

RUP y UML: Un estudio sobre ¿qué es?, ¿para qué se usa? y ¿cómo se desarrolla? Un Diagrama de Caso de Uso

Ivan Guadaña Quiroz^{1*}, Juan Daniel Pachamora Pinedo^{2†}.

¹Facultad de Ingeniería y Arquitectura, Universidad Peruana Unión

* Corresponde autor:

Universidad Peruana Unión, Facultad de Ingeniería y Arquitectura, E.A.P. Ingeniería de Sistemas

E-mail: ivan.guadania@gmail.com,

Celular: 976159687

† Corresponde autor:

Universidad Peruana Unión, Facultad de Ingeniería y Arquitectura, E.A.P. Ingeniería de Sistemas

E-mail: daniel.pachamora@gmail.com,

Celular: 981710189

Resumen

El presente artículo tiene como propósito, describir en lenguaje natural, la funcionalidad completa de un sistema a desarrollar y su relación con el entorno. Los diferentes tipos de información que se han desarrollado con propósitos diversos, han llevado consigo que el Lenguaje Unificado de Modelamiento (UML) como modelo para la construcción de software se haya extendido en los últimos años[1]. Asimismo el Rational Unified Process (RUP) interactúa a este desarrollo brindando una forma disciplinada al asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo). Es allí donde entran a tallar los diagramas de casos de uso, que es un esquema (modelo) para abstraer una vista del sistema del mundo real, considerando un cierto propósito, y describir el comportamiento de un actor y su interacción con el sistema. Estos diagramas de casos de uso son utilizados al analizar un proyecto, identificar y dividir la funcionalidad del negocio. Mediante este estudio reconocemos la importancia de los Diagramas de Casos de Uso y podemos asegurar que a través de su correcta elaboración, servirá como una herramienta para el Desarrollo de Software Orientado a Objetos (DSOO), gracias a la asertividad de las relaciones entre los actores y los casos de uso.

Palabras clave: RUP, UML, diagramas de casos de uso

1. Introducción

Variedad de atribuciones representan diferentes temas o asuntos del problema base en el proceso de desarrollo de software. Toda aplicación tendrá competencias base para funciones específicas. Logrando una separación de atribuciones, se disminuye la complejidad a la hora de trabajar con ellas, y es posible cumplir con requerimientos relacionados con la calidad como adaptabilidad, mantenimiento, extensibilidad y reusabilidad[2].

El Desarrollo de Software Orientado a Objetos, establece que no es posible lograr una completa y absoluta modularización en el desarrollo de aplicaciones y/o sistemas, dando nacimiento al paradigma del Desarrollo de Software Orientado a Aspectos (DSOA) y la Programación Orientada a Objetos (POA).

Gran parte de las estrategias de evaluación utilizadas para la automatización al momento de mejorar y producir software de alta calidad es ofrecida por el Unified Modeling Language, manejando la complejidad de los sistemas, así ellos aumenten en ámbito o en escala. Proporcionando mecanismos de modelamiento visual (diagramas) de tal forma que permita desarrollar e intercambiar modelos con significado, de los cuales destacamos los Diagramas de Casos de Uso[3].

Además se presenta un esquema de trabajo donde se mencionan las actividades que se deben realizar, la utilización correcta de actores, casos de uso y los errores que no se deben cometer en cada una de las actividades, para elaborar fielmente un Diagrama de Caso de Uso.

2. Lenguaje Unificado de Modelado (UML)

UML es un lenguaje de propósito general para el modelado orientado a objetos, que combina notaciones provenientes desde: Modelado Orientado a Objetos, Modelado de Datos, Modelado de Componentes, Modelado de Flujos de Trabajo (Workflows)[4].

Descripción de Diagramas

Un modelo (diagrama) captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.

Un diagrama es una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos como se observa en la figura 1. Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto desde cada una de las perspectivas de interés. Es aquí donde se hace evidente la importancia de UML en el contexto de un proceso de desarrollo de software.

El código fuente del sistema es el modelo más detallado del sistema (y además es ejecutable). Sin embargo, se requieren otros modelos.

Varios modelos aportan diferentes vistas de un sistema los cuales nos ayudan a comprenderlo desde varios frentes. Así, UML recomienda la utilización de nueve diagramas que, para representar las distintas vistas de un sistema.

Estos diagramas de UML se describen a continuación:

Diagrama de Casos de Uso: Diagrama de Clases, Diagrama de Objetos

Diagramas de Comportamiento: Diagrama de estados, Diagrama de actividades, Diagramas de iteración, Diagrama de secuencia, Diagrama de colaboración.

Diagrama de implementación: Diagrama de componentes, Diagrama de despliegue[3].

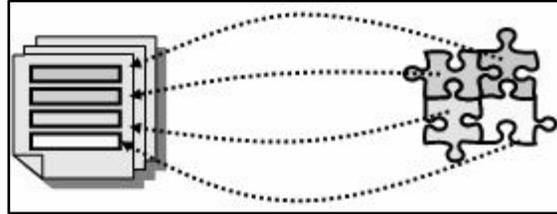


Fig. 1. Relaciones de enlaces entre modelos

2.1. ¿Qué es un Diagrama de Caso de Uso?

Los Diagramas de Caso de Uso, son diagramas que describen el comportamiento del sistema cuando algo o alguien (actor) interactúan con el sistema en relación con el negocio. Este comportamiento se puede explicar de forma gráfica y/o textual, describiendo la naturaleza del estímulo que proyecta el caso de uso. Esto conlleva a puntualizar, que ningún sistema se encuentra aislado. Ver figura 2.

Los Diagramas de Caso de Uso han proveído un medio para que los desarrolladores, los usuarios finales del sistema y los expertos del dominio lleguemos a una comprensión común del sistema. Perteneciendo a la segunda fase del Rational Unified Process (RUP), **elaboración**[3].

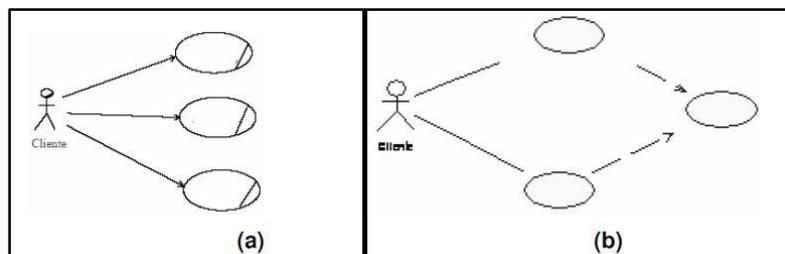


Fig. 2. Comparación entre diagramas de casos de uso (a) RUP (b) UML

2.2. ¿Pará que se usan los Diagramas de Caso de Uso?

Los Diagramas de Casos de Uso se utilizan durante la fase de análisis de un proyecto para identificar y dividir la funcionalidad del sistema. Normalmente contienen: casos de uso, actores y relaciones entre ellos: de asociación, de dependencia y/o de generalización.

Asimismo son utilizados de manera fundamental para la identificación de requerimientos en el DSOO. Como DSOA es una extensión de DSOO, es posible la aplicación de Diagramas de Casos de Uso en DSOA[5].

3. ¿Cómo desarrollar un Diagrama de Casos de Uso?

3.1. Símbolos de un Diagrama de Casos de Uso

Un diagrama de caso de uso (Business Use Case Diagram - BUCD) contiene el actor y símbolos de caso de uso, junto con líneas de conexión. A continuación detallamos cada símbolo del diagrama respectivamente.

3.1.1. Caso de Uso (Business Use Case – BUC)

El caso de uso describe un conjunto de secuencias de interacciones entre actores y el sistema, también se puede decir que los casos de uso describen el comportamiento del sistema cuando uno de los actores envía un estímulo concreto. Por ejemplo en una biblioteca se podrían encontrar varios casos de uso: Reservar Libro, Préstamo Libro, Devolver Libro, Extender Préstamo, etc. Ver figura 3.

3.1.2. Actor (*business actor - BA*)

El término *actor* se refiere a un papel singular de un usuario del sistema, roles jugados por personas, dispositivos u otros sistemas y non forman parte del sistema. Dicho de otra manera: *Es todo aquello que está fuera del sistema pero que interactúa con el sistema*. Por ejemplo un actor podría ser un empleado, pero también podría ser un cliente. Ver figura 3.

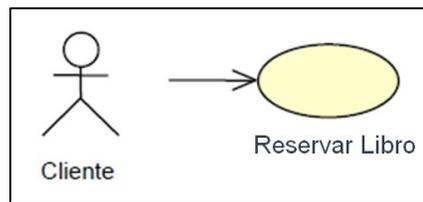


Fig. 3. Actor – Caso de uso

3.1.3. Relaciones

Hay cuatro tipos básicos de relaciones de comportamiento: *comunica*, *incluye*, *extiende* y *generaliza*. Todos estos términos son verbos de acción. En la tabla 1 se muestra las flechas y líneas usadas para diagramar cada uno de los cuatro tipos de relaciones de comportamiento. A continuación se presentan ejemplos de las relaciones[6].

Tabla 1.

Comunica		Un actor se conecta a un caso de uso usando una línea sin puntas de flecha.
Incluye		Un caso de uso contiene un comportamiento que es más común que otro caso de uso. La flecha apunta al caso de uso común.
Extiende		Un caso de uso diferente maneja las excepciones de caso de uso básico. La flecha apunta desde el caso de uso extendido hacia el básico.
Generaliza		Una “cosa” de UML es más general que otra “cosa”. La flecha apunta a la “cosa” general.

3.1.3.1. Comunica

Ejemplo: El actor **Estudiante** se comunica con el caso de uso **Matricularse en curso**.

Ver figura 4

3.1.3.2. Incluye

Ejemplo: El caso de uso **Pago de cuotas del estudiante** se incluye en **Matricula en el curso** y **Arreglar residencia estudiantil**, debido a que en ambos casos los estudiantes deben pagar sus cuotas. Ver figura 4.

3.1.3.3. Extiende

Ejemplo: El caso de uso **Seguro médico del estudiante** extiende el caso de uso básico **Pago de cuotas del estudiante**. La flecha va del extendido al básico. Ver figura. 4.

3.1.3.4. Generaliza

Ejemplo: Los actores **Estudiante regular** y **Estudiante irregular** generalizan a un actor **Estudiante**. Ver figura 4.

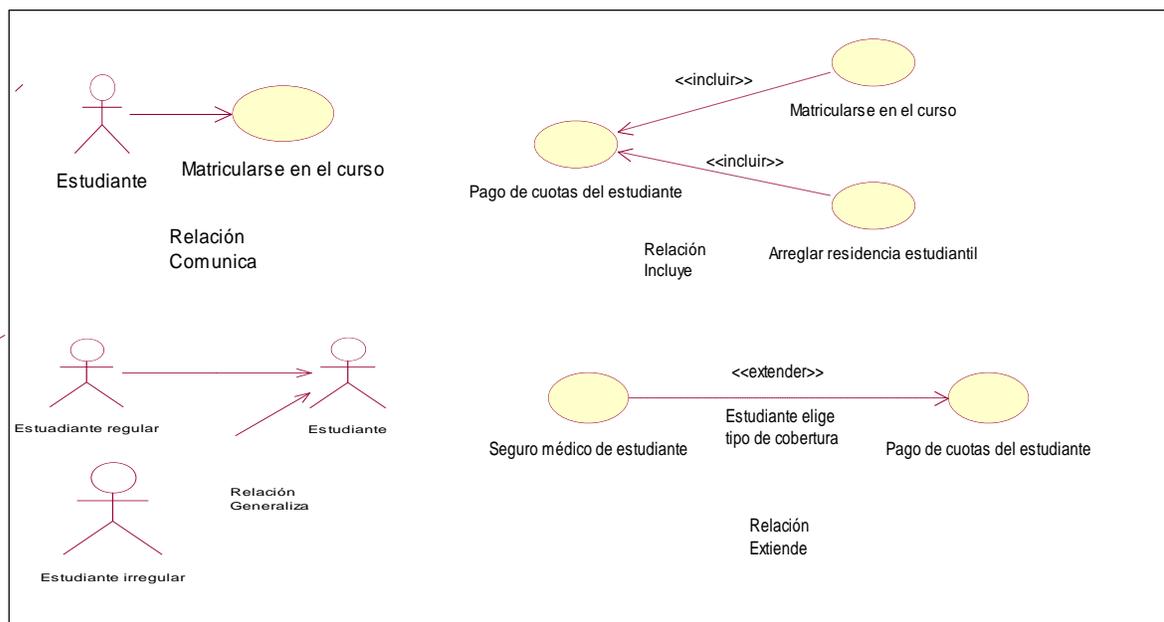


Fig. 4. Ejemplo de Relaciones



3.2. Recomendaciones para desarrollar un Diagrama de Casos de Uso

Hemos recopilado consejos de varios autores y a continuación mostramos seis pasos para un buen desarrollo de un Diagrama de Casos de Uso[5].

3.2.1. Paso 1: Identificar Requisitos

En esta actividad, deberemos responder a los siguientes cuestionamientos: ¿Qué le permite hacer, el sistema de software o negocio, al usuario? y ¿El cliente o usuario me solicita alguna restricción para construir el sistema de software? Contestando a esas preguntas se deberá realizar una lista que contendrá los requisitos del sistema, esta lista representará los servicios o funciones ofrecidos por el sistema.

3.2.2. Paso 2: Identificar Actores

Luego de identificar las funciones y servicios del sistema se procede a identificar actores del sistema. Se puede buscar en las categorías de personas, otro sistema o software, dispositivos de hardware o redes de computadoras.

3.2.3. Paso 3: Identificar Escenarios

Un escenario muestra la secuencia de pasos que se produce cuando un actor interactúa con el sistema en una situación específica y un tiempo determinado. Su propósito es servir en la identificación de casos de uso.

3.2.4. Paso 4: Identificar Casos de Uso

El *caso de uso* es el que especifica todos los escenarios posibles para una parte de funcionalidad dada, es decir, todos los escenarios todos los escenarios similares se agrupan en un solo *caso de uso*.

3.2.5. Paso 5: Especificar Casos de Uso

Luego de haber identificado los casos de uso, se tiene que indicar la forma en que el actor interactúa con el sistema.

3.2.6. Paso 6: Identificar Relaciones entre Casos de Uso y entre Actores

En esta actividad se identifican, en base a las especificaciones de casos de uso y de actores, las relaciones “incluye”, “extiende” y “generaliza” entre casos de uso y actores respectivamente, Es importante resaltar que las relaciones para casos de uso es opcional.

3.3. Errores Comunes

3.3.1. Errores en la identificación de actores.

Estos se deben principalmente a no comprender quiénes son los actores del sistema. En algunos casos se incluye actores que realmente no lo son. Ver figura 5.

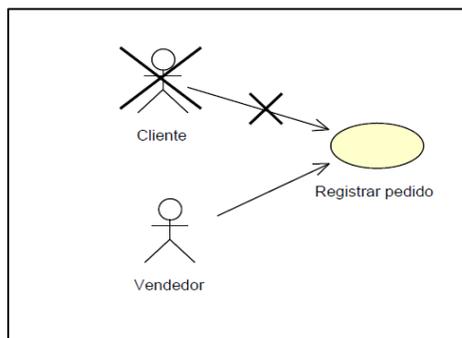


Fig. 5. Error en identificación de actores

3.3.1. Errores en la identificación de casos de uso

Un error muy extendido, es considerar las opciones de del menú o funciones del sistema como *casos de uso*. Ver figura 6.

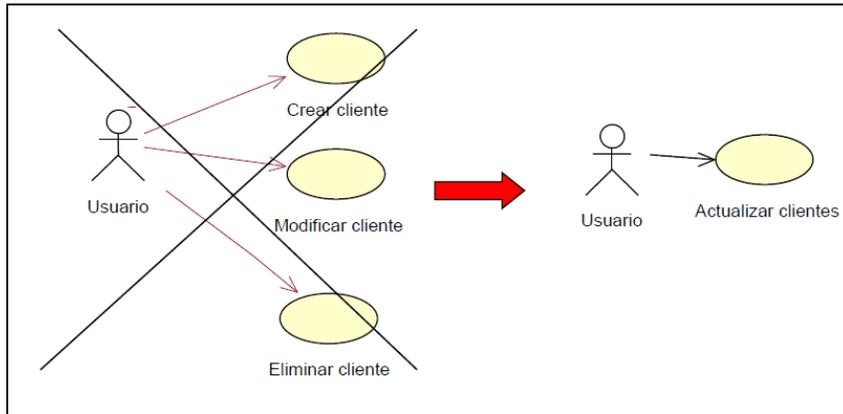


Fig. 6. Error de identificación de casos de uso

4. Diagrama de Casos de Uso aplicado al Plan de Salvación

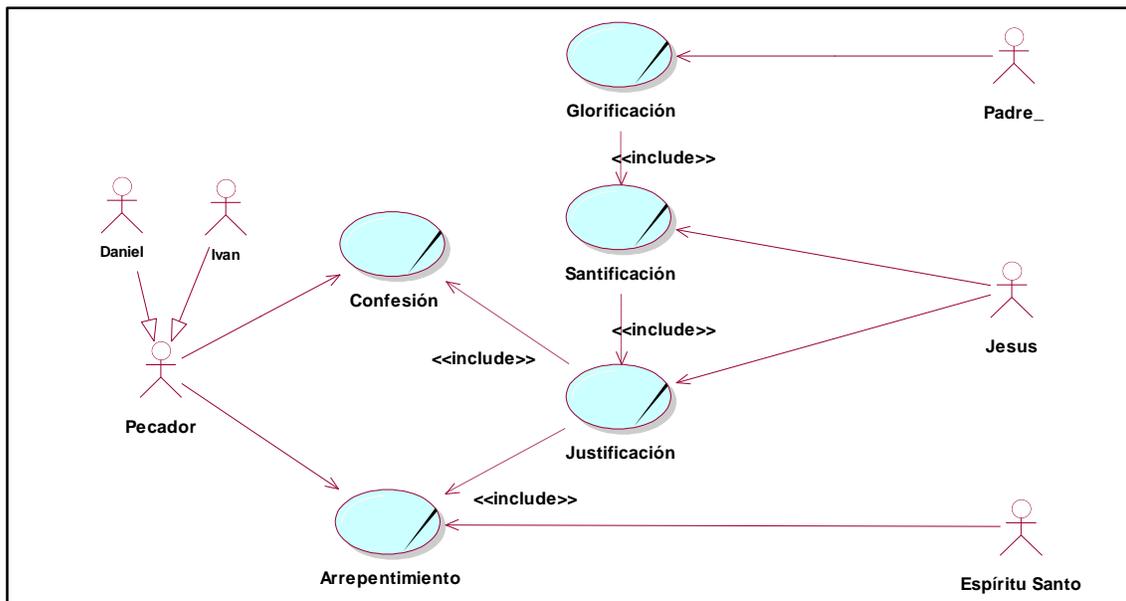


Fig. 7. Diagrama de Casos de Usos del Plan de Salvación

5. Conclusiones

Gracias a este artículo hemos podido mostrar desde los conceptos básicos de diagramas de casos de uso, sus componentes hasta los pasos para elaborar un diagrama, permitiendo minimizar los errores en las etapas de análisis y diseño.

Se puede reducir el tiempo de desarrollo de un Sistema de Software, aplicando la metodología RUP y UML ya que permite lograr de una manera fiable y rápida el desarrollo del Sistema deseado, donde están incluidos los diagramas de casos de uso.

A través del Diagrama de Casos de Uso, hemos podido comprender mejor el maravilloso Plan de Salvación, teniendo como principales actores: El Padre, Jesús, El Espíritu Santo y el pecador, quienes activan desde la confesión y el arrepentimiento hasta la santificación y glorificación; considerados como casos de uso.

El objetivo final es poder identificar los actores a partir de los requerimientos; y los casos de usos a partir de los escenarios de manera adecuada, sabiendo que el diagrama de casos de uso es de gran utilidad para el desarrollo de Software Orientado a Objetos.

Referencias

- [1] V. M. J. Coate Rosales, Edmundo, Saavedra Medina Nidia Carolina Enrique, “UML_y_RUP,” 2010. [Online]. Available: <https://www.google.com.pe/#q=para+que+se+utilizan+los+diagramas+de+casos+de+uso+del+rup>.
- [2] D. Levano Rodriguez, “Desarrollo de software orientado a objetos,” 2013, p. 97.
- [3] Q. C. Vilma, H. Solorzano, D. Harry, V. Yupanqui, and J. Luis, “MONOGRAFIA METODOLOGIA RUP (RATIONAL UNIFIED PROCESS),” 2011. [Online]. Available: <http://msdn.microsoft.com/es-es/library/dd409427.aspx>. [Accessed: 27-Nov-2013].
- [4] R. U. Process, “Rational Unified Process : A Best Practices Approach Topics What is RUP ? RUP best practices Software economics Adapt the process,” 2003. [Online]. Available: <http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html>.
- [5] A. Pow, S. Portillo, and S. Miguel, “La Especificación de Requisitos con Casos de Uso : Buenas y Malas Prácticas Introducción,” 2010. [Online]. Available: <http://web.ebscohost.com/ehost/detail?vid=5&sid=3cd21fb4-fd63-4810-a0f2-a05c4bd80d15%40sessionmgr4001&hid=126&bdata=Jmxhbm9ZXMmc210ZT1laG9zdC1saXZl>.
- [6] K. E. Kendall, J. E. Kendall, A. N. Ramos, and H. Cárdenas, *ANÁLISIS Y DISEÑO*, Sexta edic. México, 2005, p. 752.