UNIVERSIDAD NACIONAL DE COLOMBIA

Transformación de Java a Archimate en el dominio de aplicaciones utilizando Model Driven Reverse Engineering

Carlos Eduardo García Amaya

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá D.C., Colombia

2016

Transformación de Java a Archimate en el dominio de aplicaciones utilizando Model Driven Reverse Engineering

Carlos Eduardo García Amaya

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:

Magister en Ingeniería – Ingeniería de Sistemas y Computación

Director (a):

MsC Henry Roberto Umaña Acosta

Línea de Investigación:

Ingeniería de Software

Grupo de Investigación:

Colectivo de Investigación en Ingeniería de Software (ColSWE)

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá D.C., Colombia

2016

(Dedicatoria o lema)

Dedicada a Dios y a mis padres.

“Nunca olvides que basta una persona o una idea para cambiar tu vida para siempre, ya sea para bien o para mal”
Brown, J.

Agradecimientos

A Dios por permitirme estudiar en tan prestigiosa Universidad, a mis padres por todo el apoyo que me dieron, al profesor Henry Roberto Umaña Acosta director del presente trabajo de investigación quien me brindo orientación y los espacios necesarios para revisar avances, a Jhon Alexander Cruz Castelblanco, egresado de la Maestría en Ingeniería de Sistemas y Computación de la Universidad Nacional de Colombia, quien me brindo de su tiempo y conocimiento con apoyo en la solución de inquietudes que surgieron en el desarrollo del presente trabajo.

Resumen

El desarrollo de un BASELINE de arquitectura empresarial requiere de un esfuerzo significativo dependiendo del tamaño de una empresa, del conocimiento de sus funcionarios y las habilidades de los arquitectos empresariales; el objetivo principal de este trabajo de investigación es crear una herramienta MDRE llamada JTOGAF que facilite la documentación del BASELINE de arquitectura empresarial en el dominio de Aplicaciones para empresas. JTOGAF utiliza como entrada código fuente desarrollado en Java que cumpla el estándar J2EE y genera como salida un archivo con extensión .archimate el cual es la representación gráfica de los componentes relevantes para una arquitectura empresarial en el dominio de aplicaciones. La herramienta se desarrolló bajo una metodología apropiada y fue probada a través de un caso de estudio en donde se encontró que además de documentar un BASELINE de arquitectura, puede mantener actualizada la documentación de los sistemas de información con artefactos basados en Archimate. Sin embargo, una desventaja de la herramienta es que no permite visualizar las interacciones entre sistemas de información, lo cual se plantea como trabajo para desarrollar a futuro en otras tesis de maestría o doctorado.

Palabras clave: Archimate, Arquitectura Empresarial, Ingeniería inversa dirigida por modelos, TOGAF.

Abstract

The development of an Enterprise Architecture BASELINE requires a significant effort depending of size Company, officers knowledge and skills of Enterprise architects; the main goal of this research is to develop a MDRE tool called JTOGAF to provide BASELINE documentation in the Enterprise application domain. JTOGAF uses as input Java source code that meets the J2EE standard and generates as output a file with extension .archimate which is the graphical representation of the relevant components for an Enterprise architecture in the applications domain. The tool was developed under an appropriate methodology and was tested through a case study where we found that in addition to document an architectural BASELINE, you keep updated the documentation of information systems with artifacts based in Archimate. However, a disadvantage of this tool is that it does not allow view the interaction between information systems, which is proposed as a future work in a doctorate study.

Keywords: Archimate, Enterprise Architecture, Model Driven Reverse Engineering, TOGAF.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

1. Introducción

La arquitectura empresarial busca la alineación entre las tecnologías de información y los objetivos estratégicos de una empresa con el objetivo de hacerla más competitiva; al ser un gran reto, una buena práctica es el uso de un marco de trabajo o modelo de referencia entre los cuales se destaca The Open Group Architecture Framework TOGAF.

TOGAF es un marco de trabajo que proporciona un enfoque a través de un método iterativo e incremental llamado Architecture Development Method ADM en donde existe un enfoque de diseño, planeación, implementación y gobierno de TI haciendo análisis en cuatro dominios:

- Dominio de Negocio: Define la estrategia de negocios, procesos de la organización, misión, visión y gobierno.
- Dominio de Aplicaciones: Provee blueprints de los sistemas de información que posee la organización, sus relaciones y de los sistemas que se requieren implantar para alcanzar los objetivos del negocio.
- Dominio de Datos: Describe la estructura de los datos de la organización, tanto físicos como lógicos y la gestión de los mismos.
- Dominio de Tecnología: Provee la estructura de hardware, software y redes que se requieren para soportar las aplicaciones o sistemas de información de la empresa.

En cada uno de los dominios, se obtiene un baseline de arquitectura el cual consta de varios workproducts que representan el estado actual de una empresa; posteriormente y tomando como base los objetivos de la empresa, se realiza un target el cual consta de varios workproducts que representan el cómo debería ser el dominio para que esté alineado a los otros dominios. Con el baseline y target de arquitectura se realiza un análisis GAP el cual consiste en analizar desde el baseline de arquitectura la forma de llegar al target para cada uno de los dominios teniendo en cuenta que deben estar alineados entre sí; para ello se puede ver la necesidad de crear, modificar, actualizar o eliminar componentes de arquitectura en cada uno de los dominios.

Obtener un baseline de arquitectura empresarial es un trabajo que puede tomar años dependiendo del tamaño de la empresa y de la cantidad de información que se obtenga; por lo tanto, en un proyecto de arquitectura empresarial existen roles especializados en cada uno de los dominios como el Arquitecto de Negocio que se dedica a analizar el dominio de Negocio, el Arquitecto de Aplicaciones o Arquitecto de Software que analiza

Aplicaciones y Datos, el Arquitecto de Infraestructura que analiza el dominio de Tecnología y el Arquitecto Líder que coordina, orquesta e integra la información para cumplir el objetivo del proyecto.

Desde el punto de vista del arquitecto de Aplicaciones, la persona que desempeñe este rol debe tener conocimientos en Ingeniería de Software para analizar y obtener correctamente la información para los dominios de Aplicaciones y Datos; adicionalmente al momento de iniciar el levantamiento de información, debe identificar a las personas que poseen el conocimiento de los sistemas de información para entrevistarlas y obtener la mayor cantidad de información relevante que esté relacionada con el dominio de negocio. En este proceso pueden surgir dificultades en caso tal que en la empresa en donde se está ejecutando el proyecto de arquitectura empresarial exista una alta rotación del personal y/o haya documentación insuficiente o inexistente.

En el mercado existe variedad de herramientas que permiten realizar ingeniería inversa a sistemas de información utilizando su código fuente; generalmente el resultado de esta transformación queda representado en el lenguaje de modelado UML; sin embargo, resulta complicado integrar los artefactos generados en este lenguaje con los dominios de negocio y tecnología ya que UML [1] está diseñado para procesos de arquitectura de software desde el punto de vista del desarrollo de aplicaciones. Por otro lado, The Open Group ha desarrollado un lenguaje de modelado de arquitectura empresarial el cual se integra perfectamente con el marco de trabajo TOGAF, sin embargo se realizaron varias revisiones bibliográficas en donde se observó que existen herramientas que permiten generar documentación semi-automática de arquitectura empresarial con código fuente de aplicaciones pero no se encontró específicamente que transforme código Java en documentación de arquitectura empresarial, por lo tanto se ve la necesidad de una herramienta tecnológica que ayude al arquitecto de aplicaciones en esta labor.

En este trabajo de investigación se analiza esta necesidad y como resultado se desarrolla la herramienta Java2Archimate, la cual es un plugin para el IDE Eclipse en la versión Luna en donde se utilizaron conceptos de Model Driven Reverse Engineering MDRE, lenguaje Java y herramientas como Xtend [2] y Modisco [3]. La herramienta desarrollada permite transformar un proyecto Java en artefactos del lenguaje de modelado Archimate, generando un archivo con extensión .archimate el cual puede ser visualizado con la aplicación Archi.

1.1 Capítulos de este trabajo

Este trabajo está dividido en 5 capítulos. En el capítulo 1 se realiza una introducción del presente trabajo de investigación y se menciona como el problema de investigación es resuelto a través de la consecución de los objetivos propuestos. En el capítulo 2 se

describe la motivación de la investigación, tomando de base un marco de referencia conceptual; también se describe el estado del arte relacionando los temas Model Driven Engineering (MDE), Model Driven Reverse Engineering (MDRE), Archimate y TOGAF; se estructura el problema de investigación junto con una hipótesis de trabajo, lo anterior se propone validar a través de 1 objetivo general y 4 objetivos específicos. En el capítulo 3 se explica todo sobre Java2Archimate herramienta resultante del presente trabajo de investigación y de la librería JTOGAF_9-1_v1.jar. En el capítulo 4 se realiza una prueba de concepto explorando dos casos de estudios para evaluar la funcionalidad de la herramienta desarrollada. En el capítulo 5 se realiza un análisis a los datos obtenidos en la prueba de concepto, se describen los beneficios y restricciones encontrados de la herramienta JTOGAF y finalmente unas conclusiones generales y trabajo futuro.

2. Planteamiento del problema

2.1 Motivación

Originalmente la investigación surgió como una alternativa para las organizaciones que cuentan con un recurso humano limitado y necesitan desarrollar el listado de proyectos de software determinados en el Roadmap de una Arquitectura Empresarial; en este tipo de situaciones algunas organizaciones optan por realizar Outsourcing [4] lo cual puede resultar costoso si no se tienen claramente definidas, analizadas, diseñadas y especificadas las necesidades en términos de un lenguaje de modelado como por ejemplo UML; esto sin contar que dependiendo de la complejidad de los proyectos se puede requerir realizar interventoría al Outsourcing para controlar que se están cumpliendo los objetivos y las normas de calidad definidas, lo cual incrementa los costos y puede influir en el éxito o fracaso de los proyectos de software. [5] [6] [7]

Para esta problemática, se planteó una propuesta en la cual se pretendía analizar cada Workproduct correspondiente al Target de las fases B, C y D de TOGAF para obtener de cada uno de ellos una especificación de requerimientos de software que permitiera a través de una herramienta de transformación de Modelo a Código generar una aplicación funcional con base en el Roadmap y la lista de proyectos definitiva obtenida de la fase E de TOGAF; todo lo anterior motivado en trabajos de investigación orientados a Model Driven Architecture y frameworks de Arquitectura Empresarial [8] [9].

Se encontró que en la mayoría de las veces y dependiendo de la experticia del arquitecto empresarial el Target de las fases B, C y D en TOGAF se plantea a un nivel muy general sin tener en cuenta detalles particulares de los sistemas de información, esto debido a que el objetivo principal de la arquitectura empresarial es generar un

diagnóstico con el cual se obtiene un listado de proyectos a desarrollar para alinear las tecnologías de información con los objetivos del negocio de la organización y no el desarrollo de software[9]. Por lo general se utiliza Archimate para modelar la información del baseline y target; sin embargo con la información del modelo en Archimate no es posible generar software automáticamente. [10]

Teniendo en cuenta la conclusión obtenida y el gran interés por parte del autor del presente trabajo de investigación por integrar los conocimientos de Model Driven Reverse Engineering y TOGAF para apoyar el desarrollo de arquitecturas empresariales; se realizó una retroalimentación de los resultados de la investigación original con el profesor Henry Roberto Umaña Acosta de la Universidad Nacional de Colombia y el candidato a Magister de la Universidad Nacional de Colombia Jhon Alexander Cruz Castelblanco dando como resultado una nueva propuesta de investigación concluyéndose en el presente trabajo.

2.2 Marco teórico

La Arquitectura Empresarial es un marco de referencia que permite a las empresas articular las tecnologías de información y comunicación con las estrategias de negocio [11]; esto con el fin de aumentar la productividad, mejorar la competitividad y soportar la toma decisiones con respecto a la tecnología.[1] Desde esta perspectiva se contemplan los siguientes dominios:

- Arquitectura de Negocio
- Arquitectura de Información
- Arquitectura de Aplicaciones
- Arquitectura de Tecnología

[1] Curso Arquitectura de Software, Especialización en Ingeniería de Software, Profesor Jorge Mario Calvo, Universidad Distrital Francisco José de Caldas, Bogotá – Colombia 2011.

La arquitectura empresarial es la integración de la tecnología (TICs Tecnologías de la información y comunicaciones) con las estrategias y objetivos del negocio de una forma integral. [12]

Para construir una arquitectura empresarial se deben tener en cuenta los siguientes artefactos:

- Objetivos del Negocio
- Información y Conocimiento
- Procesos y Personas
- Software
- Infraestructura de TIC's.

Para desarrollar una arquitectura empresarial en cualquier organización, las mejores prácticas indican que se debe utilizar un Framework o marco de trabajo reconocido y probado, esto debido a que permite agilizar y simplificar la definición y el desarrollo de la arquitectura asegurando un cubrimiento más completo de la solución diseñada y porque diseñar una arquitectura empresarial es un proceso técnicamente complejo. En la Ilustración 1 se observan los diferentes Frameworks de arquitectura empresarial desde sus orígenes hasta su evolución.

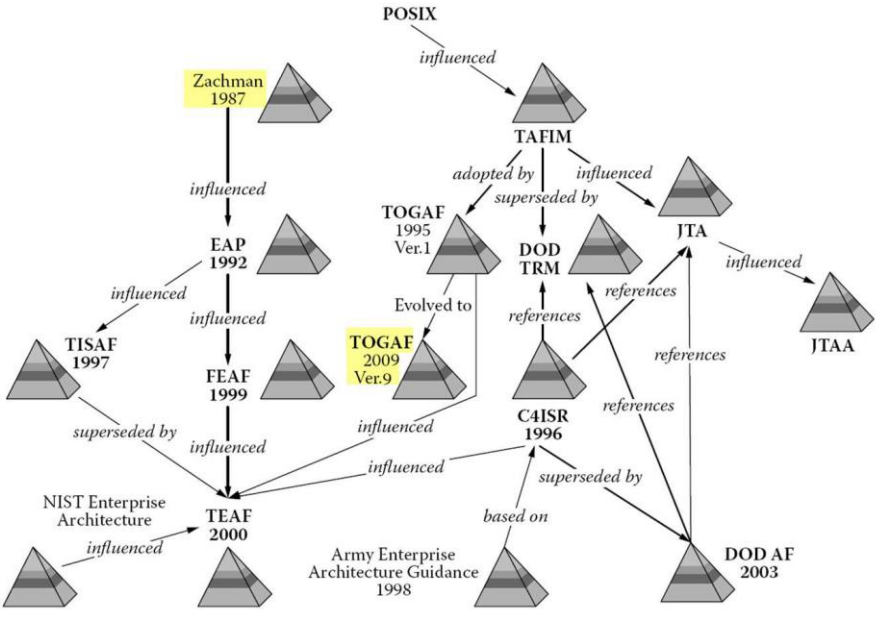


Ilustración 1 Árbol genealógico de Frameworks de Arquitectura Empresarial [15]

Los marcos de trabajo o frameworks de arquitectura empresarial más reconocidos a nivel mundial son Zachman [13] [14] por su antigüedad y.TOGAF; sin embargo, el uso de cualquier Framework de Arquitectura Empresarial dependerá de las necesidades de las organizaciones. [16]

Para el presente trabajo de investigación, se tomará como base TOGAF debido a que fue desarrollado por el mismo consorcio que el lenguaje de modelado de arquitectura empresarial Archimate.

2.2.1 TOGAF

The Open Group Architecture Framework es un framework o marco de trabajo para trabajar la arquitectura empresarial en el cual se trabaja la estructura de organización de los elementos de la arquitectura con entregables para obtener elementos que permitan alinear las TIC con los objetivos estratégicos de una organización [15].

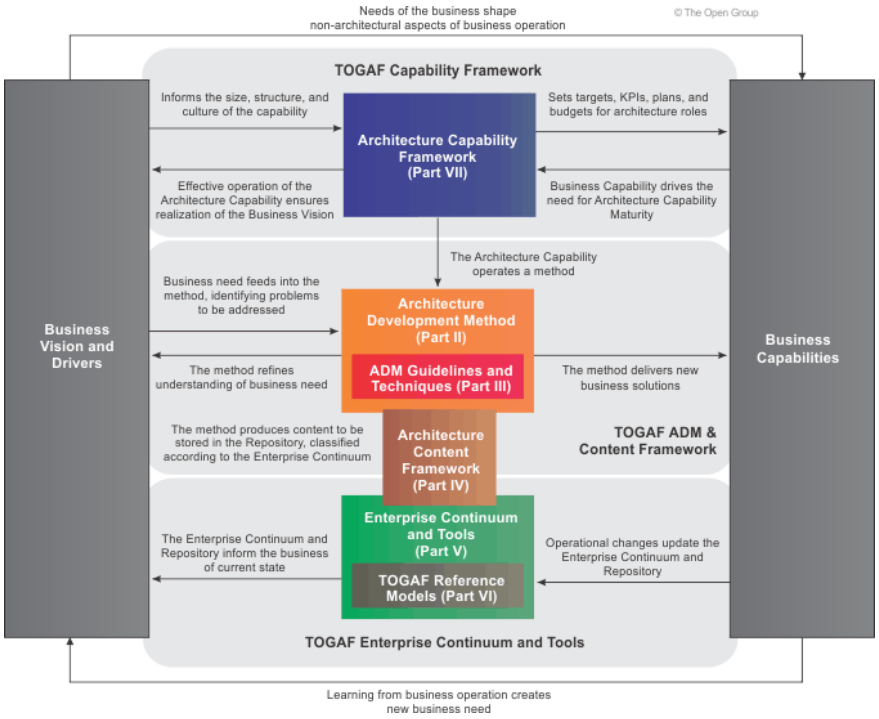


Ilustración 2 Estructura de TOGAF[2]

[2] Tomado de: http://pubs.opengroup.org/architecture/togaf9-doc/arch/

En términos generales, el Framework define unos dominios con los cuales se pretende contener un conjunto de herramientas tecnológicas que permitan unificar el lenguaje para mantener una metodología estándar, estos subsistemas son:

- Arquitectura de Negocio: Especifica las estrategias y procesos importantes del negocio de la organización o de la empresa.
- Arquitectura de Datos: Especifica la forma en que se deben administrar los datos del negocio.
- Arquitectura de Aplicaciones: Especifica las interacciones que existen entre las aplicaciones de software (Sistemas) del negocio.
- Arquitectura de Tecnología: Especifica los componentes de Hardware, Software, Comunicaciones, Redes e Infraestructura necesaria para soportar el Core del Negocio.

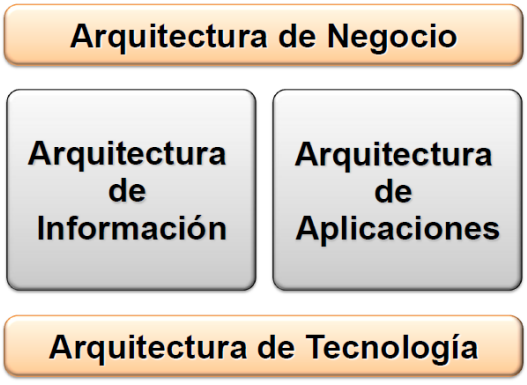


Ilustración 3 Dominios de la Arquitectura Empresarial, Fuente: Adaptado de [16]

El marco de trabajo TOGAF, de acuerdo a la Ilustración 2, está compuesto de varias secciones las cuales se detallan a continuación:

- **Part I: Introduction**: Se muestran conceptos claves y se explica la estructura del documento de TOGAF.

- **Part II: ADM**: Es el método que utiliza TOGAF para desarrollar arquitecturas empresariales. El Architecture Development Method (ADM) es una metodología compuesta de 8 fases las cuales se trabajan de forma iterativa e incremental para

el desarrollo de una arquitectura empresarial; el método puede ser ajustado y personalizado de acuerdo a las necesidades de las organizaciones. Ver Ilustración 4.

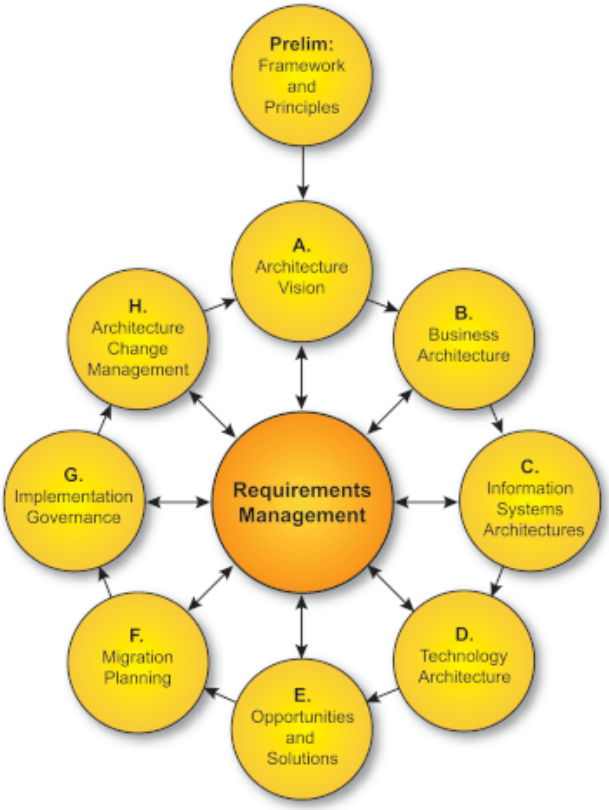


Ilustración 4 Architecture Development Method, Fuente: Tomado de The Open Group

Las fases de TOGAF ADM de acuerdo a la Ilustración 4 son:

- A: Visión de la Arquitectura
- B: Arquitectura de Negocio
- C: Arquitecturas de los Sistemas de Información
- D: Arquitectura de Tecnología
- E: Oportunidades y Soluciones
- F: Planeación de la Migración.
- G: Implementación de Gobierno
- H: Arquitectura de Gestión del Cambio.

El ADM se utiliza para definir el estado actual del negocio con respecto a los subsistemas de la arquitectura empresarial de software, esta parte es llama AS – IS; posteriormente se plantea una visión, unos objetivos en donde se describe como deberían ser los subsistemas de la arquitectura empresarial de software para cumplir con los objetivos estratégicos del negocio, esta parte es llama TO – BE. Para pasar del AS – IS al TO – BE deben definirse una serie de proyectos y desarrollarse teniendo en cuenta el ADM para alinear las TICs con las estrategias del negocio [12].

- **Part III: Guidelines and Techniques**: Se dan a conocer guías y técnicas que pueden ser aplicadas en el desarrollo del ADM.

- **Part IV: Architecture Content Framework:** Es un marco de trabajo contenido dentro de TOGAF, el cual permite clasificar y ordenar las salidas de cada una de las fases del ADM de TOGAF en 3 categorías denominadas Workproducts [17]:

    - **Artefacto**: Describe un aspecto de la arquitectura; un artefacto puede ser un catálogo, matriz o diagrama.

    - **Entregable**: Es un producto de trabajo que ha sido contractualmente especificado y debe haber sido revisado formalmente, acordado y firmado por los interesados.

    - **Building Block**: Componente reutilizable de arquitectura; puede representar componentes de negocio, IT o capacidades de arquitectura.

Para representar Building Blocks existen diferentes lenguajes de modelado; el más conocido y que se integra perfectamente con TOGAF es Archimate [18] debido a que es desarrollado por el mismo consorcio (The Open Group) y con el objetivo de que permita describir diferentes dominios de arquitectura dentro del ciclo del ADM.

- **Part V: Enterprise Continuum and Tools:** En esta sección se habla de un repositorio de arquitectura, la idea principal es buscar la forma de almacenar toda la documentación generada y hacerla de fácil acceso; el Enterprise Continuum está clasificado en:
    - Architecture Continuum: Ilustra cómo se desarrolló y evolucionó la arquitectura a través de un continuo que inicia desde arquitecturas fundacionales como la proporcionada por TOGAF, hasta llegar al punto de madurez de una Arquitectura especifica de organización.

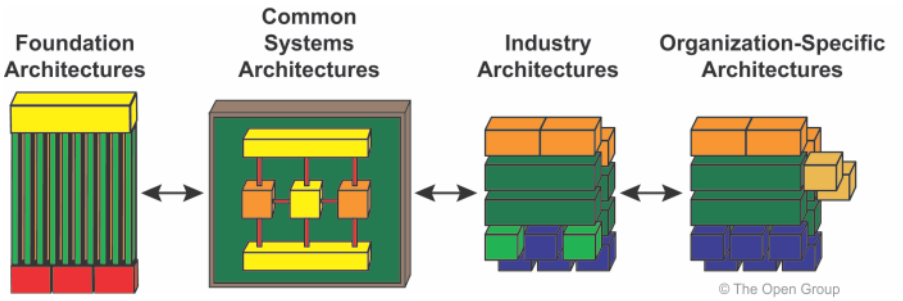


Ilustración 5 Architecture Continuum, Tomado de [19]

    - Solution Continuum: Representa la especificación detallada y construcción de las arquitecturas en los niveles correspondientes del Architecture Continuum. Cada nivel de detalle, está compuesto de Building Blocks que representan la solución en dicho nivel.

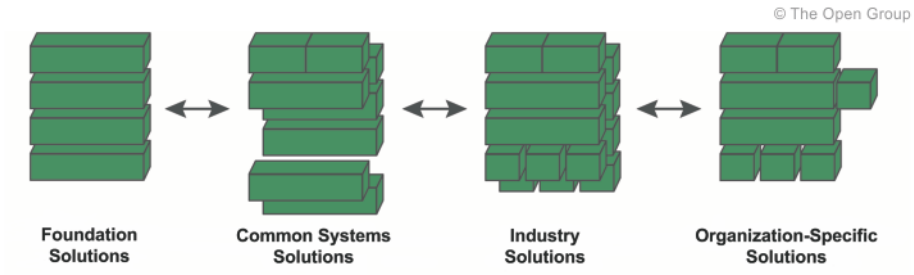


Ilustración 6 Solution Continuum, Tomado de [19]

- **Part VI: Reference Models:** Modelos de referencia de arquitecturas fundacionales (Ver el Enterprise Continuum).

- **Part VII: Architectue Capability Framework:** Es un marco de trabajo que permite establecer el nivel de madurez de arquitectura empresarial de una organización; es útil para las fases preliminar y A del ADM de TOGAF.

TOGAF es ampliamente utilizado y cuenta con 2 tipos de certificaciones:

1. TOGAF 9 Foundation con un total de 16415 personas certificadas
2. TOGAF 9 Certified con un total de 38156 personas certificadas

En total, a la fecha hay 54571 personas certificadas.[3]

2.2.2 Archimate

Es un lenguaje de modelado desarrollado por The Open Group [20] el cual permite representar una arquitectura empresarial visualmente. Este lenguaje está alineado con el marco de trabajo TOGAF y por lo tanto sus elementos permiten representar los dominios de Negocio, Aplicaciones, Tecnología y sus correspondientes interacciones para las fases B, C y D del ADM definido por TOGAF. [21]

[3] Tomado de https://togaf9-cert.opengroup.org/certified-individuals, consultado el 3 de Abril de 2016.

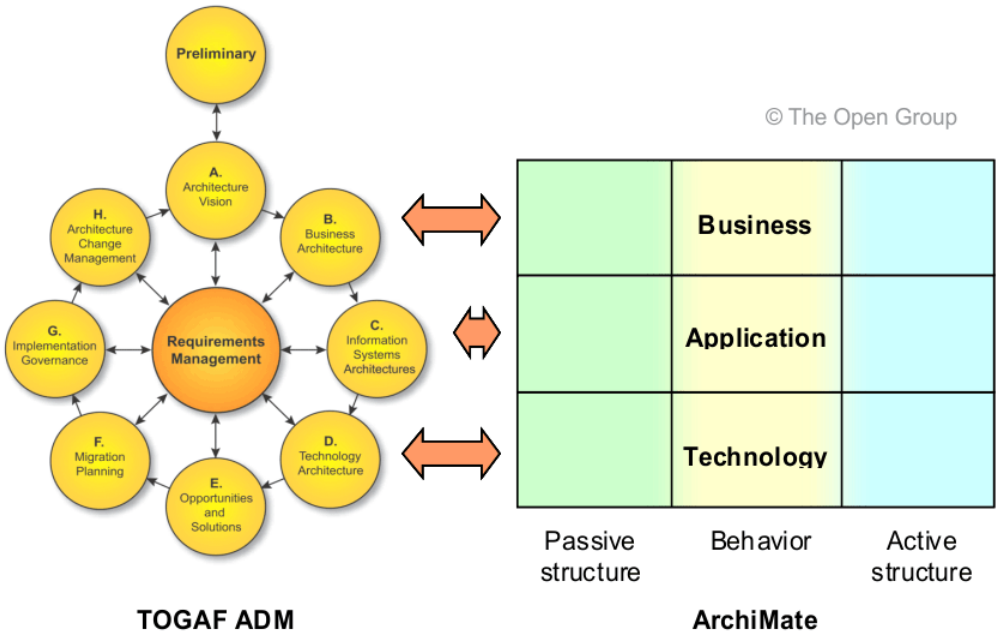


Ilustración 7: Correspondencia entre Archimate y TOGAF[4]

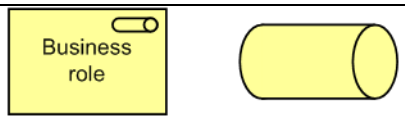
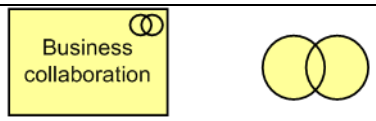
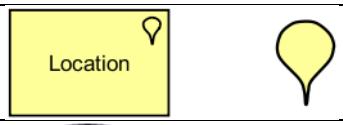
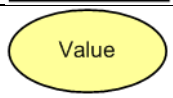
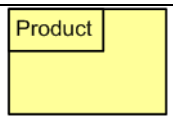
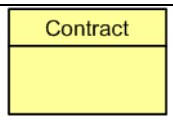
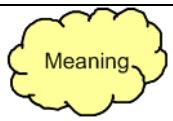
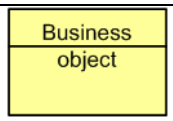
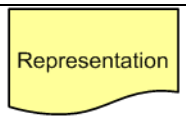
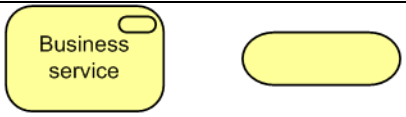
Archimate caracteriza sus elementos en tres tipos: Estructura pasiva, comportamiento y estructura activa; en cada uno de los dominios definidos se maneja un concepto similar para cada tipo; a continuación se describen los elementos del lenguaje de modelado Archimate basado en el tipo y dominio al que pertenece.

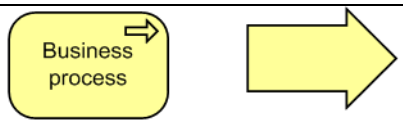
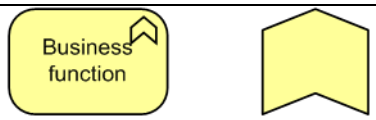
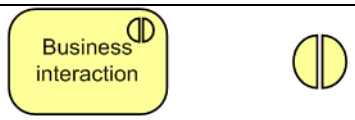
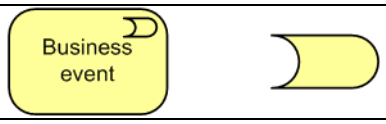
- Dominio de Negocio

Tabla 1 Elementos Archimate del Dominio de Negocio, tomado y traducido al español de [22]

Tipo	Nombre	Descripción	Representación
Estructura Activa	**Business Actor**	Entidad organizacional que es capaz de realizar un comportamiento.	Business actor 
	Business Interface	Punto de acceso donde un "**Business Service**" está disponible para uno o varios ambientes.	Business interface 

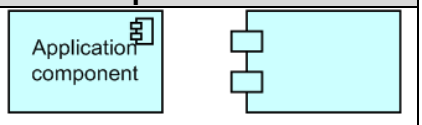
[4] Tomado de: http://pubs.opengroup.org/architecture/archimate2-doc/chap02.html#_Toc371945144. – Figure 7: Correspondence between Archimate and TOGAF

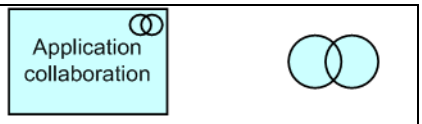
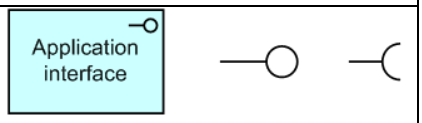
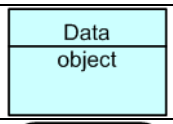
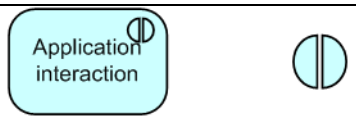
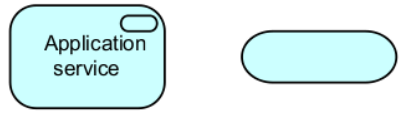
	Business Role	Responsabilidad de realizar un comportamiento específico, se le puede asignar un actor específico.	
	Business Collaboration	Agregación de dos o más "**Business Role**" que trabajan junto para realizar un comportamiento colectivo.	
	Location	Punto conceptual o representación de una ubicación.	
Estructura Pasiva	**Value**	Valor relativo, utilidad o importancia de un "**Business Service**" o "**Product**".	
	Product	Colección coherente de servicios acompañados de un "**Contract**". Un "**Product**" se ofrece a clientes (internos o externos).	
	Contract	Una especificación formal o informal de acuerdos en donde se especifican los derechos y obligaciones relacionados a un "**Product**"	
	Meaning	Conocimiento o experiencia presente de un "**Business Object**" en un contexto particular.	
	Business Object	Elemento pasivo el cual obtiene relevancia desde una perspectiva empresarial.	
	Representation	Forma de percibir la información contenida en un "**Business Object**".	
Comportamiento	**Business Service**	Servicio que satisface una necesidad de negocio para un cliente (Interno o Externo).	

	Business Process	Elemento que agrupa comportamientos basados en un conjunto ordenado de actividades. Su intención es producir definiciones para "**Product**"(s) o "**Business Service**"(s).	
	Business Function	Elemento que agrupa comportamientos basados en un conjunto seleccionado de criterios (típicamente requerimientos de recursos empresariales y/o competencias).	
	Business Interaction	Describe el comportamiento que tiene un "**Business Collaboration**".	
	Business Event		

- Dominio de Aplicaciones y Datos

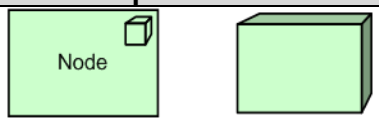
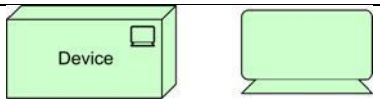
Tabla 2 Elementos Archimate del Dominio de Aplicaciones, tomado y traducido al español de [23]

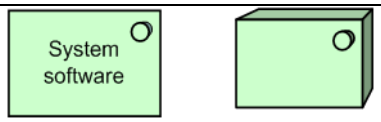
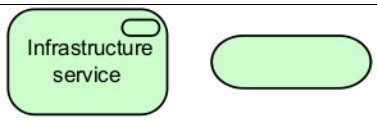
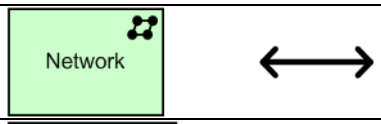
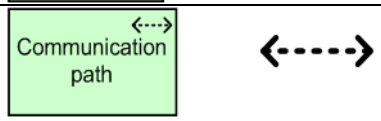
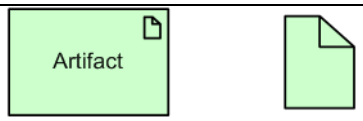
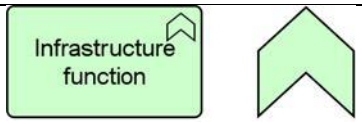
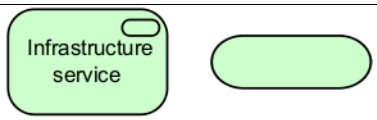
Tipo	Nombre	Descripción	Representación
Estructura Activa	**Application Component**	Parte modular, desplegable y reemplazable de un sistema de software que encapsula comportamiento y datos exponiéndolos a través de un conjunto de interfaces.	

	Application Collaboration	Conjunto de dos o más "**Application Component**" que trabajan juntos para realizar un comportamiento colectivo.	
	Application Interface	Punto de acceso en donde un **"Application Service"** está disponible para un usuario u otro "**Application Component**".	
Estructura Pasiva	**Data Object**	Elemento pasivo idóneo para procesos automatizados.	
Comportamiento	**Application Function**	Elemento que agrupa comportamiento automatizado que puede realizar un "**Application Comonent**"	
	Application Interaction	Elemento que describe el comportamiento de un "**Application Collaboration**"	
	Application Service	Un servicio que expone comportamiento automatizado.	

- Dominio de Tecnología

Tabla 3 Elementos Archimate del Dominio de Tecnología, tomado y traducido al español de [24]

Tipo	Nombre	Descripción	Representación
Estructura Activa	**Node**	Recurso computacional en donde los "**Artifact**" pueden ser almacenados o desplegados para su ejecución.	
	Device	Recurso de hardware en donde los "**Artifact**" pueden ser	

		almacenados o desplegados para su ejecución.	
	System Software	Ambiente de software para específicos tipos de objetos y componentes que pueden desplegados en forma de "**Artifact**".	
	Infraestructure Interface	Punto de acceso en donde los servicios de infraestructura ofrecidos por un "**Node**" pueden ser accedidos por otros "**Node**" y "**Application Component (Ver Dominio de Aplicaciones)**"	
	Network	Medio de comunicación entre dos o más "**Device**"	
	Communication Path	Enlace entre dos o más "**Node**", a través del cual pueden intercambiar datos.	
Estructura Pasiva	**Artifact**	Pieza física de datos que se utiliza o produce en un proceso de desarrollo de software o por la implementación y operación de un sistema.	
Comportamiento	**Infraestructure Function**	Elemento que agrupa comportamiento de infraestructura que puede ser realizado por un "**Node**".	
	Infraestructure Service	Unidad de funcionalidad visible externamente, provista por uno o más "**Node**" expuesta a través de interfaces bien definidas y	

		significativas para el ambiente.	

Como se observa en la Ilustración 7, Archimate integra todos sus elementos con el marco de trabajo TOGAF; adicionalmente existen unas extensiones del lenguaje que permiten modelar otras fases del ADM; para el caso de las fases E, F y G hay una extensión llamada “Implementation & Migration” [25] y para las fases Preliminar, H, A y Gestión de Requerimientos hay otra extensión llamada “Motivation” [26]; sin embargo, en un trabajo de arquitectura empresarial existen elementos que no se puedan cubrir o documentar completamente con Archimate; estos elementos son:

- Objetivos, principios y requerimientos
- Riesgo y Seguridad
- Gobierno
- Políticas y reglas de negocio
- Costos
- Rendimientos
- Tiempo
- Planeación y evolución

2.2.3 Conceptos relacionados en Model Driven

Existen varios acrónimos que hacen referencia a diferentes conceptos en el ámbito de Model Driven; estos conceptos fueron desarrollados por un consorcio internacional denominado Object Management Group OMG [27]. En la Ilustración 8 se puede observar la relación y su correspondiente prelación entre estos conceptos.

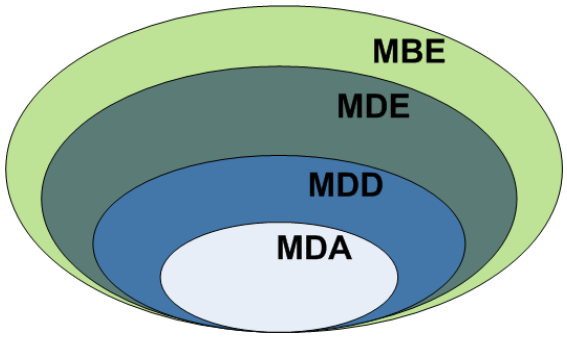


Ilustración 8 MBE vs MDE vs MDD vs MDA[5]

[5] Modeling-languages, disponible en: http://modeling-languages.com/clarificando-conceptos-mbe-vs-mde-vs-mdd-vs-mda/

- **MBE Model Based Engineering:** Describe un proceso de desarrollo de software donde los modelos juegan un papel importante pero no son los motores del desarrollo. [28]

- **MDE Model Driven Engineerig:** Va más allá de las actividades del proceso de desarrollo de software, teniendo en cuenta la evolución del sistema. [29]

- **MDD Model Driven Development:** Es un paradigma de proceso de desarrollo de software en donde los modelos son el principal insumo para construir software. (Generar código de forma semiautomática con los modelos, estándares OMG). [30]

- **MDA Model Driven Architecture:** Visión particular de la OMG (Object Management Group) con respecto a MDD (Se basa en el uso de los estándares de la OMG). [31]

- **MDRE Model Driven Reverse Engineering**: Se busca obtener un modelo a partir del código fuente de una aplicación. Para profundizar más en este tema, ver la sección 2.2.4.

2.2.4 Model Driven Reverse Engineering

Para definir Model Driven Reverse Engineering MDRE se requiere contextualizar sobre los siguientes conceptos:

- Ingeniería Inversa (Reverse Engineering): Es el proceso de comprensión de software a través de la generación de un modelo de un alto nivel de abstracción, adecuado para la documentación, mantenimiento o reingeniería. La ingeniería inversa ha demostrado ser útil en proyectos en donde no se posee documentación y a partir del código fuente se pueda entender la composición y estructura del mismo. [32]
- Modelo (Model): Es una representación de alto nivel de algunos aspectos de un sistema de software. [33]

En un proceso clásico de ingeniería de software, posterior a la etapa de levantamiento y especificación de requerimientos, se procede a diseñar el sistema antes de construirlo, inclusive existen herramientas capaces de generar automáticamente el código fuente del sistema partiendo de un modelo especifico de plataforma PSM [9] y posteriormente se puede realizar unas transformaciones hasta llegar al código fuente en un lenguaje especifico de programación. Sin embargo, existen numerosos sistemas de software que se desarrollaron y que actualmente funcionan sin haber pasado por una fase de diseño; por lo anterior al momento de que un ingeniero encargado de realizar reingeniería o mantenimiento requiere modificar algunos componentes del sistema, se convierte en un proceso complejo debido al desconocimiento de la arquitectura del software. [34]

Model Driven Reverse Engineering es una orientación metodológica diseñada para superar estas dificultades. MDRE utiliza las características de las tecnologías de modelado pero las aplica de forma inversa, es decir, a partir del código fuente generar un modelo.

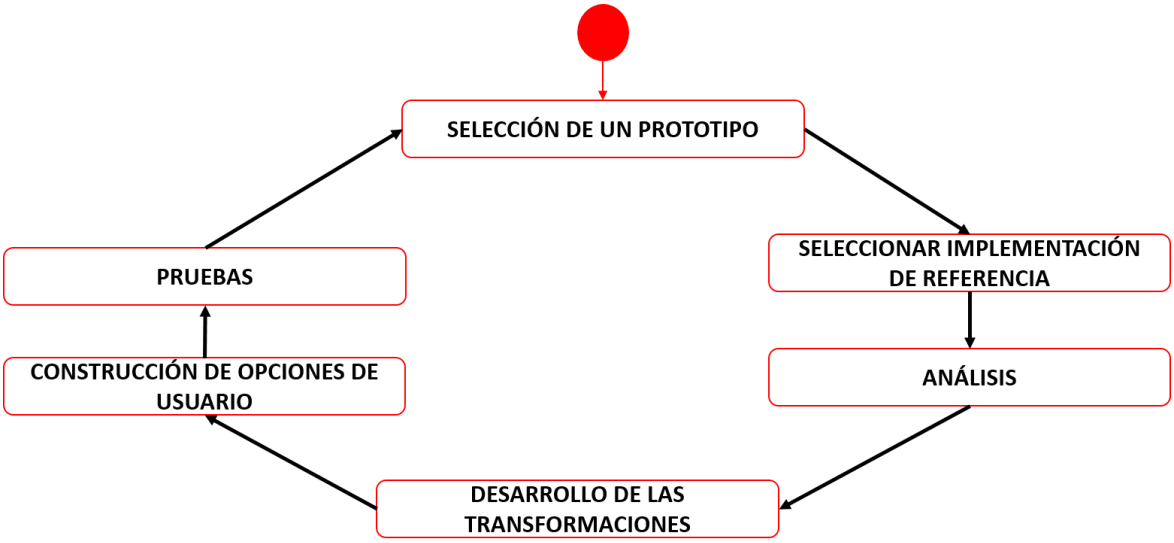


Ilustración 9 Metodología MDRE, adaptado de [35]

La metodología MDRE utilizada para el desarrollo de la presente investigación está compuesta de 6 fases y fue adaptada de [35], a continuación se define cada una de las fases de la metodología:

- **Selección de un prototipo:**

  Esta fase consiste en la selección de un prototipo que permita realizar las diferentes transformaciones planteadas en la metodología.

- **Seleccionar implementación de referencia:**
  Esta fase consiste en la selección de una implementación de destino; es decir, el modelo destino al cual se quiere llegar desde el prototipo.

- **Análisis**
  Esta fase consiste en analizar y conceptualizar tanto el modelo seleccionado como prototipo como la implementación de referencia. Posterior a la conceptualización, se debe definir los elementos del modelo origen que pueden ser convertidos a elementos del modelo destino.

- **Desarrollo de las transformaciones**
  En esta fase se debe realizar el desarrollo de las transformaciones planteadas en la fase anterior; para ello se requieren habilidades en programación de computadores y manejo de herramientas que faciliten la obtención de cada uno de los modelos.

- **Construcción de opciones de usuario**
  Esta fase consiste en la programación o desarrollo de la interfaz gráfica que tendrá el usuario final para el uso de la herramienta Model Driven Reverse Engineering desarrollada. Es muy importante la amigabilidad de las opciones para que su uso sea claro.

- **Pruebas**
  Esta fase consiste en el desarrollo de varias pruebas utilizando el prototipo seleccionado en la fase 1. Se debe tener un registro de cada uno de los errores encontrados en una lista para posteriormente solucionarlos.

2.3 Estado del arte

Se realizó búsqueda de información relacionada con Model Driven Engineering (MDE), Model Driven Reverse Engineering (MDRE), Archimate y TOGAF con el objetivo de revisar

todos los trabajos previos; las conclusiones de este trabajo se pueden ver reflejadas en la Tabla 4.

Tabla 4: Estado del arte MDE, MDRE, Archimate, TOGAF y arquitectura empresarial

Año	Referencia	TEMA INVOLUCRADO				Descripción
		Archimate	TOGAF o Arquitectura Empresarial	MDE	MDRE	
2014	[36]		X	X	X	Se propone una metodología para estandarizar la documentación que poseen las empresas a través de un meta-modelo de documentación arquitectura empresarial el cual sería implementado en el repositorio de Arquitectura Empresarial; la idea es reducir el esfuerzo de mantener actualizada la documentación haciéndolo un proceso semiautomático.
2013	[37]				X	Se plantea la existencia de modelos de rendimiento para aplicaciones JEE en donde la idea es mejorar su performance; sin embargo el esfuerzo para desarrollar dichos modelos es muy grande, por lo tanto en el artículo se propone una forma para obtenerlo automáticamente.
2013	[38]	X				Guía de buenas prácticas para modelar con Archimate. En la guía se presentan preguntas que puede afrontar un Arquitecto Empresarial en el ámbito práctico y una solución.
2013	[39]			X		Se presenta una metodología para modelar un Computation Independent Model para que posteriormente este pueda ser transformado a un Platform Independent Model. Se puede hacer una relación del CIM con la forma de obtener los Workproducts en TOGAF y posiblemente siguiendo la metodología se pueda generar el modelo PIM el cual con una herramienta de transformación se puede convertir a código,
2012	[40]		X	X	X	En el artículo se presentan los principales retos de la generación automática de documentación de arquitectura empresarial a través de un ejemplo, revisión bibliográfica y encuesta a 123 expertos.
2012	[41]	X	X	X	X	Se propone la documentación automática de arquitectura empresarial realizando una recopilación automática de datos de la infraestructura tecnológica de una red utilizando una herramienta de exploración y posteriormente mostrarla en Archimate.
2012	[42]	X				Especificación oficial de The Open Group para la versión 2.0 de Archimate.
2012	[43]	X	X	X		Generación de un modelo en Archimate realizando transformaciones de código a un Enterprise Service Bus SAP PI.
2012	[44]	X		X		Se presenta una propuesta de transformar modelos DEMO (Design & Engineering Methodology for Organizations) los cuales sirven para representar procesos de negocio junto con e3value los cuales son modelos que sirven para reducir la brecha entre grupos de negocio y TI, a Archimate.
2012	[45]	X	X	X		Uso de archivos XML resultantes de las salidas de herramientas de análisis de vulnerabilidad de sistemas de información a través de la red para generar un modelo Archimate.
2011	[46]	X	X			El artículo se enfoca en cómo Archimate complementa a TOGAF en la documentación que se genera en el desarrollo de una arquitectura empresarial dentro del ADM.
2010	[47]	X	X	X		Se plantea una forma de crear y personalizar nuevos

Año	Referencia	TEMA INVOLUCRADO				Descripción
		Archimate	TOGAF o Arquitectura Empresarial	MDE	MDRE	
						Viewpoints en Archimate utilizando un enfoque de Model Driven Engineering. El resultado de la investigación es una propuesta de extender Archimate.
2010	[48]	X	X			Se presenta la relación que hay entre Archimate y TOGAF y el valor que aporta al documentar una arquitectura empresarial para el análisis GAP, los Baseline y Targets.
2009	[49]	X		X		Se presenta una tesis en donde se exploran las posibilidades de transformar desde el enfoque planteado por el marco de trabajo IAF Integrated Architecture Framework a Archimate llevando estos dos conceptos a metamodelos y posteriormente realizar un match para identificar los elementos que se pueden transformar. Parte del objetivo de la tesis planteada es verificar si el marco de trabajo se puede desarrollar utilizando Archimate directamente.
2007	[50]				X	Se presenta un método para la transformación de modelos utilizando Model Driven Reverse Engineering. En el artículo se realiza un ejemplo con un software de cartografía en donde se toma un modelo semántico y se transforma a un modelo simbólico.
2006	[51]	X	X	X		Los documentos XML hacen parte de los Workproducts de una arquitectura empresarial, se propone a partir de estos documentos generar modelos en Archimate utilizando una herramienta de transformación de código llamada RML (Rule Markup Language)

De los artículos encontrados en la revisión bibliográfica, los más relevantes debido a la relación directa con el tema de investigación fueron los siguientes: Artículo [36], [40], Articulo [41], Articulo [43]. En la Tabla 5 se realiza una comparación.

Tabla 5 Artículos relevantes Estado del Arte vs Tesis

Alternativa estado del arte	Propuesta articulo estado del arte	JAVA2ARCHIMATE (Tesis)
[36] Farwick Matthias, Schweda Christian, Breu Ruth, Hanshke Inge. (2014).	• Mantenimiento automático de arquitectura empresarial con el uso de plantillas estandarizadas en todos los dominios. • Aborda todo los dominios, pero requiere configuración de meta-	• Genera automáticamente documentación de Arquitectura Empresarial únicamente en el dominio de aplicaciones sin necesidad de utilizar plantillas.

	modelo para extracción de datos automáticos.	• No requiere configuraciones de meta-modelo.
[40] Hauder Matheus, Matthes Florian, Roth Sascha. (2012).	• Retos en obtención automática de documentación de AE.	• Se tienen en cuenta los retos mencionados y con estos se delimita alcance de la investigación realizada en el dominio de aplicaciones.
[41] Holm Hannes, Buschle Markus, Legerstrom Robert, Ekstedt Mathias. (2012).	• Documenta automáticamente el dominio de infraestructura. • Documenta en Archimate.	• Documenta automáticamente el dominio de aplicaciones. • El estándar de documentación es Archimate, lo cual permite la integración de la salida resultante con el articulo [41]
[43] Buschle Marcus, Grunow Sebasatian, Mathes Florian, Ekstedt Mathias, Hauder Matheus, Roth Sascha. (2012).	• Documenta el dominio de aplicaciones utilizando información de en ESB de SAP.	• Documenta el dominio de aplicaciones a partir de sistemas de información desarrollados en Java. •

2.4 Formulación del problema

Teniendo en cuenta el contexto descrito en el presente documento y la información acerca del uso de TOGAF a nivel mundial descrito en las páginas 6 y 7, se plantea realizar una investigación sobre documentación automática de arquitectura empresarial en el dominio de aplicaciones representada en un modelo Archimate para sistemas de información desarrollados en lenguaje Java. Se propone en un modelo Archimate debido a que este se integra perfectamente con el framework TOGAF tal y como se visualiza en la Ilustración 7.

2.5 Pregunta de investigación

¿Cómo obtener documentación automática de arquitectura empresarial en el dominio de aplicaciones a partir de los sistemas de información existentes en una organización?

2.6 Hipótesis de trabajo

Es posible obtener documentación automática de arquitectura empresarial en Archimate para el dominio de aplicaciones, con una herramienta Model Driven Reverse Engineering (MDRE) que obtenga la información requerida de un código fuente escrito en lenguaje Java.

2.7 Objetivo General

Desarrollar una propuesta para transformar un proyecto Java en un modelo Archimate del dominio de aplicaciones utilizando Model Driven Reverse Engineering.

2.8 Objetivos Específicos

1. Establecer una propuesta para representar elementos del metamodelo Java en elementos del modelo Archimate por medio de la conceptualización de los mismos.
2. Construir una forma de comunicación entre Java y Archimate para facilitar la identificación de los elementos Java a transformar.
3. Desarrollar una herramienta utilizando Model Driven Reverse Engineering.
4. Realizar una prueba de concepto.

2.9 Justificación de la Investigación

2.9.1 Aporte teórico

El documento resultado de la presente investigación permitirá continuar con el desarrollo de la herramienta descrita en la sección 3. Java2Archimate y será una base para

estudiantes de la Universidad que deseen continuar con el desarrollo y evolución de la misma.

2.9.2 Utilidad metodológica

La metodología utilizada en el presente proyecto de investigación servirá de base para futuros trabajos en el área de Model Driven Revserse Engineering.

2.9.3 Implicación práctica

El resultado de la presente investigación tiene una aplicación concreta en el área de la Arquitectura Empresarial en el dominio de aplicaciones debido a que la herramienta resultante de la presente investigación permitirá documentar automáticamente el baseline de arquitectura empresarial del dominio de aplicaciones y mantenerla actualizada bajo las condiciones descritas en el presente documento.

3. Java2Archimate

Java2Archimate es el nombre seleccionado para la herramienta desarrollada como resultado de este trabajo de investigación. Esta herramienta permite a la persona que tenga el rol de Arquitecto de Aplicaciones o Arquitecto de Software en un proyecto de arquitectura empresarial, obtener un baseline de arquitectura empresarial documentado en un modelo Archimate a partir del código Java de las aplicaciones de la organización que cumplan con el estándar JEE o que utilicen la librería JTOGAF_9-1_v1.jar la cual también es resultado del presente trabajo de investigación.

La intención de Java2Archimate es facilitar la labor del Arquitecto de Aplicaciones o Arquitecto de Software en la elaboración de documentación y aportando a la línea de investigación de generación automática de documentación para arquitectura empresarial.

3.1 Requerimientos

Los requerimientos obtenidos para el desarrollo de la herramienta están basados en la hipótesis de trabajo planteada en el presente proyecto de investigación; para ello se analiza literatura relacionada con la generación de documentación automática de arquitectura empresarial. Los requerimientos que se definieron para la herramienta son los siguientes:

1. **Transformación de un código Java a documentación en un modelo Archimate en el dominio de Aplicaciones**: Teniendo en cuenta que archimate es un lenguaje de modelado que permite documentar arquitectura empresarial, con este requerimiento se estaría aportando a la generación automática de arquitectura empresarial; sin embargo, la generación de la documentación sería solo para el dominio de aplicaciones; por lo tanto se verían 7 componentes para diferentes con sus respectivas relaciones para representar el modelo.

2. **Fácil de utilizar**: Los usuarios finales que utilizaran la herramienta serán los arquitectos de aplicaciones o de arquitectos de software que estén trabajando en proyectos de arquitectura empresarial; por lo tanto la herramienta permite que el arquitecto únicamente deba encargarse de verificar que la documentación

generada esté relacionada con el dominio de negocio que está analizando; el arquitecto deberá escoger las clases de código Java que requiere documentar basado en la orientación del arquitecto de negocio. Al finalizar el proceso, el arquitecto de negocio junto con el arquitecto de aplicaciones deberán crear manualmente las relaciones entre los dominios; posteriormente la documentación debe ser facilitada al arquitecto de infraestructura para completar la información del dominio de tecnología.

3. **Permitir la lectura de código Java independientemente del IDE en el que se haya desarrollado**: Una limitante de los proyectos desarrollados en lenguaje Java es que estos se abren fácilmente en el IDE en el cual fueron desarrollado; al tratar de abrir un proyecto Java en otro IDE, se generan errores y dolores de cabeza con la modificación de archivos de configuración; por lo tanto, se ve la necesidad de que la herramienta funcione sin que se requieran hacer mayores configuraciones.

4. **Orientación a Arquitectura Empresarial**: En un proceso de arquitectura empresarial se parte del dominio de negocio; teniendo en cuenta el segmento que se está trabajando, se deben trabajar los dominios de aplicaciones e infraestructura; por lo tanto, la herramienta desarrollada debe permitir la documentación de un conjunto de clases Java que el arquitecto de aplicaciones previamente seleccione alineado con el análisis realizado por el arquitecto de negocio.

3.2 Selección de prototipo

En la primera fase de la metodología Model Driven Reverse Engineering se buscó un proyecto Java muy sencillo que cumpliera con el estándar J2EE. Se encontró el proyecto DEMO Ticket-monster la cual es una aplicación que permite realizar reserva de tiquetes.

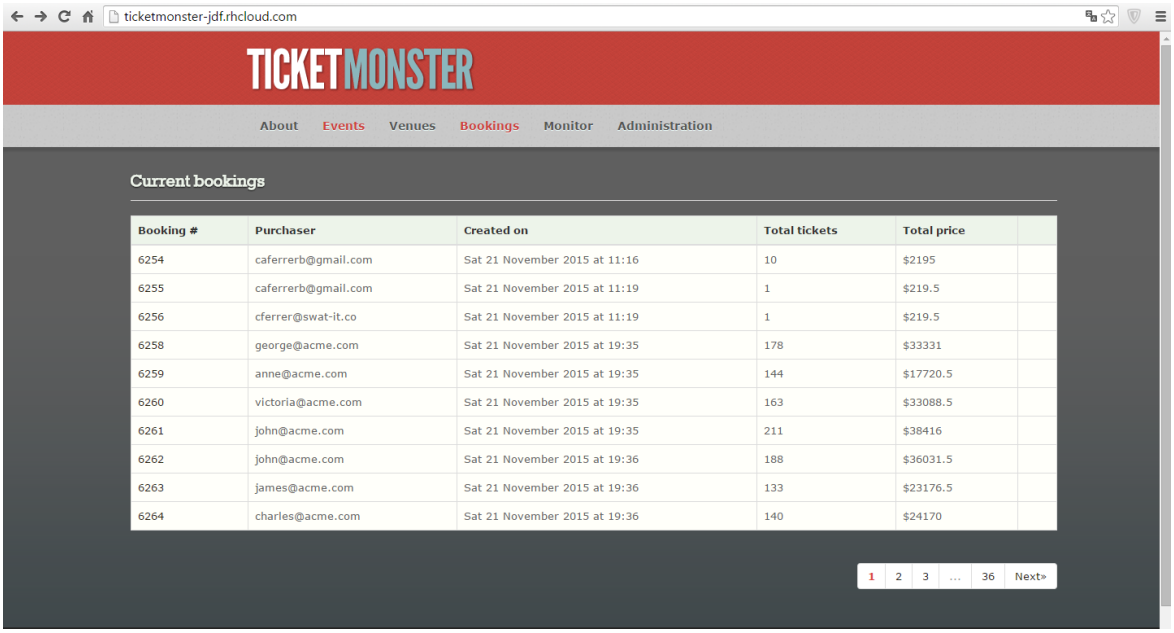


Ilustración 10 Vista Previa – Ticket-Monster Jboss en la nube

Esta aplicación se ejecuta en una arquitectura Web Cliente – Servidor y se despliega bajo el servidor de aplicaciones JBoss.

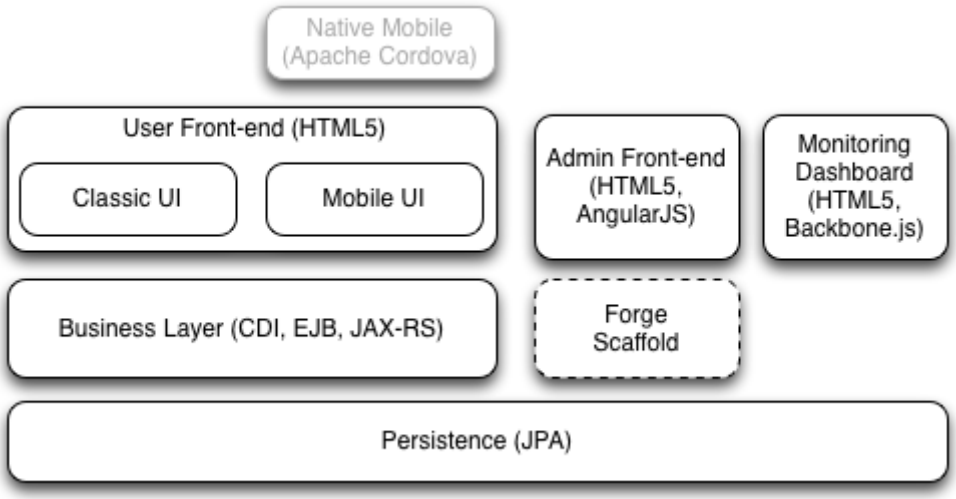


Ilustración 11 Arquitectura de Software Ticket-Monster[6]

[6] Tomado de: http://developers.redhat.com/ticket-monster/whatisticketmonster/#_architecture

Las tecnologías utilizadas en esta aplicación son HTML5 para la interfaz gráfica o Vista, para la capa de negocio se utilizan Enterprise Java Beans EJB 3.1, sin embargo el código de la aplicación refleja que los EJB se identifican con las anotaciones @EJB dentro del mismo proyecto WEB y no hay un proyecto EJB independiente. Para la parte de persistencia se utiliza JPA 2.0. La base de datos por defecto es H2.

Las fuentes del proyecto Ticket-Monster vienen organizadas en 3 carpetas:

Carpeta Java: Contiene 6 paquetes con sus correspondientes clases organizadas de la siguiente forma:

- Controller
  - MemberController.java
    - 84 Líneas de código
- Data
  - MemberListProducer.java
    - 54 Líneas de código
  - MemberRepository.java
    - 60 Líneas de código
- Model
  - Member.java
    - 93 Líneas de código
- Rest
  - JaxRsActivator.java
    - 33 Líneas de código
  - MemberResourceRESTService.java
    - 185 Líneas de código
- Service
  - MemberRegistration.java
    - 45 Líneas de código
- Util
  - Resources.java
    - 58 Líneas de código

Para un total de 612 Líneas de código correspondientes al lenguaje Java.

Carpeta resources: Contiene un archivo XML con la configuración de JPA con la tecnología Hibernate y un archivo SQL llamado import con una línea SQL para realizar una inserción en la tabla Member.

Carpeta WebApp: Contiene el archivo Index.xhtml; adicionalmente contiene la configuración para el uso de la tecnología JSF junto con archivos de estilos css e imágenes que se utilizan en la vista principal (Archivo index.xhtml).

Por otro lado, se encuentra un componente de pruebas el cual contiene una carpeta llamada test, que contiene una clase llamada MemberRegistrationTest.java la cual contiene 66 líneas de código y su finalidad es realizar una prueba unitaria, que consiste en ingresar un miembro a través de la clase MemberRegistration.java utilizando la herramienta JUnit.

Al compilar el proyecto se genera un archivo .war el cual se carga en el servidor de aplicaciones JBoss para ser desplegado y ejecutado.

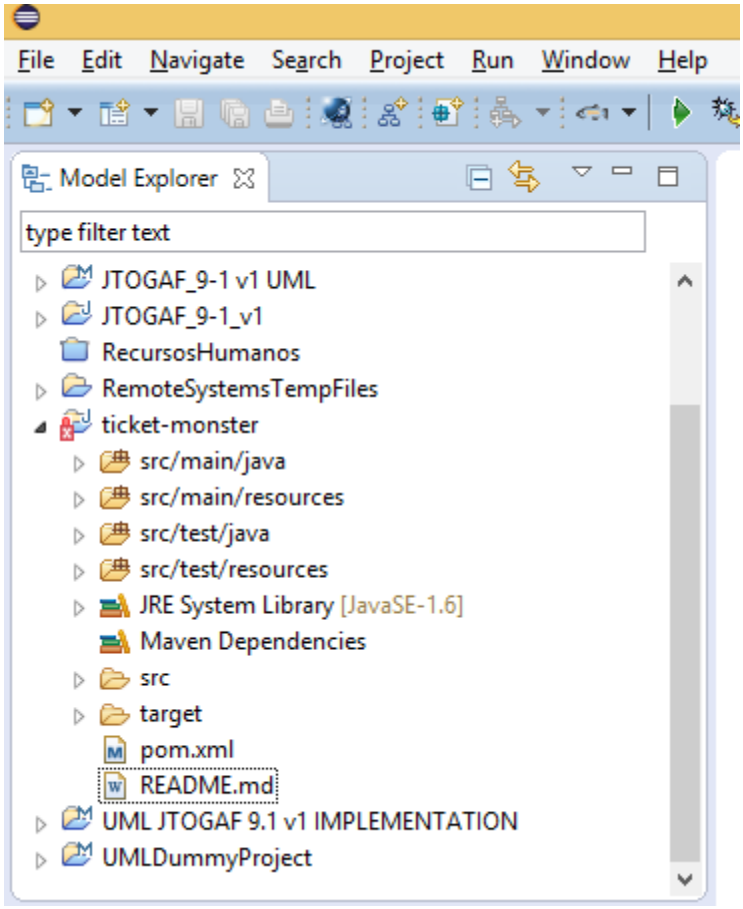


Ilustración 12 Workspace con proyecto Ticket-monster cargado

Es importante resaltar que para las pruebas efectuadas con la herramienta Java2Archimate, se modificó el proyecto ticket-monster implementando el uso de la librería JTOGAF_9-1_v1.jar en nuevas clases con el objetivo de visualizar en la salida la totalidad

de los elementos archimate en el dominio de aplicaciones; por lo anterior, la carpeta Java del proyecto quedó estructurada de la siguiente forma:

- Controller
  - MemberController.java
  - *TicketController.java*: Clase que implementa la anotación @ApplicationInterface de la librería JTOGAF.
- Data
  - MemberListProducer.java
  - MemberRepository.java
- Model
  - Member.java
  - *Reservation.java*: Implementa anotaciones JPA
  - *Ticket.java*: Implementa la anotación @DataObject de la librería JTOGAF.
- Rest
  - JaxRsActivator.java
  - MemberResourceRESTService.java
- Service
  - MemberRegistration.java
  - *TicketBooking.java:* Implementa la anotación @ApplicationComponent de la librería JTOGAF.
  - *TicketRegistration.java:* Implementa anotaciones J2EE
- Util
  - Resources.java

3.3 Implementación de referencia

La implementación de referencia corresponde al modelo de salida que se obtendrá del proceso de transformación del prototipo seleccionado; para el presente proyecto de investigación, la implementación de referencia es el dominio de aplicaciones del lenguaje de modelado para arquitecturas empresariales Archimate.

Para representar computacionalmente el modelo en archimate, el resultado de la transformación será un archivo con extensión .archimate con una estructura basada en XML; para la visualización del archivo se utiliza la herramienta Archi.

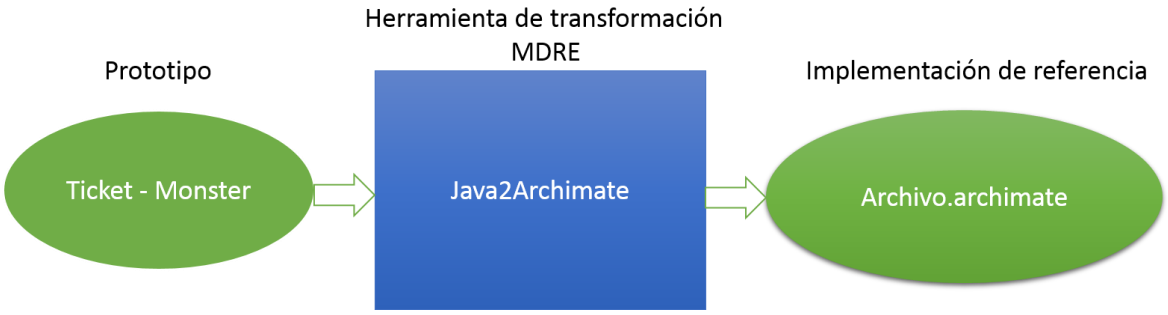


Ilustración 13 Proceso general de transformación del prototipo a la implementación de referencia

3.4 Análisis

En la fase de análisis de la metodología, se identifican elementos del modelo correspondiente al prototipo que puedan ser transformados en elementos de la implementación de referencia; para ello se conceptualizan cada uno de los modelos a través de una tabla en donde se identifique su correspondencia.

Para la elaboración de la Tabla 7, se identificaron los elementos esenciales del modelo del proyecto Java ticket-monster basado en el estándar J2EE; para ello se utilizó una utilidad llamada ModelDiscovery de la herramienta Modisco. Por otro lado, se revisó la definición para cada uno de los elementos definidos en el dominio de aplicaciones de Archimate y se comparó con las etiquetas J2EE.

Durante el análisis se detectó que cuando un proyecto Java no cumpla con el estándar J2EE, las transformaciones a Archimate en el dominio de aplicaciones serán imposibles; por lo tanto se desarrolló JTOGAF_9-1_v1.jar. La cual es una librería desarrollada en Java la que contiene un conjunto de anotaciones que permiten representar los elementos de la capa de aplicaciones de archimate al incluir dicha librería en un proyecto Java.

3.4.1 Estructura de un archivo .Archimate

Un archivo Archimate está basado en el lenguaje XML; a continuación se indica su correspondiente estructura:

Tabla 6 Conceptualización del lenguaje de representación de un modelo Archimate

Etiqueta	Definición
<?xml version="1.0" encoding="UTF-8?>	Define la versión de XML a utilizar en el archivo y la codificación para los caracteres.
<archimate:model xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:archimate="http://www.archimatetool.com/archimate" name="C:\carlos\prueba" id="570928ca" version="3.1.1"> </archimate:model>	Etiqueta cuerpo que contiene la definición de los elementos Archimate a representar. **Atributos:** • Xmlns: Sirve para importar librerías. • Name: Indica la ruta en disco en donde está almacenado el archivo. • Id: Identificador del archivo de 8 caracteres. • Versión: Indica la versión del software Archi con el que se puede leer el archivo; se deja por defecto la 3.1.1.
<folder name="Business" id="60b5f463" type="business"/> <folder name="Application" id="56d708ac" type="application"/> <folder name="Technology" id="12d9c5a2" type="technology"/> <folder name="Motivation" id="532defd8" type="motivation"/> <folder name="Implementation & Migration" id="996cf5af" type="implementation_migration"/> <folder name="Connectors" id="22764ac2" type="connectors"/> <folder name="Relations" id="a5fc2571" type="relations">	Etiqueta que define una carpeta para organizar los elementos Archimate por dominios de arquitectura empresarial. Por defecto siempre se tienen 8 carpetas: • Business: Elementos correspondientes al dominio de negocio. • Application: Elementos correspondientes al dominio de aplicaciones. • Technology: Elementos correspondientes al dominio de tecnología. • Motivation: Elementos que permiten representar artefactos de las fases H, A, Preliminar y Gestión de Requerimientos del ADM de TOGAF. • Implementation & Migration: Elementos que permiten

	representar artefactos de las fases E, F y G del ADM de TOGAF. • Connectors: Conectores que se utilizan. • Relations: Elementos de tipo relación para representar asociaciones entre elementos. • Views: En esta carpeta se almacenan los Viewpoints del proyecto Archimate; para la presente investigación, se deja por defecto un viewpoint para el dominio de aplicaciones. **Atributos:** • Name: Nombre que se le desea dar a la carpeta. • Type: Se debe asignar de acuerdo a los elementos que se desean instanciar según los nombres de las carpetas por defecto. • Id: Identificador de la carpeta de 8 caracteres.
<element xsi:type="archimate:ApplicationInteraction" id="ai000001" name="processTickets"/>	Etiqueta que permite declarar un elemento Archimate. Esta etiqueta debe estar dentro de una etiqueta <folder> **Atributos:** • Xsi:type: Indica el elemento archimate a instanciar. Depende de la carpeta en la que se esté definiendo. • Id: Identificador del elemento compuesto de 8 caracteres. • Name: Nombre del elemento.
<child xsi:type="archimate:DiagramObject" id="aiv00002" textAlignment="2" targetConnections="cal20006 " archimateElement="ai000001"/>	Etiqueta que debe ser utilizada dentro de una etiqueta <element> que este contenida en la etiqueta <folder> con atributo type=”diagrams”. Permite representar gráficamente un elemento Archimate instanciado en

	alguno de los <folder> diferentes al type=”diagrams”. **Atributos:** • xsi:type=”archimate:DiagramObject”: representa un elemento a pintar en el folder Views. • Id: Identificador del elemento compuesto de 8 caracteres. • textAlignment: Permite alinear el texto de izquierda a derecha con un valor numérico. • targetConnections: Identificador de 8 caracteres de los elementos instanciados asociados a través de una Relación al elemento visual. • archimateElement: Identificador de 8 caracteres del elemento Archimate previamente instanciado que se está visualizando.
<bounds x="150" y="0" width="100" height="100"/>	Etiqueta que debe estar contenida en una etiqueta <child>. Permite asignar un tamaño y ubicación en la pantalla al elemento Archimate que se está visualizando con la etiqueta <child>. **Atributos:** • x: Valor numérico que representa una posición en el eje horizontal para ubicar el elemento. • y: Valor numérico que representa una posición en el eje vertical para ubicar el elemento. • width: Valor numérico que representa el ancho del elemento. • height: Valor numérico que representa el alto del elemento.

<sourceConnection xsi:type="archimate:Connection" id="car20004" source="aiv00002" target="dov00003" relationship="car00004"/>	Etiqueta que permite representar una conexión o relación con otro elemento Archimate. Esta etiqueta debe estar contenida en la etiqueta <child> y el valor del atributo source debe ser igual al identificador del elemento <child>. **Atributos:** • xsi:type=”archimate:Connection”: Indica que se va a representar gráficamente una relación o asociación entre dos elementos archimate. • id: Identificador del elemento gráfico compuesto de 8 caracteres. • source: Origen desde donde se realiza la relación; en este campo debe ir el id del elemento <child> o elemento que representa gráficamente otro elemento. • target: Destino a donde se realiza la relación o asociación; en este campo debe ir el id del elemento <child> o elemento que representa gráficamente otro elemento. • relationship: identificador del tipo de asociación que previamente debido haber sido instanciado en el <folder> correspondiente al type = “relations”.

3.4.2 Estructura de J2EE

En el estándar de Java 2 Enterprise Edition, aparece el concepto de anotación. Una anotación permite adicionar funcionalidad o comportamiento a una clase a través de un llamado a una interface con el operador “@”.

Se pueden clasificar las anotaciones utilizadas en el estándar J2EE de acuerdo al uso que se le quiera dar; en el caso de los Enterprise Java Bean se puede realizar a través de las siguientes anotaciones:

- @EJB: Define un EJB; posterior a la definición, se le debe asignar el comportamiento.

Comportamiento de un EJB:

- @Stateless: El EJB tendrá un comportamiento en el cual no guardara estados.
- @Statefull: El EJB tendrá un comportamiento en el cual guardará los estados.

Por otro lado, para el manejo de vistas con tecnología JSF o para exponer interfaces a través de servicios, se utilizan los ManagedBeans

- @ManagedBean: Define un ManagedBean; posterior a la definición, se le debe asignar el comportamiento.

Comportamiento de los ManagedBeans:

- @ApplicationScoped: El bean estará vivo mientras la aplicación este en ejecución.
- @SessionScoped: El bean estará vivo mientras exista una sesión.
- @RequestScoped: El bean solo se creará cuando se realice una petición y su tiempo de vida será hasta que responda la petición.
- @ViewScoped: El bean estará vivo mientras se utilice la vista a la cual está relacionado.

Para identificar un bean que expone servicios a través de la tecnología de WebServices, se realiza a través de la etiqueta @WebService.

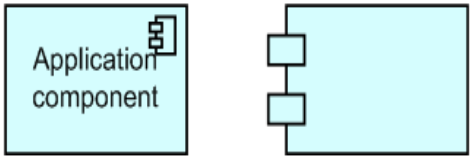
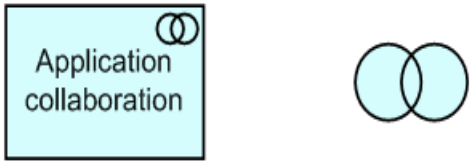
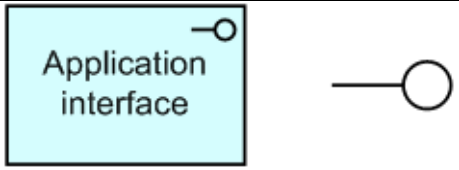
3.4.3 Java Persistence API

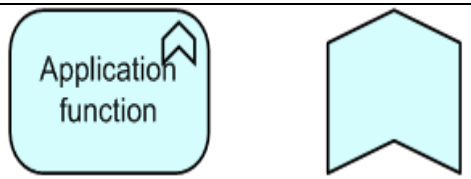
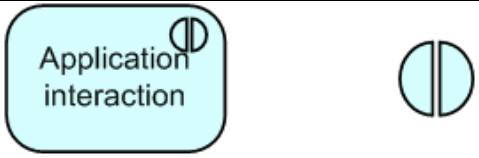
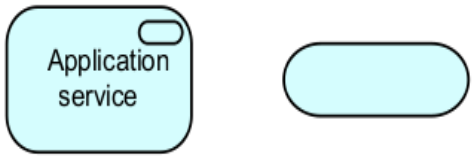
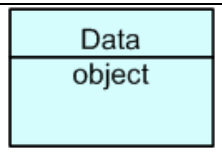
Java Persistence API o JPA, es un API independiente del estándar J2EE el cual posee un conjunto de anotaciones para identificar clases Java que manejaran la persistencia de una aplicación contra una base de datos. Una de las ventajas de esta API es que se puede trabajar en el lenguaje JPQL el cual es una variación del SQL y permite migrar de motor de base de datos fácilmente sin necesidad de modificar el código fuente; adicionalmente permite crear automáticamente las tablas y sus relaciones en el nuevo motor de base de datos.

Una entidad representa la estructura de una tabla de la base de datos; al instanciar el objeto durante la ejecución de una aplicación Java, puede contener la información de un objeto de datos. Para identificar un objeto entidad con JPA se realiza con la anotación @Entity.

3.4.4 Equivalencia entre J2EE y el dominio de aplicaciones Archimate

Tabla 7 Correspondencia entre Archimate en el Dominio de Aplicaciones y J2EE

No	ARCHIMATE		JAVA	
	CONCEPTO	NOTACIÓN	IDENTIFICACION	REGLAS
1	Application Component		J2EE Annotations: @EJB @Stateless @Statefull JTOGAF Annotations: @ApplicationComponent	En un proyecto Java Web JEE, se revisan las clases que posean las anotaciones @Stateless o @Statefull; dicha clase representará el elemento Archimate ApplicationComponent y el nombre de la clase será el nombre del componente.
2	Application Collaboration		J2EE Annotations: EJB con llamado de 2 o más EJBs. JTOGAF Annotations: @ApplicationCollaboration	Se realiza validando en el proyecto Java WEB JEE la clase EJB donde se realiza import de 2 o más componentes EJBs.
3	Application Interface		J2EE Annotations: @ApplicationScoped @SessionScoped @RequestScoped @ViewScoped @ManagedBean JTOGAF Annotations: @ApplicationInterface	Interface donde se llama la lógica de negocio para que pueda ser utilizada por un usuario o por otra aplicación. Un Bean que haga llamado a un EJB mapeado a elemento archimate como ApplicationComponent o ApplicationCollaboration. Un bean puede ser identificado por alguna de las anotaciones JEE que denotan el scope.

				También puede ser identificado un bean como un ManagedBean que sería el Backend de una vista JSF.
4	Application Function		J2EE Annotations: Función de una clase JAVA que ha sido mapeada a un elemento Archimate de tipo ApplicationComponent. JTOGAF Annotations: @ApplicationFunction	En un proyecto Java WEB JEE, se revisa una clase EJB que haya sido previamente identificada como ApplicationComponent y se verifican las funciones que hagan llamados a una clase que haya sido previamente identificada como DataObject.
5	Application Interaction		J2EE Annotations: Función de un EJB que hace llamados a 2 o más funciones de diferentes EJBs. JTOGAF Annotations: @ApplicationInteraction.	En un proyecto Java Web JEE, se revisa la clase que haya sido previamente identificada como ApplicationCollaboration y posteriormente se identifica la función que implementa los EJBs instanciados.
6	Application Service		J2EE Annotations: @WebService JTOGAF Annotations: @ApplicationService	En un proyecto Java Web JEE donde se haga uso de la anotación @WebService de la utilidad JAX-WS o @ApplicationService descrita en la librería JTOGAF9-1.jar
7	Data Object		JPA Annotations: @Entity JTOGAF Annotations: @DataObject	En un proyecto Java WEB JEE, se debe verificar lo siguiente: • Si el proyecto utiliza JPA, un entity el cual está identificado por el estándar JPA con la

				anotación @Entity puede representar un DataObject. • Si el proyecto no utiliza JPA, se debe utilizar la anotación • n @DataObject de la biblioteca JTOGAF.jar

3.5 Desarrollo de las transformaciones

El insumo principal para la fase de desarrollo fue la Tabla 7; por lo tanto se diseñó una estrategia partiendo desde la selección de las herramientas a utilizar, selección de las mismas en un workspace, diseño de la arquitectura de software y código fuente.

3.5.1 Herramientas utilizadas

Para el desarrollo de JTOGAF se utilizaron las siguientes herramientas:

- Sistema Operativo: Windows 7 Professional
  - Modisco: Es una herramienta que facilita realizar Model Driven Engineering o Model Driven Reverse Engineering debido a que genera un modelo de un proyecto de software siguiendo el estándar MOF (Meta Object Facility).

Ilustración 14 MODISCO Logo

Si se desea obtener más información sobre esta herramienta, se recomienda visitar los siguientes links:

- Sitio web oficial: https://eclipse.org/MoDisco/
- Foro: https://www.eclipse.org/forums/index.php/t/323226/
- Tutorial: https://www.youtube.com/watch?v=9PAspfzJn2E

- Xtend: Es un lenguaje de programación basado en Java el cual facilita la generación de código el cual puede ser almacenado en archivos con diferentes tipos de extensiones; para el caso del presente proyecto; se utiliza para generar archivos con estructura XML y extensión .archi.

Ilustración 15 Xtend Logo

Si se desea obtener más información sobre esta herramienta, se recomienda visitar los siguientes links:

- Sitio web oficial: http://www.eclipse.org/xtend/

- Tutorial: http://rtsys.informatik.uni-kiel.de/confluence/display/WS12EclPract/Code+Generation+with+Xtend
- Video Tutorial: https://www.youtube.com/watch?v=EzH5MPd13iI

- JDK 1.8: Java Development Kit en la versión 1.8

Ilustración 16 JDK 8

- IDE Eclipse versión Luna: Ambiente de desarrollo de software utilizado para integrar la herramienta desarrollada.

Ilustración 17 Eclipse Luna

- Archi: Herramienta que permite el modelado de building blocks en Archimate para arquitectura empresarial. Se utilizó para la visualización del resultado del código transformado a Archimate.

Ilustración 18 Archi versión 3.2.1

- JBoss-eap-6.3: Servidor de aplicaciones utilizado para el despliegue de la aplicación Ticket-Monster.

Ilustración 19 JBoss Logo

- J2EE: Java 2 Enterprise Edition; se utilizó para el funcionamiento de la aplicación Java Web Ticket Monster y conceptualización de las etiquetas propias de este estándar para las posteriores transformaciones al modelo Archimate en el dominio de aplicaciones.

- Coloso versión Icaro: Herramienta de modelado con licencia académica utilizada para la documentación de arquitectura de software de la herramienta Java2Archimate. Esta herramienta de modelado fue desarrollada por el profesor Ing Msc Sandro Bolaños de la Universidad Distrital Francisco José de Caldas en Bogotá – Colombia.

Ilustración 20 Coloso Versión Icaro

Se puede obtener más información de la herramienta en el siguiente link: http://www.colosoft.com.co/

3.5.2 Arquitectura de software

Java2Archimate se diseñó para trabajar bajo una plataforma Java utilizando el paradigma de programación orientada objetos; por lo anterior se realizó un diseño en donde se utilizaron técnicas de herencia, modularidad, encapsulamiento y polimorfismo.

- **Casos de Uso**

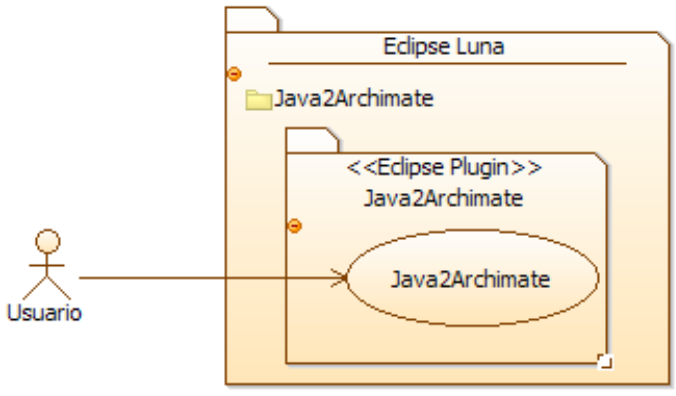


Ilustración 21 Diagrama de casos de uso Java2Archimate

Java2Archimate se desarrolló como plugin del IDE Eclipse Luna y el usuario solo tiene interacción con una opción llamada “Transformar”.

Tabla 8 Especificación caso de uso Transformar

Nombre	Java2Archimate	
Autor	Carlos Eduardo García Amaya	
Actores	Usuario	
Descripción	Transformar un proyecto Java en un archivo con extensión .Archimate.	
Pre-condición	1. El proyecto Java debe estar referenciado o importado en el IDE Eclipse Luna. 2. El proyecto Java debe estar abierto en el IDE Eclipse Luna. 3. Las clases del proyecto Java deben contener anotaciones de J2EE o implementar la librería JTOGAF_9-1_v1.jar. 4. El usuario debe tener instalado en su máquina el software Archi.	
Post-condición	1. Transformación exitosa 2. El archivo .archimate debe poderse abrir y visualizar con el software Archi.	
Flujo normal de eventos	**Actor**	**Evento**
	Usuario	1. Ejecutar el IDE Eclipse con el plugin Java2Archimate instalado
	Eclipse Luna (Sistema)	2. Mostrar las opciones al usuario.
	Usuario	3. Crear un proyecto Java e incluir las clases java que se desean documentar.
	Usuario	4. Seleccionar un proyecto, hacer clic en el menú Project y posteriormente en la opción Java2Archimate; o hacer clic derecho sobre el proyecto y seleccionar la opción Java2Archimate.
	Java2Archimate (Sistema)	5. Abrir ventana de selección de ruta de guardado y nombre de archivo.
	Usuario	6. Ingresar nombre al archivo de salida y la

		ubicación en el equipo.
	Java2Archimate (Sistema)	7. Mostrar resultado exitoso de la transformación.
	Usuario	8. Abrir el archivo generado con el software archi.
Flujo alternativo de eventos	Java2Archimate	En el paso 4. La opción Java2Archimate puede no aparecer habilitada si el usuario no selecciona un proyecto que este abierto en el IDE o si no selecciona la raíz del proyecto.
	Usuario	En el paso 6. El actor puede cancelar el proceso.
	Java2Archimate (Sistema)	En el paso 7 El resultado puede no ser exitoso por el no cumplimiento de una precondición.

- **Diagrama de componentes**

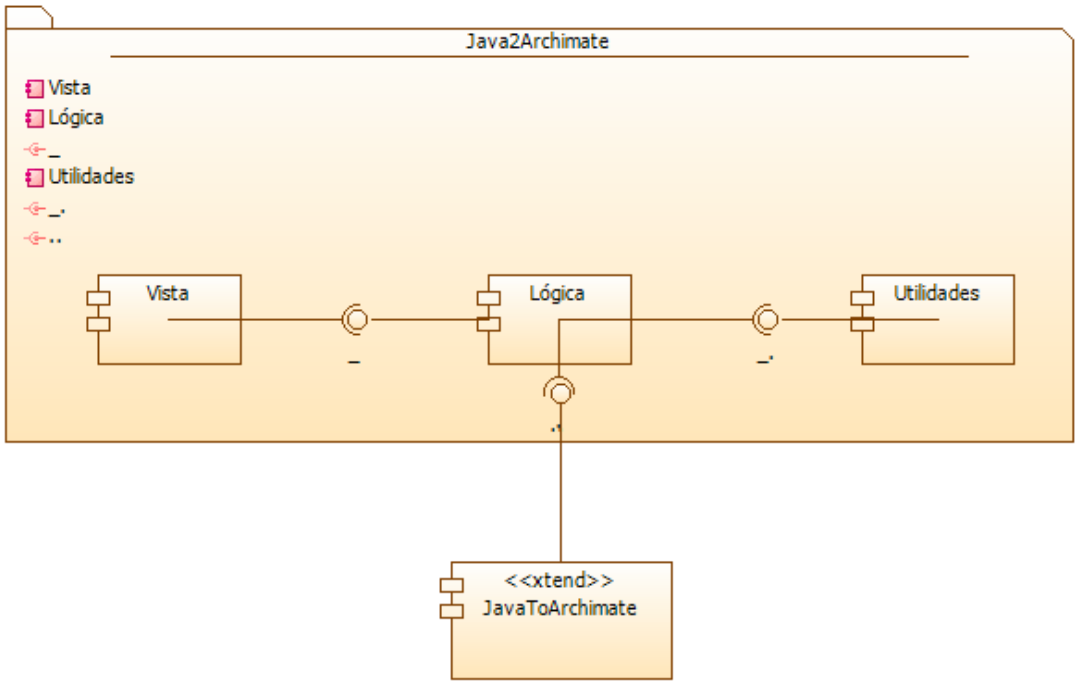


Ilustración 22 Diagrama de componentes Java2Archimate

Como se observa en la Ilustración 22, se definió un componente llamado JavaToArchimate el cual es una librería en lenguaje Xtend que facilita la generación de archivos a través de plantillas; una ventaja de este lenguaje es que al compilar sus clases, estas se transforman en archivos .class lo cual permite empaquetarlas en un archivo .jar facilitando la integración de las funcionalidades en un proyecto Java.

El archivo .jar se importa al proyecto Java2Archimate y se utiliza en el componente Lógica.

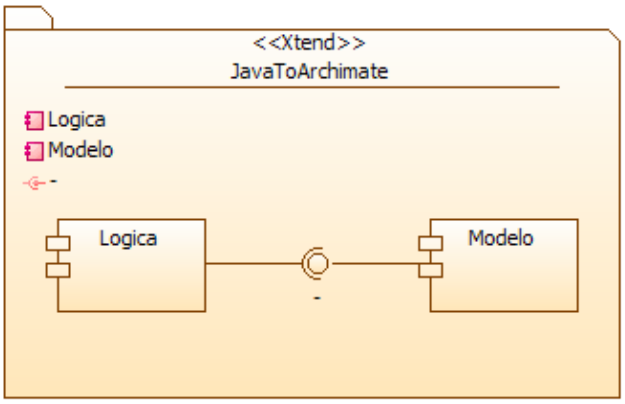


Ilustración 23 Diagrama de componentes JavaToArchimate

Cada componente descrito para el presente proyecto de investigación, es un proyecto de desarrollo independiente; es decir, el componente Java2Archimate es un proyecto Java que se desarrolló para ser integrado al IDE Eclipse Luna a través de un plugin; y el proyecto JavaToArchimate es un componente desarrollado en lenguaje Xtend, el cual permite y facilita la generación del archivo salida .archimate.

.

- **Diagrama de clases**

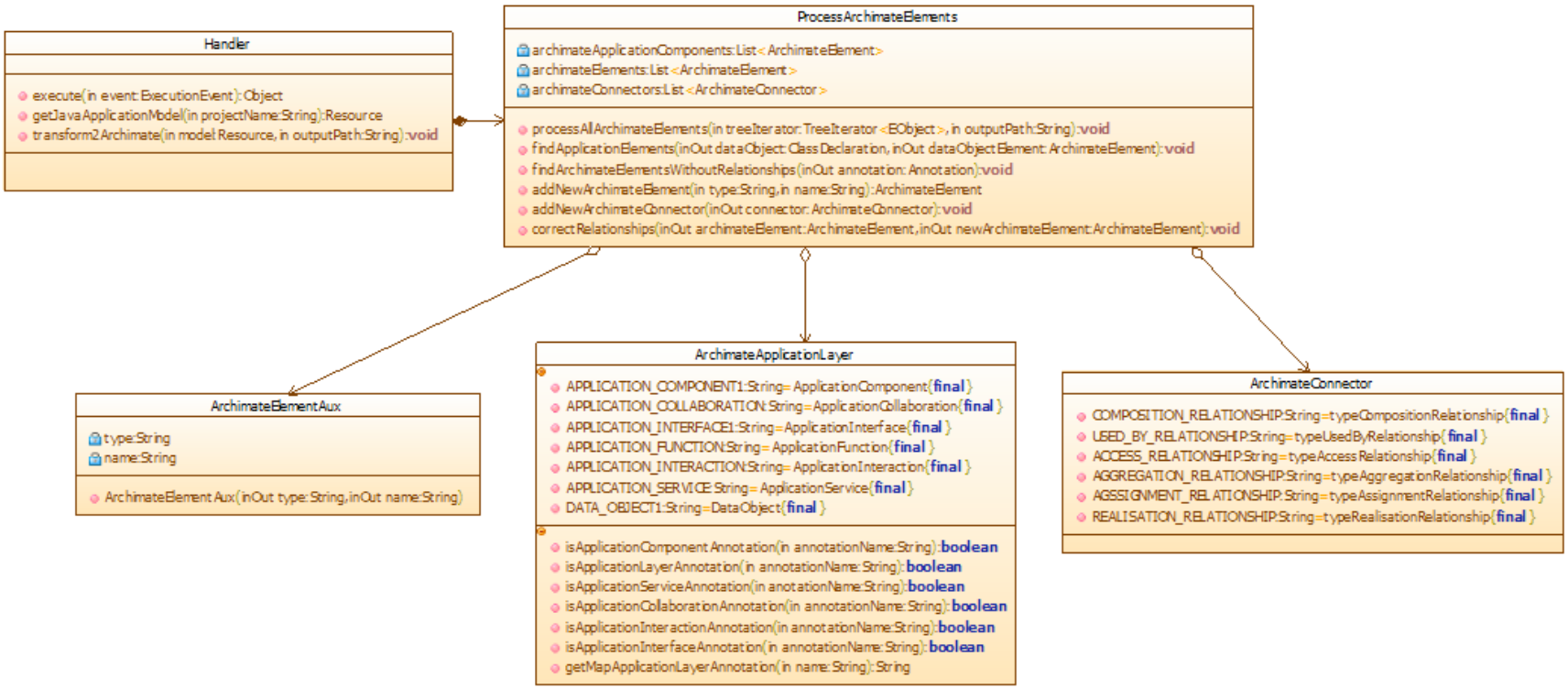


Ilustración 24 Diagrama de clases Java2Archimate

Como se observa en la Ilustración 24, existen 5 clases principales de la aplicación Java2Archimate las cuales son:

- SampleHandler: Clase principal desde donde se ejecuta toda la aplicación.
- ProcessArchimateElements: Clase que contiene la lógica de negocio.
- ArchimateElementAux: Clase que representa un elemento Archimate; contiene la palabra AUX al final de su nombre debido a que en la librería JavaToArchimate que está desarrollada en Xtend, existe una clase con el nombre “ArchimateElement” con un objetivo de uso distinto.
- ArchimateApplicationLayer: Clase en donde se encuentran definidas variables estáticas que sirven para representar, caracterizar e identificar en el código fuente los elementos Archimate del dominio de aplicaciones.
- ArchimateConnector: Clase que representa un conector Archimate del dominio de aplicaciones.

Para el componente JavaToArchimate, se tiene una organización de clases de tal forma que la interacción entre las mismas está orientada a los conceptos de la programación orientada a objetos. En este componente se utilizaron técnicas de agregación y composición tal y como se visualiza en la Ilustración 25.

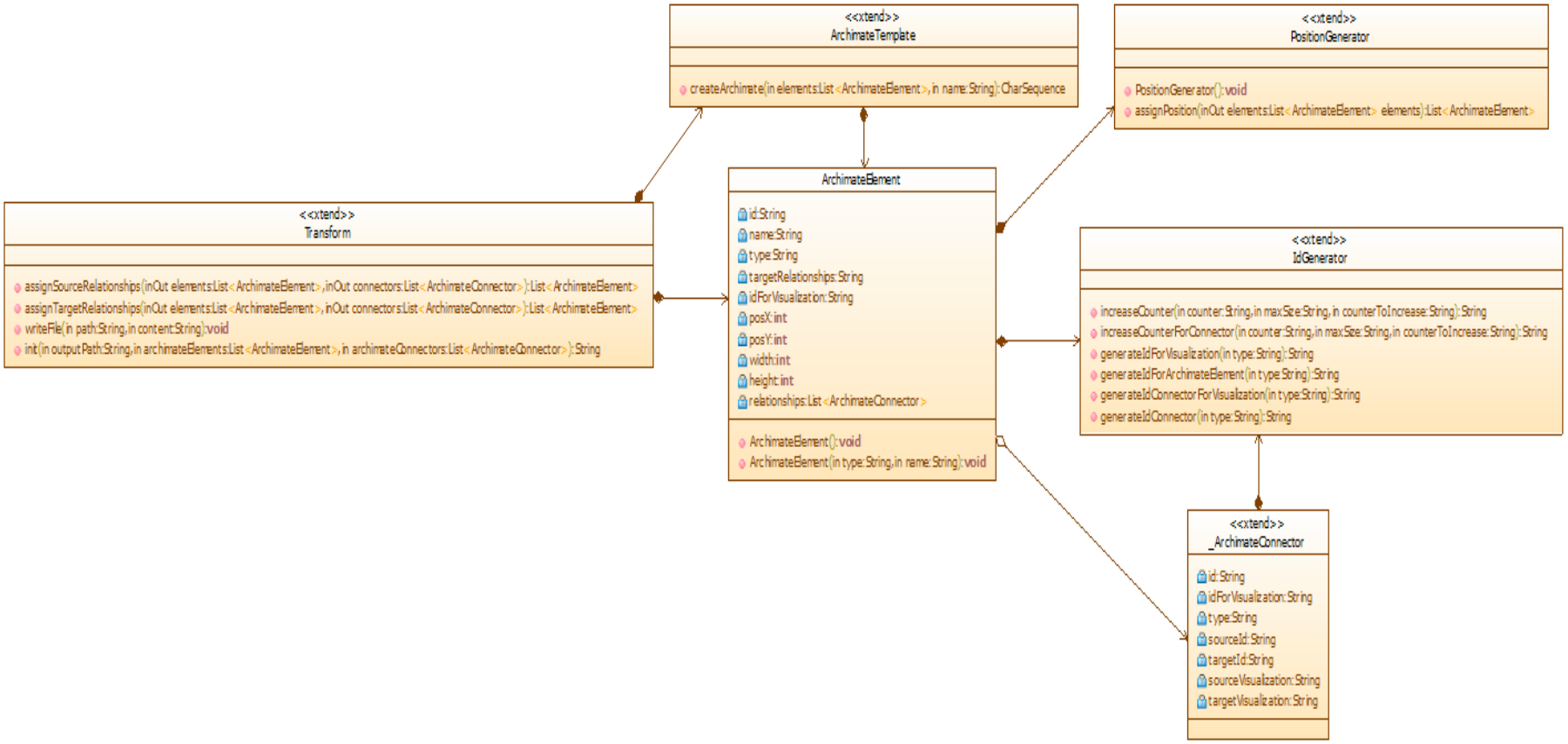


Ilustración 25 Diagrama de clases JavaToArchimate

En el caso de que un proyecto Java desee ser documentado en Archimate y no cumpla con el estándar J2EE, es posible realizarlo a través de la implementación de una librería que se desarrolló llamada JTOGAF_9-1_v1.jar.

JTOGAF_9-1_v1.jar contiene un conjunto de anotaciones las cuales facilitan la lectura del código Java para la transformación a un modelo Archimate. En caso de necesitar documentar en Archimate en el dominio de aplicaciones un Sistema de Información o Software, se sugiere a los analistas de sistemas de información utilizar las anotaciones en los siguientes casos:

1. Inicio del desarrollo de un software que no cumplirá el estándar J2EE; por ejemplo una aplicación de escritorio Stand Alone.
2. Refactoring a una aplicación previamente desarrollada que no cumple el estándar J2EE.

Para poder utilizar las anotaciones de la librería JTOGAF_9-1_v1.jar, basta con importar la librería JTOGAF_9-1_v1 al proyecto y referenciar las anotaciones de acuerdo a la siguiente explicación:

- ApplicationCollaboration: Se debe utilizar esta anotación sobre una clase Java que contenga 2 o más clases Java que hayan sido marcadas como ApplicationComponent.
- ApplicationComponent: Se debe utilizar esta anotación sobre las clases Java que contenga lógica de negocio.
- ApplicationInteraction: Se debe utilizar esta anotación sobre las funciones que hagan llamado a clases Java marcadas como ApplicationComponent; esta anotación solo debe ser utilizada en clases Java marcadas con la anotación ApplicationCollaboration.
- ApplicationInterface: Se debe utilizar esta anotación sobre las clases Java que sirvan de BackEnd para una vista de la aplicación o que sean clases Java utilizadas para representar Vistas donde un usuario tendrá interacción.
- ApplicationService: Se debe utilizar esta anotación sobre las clases Java que expongan servicios; por ejemplo Web Services.
- DataObject: Se debe utilizar esta anotación sobre las clases Java que sean VO u Objetos de Valor que se vayan a persistir en la base de datos.

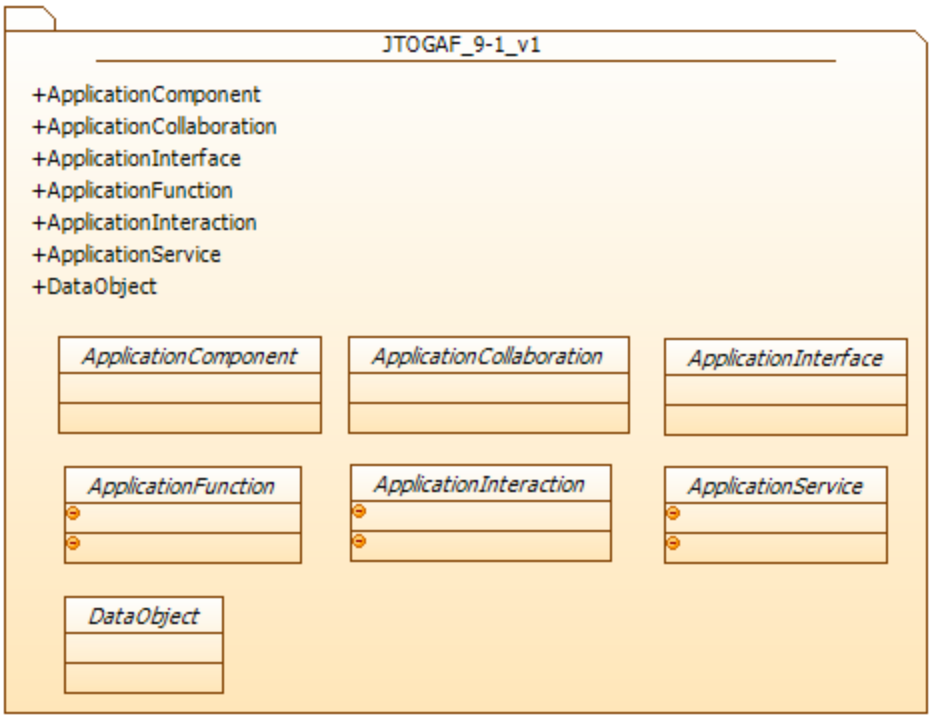


Ilustración 26: Diagrama de Clases JTOGAF_9-1_v1.jar

- **Diagrama de secuencia**

La intención del diagrama de secuencia es modelar la interacción que hay entre los diferentes objetos de la herramienta Java2Archimate y observar cómo se relacionan entre sí para cumplir el objetivo de generar documentación automática en un modelo Archimate.

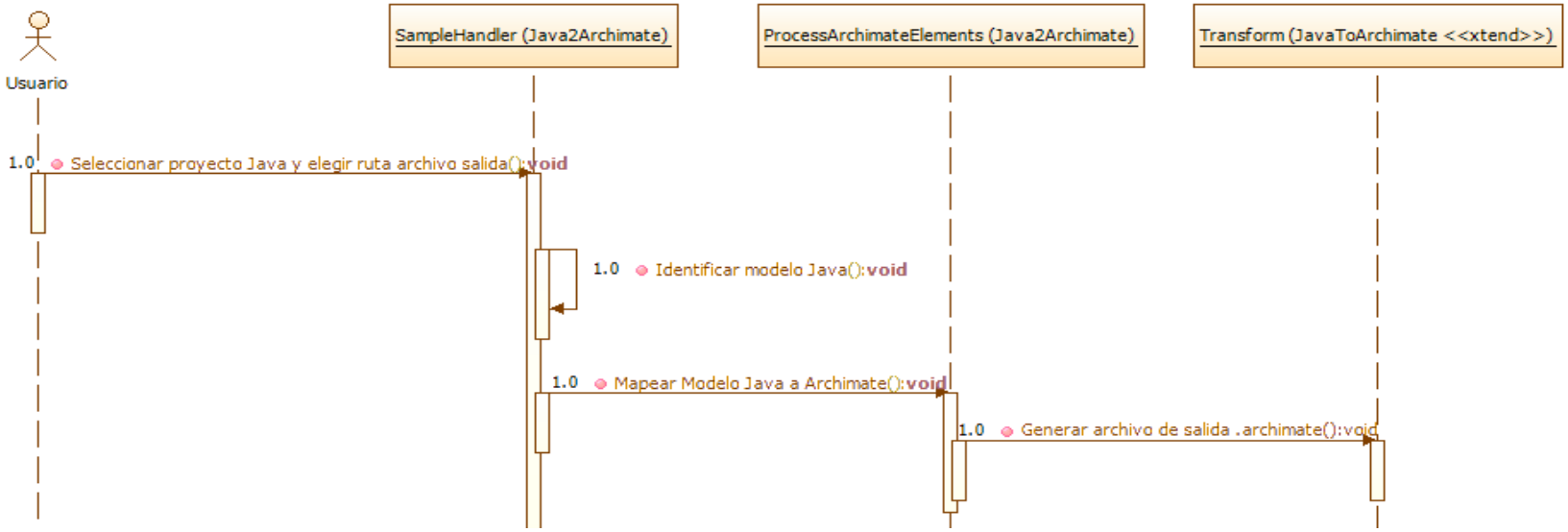


Ilustración 27 Diagrama de secuencia Java2Archimate

En la Ilustración 27, se observa que la secuencia la inicia el usuario con la selección de un proyecto Java y elección de la ruta destino donde quedará almacenado el archivo de salida .archimate, lo cual se realiza haciendo una petición a la clase SampleHandler. En la clase SampleHandler se procede a realizar una identificación del modelo Java que consiste en recorrer el código fuente del proyecto Java seleccionado y la construcción de un árbol jerarquico en un archivo xmi que se maneja internamente para efectos de las transformaciones. Posteriormente se hace la petición para mapear las etiquetas Java a Archimate haciendo uso de la clase ProcessArchimateElements, el resultado de esta petición, genera una lista de los objetos archimate identificados; dicha lista es el insumo para la generación del archivo .archimate; para ello se hace una petición a la clase Transform del proyecto JavaToArchimate desarrollado en lenguaje Xtend

.

- **Integración de herramientas**

Las herramientas mencionadas en la sección 3.5.1 se integran como se muestra en el diagrama de componentes expuesto en la Ilustración 28.

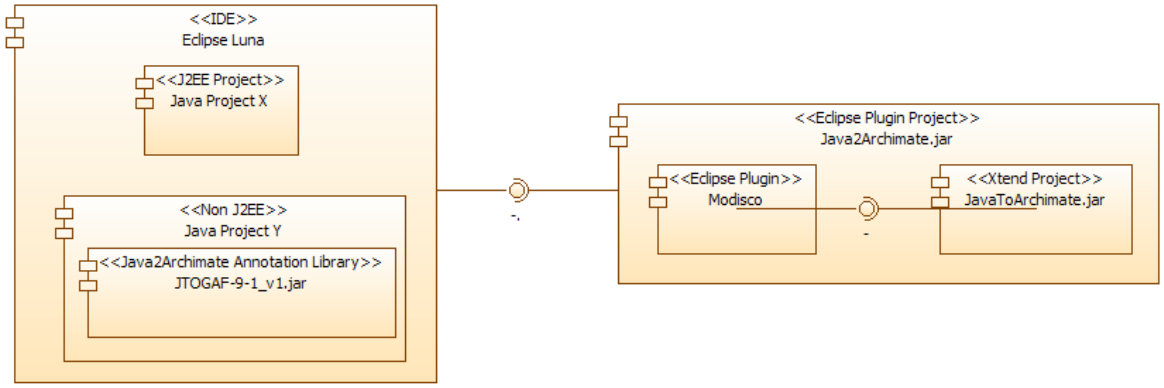


Ilustración 28 Diagrama de Componentes - Integración herramientas

En la parte izquierda se ilustra el IDE Eclipse Luna el cual puede contener desde 1 hasta n proyectos cargados. Se debe tener en cuenta que los proyectos deben estar en lenguaje Java y deben cumplir con el estándar J2EE; en caso de no cumplir con mencionado estándar, el proyecto Java debe importar la librería JTOGAF-9-1_v1.jar que se desarrolló en esta investigación, y de acuerdo a lo que se explica en la página 53.

En la parte derecha de la Ilustración 28, se tiene un Eclipse Plugin Project que corresponde al proyecto Java2Archimate el cual requiere utilizar el componente Eclipse Luna para poder ejecutarse. Dentro del proyecto Java2Archimate se encuentra el uso de la herramienta Modisco, la cual se encarga de obtener el metamodelo de un proyecto Java previamente seleccionado; posteriormente el metamodelo obtenido es utilizado para realizar la correspondencia entre Java y Archimate, generando una listado de elementos a transformar con sus correspondientes relaciones y posteriormente se construye el lenguaje de modelado de Archimate con ayuda del componente JavaToArchimate desarrollado en Xtend. Para una explicación más clara, se presenta la Ilustración 29.

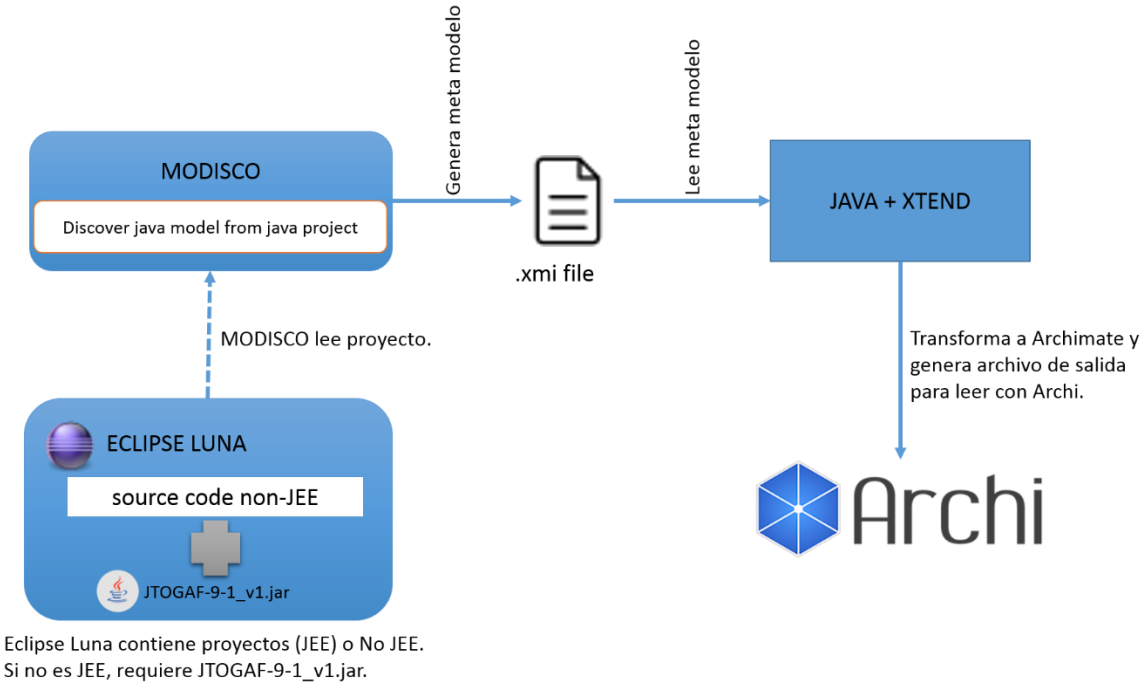


Ilustración 29 Proceso general de transformación Java2Archimate

3.6 Construcción de opciones de usuario

La construcción de las opciones de usuario fue basada en el único caso de uso que surgió del análisis de requerimientos y que se expresó en la sección “Arquitectura de Software”. Para ello, se diseñó una opción llamada “Java2Archimate”en el menú Project del IDE Eclipse Luna; también se puede acceder a la opción haciendo clic derecho sobre el proyecto. Al hacer clic sobre esta opción de menú, se despliega una única opción llamada “Java2Archimate” como se visualiza en la Ilustración 30 e Ilustración 31.

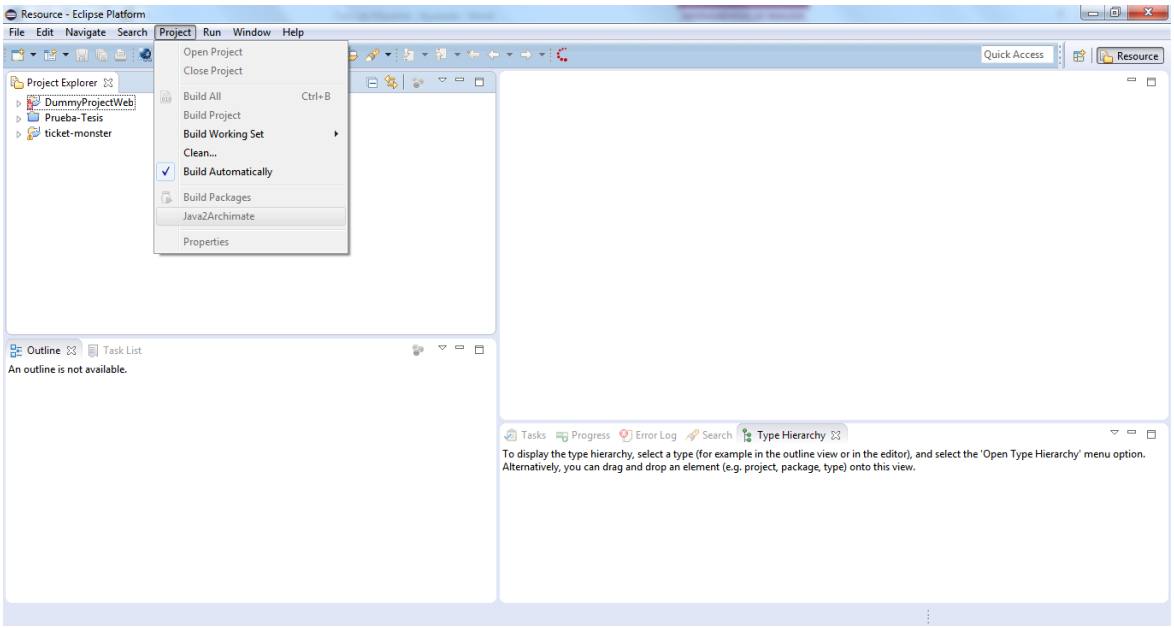


Ilustración 30 Opciones de usuario: Java2Archimate menú Project

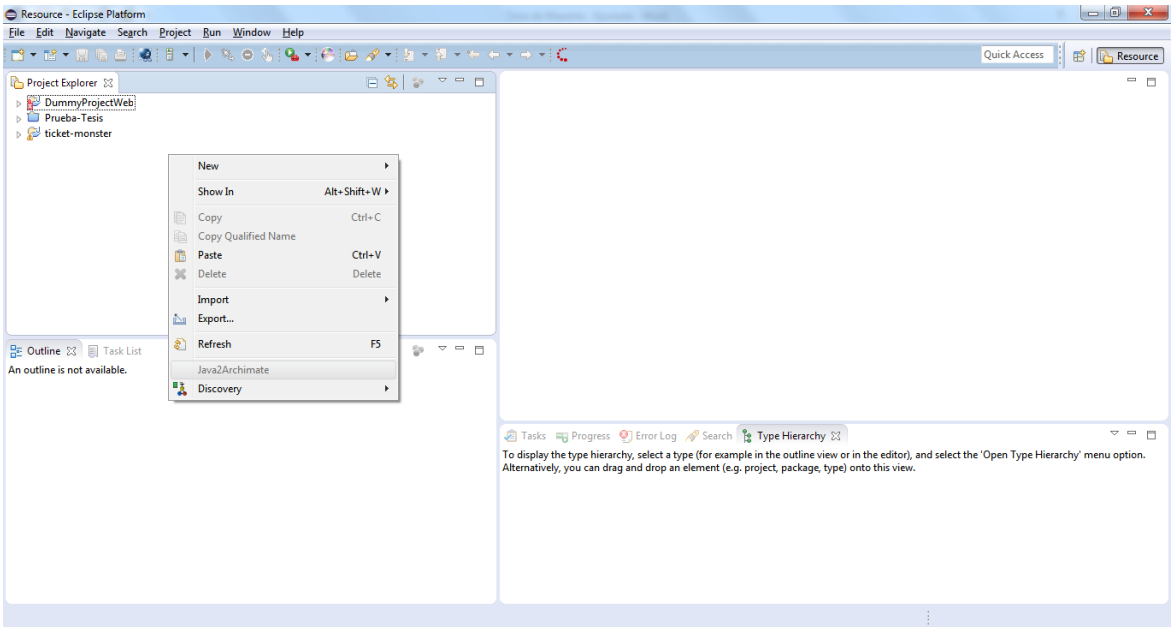


Ilustración 31 Opciones de usuario: Java2Archimate menú contextual

Como se observa en las ilustraciones anteriores, la opción Java2Archimate aparece deshabilitada debido a que no hay seleccionado un proyecto que esté abierto. En ese caso, aparecería como se observa en la Ilustración 32 e Ilustración 33.

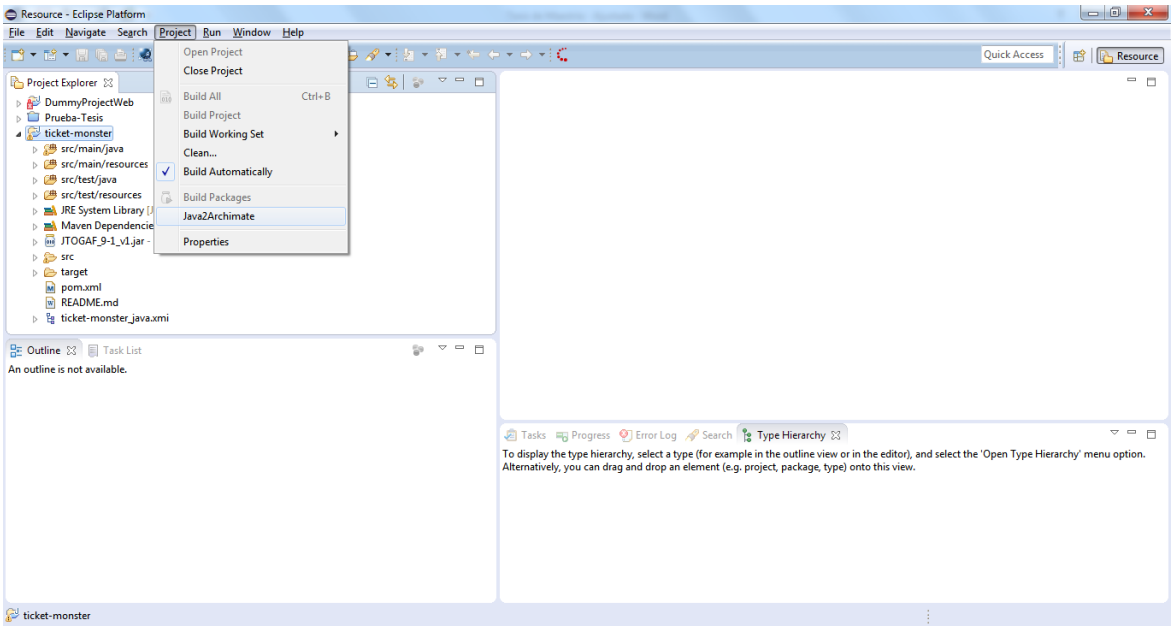


Ilustración 32 Java2Archimate habilitado en el menú Project

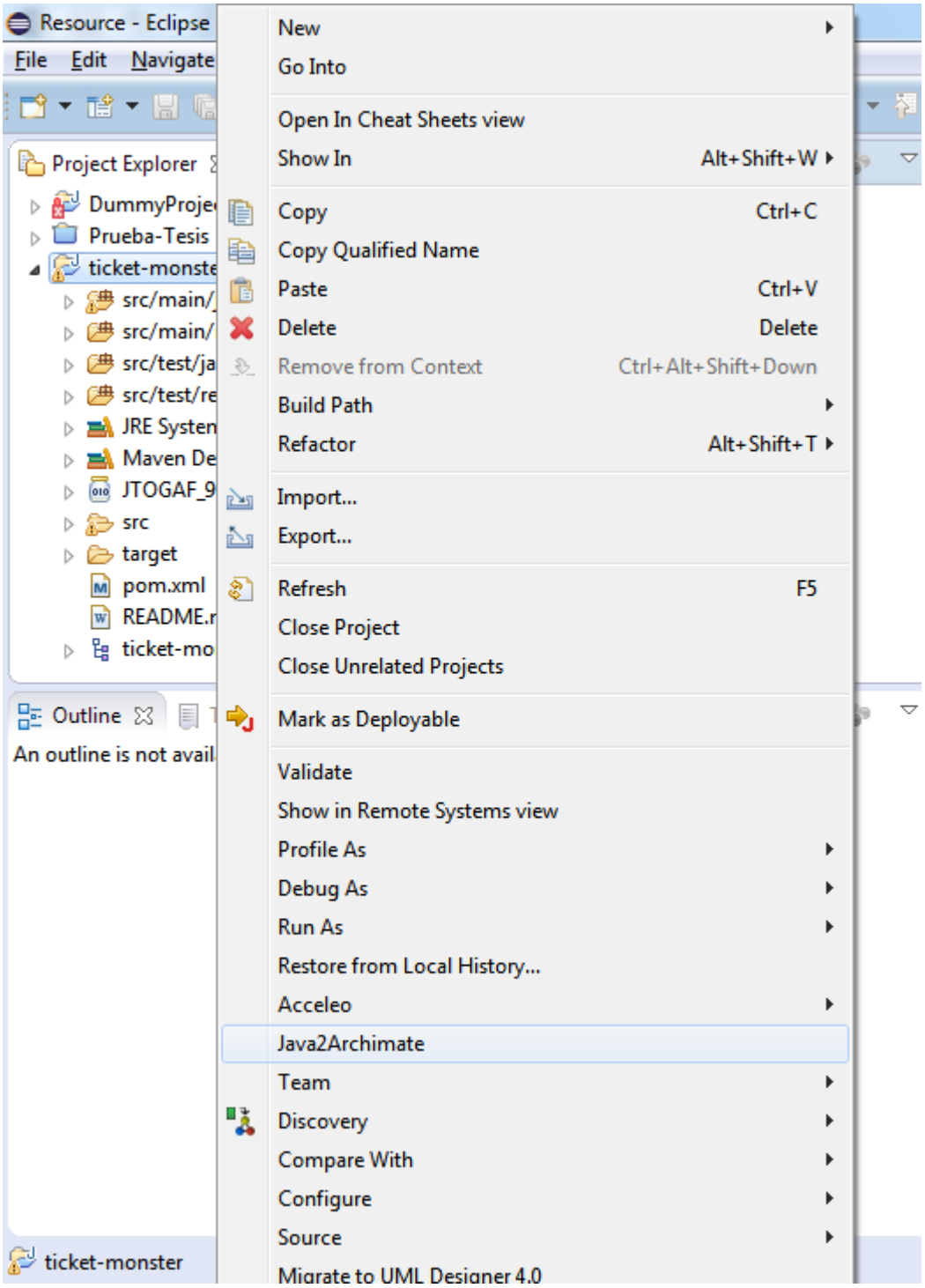


Ilustración 33 Java2Archimate habilitado en menú contextual

Luego de hacer clic sobre la opción Java2Archimate, al usuario se le despliega una nueva ventana en la cual puede elegir la ruta en donde desea guardar el archivo resultante de la transformación.

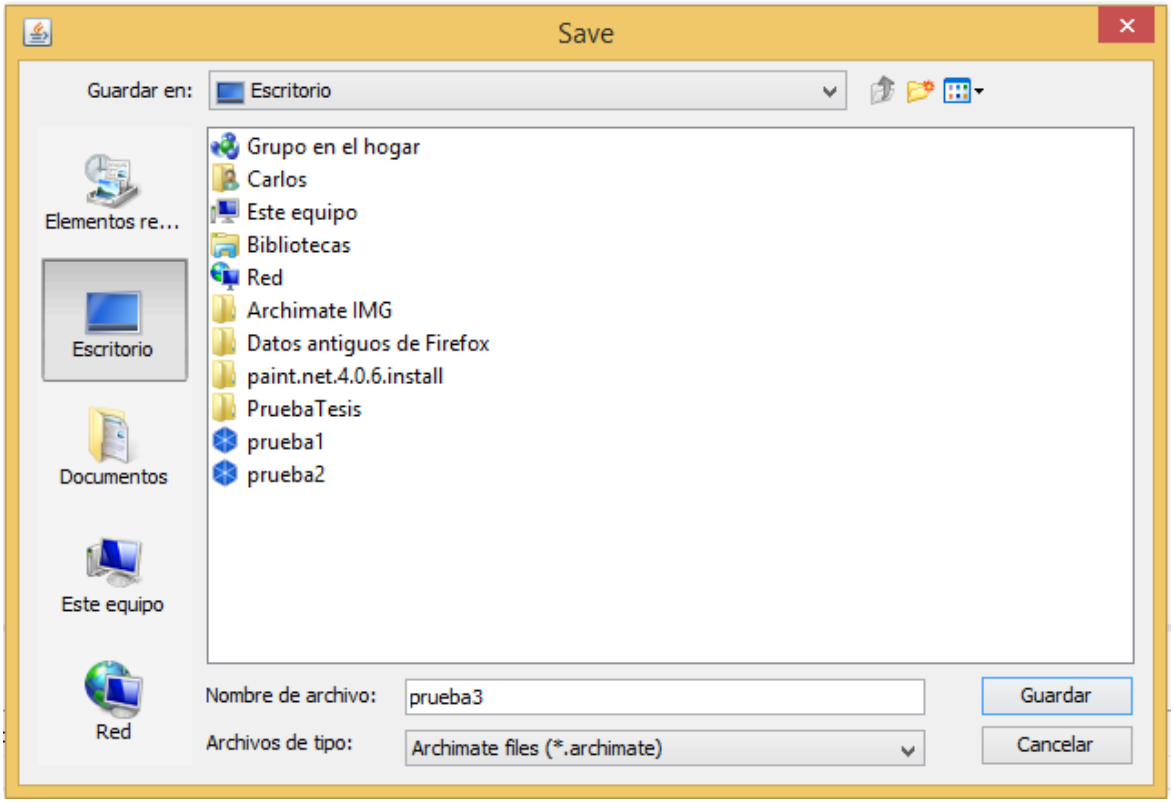


Ilustración 34 Opciones de usuario: selección ruta archivo salida

Finalmente, cuando la herramienta Java2Archimate termina el proceso de transformación, se despliega una ventana notificando al usuario del resultado del proceso.

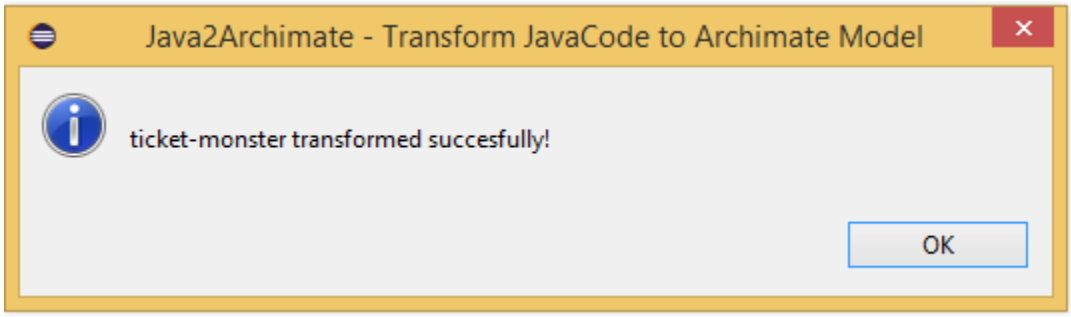


Ilustración 35 Opciones de usuario: Resultado proceso de transformación

4. Prueba de concepto y evaluación

Se decidió realizar dos pruebas de concepto para probar la efectividad de Java2Archimate. Para la prueba número 1 se seleccionó el sistema Ticket-monster el cual fue utilizado en la metodología del presente trabajo de investigación para el desarrollo de cada una de las fases de la metodología planteada.

Para la prueba número 2, se seleccionó el SPOPA, el cual es un sistema que se utiliza en la Universidad Nacional de Colombia para administrar las convocatorias de prácticas y pasantías que pueden tomar los estudiantes de la Universidad.

Las pruebas de concepto son satisfactorias si y solo si se cumplen las siguientes condiciones:

1. No se genera ningún error al momento de hacer clic sobre la opción transformar en cualquiera de los proyectos Java importados en Eclipse Luna sin tener la necesidad de configurarlo para su ejecución.
2. Se genera correctamente un archivo .xmi dentro de la ruta del proyecto JAVA. Este archivo representa el modelo Java obtenido del código fuente y es generado con ayuda de Modisco.
3. Se genere un archivo .archimate en la ruta seleccionada y al hacer doble clic, la herramienta de software Archi lo pueda leer sin ningún inconveniente.

Los elementos Archimate generados, deben corresponder con las clases Java que se encuentran en cada uno de los proyectos Java según la tabla de transformaciones definida en el capítulo 3.4 Análisis para validar que la propuesta de transformación es viable.

4.1 Prueba número 1

4.1.1 Preparación de la prueba

El objetivo de la prueba es validar los siguientes puntos:

1. Validar la funcionalidad de Activo / Inactivo del botón Java2Archimate bajo la condición de que debe haber un proyecto abierto seleccionado en el workspace del IDE.
2. Comprobar que la propuesta de transformación de Java a Archimate aplica para el proyecto ticket-monster.
3. Comprobar el uso de la librería JTOGAF junto con anotaciones J2EE.

Para realizar la prueba, se prepara el workspace de la siguiente forma:

1. Se inicializa el IDE Eclipse Luna con el plugin Java2Archimate configurado.
2. Se realiza import del código fuente del proyecto ticket-monster en el IDE; esto se puede evidenciar en la Ilustración 30.
3. Se utilizan las anotaciones de la librería JTOGAF en el proyecto ticket-monster creando unas clases de prueba:
    a. TicketController.java: contiene la anotación @ApplicationInterface

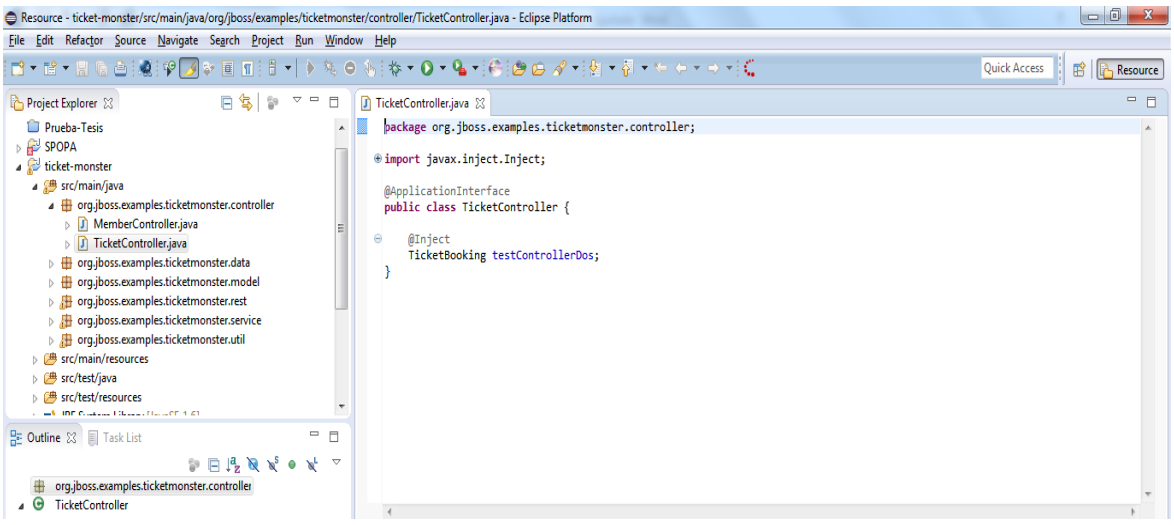


Ilustración 36 Clase TicketController.java con JTOGAF

b. Ticket.java: Contiene la anotación @DataObject

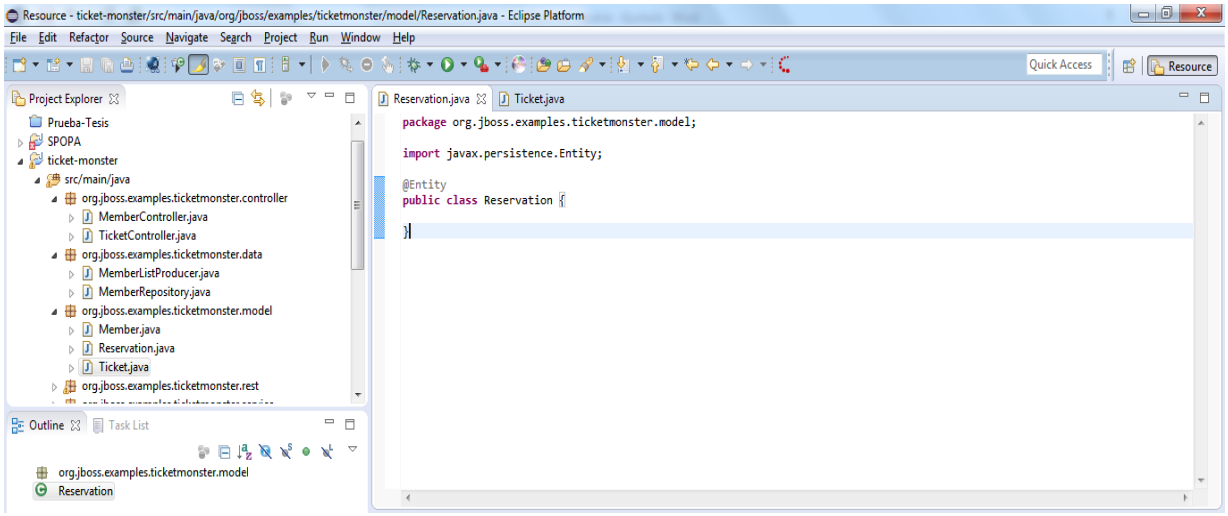


Ilustración 37 Clase Ticket.java con JTOGAF

c. TicketBooking.java: Contiene la anotación @ApplicationComponent

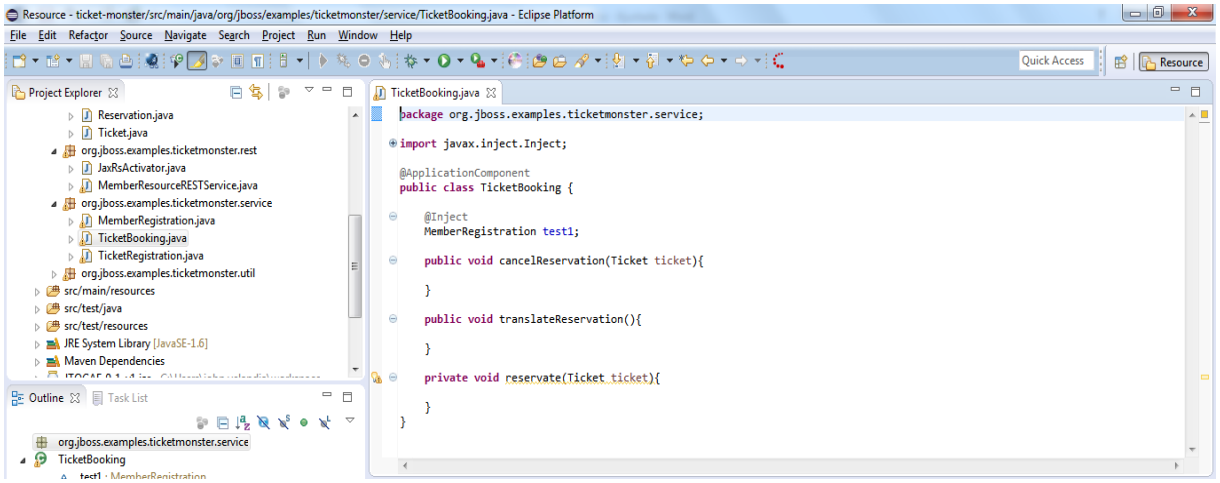


Ilustración 38 Clase TicketBooking.java con JTOGAF

4.1.2 Desarrollo de la prueba

1. Se crea un proyecto de prueba en el IDE llamado “Prueba-Tesis” y posteriormente se cierra para realizar la validación de Activo / Inactivo del botón Java2Archimate, para ello se prueban los siguientes casos:

    a. Menú Project sin seleccionar ningún proyecto Ilustración 39.
    b. Menú contextual sin seleccionar ningún proyecto Ilustración 40.
    c. Menú Project seleccionando el proyecto “Prueba-Tesis” que se encuentra en estado cerrado Ilustración 41.
    d. Menú contextual seleccionando el proyecto “Prueba-Tesis” que se encuentra en estado cerrado Ilustración 42.
    e. Menú contextual seleccionando una clase del proyecto “ticket-monster” para validar que la opción solo se habilita al seleccionar la raíz del proyecto. Ilustración 43

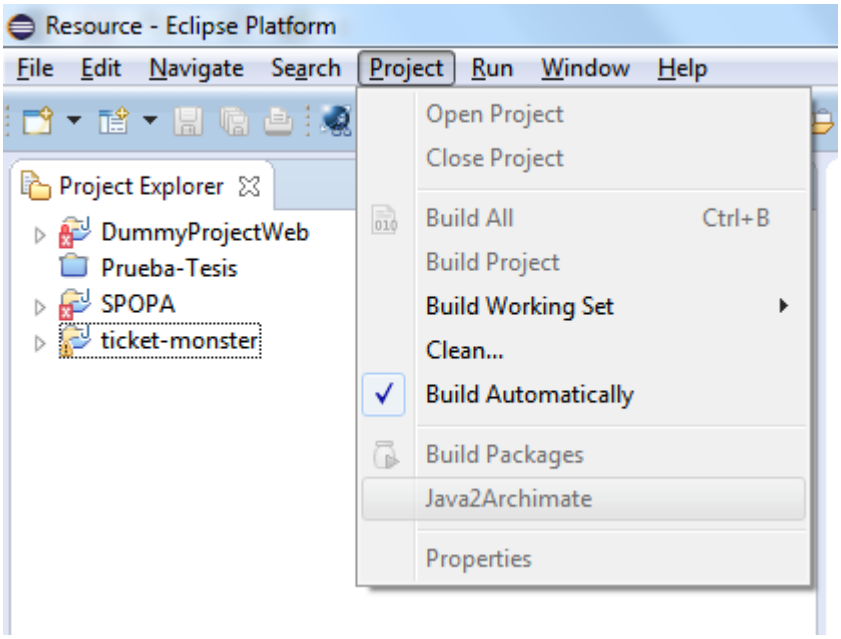


Ilustración 39 Evidencia Java2Archimate en menú Project deshabilitado si no hay proyecto seleccionado

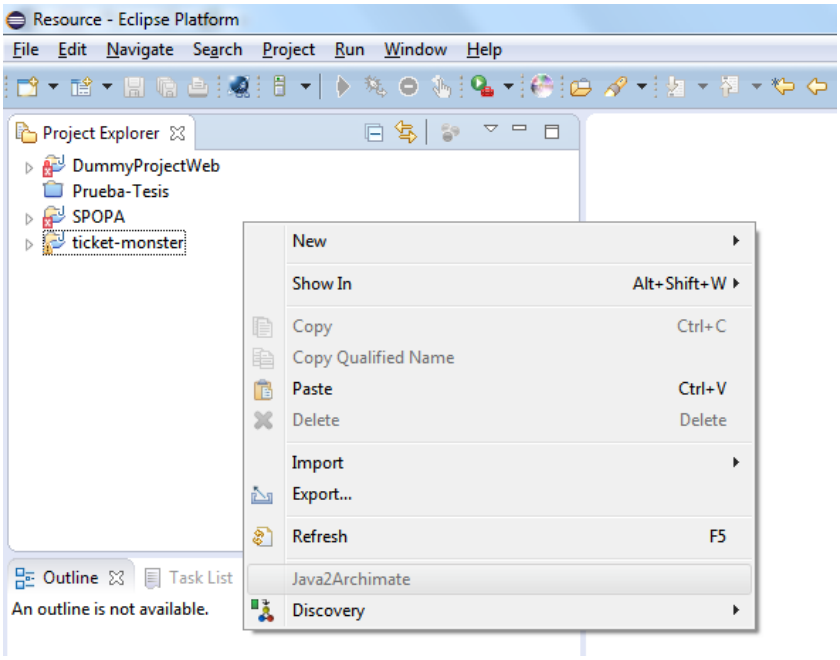


Ilustración 40 Evidencia Java2Archimate en menú contextual deshabilitado si no hay proyecto seleccionado

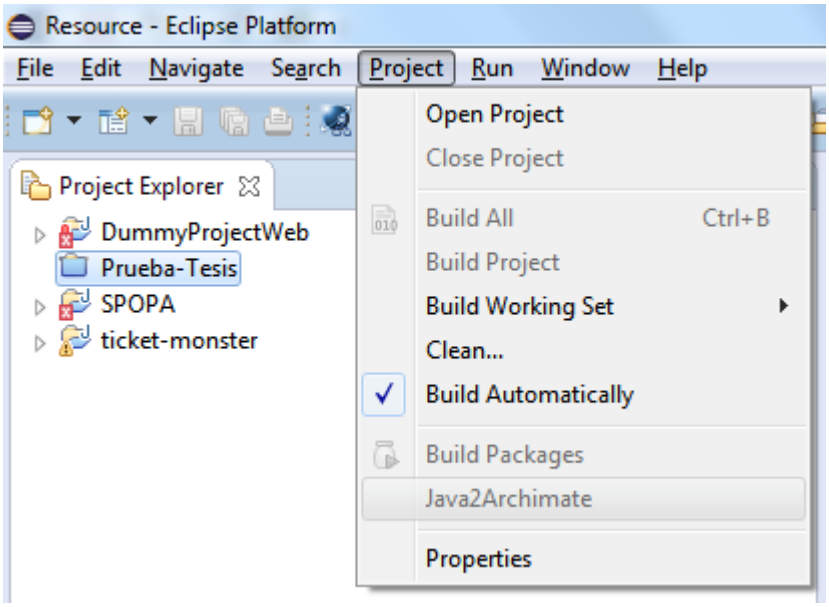


Ilustración 41Evidencia Java2Archimate en menú Project deshabilitado si el proyecto seleccionado está cerrado.

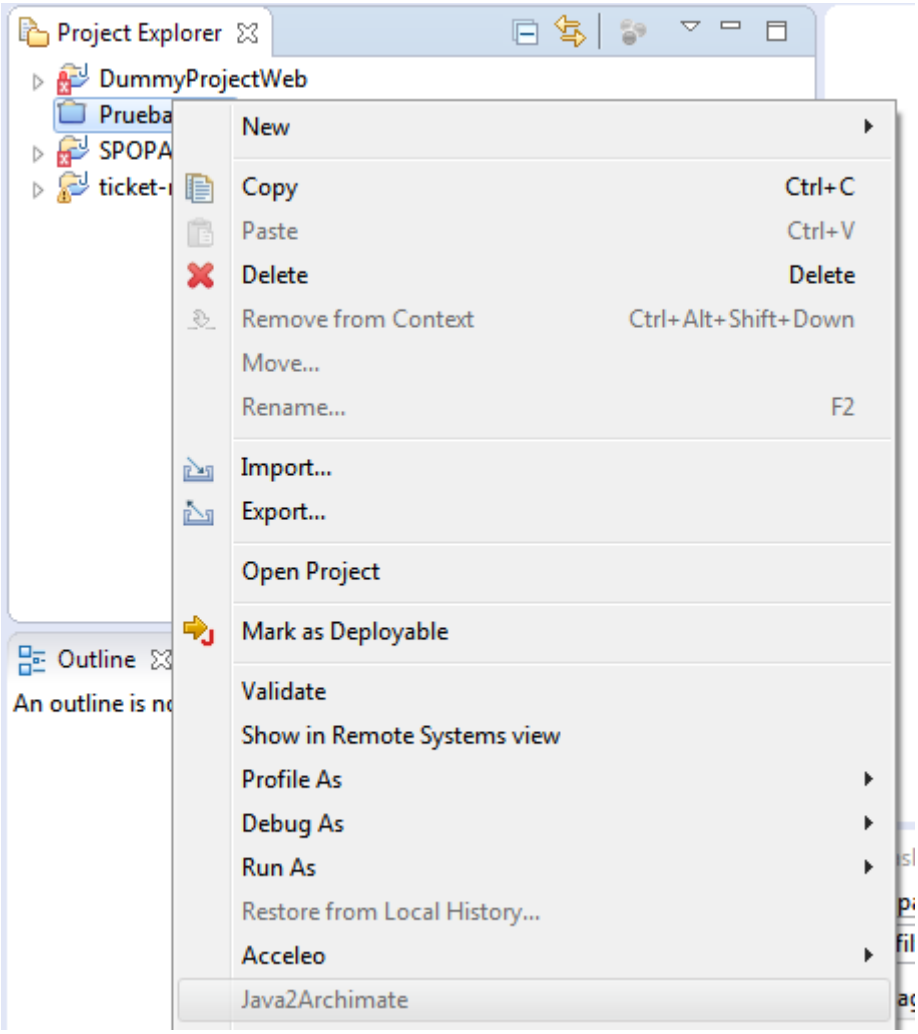


Ilustración 42 Evidencia Java2Archimate menú contextual deshabilitado si el proyecto seleccionado está cerrado

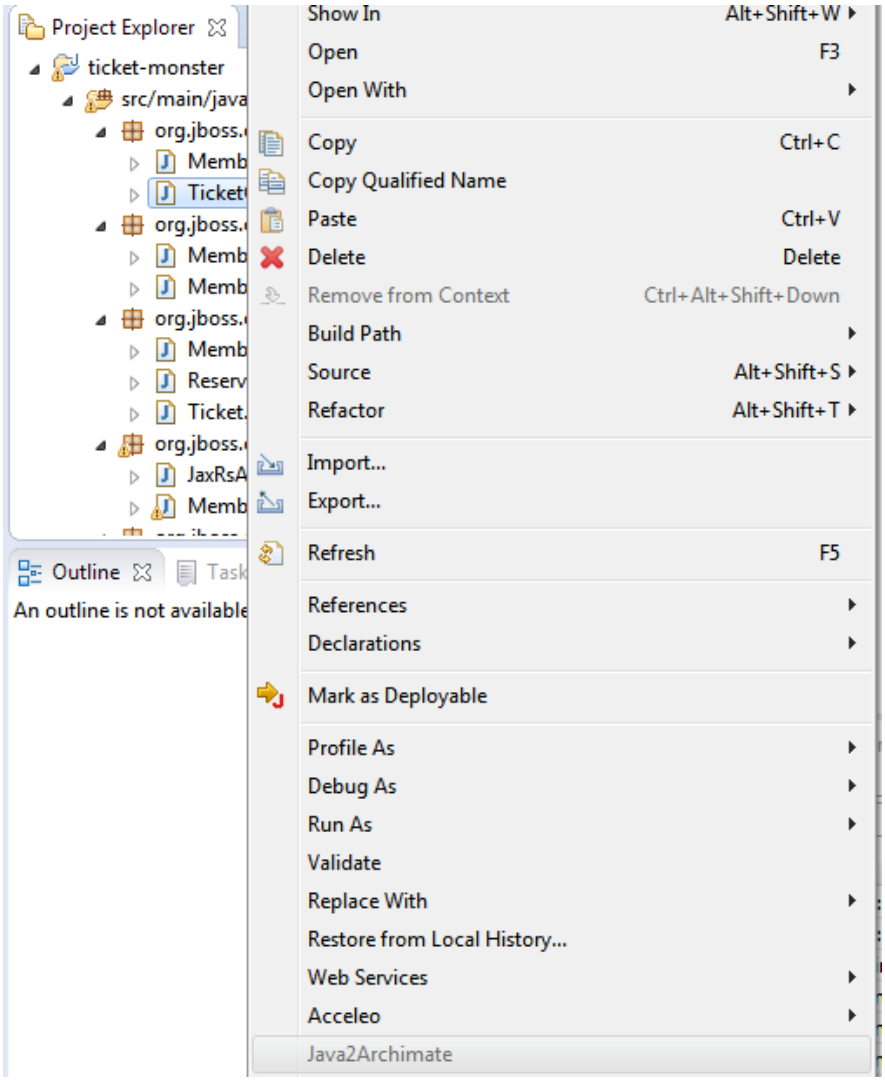


Ilustración 43 Evidencia Java2Archimate menú contextual deshabilitado si no se selecciona la raíz del proyecto.

2. Se crea una carpeta en donde quedara el archivo .archimate resultante.
3. Se ejecuta la lógica del botón Java2Archimate.

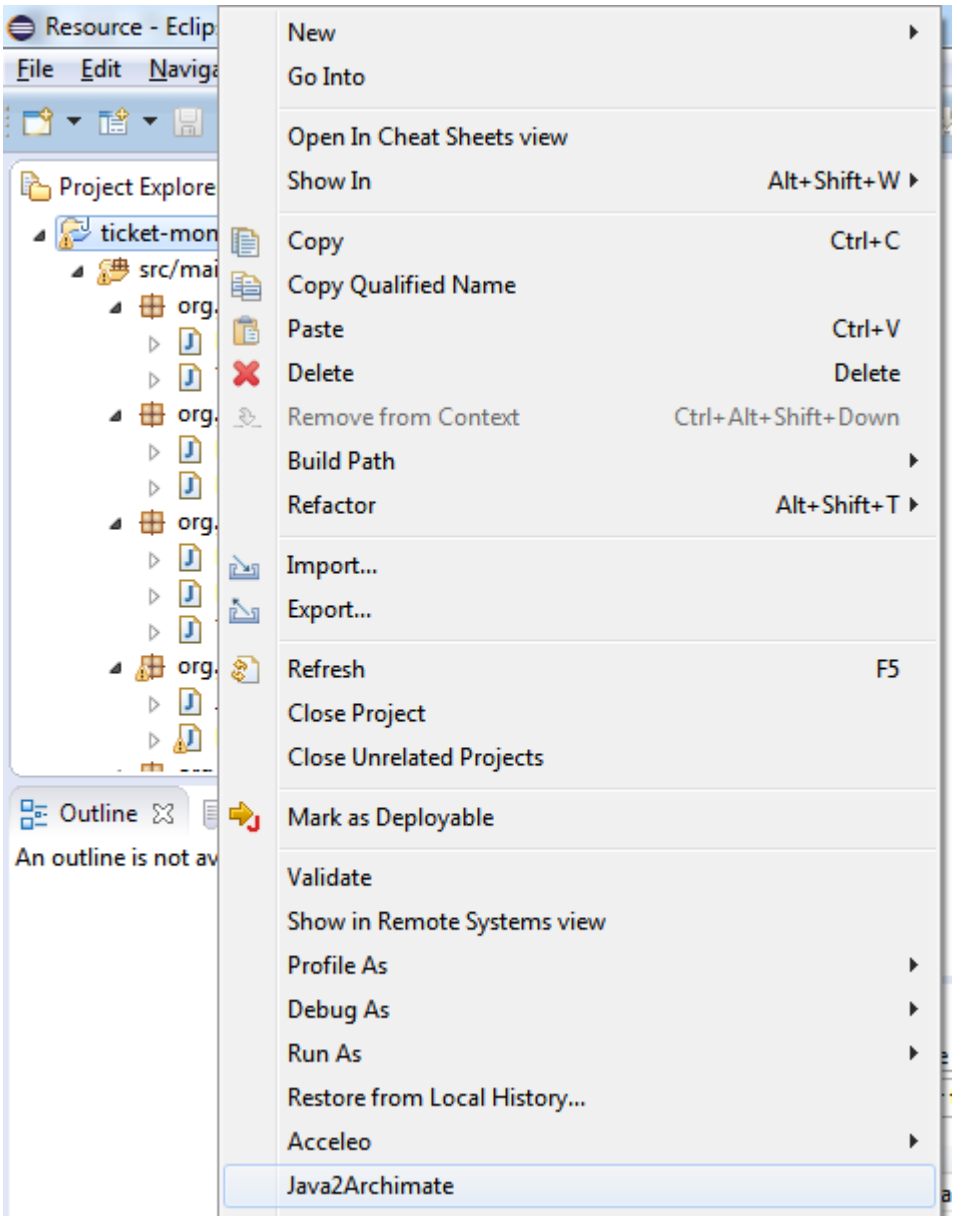


Ilustración 44 Java2Archimate desde menú contextual habilitado al seleccionar raíz de ticket-monster

4. Se elige la ruta donde se guardará el archivo resultante; en este caso la carpeta creada en el paso 2.

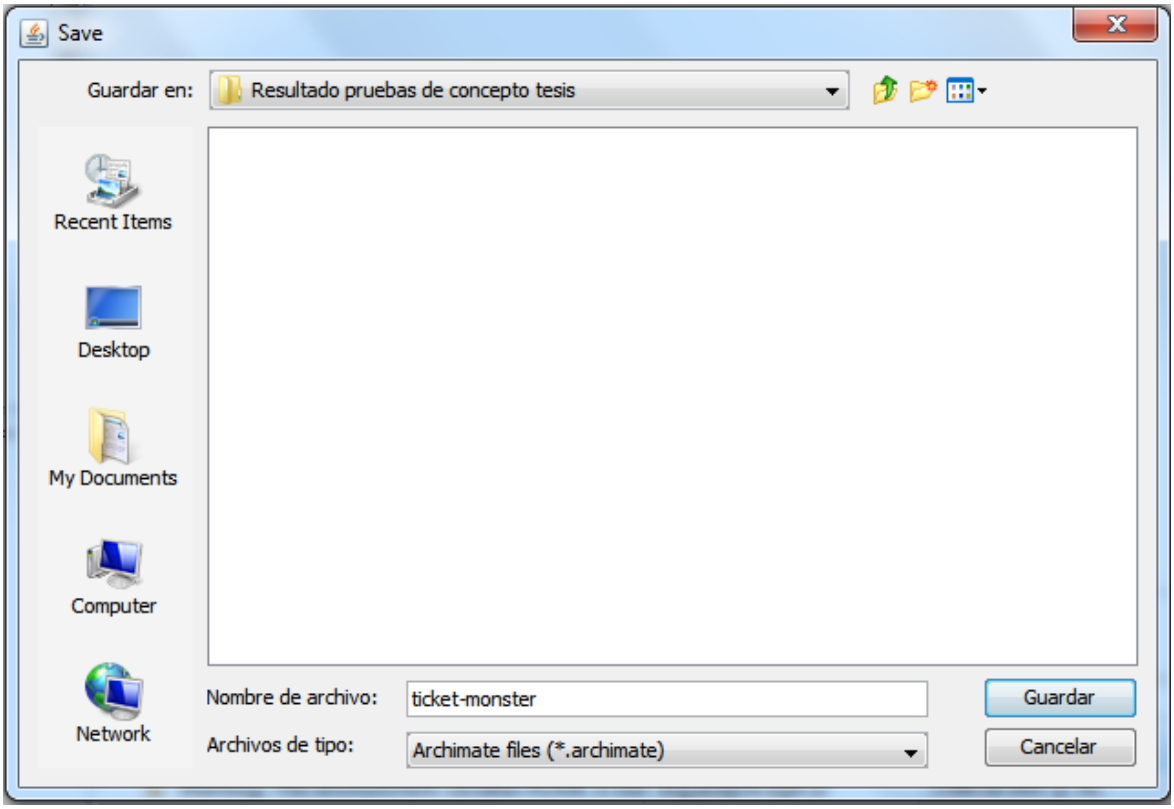


Ilustración 45 Evidencia selección de carpeta y nombre de archivo para ticket-monster

5. Se valida que el proceso haya sido exitoso de la siguiente forma:
    a. La herramienta muestra mensaje de transformación exitosa

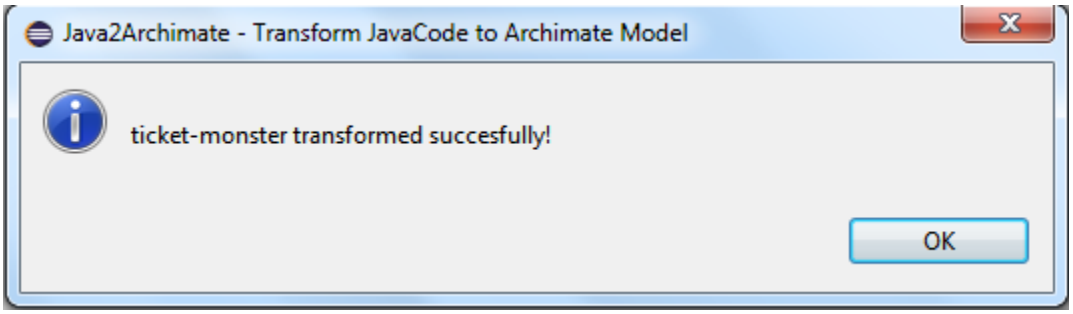


Ilustración 46 Evidencia transformación exitosa Ticket-Monster

    b. Se obtuvo el metamodelo del código fuente del proyecto, el cual quedó en un archivo con extensión .xmi, almacenado en la misma ruta del proyecto.

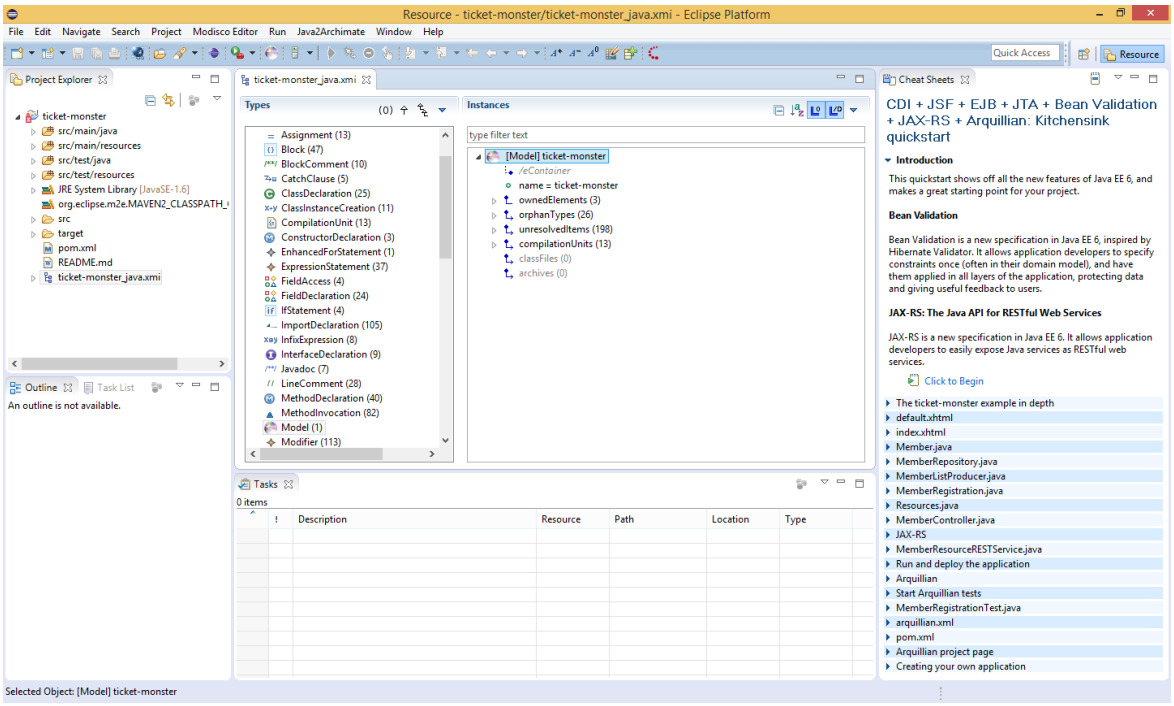


Ilustración 47 Evidencia archivo XMI para ticket-monster

c. Se creó el archivo .archimate en la carpeta creada previamente

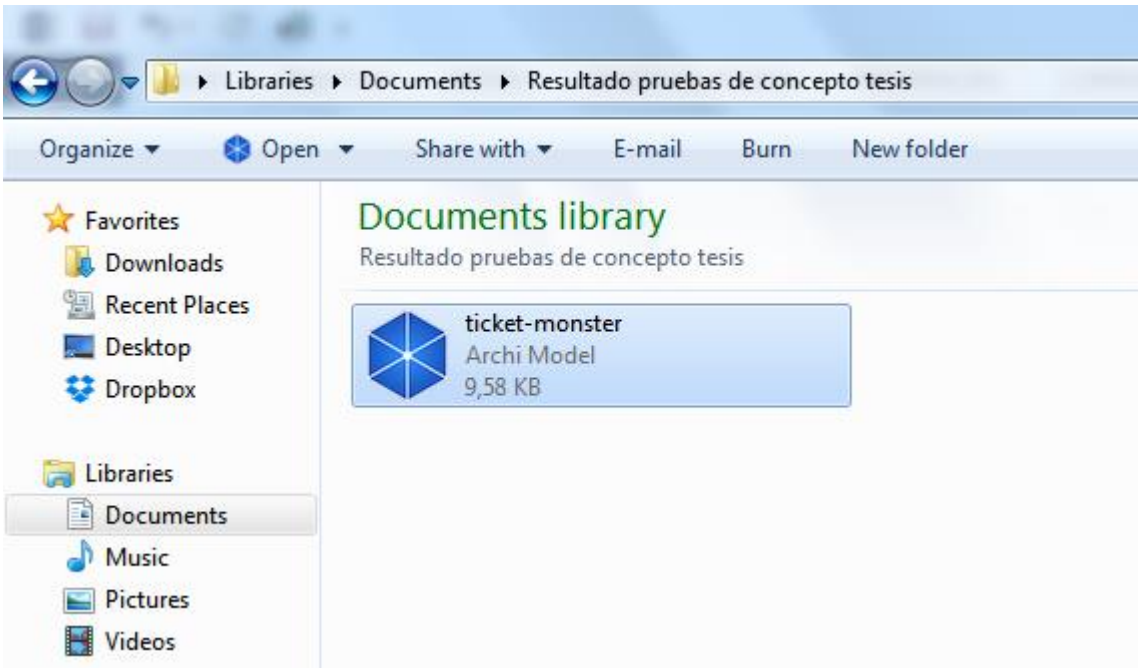


Ilustración 48 Evidencia archivo .archimate generado para ticket-monster

d. Se verifica que el archivo .archimate generado, pueda ser interpretado por la herramienta Archi.

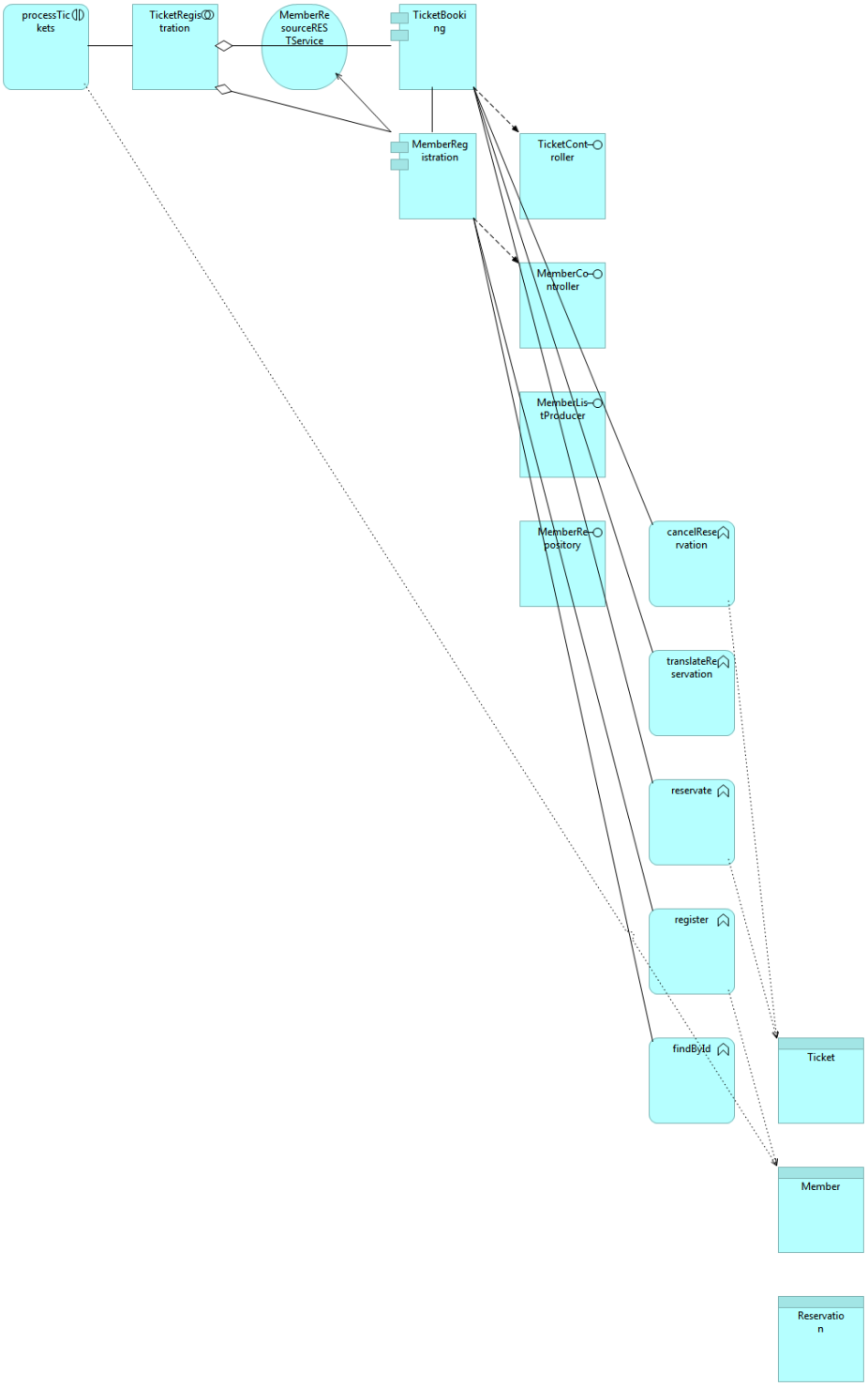


Ilustración 49 Evidencia modelo Archimate generado para ticket-monster

Al validar el código fuente de ticket-monster con la Tabla 7, se observa que la herramienta realiza las transformaciones correctamente de acuerdo a las anotaciones J2EE y JTOGAF contenidas en el código. Para una mejor visualización, se organizó manualmente el modelo en la herramienta Archi. Ver la Ilustración 50

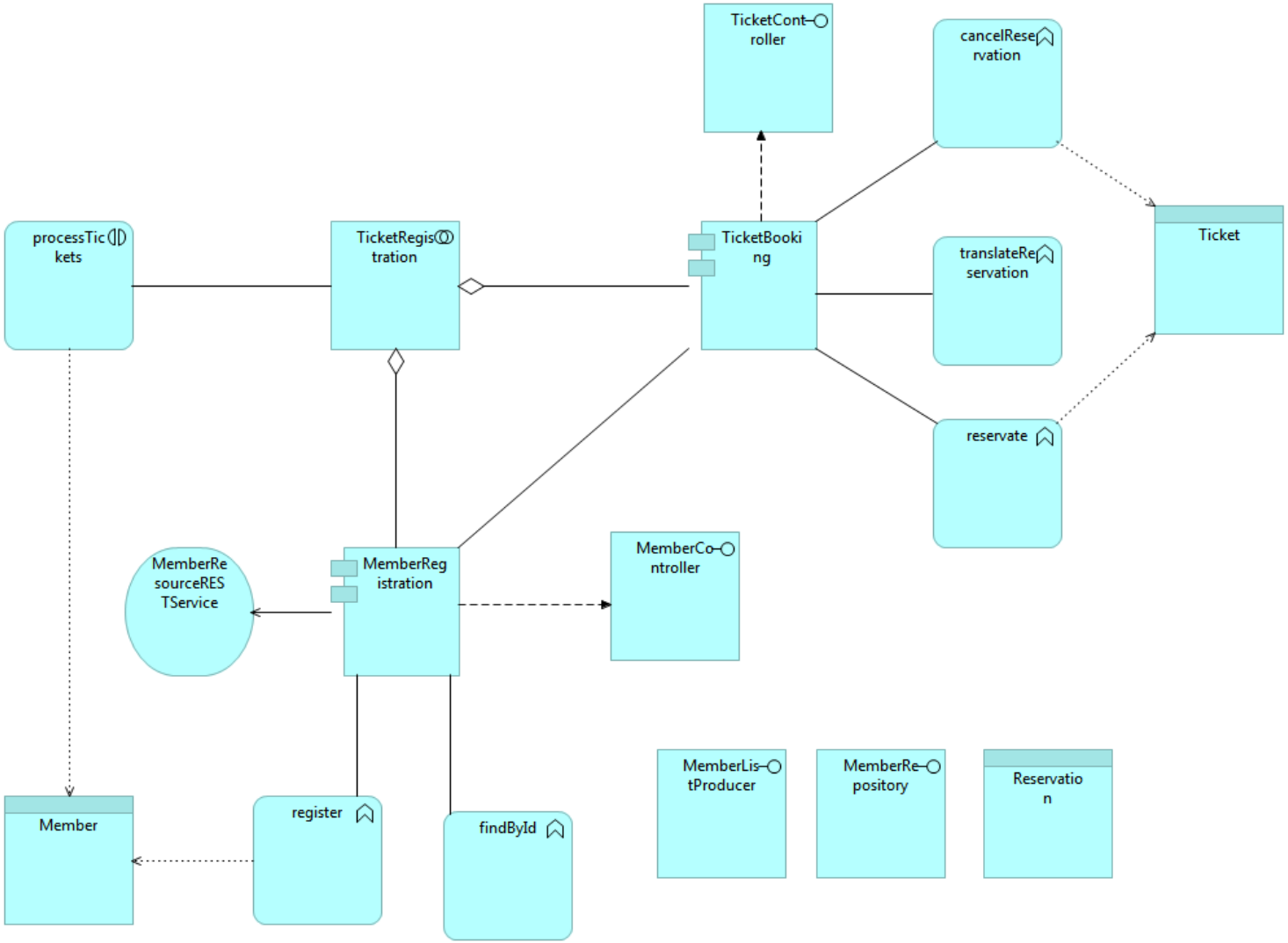


Ilustración 50 Evidencia modelo Archimate organizado manualmente para ticket-monster

4.2 Prueba número 2

4.2.1 Preparación de la prueba

La Universidad Nacional de Colombia cuenta con un sistema que permite administrar las prácticas y pasantías que ofrecen las empresas a los estudiantes de la Universidad, lo cual facilita el control de las ofertas y la inclusión de los estudiantes en el mundo laboral. Este sistema se llama SPOPA y para poder realizar la prueba de concepto, se debieron solicitar permisos a la Dirección Nacional de Tecnologías de la Información y Comunicaciones DNTIC para utilizar el código fuente con fines académicos y haciendo un pacto de confidencialidad de no difundir ni utilizar el código para beneficio personal; por lo anterior, a grosso modo se expone que SPOPA está desarrollado en el IDE Netbeans, cuenta con más de 300 clases desarrolladas en Java en una arquitectura n-tier; está compuesto de un Enterprise Application Project dentro del cual hay un proyecto EJB y un proyecto Web.

Para la prueba de concepto, se seleccionaron paquetes de código Java del proyecto EJB y del proyecto WEB agrupándolos en un solo proyecto e importándolos al IDE Eclipse Luna.

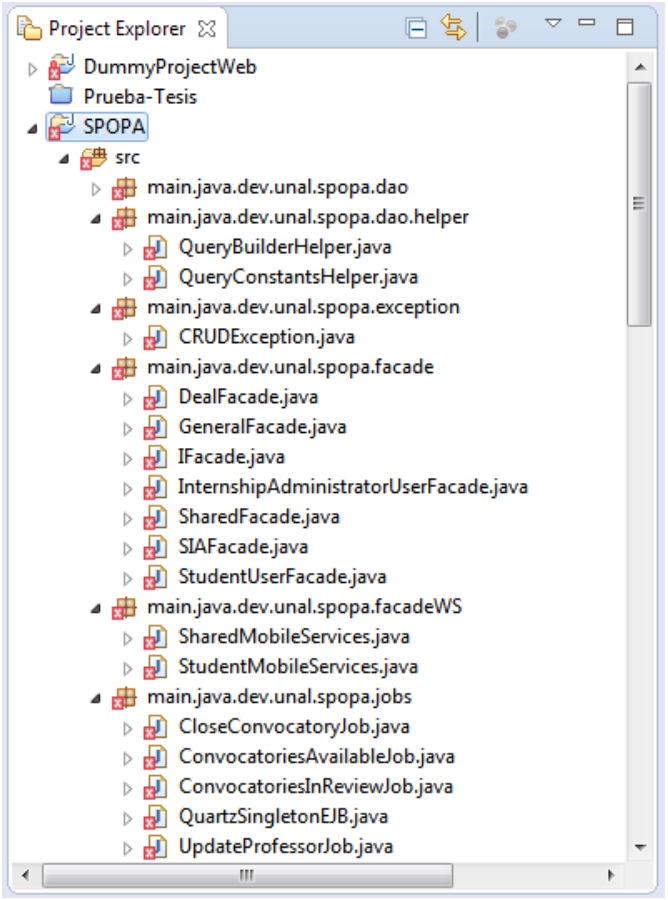


Ilustración 51 Evidencia SPOPA cargado en IDE Eclipse Luna

El objetivo de esta prueba de concepto es validar que la propuesta de transformación de Java a Archimate aplica para sistemas reales y observar el comportamiento de la herramienta Java2Archimate para identificar posibles mejoras y/o trabajo futuro.

4.2.2 Desarrollo de la prueba

Para validar el objetivo de la prueba, se ejecuta la opción Java2Archimate desde el menú Project.

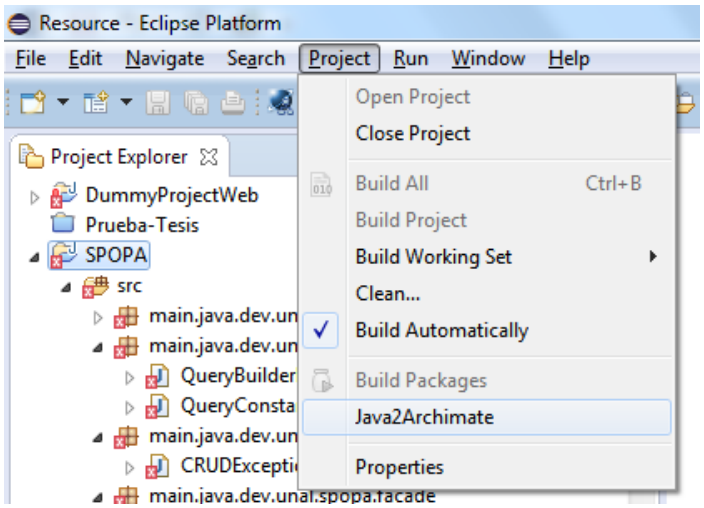


Ilustración 52 Java2Archimate habilitado desde el menú Project para el proyecto SPOPA

Se selecciona la carpeta creada previamente en la prueba de concepto número 1.

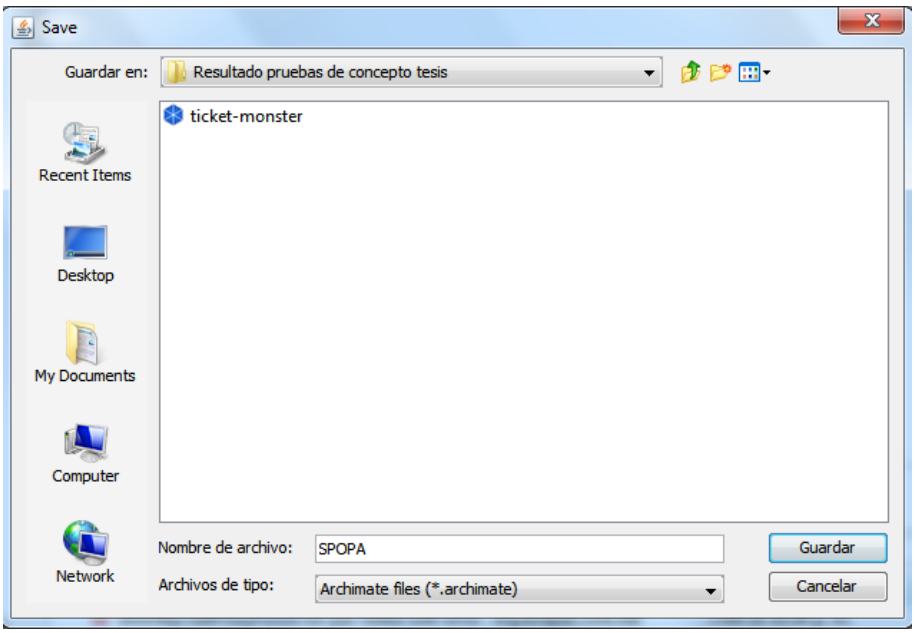


Ilustración 53 Evidencia selección de carpeta y nombre de archivo para SPOPA

Se realizan las siguientes verificaciones para comprobar que la transformación fue exitosa:

1. Se verifica el archivo .xmi con el metamodelo de SPOPA dentro del proyecto.

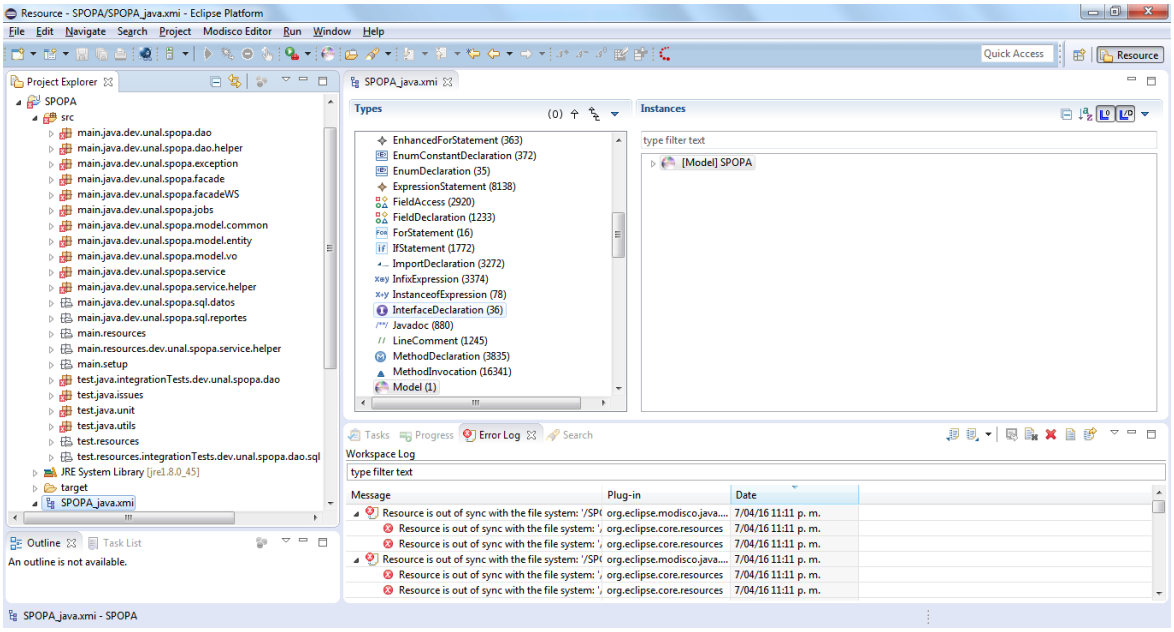


Ilustración 54 Evidencia archivo XMI para SPOPA

2. Se verifica que el archivo .archimate haya sido generado correctamente

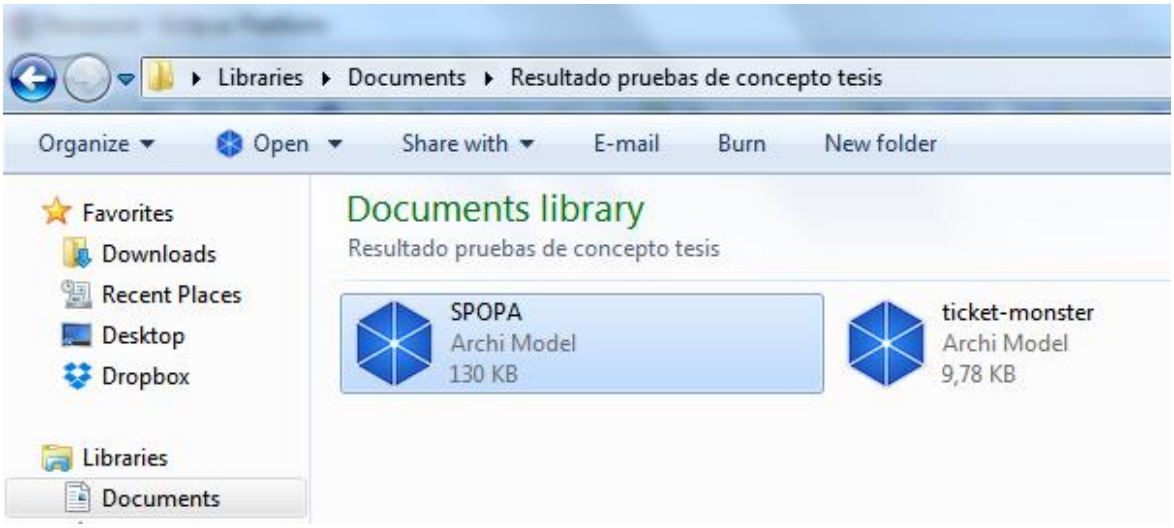


Ilustración 55 Evidencia archivo .archimate generado para SPOPA

3. Se verifica que el archivo archimate pueda ser interpretado por la herramienta .archi

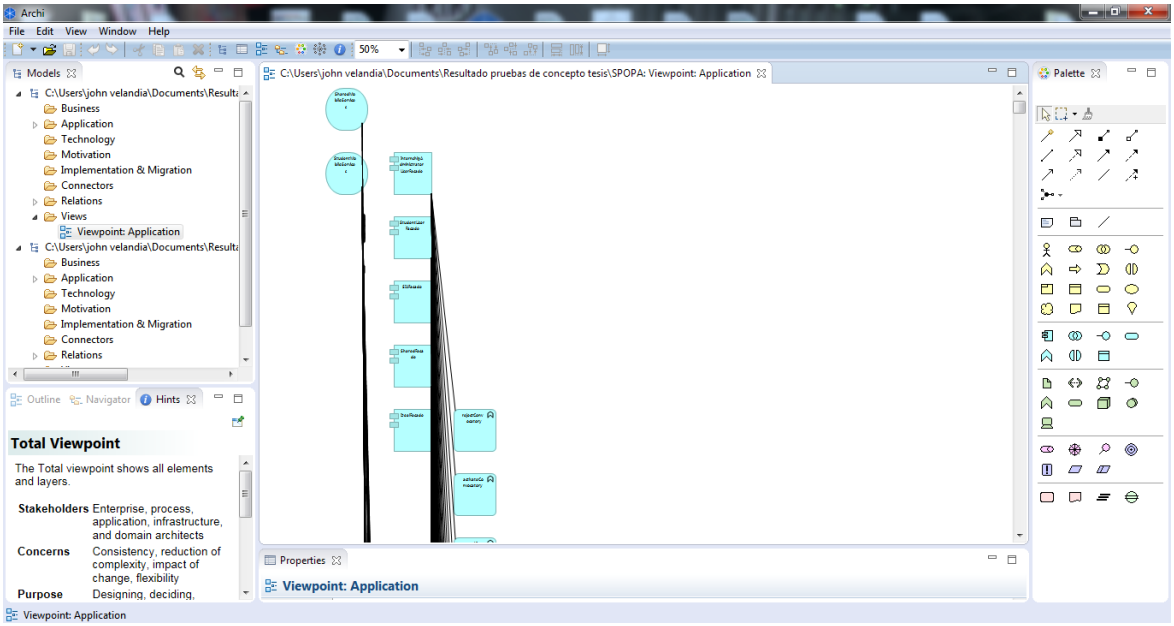


Ilustración 56 Evidencia Java2Archimate modelo generado para SPOPA

Se puede observar que por el tamaño del sistema SPOPA y la cantidad de clases Java con anotaciones J2EE, la lectura y visualización del modelo se dificulta bastante; por lo anterior, se realizó una organización manual del modelo generado junto con la eliminación de algunos elementos.

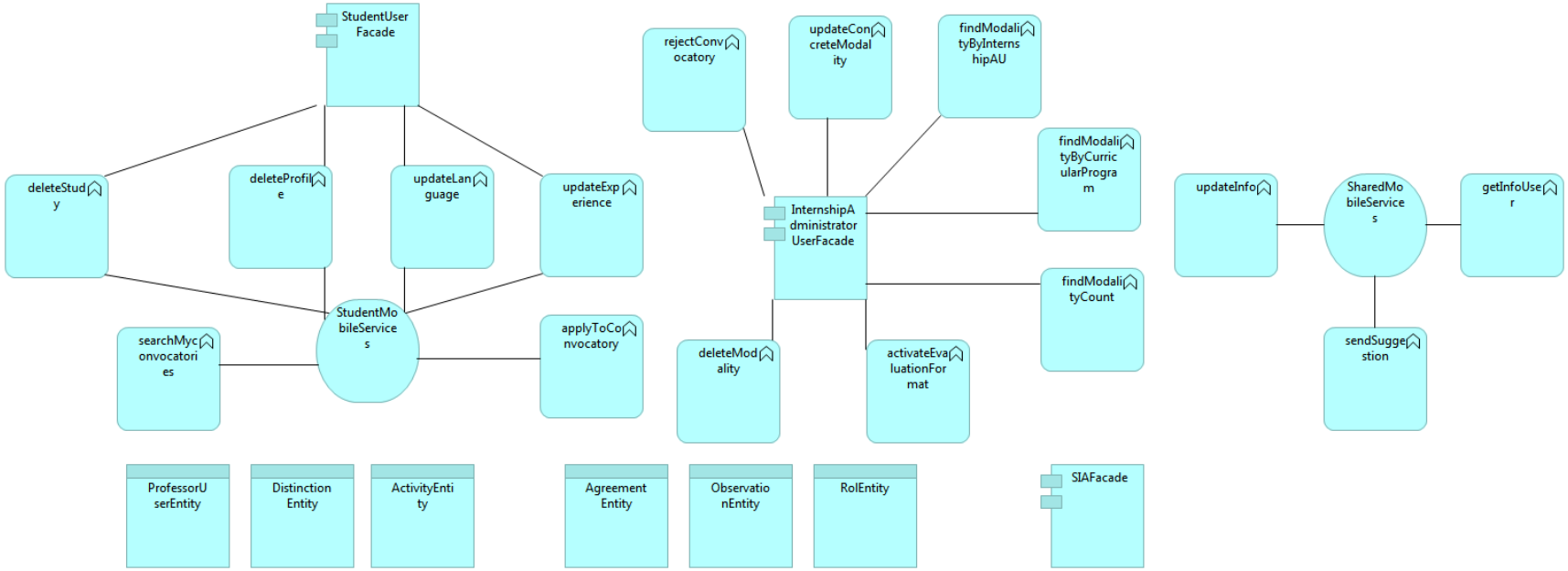


Ilustración 57 Modelo archimate generado para SPOPA organizado y filtrado manualmente

4.3 Análisis de los resultados obtenidos

1. Posterior a la comparación del código en ambas pruebas de concepto con el modelo generado, se encuentra que la equivalencia fue correcta según la lógica establecida en la Tabla 7.
2. Al realizar análisis del código fuente de la prueba de concepto número 2 contra el modelo generado, se encontró que las clases Entity identificadas con la anotación JPA como DataObject en Archimate, no están relacionadas a con otros elementos identificados de Archimate; lo anterior se debe a que se utiliza una clase VO intermediaria entre el Entity y las funciones de negocio; por lo tanto al no tener una anotación del estándar J2EE, no se visualiza en el modelo, en este caso se hace necesario el uso de la librería JTOGAF.
3. Cuando se realizó el filtrado manual en la prueba de concepto 2, se encontró que habían funciones que aparentemente eran compartidas con otras clases, algo que lógicamente no tiene sentido; sin embargo se encontró que esto ocurre porque la herramienta Java2Archimate determina elementos compartidos utilizando el nombre como identificador para los casos de encontrar elementos de tipo ApplicationCollaboration o ApplicationInteraction. Se concluye de lo anterior que el significado de ese caso es que los componentes tienen una función implementada con el mismo nombre.

5. Conclusiones

5.1 Prueba de concepto

- La propuesta planteada es viable y se valida con la herramienta Java2Archimate que realiza las transformaciones correctamente siguiendo lo planteado en la Tabla 7 Correspondencia entre Archimate en el Dominio de Aplicaciones y J2EE.
- Hay escenarios en los que se visualizan elementos Archimate sueltos o sin relaciones; en estos casos, la visualización muestra que dichos elementos no son utilizados por ninguna otra clase que haya sido identificada de acuerdo a la Tabla 7 Correspondencia entre Archimate en el Dominio de Aplicaciones y J2EE y por lo tanto, se sugiere que el desarrollador de software verifique si efectivamente esta clase es utilizada en el proyecto.
- Los componentes Archimate que se visualizan sueltos no significan que no estén siendo utilizados por otra clase; esto depende de las anotaciones.
- Con sistemas de información grandes, la documentación que se genera es difícil de interpretar por la cantidad de elementos que se determinan, por lo tanto se sugiere que al realizar el levantamiento del Baseline de arquitectura en el dominio de aplicaciones, utilizar la metodología del ADM de TOGAF; en el caso particular de la prueba de concepto número 2, se debe trabajar en equipo con el Arquitecto de Negocio para definir un alcance, posteriormente seleccionar el código fuente que se desea documentar para construir el artefacto de arquitectura alineado al negocio.
- Las clases utilizadas como VO, pueden ser transformadas a un elemento Archimate de tipo DataObject. Para ello se puede utilizar la librería JTOGAF con la anotación @DataObject en las clases VO y excluir del proyecto las clases con la anotación @Entity lo cual queda como labor del desarrollador de software.
-

5.2 Beneficios

- Reducción en el tiempo de obtención de información para un baseline de arquitectura empresarial en el dominio de aplicaciones debido a que la herramienta genera automáticamente la documentación en Archimate del código fuente seleccionado.
- Documentación semi-automática de arquitectura empresarial en el dominio de aplicaciones; se habla de semi-automática debido a que alguien debe seleccionar las clases Java que serán documentadas en Archimate.
- Repositorio de arquitectura empresarial actualizable fácilmente.
- La herramienta hace lectura del código Java independientemente del IDE en el que haya sido desarrollada.

5.3 Restricciones

- La documentación de arquitectura empresarial que se genera solo está enfocada en el dominio de aplicaciones; para documentar los otros dominios y ver la integración es necesaria la intervención de un arquitecto líder o un arquitecto de negocio y uno de infraestructura.
- La documentación generada está representada en un modelo Archimate, lo cual limita a trabajar únicamente en este lenguaje de modelado un proceso de arquitectura empresarial.
- Las clases Java que serán documentadas, deben ser importadas en Eclipse Luna a través de un Java Project.
- La versión actual de la herramienta desarrollada solo permite documentar un proyecto Java a la vez y no permite ver las interacciones con otros proyectos Java.
- La herramienta desarrollada no tiene soporte para la regeneración del modelo generado inicialmente; es decir, que si un usuario modifica el modelo, al regenerar el modelo se pierden los cambios realizados por el usuario en su modelo.

5.4 Trabajo futuro

Se proponen los siguientes temas para desarrollar en uno o varios trabajos de investigación:

- Mejora y optimización del algoritmo encargado del posicionamiento espacial de los elementos Archimate encontrados en el código. Al generar el modelo en Archimate, un algoritmo básico se encarga de ubicar espacialmente las “cajas” que representan los elementos del modelo, se propone mejorar este algoritmo.
- Lectura simultanea de varios proyectos Java permitiendo la visualización de las interacciones; por ejemplo; cómo interactúan uno o varios EJB Projects con un Web-Project.
- Soporte en la regeneración de modelos. Si un usuario modifica el modelo generado, tener zonas protegidas para que al regenerar el modelo no se pierdan los cambios realizados por el usuario.
- Ampliar la lectura de clases sin necesidad de implementar la librería JTOGAF: La librería JTOGAF contiene un conjunto de anotaciones que permiten a la herramienta Java2Archimate facilitar la lectura y transformación del código a un artefacto Archimate en el dominio de aplicaciones; sin embargo, el uso de esta librería queda sujeta a la experticia y conocimiento del desarrollador de software; por lo tanto, un trabajo futuro es eliminar por completo el uso de la librería, buscando alternativas adicionales para identificar clases candidatas a transformar por ejemplo, el caso de las clases VO.

5.5 Conclusiones generales

De acuerdo a los objetivos planteados para el presente proyecto de investigación, se considera que el objetivo general se cumple satisfactoriamente debido a que se presenta una propuesta concreta con una herramienta para la transformación de un proyecto Java en un modelo Archimate del dominio de aplicaciones utilizando los conceptos de Model Driven Reverse Engineering. Adicionalmente se cumplen cada uno de los objetivos específicos de la siguiente forma:

- **Objetivo específico 1:** Establecer una propuesta para representar elementos del metamodelo Java en elementos del modelo Archimate por medio de la conceptualización de los mismos. Este objetivo se cumple en la fase de análisis de la metodología planteada; se puede ver reflejado en el capítulo 3.4 del presente trabajo de investigación.

- **Objetivo específico 2:** Construir una forma de comunicación entre Java y Archimate para facilitar la identificación de los elementos Java a transformar. Este objetivo se cumple en la fase de análisis de la metodología planteada; se puede ver reflejado en el capítulo 3.4.4.

- **Objetivo específico 3:** Desarrollar una herramienta utilizando Model Driven Reverse Engineering. Este objetivo se cumple en la fase de “Desarrollo de las transformaciones” de la metodología planteada; se puede ver reflejado en el capítulo 3.5 y en el código fuente adjunto en medio magnético al presente trabajo de investigación.

- **Objetivo específico 4:** Realizar una prueba de concepto. Este objetivo se cumple en la fase “Pruebas” de la metodología planteada; se puede ver reflejado este logro en el capítulo 4 del presente trabajo de investigación.

A. Anexo: Configuración Workspace

Instalación de Modisco:

Para la instalación de Modisco es necesario previamente haber instalado el JDK 8 y posteriormente haber instalado y configurado Eclipse Luna. A continuación se indican los pasos realizados para la instalación del plugin:

1. Abrir Eclipse e Ingresar por el menú: Help / Install new software
2. Al hacer clic en la opción Install new software, se debe seleccionar el siguiente repositorio: http://download.eclipse.org/releases/luna/
3. Antes de realizar cualquier búsqueda, se debe validar que en la parte inferior de la ventana, solo aparezca chequeada la opción “Group ítems by category”.
4. Posteriormente, en el cuadro de texto que permite buscar complementos, escribir “Modisco”.
5. En los resultados que aparecen en la ventana, chequear “MoDisco SDK”. En caso de que aparezcan varios resultados, se debe seleccionar la que tenga mayor versión.

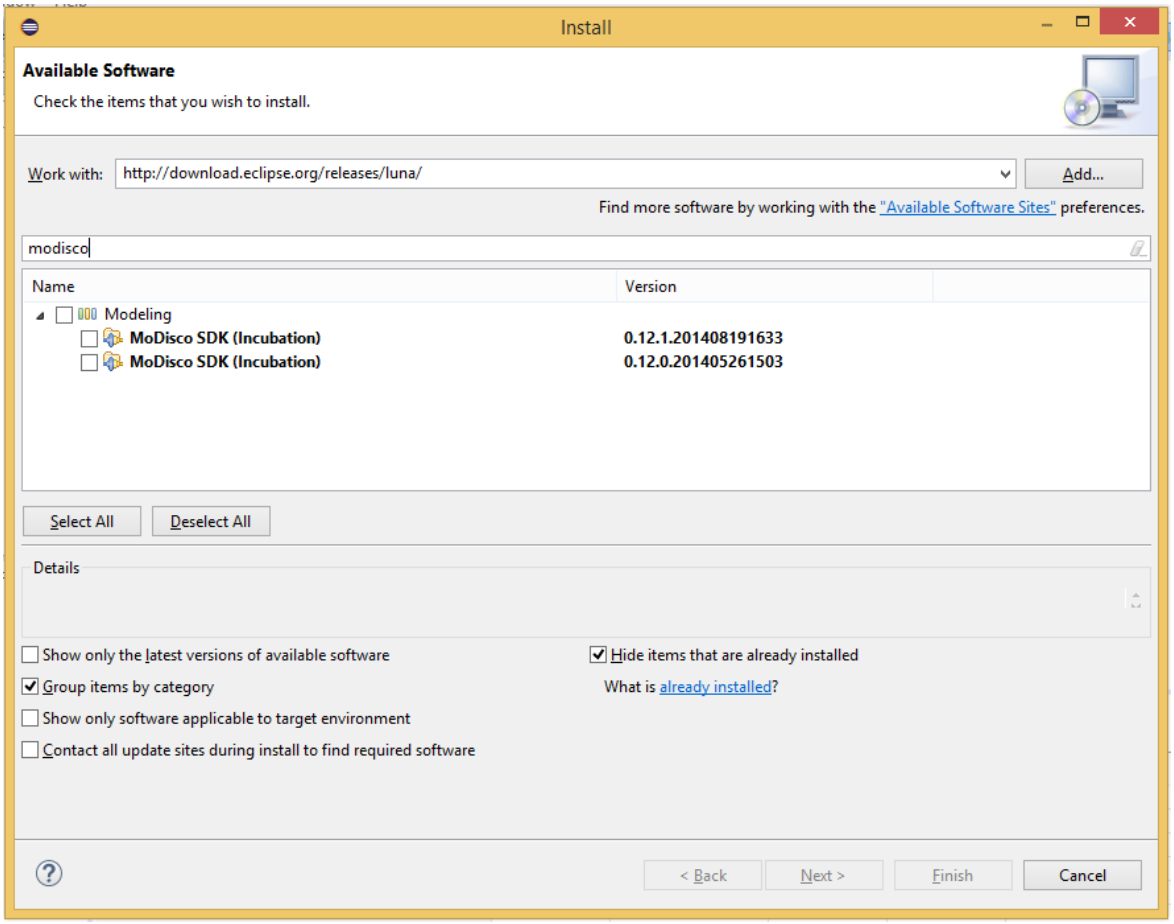


Ilustración 58 Instalación del plugin MODISCO

Instalación de Xtend

Para la instalación de Xtend, se realizaron los siguientes pasos:

1. Abrir Eclipse e ingresar por el menú: Help / Eclipse Market Place
2. En la ventana que se abre al ingresar a la opción, ubicarse en la pestaña Search y en el campo Find escribir “Xtend”.
3. En el panel de resultados que muestra la ventana, hacer clic sobre el botón Install que se encuentre ubicado sobre la casilla “Eclipse Xtend 2.8.3”. En la Ilustración 59 se visualiza la forma de realizarlo; en este caso, como el plugin ya se encuentra instalado, aparecen dos opciones: “Update” y “Uninstall”.

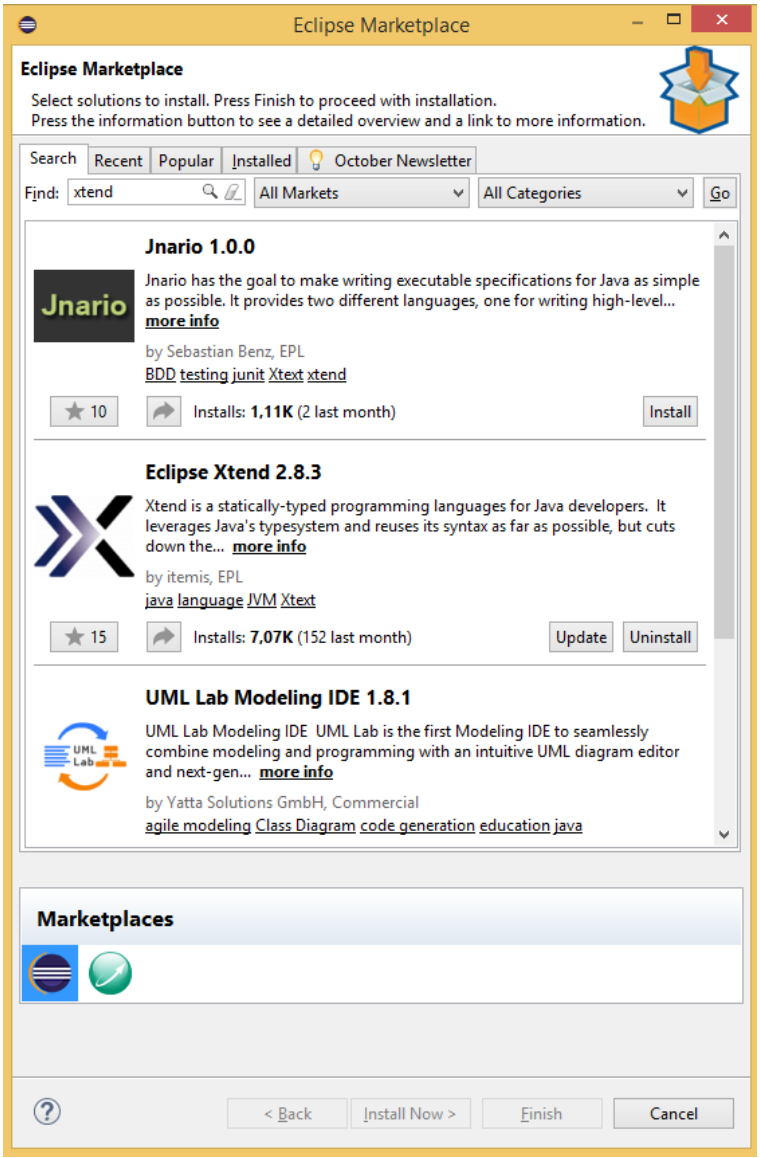


Ilustración 59 Instalación de Xtend en Eclipse Luna

Instalación de JBoss Tools

Teniendo en cuenta que en la selección del prototipo se escogió trabajar con la aplicación ticket-monster, se tomó la decisión de instalar Jboss Tools para integrar las herramientas de desarrollo de JBoss y configurar más fácilmente el proyecto J2EE ticket-monster. Para ello, dentro del Eclipse Market Place, se buscó “Jboss tolos”; en los resultados de la búsqueda se instalaron “JBoss Tools 4.2.3 Final” y “Red Hat JBoss Developer Studio 8.1.0 GA”.

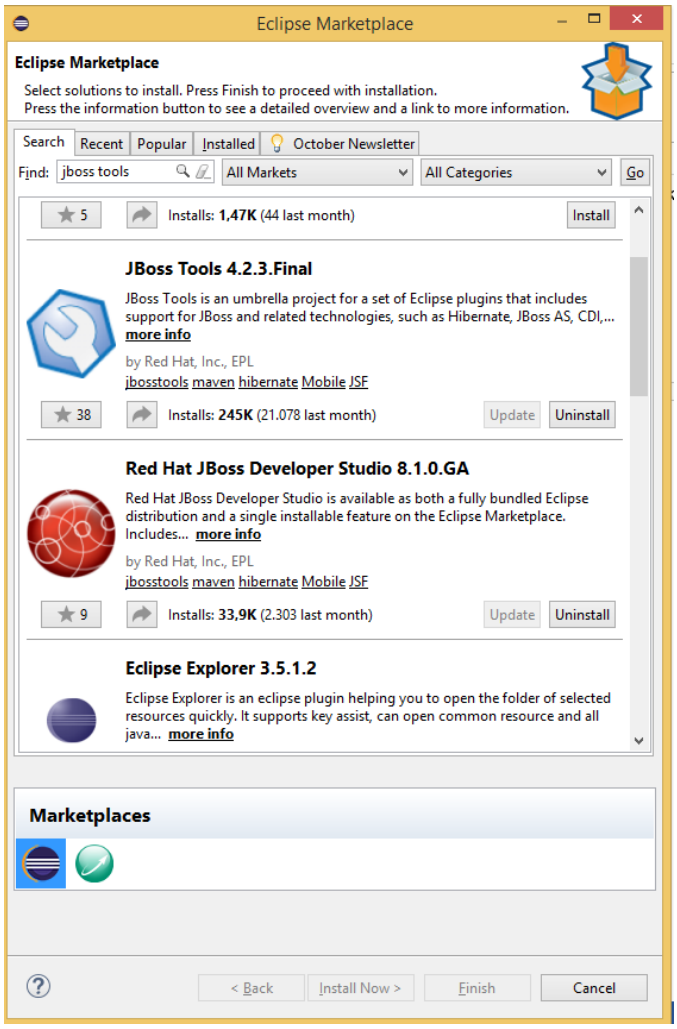


Ilustración 60 Instalación de JBoss Developer Studio y JBoss Tools en Eclipse Luna

Instalación JBoss EAP 6.3

Para la instalación del servidor de aplicaciones JBoss EAP 6.3 se realizaron los siguientes pasos:

1. Ingresar a la URL: http://www.jboss.org/products/eap/download/
2. Ir a la sección View Older Downloads
3. Descargar la versión 6.3.0 GA haciendo clic sobre el link “Installer”. Es importante estar registrado en la página debido a que para descargar el archivo se solicitan las credenciales.

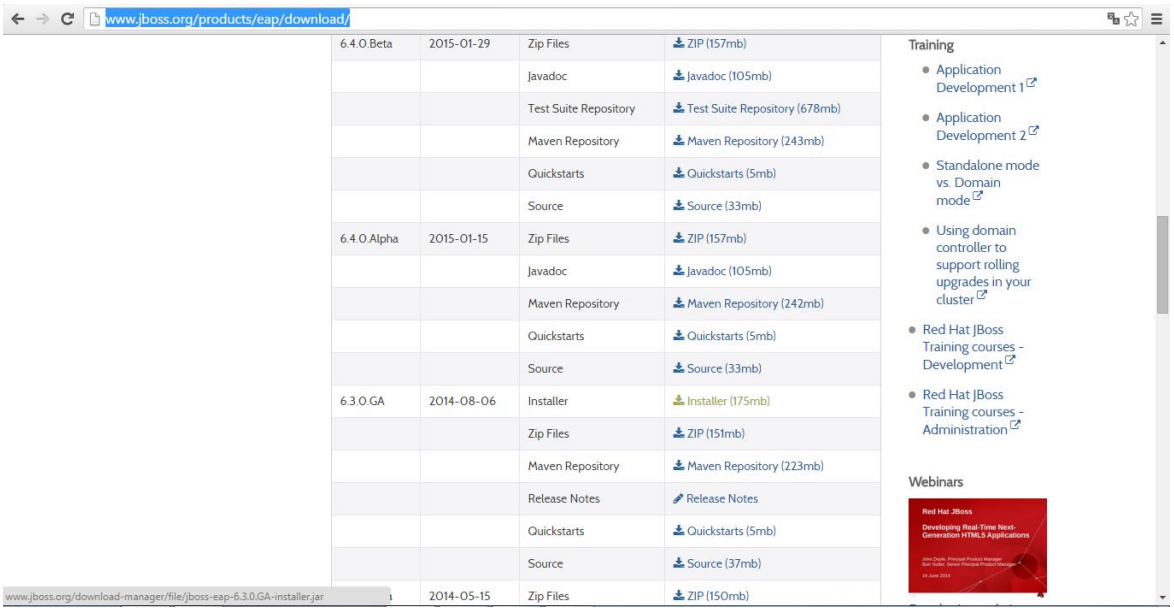


Ilustración 61 Descarga de JBoss EAP 6.3

4. El archivo .jar descargado se debe ejecutar haciendo doble clic sobre el mismo; continuar con el proceso de instalación normal. Se hace énfasis en el paso de creación de usuario administrador debido a que en la configuración del proyecto Ticket-Monster se requieren las credenciales.

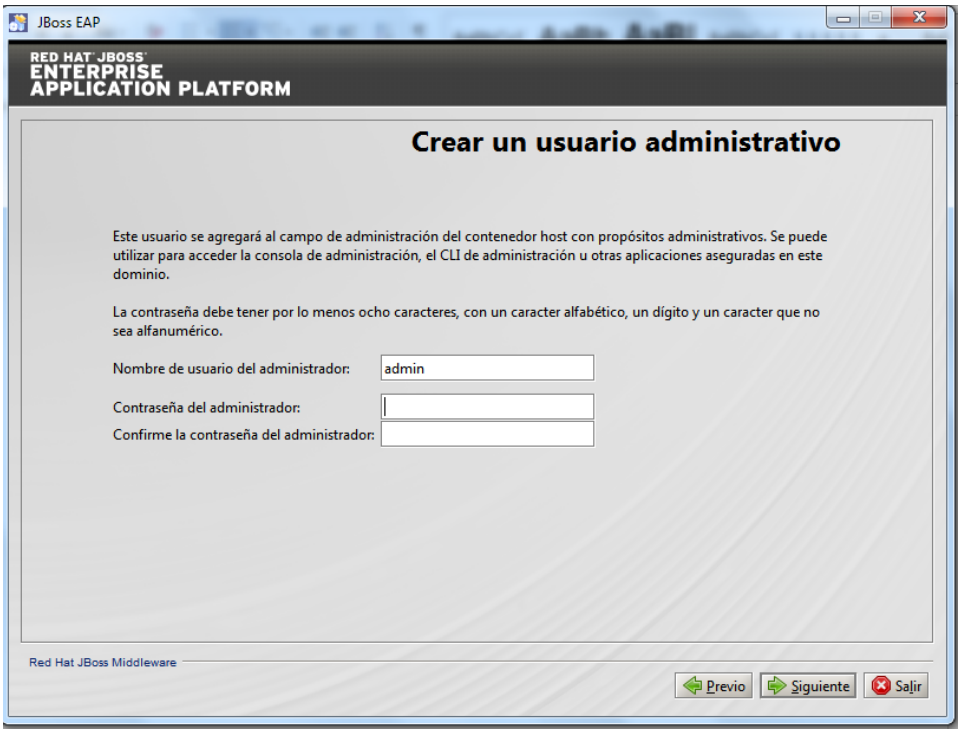


Ilustración 62 Configuración usuario administrador JBoss EAP 6.3

5. Al hacer clic en el botón Siguiente en el paso visualizado en la Ilustración 63, se inicia el proceso de instalación.

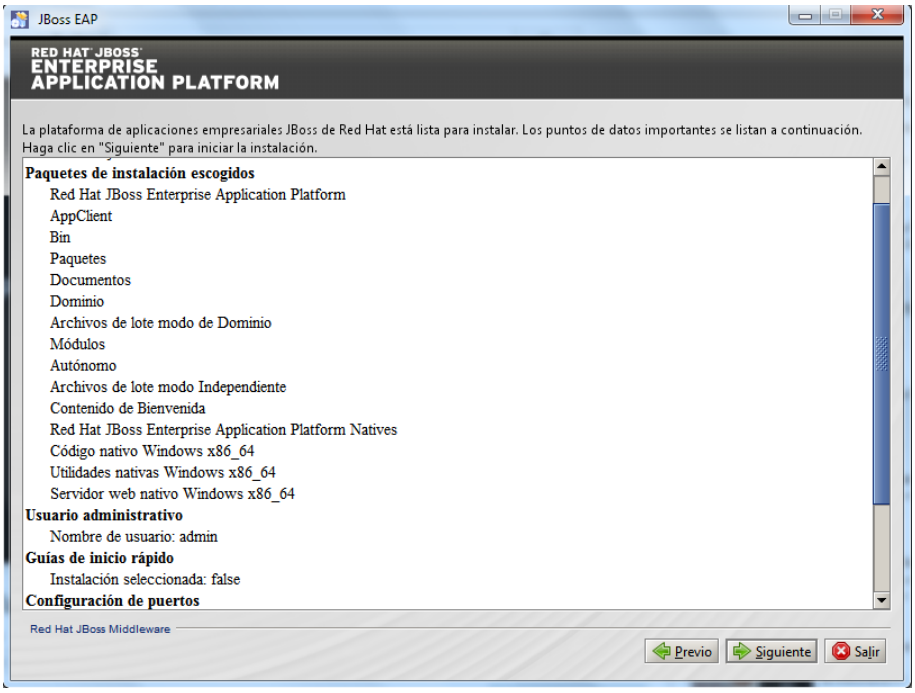


Ilustración 63 Confirmación configuración instalación JBoss EAP 6.3

6. En los siguientes pasos se deben dejar marcadas las opciones por defecto.

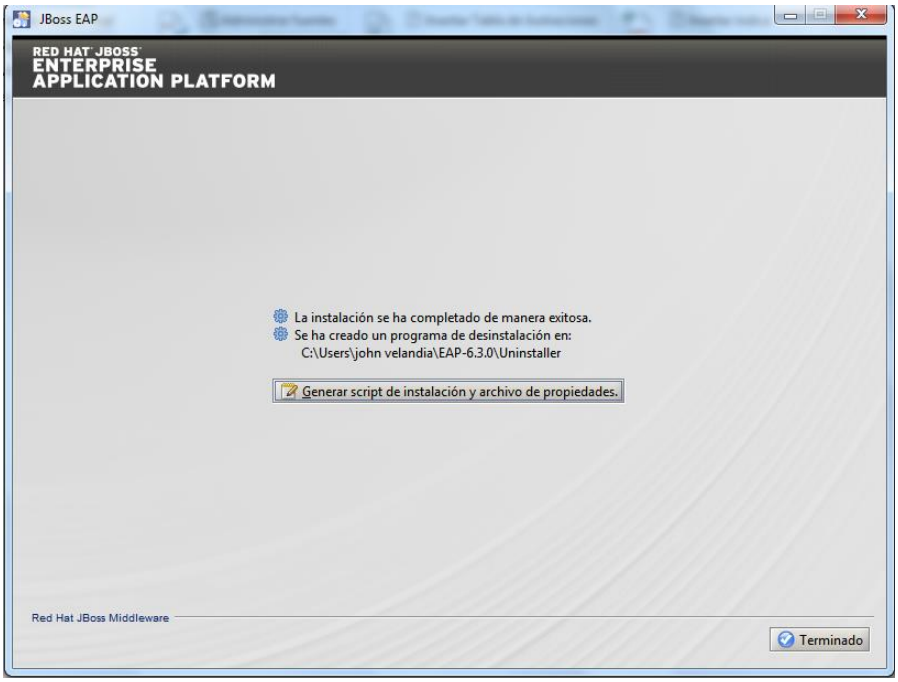


Ilustración 64 Instalación éxitos JBoss EAP 6.3

7. Al finalizar todos los pasos y la configuración del servidor de aplicaciones, aparecerá la ventana de la Ilustración 64 en la cual solo se debe hacer clic en la opción terminar.
8. Para la integración del servidor de aplicaciones en el IDE Eclipse, se debe ingresar a Eclipse y posteriormente ir a la opción Window / Preferences.
9. En la ventana que aparece, se debe ir al ítem JBoss Tools / JBoss Runtime Detection.
10. Hacer clic en el botón Add y seleccionar la ruta en donde quedó instalado el servidor de aplicaciones.

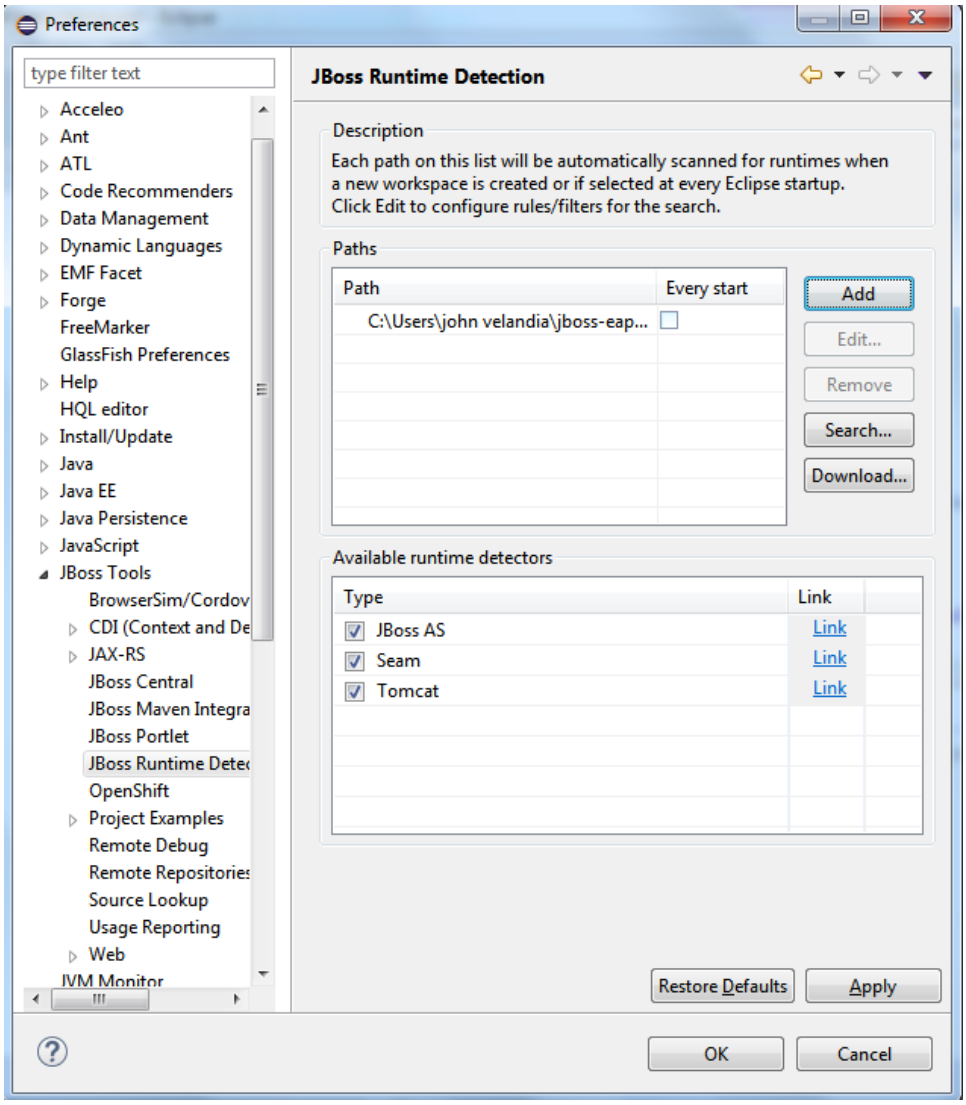


Ilustración 65 Integración JBoss EAP 6.3 con IDE Eclipse

Configuración del proyecto Ticket – Monster

Para la instalación y configuración del proyecto Ticket-Monster, se realizaron los siguientes pasos:

1. Crear un proyecto WEB en el IDE Eclipse ingresando por la opción File / New / Other.
2. En el Wizard, buscar la carpeta JBoss Central y seleccionar Java EE Web Project como se visualiza en la Ilustración 66

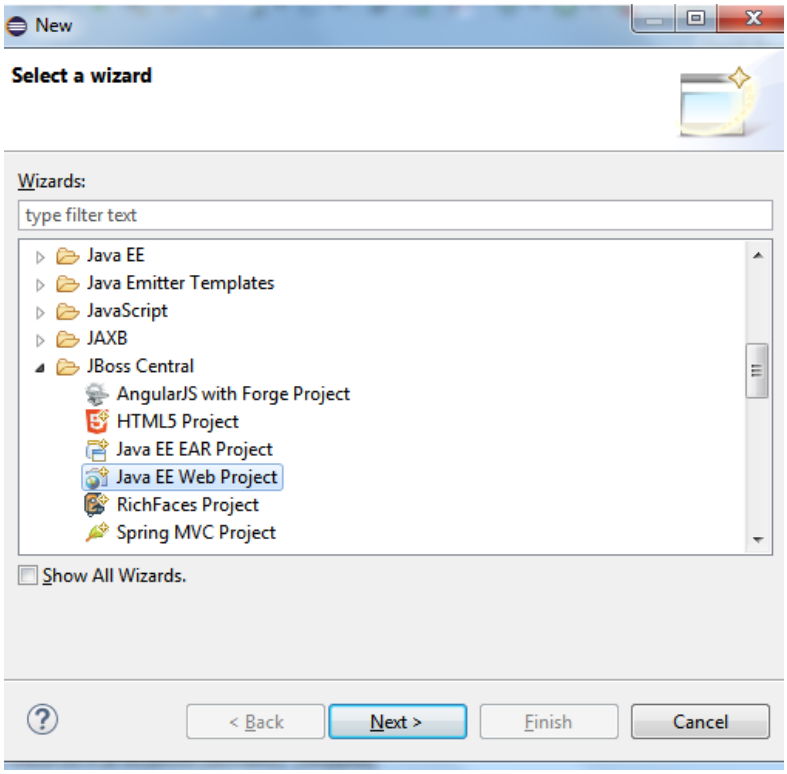


Ilustración 66 Creación proyecto Java EE Web para configurar Ticket-Monster

3. En la siguiente ventana, en Target Runtime dejar seleccionado jboss-eap-6.3 Runtime y hacer clic en Next.
4. En la ventana que se visualiza en la Ilustración 67, se debe tener cuidado con el nombre del proyecto, este debe ser ticket-monster; en el campo Package se debe dejar el que está por defecto seguido de ".ticketmonster".

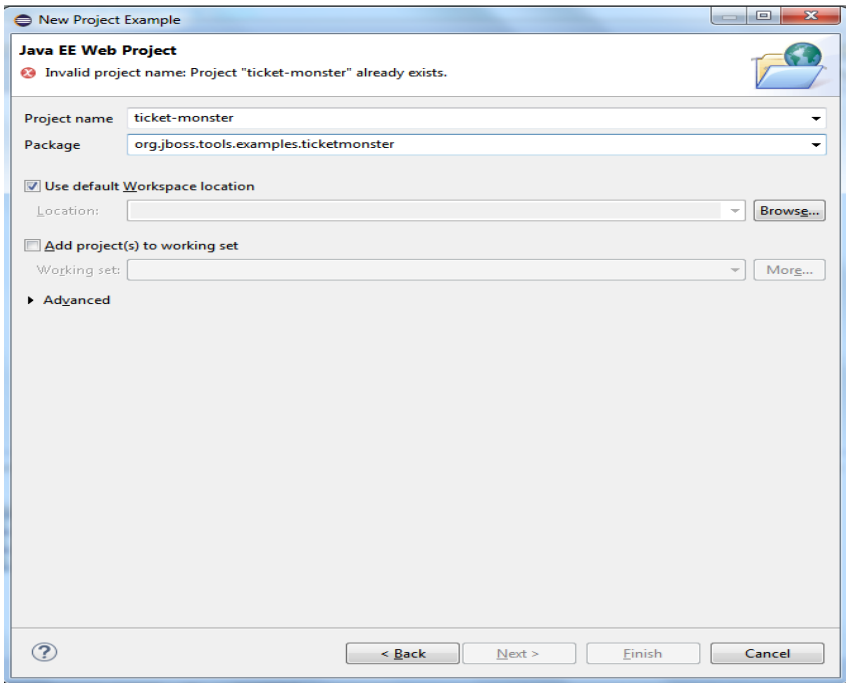


Ilustración 67 Creación proyecto Ticket-monster

5. Posteriormente se debe hacer clic en el botón Finish; al realizar esta acción automáticamente se descarga y configura el proyecto ticket-monster para ser desplegado.

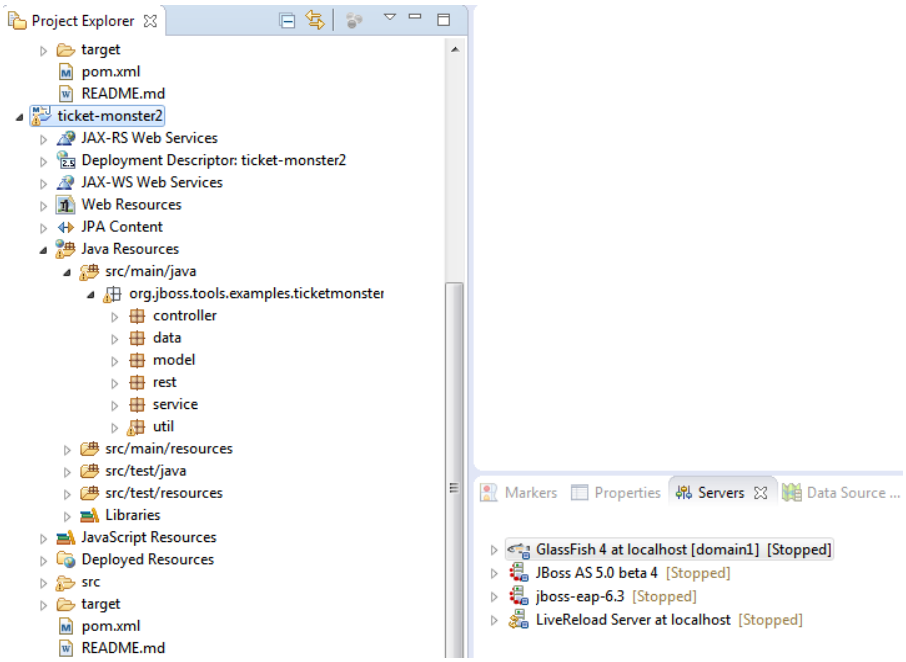


Ilustración 68 Proyecto Ticket-monster

En caso de que se requiera información adicional para la configuración del proyecto Ticket-Monster en el IDE Eclipse Luna, se recomienda revisar la referencia bibliográfica número [52].

6. Bibliografía

[1] J. Rumbaugh, I. Jacobson y G. Booch, Unified Modeling Language Reference Manual, The (2nd Edition), Pearson, 2004.

[2] L. Bettini, Implementing Domain-Specific Languages with Xtext and Xtend, PACKT, 2013.

[3] H. Bruneliere, J. Cabot, F. Jouault y F. Madiot, «Modisco: a generic and extensible framework for model driven reverse engineering,» *ASE '10 Proceedings of the IEEE/ACM international conference on Automated software engineering,* pp. 173 - 174, 2010.

[4] M. Lacity y R. Hirscheim, «Information Systems Outsourcing: Myths, Metaphors, and Realities,» 1993, pp. 99 - 107.

[5] J. Zavala Ruiz, «¿Por qué fracasan los proyectos de software?; un enfoque organizacional,» de *Congreso Nacional de Software libre.*, Mexico, D.F., 2004.

[6] F. Brooks, «No Silver Bullet; Escence and Accidents of Software Engineering,» *Computer Magazine,* 1987.

[7] Standish Group, «The Chaos Report 1994, Extreme CHAOS, summary,» Standish Group, 1994. [En línea]. Available: http://www.standishgroup.com/.

[8] D. Ortega, E. Uzcátegui y M. Guevara, «EAIF: Un framework de arquitectura empresarial orientado a servicio en correspondencia con MDA,» *Departamento de Computación, Facultad de Ciencias y Tecnología Universidad de Carabobo, Venezuela,* 2012.

[9] T. J. Blevins, J. Spencer y F. Waskiewiez, «The Power of Synergy. The Open Group and OMG,» [En línea]. Available: http://www.opengroup.org/cio/MDA-ADM/MDA-TOGAF-R1-070904.pdf. [Último acceso: 2016 Marzo 28].

[10] R. Seguel, «Is there a relation among Archimate and MDA?,» [En línea]. Available: http://rseguel.bpm-latam.org/2009/12/is-there-relation-among-archimate-and.html. [Último acceso: 22 Noviembre 2015].

[11] M. D. Arango Serna, J. E. Londoño Salazar y J. A. Zapata Cortés, «Arquitectura Empresarial - Una Visión General,» *Revista Ingenierías Universidad de Medellín,* vol. 9, nº 16, pp. 101 - 111, 2010.

[12] A. Josey, TOGAF Version 9 – A Pocket Guide, The Open Group, 2009.

[13] Sergio Sotelo, «Arquitectura Empresarial y Frameworks de Industria,» IBM, [En línea]. Available: https://www.ibm.com/developerworks/community/files/form/anonymous/api/library/9febf797-1183-40b9-a444-1c0eec37576d/document/56c06699-cc3e-47e5-b4f8-13b00b5308b6/media/02%20Arch%20Summit%20Enterprise%20Architecture.pdf. [Último acceso: 03 04 2016].

[14] Zachman, «A Framework for Information Systems Architecture,» *IBM Systems Journal,* vol. 26, nº 3, 1987.

[15] Zachman, «Concepts of the framework for enterprise architecture, background, description and utility,» 2004. [En línea]. Available: http://slashdemocracy.org/links/files/Zachman_ConceptsforFrameworkforEA.pdf. [Último acceso: 22 Noviembre 2015].

[16] Microsoft, «A Comparison of the Top Four Enterprise-Architecture Methodologies,» Microsoft, Mayo 2007. [En línea]. Available: https://msdn.microsoft.com/en-us/library/bb466232.aspx#eacompar_topic5. [Último acceso: 03 04 2016].

[17] A. Josey, «An Introduction to TOGAF 9.1.,» *White Paper Open Group Document,* nº Document No W118, 2011.

[18] J. M. Calvo, *Arquitectura Empresarial 8.0. Presentación Clase Arquitectura de Software. Especialización en Ingeniería de Software,* Bogotá: Universidad Distrital Francisco José de Caldas, 2012.

[19] The Open Group, «TOGAF 9.1 > Part IV: Architecture Content Framework > Introduction,» [En línea]. Available: http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap33.html. [Último acceso: 22 Noviembre 2015].

[20] M. Lankhorst y H. van Drunen, «Enterprise Architecture Development and Modelling. Combining TOGAF and Archimate,» 2007.

[21] The Open Group, «TOGAF 9.1 > Part V: Enterprise Continuum and Tools > Enterprise Continuum,» The Open Group, [En línea]. Available: http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap39.html. [Último acceso: 22 Noviembre 2015].

[22] The Open Group, «About us,» 3 Noviembre 2015. [En línea]. Available: http://www.opengroup.org/standards/ea.

[23] The Open Group, «About Archimate,» The Open Group, [En línea]. Available: http://www.opengroup.org/subjectareas/enterprise/archimate. [Último acceso: 22 Noviembre 2015].

[24] The Open Group, «Summary of Business Layer Components,» The Open Group, [En línea]. Available: http://pubs.opengroup.org/architecture/archimate2-doc/chap03.html#_Toc371945155. [Último acceso: 22 Noviembre 2015].

[25] The Open Group, «Summary of Application Layer Components,» The Open Group, [En línea]. Available: http://pubs.opengroup.org/architecture/archimate2-doc/chap04.html#_Toc371945177. [Último acceso: 22 Noviembre 2015].

[26] The Open Group, «Summary of Technology Layer Concepts,» The Open Group, [En línea]. Available: http://pubs.opengroup.org/architecture/archimate2-doc/chap05.html#_Toc371945190. [Último acceso: 22 Noviembre 2015].

[27] H. Jonkers,, H. van den Berg y M. E. Iacob, «ArchiMate® Extension for Modeling the TOGAF Implementation and Migration Phases,» *The Open Group,* 2010.

[28] The Open Group, «Motivation Extension,» The Open Group, [En línea]. Available: http://pubs.opengroup.org/architecture/archimate2-doc/chap10.html. [Último acceso: 22 Noviembre 2015].

[29] Object Management Group, «OMG,» 16 Septiembre 2014. [En línea]. Available: http://www.omg.org/gettingstarted/gettingstartedindex.htm.

[30] OMG, «Model-Based Engineering,» [En línea]. Available: http://www.modelbasedengineering.com/glossary/. [Último acceso: 22 Noviembre 2015].

[31] J. Bézivin, «In Search of a Basic Principle for Model Driven Engineering,» *The European Journal of Informatics Professional,* vol. 2, pp. 21 - 24, 2004.

[32] C. Atkinson y T. Kuhne, «Model-Driven development: a metamodeling foundation,» *IEEE,* vol. 20, pp. 36 - 41, 2003.

[33] J. Bézivin y O. Gerbe, «Towards a precise definition of the OMG/MDA framework,» *Automated Software Engineering, 2001. (ASE 2001). Proceedings. 16th Annual International Conference on,* pp. 273 - 280, 2001.

[34] H. A. Muller, M. A. Orgun, S. R. Tilley y J. S. Uhl, «A reverse-engineering approach to subsystem structure identification,» *Journal of Software Maintenance: Research and Practice,* vol. 5, pp. 181 - 204, 2006.

[35] F. Barraza, «Modelado y diseño de arquitectura de software,» [En línea]. Available: http://cic.javerianacali.edu.co/wiki/lib/exe/fetch.php?media=materias:s2_conceptosdemodelado.pdf. [Último acceso: 22 Noviembre 2015].

[36] S. Rugaber y K. Stirewalt, «Model-Driven Reverse Engineering,» *IEEE Computer Society,* vol. 21, pp. 45 - 53, 2004.

[37] J. A. Cruz Castelblanco, «Model-Driven Software Development,» de *A modular model-driven engineering approach to reduce efforts in software development teams*, Bogotá, Universidad Nacional de Colombia, 2014, p. 9.

[38] M. Farwick, C. Schweda, R. Breu y I. Hanshke, «A situational method for semi-automated Enterprise Architecture Documentation,» *Springer-Verlag Berlin Heidelberg,* 2014.

[39] A. Brunnert, C. Vogele y H. Krcmar, «Automatic Performance Model Generation for Java Enterprise Edition (EE) Applications. Computer performance Engineering,» *Lecture Notes in computer science,* vol. 8168, pp. 74 - 88, 2013.

[40] H. van den Berg, A. Bielowski, G. Dijk, H. van Drunen, G. Faber, J. van Gijsen, R. de Kok, G. Osting, W. Schut, F. Smit, J. van de Wetering, E. Willemsz, H. Bosma, F. Langeveld, J. Luijpers y R. Slagter, «ArchiMate Made Practical,» *Company Workgroup ArchiMate Usage/Tools.,* 2013.

[41] B. Bousetta, O. El Beggar y T. Gadi, «A methodology for CIM modelling and its transformation to PIM,» *Journal of Information Engineering and Applications,* 2013.

[42] M. Hauder, F. Matthes y S. Roth, «Challenges for Automated Enterprise Architecture Documentation. Trends in Enterprise Architecture Research and

Practice-Driven Research on Enterprise Transformation,» *Lecture Notes in Business Information Processing,* vol. 131, pp. 21 - 39, 2012.

[43] H. Holm, M. Buschle, R. Legerstrom y M. Ekstedt, «Automated data collection for enterprise architecture models,» *Springer-Verlag,* 2012.

[44] The Open Group, de *Archimate 2.0 Specification*, The Open Group, 2012, pp. 17 - 18.

[45] M. Buschle, S. Grunow, F. Mathes, M. Ekstedt, M. Hauder y S. Roth, «Automating Enterprise Architecture Documentation using an Enterprise Service Bus,» *18th Americas Conference on Information Systems,* 2012.

[46] S. de Kinderen, K. Gaaloul y H. A. Proper, «Bridging value modelling to ArchiMate via transaction modelling,» *Springer-Verlag Berlin Heidelberg,* 2012.

[47] M. Buschle, H. Holm, T. Sommestad, M. Ekstedt y K. Shahzad, «A Tool for Automatic Enterprise Architecture Modeling,» *Lecture Notes in Business Information Processing,* vol. 107, pp. 1 - 15, 2012.

[48] H. Jonkers, E. Proper, M. Lankhorst y M. E. Iacob, «Archimate for integrated modelling Throughout the Architecture Development and Implementation Cycle,» *IEEE Conference on Commerce and Enterprise Computing,* 2011.

[49] C. Peña y J. Villalobos, «An MDE approach to design enterprise architecture viewpoints,» *12th IEEE International Conference on Commerce and Enterprise Computing,* vol. 12th, 2010.

[50] R. Blom, «ArchiMate®: Adding value to TOGAF™,» 2010. [En línea]. Available: http://www.opengroup.org/public/member/proceedings/q110/archimate.pdf. [Último acceso: 22 Noviembre 2015].

[51] W. Stein, «Enterprise Architecture Model Transformation (Master Thesis),» University of Twente, 2009.

[52] S. Buckl, A. Erns y J. Lankes, «Generating visualizations of Enterprise Architectures using Model Transformations,» *Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures EMISA,* 2007.

[53] A. Stam, J. Jacob, F. de Boer, M. Bonsangue y L. van der Torre, «Using XML Transformations for Enterprise Architectures,» *Lecture Notes in Computer Science,* vol. 4313, pp. 42 - 56, 2006.

[54] RedHat, «JBoss Ticket Monster,» [En línea]. Available: https://www.jboss.org/ticket-monster/introduction/. [Último acceso: 27 Noviembre 2015].