiente: cuando una clase de tipo Entity (o una clase con el estereotipo Entity aplicado) aparece en el modelo origen (en este caso el PIM), debe ser transformada en una clase de tipo

PersistentEntity en el modelo destino (en este caso el ASM), mediante la aplicación del estereotipo PersistentEntity.

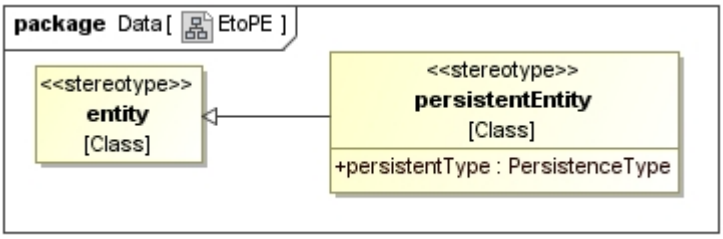


Figura 4.9: Mapeo–P. Móvil–Herencia – Entity–PersistentEntity

Propiedades de las entidades En la figura 4.10 se puede observar una relación de tipo herencia detectada en el perfil correspondiente a persistencia móvil, la cual implica lo siguiente: cuando una clase de tipo EntityProperty (o una clase con el estereotipo EntityProperty aplicado) aparece en el modelo origen (en este caso el PIM), debe ser transformada en una clase de tipo PersistentEntityProperty en el modelo destino (en este caso el ASM), mediante la aplicación del estereotipo PersistentEntityProperty.

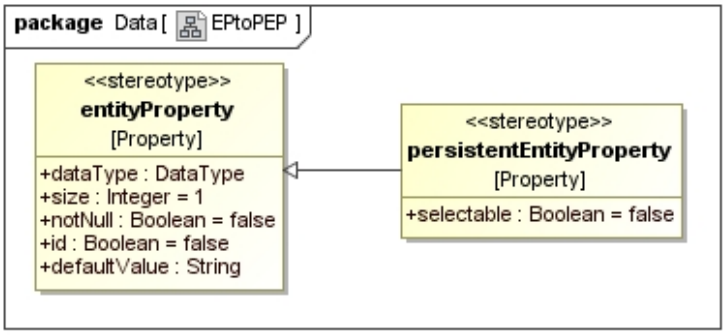


Figura 4.10: Mapeo–P. Móvil–Herencia – EntityProperty–PersistentEntityProperty

4.4.2. Mapeo #2 (otro tipo de relaciones)

Cuando la relación entre dos clases del perfil no es de tipo herencia, el mapeo se hace un poco más difícil. En el caso del perfil para RIA encontramos dos relaciones de este tipo, una entre las clases ServiceState y AsynchronousCall (ver Figura 4.11), y la otra entre las clases ServiceSate y RichTable. Luego de analizar profundamente estas relaciones pudimos concluir que la clase AsynchronousCall debe ser creada (en el ASM) cuando está relacionada (en el PIM) con una clase ServiceSate que, además, representa a un servicio asíncrono, condición que puede ser detectada de manera automática, permitiendo la automatización de la transformación.

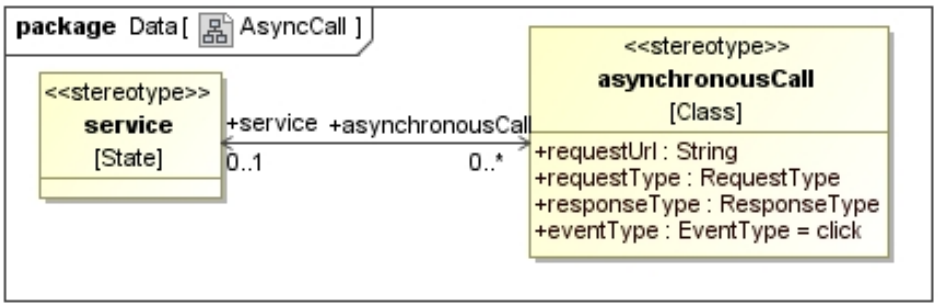


Figura 4.11: Mapeo–RIA–Servicio asíncrono

4.5. Reglas de transformación

En esta sección se presentará la totalidad de las reglas de transformación M2M definidas tanto para RIA como para persistencia móvil. Además, se presentará el código Java que permite el procesamiento de los archivos de configuración.

4.5.1. Encabezado

La primera sección de código en ser presentada será el Encabezado. Esta es una sección muy importante de las reglas de transformación ya que detalla el compilador, los meta-modelos, el modo de ejecución y los perfiles utilizados.

Encabezado para RIA

```
-- @atlcompiler emftvm
-- @nsURI UML2=http://www.eclipse.org/uml2/2.0.0/UML

module ReglasM2M;

create OUT : UML2 refining IN : UML2, RIA_PROFILE : UML2, CONTENT_PROFILE : UML2;
```

Encabezado para persistencia móvil

```
-- @atlcompiler emftvm
-- @nsURI UML2=http://www.eclipse.org/uml2/2.0.0/UML

module ReglasM2M;

create OUT : UML2 refining IN : UML2, MOBILE_PROFILE : UML2, CONTENT_PROFILE : UML2;
```

4.5.2. Variables globales (Helper Rules de tipo atributo)

La segunda sección de código en ser presentada será la correspondiente a la declaración de las variables globales.

Variables globales para RIA

```
---
helper def: Global_RIAProfile : UML2!Profile =
    UML2!Profile.allInstancesFrom('RIA_PROFILE')
        ->select(p|p.name='RIA Profile')->first();
```

Variables globales para persistencia móvil

```
---
helper def: Global_MobileProfile : UML2!Profile =
    UML2!Profile.allInstancesFrom('MOBILE_PROFILE')
        ->select(p|p.name='Mobile Profile')->first();

---
helper def: Selectable_Control : Sequence(String) = Sequence{''};
```

4.5.3. Helper Rules (de tipo funcional)

Los *Helper Rules* pueden ser considerados como los equivalentes en ATL de los métodos de Java, puesto que se utilizan para factorizar código que luego puede ser invocado desde diferentes puntos de la transformación [13]. La principal característica de los *helpers* es que no pueden crear elementos en el modelo destino.

Helper Rules para RIA

```
--- Evaluates whether the context has the given stereotype or not.
helper context UML2!Element def: hasStereotype(stereotype : String) : Boolean =
    self.getAppliedStereotypes() -> collect( st | st.name ) -> includes(stereotype);

---
helper context UML2!Element def : getStereotype(name : String) : UML2!Stereotype =
    self.getAppliedStereotypes() -> any( e | e.name = name );

---
helper def : getStereotype(name : String) : UML2!Stereotype =
    UML2!Stereotype.allInstances() -> any( e | e.name = name );

---
helper def : getState(name : String) : UML2!State =
    UML2!State.allInstances() -> any( e | e.name = name );

---
helper def : getProperty(name : String) : UML2!Property =
    UML2!Property.allInstances() -> any( e | e.name = name );

---
helper def : getClass(name : String) : UML2!Class =
```

```
    UML2!Class.allInstances() -> any( e | e.name = name );

---
helper context UML2!Element
def: hasValTagValue(stereotype:String, tag:String) : Boolean =
    if (self.hasStereotype(stereotype))
    then
        if (self.hasValue(self.getStereotype(stereotype),tag))
        then true
        else false
        endif
    else false
    endif;

---
helper context UML2!Element
def: getTagValue(stereotype:String, tag:String) : UML2!EObject =
    if(self.hasStereotype(stereotype))
    then self.getValue(self.getStereotype(stereotype),tag)
    else OclUndefined
    endif;

---
helper context UML2!Element
def: setTagValue(stereotype:String, tag:String, value:UML2!EObject) : UML2!Element =
    if(self.hasStereotype(stereotype))
    then self.setValue(self.getStereotype(stereotype),tag,value)
    else OclUndefined
    endif;

---
helper context UML2!State def: isAsynchronousService() : Boolean =
    if ( self.hasStereotype('service')
        and UML2!Transition.allInstances()
                -> exists( e | e.hasStereotype('controlFlow')
                    and e.target.hasStereotype('virtualState')
                    and e.source.name = self.name)
        and UML2!Transition.allInstances()
                -> exists( e | e.hasStereotype('hiperLink')
                    and e.source.hasStereotype('virtualState')
                    and e.target.name = self.name) )
    then
        let r1 : UML2!Transition = UML2!Transition.allInstances()
                -> any( e | e.hasStereotype('controlFlow')
                    and e.target.hasStereotype('virtualState')
                    and e.source.name = self.name) in
        let r2 : UML2!Transition = UML2!Transition.allInstances()
                -> any( e | e.hasStereotype('hiperLink')
                    and e.source.hasStereotype('virtualState')
                    and e.target.name = self.name) in
        if ( r1.target.name = r2.source.name )
        then true
        else false
        endif
    else false
    endif;
```

```
--- Retorna las propiedades candidatas a referenciar servicios asincronos.
helper context UML2!Package def: candidates : Sequence(UML2!Property) =
    UML2!Property.allInstances() ->
        select( e | e.getNearestPackage().name = self.name ) ->
        select( e | e.hasStereotype('anchor') or
                    e.hasStereotype('button') or
                    e.hasStereotype('submitButton') );

--- Obtiene el estereotipo de una propiedad candidata a
--- referenciar servicios asincronos.
helper context UML2!Property def: st() : String =
    let default_response : String = '' in
    if (self.hasStereotype('anchor'))       then 'anchor' else
    if (self.hasStereotype('button'))       then 'button' else
    if (self.hasStereotype('submitButton')) then 'submitButton' else
    default_response endif endif endif;

--- Filtra las propiedades candidatas
--- y solo devuelve las propiedades con servicio asincrono.
helper context UML2!Package def: propertiesWithAS : Sequence(UML2!Property) =
    self.candidates ->
        select( p | p.hasValTagValue(p.st(),'service')) ->
        select( p | p.getTagValue(p.st(),'service').isAsynchronousService() ) ;

---
helper context UML2!Package def: hasAsynchronousService() : Boolean =
    if ( self.propertiesWithAS -> notEmpty() ) then true else false endif;

---
helper context Integer def : makeSequence() : Sequence(Integer) =
    if self <= 1 then
        Sequence{1}
    else
        (self - 1).makeSequence()->append(self)
    endif;

---
helper context UML2!Package def: getAsynchronousServicesNames() : Sequence(String) =
    self.propertiesWithAS
        -> collect(p | p.getTagValue(p.st(),'service').name)
        -> flatten() ;

---
helper context Sequence(String) def: toClassNames() : Sequence(String) =
    self -> collect(s|s.replace(' ','').replaceAll('^.',s.substring(1,1).toLower()))
         -> flatten();

---
helper context UML2!Package def: getNamesOfPropertiesWithAS() : Sequence(String) =
    self.propertiesWithAS -> collect( p | p.name ) -> flatten();

---
helper context UML2!Package def: getStNamesOfPropertiesWithAS() : Sequence(String) =
    self.propertiesWithAS
        -> collect( p | p.getStereotype(p.st()).name ) -> flatten();

---
```

```
helper context UML2!Package def: getClassesOfPropertiesWithAS() : Sequence(String) =
    self.propertiesWithAS -> collect( p | p.class.name ) -> flatten();

---
helper context Integer def : makeMap(val: Sequence(String)) : Map(Integer,String) =
    if self <= 1 then
        Map{(1,val.at(1))}
    else
        (self - 1).makeMap(val)->including(self,val.at(self))
    endif;
```

Helper Rules para persistencia móvil

```
---
helper def : getStereotype(name : String) : UML2!Stereotype =
    UML2!Stereotype.allInstances() -> any( e | e.name = name );
```

4.5.4. Detección de servicios asíncronos

Una sección que merece destaque es la *Helper Rule* que detecta cuando un servicio es asíncrono, permitiendo de esta manera la creación automática de la clase AsynchronousCall. Es importante señalar que esta regla solo aplica para el paquete de transformaciones RIA.

```
helper context UML2!State def: isAsynchronousService() : Boolean =
    if ( self.hasStereotype('service')
        and UML2!Transition.allInstances()
                -> exists( e | e.hasStereotype('controlFlow')
                    and e.target.hasStereotype('virtualState')
                    and e.source.name = self.name)
        and UML2!Transition.allInstances()
                -> exists( e | e.hasStereotype('hiperLink')
                    and e.source.hasStereotype('virtualState')
                    and e.target.name = self.name) )
    then
        let r1 : UML2!Transition = UML2!Transition.allInstances()
                -> any( e | e.hasStereotype('controlFlow')
                    and e.target.hasStereotype('virtualState')
                    and e.source.name = self.name) in
        let r2 : UML2!Transition = UML2!Transition.allInstances()
                -> any( e | e.hasStereotype('hiperLink')
                    and e.source.hasStereotype('virtualState')
                    and e.target.name = self.name) in
        if ( r1.target.name = r2.source.name )
        then true
        else false
        endif
    else false
    endif;
```

4.5.5. Lazy Rules

Este tipo de reglas deben ser explícitamente invocadas desde otra regla y, a diferencia de los *helpers*, sí pueden crear elementos en el modelo destino [13]. No fue necesario utilizar *lazy rule* en el paquete de reglas de transformación para persistencia móvil.

```
---
lazy rule CreateClass {
    from datos : TupleType(name : String, propertiesNames : Sequence(String))
    to t : UML2!"uml::Class" (
        name <- datos.name,
        ownedAttribute <- datos.propertiesNames
            -> collect(pn | thisModule.CreateProperty(pn)))
}

---
lazy rule CreateProperty {
    from propertyName : String
    to t : UML2!"uml::Property" (
        name <- propertyName)
}

---
lazy rule CreateDependency {
    from datos : TupleType(name : String, client : UML2!Class, supplier : String)
    to t : UML2!"uml::Dependency" (
        name <- datos.name,
        client <- Sequence{datos.client},
        supplier <- Sequence{thisModule.getProperty(datos.supplier)}
    )
}
```

4.5.6. Called Rules

Las *Called Rules* coinciden en propósito y comportamiento con las *Lazy Rules*, aunque difieren ligeramente en su sintaxis y no tienen ningún campo obligatorio [13].

Called Rules para RIA

```
---
rule applyRIAStereoTypes(required : Boolean, s : UML2!Element, t : UML2!Element)
{
    using {

        new_stereotype : UML2!Stereotype = '';
        change_stereotype : Boolean = false;

        container : String = s.namespace.name.toString();
        element : String = s. name.toString();
        package : String = s.getNearestPackage().name;

    }
```

```
do
{

    for (stereotype in s.getAppliedStereotypes())
    {

        if (stereotype.getName() = 'valueObject')
        {
            new_stereotype <- thisModule.getStereotype('clientValueObject');
        }
        else if (stereotype.getName() = 'staticObject')
        {
            new_stereotype <- thisModule.getStereotype('clientStaticObject');
        }
        else if (stereotype.getName() = 'table')
        {
            new_stereotype <- thisModule.getStereotype('richTable');
        }
        else if (stereotype.getName() = 'form')
        {
            new_stereotype <- thisModule.getStereotype('richForm');
        }
        else if (stereotype.getName() = 'textInput')
        {
            new_stereotype <- thisModule.getStereotype('richTextInput');
        }

        --- Se decide si se cambia o no el estereotipo
        if (container = package) {
            --- si container es igual a package,
            --- entonces el elemento es una clase
            change_stereotype <- let object : "#native"!"atl::conf::ConfM2M" =
                "#native"!"atl::conf::ConfM2M".newInstance()
                    in object.aplicarEstereotipo(container,element);
        } else {
            --- si container NO es igual a package,
            --- entonces el elemento es una propiedad
            change_stereotype <- let object : "#native"!"atl::conf::ConfM2M" =
                "#native"!"atl::conf::ConfM2M".newInstance()
                    in object.aplicarEstereotipo(package,container,element);
        }

        if ( new_stereotype <> '' and change_stereotype)
        {
            --if the UML!Element does not get its stereotype automically applied
            --(i.e. UML not required) then we must apply it manually
            --note:surprisingly the "required" property doesn't seem to be
            --exposed by UML2.
            if (not required)
            {
                t.unapplyStereotype(stereotype);
                t.applyStereotype(new_stereotype);
            }

            for (property in stereotype.getAllAttributes())
            {
                --apply the value if there is one. don't apply the base type
```

```
                --property as ATL cannot handle this. also don't touch
                --read-only properties.
                if(s.hasValue(stereotype,property.name)
                    and not property.name.startsWith('base_')
                    and not property.isReadOnly())
                {
                    t.setValue(stereotype,
                               property.name,
                               s.getValue(stereotype,property.name));
                }
            }
        }

    }
}
}
```

Called Rules para persistencia móvil

```
---
rule applyMobileStereoTypes(required : Boolean, s : UML2!Element, t : UML2!Element)
{
    using {

        new_stereotype : UML2!Stereotype = '';
        change_stereotype : Boolean = false;

        container : String = s.namespace.name.toString();
        element : String = s.name.toString();
        package : String = s.getNearestPackage().name;

    }
    do
    {

        for (stereotype in s.getAppliedStereotypes())
        {

            if (stereotype.getName() = 'entity')
            {
              new_stereotype <- thisModule.getStereotype('persistentEntity');
            }
            else if (stereotype.getName() = 'entityProperty')
            {
              new_stereotype <- thisModule.getStereotype('persistentEntityProperty');
            }

            --- Se decide si se cambia o no el estereotipo
            if (container = package) {
               --- si container es igual a package, entonces el elemento es una clase
               change_stereotype <- let object : "#native"!"atl::conf::ConfM2M" =
```

```
                "#native"!"atl::conf::ConfM2M".newInstance()
                    in object.aplicarEstereotipo(container,element);
            } else {
               --- si container NO es igual a package, entonces el elemento es
               --- una propiedad
               change_stereotype <- let object : "#native"!"atl::conf::ConfM2M" =
                   "#native"!"atl::conf::ConfM2M".newInstance()
                       in object.aplicarEstereotipo(package,container,element);
            }
            --change_stereotype -> debug('Se cambia el estereotipo?');
            ---

            if ( new_stereotype <> '' and change_stereotype)
            {
                --if the UML!Element does not get its stereotype automically applied
                --(i.e. UML not required) then we must apply it manually
                --note:surprisingly the "required" property doesn't seem to be
                --exposed by UML2.
                if (not required)
                {
                    t.unapplyStereotype(stereotype);
                    t.applyStereotype(new_stereotype);
                }

                for (property in stereotype.getAllAttributes())
                {
                    --apply the value if there is one. don't apply the base type
                    --property as ATL cannot handle this. also don't touch
                    --read-only properties.
                    if(s.hasValue(stereotype,property.name)
                        and not property.name.startsWith('base_')
                        and notproperty.isReadOnly())
                    {
                        t.setValue(stereotype,
                                   property.name,
                                   s.getValue(stereotype,property.name));
                    }
                }

                --marca la propiedad 'selectable' como 'true' para la 1er propiedad
                --de la clase
                if (new_stereotype.getName() = 'persistentEntityProperty')
                {
                    if (not thisModule.Selectable_Control.includes(t.class.name)) {
                        t.setValue(new_stereotype,'selectable',true);
                        thisModule.Selectable_Control
                            <- thisModule.Selectable_Control.including(t.class.name);
                    }
                }

            }

        }
    }
}
```

4.5.7. Matched Rules

Este tipo de regla constituye el núcleo de la transformación declarativa de ATL, puesto que permite especificar i) qué elementos del modelo (o modelos) destino deben ser generados a partir de qué elementos del modelo (o modelos) origen, y ii) la manera en la que se deben inicializar las propiedades de dichos elementos [13].

Matched Rules para RIA

```
--- Se debe ejecutar primero!! Sino, no puedo aplicar los estereotipos de RIA_PROFILE
rule Model {
    from s : UML2!"uml::Model" in IN
    to t : UML2!"uml::Model" (
        __xmiID__ <- s.__xmiID__,
        name <- s.name,
        visibility <- s.visibility,
        viewpoint <- s.viewpoint,
        eAnnotations <- s.eAnnotations,
        ownedComment <- s.ownedComment,
        nameExpression <- s.nameExpression,
        elementImport <- s.elementImport,
        packageImport <- s.packageImport,
        ownedRule <- s.ownedRule,
        templateParameter <- s.templateParameter,
        templateBinding <- s.templateBinding,
        ownedTemplateSignature <- s.ownedTemplateSignature,
        packageMerge <- s.packageMerge,
        packagedElement <- s.packagedElement,
        profileApplication <- s.profileApplication)
    do {
        t.applyProfile(thisModule.Global_RIAProfile);
    }
}



---
rule Package {
    from s : UML2!"uml::Package" in IN ( s.oclIsTypeOf(UML2!"uml::Package")
                                          and s.hasAsynchronousService() )
    using {

        --- DATOS INVARIABLES
        class_stereotype : UML2!Stereotype =
            thisModule.getStereotype('asynchronousCall');

        --- DATOS QUE PUEDO INFERIR
        FLAG : Integer = s.propertiesWithAS.size();

        keys : Sequence(Integer) =
            FLAG.makeSequence();
        services : Map(Integer,String) =
            FLAG.makeMap(s.getAsynchronousServicesNames());
```

```
    classes : Map(Integer,String) =
        FLAG.makeMap(s.getAsynchronousServicesNames().toClassNames());
    suppliers : Map(Integer,String) =
        FLAG.makeMap(s.getNamesOfPropertiesWithAS());
    suppliers_stereotypes : Map(Integer,String) =
        FLAG.makeMap(s.getStNamesOfPropertiesWithAS());
    suppliers_classes : Map(Integer,String) =
        FLAG.makeMap(s.getClassesOfPropertiesWithAS());

    --- DATOS QUE NECESARIAMENTE DEBEN SER DEFINIDOS DE MANERA MANUAL
    properties : Map(Integer,Sequence(String)) =
        let object : "#native"!"atl::conf::ConfM2M" =
            "#native"!"atl::conf::ConfM2M".newInstance()
                in object.getPropiedades(suppliers_classes,suppliers);
    properties_stereotypes : Map(Integer,Map(String, String)) =
        let object : "#native"!"atl::conf::ConfM2M" =
            "#native"!"atl::conf::ConfM2M".newInstance()
                in object.getEstereotipoDePropiedades(suppliers_classes,
                                                      suppliers);

}
to t1 : UML2!"uml::Package" (
    __xmiID__ <- s.__xmiID__,
    name <- s.name,
    visibility <- s.visibility,
    eAnnotations <- s.eAnnotations,
    ownedComment <- s.ownedComment,
    nameExpression <- s.nameExpression,
    elementImport <- s.elementImport,
    packageImport <- s.packageImport,
    ownedRule <- s.ownedRule,
    templateParameter <- s.templateParameter,
    templateBinding <- s.templateBinding,
    ownedTemplateSignature <- s.ownedTemplateSignature,
    packageMerge <- s.packageMerge,
    packagedElement <- s.packagedElement ->
                    union( keys -> collect( k |
                        let pKey : String =
                            suppliers_classes.get(k).concat('#')
                                .concat(suppliers.get(k))
                        in let clase : UML2!Class =
                            thisModule.CreateClass(
                                Tuple{
                                    name = classes.get(k),
                                    propertiesNames = properties.get(pKey)
                                }
                            )
                        in let dependencia : UML2!Dependency =
                            thisModule.CreateDependency(
                                Tuple{
                                    name = suppliers.get(k),
                                    client = clase,
                                    supplier = suppliers.get(k)
                                }
                            )
                        in Sequence{clase,dependencia} )
                    ) -> flatten(),
```

```
        profileApplication <- s.profileApplication)
    do {

            for (k in keys) {
                thisModule.getClass(classes.get(k))
                    .applyStereotype(class_stereotype);
            }

            for (k in keys) {
                thisModule.getClass(classes.get(k)).setValue(class_stereotype,
                    'service', thisModule.getState(services.get(k)));
             }

            for (k in keys) {
                for (t in thisModule.getClass(classes.get(k)).getOwnedAttributes())
                {
                    let pKey : String =
                        suppliers_classes.get(k).concat('#').concat(suppliers.get(k))
                    in t.applyStereotype(
                        thisModule.getStereotype(properties_stereotypes.get(pKey)
                            .get(t.name))
                    );
                }
            }

            for (k in keys) {
                let prop : UML2!Property =
                    thisModule.getProperty(suppliers.get(k)) in
                let prop_stro : UML2!Stereotype =
                    prop.getStereotype(suppliers_stereotypes.get(k)) in
                        prop.setValue (prop_stro, 'service', OclUndefined);
            }

    }
}



---
rule Class {
    from s : UML2!"uml::Class" in IN (s.oclIsTypeOf(UML2!"uml::Class"))
    to t : UML2!"uml::Class" (
        __xmiID__ <- s.__xmiID__,
        name <- s.name,
        visibility <- s.visibility,
        isLeaf <- s.isLeaf,
        isAbstract <- s.isAbstract,
        isActive <- s.isActive,
        eAnnotations <- s.eAnnotations,
        ownedComment <- s.ownedComment,
        nameExpression <- s.nameExpression,
        elementImport <- s.elementImport,
        packageImport <- s.packageImport,
        ownedRule <- s.ownedRule,
        templateParameter <- s.templateParameter,
        templateBinding <- s.templateBinding,
        ownedTemplateSignature <- s.ownedTemplateSignature,
```

```
        generalization <- s.generalization,
        powertypeExtent <- s.powertypeExtent,
        redefinedClassifier <- s.redefinedClassifier,
        substitution <- s.substitution,
        representation <- s.representation,
        collaborationUse <- s.collaborationUse,
        ownedUseCase <- s.ownedUseCase,
        useCase <- s.useCase,
        ownedAttribute <- s.ownedAttribute,
        ownedConnector <- s.ownedConnector,
        ownedBehavior <- s.ownedBehavior,
        classifierBehavior <- s.classifierBehavior,
        interfaceRealization <- s.interfaceRealization,
        nestedClassifier <- s.nestedClassifier,
        ownedOperation <- s.ownedOperation,
        ownedReception <- s.ownedReception)
    do {
        thisModule.applyRIAStereoTypes(false,s,t);
    }
}



---
rule Property {
    from s : UML2!"uml::Property" in IN (s.oclIsTypeOf(UML2!"uml::Property"))
    to t : UML2!"uml::Property" (
        __xmiID__ <- s.__xmiID__,
        name <- s.name,
        visibility <- s.visibility,
        isLeaf <- s.isLeaf,
        isStatic <- s.isStatic,
        isOrdered <- s.isOrdered,
        isUnique <- s.isUnique,
        isReadOnly <- s.isReadOnly,
        isDerived <- s.isDerived,
        isDerivedUnion <- s.isDerivedUnion,
        aggregation <- s.aggregation,
        eAnnotations <- s.eAnnotations,
        ownedComment <- s.ownedComment,
        nameExpression <- s.nameExpression,
        type <- s.type,
        upperValue <- s.upperValue,
        lowerValue <- s.lowerValue,
        templateParameter <- s.templateParameter,
        deployment <- s.deployment,
        redefinedProperty <- s.redefinedProperty,
        defaultValue <- s.defaultValue,
        subsettedProperty <- s.subsettedProperty,
        association <- s.association,
        qualifier <- s.qualifier)
    do {
            thisModule.applyRIAStereoTypes(false,s,t);
    }
}
```

Matched Rules para persistencia móvil

```
--- Se debe ejecutar primero!! Sino, no puedo aplicar los estereotipos de RIA_PROFILE
rule Model {
    from s : UML2!"uml::Model" in IN
    to t : UML2!"uml::Model" (
        __xmiID__ <- s.__xmiID__,
        name <- s.name,
        visibility <- s.visibility,
        viewpoint <- s.viewpoint,
        eAnnotations <- s.eAnnotations,
        ownedComment <- s.ownedComment,
        nameExpression <- s.nameExpression,
        elementImport <- s.elementImport,
        packageImport <- s.packageImport,
        ownedRule <- s.ownedRule,
        templateParameter <- s.templateParameter,
        templateBinding <- s.templateBinding,
        ownedTemplateSignature <- s.ownedTemplateSignature,
        packageMerge <- s.packageMerge,
        packagedElement <- s.packagedElement,
        profileApplication <- s.profileApplication)
    do {
        t.applyProfile(thisModule.Global_MobileProfile);
    }
}
```

```
---
rule Class {
    from s : UML2!"uml::Class" in IN (s.oclIsTypeOf(UML2!"uml::Class"))
    to t : UML2!"uml::Class" (
        __xmiID__ <- s.__xmiID__,
        name <- s.name,
        visibility <- s.visibility,
        isLeaf <- s.isLeaf,
        isAbstract <- s.isAbstract,
        isActive <- s.isActive,
        eAnnotations <- s.eAnnotations,
        ownedComment <- s.ownedComment,
        nameExpression <- s.nameExpression,
        elementImport <- s.elementImport,
        packageImport <- s.packageImport,
        ownedRule <- s.ownedRule,
        templateParameter <- s.templateParameter,
        templateBinding <- s.templateBinding,
        ownedTemplateSignature <- s.ownedTemplateSignature,
        generalization <- s.generalization,
        powertypeExtent <- s.powertypeExtent,
        redefinedClassifier <- s.redefinedClassifier,
        substitution <- s.substitution,
        representation <- s.representation,
        collaborationUse <- s.collaborationUse,
```

```
        ownedUseCase <- s.ownedUseCase,
        useCase <- s.useCase,
        ownedAttribute <- s.ownedAttribute,
        ownedConnector <- s.ownedConnector,
        ownedBehavior <- s.ownedBehavior,
        classifierBehavior <- s.classifierBehavior,
        interfaceRealization <- s.interfaceRealization,
        nestedClassifier <- s.nestedClassifier,
        ownedOperation <- s.ownedOperation,
        ownedReception <- s.ownedReception)
    do {
        thisModule.applyMobileStereoTypes(false,s,t);
    }
}
```

```
---
rule Property {
    from s : UML2!"uml::Property" in IN (s.oclIsTypeOf(UML2!"uml::Property"))
    to t : UML2!"uml::Property" (
        __xmiID__ <- s.__xmiID__,
        name <- s.name,
        visibility <- s.visibility,
        isLeaf <- s.isLeaf,
        isStatic <- s.isStatic,
        isOrdered <- s.isOrdered,
        isUnique <- s.isUnique,
        isReadOnly <- s.isReadOnly,
        isDerived <- s.isDerived,
        isDerivedUnion <- s.isDerivedUnion,
        aggregation <- s.aggregation,
        eAnnotations <- s.eAnnotations,
        ownedComment <- s.ownedComment,
        nameExpression <- s.nameExpression,
        type <- s.type,
        upperValue <- s.upperValue,
        lowerValue <- s.lowerValue,
        templateParameter <- s.templateParameter,
        deployment <- s.deployment,
        redefinedProperty <- s.redefinedProperty,
        defaultValue <- s.defaultValue,
        subsettedProperty <- s.subsettedProperty,
        association <- s.association,
        qualifier <- s.qualifier)
    do {
            thisModule.applyMobileStereoTypes(false,s,t);
    }
}
```

4.6. Archivos de configuración

Los archivos de configuración permiten al diseñador controlar algunos aspectos del proceso de transformación, permitiendo la consecución de dos importantes capacidades: por un lado, la posibilidad de capturar decisiones de diseño específicas del sistema bajo desarrollo que de otra manera no podrían ser automatizadas, y por el otro, la posibilidad de que el diseñador tenga cierto grado de injerencia sobre las reglas de transformación, incluso sin tener acceso a ellos.

Han sido contemplados dos archivos de configuración diferentes. Uno que permite indicar cuales elementos del modelo sí deben ser transformados, y cuales no deben serlo, al verse afectados por las reglas de transformación, cuyo nombre es "ArchConfTransformacion.yaml" (e.g., figura 4.12), y otro que permite especificar algunas propiedades para las clases creadas automáticamente como consecuencia de la ejecución de las reglas de transformación M2M, llamado "ArchConfAsynchronousCall.yaml" (e.g., figura 4.13).

El archivo de configuración "ArchConfTransformacion.yaml" dispone de dos modos de funcionamiento. El modo 1, que permite especificar cuales elementos del modelo sí deben transformarse, y el modo 2, que permite especificar cuales no. Indicación que solo aplica para elementos del modelado que machean con las reglas de transformación. La estructura de este archivo es muy simple. Dispone de tres secciones: en la primera se indica el modo de funcionamiento; en la segunda, las clases afectas; y en la tercera, las propiedades afectadas. En el archivo, una clase es referida indicando el paquete que la contiene y su nombre, mientras que una propiedad es referida indicando el paquete que la contiene, la clase que la contiene, y finalmente, su nombre.

El archivo de configuración "ArchConfAsynchronousCall.yaml" especifica las pro-

```
mod: 1
clases:
    - paquete     : paquete_que_contiene_a_la_clase
      clase       : nombre_de_la_clase
propiedades:
    - paquete     : paquete_que_contiene_a_la_propiedad
      clase       : clase_que_contiene_a_la_propiedad
      propiedad   : nombre_de_la_propiedad
```

Figura 4.12: ArchConfTransformacion.yaml – Ejemplo

```
clases:
    - clasePIM   : nombre_de_la_clase_en_el_PIM
      atributoPIM: nombre_de_la_propiedad_en_el_PIM
      propiedades:
        - nonmbre    : nombre_de_la_nueva_propiedad
          estereotipo: estereotipo_de_la_nueva_propiedad
```

Figura 4.13: ArchConfAsynchronousCall.yaml – Ejemplo

piedades a ser agregadas a las clases asíncronas creadas automáticamente. Tiene una sola sección que permite identificar las clases a las que se deben vincular las propiedades. Como al momento de identificar estas clases, las mismas todavía no fueron creadas, se las identifica de manera indirecta a partir de elementos del PIM, concretamente, la clase y el atributo que contienen al servicio asíncrono que justifican la creación automática de cada una de ellas.

Ambos archivos de configuración son procesados mediante la llamada a código Java nativo, directamente desde las reglas de transformación M2M definidas. El código Java en cuestión fue desarrollado a medida para este trabajo y se encuentra expuesto en el apéndice C, donde también se detalla el proceso de configuración que permitió que los ejecutables Java estén disponibles el classpath del IDE utilizado, algo primordial para la correcta resolución de las llamadas.

Antes de cerrar esta sección es necesario dejar en claro algunas consideraciones que deben ser tenidas en cuenta a la hora de utilizar los archivos de configuración propuestos.

- Las indicaciones del archivo "ArchConfTransformacion.yaml" solo aplican para elementos del modelado que machean con las reglas de transformación M2M ejecutadas al momento de consultar el archivo. Es decir, el archivo solo es consultado al momento de aplicar alguna transformación sobre algún elemento del modelo.

- La no existencia del archivo "ArchConfTransformacion.yaml" implica que las reglas de transformación serán ejecutadas tal cual fueron definidas, sin ningún tipo de personalización o consideración adicional, es decir, que todos los elementos del modelo que son referidos por las reglas de transformación serán transformados.

- También es posible la no existencia del archivo "ArchConfAsynchronousCall.yaml", en este caso lo ocurre es que las clases asíncronas son creadas sin las propiedades que hubiesen sido especificadas en el archivo, y posiblemente sea necesario establecerlas manualmente después, directamente en el modelo generado.

4.7. Desafíos de la transformación M2M

Entre los retos más desafiantes que han sido abordados a la hora de definir y ejecutar las reglas de transformación M2M definidas, merecen destaque:

- La imposibilidad de trabajar con perfiles en modo refinamiento, que forzó la utilización de un compilador diferente al propuesto por defecto.

- La identificación de servicios asíncronos, indispensable para automatizar la creación de las clases asíncronas.

- La correcta configuración del IDE para acceder a métodos Java nativos que permitan el procesamiento de los archivos de configuración.

4.8. Síntesis del capítulo

Este capítulo se introduce con una explicación y una justificación de las decisiones tomadas y las actividades llevadas a cabo en el marco del presente trabajo de tesis. Para posteriormente exponer de manera detallada, el proceso de desarrollo propuesto, los metamodelos y perfiles utilizados, el mapeo y las reglas de transformación derivadas de este mapeo, los archivos de configuración definidos, y finalmente, los desafíos que han sido identificados y abordados.

La primera sección expone algunas consideraciones preliminares, como por ejemplo las taxonomías dentro de las que se enmarcan las reglas de transformación M2M definidas (ver tabla 4.1), el lenguaje de transformación M2M utilizado para la definición de las reglas (ATL), el modo de ejecución de las transformaciones M2M (el modo de ejecución por refinamiento), el IDE para ATL y el motor de transformación utilizados (Eclipse ATL y EMFTVM, respectivamente).

La segunda sección presenta el proceso de desarrollo propuesto, de una manera minuciosa y detallada. Es decir, plantea una serie de herramientas y tecnologías que deben ser utilizados, junto con una secuencia de pasos que deben ser seguidas, para la ejecución exitosa de las transformaciones M2M y M2T. De manera muy resumida, el proceso involucra los siguientes pasos, herramientas y tecnologías: i) modelar el sistema a ser desarrollado utilizando MagicDraw y los metamodelos y perfiles de MoWebA, ii) exportar los modelos creados al formato XMI e importarlos en el IDE Eclipse configurado para la ejecución de las transformaciones M2M, iii) ejecutar las transformaciones M2M, iv) importar los modelos resultantes, todavía en formato XMI, al IDE Eclipse preparado para ejecución de las transformaciones M2T, y finalmente, v) ejecutar las transformaciones M2T.

En la tercer sección del presente capítulo, se explica cuales fueron los metamodelos y perfiles utilizados en cada parte del proceso. Así, para el modelado del sistema bajo desarrollo se utilizó un perfil de MoWebA creado a partir de la unión de los perfiles definidos en [22] y [23] para RIA y persistencia móvil respectivamente. Para el mapeo se utilizó este perfil unificado de MoWebA, el cual permite modelar tanto RIA como persistencia móvil. Mientras que para la ejecución de las reglas de transformación se utilizó el metamodelo definido para UML2 por el proyecto MDT de la fundación Eclipse, junto con el perfil unificado antes mencionado.

La cuarta sección, por otro lado, detalla las reglas de mapeo que fueron detectadas a partir del análisis de los perfiles de MoWebA orientados al modelado de arquitecturas RIA y al modelado de elementos arquitectónicos propios de la persistencia móvil.

La quinta sección expone las reglas de transformación M2M definidas a partir de reglas de mapeo detectadas y señaladas en la sección anterior. Presenta aquellas reglas de transformación M2M definidas tanto para la transformación PIM-ASM de modelos orientados a RIA como para la transformación PIM-ASM de modelos orientados a persistencia móvil. Mientras la sexta sección explica que son y para qué sirven los archivos de configuración definidos, y que complementan a las reglas de transformación M2M definidas en la quinta sección.

Finalmente, la séptima sección describe los desafíos más importantes a la hora de definir, configurar y ejecutar el proceso de desarrollo propuesto.

Capítulo 5

Validación

Las reglas de transformación M2M presentadas en capítulo anterior (capítulo 4, sección 4.5) han sido utilizadas para la ejecución de un caso práctico, cuyo principal objetivo consistió en mostrar que es perfectamente posible obtener varios modelos ASM a partir de un único modelo PIM.

Por lo tanto, para la ejecución del caso práctico, se utilizó un único PIM, a partir del cual, mediante la utilización de las reglas de transformación M2M definidas en el capítulo anterior, se buscó obtener dos modelos ASM diferentes, uno orientado a arquitecturas RIA y otro orientado a persistencia móvil.

Una vez finalizadas las transformaciones M2M y obtenidos los modelos ASM, estos fueron comparados con los ASM definidos en [22] (en el caso de RIA) y en [23] (en el caso de persistencia móvil). La idea detrás de esto es que, si los modelos obtenidos mediante las transformaciones M2M definidas son idénticos a los definidos en ambos trabajos, entonces sería posible concluir que, además de haber obtenido los ASM correctos, estos podrían ser utilizados para la obtención del código final, algo alcanzado de manera exitosa en ambos trabajos.

Para la experiencia de transformación, se diseñó un sistema de marcación de empleados. Este sistema permite que un usuario invitado se registre como empleado para luego iniciar sesión y poder realizar marcaciones de entrada y salida. Además, si el usuario es un supervisor, puede observar las marcaciones realizadas por los empleados.

Al modelar el sistema de marcación de empleados, se crearon todos los modelos y diagramas correspondientes al CIM (modelos de casos de uso) y PIM (modelos de entidades, modelo navegacional, modelos de comportamiento, modelos de presentación y modelos de usuario), todos siguiendo las directrices establecidas por MoWebA.

Tanto los diagramas correspondientes al PIM, como aquellos diagramas correspondientes al ASM para RIA y al ASM para persistencia móvil, que sufrieron variaciones con respecto al PIM, son presentados en las siguientes secciones.

Con el objetivo de facilitar la identificación de las transformaciones M2M realizadas de manera automática, en cada uno de los diagramas correspondientes a los modelos ASM, las clases que sufrieron variaciones fueron señaladas con el color azul, mientras que las clases nuevas, es decir, aquellas creadas por las reglas de transformación M2M, fueron señaladas con el color verde. Además, como en las siguientes secciones no se muestran todos los diagramas correspondientes a los ASM generados para RIA y persistencia móvil, los restantes fueron ubicados en los apéndices D y E.

5.1. Modelado del PIM

A continuación se detalla el diseño del sistema de marcación de empleados, concretamente el PIM, el cual ha sido modelado siguiendo las directrices establecidas por MoWebA, el enfoque metodológico MDD abordado por este trabajo de tesis y utilizado en la ejecución del caso práctico.

5.1.1. Modelado del Árbol Navegacional

El *Árbol Navegacional* representa el espacio navegacional de la aplicación y está compuesto por nodos y/o enlaces. Un nodo navegacional representa una unidad funcional básica del sistema, y se conecta con otros nodos por medio de enlaces. Los enlaces fuertes (hLinks), denotan una jerarquía en el árbol de navegación, mientras los débiles (hLinks) modelan navegaciones no jerárquicas.

El árbol navegacional correspondiente al sistema de marcación de empleados se detalla en la figura 5.1.

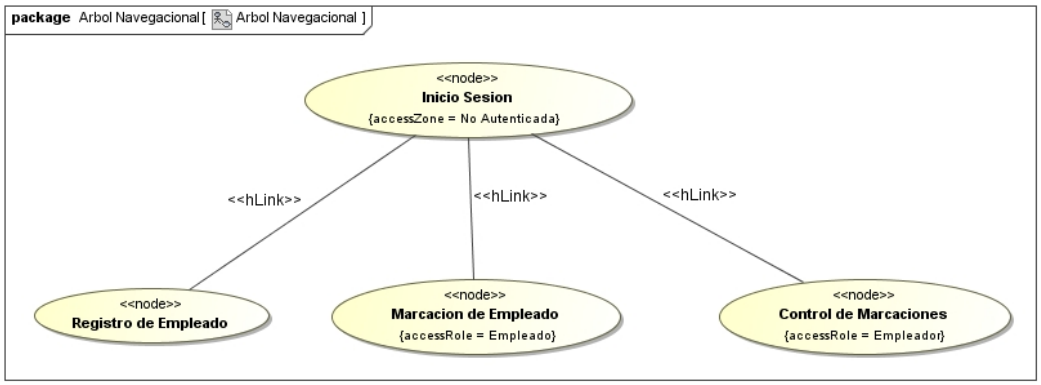


Figura 5.1: Diagrama de Árbol Navegacional – PIM

5.1.2. Modelado de los Diagramas de Contenido

Los *Diagramas de Contenido*, son diagramas que permiten especificar los diferentes elementos que serán presentados a los usuarios del sistema. Estos diagramas se componen de páginas de presentación (clases estereotipadas con «presentationPage»), que a su vez se componen de uno o más elementos. Estos elementos pueden ser: formularios, estereotipados con «form»; tablas, con «table»; elementos compuestos de interfaz, con «compositeUIElement»; objetos valuados, con «valueObject»; y objetos estáticos, estereotipados con «staticObject» (encapsulan datos que no dependen de ninguna entidad).

Los elementos compuestos de interfaz definen atributos de la interfaz de usuario como títulos, texto informativo, el submit de un formulario, etc. Los objetos valuados encapsulan datos que dependen de una o más entidades, mientras que los objetos estáticos encapsulan datos que no dependen de ninguna entidad.

5.1.2.1. Iniciar Sesión

El diagrama de contenido para el inicio de sesión es presentado en la figura 5.2.

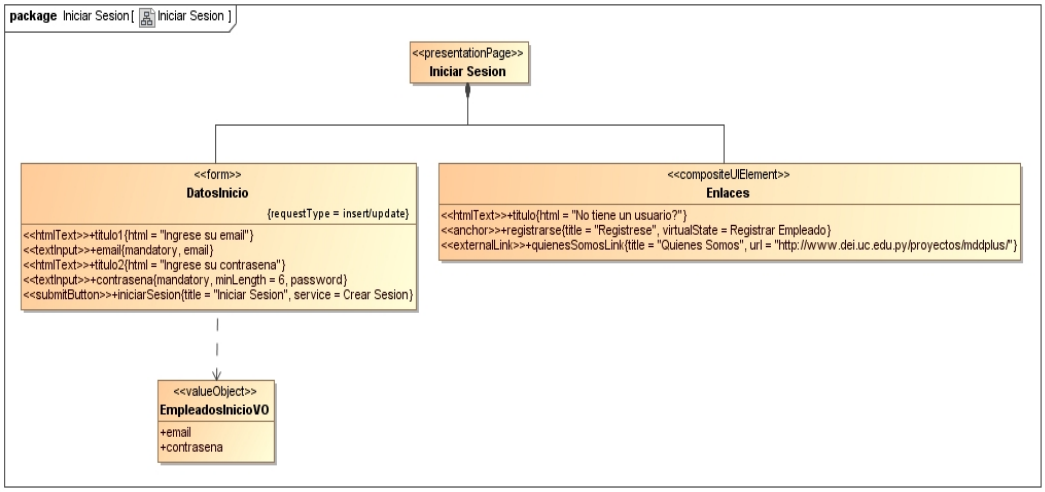


Figura 5.2: Diagrama de Contenido – Iniciar Sesion – PIM

5.1.2.2. Controlar Marcaciones

El diagrama de contenido para el control de marcaciones es presentado en la figura 5.3.

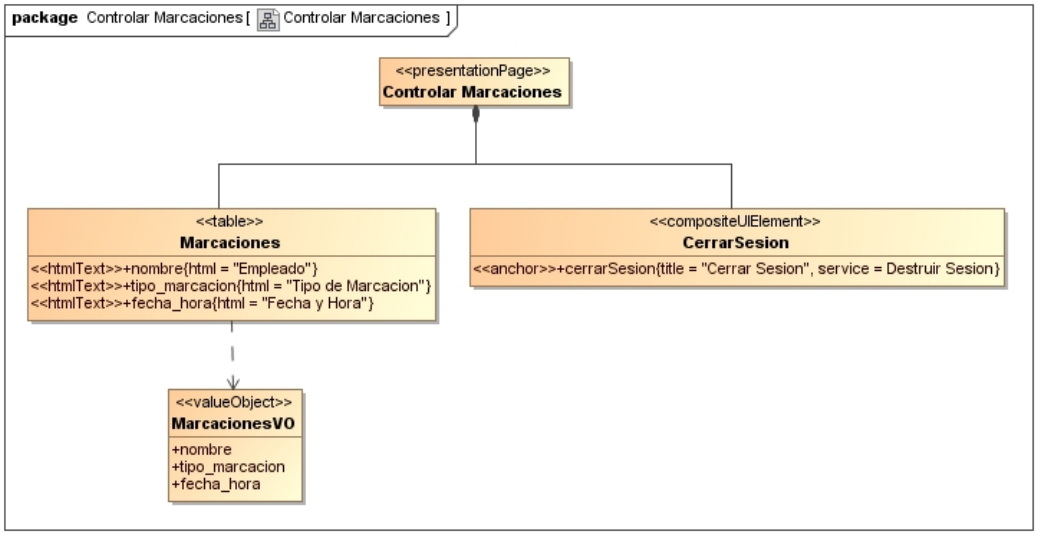


Figura 5.3: Diagrama de Contenido – Controlar Marcaciones – PIM

5.1.2.3. Realizar Marcación

El diagrama de contenido para la realización de una marcación es presentado en la figura 5.4.

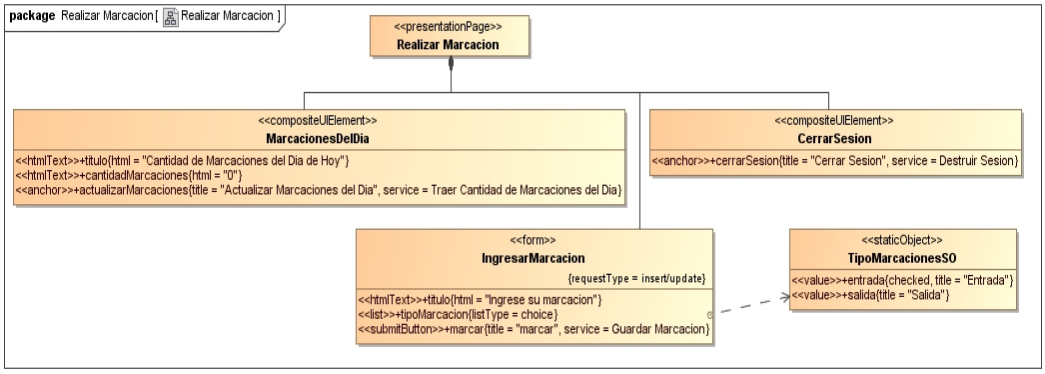


Figura 5.4: Diagrama de Contenido – Realizar Marcación – PIM

5.1.2.4. Registrar Empleado

El diagrama de contenido para el registro de un nuevo empleado es presentado en la figura 5.5.

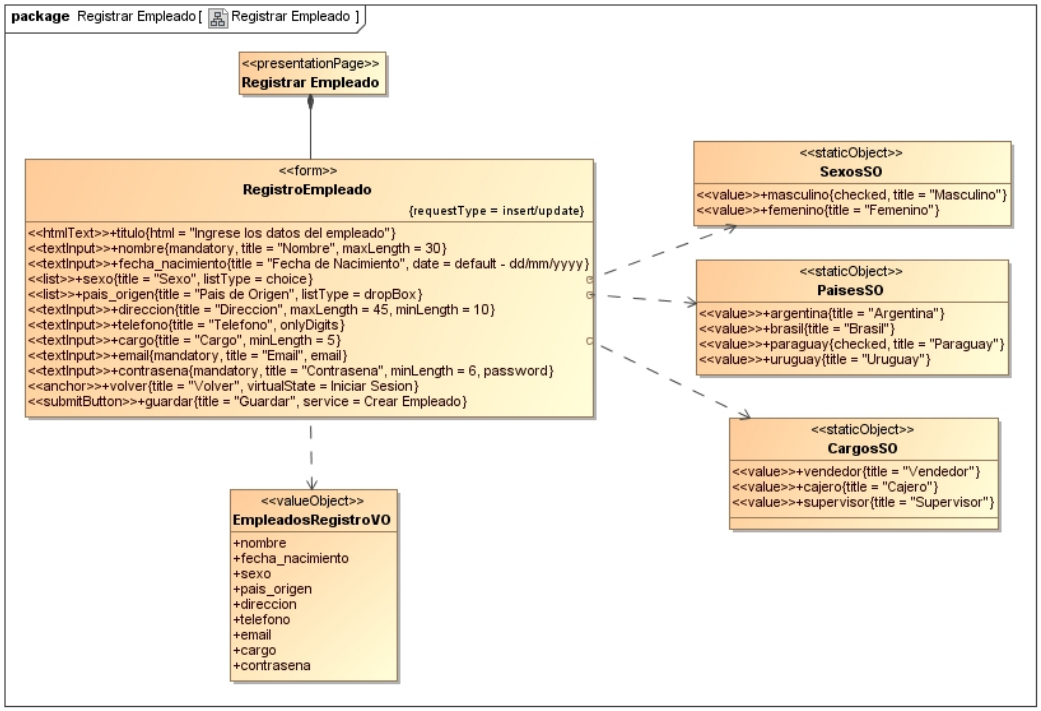


Figura 5.5: Diagrama de Contenido – Registrar Empleado – PIM

5.1.3. Modelado del Diagrama de Entidades

El *Diagrama de Entidades* define la estructura y las relaciones estáticas entre clases identificadas en el dominio del problema. Los elementos que componen este diagrama se denominan entidades, y son representadas en el diagrama con clases estereotipadas con «entity». La figura 5.6 presenta el diagrama de entidades modelado.

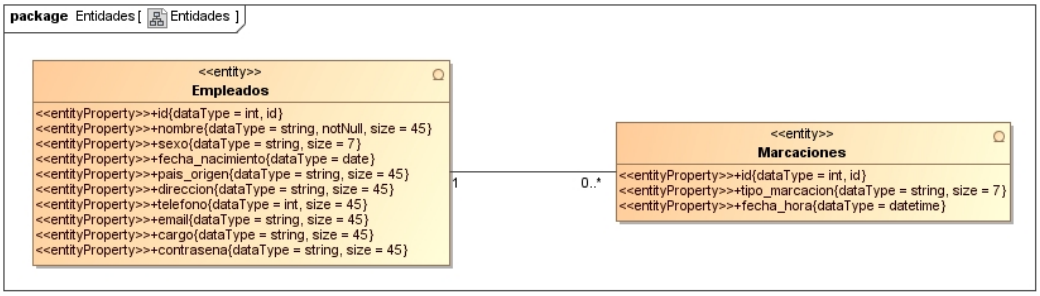


Figura 5.6: Diagrama de Entidades – PIM

5.1.4. Modelado del Diagrama Lógico

El *Diagrama Lógico* es un diagrama orientado a modelar la lógica del negocio. Permite el modelado de objetos valuados, objetos estáticos, procesos del negocio (clases estereotipadas con «tProcess»). El diagrama lógico modelado para el sistema de marcación de empleados se presenta en la figura 5.7.

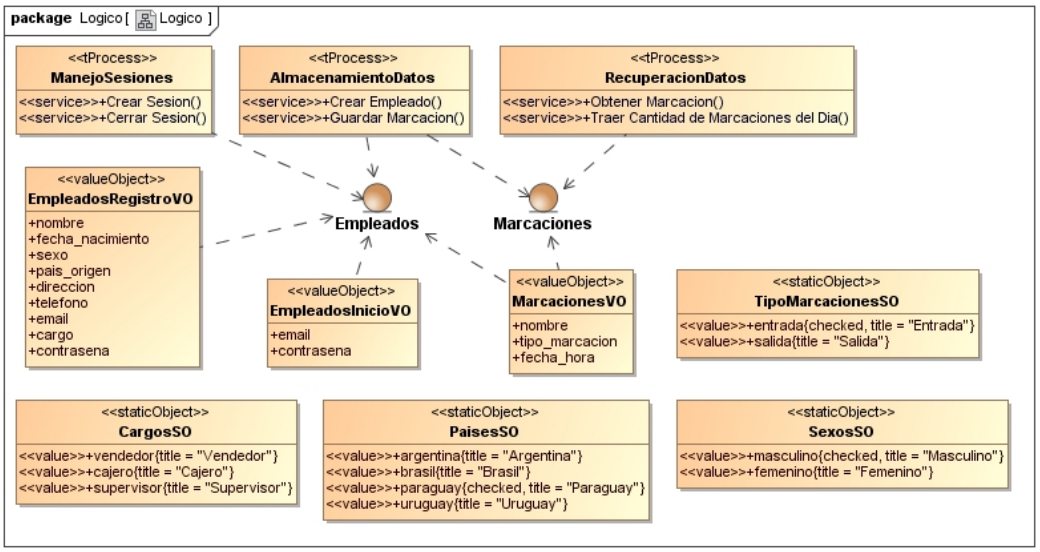


Figura 5.7: Diagrama Lógico – PIM

5.1.5. Modelado de los Diagramas de Nodos

Los *Diagramas de Nodos* permiten modelar aspectos de comportamiento relacionados con navegaciones dinámicas obtenidas de interacciones con el usuario y representan la navegación interna de cada nodo del árbol navegacional.

5.1.5.1. Inicio Sesión

El diagrama de nodos para el inicio de sesión es presentado en la figura 5.8.

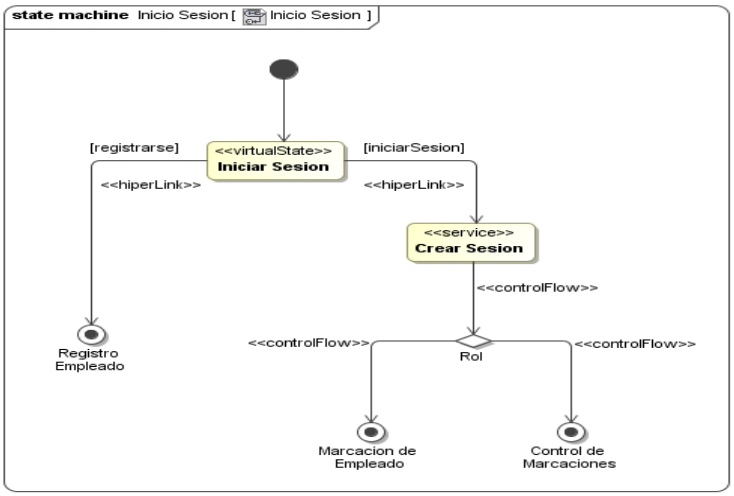


Figura 5.8: Diagrama de Nodos – Inicio Sesion – PIM

5.1.5.2. Control de Marcaciones

El diagrama de nodos para el control de marcaciones se presenta en la figura 5.9.

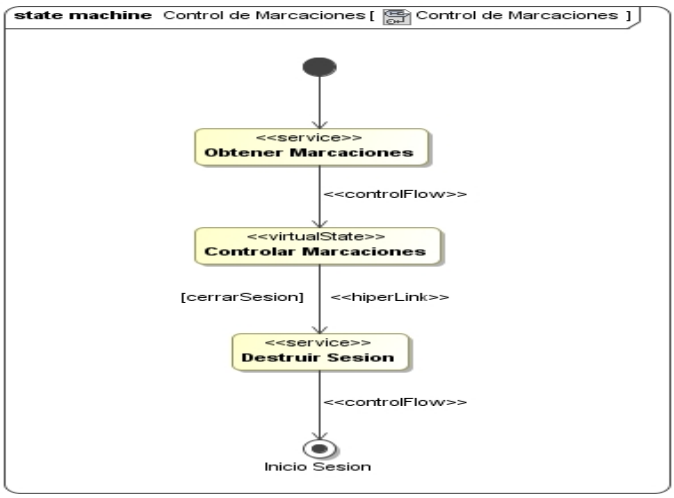


Figura 5.9: Diagrama de Nodos – Control de Marcaciones – PIM

5.1.5.3. Marcación de Empleado

El diagrama de nodos para la marcación de un empleado es presentado en la figura 5.10.

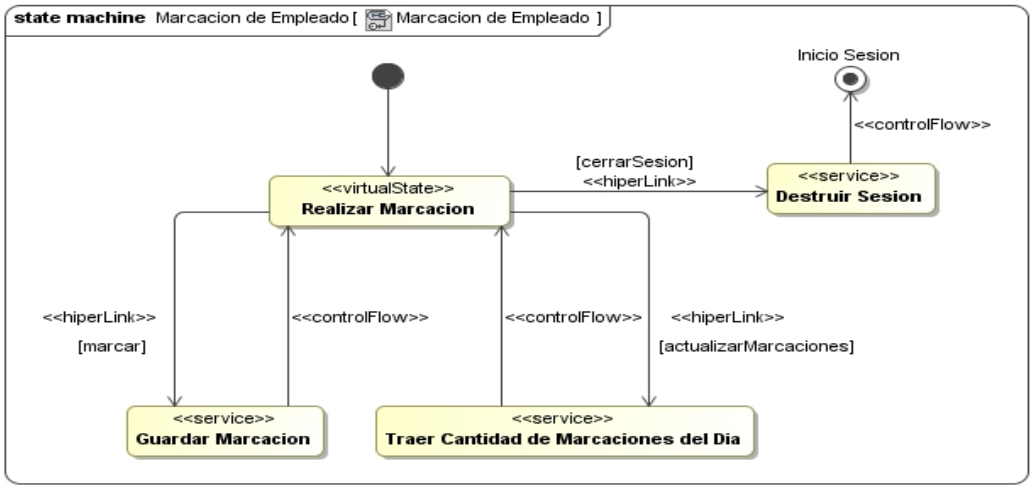


Figura 5.10: Diagrama de Nodos – Marcación de Empleado – PIM

5.1.5.4. Registro de Empleado

El diagrama de nodos para el registro de un empleado nuevo es presentado en la figura 5.11.

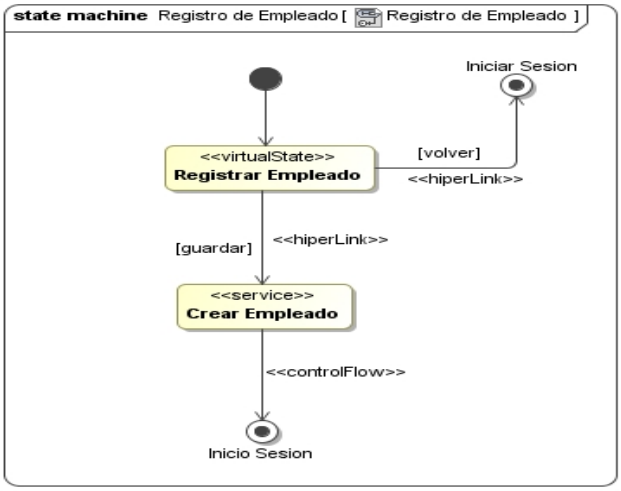


Figura 5.11: Diagrama de Nodos – Registro de Empleado – PIM

5.1.6. Modelado del Diagrama de Roles

El *Diagrama de Roles* representa las jerarquías de roles de usuarios. Se utiliza la notación de jerarquía de actores del diagrama de casos de uso. El diagrama de roles modelado para el sistema de marcación de empleados es presentado en la figura 5.12.

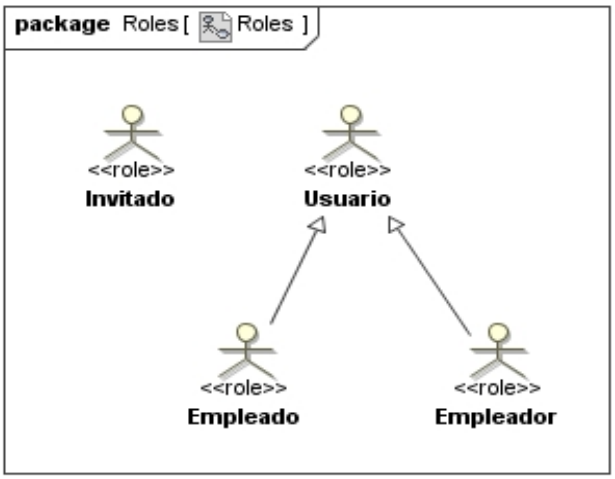


Figura 5.12: Diagrama de Roles – PIM

5.1.7. Modelado del Diagrama de Zonas

El *Diagrama de Zonas* define zonas de navegación que representan a contextos con determinados perfiles de comportamiento que mantienen relación entre sí. El diagrama de zonas modelado es presentado en la figura 5.13.

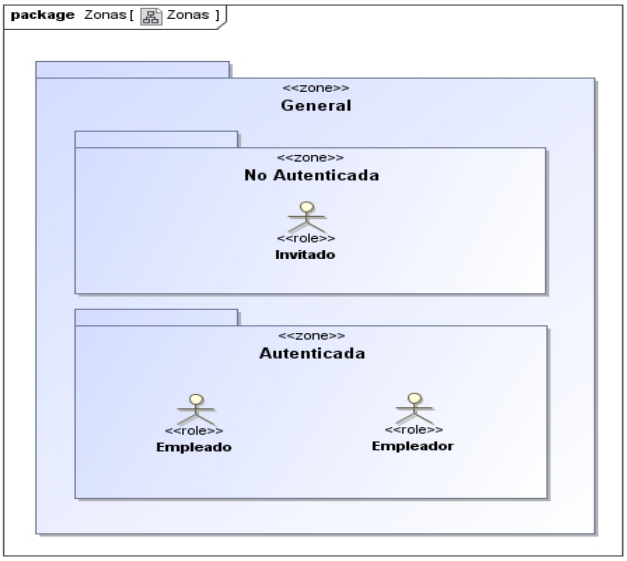


Figura 5.13: Diagrama de Zonas – PIM

5.2. Definición de los archivos de configuración

Aquí se detallan los archivos de configuración utilizados, durante la ejecución del caso práctico, para i) personalizar las transformaciones M2M definidas y ii) adaptarlas a las particularidades del sistema modelado.

Como solo fue necesario definir archivos de configuración para complementar las reglas orientadas RIA, los dos archivos a continuación refieren a esta arquitectura.

5.2.1. Complemento para la transformación de clases

Archivo externo: ArchConfTransformacion.yaml

```
mod: 1
clases:
    - paquete     : Logico
      clase       : CargosSO
    - paquete     : Logico
      clase       : PaisesSO
    - paquete     : Iniciar Sesion
      clase       : DatosInicio
    - paquete     : Controlar Marcaciones
      clase       : Marcaciones
    - paquete     : Realizar Marcacion
      clase       : IngresarMarcacion
    - paquete     : Registrar Empleado
      clase       : RegistroEmpleado
propiedades:
    - paquete     : Iniciar Sesion
      clase       : DatosInicio
      propiedad   : contrasena
    - paquete     : Registrar Empleado
      clase       : RegistroEmpleado
      propiedad   : nombre
```

La línea uno define el modo de funcionamiento. El modo 1 implica que el archivo señala clases y propiedades que deben ser transformadas al machear con alguna regla. Para más información ver la sección 4.6.

5.2.2. Complemento para la creación de nuevas clases

Archivo externo: ArchConfAsynchronousCall.yaml

```
clases:
    - clasePIM    : MarcacionesDelDia
      atributoPIM: actualizarMarcaciones
      propiedades:
        - nombre     : fechaActual
          estereotipo: callParameter
```

Este archivo de configuración especifica algunas de las propiedades para las clases que serán creadas automáticamente al ejecutarse las transformaciones M2M. Para más información ver la sección 4.6.

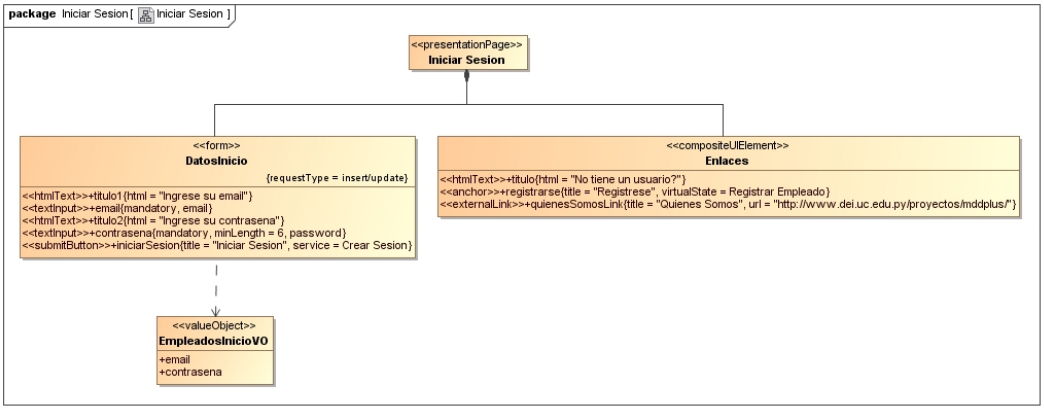
5.3. Ejecución de las transformaciones M2M

En esta sección se presenta una breve discusión acerca de las transformaciones derivadas de la ejecución de las reglas de transformación M2M definidas en el capítulo anterior, concretamente en la sección 4.5.

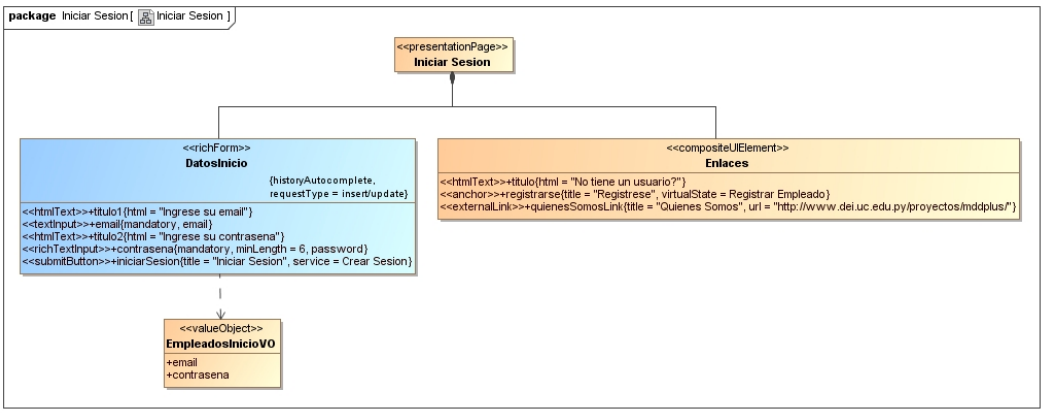
La intención de esta sección es explicar, de manera detallada, cada uno de los diagramas que sufrieron cambios al ser ejecutadas las transformaciones M2M definidas. Se buscó, sobre todo, identificar cada una de las diferencias entre los diagramas correspondientes al PIM y al ASM, señalarlas, y justificar a partir del mapeo presentado en la sección 4.4, el porqué de cada una de ellas.

5.3.1. Resultados para RIA

La figura 5.14 presenta el diagrama de contenido para el inicio de sesión. Este diagrama es el más sencillo de los diagramas de contenido. En él se puede observar una página de presentación[1] denominada "Iniciar sesión", un formulario denominado "DatosInicio", un elemento compuesto de interfaz de usuario[2] denominado "Enlaces" y, finalmente, un objeto valuado denominado "EmpleadosInicioVO".



(a) PIM

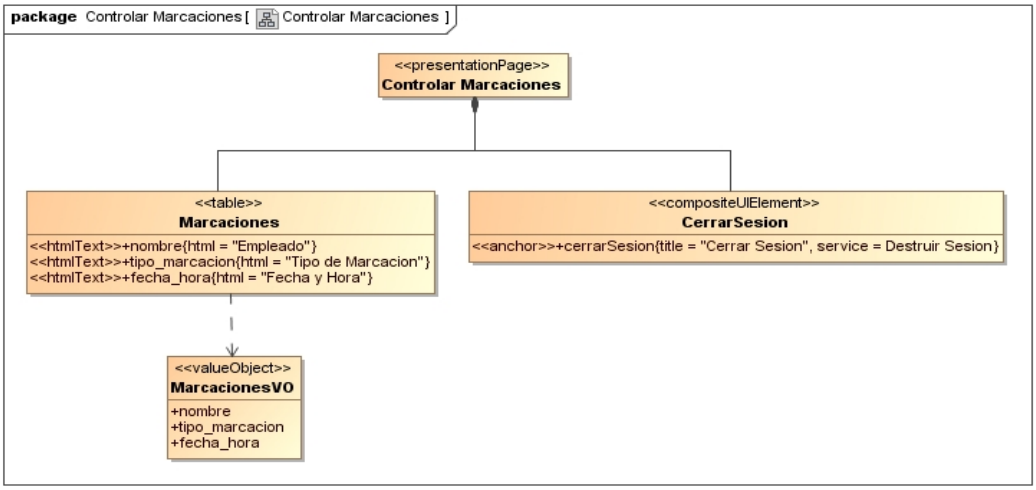


(b) ASM

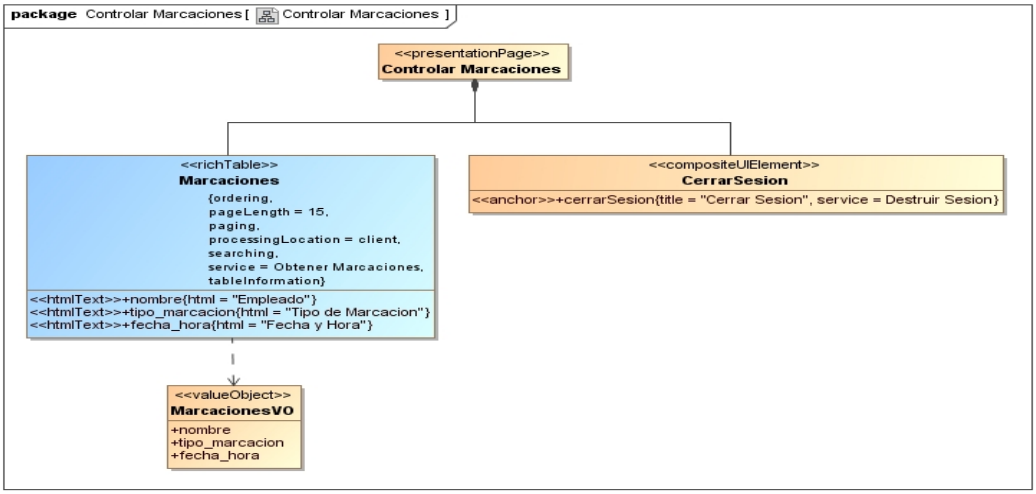
Figura 5.14: Diagrama de Contenido – Iniciar Sesión – PIM vs ASM

De estos elementos, el único que presenta cambios en el diagrama correspondiente al ASM, es el formulario "DatosInicio". Como ya se ha explicado en la sección 4.4, específicamente en el apartado 4.4.1 a través de la figura 4.7, toda vez que una clase con el estereotipo «form» aparece en el PIM (ver figura 5.14a), debe ser transformada a una clase con el estereotipo «richForm» en el ASM (ver figura 5.14b).

Por otro lado, la figura 5.15 presenta el diagrama de contenido para el control de las marcaciones. Este diagrama de contenido cuenta con una página de presentación[1]



(a) PIM



(b) ASM

Figura 5.15: Diagrama de Contenido – Controlar Marcaciones – PIM vs ASM

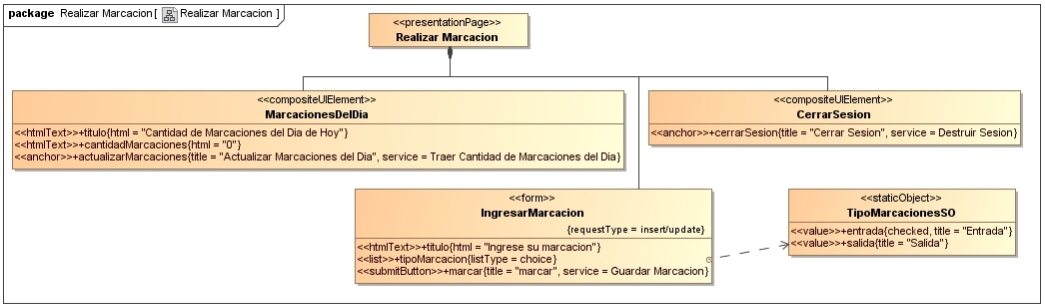
denominada "Controlar Marcaciones", la cual está conformada por dos clases. La tabla "Marcaciones", conformada a su vez por la clase de tipo «valueObject» denominada "MarcacionesVO", y el elemento compuesto de interfaz de usuario[2] denominado

[1]Clase con el estereotipo «presentationPage»
[2]Clase con el estereotipo «compositeUIElement»

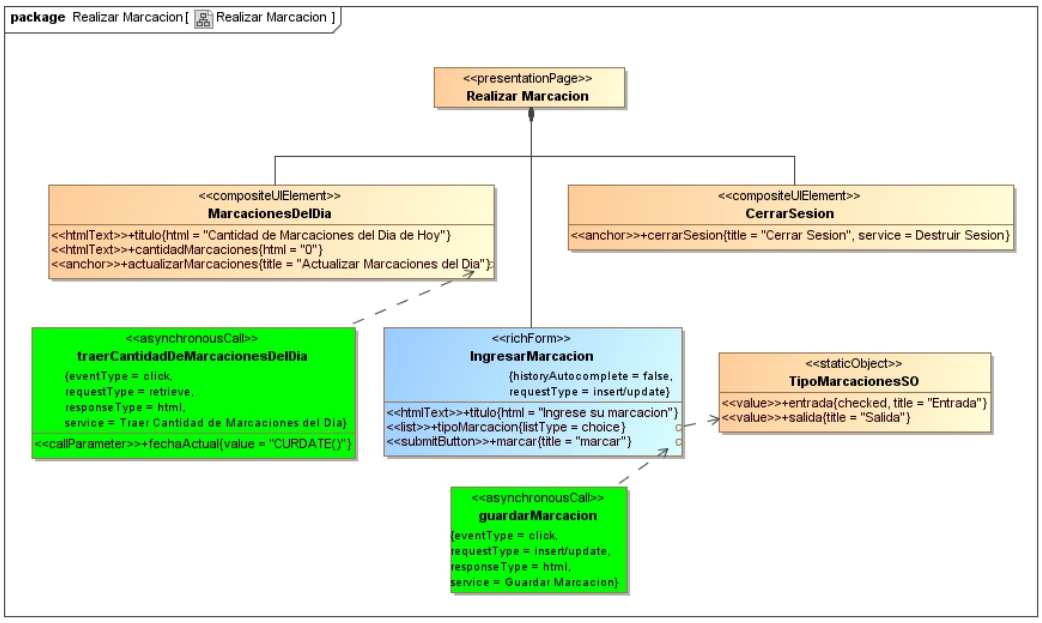
"CerrarSesion".

Al igual que en el caso anterior, aquí también se da que solo uno de los elementos que conforman el diagrama correspondiente al PIM (ver figura 5.15a), la tabla "Marcaciones", presenta cambios en el diagrama correspondiente al ASM (ver figura 5.15a). La justificación para esta transformación también la podemos encontrar en la sección 4.4, esta vez en el apartado 4.4.1, el cual por medio del análisis de la figura 4.6 explica que, toda vez que una clase con el estereotipo «table» aparece en el PIM, ésta debe ser transformada en una clase con el estereotipo «richTable» en el ASM.

La figura 5.16, sin embargo, presenta el diagrama de contenido para la realización de las marcaciones. Este diagrama de contenido cuenta con una página de presenta-



(a) PIM



(b) ASM

Figura 5.16: Diagrama de Contenido – Realizar Marcación – PIM vs ASM

ción[1] denominada "Realizar Marcación", la cual está conformada por dos elementos compuestos de interfaz de usuario[2] denominados "MarcacionesDelDia" y "CerrarSesion", y por el formulario "IngresarMarcacion" relacionado, a su vez, con una clase de tipo «staticObject» denominada "TipoMarcacionesSO".

Como se puede ver en la figura 5.16, la clase “IngresarMarcación” tiene el estereotipo «form» en el PIM (ver figura 5.16a) y el estereotipo «richForm» en el ASM (ver figura 5.16b). Esto es así porque, como ha sido expuesto en el mapeo presentado en la sección 4.4, específicamente en el apartado 4.4.1 a través del análisis de la figura 4.7, siempre que aparece en el PIM una clase con el estereotipo «form», esta debe transformarse en una con el estereotipo «richForm» en el ASM.

Una situación que merece destaque es la que se da con las clases “MarcacionesDelDía” e “IngresarMarcacion” presentes en este diagrama de contenido. Resulta que la clase “MarcacionesDelDia” posee la propiedad “actualizarMarcaciones” (la del estereotipo «anchor» en la figura 5.16), la cual tiene la particularidad de que, al hacer clic en ella, inicia la ejecución del servicio “Traer Cantidad de Marcaciones del Dia” cuya responsabilidad consiste en actualizar la propiedad “Cantidad de Marcaciones del Dia de Hoy” de manera asíncrona. Como podrá haber notado, todo lo mencionado cumple con lo estipulado para la creación automática de clases de tipo «aynchronousCall» (ver subsección 4.4.2). Una situación similar se da con la propiedad de tipo «submitButton» de nombre “marcar” presente en la clase “IngresarMarcacion”, la cual ejecuta de manera asíncrona el servicio “Guardar Marcación”.

Resumiendo, la transformación de PIM a ASM del diagrama de contenido para la realización de marcaciones requirió la transformación del formulario “IngresarMarcacion” y la creación de dos nuevas clases de tipo «asynchronousCall», denominadas “traerDeMarcacionesDelDia” y “guardarMarcacion” (ver figura 5.16b).

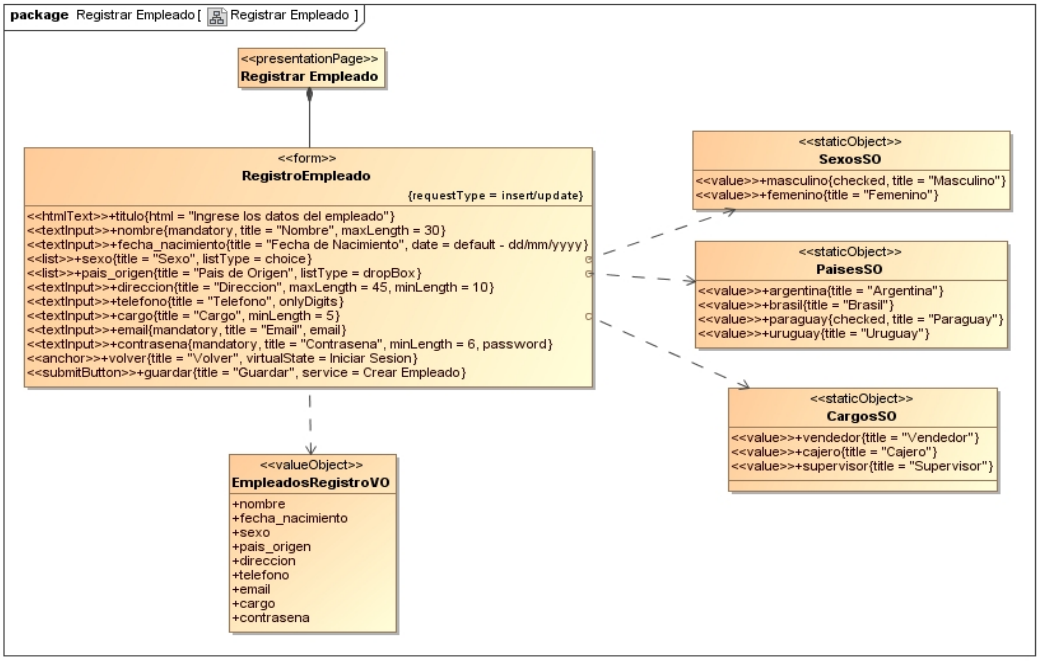
El último de los diagramas de contenido es presentado en la figura 5.17, y es el diagrama de contenido para el registro de empleados. Por su parte, este diagrama de contenido está conformado por la página de presentación “Registrar Empleado”, la que a su vez está relacionada con tres objetos estáticos[3] llamados “SexosSO”, “PaisesSO” y “CargosSO”, y con una clase de tipo «valueObject» llamada “EmpleadosRegistroVO”.

En base al mapeo presentado en la sección 4.4, y luego de analizar cada uno de los elementos en este diagrama de contenido, se llegó a la conclusión de que cinco de ellos pueden sufrir cambios como resultado de la ejecución de las transformaciones M2M definidas. Sin embargo, solo tres de estos cinco elementos registran realmente cambios en el ASM (ver figura 5.17b). La explicación para esto son los archivos de configuración de las transformaciones M2M.

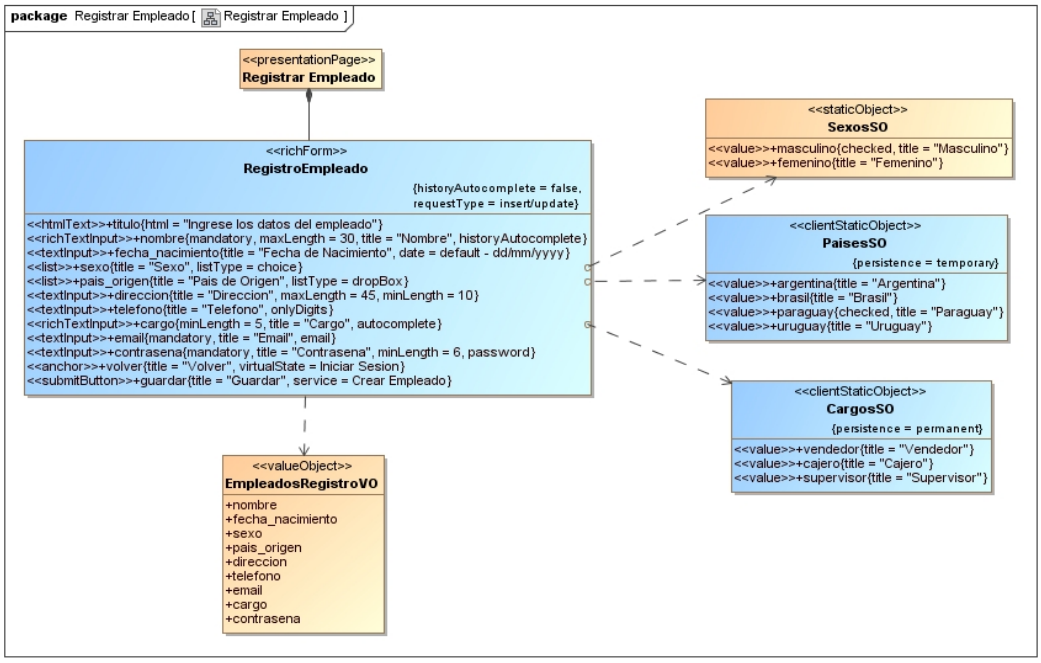
Como ya fue mencionado antes, específicamente en la sección 4.6, el archivo de configuración, entre otras cosas, permite al diseñador cierto grado de ingerencia sobre la ejecución de las transformaciones M2M, sin que para ello tenga que escribir código en ATL, sino que solamente definiendo en los archivos de configuración cuales de los elementos en los diagramas deben ser transformados y cuales, a pesar de cumplir las condiciones necesarias para ello, no deben serlo. Así fue que tanto el objeto estático[3] “SexosSO” como el objeto valuado[4] “EmpleadosRegistroVO”, no fueron transformados. Además, esto se repite en todos los diagramas presentados, solo que no fue explicado hasta ahora.

Los elementos que sí sufrieron cambios luego de la ejecución de las transformaciones M2M, son el formulario “RegistroEmpleado” y los objetos estáticos[3] “PaisesSO” y “CargosSO”.

[3]Clases con el estereotipo «staticObject»
[4]Clases con el estereotipo «valueObject»



(a) PIM



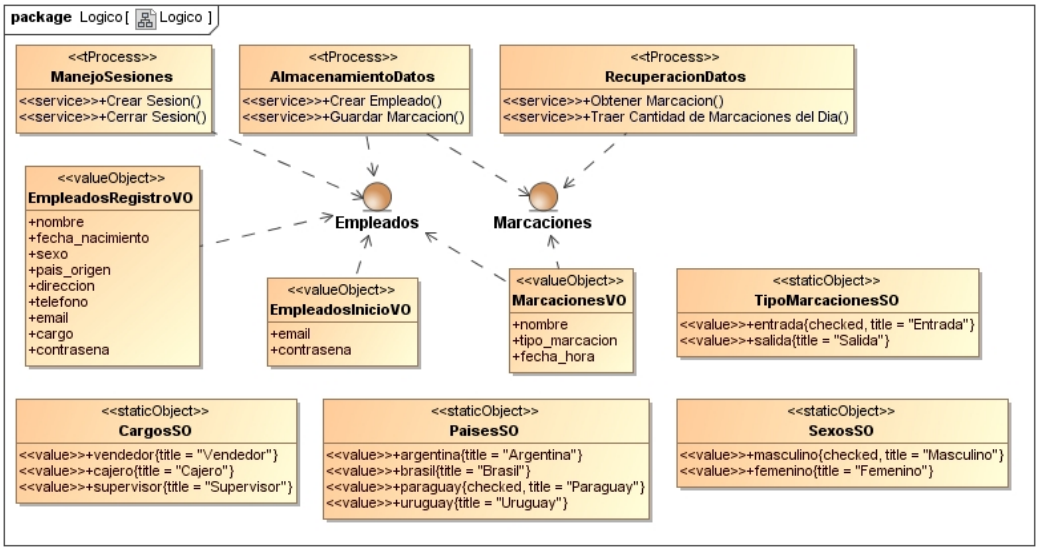
(b) ASM

Figura 5.17: Diagrama de Contenido – Registrar Empleado – PIM vs ASM

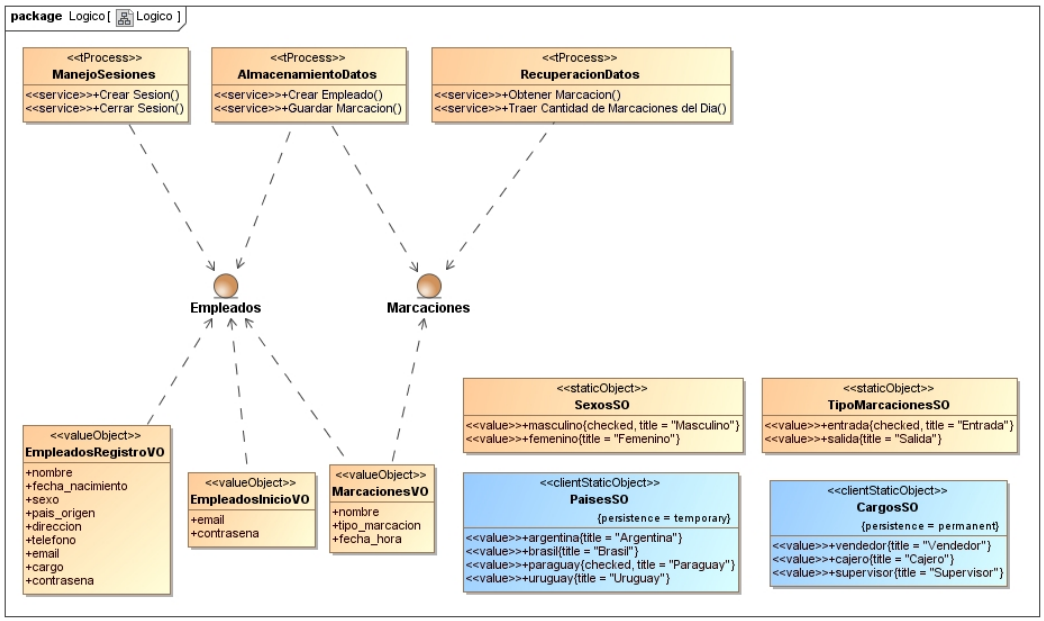
La justificación para la transformación de la clase "RegistroEmpleado", como ya se mencionó antes, se encuentra en el apartado 4.4.1. Mientras que la justificación para la transformación de los objetos estáticos[3] "PaisesSO" y "CargosSO", se encuentra

en el apartado 4.4.1 y, además, en los archivos de configuración presentados en la sección 5.2, donde se indica explícitamente que solo estos dos objetos estáticos deben ser transformados. Sin embargo, a pesar de que la transformación puede ser observada en el diagrama de contenido para el registro de empleados, la transformación no fue realizada allí, puesto que allí solo se la expone. El diagrama en el que se ejecuta realmente la transformación, es el diagrama lógico presentado a continuación.

Finalmente, la figura 5.18a presenta el diagrama lógico de la aplicación. Este define



(a) PIM



(b) ASM

Figura 5.18: Diagrama Lógico – PIM vs ASM

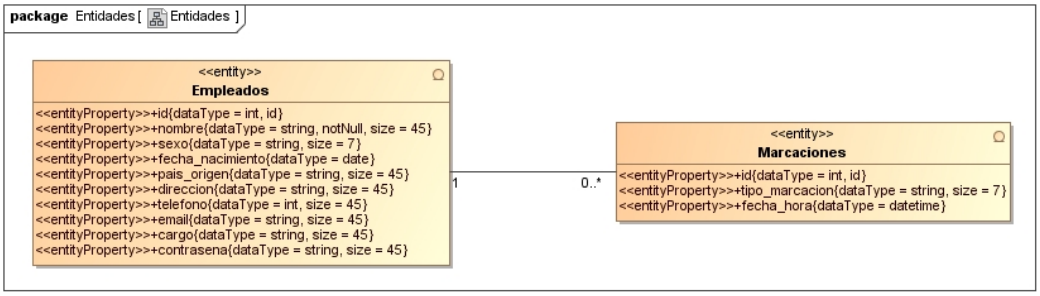
procesos de negocios[5] en las clases denominadas “ManejoSesiones”, “Almacenamiento-Datos” y “RecuperacionDatos”, y objetos valuados[4] en las clases denominadas “EmpleadosRegistroVO”, “EmpleadosInicioVO” y “MarcacionesVO”. Además introduce cuatro objetos estáticos[3] a través de las clases denominadas “SexosSO”, “TipoMarcacionesSO”, “PaisesSO” y “CargosSO”.

En esta figura se pueden identificar muchas clases que pueden ser transformadas siguiendo el mapeo presentado en la sección 4.4. Concretamente, todos los objetos estáticos, junto con todos los objetos valuados. Sin embargo, en la figura 5.18b se observa claramente que solo los objetos estáticos “PaisesSO” y “CargosSO” son afectados por las transformaciones M2M. Esto se debe a lo especificado en el archivo de configuración externo presentado en la sección 5.2, en el apartado 5.2.1. Donde se detallan aquellas clases que deben ser transformadas en caso de verse afectadas por las reglas de transformación M2M definidas.

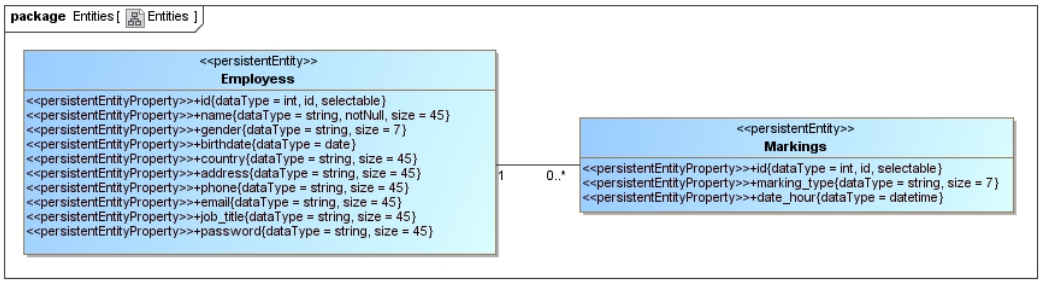
Al igual que en los casos anteriores, la justificación para estas transformaciones se encuentra en la sección 4.4, en el apartado 4.4.1, donde se explica que siempre que una clase con el estereotipo «staticObject» aparezca en el PIM, esta debe ser transformada en una clase con el estereotipo «clientStaticObject» en el ASM.

5.3.2. Resultados para Persistencia Móvil

El diagrama de entidades de la Figura 5.19a consta de dos entidades[6], “Emplea-



(a) PIM



(b) ASM

Figura 5.19: Diagrama de Entidades – PIM vs ASM

dos” y “Marcaciones” con una relación de uno a muchos entre ellas. Se observa que

[5]Clases con el estereotipo «tProcess»
[6]Clases con el estereotipo «entity»

las propiedades de las entidades poseen el estereotipo «entityProperty» permitiendo agregar valores etiquetados para los tipos de datos int, string, date, tamaños máximos (7 y 45) y restricciones id y notNull.

En base al mapeo definido en la sección 4.4, concretamente, en base al análisis de la figura 4.9 presentada en el apartado 4.4.1, es posible apreciar que las entidades con estereotipo «entity» en el PIM, se deben transformar en dos entidades con el estereotipo «persistentEntity» en el ASM. Así también, en base al mapeo definido en la sección 4.4, pero esta vez, observando el análisis de la figura 4.10 presentada en el apartado 4.4.1, las propiedades de estas entidades, todas con el estereotipo «entityProperty» en el PIM, deben transformarse en propiedades con el estereotipo «persistentEntityProperty» en el ASM (ver Figura 5.19b).

5.4. Discusión

La experiencia realizada fue muy positiva debido a que fue posible transformar todas las clases que así lo requerían. Además, el 100 % de clases del ASM fueron derivadas automáticamente a partir de las reglas de transformación M2M definidas en la sección 4.5, y hasta fue posible automatizar la creación de nuevas clases para servicios asíncronos.

Haciendo un recuento rápido, las clases afectadas por las reglas de transformación M2M, es decir, aquellas que sufrieron cambios en el proceso de transformación del PIM al ASM, representan cerca de un tercio de todas las clases presentes en los diagramas correspondientes a los modelos ASM para RIA y persistencia móvil. E incluso, podrían haber sido afectadas un mayor porcentaje de las mismas si se hubiese ignorado lo especificado en los archivos de configuración presentados en la sección 5.2. Esto debido a que, al capturar las decisiones de diseño del caso modelado, los archivos de configuración detuvieron la transformación de varias clases que de otro modo hubiesen sido transformadas.

Además, los tiempos de creación del ASM se redujeron considerablemente en comparación con su creación manual a partir de la modificación del PIM, incluso considerando el tiempo de creación de los archivos de configuración, cuya creación es mucho más simple y rápida que la aplicación manual de los ajustes allí detallados.

Es importante remarcar que al finalizar la ejecución de las reglas de transformación M2M definidas, se han obtenido dos modelos ASM diferentes a partir del único PIM modelado para el sistema de marcación de empleados, es decir, se ha logrado una transformación M2M completa para dos arquitecturas diferentes partiendo del mismo PIM. Además, gracias a la utilización de los archivos de configuración externos, el modelo ASM generado para la arquitectura RIA no requirió de ningún ajuste adicional posterior, algo que sí fue necesario para el modelos ASM correspondientes a la persistencia móvil. Sin embargo, esto es algo que se sabía iba a ocurrir puesto que las reglas de transformación M2M para persistencia móvil no cubrieron la creación de clases nuevas, sino solo la transformación de aquellas pre-existentes.

Una vez finalizada la transformación del PIM al ASM para las dos arquitecturas abordadas (RIA y persistencia móvil), y ya con los objetivos de esta tesis cubiertos, se procedió a analizar la posibilidad de obtener código a partir de los modelos ASM generados para cada una de estas arquitecturas.

Ocurre que la obtención de código a partir de los modelos ASM generados es factible gracias a las reglas de transformación M2T definidas en dos trabajos previos, [22] y [23], llevados a cabo en el marco del mismo proyecto de investigación que permitió la realización de este trabajo de tesis.

El primer paso del análisis consistió en la comparación de los ASM generados, con aquellos definidos en ambos trabajos. Así, el ASM para RIA, generado a partir de la ejecución de las reglas de transformación M2M definidas en la sección 4.5 para este tipo de arquitecturas, fue comparado con el ASM definido en [22]. Mientras que el ASM para persistencia móvil, generado a partir de la ejecución de las reglas de transformación M2M definidas en la sección 4.5 para este tipo de arquitecturas, fue comparado con el ASM definido en [23].

En el primer caso, el ASM para RIA obtenido de manera automática a partir de la ejecución de las reglas de transformación M2M definidas, resulto ser idéntico al ASM definido y utilizado para la obtención de código (a través de transformaciones M2T) en el estudio llevado a cabo en [22]. Por lo tanto, los resultados obtenidos en nuestra experiencia, en términos de la cantidad de código generado y los ajustes manuales realizados al código luego de la transformación M2T, son idénticos a los obtenidos en el mencionado estudio. Más aún, el código final que ha sido obtenido con las herramientas de generación automática cumple con los requerimientos iniciales del sistema [24].

El el caso del ASM generado para persistencia móvil, la comparación con el ASM definido en [23] no pudo ser directa, puesto que el sistema modelado en ambos casos es diferente. Sin embargo, limitando el análisis al diagrama de persistencia únicamente, e ignorando los demás diagramas, se pudo notar que son sumamente similares. Lo que de alguna manera respalda la correctitud del ASM generado. Además, debido a esta diferencia en cuanto al sistema modelado, la obtención de código a partir del ASM para persistencia móvil y presentado en el apéndice E, a pesar de ser posible, no tiene mucho valor ya que no puede ser comparado con el código generado en [23].

Todas la herramientas necesarias para la realización de estas transformacaciones M2M y M2T están disponibles en el siguiente link: `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/`.

5.5. Observaciones finales y oportunidades

Gracias a la experiencia realizada ha sido posible verificar que las transformaciones M2M permiten la obtención de modelos ASM para diferentes arquitecturas a partir de un único modelo PIM, sin necesidad modificarlo, extenderlo o adaptarlo.

Si bien en la experiencia se han considerado solo dos arquitecturas, y en consecuencia, solo se han obtenido dos modelos ASM diferentes (uno orientado a modelar arquitecturas RIA y otro orientado a modelar elementos arquitectónicos propios de la persistencia móvil), se intuye posible obtener/generar modelos ASM para otras arquitecturas diferentes.

La principal ventaja de obtener varios modelos ASM partiendo de un único modelo PIM es que la portabilidad del PIM se conserva de manera considerable y significativa. Es importante recordar que este enfoque (es decir, la captura de los elementos arquitectónicos en el modelo ASM), es muy diferente al de la mayoría de las metodologías

actuales que, a diferencia de MoWebA, tienden a agregar los elementos específicos de la arquitectura al PIM (mediante la adaptación/extensión del mismo).

Aunque el ASM que ha sido generado por las reglas de transformación M2M definidas para la arquitectura de persistencia móvil es muy simple, y por lo tanto, la obtención de mencionado ASM podría ser percibida hasta como trivial. Considerando que de todas formas constituye un modelo ASM diferente, obtenido a partir del mismo PIM utilizado para la obtención del ASM para RIA, y que, a pesar de ser pocos, se logró automatizar la transformación de todos aquellos elementos del PIM que así lo requerían. El caso práctico presentado en este capítulo, todavía constituye un ejemplo válido a partir del cual inferir conclusiones sobre el impacto del ASM para con la portabilidad del PIM. Sin embargo, también es cierto que, a causa de esto, los resultados obtenidos solo pueden ser considerados como preliminares y que, por lo tanto, es necesario que validaciones más rigurosas los confirmen.

Por otra parte, la presente experiencia permitió comprobar que mediante la utilización de archivos de configuración externos, es posible mejorar, y en gran medida, el grado de automatización de las transformaciones M2M orientadas a transformar modelos PIM a modelos ASM. Más aún cuando estos archivos son procesados mediante la invocación de código Java nativo.

Considerando además que, ni el metamodelo ni los perfiles ASM utilizados como base para la definición del mapeo y las transformaciones M2M presentadas, fueron diseñados/creados con el objetivo de maximizar la automatización de la transformación M2M (es decir, la transformación de modelos PIM a modelos ASM), sino que pensando en la automatización de las transformaciones M2T (es decir, la transformación de modelos ASM a código), el grado de automatización alcanzado es muy alentador, sobre todo al considerar el gran potencial de mejora que esto conlleva.

También han sido detectadas algunas amenazas a la validez de la experiencia llevada a cabo y reportada en este capítulo. A continuación se las mencionan, junto con algunas sugerencias en cuanto a cómo abordarlas:

- La facilidad para obtener modelos ASM para otras arquitecturas diferentes podría deberse a que no ha sido tomada en cuenta ninguna característica distintiva de los modelos ASM. Por lo tanto, en trabajos futuros, estas características deben ser tenidas en cuenta en orden a corroborar si se mantiene la tendencia detectada en este estudio.

- Así también, el alentador grado de automatización alcanzado podría deberse a que no hay mucho poder distintivo entre los modelos ASM (lo que conduce a transformaciones triviales). Por lo tanto, para disipar estas dudas, los trabajos futuros deben abordar casos prácticos más complejos.

5.6. Síntesis del capítulo

Una vez definidas y validadas las reglas de transformación PIM-ASM, se procedió a utilizarlas en un caso práctico que, además de probarlas, permitió recabar datos para el análisis de los siguientes aspectos del ASM: i) la posibilidad de generar múltiples ASM a partir de un único PIM, y ii) el grado de portabilidad del PIM alcanzado como consecuencia de la utilización del ASM.

Para la experiencia de transformación se procedió al diseño de un sistema de marcación de empleados. Hecho que implicó la creación de todos los diagramas correspondientes al PIM (modelos de entidades, modelo navegacional, modelos de comportamiento, modelos de presentación y modelos de usuario).

La primera sección del presente capítulo presenta y explica de manera detalla cada uno de estos diagramas. Mientras la segunda sección presenta los archivos de configuración creados a partir de ellos.

Por otro lado, la tercera sección del capítulo presenta una explicación detallada de cada uno de los diagramas que sufrieron modificaciones al pasar del PIM al ASM. Entre los aspectos abordados en la explicación resaltan la justificación de cada uno de los cambios y la comparación visual de los diagramas afectados.

Finalmente, las secciones cuarta y quinta detallan y explican las conclusiones alcanzadas a partir del análisis de los resultados recabados durante y después de la ejecución del caso práctico. Sin embargo, validaciones más rigurosas deben confirmar estos resultados preliminares.

Capítulo 6

Conclusiones

Como parte de la revisión bibliográfica del presente trabajo final de carrera, fue llevado a cabo un mapeo sistemático de la literatura (SMS), con el cual se buscó obtener una visión general sobre las áreas de investigación relacionadas con MDD y MDA, especialmente en lo que respecta a propuestas y trabajos de investigación que contemplan la definición de reglas de transformación M2M para arquitecturas RIA. Interesó, sobre todo, la identificación de áreas de estudio descuidadas, trabajos inconclusos y/o pendientes y, de manera más general, cualquier oportunidad de investigación que eventualmente pueda o requiera ser abordada en el futuro.

Algunas de las principales contribuciones del SMS, para con el presente proyecto de fin de carrera, fueron: i) la identificación de la necesidad de contribuir con metodologías MDD existentes en lugar de proponer nuevas metodologías, puesto que muchas de ellas se encuentran inconclusas; ii) la necesidad de definir y utilizar reglas de transformación M2M, puesto que es frecuente que las metodologías propuestas las contemplen pero no las definan ni utilicen, optando en su lugar por las transformaciones M2T; iii) la identificación del bajo porcentaje de propuestas que cuentan con herramientas que las soporten; y iv) la existencia y la falta de abordaje por parte de la comunidad científica, del problema de portabilidad del PIM (derivado de la tendencia actual de las metodologías MDD existentes, a extender la notación del PIM con elementos propios de cada nueva arquitectura a ser modelada).

Con esto en mente, se decidió adoptar MoWebA y contribuir con ella mediante el análisis de la nueva fase que propone, la fase ASM.

Esta elección permitió el abordaje de las cuatro problemáticas señaladas en el punto anterior, ya que para el análisis de la fase ASM de MoWebA: i) fueron definidas reglas de transformación M2M que permitieron obtener modelos ASM a partir de modelos PIM, algo que, además, aún no había sido realizado para MoWebA, y ii) fue definido un proceso de desarrollo que permite aprovechar herramientas existentes para ejecutar las reglas de transformación M2M definidas, y iii) fueron obtenidos dos modelos ASM diferentes a partir de un único modelo PIM, uno de los objetivos perseguidos por el ASM de MoWebA y una importante contribución para con la resolución del problema de portabilidad del PIM.

Las arquitecturas abordadas en el análisis fueron RIA y persistencia móvil. Ambas fueron seleccionadas debido a su relevancia actual. Sin embargo, RIA fue adoptada como arquitectura principal, con ella se buscó identificar el grado máximo de auto-

matización posible y se buscó cubrir el ciclo de desarrollo completo, desde el PIM hasta el código. La persistencia móvil, por otro lado, fue abordada únicamente con el objetivo de verificar si es posible o no obtener varios modelos ASM a partir de un único modelo PIM y, posteriormente, analizar las implicancias de esto.

Lo anterior es muy relevante para entender los resultados obtenidos. Por un lado, al analizar los resultados para RIA se ha llegado a las siguientes conclusiones:

- La experiencia realizada fue muy positiva debido a que fue posible transformar todas las clases que así lo requerían. Además, el 100 % de clases del ASM fueron derivadas automáticamente a partir de las reglas de transformación M2M definidas en la sección 4.5, incluyendo la creación de nuevas clases para los servicios asíncronos detectados.
- Además, gracias a la utilización de los archivos de configuración externos, el modelo ASM generado para la arquitectura RIA no requirió de ningún ajuste adicional posterior.
- El modelo ASM para RIA obtenido de manera automática a partir de la ejecución de las reglas de transformación M2M definidas, resulto ser idéntico al ASM definido y utilizado para la obtención de código (a través de transformaciones M2T) en el estudio llevado a cabo por Guido Nuñez et al. [22]. Por lo tanto, los resultados obtenidos en nuestra experiencia, en términos de la cantidad de código generado y los ajustes manuales realizados al código luego de la transformación M2T, son idénticos a los obtenidos en el mencionado estudio. Más aún, el código final que ha sido obtenido con las herramientas de generación automática cumple con los requerimientos iniciales del sistema [24].

Por otro lado, al analizar los resultados para persistencia móvil en combinación con los resultados obtenidos para RIA, se ha llegado a las siguientes conclusiones:

- Gracias a la experiencia realizada ha sido posible verificar que las transformaciones M2M permiten la obtención de modelos ASM para diferentes arquitecturas a partir de un único modelo PIM, sin necesidad de modificarlo, extenderlo o adaptarlo.
- Si bien en la experiencia se han considerado solo dos arquitecturas, y en consecuencia, solo se han obtenido dos modelos ASM diferentes (uno orientado a modelar arquitecturas RIA y otro orientado a modelar elementos arquitectónicos propios de la persistencia móvil), se intuye posible obtener/generar modelos ASM para otras arquitecturas diferentes.
- La principal ventaja de obtener varios modelos ASM partiendo de un único modelo PIM es que la portabilidad del PIM se conserva de manera considerable y significativa. Es importante recordar que este enfoque (es decir, la captura de los elementos arquitectónicos en el modelo ASM), es muy diferente al de la mayoría de las metodologías actuales que, a diferencia de MoWebA, tienden a agregar los elementos específicos de la arquitectura al PIM (mediante la adaptación/extensión del mismo).

- Si bien el ASM obtenido para persistencia móvil es muy sencillo, y su obtención resultó hasta trivial. Considerando que de todas formas constituye un ASM diferente, obtenido a partir del mismo PIM utilizado para RIA, y que, a pesar de ser pocos, se logró automatizar la transformación de todos aquellos elementos del PIM que así lo requerían. Todavía constituye un ejemplo válido a partir del cual inferir conclusiones sobre el impacto del ASM para con la portabilidad del PIM. Aunque a causa de esto, los resultados obtenidos se consideran preliminares y se requiere que validaciones más rigurosas los confirmen.

6.1. Principales contribuciones

- *El Mapeo Sistemático de la Literatura (SMS).*

  El SMS presentado en el capítulo 3 constituye un importante aporte al campo, puesto que: i) como fue demostrado en el SMS mismo, no se detectó ningún otro SMS que aborde MDD y RIA en conjunto, y ii) el SMS en sí, al ser un estudio definido de manera formal y ejecutado de manera sistemática y metódica, constituye un estudio muy valorado por la comunidad científica.

- *La contribución con MoWebA en lugar de proponer una nueva metodología.*

  En consonancia con una de las conclusiones del SMS, concretamente la que señala la gran cantidad de propuestas inconclusas e identifica en esto una importante oportunidad de investigación futura (la necesidad de contribuir con metodologías existentes en lugar proponer nuevas), en este trabajo se decidió adoptar MoWebA y contribuir con ella. Ver apartado 3.5.

- *La unificación de los Perfiles UML orientados a RIA y persistencia móvil.*

  Si bien los Perfiles UML de MoWebA para RIA y persistencia móvil fueron creados en trabajos previos a este, la unificación de los mismos para su posterior utilización en la definición de las reglas de transformación M2M requeridas, constituye un importante aporte de este trabajo de tesis. Ver apartado 4.3.

- *Las reglas de mapeo y las reglas de transformación M2M.*

  En vista de que MoWebA aún no disponía de mapeos y reglas de transformación M2M que posibiliten la obtención de modelos ASM a partir de modelos PIM para arquitectura alguna, el hecho de dotarla de esta capacidad constituye un importante aporte para con ella. Ver apartados 4.4 y 4.5.

- *Los archivos de configuración externos.*

  Los archivos de configuración externos también constituyen un aporte del presente trabajo de tesis. El valor de los mismos radica en que conceden cierto nivel de injerencia sobre las transformaciones M2M, al permitir i) personalizar las transformaciones M2M definidas y ii) adaptarlas a las particularidades del sistema modelado. Ver apartado 4.6.

- *El proceso de desarrollo propuesto.*

  Como con MoWebA aún no se había intentado obtener modelos ASM de manera automática, el proceso para hacerlo seguía siendo una incógnita. Este trabajo responde a dicha incógnita presentando un conjunto de herramientas a utilizar y una serie de pasos a seguir para obtener modelos ASM de manera automática. Ver apartado 4.2.

- *El aporte para con el ASM.*

  Este aporte refiere al hecho de que, mediante el trabajo llevado a cabo en el marco de la presente tesis, trabajo consolidado en el ejemplo práctico del capítulo 5, se ha podido demostrar que, efectivamente, la idea detrás del modelo ASM propuesto por MoWebA, es factible. Es decir, con este trabajo se a probado que es posible obtener varios y diferentes modelos ASM partiendo de un único modelo PIM. Ver capítulo 5.

- *El aporte para con el problema de portabilidad del PIM.*

  Este aporte está intrínsecamente relacionado con el aporte anterior. Refiere al hecho de que, al haberse demostrado que es posible obtener varios modelos ASM a partir de un único modelo PIM, entonces es muy probable que parte del problema de portabilidad del PIM esté resulto. Ver capítulo 5.

- *La publicación de artículos científicos.*

  Fruto del trabajo llevado a cabo en el marco de la presente tesis, fueron realizadas algunas publicaciones científicas en congresos y revistas. Ver apartado 1.4.

6.2. Resultados vs Objetivos

Objetivo General

- *En el marco del enfoque MoWebA, el objetivo general del presente trabajo de tesis se centra en analizar algunos aspectos relevantes del nuevo nivel de abstracción planteado por esta propuesta, el modelo ASM.*

  Se demostró que es posible obtener varios modelos ASM a partir de un único modelo PIM, y además, se determinó que, efectivamente, el ASM contribuye a la portabilidad del PIM en la medida de lo esperado.

Objetivos Específicos

- *Identificar una(s) arquitectura(s) relevante(s) para la especialización del ASM.*

  Se seleccionaron las arquitecturas RIA y de persistencia móvil, sin embargo, la de persistencia móvil solo se abordó luego de identificar los aspectos del ASM que serían analizados, ya que se observó que para el análisis de los mismos se requería del abordaje de al menos dos arquitecturas.

- *Definir reglas de transformación que permitan transformar modelos PIM a modelos ASM para la(s) arquitectura(s) seleccionada(s).*

  Estas fueron definidas de manera exitosa y se presentan en las secciones 4.4 y 4.5 respectivamente. Al momento de definir las reglas para persistencia móvil no se contempló la creación de nuevas clases puesto que esto resultó no ser necesario para los objetivos perseguidos con la validación realizada.

- *Validar el proceso de transformación PIM-ASM, garantizando que las reglas definidas se ajusten adecuadamente tanto al enfoque MoWebA como a la(s) arquitectura(s) seleccionada(s).*

  La validación del proceso de transformación PIM-ASM fue realizada por los profesores guía mediante la puesta en práctica de una validación basada en el juicio de expertos. Los aspectos validados fueron, en concreto, los mapeos detallados en la sección 4.4 y las reglas de transformación presentadas en la sección 4.5.

- *Identificar un(unos) aspecto(s) relevante(s) del ASM para el análisis de este nuevo nivel de abstracción.*

  En base las conclusiones obtenidas en el SMS, sobre todo las relacionadas al problema de portabilidad del PIM, los aspectos del ASM seleccionados como objeto de análisis fueron los siguientes: i) la posibilidad de generar múltiples ASM a partir de un único PIM, y ii) el grado de portabilidad del PIM alcanzado como consecuencia de la utilización del ASM.

- *Definir y llevar adelante experiencias que aporten datos relevantes sobre el(los) aspecto(s) del ASM a ser analizado(s).*

  La experiencia realizada consistió en la ejecución de un caso práctico en el que se buscó: i) validar las reglas de transformación M2M definidas en la sección 4.5, ii) determinar si es posible o no obtener modelos ASM diferentes partiendo de un único modelo PIM, y iii) determinar si el ASM contribuye a la portabilidad del PIM en la medida de lo esperado.

6.3. Trabajos futuros

En cuanto a las posibilidades de trabajo futuro identificadas a lo largo del presente proyecto final de carrera, estas fueron detectadas en dos etapas clave del desarrollo: la etapa de revisión bibliográfica y la etapa de validación.

Las posibilidades de trabajo futuro detectadas en la etapa de revisión bibliográfica se corresponden con las posibilidades de mejora que surgieron durante la ejecución del protocolo del SMS y/o al finalizarlo, junto con aquellas señaladas por los revisores de las publicaciones relacionadas derivadas de este trabajo (ver sección 1.4).

A continuación se listan las posibilidades de trabajo futuro que fueron detectadas en la etapa de revisión:

- Llevar a cabo una SLR que cubra los mismos temas que el SMS realizado, de manera a excluir todos los estudios no validados y así obtener una mirada más rigurosa sobre el tema.
- Llevar a cabo experimentos que permitan validar las propuestas MDD existentes y que aún no hayan sido validadas con ese nivel de rigurosidad.
- Llevar a cabo un SMS que permita detectar y analizar todas las herramientas MDD existentes y averiguar si son capaces de adaptarse a RIA.
- Interactuar con la industria con el fin de desarrollar más y mejores herramientas para MDD y RIA y/o para aumentar la capacidad de las herramientas existentes.
- Investigar cuales son los lenguajes de modelado de arquitecturas RIA utilizados en las propuestas MDD orientadas a RIA.

Por otro lado, las posibilidades de trabajo futuro detectadas en la etapa de validación se corresponden con las posibilidades de mejora que surgieron durante la ejecución del caso práctico llevado a cabo con objeto de: i) validar las reglas de transformación M2M definidas en la sección 4.5, ii) comprobar la posibilidad de generar múltiples ASM a partir de un único PIM, y finalmente, iii) comprobar si el ASM contribuye a la portabilidad del PIM en la medida de lo esperado.

A continuación se listan las posibilidades de trabajo futuro que fueron detectadas en la etapa de validación:

- Realizar una validación más formal, un primer paso podría ser un experimento con alumnos y luego un caso de estudio en la industria.
- Mejorar el porcentaje de automatización alcanzado para la arquitectura de persistencia móvil, específicamente en lo referente a la automatización de la creación de clases.
- Aumentar el número de arquitecturas soportado, de manera a posibilitar experiencias futuras de mayor complejidad.
- Analizar la calidad de los modelos definidos y generados, así como también la del software resultante en general.
- Aumentar progresivamente la complejidad de los sistemas modelados, abordando cada vez casos prácticos más complejos.

Apéndice A

Metamodelo de MoWebA

A.1. Visión global del metamodelo de MoWebA

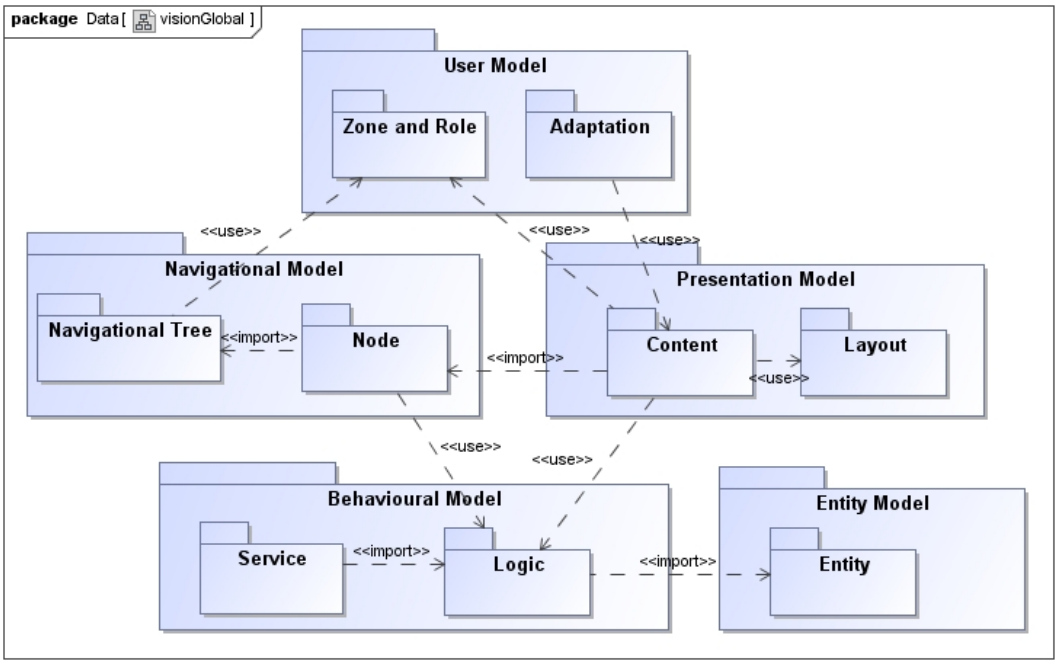


Figura A.1: Visión global del metamodelo de MoWebA

A.2. Metamodelo de MoWebA

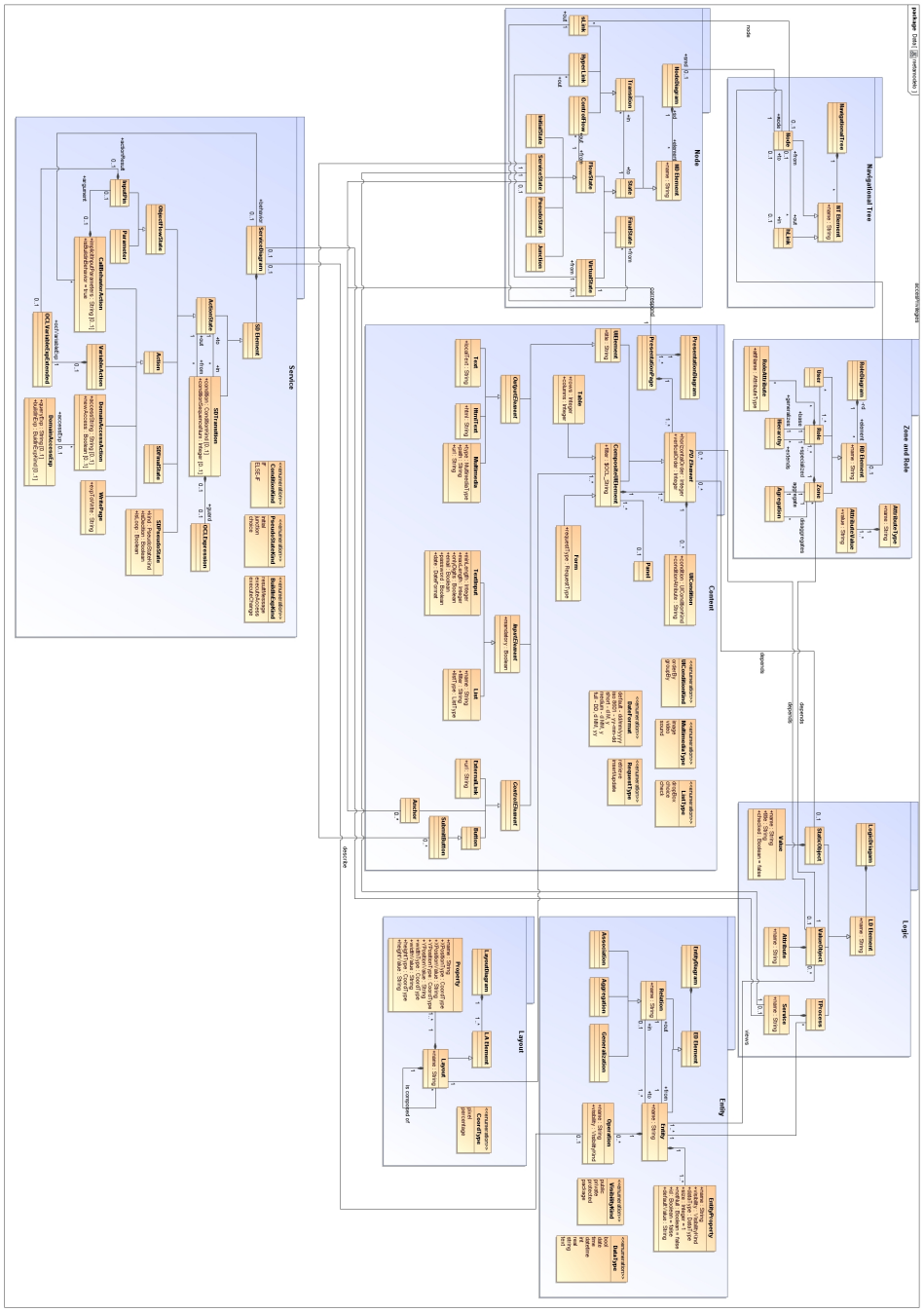


Figura A.2: Metamodelo de MoWebA

Apéndice B

Perfiles de MoWebA

B.1. Perfil de Entidad

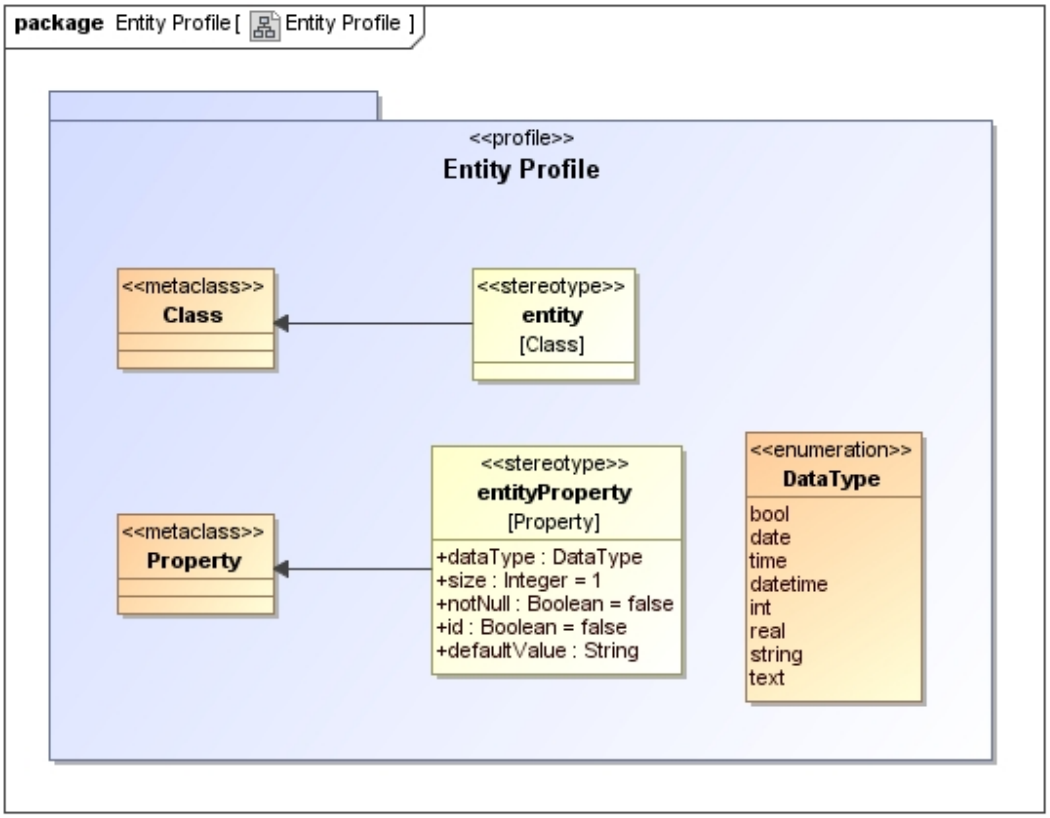


Figura B.1: Perfiles de MoWeba – Entidades

B.2. Perfil de Contenido

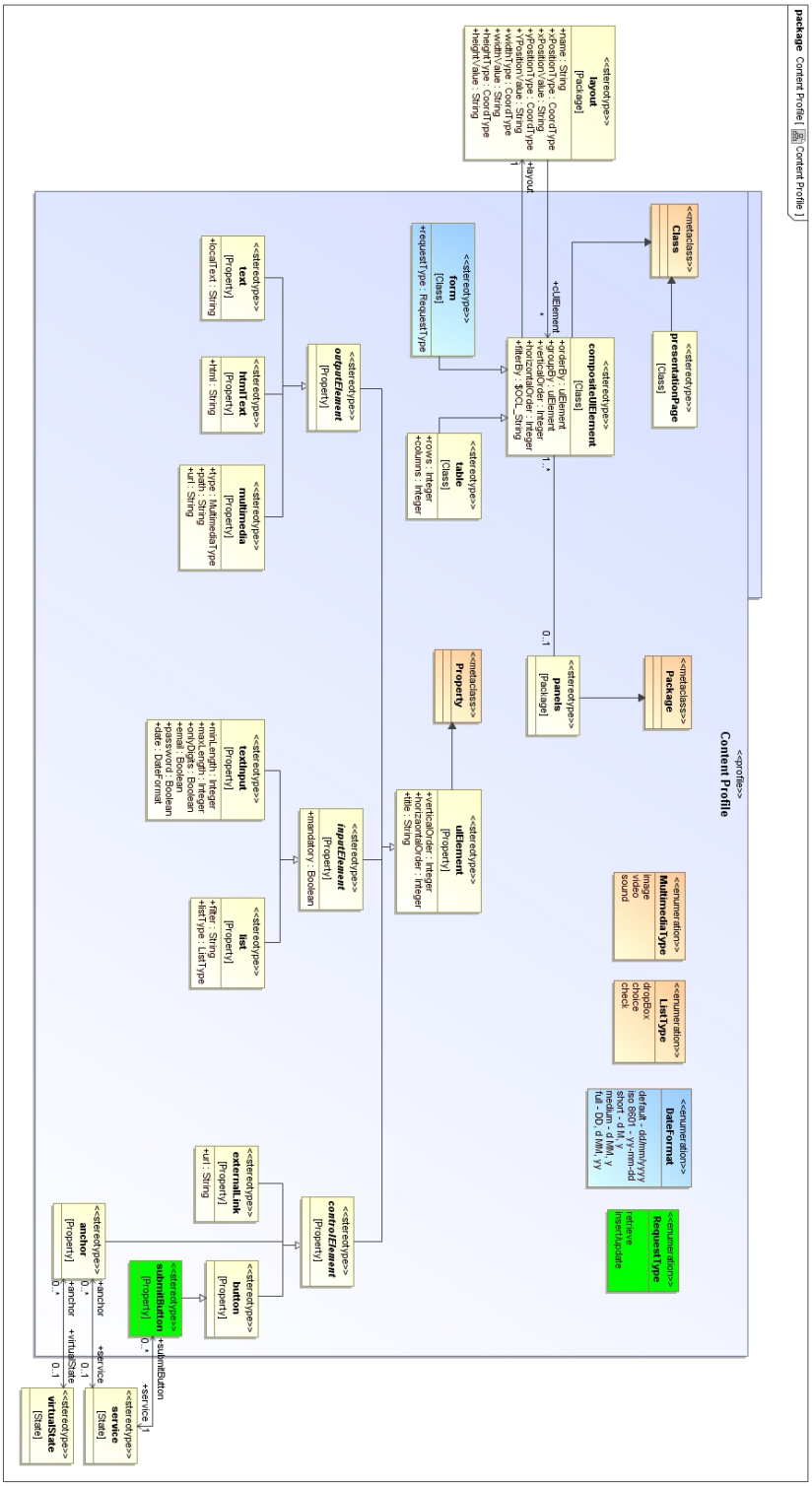


Figura B.2: Perfiles de MoWeba – Contenido

B.3. Perfil Lógico

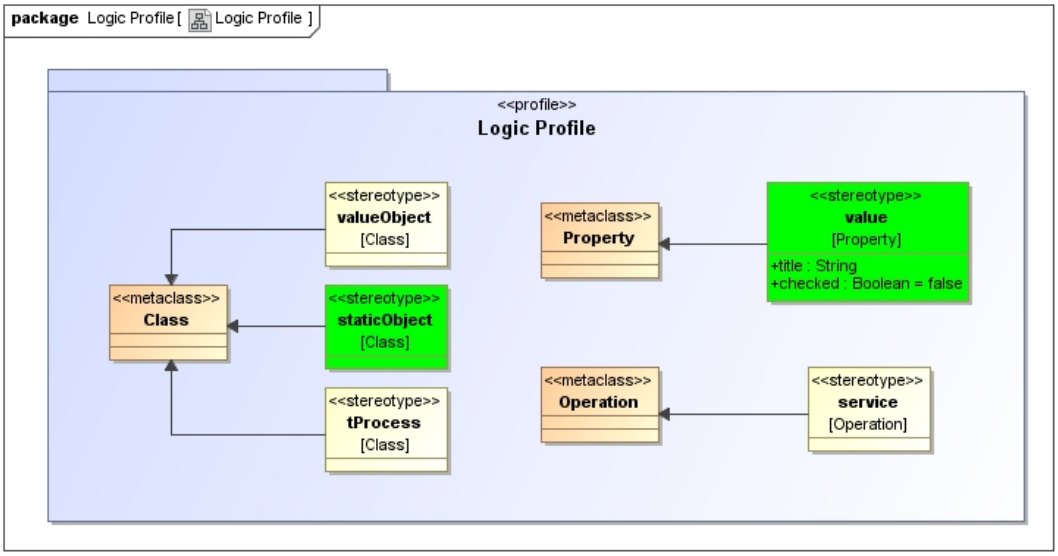


Figura B.3: Perfiles de MoWeba – Lógico

B.4. Perfil de Árbol Navegacional

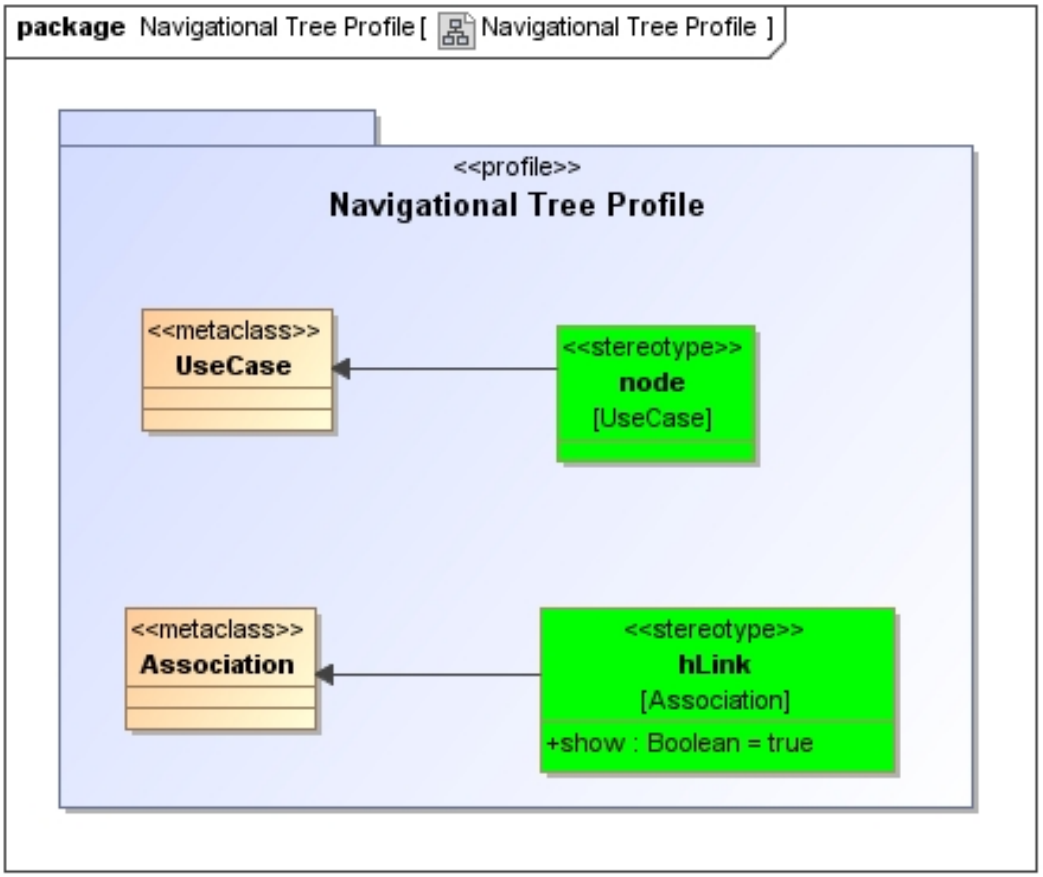


Figura B.4: Perfiles de MoWeba – Árbol navegacional

B.5. Perfil de Nodos

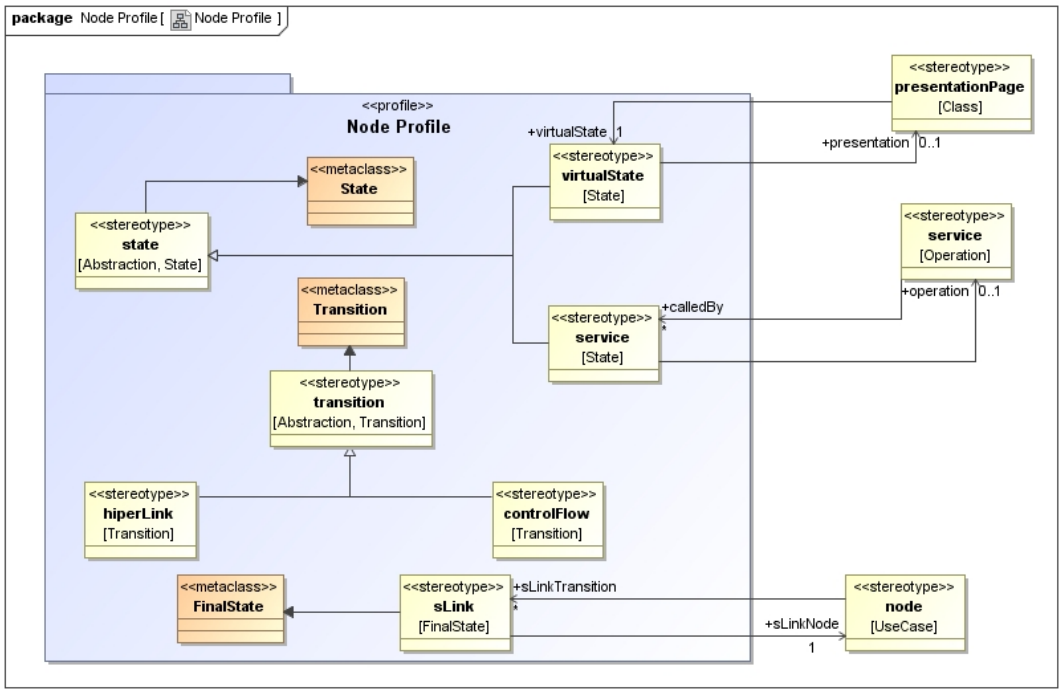


Figura B.5: Perfiles de MoWeba – Nodos

B.6. Perfil de Roles

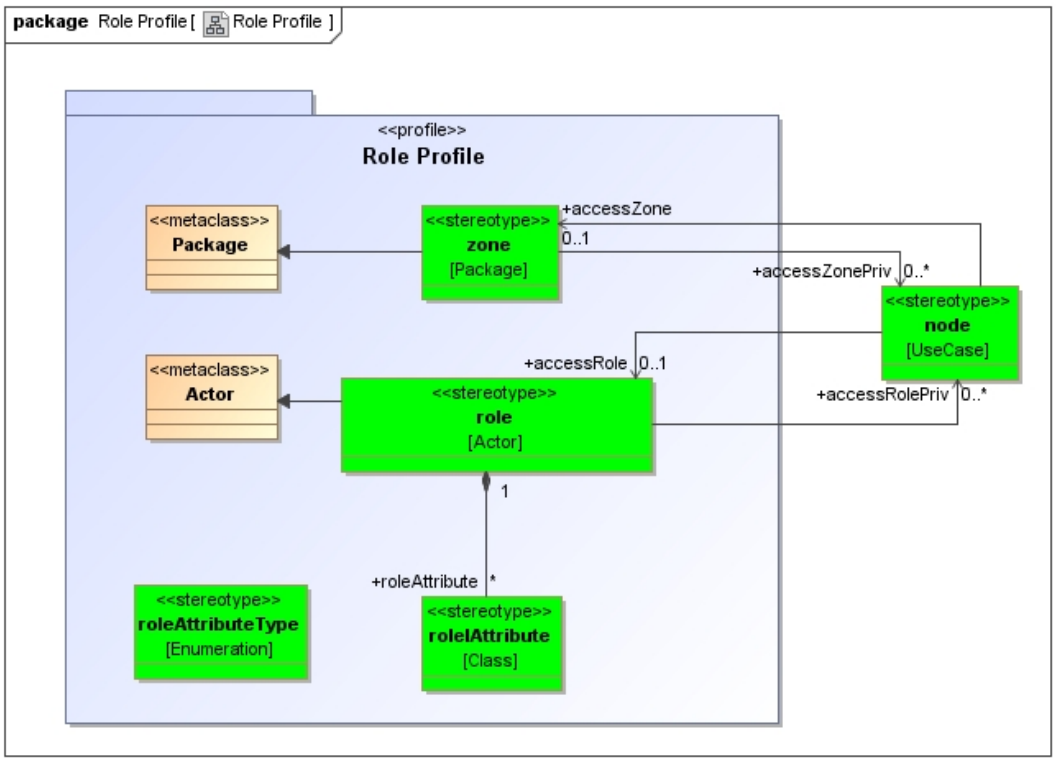


Figura B.6: Perfiles de MoWeba – Roles

B.7. Perfil de Servicios

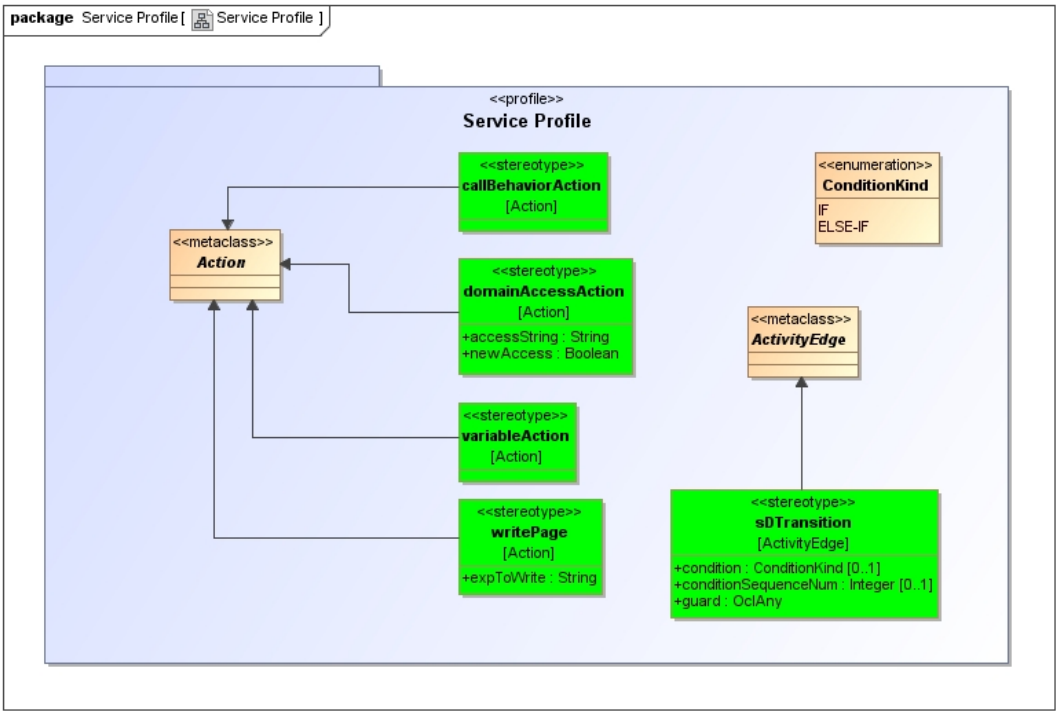


Figura B.7: Perfiles de MoWeba – Servicios

B.8. Perfil de Estructura

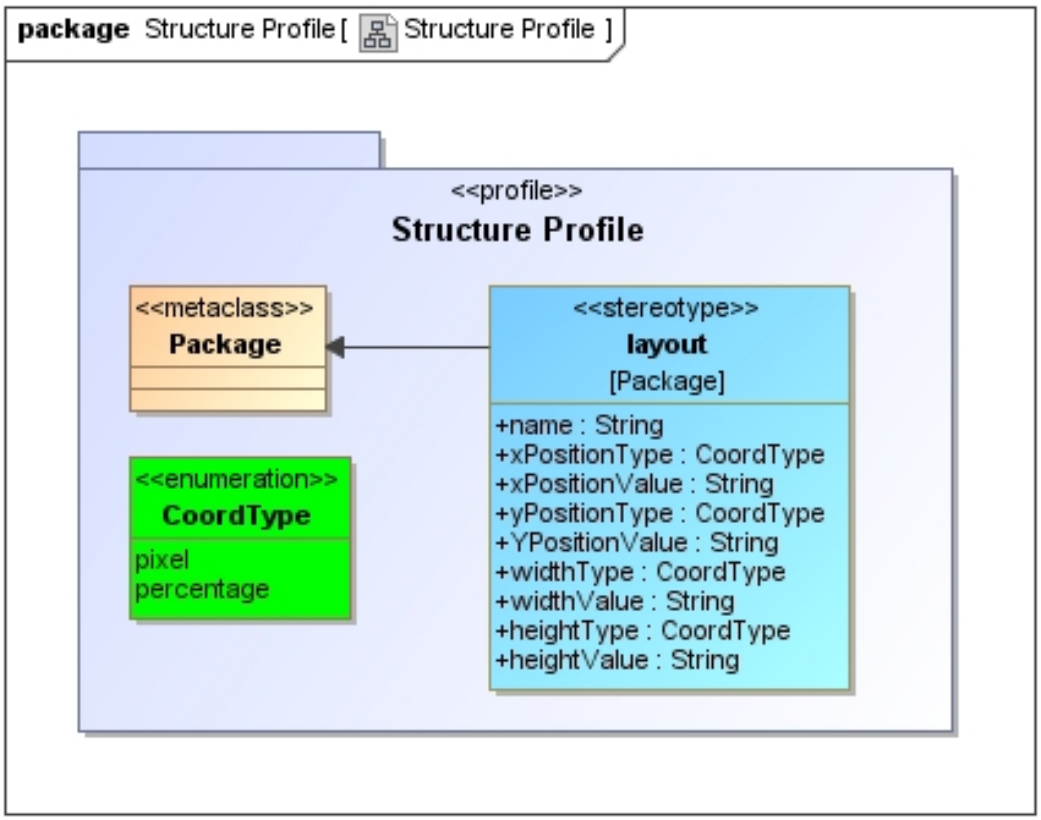


Figura B.8: Perfiles de MoWeba – Estructura

Apéndice C

Código Java para el procesamiento de los Archivos de Configuración

C.1. Configuración del IDE

Para poder llamar al código Java desde las reglas de transformación M2M fue necesario agregar, al classpath del IDE utilizado (ver apartado 4.2), los ejecutables generados a partir de ellos.

El IDE utilizado para la ejecución de las transformaciones M2M, contempla una manera de agregar ejecutables Java a su classpath. La estrategia definida requiere del agregado de una línea, con un comando especial, al archivo eclipse.ini ubicado en la misma carpeta que el ejecutable que permite lanzarlo. La línea en cuestión es la siguiente: `-Xbootclasspath/a:workspace\...\.jar;workspace\...\.jar`

C.2. Estructura del código

Para el desarrollo del procesador de los archivos de configuración se utilizó el patrón de diseño denominado "Fachada" o "Facade", el cual es un tipo de patrón de diseño estructural que viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre estos [87].

Los participantes definidos por este patrón de diseño estructural son los siguientes:

1. **Fachada** (Facade): conoce qué clases del subsistema son responsables de una determinada petición, y delega esas peticiones de los clientes a los objetos apropiados del subsistema [87].

2. **Subclases** (Modulo A, Modulo B): implementan la funcionalidad del subsistema. Realizan el trabajo solicitado por la fachada. No conocen la existencia de la fachada [87].

C.3. Fachada

C.3.1. ConfM2M.java

```
package atl.conf;

import org.yaml.snakeyaml.Yaml;

import java.io.FileNotFoundException;
import java.util.List;
import java.util.Map;

import atl.conf.creacion.CreArchConf;
import atl.conf.transformacion.TraArchConf;

public class ConfM2M {

    private TraArchConf conf_tra = new TraArchConf();
    private CreArchConf conf_cre = new CreArchConf();
    private Boolean seCargo_TraArchConf = true;
    private Boolean seCargo_CreArchConf = true;

    public ConfM2M() {
        Yaml yaml = new Yaml();
        try {
            this.conf_tra = yaml.loadAs(TraArchConf.getInputStream(),
                TraArchConf.class);
            if (this.conf_tra == null) this.conf_tra = new TraArchConf
                (); //por si el archivo esta vacio

        } catch (FileNotFoundException e) {
            this.seCargo_TraArchConf = false;
        }
        try {
            this.conf_cre = yaml.loadAs(CreArchConf.getInputStream(),
                CreArchConf.class);
            if (this.conf_cre == null) this.conf_cre = new CreArchConf
                (); //por si el archivo esta vacio

        } catch (FileNotFoundException e) {
            this.seCargo_CreArchConf = false;
        }
    }

    public boolean aplicarEstereotipo(String contenedor, String
        elemento) {
        if (this.seCargo_TraArchConf) return this.conf_tra.
            aplicarEstereotipo(contenedor, elemento);
        return true;
    }

    public boolean aplicarEstereotipo(String paquete, String contenedor
        , String elemento) {
        if (this.seCargo_TraArchConf) return this.conf_tra.
            aplicarEstereotipo(paquete, contenedor, elemento);
        return true;
    }
```

```
    public Map<String,List<String>> getPropiedades(Map<Integer,String>
        clasesPIM,Map<Integer,String> atributosPIM) {
        if (this.seCargo_CreArchConf) return this.conf_cre.
            getPropiedades(clasesPIM,atributosPIM);
        return this.conf_cre.iniPropiedades(clasesPIM,atributosPIM);
    }

    public Map<String,Map<String,String>> getEstereotipoDePropiedades(
        Map<Integer,String> clasesPIM,Map<Integer,String> atributosPIM)
         {
        if (this.seCargo_CreArchConf) return this.conf_cre.
            getEstereotipoDePropiedades(clasesPIM,atributosPIM);
        return this.conf_cre.iniEstereotipoDePropiedades(clasesPIM,
            atributosPIM);
    }

    protected String getClassPath() {
        return System.getProperty("java.class.path");
    }

}
```

C.4. Módulo A: (creación)

C.4.1. Clase.java

```
package atl.conf.creacion;

import java.util.List;

public class Clase {

    public String clasePIM;
    public String atributoPIM;
    public List<Propiedad> propiedades;

}
```

C.4.2. Propiedad.java

```
package atl.conf.creacion;

public class Propiedad {

    public String nombre;
    public String estereotipo;

}
```

C.4.3. CreArchConf.java

```
package atl.conf.creacion;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CreArchConf {

    public List<Clase> clases;

    public CreArchConf() {
        this.clases = new ArrayList<Clase>();
    }

    public Map<String,List<String>> iniPropiedades(Map<Integer,String>
        clasesPIM,Map<Integer,String> atributosPIM)
    {
```

```
        Map<String,List<String>> r = new HashMap<String,List<String>>()
            ;

        if (clasesPIM.size() == atributosPIM.size()) {

            Integer bandera = clasesPIM.size();

            for (int i = 1; i <= bandera; i++) {
                String cPIM = clasesPIM.get(i);
                String aPIM = atributosPIM.get(i);
                String key = cPIM.concat("#").concat(aPIM);
                List<String> propiedades = new ArrayList<String>();
                r.put(key, propiedades);
            }

        }

        return r;

    }

    public Map<String,List<String>> getPropiedades(Map<Integer,String>
        clasesPIM,Map<Integer,String> atributosPIM)
    {
        Map<String,List<String>> r = this.iniPropiedades(clasesPIM,
            atributosPIM);
        if (clasesPIM.size() == atributosPIM.size()) {
            for (int i = 1; i <= this.clases.size(); i++) {
                String cPIM = this.clases.get(i-1).clasePIM;
                String aPIM = this.clases.get(i-1).atributoPIM;
                String key = cPIM.concat("#").concat(aPIM);
                List<String> propiedades = new ArrayList<String>();
                for (int j = 1; j<= this.clases.get(i-1).propiedades.
                    size(); j++)
                    propiedades.add(this.clases.get(i-1).propiedades.
                        get(j-1).nombre);
                r.put(key,propiedades);
            }
        }
        return r;
    }

    public Map<String,Map<String,String>> iniEstereotipoDePropiedades(
        Map<Integer,String> clasesPIM,Map<Integer,String> atributosPIM)
    {
        Map<String,Map<String,String>> r = new HashMap<String,Map<
            String,String>>();
        if (clasesPIM.size() == atributosPIM.size()) {
            Integer bandera = clasesPIM.size();
            for (int i = 1; i <= bandera; i++) {
                String cPIM = clasesPIM.get(i);
                String aPIM = atributosPIM.get(i);
                String key = cPIM.concat("#").concat(aPIM);
                Map<String,String> estereotipos = new HashMap<String,
                    String>();
                r.put(key, estereotipos);
            }
```

```
        }
        return r;
    }

    public Map<String,Map<String,String>> getEstereotipoDePropiedades(
        Map<Integer,String> clasesPIM,Map<Integer,String> atributosPIM)
    {
        Map<String,Map<String,String>> r = new HashMap<String,Map<
            String,String>>();
        if (clasesPIM.size() == atributosPIM.size()) {
            for (int i = 1; i <= this.clases.size(); i++) {
                String cPIM = this.clases.get(i-1).clasePIM;
                String aPIM = this.clases.get(i-1).atributoPIM;
                String key = cPIM.concat("#").concat(aPIM);
                Map<String,String> estereotipos = new HashMap<String,
                    String>();
                for (int j = 1; j<= this.clases.get(i-1).propiedades.
                    size(); j++)
                    estereotipos.put(this.clases.get(i-1).propiedades.
                        get(j-1).nombre,this.clases.get(i-1).
                        propiedades.get(j-1).estereotipo);
                r.put(key,estereotipos);
            }
        }
        return r;
    }

    public static InputStream getInputStream() throws
        FileNotFoundException {
        InputStream input = null;
        String path_to_file = "workspace/TesisImpl_v1/complementos/
            ArchConfAsynchronousCall.yaml";
        input = new FileInputStream(path_to_file);
        return input;
    }

}
```

C.5. Módulo B: (transformación)

C.5.1. Clase.java

```
package atl.conf.transformacion;

public class Clase {

    public String paquete;
    public String clase;

}
```

C.5.2. Propiedad.java

```
package atl.conf.transformacion;

public class Propiedad {

    public String paquete;
    public String clase;
    public String propiedad;

}
```

C.5.3. TraArchConf.java

```
package atl.conf.transformacion;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class TraArchConf {

    public Integer mod;
    public List<Clase> clases;
    public List<Propiedad> propiedades;

    public TraArchConf(){
        this.mod = 1;
        this.clases = new ArrayList<Clase>();
        this.propiedades = new ArrayList<Propiedad>();
    }

    private boolean contains(String contenedor, String elemento) {
        for (Clase cl : this.clases) {
            if (cl.paquete.equals(contenedor) &&
                cl.clase.equals(elemento)) return true;
        }
```

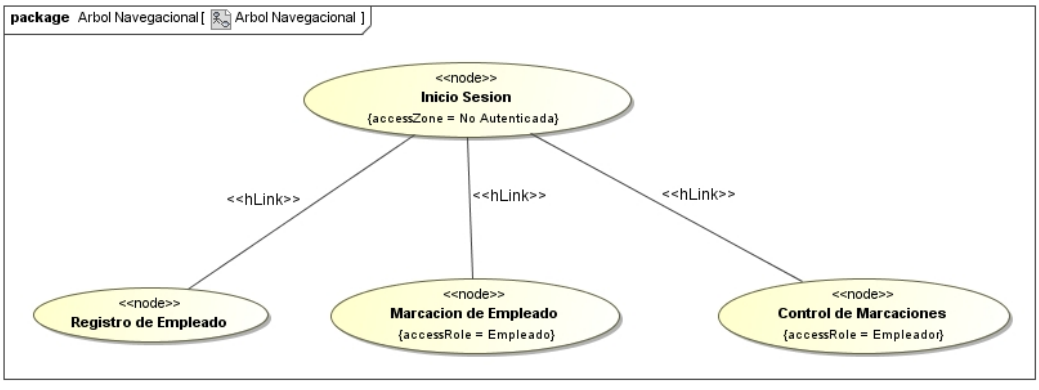
```
        return false;
    }

    public boolean aplicarEstereotipo(String contenedor, String
        elemento) {
        if (mod == 1) return this.contains(contenedor, elemento);
        if (mod == 2) return !this.contains(contenedor, elemento);
        return false;
    }

    private boolean contains(String paquete, String contenedor, String
        elemento) {
        for (Propiedad pr : this.propiedades) {
            if (pr.paquete.equals(paquete) &&
                pr.clase.equals(contenedor) &&
                pr.propiedad.equals(elemento)) return true;
        }
        return false;
    }

    public boolean aplicarEstereotipo(String paquete, String contenedor
        , String elemento) {
        if (mod == 1) return this.contains(paquete, contenedor,
            elemento);
        if (mod == 2) return !this.contains(paquete, contenedor,
            elemento);
        return false;
    }

    public static InputStream getInputStream() throws
        FileNotFoundException {
        InputStream input = null;
        String path_to_file = "workspace/TesisImpl_v1/complementos/
            ArchConfTransformacion.yaml";
        input = new FileInputStream(path_to_file);
        return input;
    }

}
```

Apéndice D

ASM generado para RIA (completo)

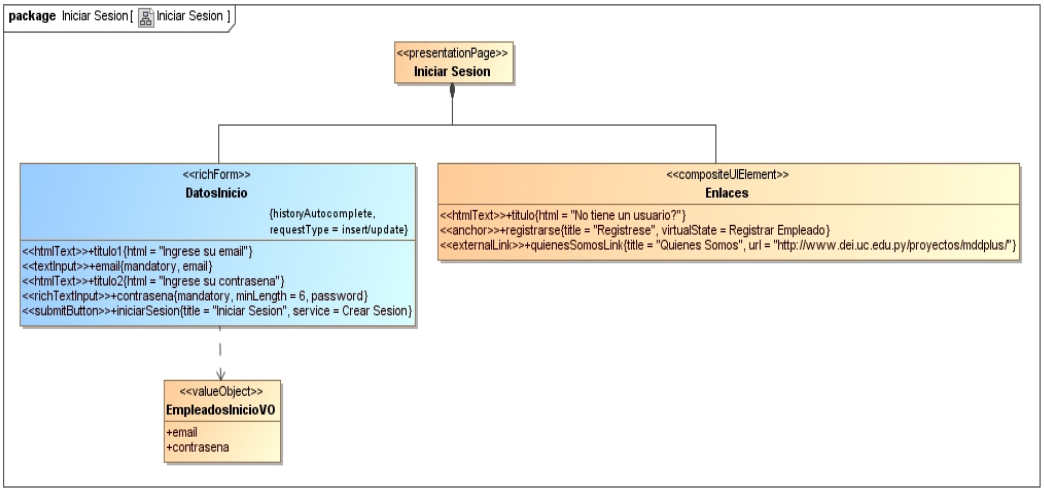
A continuación se detallan los diagramas resultantes de la aplicación de las reglas de transformación M2M definidas para la obtención del ASM orientado a arquitecturas RIA. Todos estos diagramas fueron obtenidos de manera automática a partir del PIM presentado en la sección 5.1.

D.1. Árbol Navegacional

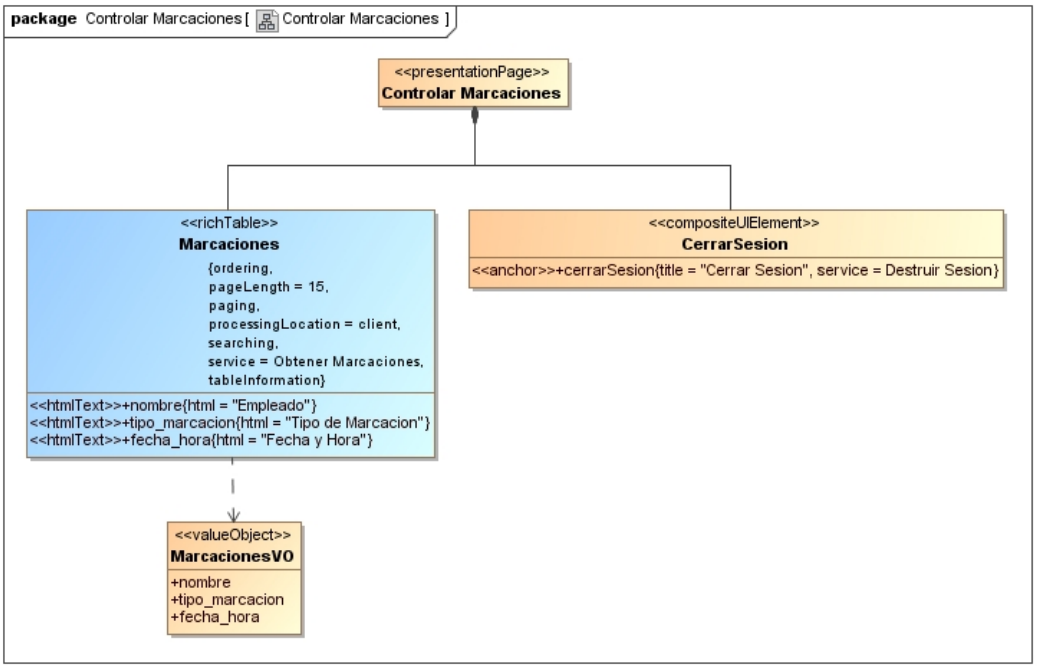


D.2. Diagramas de Contenido

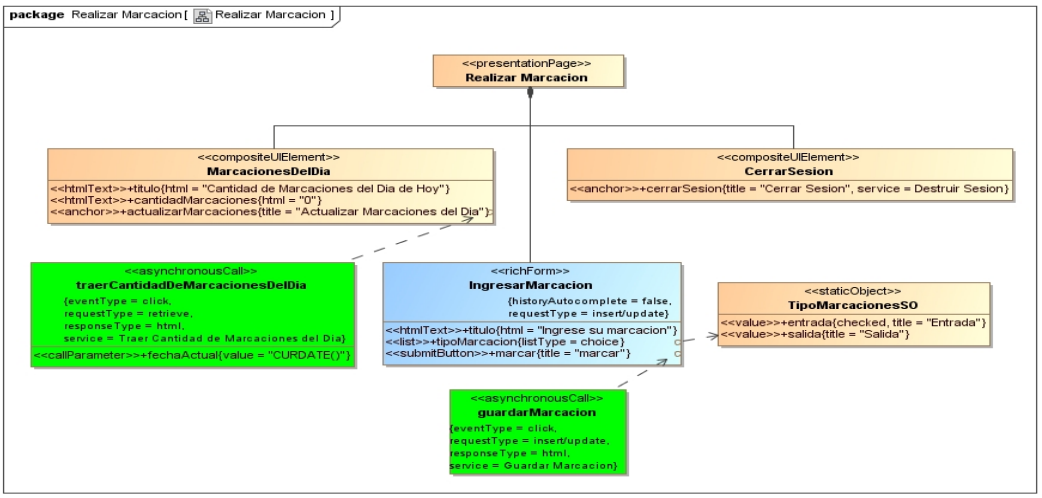
D.2.1. Iniciar Sesión



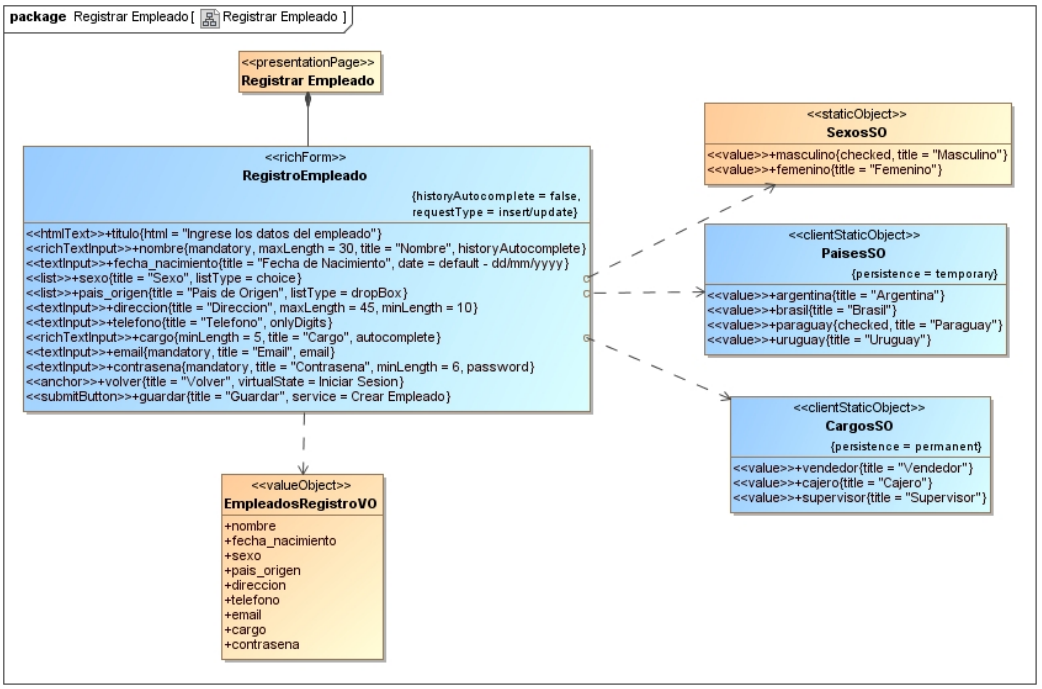
D.2.2. Controlar Marcaciones



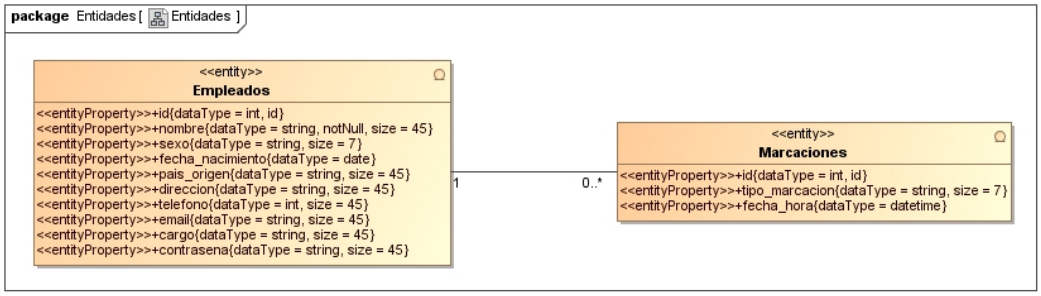
D.2.3. Realizar Marcación



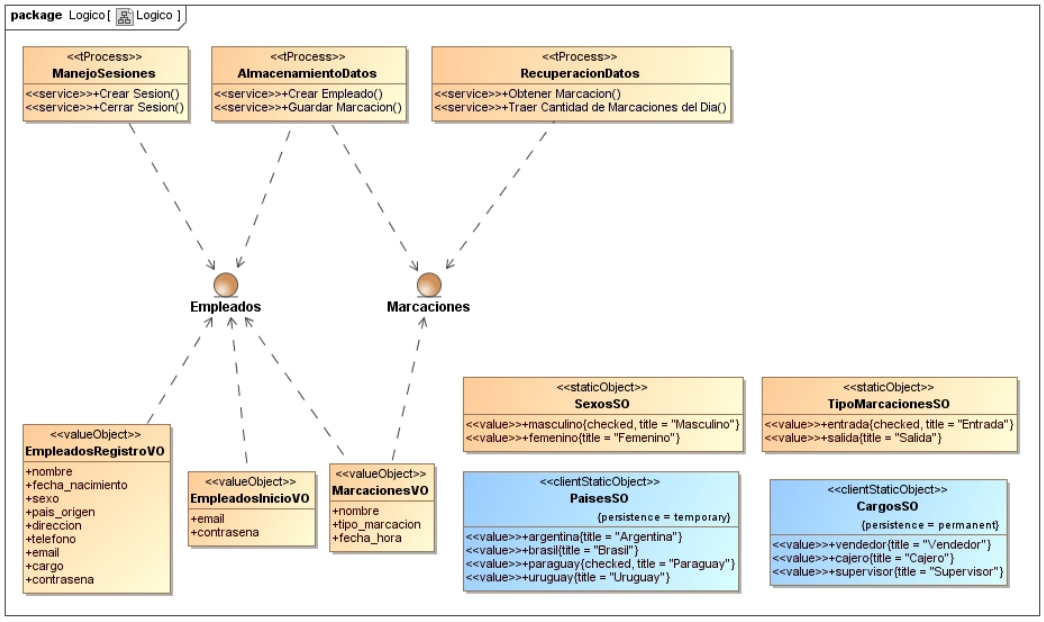
D.2.4. Registrar Empleado



D.3. Diagrama de Entidades

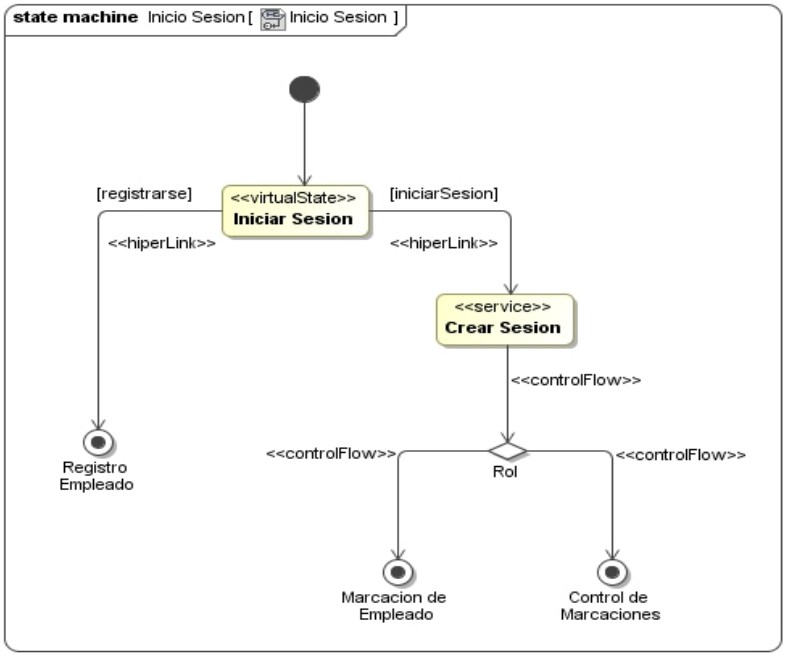


D.4. Diagrama Lógico

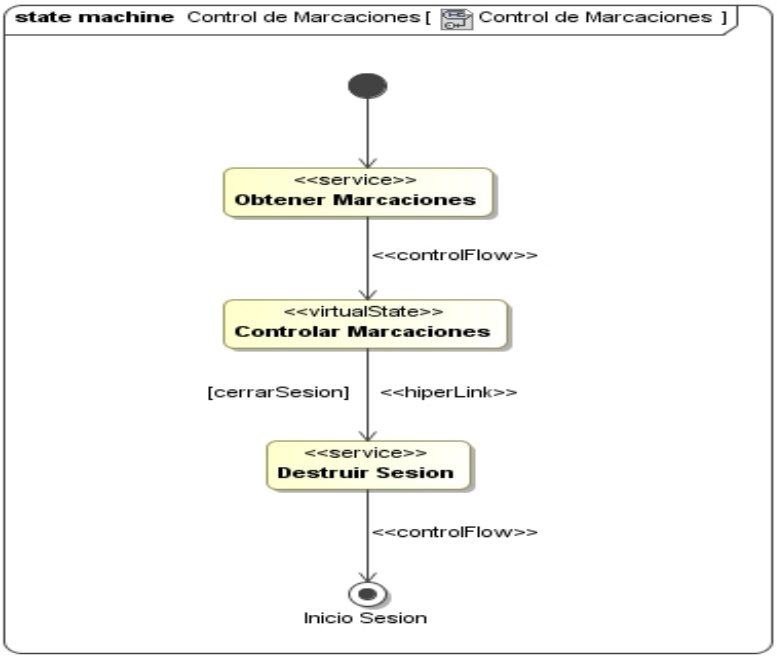


D.5. Diagramas de Nodos

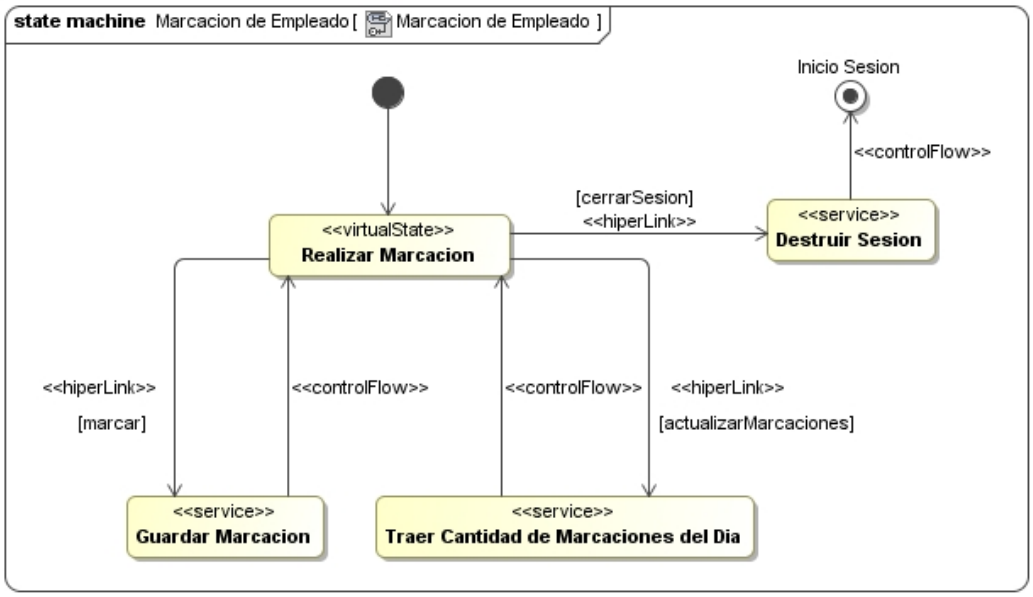
D.5.1. Inicio Sesión



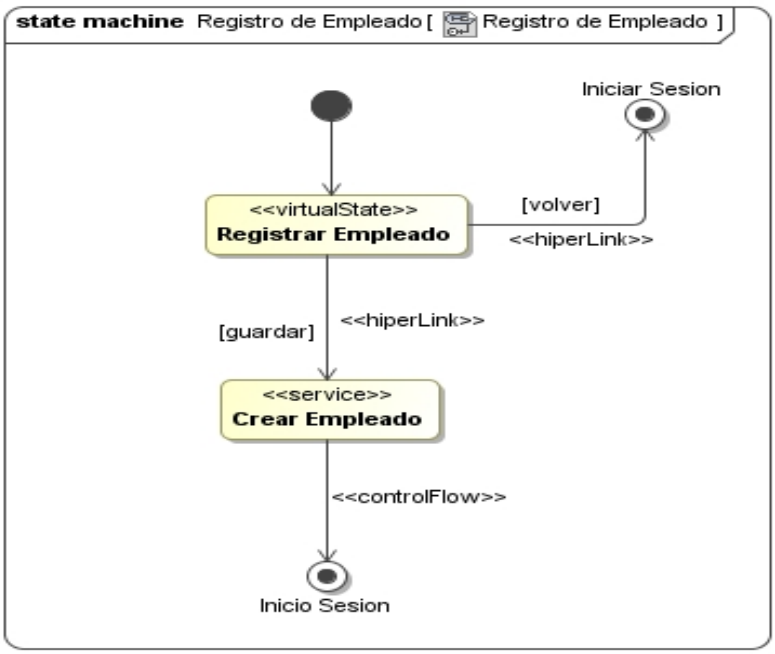
D.5.2. Control de Marcaciones



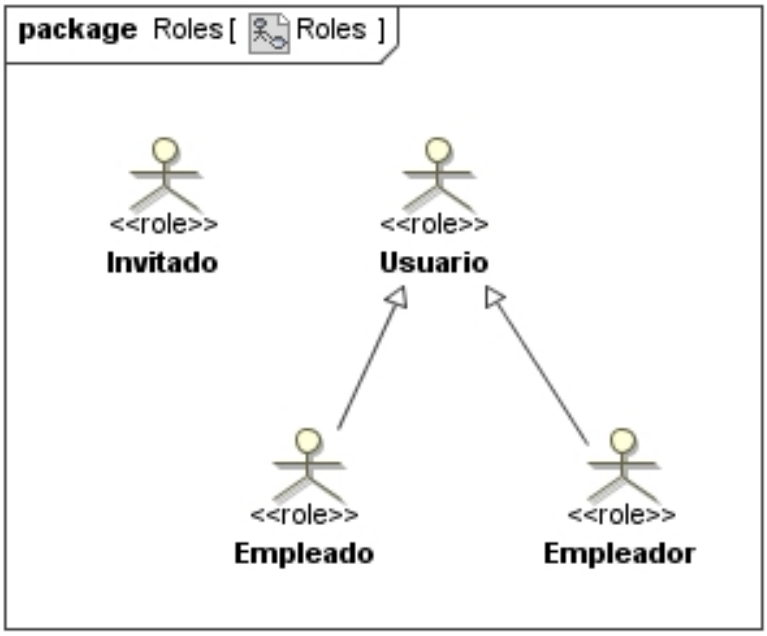
D.5.3. Marcación de Empleado



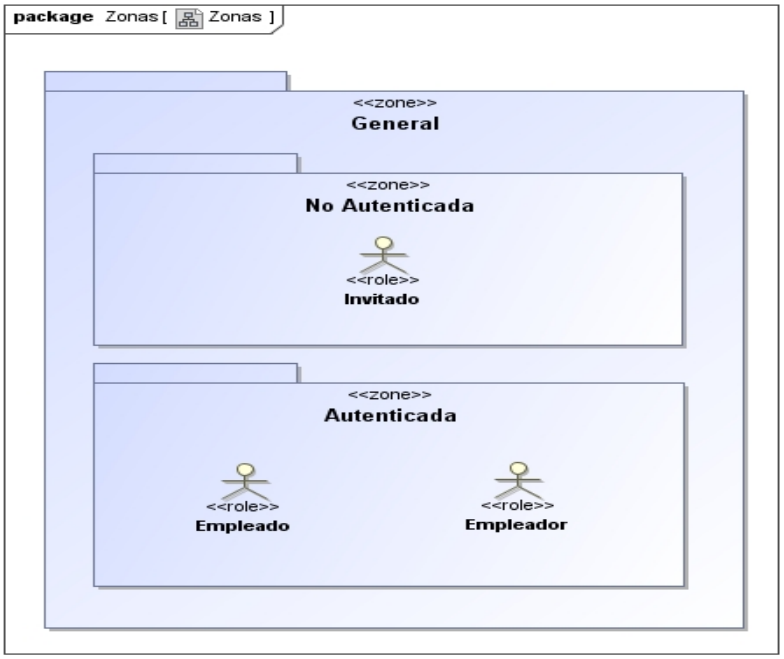
D.5.4. Registro de Empleado



D.6. Diagrama de Roles



D.7. Diagrama de Zonas

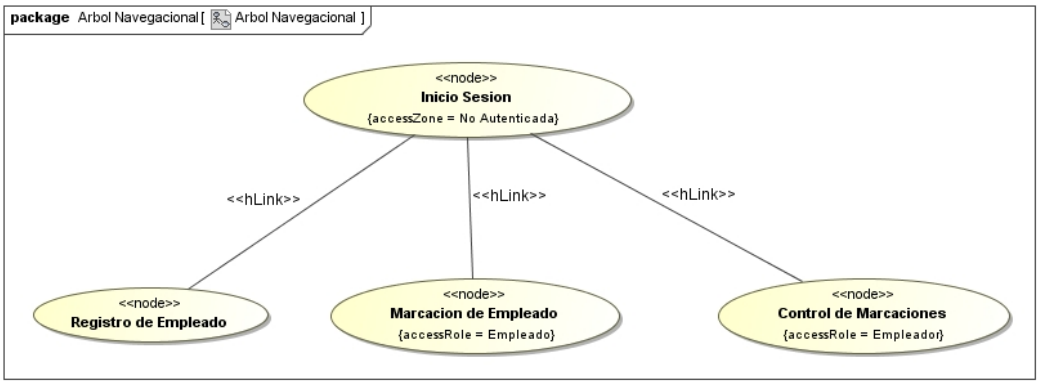


Apéndice E

ASM generado para Persistencia Móvil (completo)

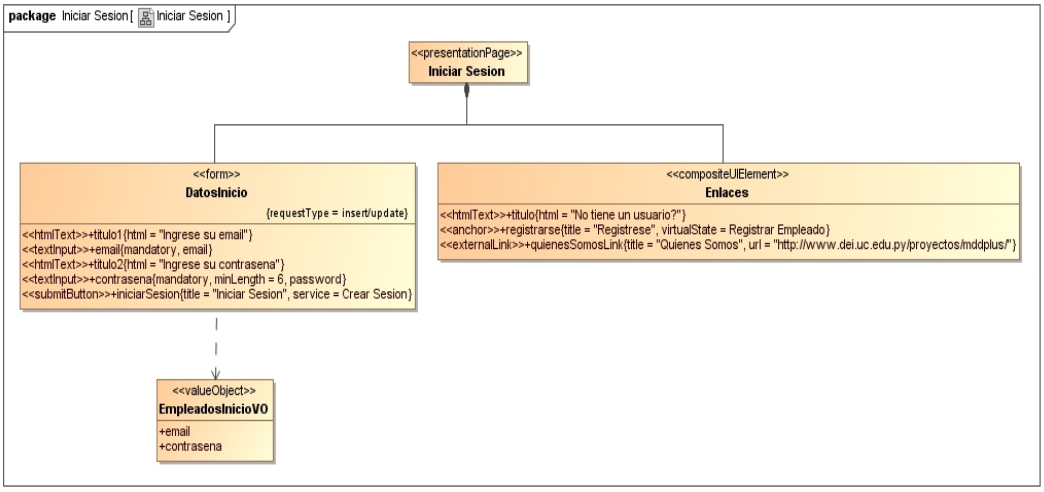
A continuación se detallan los diagramas resultantes de la aplicación de las reglas de transformación M2M definidas para la obtención del ASM orientado a persistencia móvil. Todos estos diagramas fueron obtenidos de manera automática a partir del PIM presentado en la sección 5.1.

E.1. Árbol Navegacional

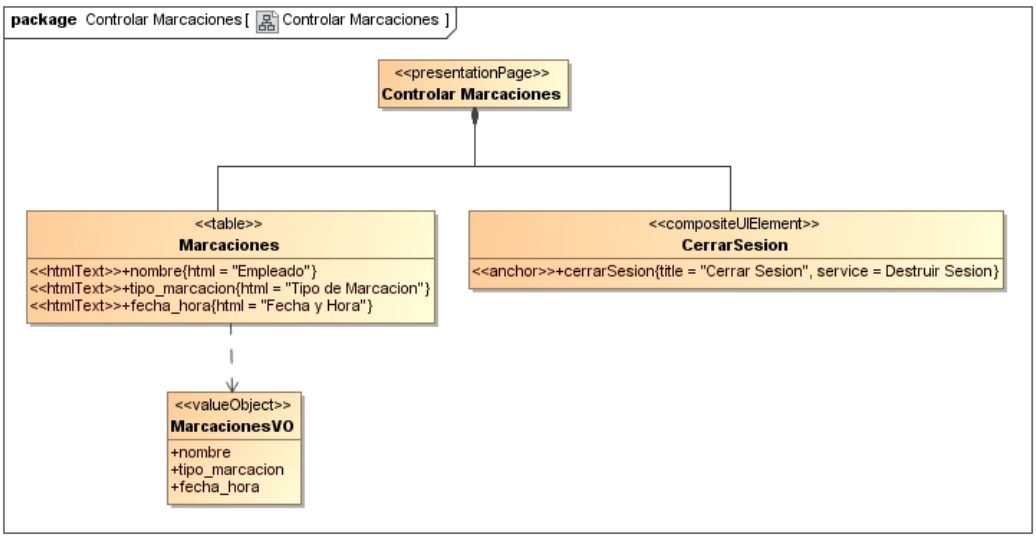


E.2. Diagramas de Contenido

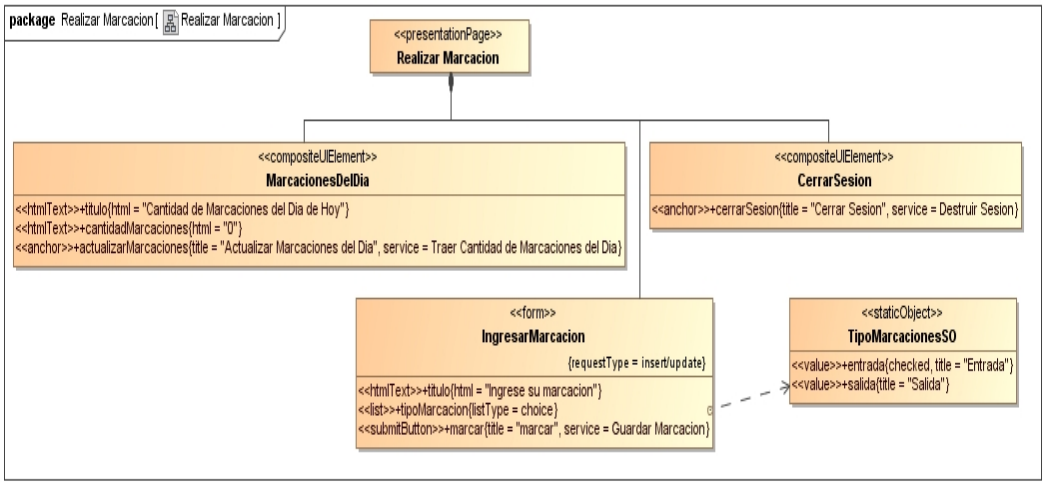
E.2.1. Iniciar Sesión



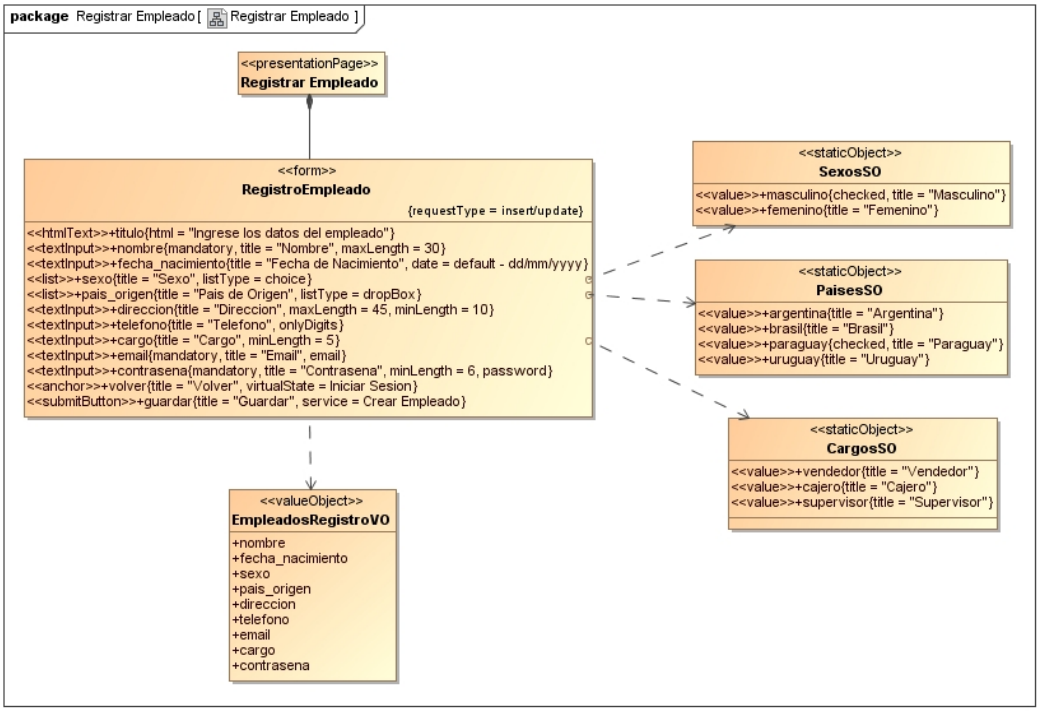
E.2.2. Controlar Marcaciones



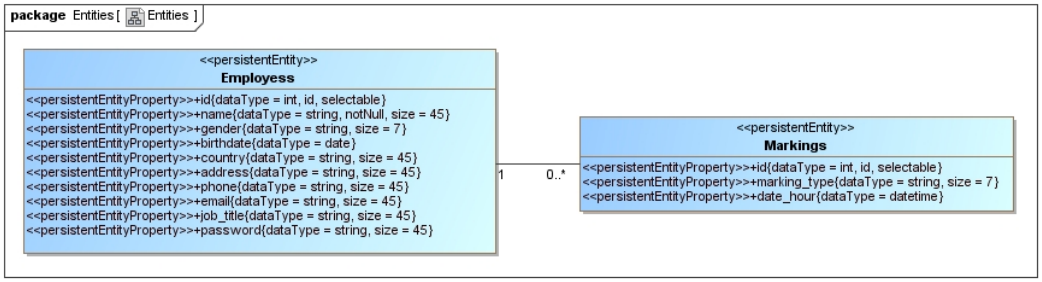
E.2.3. Realizar Marcación



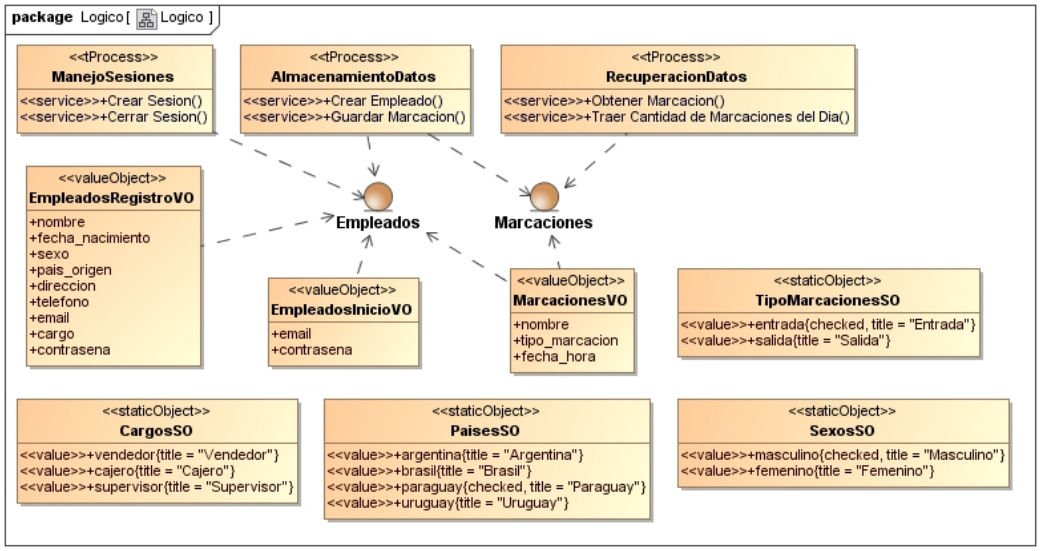
E.2.4. Registrar Empleado



E.3. Diagrama de Entidades

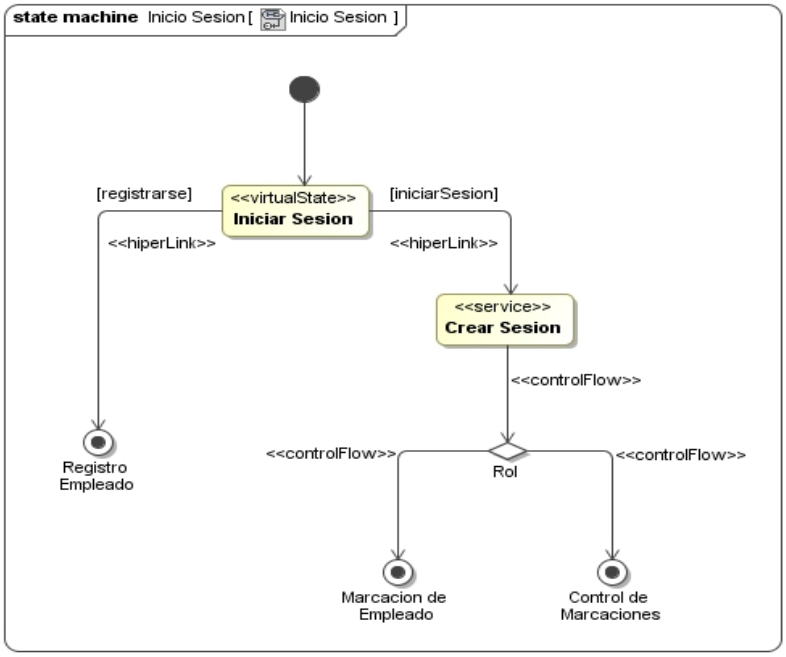


E.4. Diagrama Lógico

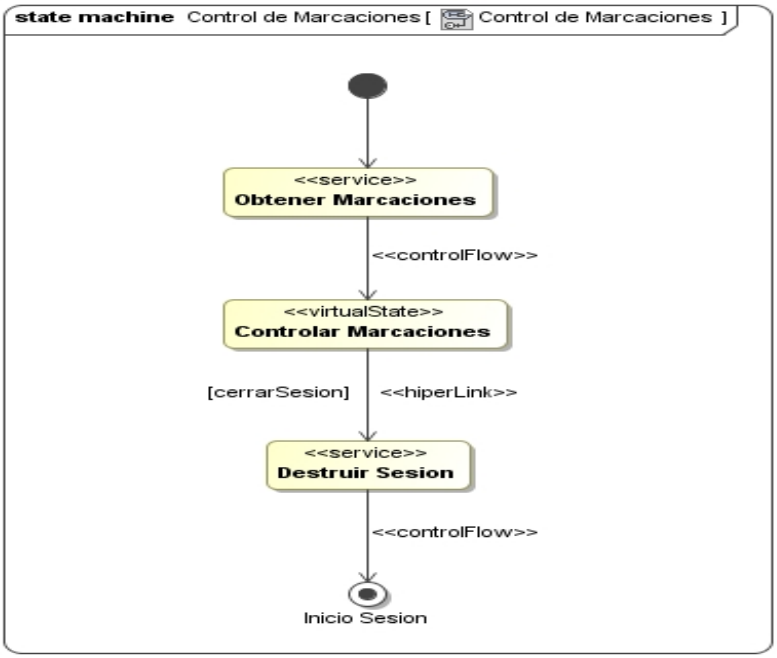


E.5. Diagramas de Nodos

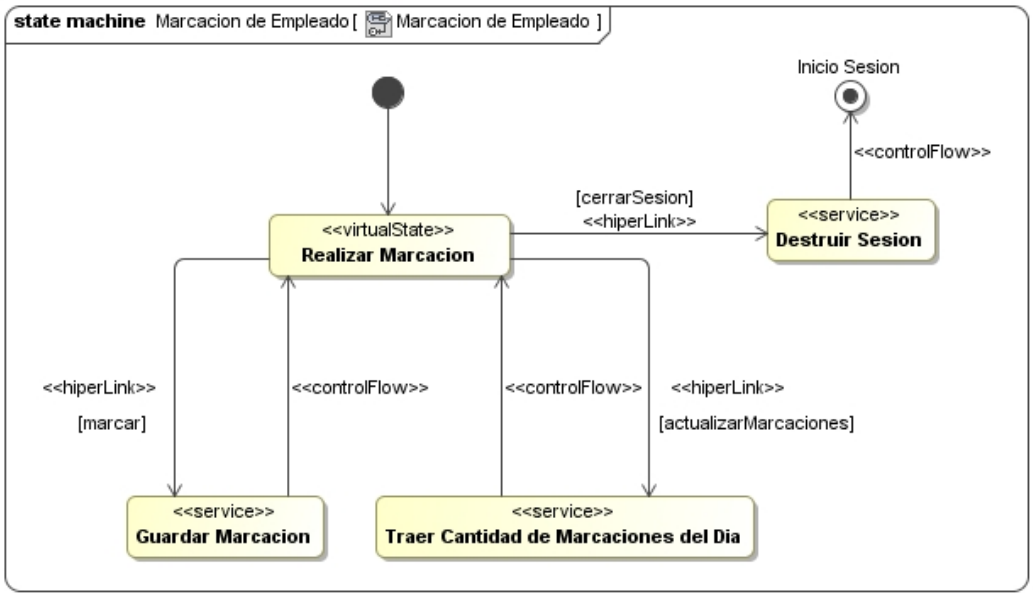
E.5.1. Inicio Sesión



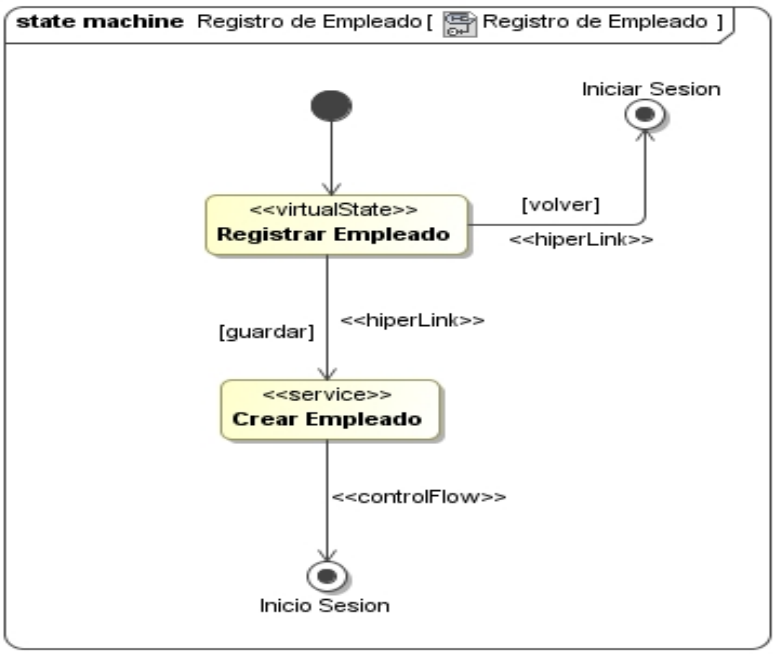
E.5.2. Control de Marcaciones



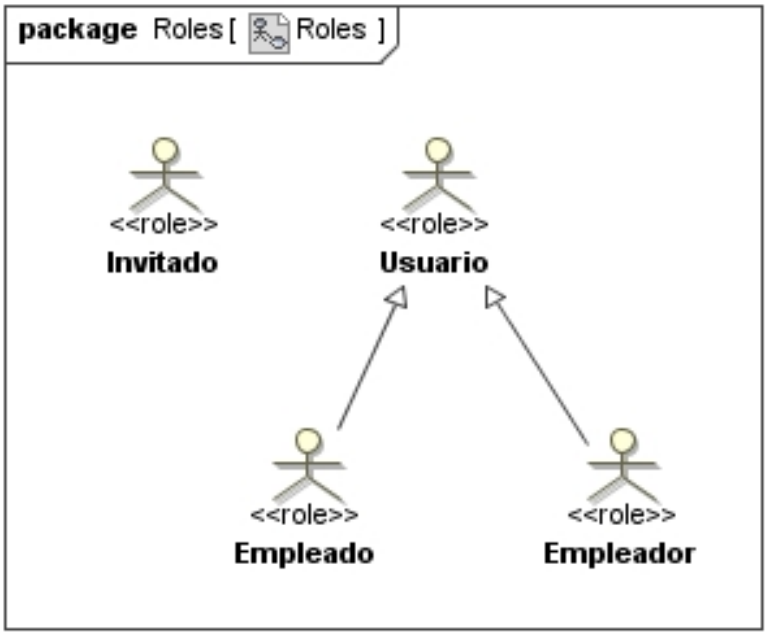
E.5.3. Marcación de Empleado



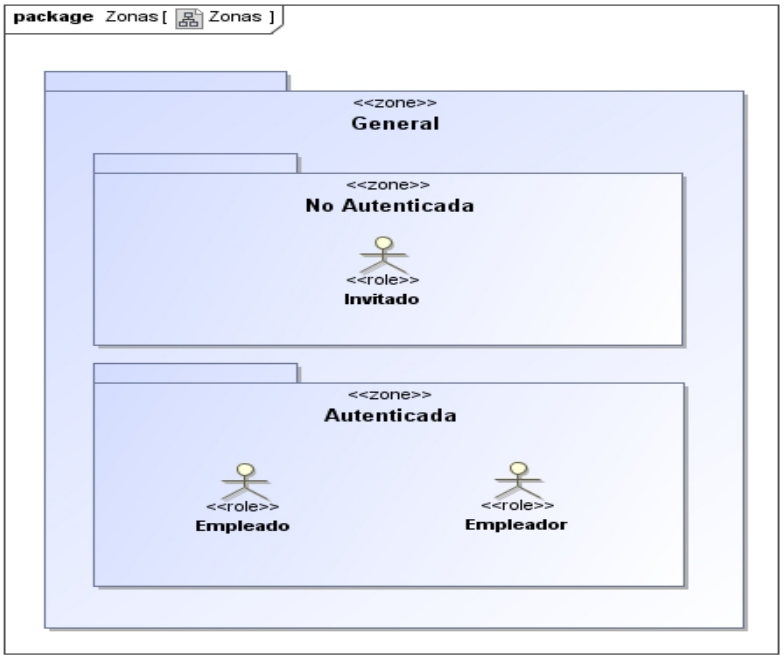
E.5.4. Registro de Empleado



E.6. Diagrama de Roles



E.7. Diagrama de Zonas



Bibliografía

[1] Y. Deshpande, S. Murugesan, A. Ginige, S. Hansen, D. Schwabe, M. Gaedke y B. White, «Web Engineering», *Journal of Web Engineering*, vol. 1, n.º 1, págs. 3-17, 2002. dirección: http://www.rintonpress.com/xjwe1/jwe-1-1/003-017.pdf.

[2] R Pressman y D Lowe, *Web Engineering: A Practitioner's approach*. New York, NY, USA: McGraw-Hill, Inc., 2009, ISBN: 978-0-07-352329-3.

[3] M. Brambilla, J. Cabot y M. Wimmer, *Model-Driven Software Engineering in Practice*, 1st, ép. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2012, ISBN: 9781608458837. DOI: 10.2200/S00441ED1V01Y201208SWE001. dirección: https://doi.org/10.2200/S00441ED1V01Y201208SWE001.

[4] A. Bozzon, S. Comai, P. Fraternali y G. T. Carughi, «Capturing RIA concepts in a web modeling language», en *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, L. Carr, D. D. Roure, A. Iyengar, C. A. Goble y M. Dahlin, eds., ACM, 2006, págs. 907-908. DOI: 10.1145/1135777.1135938. dirección: http://doi.acm.org/10.1145/1135777.1135938.

[5] N. Koch, M. Pigerl, G. Zhang y T. Morozova, «Web Engineering: 9th International Conference, ICWE 2009 San Sebastián, Spain, June 24-26, 2009 Proceedings», en, M. Gaedke, M. Grossniklaus y O. Díaz, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, cap. Patterns for the Model-Based Development of RIAs, págs. 283-291, ISBN: 978-3-642-02818-2. DOI: 10.1007/978-3-642-02818-2_23. dirección: http://dx.doi.org/10.1007/978-3-642-02818-2_23.

[6] S. Meliá, J. Gómez, S. Pérez y O. Díaz, «A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA», en *Web Engineering, 2008. ICWE '08. Eighth International Conference on*, IEEE, 2008, págs. 13-23. DOI: 10.1109/ICWE.2008.36.

[7] H. Heitkötter, T. A. Majchrzak y H. Kuchen, «Cross-platform model-driven development of mobile applications with md^2», en *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*, 2013, págs. 526-533. DOI: 10.1145/2480362.2480464. dirección: http://doi.acm.org/10.1145/2480362.2480464.

[8] A. Ranabahu, E. M. Maximilien, A. P. Sheth y K. Thirunarayan, «A domain specific language for enterprise grade cloud-mobile hybrid applications», en *Conference on Systems, Programming, and Applications: Software for Humanity, SPLASH '11, Proceedings of the compilation of the co-located workshops, DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, and VMIL'11, Portland, OR, USA, October 22 - 27, 2011*, 2011, págs. 77-84. DOI: 10.1145/2095050.2095064. dirección: http://doi.acm.org/10.1145/2095050.2095064.

[9] M. González, L. Cernuzzi y O. Pastor, «A navigational role-centric model oriented web approach - MoWebA», *Int. J. Web Eng. Technol.*, vol. 11, n.º 1, págs. 29-67, 2016. DOI: 10.1504/IJWET.2016.075963. dirección: http://dx.doi.org/10.1504/IJWET.2016.075963.

[10] T. Mikkonen, R. Pitkänen y M. Pussinen, «On the Role of Architectural Style in Model Driven Development», en *Software Architecture, First European Workshop, EWSA 2004, St Andrews, UK, May 21-22, 2004, Proceedings*, 2004, págs. 74-87. DOI: 10.1007/978-3-540-24769-2_6. dirección: http://dx.doi.org/10.1007/978-3-540-24769-2_6.

[11] C. Pons, R. Giandini y G. Pérez, *Desarrollo de Software dirigido por Modelos (Conceptos teóricos y su aplicación práctica)*. Edulp - Editorial de la Universidad de la Plata, 2010.

[12] M. González, L. Cernuzzi, N. Aquino y O. Pastor, «Developing web applications for different architectures: The MoWebA approach», en *Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016, Grenoble, France, June 1-3, 2016*, 2016, págs. 1-11. DOI: 10.1109/RCIS.2016.7549344. dirección: http://dx.doi.org/10.1109/RCIS.2016.7549344.

[13] J. G. Molina, F. O. G. Rubio, V. Pelechano, A. Vallecillo, J. M. Vara y C. Vicente-Chicote, *Desarrollo de Software Dirigido por Modelos: Conceptos, Métodos y Herramientas*. Ra-Ma Editorial, 2013, ISBN: 978-84-9964-215-4.

[14] M. González, L. Cernuzzi y O. Pastor, «Una Aproximación para Aplicaciones Web: MoWebA», *Congreso Iberoamericano en "Software Engineering"(CIbSE)*, 2011.

[15] A. Bozzon, S. Comai, P. Fraternali y G. T. Carughi, «Conceptual Modeling and Code Generation for Rich Internet Applications», en *Proceedings of the 6th International Conference on Web Engineering*, ép. ICWE '06, Palo Alto, California, USA: ACM, 2006, págs. 353-360, ISBN: 1-59593-352-2. DOI: 10.1145/1145581.1145649. dirección: http://doi.acm.org/10.1145/1145581.1145649.

[16] D. Bonhaure, M. González, N. Aquino, L. Cernuzzi y C. Pons, «Model-to-model transformations for RIA architectures: A systematic mapping study», en *2016 XLII Latin American Computing Conference (CLEI)*, 2016, págs. 1-11. DOI: 10.1109/CLEI.2016.7833405.

[17] D. Bonhaure, M. González, N. Aquino, L. Cernuzzi y C. Pons, «Exploring Model-to-Model Transformations for RIA Architectures by means of a Systematic Mapping Study», *CLEI Electronic Journal*, vol. 20, n.º 3, dic. de 2017, (to apper). DOI: 10.19153/cleiej.20.3.5.

[18] T. DeMarco, *Structured Analysis and System Specification*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1979, ISBN: 0138543801.

[19] J. Parodi y D. S. Frankel, *The MDA journal: model driven architecture straight from the masters*. Meghan-Kiffer Press, 2004.

[20] T. Stahl, M. Voelter y K. Czarnecki, *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006, ISBN: 0470025700.

[21] M. González, J. Casariego, J. J. Bareiro, L. Cernuzzi y O. Pastor, «A MDA Approach for Navigational and User Perspectives», *CLEI Electron. J.*, vol. 14, n.º 1, 2011. dirección: http://www.clei.org/cleiej/paper.php\?id=208.

[22] G. Nuñez, M. González, N. Aquino y L. Cernuzzi, «Un enfoque de Desarrollo Dirigido por Modelos para Aplicaciones Web Enriquecidas», en *2017 XLIII Latin American Computing Conference (CLEI)*, (in press), Córdoba, Argentina, 2017.

[23] M. Nuñez, «Un Enfoque MDD para el desarrollo de Aplicaciones Móviles Nativas enfocadas en la Capa de Datos.», http://www.dei.uc.edu.py/proyectos/mddplus/wp-content/uploads/2017/02/Proyecto-Final-de-Carrera-Guido-Nu%C3%B1ez.pdf, Universidad Católica "Nuestra Señora de la Asunción", Tte. Cantaluppi y G. Molinas, Asunción, ene. de 2017.

[24] G. Nuñez, «Un Enfoque MDD para el desarrollo de RIA», http://www.dei.uc.edu.py/proyectos/mddplus/wp-content/uploads/2017/05/Proyecto-Final-de-Carrera-Manuel-Nu%C3%B1ez.pdf, Universidad Católica "Nuestra Señora de la Asunción", Tte. Cantaluppi y G. Molinas, Asunción, ene. de 2017.

[25] M. Brambilla, J. C. Preciado, M. L. Trigueros y F. Sánchez-Figueroa, «Business Process-Based Conceptual Design of Rich Internet Applications», en *Proceedings of the Eighth International Conference on Web Engineering, ICWE 2008, 14-18 July 2008, Yorktown Heights, New York, USA*, D. Schwabe, F. Curbera y P. Dantzig, eds., IEEE Computer Society, 2008, págs. 155-161. DOI: 10.1109/ICWE.2008.22. dirección: https://doi.org/10.1109/ICWE.2008.22.

[26] P. Fraternali, G. Rossi y F. Sánchez-Figueroa, «Rich Internet Applications», *IEEE Internet Computing*, vol. 14, n.º 3, págs. 9-12, 2010. DOI: 10.1109/MIC.2010.76. dirección: https://doi.org/10.1109/MIC.2010.76.

[27] M. Busch y N. Koch, «Rich Internet Applications: State-of-the-Art», *Ludwig-Maximilians-Universität München, München, Germany, Rep*, vol. 902, 2009.

[28] M. Linaje, J. C. Preciado y F. Sánchez-Figueroa, «A method for model based design of rich internet application interactive user interfaces», en *Web Engineering*, Springer, 2007, págs. 226-241.

[29] J. L. H. Agustin, «Model-driven web applications», en *Science and Information Conference (SAI), 2015*, IEEE, 2015, págs. 954-964.

[30] M. L. Bernardi, G. A. Di Lucca y D. Distante, «Model-driven fast prototyping of RIAs: From conceptual models to running applications», en *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, IEEE, 2014, págs. 250-258.

[31] D. Manset, H. Verjus, R. McClatchey y F. Oquendo, «A Formal Architecture-Centric Model-Driven Approach for the Automatic Generation of Grid Applications», en *ICEIS 2006 - Proceedings of the Eighth International Conference on Enterprise Information Systems: Databases and Information Systems Integration, Paphos, Cyprus, May 23-27, 2006*, 2006, págs. 322-330.

[32] E. Marcos, C. J. Acuña y C. E. Cuesta, «Integrating Software Architecture into a MDA Framework», en *Software Architecture, Third European Workshop, EWSA 2006, Nantes, France, September 4-5, 2006, Revised Selected Papers*, 2006, págs. 127-143. DOI: 10.1007/11966104_10. dirección: http://dx.doi.org/10.1007/11966104_10.

[33] M. L. Sanz y E. Marcos, «ArchiMeDeS: A model-driven framework for the specification of service-oriented architectures», *Inf. Syst.*, vol. 37, n.º 3, págs. 257-268, 2012. DOI: 10.1016/j.is.2011.11.002. dirección: http://dx.doi.org/10.1016/j.is.2011.11.002.

[34] M. López-Sanz y E. Marcos, «Modeling Platform-Independent and Platform-Specific Service Architectures with UML and the ArchiMeDeS Framework», en. 2014, cap. 11, págs. 254-277. DOI: doi:10.4018/978-1-4666-6026-7.ch011.

[35] N. Elleuch, A. Khalfallah y S. B. Ahmed, «ArchMDE Approach for the Formal Verification of Real Time Systems», en *11th IEEE International Conference on Computer and Information Technology, CIT 2011, Pafos, Cyprus, 31 August-2 September 2011*, 2011, págs. 533-538. DOI: 10.1109/CIT.2011.39. dirección: http://dx.doi.org/10.1109/CIT.2011.39.

[36] K. Petersen, S. Vakkalanka y L. Kuzniarz, «Guidelines for conducting systematic mapping studies in software engineering: An update», *Information & Software Technology*, vol. 64, págs. 1-18, 2015. DOI: 10.1016/j.infsof.2015.03.007. dirección: http://dx.doi.org/10.1016/j.infsof.2015.03.007.

[37] S. Casteleyn, I. Garrigós y J.-N. Mazón, «Ten Years of Rich Internet Applications: A Systematic Mapping Study, and Beyond», *ACM Trans. Web*, vol. 8, n.º 3, 18:1-18:46, jul. de 2014, ISSN: 1559-1131. DOI: 10.1145/2626369. dirección: http://doi.acm.org/10.1145/2626369.

[38] G. Aragón, M.-J. Escalona, M. Lang y J. R. Hilera, «An analysis of model-driven web engineering methodologies», *International Journal of Innovative Computing, Information and Control (IJICIC)*, vol. 9, n.º 1, págs. 413-436, 2013.

[39] J. Allaire, «Macromedia Flash MX-A next-generation rich client», *Macromedia White Paper*, págs. 1-2, 2002.

[40] B. A. Kitchenham, D. Budgen y O. P. Brereton, «Using mapping studies as the basis for further research - A participant-observer case study», *Information & Software Technology*, vol. 53, n.º 6, págs. 638-651, 2011. DOI: 10.1016/j.infsof.2010.12.011. dirección: http://dx.doi.org/10.1016/j.infsof.2010.12.011.

[41] M. Genero Bocco, J. Cruz-Lemos y M. Piattini Velthuis, *Métodos de investigación en ingeniería del software*. Ra-Ma Editorial, 2014, ISBN: 978-84-9964-507-0.

[42] B. Kitchenham y S. Charters, «Guidelines for performing Systematic Literature Reviews in Software Engineering», Keele University y Durham University Joint Report, inf. téc. EBSE 2007-001, 2007.

[43] K. Petersen, R. Feldt, S. Mujtaba y M. Mattsson, «Systematic Mapping Studies in Software Engineering», en *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, University of Bari, Italy, 26-27 June 2008*, G. Visaggio, M. T. Baldassarre, S. G. Linkman y M. Turner, eds., ép. Workshops in Computing, BCS, 2008. dirección: `http://ewic.bcs.org/content/ConWebDoc/19543`.

[44] C. Wohlin, P. Runeson, P. A. da Mota Silveira Neto, E. Engström, I. do Carmo Machado y E. S. de Almeida, «On the reliability of mapping studies in software engineering», *Journal of Systems and Software*, vol. 86, n.º 10, págs. 2594-2610, 2013. DOI: `10.1016/j.jss.2013.04.076`. dirección: `http://dx.doi.org/10.1016/j.jss.2013.04.076`.

[45] T. Dybå, T. Dingsøyr y G. K. Hanssen, «Applying Systematic Reviews to Diverse Study Types: An Experience Report», en *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, September 20-21, 2007, Madrid, Spain*, ACM / IEEE Computer Society, 2007, págs. 225-234. DOI: `10.1109/ESEM.2007.59`. dirección: `http://dx.doi.org/10.1109/ESEM.2007.59`.

[46] M. Petticrew y H. Roberts, *Systematic reviews in the social sciences: A practical guide*. Blackwell Publishing Ltd, 2008, ISBN: 9780470754887. DOI: `10.1002/978047075488`. dirección: `http://dx.doi.org/10.1002/9780470754887`.

[47] S. Link, T. Schuster, P. Hoyer y S. Abeck, «Focusing Graphical User Interfaces in Model-Driven Software Development», en *Advances in Computer-Human Interaction, 2008 First International Conference on*, IEEE, 2008, págs. 3-8. DOI: `10.1109/ACHI.2008.16`.

[48] M. L. Bernardi, G. A. D. Lucca y D. Distante, «Model-driven fast prototyping of RIAs: From conceptual models to running applications», en *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, IEEE, 2014, págs. 250-258. DOI: `10.1109/ICACCI.2014.6968522`.

[49] S. Pérez, O. Díaz, S. Meliá y J. Gómez, «Facing Interaction-Rich RIAs: The Orchestration Model», en *Web Engineering, 2008. ICWE '08. Eighth International Conference on*, IEEE, 2008, págs. 24-37. DOI: `10.1109/ICWE.2008.12`.

[50] J. C. Preciado, M. Linaje, R. Morales-Chaparro, F. Sanchez-Figueroa, G. Zhang, C. Kroiß y N. Koch, «Designing Rich Internet Applications Combining UWE and RUX-Method», en *Web Engineering, 2008. ICWE '08. Eighth International Conference on*, IEEE, 2008, págs. 148-154. DOI: `10.1109/ICWE.2008.26`.

[51] S. Meliá, J. Gómez, S. Pérez y O. Díaz, «Facing Architectural and Technological Variability of Rich Internet Applications», *IEEE Internet Computing*, vol. PP, n.º 99, págs. 1-1, 2010, ISSN: 1089-7801. DOI: `10.1109/MIC.2010.53`.

[52] E. Sosa, P. J. Clemente, J. M. Conejero y R. Rodríguez-Echeverría, «A model-driven process to modernize legacy web applications based on service oriented architectures», en *Web Systems Evolution (WSE), 2013 15th IEEE International Symposium on*, IEEE, 2013, págs. 61-70. DOI: 10.1109/WSE.2013.6642418.

[53] S. Roubi, M. Erramdani y S. Mbarki, «A model driven approach to generate graphical user interfaces for Rich Internet Applications using Interaction Flow Modeling Language», en *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, IEEE, 2015, págs. 272-276. DOI: 10.1109/ISDA.2015.7489237.

[54] R. Esbai y M. Erramdani, «Model-to-model transformation in approach by modeling: From UML model to Model-View-Presenter and Dependency Injection patterns», en *2015 5th World Congress on Information and Communication Technologies (WICT)*, IEEE, 2015, págs. 1-6. DOI: 10.1109/WICT.2015.7489648.

[55] V. Torres, P. Giner y V. Pelechano, «Developing BP-driven web applications through the use of MDE techniques», *Software & Systems Modeling*, vol. 11, n.º 4, págs. 609-631, 2012, ISSN: 1619-1374. DOI: 10.1007/s10270-010-0177-5. dirección: http://dx.doi.org/10.1007/s10270-010-0177-5.

[56] E. Robles Luna, M. J. Escalona y G. Rossi, «Software and Data Technologies: 5th International Conference, ICSOFT 2010, Athens, Greece, July 22-24, 2010. Revised Selected Papers», en, J. Cordeiro, M. Virvou y B. Shishkov, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, cap. Modelling the Requirements of Rich Internet Applications in WebRe, págs. 27-41, ISBN: 978-3-642-29578-2. DOI: 10.1007/978-3-642-29578-2_2. dirección: http://dx.doi.org/10.1007/978-3-642-29578-2_2.

[57] Ó. Pastor, M. Ruiz y S. España, «Software and Data Technologies: 6th International Conference, ICSOFT 2011, Seville, Spain, July 18-21, 2011. Revised Selected Papers», en, M. J. Escalona, J. Cordeiro y B. Shishkov, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, cap. From Requirements to Code: A Full Model-Driven Development Perspective, págs. 56-70, ISBN: 978-3-642-36177-7. DOI: 10.1007/978-3-642-36177-7_4. dirección: http://dx.doi.org/10.1007/978-3-642-36177-7_4.

[58] J. M. Hermida, S. Meliá, J.-J. Martínez, A. Montoyo y J. Gómez, «Current Trends in Web Engineering: ICWE 2012 International Workshops: MDWE, ComposableWeb, WeRE, QWE, and Doctoral Consortium, Berlin, Germany, July 23-27, 2012, Revised Selected Papers», en, M. Grossniklaus y M. Wimmer, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, cap. Developing Semantic Rich Internet Applications with the Sm4RIA Extension for OIDE, págs. 20-25, ISBN: 978-3-642-35623-0. DOI: 10.1007/978-3-642-35623-0_3. dirección: http://dx.doi.org/10.1007/978-3-642-35623-0_3.

[59] R. Rodríguez-Echeverría, J. M. Conejero, M. Linaje, J. C. Preciado y F. Sánchez-Figueroa, «Web Engineering: 10th International Conference, ICWE 2010, Vienna Austria, July 5-9, 2010. Proceedings», en, B. Benatallah, F. Casati, G. Kappel y G. Rossi, eds. Berlin, Heidelberg: Springer Berlin Heidelberg,

2010, cap. Re-engineering Legacy Web Applications into Rich Internet Applications, págs. 189-203, ISBN: 978-3-642-13911-6. DOI: 10.1007/978-3-642-13911-6_13. dirección: http://dx.doi.org/10.1007/978-3-642-13911-6_13.

[60] S. Meliá, J.-J. Martínez, S. Mira, J. A. Osuna y J. Gómez, «Web Engineering: 10th International Conference, ICWE 2010, Vienna Austria, July 5-9, 2010. Proceedings», en, B. Benatallah, F. Casati, G. Kappel y G. Rossi, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, cap. An Eclipse Plug-in for Model-Driven Development of Rich Internet Applications, págs. 514-517, ISBN: 978-3-642-13911-6. DOI: 10.1007/978-3-642-13911-6_41. dirección: http://dx.doi.org/10.1007/978-3-642-13911-6_41.

[61] J. M. Hermida, S. Meliá, A. Montoyo y J. Gómez, «Web Information Systems Engineering – WISE 2010 Workshops: WISE 2010 International Symposium WISS, and International Workshops CISE, MBC, Hong Kong, China, December 12-14, 2010, Revised Selected Papers», en, D. K. W. Chiu, L. Bellatreche, H. Sasaki, H.-f. Leung, S.-C. Cheung, H. Hu y J. Shao, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, cap. Developing Semantic Rich Internet Applications Using a Model-Driven Approach, págs. 198-211, ISBN: 978-3-642-24396-7. DOI: 10.1007/978-3-642-24396-7_16. dirección: http://dx.doi.org/10.1007/978-3-642-24396-7_16.

[62] R. Rodríguez-Echeverría, J. M. Conejero, P. J. Clemente, V. M. Pavón y F. Sánchez-Figueroa, «Current Trends in Web Engineering: ICWE 2012 International Workshops: MDWE, ComposableWeb, WeRE, QWE, and Doctoral Consortium, Berlin, Germany, July 23-27, 2012, Revised Selected Papers», en, M. Grossniklaus y M. Wimmer, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, cap. Model Driven Extraction of the Navigational Concern of Legacy Web Applications, págs. 56-70, ISBN: 978-3-642-35623-0. DOI: 10.1007/978-3-642-35623-0_6. dirección: http://dx.doi.org/10.1007/978-3-642-35623-0_6.

[63] H. Casalánguida y J. E. Durán, «Current Trends in Web Engineering: ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013. Revised Selected Papers», en, Q. Z. Sheng y J. Kjeldskov, eds. Cham: Springer International Publishing, 2013, cap. A Method for Integrating Process Description and User Interface Use during Design of RIA Applications, págs. 172-186, ISBN: 978-3-319-04244-2. DOI: 10.1007/978-3-319-04244-2_16. dirección: http://dx.doi.org/10.1007/978-3-319-04244-2_16.

[64] R. Rodríguez-Echeverría, V. M. Pavón, F. Macías, J. M. Conejero, P. J. Clemente y F. Sánchez-Figueroa, «Web Information Systems Engineering – WISE 2013: 14th International Conference, Nanjing, China, October 13-15, 2013, Proceedings, Part II», en, X. Lin, Y. Manolopoulos, D. Srivastava y G. Huang, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, cap. Generating a Conceptual Representation of a Legacy Web Application, págs. 231-240, ISBN: 978-3-642-41154-0. DOI: 10.1007/978-3-642-41154-0_17. dirección: http://dx.doi.org/10.1007/978-3-642-41154-0_17.

[65] R. Rodríguez-Echeverría, J. M. Conejero, P. J. Clemente, M. D. Villalobos y F. Sánchez-Figueroa, «Web Engineering: 12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012. Proceedings», en, M. Brambilla, T. Tokuda y R. Tolksdorf, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, cap. Extracting Navigational Models from Struts-Based Web Applications, págs. 419-426, ISBN: 978-3-642-31753-8. DOI: 10.1007/978-3-642-31753-8_35. dirección: http://dx.doi.org/10.1007/978-3-642-31753-8_35.

[66] A. Pleuß y H. Hußmann, «Human-Computer Interaction. Interaction Design and Usability: 12th International Conference, HCI International 2007, Beijing, China, July 22-27, 2007, Proceedings, Part I», en, J. A. Jacko, ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, cap. Integrating Authoring Tools into Model-Driven Development of Interactive Multimedia Applications, págs. 1168-1177, ISBN: 978-3-540-73105-4. DOI: 10.1007/978-3-540-73105-4_127. dirección: http://dx.doi.org/10.1007/978-3-540-73105-4_127.

[67] I. Garrigós, S. Meliá y S. Casteleyn, «Web Information Systems Engineering - WISE 2009: 10th International Conference, Poznań, Poland, October 5-7, 2009. Proceedings», en, G. Vossen, D. D. E. Long y J. X. Yu, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, cap. Personalizing the Interface in Rich Internet Applications, págs. 365-378, ISBN: 978-3-642-04409-0. DOI: 10.1007/978-3-642-04409-0_37. dirección: http://dx.doi.org/10.1007/978-3-642-04409-0_37.

[68] L. Dannecker, M. Feldmann, T. Nestler, G. Hübsch, U. Jugel y K. Muthmann, «Current Trends in Web Engineering: 10th International Conference on Web Engineering ICWE 2010 Workshops, Vienna, Austria, July 2010, Revised Selected Papers», en, F. Daniel y F. M. Facca, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, cap. Rapid Development of Composite Applications Using Annotated Web Services, págs. 1-12, ISBN: 978-3-642-16985-4. DOI: 10.1007/978-3-642-16985-4_1. dirección: http://dx.doi.org/10.1007/978-3-642-16985-4_1.

[69] S. Pérez, F. Durao, S. Meliá, P. Dolog y O. Díaz, «Web Information Systems Engineering – WISE 2010 Workshops: WISE 2010 International Symposium WISS, and International Workshops CISE, MBC, Hong Kong, China, December 12-14, 2010, Revised Selected Papers», en, D. K. W. Chiu, L. Bellatreche, H. Sasaki, H.-f. Leung, S.-C. Cheung, H. Hu y J. Shao, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, cap. RESTful, Resource-Oriented Architectures: A Model-Driven Approach, págs. 282-294, ISBN: 978-3-642-24396-7. DOI: 10.1007/978-3-642-24396-7_22. dirección: http://dx.doi.org/10.1007/978-3-642-24396-7_22.

[70] J. M. Hermida, S. Meliá, A. Montoyo y J. Gómez, «Applying model-driven engineering to the development of Rich Internet Applications for Business Intelligence», *Information Systems Frontiers*, vol. 15, n.º 3, págs. 411-431, 2013, ISSN: 1572-9419. DOI: 10.1007/s10796-012-9402-9. dirección: http://dx.doi.org/10.1007/s10796-012-9402-9.

[71] L. Mainetti, R. Paiano y A. Pandurino, «Web Engineering: 12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012. Proceedings», en, M. Brambilla, T. Tokuda y R. Tolksdorf, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, cap. MIGROS: A Model-Driven Transformation Approach of the User Experience of Legacy Applications, págs. 490-493, ISBN: 978-3-642-31753-8. DOI: 10.1007/978-3-642-31753-8_51. dirección: http://dx.doi.org/10.1007/978-3-642-31753-8_51.

[72] F. Valverde y O. Pastor, «Web Information Systems Engineering - WISE 2009: 10th International Conference, Poznań, Poland, October 5-7, 2009. Proceedings», en, G. Vossen, D. D. E. Long y J. X. Yu, eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, cap. Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach, págs. 131-144, ISBN: 978-3-642-04409-0. DOI: 10.1007/978-3-642-04409-0_18. dirección: http://dx.doi.org/10.1007/978-3-642-04409-0_18.

[73] E. Yigitbas, B. Mohrmann y S. Sauer, «Model-driven UI Development Integrating HCI Patterns», en *CEUR Workshop Proceedings*, cited By 0, vol. 1380, 2015, págs. 42-46. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84938499474\&partnerID=40\&md5=32c4ac570d4447073bdce9cbc7f47474.

[74] Y.-C. Huang y C.-P. Chu, «Developing web applications based on model driven architecture», *International Journal of Software Engineering and Knowledge Engineering*, vol. 24, n.º 2, págs. 163-182, 2014, cited By 0. DOI: 10.1142/S0218194014500077. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84903266176\&partnerID=40\&md5=07b722e42f882c7852592ff6cb04fd85.

[75] M. Urbieta, M. Urbieta, G. Rossi, G. Rossi, J. Ginzburg, J. Ginzburg, D. Schwabe y D. Schwabe, «Designing the Interface of Rich Internet Applications», en *Web Conference, 2007. LA-WEB 2007. Latin American*, IEEE, 2007, págs. 144-153. DOI: 10.1109/LA-Web.2007.14.

[76] J. C. Preciado, M. Linaje, S. Comai y F. Sanchez-Figueroa, «Designing Rich Internet Applications with Web Engineering Methodologies», en *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on*, IEEE, 2007, págs. 23-30. DOI: 10.1109/WSE.2007.4380240.

[77] K. Wakil y D. Jawawi, «Model driven web engineering: A systematic mapping study», *E-Informatica Software Engineering Journal*, vol. 9, n.º 1, págs. 87-122, 2015, cited By 2. DOI: 10.5277/E-INF150106. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84953859883\&partnerID=40\&md5=f558d9841ce4a15a87afb5faf49725c0.

[78] I. Ruiz-Rube, J. M. Dodero, M. Palomo-Duarte, M. Ruiz y D. Gawn, «Uses and applications of Software & Systems Process Engineering Meta-Model process models. A systematic mapping study», *Journal of Software: Evolution and Process*, vol. 25, n.º 9, págs. 999-1025, 2013, ISSN: 2047-7481. DOI: 10.1002/smr.1594. dirección: http://dx.doi.org/10.1002/smr.1594.

[79] T. Mens y P. Van Gorp, «A Taxonomy of Model Transformation», *Electron. Notes Theor. Comput. Sci.*, vol. 152, págs. 125-142, mar. de 2006, ISSN: 1571-0661. DOI: 10.1016/j.entcs.2005.10.021. dirección: http://dx.doi.org/10.1016/j.entcs.2005.10.021.

[80] K. Czarnecki y S. Helsen, «Classification of Model Transformation Approaches», en *2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*, vol. 45, Anaheim, CA, USA, 2003, págs. 1-17. dirección: http://www.softmetaware.com/oopsla2003/czarnecki.pdf.

[81] F. Truyen, *The Fast Guide to Model Driven Architecture*, http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf, Accessed at 30-04-2017, 2006.

[82] K. Petersen y Ç. Gencel, «Worldviews, Research Methods, and their Relationship to Validity in Empirical Software Engineering Research», en *2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, Ankara, Turkey, October 23-26, 2013*, IEEE Computer Society, 2013, págs. 81-89. DOI: 10.1109/IWSM-Mensura.2013.22. dirección: http://dx.doi.org/10.1109/IWSM-Mensura.2013.22.

[83] C. Pannucci y E. Wilkins, «Identifying and avoiding bias in research.», *Plastic and reconstructive surgery*, vol. 126, n.º 2, págs. 619-625, 2010, cited By 75. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77955354399&partnerID=40&md5=89a94af93fe186cae0a88bb582336fde.

[84] F. Jouault y D. Wagelaar, *ATL EMF Transformation Virtual Machine (research VM) - Performance*, https://wiki.eclipse.org/ATL/EMFTVM#Performance, Accessed at 19-02-2017, 2017.

[85] ——, *ATL EMF Transformation Virtual Machine (research VM) - Invoking native Java methods*, https://wiki.eclipse.org/ATL/EMFTVM#Invoking_native_Java_methods, Accessed at 19-02-2017, 2017.

[86] T. E. Foundation, *Eclipse - Model Development Tools (MDT)*, http://www.eclipse.org/modeling/mdt/, Accessed at 20-04-2017, 2017.

[87] Wikipedia, *Facade (patrón de diseño)*, https://es.wikipedia.org/wiki/Facade_(patr%C3%B3n_de_dise%C3%B1o), Accessed at 20-06-2017, 2017.