UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 5 ENTORNOS VIRTUALES

Mejora de Calidad en el proceso de desarrollo de software dirigido por casos de uso

Trabajo para optar por el Título de Ingeniería en Ciencias Informáticas

Autores
Yaima Bety Suárez Gijón
Janet Díaz Magdariaga

Tutor
Ing. Amado Espinosa Hidalgo
Co-Tutor
Lic. Daciel Alberto Olivera Cortina
Asesor
Ing. Jandrich Domínguez Fortién

Junio, 2007

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____	_____
Yaima B Suárez Gijón	Janet Díaz Magdariaga	Amado Espinosa Hidalgo

AGRADECIMIENTOS

A los profesores que nos han guiado durante estos cinco años de la carrera y han hecho posible que este día llegara con su apoyo y enseñanza.

A los compañeros de estudio, quienes han sido como nuestras familias, brindado cariño, apoyo y confianza durante la carrera.

Especialmente al profesor Daciel Alberto jefe de Departamento de Matemática Aplicada de la Universidad de las Ciencias Informáticas por su bondad y preocupación en todo momento que necesitamos aclarar alguna duda.

A la profesora Madelín Haro y Jandrich Domínguez por su apoyo e ideas que aportaron para seguir adelante en la investigación.

Especialmente a Pedro Rodríguez Samón por su apoyo y preocupación en todo momento.

Agradecerle a la Revolución por darnos la oportunidad de haber estudiado en la 1era. Universidad surgida en el calor de la batalla de ideas.

En fin, agradecerle a la UCI por darnos la oportunidad de utilizar todos los medios y condiciones necesarias para poder llevar a cabo la investigación.

A todos Muchas Gracias!!!!

DEDICATORIA

Especialmente a mis padres Zuzel Magdariaga y Modesto Díaz, por su apoyo, ayuda, comprensión y sobretodo por su amor durante mi vida y mi carrera.

A mi novio Andrys E. Moreno por su confianza y comprensión durante la carrera.

A mis familiares quienes me han brindado toda su ayuda y amor para salir adelante.

Gracias, Janet

A mis padres Marilín Gijón y Juan E. Suárez que siempre con amor me han apoyado en todo, me han enseñado mucho en la vida y han colaborado en la formación de quien soy.

A mis abuelos paternos Bety, Ramiro y maternos Idelsa e Israel que con su bendición siempre he salido adelante en todo lo que me propongo.

A mi hermano, tías y demás familiares que con su apoyo y sus consejos me han ayudado a enfrentar cualquier problema.

A mi novio Pedro Rodríguez por darme su amor y estar junto a mí brindándome su apoyo y confianza.

Gracias, Yaima Bety

RESUMEN

El proceso de desarrollo de software dirigido por casos de uso es una de las prácticas más utilizadas en el mundo, han alcanzado gran aceptación en el desarrollo de sistemas software por su sencillez, adaptabilidad y facilidad de compresión.

El objetivo que persigue este trabajo de diploma investigativo es proponer un conjunto de técnicas basadas en el desarrollo dirigido por casos de uso que haga más eficiente el proceso de desarrollo de software y contribuya a la mejora de la calidad de los productos construidos en la facultad 5 de la UCI.

Para fundamentar la investigación se hizo un estudio en los proyectos productivos de la facultad que utilizan la metodología RUP para conocer como se lleva a cabo el proceso de desarrollo de software, donde se realizó una encuesta para detectar los problemas que presentan y a la vez hacer una propuesta que contribuya a la solución de los mismos con el fin de crear las bases a la mejora de la calidad en la realización de los productos.

Palabras claves

Proceso de desarrollo de software, casos de uso, mejora de calidad

ÍNDICE

INTRODUCCIÓN

En los últimos años la industria del software ha alcanzado un alto nivel de desarrollo, cada día los sistemas se tornan más complejos, los clientes exigen soluciones mas novedosas y con mayor calidad. La obtención de un producto con calidad implica la adopción por parte del equipo de desarrollo de metodologías y estándares para la construcción del software. Permitiendo uniformar los esquemas de trabajo, normalizando cada área de conocimiento de la disciplina encargada del estudio del desarrollo de sistemas informáticos. Esta es llamada desde la década de los 80 como Ingeniería de Software.

En nuestro país el desarrollo de software ha tenido un impacto económico-social en la venta de sus productos. Como aporte a la economía del país surge la Universidad de las Ciencias Informáticas (UCI) con el objetivo de formar ingenieros con un nivel altamente calificado en la rama de la informática y que durante su formación estén vinculados a la producción de software.

Actualmente en los proyectos productivos de la Facultad 5 de la UCI se han detectado irregularidades en el proceso de desarrollo, donde la planificación, especificación de requerimientos y pruebas son algunas de la áreas mas afectadas [1]. Impactando los atributos de calidad fijados para cada uno de los productos a desarrollar. Este trabajo forma parte de un conjunto de investigaciones dirigidas a mejorar las técnicas relacionadas con la producción de software en procesos dirigidos por casos de uso.

Por tanto el **problema** a resolver queda formulado a modo de interrogante de la siguiente forma: ¿Cómo mejorar los procesos de desarrollo de software dirigidos por casos de uso en la facultad 5 de la Universidad de las Ciencias Informáticas?

El proceso de desarrollo del software **es el objeto de estudio,** y el **campo de acción** que abarca este trabajo es el desarrollo dirigido por casos de uso.

El **objetivo general** de esta investigación es proponer un conjunto de técnicas basadas en el desarrollo dirigido por casos de uso que haga más eficiente el proceso de desarrollo de software y contribuya a la mejora de la calidad de los productos construidos en la facultad 5 de la UCI.

Como **idea a defender** se propone ajustar el proceso de desarrollo de software de los proyectos de la facultad 5 aplicando técnicas centradas en los casos de uso creando las bases a la mejora de la calidad.

De ahí se derivan las siguientes **tareas de investigación**:

- Estudiar la conceptualización del proceso de desarrollo de software dirigido por casos de uso.
- Estudiar la metodología RUP (que es la que esta dirigida por casos de uso).
- Realizar encuesta a los proyectos de la facultad que utilizan la metodología RUP.
- Analizar los resultados obtenidos de las encuestas aplicadas para conocer la forma que desarrollan sus productos y las principales deficiencias que presentan al hacerlo.
- Proponer el conjunto de técnicas que den solución al objetivo planteado.

Con el propósito de desarrollar las tareas planteadas, se utilizaron los métodos de investigación siguientes:

El método teórico utilizado en el presente trabajo es el Analítico – Sintético ya que mediante la consulta de libros, revistas, monografías, sitios web, catálogos, publicaciones se ha hecho un análisis profundo del objeto de estudio para sintetizar todas las citas, apuntes, datos, resúmenes tomados al respecto.

También se puso en práctica el método Empírico donde se usó la encuesta para la recogida de información sobre cómo se realiza el proceso de desarrollo de software dirigido por casos de uso en los proyectos productivos de la facultad 5 y la entrevista para recoger las opiniones de la situación actual sobre el tema investigado.

Estructuración del contenido y breve explicación

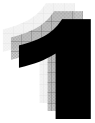
Este trabajo esta conformado por tres capítulos:

En el Capítulo 1. Fundamentación Teórica se abordan aspectos y conceptos generales de las definiciones actuales del proceso de desarrollo de software dirigido por casos de uso, los cuales servirán de guía a lectores interesados en el tema. Además se expresa lo referente a este tema en cuanto al estado del arte en el Mundo y en Cuba.

En el *Capitulo 2. Estudio Preliminar* se desarrolló un análisis en los proyectos en cuanto al uso de la metodología RUP y las técnicas de casos de uso, donde se aplicó una encuesta a cada uno de ellos para

detectar las irregularidades que cometen a la hora de hacer sus productos y se realizó un estudio estadístico para demostrar los problemas existentes.

En el *Capitulo 3. Propuesta de solución* se proponen las técnicas basadas en casos de uso para mejorar el proceso de desarrollo de software a partir de los problemas detectados en el capítulo 2.



Capítulo 1 Fundamentación Teórica

Introducción

En la actualidad los casos de uso son un método que, justamente, ayudan al Ingeniero de Software a llevar adelante el desarrollo de un sistema de software. En el presente capítulo se abordan aspectos y conceptos generales de las definiciones actuales del proceso de desarrollo de software dirigido por casos de uso, los cuales servirán de guía a lectores interesados en el tema. Además se expresa lo referente a este tema en cuanto al estado del arte en el Mundo y en Cuba.

1.1 Aceptación de casos de uso en la industria del software

Actualmente la Industria Cubana del Software (ICSW) está llamada a convertirse en una significativa fuente de ingresos nacional, como resultado del correcto aprovechamiento de las ventajas del considerable capital humano disponible. La promoción de la industria cubana del software en el ámbito internacional ha tenido como línea estratégica aprovechar la enorme credibilidad que tiene Cuba en sectores tales como la salud, la educación y el deporte. Continuar la producción sostenida de software de alta calidad en prestaciones, imagen y soporte, para satisfacer las necesidades nacionales en estos sectores, tiene ya una positiva repercusión en el incremento de la exportación. [2]

La Universidad de las Ciencias Informáticas y el sistema de empresas cubanas vinculadas a la producción de software jugarán un papel importante en el desarrollo de la Industria Cubana del Software, y en la materialización de los proyectos asociados al programa cubano de informatización.

En la actualidad el proceso de desarrollo de software dirigido por casos de uso es una de las mejores prácticas utilizadas en el mundo, la cual permite capturar y analizar la funcionalidad de un sistema desde el punto de vista del usuario, que desarrollado a través de diferentes modelos permite su implementación y posterior actualización.

Los casos de uso representan las condiciones y capacidades que el sistema de software deberá cumplir y tener respectivamente, desde el punto de vista funcional y son las únicas piezas de información utilizadas por todos los participes del sistema software. Debido a la aceptación que han alcanzado en el mundo en cuanto al desarrollo de software, se realizaron visitas a empresas cubanas con el fin de conocer si utilizaban los casos de uso para el desarrollo de sus productos, a raíz de la misma se puede decir de manera general que las empresas cubanas están utilizando los casos de uso, aunque no en todo el proceso de desarrollo. Por ejemplo en la empresa Tecnomática del Minbas utilizan los casos de uso en la captura de requisitos ya que les permite recoger al detalle los requisitos del sistema definiendo todo lo que el cliente quiere que su sistema haga y además facilitando la comprensión del mismo.

Cuando los casos de uso son bien aplicados, las probabilidades de éxito de los proyectos de ingeniería de software se incrementa de manera importante. Se utilizan básicamente en el proceso de modelado de sistemas, partiendo de una percepción o perspectiva que nos plantea el paradigma de la orientación a objetos, ya sea desde una petición de un actor o bien desde la invocación desde otro caso de uso. Se centran en describir cómo es el diálogo entre el usuario y el sistema y se prolongan a lo largo del tiempo mientras dure la interacción del usuario con el sistema.

Todo sistema de software ofrece a su entorno una serie de servicios, donde el caso de uso es una forma de expresar cómo alguien o algo externo a un sistema lo usa. Cuando decimos “alguien o algo” hacemos referencia a que los sistemas son usados no sólo por personas, sino también por otros sistemas de hardware y software.

Es importante destacar que los casos de uso poco tienen que ver con entender a un sistema como un conjunto de objetos que interactúan, que es la premisa básica del análisis orientado a objetos. En este sentido, el éxito de los casos de uso no hace más que dar la razón al análisis estructurado, que propone

que la mejor forma de empezar a entender un sistema es a partir de los servicios o funciones que ofrece a su entorno, independientemente de los objetos que interactúan dentro del sistema para proveerlos.

Los casos de uso son independientes del método de diseño que se utilice, y por lo tanto del método de programación. Luego de documentar los requerimientos de un sistema con casos de uso, se puede diseñar un sistema estructurado (manteniendo una separación entre datos y funciones), o un sistema Orientado a Objetos, sin que la técnica sea de mayor o menor utilidad en alguno de los dos casos. Esto da más flexibilidad al método, y probablemente contribuya a su éxito.

A partir de la publicación del libro de Jacobson, gran parte de los más reconocidos especialistas en métodos Orientados a Objetos coincidieron en considerar a los casos de uso como una excelente forma de especificar el comportamiento externo de un sistema. De esta forma, la notación de los casos de uso fue incorporada al lenguaje estándar de modelado UML (Unified Modelling Language) propuesto por Ivar Jacobson, James Rumbaugh y Grady Booch, tres de los precursores de las metodologías de Análisis y Diseño Orientado a Objetos, y avalada por las principales empresas que desarrollan software en el mundo. [3]

Se han escrito diversas definiciones de casos de uso, algunas de ellas serán mostradas a continuación y luego se plasmará una definición propia.

“La perspectiva que proporcionan los casos de uso refuerza el objetivo ultimo de la ingeniería del software: la creación de productos que permitan a los clientes realizar un trabajo útil**’’**

Karl Wieger,

“Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto**’’**

Ivar Jacobson, Grady Booch y James Rumbaugh

“El caso de uso es un documento narrativo que describe la secuencia de eventos de un actor que utiliza un sistema para completar un proceso**’’**

Jacobson 92

1.2 Casos de uso

1.2.1 ¿Qué es Casos de Uso?

"Un caso de uso es una técnica para la captura de requisitos de un nuevo sistema o una actualización software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico**''**. [4]

A partir de las definiciones anteriormente reflejadas se puede decir que un caso de uso no es más que las funcionalidades que componen a un sistema, o a un negocio. Describe qué acciones debe desarrollar el producto que se va a desarrollar y cómo se deben realizar dichas acciones desde el punto de quienes utilizarán el sistema.

1.2.2 Características principales de los casos de uso

Las características principales de los casos de uso son que están expresados desde el punto de vista del actor, se documentan con texto informal describiendo la interacción, se describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él, aunque el énfasis está puesto en la interacción, son iniciados por un único actor y están acotados al uso de una funcionalidad claramente definida del sistema.

1.2.3 Identificación de los casos de uso

La identificación de casos de uso requiere un conocimiento medio acerca de los requisitos, y se basa en la revisión de los documentos de requisitos existentes sobre la especificación de los requerimientos y en el uso de la técnica de brainstorming (tormenta de ideas) entre los miembros del equipo de desarrollo.

Un método con que se identifican los casos de uso se basa en los actores

- Se identifican los actores relacionados con un sistema o empresa.
- En cada actor, se identifican los procesos que inician o en que participan.

Un segundo método de identificación de los casos de uso se basa en eventos

- Se identifican los eventos externos a los que un sistema ha de responder.
- Se relacionan los eventos con los actores y con los casos de uso.

Las funciones del sistema identificadas durante la especificación previa de requerimientos deben asignarse a los casos de uso. Debe ser posible verificar, mediante la sección de referencias cruzadas, que todas las funciones hayan sido asignadas, logrando un vínculo importante respecto a la trazabilidad entre los artefactos. Todas las funciones y casos de uso del sistema deberían analizarse hasta la implementación y la aplicación de pruebas.

En la práctica, los casos de uso pueden expresarse con diverso grado de detalle y de aceptación de las decisiones concernientes al diseño. En otras palabras, un mismo caso de uso pueden escribirse en diferentes formatos y con diversos niveles de detalle.

Formato de alto nivel: Un caso de uso de alto nivel describe un proceso muy brevemente, casi siempre en dos o tres enunciados.

Formato expandido: Un caso de uso expandido describe un proceso más a fondo que el de alto nivel. La diferencia básica con el caso de uso de alto nivel consiste en que tiene una sección destinada al curso normal de los eventos que los describe paso por paso.

1.2.4 Objetivos de los casos de uso

Los casos de uso fueron creados con el objetivo de capturar los requisitos funcionales del sistema y describir los mismos en el proceso de desarrollo de software de forma legible y precisa. Estos permiten simplificar la construcción de los modelos de objetos y servir de base para las pruebas del sistema. Los casos de uso ayudan a los desarrolladores a encontrar las clases, además que ayudan a los jefes de proyecto a planificar, asignar, y controlar muchas de las tareas que los desarrolladores realizan.

Por cada caso de uso, un jefe de proyecto puede identificar unas cuantas tareas como por ejemplo: cada caso de uso a especificar, diseñar y probar es una tarea. Las tareas identificadas a partir de los casos de uso ayudan a los jefes a estimar el tamaño del proyecto y los recursos necesarios. Un caso de uso describe un proceso, un proceso de negocios. Un proceso describe, de comienzo a fin, una secuencia de eventos, de acciones y transacciones que se requieren para producir u obtener algo de valor para una empresa o actor.

Cada caso de uso describe una forma de utilizar el sistema, son el punto de partida ideal para explicar cómo puede el usuario interactuar con el sistema. Los casos de uso están diseñados para cumplir los deseos del usuario cuando utiliza el sistema. Los casos de usos enlazan todas las actividades del desarrollo y dirigen el proceso de desarrollo, este es quizás el beneficio más importante de la aproximación dirigida por los casos de usos.

1.2.5 Ventajas de los casos de uso

Los casos de usos proporcionan beneficios innumerables a lo largo del ciclo vida del desarrollo del software, a continuación se mencionan los más significativos:

- Ayudan a asegurar que se desarrolla el sistema correcto.
- Excelente forma de comunicación con los clientes y los usuarios.
- Ayudan a gestionar la complejidad de los proyectos grandes.
- Ofrecen una buena base para la verificación y validación.
- Modo objetivo para el seguimiento del proyecto.
- Documentan las respuestas funcionales de caja negra.
- Proporcionan el fundamento de los mensajes.
- Pueden servir como base para especificar respuestas a aplicaciones de control y tiempo real.

1.2.6 Desventajas de los casos de uso

Al igual que los casos de uso proporcionan innumerables beneficios en todo el ciclo de vida, estos también conllevan a una descomposición funcional del sistema, por ejemplo:

- Los casos de uso son funcionales por naturaleza (esto es, localizan la información en torno a las funciones).
- Debe tenerse cuidado cuando se utilizan dentro de un desarrollo orientado a objetos.
- Los problemas pueden surgir cuando en un desarrollo software se utilizan diferentes estrategias.

Violación de la ocultación de la información.

- Cuando se describen casos de uso, se debe conocer no sólo el elemento para el que se desarrolla el caso de uso, sino también la interfaz pública definida de cada elemento.
- Los autores de casos de uso deben evitar la tentación de ir más allá de la interfaz pública, e intentar describir la estructura interna del elemento.

Falta de formalidad.

- La informalidad de los casos de uso lleva a la gente aun falso sentido de seguridad.
- La gente se olvida de las normas para los nombres y otras convenciones de estilo.
    - Aumenta la posibilidad de cometer errores
    - Disminuye la probabilidad de reutilizar el caso de uso

No saber cuando parar.

- Existe una gran confusión entre la adquisición de los requisitos y los casos de uso y entre el diseño y los casos de uso.

La cobertura es el mayor problema de quien usa los casos de uso:

- Una cosa es decir que “el conjunto de todos los casos de uso especifican la totalidad de la funcionalidad del sistema” …
- Otra cosa es demostrar que se ha capturado por completo la funcionalidad del sistema en un conjunto de casos de uso.

1.3 Requisitos del software

1.3.1 ¿Qué es captura de requisitos?

La aplicación principal de los casos de uso es en el proceso de análisis y diseño, pero de manera particular en la definición de requerimientos del usuario, siendo una excelente herramienta de comunicación debido a la sencillez de su elaboración así como su comprensión.

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae de cualquier fuente de información disponible las necesidades que debe cubrir dicho sistema.

El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa.

1.3.2 Técnicas para la captura de requisitos

Existen varias técnicas para la captura de requisitos, por ejemplo entrevistas, desarrollo conjunto de aplicaciones, tormenta de ideas, mapas conceptuales, croquis, casos de uso, cuestionarios y listas de chequeo, comparación de terminología, las cuales se especifican a continuación.

- Entrevistas: resultan ser una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista). A pesar de que las entrevistas son esenciales en el proceso de la captura de requisitos y con su aplicación el equipo de desarrollo puede obtener una amplia visión del trabajo y las necesidades del usuario, es necesario destacar que no es una técnica sencilla de aplicar. Requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado.

- Desarrollo conjunto de aplicaciones: esta técnica resulta una alternativa de las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional, es decir, durante la aplicación de la técnica se trabajará sobre lo que se generará. Tras una fase de preparación del Desarrollo conjunto de aplicaciones al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación. Durante la sesión se discute en grupo sobre estos temas, llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema. Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados.

- Tormenta de ideas: es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. Como técnica de captura de requisitos es sencilla de usar y de aplicar, contrariamente a la técnica Desarrollo conjunto de aplicaciones, puesto que no requiere tanto trabajo en grupo como éste. Además

suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

- Mapas conceptuales: Los mapas conceptuales son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy subjetivos y pueden llegar a ser ambiguos en casos complejos, si no se acompaña de una descripción textual.

- Croquis: Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces del usuario. Estos croquis pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación. Se relaciona con la técnica Historia del usuario (storyboard).

- Casos de Uso: Aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para la captura de requisitos. Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas, etc.) que interactúan con el sistema como si se tratara de una caja negra. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores. La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades.

- Cuestionarios y Listas de chequeo: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (lista de chequeo). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.

- Comparación de terminología: Uno de los problemas que surge durante la elicitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es

utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste).

Existen más técnicas para la captura de requisitos (Storyboard(historia del usuario), análisis de otros sistemas, estudio de la documentación, entre otras), incluso también es común encontrar alternativas que combinen varias de estas técnicas. Sin embargo, las presentadas ofrecen un conjunto representativo de las más utilizadas.

1.4 Atributos de requisitos

Un atributo de requisito es un campo descriptivo de información asociada con un requisito. Los atributos de requisitos son atributos del sistema (definido por un producto del Rational integrado) o atributos definidos por el usuario (definidos por el propietario del proyecto). [5]

1.5 Gestión de Requisitos

La Gestión de Requisitos en Ingeniería de Software, es el proceso encargado de la identificación, asignación y seguimiento de los requisitos, incluyendo la verificación, modificación y control del estado a lo largo del ciclo de vida. Es el conjunto de actividades que se concentra en el aseguramiento de las especificaciones. Es el proceso que inicia con la concepción de un proyecto y continúa hasta el resultado final del producto. Los cambios de requisitos deben ser gestionados para asegurar que la calidad de los mismos se mantenga, los problemas suscitados por los cambios de requisitos podrían incurrir en altos costos, siendo el requisito factor crítico de riesgo. [6]

El objetivo de la gestión de requisitos es gestionar los requisitos de los elementos del proyecto y sus componentes e identificar inconsistencias entre estos requisitos, el plan de proyectos y los elementos de trabajo.

En este proceso se deben gestionar todos los requisitos del proyecto, tanto los requisitos técnicos como los no técnicos.

Estos requisitos han de ser revisados conjuntamente con la fuente de los mismos así como con las personas que se encargarán del desarrollo posterior. [7]

1.5.1 Principales tareas de la Gestión de Requisitos

La Gestión de requisitos se basa en cuatro actividades principales (Recolección, Documentación, Verificación y Gestión de cambios).

Recolección: Recolección y documentación de requisitos es una actividad de comunicación iterativa entre clientes, gerentes y practicantes (stakeholders del proyecto), para descubrir, definir, refinar y registrar una representación precisa de los requisitos del producto. Varios métodos son utilizados para la recolección de requisitos. Algunos análisis iniciales como es la agrupación, categorización, priorización, son desarrollados durante esta actividad.

Documentación: Después que los requisitos han sido recolectados, hay que analizarlos a detalle y documentarlos en una especificación de requisitos. El resultado de la especificación de requisitos y de cualquier especificación de requisitos de componentes hardware/software derivado sirve como registro de convenio con el cliente y compromiso con el proveedor. Estas especificaciones son rastreadas utilizando una matriz de trazabilidad de requerimientos y son sujetos a verificación y gestión de cambio a través del ciclo de vida del producto.

Verificación: Una vez que la especificación de requisitos ha sido desarrollada, los requisitos son verificados. La verificación de requisitos es un proceso para asegurar que la especificación de requisitos del producto es una representación exacta de las necesidades del cliente. Este proceso también asegura que los requisitos sean trazados y verificados a través de varias fases del ciclo de vida; particularmente en el diseño, implementación y pruebas. Los requisitos deben ser trazados desde fuentes externas, tales como los clientes, para derivar requisitos del nivel del sistema, para especificar requisitos del producto hardware/software. Además, todos estos requerimientos deben ser trazados al diseño, implementación y pruebas para asegurarse que los requerimientos han sido satisfechos.

Gestión de Cambios: Gestión de cambios es un proceso formal para identificar, evaluar, trazar y reportar cambios propuestos y aprobados a la especificación del producto. Como el proyecto va evolucionando, los requerimientos pueden cambiar o expandirse para ajustar algunas modificaciones en el alcance o diseño del proyecto. Un proceso de gestión de cambios proporciona un rastreo completo y preciso de todos los cambios que son pertinentes al proyecto.

1.5.2 Descripción de las Actividades de la Gestión de Requisitos:

1.- Recolección de requisitos
  1.1.- Planeación de Elicitación
  1.2.- Formatos de representación
  1.3.- Análisis de requerimientos inicial
  1.4.- Especificación de requerimientos
2.- Documentación de requerimientos
  2.1.- Formatos de representación
  2.2.- Análisis de requerimientos detallado
  2.3.- Especificación de requerimientos
  2.4.- Matriz de trazabilidad
3.- Verificación de requerimientos
  3.1.- Verificación – una actividad del ciclo de vida
  3.2.- Revisión de la verificación
4.- Gestión de cambio de requerimientos
  4.1.- Proceso de control de cambio
  4.2.- Aceptación de la línea base

1.6 Trazabilidad de los requisitos

Con el objetivo de establecer las dependencias entre los requisitos aparece el término trazabilidad que no es más que la habilidad para describir y darle seguimiento a los requisitos a través de todo el proceso de desarrollo, hacia sus orígenes o hacia su implementación, y mostrando el impacto de un cambio a través de todas las especificaciones generadas durante el proceso de desarrollo de software. Es un proceso que se define para seguir todas las relaciones entre los elementos software (artefactos) que se van creando a partir de los requisitos.

En los casos de uso es posible la trazabilidad y aunque, puede ser llevada de manera manual, la misma dinámica de los procesos de desarrollo de software exigen contar con herramientas que permitan

automatizar la mayoría de los procesos operativos y que permitan al arquitecto realizar tareas más creativas en beneficio del usuario y de la organización.

1.6.1 Estrategias de trazabilidad

Existen diversas estrategias de trazabilidad que se pueden usar para facilitar el proceso de administración de requerimientos; incluso dentro del framework del RUP muchos acercamientos son posibles. Cinco de los más comunes son:

1.6.1.1 Solo Modelo de Casos de uso

En esta estrategia el modelo de casos de uso es la manifestación única de los requisitos del sistema. Los proyectos que escogen esta estrategia se caracterizan por la estrecha asociación y confianza que existe entre los clientes y los desarrolladores.

El modelo de casos de uso, el glosario y los requisitos adicionales forman la manifestación completa de los requisitos del sistema. Se usa normalmente en proyectos internos, de baja responsabilidad donde los desarrolladores esperan probar o ganar una clara comprensión de los requisitos desarrollando el modelo de casos de uso junto (o aprobados por) con los clientes.

Ventajas

Esta estrategia requiere de un conjunto mínimo de documentos y permite un esfuerzo mínimo en la administración de requisitos, brindando así un buen apoyo para el alcance de la administración de requisitos, en la que incide el análisis y el desarrollo incremental de la misma, y los casos de uso son fáciles de entender.

Desventajas

En esta estrategia no hay ninguna relación entre las necesidades de los involucrados (stakeholders). No se hace ningún esfuerzo real para analizar el problema antes de comenzar la definición de la solución. Si se llevan a cabo versiones (releases) regulares puede resultar difícil administrar el producto y las expectativas de los involucrados sin alguna información en un nivel más alto que los mismos casos de uso. Sin comprender cualquier necesidad puede ser difícil saber cuando el modelo de casos de uso describe una solución satisfactoria.

Entre las principales características de esta estrategia están las siguientes:

Trazabilidad explícita: No se requiere trazabilidad explicita. La trazabilidad implícita que se provee adoptando una estrategia basada en casos de uso se considera suficiente. Probablemente ninguna trazabilidad explícita se mantiene, así que no se usa ninguna herramienta de administración de requisitos.

Integridad: Aunque un modelo de casos de uso facilita establecer la integridad de las especificaciones de los requisitos del software la falta de trazabilidad de las necesidades de los involucrados a menudo guían a la producción de un sistema incompleto o sobre elaborado.

Entendimiento: El modelo de casos de uso es fácil de entender para todos los involucrados en el proyecto.

1.6.1.2 El modelo de Casos de uso define la Característica (Feature) del producto

Se usa el modelado de los Casos de uso como el principal método para sacar los requisitos y el Modelo de los Casos de uso llega a ser la definición de las características del producto que proveerá el sistema así como la declaración de los requisitos de software.

Esta estrategia es solo conveniente para pequeños desarrollos, con ciclos de vida cortos. Mientras el sistema evoluciona, es cada vez menos probable que las nuevas características de cada versión tomen la forma de nuevos casos de uso.

Ventajas

En este caso el modelo de casos de uso se relaciona con las necesidades de los involucrados, los cuales ayudan a evaluar la conveniencia del modelo de casos de uso.

Desventajas

Los casos de uso pueden aparecer para definir las características del sistema en las primeras etapas del proyecto, pero los dos conceptos discreparán mientras el proyecto madura.

Los casos de uso no son características (lo que parece ser una estrategia de ahorro de tiempo, puede convertirse en poco tiempo en un desorden difícil de mantener).

Entre las principales características de esta estrategia están las siguientes:

Trazabilidad explícita: Como Solo modelo de casos de uso. Solo habrá un pequeño conjunto de necesidades y este pequeño conjunto de trazabilidad adicional aún resulta que se requieran en pequeñas cantidades de trazabilidad explícita.

Integridad: El modelo de casos de uso facilita establecer la integridad de la especificación de la trazabilidad adicional de los requerimientos de software que apoyan a las necesidades de los involucrados y ayudan a validar la aplicabilidad del modelo de casos de uso.

Entendimiento: Las necesidades y el modelo de casos de uso son fáciles de entender para todos los involucrados en el proyecto.

1.6.1.3 Las características guían al modelo de casos de uso

El modelo de casos de uso y las especificaciones adicionales forman una completa especificación de los requisitos del software. En este caso el modelo de casos de uso se comporta como la principal manifestación de los requisitos del software. Esto está complementado por una especificación adicional que contienen los requisitos del software que no se pueden expresar fácilmente en los mismos casos de uso. El modelo de casos de uso y los requisitos adicionales se complementan con las necesidades y las características del producto además del glosario. Definiendo así una política de trazabilidad. Esta estrategia es aplicable a todos los proyectos donde los casos de uso se aceptan como un formato apropiado para la expresión de la mayoría de los requisitos de software.

Ventajas

Esta estrategia es fácil de aplicar. Permite expresar en una forma fácil de entender los requisitos del software. Posee una capacitación detallada y una trazabilidad formal. En esta estrategia el impacto de análisis de los cambios de requisitos se facilita por esta estrategia de trazabilidad, el impacto de no llevar a cabo una característica o una sección de caso de uso puede entenderse claramente. Además de esto se minimiza el esfuerzo en la dirección de requisitos.

Desventajas:

Esta estrategia no es aceptable en todas las organizaciones. Algunas personas consideran difícil escribir un contrato basado en requisitos del software expresados principalmente como un modelo de casos de uso aunque muchas organizaciones han logrado esto con éxito.

Entre las principales características de esta estrategia están las siguientes

Trazabilidad explícita: Además de la trazabilidad implícita del modelo de casos de uso, en este caso, tenemos que mantener la trazabilidad explícitamente entre las necesidades, los requerimientos y el modelo de casos de uso.

Integridad: Teniendo a ambos, las características y la perspectiva de los casos de uso en los requisitos del software posibilita alcanzar un alto nivel de integridad con respecto a la captura y prioridad de requisitos.

Enfoque: La suma de necesidades y características al modelo de casos de uso aumentan el enfoque de la actividad de requisitos para abarcar mas activamente los administradores del producto y todos los demás involucrados, así como los usuarios. Las características son una herramienta muy poderosa para dirigir las expectativas de los usuarios y proveer una gran cantidad a las perspectivas de los casos de uso de los requisitos del software.

Entendimiento: La definición de necesidades y las características junto al modelo de casos de uso con especificaciones adicionales provee un modelo de requisitos que es fácilmente entendible por todos los involucrados en el proyecto.

1.6.1.4 El modelo de Casos de uso es una interpretación de la especificación de requerimientos de software

En esta estrategia el modelo de casos de uso es una interpretación de una formal y tradicional especificación de los requisitos del software. Las características se analizan en el documento de especificación de requisitos del software (como en una administración formal de requisitos) pero entonces los requisitos del software se analizan explícitamente en el modelo de casos de uso. Esta estrategia es útil para compañías productoras de software que usan técnicas de desarrollo basadas en caso de uso que dan una especificación de requisitos de software tradicional como parte de su contrato.

Ventajas

Esta estrategia habilita una trazabilidad nivelada y detallada, en la misma se expresan los requisitos de software en una forma más fácil de entender. En este caso el modelo de casos de uso está finalmente relacionado recíprocamente a las necesidades de los involucrados por vía de los requisitos de software y

las características del producto que ayudan a los involucrados a evaluar la conveniencia del modelo de casos de uso. Los requisitos individuales tienen el contexto proporcionado por los casos de uso y/o las características del producto.

Desventajas

La estrategia debe mantener un conjunto de documentos muy grande.

Esta estrategia implica un costo y un mantenimiento muy alto.

1.6.1.5 El Modelo de Casos de uso reconcilia múltiples requisitos de software tradicionales

Ésta estrategia es una variación del modelo de casos de uso, es una interpretación de la especificación de requisitos del software excepto en este caso que hay SRS tradicionales proporcionadas por conjuntos múltiples e independientes de involucrados y proporciona la especificación de un sencillo sistema común. Esta situación se levanta a menudo para Casas de software que están desarrollando una sola aplicación para satisfacer los requisitos de muchos clientes desconectados, independientes y diferentes. En este caso cada involucrado tiene su propio conjunto de características del producto y requisitos del software, los cuales se detallan al alcance de sus propios documentos Visión y de SRS. Esta estrategia puede ser muy efectiva cuando se procede con un gran conjunto de involucrados independientes. Mediante la investigación de requisitos individuales y el Modelo de Casos de uso se les permite a los diseñadores evaluar cuan bien está evaluando las necesidades de varios involucrados.

Ventajas y desventajas:

Esta estrategia es una variación de la anterior (el modelo de casos de uso es una interpretación de la especificación de requisitos de software), y tiene las mismas ventajas y desventajas.

El beneficio de esta estrategia que la diferencia de las otras es su habilidad de lidiar y conservar los puntos de vista de los involucrados independientes en las planillas de sus propias especificaciones de requisitos de software formales e individuales.

También tiene la desventaja adicional de generar un documento aun más grande para el mantenimiento y el análisis.

1.7 Planificación y Estimación

La Planificación y la Estimación de un software es lo primero que se realiza para la modelación del mismo, la cuál da una visión global de la aplicación. Aunque estas no son lo suficientemente exactas, ayudan a organizar y planificar el trabajo individual y el del equipo. La Planificación implica la Estimación, su intento por determinar esfuerzo humano requerido, la duración cronológica del proyecto y el costo. [8]

La planeación efectiva de un proyecto de software depende de la planeación detallada de su avance, anticipando problemas que puedan surgir y preparando con anticipación soluciones tentativas a ellos. El administrador del proyecto es responsable de la planeación desde la definición de requisitos hasta la entrega del sistema terminado. [8]

1.7.1 Estrategias para la planificación

Una mejor práctica para planificar los proyectos es establecer los planes en los requisitos del proyecto. Para desarrollar un plan inicial, los requisitos no necesitan estar completos, pero se debe tener por lo menos una idea bastante clara acerca del alcance del proyecto. La planificación basada en requisitos se puede realizar siguiendo tres tipos de estrategias:

- ✓ Dirigido por Casos de uso.
- ✓ Dirigido por requerimientos.
- ✓ Dirigido por la interfaz de usuario.

1.7.1.1 Dirigido por Casos de uso

Esta estrategia es promovida por el RUP en la cual se organiza el proyecto en iteraciones donde se desarrolla totalmente y potencialmente despliega (por lo menos internamente) una porción del sistema. Siguiendo esta estrategia, se asignan uno o más casos de uso a cada iteración. Esta estrategia tiene varias ventajas y desventajas.

Ventajas

- Se desarrollan valiosas y coherentes porciones del sistema (Casos de uso), proporcionando el valor del negocio.
- Facilita la comprensión del plan construido con la información brindada por los usuarios.

- Los diseñadores aprenden el negocio mientras están trabajando con los casos de uso, ya que los casos de uso describen la funcionalidad fundamental del negocio.

Desventajas

- Los Casos de uso no son una definición completa de los requisitos, de modo que se necesita tener en cuenta esto al definir las iteraciones.
- La mayoría de los Casos de uso requieren el desarrollo de fin a fin (interfaz de usuario, las clases del negocio, base de datos), exigiendo todos los aspectos del sistema para evolucionar simultáneamente, se requiere más personas o más generalizaciones en el equipo.
- Todos los Casos de uso no se crean igual (algunos son significativamente mas complejos que otros), mientras que se produzcan iteraciones de tamaños variables o números diferentes de Casos de uso asignado a cada iteración.
- La mayor dirección puede tener dificultad para aceptar la estrategia si ellos no están familiarizados con los Casos de uso.

1.7.1.2 Dirigido por requerimientos

Con esta estrategia, los requerimientos individuales como el control de seguridad de acceso o la manipulación de los clientes basados en la información se asignan a las iteraciones individuales.

Ventajas

- Los requerimientos están típicamente mejor formados que los Casos de uso que facilitan asignar las cantidades de funcionalidad a cada iteración.
- A los diseñadores les gusta porque no requiere el conocimiento significante de cómo se usa el sistema.
- A la dirección le gusta porque es más largo que las líneas de planificación tradicional.
- Negarse es más fácil porque pueden desarrollarse los rasgos comunes temprano en el proyecto.

Desventajas

- La principal desventaja de esta estrategia es que es mucho más difícil garantizar que se asigne un valor a los usuarios en el orden que ellos lo necesitan. Los usuarios necesitan que se les asignen las

colecciones de requerimientos para obtener el valor del negocio, y se necesita algo (como Casos de uso) que una esa funcionalidad.

1.7.1.3 Dirigido por la interfaz de usuario

Con esta estrategia, los elementos individuales de la interfaz de usuario, las páginas HTML, pantallas e informes se asignan a las iteraciones.

Ventajas

- Se desarrollan valiosas, coherentes porciones del sistema (elementos de interfaz de usuario), mientras se proporciona el valor del negocio a los usuarios.
- A los diseñadores les gusta porque no requiere el conocimiento significante de cómo se usa el sistema.
- A la dirección le gusta porque es más largo que las líneas de planificación tradicional.

Desventajas

- Muchos elementos de interfaces de usuario requieren interacción con varias clases de negocio, qué a su vez actúa recíprocamente con varios aspectos de su base de datos, exigiendo todos los aspectos de su sistema para evolucionar simultáneamente.
- Todos los elementos de interfaz de usuario no se crean igual (algunos son significativamente mas complejos que otros), mientras produce iteraciones de tamaños variantes o los números diferentes de elementos de la interfaz de usuario que se asignan a cada iteración.
- Negarse es difícil porque normalmente entre los elementos de la interfaz de usuario no puede reconocerse tarde hasta en el proyecto.

1.8 Pruebas por casos de uso

El principal objetivo del flujo de prueba es valuar la calidad del producto que se desarrolla a través de las diferentes fases por las cuales este pasa. Verifica que el producto funcione como se diseñó y que los requerimientos son satisfechos completamente. Brinda soporte a los flujos de trabajos, con el objetivo de encontrar y documentar defectos en la calidad del software, notifica la calidad percibida del software, provee un medio de validación para las suposiciones hechas en el diseño y especificaciones de requerimientos por medio de demostración concreta y valida las funciones del producto de software según

lo diseñado. Es recomendable que esté presente en todo el ciclo de vida del desarrollo del sistema para ir refinándolo y no esperar al final del mismo.

En el mundo, es muy habitual desarrollar pruebas del sistema a partir de los casos de uso[9], pero es imprescindible garantizar que estén correctamente implementados para lograr el éxito.

Las pruebas generadas del sistema consisten en una explosión combinatoria, para identificar todos los caminos de ejecución posibles y generar una prueba por cada camino. Dichas pruebas se completan con datos de prueba, el resultado esperado, los actores participantes y las precondiciones y poscondiciones de cada requisito funcional.

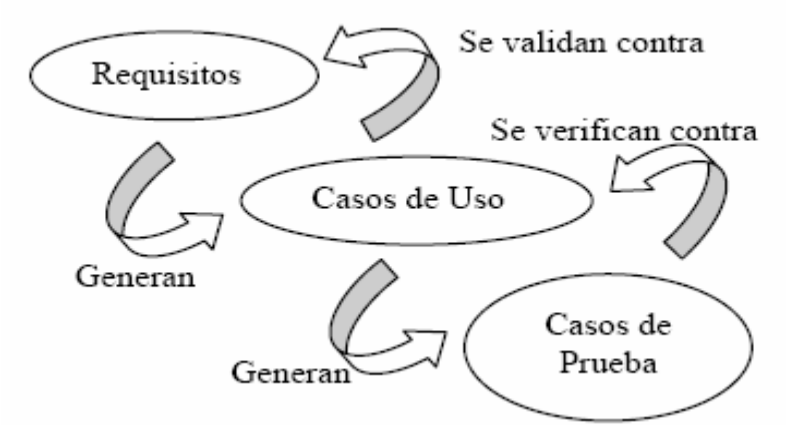


Fig1. Relación entre Requisitos, Casos de Uso y Casos de Prueba.

Según Evans en el año 2002 planteo que los casos de uso deben probarse contra los requisitos, constituyendo este proceso la validación del sistema, mientras que son los propios casos de uso los que deben utilizarse como autoridad de prueba para probar el código, es decir para realizar la verificación del sistema (figura 1).

1.9 Metodologías de desarrollo de software

En el mundo de la informática todo desarrollo de software es riesgoso y difícil de controlar, pero sino se usa una metodología adecuada, lo que se obtiene es un producto sin calidad y esto trae como consecuencia que los clientes queden insatisfechos con el mismo.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software, indican paso a paso todas las actividades a realizar para lograr el producto informático deseado, mostrando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. Tiene como objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo, por medio de una mayor transparencia y control sobre el proceso; "producir lo esperado en el tiempo esperado y con el coste esperado".[10]

Existen varias metodologías de desarrollo de software, de las cuales se pueden mencionar la Extreme Programming (XP), Microsoft Solution Framework (MSF), Feature Driven Development (FDD) y Rational Unified Process (RUP).

1.9.1 RUP (Proceso Unificado de Software)

El Proceso Unificado de Rational es un proceso de ingeniería del software, el cual está definido y estructurado. Es un proceso que proporciona adaptabilidad en dependencia de las necesidades y características que requiera cada proyecto específico. Al ser una de las metodologías más generales en la actualidad tiene como objetivo asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles. Este Proceso Unificado de Desarrollo de Software está definido por tres características fundamentales: "dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental."[11] Está dirigido por casos de uso ya que el desarrollo de este es a través de los flujos de trabajo generados por los casos de uso. Los casos de uso se especifican y se diseñan en el inicio de cada iteración, son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba y describen la funcionalidad total del sistema. La estrategia que propone es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

RUP se repite a lo largo de una serie de ciclos que contienen los procesos, las actividades y las tareas involucradas en el desarrollo y el mantenimiento de un producto software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso. “Cada ciclo concluye con una versión del producto para los clientes”.[11] El ciclo de vida del software comprende cuatro grandes fases: concepción o inicio, elaboración, construcción y transición. En la fase de inicio el objetivo es determinar la visión del proyecto, en la fase de elaboración el objetivo es determinar la arquitectura óptima, en la fase de construcción el objetivo es llevar a obtener la capacidad operacional inicial y por último en la fase de transición el objetivo es llegar a obtener el release del proyecto.

Para el desarrollo de RUP tenemos en cuenta 9 flujos de trabajos, flujo de trabajo no es más que una secuencia de actividades para producir determinados artefactos que aumentarán el desarrollo del proyecto. Dentro de los flujos de trabajo se encuentran el modelo de negocio, requerimientos, análisis y diseño, implementación, pruebas, la configuración y administración del cambio, administrando el proyecto, ambiente y distribución.

Capítulo 2 Estudio Preliminar

Introducción

En este capitulo se abordará la situación actual del uso y aplicación de técnicas enfocadas a los casos de uso en los proyectos productivos en la facultad, específicamente aquellos que utilizan la metodología RUP. Para justificar dicho levantamiento se realizó una encuesta a los proyectos, con el fin de tener la información actual de cómo están realizando el proceso de desarrollo de sus productos y de ahí se realizará un análisis de la información recopilada.

2.1 Situación actual de los proyectos de la facultad

La facultad desde sus inicios se ha caracterizado por tener un alto nivel de compromiso y responsabilidad con la patria y con las tareas orientadas por la universidad. Una de las principales tareas asignadas fue el desarrollo de diversos software, como por ejemplo: la Versión de Auto-teórico para República Dominicana, el Simulador Práctico para Guatemala, A jugar (MINED-Primaria) y un Demo del Simulador de Tiro, de los cuales se obtuvieron experiencias positivas y negativas .

Por la importancia que requiere el desarrollo de estos software y el auge que han alcanzado hasta el momento, nos hemos visto en la necesidad de hacer un estudio profundo en los proyectos de la facultad que utilizan la metodología RUP con el objetivo de detectar las principales deficiencias que estos presentan en el desarrollo de sus productos, por lo que se realizó una encuesta a estos proyectos, recogiendo la información necesaria que se muestra a continuación **(Anexo 1).**

Luego se exponen los problemas detectados, los cuales fueron obtenidos de las respuestas que dio el responsable de cada proyecto en las encuestas aplicadas a los mismos **(Anexo 2).**

1- Deficiencia en la captura de requisitos del software.

La mayoría de los proyectos de la facultad no realizan la captura de requisitos del software, y los que la hacen lo realizan de forma incorrecta. No se definen ni se usan los atributos de los requisitos. No se sigue la trazabilidad de un requisito y todos los artefactos generados a su alrededor a lo largo del proceso de desarrollo.

2- Deficiencias en la identificación de los casos de uso.

En los proyectos de la facultad no se identifican los casos de uso. No se guían por la plantilla de RUP que asigna la Universidad.

3- Deficiencias en el nivel de detalle de especificación de los casos de uso.

Los proyectos de la facultad no usan en el momento preciso los niveles de detalle. No priorizan los casos de uso mas importante y de mayor influencia para el desarrollo de los software.

4- Poco aprovechamiento de los casos de uso.

No todos los casos de uso que identifican en los proyectos de la facultad se le dan seguimiento a lo largo del proceso de desarrollo del software.

5- Deficiencias en la estimación de los proyectos.

La mayoría de los proyectos de la facultad no aplican ninguna técnica basada en los casos de uso para estimar el mismo. Para estimar el proyecto se basan en experiencias obtenidas anteriormente en otros productos desarrollados.

6- Los casos de uso no aportan a la planificación de las iteraciones.

No se realiza un cronograma de trabajo al inicio de cada proyecto. Proyectos con plazos de entrega muy cortos. No se fijan plazos de cumplimiento de las tareas. Debido a las pobres especificaciones de los casos de uso, los mismos aportan muy poco para estructurar las iteraciones.

7- Los casos de uso no aportan a las pruebas del software.

Al no identificar y especificar los casos de uso en los proyectos estos no saben cuanto puede aportarle a las pruebas. No se elaboran casos de prueba a partir de ellos.

Para hacer un análisis más profundo de los problemas detectados en los proyectos productivos de la facultad se hizo necesario escoger los problemas que más afectan el proceso de desarrollo de software, los cuáles se mencionan a continuación:

2.2 Principales problemas detectados en los proyectos encuestados

1. Deficiencias en la captura de requisitos.
2. Deficiencias en la especificación de los casos de uso.
3. Deficiencias en la estimación del proyecto.
4. Deficiencias en la planificación del proyecto.
5. Deficiencias en la realización de las pruebas a partir de los casos de uso.

Estos problemas traen consigo una demora en la entrega del producto, provocando que para cumplir con el tiempo de desarrollo que le impuso el cliente, se haga todo en muy poco tiempo, con lo cual el producto final no tendrá la calidad requerida.

A continuación se muestra una tabla resumen de los resultados de los principales problemas encontrados en los proyectos encuestados:

Parámetros de la encuesta	Auto-Teórico		Simulador de Tiro		Juegos Virtuales		Herramienta		Scada		Emedia		Quirúrgico	
	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Capturan requisitos de software.		X		X	X			X	X		X		X	
Especifican los requisitos en forma de casos de uso.	X			X	X			X	X		X		X	
Aplican alguna técnica basada en los casos de uso		X		X		X		X		X		X		X

para la estimación del proyecto.														
Consideran que los casos de uso tienen influencia en la planificación del proyecto.		X		X	X			X		X	X			X
Realizan pruebas a partir de los casos de uso.		X		X		X		X		X	X			X

2.3 Determinación y análisis de la muestra

Definiendo que:

Población, también llamada universo o colectivo es el conjunto de elementos de referencia sobre el que se realizan las observaciones. [12]

La *muestra* es un subconjunto de casos o individuos de una población. [13]

En el caso de nuestra investigación se utilizó un muestreo aleatorio simple (M.A.S) donde cada elemento de la población tiene igual probabilidad de ser seleccionado.

La población está compuesta por los 11 proyectos de la facultad que utilizan la metodología RUP en el desarrollo de sus productos. Estos proyectos fueron seleccionados por las experiencias adquiridas en la realización de software, los cuáles fueron asignados por parte de la universidad, donde algunos fueron de exportación para otros países.

La muestra seleccionada fue de 7 proyectos que representa el 63,63% del total lo cual es una buena cantidad para el tamaño de muestra para cualquier propósito, de los 7 proyectos seleccionados utilizando M.A.S 2 son investigativos y 5 productivos. Dentro de los investigativos se encuentran Herramientas y

Emedia y dentro de los productivos está Auto-teórico, Simulador de tiro, Quirúrgico, Juegos virtuales y Scada.

Para probar estadísticamente la investigación realizada en los proyectos de la facultad 5 que utilizan la metodología RUP, es necesario aplicar un método óptimo que nos demuestre el nivel de dificultad de los problemas existentes en estos proyectos.

2.3.1 Calculo de la proporción

Para conocer el por ciento de los proyectos que no cumplen cada parámetro medido en la encuesta realizada se calcula la proporción de los resultados positivos.

$p = \frac{r}{n}$ donde r es los resultados positivos, es decir la cantidad de proyectos que cumplen el parámetro analizado y n es el tamaño de la muestra, es decir la cantidad de proyectos encuestados.

Captura de requisitos

r = 4, n = 7

$p = \frac{4}{7} = 0.57$

Especificación de requisitos en forma de casos de uso

r = 5, n = 7

$p = \frac{5}{7} = 0.71$

Estimación de alguna técnica basada en los casos de uso

r = 0, n = 7

$p = \frac{0}{7} = 0$

Influencia de los casos de uso en la planificación de los proyectos

r = 2, n = 7

$p = \frac{2}{7} = 0.29$

Realización de las pruebas a partir de los casos de uso

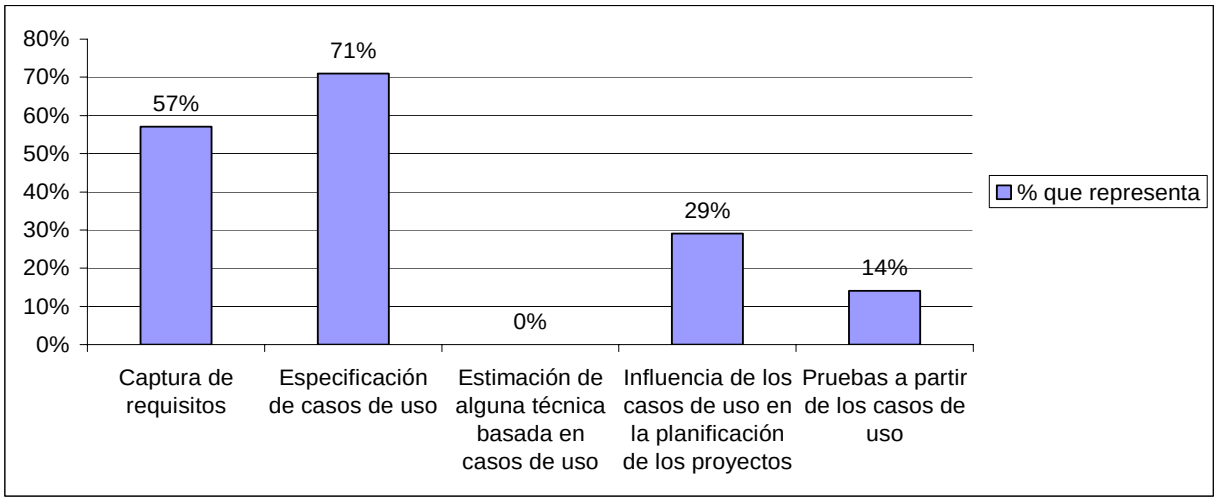
r =1, n = 7

$p = \frac{1}{7} = 0.14$

A continuación se muestra una tabla resumen con el por ciento que representa el cumplimiento de los parámetros medidos en la encuesta aplicada a los proyectos de la facultad 5:

Parámetros medidos en la encuesta.	Por ciento que representa el cumplimiento de los parámetros medidos en la encuesta de los proyectos de la facultad 5.
Capturan requisitos de software.	57%
Especifican los requisitos en forma de casos de uso.	71%
Aplican alguna técnica basada en los casos de uso para la estimación del proyecto.	0%
Consideran que los casos de uso tienen influencia en la planificación del proyecto.	29%
Realizan pruebas a partir de los casos de uso.	14%

2.3.2 Representación gráfica de los resultados obtenidos de la información recopilada.



De los 7 proyectos encuestados de la facultad que utilizan la metodología RUP solamente 4 realizan la captura de requisitos, lo cual representa un 57%, 5 especifican los requisitos en forma de casos de uso, lo cual representa un 71%, ninguno aplica una técnica basada en los casos de uso para la estimación del proyecto, lo cual representa un 0%, 2 consideran que los casos de uso tienen influencia en la planificación del proyecto, lo cual representa un 29%, 1 realiza pruebas al software a partir de los casos de uso, lo cual representa un 14%.

Con esto podemos afirmar que en los proyectos de la facultad hay problemas en cuanto a los parámetros anteriormente mencionados, esto nos demuestra que los software no se han desarrollado de la mejor forma, es decir ningún proyecto cumple al máximo % estos parámetros, que en realidad lo idóneo sería que los cumpliera al 100% para que el producto al finalizar este proceso tenga la calidad requerida.

Estos porcientos calculados en la muestra son estimadores de los porcientos de la población completa. Para ilustrar este resultado de una forma más general y con un error deseado hallaremos los intervalos de confianza de las diferentes proporciones para la población total con un 95% de probabilidad.

2.3.3 Cálculo de Intervalos de Confianza para la proporción de la población

Este intervalo será hallado mediante el método Wilson (1927), conocido como método score, el cual posee buenas propiedades para el análisis, es el más óptimo y es el que mejor trabaja cuando se tienen proporciones igual a cero y a uno como es en el caso de nuestra investigación, ya que tenemos un parámetro con proporción igual a cero. No decidimos usar el método Wald ya que aunque sea sencillo su cálculo en el caso que las proporciones sean 0 o 1 el no brinda un intervalo tan ilustrativo como esperamos. Y más que no estamos calculando el error que comete.

Para conocer el intervalo de confianza para cada parámetro se debe resolver las ecuaciones siguientes, donde r representa la cantidad de proyectos que cumplen el parámetro analizado y n es la cantidad de proyectos encuestados. Para mayor información sobre este método referirse a [14].

De esta manera, con una probabilidad de 95% y z = 1.96, se pueden calcular los componentes de las ecuaciones de la (1) a la (4) para cada parámetro.

(1) $A = 2r + Z^2$

(2) $B = Z\sqrt{Z^2 + 4r(1 - \frac{r}{n})}$

(3) $C = 2(n + Z^2)$

Luego, el intervalo de confianza está dado por:

(4) $(A \pm B) / C$

Captura de requisitos

r = 4, n = 7

$A = 2*4 + (1.96)^2 = 11,8416$

$B = 1.96\sqrt{(1.96)^2 + 4*4(1-\frac{4}{7})} = 6,4111$

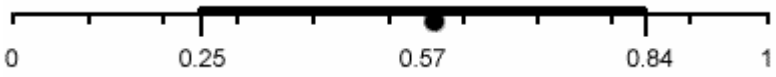
$C = 2\,[7 + (1.96)^2] = 21,6832$

$(A \pm B) / C$

((11,8416 - 6,4111) / 21,6832; (11,8416 + 6,4111) / 21,6832)

(0.25, 0.84)

Para este parámetro se puede afirmar con un 95% de probabilidad de que de los 11 proyectos el cumplimiento de este parámetro está entre un 25% y un 84%.



Especificación de requisitos en forma de casos de uso

r = 5, n = 7

$A = 2*5 + (1.96)^2 = 13,8416$

$B = 1.96\sqrt{(1.96)^2 + 4*5(1-\frac{5}{7})} = 6,059$

$C = 2\,[7 + (1.96)^2] = 21,6832$

((13,8416-6,059/21,6832) , (13,8416+6,059/21,6832))

(0.36, 0.92)

Para este parámetro se puede afirmar con un 95% de probabilidad de que de los 11 proyectos el cumplimiento de este parámetro está entre un 36% y un 92%.

Estimación de alguna técnica basada en los casos de uso

r = 0, n = 7

A = 2*0 + $(1.96)^2$ = 3,8416

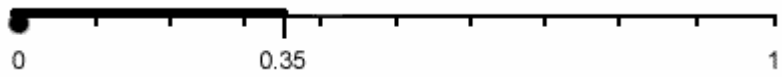
B = 1.96 $\sqrt{(1.96)^2 + 4*0(1-\frac{0}{7})}$ = 3,8416

C = 2 [7 + $(1.96)^2$] = 21,6832

((3,8416 - 3,8416/21,6832) , (3,8416 + 3,8416/21,6832))

(0.00, 0.35)

Para este parámetro se puede afirmar con un 95% de probabilidad de que de los 11 proyectos el cumplimiento de este parámetro está entre un 0% y un 35%.



Influencia de los casos de uso en la planificación de los proyectos

r = 2, n = 7

A = 2*2 + $(1.96)^2$ = 7,8416

B = 1.96 $\sqrt{(1.96)^2 + 4*2(1-\frac{2}{7})}$ = 6,0589

C = 2 [7 + $(1.96)^2$] = 21,6832

((7,8416 - 6,0589/21,6832) , (7,8416 + 6,0589/21,6832))

(0.082, 0.64)

Para este parámetro se puede afirmar con un 95% de probabilidad de que de los 11 proyectos el cumplimiento de este parámetro está entre un 8,2% y un 64%.

Realización de las pruebas a partir de los casos de uso

r =1, n = 7

A = 2*1 + $(1.96)^2$ = 5,8416

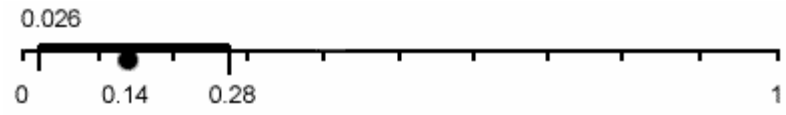
B = 1.96 $\sqrt{(1.96)^2 + 4*1(1-\frac{1}{7})}$ = 5,2847

C = 2 [7 + $(1.96)^2$] = 21,6832

((5,8416 - 5,2847/21,6832) , (5,8416 - 5,2847/21,6832))

(0.026, 0.28)

Para este parámetro se puede afirmar con un 95% de probabilidad de que de los 11 proyectos el cumplimiento de este parámetro está entre un 2,6% y un 28%.



Hallando una media de las 5 proporciones muestrales se obtiene la proporción media de cumplimiento de los 5 parámetros, la cual es de 0,342.

Para los 11 proyectos con un intervalo de confianza para la media de la proporción de cumplimiento de los 5 parámetros es [0.058, 0.754]. El promedio de cumplimiento de los 5 parámetros esta en ese intervalo.

Capítulo 3 Propuesta de Solución

Introducción

En este capítulo se mostrará la propuesta de las técnicas que darán solución a los problemas detectados en el análisis realizado en los proyectos de la facultad. Las cuáles harán más eficiente el proceso de desarrollo del software, contribuyendo a la mejora de la calidad de los productos que se desarrollen en los proyectos de la facultad.

3.1 Propuesta de técnicas basadas en el desarrollo dirigido por casos de uso.

Basado en el estudio realizado en los proyectos de la facultad, donde se detectaron irregularidades en el proceso de desarrollo de software, lo cuál conllevó a que los productos desarrollados no hubieran alcanzado la calidad esperada, surgió la idea de proponer técnicas basadas en el desarrollo dirigido por casos de uso para perfeccionar el proceso de desarrollo del software y contribuir a la mejora de la calidad de los mismos. Se decidió proponer las técnicas basadas en casos de uso ya que en la mayoría de los proyectos tanto de la Universidad como los de la facultad, el proceso de desarrollo que utilizan son los dirigidos por caso de uso, específicamente RUP.

En la fase de inicio, específicamente en el flujo de trabajo Levantamiento de requisitos se propone que se aplique la técnica Storyboard para lograr entender las funcionalidades del sistema y así definir los requisitos. Luego se utiliza la técnica Taller de casos de uso con el objetivo de definir los actores, casos de usos y requisitos adicionales. Estas técnicas se describen a continuación:

3.1.1 Técnica “Storyboard”

Esta técnica brinda el punto de entrada a los posibles caminos de cada uno de los casos de uso. El propósito principal de esta técnica es entender el flujo global y las interacciones del sistema. Permite realizar la descripción lógica y conceptual de las funcionalidades del sistema para un escenario específico, e incluso la interacción requerida entre los usuarios y el sistema.

Proporciona importantes mecanismos de retroalimentación para descubrir requisitos inciertos o desconocidos. Explora, clarifica y captura la interacción prevista por el usuario como parte de adquisición de requisitos. Es utilizada para diseñar la interfaz de usuario y construir un prototipo de la misma. En las pruebas se usa para probar las características del sistema. Se usa para planear y llevar a cabo el análisis y diseño del trabajo.

3.1.2 Técnica “Taller de casos de uso”

El taller de casos de uso se comporta como una especialización de Tormenta de ideas. Es una reunión organizada donde las personas que participan deben tener un amplio conocimiento del proceso de desarrollo de software, es decir, el grupo contendrá personas con diferentes ideas y experiencias, debe ser pequeño (menos de diez personas), se divide en dos grupos, una mitad representando el equipo de desarrollo y la otra representando a los usuarios. En el medio de los dos grupos se encuentra el facilitador, quien debe hacer el papel de un moderador, un catalizador para todas las ideas y deseos. [5]

Las principales acciones a realizar en esta técnica son:

Definición de los actores

Se identifican los actores y se hace una pequeña descripción de cada uno, listando con viñetas los papeles que juegan con respecto al sistema y sus responsabilidades, muchos pueden identificarse directamente a través de sus posiciones regulares en la organización, pero hay que tener en cuenta que puede darse el caso que una posición en la organización podría corresponder a más de un rol para el sistema por lo que hay que definir bien claro quienes son los actores y los usuarios del sistema para luego quitar a los que no lo son y a los duplicados (otro actor juega el mismo rol), así sucesivamente se irán definiendo los verdaderos actores del sistema. El nombre dado a los actores debe reflejar el rol que desempeña.

Definición de los casos de uso

Entre más interrogantes se tenga sobre lo que quiere el cliente que haga su sistema, se obtendrá una mayor información del mismo, y así los desarrolladores podrán definir con más claridad los casos de uso. Para facilitar el entendimiento y definición del sistema se hace un dibujo de una vista del mismo, que no es más que una abstracción del sistema. Mediante esta vista se muestra como el sistema trabajará, la cual ayuda a los casos de uso extenderse de un límite al otro del sistema, y esto apuntará implícitamente a diferentes estados del sistema.

Al analizar los estados pueden surgir más casos de uso, y al chequear si las diferentes comunicaciones se pueden eliminar o no, es que surgen los flujos alternativos en los casos de uso. Cada actor debe tener por lo menos un caso de uso. Si no es así, es porque el actor es un duplicado o porque el actor realmente no es un usuario directo del sistema.

Descripción de los casos de uso

Después que se hayan definido todos los posibles casos de uso se trabaja uno por uno para hacer una breve descripción de los mismos. Luego se hace la descripción paso a paso del flujo de eventos y se escriben en orden las diferentes acciones del mismo. En la descripción se enumeran los pasos para ayudar a entender el nivel de detalle del caso de uso. Durante la definición y descripción surgen varios problemas, de los cuales muchos de ellos no se resuelven hasta llegar al Análisis y Diseño, y otros necesitan ser resueltos rápidamente para que los especificadores de requisitos puedan detallar el flujo de eventos correctamente.

Capturar los requisitos adicionales

En el sistema habrá varios requisitos que no se reflejarán en ningún caso de uso. Cuando se esta analizando la información (lo que hará el sistema) brindada por lo clientes, por lo general existirán partes de esta información que no estarán claras, por lo que se van almacenando en un documento aparte, esto servirá de base para la identificación de los requisitos adicionales.

Tracear los requisitos para Casos de uso

Se analizan los requisitos importantes de los involucrados (stakeholders) y cada característica en el documento Visión, verificando si están recogidos en el modelo de casos de uso y se discute cuales necesidades de los usuarios son rastreadas a los casos de uso. Leyendo uno por uno los requisitos del documento Visión, se va identificando a que caso de uso pertenece y se registra en una pequeña nota.

Siempre van a existir varios requisitos que no estarán conectados a ningún caso de uso, pueden ser requisitos específicos que se pospusieron para el diseño (diseño de requisitos), los que forman la lista de

los requisitos adicionales, los que fueron olvidados y luego requeridos en un caso de uso nuevo o alguna modificación a uno existente.

Además de aplicar estas técnicas es necesario determinar los atributos de los requisitos definidos en el proceso para saber que función cumple cada caso de uso, por lo que se propone que los atributos que no pueden dejar de precisarse son los siguientes.

3.1.3 Definición de Atributos

Atributo “estado”

El estado de los requisitos se establecen después de la negociación y revisión por el equipo de dirección de proyecto y su rastreo progresa durante la definición de la línea del proyecto. Los estados de los requisitos pueden clasificarse en:

- Propuestos: Los requisitos que están en discusión pero que no se han revisado y aceptado por la "vía oficial", como un grupo activo que consiste en representantes del equipo del proyecto, dirección del producto y usuario o comunidad de cliente.
- Aprobados: Los requisitos que se juzgan útiles, factibles y que han sido aceptados para la aplicación por la vía oficial.
- Incorporados: Los requisitos incorporados a tiempo en la línea del producto en un momento específico.

Atributo “prioridad”

La prioridad de los requisitos hace referencia al orden temporal en que debe realizarse ese requisito e indica en las fases que se incluirá cada uno. Este atributo es establecido por el líder del proyecto. Se clasifica en alto, medio y bajo.

Atributo “Autor” o “Fuente”

Este atributo permite conocer la procedencia de un requisito, de posibilitar el acceso a una mayor información sobre el requisito, que puede ser necesaria en caso de modificaciones o resolución de incompatibilidades. Podría indicar la fuente de donde se han inspirado para elaborar un requisito, de modo que esta información pueda serles útil más adelante si deben especificar el requisito.

Atributo “Esfuerzo”

Este atributo es establecido por el equipo de desarrollo. Algunos casos de uso requieren más tiempo y recursos que otros, estimando el número de equipo o persona por semanas, las líneas de código requeridas, por ejemplo, es la mejor manera de evaluar la complejidad y establecer las expectativas de lo que puede o no lograrse en un horario dado.

Atributo “Riesgo”

Establecido por el equipo de desarrollo basado en la probabilidad de que el caso de uso experimentará eventos indeseables, como los desbordamientos de esfuerzo, retrasos del horario o la cancelación. La mayoría de líderes de proyectos encuentran la categorización de riesgos como alto, el medio, y bajo. El riesgo puede evaluarse a menudo indirectamente midiendo la incertidumbre (el rango) de los horarios estimados de los proyectos de equipos. El riesgo puede ser:

- Alto: El impacto del riesgo combinado con la probabilidad de ocurrencia del riesgo es alto.
- Medio: El impacto del riesgo es menos severo y/o la probabilidad de ocurrencia del riesgo es menor.
- Bajo: El impacto del riesgo es mínimo y la probabilidad de ocurrencia del riesgo es baja.

Para facilitar el proceso de administración de requerimientos se propone aplicar la siguiente estrategia de trazabilidad:

3.1.4 Técnica de trazabilidad

✓ El modelo de Casos de uso es una interpretación de la especificación de requerimientos de software.

La organización de desarrollo se proporciona con una SRS tradicional, por el cliente, como el punto de partida para el desarrollo de su sistema.

El documento es un obligatorio o regulador entregable a principios del ciclo de vida del proyecto. Cada proyecto debe tener un documento de SRS tradicional y formal que exprese los requisitos del sistema en la misma forma que me puso que todos los demás proyectos.

En este caso el modelo de casos de uso se usa para modelar y reinterpretar todos los requisitos de software dentro del alcance del proyecto. Cuando se adopta esta estrategia es usual que se haga primero la especificación de los requisitos del software, hay otras técnicas disponibles para dar la información sostenida por el modelo de casos de uso en un formato que parece formal, la SRS tradicional (sobre todo

cuando se asume una estrategia donde los requerimientos guían al modelo de casos de uso) sin crear una segunda definición de requisitos de software.
Al adoptar esta estrategia no hay necesidad de que el conjunto de requisitos del software tradicionales sea una declaración completa de la funcionalidad requerida ya que el modelo de casos de uso suministra o asegura la integridad de la especificación funcional. Los requisitos del software tradicionales pueden usarse para capturar los requisitos del software identificados directamente o pueden capturarse por los involucrados (stakeholders).

Esta técnica proporciona diversas ventajas. Una de esta es que facilita el análisis de cambio de los requisitos. La presencia del modelo de casos de uso facilita identificar los subconjuntos significantes de requisitos, de esta forma facilita una entrega incremental del producto. Esta estrategia se usa en los casos de uso iniciales como una forma de los procesos de los requisitos paralelos (el proyecto está ejecutando la forma vieja y la nueva) o puede adoptarse para esconder el hecho que los diseñadores están usando los casos de uso.

Entre las principales características de esta estrategia están las siguientes:
- Esta estrategia requiere la trazabilidad explícita para mantener la SRS formal, más allá está la sobrecarga adicional de rastrear los requisitos tradicionales del software en el modelo de casos de uso.
- El complemento de la SRS tradicional con un modelo de casos de uso hace que esta estrategia tenga una integridad muy alta. En este caso el modelo de casos de uso asegurará una especificación completa de la funcionalidad de los sistemas. El conjunto de especificaciones de requisitos del software no necesita estar al mismo nivel de integridad.
- La adopción de esta estrategia se maneja por la necesidad de cumplir un contrato existente, expresada por una SRS tradicional, o encaja en una metodología de desarrollo existente que requiere un SRS como el contrato entre los diseñadores y los clientes.
- La producción de dos definiciones de requisitos de software puede estar confundida al principio pero el modelo de casos de uso como patrón de la definición de los requisitos de software debe guiar a una fácil y entendible declaración de los requisitos del sistema.

Luego se aplica la descripción detallada de los casos de uso definidos empleando el formato usecase.org.

3.1.5 Especificación de los casos de uso "Formato usecases.org".

Esta plantilla se puede representar de dos formas, en una o dos columnas, pero se recomienda que sea en 2 ya que es más compacto y fácil de modificar, posee una clara separación visual de la conversación (cliente, sistema…), lo que hace más sencillo identificar visualmente las diferentes partes en la misma.

Formato usecases.org

Caso de uso: Se refleja el nombre del caso de uso, el que cual tiene que ser claro y con solo leerlo de a entender lo que hará el caso de uso.

Actor principal: El actor principal que recurre a los servicios del sistema para cumplir un objetivo. Es el que inicia el caso de uso.

Personal involucrado e intereses: Se registran todo las personas que están involucradas al caso de uso y se describen los intereses de las mismas con el objetivo de conocer que es lo que va a hacer el caso de uso.

Precondiciones: En las precondiciones se establece lo que debe cumplirse antes de comenzar un escenario en el caso de uso, son condiciones que se asumen que son verdad. Implica un escenario de otro caso de uso que se ha completado con éxito.

Garantías de éxito (Poscondiciones): En las poscondiciones se establecen qué debe cumplirse cuando el caso de uso se completa con éxito -o bien el escenario principal de éxito o algún camino alternativo. La garantía debería satisfacer las necesidades de todo el personal involucrado.

Escenario principal de éxito (o Flujo Básico): Describe el camino de éxito que satisface los intereses del personal involucrado. A menudo, *no* incluye ninguna condición o bifurcación. Aunque no es incorrecto reflejarlo en esta sección, se recomienda ponerlo en la sección de Extensiones.

Extensiones (o Flujos Alternativos): Se emplea cuando se llega a una alternativa compleja de explicar, la cual puede ser larga como se necesite para entender los eventos asociados a esta alternativa. Indican todos los otros escenarios o bifurcaciones, tanto de éxito como de fracaso.

Requisitos especiales: Son los requisitos no funcionales, atributos de calidad o restricciones que se relacionan de manera específica con el caso de uso descrito.

Lista de tecnología y variaciones de datos: Se registran las restricciones técnicas impuestas por el personal involucrado con respecto a las tecnologías de entrada o salida de datos. También es útil registrar las variaciones en los datos que podrían capturarse en un paso particular.

Como ejemplo de la descripción de un caso de uso referirse al **(Anexo 3)**

Utilizando la descripción textual de los casos de uso se aplica el "Método de cálculo de Puntos de Casos de Uso" para obtener el tamaño del producto software y a partir del mismo generar las estimaciones del esfuerzo ayudando a organizar y planificar el trabajo individual y el del equipo. Para lograr una mejor planificación se propone usar la estrategia "Dirigido por Casos de Uso". A continuación se explica la técnica y la estrategia anteriormente mencionadas:

3.1.6 Técnica "Estimación por Puntos de Casos de Uso"

Este método se basa fundamentalmente en las cuentas realizadas a los casos de uso.

1. Cuantificación de características funcionales del Sistema:

El primer paso a desarrollar es la cuantificación de los requerimientos funcionales. Esto consiste en la extracción de información del modelo de caso de uso en su forma textual de acuerdo a una clasificación de Actores y Transacciones de los Casos de Uso.

Clasificación de Actores y obtención del **P**eso de **A**ctores **S**in **A**justar (PASA).

Se debe realizar un registro de todos los actores del sistema y deben ser clasificados como Simple, Promedio y Complejo, de acuerdo al siguiente criterio:

- ✓ Actor Simple: Se trata de otro sistema interactuando a través de una interfaz de programación definida y conocida (API).
- ✓ Actor Promedio: Es otro sistema interactuando a través de un protocolo (como TCP/IP).
- ✓ Actor Complejo: se trata de una persona interactuando con el sistema a través de una interfaz gráfica de usuario (GUI) o página Web.

Junto a la cuenta y clasificación de los actores se debe asociar un factor de peso de acuerdo a la siguiente tabla:

Tipo de actor	Descripción	Factor
Simple	Interfaz de programación de aplicaciones	1

Promedio	Interfaz de comunicación vía protocolo	2
Complejo	Interfaz gráfica de usuario	3

Finalmente, se cuentan los actores de acuerdo a su clasificación o grado de complejidad, multiplicando cada subtotal por su factor de complejidad y sumando cada producto obteniéndose el peso de los actores sin ajustar (PASA).

Clasificación de los Casos de Uso y obtención del **P**eso de **T**ransacciones **S**in **A**justar (PTSA)

Teniendo el modelo de casos de uso, cada uno de ellos debe clasificarse como Simple, Medio o Complejo, de acuerdo al número de transacciones descritas en el caso de uso, incluyendo los cursos de acción alternativos. La cuenta del número de transacciones puede ser hecha a través de la cuenta de los pasos descritos en el caso de uso en forma textual según el siguiente criterio:

- ✓ Casos de Uso Simple: contiene entre 1 a 3 Transacciones (o pasos).
- ✓ Casos de Uso Promedio: contiene entre 4 a 7 Transacciones.
- ✓ Casos de Uso Complejos: contiene más de 7 Transacciones.

Los factores de peso asociados a la clasificación son los siguientes:

Tipo de caso de uso	Descripción	Factor
Simple	de 1 a 3 transacciones	5
Promedio	de 4 a 7 transacciones	10
Complejo	más de 7 transacciones	15

Al igual que las clasificación de los actores las cuentas de las transacciones de los casos de uso se multiplican por los factores de complejidad y finalmente se suman los productos obteniéndose el peso de las transacciones sin ajustar (PTSA)

Obtención del Peso o **P**untos de **C**asos de **U**so **S**in **A**justar (PCUSA).

Es la suma del Peso de los Actores Sin ajustar más el Peso de las Transacciones Sin Ajustar, es decir se calcula mediante la ecuación siguiente:

PCUSA = PASA + PTSA

2. *Cuantificación de características no funcionales del Sistema:*

El método considera las características de complejidad técnica tomando en cuenta algunos requerimientos no funcionales como un factor de ajuste al sistema, y además, factores ambientales que se concentran en las características del equipo de desarrollo.

En ambos casos, se debe evaluar cada Factor multiplicado por un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante:

Clasificación de **F**actores de **C**omplejidad **T**écnica (FCT)

Se adjunta tabla con los factores de peso que incorporan la complejidad técnica del sistema y algunas características no funcionales, en este caso, en cada uno de los ítems se tomaron en cuenta factores de complejidad propios de sistemas desarrollados bajo orientación a objetos.

Factor	Descripción	Factor de peso
T1	Sistema Distribuido	2
T2	Rendimiento o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Complejidad de procedimiento interno	1
T5	Reusabilidad del código	1
T6	Facilidades de instalación	0,5
T7	Facilidad de uso	0,5
T8	Portabilidad	2
T9	Facilidades de cambio	1
T10	Concurrencia	1
T11	Características de seguridad	1
T12	Provee acceso directo a terceras partes	1

T13	Requerimientos de entrenamiento especial	1

Para obtener el factor final se debe multiplicar cada item (T1 a T13) por el grado de influencia sobre el sistema, es decir mediante la siguiente fórmula:

FCT = 0.6 + 0.01 x ∑ (Peso i x Valor i)

Clasificación de **F**actores **A**mbientales (FA)

Las características del equipo de desarrollo en cuanto a perfiles, experiencia y capacidad técnica son los factores que se contemplan en el Cálculo del Factor de Ambiente.

Factor	Descripción	Factor de Peso
F1	Conocimiento del proceso de desarrollo	1,5
F2	Experiencia en la aplicación	0,5
F3	Experiencia en Orientación a objetos	1
F4	Capacidad de liderazgo de los analistas	0,5
F5	Motivación	1
F6	Estabilidad de los requerimientos	2
F7	Personal a tiempo compartido	-1
F8	Dificultad de los lenguajes de programación	-1

El cálculo del mismo es similar al cálculo del Factor de Complejidad Técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5, se calcula mediante la siguiente Fórmula:

FA = 1.4 - 0.03 x ∑ (Peso i x Valor i)

Cálculo de **P**untos de **C**asos de **U**so **A**justados (PCUA)

Finalmente, se obtiene la siguiente fórmula que representa los puntos de casos de uso ajustados:

PCUA = PCUSA* FCT*FA

Cálculo del Esfuerzo de Desarrollo (E)

Una vez que se han obtenido los puntos de casos de uso ajustados se puede estimar el esfuerzo de desarrollo mediante la siguiente fórmula:

E = PCUA x FC

Donde:

FC: Factor de Conversión

PCUA: Puntos de Casos de Uso Ajustados

E: Esfuerzo estimado en horas-hombre

Gustav Karner (1993) originalmente sugirió que cada Punto de Casos de Uso requiere 20 horas-hombre. Posteriormente, surgieron otros refinamientos que proponen una granularidad algo más fina, según el siguiente criterio.

Para conocer el valor del factor de conversión se contabilizan cuántos factores de los que afectan al Factor de ambiente están:

- por debajo del valor medio (3), para los factores E1 a E6,
- por encima del valor medio (3), para los factores E7 y E8.

Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre, 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre y mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

Se debe tener en cuenta que éste método proporciona una estimación del esfuerzo contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso. Para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software, las cuales se muestran en la siguiente tabla:

Tipo de actividad	**Por ciento que representa**
Análisis	10%
Diseño	20%

Implementación	40%
Prueba	15%
Sobre Cargas (Otras actividades)	15%
Total	100%

Cálculo de Tiempo

Luego se calcula el tiempo de desarrollo del proyecto de software mediante la siguiente formula:

TDES (total) = E (total) / CH (hombres) donde:

TDES: Tiempo de Desarrollo

E: Esfuerzo

CH: Cantidad de Hombres

Como ejemplo de los cálculos anteriormente explicados referirse al **(Anexo 4).**

Para lograr que este método tenga efectividad se debe chequear que los analistas hayan hecho una buena descripción detallada de los casos de uso, si el mismo no se hace de una forma correcta repercutirá negativamente en los resultados de la realización de los cálculos explicados anteriormente, es decir, el resultado de los mismos no nos dará una buena medida de tamaño y estimación del software analizado.

3.1.7 Estrategia de planificación "Dirigido por Casos de uso"

Para realizar la planificación del proyecto se propone que se aplique la estrategia dirigido por casos de uso, en la cual se comienza priorizando los requisitos para llevar a cabo los de prioridad más alta, con el objetivo de minimizar el riesgo de que el proyecto sea cancelado debido a la falta de confianza. Se reduce el riesgo de construir un sistema equivocado ya que se puede obtener la retroalimentación de los usuarios para validar que lo que se está construyendo es correcto.

Brinda diversos beneficios, por ejemplo, desarrolla valiosos y coherentes casos de uso, permite a los diseñadores adquirir un alto entendimiento del negocio mientras están trabajando con los casos de uso, ya que estos describen la funcionalidad fundamental del negocio, facilita la comprensión del plan construido

con la información brindada por el usuario, ya que en los casos de uso se recoge de forma detallada y clara.

A partir de las descripciones de los casos de uso definidos en el sistema se propone usar la técnica "Pruebas tempranas" ya que permite generar pruebas ejecutables y mejorar la calidad de los requisitos detectando errores, omisiones, inconsistencias en los mismos.

3.1.8 Técnica "Pruebas tempranas"

Es diseñada específicamente para ser aplicada cuando el sistema aún no está construido o, incluso, diseñado. La principal ventaja que proporciona esta técnica es la corrección de los errores detectados en las etapas tempranas ya que los costes serían menores.

La generación de pruebas en etapas tempranas del desarrollo permite corregir los siguientes defectos:

- *Omisión de escenarios.*

Al realizar el análisis combinatorio de todos los posibles caminos de ejecución se localizan las interacciones (manejo de errores, escenarios alternativos, entre otros) no documentadas en los requisitos funcionales. De esta manera es que se detectan los escenarios omitidos.

- *Resultados sin escenarios o escenarios sin resultado*.

Para la elaboración de las pruebas no solo se necesita un escenario, sino su resultado, lo cual permite evaluar la ejecución satisfactoria o insatisfactoria y la correcta implantación en el sistema del mismo.

Al analizar todas las posibles combinaciones de caminos de ejecución se pueden detectar escenarios que no conduzcan a ningún resultado documentado y resultados esperados a los que no conduzca ningún escenario.

- *Nuevos requisitos funcionales*.

Los posibles escenarios encontrados con algún resultado esperado, es mejor expresarlo mediante la definición de nuevos requisitos funcionales.

- *Ambigüedad*.

Para definir casos de uso se emplea con mucha frecuencia tablas y plantillas en lenguaje natural, esto trae como consecuencia ambigüedad ya que las mismas palabras pueden tener significados diferentes según quien las lea. La aplicación de esta técnica permite detectar aquellos términos ambiguos o que puedan tener distintos significados.

A medida que aumenta la complejidad de los sistemas software y la demanda de calidad, es necesario realizar procesos y métodos mediante buenos casos de pruebas para verificar la correcta implementación de los requisitos funcionales. Sin embargo, muchas veces la fase de prueba del sistema es demasiado corta para diseñar un buen conjunto de pruebas, ejecutarlas y analizar sus resultados.

Un caso práctico de como la generación de pruebas permite detectar y corregir los errores descritos anteriormente, referirse al **(Anexo 5).**

Conclusiones

A partir de los problemas detectados en los proyectos de la facultad mediante la encuesta se recomienda a los líderes de proyectos o responsables del mismo aplicar la propuesta planteada anteriormente, cada responsable de proyecto analizará los resultados de la puesta en práctica y de esta forma retroalimentar y desarrollar las conclusiones de la investigación.

CONCLUSIONES GENERALES

La realización de este Trabajo de Diploma fue de un gran aporte a los autores por los conocimientos y resultados obtenidos, que fueron los esperados. Se cumplieron los objetivos propuestos de forma satisfactoria.

Se desarrolló un análisis en los proyectos de la facultad en cuanto al uso de la metodología RUP y las técnicas de casos de uso, donde se aplicó una encuesta a los líderes para detectar las irregularidades que presenta el proceso de desarrollo de software.

A partir de la información recopilada se llegó a la conclusión que los productos no se han desarrollado de la mejor forma, lo que trajo como consecuencia que al finalizar el proceso no alcanzaran la calidad esperada. Por lo que se plantea aplicar las técnicas propuestas anteriormente y que cada responsable de proyecto pueda analizar los resultados de la puesta en práctica y de esta forma retroalimentar y desarrollar conclusiones de la investigación.

Este trabajo servirá de referencia para futuros cursos, para generar nuevos temas de investigación y para paulatinamente aplicar los resultados en todos los proyectos que utilicen la metodología RUP.

RECOMENDACIONES

Se recomienda a todas las personas interesadas en la investigación del trabajo de diploma realizar un análisis más detallado en cuanto al comportamiento de los casos de uso en cada flujo de trabajo que propone RUP.

A los líderes de proyectos poner en práctica las técnicas propuestas, analizar los resultados de las mismas, y de esta forma retroalimentar todo el proceso de software.

REFERENCIAS BIBLIOGRÁFICAS

1. Fortién, J.D. (2006) *Mejora Continua: Reto y Necesidad*.
2. *La informatización en Cuba*. 2004 [cited 2006 13 de noviembre]; Available from: http://www.cubaminrex.cu/Sociedad_Informacion/Cuba_SI/Informatizacion.htm.
3. Ceria, S. *Un Método Práctico para Explorar Requerimientos*.
4. *Caso de uso*. 2007 [cited 2006 13 de noviembre]; Available from: http://es.wikipedia.org/wiki/Caso_de_uso.
5. IBM (2005) *Rational Unified Process*.
6. Landazuri, B.A.M., *Definición de Perfiles en Herramientas de Gestión de Requisitos*, in *Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software*, Facultad de Informática, Universidad Politécnica de Madrid.
7. Gracia, J. *CMM-CMMI: Gestión de Requisitos o Requirimientos*. 2005 [cited 2007 12 de marzo]; Available from: http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php.
8. Giraldo, O.P. *Métricas, Estimación y Planificación en Proyectos de Software*. 2005 [cited 2007 15 de marzo]; Available from: http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php.
9. *Calidad del Sofware*. 2006 [cited 2007 12 de mayo]; Available from: http://www.calidaddelsoftware.com/modules.php?name=News&file=article&sid=157.
10. Molpeceres, A. *Procesos de desarrollo: RUP, XP y FDD*. 2005 [cited 2006 12 de noviembre]; Available from: http://www.javahispano.org/articles.article.action?id=76.
11. Ivar Jacobson, G.B.y.J.R., *El procesos Unificado de Desarrollo de Software*. Vol. I. 2004, La Habana, Cuba.
12. Wikimedia Foundation, I. *Población estadística*. 2007 [cited 2007 10 de mayo]; Available from: http://es.wikipedia.org/wiki/Poblaci%C3%B3n_estad%C3%ADstica.
13. Wikimedia Foundation, I. *Muestra estadística*. 2007 [cited 2007 23 de mayo]; Available from: http://es.wikipedia.org/wiki/Muestra_estad%C3%ADstica.
14. SOTO, R.G.N.Y.C.M. (2006) *INTERVALOS DE CONFIANZA PARA LAS ESTIMACIONES DE PROPORCIONES Y LAS DIFERENCIAS ENTRE ELLAS*. Interdisciplinaria

BIBLIOGRAFÍA CONSULTADA

Koch, M.J.E.y.N. *Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo*. 2002 [cited 2007 20 de febrero]; Available from: http://lsiweb.lsi.us.es/docs/informes/LSI-2002-4.pdf

J. J. Gutiérrez, M.J.E., M. Mejías, J. Torres. *Mejora de la calidad de los requisitos mediante la generación de pruebas*. [cited 2007 15 de abril]; Available from: http://www.lsi.us.es/~javierj/investigacion_ficheros/Mejorando%20casos%20de%20uso%2003.pdf.

Javier J. Gutiérrez, M.J.E., Arturo H. Torres, Manuel Mejías y Jesús Torres. *HACIA UNA PROPUESTA DE PRUEBAS TEMPRANAS DEL SISTEMA*. [cited 2007 20 de mayo]; Available from: http://in2test.lsi.uniovi.es/pris2006/PRIS2006-GutierrezEscalonaMejiasTorres.pdf.

Javier J. Gutiérrez, M.J.E., Manuel Mejías y Antonia M. Reina. *MODELOS DE PRUEBAS PARA PRUEBAS DEL SISTEMA*. [cited 2007 20 de mayo]; Available from: http://www.lsi.us.es/~javierj/publications/MDA14.pdf.

Peralta, M. *ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO*. 2004 [cited 2007 21 de febrero]; Available from: http://www.itba.edu.ar/capis/rtis/rtis-6-1/estimacion-del-esfuerzo-basada-en-casos-de-usos.pdf.

Gracia, J. *Desarrollo de Software Orientado a Objetos*. 2003 [cited 2006 25 de noviembre]; Available from: http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php.

Javier J. Gutiérrez, M.J.E., Manuel Mejías y Antonia M. Reina. *Definición de casos de uso para procesos de generación automática de pruebas del sistema*. [cited 2007 24 de febrero]; Available from: http://www.lsi.us.es/~javierj/publications/Hito02.pdf.

Mestras, J.P. *Proceso Unificado y la captura de requisitos*. [cited 2007 10 de enero]; Available from: http://www.fdi.ucm.es/profesor/jpavon/is2/04Requisitos.pdf.

Larman, C., *UML y Patrones*. 2da edición.

Probasco, I.S.a.L. (2000) *Traceability Strategies for Managing Requirements with Use Cases*. http://materjalid.tmk.edu.ee/kaarel_allik/ISP/traceabilityStrategies.pdf

Smith, J. (2003) *The Estimation of Effort Based on Use Cases*. ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/finalTP171.pdf

GLOSARIO DE TÉRMINOS

UCI: Universidad de las Ciencias Informáticas

RUP: Proceso Unificado de Software.

CMM: Modelo de Capacidad y madurez.

Feature: Es un tipo de requisito de alto nivel que se captura en el modelo del negocio en la captura de requisitos, se refleja en el documento visión y se rastrea para los casos de uso.

SRS: Especificación de requisitos de software.

Brainstorming: Técnica de captura de requisitos, llamada Tormenta de ideas.

Camino de ejecución o escenario: Es el conjunto de posibles interacciones entre los actores implicados y el sistema.

ANEXOS

Anexo 1 Modelo de Encuesta

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen?

2. ¿Se especifican los requisitos en forma de casos de uso? ¿Cómo?

3. ¿Qué nivel de detalle aplica a sus especificaciones de casos de uso?
   - Formal
   - Informal
   - Alto nivel
   - Expandido

4. ¿Cuánto usan los casos de uso en el proyecto? ¿Explique?
   - Nada
   - Poco
   - Algo
   - Mucho

5. ¿Se estima el proyecto usando alguna técnica relacionada con los casos de uso?

6. ¿Cuánto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?
   - Nada
   - Poco
   - Algo
   - Mucho

7. ¿Cómo distribuye el trabajo en su equipo para una iteración determinada?
   Por subsistemas.
   Por requisitos.
   Por casos de uso.
   Por subsistemas y casos de uso.

8. ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran?
   - Por subsistemas.
   - Por capas
   - Por casos de uso

9. ¿Cuánto aportan los casos de uso a estas pruebas?
   - Nada
   - Poco
   - Algo
   - Mucho

Anexo 2 Respuestas de las encuestas aplicadas a los responsables de los proyectos de la facultad.

Nombre del Proyecto: **Auto teórico**

Nombre del entrevistado: Juan Carlos Quevedo Lussón

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen?
Si, otro grupo de personas se reunieron con el cliente para recoger lo este quería que su sistema hiciera y luego esa información fue entregada al líder y a los desarrolladores.

2. ¿Se especifican los requisitos en forma de casos de uso? ¿Cómo?
Si, lo que se entregó fue lo que expresaron en forma de casos de uso.

3. ¿Qué nivel de detalle aplica a sus especificaciones de casos de uso?
Formal
Informal
Alto nivel
X Expandido

4. ¿Cuánto usan los casos de uso en el proyecto? ¿Explique?
Nada
X Poco, porque no todos los casos de uso se programaron
Algo
Mucho

5. ¿Se estima el proyecto usando alguna técnica basada en los casos de uso? No

6. ¿Cuánto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?
Nada
X Poco, porque no se usaron los casos de uso
Algo
Mucho

7. ¿Cómo distribuye el trabajo en su equipo para una iteración determinada?
X Por subsistemas.
Por requisitos.
Por casos de uso.
Por subsistemas y casos de uso.

8. ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran?
X Por subsistemas.
Por capas
Por casos de uso

9. ¿Cuánto aportan los casos de uso a estas pruebas?
Nada
Poco
X Algo
Mucho

Nombre del Proyecto: **Simulador de Tiro**

Nombre del entrevistado: Yanesky Montero Martínez

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen?
NO

2. ¿Se especifican los requisitos en forma de casos de uso? ¿Cómo?
NO

3. ¿Qué nivel de detalle aplica a sus especificaciones de casos de uso? NO
Formal
Informal
Alto nivel
Expandido

4. ¿Cuánto usan los casos de uso en el proyecto? ¿Explique?
X Nada, porque no se especifican los casos de uso
Poco
Algo
Mucho

5. ¿Se estima el proyecto usando alguna técnica basada en los casos de uso?
NO

6. ¿Cuánto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?
X Nada, porque no se especifican los casos de uso
Poco
Algo
Mucho

7. ¿Cómo distribuye el trabajo en su equipo para una iteración determinada?
X Por subsistemas.
Por requisitos.
Por casos de uso.
Por subsistemas y casos de uso.

8. ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran?
X Por subsistemas.
Por capas
Por casos de uso

9. ¿Cuánto aportan los casos de uso a estas pruebas?
X Nada No se especifican los casos de uso
Poco
Algo
Mucho

Nombre del Proyecto: Juego de Mesa

Nombre del entrevistado: Yeisnier Dominguez Silva

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen? NO

2. ¿Se especifican los requisitos en forma de casos de uso? ¿Cómo? NO

3. ¿Qué nivel de detalle aplica a sus especificaciones de casos de uso? NO

Formal
Informal
Alto nivel
Expandido

4. ¿Cuánto usan los casos de uso en el proyecto? ¿Explique?

X Nada
Poco
Algo
Mucho

5. ¿Se estima el proyecto usando alguna técnica relacionada con los casos de uso?
Empírico, por experiencia.

6. ¿Cuánto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?

X Nada
Poco
Algo
Mucho

7. ¿Cómo distribuye el trabajo en su equipo para una iteración determinada?

X Por subsistemas.
Por requisitos.
Por casos de uso.
Por subsistemas y casos de uso.

8. ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran?

X Por subsistemas.
Por capas
Por casos de uso

9. ¿Cuánto aportan los casos de uso a estas pruebas?

X Nada
Poco
Algo
Mucho

Nombre del Proyecto: **Herramientas de desarrollo para sistemas de realidad virtual**

Nombre del entrevistado: Yanosky Camacho Román

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen? NO

2. **¿Se especifican los requisitos en forma de casos de uso? ¿Cómo?** NO

3. **¿Que nivel de detalle aplica a sus especificaciones de casos de uso?** No se hace
Formal
Informal
Alto nivel
Expandido

4. **¿Cuánto usan los casos de uso en el proyecto? ¿Explique?**
X Nada
Poco
Algo
Mucho

5. **¿Se estima el proyecto usando alguna técnica basada en los casos de uso?**
Estimaciones Empíricas

6. **¿Cuanto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?** No los usan
Nada
Poco
Algo
Mucho

7. ¿Como distribuye el trabajo en su equipo para una iteración determinada?
X Por subsistemas.
Por requisitos.
Por casos de uso.
Por subsistemas y casos de uso.

8 . ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran?
X Por subsistemas.
Por capas
Por casos de uso

9. **¿Cuanto aportan los casos de uso a estas pruebas?**
X Nada
Poco
Algo
Mucho

Nombre del Proyecto: SCADA

Nombre del entrevistado: Juan Fung

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen?
Si. Entrevistas, criterios de experto, ingeniería inversa.

2. ¿Se especifican los requisitos en forma de casos de uso? ¿Cómo?
Algunos, hay requerimientos que no se pueden expresar mediante casos de uso.

3. ¿Que nivel de detalle aplica a sus especificaciones de casos de uso?
Formal
Informal
Alto nivel
X Expandido. Siempre que esto es posible.

4. ¿Cuánto usan los casos de uso en el proyecto? ¿Explique?
Nada
Poco
Algo
X Mucho, Mucho cuando se aplica.

5. ¿Se estima el proyecto usando alguna técnica basada en los casos de uso? NO

6. ¿Cuanto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?
X Nada. El proyecto se pactó a partir de las necesidades y restricciones de tiempo del cliente.
Poco
Algo
Mucho

7. ¿Como distribuye el trabajo en su equipo para una iteración determinada?
X Por subsistemas. Si descompone el trabajo hasta el nivel de atomicidad que mejor permita cada caso.
Por requisitos.
Por casos de uso.
Por subsistemas y casos de uso.

8. ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran?
X Por subsistemas. Igual que el anterior, partiendo de los subsistemas pero hay pruebas en las que deberemos probar la integración, algunas ya han tenido lugar.
Por capas
Por casos de uso

9. ¿Cuanto aportan los casos de uso a estas pruebas?
Nada
Poco
X Algo. Depende de cada caso. Te explicaba antes que no siempre aplica esto a nuestro proyecto.
Mucho

Nombre del Proyecto: Simpro

Nombre del entrevistado: Lázaro Abreu

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen? NO

2. **¿Se especifican los requisitos en forma de casos de uso? ¿Cómo?** NO

3. **¿Que nivel de detalle aplica a sus especificaciones de casos de uso?** No se utiliza ningún nivel de detalle

Formal
Informal
Alto nivel
Expandido

4. ¿Cuánto usan los casos de uso en el proyecto? ¿Explique?

X Nada
Poco
Algo
Mucho

5. **¿Se estima el proyecto usando alguna técnica relacionada con los casos de uso?** NO

6. **¿Cuanto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?**

X Nada
Poco
Algo
Mucho

7. ¿Como distribuye el trabajo en su equipo para una iteración determinada? NO se distribuye el trabajo.

Por subsistemas.
Por requisitos.
Por casos de uso.
Por subsistemas y casos de uso.

8. ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran? NO se realiza pruebas de funcionalidad.

Por subsistemas.
Por capas
Por casos de uso

9. **¿Cuanto aportan los casos de uso a estas pruebas?**

X Nada
Poco
Algo
Mucho

Nombre del Proyecto: Simulador Quirúrgico

Nombre del entrevistado:Juan M. Mederos

1. ¿Se capturan los requisitos de software en su proyecto? ¿Cómo lo hacen? Si. Se capturan mediante entrevista con el cliente.

2. ¿Se especifican los requisitos en forma de casos de uso? ¿Cómo? Si

3. ¿Que nivel de detalle aplica a sus especificaciones de casos de uso?

Formal
X Informal
Alto nivel
Expandido

4. ¿Cuánto usan los casos de uso en el proyecto? ¿Explique?

Nada
Poco
X Algo Se están definiendo requisitos todavía.
Mucho

5. ¿Se estima el proyecto usando alguna técnica basada en los casos de uso?

No

6. ¿Cuanto aportan los casos de uso en la planificación de las iteraciones? ¿Por qué?

Nada
X Poco El desarrollo de algoritmos pesa mas en plan de iteración.
Algo
Mucho

7. ¿Como distribuye el trabajo en su equipo para una iteración determinada?

X Por subsistemas.
Por requisitos.
Por casos de uso.
Por subsistemas y casos de uso.

8. ¿Se realizan pruebas de funcionalidad? ¿Cómo las estructuran?

No. No se ha concluido finalmente ninguna.
Por subsistemas.
Por capas
Por casos de uso

9. ¿Cuanto aportan los casos de uso a estas pruebas?

Nada
Poco
Algo
Mucho

Anexo 3 Ejemplo de la descripción de un caso de uso.

Caso de uso: Procesar venta

Actor principal: Cajero.

Personal involucrado e intereses:

- Cajero: quiere entradas precisas, rápidas, y sin errores de pago, ya que las pérdidas se deducen de su salario.
- Vendedor: quiere que las comisiones de las ventas estén actualizadas.
- Compañía: Quiere registrar las transacciones con precisión y satisfacer los intereses de los clientes. Quiere asegurar que se registran los pagos aceptados por el Servicio de Autorización de Pagos. Quiere cierta tolerancia a fallos que permita capturar las ventas incluso si los componentes del servidor (ej. validación remota de crédito) no están disponibles. Quiere actualización automática y rápida de la contabilidad y el inventario.
- Agencia Tributaria del Gobierno: quiere recopilar los impuestos de cada venta. Podrían ser múltiples agencias: nacional, provincial y local.
- Servicio de Autorización de Pagos: Quiere recibir peticiones de autorización digital con el formato y protocolo correctos. Quiere registrar de manera precisa las cuentas por cobrar de la tienda.

Precondiciones: El cajero se identifica y autentica.

Garantías de éxito (Poscondiciones): Se registra la venta. El impuesto se calcula de manera correcta. Se actualizan la contabilidad y el inventario. Se registran las comisiones. Se genera el recibo. Se registran las autorizaciones de pago aprobadas.

Escenario principal de éxito (o Flujo Básico):

1. El Cliente llega a un terminal PDV con mercancías y/o servicios que comprar.
2. El Cajero comienza una nueva venta.
3. El Cajero introduce el identificador del artículo.
4. El Sistema registra la línea de la venta y presenta la descripción del artículo, precio y suma parcial. El precio se calcula a partir de un conjunto de reglas de precios.

El Cajero repite los pasos 3-4 hasta que se indique.

5. El Sistema presenta el total con los impuestos calculados.
6. El Cajero le dice al Cliente el total y pide que le pague.
7. El Cliente paga y el Sistema gestiona el pago.

8. El Sistema registra la venta completa y envía la información de la venta y el pago al sistema de Contabilidad externo (para la contabilidad y las comisiones) y al sistema de Inventario (para actualizar el inventario).

9. El Sistema presenta el recibo.

10. El Cliente se va con el recibo y las mercancías (si es el caso).

Extensiones (o Flujos Alternativos):

*a. En cualquier momento el Sistema falla:

Para dar soporte a la recuperación y registro correcto, asegura que todos los estados y eventos significativos de una transacción puedan recuperarse desde cualquier paso del escenario.

1. El Cajero reinicia el Sistema, inicia la sesión, y solicita la recuperación al estado anterior.

2. El Sistema reconstruye el estado anterior.

2a. El Sistema detecta anomalías intentando la recuperación:

1. El Sistema informa del error al Cajero, registra el error, y pasa a un estado limpio.
2. El Cajero comienza una nueva venta.

3a. identificador no válido:

1. El Sistema señala el error y rechaza la entrada.

3b. Hay muchos artículos de la misma categoría y tener en cuenta una única identidad del artículo no es importante (ej. 5 paquetes de hamburguesas vegetales):

1. El Cajero puede introducir el identificador de la categoría del artículo y la cantidad.

3-6a. El Cliente le pide al Cajero que elimine un artículo de la compra:

1. El Cajero introduce el identificador del artículo para eliminarlo de la compra.
2. El Sistema muestra la suma parcial actualizada.

3-6b. El Cliente le pide al Cajero que cancele la venta:

1. El Cajero cancela la venta en el Sistema.

3-6c. El Cajero detiene la venta:

1. El sistema registra la venta para que esté disponible su recuperación en cualquier terminal PDV.

4a. El Sistema genera el precio de un artículo que no es el deseado (ej. el Cliente se queja por algo y se le ofrece un precio más bajo):

1. El Cajero introduce el precio alternativo.
2. El Sistema presenta el precio nuevo.

5a. El sistema encuentra algún fallo para comunicarse con el servicio externo del sistema de cálculo de impuestos.

1. El Sistema reinicia el servicio en el nodo *PDV y* continúa

1a. El Sistema detecta que el servicio no se reinicia.

1. El Sistema señala el error.

2. El Cajero podría calcular e introducir manualmente el impuesto, o cancelar la venta.

5b. El Cliente dice que le son aplicables descuentos (ej. empleado, cliente preferente):

1. El Cajero señala la petición de descuento.

2. El Cajero introduce la identificación del Cliente.

3. El Sistema presenta el descuento total, basado en las reglas de descuento.

5c. El Cliente dice que tiene crédito en su cuenta, para aplicar a la venta: 1. El Cajero señala la petición de crédito.

1. El Cajero introduce la identificación del Cliente.

2. El Sistema aplica el crédito hasta que el precio = 0, y reduce el crédito que queda.

6a. El Cliente dice que su intención era pagar en efectivo pero que no tiene suficiente:

1a. El Cliente utiliza un método de pago alternativo.

1 b. El Cliente le dice al Cajero que cancele la venta. El Cajero cancela la venta en el Sistema.

7a. Pago en efectivo:

1. El Cajero introduce la cantidad de dinero en efectivo entregada.

2. El Sistema muestra la cantidad de dinero a devolver y abre el cajón de caja.

3. El Cajero deposita el dinero entregado y devuelve el cambio al Cliente.

4. El Sistema registra el pago en efectivo.

7b. Pago a crédito:

1. El Cliente introduce la información de su cuenta de crédito.

2. El Sistema envía la petición de autorización del pago al Sistema externo de Servicio de Autorización de Pagos, y solicita la aprobación del pago.

2a. El Sistema detecta un fallo en la colaboración con el sistema externo:

1. El Sistema señala el error al Cajero.

2. El Cajero le pide al Cliente un modo de pago alternativo.

3. El Sistema recibe la aprobación del pago y lo notifica al Cajero.

3a. El Sistema recibe la denegación del pago:

1. El Sistema señala la denegación al Cajero.
2. El Cajero le pide al Cliente un modo de pago alternativo.
4. El Sistema registra el pago a crédito, que incluye la aprobación del pago
5. El Sistema presenta el mecanismo de entrada para la firma del pago a crédito.
6. El Cajero le pide al Cliente que firme el pago a crédito. El Cliente introduce la firma.

7c. Pago con cheque...

7d. Pago a cuenta...

7e. El Cliente presenta cupones:

1. Antes de gestionar el pago, el Cajero recoge cada cupón y el Sistema reduce el pago como sea oportuno. El sistema registra los cupones utilizados por razones de contabilidad.

1a. El cupón introducido no es válido para ninguno de los artículos comprados

1. El Sistema señala el error al Cajero.

8a. Hay rebajas en los artículos:

1. El Sistema presenta los formularios de rebaja y los recibos de descuento para cada artículo con una rebaja.

8b. El Cliente solicita un vale-regalo (sin precio visible):

1. El Cajero solicita el vale-regalo y el Sistema lo proporciona.

Requisitos especiales:

- Interfaz de Usuario con pantalla táctil en un gran monitor de pantalla plana. El texto debe ser visible a un metro de distancia.
- Tiempo de respuesta para la autorización de crédito de 30 segundos el 90% de las veces.
- De algún modo, queremos recuperación robusta cuando falla el acceso a servicios remotos, como el sistema de inventario.
- Internacionalización del lenguaje del texto que se muestra.
- Reglas de negocio que se puedan añadir en tiempo de ejecución en los pasos 3 y 7.

Lista de tecnología y variaciones de datos:

- 3a. El identificador del artículo se introduce mediante un escáner láser de código de barras (si está presente el código de barras) o a través del teclado.
- 3b. El identificador del artículo podría ser cualquier esquema de código UPC, EAN, JAN o SKU.

- 7a. La entrada de la información de la cuenta de crédito se lleva a cabo mediante un lector de tarjetas o el teclado.
- 7b. La firma de los pagos a crédito se captura en un recibo de papel. Pero en dos años, pronosticamos que muchos clientes querrán que se capture la firma digital.

Anexo 4 Ejemplo del método "Estimación por puntos de casos de uso"

Obtención del **P**eso de **A**ctores **S**in **A**justar (PASA).

Clasificación de la complejidad de los actores de acuerdo a la naturaleza de los mismos.

Actores	**Factor**	**Tipo de actor**
Almacenero	3	Complejo
Contador	3	Complejo
Esp. De Inventario	3	Complejo
Administrador	3	Complejo
Comercial	3	Complejo

Se tienen 5 actores todos con peso 3 por lo tanto:

PASA = 5 x 3 = 15

Clasificación de los Casos de Uso y obtención del **P**eso de **T**ransacciones **S**in **A**justar (PTSA)

Clasificación de la complejidad de los Casos de Uso de acuerdo al número de transacciones.

Casos de Uso	**Transacciones**	**Factor de peso**	**Tipo de caso de uso**
Ajustar Precios	3	5	Simple
Autenticarse	2	5	Simple
Cambiar Contraseña	2	5	Simple
Cancelar Movimientos	3	5	Simple
Fijar fecha	2	5	Simple
Gestionar Ajustes	9	15	Complejo
Gestionar Almacenes	9	15	Complejo
Gestionar Control Interno de Productos	9	15	Complejo
Gestionar Cuenta de Control de Inventario	3	5	Simple
Gestionar Opciones Contables	9	15	Complejo
Gestionar Órdenes de Trabajo	9	15	Complejo
Gestionar Roles	9	15	Complejo

Gestionar Tarjetero de Productos	6	10	Medio
Gestionar Usuarios	12	15	Complejo
Gestionar Vales de Entrada/Salida	7	10	Medio
Mostrar Reportes	17	15	Complejo
Mostrar Tipos de Movimientos	2	5	Simple
Realizar Cambio de Cuentas	3	5	Simple
Realizar Cambio de CUP	3	5	Simple
Realizar Cierre de Apertura	3	5	Simple
Realizar Cierres Año	6	10	Medio
Realizar Cierres Día	4	10	Medio
Realizar Cierres Mes	5	10	Medio
Realizar Informe de Recepción	2	5	Simple
Realizar Informe de Recepción a ciegas	2	5	Simple
Realizar Inventario Físico	10	15	Complejo
Realizar Transferencia de Materiales	3	5	Simple
Realizar Transferencia entre Almacenes	4	10	Medio
Ver Disponibilidad	2	5	Simple

Se tienen 29 casos de uso, de ellos 14 tienen complejidad simple con factor de peso 5, 6 tienen complejidad media con factor de peso 10 y 9 son complejos con factor de peso 15, de ahí se tiene que:

PTSA = (14 x 5) + (6 x 10) + (9 x 15) = 265

Obtención del Peso o **P**untos de **C**asos de **U**so **S**in **A**justar (PCUSA).

Luego de tener el peso de actores sin ajustar y el peso de transacciones sin ajustar se podrá obtener los puntos de casos de uso sin ajustar de la siguiente forma:

PCUSA = PASA + PTSA

PCUSA = 15 + 265 = 280

Clasificación de **F**actores de **C**omplejidad **T**écnica (FCT)

Clasificación del Factor de Complejidad Técnica del Sistema

Factor	Descripción	Factor de Peso	Valor	∑ (peso, valor)
T1	Sistema Distribuido	2	3	6
T2	Rendimiento o tiempo de respuesta	1	5	5
T3	Eficiencia del usuario final	1	5	5
T4	Complejidad de procedimiento interno	1	5	5
T5	Reusabilidad del código	1	4	4
T6	Facilidades de instalación	0.5	5	2,5
T7	Facilidad de uso	0.5	5	2,5
T8	Portabilidad	2	5	10
T9	Facilidades de cambio	1	4	4
T10	Concurrencia	1	4	4
T11	Características de seguridad	1	4	4
T12	Provee acceso directo a terceras partes	1	0	0
T13	Requerimientos de entrenamiento especial	1	4	4

Para obtener el factor final se utilizó la siguiente formula

FCT = 0.6 + 0.01 x ∑ (Peso i x Valor i)

FCT = 0.6 + 0.01 x (6 + 5 + 5 + 5 + 4 + 2,5 + 2,5 + 10 + 4 + 4 + 4 + 0 + 4)

FCT = 1, 16

Clasificación de **F**actores **A**mbientales (FA)

Clasificación del Factor de Ambiente del Sistema

Factor	Descripción	Peso	Valor	∑ (peso, valor)
E1	Conocimiento del proceso de desarrollo	1.5	4	6
E2	Experiencia en la aplicación	0.5	4	2
E3	Experiencia en Orientación a objetos	1	4	4

E4	Capacidad de liderazgo de los analistas	0.5	4	2
E5	Motivación	1	5	5
E6	Estabilidad de los requerimientos	2	2	4
E7	Personal a tiempo compartido	-1	2	-2
E8	Dificultad de los lenguajes de programación	-1	3	-3

Se calcula mediante la siguiente formula

FA = 1.4 - 0.03 x ∑ (Peso i x Valor i)

FA = 1.4 - 0.03 x (6 + 2 + 4 + 2 + 5 + 4 - 2 - 3)

FA = 0,86

Cálculo de **P**untos de **C**asos de **U**so **A**justados (PCUA)

Finalmente, se obtiene la siguiente fórmula que representa los puntos de casos de uso ajustados:

PCUA = PCUSA* FCT*FA

PCUA = 280 * 1, 16 * 0,86

PCUA = 385,28

Cálculo del Esfuerzo de Desarrollo (E)

Analizando el factor ambiente se observa que por debajo del valor medio (3) para E1...E6 existen 2 valores, mientras que por encima para E7...E8 no existe ningún valor por lo que el total es 2. Por lo anterior se plantea que FC = 20 horas-hombre y tomando el valor de los puntos de casos de uso ajustados calculado anteriormente, se puede calcular el esfuerzo estimado:

E = PCUA x FC

E = 385,28 x 20

E = 7705,6 Horas/Hombre

Para la estimación de la duración total del proyecto, se agrega a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software mostradas a continuación:

Tipo de actividad	Por ciento que representa	Horas-Hombre
Análisis	10%	19264
Diseño	20%	3852,8
Implementación	40%	7705,6
Prueba	15%	2889,6
Sobre Cargas (Otras actividades)	15%	2889,6
Total	100%	19264

El desarrollo de este proyecto es llevado a cabo por dos personas:

TDES = 19264 /2

TDES = 9632 Horas

Anexo 5 Ejemplo práctico de la técnica "Pruebas tempranas"

Se emplea un caso de uso extraído de un sistema médico para la detección y evaluación de minusvalías. Este sistema fue implementado en el año 2004 y actualmente está en explotación [Villadiego 2004].

4.1 Descripción del caso de uso

La descripción del caso de uso para controlar el acceso de un médico al sistema se muestra en la tabla 1.

Tabla 1. Definición del caso de uso.

Nombre del Caso de Uso:	Autenticación del médico
Actores:	Médico
Propósito:	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el médico intente acceder al sistema.
Referencias	RF-01, RF-02
Flujo de Eventos	
Flujo Básico	
Acción del Actor	**Respuesta del Sistema**
1 El Médico solicita entrar en el sistema. 3 El Médico proporciona identificador y contraseña.	2 El sistema solicita las credenciales al usuario. 4 Si el usuario está registrado en el sistema, el sistema permite el acceso al usuario, y se realiza el caso de uso Gestión de pacientes.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3 Si el identificador o la contraseña son incorrectos, el sistema vuelve al paso 2 y continúa el caso de uso.

Después de la descripción del caso de uso se hace el modelo de comportamiento, en este caso se escogió el diagrama de actividades ya que permite expresar fácilmente caminos alternativos y la interacción entre el sistema y los actores externos, identificando claramente a cada uno de los participantes. El diagrama correspondiente al caso de uso de la tabla 1 se muestra en la figura 3.

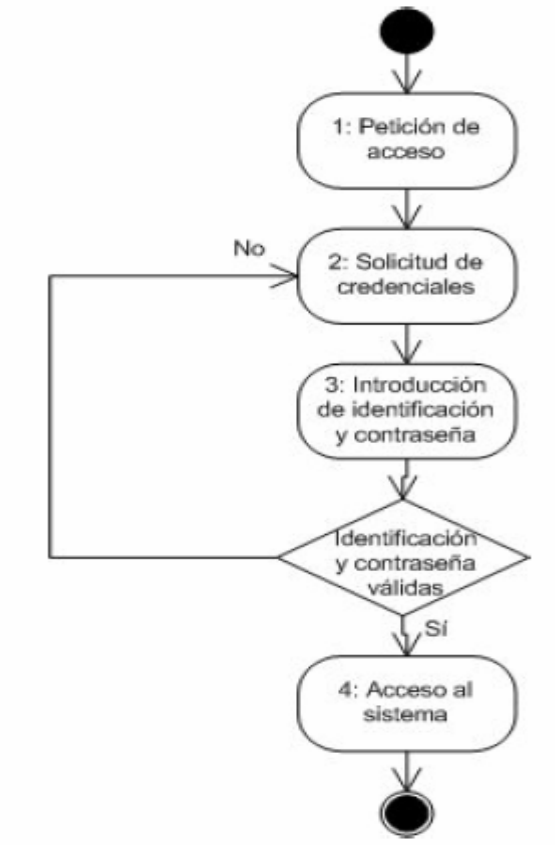


Fig. 3 Diagrama de actividades para el caso de uso “Autenticación del Médico”.

4.2 Generación de las pruebas

Pasos para la generación de pruebas a partir de la descripción del caso de uso anteriormente planteado.

Pasos	Tarea	Resultado
1	Identificar Caminos de ejecución.	Todos los caminos de ejecución posibles.
2	Seleccionar valores de prueba.	Conjunto de valores adecuados para cada

		camino.
3	Identificación de resultados.	Resultado esperado de cada camino.

En el primer paso se identifican todas las posibles secuencias de iteraciones entre el sistema y los actores participantes. Cada una de estas iteraciones será un camino de ejecución. Mediante la exploración de todos los posibles caminos se garantiza la cobertura de todo el comportamiento recogido en el caso de uso. A continuación se identifican los conjuntos de valores de entrada del caso de uso, o precondiciones necesarias, para recorrer cada uno de los caminos. Por último se identifica cual es el resultado esperado de cada camino de ejecución según los valores del paso dos. Este resultado será verificado por la prueba del sistema para evaluar si ha sido superada o no.

4.2.1 Identificación de caminos de ejecución

Se aplica un criterio de cobertura para recorrer todos los posibles caminos de ejecución del caso de uso. En este ejemplo el criterio utilizado consiste en pasar, al menos una vez, por cada nodo y cada transición. Basándonos en la información recogida en el caso de uso, los posibles caminos de ejecución identificados a partir de la Flujo Básico y Alterno se muestran en la tabla 3.

Tabla 3. Caminos de ejecución.

Caminos	Interacciones
1	1, 2, 3, 4
2	1, 2, 3, 2, 3, 4
3	1, 2, 3, 2, 3, 2, 3, 4
⋮	⋮
N	1, (2, 3)+, 4

El camino de ejecución número 1 describe la siguiente secuencia: un usuario perteneciente al rol médico solicita entrar en el sistema, el sistema solicita las credenciales del usuario, el usuario proporciona un identificador y contraseña registradas en el sistema y el sistema permite el acceso al usuario y se realiza el caso de uso Gestión de pacientes.

El camino de ejecución número 3 describe la secuencia de un usuario que escribe dos veces un identificador o contraseña incorrectos y, al tercer intento, consigue acceso al sistema.

Según la definición del caso de uso, los pasos 2 y 3 podrían repetirse infinitas veces, lo cual da como resultado infinitos caminos de ejecución posibles. Esto significa que cualquier usuario, puede intentar identificarse en el sistema un número infinito de veces. Esto es un aviso para consultar esta circunstancia con los ingenieros de requisitos y usuarios para verificar si esto es correcto o es necesario refinar el caso de uso.

También se pone de manifiesto que, tal y como está especificado en la tabla 1, el caso de uso solo permite un final exitoso con el acceso del médico al sistema, no contemplando que pueda tener un final distinto, por ejemplo denegación de acceso después de un límite de fallos.

Para la generación de casos de prueba se elige solo los dos primeros caminos de ejecución, mostrados en la tabla 4, ya que los demás no aportan nada nuevo al sistema.

Tabla 4. Caminos de ejecución seleccionados.

Camino	Descripción	Interacciones
1	Verifica el correcto acceso de un médico al sistema.	1, 2, 3, 4
2	Verifica el correcto acceso de un médico al sistema después de introducir un nombre erróneo.	1, 2, 3, 2, 3, 4

Si el caso de uso especificara qué debe suceder cuando se introduce un nombre incorrecto, por ejemplo un mensaje de advertencia, incluiríamos un camino de ejecución para comprobar ese resultado.

4.2.2 Selección de valores de prueba

A continuación seleccionamos los valores de prueba necesarios para cada camino de ejecución identificado en el apartado anterior. Para ello identificamos las condiciones que los valores de prueba deben cumplir para poder ejecutar los caminos identificados en la tabla 4.

Dado que el primer camino de ejecución permite el acceso al sistema, los valores de prueba serán cualquier nombre de médico y cualquier clave válida.

El segundo camino de ejecución admite dos categorías de valores posibles: un nombre de usuario inválido y una contraseña cualquiera, sea válida o no, y un nombre de usuario y clave válidos. Estos valores se recogen en la tabla 5. Se ha añadido una notación en dicha tabla, mediante corchetes, para indicar en que

interacción debe introducirse cada uno de esos valores.

Tabla 5. Valores de prueba.

Id	Caminos	Valores de pruebas
1	1, 2, 3[1], 4	[1] Nombre: Médico Válido Clave: Clave Válida
2	1, 2, 3[2], 2, 3[1], 4	[1] Nombre: Médico Válido Clave: Clave Válida [2] Nombre: Médico No válido Clave: Clave No válida

A la hora de establecer un nombre y una clave válida, comprobamos que el caso de uso no indica las reglas o restricciones que tienen que cumplir: tamaño mínimo y máximo caracteres permitidos y no permitidos, distinción entre mayúsculas y minúsculas. Tampoco indica ninguna referencia a donde estas reglas puedan encontrarse, como por ejemplo un requisito de almacenamiento.

Este tipo de restricciones pueden estar especificadas en requisitos independientes ya que pueden ser de aplicación a más de un caso de uso o a todo el sistema en general. Se identifican los requisitos de almacenamiento, para especificar las condiciones que tienen que cumplir los datos nombre de usuario y contraseña.

También se debe advertir de este hecho al equipo de requisitos, ya que, incluso, pueda darse el caso de que no se hayan tenido en cuenta este tipo de restricciones, o que se de por sobrentendido el concepto de nombre de usuario y clave válida cuando no es así.

4.2.3 Identificación de resultados

A continuación es necesario identificar claramente cual será el resultado observable de cada camino de ejecución. Este resultado debe expresarse de manera que sea posible observarlo desde el exterior del sistema. También debe estar definido con precisión ya que este resultado será lo que la prueba verificará para evaluar si ha sido superada satisfactoriamente o no.

En el caso de uso de ejemplo, se aprecia que todos los caminos de ejecución terminan con el mismo

resultado: la ejecución del caso de uso 2. Este caso de uso recoge la funcionalidad de la pantalla de control de datos del paciente, por lo que, aunque el caso de uso no lo recoge explícitamente, el resultado esperado observable será el acceso o no a esta pantalla. Los resultados esperados de los caminos de ejecución se muestran en la tabla 6.

Tabla 6. Resultados esperados.

Id	Descripción	Resultado esperado
1	Verifica el correcto acceso de un médico al sistema	Pantalla para poder ejecutar las funcionalidades del caso de uso Gestión de pacientes.
2	Verifica el correcto acceso de un médico al sistema después de introducir un nombre erróneo.	Pantalla para poder ejecutar las funcionalidades del caso de uso Gestión de pacientes.

Nuevamente se aprecia como el único resultado posible de este caso de uso es acceder a la pantalla de control de datos del paciente o que, el usuario desista de seguir intentando el acceso.

4.2.4 Informe de errores

Una vez realizado este proceso y comprobado que hay errores y omisiones en este caso de uso, se debe redactar un informe informando de ello y remitirlo a los ingenieros de requisitos para que procedan a realizar las correcciones necesarias. En las tabla 7 se muestra un resumen con todos los errores identificados en los distintos pasos del proceso de generación de prueba.

Tabla 7. Errores detectados.

Error	Descripción
Omisión de caminos	No existe ningún camino que limite el número de intentos de acceso.
Camino insuficientemente detallado	No se detalla cual es el resultado de introducir un nombre o contraseña inválida.
Ambigüedad	No se detalla que se entiende nombre de usuario válido o inválido. Lo mismo para la clave.

Falta de información	No se detalla, ni se ofrecen referencias, sobre las características que deben tener los nombres de usuario y las claves.
Resultado sin camino.	No existe ningún camino que conduzca a la denegación de acceso, salvo el desistimiento del usuario.

Los ingenieros de requisitos, junto con los usuarios del sistema o clientes, deberán evaluar estos errores y modificar el caso de uso, o añadir casos de uso adicionales, para corregirlos.

4.3 Caso de uso corregido

Una vez corregidos todos los errores detectados en el apartado anterior, la nueva redacción del caso de uso se muestra en la tabla 8 y en la figura 4. Además se ha desarrollado un requisito de almacenamiento, recogido en la tabla 9, para definir las condiciones a cumplir por el identificador y la clave.

RF-01 Autenticación del médico

RF-02 Gestión de pacientes

Tabla 8. Descripción del caso de uso corregido.

Nombre del Caso de Uso:	Autenticación del médico
Actores:	Médico
Propósito:	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el médico intente acceder al sistema.
Referencias	RF-01, RF-02, SR-01
Flujo de Eventos	
Flujo Básico	
Acción del Actor	**Respuesta del Sistema**
1 El Médico solicita entrar en el sistema. 3 El Médico proporciona identificador y contraseña.	2 El sistema solicita las credenciales al usuario. 4 Si el usuario está registrado en el sistema, el sistema permite el acceso al usuario, y se realiza el caso de uso Gestión de pacientes.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3. Si el usuario introdujo un identificador incorrecto 3 veces consecutivas, el sistema considera que no es médico y muestra un mensaje impidiéndole el acceso. 4. Si el identificador no está registrado, el sistema vuelve al paso 2, muestra un mensaje de aviso y continúa el caso de uso. 5. Si la contraseña no coincide con la contraseña registrada para el identificador, el sistema vuelve al paso 2 y muestra una opción para reenviar la contraseña al usuario.

Tabla 9. Requisito de almacenamiento para el identificador y la contraseña.

Nombre	SR-01. Datos de acceso del médico	
Casos de uso asociados	Autenticación del médico	
Datos específicos	Nombre	Dominio
	Identificador	Cadena de texto
	Contraseña	Cadena de texto
Restricciones	Nombre	Restricción
	Identificados	Al menos 5 caracteres.
	Contraseña	Al menos 5 caracteres.

Se han añadido todas las carencias detectadas, como lo que sucede si alguien que no sea médico intenta acceder al sistema. También se pueden identificar posibles casos de uso adicionales, como un caso de uso para recordar la contraseña de un médico que la hubiese olvidado.

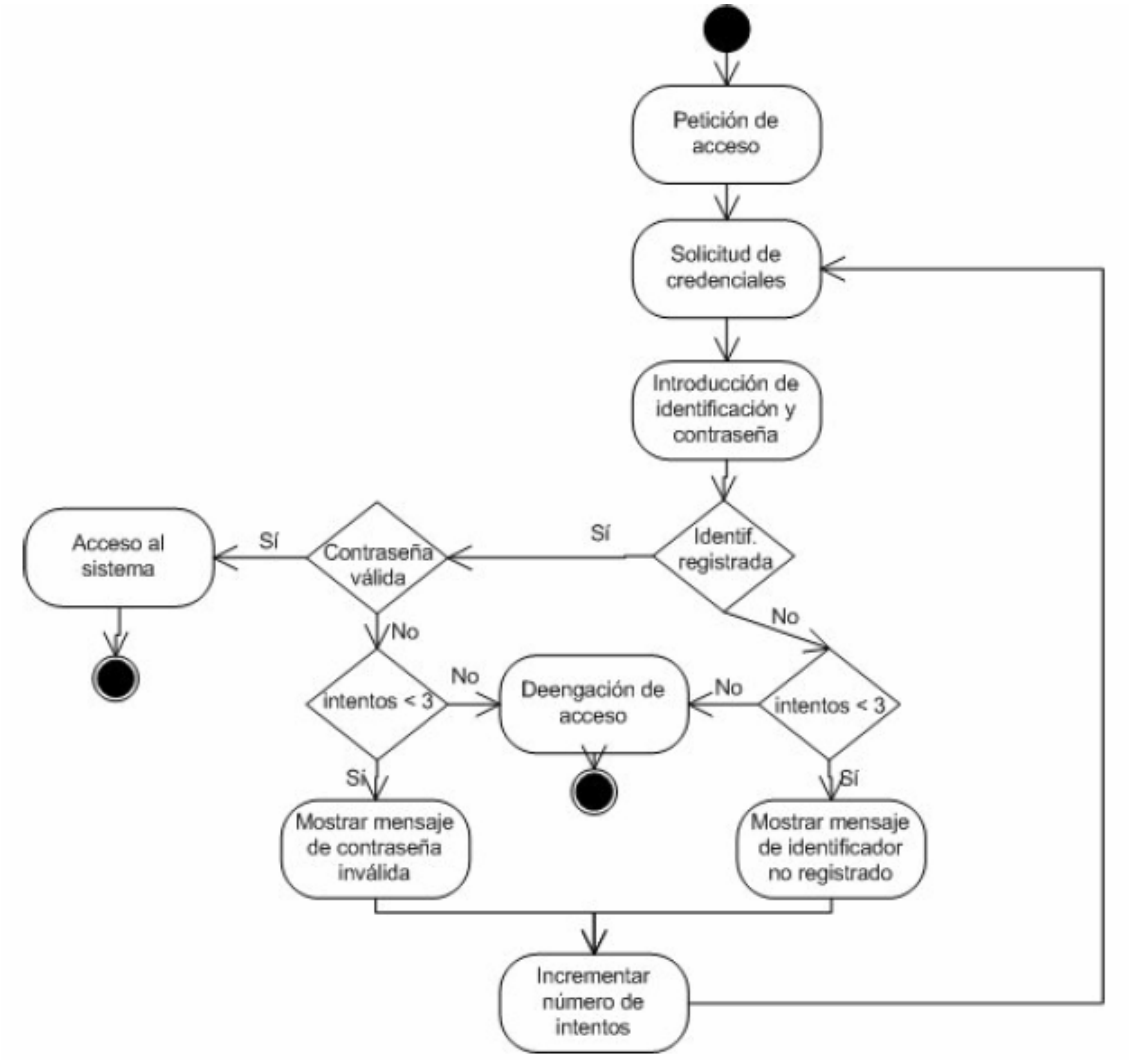


Fig. 4. Diagrama de actividades del caso de uso corregido.

Como se ha mencionado en apartados anteriores, este caso de uso pertenece a un sistema real, que se implementó y que actualmente se utiliza en asociaciones médicas.

Todas las carencias detectadas tuvieron que ser resueltas durante las fases de diseño e implementación. Sin embargo, es muy beneficioso detectar estas carencias en etapas tempranas, y, más concretamente, durante la elicitación de requisitos, que tener que aplicar parches improvisados durante la construcción del sistema.