La importancia del Análisis de Requerimientos

Yois Smith Pascuas Rengifo [1]

Resumen

Cuando se inicia el proceso de desarrollo de software, se debe comenzar con la recolección de requerimientos de usuario. Para lograr un mayor acercamiento y entendimiento a éstos requerimientos, se deben analizar y describir diferentes enfoques, logrando así un diagnóstico de la situación actual del negocio.

Palabras clave: Métodos, metodologías, artefactos, procesos, objetos.

The importance of the Analysis of Requirements

Abstract

When the process of development of software begins, is due to begin with the harvesting of user requirements. In order to obtain a greater approach and understanding to these requirements, they are due to analyze and to describe to different approaches, obtaining therefore a diagnosis of the present situation of the business.

Key words: Methods, methodologies, devices, processes, objects.

Introducción

La importancia que hoy en día se le da al software radica en que prácticamente todas las organizaciones dependen de éste para realizar sus funciones diarias, también se considera la Tecnología Informática como estrategia para obtener ventaja competitiva. Por éstas razones y muchas más, el desarrollo de proyectos software se ha convertido en una de las áreas con mayor campo de acción dentro de las disciplinas tecnológicas.

Pero el desarrollo de software no es sencillo, ya que por medio de éste se modelan las principales funcionalidades ofrecidas por el negocio, se abstrae el funcionamiento de la organización y por lo mismo, se vuelve más complejo en tanto más compleja sea la organización.

Para trabajar en el desarrollo de un software, existen metodologías que se dividen en varias etapas que proporcionan procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a crear software de calidad. La metodología indica cómo hay que obtener los distintos productos parciales y finales. Entre las metodologías más importantes se encuentran la Metodología de Rational Unified Process (RUP), Extreme Programing (XP), Microsoft Solution Framework (MSF), el modelo en espiral, en cascada, espiral, prototipos y Método en V [1].

A pesar de la aplicación de las metodologías y estrategias de la ingeniería en el desarrollo de software, se observa, con el pasar de los años que el desarrollo de los proyectos de software en la mayoría de los casos no culmina exitosamente. El 75 % de los productos de software grandes se entregaron a los clientes pero tienen fallas, son un fracaso porque no se usan o no cumplen los requerimientos del cliente. Lo que provoca grandes tasas de fracasos por la indefinición precisa de las necesidades o requerimientos. Esto se puede notar a partir del origen de los errores en el software; se estima que el 56% de los errores en los proyectos se da por el mal desarrollo en la etapa del Estudio y Análisis, 10% en el diseño, 7% en el código y el 27% en errores varios [2]. Las anteriores cifras muestran la realidad de los fracasos de proyectos software, por eso es indispensable que se reconozca la importancia de cada una de las etapas de las metodologías de desarrollo de software.

Por todo esto existe la Ingeniería de Software (IS). La IS es la rama de la ingeniería que crea y mantiene las aplicaciones de software aplicando tecnologías y prácticas de las ciencias computacionales, manejo de proyectos, ingeniería, el ámbito de la aplicación, y otros campos [3]. La IS se encarga del desarrollo de un producto de software a bajo costo y alto rendimiento. [4]

Resulta indispensable tomar en cuenta de que la parte fundamental para el desarrollo de software es la identificación de requerimientos, la cual se constituye como una disciplina de la Ingeniería de Software llamada Ingeniería de Requerimientos (IR). La IR, es una etapa en la que el usuario define lo que quiera que haga el sistema; identifica las funciones principales del sistema, el alcance del modelamiento del mundo y documenta los

[1] Estudiante de Maestría en Ciencias de la Información y las Comunicaciones de la Universidad Distrital Francisco José de Caldas.

El 75 % de los productos de software grandes se entregaron a los clientes pero tienen fallas, son un fracaso porque no se usan o no cumplen los requerimientos del cliente

procesos principales y las políticas que el sistema va a soportar. No se definen pasos formales, ya que éstos dependen de qué tan nuevo es el proyecto, la disponibilidad de expertos y usuarios y la disponibilidad de documentos adicionales [5].

Es crucial conocer la importancia de la IR. El siguiente documento muestra la Ingeniería de Requerimientos dentro del desarrollo de la metodología aplicada en la construcción de software. En la primera sección se muestran los conceptos básicos, generales, las actividades de la IR y la clasificación de los requerimientos. En la segunda sección las metodologías para el desarrollo de software utilizadas y específicamente la metodología orientada a objetos, las etapas del proceso unificado, como se hace análisis de Requerimientos en Rup y por último se da un ejemplo del análisis de requerimientos.

1. ANÁLISIS DE REQUERIMIENTOS

Para comenzar con éste tema se debe tener muy claro la importancia, actividades de la IR, el concepto de requerimiento y los tipos de requerimientos. La IR se define, como un "conjunto de actividades en las cuales, utilizando técnicas y herramientas, se analiza un problema y se concluye con la especificación de una solución (a veces más de una)." [Ortas 1997]. Entonces, la IR se utiliza para definir todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requerimientos para un producto determinado. El uso del término "ingeniería" implica que se deben utilizar técnicas sistemáticas y repetibles para asegurar que los requerimientos del sistema estén completos, sean consistentes y relevantes [6].

Su importancia estriba en que, de la definición de los requerimientos dependerá la definición de las etapas subsecuentes del desarrollo de software, es decir, que si no se descubren los requerimientos que se encuentran en el ambiente del sistema ó son encontrados en una etapa avanzada del desarrollo del sistema, se tendrá que retroceder nuevamente a la etapa de requerimientos y esto provocaría cambios en el sistema y consecuentemente retraso en la entrega del sistema. Un caso peor, es que no se encontraran y especificaran todos los requerimientos del sistema en un proceso de desarrollo de software, lo cual produciría la entrega de un producto de software incompleto o poco funcional. [7]

Usualmente se puede dividir las prácticas de la IR en cuatro actividades, a saber:

1. Extracción
2. Análisis
3. Especificación
4. Validación

Como toda división de tareas, no es una estricta representación de la realidad, sino que se hace con el fin de sistematizar la realización de la IR. En general la delimitación entre una actividad y la otra no es tan clara, ya que están sumamente interrelacionadas, existiendo un alto grado de iteración y retroalimentación entre una y otra. En la figura 1 se aprecian las tareas de la IR y la iteración entre sus tareas también se resalta la tarea más importante de la IR que es el análisis de requerimientos:

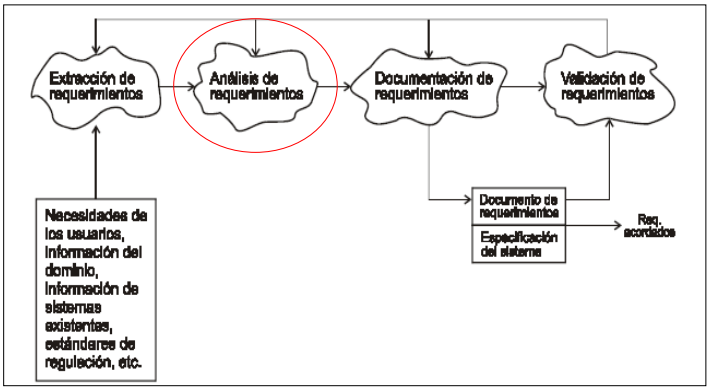


Figura 1. Tareas de la IR

Para profundizar un poco en nuestro tema, según la Real Academia Española, requerir (**requerimiento**) es reconocer o examinar el estado en que se halla algo. Pero el concepto que da [IEEE, 1990] de requerimiento nos lleva a varios puntos de vista, primero define un requerimiento como una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo, segundo una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal y tercero una representación en forma de documento de una condición o capacidad como las expresadas en la primera o la segunda.

Pero existen dos **clases de requerimientos**, los **requerimientos funcionales** (RF) y los **requerimientos no funcionales** (RNF). Los **RF** del sistema son aquellos servicios que el usuario espera

del sistema. En general, al usuario no le interesa cómo se aplican esos servicios, así que el ingeniero de software debe evitar la inclusión de conceptos de aplicación en esta sección del documento de los requisitos. En principio, los requisitos funcionales de un sistema deben ser completos y consistentes. Por completo se entiende que todos los servicios requeridos por el usuario deben especificarse, y la consistencia significa que ninguna definición de requisitos debe contradecir a otra. En la práctica, y para sistemas grandes y complejos, es casi imposible lograr que los requisitos sean consistentes y completos en la versión inicial del documento. A medida que se descubren los problemas durante las revisiones o en las etapas posteriores del ciclo de vida, el documento debe modificarse en consecuencia, hay tres maneras de expresar los requisitos funcionales de un sistema:

1. en Lenguaje Natural
2. en un lenguaje estructurado o en un formato que tenga algunas reglas, pero no una especificación sintáctica o semántica rigurosa,
3. en un lenguaje formal de especificación con una sintaxis y semántica definidas

De estas notaciones, la más utilizada es el ***lenguaje natural***, por la simple razón de que es el más expresivo y porque puede ser comprendido tanto por los usuarios como por los desarrolladores del sistema. Se han utilizado algo los lenguajes estructurados. Estos lenguajes tienen la ventaja de que, debido a las reglas que gobiernan su uso, pueden construirse dispositivos de software para verificar si son consistentes y completos.

Los RNF, son los requisitos no funcionales de un sistema, son restricciones u obligaciones impuestas al servicio de éste. Ejemplos de requisitos no funcionales son las obligaciones impuestas a los tiempos de respuesta del sistema, las limitaciones en la cantidad de memoria que ocupará el software y las restricciones en la representación de los datos del sistema.

Aunque tanto los requisitos funcionales como los no funcionales están sujetos a cambios, los requisitos no funcionales se ven especialmente afectados por los cambios en la tecnología de hardware. Puesto que el tiempo de desarrollo de un gran sistema puede ser de varios años, es probable que el hardware disponible al concluir el proyecto sea más potente que el potente que el disponible cuando se concibió el proyecto. Además, el hardware evolucionará a través del tiempo de vida del software desarrollado y los requisitos no funcionales se modificarán mientras el software esté en uso.

Existe la posibilidad de anticipar los cambios que se pueden producir en el hardware mientras se desarrolla el software. Los requisitos no funcionales dependientes del hardware pueden especificarse de modo que presuponga las capacidades de hardware que existirán a la terminación del proyecto, aunque quizá no sea ese el caso al comienzo del proyecto. No se puede hacer tal anticipación para los cambios producidos durante el tiempo de vida del proyecto, y una característica importante de los requisitos dependientes del hardware es que deben especificarse de manera que sea posible modificarlos con facilidad.

Los requisitos no funcionales son tales que tienen a estar en conflicto y actuar recíprocamente con otros requisitos funcionales del sistema. El conflicto entre los requisitos de velocidad de ejecución y los de memoria es aquí el ejemplo obvio. Una definición de requisitos no funcionales debe presentar esos requisitos de manera que los conflictos sean claros y se puede discernir entre los posibles intercambios [8].

2. METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE

Existen dos ramas importantes dentro de la IS, que son el análisis y el diseño estructurado o *metodología de desarrollo de software estructurada (ME)* que se ha utilizado mayoritariamente hasta hace pocos años y que todavía se utiliza en muchos de los proyectos software desarrollados hoy en día. También se encuentra la *metodología desarrollo de software orientado a objetos* (OO) que ya es utilizada en los proyectos de desarrollo de software actuales.

En la ME, un problema se asimila a una función o procedimiento, que se descomponen en otros más pequeños, que a su vez se descomponen en otras funciones más pequeñas, hasta llegar a problemas indescomponibles. Gracias a estos métodos, los sistemas de software fueron poco a poco creciendo en tamaño y complejidad, pero generaron nuevos problemas de más difícil solución, como por ejemplo el mantenimiento del software, más costos en las depuraciones, escasas posibilidades de reutilización de código, etc. [9]

La OO, se puede describir como el conjunto de disciplinas (ingeniería) que desarrollan y modelizan software que facilitan la construcción de sistemas complejos a partir de componentes. El atractivo intuitivo de la orientación a objetos es que proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan

fielmente como sea posible. Las ventajas de la orientación a objetos son muchas en programación y modelación de datos. [10]

Pero existen un cierto número de ventajas entre el modelo orientado a objetos con respecto al modelo estructurado, entre estas tenemos las más importantes:

- ✓ Un modelo de objetos es más cercano a la realidad que un modelo funcional.
- ✓ Un desarrollo realizado con el modelo orientado a objetos es más fácil de mantener y de reutilizar.
- ✓ El modelo orientado a objetos evita la redundancia en los procesos luego los códigos son más entendibles y resumidos.
- ✓ La integridad que dan los objetos a los datos evita ambigüedades en su uso, dando mayor seguridad en los resultados.
- ✓ El modelo orientado a objetos facilita la integridad de módulos que hallan sido realizados por separado sin correr riesgos en el manejo de los datos. [11]

Viendo las ventajas de la OO, la tendencia futura es a que se vaya imponiendo sobre la metodología estructurada clásica. A continuación se explicará el análisis de requerimientos de una Metodología Orientada a Objetos.

2.1 METODOLOGÍA ORIENTADA A OBJETOS

La metodología más utilizada en el desarrollo de software OO es la RUP *(Rational Unified Process, por sus siglas en inglés)* es un proceso de desarrollo de Software que proporciona una guía en el orden de las actividades de un equipo, dirige las tareas individuales de los desarrolladores, especifica que productos deberían ser desarrollados y ofrece criterios para monitorear y medir los productos y actividades de un proyecto.

Para la OO, se necesita realizar un análisis y diseño orientado a objetos. El modelamiento visual es la clave para realizar el análisis OO. Desde los inicios del desarrollo de software OO han existido diferentes metodologías para hacer esto del modelamiento, pero sin lugar a duda, el Lenguaje de Modelamiento Unificado (UML) puso fin a la guerra de metodologías. Según los mismos diseñadores del lenguaje UML, éste tiene como fin modelar cualquier tipo de sistemas (no solamente de software) usando los conceptos de la orientación a objetos. Y además, este lenguaje debe ser entendible para los humanos y máquinas.

Actualmente en la industria del desarrollo de software tenemos al UML como un estándar para el modelamiento de sistemas OO. Fue la empresa Racional que creó estas definiciones y especificaciones del estándar UML, y lo abrió al mercado. La misma empresa creó uno de los programas más conocidos hoy en día para este fin; el Racional Rose, pero también existen otros programas como el Poseidon que trae licencias del tipo community edition que permiten su uso libremente. El UML consta de todos los elementos y diagramas que permiten modelar los sistemas en base al paradigma orientado a objetos. Los modelos orientados a objetos cuando se construyen en forma correcta, son fáciles de comunicar, cambiar, expandir, validar y verificar. Este modelamiento en UML es flexible al cambio y permite crear componentes plenamente reutilizables [12].

2.1.1. ETAPAS DEL PROCESO UNIFICADO

Este proceso de desarrollo considera que cualquier desarrollo de un sistema software debe pasar por cuatro fases que se describirán a continuación, la figura 2 muestra las fases de desarrollo y los diversos flujos de trabajo involucrados dentro de cada fase con una representación gráfica en cual de los flujos se hace mayor énfasis según la fase, cabe destacar el flujo de trabajo concerniente al negocio.

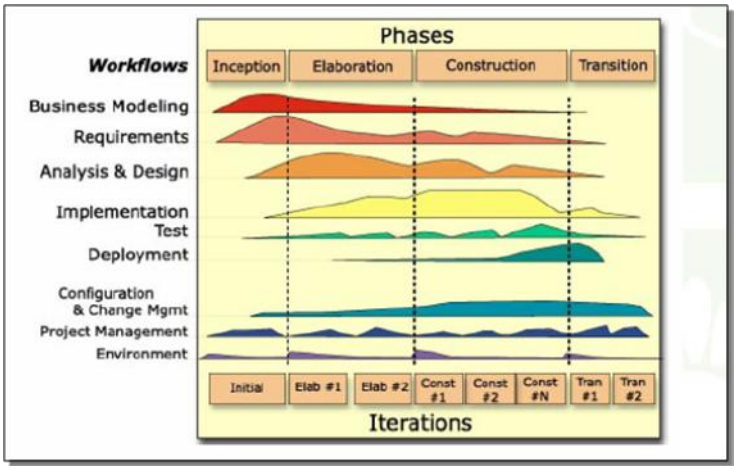
1. Inicio
Visión aproximada, análisis del quehacer de la empresa cliente ("el negocio"), alcance del proyecto, estimaciones (imprecisas) de plazos y costos. Se define la viabilidad del proyecto.

2. Elaboración
Visión refinada, implementación iterativa del núcleo central de la aplicación, resolución de los riesgos más altos, identificación de nuevos requisitos y nuevos alcances, estimaciones más ajustadas.

3. Construcción
Implementación iterativa del resto de los requisitos de menor riesgo y elementos más sencillos, preparación para el despliegue (entrega, instalación y configuración).

4. Transición
Pruebas beta, despliegue.

Figura 2. Fases del RUP [14]



La figura 3 resume las disciplinas del UP y sus artefactos asociados, indicando también, para las siguientes fases, el grado aproximado de desarrollo de cada uno de estos artefactos.

c: comienzo de la construcción del artefacto.
r: refinamiento del artefacto (ampliación, corrección).

Figura 3. Requerimientos y el análisis de requerimientos con el RUP

2.1.1.1 Análisis de Requerimientos en Rup

El análisis da como resultado un modelo del sistema que pretender ser correcto, completo, consistente y verificable. Los desarrolladores formalizan la especificación del sistema producida durante la obtención de requerimientos y examinan con mayor detalle las condiciones de frontera y los casos excepcionales. También corrigen y aclaran la especificación del sistema si es que encuentran errores o ambigüedades. El cliente y el usuario están involucrados, por lo general, en esta actividad, en especial cuando se necesita cambiar la especificación del sistema y cuando se necesita recopilar la información adicional.

En el análisis orientado a objetos, los desarrolladores construyen un modelo que describe al dominio de aplicación. Por ejemplo, el modelo de análisis de un reloj describe la manera en que el reloj representa al tiempo (por ejemplo, ¿ sabe el reloj acerca de los años bisiestos? ¿sabe acerca del día de la semana? ¿sabe acerca de las fases de la luna?) el modelo de análisis se extiende luego para describir la manera en que interactúan los actores y el sistema para manipular el modelo del dominio de aplicación (por ejemplo ¿cómo ajusta la hora el propietario del reloj? ¿cómo ajusta el día de la semana?). Los desarrolladores usan el modelo de análisis, junto con los requerimientos no funcionales, para preparar la arquitectura del sistema que se desarrolla durante el diseño de alto nivel.

Componentes del UP		Fases del UP			
Disciplina	Artefacto Iteraciones:	Inicio I1	Elaboración E1...En	Construcción C1...Cn	Transición T1...Tn
Modelado del negocio	Modelo del dominio		c		
Requerimientos	Modelo de Casos de Uso Visión y Análisis del Negocio Especificación Complementaria Glosario	c c c c	r r r r		
Diseño	Modelo de Diseño Documento de Arquitectura Modelo de Datos		c c c	r r	
Implementación	Modelo de implementación		c	r	r
Gestión del proyecto	Plan de desarrollo	c	r	r	r
Pruebas	Modelo de Pruebas		c	r	
Entorno	Marco de desarrollo	c	r		

El análisis se enfoca en la producción de un modelo del sistema, llamado el modelo de análisis, que es correcto, completo, consistente y verificable. El análisis se diferencia de la obtención de requerimientos en que los desarrolladores se enfocan en la estructuración y formalización de los requerimientos obtenidos de los usuarios. El modelo de análisis está compuesto por tres modelos individuales: el modelo funcional, representado por casos de uso y escenarios, el modelo de objetos de análisis, representado por diagramas de clase y objeto, y el modelo dinámico, representado por gráficas de estado y diagramas de secuencia.

Para realizar análisis a los requerimientos en la metodología RUP se utilizan los diagramas de casos de uso y plantillas que contienen información que los describen (Nombre Caso de Uso, Autor, Identificador, Prioridad, Fecha, Categoría, Resumen, Curso Básico Eventos, Caminos Alternativos, Caminos de Excepción, Puntos de Extensión, Pre – Condiciones, Post- Condiciones, Actores involucrados). Un *diagrama* es la representación gráfica de un conjunto de elementos, visualizado la mayoría de las veces como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se dibujan para visualizar un sistema desde diferentes perspectivas, de forma que un diagrama es una proyección de un sistema. [15]

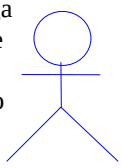
Una de las características de RUP es que es un proceso dirigido o conducido por *Casos de Uso*, los cuales son usados para capturar los requerimientos funcionales de una aplicación. RUP define entonces un proceso de administración de requerimientos basado en casos de uso. [13]

Diagrama de Casos de Uso [15]

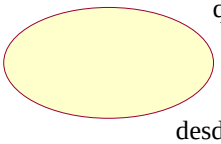
El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en

como los elementos interactúan (operaciones o casos de uso). Un diagrama de casos de uso consta de los siguientes elementos: Actor, Casos de uso, Relaciones de uso, herencia y comunicación.

Actor: Es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra **rol**, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema..



Caso de Uso: Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.



Relaciones:

Asociación

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

Dependencia o Instanciación

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

Generalización

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de **Uso** (<<uses>>) o de **Herencia** (<<extends>>). Este tipo de relación esta orientado exclusivamente para casos de uso (y no para actores).

Extends: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

Uses: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica. De lo anterior cabe mencionar que tiene el mismo paradigma en diseño y modelamiento de clases, en donde esta la duda clásica de **usar** o **heredar**.

3. CASO ESTUDIO

3.1. Captura de Requisitos para Videoclub Automático

El sistema a desarrollar es un sistema software que hay que incorporar dentro de un dispensador automático de películas, un videoclub automático, como los que hay en gasolineras y tiendas abiertas 24h. Este videoclub tiene las facilidades usuales, como sacar películas, devolverlas, etc., y también otras como que se puede reservar una película notificando al usuario que ya esta disponible mediante un mensaje SMS o enviar recordatorios a los clientes que no han devuelto películas vía SMS. El pago ha de hacerse con tarjeta de crédito. El videoclub reporta a la central de la empresa vía Internet.

3.2. Algunos Casos de Uso identificados

Alquilar Película
El *Usuario* utiliza este caso de uso para mirar qué películas hay, seleccionar una, pagarla y retirarla del dispensador. Debe obtener un recibo.

Devolver Película
El *Usuario* inicia este caso cuando devuelve la película. Recibe una notificación confirmándole la devolución. Si ha tenido retraso deberá abonar una penalización.

Dar de Alta
El *Operario* inicia este caso cuando introduce una nueva película en el dispensador.

Dar de baja
El *Operario* inicia este caso cuando retira una película del dispensador.

Mandar Recordatorio
El *Operario* inicia este caso cuando fija en el sistema el tiempo máximo permitido por el alquiler de cada película.

3.3. Actores Identificados

Usuario
Un Usuario es la persona que utilizará el sistema para buscar películas, solicitarlas, pagarlas y obtenerlas del dispensador.

Operario

Un Operario es la persona que realizará las tareas de mantenimiento del dispensador de películas, es decir, sustituirá viejas películas por nuevas. Para ello utilizará el sistema para dar de alta y baja las películas.

Banco
El Banco representa a la entidad externa con la que el sistema debe contactar para admitir y realizar los pagos con tarjeta de los clientes.

Dispensador
El Dispensador representa al dispositivo mecánico donde están almacenadas las películas que el sistema debe controlar para dar la película alquilada al cliente o que éste la pueda devolver.

Operador Telefónico
El Operador Telefónico representa la entidad externa con la que el sistema debe contactar para enviar mensajes SMS a los clientes.

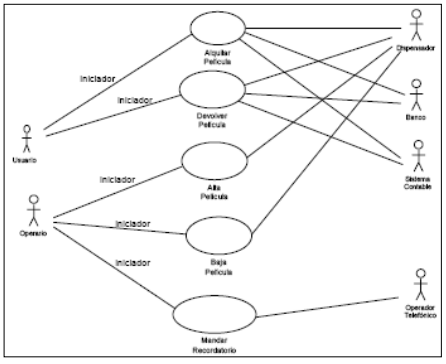
Sistema Contable
El Sistema Contable representa el sistema informático externo de la empresa con el que el sistema videoclub debe contactar para notificar los pagos realizados en cada momento.

Precondición
El usuario ha pulsado en el videoclub "Alquilar Película"
Flujo de Eventos
Camino Básico
1. El Usuario inicia el caso de uso buscando en la pantalla la película que más le gusta. El sistema le muestra la información de cada película, por orden alfabético, así como si está prestada, reservada o disponible.
2. El usuario selecciona una película en la pantalla. El sistema le pide los datos de su tarjeta de crédito y gestiona el pago con la entidad bancaria adecuada. El banco emite una verificación y el pago es aceptado.
3. El sistema ordena al dispensador que proporcione la película elegida y que le genere un recibo. Asimismo notifica al sistema contable la transacción bancaria realizada. Finalmente actualiza los datos históricos del usuario y marca la película como alquilada y no disponible si no hay más copias.

Caminos Alternativos
4. En 1 puede que la película esté alquilada y el usuario quiera reservarla. En este caso hay que emitir una notificación de reserva.
5. En 2 puede que el pago no sea aceptado por el banco o que el usuario haya alcanzado el tope de sus alquileres por día. Hay que notificárselo al usuario y terminar.

Poscondición
El caso acaba cuando el usuario tiene la película y un recibí, o sin película y una notificación de pago rechazado, o sin película pero con la reserva hecha, o sin película porque no le gustara ninguna.
Atributos del Caso de Uso
Película: prestada, reservada o disponible
Cliente (Usuario): datos tarjeta de crédito, número de películas alquiladas
Información a mostrar: Sobre películas, sobre petición de datos, sobre resultado del alquiler realizado.

3.4. Diagrama de Casos de Uso



3.5 Descripción del Caso de Uso "Alquilar Película" del Videoclub [16]

CONCLUSIONES

- Se cuenta con una visión sobre el proceso de desarrollo de software y se debe asumir el compromiso de establecer los requerimientos del sistema como parte esencial en el éxito de los proyectos.
- Es muy importante definir la visión de los involucrados en el desarrollo de software, en términos de sus necesidades y características esperadas del producto.
- Todo proyecto de desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que se obtiene son clientes insatisfechos con el resultado.
- Los proyectos que generan problemas, son los que aumentan el presupuesto, se demoran más del tiempo estimado o simplemente no cumplen con las expectativas del cliente.

REFERENCIAS

[1] Metodologías de Desarrollo de Software, disponible en, http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf

[2] J. Jesús María Zavala Ruiz, ¿Por Qué Fracasan los Proyectos de Software?; Un Enfoque Organizacional, 2004 disponible en, http://www.consol.org.mx/2004/material/63/por-que-fallan-los-proy-de-soft.pdf

[3] Ingeniería de software , disponible en, http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software

[4] ZELKOVITZ, M.V. Shaw, A.C. y GANNON, J.D. Principles of Software Engineering and Design.

[5] Object Oriented Design, por Grady Booch, disponible en, http://www.cs.ualberta.ca/~pfiguero/soo/metod/ood.html

[6] Ingeniería de software, disponible en, http://www.monografias.com/trabajos12/ingreq/ingreq.shtml

[7] Líneas de Investigación de la Ingeniería de Software, disponible en, http://delta.cs.cinvestav.mx/~pmejia/investiga.ppt

[8] SOMMERVILLE, Ian, Ingeniería de Software Segunda Edición, University Of Strathclyde Pag 20-21 y 34, 1998, por Addison-Wesley Iberoamericana, S.A.

[9] http://www.foros-fiuba.com.ar/download.php?id=59

[10] JOYANES AGUILAR Luis, Programación Orientada a Objetos- Conceptos, Modelado, Diseño y Codificación en C++ McGraw-Hill INTERAMERICANA DE ESPAÑA, S.A. 1996 pp 16.

[11] Programación Orientadas a Objetos, disponible en, http://ieee.udistrital.edu.co/concurso/programacion_orientada_objetos/poo/poovsest.html

[12] Tecnología Orientadas a Objetos, disponible en,http://java.ciberaula.com/articulo/tecnologia_orientada_objetos/

[13] Análisis, diseño e implementación de un sistema de información para la gestión de calidad basado en la arquitectura multicapa, disponible en,http://ingenieria.sanmartin.edu.co/graduacion/ProcesoGraduacion//Graduacion/APublicar/Ejemplos/2006_07_Articulo.pdf

[14] El Proceso Unificado (UP), http://iie.fing.edu.uy/ense/asign/desasoft/Teorico2/ProcesoUnificado.pdf

[15] BOOCH Grady, Unified Modeling Language. El Lenguaje Unificado de Modelado.

[16] Ejemplo de Casos de Uso, gestión de un Video Club, disponible en, www.dsic.upv.es/asignaturas/facultad/lsi/trabajos/032000.ppt

[IEEE, 1990] *IEEE Recommended Practice for Software Requirements Specifi*cations. IEEE/ANSI Stan

Autora

Ingeniera de Sistemas de la Universidad Distrital Francisco José de Caldas en convenio con la Universidad de la Amazonia, en la de Ciudad de Florencia, Colombia. Es estudiante de Maestría en Ciencias de la Información y las Comunicaciones de la Universidad Distrital Francisco José de Caldas.

Actualmente se desempeña como docente en el área de informática en la Universidad de la Amazonia de Florencia, Colombia, y pertenece como investigador al grupo GIIE (Grupo de Investigación en Informática Educativa).
e-mail: yoispascuas@hotmail.com

SOCIEDAD DE LA INFORMACION

www.sociedadelainformacion.com

Edita:

Director: José Ángel Ruiz Felipe
Jefe de publicaciones: Antero Soria Luján
D.L.: AB 293-2001
ISSN: 1578-326x