Modelo de requerimientos y de funcionalidad de software basado en MDA y UML para la gestión de proyectos y convenios globales

Luis Alberto Lujan Campos

Escuela de Post Grado, Universidad Nacional Federico Villarreal, Jr. Camana 1014, Lima, Perú

Recibido el 01 de junio del 2017. Aceptado el 1 de julio del 2017

Resumen

El propósito es elaborar modelos de la arquitectura que son independientes de la plataforma de software para la gestión de información de proyectos y convenios globales. Que sean orientados por el enfoque Model Driven Architecture (MDA, Arquitectura Dirigida por Modelos) que puede ayudar a resolver al mismo tiempo tanto los problemas de la industrialización y las infraestructuras de software en constante cambio [1]. Este enfoque de arquitectura de software promueve la separación de la especificación de la funcionalidad de la implementación de esta funcionalidad en plataformas específicas. En la arquitectura del sistema de software para la gestión de proyectos y convenios globales se determinó los módulos principales: módulo de cooperantes, módulo de datos del proyecto o convenio, módulo de presupuesto, módulo de finanzas y módulo de seguimiento de actividades y se logró elaborar los siguientes modelos: requisitos, estructura de clases de objetos y de secuencia. Estos modelos son los conductores primarios en todos los aspectos del desarrollo de software. Los modelos tienen diversas vistas como: Modelo de la estructura del sistema y de la funcionalidad de los módulos. Para la elaboración de los modelos se ha utilizado el Unified Modeling Language (UML, Lenguaje Unificado de Modelado). UML es un lenguaje de modelado y no un método, es la notación (principalmente gráfica) del que se valen los métodos para expresar los diseños y que en el proceso de software es la orientación que nos dan sobre los pasos a seguir para hacer el diseño [2].

Descriptores: *Arquitectura Dirigida por Modelos, Lenguaje Unificado de Modelado, Diseño de software*

Abstract

The purpose is to develop architecture models that are independent of the software platform for the management of information of global projects and agreements. That is guided by the Model Driven Architecture (MDA) approach that may help solving at the same time both problems of industrialization and ever-changing software infrastructures [1]. This software architecture approach promotes the separation of the specification from the functionality of the implementation of this functionality into specific platforms. In the architecture of the software system for the management of global projects and conventions, the main modules were determined: cooperators module, data module of the project or agreement, budget module, finance module and activity tracking module and elaboration The following models: requirements, structure of object classes and sequence. These models are the primary drivers in all aspects of software development. The models have different views such as: Model of the structure of the system and the functionality of the modules. For the elaboration of the models the Unified Modeling Language (UML) has been used. UML is a modeling language and not a method, it is in the notation (mainly graph) of which the methods are used to express the designs and that in the software process is the orientation that they do not give on the steps to follow to do the design [2].

Keywords: *Model Driven Architecture (MDA), Unified Modeling Language (UML), Software Design*

1. Introducción

Model Driven Architecture (MDA-Arquitectura Dirigida por Modelos) es un enfoque que tiene

como objetivo obtener modelos independientes de plataformas en las cuales se implementaría un sistema de software. Unified Modeling Language (UML-Lenguaje Unificado de Modelado) es una notación con diversos elementos gráficos que se utiliza para representar sistemas. Mediante el análisis de la gestión de información de proyectos y convenios globales, con el enfoque de MDA y con el uso del Lenguaje UML se elaboró el modelo de requisitos y modelos de la funcionalidad del software. En la investigación se diseñó modelos en base a módulos del software que son: módulo de cooperantes, módulo de datos del proyecto o convenio, módulo de presupuesto, módulo de finanzas y módulo de seguimiento de actividades. Se Obtuvo más de quince modelos elaborados mediante el uso del software Rational Rose.

La arquitectura dirigida por modelos (Model-Driven Architecture o MDA) es un acercamiento al diseño de software, la funcionalidad del sistema será definida en primer lugar como un modelo independiente de la plataforma (Platform-Independent Model o PIM) a través de un lenguaje específico para el dominio del que se trate. Dado un modelo de definición de la plataforma (Platform Definition Model o PDM), el modelo PIM puede traducirse entonces a uno o más modelos específicos de la plataforma (Platform-Specific Models o PSMs), esta teoría se utilizará en la presente investigación [3].

Una guía de desarrollo puede contribuir a ahorrar tiempo, costo y esfuerzo con la calidad que todos buscamos, reutilización al máximo del 70%, el factor de seguridad se puede lograr hasta un 99%. Por lo tanto, atributos como la reutilización y los factores de seguridad se puede mejorar de manera significativa que se traduce en la consecución de la alta calidad de los sistemas de software y reducir los costos de desarrollo de software [4]. Apoyando el diseño dirigido por modelos de la presente investigación.

La presente investigación se justificó por el aporte en el desarrollo del diseño y prototipo de software para el apoyo a las actividades de gestión de información relacionada a convenios y proyectos.

El objetivo es elaborar modelos integrados basados en MDA y UML de los requerimientos y de la funcionalidad de software para la gestión de información de proyectos y convenios globales mediante una herramienta de software.

2. Metodología

En el trabajo de investigación para la elaboración del modelo de requisitos y de los modelos de la funcionalidad del software se tuvo como fuente principal una entidad nacional. Se usó el enfoque MDA con la herramienta Rational Rose por estar diseñado con el enfoque MDA e incluye la notación UML.

Se ejecutó lo planificado según lo siguiente:
Definición general de los requerimientos.
Modelamiento de los requerimientos.
Definición de Clases de Objetos.
Modelo de Clases de Objetos.
Modelamiento de la funcionalidad de la gestión de datos de Proyectos.
Modelamiento de la funcionalidad de la gestión de datos de Unidades Ejecutoras y Direcciones.
Modelamiento de la funcionalidad de la gestión de datos del cronograma de actividades.
Modelamiento de la funcionalidad de la gestión de datos de la ejecución física.
Modelamiento de la funcionalidad de la gestión de datos de cooperantes.
Modelamiento de la funcionalidad de la gestión de datos de Presupuesto.
Modelamiento de la funcionalidad de la gestión de datos de Finanzas [5].
El modelo MDA que se utilizó se presenta a continuación brevemente.

2.1. Model Driven Architecture

OMG Model Driven Architecture ofrece un enfoque abierto y neutral a los proveedores para el reto de cambiar los negocios y la tecnología. Basado en los estándares establecidos por OMG, el MDA separa la lógica de negocios de las aplicaciones en plataformas tecnológicas [6].

Se explica en relación al enfoque para el desarrollo de software con mucho énfasis que los modelos están considerados actualmente como los elementos más importantes para el desarrollo del software, ya que ayudan a entender problemas complejos y sus soluciones potenciales a través del uso de la abstracción. Que la aplicación de Model Driven Development (MDD) mejora la productividad y calidad del proceso de desarrollo, aplicando un enfoque centrado en modelos que utiliza modelos a distinto nivel de abstracción y transformaciones entre dichos modelos. Refiriéndose al software a elaborar indica que el sistema debe ser modelado a un alto nivel de abstracción de forma independiente a la plataforma y a continuación es transformado a un modelo

especifico de la plataforma y finalmente a código [7].

Se presentan los elementos de Model Driven Architecture en la Figura 1, y a continuación se explican los modelos del MDA.

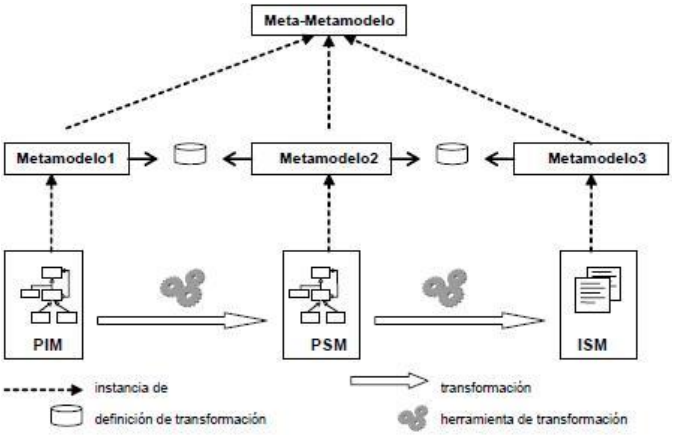

Figura 1: Arquitecta Dirigida por Modelos

Modelos Independientes de la Computación

(Computation Independent Model o CIM) también llamado modelo del dominio muestra el sistema en el ambiente en que operará y lo que se espera que haga.

Es útil no sólo para entender el problema, sino también como fuente de un vocabulario común para usar en los otros modelos.

Modelos Independientes de la Plataforma

(Platform Independent Model o PIM) tiene un alto nivel de abstracción y es independiente de cualquier tecnología de implementación.

Se centra en el funcionamiento de un sistema mientras oculta los detalles necesarios para una plataforma particular.

Este modelo muestra aquella parte de la especificación completa que no cambia de una plataforma a otra, por ejemplo J2EE.

Modelos Específicos de la Plataforma

(Platform Specific Model o PSM) combina el punto de vista independiente de la plataforma con un enfoque adicional en el detalle del uso de una plataforma específica para un sistema.

Es adecuado para especificar el sistema en términos de construcciones de implementación que son disponibles en una tecnología de implementación específica.

Por ejemplo, un PSM Enterprise JavaBeans (EJB) es un modelo del sistema en términos de estructuras EJB. Típicamente contendrá términos específicos EJB, como “home interface”, “entity bean”, etc.

Modelo Específico a la implementación

(Implementation Specific Model o ISM) muestra el sistema a nivel de código, se refiere a componentes y aplicaciones escritos en lenguajes de programación tales como Java o Eiffel [8].

La visualización, la refactorización y las capacidades de generación automática en una herramienta de fuerte desarrollo dirigido por modelos mejoran en gran medida su capacidad para tomar estas decisiones de reutilización. El enfoque Model Driven (Dirigido por Modelos) no limita la codificación, las nuevas aplicaciones se pueden empezar a escribir y entregar ya que no se empieza desde cero [9].

2.2. Unified Modeling Language

El enfoque de desarrollo basado en lenguaje unificado para modelado (UML) es muy útil en el proyecto. Proporciona un mayor nivel de abstracción (en comparación con el código fuente) y el análisis de: habilitado, verificación y comunicación del equipo en los algoritmos de control, sin entrar en detalles de implementación a nivel de código [10].

UML comprende un conjunto de herramientas para documentar el análisis de un sistema, describir y evaluar el funcionamiento de complejos sistemas, que hay ventajas claras, especialmente en términos de claridad de la comunicación y de repetitividad, si uno usa éste estándar de riguroso notación que se emplea ampliamente [11].

Considera entre otros los siguientes diagramas:

Diagrama de casos de uso

Un caso de uso es, en esencia, una interacción típica entre un usuario y un sistema de cómputo. Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, esta es una

herramienta valiosa, ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista del usuario. Esto es importante si la finalidad es crear un sistema que pueda ser utilizado por la gente en general (no solo por expertos en computación).

Diagramas de clase

Las cosas tienen atributos (características-propiedades) y que realizan determinadas acciones. Podríamos imaginar cada una de esas acciones como un conjunto de tareas.

Encontrará que las cosas se pueden agrupar en categorías (automóviles, mobiliario, lavadoras, etc.) a tales categorías las llamaremos clases.

Una clase es una categoría o grupo de cosas que tienen atributos y acciones similares.

Por ejemplo: cualquier cosa dentro de la clase lavadoras tiene atributos como la marca, el modelo, el número de serie y la capacidad. Entre las acciones de las cosas de esta clase se encuentran. "agregar ropa", "agregar detergente", "activarse" y "sacar la ropa".

Diagrama de secuencia

Los diagramas de clase representan información estática. No obstante, en un sistema funcional los objetos interactúan entre sí y tales interacciones suceden con el tiempo.

El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos. Continuando con el ejemplo de la lavadora, entre los componentes de la lavadora se encuentran una manguera de agua (para obtener agua fresca), un tambor (donde se coloca la ropa) y un sistema de drenaje. Por supuesto, estos también son objetos (como vera, un objeto puede estar conformado por otros objetos) [12].

3. Resultados

Modelo de requerimientos

En la Figura 2 y 3 se presenta parte del modelo.

Los requerimientos se representan en la Figura 3.

Modelo de funcionalidad del software

En esta sección se presentaran los principales modelos (figuras) de los módulos principales ya que son más de quince.

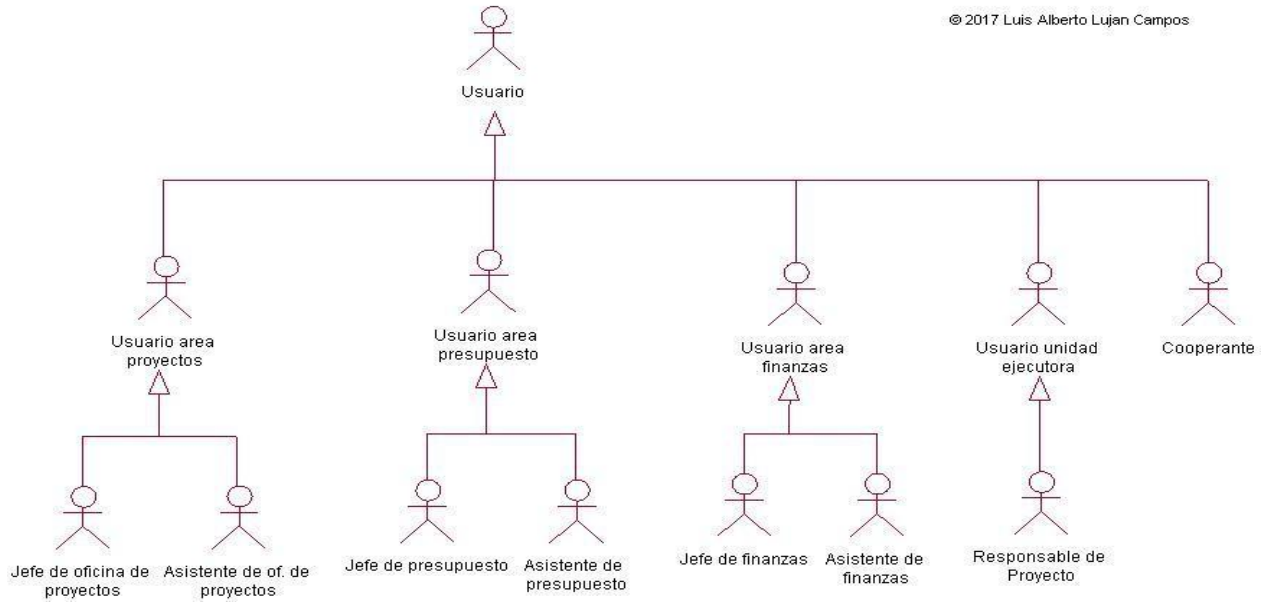

Figura 2: Diagrama de actores.

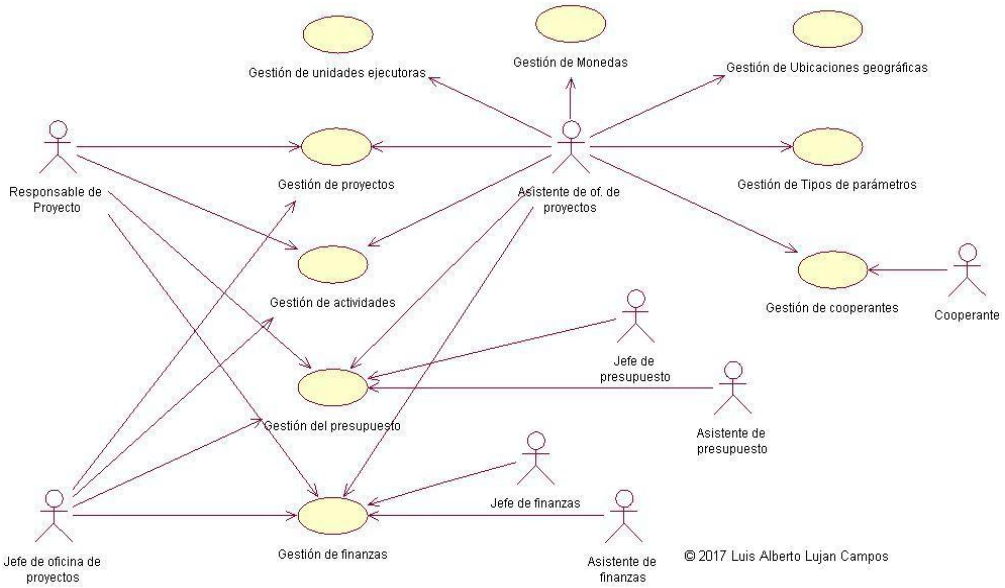

Figura 3: Diagrama de requerimientos.

Modelo de clases de objetos

En la figura 4 y 5 se muestran las clases de objetos y sus asociaciones en el sistema.

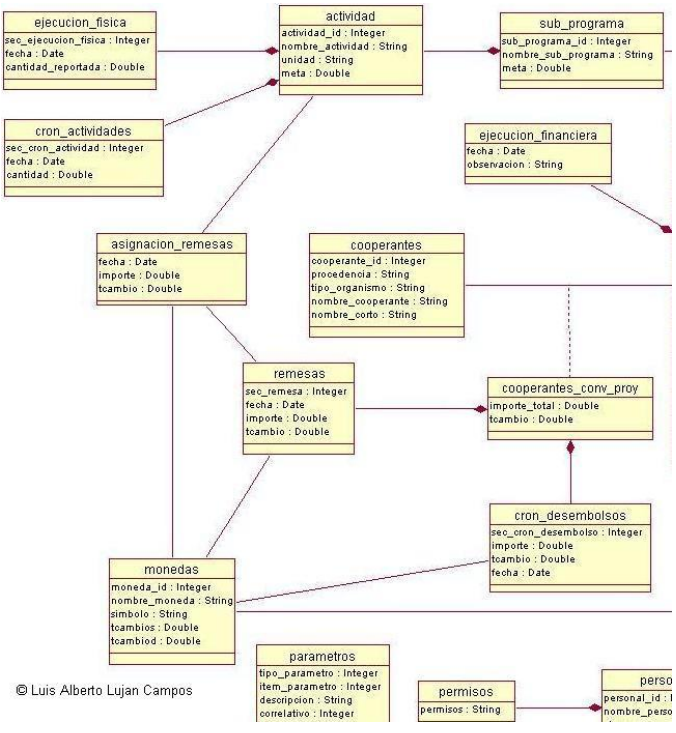

Figura 4: Diagrama de clases de objetos (parte 1).

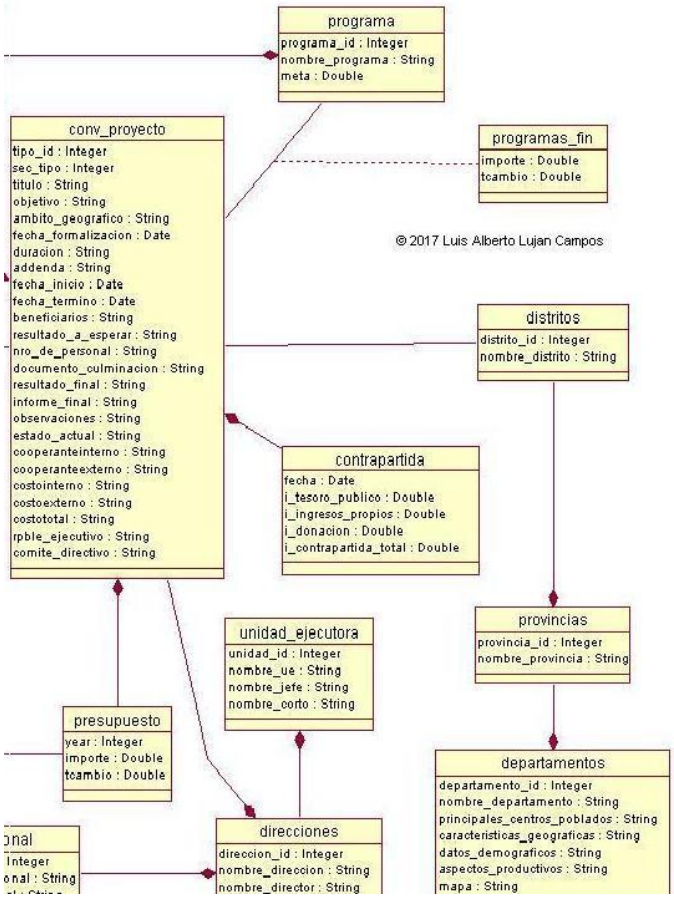

Figura 5: Diagrama de clases de objetos (parte 2).

Módulo de finanzas

En la figura 6 se muestra la funcionalidad de gestión de datos de finanzas.

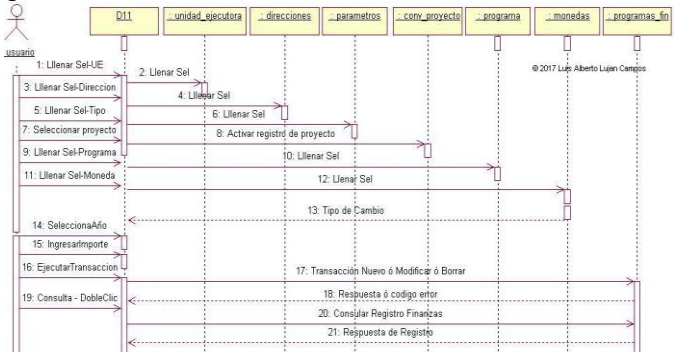

Figura 6: Diagrama de gestión de datos de finanzas.

Módulo de presupuesto

En la figura 7 se muestra la funcionalidad de gestión de datos de presupuesto.

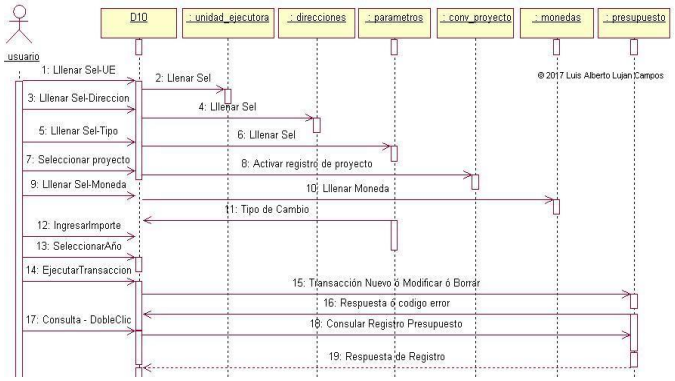

Figura 7: Diagrama de gestión de datos de presupuesto.

Módulo de proyectos o convenios

En la figura 8 y 9 se muestra la funcionalidad de gestión de datos de actividades.

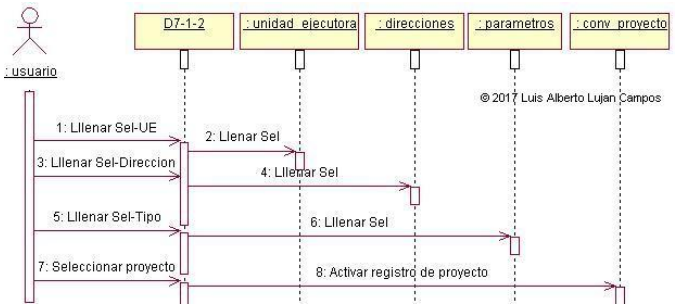

Figura 8: Diagrama de gestión de datos de actividades de proyectos o convenios (parte 1).

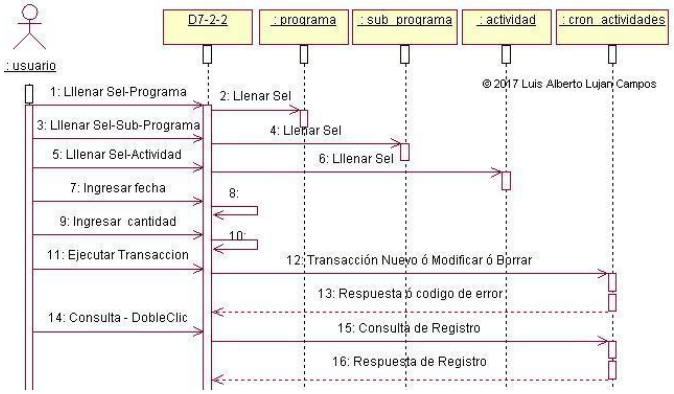

Figura 9: Diagrama de gestión de datos de actividades de proyectos o convenios (parte 2).

4. Interpretación de los resultados

Cumple con el objetivo de modelar los requerimientos y funcionalidad del software, en el trabajo de investigación para la gestión de información de proyectos y convenios globales se logró modelar los módulos de: módulo de cooperantes, módulo de datos del proyecto o convenio, módulo de presupuesto, módulo de finanzas y módulo de seguimiento de actividades, se elaboró más de quince diagramas. Los modelos son independientes de plataformas de software. Utilizando la herramienta Rational Rose pueden generarse diversos elementos como clases de objetos, tablas relacionadas, código de cabecera de clases entre otros para una plataforma de software especifica.

Model Driven Architecture (MDA) es una nueva propuesta de la OMG (Object Management Group) cuyo objetivo es modelar sistemas basándose únicamente en el modelado del dominio, independientemente de los aspectos tecnológicos. A partir de este modelado, MDA propone obtener por transformación elementos técnicos capaces de funcionar dentro de una plataforma de software como Java o .NET [13].

5. Conclusiones

Los modelos de requerimientos y los modelos de funcionalidad del software para la gestión de información de proyectos y convenios globales que están conformados por más de quince son independientes de las plataformas de software y de los aspectos tecnológicos.

Los modelos de los módulos: módulo de cooperantes, módulo de datos del proyecto o

convenio, módulo de presupuesto, módulo de finanzas y módulo de seguimiento de actividades están integrados con software Rational Rose.

El modelo de funcionalidad de software representa la interacción entre las clases de objetos mediante el flujo de información en la gestión de información de proyectos y convenios globales.

Los modelos de requerimientos y los modelos de funcionalidad del software son independientes y sirven para la ingeniería del desarrollo y mantenimiento de aplicaciones de software para la gestión de información de proyectos y convenios globales.

Referencias

[1] T. Ndie, C. Tangha, F. Ekwoge, MDA (model-driven architecture) as a software industrialization pattern: An approach for a pragmatic software factories. Journal of Software Engineering and Applications, 3(6), (2010), 561. Visitado el 10/06/2017: https://file.scirp.org/pdf/JSEA20100600007_83854981.pdf

[2] M. Fowler, S. Kendall, UML Gota a Gota. Spanish Edition (Addison Wesley Longman Mexico, 1999, ISBN: 968-444-364-1), pp. 1.

[3] T. Ndie, C. Tangha, F. Ekwoge, MDA (model-driven architecture) as a software industrialization pattern: An approach for a pragmatic software factories. Journal of Software Engineering and Applications, 3(6), (2010), 562

[4] M. Ramachandran, Guidelines based software engineering for developing software components. Journal of Software Engineering and Applications, 5(1), (2012), 1-6. doi: 10.4236/jsea.2012.51001.

[5] L. Lujan, Tesis de Maestría, Escuela de Post Grado, Universidad Nacional Federico Villarreal, 2013. pp. 122-123.

[6] OMG Object Management Group, Model Driven Arquitecture, (2008). En www.omg.org/mda, pp. 1, Visitado el 20/05/2017.

[7] C. Calero, A. Moraga, M. Piattini, Calidad del Producto y Proceso de Software, (Rama. D.J. España, 2010). Pp. 460.

[8] L. Martínez, Tesis de Maestría, Facultad de Informática, Universidad Nacional de La Plata, 2008, pp. 11. Visitado el 20/05/2017 http://sedici.unlp.edu.ar/handle/10915/4147

[9] R. Patil, Model-driven development methods for multicore-based software. Electronic Engineering Times, (1597), (2011), 40. http://www.eetimes.com/document.asp?doc_id=1278616&print=yes Visitado el 10/06/2017.

[10] V. Garousi, Experience in developing a robot control software. Computer and Information Science, 4(1), (2011), 3-13.

[11] Ch. Vasilakis, D. Lecnzarowicz, Ch. Lee, Application of Unified Modelling Language (UML) to the Modelling of Health Care Systems: An Introduction and Literature Survey, International Journal of Information Technology and Web Engineering, 3(4), (2008), 39-52.

[12] M. Fowler, S. Kendall, UML Gota a Gota. Spanish Edition (Addison Wesley Longman Mexico, 1999), pp. 49-124.

[13] Debrauwer, Laurent y Heyde, Fien van der, Unified Model Languaje, (Editorial ENI, 2009). pp. 26.