Mejora de la productividad en el diseño de arquitectura de software. Caso de estudio de sistema de costos para servicios de atención médica

Luis Alberto Lujan Campos

Universidad Nacional de Ingeniería, Av. Túpac Amaru 210, Lima, Perú

Recibido el 23 de abril del 2023, aceptado el 07 de julio del 2023

https://doi.org/10.33017/RevECIPeru2023.0006/

Resumen

El objetivo es mejorar la productividad al diseñar la arquitectura de software, del sistema de costos para servicios de atención médica. La recolección de datos se realizó en una institución de salud pública. Para el análisis y diseño de los modelos se usó el enfoque Arquitectura Dirigida por Modelos (Model Driven Architecture - MDA) y para el modelado se usó el software Rational Rose, por tener enfoque MDA e incluir la notación del Lenguaje Unificado de Modelado (Unified Modeling Language - UML). El enfoque MDA permite diseñar modelos de la funcionalidad del sistema independientes de la plataforma y presenta el beneficio de generar código para el sistema desde los modelos elaborados con UML. Las reglas de transformación MDA se expresan usando el software, que puede generar código, por ejemplo, mediante el modelo de clases del sistema, se generan clases de objetos respetando asociación y herencia [1]. Se determinó los módulos siguientes: depreciaciones, grupos de servicios, servicios, actividades, gastos indirectos, materiales, personal y estructura de costos de actividades y la estructura de clases de objetos. Se elaboraron 26 modelos y se mejoró la productividad con el uso de herramientas de software de ingeniería de software asistida por computadora.

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

Abstract

The objective is to improve productivity when designing the software architecture of the cost system for health care services. Data collection was performed in a public health institution. For the analysis and design of the models, the Model Driven Architecture (MDA) approach was used and the Rational Rose software was used for modeling, since it has an MDA approach and includes the Unified Modeling Language notation. Language - UML). The MDA approach allows platform-independent models of system functionality to be designed and has the benefit of generating code for the system from models built with the UML. MDA transformation rules are expressed using software, which can generate code, for example, using the system class model, object classes are generated respecting association and inheritance [1]. The following modules were determined: depreciation, groups of services, services, activities, indirect expenses, materials, personnel and cost structure of activities and the structure of classes of objects. 26 models were made and productivity was improved with the use of computer aided software engineering software tools.

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

1. Introducción

El diseño de modelos de arquitectura de software, presenta desafíos, debido a la diversidad de dispositivos, la heterogeneidad de los datos y a los grandes volúmenes de datos, se ha vuelto crítico para garantizar un funcionamiento eficiente y seguro del sistema. El propósito de ésta investigación es el diseño de los modelos para obtener la arquitectura de software del sistema de

costos para servicios de atención médica en una institución pública y mejorar la productividad, con el enfoque de Arquitectura Dirigida por Modelos (Model Driven Architecture - MDA).

Para obtener la arquitectura, se usa el desarrollo basado en modelos, que es muy útil para reducir los esfuerzos, tiempo y reducir errores, como se mostró en la arquitectura basada en modelos en el área de aplicaciones web, para reducir los problemas [2].

El refinamiento del modelo, es un elemento clave de nuestro enfoque. La información de diseño se agrega paso a paso. El modelo inicial puede ser tan abstracto como sea apropiado, debe servir para las necesidades de codificación automática y se acepta la incompletitud de los modelos para mejorarlos [3].

Según Object Management Group (OMG), el enfoque Arquitectura Dirigida por Modelos usa lenguajes de modelado para elevar el nivel de abstracción, mediante el modelado de requisitos, mediante un Modelo Independiente de Computación (Computation Independent Model - CIM), éste enfoque de arquitectura impulsada por modelos ofrece una interesante y dimensión original al dominio de ingeniería de software [4].

El método de desarrollo de la interfaz de usuario basado en MDA, puede resolver el problema de manera efectiva, para interfaces de usuario específicas, cuando la aplicación necesita ejecutarse en diferentes plataformas. Con éste método se diseña el modelo independiente de la plataforma, luego se transforma en una variedad de modelos específicos de forma automática o semiautomáticamente, y finalmente transforma en un código específico de la plataforma mediante meta modelos [5].

Se propone el enfoque MDA para el proceso de producir modelos que se pueden aplicar a cualquier implementación de tecnología y plataformas, a través de modelos que se transforman posibilitando la portabilidad de los sistemas. Se puede apreciar el detalle en el proceso de desarrollo y los nexos de comunicación para construir relaciones entre modelos [6].

Los modelos sirven como soporte del desarrollo, con el uso óptimo pueden contribuir a ahorrar costo, esfuerzo y tiempo con la calidad que todos buscamos, reutilización al máximo. La reutilización y los factores de seguridad se pueden mejorar de manera significativa, que se traduce en logros de la alta calidad de los sistemas de software y reducir los costos de desarrollo de software [7].

En el caso en estudio se logró mejorar la productividad y la elaboración del diseño del conjunto de modelos que forman parte de la arquitectura de software del sistema de costos para servicios de atención médica.

2. Metodología

Se realizaron entrevistas al personal de una institución de salud pública. Se analizaron documentos, flujo de datos y actividades enfocados en el objetivo de la investigación. Se estimó y elaboró lista de tiempos para cada entregable. La metodología usada considera análisis, modelado y diseño utilizando el enfoque MDA y el Lenguaje Unificado de Modelado (Unified Modeling Language - UML). Se utilizó software Rational Rose, según [8] mantiene integrado todos los modelos, permite el modelado de sistemas complejos, soporte para MDA, incluye la notación UML y diversas transformaciones de modelos.

Los modelos de la arquitectura de software, del sistema de costos para servicios de atención médica son:
Definición de actores, funcionalidades y modelado de los requerimientos.
Definición de clases, asociaciones y modelado de la estructura.
Modelado de la funcionalidad de la gestión de datos de: grupos de servicios, servicios, actividades, personal, materiales, depreciaciones, gastos indirectos y la ficha de estructura de costos de actividades.

2.1. Model Driven Development y Model Driven Architecture

Los enfoques de Desarrollo Dirigido por Modelos (Model Driven Development - MDD) ofrecen soluciones a diversas limitaciones, permitiendo crear aplicaciones a través de la especificación de modelos y generar código a partir de ellos. Específicamente, en el caso de que estos enfoques adopten un estándar MDA, se ve facilitada la tarea de dirigir el proceso de desarrollo y garantizar una mayor interoperabilidad y portabilidad entre sistemas. Se define lo necesario como meta modelos y perfiles con UML, así como reglas de transformación que te permiten generar código basado en HTML5, JavaScript, jQuery, jQuery Datatables y jQuery UI. La validación preliminar de la propuesta muestra evidencias positivas en cuanto

a la eficacia, eficiencia y satisfacción de los usuarios con respecto a los procesos de modelado y generación de código [9].

MDA define capas de abstracción y transformaciones entre modelos hasta la generación de código. Las capas de abstracción se utilizan para permitir una separación coherente de las partes. El modelado y la generación de código es compatible con el meta modelado y MDA se centra en la ingeniería de software. El MDA define cuatro capas de abstracción denominadas modelo independiente de computación (CIM), modelo independiente de la plataforma (PIM), modelo específico de la plataforma (PSM) e Implementación real [10]. Según [10] las principales capas se definen de la siguiente manera:

Computation Independent Model

El CIM permite que no se muestre detalles de la estructura de los sistemas. Un CIM se conoce con frecuencia como un modelo de dominio, y su especificación utiliza un vocabulario que es común a los usuarios del área de dominio.

Los CIM son cruciales para cerrar la brecha entre los expertos del dominio, los expertos del diseño y personal de construcción de los artefactos que juntos cumplen con los requisitos del dominio.

Los CIM ayudan a presentar exactamente lo que se espera del sistema. Los CIM son valiosos, no solo para comprender un problema, sino también como un instrumento para compartir vocabulario con otros modelos.

Platform Independent Model

El Modelo Independiente de la Plataforma o PIM describe y se centra en el funcionamiento del sistema, pero no muestra detalles de su uso en su plataforma, es independiente de cualquier tecnología de implementación.

La transformación de CIM a PIM permite utilizar los CIM no sólo como un medio de comunicación entre el experto del dominio y desarrolladores de software. Esta transformación es un paso clave para garantizar que la información comercial se transmita y sea respetada a lo largo del proceso de MDA.

Platform Specific Model

El modelo específico de la plataforma o PSM combina las especificaciones en el PIM con los detalles, que especifican cómo ese sistema utiliza un tipo particular de plataforma. La transformación permite vínculos de trazabilidad entre las especificaciones de requisitos, representadas en los CIM, y PIM y artefactos PSM que los implementan [11].

Estos vínculos contribuyen en asegurar la calidad en el proceso de desarrollo de software, facilitando que cualquier cambio en el nivel del CIM se refleje en el PIM y los niveles de PSM [12].

Según el autor [13], en la Figura 1, se muestran los elementos fundamentales de la arquitectura MDA.

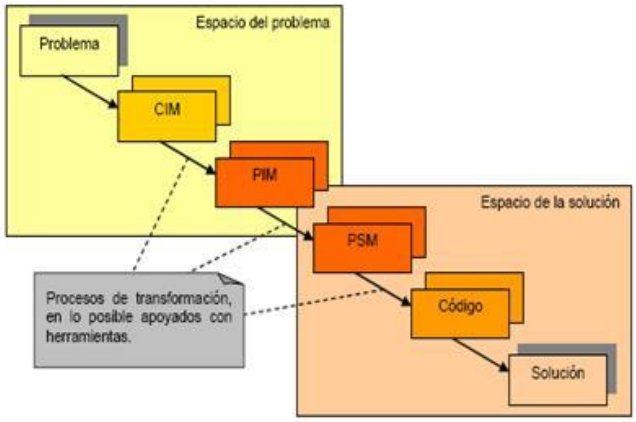

Figura 1: Arquitecta Dirigida por Modelos

2.2. Unified Modeling Language

Basado en los requisitos de la investigación, se utilizó UML como el medio principal para describir la arquitectura del sistema, especialmente el diagrama de casos de uso, el de clases y el de secuencia. UML es un lenguaje de modelado estandarizado que permite a los desarrolladores especificar, visualizar, construir y documentar artefactos de sistemas de software.

Por lo tanto, UML hace que estos artefactos sean escalables, seguros y robustos. UML es un aspecto importante involucrado en el desarrollo de software orientado a objetos. Utiliza notación gráfica para crear modelos visuales de sistemas de software [14].

Según [15], entre los principales diagramas se definen de la siguiente manera:

Use case

El diagrama de casos de uso, es aplicable en todas las fases del ciclo de vida del software desde los

requisitos, análisis, hasta la prueba del sistema. En la fase de análisis de requisitos, los requisitos funcionales del sistema pueden ser descritos a través del modelado de casos de uso.

En la prueba del sistema, utilizando el diagrama de casos de uso permite verificar el comportamiento del sistema, y en la prueba de aceptación, permite verificar que los resultados de la prueba del sistema cumplen con los requisitos definidos en el análisis. El modelado de requisitos describe la funcionalidad del sistema a través del diagrama de casos de uso.

Class diagram

El diagrama de clases, **e**n la fase de análisis, muestra las clases en el dominio del problema y sus relaciones se identifican a través del diagrama de clases.

Los detalles técnicos de la clase se introducen en la fase de diseño. Usando diagrama de clases y diagrama de paquete, el modelado estático describe la estructura estática del sistema.

Sequence diagram

El diagrama de secuencia, es parte del modelado dinámico, a través del diagrama de secuencia, se describe la dinámica del comportamiento del sistema, es decir, cómo los objetos en el sistema interactúan dinámicamente durante la ejecución [16].

2.3. Productividad

En relación al enfoque para el desarrollo de la arquitectura del software, se explica, con mucho énfasis, el uso de modelos, 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. La aplicación de 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. Al elaborar el diseño del software, debe ser modelado a un alto nivel de abstracción de forma independiente a la plataforma y a continuación puede ser transformado a modelos específicos de plataformas y finalmente a código [17]**.**

3. Resultados

Modelo de requerimientos

En la Figura 2, se presenta los tipos de actores.

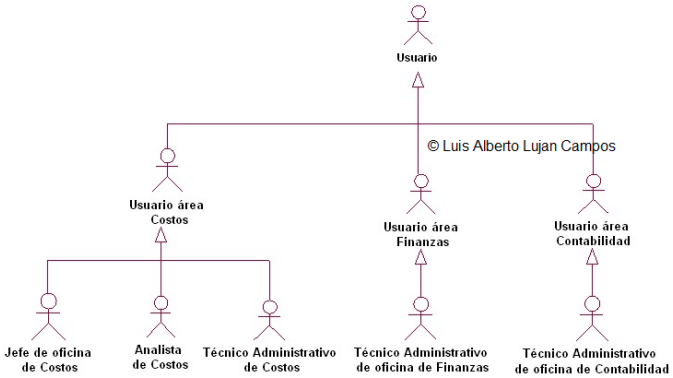

Figura 2: Actores.

Los requerimientos se representan en la Figura 3.

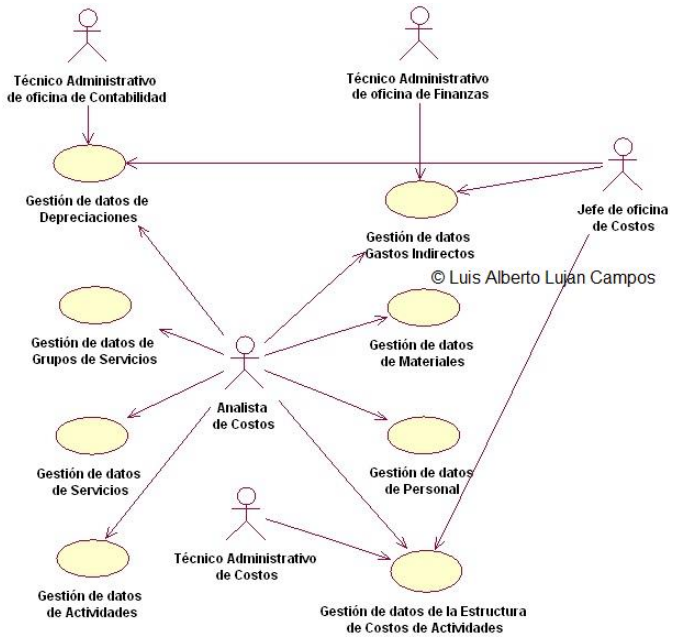

Figura 3: Requerimientos.

Modelos de estructura y funcionalidad

En la figura 4, se aprecian las clases de objetos con atributos y las asociaciones. Desde la Figura 5 hasta 8, se aprecian modelos de funcionalidad de los módulos de costos y de actividades.

Modelo de clases de objetos

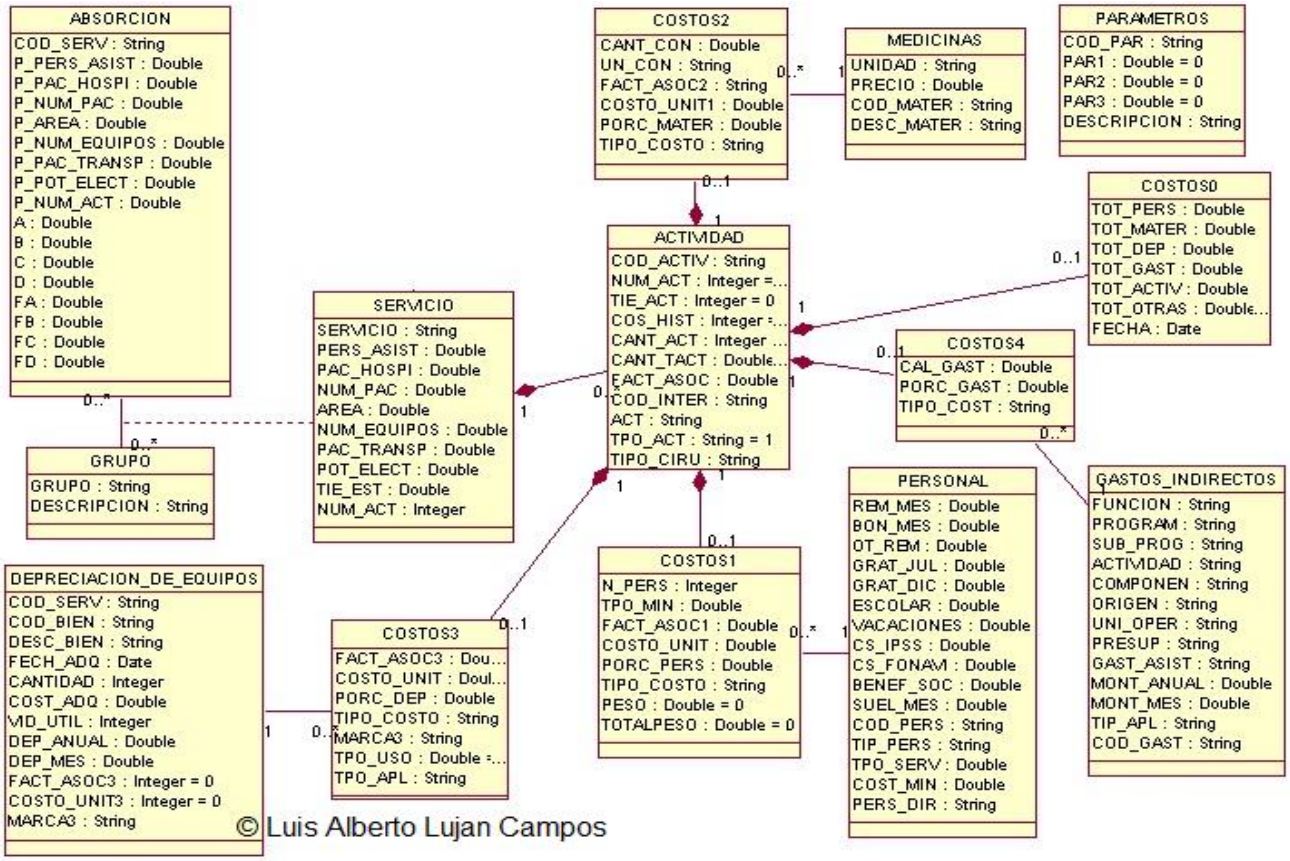

Figura 4: Clases de objetos

Módulo de costos: vista de materiales

En la figura 5, se muestra la funcionalidad de la gestión de datos de materiales en el módulo costos.

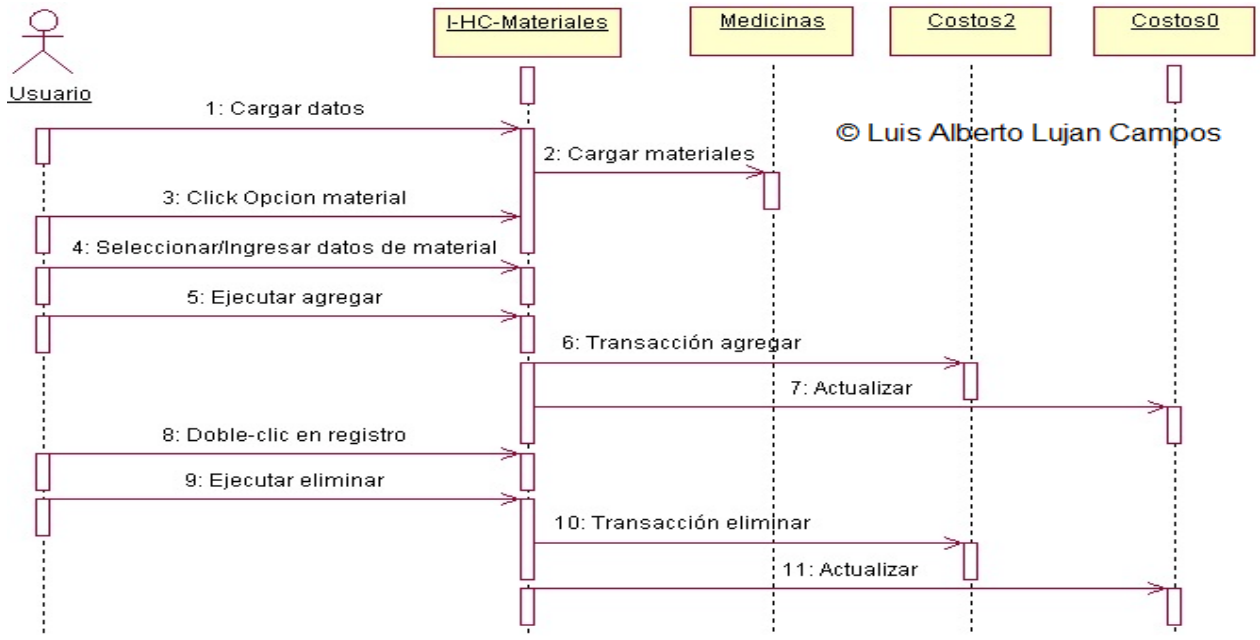

Figura 5: Gestión de datos de materiales.

En la Tabla 1 se muestra la lista de modelos de la arquitectura.

Tabla 1: Lista de entregables

Modelos	
Modelos de requisitos	
1	Diagrama de actores
2	Modelamiento de los requerimientos
Modelos de la interfaz de la gestión de datos de:	
3	Grupos de servicios
4	Servicios
5	Personal
6	Materiales
7	Actividades
8	Depreciaciones
9	Gastos Indirectos
10	Costos: vista personal directo
11	Costos: vista de materiales
12	Costos: vista de depreciaciones
13	Costos: vista de gastos Indirectos
14	Modelo de clases de objetos
Modelos de la funcionalidad de la interfaz:	
15	Grupos de servicios
16	Servicios
17	Personal
18	Materiales
19	Actividades
20	Depreciación
21	Gastos indirectos
22	Costos: vista personal directo
23	Costos: vista de materiales
24	Costos: vista de depreciaciones
25	Costos: vista de gastos indirectos
26	Modelo de componentes

Algunos modelos de la funcionalidad de la interfaz, se muestran desde la Figura 5 hasta la 8, son diagramas de secuencia, que corresponden respectivamente a uno de los modelos de interfaz de la gestión de datos.

Los modelos están integrados con el modelo de clases de objetos, con los modelos de requerimientos y con el modelo de componentes, los 26 modelos forman parte de la arquitectura del sistema de costos para servicios de atención médica.

En la Tabla 2 se muestra los tiempos para el cálculo de la productividad.

Tabla 2: Datos para la productividad

Modelo	Recurso tiempo		Productividad	
	Estimado (día de 24 horas)	Obtenido (día de 24 horas)	Estimada	Obtenida
1	3	2	0.333	0.500
2	17	16	0.059	0.063
3	2	1	0.500	1.000
4	2	1	0.500	1.000
5	2	1	0.500	1.000
6	3	2	0.333	0.500
7	5	4	0.200	0.250
8	4	3	0.250	0.333
9	5	4	0.200	0.250
10	5	4	0.200	0.250
11	5	4	0.200	0.250
12	5	4	0.200	0.250
13	7	6	0.143	0.167
14	19	17	0.053	0.059
15	6	4	0.167	0.250
16	3	2	0.333	0.500
17	3	2	0.333	0.500
18	3	2	0.333	0.500
19	3	2	0.333	0.500
20	3	2	0.333	0.500
21	3	2	0.333	0.500
22	4	3	0.250	0.333
23	4	3	0.250	0.333
24	4	3	0.250	0.333
25	4	3	0.250	0.333
26	4	3	0.250	0.333
	128	100	7.088	10.788

Con los valores de los totales de la segunda y tercera columna de la Tabla 2, se calcula la productividad, obteniendo la siguiente expresión:

[(128-100)*100] / 128

Efectuando la expression se obtiene el 21.88. El valor 21.88 representa la mejora porcentual en la productividad basada en la diferencia entre los valores estimados y obtenidos de la Tabla 2.

Módulo de costos: vista depreciación

En la figura 6, se muestra la funcionalidad de gestión de datos de depreciación.

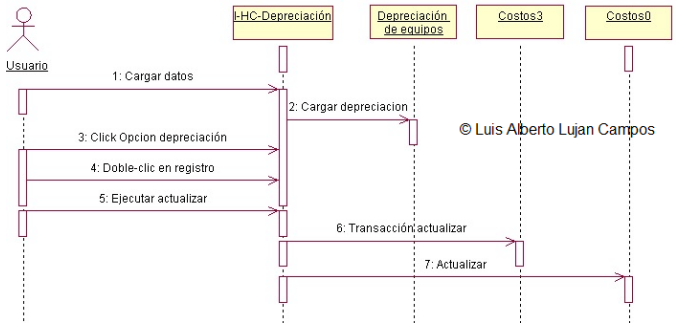

Figura 6: Gestión de datos de depreciación.

Módulo de costos: vista gastos indirectos

En la figura 7 se muestra la funcionalidad de gestión de datos de gastos indirectos.

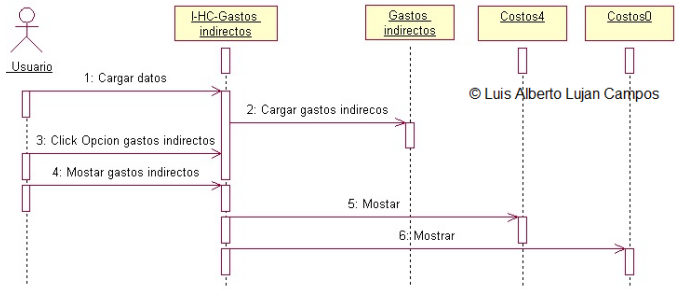

Figura 7: Gestión de datos de gastos indirectos.

Módulo de actividades

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

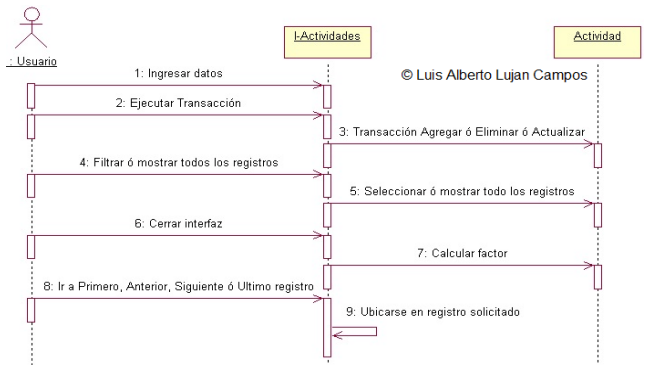

Figura 8: Gestión de datos de actividades.

4. Conclusiones

Se obtuvo el diseño de la arquitectura de software del sistema de costos para servicios de atención médica, que está formado por 26 modelos.

Se obtuvo el 21.88% de mejora de la productividad, se debe a la capacitación y al uso de herramientas de software de ingeniería de software asistida por computadora.

Los modelos de los módulos: depreciaciones, grupos de servicios, servicios, actividades, gastos indirectos, materiales, personal y estructura de costos de actividades están integrados con el software Rational Rose.

Los modelos de requerimientos y los modelos de funcionalidad del sistema son independientes y sirven para la ingeniería de desarrollo, mantenimiento y la gestión de información del sistema de costos para servicios de atención médica.

Referencias

[1] H. Benouda, M. Azizi, M. Moussaoui and R. Esbai, Automatic code generation within MDA approach for cross-platform mobiles apps. 2017 First International Conference on Embedded & Distributed Systems (EDiS), Oran, Algeria (2017) pp. 1-5, doi.org/10.1109/EDIS.2017.8284045

[2] F. Deeba, S. Kun, M. Shaikh, F. A. Dharejo, S. Hayat and P. Suwansrikham, Data transformation of UML diagram by using model driven architecture, 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China (2018) pp. 300-303, doi.org/10.1109/ICCCBDA.2018.8386531

[3] W. Ecker, K. Devarajegowda, M. Werner, Z. Han and L. Servadei, Embedded Systems Automation following OMG's Model Driven Architecture Vision, 2019 Design, Automation & Test in Europe Conference & Exhibition (date), Florence, Italy (2019) pp. 1301-1306, doi.org/10.23919/DATE.2019.8715154

[4] I. Essebaa and S. Chantit, Model Driven Architecture and Agile Methodologies: Reflexion and Discussion of Their Combination, 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), Poznan, Poland (2018) pp. 939-948, doi.org/10.3390/computers8040089

[5] G. Miao, L. Hongxing, X. Songyu and L. Juncai, Research on User Interface Transformation Method Based on MDA, 2017 16th International

Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), Anyang, China (2017) pp. 150-153 doi.org/10.1109/DCABES.2017.40

[6] H. Zhai, L. Cui, W. Zhang, L. Zhou and X. Zhang, Designing and Developing High-Confidence Petroleum Extraction Cyber-Physical System Using the Model-Driven Architecture, 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China (2019) pp. 975-978, doi.org/10.1109/ICPADS47876.2019.00148

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

[8] L. Lujan, Modelo de requerimientos y de funcionalidad de software basado en MDA y UML para la gestión de proyectos y convenios globales, Revista ECIPerú, Lima, Perú, (2017), 14(1), pp. 13-19, doi.org/10.33017/RevECIPeru2017.0002

[9] G. Nuñez, M. González, N. Aquino and L. Cernuzzi, A model-driven approach to develop rich web applications, 2017 XLIII Latin American Computer Conference (CLEI), Cordoba, Argentina (2017) pp. 1-10, doi.org/10.1109/CLEI.2017.8226424

[10] A. Rautakoura, M. Käyrä, T. D. Hämäläinen, W. Ecker, E. Pekkarinen and M. Teuho, Kamel: IP-XACT compatible intermediate meta-model for IP generation, 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovenia (2020) pp. 325-331, doi.org/10.1109/DSD51259.2020.00060

[11] N. Silega, M. Noguera, Y. I. Rogozov, V. S. Lapshin and T. González, Transformation From CIM to PIM: a Systematic Mapping, in IEEE Access (2022) vol. 10, pp. 90857-90872, doi.org/10.1109/access.2022.3201556

[12] N. Silega & M. Noguera, Applying an MDA-based approach for enhancing the validation of business process models. Procedia Computer Science 184 (2021) 761-766, doi.org/10.1016/j.procs.2021.03.094

[13] R. Moreno, CASE jMDA de Arquitectura Dirigida por Modelos para Sistemas de Información. Revista Cubana de Ciencias Informáticas (2020) 14(4), 210-223, redalyc.org/articulo.oa?id=378365835013

[14] X. Dong, J. Ding, J. Cui, H. Duan and F. Meng, The Architecture Design of Government Microblog Matrix Management System Based on Unified Modeling Language, 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Shenzhen, China (2020) pp. 75-79, doi.org/10.1109/aemcse50948.2020.00024

[15] L. Jin and J. Liang, Modeling of vehicle administrative management system based on unified modeling language, 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China (2017) pp. 50-54, doi.org/10.1109/ITOEC.2017.8122372

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

[17] L. Zepeda, E. Ceceña, J. Sosa, C. Angulo and R. González, A model driven method for data migration: Data migrattion with MDA, 2017 6th International Conference on Software Process Improvement (CIMPS), Zacatecas, México (2017), pp. 1-9, doi.org/10.1109/CIMPS.2017.8169958

Luis Alberto Lujan Campos:
llujanc@uni.edu.pe luislujan@neosistemas.org