UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS

LICENCIATURA EN SISTEMAS DE INFORMACIÓN

TRABAJO FINAL DE GRADUACIÓN

APLICACIÓN DE ETNOGRAFÍA EN UNA HERRAMIENTA DE GESTIÓN DE REQUISITOS EN UN ENTORNO XP

Autor:

RITA DANIELA VERA

Profesora Guía**:**

DIANA PALLIOTTO

Noviembre de 2006

UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS

TRABAJO FINAL DE GRADUACIÓN DE LA LICENCIATURA EN SISTEMAS DE INFORMACIÓN

APLICACIÓN DE ETNOGRAFÍA EN UNA HERRAMIENTA DE GESTIÓN DE REQUISITOS EN UN ENTORNO XP

Autor:

.....

Rita Daniela Vera

Profesora Guía:

.....

Diana Palliotto

* ——————————————————— * ——————————————————— *

Aprobado el día ..... del mes de ..... del año 20.....

por el Tribunal integrado por

....

....

Santiago del Estero – Argentina

A la memoria de Boni.

Para Aldo, Alba, Maria y Gustavo;
quienes son la gran motivación
de mi vida.

Rita Daniela Vera

Agradecimientos

Antes que nada, deseo expresar mi enorme agradecimiento a mi profesora guía Ing. Diana Palliotto por su desinteresada colaboración y aporte de conocimientos, traducido en comentarios, correcciones y críticas que han enriquecido mi Trabajo Final de Graduación.

Una mención especial al momento de agradecer es para el Ing. Héctor Paz por su constante predisposición y buena voluntad, brindándome reflexiones, consideraciones y todo lo que estuvo a su alcance para que mi labor se simplifique.

También, hago extensivo mi reconocimiento a todos los integrantes de la Cátedra de Fundamentos de la Programación de la FCEyT por su paciencia, generosidad y buenos momentos compartidos en estos años de ayudantía, que a través del intercambio de ideas y conocimientos constituyeron un pilar esencial en mi formación.

Finalmente, mi agradecimiento a mis compañeros de la facultad y en especial a mis amigos de la vida Cesar, Daniel y Fernando.

R.V.

Santiago del Estero, Argentina

Noviembre de 2006

CONTENIDO

RESUMEN

El ciclo de vida en la construcción de un sistema software comienza con actividades de identificación y análisis de requisitos. Está comprobado que estas actividades iniciales del desarrollo son las más críticas. Si no se realiza el análisis, es muy probable que se construya una solución software que resuelva incorrectamente el problema. Como resultado se pierde dinero y tiempo, además crea frustración en equipo de desarrollo e insatisfacción en los clientes. Por lo tanto, uno de los retos más importantes de la elicitación de requisitos es garantizar que los requisitos del sistema sean consistentes con las necesidades de la organización donde se utilizará el mismo y con las futuras necesidades de los usuarios.

Para ello, se propone el diseño de una herramienta software aplicable a un entorno de desarrollo XP[1] llamada HGH (Herramienta de Gestión de Historias), que permita la especificación, el control y la actualización de las historias de usuario. Además, se expone una serie de lineamientos para aplicar etnografía durante la fase de captura de requisitos (elicitación); se plantea el uso de los estudios etnográficos en esta fase de desarrollo de software, ya que pueden aportar los requisitos que justamente suelen escapar cuando se aplican otros métodos más conocidos. El uso de la etnografía tiene como objetivo el suministrar una mejor comprensión de las prácticas de trabajo y del comportamiento de los grupos que las llevan a cabo. De esta manera, se pueden incluir datos, no sólo de las actualizaciones de las historias construidas por los usuarios, sino también información obtenida por los desarrolladores a partir de la aplicación de lo que se conoce como el método etnográfico.

Para demostrar que HGH y los lineamientos de aplicación de etnografía durante la captura de requisitos permite a usuarios y desarrolladores a obtener una mejor documentación de las historias de usuario del sistema, de manera de evitar problemas en

[1] XP: Programación Extrema

el diseño y facilitar el mantenimiento, se construyó un prototipo de la herramienta que se utilizó durante el desarrollo de dos sistemas.

Los resultados de la aplicación del prototipo se sometieron a métricas de calidad de la especificación propuestas por A. Davis, con lo cual se pudo comprobar que las especificaciones obtenidas cumplen con las características de calidad evaluadas.

Palabras claves: requisitos; programación extrema; historias de usuario; etnografía.

INTRODUCCIÓN

El proceso de desarrollo de software comienza con un cliente que establece lo que quiere que haga el sistema en función de una solución que cree que es la mejor aunque, muchas veces, no determina qué es lo que realmente necesita; en este caso, es preferible clientes con escasos conocimientos en software, ya que los que conocen del tema creen que ya tienen la solución, pero usualmente la misma conduce a fracasos en la práctica.

Una vez que el cliente estableció lo que necesita, los desarrolladores especifican lo que ellos entendieron a partir de lo recabado durante la fase de análisis de requisitos (por razones de tiempo, generalmente no se documenta nada, se prefiere avanzar en el desarrollo). Luego se diseña el sistema en función de las especificaciones realizadas. Sin embargo, a menudo sucede que los requisitos[2] iniciales cambian y el desarrollo no se adapta a esos cambios, ya sea porque no existe documentación o, si existe, no se actualiza; o bien, surgen dudas respecto a lo especificado y el cliente ya no está para aclararlas.

En algunos casos se tiene la oportunidad de consultar a un asesor externo (en nuestro medio esto no ocurre con frecuencia, pues el equipo de desarrollo está formado por pocas personas), que plantea alguna solución que no se adapta al contexto del sistema.

Finalmente, siempre lo mismo, es imposible realizar el mantenimiento del sistema debido a la falta de una buena documentación de todo el proceso. Además, el producto desarrollado no satisface las necesidades, el cliente está descontento y el equipo de desarrollo se encuentra en problemas (véase Figura 1).

[2] Los requisitos son una lista de propiedades que el sistema debería tener para ser exitoso en el entorno en el que se usará

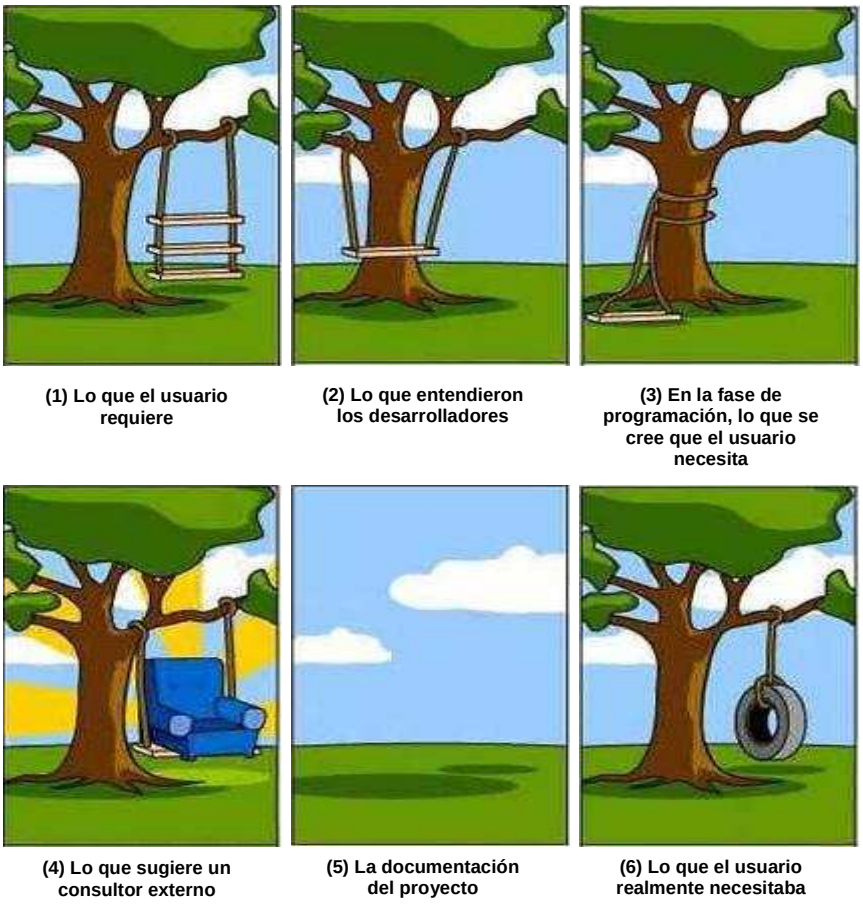

Figura 1. Caricatura del proceso de desarrollo de Software
(adaptada de http://www.marquetti-asociados.com.ar/xp.htm)

Por todo lo dicho anteriormente, lo que se plantea en el presente proyecto es el diseño de una herramienta software para la captura, la especificación y el control de requisitos, que permita solucionar los errores provocados por los requerimientos cambiantes o por la falta de entendimiento entre lo que el cliente quiere y lo que el desarrollador entiende. Para ello, se propone el uso de la etnografía en la captura de requisitos y que se aplique en un entorno de desarrollo ágil como es la programación extrema.

En el primer capítulo se delimita y se plantea el problema a tratar; además, se especifican los antecedentes y el alcance de la investigación, se expresan los objetivos que se plantearon al comenzar el estudio, y se presenta la hipótesis que dirige el proceso de investigación.

El segundo capítulo presenta los marcos referenciales: teórico, metodológico y empírico. En el marco teórico se expresan las teorías y los conceptos sobre los que se apoyan los conocimientos que apuntan a describir, comprender y explicar el problema a investigar. En el marco metodológico se hace referencia a la metodología a utilizar. Y, finalmente, en el marco empírico se presentan los sistemas que se someterán a las

recomendaciones de aplicación de etnografía durante la captura de requisitos y a la herramienta propuesta.

En los capítulos tercero y cuarto se exponen los lineamientos antes mencionados de la aplicación de etnografía en la captura de requisitos, y se presenta el diseño de la herramienta de gestión de historias de usuario respectivamente.

En el quinto capítulo se describe un prototipo de la herramienta. Y, por último, en el último capítulo se evalúan los resultados utilizando la lista de características para valorar la calidad de la especificación de requisitos propuesta por A. Davis.

CAPÍTULO I

PROBLEMAS, OBJETIVOS E HIPÓTESIS

La finalidad del presente capítulo es delimitar y plantear el problema a abordar, así como especificar los antecedentes a tener en cuenta para llevar a cabo el trabajo. Además, se procede a definir el alcance de la investigación y los objetivos que se plantearon al comenzar el estudio. Finalmente, se presenta la hipótesis que conduce el proceso de investigación.

I.1. DELIMITACIÓN DEL PROBLEMA

> *"El éxito de cualquier proyecto se mide en base a cuatro parámetros: plazo, costo, funcionalidad y calidad"* [3, 34].

Los proyectos de software habitualmente se exceden en tiempo y costo. Además, en muchos casos, el producto obtenido no es lo que el usuario necesita. Esto se debe a la falta de información y a los cambios permanentes [3], por lo que la capacidad para lidiar con ellos es esencial para el éxito.

Entonces, para lograr el éxito en el desarrollo de software, es importante determinar en forma clara los requisitos de la aplicación. Sin embargo, frecuentemente nos enfrentamos con situaciones en las que los requisitos cambian continuamente o que el cliente no puede definir claramente los requisitos para su producto o, también, el problema está en la falta de entendimiento entre lo que el cliente quiere y lo que el desarrollador entiende. Pueden surgir otros tipos de cambios ajenos al proyecto pero que afectan a los requerimientos del producto, por ejemplo, cambios en el negocio, en las tecnologías, las limitaciones financieras, cambios en el mercado, legislación o condiciones laborales. En consecuencia, la funcionalidad del producto final puede no tener relación con lo que el cliente realmente necesita [3, 41].

Los Métodos Tradicionales (MT) definen un proceso secuencial para el desarrollo de software. Cada etapa depende de la anterior. Por lo tanto, realizar cambios para solucionar problemas del producto cuando éste ya se ha entregado, cuesta mucho más que realizar cambios en las etapas tempranas. El costo del cambio está muy relacionado con la metodología que se elige para llevar a cabo el proceso, y con la complejidad (cuanto más complejo es un proyecto, mayor será el costo del cambio). En los MT, la curva de costo del cambio crece exponencialmente a medida que transcurre el tiempo. Sin embargo, los defensores de los Métodos Ágiles (MA), afirman que se puede convertir esta curva exponencial en una curva logarítmica; expresan que *"una decisión que no ha sido tomada es evidentemente mucho más fácil de cambiar"* [5], por lo que retrasar las decisiones es una de las estrategias que permiten reducir el costo del cambio [1, 5, 31].

El MA más popular es la ***Programación Extrema*** (XP). La filosofía de XP es satisfacer por completo las necesidades del cliente, motivo por el cual se lo integra como un miembro más del equipo de desarrollo [41]. XP rechaza la idea de un análisis completo y de la comunicación del diseño a través de especificaciones textuales y diagramas UML. En lugar de eso propone conversaciones, metáforas, anotaciones en tarjetas, refactorización y código como documentación. Para la documentación de requisitos se prefiere la comunicación oral, para la documentación del diseño las tarjetas CRC (Clases, Responsabilidades y Colaboraciones), y para documentar el código se recomienda clarificarlo en lugar de comentarlo [1].

Por lo expuesto anteriormente, aparentemente con los MA, y sobre todo para aquellos que les gusta programar y, poco o nada, encargarse de la fase de requisitos y su documentación, ¡ESTÁN EN EL PARAISO! No obstante, no se puede negar el hecho que estos métodos, y en especial XP, desprecian la recolección de requisitos (se reduce a la especificación de las historias de usuario[1]) y, como se sabe, "*…los requisitos deficientes son la mayor causa de los fallos en los proyectos de software…*" [1].

XP propone la comunicación oral con el cliente y las fichas para asentar las historias de usuario; pero, en su aplicación, son notables sus falencias, tales como buena documentación y gestión de requisitos (tratamiento, control de actualizaciones y cambios). Esto provoca algunos problemas:

[1] Por lo general, las historias de usuario son fichas de papel, donde los clientes escriben, con su vocabulario, tal y como ellos ven las necesidades del sistema.

- La comunicación oral de la que se habló anteriormente, es difícil de salvaguardar en el transcurso del tiempo.
- Algunas veces los requisitos son difíciles de expresar en palabras (el lenguaje es ambiguo por lo tanto las historias son ambiguas), y esto a menudo promueve a la falta de entendimiento entre usuarios y desarrolladores.
- Pueden existir muchos tipos de requerimientos con diferentes niveles de detalle (en XP no se especifica qué información deben contener las historias de usuario, sólo se proporcionan plantillas).
- Los requerimientos están relacionados unos con otros, y a su vez se relacionan con otras partes del proceso (trazabilidad).
- Cada requerimiento tiene propiedades únicas y abarcan áreas funcionales específicas.

Debido a los problemas mencionados, para la captura de requisitos se puede utilizar la etnografía, la cual es útil para estudiar grupos pequeños donde es posible que se conozcan todos los miembros.

Por lo general, a las personas les resulta difícil escribir lo que hacen y, muchas veces, la mejor manera es observarlas en su trabajo (realizando trabajo etnográfico u observación de campo). Entonces, con la aplicación de estudios etnográficos, como técnica de análisis de requisitos, se puede mejorar:

- la descripción del contexto, del lugar de trabajo y cómo las personas realizan su trabajo;
- el estudio de las relaciones entre las personas y los objetos; y
- la información acerca de las tareas que se realizan.

Por lo tanto, el problema se puede plantear de la siguiente manera: ***"Los métodos tradicionales no son aptos para trabajar en entornos de requerimientos cambiantes; además, antes de programar es imposible pensar en todos los problemas que se podrían presentar y, a menudo, los clientes están insatisfechos con el proceso de desarrollo ya que no ven resultados hasta que el proyecto ha finalizado. Con los métodos ágiles (en especial con XP), los problemas mencionados anteriormente encuentran solución ya que hay mayor capacidad de respuesta a los cambios, el cliente nota el avance del proyecto a partir de pequeñas entregas e incluso funcionalidades que puede empezar a usar. No obstante, XP no es un método***

perfecto, ya que su principal falencia es la falta de documentación, de control y especificación de requisitos, lo cual produce problemas en el diseño y no facilita el mantenimiento. Pero, lo peor que puede pasar es no seguir ningún método, o intentar definir todo de antemano guiados por los métodos tradicionales, o bien no hacer ningún análisis de requisitos pensando que así se está aplicando un método ágil, o lo que es peor, se oscila entre un extremo y otro".

Entonces: *¿es posible diseñar una herramienta software, apta para ser usada tanto por usuarios como por desarrolladores, que apoye la captura, la especificación y el control de requisitos, para la cual se propone que la elicitación de requisitos se lleve a cabo mediante la técnica de la etnografía aplicada y la información obtenida sea incorporada (a dicha herramienta) por los desarrolladores; y para la especificación y control se desarrollará un sistema de control de versiones propio, aplicable en un entorno XP (manteniendo la agilidad), capaz de remediar las deficiencias que este método presenta?*

I.2. ANTECEDENTES

El primer antecedente a tener en cuenta es **IEEE-STD-830-1998, "Especificaciones de requisitos de software (ERS)"** [23], que establece las consideraciones para producir un buen documento ERS y la información que debe incluir como, por ejemplo, propósito, alcance, definiciones, siglas y abreviaciones, referencias, visión global, perspectiva del producto, funciones del producto, características del usuario, restricciones, dependencias, y requisitos específicos.

Es importante mencionar algunas herramientas para la gestión de requisitos de propósito general que se pueden aplicar:

- **RequisitePro:** es una herramienta centrada en documentos; almacena los requisitos asociándolos a documentos. Ayuda especialmente en el control de cambios de requisitos, con trazabilidad (característica que relaciona a los requerimientos entre sí) para especificaciones y pruebas de software. Permite el uso de Oracle sobre Unix o Windows, y también soporta SQL Server sobre Windows [12].
- **IRqA:** es una herramienta de ingeniería de requisitos especialmente diseñada para soportar el proceso completo de ingeniería de requisitos. El ciclo de especificación

incluye captura de requisitos, análisis, especificación de sistema, validación y organización de requisitos que es soportada por modelos estándares [12, 47].

- **CaliberRM:** es para sistemas grandes y complejos, y proporciona una base de datos de requisitos con trazabilidad. Está basado en Internet y maneja referencias de documentos, responsabilidades de usuario, trazabilidad y prioridad. Está diseñado para capturar y gestionar los requisitos de negocio, técnicos, funcionales, y operacionales; también permite la colaboración eficaz a través de la organización destinada a entregar los proyectos cumpliendo el tiempo, el presupuesto y las especificaciones [12, 46].
- **DOORS:** a diferencia del resto de las herramientas, considera los requisitos como objetos y los documentos como módulos. Es una herramienta para organizaciones grandes que necesitan controlar complejos conjuntos de usuarios y requisitos de sistemas con una completa trazabilidad [12].
- **Reconcile:** permite entregar los proyectos a tiempo controlando el presupuesto y las especificaciones. También ayuda a definir y perseguir las metas del proyecto a lo largo del ciclo de vida, previniendo los errores comunes que pueden causar el fracaso del proyecto. Usando esta herramienta se promueve la comunicación; además, el desarrollo del software es acelerado pero siempre manteniendo la calidad [49].

También existen herramientas para su utilización en proyectos de desarrollo XP, como por ejemplo:

- **VOLT:** en una herramienta que permite gestionar las historias de usuario a través de un navegador Web, incluyendo control de versiones. Para ello, codifica cada historia en XML y la registra en un repositorio SUBVERSION (sistema se control de versiones, del cual se hablará en apartados posteriores). También ofrece una interfaz personalizada a cada tipo de rol de usuario [36].
- **SQS-RequirementsWORKFLOW:** es la herramienta creada por SQS S.A. para la gestión de requisitos en proyectos de desarrollo de software basados en XP. Esta herramienta gestiona historias de usuario de una manera cómoda y sencilla. Puede clasificar las historias de usuario, dividirlas en diferentes niveles de detalle, asignarlas a programadores y a múltiples proyectos, planificar su implementación asignándoles recursos y hacer el seguimiento de su grado de implantación. [37].

- **XPLANNER:** es una herramienta para la planeación y el seguimiento de actividades para equipos de desarrollo que trabajen mediante el método XP. Permite hacer estimaciones de tiempo, costo, personal, y ver el avance del proyecto detectando lo antes posible las desviaciones respecto a las estimaciones [52].

El *Sistema de Control de Versiones* es un sistema capaz de gestionar las versiones facilitando la administración de las mismas, para cada producto desarrollado, junto a las posibles especializaciones realizadas para algún cliente específico [50]. Los más conocidos son:

- **CVS (Concurrent Version System):** permite el control de versiones y el control de concurrencia. Mantiene el registro de todo el trabajo, los cambios en la implementación de un proyecto, y permite que distintos desarrolladores colaboren. Se caracteriza porque varios clientes pueden sacar copias del proyecto al mismo tiempo [43].
- **SUBVERSION:** es un software diseñado específicamente para reemplazar al popular CVS. Es software libre bajo una licencia de tipo Apache/BSD. Se caracteriza porque sigue la historia de los archivos y directorios a través de copias y renombrados, las modificaciones son atómicas y maneja eficientemente archivos binarios [51].

En cuanto a la aplicación de la etnografía en la Ingeniería de Software, hay algunos precedentes:

- **AMENITIES:** es una metodología basada en modelos de comportamiento y tareas para el análisis, el diseño y el desarrollo de sistemas cooperativos. Parte de marcos teóricos cognitivos y metodológicos, permitiendo realizar un modelado conceptual del sistema cooperativo. El objetivo de la metodología es abordar de forma sistemática el análisis y el diseño del sistema cooperativo facilitando el desarrollo posterior del software. Esta metodología se ha aplicado a diversas áreas como la gestión de recursos humanos en un sistema de control de emergencias, o a la representación de la colaboración para la gestión del conocimiento compartido. El proceso de elicitación de requisitos se lleva a cabo mediante la técnica de la etnografía aplicada y los diagramas de casos de uso de UML. En primer lugar, la aplicación de la etnografía permite reunir y documentar informalmente los requisitos del sistema. A continuación, a partir de la información obtenida, se utilizan los diagramas de casos de uso para estructurar y especificar los requisitos funcionales, roles y actores en el sistema [18].

I.3. OBJETIVOS

El propósito de este proyecto es aplicar la etnografía para la captura de requisitos como técnica de análisis para obtener información, la cual se incorporará a una herramienta software (adecuada para usuarios y desarrolladores), que permita la especificación y el control de requisitos en un entorno de desarrollo XP.

OBJETIVOS GENERALES

- Apoyar a los desarrolladores de software en la captura y la gestión de requisitos en un ambiente de desarrollo XP.
- Propiciar el uso de etnografía en el desarrollo de software como técnica de elicitación de requisitos.
- Mantener los principios de los MA, sin menospreciar la recolección de requisitos.
- Facilitar el diseño y el mantenimiento del sistema.

OBJETIVOS ESPECÍFICOS

- Aplicar la técnica de etnografía para la elicitación de requisitos.
- Diseñar una herramienta software de gestión de requisitos que centralice toda la información del problema.
- Mejorar la actualización y la trazabilidad de las historias de usuario, y favorecer la comunicación entre usuarios y desarrolladores.

I.4. CARACTERÍSTICAS DE LA INVESTIGACIÓN

La presente investigación es de tipo *exploratoria* y *descriptiva*.

Es *exploratoria* porque, aún cuando existe suficiente material informativo relacionado a XP, hay pocas herramientas de gestión de requisitos en un ambiente ágil, y no se encuentran datos de la aplicación de la etnografía para la captura de requisitos en las herramientas diseñadas para ese fin.

También es *descriptiva*, ya que la descripción del fenómeno estudiado y su marco teórico se realizará en función de la información aportada a partir de distintas fuentes referentes a XP, requisitos y etnografía aplicada. Luego, toda la información recabada se relacionará de modo de justificar el fin que persigue la presente investigación. Cabe destacar que hay gran cantidad de información referida a XP, en especial en la Web (a pesar de ser una metodología que tiene unos pocos años); con respecto a la etnografía

aplicada a la captura de requisitos existen algunos antecedentes, pero es una técnica relativamente nueva en el contexto del desarrollo de software.

I.5. ALCANCES

Si se piensa en el ámbito donde se desarrollará nuestra actividad profesional, es indispensable considerar las limitaciones que se enfrentarán, ya sea de tipo económico o bien limitaciones de recursos (en lo que se refiere a nuevas tecnologías).

Los proyectos, por lo general, se caracterizan porque se exceden en tiempo, debido a nuevos requerimientos que van apareciendo o a cambios en los iniciales. El problema deviene a partir de la falta de entendimiento entre lo que el cliente quiere y lo que se entiende acerca del problema. Además, al cliente no le satisface el hecho de saber que se está trabajando en el proyecto, por ejemplo, documentando; quiere ver el avance del proyecto en entregas frecuentes, es decir, funcionalidades que pueda empezar a usar. La captura de requisitos, en los MT, es una actividad productora de documentos que especifican un sistema con el detalle suficiente como para que el desarrollo de software pueda guiarse a partir de ellos. Es importante recordar que el propósito primario de un proyecto de desarrollo es entregar un sistema de software que genera valor para el cliente (los modelos y documentos son un derivado del desarrollo, no generan el valor de negocio). Además, los documentos son selectivos y unidireccionales. Selectivos, pues los requisitos describen a menudo lo que se necesita en función a una solución deseada en vez de explicar el contexto del problema (no se habla sobre las necesidades de los usuarios). Es unidireccional, ya que no hay ninguna oportunidad para hacer preguntas respecto a lo documentado.

Por lo expuesto anteriormente, nuestro medio es un ámbito propicio para la aplicación de MA, en especial de XP, debido a que los equipos de desarrollo son pequeños, los clientes se involucran en el proyecto, los requerimientos son cambiantes y se solicitan entregas frecuentes del producto software.

Sin embargo, como ya se mencionó anteriormente, XP presenta serios problemas debido a deficiencias en cuanto a su aplicación, pues carece de una buena documentación y de una eficiente gestión de requisitos.

Por lo tanto, el alcance del trabajo de investigación será llegar al diseño de la herramienta y la construcción del prototipo de la misma. Será una herramienta software de

gestión de historias de usuario en una ambiente ágil, la cual poseerá su propio sistema de control de versiones y, además, se propone al método etnográfico para la captura de requisitos. Se espera que el uso de esta herramienta proporcione a la organización donde se la aplique y al equipo de desarrollo que la utilice:

- mejoras en la captura de requisitos a través del uso de etnografía permitiendo así realizar ahorros en el costo de especificación y minimizar el impacto de errores;
- mejoras en la calidad, permitiendo controlar y administrar más fácilmente las especificaciones;
- centralización de toda la información del problema, proporcionando una trazabilidad entre todas las especificaciones (historias de usuario).

I.6. HIPÓTESIS Y VARIABLES

"La herramienta propuesta de gestión de historias de usuario, colabora con usuarios y desarrolladores a lograr una adecuada captura, especificación y control de requisitos, manteniendo los principios de los MA".

Tabla 1.1. Operalización de Variables

Tipo de Variable	Nombre de la Variable	Definición Conceptual
Independiente	Herramienta de gestión de historias de usuario	Herramienta que permite la captura de requisitos a través de etnografía aplicada, el control y la actualización de las versiones de las historias de usuario.
Dependiente	Adecuada captura, especificación y control de requisitos, manteniendo los principios de los MA	Información obtenida de una eficiente captura de requisitos (aplicando etnografía), y que permita la especificación, el control y la actualización de las historias de usuario. De esta manera, se obtiene información detallada de los requerimientos, la cual es necesaria para realizar determinadas tareas o funciones.
Unidad de Análisis o de Observación	Usuarios y desarrolladores	Los usuarios son aquellos que se encargarán de escribir las distintas versiones de las historias; y los desarrolladores son los que integran el equipo y se encargarán de incorporar, a la herramienta, la información obtenida a través de la etnografía aplicada, y utilizar dicha información y las distintas versiones de las historias para el diseño y posterior mantenimiento.

Para corroborar, o refutar, la hipótesis se procederá a construir un prototipo de la herramienta propuesta que se utilizará, en principio, durante el desarrollo de dos sistemas en un entorno XP, en una organización del medio. Con la aplicación se espera conseguir las bases que guiarán el diseño y facilitarán el mantenimiento del sistema.

Los resultados de la aplicación del prototipo se someterán a métricas de la calidad de la especificación, las cuales proponen una lista de características que pueden emplearse para valorar la calidad de la especificación de requisitos: ausencia de ambigüedad, compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización.

CAPÍTULO II

MARCOS REFERENCIALES

En este capítulo, se presenta los marcos referenciales: teórico, metodológico y empírico.

En el marco *teórico* se expresan las teorías y conceptos; los cuales permiten obtener conocimientos sólidos que permitan describir, comprender y explicar el problema a investigar. En este marco se incluyen y definen todos términos necesarios, con el propósito que el lenguaje empleado en el informe sea claro y preciso. Para ello se utilizó el material bibliográfico obtenido luego de un proceso de clasificación y de selección, consiguiendo así las bases para la construcción de dicho marco.

En el marco *metodológico* se hace referencia a la metodología a utilizar; es decir se presenta la metodología que se utilizará tanto en la definición de los lineamientos a considerar a la hora de aplicar la etnografía en la captura de requisitos, como en el diseño de la herramienta de gestión propuesta.

Finalmente, en el marco *empírico* se presentan los sistemas que serán sometidos a las recomendaciones de aplicación de etnografía durante la captura de requisitos y a la herramienta propuesta, permitiendo así, corroborar la hipótesis planteada en el capítulo anterior.

II.1. MARCO TEÓRICO

II.1.1. REQUISITOS

II.1.1.1. Definición de Ingeniería de Requisitos

El IEEE en el "Glosario de Terminologías de Ingeniería de Software", define a un requisito como: "*(1) una condición o capacidad que necesita un usuario para resolver un problema o para cumplir un objetivo; (2) una condición o capacidad que debe tener un*

sistema o componente para cumplir un contrato, estándar, especificación o cualquier otro documento impuesto formalmente; (3) una representación documentada de una condición o capacidad como en (1) o (2)" [22].

Algunas definiciones de Ingeniería de Requisitos son se las siguientes [33]:

- es un proceso de descubrimiento, refinamiento, modelado y especificación; en el cual se refinan detalles los requisitos del sistema, se crean modelos de los requisitos de datos, flujo de información y control, y del comportamiento operativo.
- es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un costo reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga las necesidades del usuario.

II.1.1.2. Tipos de Requisitos [2,13, 21, 22, 23, 33, 45]

Los requisitos pueden ser clasificados como:

- *Funcionales*: describen la funcionalidad o servicios que se espera que el sistema provea (entradas, salidas, etc.). Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.
- *No Funcionales*: describen propiedades como la fiabilidad, el tiempo de respuesta, mantenibilidad, sistema operativo, plataforma hardware, etc. También describen restricciones de cómo se implementarán los requisitos funcionales.

II.1.1.3. Análisis de Requisitos

En la fase de Análisis de Requisitos se estudia el dominio del problema interactuando constantemente con clientes y usuarios para detectar información sobre sus verdaderas necesidades. La clave es la comunicación con el cliente. Para facilitar esta comunicación se han desarrollado varias técnicas. Las técnicas más comunes en la captura de requisitos son las entrevistas, el Desarrollo Conjunto de Aplicaciones (JAD), la tormenta de ideas, la utilización de escenarios o casos de uso y la etnografía como técnicas de adquisición. A estas técnicas las suelen acompañar la observación, el estudio de documentación, los cuestionarios, la inmersión en el negocio del cliente o haciendo que los ingenieros de requisitos sean aprendices del cliente. A continuación se describen brevemente las técnicas [2, 13]:

- *Entrevistas*: constituyen la técnica de recolección de requisitos más usada y son prácticamente inevitables en cualquier proyecto de desarrollo. En ellas no conviene improvisar por lo cual es necesario conocer el entorno del usuario para poder entender sus necesidades y ganar así su confianza en cuanto a que el ingeniero de requisitos entiende sus problemas. También se debe seleccionar a las personas a las que se va a entrevistar para disminuir el número de entrevistas a realizar. Además, para minimizar el tiempo de la entrevista es fundamental fijar el objetivo que se pretende alcanzar y determinar previamente su contenido.
- *JAD*: se desarrolla a lo largo de un conjunto de reuniones en grupo. En estas reuniones se ayuda a los clientes y usuarios a formular problemas y explorar posibles soluciones, involucrándolos y haciéndolos sentirse partícipes del desarrollo. Tiene dos grandes pasos, el JAD/Plan cuyo objetivo es elicitar y especificar requisitos, y el JAD/Design, en el que se aborda el diseño del software.
- *Tormenta de Ideas*: es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Esta técnica ayuda a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de elicitación, cuando los requisitos son todavía muy vagos.
- *Utilización de escenarios o casos de uso*: son una técnica para la especificación de requisitos funcionales que forma parte de la propuesta de UML. Un caso de uso es la descripción de una secuencia de interacciones entre el sistema y uno o más actores (personas u otros sistemas que interactúan con el sistema cuyos requisitos se están describiendo). La ventaja que tiene esta técnica sobre la descripción puramente textual de los requisitos funcionales es que facilitan la elicitación de requisitos y son fácilmente comprensibles por los clientes y usuarios. Algunos autores proponen esta técnica tanto como de elicitación, como de especificación de los requisitos funcionales del sistema.
- *La etnografía*: permite hacer adecuadas interpretaciones de sucesos, acciones, individuos y roles para tener en cuenta sus significados y transmitirlos en la interfaz. En definitiva, constituye una técnica adecuada y altamente recomendable para realizar la actividad de observación de campo. Realizar un estudio etnográfico ayuda en la fase del Análisis de Requisitos,

puesto que se podrá detallar y relacionar los objetos que las personas involucradas utilizan (directa o indirectamente), aumentar la información relativa a la organización de las tareas y a su consecución (la experiencia del día a día de los trabajadores aporta gran información debido a que cada uno ve una misma tarea de diferente manera). Se profundizará sobre esta técnica en apartados posteriores.

II.1.1.4. Especificación de los Requerimientos del Sistema

La especificación de requisitos puede definirse como: *"un conjunto o colección estructurada de información que contiene los requerimientos del sistema"* [21].

"*… tradicionalmente ha sido vista como un documento que comunica los requerimientos del cliente a la comunidad técnica que especificarán y construirán el sistema. La colección de requerimientos que constituyen la especificación y su representación actúan como el puente entre los dos grupos y debe ser entendible tanto por el cliente como por la comunidad técnica. Una de las tareas más difíciles en la creación de un sistema, es aquella de comunicar a todos los subgrupos, especialmente en un solo documento…"* [21].

El conjunto de requerimientos debería tener las siguientes propiedades:

- Cada requerimiento debe declararse sólo una vez.
- Estar normalizado (es decir, no se deben referir a otros requerimientos ni a las capacidades de otros requerimientos).
- Se deben definir explícitamente las relaciones entre los requerimientos individuales para mostrar cómo están relacionados para formar el sistema completo (interdependientes).
- La especificación de requerimientos debe incluir todos los requerimientos dados por el cliente, y todos los requerimientos necesarios para la definición del sistema.
- La especificación de requerimientos debe ser consistente y sin contradicciones en el nivel de detalle, estilo de la declaración de los requerimientos y en la presentación del material.
- Deben identificarse los límites, el alcance y el contexto de los requerimientos del sistema.
- La especificación de requerimientos debería ser modificable.

II.1.2. MÉTODOS ÁGILES

Los MT definen un proceso secuencial, donde cada proceso depende del anterior. Imponen un proceso disciplinado sobre desarrollo de software con el fin de hacerlo más predecible y eficiente. Se basan en documentar todo el proceso para minimizar el riesgo de cada proyecto y controlar así su evolución [18].

Sin embargo, presentan algunos inconvenientes:

- El problema ha cambiado y el software no se ha adaptado.
- El software no resuelve los requisitos planificados al principio (requerimientos cambiantes o mal comprendidos).
- Es imposible pensar en todos los problemas que se presentan al programar; es decir que no fueron tenidos en cuenta durante el diseño.
- El cliente está insatisfecho porque no ve resultados.

El continuo cambio en los requerimientos funcionales genera costos elevados al cliente, quien en la mayoría de las situaciones no percibe la dificultad que implica rediseñar el sistema. Estos cambios o nuevas mejoras en el sistema pueden generar que un proyecto se atrase o, en el peor de los casos, que falle.

Debido a estos problemas y a la necesidad de tener un modo eficiente que guíe el proceso de desarrollo cuando el entorno es cambiante y los requisitos no se conocen con exactitud, surgieron los MA destinados a equipos de desarrollo pequeños. Por dicha razón en el 2001, se creó la organización “The Agile Alliance” sin fines de lucro, dedicada a promover los conceptos del desarrollo ágil. En la misma fecha surgió el Manifiesto Ágil de Desarrollo, el cual incluye una serie de principios de desarrollo ágil que lo diferencian de uno tradicional [1, 29, 31]:

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
- Dar la bienvenida a los cambios.
- Realizar frecuentemente entregas de software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- Los clientes y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener un paso constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad. Producir código claro y robusto es la clave para avanzar más rápidamente en el proyecto.
- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y entonces ajusta su comportamiento.

La diferencia principal entre estos métodos y los tradicionales, es que los primeros se preocupan más en la capacidad de respuesta a los cambios antes de seguir un plan, son menos burocráticos, están orientados a la gente y no al proceso.

Entre los MA más difundidos están: Extreme Programming (de Kent Beck), Crystal Clear (de Alistair Cockburn), SCRUM, DSDM (Dynamic System Development Method), FDD (Feature Driven Development), Adaptative Software Development (de Jim Highsmith).

XP fomenta cuatro valores, ellos son, la simplicidad (de las soluciones), la valentía (para hacer frente a los cambios), la retroalimentación (entre el cliente y los desarrolladores) y la comunicación (entre los participantes del proyecto).

II.1.2.1. Programación Extrema (XP)

XP se basa en ir analizando, diseñando e implementando pequeñas piezas de código. Esto permite mayor flexibilidad en el caso que se requiera cambios en la funcionalidad.

Entre todos los MA, éste es el que ha cobrado mayor notoriedad en los últimos tiempos. Sus objetivos son: potenciar al máximo el trabajo en grupo y tratar de conseguir un desarrollo acelerado, flexible y cambiante, con entregas frecuentes y evaluaciones constantes por parte del cliente, con el fin de proporcionarle el software

que necesita y cuando lo necesita, es decir se debe responder muy rápido a las necesidades del cliente.

Este método persigue la satisfacción del cliente. El mayor fracaso de los MT ha sido, y es, el no mantener contento al cliente. El cliente se impacienta porque no ve avances en el proyecto, el jefe de proyecto se desespera, los analistas y programadores no pueden tener a tiempo los cambios requeridos [9].

II.1.2.2. Roles en XP [9, 29]

- *Programador*: es el encargado de escribir las pruebas y el código del sistema. Es el responsable de las decisiones técnicas. Un programador XP se caracteriza por poner mayor énfasis en la comunicación con otras personas; escribe continuamente pruebas para demostrar aspectos vitales del software; descompone los programas en piezas más pequeñas y busca la coherencia entre las piezas. Siendo un programador XP se tiene mayor capacidad de observación que siendo un programador de otra disciplina.
- *Cliente*: es parte del equipo de proyecto, es el encargado de determinar qué construir y cuándo. Establece las pruebas de aceptación. Para poder llevar a cabo su trabajo debe aprender a hacer cosas nuevas (como escribir historias de usuarios) y estar preparado para afrontar posibles cambios durante todo el desarrollo.
- *Encargado de pruebas*: ayuda al cliente con las pruebas de aceptación y se asegura de que las pruebas aceptación se superen. Se encarga de transmitir los resultados y se asegura del adecuado funcionamiento de las herramientas de prueba.
- *Encargado de Seguimiento*: se encarga de mantener los datos históricos. También de ocupa de verificar el grado de acierto entre las estimaciones y el tiempo real dedicado. Evalúa si los objetivos son alcanzables con las restricciones de tiempo y costo.
- *Entrenador (coach)*: es el responsable del proceso. Una cualidad muy importante del coach es su capacidad para trabajar indirectamente. Es decir, cuando detecta un error de diseño, debe medir su impacto y decidir hasta que punto intervenir y la sutileza con que debe hacerlo. Se encarga de trabajar con el equipo de desarrollo para ayudar a reorganizar los roles y las responsabilidades.

- *Gestor*: es el encargado de organizar y guiar las reuniones; es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje eficientemente creando las condiciones adecuadas. Para ello, debe esmerarse en cuidar un clima de comunicación honesta con el equipo de desarrollo.
- *Consultor*: cuando ocurren problemas que se necesita solucionar, referidos a algún aspecto técnico en particular, entonces se hace imprescindible que el equipo recurra a un consultor, el cual es un miembro externo con algún conocimiento específico necesario para el proyecto.

II.1.2.3. Prácticas comunes en XP [1, 7, 9, 15, 27, 29, 37, 48]

a) El Juego de la Planificación

No se trata de una etapa de planificación detallada; en realidad, da comienzo a una planificación general de lo que se piensa hacer para llevar a cabo el proyecto. Se realiza una estimación del esfuerzo requerido para implementar las historias de usuario (técnica utilizada en XP para especificar los requisitos del software), se decide sobre el alcance y el tiempo de las entregas, y de las iteraciones. También, se ordenan las historias según prioridad y esfuerzo asociado. Además, se determina el alcance de la próxima entrega, combinando las prioridades de negocios con los estimados técnicos. Cuando la realidad sobrepasa el plan, se tendrá que adaptar el plan.

b) *Entregas Pequeñas*

Cada versión debe ser lo más pequeña que se pueda, conteniendo los requerimientos más importantes, es decir, deben ser operativas aunque no cuenten con toda la funcionalidad del sistema. De esta manera, cada versión proporciona el código para un pequeño conjunto de funciones. Las versiones deberían ser entregadas cada dos o tres semanas de manera que los clientes puedan ver y tocar el producto funcionando de manera regular, una entrega no debería tardar más de tres meses.

c) Metáfora

Sirve para guiar todo el desarrollo, pues es una historia compartida entre el cliente y el equipo de desarrollo que describe cómo debería funcionar el sistema. Se originó a partir de la propuesta de Ward Cunningham de un sistema de nombres, *"… la metáfora consiste en formar un conjunto de*

nombres que actúen como vocabulario para hablar sobre el dominio del problema… ". Las metáforas ayudan a cualquier persona a entender el objeto del sistema es decir, indica cómo trabaja (o como debería trabajar) el mismo.

d) Diseño Simple

Las soluciones complejas a pesar de que sean buenas, llegan a tomarse un tiempo de diseño bastante grande, comparado con los resultados que retornan. Es por ello que en XP se brega por un diseño simple, puesto que los tiempos de entrega no se pueden prolongar; después de la entrega se puede perfeccionar lo que ya funciona. "*En cualquier momento el diseño adecuado para el software es aquel que supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la invención de implementación de los programadores y tiene el menor número posible de clases y métodos"* [4]. Por lo tanto, la complejidad innecesaria y el código duplicado debería removerse; a menos código, menos errores.

e) Pruebas

Las pruebas unitarias se establecen antes de escribir el código, y se ejecutan antes de cada modificación del sistema. Es decir, las pruebas deben ir a la par de las entregas del proyecto, nunca más adelante ni detrás de las mismas. Los desarrolladores continuamente escriben las pruebas unitarias, las cuales deben ejecutarse sin error para que el desarrollo pueda continuar. Cuando se detecta un error en una corrida, su reparación pasa a ser la máxima prioridad para el equipo de desarrollo. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse, de esta manera prueban qué funcionalidades están terminadas de acuerdo a sus expectativas.

f) *Refactorización*

Consiste en llevar a cabo modificaciones en el código de la aplicación sin modificar su comportamiento; es decir, no es una técnica para encontrar y corregir errores de la aplicación, su objetivo es mantener el código sencillo y bien estructurado. Refactorizar propone seguir las técnicas matemáticas que consiguen reducir fórmulas muy complejas en fórmulas equivalentes pero más sencillas. Un código simple es aquel que funciona, no hay código duplicado, el código permite entender el diseño y minimiza el número de clases y métodos. La particularidad de esta técnica consiste en aplicar una serie de pequeños

cambios en el código manteniendo su comportamiento. Esta práctica no sólo sirve para mantener el código legible y sencillo, como se dijo anteriormente, sino también puede utilizarse cuando resulta conveniente modificar código existente para hacer más fácil implementar una nueva funcionalidad. Según Martin Fowler [16] *"la refactorización es realizar modificaciones en el código con el fin de mejorar su estructura interna, sin alterar su comportamiento externo"*. Por lo tanto, es un medio para mantener el diseño lo más sencillo posible y de calidad.

g) *Programación en parejas*

No está bien vista ni por los programadores ni por las empresas, los primeros prefieren programar cada uno en su propia computadora, mientras los segundos consideran que un programador de la pareja está ocioso. La ventaja de esta práctica es que muchos errores se detectan cuando se introducen en el código ("*cuatro ojos ven mejor que dos"*). Se posibilita la transferencia de conocimientos de programación entre los miembros del equipo, más personas conocen de distintas partes del sistema, el código siempre está siendo revisado por otra persona (la revisión de código es el método más eficaz de conseguir código de calidad), y además todas las decisiones son tomadas al menos por dos personas proporcionando un mecanismo de seguridad valioso. Se trata de dos personas compartiendo el mismo monitor y teclado, es decir que hay uno que codifica y su compañero se encargara de pensar en forma más estratégica, como por ejemplo, ¿qué es lo que podría fallar?, ¿qué deberíamos comprobar en las pruebas? o ¿hay alguna manera de simplificar el sistema?; estos roles son intercambiables. Del mismo modo, la composición de la pareja puede cambiar si alguno de sus integrantes es solicitado por otro miembro del equipo para que le ayude con su código. *"El mejor método de desarrollo por parejas consiste en sentarse uno junto al otro, compartiendo ordenador* (la computadora), *y dejar que uno se encargue del desarrollo mientras el otro piensa en la mejor forma de afrontar los problemas. Esta actividad debería pasar de uno a otro periódicamente"*.

h) *Propiedad colectiva del código*

Esta práctica evita que un programador sea imprescindible para realizar cambios en una parte del código, ya que cualquier programador puede

cambiar cualquier parte del código en cualquier momento. Esta práctica es diferente a los métodos tradicionales en los que un programador posee un conjunto de códigos. Los defensores de XP sostienen que de esta manera, realizar cambios es un esfuerzo de equipo, y no significa una sobrecarga de trabajo individual para un solo programador. Si el código tiene un solo dueño, entonces, para realizar un cambio es necesario su autor; si se requiere hacer un cambio sobre una parte del código se la hace y listo.

i) *Integración continua*

Cada pieza de código es integrada al sistema cuando esté lista. Es decir, un sistema puede ser integrado varias veces al día. La integración no es una prueba, pero para poder integrar una parte del código al sistema es necesario que pase todas las pruebas unitarias. Esta práctica favorece a la comunicación entre los desarrolladores y permite que el código pueda reutilizarse y compartirse. Una pareja de programadores se encargara de integrar todo el código en una maquina, llamada de integración, y realizar todas las pruebas hasta que estas funcionen al 100%. Si al añadir un bloque nuevo junto con todas sus pruebas unitarias, el sistema completo sigue funcionando correctamente, entonces pasa la prueba y los programadores darán por finalizada esa tarea. Si no, se encargaran de dejar el sistema de nuevo con las pruebas funcionando al 100%. La integración poco frecuente trae serios problemas a un proyecto de software, la integración es crítica para lograr un código que funcione bien.

j) *40 horas semanales*

En XP se debe trabajar manteniendo el equilibrio de 40 horas semanales. Está bien trabajar tiempo extra cuando se requiere, pero no hay que hacerlo dos semanas seguidas, el trabajo extra desmoraliza al equipo, por lo tanto, habiendo descansado, los desarrolladores motivados aceleran la productividad. Además, si el proyecto requiere trabajos extras para tratar de cumplir con los plazos, probablemente está ocurriendo un problema que debe corregirse. Esta práctica está dirigida a un entorno laboral, se basa en el hecho de que la capacidad creativa no se puede alargar más allá de un cierto tiempo. El desarrollador necesita una serie de condiciones para poder ejercerla en su máxima expresión y una de ellas es que el ambiente laboral sea agradable.

k) Cliente in-situ

El cliente debe estar disponible full-time para el equipo. La comunicación oral es más efectiva que la escrita, ya que esta última toma mucho más tiempo en generarse y corre mayor riesgo de ser mal interpretada. Como es el cliente el que conduce el trabajo, el hecho de que este presente siempre, permite a que los programadores pueden resolver de manera inmediata cualquier duda que surja. El cliente asignado para esta tarea, es una persona que trabaja con el equipo y está disponible para responder preguntas, resolver asuntos y establecer prioridades. Sin embargo, esta práctica presenta un problema, ya que es difícil que el cliente ceda una persona que conozca el negocio para que participe del equipo, entonces será responsabilidad del equipo de desarrollo hacerles ver que será mejor para su negocio tener un software pronto en funcionamiento que cubra sus necesidades, y esto no implica que el cliente no pueda realizar otro trabajo.

l) Estándares de codificación

Para lograr la comunicación entre los programadores a través del código, es necesario que se sigan estándares de programación. Si los programadores van a estar tocando partes distintas del sistema, intercambiando parejas, refactorizando, entonces se debe establecer un estándar de codificación, el cual debe ser aceptado e implantado por todo el equipo. Esto mantiene el código legible para los miembros del equipo, facilitando así los cambios. Es decir, que en el equipo de desarrollo todos los programadores deben escribir de una manera parecida, de modo que todos lo entiendan, y si ingresa un nuevo programador al equipo pueda entender rápidamente el trabajo ya realizado.

Estas prácticas no son exclusivas de XP, sino que ya existían. XP lo que se hizo fue unirlas, de manera de aplicarlas juntas, integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

Se puede decir que son prácticas complementarias entre sí y su correcto funcionamiento se produce a partir de la aplicación de todas ellas; es decir, que el mayor beneficio se logra con su aplicación conjunta y equilibrada ya que unas se apoyan en otras. No es recomendable seguir algunas de ellas y otras no, ya que cada práctica

soporta a las otras, las debilidades de una son subsanadas con las fortalezas de otras (Ver Figura 2.1).

"*Ninguna práctica funciona bien por sí sola (con la excepción de las pruebas). Requieren de las otras prácticas para equilibrarse*" [7, 8].

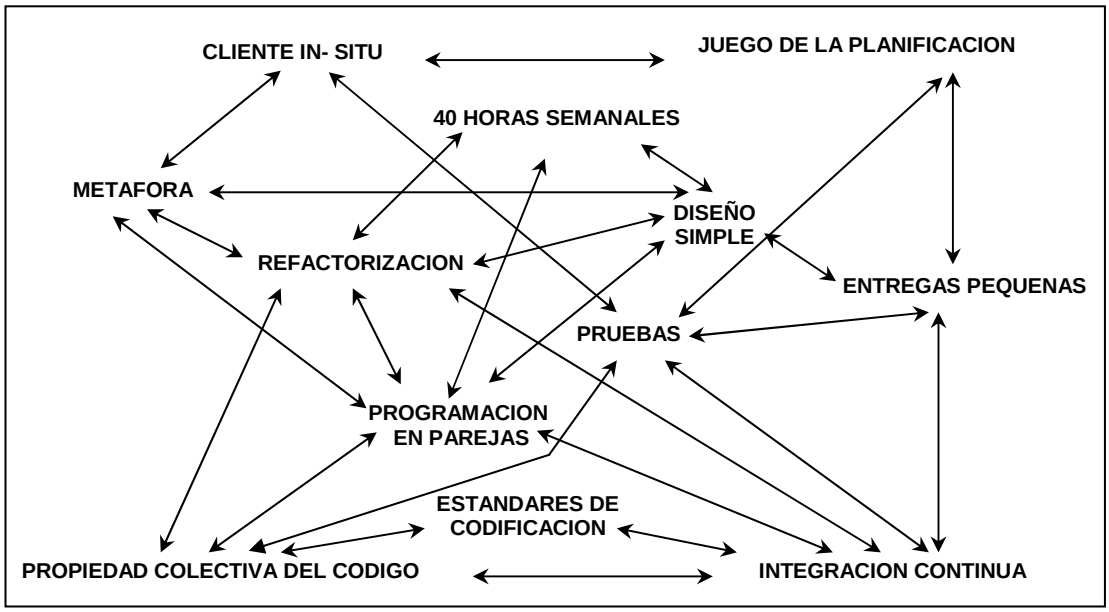

Figura 2.1. Prácticas comunes en XP [4]

En la figura anterior, una línea entre dos prácticas significa que éstas se refuerzan entre sí.

II.1.2.4. Historias de Usuario [7, 8, 9, 29]

El objetivo de las historias de usuario es guiar al cliente para escribir en detalle sus requerimientos. Las escriben los propios clientes, tal y como ellos ven las necesidades del sistema. Por lo tanto, serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica.

Las historias de usuario sirven para crear el plan estimado de entregas. Además, servirán para definir las pruebas de aceptación, las cuales se realizarán para verificar que las historias se están implementando correctamente.

El nivel de detalle de las historias de usuario debe ser el mínimo posible que permita hacerse una ligera idea de cuánto costará implementar el sistema. Cuando se

llegue a la fase de implementación, los desarrolladores podrán acudir al cliente para ampliar detalles.

Las historias de usuarios constituyen el núcleo de la planificación en XP. Tienen el mismo propósito que los casos de uso, pero no son iguales. Como se dijo anteriormente, se utilizan para calcular el tiempo estimado de las reuniones de planificación y el tiempo estimado de entrega.

Por lo expuesto, entonces, las prácticas más comunes que se realizan para generar las historias de usuario son:

- Reunión del coordinador del proyecto con los representantes del cliente. El cliente escribirá sus requerimientos de forma detallada en tarjetas, donde asentará lo que pretende del sistema. Como están hechas por el cliente, se debe evitar que las tarjetas contengan detalles específicos de tecnología o algoritmos, sólo se debe manejar el lenguaje del cliente. Cada tarjeta contiene una historia de usuario.
- Luego se realiza una revisión de las historias de usuario creadas, donde el cliente debe colocar la prioridad de la historia en cada tarjeta.
- El coordinador del proyecto realiza un estimado del tiempo de realización cada historia de usuario y lo coloca en la misma tarjeta.
- El cliente describe para cada historia de usuario uno o varios escenarios de prueba (aceptación).

En cuanto a la información que debe contener una tarjeta, aun no existe un acuerdo, sin embargo hay algunas plantillas diseñadas para ese fin, como la que se presenta a continuación en la Figura 2.2

TARJETA DE HISTORIA DE USUARIO			
Fecha: ……..	**Tipo de Actividad:** ……		
Nro. de Historia:…….	**Prioridad del Cliente:** ……	**Prioridad Técnica:** ……	
Referencia a otra historia: ……..		**Riesgo:** ……	
Descripción de la historia:			
Notas:			
Fecha	**Estado**	**Hacer**	**Comentarios**

Figura 2.2. Tarjeta de Historia de Usuario [4]

Las historias tienen dos partes esenciales: la tarjeta (implica una conversación) y la confirmación [42]:

- *Las tarjetas:* es un medio de comunicación y un mecanismo del almacenamiento. Aunque la conversación proporciona un medio de comunicación rico, las palabras pueden perderse, por lo tanto es importante registrarlas. Como se dijo antes, no hay ningún formulario formal para una tarjeta de la historia. Cada día, el equipo celebra una reunión alrededor del plan para repasar el trabajo que se realizará durante la semana. La tarjeta de historia no constituye un documento de requisitos en el sentido tradicional. La fuente principal de información durante el desarrollo es la conversación
- *La confirmación:* la primera tarea es crear un juego de pruebas que clarifican el alcance de la historia. Estas pruebas se usan para confirmar que la historia se ha llevado a cabo como se esperaba.

II.1.2.5. Ciclo de Vida XP [29, 30, 37]

XP se caracteriza por la entrega de versiones pequeñas que contienen requerimientos, que son operativas pero no cuentan con toda la funcionalidad del sistema. La entrega de versiones permite a los clientes ver y tocar el producto funcionando de manera regular. Se espera que cuanto más rápido se le entreguen las versiones funcionando al cliente, más retroalimentación se va a obtener y esto va a permitir lograr mejor calidad del producto a largo plazo.

Dependiendo de los autores, el ciclo de vida de XP, puede tener cinco o seis fases, en el presente proyecto de investigación se opta por definir el ciclo de vida en cinco fases: exploración, planeamiento, producción, mantenimiento y muerte (Ver Figura 2.3). En todas estas fases, debe estar presente un representante del cliente (cliente in-situ), es decir que participe en todas las reuniones del equipo.

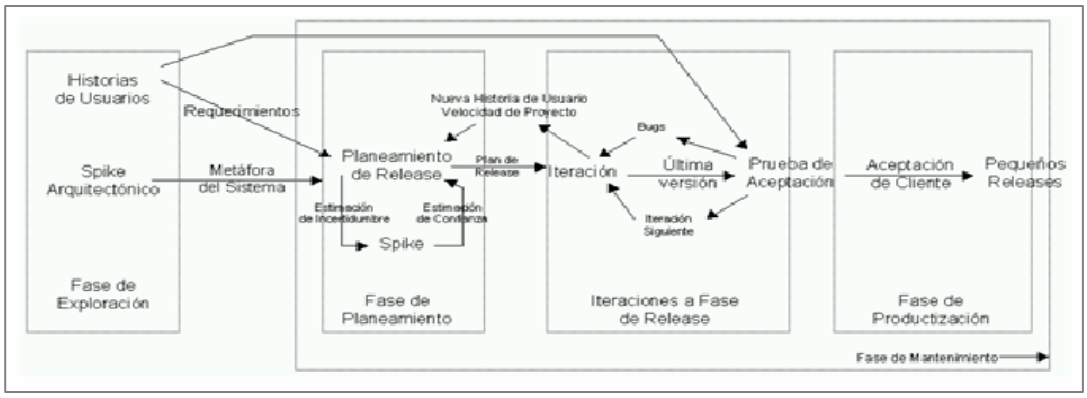

Figura 2.3. Ciclo de Vida XP [27]

Las siguientes son las fases del ciclo de vida XP:

1. *Exploración*: los clientes escriben en las tarjetas de historia de usuario lo que desean sea incluido en cada entrega (release). A las historias las escriben los propios clientes, son descripciones cortas y escritas en el lenguaje del usuario, e incluyen alguna característica que se agregará en el programa. Durante esta fase equipo del proyecto se habitúa con las herramientas y la tecnología existente. Esta fase puede tomar algunas semanas, o incluso algunos meses.
2. *Planeamiento:* el cliente establece la prioridad de cada historia de usuario y los programadores realizan una estimación del esfuerzo asociado a cada una de ellas, entonces se define el cronograma. Las historias de usuario dividen el proyecto en iteraciones. A partir de estas iteraciones se harán versiones pequeñas del proyecto. La fase de planeamiento toma un par de días.

   El plan de entrega está compuesto por iteraciones cuya duración es de una a cuatro semanas en ejecución. Con la primera iteración se trata de establecer la arquitectura del sistema. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. Cuando termina la última iteración el sistema está listo para la siguiente fase.
3. *Producción*: se realizan pruebas del funcionamiento del sistema, antes de que este sea llevado al entorno del cliente. Si se producen cambios, entonces debe decidirse si se incluyen o no en la actual entrega.

   Una vez realizada la primera entrega exitosa para uso del cliente, el proyecto debe mantener el funcionamiento del sistema mientras se realizan nuevas iteraciones.
4. *Mantenimiento*: cuando la primera versión se encuentra en producción, se debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. La velocidad del desarrollo puede disminuir después de que el sistema esté en producción. Durante esta fase se puede requerir la incorporación de nueva gente y cambiar la estructura del equipo.
5. *Muerte*: puede ocurrir por dos razones o porque el cliente no tiene más historias para que se desarrollen, es decir el sistema satisface las necesidades del cliente respecto al funcionamiento y calidad del producto software obtenido; o bien, puede suceder que el sistema no satisface los resultados esperados o es demasiado costoso para el desarrollo.

II.1.3. ETNOGRAFÍA

La etnografía (del griego: ethnos (εθνος) - tribu, pueblo, grapho (γραφω) - literalmente descripción de los pueblos) es un término derivado de la antropología que puede considerarse también como un método de trabajo de ésta; se traduce, etimológicamente, como el estudio de las etnias y significa el análisis del modo de vida de una raza o grupo de individuos, mediante la observación y descripción de lo que la gente hace, cómo se comportan y cómo interactúan entre sí, para describir sus creencias, valores, motivaciones, perspectivas y cómo éstos pueden variar en diferentes momentos y circunstancias; entonces podríamos decir que describe las múltiples formas de vida de los seres humanos.

Realizar un análisis etnográfico implica utilizar lo que se conoce como "método etnográfico", mediante el cual se realiza una investigación por medio de la observación contextual. La etnografía es un método de investigación, donde el investigador intenta entrar en la cultura de un grupo particular. Por lo tanto, para *"hacer etnografía"* es necesario introducirse en el grupo, aprender su lenguaje, sus costumbres, haciendo una observación participativa. En fin, se debe ingresar al grupo o cultura a estudiar hasta el punto en que se llegue a confundir al etnógrafo como un miembro mismo de la comunidad. No sólo se trata de observar los datos externos, sino de analizar los puntos de vista de los sujetos y las condiciones histórico-sociales en las que se dan. Por lo expuesto, una investigación de este tipo (etnográfica) consiste de una parte práctica y una teórica. La parte práctica trata de recoger e identificar los datos, la teórica consiste en reflejar e interpretar los datos de los problemas que se investigaron [53].

Los principios básicos de la Etnografía [53]

- Los estudios deben ser a personas reales y a sus actividades, operando en su medio.
- La duración puede ser prolongada, debido a que los etnógrafos pueden no tener una idea clara de lo que encontrarán, o el dominio puede ser muy técnico.
- Entiende el mundo desde el punto de vista de aquellos que lo habitan.

II.1.3.1. Etnografía y el Desarrollo del Software

Lo que se consigue con un análisis etnográfico, en el contexto del desarrollo de software, es realizar adecuadas interpretaciones de los sucesos, acciones, individuos y

roles para tener en cuenta sus significados y transmitirlos en la interfaz. Cabe destacar que, en los últimos años, muchas compañías norteamericanas están reclutando antropólogos para comprender mejor a sus clientes y a sus trabajadores y para definir los requisitos necesarios que conduzcan a diseñar productos que reflejen mejor las tendencias culturales emergentes. Es decir, la etnografía contribuye a la comprensión de los usuarios, sus contextos y sus aptitudes, habilidades y motivaciones.

Concretamente, con la aplicación de estudios etnográficos como técnica utilizada en el análisis de requisitos durante el desarrollo de sistemas, se conseguirá [18]:

- Describir el contexto, el lugar de trabajo y cómo las personas realizan sus tareas.
- Detallar y entender las relaciones entre las personas y los objetos que dichas personas, directa o indirectamente, utilizan.
- Aumentar la información relativa a la organización de las tareas y a su realización. La experiencia del "día a día" de los trabajadores aporta gran información debido a que cada uno ve una misma tarea de diferente manera.
- Requisitos funcionales relacionados con las prácticas de trabajo, recursos, etc.
- Aumentar cualitativa y cuantitativamente las conclusiones obtenidas en una observación de campo.
- Cuestiones socioculturales que pueden afectar a la interacción y la comunicación entre participantes (uso de artefactos tales como videoconfencia por computadora, etc.).

II.2. MARCO METODOLÓGICO

Las pautas que se seguirán para facilitar la generación de datos útiles, a partir de la aplicación de etnografía en la etapa de requisitos, son las siguientes [6]:

- *Preparación*: consiste en entender las políticas de la organización y la cultura de trabajo, familiarizarse con el sistema y su historia, y fijar las metas.
- *Estudio de campo:* se establece la relación con gerentes y usuarios, se observa a los usuarios en su lugar de trabajo.
- *Análisis:* se recopilan e interpretan los datos reunidos, se cuantifican los datos y se realizan estadísticas.
- *Reporte:* se prepara un informe y se presentan los resultados.

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema software. Para diseñar la herramienta propuesta se utilizará el Proceso Unificado de Desarrollo, el cual es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software. Este proceso está basado en componentes interconectados a través de interfaces bien definidas. Se caracteriza por utilizar casos de uso, estar centrado en la arquitectura y ser iterativo e incremental. Se utiliza UML para preparar todos los esquemas de un sistema, es decir, para visualizar, especificar, construir y documentar los artefactos de un sistema [25, 26].

El proceso se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo termina con una versión del producto y consta de cuatro fases, cada fase se subdivide a su vez en iteraciones, y cada iteración pasa por cinco flujos de trabajo (requisitos, análisis, diseño, implementación y prueba) [22].

- *Fase de Inicio:* se realiza una descripción del producto final, se identifican y priorizan los riesgos, se planifica en detalle la fase de elaboración y se estima la duración del proyecto.
- *Fase de Elaboración:* se especifica en detalle los casos de uso del producto y se diseña la arquitectura del sistema.
- *Fase de Construcción:* se crea el producto.
- *Fase de Transición:* los desarrolladores corrigen los problemas e incorporan algunas mejoras.

II.3. MARCO EMPÌRICO

Se construirá un prototipo de la herramienta propuesta, para lo cual se empleará C++ Builder 6.0. Como sistema de gestión de base de datos, se utilizará Interbase 6.0 debido a que genera bases de datos bastante estables y ha demostrado ser eficiente en el manejo de colisiones, además, ha sido probada en un sistema de control de stock y facturación durante doce meses, demostrando ser muy confiable.

Una vez desarrollado el prototipo, se lo empleará durante el desarrollo, en un entorno XP, de dos sistemas en organizaciones del medio, ellos son: "*Sistema de*

Control de Deposito de Materiales Eléctricos" y *"Sistema de Inscripción y Puntajes en Tiro a la Hélice"*.

- *Sistema de Control de Deposito de Materiales Eléctricos:* las funciones principales son control de usuarios, log, alta, bajas y modificaciones de productos, entrada y salida de materiales, y reportes mensuales de movimientos de materiales eléctricos. Desarrollado con Delphi 6.0 y como sistema de gestión de base de datos My Sql.
- *Sistema de Inscripción y Puntajes en Tiro a la Hélice:* las funciones principales son administración de inscripciones, determinación de puntajes, series por categorías. Desarrollado con Visual C# 2005 y como sistema de gestión de base de datos SQL Server 2005.

Al final de la aplicación del prototipo, se espera que el mismo permita una correcta captura (etnografía aplicada), especificación y control de requisitos, preservando siempre los principios básicos de los MA.

CAPÍTULO III

APLICACIÓN DE ETNOGRAFÍA

La comprensión del problema es un factor fundamental en el desarrollo de un proyecto, y ha sido uno de los temas más tratados a lo largo de las últimas cuatro décadas a raíz de la crisis del software, ya que a medida que avanza la tecnología, los sistemas se tornan más complejos. Para resolver un problema, lo primero que hay que hacer es conocerlo y por lo tanto se necesita saber cuáles son los requisitos del mismo.

El análisis y la especificación de los requisitos puede parecer una tarea sencilla, pero no es así. Sobran las ocasiones para las malas interpretaciones y la falta de información; esto puede ocasionar que se construya una solución software, que sea muy elegante, pero que no resuelva el problema [33].

III.1. ANÁLISIS DE REQUISITOS

La captura de los requisitos es el primer paso, que va seguido por el análisis de los mismos. Del análisis se obtendrá una especificación de requisitos que se deberá validar frente a las especificaciones iniciales.

Esta fase es la primera del desarrollo del proyecto y es la más importante, sobre todo si se tienen en cuenta las consecuencias de una mala obtención de requisitos. Para evitar las malas consecuencias hay que comunicarse de forma efectiva con el cliente para obtener los requisitos y comprender el problema a resolver.

La comunicación con los usuarios es uno de los aspectos más relevantes para el desarrollo de sistemas software; aun así se confía más en la experiencia acumulada que en la aplicación de métodos para capturar la experiencia de los usuarios y sus verdaderas necesidades. Sin embargo, cuando se analizan las causas por las cuales no se satisfacen las necesidades de los usuarios, se suele otorgar la culpabilidad a los propios usuarios, argumentando *"que no describieron correctamente sus necesidades, que*

cambian de pensamiento fácilmente o que tienen diferentes puntos de vista". Lo que nunca se piensa es que si se hubieran aplicado correctamente las técnicas del análisis de requisitos centradas en los usuarios, se habrían ahorrado estos problemas y los usuarios estarían satisfechos [33].

En la fase del análisis de requisitos se establecen los servicios que el sistema debe proporcionar y las restricciones bajo las cuales debe operar. Se especifican las condiciones que determinan "*qué*" debe hacer el sistema y las restricciones impuestas, o sea requisitos funcionales y no funcionales.

Lo normal es recopilar los requisitos hablando con todas las personas involucradas, ya sea usuario o expertos en el tema de que trata del sistema. Generalmente, las personas con las que se habla proporcionan los requisitos de una forma vaga y desordenada. Por lo tanto, es labor del analista ordenarlos (por temas, por dificultad, por importancia para los usuarios, etc.) y asegurarse de que sean claros.

Durante el análisis se define más claramente qué es lo que va a hacer el software. Para ello se debe:

- Identificar actores, es decir los usuarios u otros sistemas con los que se pueda comunicar el sistema.
- Identificar casos de uso, es decir algo que un actor quiera hacer con el sistema.
- Especificar casos de uso detallando sentencias del estilo *"el usuario hace tal cosa y el sistema hace tal otra"*. Es decir, se especifica por escrito, desde el punto de vista del usuario, qué es lo que tiene que hacer y qué es lo que va a hacer el sistema por él.

III.2. LA ETNOGRAFÍA Y LA RECOLECCIÓN DE REQUISITOS

La correcta identificación de requerimientos, tanto los del usuario como los del producto, es indispensable en las etapas tempranas del proceso de desarrollo, el cual debería culminar en la satisfacción de las necesidades del usuario. Para ello, se estudia el dominio del problema interactuando constantemente con clientes y usuarios para detectar información sobre sus verdaderas necesidades. Las diferentes técnicas utilizadas en el proceso de elicitación se pueden clasificar en [19, 38]:

- tradicionales, incluyen el uso de cuestionarios, entrevistas y análisis de documentos existentes;
- cognoscitivas, incluyen técnicas para la adquisición de conocimientos para sistemas basados en conocimiento; y
- contextuales, son los que surgieron en la década de 1990 como una alternativa tanto a las técnicas tradicionales como a las cognoscitivas, incluyen el uso de técnicas etnográficas, tales como la observación de los participantes y el análisis de la conversación.

III.2.1. LA INVESTIGACIÓN ETNOGRÁFICA

El contacto o la interacción con los usuarios se debería realizar en el contexto donde éstos trabajan habitualmente. Por lo tanto, para lograr una aproximación con el usuario, que permita obtener la información que se necesita, se deben usar métodos de indagación, entre los cuales se distinguen [14]:

- la aproximación contextual propiamente dicha, para la cual el método característico es la i*ndagación en el contexto*; se trata, de un método estructurado de entrevista de campo, caracterizado por la necesidad de comprender el contexto y las necesidades del usuario; y,
- el e*studio etnográfico u observación de campo*, que consiste en la observación del usuario y su interacción con el producto en su entorno habitual.

Los usuarios a menudo encuentran difícil describir *"lo que hacen"*. Muchas veces la mejor manera de entenderlos es observarlos en su trabajo, o sea, realizando un trabajo etnográfico.

La *etnografía* ha sido concebida como la ciencia que estudia, describe y clasifica las culturas o los pueblos, como se mencionó en el capítulo anterior. Desde esta perspectiva, se la puede ver como una teoría de la descripción. Entonces, se puede decir que la *etnografía* permite descubrir y describir las acciones de los participantes dentro de su interacción social contextualizada, en el sentido y el significado que dan los mismos participantes a sus acciones.

La investigación etnográfica se acepta como una metodología no clásica, naturalista, básicamente descriptiva, por lo que existe la tendencia a considerarla como sinónimo de trabajo informal, libre de fundamentos y de enlaces teóricos; esto se debe

al profundo desconocimiento del verdadero alcance de la etnografía y de las elaboraciones teóricas que de la misma pueden derivarse. [36]

III.2.1.1. Indagación en el Contexto

Durante la indagación en el contexto, se averigua cómo se hacen las cosas o cómo deberían hacerse, si funciona la forma de hacerlo, si existe otra forma, etc. Se debería realizar preferentemente en las etapas tempranas del proceso de desarrollo. En estas etapas, cuando apenas se tiene la idea de que se requiere un determinado producto para satisfacer una necesidad particular [38].

III.2.1.2. Estudio Etnográfico / Observación de Campo

Una parte de la observación de campo consiste en indagar; esto es, entrevistar a los usuarios acerca de sus trabajos y la forma en que hacen uso del sistema. La otra parte consiste en la observación propiamente dicha, la del modo en que las personas harían uso del producto en el curso de su vida cotidiana. Una manera de asegurar una captación de datos apropiada es la de identificar el mayor número de artefactos y de afloramientos posibles [19, 20].

- Los artefactos son objetos físicos utilizados en un determinado lugar (por ejemplo, cuadernos, formularios, etc.).
- Los afloramientos hacen referencia a los rasgos físicos que caracterizan al lugar en cuestión (por ejemplo, el tamaño de oficinas, uniformes del personal, etc.).

III.2.1.3. Alcance de la Etnografía

Como labor descriptiva, el propósito fundamental de la etnografía es describir una cultura o algún aspecto de una o más culturas en una organización. Sin embargo, también incluye la comprensión y la interpretación de fenómenos hasta llegar a teorizaciones sobre los mismos.

Entonces, la etnografía permite reflexionar sobre la realidad, asignando significaciones a lo que se ve, se oye y se hace, desarrollando aproximaciones hipotéticas, redefiniendo continuamente, hasta llegar a construir e interpretar esa realidad sin anteponer el sistema de valores del investigador, lo cual conduce a la reconstrucción teórica.

La observación de los usuarios en su entorno habitual es, por lo general, la mejor forma de determinar sus requerimientos de usabilidad [38].

III.2.1.4. Características de la Etnografía

- Incorpora experiencias, creencias, actitudes, pensamientos, reflexiones, de los participantes; es decir, supone describir e interpretar los fenómenos sociales desde su propia perspectiva del: *" tal como son expresadas por ellos mismos y no como uno los describe"* [38].
- Estudia la cultura como unidad particular; para ello, parte de la observación de las conductas que se evidencian en la interacción de las personas y descubriendo el significado cultural de tales conductas desde la óptica de los propios participantes y del investigador [38].

III.2.2. APLICACIÓN DE LA ETNOGRAFÍA EN EL ANÁLISIS DE REQUISITOS

Para aplicar la etnografía al desarrollo de sistemas software, se proponen los siguientes lineamientos:

- Pasar tiempo suficiente con los usuarios, para tratar de conocerlos e intentar establecer una buena relación con ellos.
- Tomar notas detalladas de todas las actividades del trabajo de los usuarios, analizarlas y sacar conclusiones con ellos mismos.
- Combinar la observación con entrevistas abiertas.
- Combinar la etnografía con otras técnicas de elicitación.

Hay ciertas actividades de la vida humana que o son de difíciles capturar o simplemente no pueden capturarse. De las actividades que se repiten frecuentemente, se espera capturar sus aspectos principales, mientras que para aquellas actividades poco frecuentes, la observación directa sólo podrá hacer un registro de ellas, si suceden durante el proceso de observación.

Con la aplicación de estudios etnográficos, como técnica de análisis de requisitos durante el desarrollo de sistemas software, se podrá:

a) Describir el contexto, el lugar de trabajo y cómo las personas realizan sus tareas.

b) Detallar y entender las relaciones entre las personas y los objetos que dichas personas utilizan, directa o indirectamente.

c) Aumentar la información relativa a la organización de las tareas y a su consecución.

d) Aumentar, cualitativa y cuantitativamente, las conclusiones obtenidas en una observación de campo.

A través de una correcta elicitación de requisitos será posible obtener una apreciación total de las actividades del usuario y un mayor entendimiento de los aspectos ambientales, sociales y personales del usuario y el contexto de uso. De esta forma, se logrará identificar los elementos que serán de gran importancia para la especificación correcta de requisitos: artefactos, plataforma tecnológica, objetivos, stakeholders[1], usuarios, tareas, personajes [18, 19, 38].

III.2.2.1. Los Artefactos

Son los objetos que, en forma directa o indirecta, intervienen en el proceso de interacción entre la persona y el sistema. Pueden ser cosas físicas o bien conceptuales (contraseñas, firmas, etc.).

El uso de los artefactos constituye una importante fuente de información para el análisis del sistema, ya que, por ejemplo, un cuaderno en un lugar determinado puede significar “anotar novedades”, mientras que en otro lugar de la organización puede significar “notas revisadas” [20].

III.2.2.2. La Plataforma Tecnológica

Este elemento se refiere al hardware que se necesita y al sistema operativo donde el sistema vivirá. En función de la elección de la misma, se determinarán las posibilidades que dicha plataforma ofrece, así como las restricciones que impone. Indudablemente, estas posibilidades y restricciones se deberán tener en cuenta al momento de diseñar la interfaz de usuario.

[1] Es cualquier grupo o individuo que puede afectar o puede ser afectado por la consecución de los objetivos de la organización.

III.2.2.3. Los Objetivos del Sistema

Cuando se toma la decisión de desarrollar un sistema, se fijan ciertos objetivos a cumplimentar a través de la implementación del mismo. Para identificar los objetivos de la aplicación, se tendrán en cuenta los requisitos funcionales y los no funcionales, incluso los objetivos que se refieren a la usabilidad y accesibilidad del sistema. Por más usable y accesible que sea un sistema, de nada servirá si no realiza las tareas que se le han encomendado (objetivos funcionales).

La usabilidad se refiere a que el sistema a desarrollar sea fácil de aprender, efectivo y agradable para sus usuarios; es decir, se trata de optimizar las interacciones de las personas con el sistema en su vida diaria. Por ejemplo, que sea "fácil de aprender" significa que los usuarios podrán utilizar el sistema sin ningún tipo de aprendizaje previo, y las funciones principales estarán bien definidas y serán cómodamente accesibles.

Un punto importante a tener en cuenta, es evitar que ningún objetivo funcional entre en conflicto con objetivos de usabilidad o accesibilidad. Si esto ocurriera, lo que se podría realizar es una reunión con los stakeholders (concepto que se abordará a continuación) y realizar una especie de análisis de las cosas a favor y en contra de cada uno de ellos.

III.2.2.4. Los Stakeholders [17]

Un stakeholder es un participante en el proceso de desarrollo, que junto a cualquier otro individuo, grupo u organización, puede influenciar o ser influenciado por el desarrollo y el uso del sistema, ya sea directa (ingenieros de software responsables del desarrollo y los usuarios finales) o indirectamente (directores[2] de los usuarios y los socios y/o proveedores tecnológicos).

En todo proyecto existe un grupo de stakeholders que son elementales y cuya identificación resulta muy fácil, pero se tiene que realizar un esfuerzo mayor para la identificación de los que no son tan elementales. El objetivo es encontrar todas las personas involucradas, incluso aquellas que pueden influir negativamente en el proyecto.

[2] Los directores son los responsables del trabajo de los usuarios y de los que están relacionados con el desarrollo del sistema.

Entre los stakeholders, por lo general, se encuentran los usuarios, los desarrolladores y los que toman las decisiones. Para su identificación se tiene que:

- utilizar la observación de campo, puesto que no es lo mismo intentar identificar dichos usuarios en el lugar donde la acción se realiza que fuera de ella;
- estar muy atentos, los stakeholders pueden ser internos al equipo, internos a la organización o externos a cualquiera de ellos;
- considerar que en el ciclo completo de las actividades de negocio pueden aparecer nuevos stakeholders “por sorpresa”, que no estaban previstos, y
- considerar el ciclo de vida del desarrollo completo y no hacerlo sólo en la fase inicial.

Una vez identificados los stakeholders, se debe obtener la información relativa a la influencia que ellos pueden tener sobre el proyecto. Una de las formas más habituales de obtener esta información es realizando reuniones de stakeholders. En la tabla 3.1 se proporcionan las principales recomendaciones para preparar una reunión.

Tabla 3.1. Reuniones de Stakeholders

Antes de la reunión	Identificar puntos clave que se necesita explorar.
	Proporcionar la agenda y el listado con los puntos a tratar a todos los participantes.
Durante la reunión	Una vez discutidos los puntos clave, se deberá intentar obtener un consenso en aquellos puntos donde haya habido incertidumbre o disconformidad.
	Si se ha echado en falta información, deberá acordarse cómo se obtendrá.
	Realizar una discusión sobre temas menores.
Después de la reunión	Obtener toda la información que faltaba.
	Si la información no es fácil de obtener, organizar un estudio de campo para observar a los usuarios en su ambiente de trabajo.

III.2.2.5. Los Usuarios

La usabilidad es la medida en la cual un producto es usado por usuarios específicos para conseguir objetivos concretos con efectividad, eficiencia y satisfacción en un contexto de uso particular [24]. Entonces, un sistema es usable si los usuarios

pueden hacer rápida y fácilmente sus tareas. Por todo esto, es esencial identificar a los reales usuarios, y el desarrollador debe tener siempre presente que no será el usuario final; además, habrá que identificar las necesidades y expectativas de los usuarios primarios[3] y los secundarios[4].

Por lo tanto, será muy importante para el análisis de requisitos obtener una clasificación de los distintos tipos de usuarios y una descripción de las características más relevantes de la población potencial que usará el sistema a desarrollar.

Para clasificar a los usuarios, se puede usar los roles que éstos cumplen en la organización. De esta manera, se puede indicar clases de usuarios que tienen asignados ciertos subconjuntos de tareas, ya sea por elección propia o como resultado de la organización en la que se encuentran. Además, puede ocurrir que varios usuarios estén involucrados en un mismo rol, y un mismo usuario puede tener varios roles al mismo tiempo.

III.2.2.6. Las Tareas [17]

Durante el análisis de requisitos, se realiza un análisis de las tareas, es decir cómo las realizan los usuarios, qué patrones utilizan, si es que utilizan alguno. Con esto, se trata de entender cuáles son los objetivos del usuario.

Lo que se quiere lograr es, a través de la realización de análisis etnográfico, conocer todas las tareas que el sistema debe ser capaz de realizar en el contexto específico en el cual se desarrollan. Es decir, se trata de comprender la estructura organizativa, lo que le permitirá al analista luego entrelazar las tareas que el sistema debe realizar con el contexto de su realización.

III.2.2.7. Los Personajes [17]

Se puede definir personajes como modelo de usuario. La función de este modelo será representar las necesidades, los comportamientos, los patrones de uso y las motivaciones, identificados durante la fase de elicitación de requisitos. Este modelo permitirá simplificar el entendimiento de estructuras y relaciones complejas. Los personajes son útiles en el momento el diseño del proyecto donde se configura la idea de lo que será más adelante el sistema.

[3] Usuarios primarios: son aquellos que usan el sistema frecuentemente.
[4] Usuarios secundarios: son los usuarios que usan el sistema en forma ocasional.

Como cualquier modelo, los personajes están basados en información obtenida a través de investigaciones etnográficas de usuarios reales y observaciones del mundo real, aunque también pueden contener elementos de ficción. Los personajes tienen identidad y existencia propia, tienen edad, género, nivel de educación, historias de vida, motivaciones, preferencias y maneras de hacer las cosas.

Estos personajes (que pueden representar a usuarios) utilizan el sistema con un fin determinado, es decir tienen metas. Una meta es una condición final, y las tareas son un estadio intermedio que permite alcanzar una meta. Los personajes permiten al diseñador entender y diferenciar metas de tareas. Primero hay que identificar las metas y luego diseñar las tareas que le permitirán al usuario alcanzar sus metas.

Es por todo esto que se plantea a la etnografía como técnica de elicitación, porque no tiene sentido preguntar a los usuarios cuáles son sus metas ya que, por lo general, tratarán de inventar una respuesta para satisfacer al entrevistador. Es decir, a través del comportamiento observado en el entorno, se pueden deducir las metas. Cuando se las identifican se las trata de expresar en una oración (lo más breve posible).

Las metas pueden referirse a aspiraciones personales que van más allá del contexto del sistema (metas vitales), a cómo el usuario quiere sentirse mientras utiliza un sistema (metas de experiencia), o bien a las expectativas con respecto a los resultados, aquello que el usuario espera alcanzar por medio del sistema (metas finales). También se puede identificar metas externas, las cuales son metas empresariales y técnicas que son imprescindibles para que el producto sirva al usuario [19].

III.3. PROPUESTA DE APLICACIÓN DE LA ETNOGRAFÍA EN LA FASE DE ANÁLISIS DE REQUISITOS

La siguiente es una propuesta para aplicar la etnografía durante la fase de análisis de requisitos, la cual se puede interpretar como una lista de recomendaciones a seguir de modo que los desarrolladores puedan incorporar la información obtenida la herramienta de gestión de historias de usuario llamada HGH (*Herramienta de Gestión de Historias*), cuyo diseño se planteará en el siguiente capítulo.

Esta propuesta se divide en dos partes: la primera consiste en la indagación (entrevistar a los usuarios), y la segunda en la observación propiamente dicha.

- En cuanto a la *indagación*, los lineamientos son los siguientes:

  a) Realizar entrevistas abiertas y reuniones con el cliente y los usuarios, de modo de generar ideas para obtener distintas visiones del problema y poder formularlo de diferentes maneras.

  b) Determinar requisitos funcionales, es decir lo que se espera que el sistema haga.

  c) Determinar requisitos no funcionales, opciones de mantenibilidad, plataforma tecnológica, tiempo de respuesta, etc.

  d) Definir plazos de desarrollo.

  e) Definir características de usabilidad y accesibilidad del sistema a desarrollar.

  f) Identificar principales subsistemas.

- En cuanto a la *observación de campo* propiamente dicha, los lineamientos son los siguientes:

  a) El desarrollador debe introducirse en el grupo, hasta llegar al punto de confundirse como un miembro más de la organización.

  b) Identificar las metas de los usuarios y, si es necesario, definir personajes que ayuden a individualizarlas.

  c) Identificar los artefactos usados y determinar la importancia y los efectos de su uso en el sistema.

  d) Identificar la importancia de los afloramientos en el lugar de trabajo, y determinar la forma que ellos influyen en el sistema a desarrollar.

  e) Determinar y describir las actividades que se realizan con frecuencia.

  f) Detallar las relaciones entre los diferentes usuarios del sistema.

  g) Identificar los stakeholders y la influencia que éstos ejercen sobre el sistema. Diferenciar los stakeholders elementales de aquellos que pueden afectar negativamente al desarrollo del sistema.

  h) Identificar los usuarios potenciales del sistema y definir los roles que éstos cumplen en la organización, de manera de poder clasificarlos posteriormente.

  i) Identificar y describir las tareas que realizan los usuarios y los patrones que utilizan para llevarlas a cabo.

Con las recomendaciones propuestas se espera obtener la información necesaria para incorporar a la herramienta y así lograr correctas especificaciones. A través de la *indagación*, el analista podrá definir las tareas (subsistemas) a describir en la herramienta,

requisitos funcionales y no funcionales, establecer las restricciones y posibilidades impuestas por la plataforma tecnológica y definir algunas historias de usuario. Y con la *observación de campo,* al introducirse el analista en el grupo podrá individualizar personajes, definir roles, establecer y describir relaciones, artefactos usados y definir prioridades, todo esto lo podrá utilizar en la herramienta para crear o modificar historias, crear versiones de las historias y establecer la trazabilidad entre las historias.

A final se lograrán especificaciones del sistema por usuarios, por tareas, por prioridades, por trazabilidad de historias, etc.; es decir, se podrá conseguir una verdadera gestión de requisitos (actualización y control de requisitos) que sirva para apoyar el diseño y facilite el mantenimiento del sistema.

CAPÍTULO IV

DISEÑO DE LA HERRAMIENTA DE GESTIÓN DE HISTORIAS

En este capítulo se identifican los usuarios de HGH y se definen los requisitos candidatos.

Luego, se utiliza UML para definir el diagrama de contexto y los casos de uso observados. Finalmente, se presentan las realizaciones de los casos de uso a través de los diagramas de clase y los diagramas de secuencia.

IV.1. FASE DE INICIO

IV.1.1. IDENTIFICACIÓN DE LOS ACTORES DEL SISTEMA

1. *Usuario*: es la persona que utilizará HGH para definir las historias y las diferentes versiones de las mismas.
2. *Resto del Equipo de Desarrollo*: conformado por parte del equipo de desarrollo, el cual se encarga de
    - realizar el diseño en función de la información obtenida durante la fase de análisis,
    - realizar la programación del sistema en función del diseño y, en caso de dudas, acude a la información recabada durante el análisis, y
    - realizar el mantenimiento, teniendo en cuenta lo realizado en etapas anteriores, y los requerimientos del sistema.
3. *Analista*: se encarga de
    - recoger la información obtenida a través de las técnicas tradicionales de elicitación de requisitos,

- recoger la información obtenida a través de etnografía y observación de campo, y
- definir, en función de la información obtenida, las tareas del sistema de manera que los usuarios escriban sus historias.

4. *Cliente*: es quien establece los requisitos iniciales del sistema.
5. *Cliente que contrata el desarrollo del Sistema*: representa al responsable de la empresa que solicita la construcción del sistema.
6. *Futuro Usuario del Sistema*: representa a todas las personas que usarán el sistema una vez construido.

IV.1.2. REQUISITOS CANDIDATOS

El *Resto del Equipo de Desarrollo* necesita:

- Tener los requisitos detallados para saber, en todo momento, la diferencia entre el sistema que se quiere obtener y el que se tiene.
- Tener documentación precisa, para facilitar el mantenimiento del sistema.
- Obtener código ligero conforme a las necesidades del cliente y que pueda modificarse rápidamente.

El *Analista* necesita:

- Poder definir las tareas del sistema de modo que los usuarios escriban sus historias.
- Incorporar al usuario en el equipo de desarrollo, de manera de tener siempre al alcance su perspectiva del sistema.
- Cumplir con los plazos y la calidad del trabajo establecidos.
- Tener la capacidad para crear cambios, y que el equipo de desarrollo pueda responder y reaccionar a ellos.

El *Cliente* (Futuro Usuario de Sistema - Cliente que contrata el desarrollo del Sistema) necesita:

- Describir en forma clara y simple las funciones que debe realizar el sistema.
- Plasmar en las historias de usuario los requerimientos funcionales y no funcionales del sistema y establecer las prioridades de los mismos.
- Reflejar en forma simple y sencilla sus necesidades.
- Obtener un producto que presente la funcionalidad y la calidad deseada.

IV.1.3. DIAGRAMA DE CONTEXTO

Como se muestra en la figura 4.1, el sistema cuenta con los siguientes casos de uso:

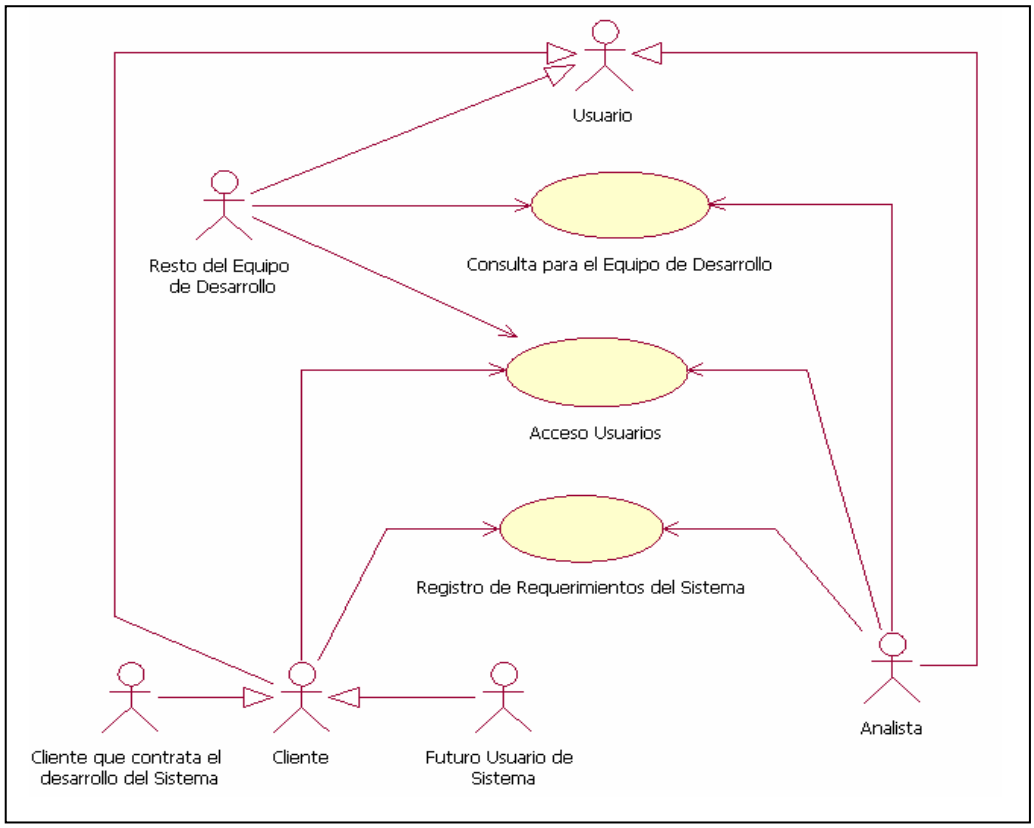

Figura 4.1. Diagrama de Contexto

1) Acceso a usuarios

   Se encarga del control de la seguridad permitiendo diversos tipos de acceso de acuerdo a las características de cada usuario (roles). Para ello lleva a cabo las siguientes operaciones:

   - Alta y modificación de datos y permisos (de acuerdo al rol) de usuarios.
   - Control de acceso mediante Nombre y Contraseña.
   - Modificación de contraseña.

2) Registro de Requerimientos del Sistema

   Se encarga de registrar los requerimientos generales del sistema, los cuales son proporcionados por el cliente, usuarios del sistema y por el analista. Se definen las tareas y los subsistemas de modo que los usuarios puedan definir sus historias, en función a la información obtenida a través de etnografía aplicada.

También, este caso de uso se ocupa de registrar las historias de usuario y realizar el control de versiones de las mismas.

3) Consulta para el Equipo de Desarrollo
   Es usado por el Resto del Equipo y por el Analista para:
   - obtener información acerca de las diferentes funciones del sistema durante el desarrollo del proceso, llámese diseño, programación, mantenimiento;
   - determinar desvianciones entre lo estimado y la duracion real.

IV.2. REQUISITOS CANDIDATOS

IV.2.1. MODELOS DE CASOS DE USO

IV.2.1.1. Acceso a Usuarios

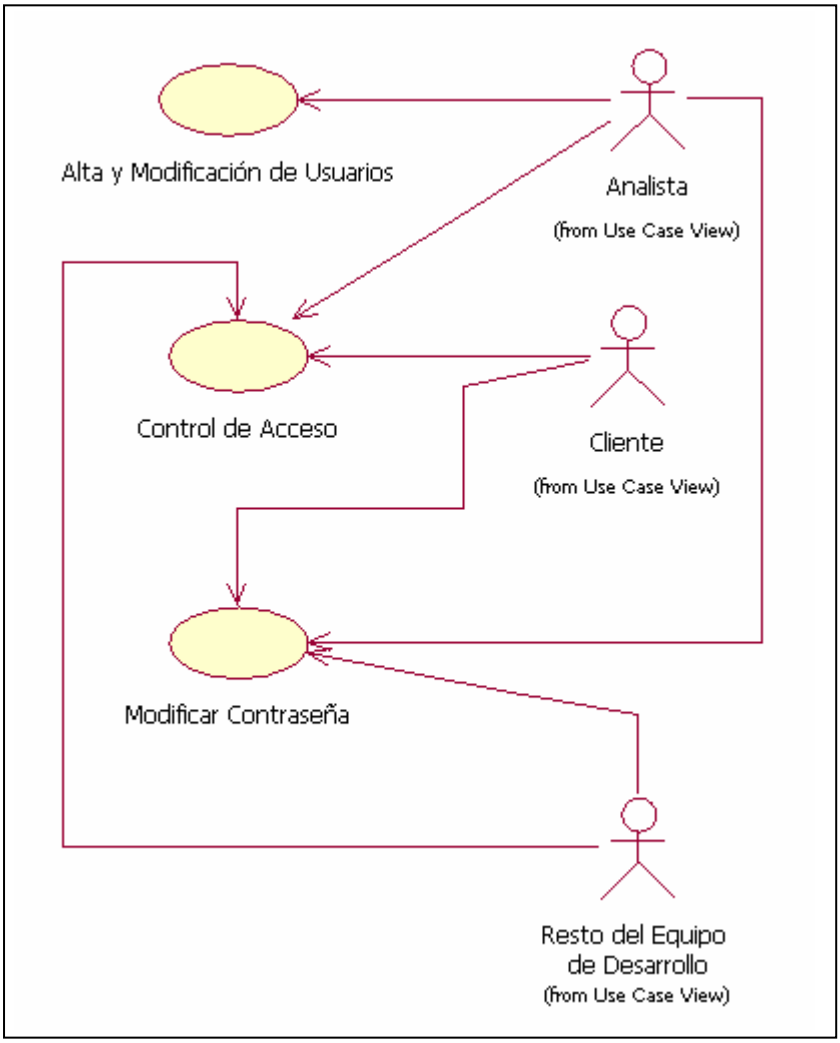

Figura 4.2. Acceso a Usuarios

Caso de Uso Control de Acceso: es usado por el usuario general de HGH ya que, dado el nombre y la contraseña de usuario, se habilitan las funciones de acuerdo a los permisos que éste posea y al rol asignado.

Iniciador	Usuario
Camino básico	1-El Usuario ingresa su nombre de usuario y contraseña. 2-Se verifican los datos ingresados en la tabla Usuarios, Roles y Usuarios_Roles. 3-Entonces se habilitan las funciones para las cuales el Usuario tiene permisos de acuerdo al rol asignado.
Poscondición	Funciones habilitadas.

Caso de Uso Alta y Modificación de Usuarios: es usado por Analista para ingresar nuevos usuarios o modificar los datos, el perfil y los permisos de algun usuario ya existente.

Iniciador	Analista
Camino básico	1-El Analista elige la operación a realizar (Alta o Modificación). 2-Luego los datos se verifican. 3-En la operación de modificación se solicita la confirmación de la misma. 4-Luego se graban los nuevos registros o se modifican los ya existentes en función de la operación seleccionada sobre la tabla Usuarios y la tabla Usuarios_Roles.
Poscondición	Se agrega o modifica un registro de Usuarios y Usuarios_Roles de acuerdo a la operación seleccionada.

Caso de Uso Modificar Contraseña: es usado por el usuario general de HGH, el cual le permite modificar su contraseña actual.

Iniciador	Usuario
Camino básico	1-El Usuario ingresa su nombre de usuario y contraseña. 2-Se escribe la nueva contraseña y la confirmación de la misma. 3-Entonces, se verifican los datos en la tabla Usuarios. 4-Se modifica la tabla Usuarios.
Poscondición	Se actualiza la tabla Usuarios.

IV.2.1.2. Registro de Requerimientos del Sistema

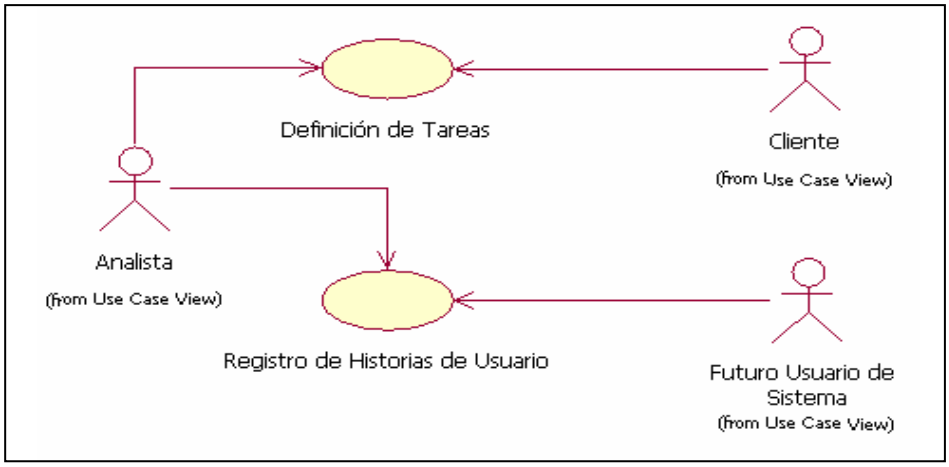

Figura 4.3. Registro de Requerimientos del Sistema

Caso de Uso Definición de Tareas: es usado por el Analista para crear las tareas del Sistema en función de los datos obtenidos a través de etnografía aplicada y otros métodos comunes de elicitación de requisitos; también lo usa junto al Cliente para modificar dichas tareas.

Iniciador	Analista, Cliente
Camino básico	1-Se elige la operación a realizar, ya sea crear o modificar una tarea. 2-Se verifican los datos ingresados en la tabla Tareas. 3-Si se trata de una modificación, se solicita confirmación de la operación. 4-De acuerdo a la operación seleccionada, se agrega un registro al final o se actualiza un registro en la tabla Tareas.
Poscondición	Se agrega un registro al final o se actualiza un registro en la tabla Tareas.

Caso de Uso Registro de Historias de Usuario: es usado por el Futuro Usuario del Sistema (Cliente) para crear las historias de usuario asignadas a una Tarea específica. También lo usa junto al Analista para modificar dichas historias o agregar versiones a una historia descripta anteriormente, aún cuando éstas ya hayan sido finalizadas por el Resto del Equipo de Desrrollo.

Iniciador	Analista, Futuro Usuario del Sistema
Camino básico	1-Se elige operación a realizar, ya sea crear, modificar una historia de usuario o agregar una versión de una historia existente. 2-Se verifican los datos en la tabla Tareas. 3-Se verifican los datos ingresados en la tabla Hitorias_Versiones. 4-Si se trata de una modificación, se solicita confirmación de la operación. 5-De acuerdo a la operación seleccionada, se agrega un registro al final, se actualiza un registro en la tabla Hitorias_Versiones o se agrega un registro en la tabla Versiones.
Poscondición	Se agrega un registro al final, se actualiza un registro en la tabla Hitorias_Versiones o se agrega un registro en la tabla Versiones.

IV.2.1.3. Consulta para el Equipo de Desarrollo

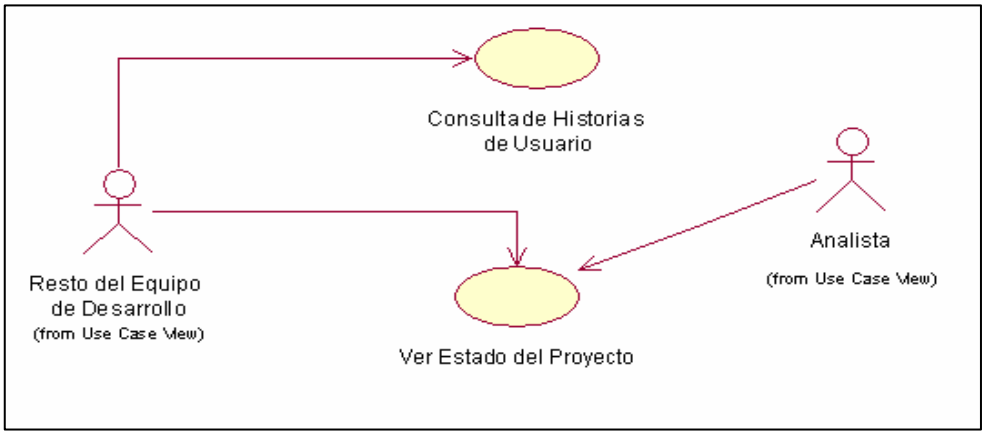

Figura 4.4. Consulta para el Equipo de Desarrollo

Caso de Uso Consulta de Historias de Usuario: es usado por el Resto del Equipo de Desarrollo para consultar las Tareas, el estado de las Historias de Usuario y sus correspondientes versiones; en el caso de iniciar o concluir una historia, ésta se marca de modo que se pueda ver el avance del proyecto.

Iniciador	Resto del Equipo de Desarrollo
Camino básico	1-Se elige la Tarea a consultar. 2-Se verifican los datos ingresados en la tabla Tareas. 3-Se elige la Historia a consultar. 4-Se verifican los datos ingresados en la tabla Historias_Versiones y Versiones. 5-Si se inicia o se termina una historia, entonces se modifica la tabla Historias_Versiones.
Poscondición	Se actualiza un registro en la tabla Historias_Versiones.

Caso de Uso Ver Estado del Proyecto: es usado por el Resto del Equipo de Desarrollo y por el Analista para consultar acerca del estado del proyecto o imprimir las especidficaciones de requisitos, para ello utiliza la información contenida las tablas Tareas e Historias_Versiones.

Iniciador	Resto del Equipo de Desarrollo, Analista
Camino básico	1-Se elige la forma de visualizar los datos del Estado del Proyecto o si se desea imprimir las especificaciones . 2-Se verifica los datos en la tabla Tareas, Usuarios, Versiones e Historias_Versiones. 3-De acuerdo a la opción de seleccionada, se muestran los datos por pantalla con posibilidad de impresión.
Poscondición	Ninguno.

IV.3. REALIZACIONES DE LOS CASOS DE USO

IV.3.1. CASO DE USO: ALTA Y MODIFICACIÓN DE USUARIOS

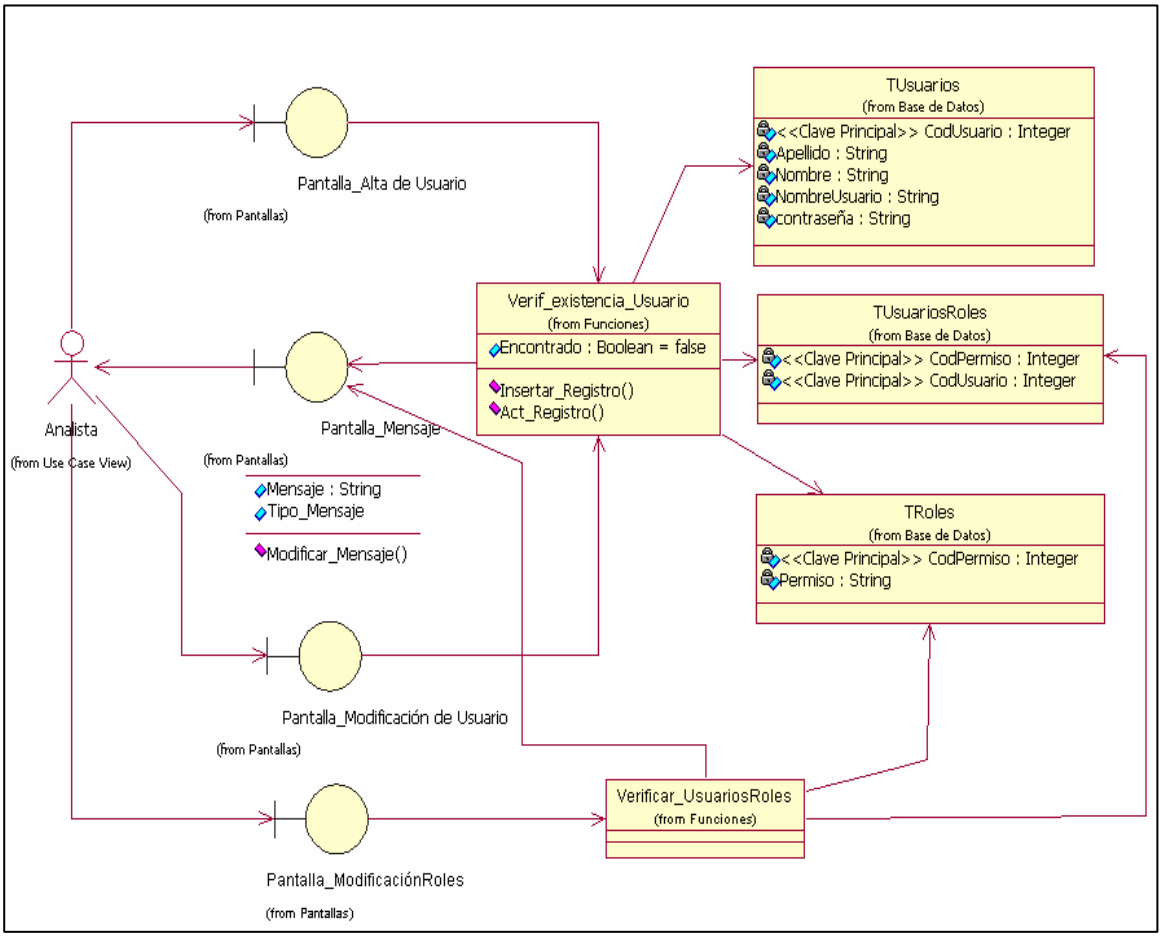

Figura 4.5. Diagrama de Clases del caso de uso Alta y Modificación de Usuarios

IV.3.1.1. Flujo de sucesos: Alta correcta de Usuarios

El Analista puede ingresar datos para dar de alta a un registro (Datos Alta – entre los mismos se encuentra el nombre, apellido, etc.). Entonces se verifican los datos de alta (Verificar Datos); si los datos son correctos y se ha especificado a los permisos de un usuario de acuerdo al rol asignado (Buscar registro en las tablas Usuario y Roles), se agrega un registro nuevo a la tabla Usuarios y varios registros en la tabla UsuariosRoles dependiendo de los permisos asignados.

Luego, se informa al Analista (Modificar Mensaje, mensaje).

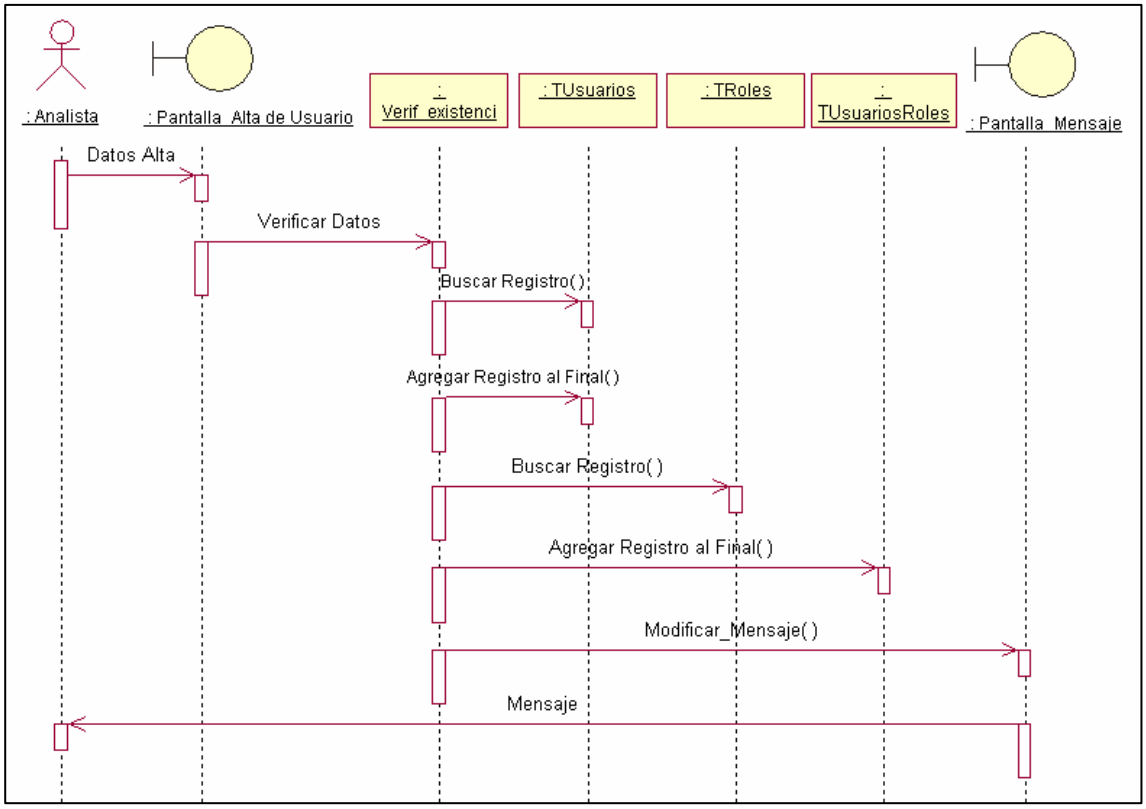

Figura 4.6. Diagrama de Secuencia de la realización de Alta Correcta de Usuarios

IV.3.1.2. Flujo de sucesos: Alta Incorrecta de Usuarios

El Analista puede ingresar datos para dar de alta a un registro (Datos Alta – entre los mismos se encuentra el nombre, apellido, etc.). Entonces se verifican los datos de alta (Verificar Datos), si los datos son incorrectos se informa al Analista de que se trata de un usuario que ya existe (Modificar_Mensaje, Mensaje).

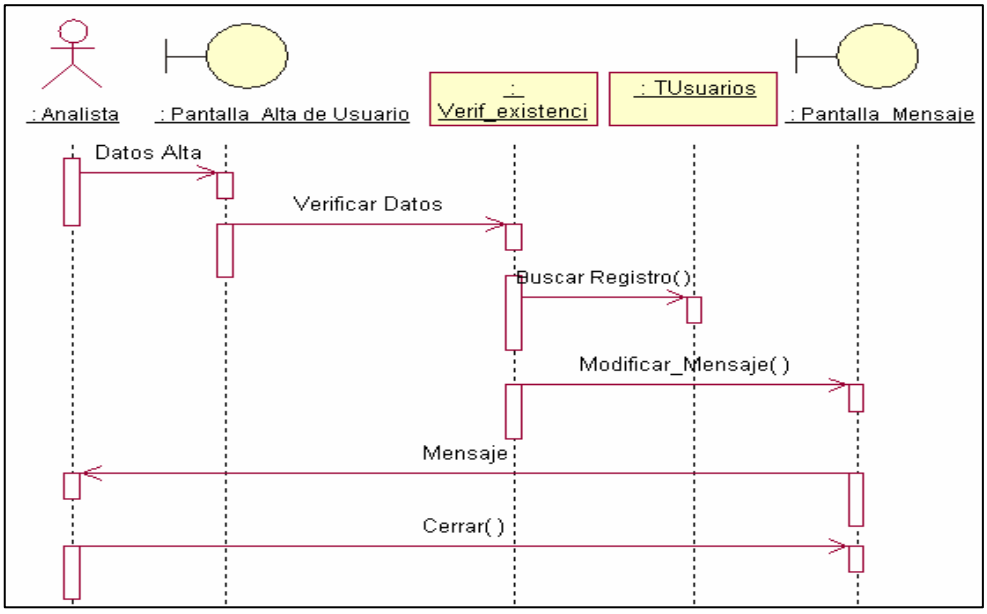

Figura 4.7. Diagrama de Secuencia de la realización de Alta Incorrecta de Usuarios

IV.3.1.3. Flujo de sucesos: Modificación Correcta de Usuarios

El Analista debe ingresar datos para realizar una modificación de los datos personales del usuario (Datos Modificación). Entonces se verifica la existencia del usuario (Existe Usuario y Buscar Registro en la tabla Usuarios), y luego se solicita la confirmación del usuario para realizar la operación (Modificar Mensaje y Mensaje).

Una vez confirmada la operación, se modifica los datos en la tabla Usuarios (Actualizar Tabla).

En el caso de que no se confirme la operación, se cierra la pantalla de mensaje (Cerrar).

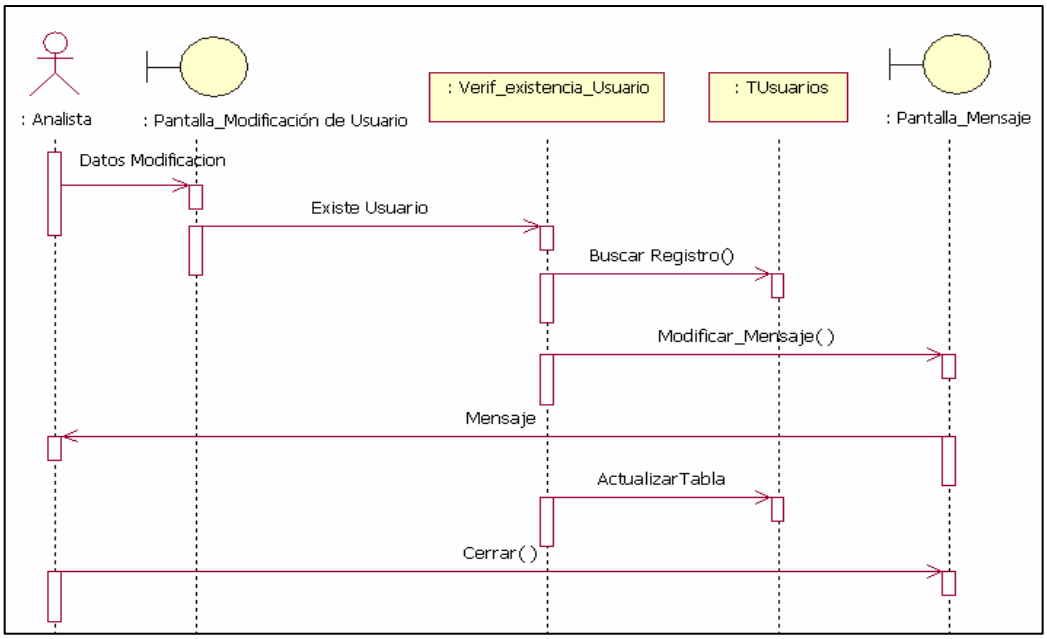

Figura 4.8. Diagrama de Secuencia de la realización Modificación Correcta de Usuarios

IV.3.1.4. Flujo de sucesos: Modificación Incorrecta de Usuarios

El Analista debe ingresar datos para realizar una modificación de los datos personales del usuario (Datos Modificación). Entonces se verifica la existencia del usuario (Existe Usuario y Buscar Registro en la tabla Usuarios), si no existe el usuario se envía un mensaje de error indicando que el usuario ingresado es inexistente (Modificar Mensaje y Mensaje).

Luego se cierra la pantalla de mensaje (Cerrar).

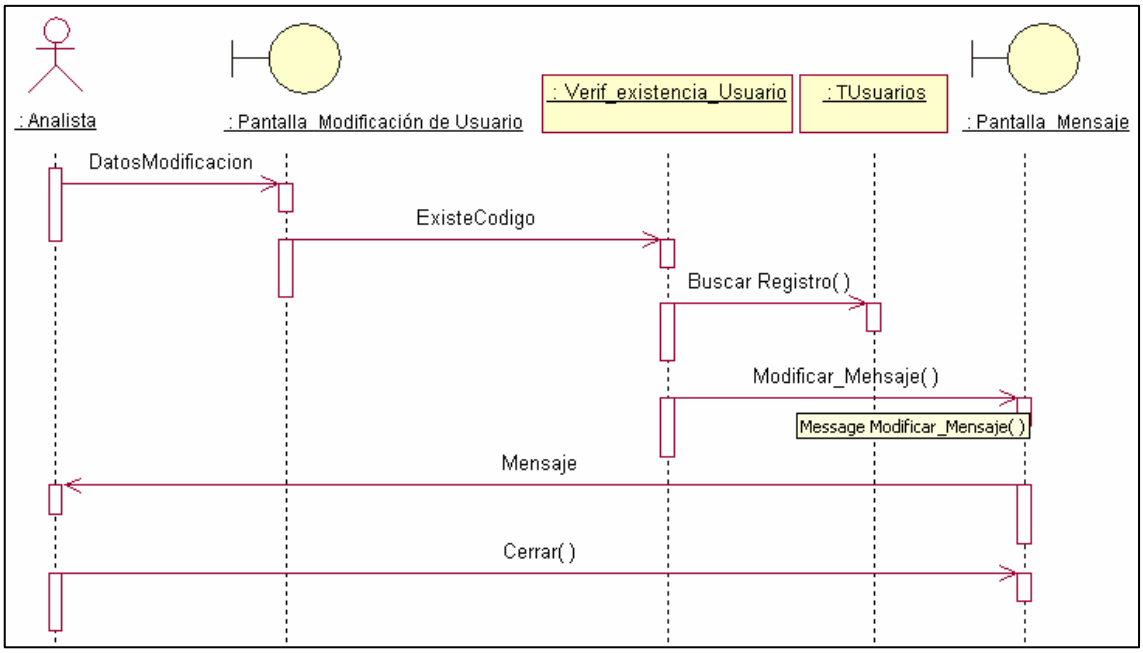

Figura 4.9. Diagrama de Secuencia de la realización Modificación Incorrecta de Usuarios

IV.3.1.5. Flujo de sucesos: Modificar Permisos de Usuarios Correcta

El Analista debe ingresar datos para realizar una modificación de los permisos del usuario de acuerdo al rol asignado (Datos Modificación). Entonces se verifica la existencia del usuario (Existe Usuario y Buscar Registro en la tabla Usuarios y en la tabla Roles), y luego se solicita la confirmación del usuario para realizar la operación (Modificar Mensaje y Mensaje).

Una vez confirmada la operación, se modifica los datos en la tabla UsuariosRoles (Actualizar Tabla).

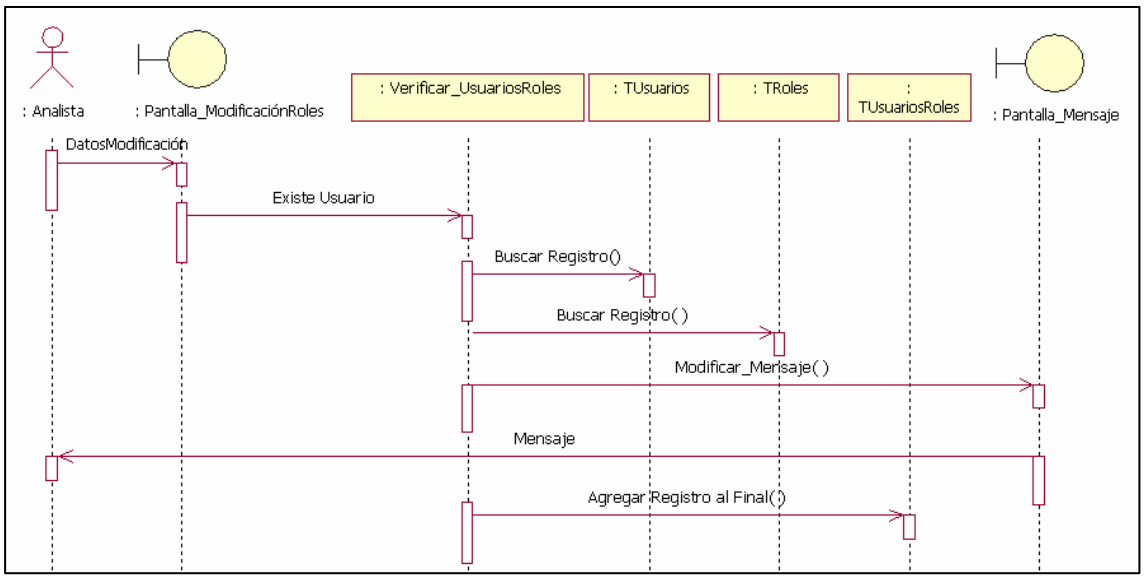

Figura 4.10. Diagrama de Secuencia de la realización Modificar Permisos de Usuario Correcta

IV.3.1.6. Flujo de sucesos: Modificar Permisos de Usuarios Incorrecta

El Analista debe ingresar datos para realizar una modificación de los permisos del usuario de acuerdo al rol asignado (Datos Modificación). Entonces se verifica la existencia del usuario (Existe Usuario y Buscar Registro en la tabla Usuarios y en la tabla Roles) y, si no existe el usuario o el permiso deseado, se envía un mensaje de error indicando la situación (Modificar Mensaje y Mensaje).

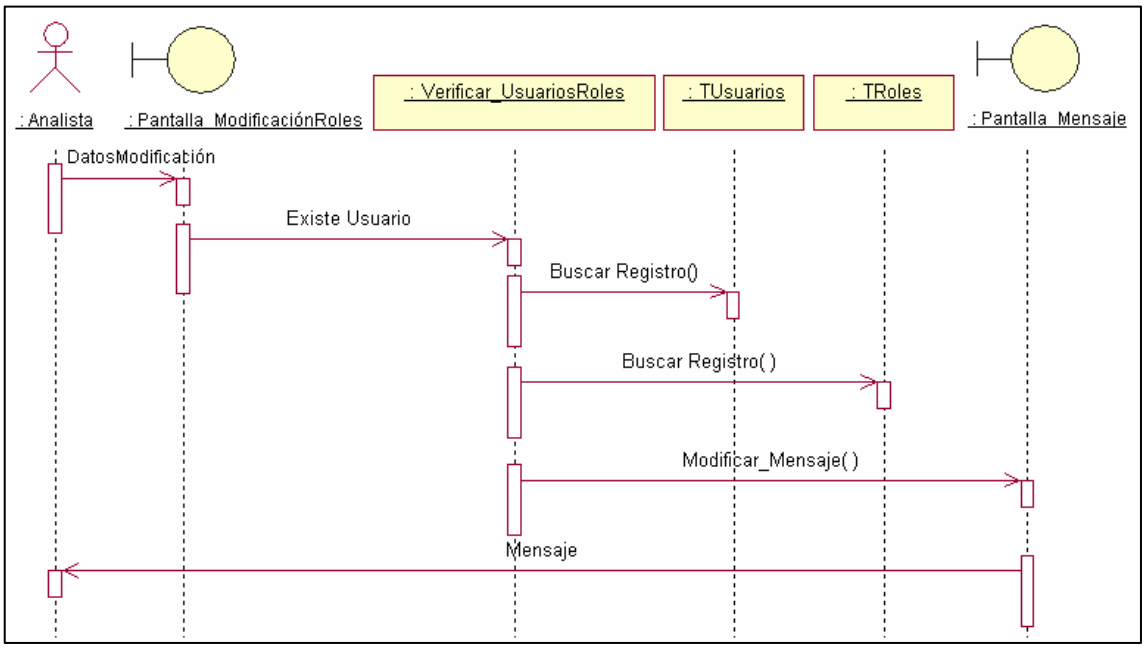

Figura 4.11. Diagrama de Secuencia de la realización Modificar Permisos de Usuario Incorrecta

IV.3.2. CASO DE USO: CONTROL DE ACCESO

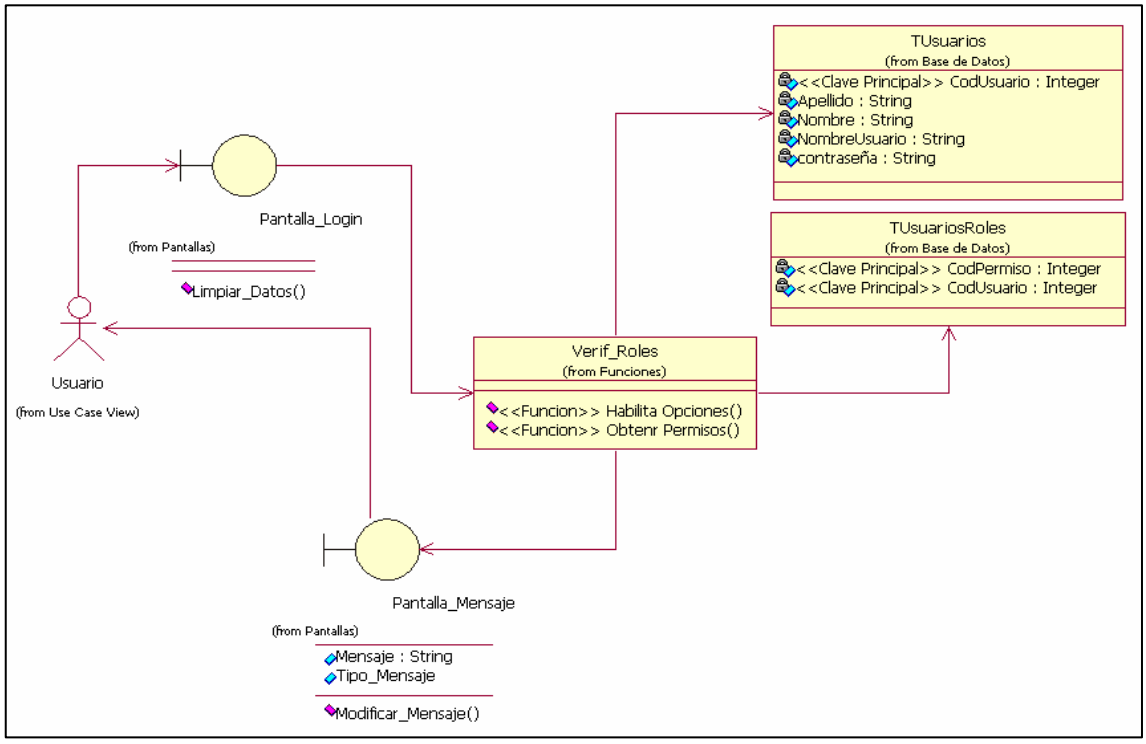

Figura 4.12. Diagrama de Clases Análisis de la realización del caso de uso Control de Acceso

IV.3.2.1. Flujo de sucesos:Control de Acceso

El usuario ingresa el nombre de usuario y contraseña para obtener los permisos de acceso de acuerdo al rol asignado (Datos Cargados), se verifica los datos ingresados comparándolos con los datos cargados en la tabla Usuarios (Verificación de Nombre y Contraseña y Buscar Registro). Si los datos ingresados son correctos, se habilitan las opciones permitidas (según la tabla UsuariosRoles) para el usuario en cuestión.

Si los datos ingresados son erróneos (Ingreso Erróneo), se solicita el reingreso de los datos (Ingresar Nombre y Contraseña).

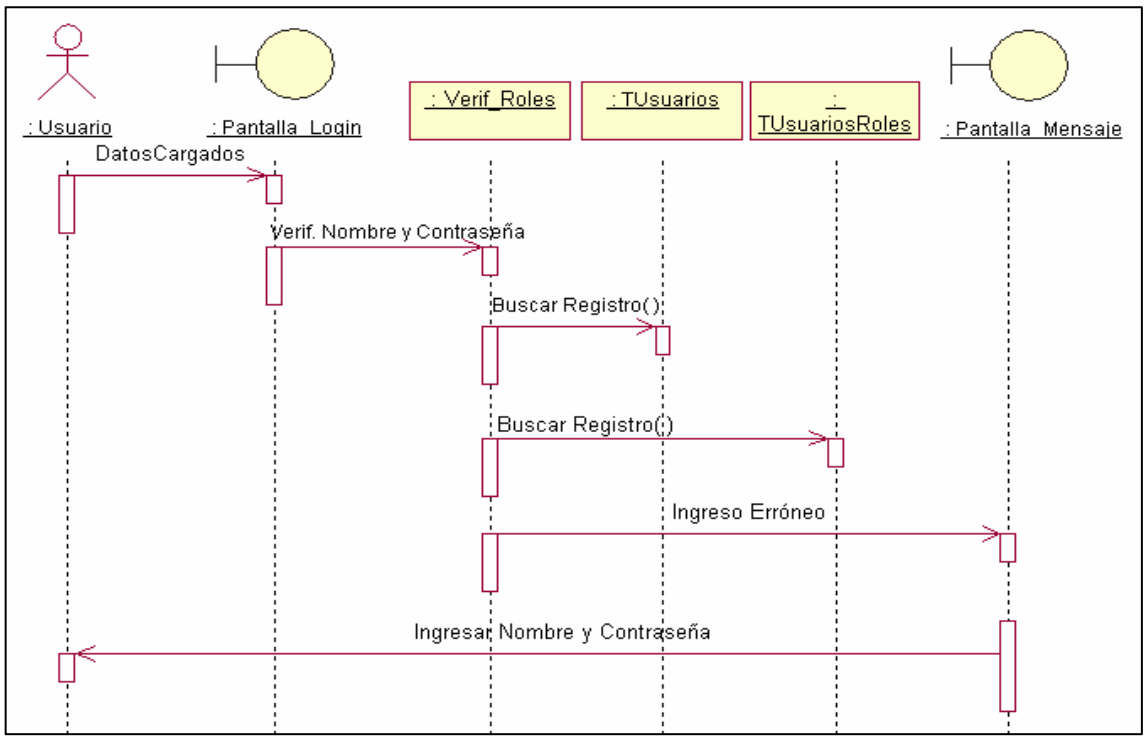

Figura 4.13. Diagrama de Secuencia de la realización Control de Acceso

IV.3.3. CASO DE USO: MODIFICAR CONTRASEÑA

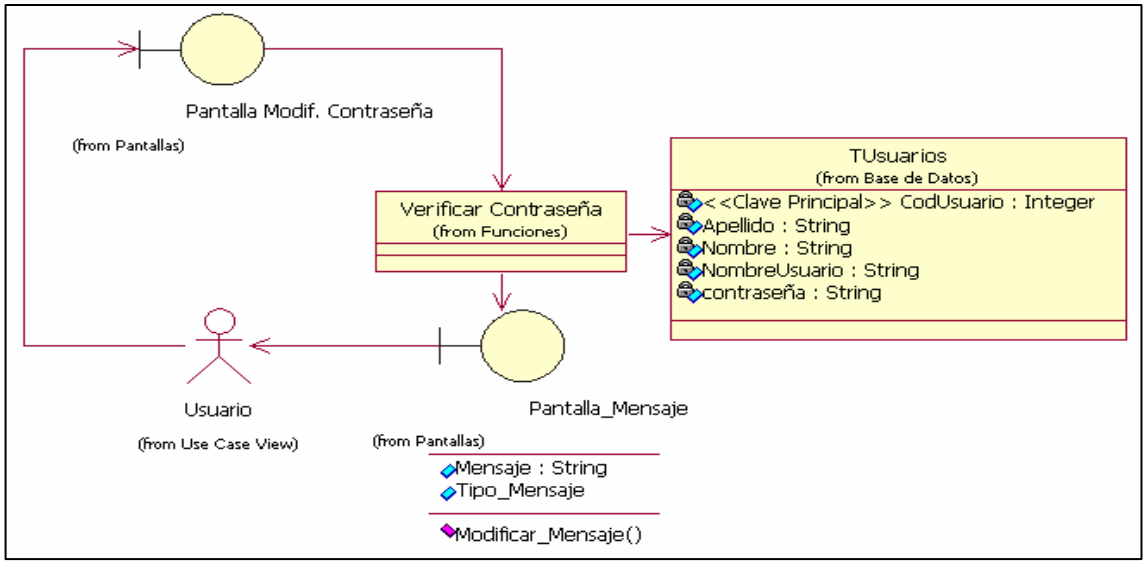

Figura 4.14. Diagrama de Clases Análisis de la realización del caso de uso Modificar Contraseña

IV.3.3.1. Flujo de sucesos: Modificar Contraseña Correcta

El usuario ingresa su nombre de usuario, contraseña, la nueva contraseña y la confirmación de la nueva contraseña (DatosModificaciónContraseña), entonces se verifica la existencia del usuario con la correspondiente contraseña y que la nueva

contraseña sea la misma a la confirmación ingresada (DatosModificaciónContraseña y Buscar Registro). Luego se solicita la confirmación de la operación (Modificar_Mensaje y Mensaje).

Una vez confirmada la operación, se modifica el registro en la tabla Usuarios (Actualizar Tabla). En el caso de que no se confirme la operación, se cierra la pantalla de mensaje (Cerrar).

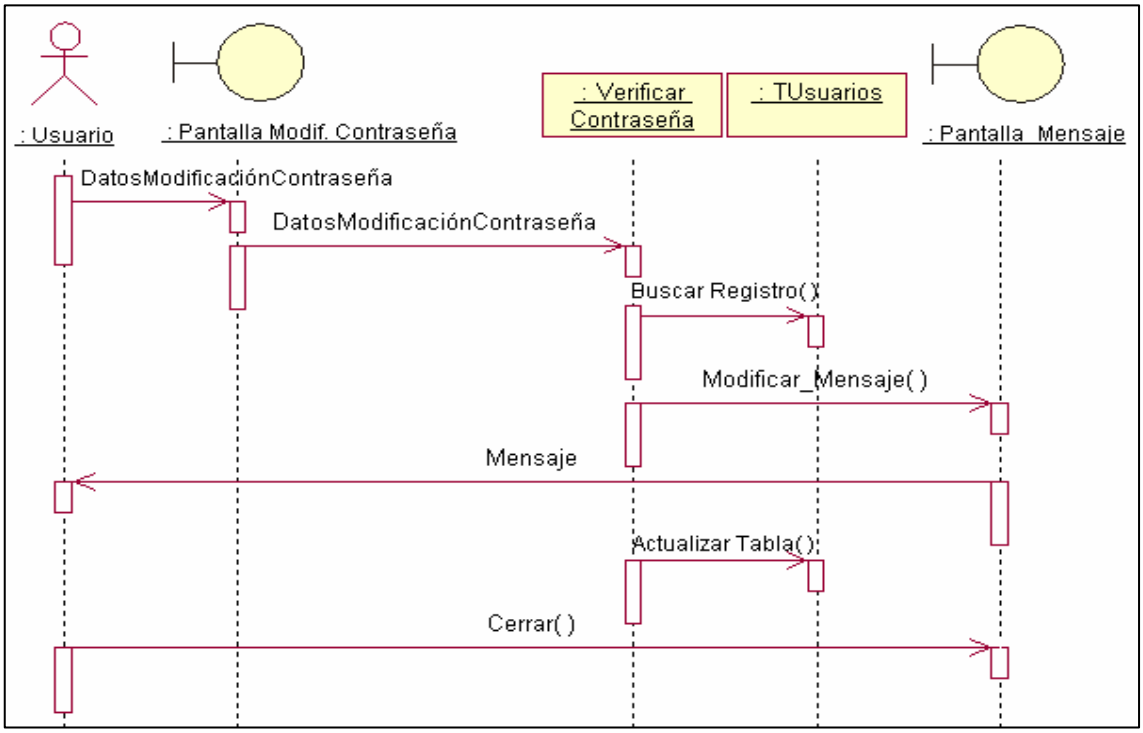

Figura 4.15. Diagrama de Secuencia de la realización de Modificar Contraseña Correcta

IV.3.3.2. Flujo de sucesos: Modificar Contraseña Incorrecta

El usuario ingresa su nombre de usuario, contraseña, la nueva contraseña y la confirmación de la nueva contraseña (DatosModificaciónContraseña), entonces se verifica la existencia del usuario con la correspondiente contraseña y que la nueva contraseña sea la misma a la confirmación ingresada (DatosModificaciónContraseña y Buscar Registro). Si no existe el usuario con esa contraseña, o si la nueva contraseña y su confirmación no coinciden, entonces se envía un mensaje de error indicando la situación (Modificar_Mensaje y Mensaje).

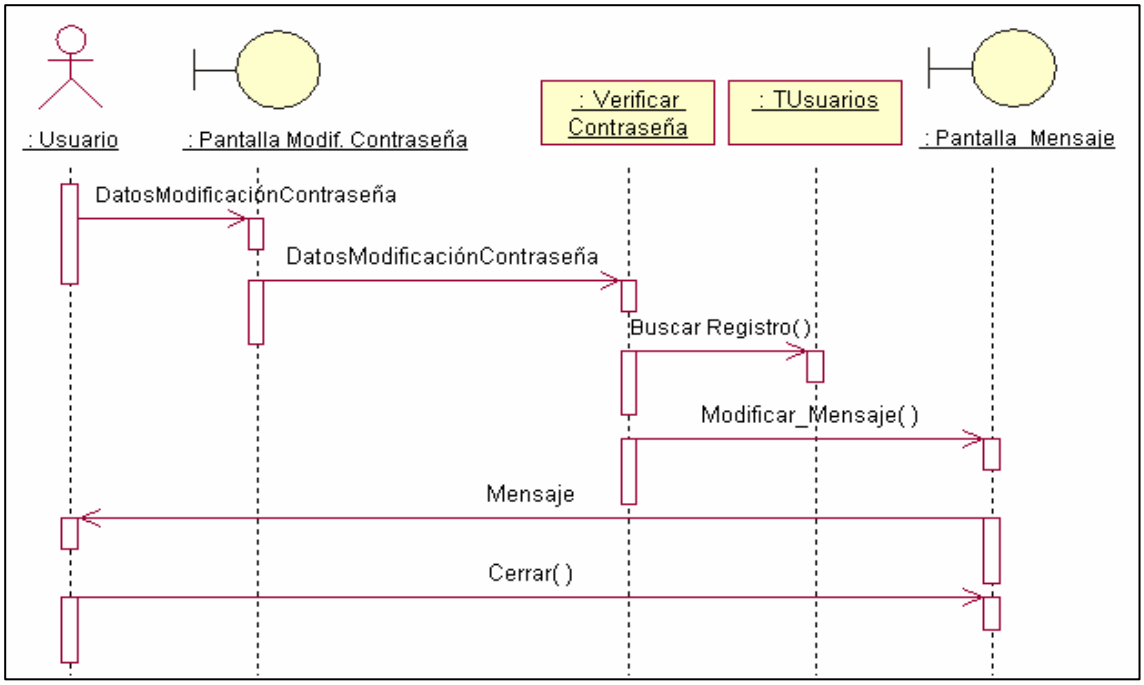

Figura 4.16. Diagrama de Secuencia de la realización de Modificar Contraseña Incorrecta

IV.3.4. CASO DE USO: DEFINICIÓN DE TAREAS

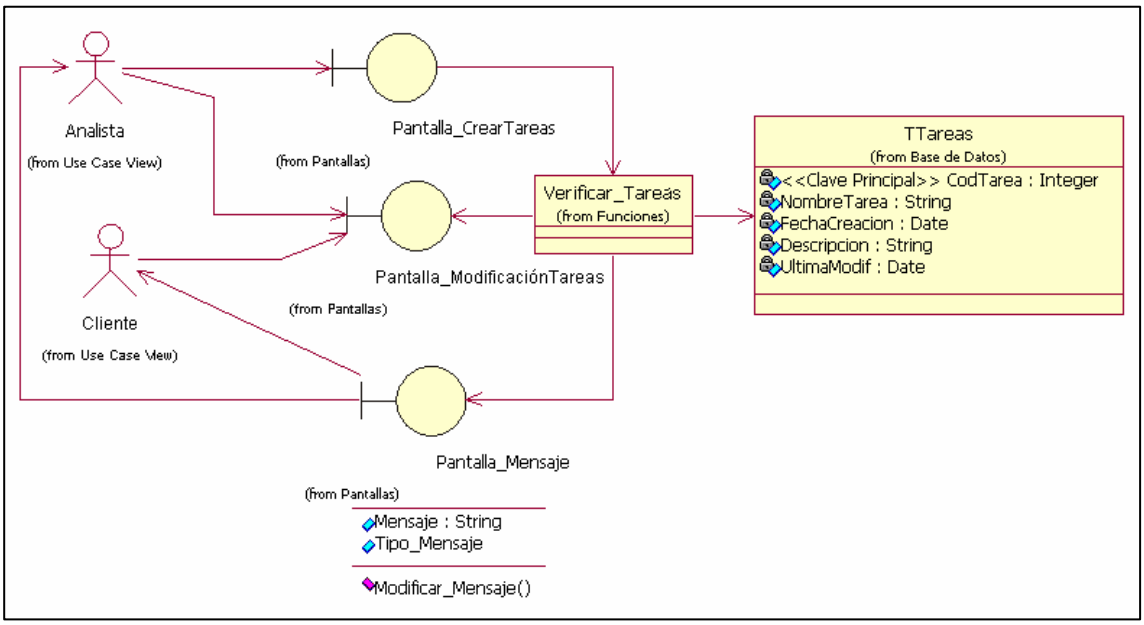

Figura 4.17. Diagrama de Clases Análisis de la realización del caso de uso Definición de Tareas

IV.3.4.1. Flujo de sucesos: Alta Tareas Correcta

El Analista debe ingresar los datos de una nueva tarea, es decir nombre, fecha de creación y descripción de la misma (Datos de Nueva Tarea), entonces se verifica la

existencia de la tarea en la tabla Tareas (Datos de Nueva Tarea y Buscar Registro). Si no existe la tarea que se quiere ingresar, entonces se agrega la tarea a la tabla Tareas (Agregar Registro al Final). Luego se informa al Analista (Modificar Mensaje, Mensaje).

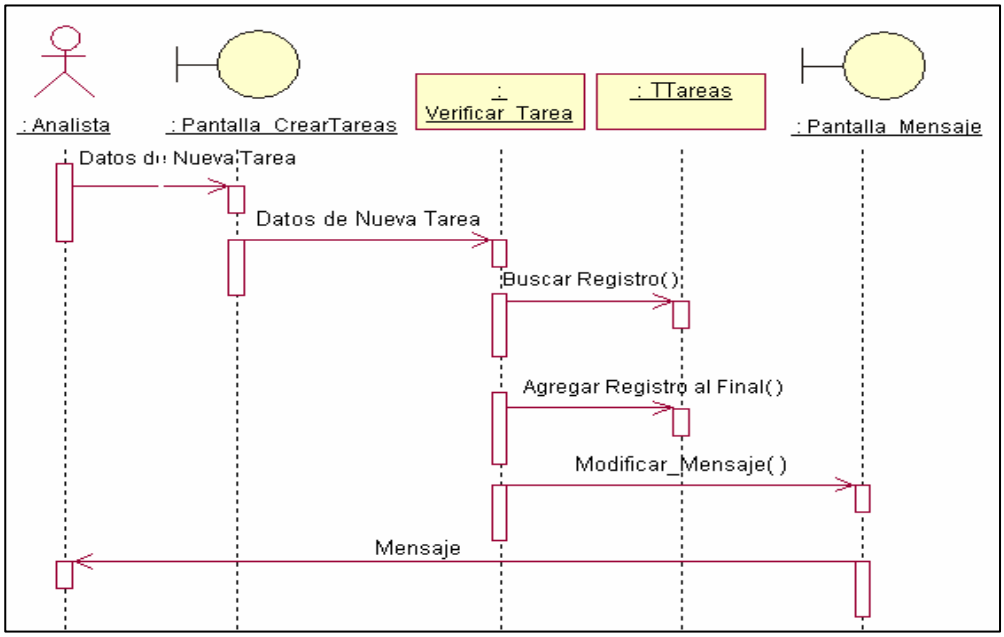

Figura 4.18. Diagrama de Secuencia de la realización de Alta Tareas Correcta

IV.3.4.2. Flujo de sucesos: Alta Tareas Incorrecta

El Analista debe ingresar los datos de una nueva tarea, es decir nombre, fecha de creación y descripción de la misma (Datos de Nueva Tarea), entonces se verifica la existencia de la tarea en la tabla Tareas (Datos de Nueva Tarea y Buscar Registro). Si existe la tarea que se quiere ingresar, entonces se informa al Analista la situación (Modificar Mensaje, Mensaje).

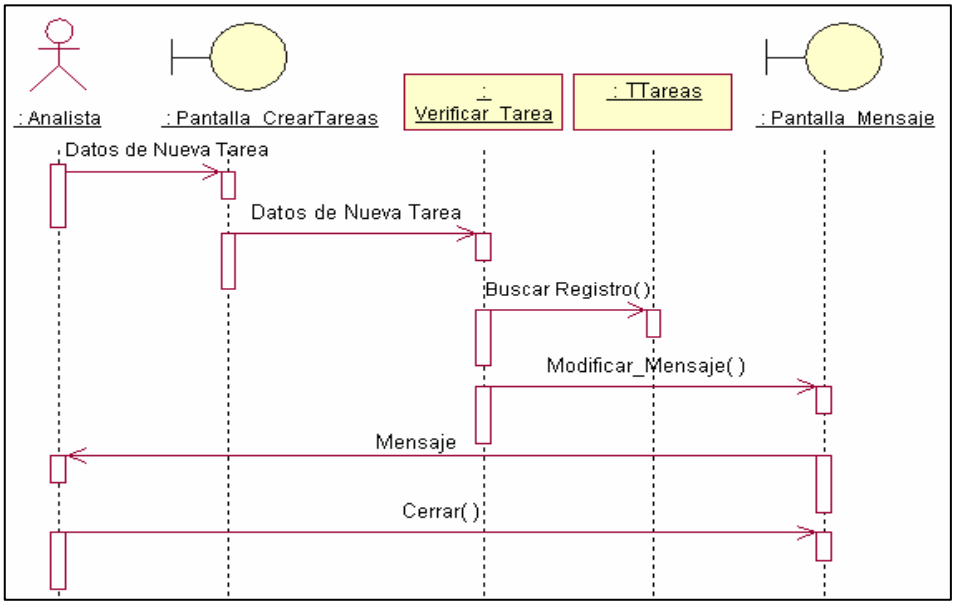

Figura 4.19. Diagrama de Secuencia de la realización de Alta Tareas Incorrecta

IV.3.4.3. Flujo de sucesos: Modificación Tareas Correcta

El Analista o el Cliente debe ingresar los datos de la tarea a modificar (Datos a Modificar), entonces se verifica la existencia de la tarea en la tabla Tareas (Datos a Modificar y Buscar Registro). Si existe la tarea que se quiere modificar, entonces se solicita al Analista o al Cliente que confirme la operación (Modificar Mensaje, Mensaje). Una vez confirmada, se actualiza la tabla Tareas (Actualizar Tabla).

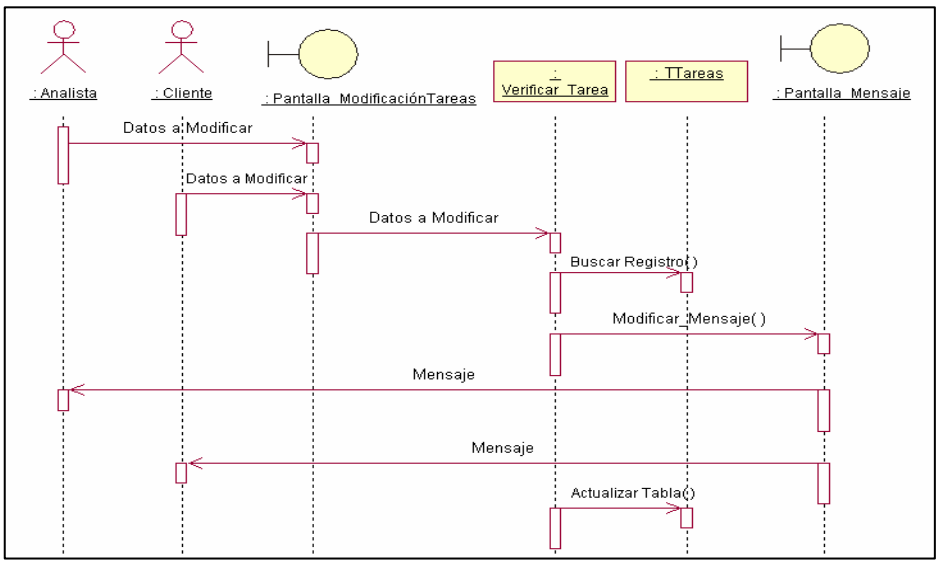

Figura 4.20. Diagrama de Secuencia de la realización de Modificación Tareas Correcta

IV.3.4.4. Flujo de sucesos: Modificación Tareas Incorrecta

El Analista o el Cliente debe ingresar los datos de la tarea a modificar (Datos a Modificar), entonces se verifica la existencia de la tarea en la tabla Tareas (Datos a Modificar y Buscar Registro).

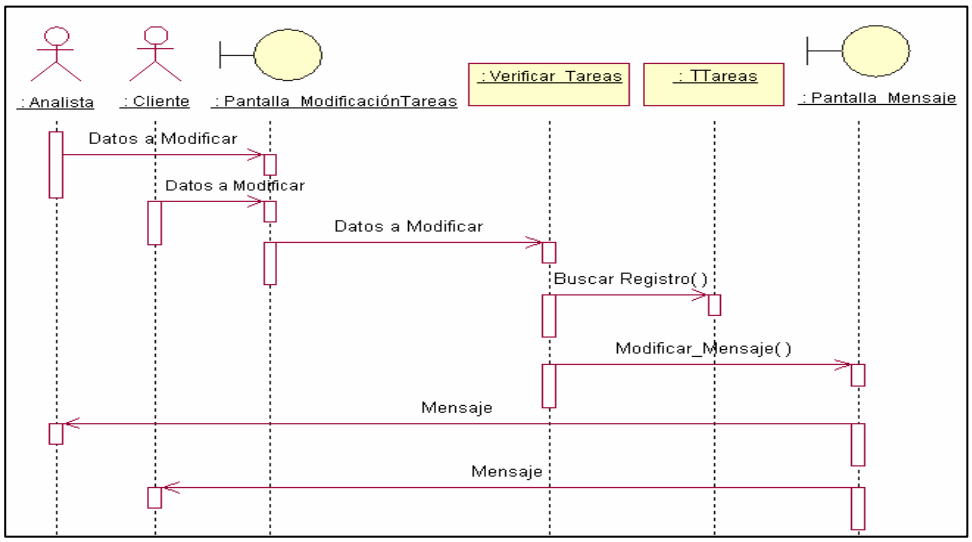

Figura 4.21. Diagrama de Secuencia de la realización de Modificación Tareas Incorrecta

Si no existe la tarea que se quiere modificar, entonces se informa al Analista o al Cliente la situación (Modificar Mensaje, Mensaje).

IV.3.5. CASO DE USO: REGISTRO DE HISTORIAS DE USUARIO

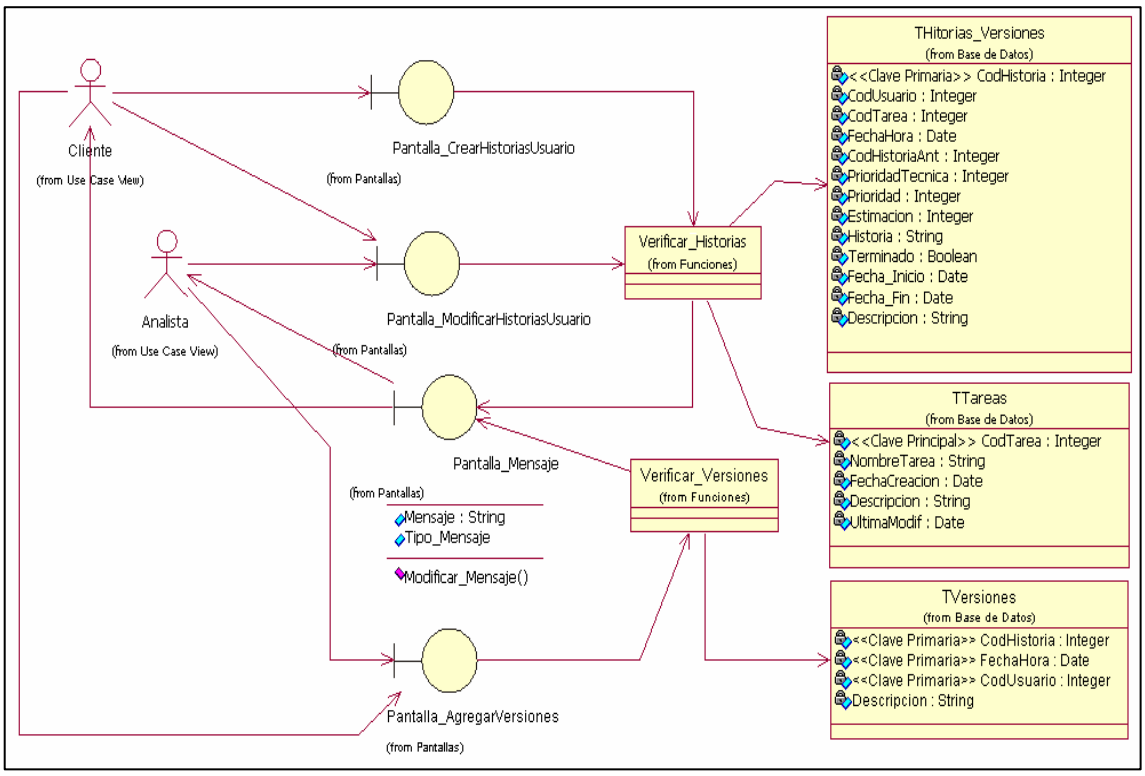

Figura 4.22. Diagrama de Clases Análisis de la realización del caso de uso Registro de Historias de Usuario

IV.3.5.1. Flujo de sucesos: Crear Historias de Usuario Correcta

El Cliente debe ingresar los datos de una nueva historia, es decir seleccionar la tarea a la que hace referencia, establecer la descripción, prioridad, etc. (Datos Historia Nueva), entonces se verifica la existencia de la tarea en la tabla Tareas y de la historia en la tabla Historias_Versiones (Datos Historia Nueva y Buscar Registro).

Si existe la tarea y no existe la historia que se quiere ingresar, entonces se agrega la historia a la tabla Hitorias_Versiones (Agregar Registro al Final). Luego se informa al Cliente (Modificar Mensaje, Mensaje).

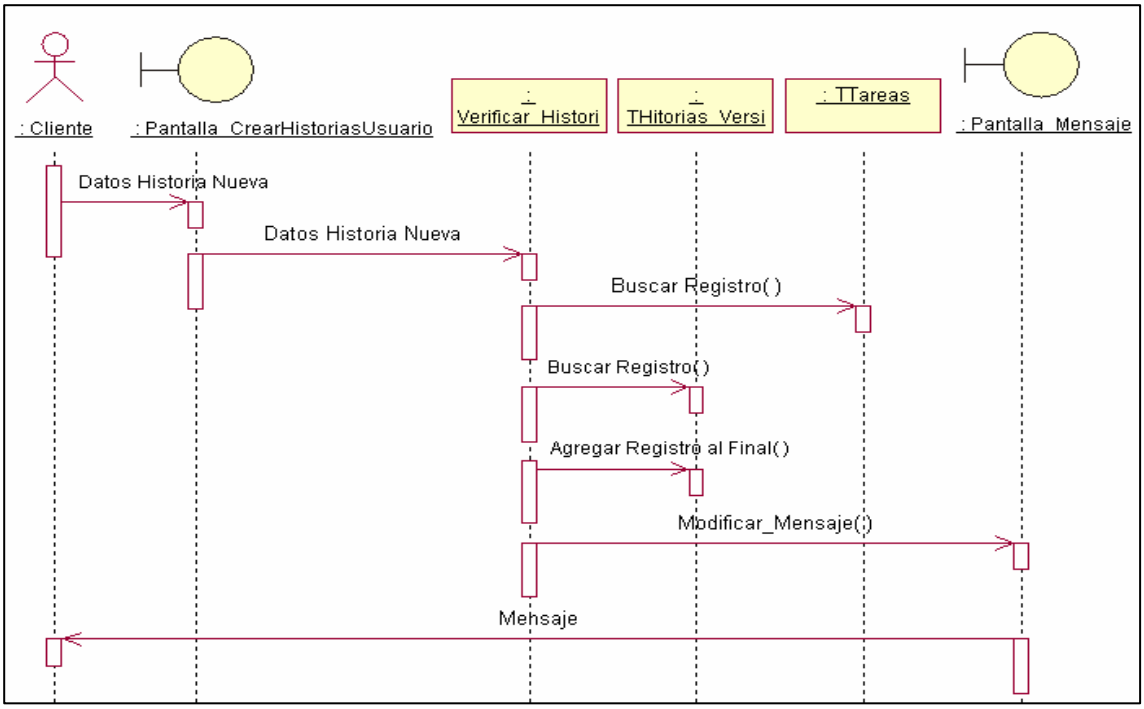

Figura 4.23. Diagrama de Secuencia de la realización de Crear Historias de Usuario Correcta

IV.3.5.2. Flujo de sucesos: Crear Historias de Usuario Incorrecta

El Cliente debe ingresar los datos de una nueva historia, es decir seleccionar la tarea a la que hace referencia, establecer la descripción, prioridad, etc. (Datos Historia Nueva), entonces se verifica la existencia de la tarea en la tabla Tareas y de la historia en la tabla Historias_Versiones (Datos Historia Nueva y Buscar Registro). Si no existe la tarea o si ya existe la historia que se quiere ingresar, entonces se informa al Cliente la situación (Modificar Mensaje, Mensaje).

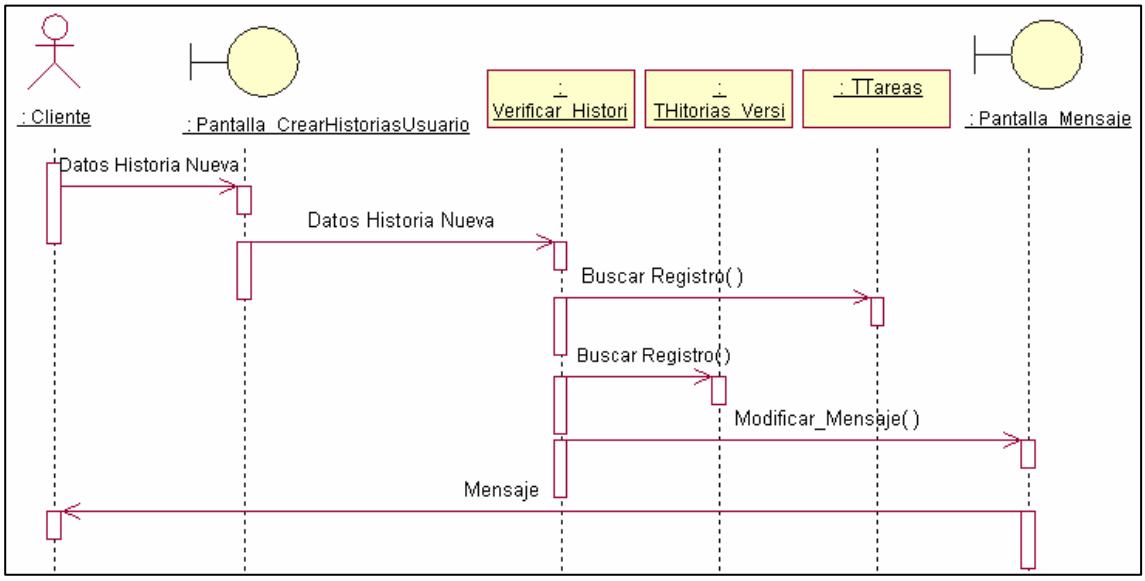

Figura 4.24. Diagrama de Secuencia de la realización de Crear Historias de Usuario Incorrecta

IV.3.5.3. Flujo de sucesos: Modificación Historias Correcta

El Analista o el Cliente debe ingresar los datos de la historia a modificar (Datos Modificación Historias), entonces se verifica la existencia de la tarea en la tabla Tareas y la historia en la tabla Historias_Versiones (Datos Modificación Historias y Buscar Registro). Si existe la historia que se quiere modificar, entonces se solicita al Analista o al Cliente que confirme la operación (Modificar Mensaje, Mensaje); una vez confirmada se actualiza la tabla Historias_Versiones (Actualizar Tabla).

En caso de que se trate de modificar una historia que ya fue marcada por el Resto del Equipo de Desarrollo como una historia terminada, entonces se informa la situación al Analista o al Cliente (Modificar Mensaje, Mensaje) y, si se confirma la operación, se marca esta tarea como no terminada de modo que posteriormente sea revisada nuevamente por el Resto del Equipo de Desarrollo.

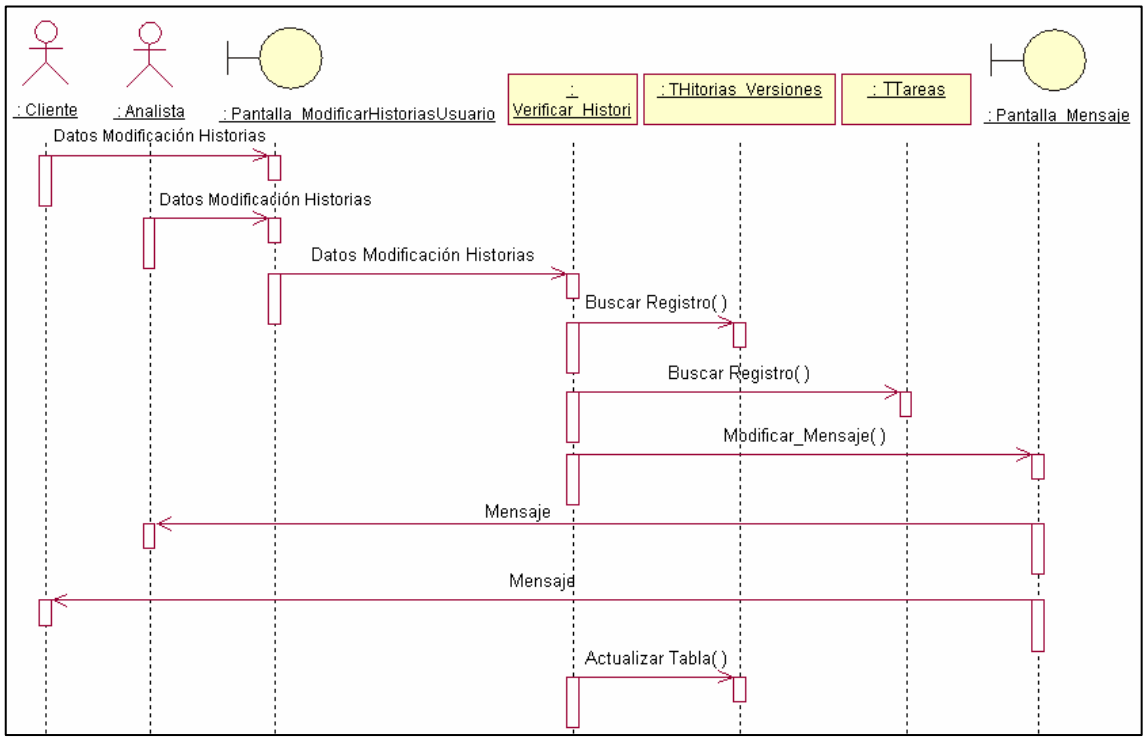

Figura 4.25. Diagrama de Secuencia de la realización de Modificación Historias Correcta

IV.3.5.4. Flujo de sucesos: Modificación Historias Incorrecta

El Analista o el Cliente debe ingresar los datos de la historia a modificar (Datos Modificación Historias), entonces se verifica la existencia de la tarea en la tabla Tareas y la historia en la tabla Historias_Versiones (Datos Modificación Historias y Buscar

Registro). Si no existe la historia que se quiere modificar, entonces se informa al Analista o al Cliente la situación (Modificar Mensaje, Mensaje).

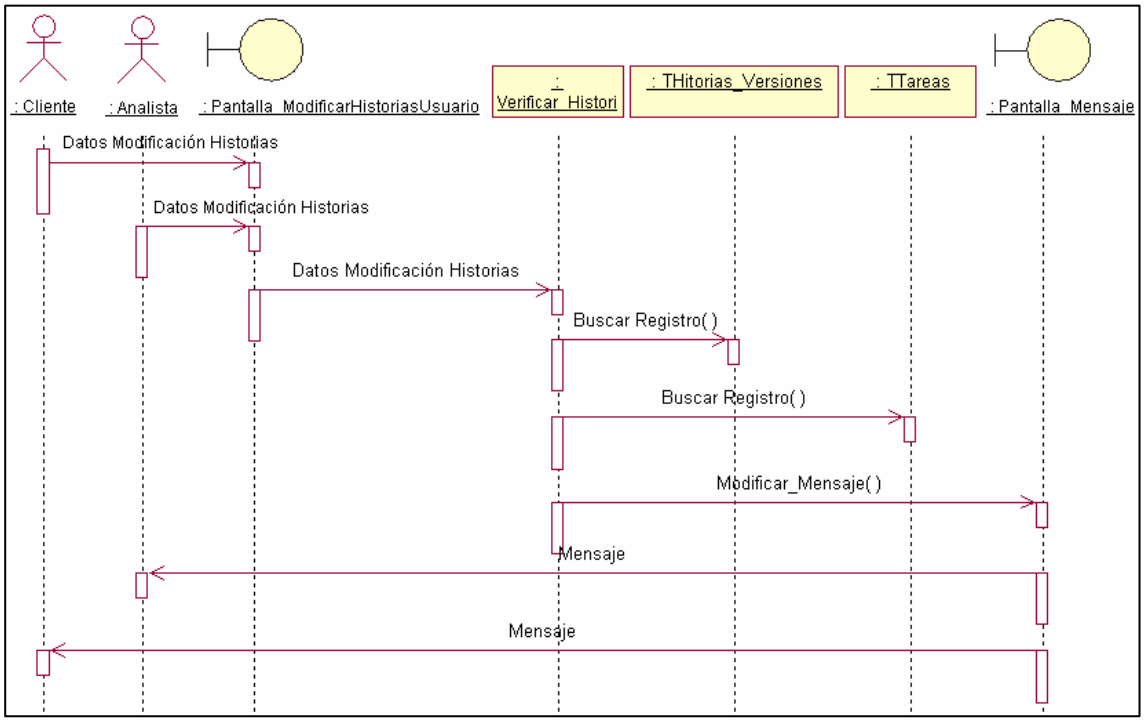

Figura 4.26. Diagrama de Secuencia de la realización de Modificación Historias Incorrecta

IV.3.5.5. Flujo de sucesos: Agregar Versiones Correcta

El Cliente o el Analista ingresan los datos de una nueva versión de una historia, es decir seleccionar la tarea y la historia a la que hace referencia, luego se descibe la versión (Datos Versión), entonces se verifica la existencia de la tarea en la tabla Tareas, de la historia en la tabla Historias_Versiones y de la versión en al tabla Versiones (Datos Versión y Buscar Registro).

Si existe la tarea y la historia, pero no existe la versión, entonces se agrega la nueva versión a la tabla Versiones (Agregar Registro al Final). Luego se informa al Cliente (Modificar Mensaje, Mensaje).

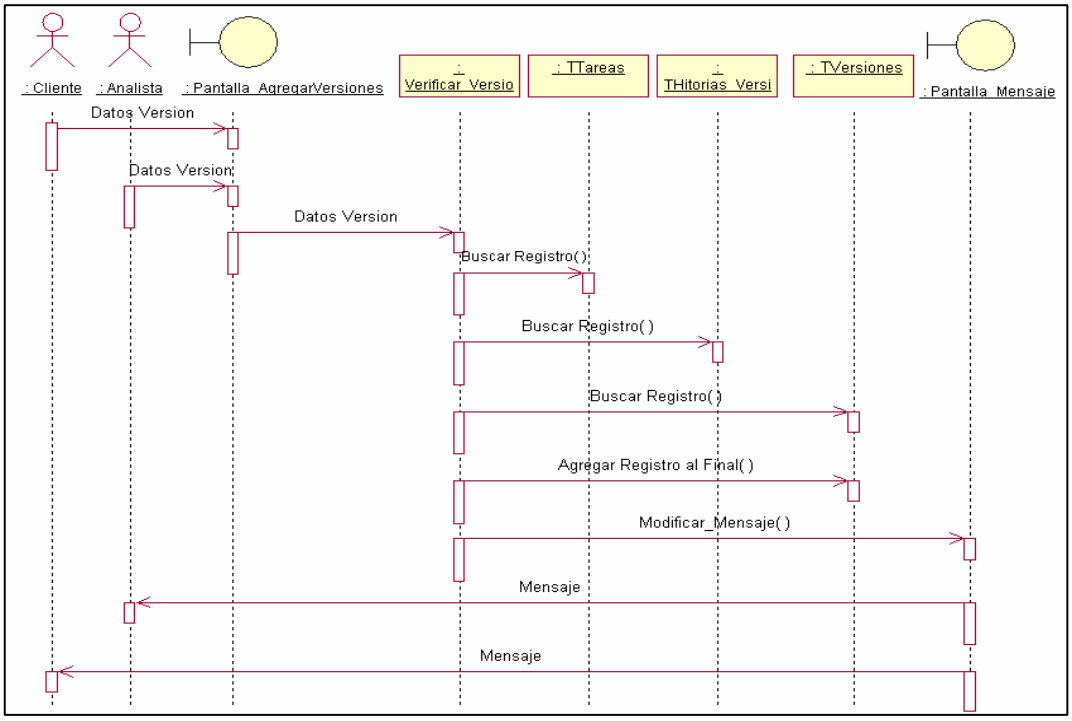

Figura 4.27. Diagrama de Secuencia de la realización de Agregar Versiones Correcta

IV.3.5.6. Flujo de sucesos: Agregar Versiones Incorrecta

El Cliente o el Analista ingresan los datos de una nueva versión de una historia, es decir seleccionar la tarea y la historia a la que hace referencia, luego se descibe la versión (Datos Versión), entonces se verifica la existencia de la tarea en la tabla Tareas, de la historia en la tabla Historias_Versiones y de la versión en al tabla Versiones (Datos Versión y Buscar Registro). Si no existe la tarea, o la historia, o si existe la versión, entonces se informa al Cliente o Analista la situación (Modificar_Mensaje, Mensaje)

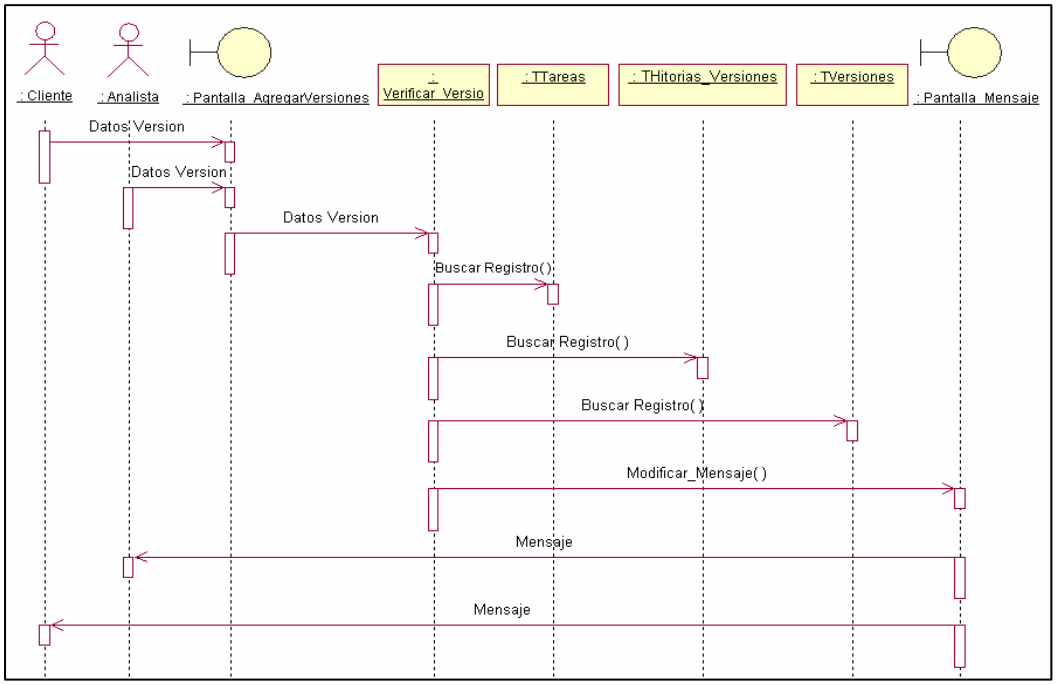

Figura 4.28. Diagrama de Secuencia de la realización de Agregar Versiones Incorrecta

IV.3.6. CASO DE USO: CONSULTA DE HISTORIAS DE USUARIO

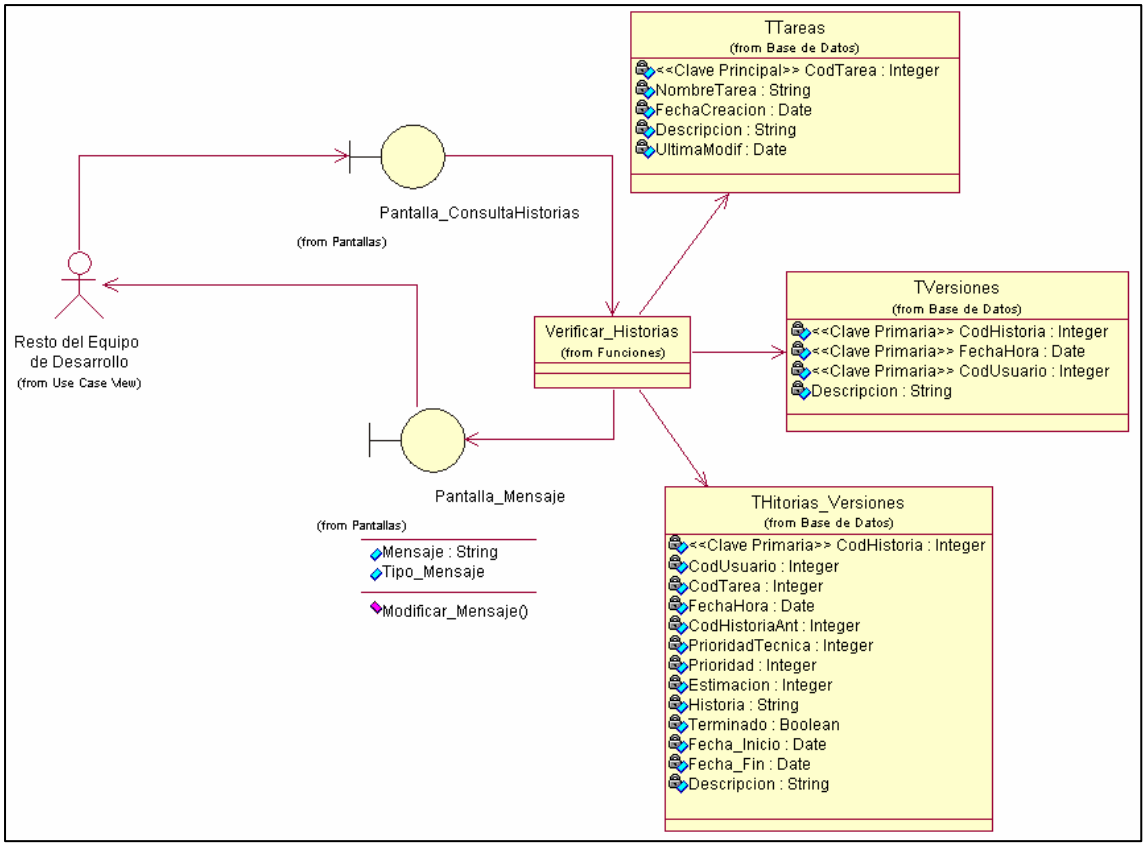

Figura 4.29. Diagrama de Clases Análisis de la realización del caso de uso Consulta de Historias de Usuario

IV.3.6.1. Flujo de sucesos: Consulta de Historias de Usuario

El Resto del Equipo de Desarrollo ingresa los datos de la tarea y la historia por consultar (Datos Tareas Historias), entonces se verifica los datos ingresados en las tablas Tareas, Historias_Versiones y Versiones (Datos Tareas Historias y Buscar Registro).

En el caso de que la historia consultada sea inicie o concluya, entonces se actualiza la tabla Historias_Versiones y se solicita confirmación de la operación (Modificar Mensaje, Mensaje, Actualizar Tabla).

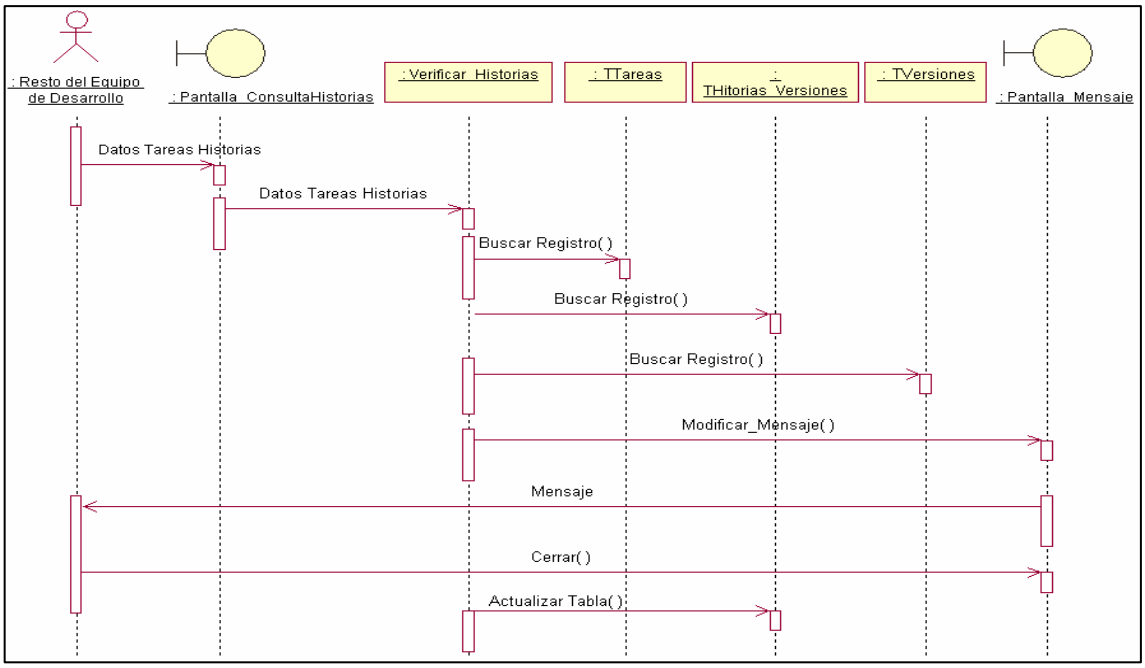

Figura 4.30. Diagrama de Secuencia de la realización de Consulta de Historias de Usuario

IV.3.7. CASO DE USO: VER ESTADO DEL PROYECTO

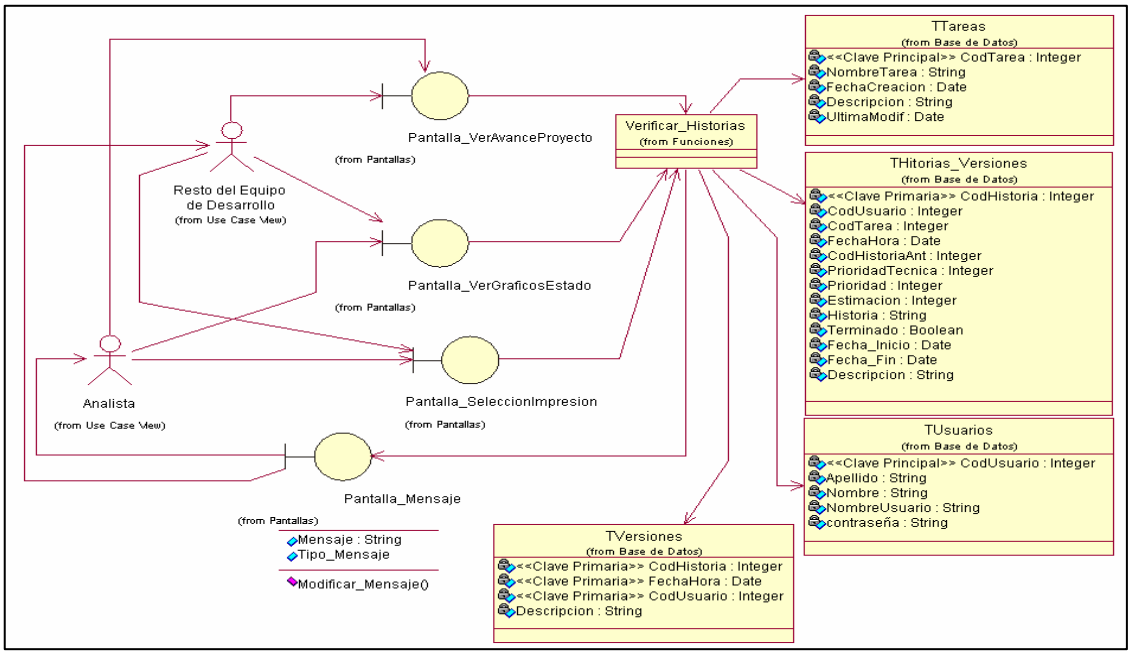

Figura 4.31. Diagrama de Clases Análisis de la realización del caso de uso Ver Estado del Proyecto

IV.3.7.1. Flujo de sucesos: Ver Avance del Proyecto

El Analista o el Resto del Equipo de Desarrollo deben ingresar los datos que le permitan ver el avance del proyecto, es decir el estado de realización de las historias de

usuario (Datos Proyecto), entonces se verifica los datos en las tablas Tareas e Historias_Versiones (Datos Proyecto y Buscar Registro).

Si se produce alguna situación no prevista, entonces se informa al Analista o al Resto del Equipo de Desarrollo la situación (Modificar Mensaje, Mensaje).

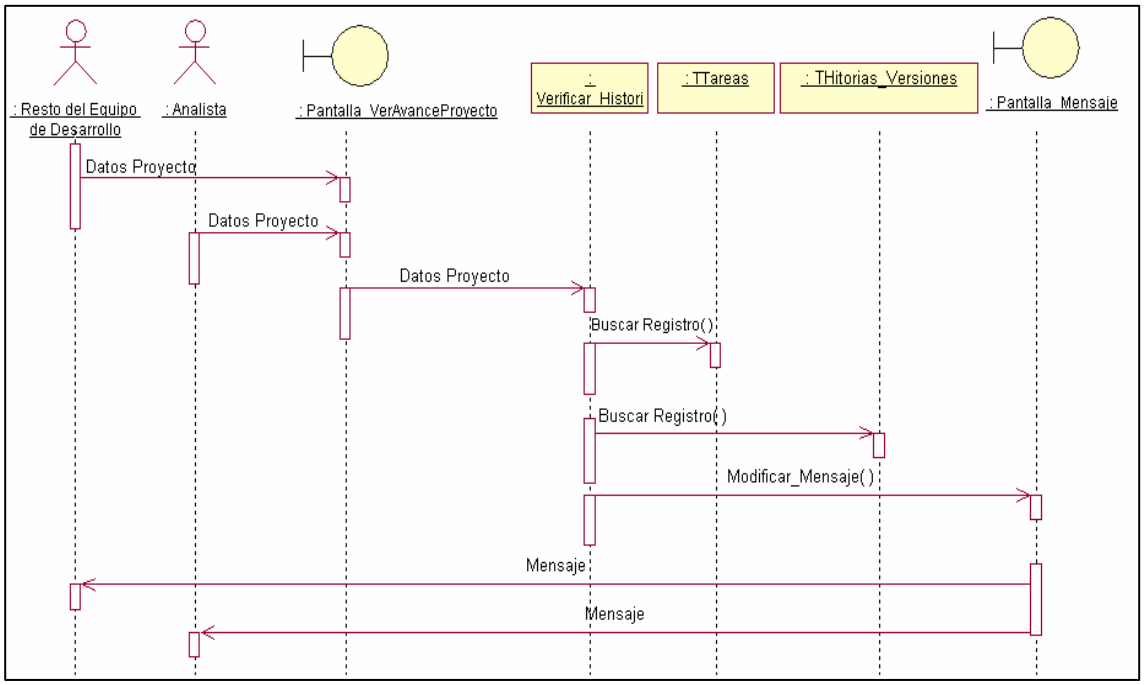

Figura 4.32. Diagrama de Secuencia de la realización de Ver Avance del Proyecto

IV.3.7.2. Flujo de sucesos: Ver Gráficos del Estado del Proyecto

El Analista o el Resto del Equipo de Desarrollo deben ingresar los datos de la tarea e historia de las cuales desean ver en forma gráfica el estado del proyecto (Datos Proyecto), entonces se verifica los datos en las tablas Tareas e Historias_Versiones (Datos Proyecto y Buscar Registro).

Si se produce alguna situación no prevista, entonces se informa al Analista o al Resto del Equipo de Desarrollo la situación (Modificar Mensaje, Mensaje).

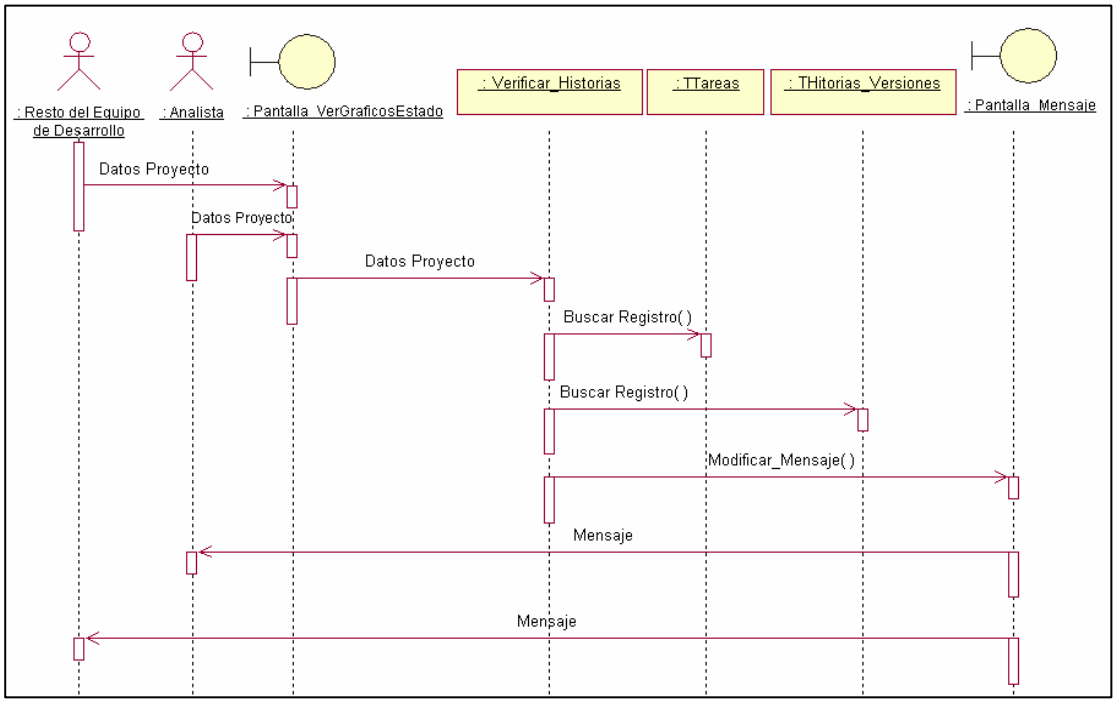

Figura 4.33. Diagrama de Secuencia de la realización de Ver Gráficos del Estado del proyecto

IV.3.7.3. Flujo de sucesos: Selección de Forma de Impresión

El Analista o el Resto del Equipo de Desarrollo selecciona la forma de impresión que se desea obtener, por eljemplo por usuario o por tareas. Una vez realizada la selección (OpcionImpresion), entonces se realizan las consultas en las tablas Tareas, Historias_Versiones, Usuarios y Versiones (Filtrar()). Luego, se informa al Analista o al Resto del Equipo de Desarrollo que se realizará la impresión (Modificar Mensaje, Mensaje).

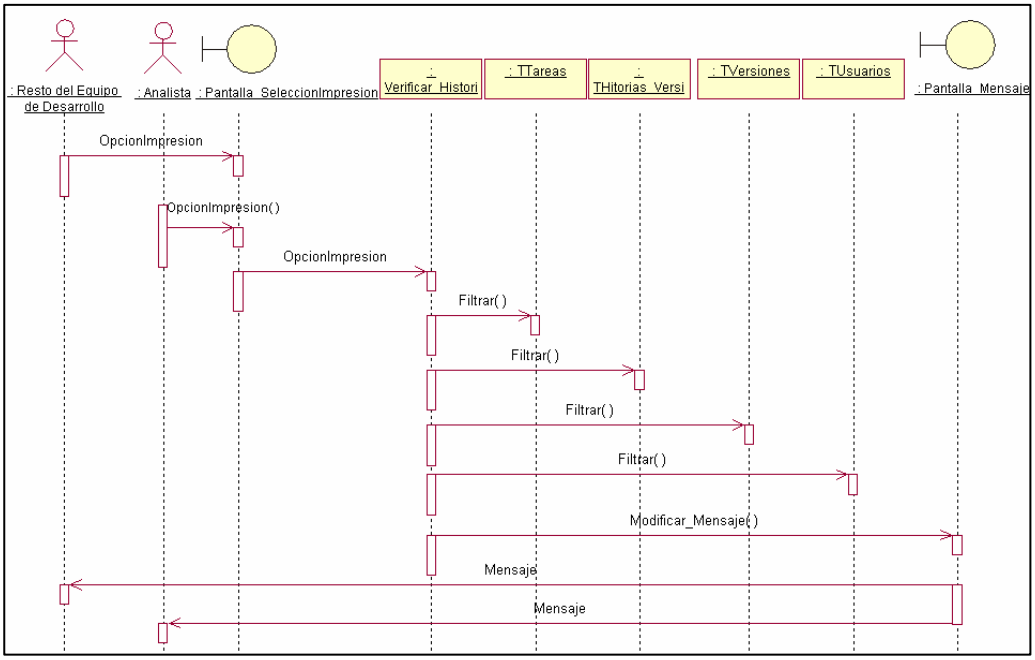

Figura 4.34. Diagrama de Secuencia de la realización de Selección de Forma de Impresión

CAPÍTULO V

DESCRIPCIÓN DEL PROTOTIPO DE HGH

V.1. DESCRIPCIÓN DEL PROTOTIPO

El prototipo de la HGH se construyó empleando C++ Builder 6.0. Como sistema de gestión de base de datos, se utilizó Interbase 6.5 de licencia libre para cinco conexiones, lo cual permite que el prototipo sea usado en red por varios usuarios en forma concurrente. A continuación se muestra el menú principal del prototipo (ver Figura 5.1).

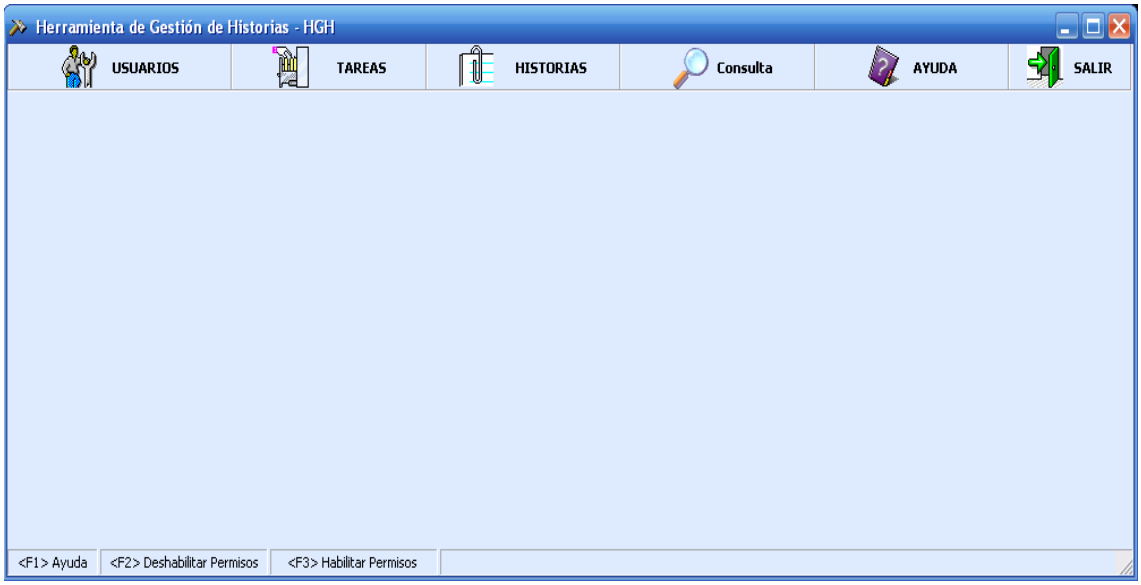

Figura 5.1. Menú principal del prototipo de la HGH

V.1.1. ADMINISTRACIÓN DE USUARIOS

V.1.1.1. Control de Acceso

El prototipo posee un control de acceso de usuarios que permite definir las opciones disponibles para todas las personas involucradas en el desarrollo del software en función del rol que cada uno desempeña; por ejemplo, se puede diferenciar al cliente, los usuarios finales del sistema, el analista, los programadores, etc. (ver figura 5.2).

Figura 5.2. Control de Acceso

V.1.1.2. Usuarios

La opción Usuarios permite crear nuevos usuarios o bien modificar los datos y roles de usuarios ya existentes (ver Figura 5.3) y cambiar la contraseña de acceso.

Figura 5.3. Administración de Usuarios

Al agregar o modificar un usuario, se puede:

- marcarlo como analista, para indicar que esta persona puede, por ejemplo, revisar o modificar todas las historia de usuario, aún las no creadas por él;
- o asignar roles, lo cual determina a qué función de la herramienta puede acceder esta persona (ver Figura 5.4).

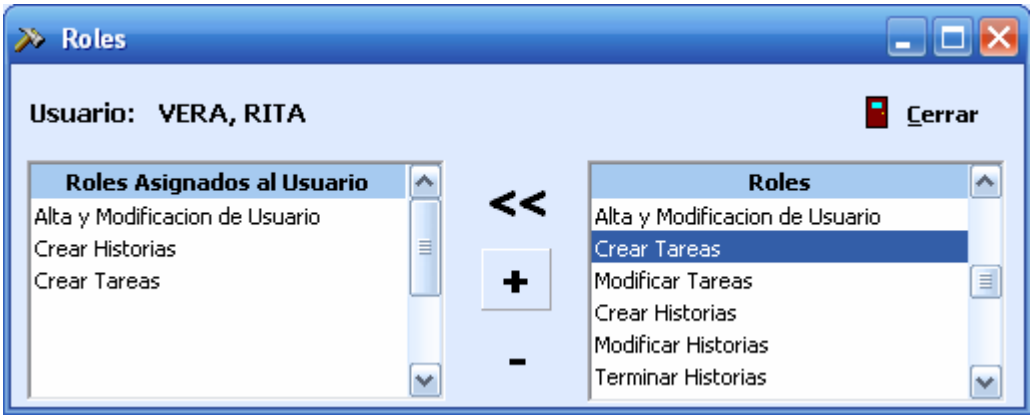

Figura 5.4. Asignar Roles

V.1.2. TAREAS

V.1.2.1. Crear Nueva Tarea

La Administración de Tareas es usada por el cliente y por el analista para crear las Tareas que el sistema debe realizar (ver Figura 5.5), para ello el analista utilizará la información recabada durante la fase de captura utilizando etnografía aplicada. La información que se debe incluir por tarea es el nombre de la misma, una breve descripción y la fecha de creación (por defecto, la fecha de creación es la fecha actual).

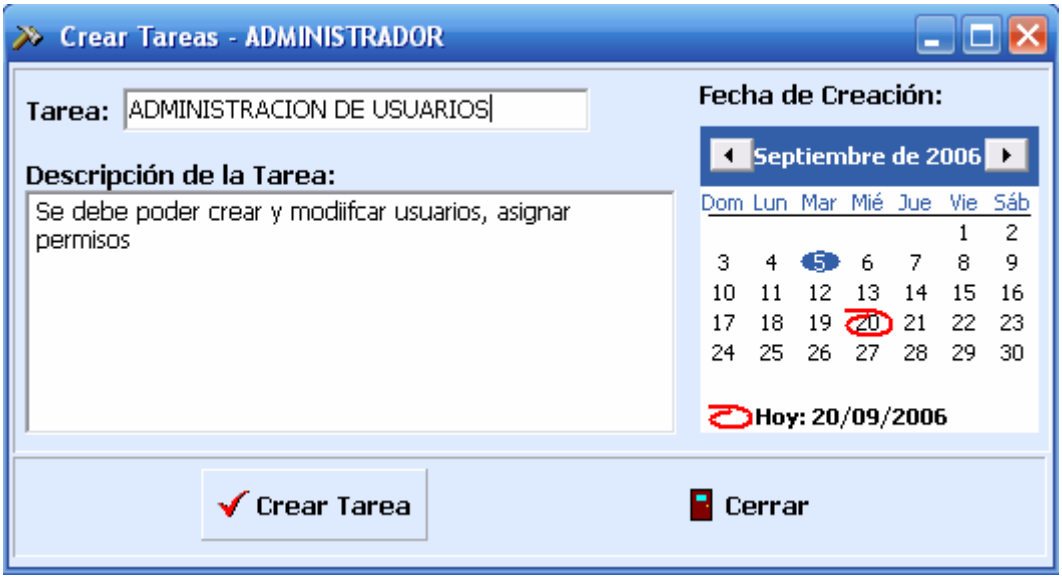

Figura 5.5. Alta de Tareas

V.1.2.2. Modificar una Tarea Existente

El cliente o el analista puede modificar una tarea ya creada (ver Figura 5.6). Una vez seleccionada la tarea, sólo se permite modificar la descripción de la misma.

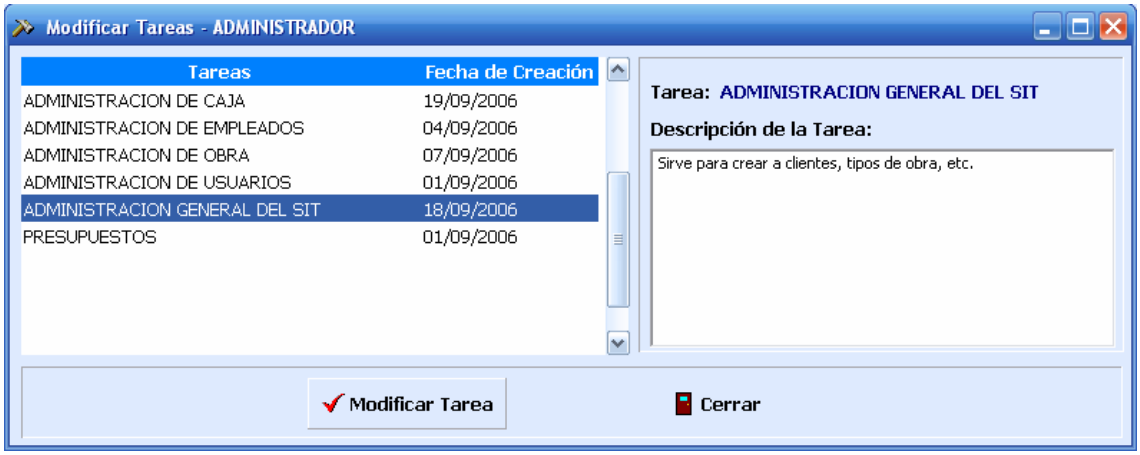

Figura 5.6. Modificar de Tareas

V.1.3. ADMINISTRACIÓN DE HISTORIAS

V.1.3.1. Crear Historia de Usuario

Cualquier persona involucrada en el desarrollo del proyecto, puede crear una historia de usuario (ver Figura 5.7); en lugar de escribir en fichas de papel (como es habitual) con su propio vocabulario, tal y como ellos ven las necesidades del sistema, se propone que ingresen la información a la herramienta.

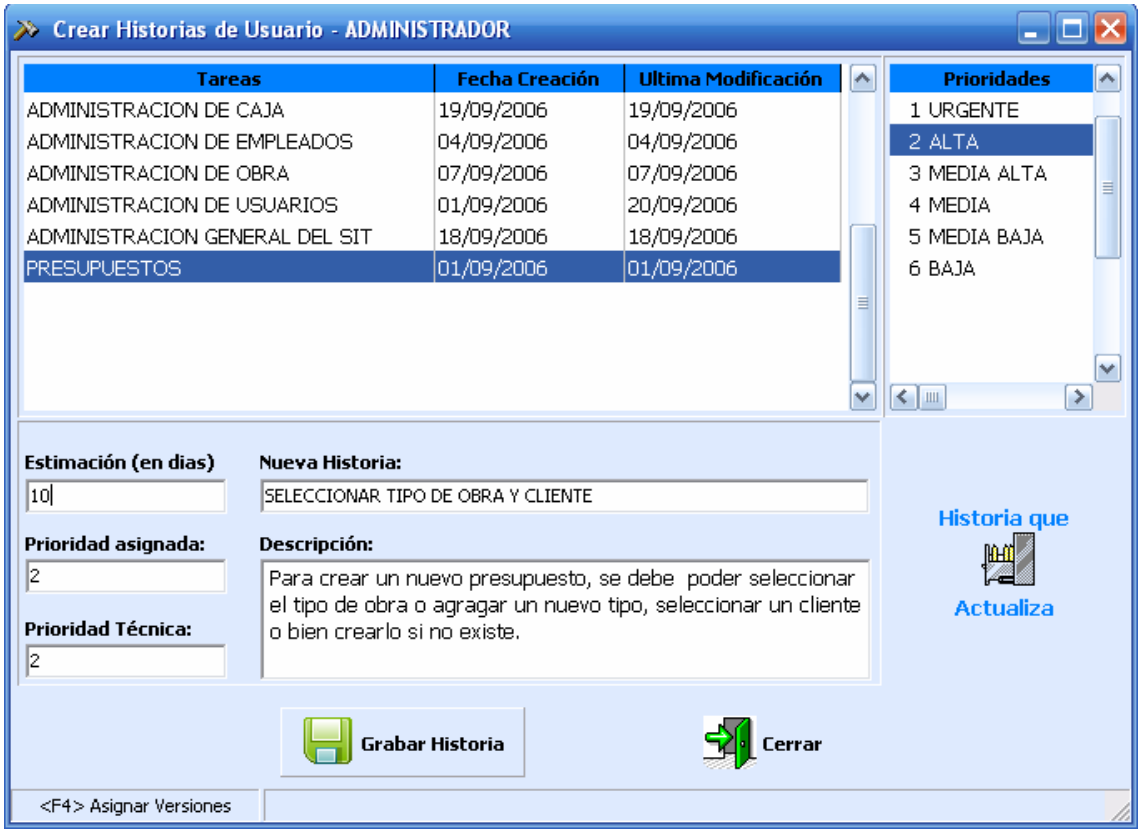

Figura 5.7. Crear Historias de Usuario

Primero se solicita seleccionar la tarea a la cual la historia hará referencia, y luego se debe determinar una estimación, la prioridad (las prioridades se las puede ver en forma tabular en la parte superior izquierda) y la prioridad técnica tentativa, un nombre y una descripción de la historia, y también se puede realizar su trazabilidad (es decir establecer una relación si es que existe) con otra historia definida anteriormente *"Historia que Actualiza"*.

V.1.3.2. Modificar Historia de Usuario

Cada usuario de la herramienta puede modificar los datos ingresados referentes a las historias creadas por él mismo; a excepción del Administrador (analista del sistema) quien puede modificar cualquier historia, puesto que es quien determina la estimación y la prioridad técnica aproximada a cada historia. Para ello, selecciona la tarea y elige la historia a modificar (ver Figura 5.8).

Figura 5.8. Modificar Historias de Usuario

V.1.3.3. Agregar versiones a las Historias de Usuario

También se pueden agregar versiones a las historia de usuario definidas (ver Figura 5.9).

Cualquier usuario de la herramienta selecciona la tarea, luego la historia, y después puede ver las distintas versiones de la historia, detalles de la historia en cuestión y se permite agregar una breve descripción de la nueva versión. Si la historia ha empezado a ser desarrollada por parte del resto de equipo de desarrollo, entonces aparece la fecha de inicio y en el caso de que haya sido terminada aparece la fecha de fin y se la marca con color amarillo.

En el caso de que a una historia terminada se le agregue una versión, entonces se la marca de modo que el resto del equipo de desarrollo pueda ver fácilmente que debe volver a revisar la historia ya desarrollada.

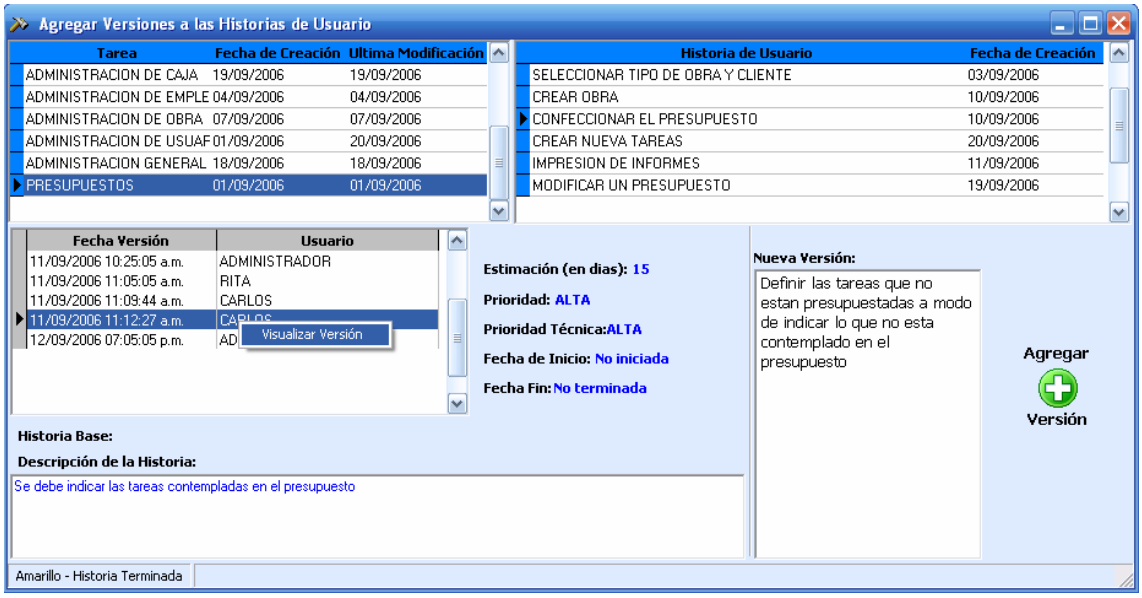

Figura 5.9. Agregar Versiones a las Historias de Usuarios

V.1.4. CONSULTA

V.1.4.1. Consultar y Grabar Historias de Usuario

El resto del equipo de desarrollo puede consultar los requisitos especificados, tanto en tareas e historias como en versiones (ver Figura 5.10).

También se utiliza esta parte de la herramienta para marcar el inicio y el fin de las historias, entonces puede suceder alguna de las siguientes situaciones:

- Si se marca el inicio de una historia y si su historia base aún no ha sido iniciada, entonces se muestra un mensaje de error indicando tal situación.
- Si se marca el fin de una historia y ésta aun no ha sido iniciada, entonces se marca como fecha de inicio el mismo día de finalización.

Del mismo modo, se utiliza esta ventana para verificar si las historias terminadas han sido modificadas o se les han agregado más versiones, para ello se marcan estas historias con color naranja, de modo que los desarrolladores puedan transformar lo realizado de acuerdo a las nuevas modificaciones.

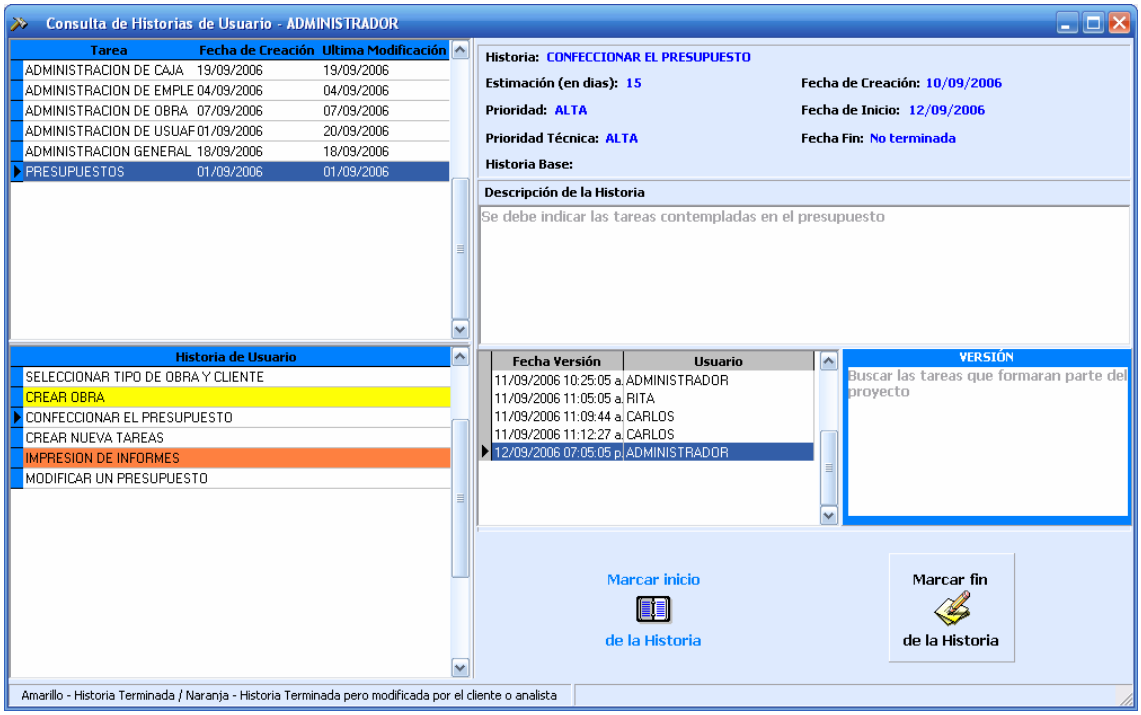

Figura 5.10. Consulta y Grabar Historias de Usuarios

V.1.4.2. Ver Información en Gráficos

La herramienta proporciona una visualización del estado del proyecto para determinar diferencias, por ejemplo entre estimaciones y duraciones reales, lo cual permitirá identificar desviaciones en el desarrollo del proyecto.

Se pueden realizar las visualizaciones por *"Historias"* o por *"Tareas"*. Por ejemplo, si se realiza la visualización *"Por Historias"*, se debe seleccionar la tarea (ver Figura 5.11).

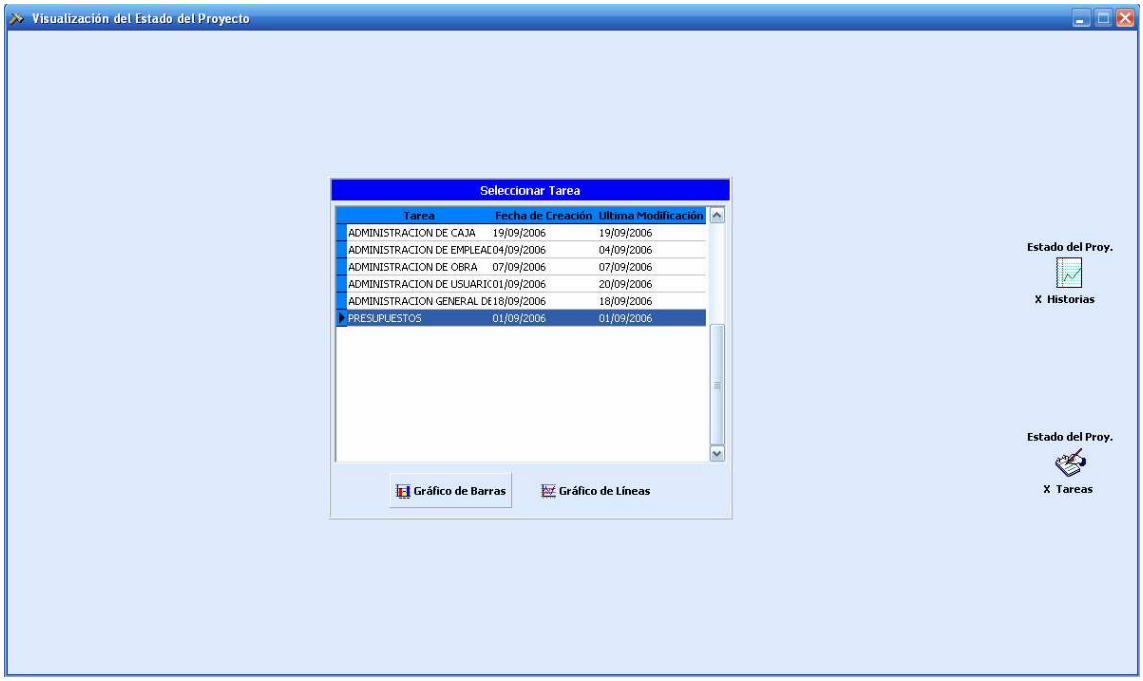

Figura 5.11. Seleccionar la tarea

Luego se puede optar por ver un gráfico de Barras (ver Figura 5.12) o de Líneas (ver Figura 5.13). Entonces aparecen graficadas las historias, con sus estimaciones y duraciones reales.

En el gráfico de barras, las fechas rojas indican la estimación real de las historias, las barras azules indican las duraciones reales de las historias terminadas, y las barras verdes indican la duración hasta la fecha actual de historias que aún no han sido terminadas.

Figura 5.12. Gráfico de Barras por Historias

En el gráfico de líneas, las líneas rojas indican la estimación real de las historias y las líneas azules las duraciones reales. En este gráfico no se distinguen las historias aún no terminadas.

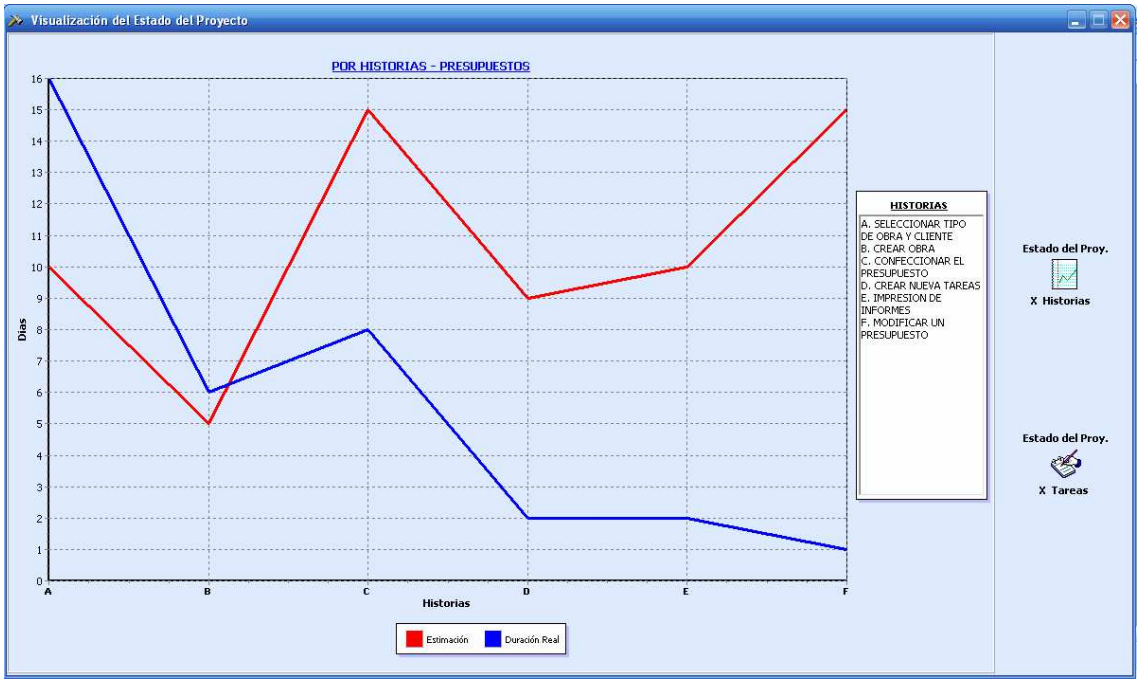

Figura 5.13. Gráfico de Líneas por Historias

En el caso de realizar la visualización *"Por Tareas"* (ver Figura 5.14), se pueden observar todas las tareas que el sistema debe realizar.

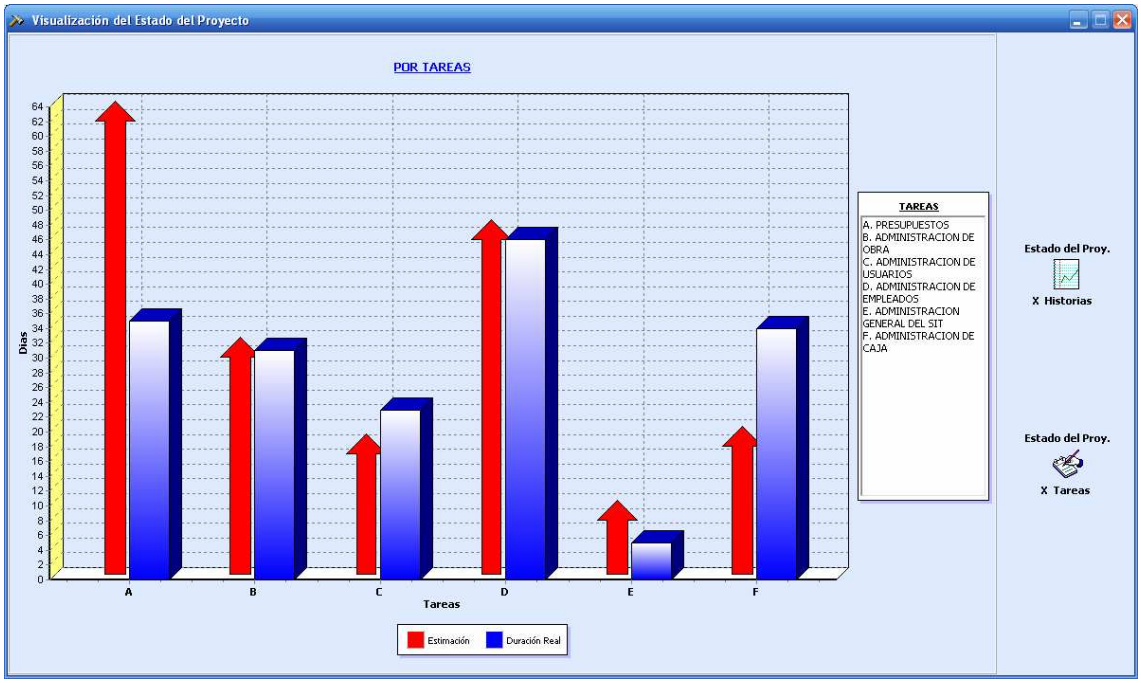

Figura 5.14. Gráfico por Tareas

Para indicar las estimaciones se usan las flechas rojas las cuales representan la suma de las estimaciones de las historias contenidas en cada tarea. Y para indicar las duraciones reales, se utilizan las barras azules, las cuales representan la suma de las duraciones reales de las historias contenidas en cada tarea, sin diferenciar las historias terminadas y las no terminadas.

V.1.4.3. Ver Avance del Proyecto

También, la herramienta permite visualizar el avance del proyecto hasta la fecha actual a través de diagramas de Gantt, ya sea por historias o por tareas.

Para obtener la visualización por historias se debe elegir *"Diagrama de Gantt por Historias"*, lo primero que se hace es seleccionar la tarea (de forma similar a la Figura 5.11) y luego se puede ver el diagrama de Gantt correspondiente (ver Figura 5.15).

En el diagrama se puede apreciar las duraciones reales de cada historia utilizando como parámetros las fechas de inicio y de fin de cada historia, en caso de que ésta haya sido marcada como terminada, o fecha actual para las no terminadas.

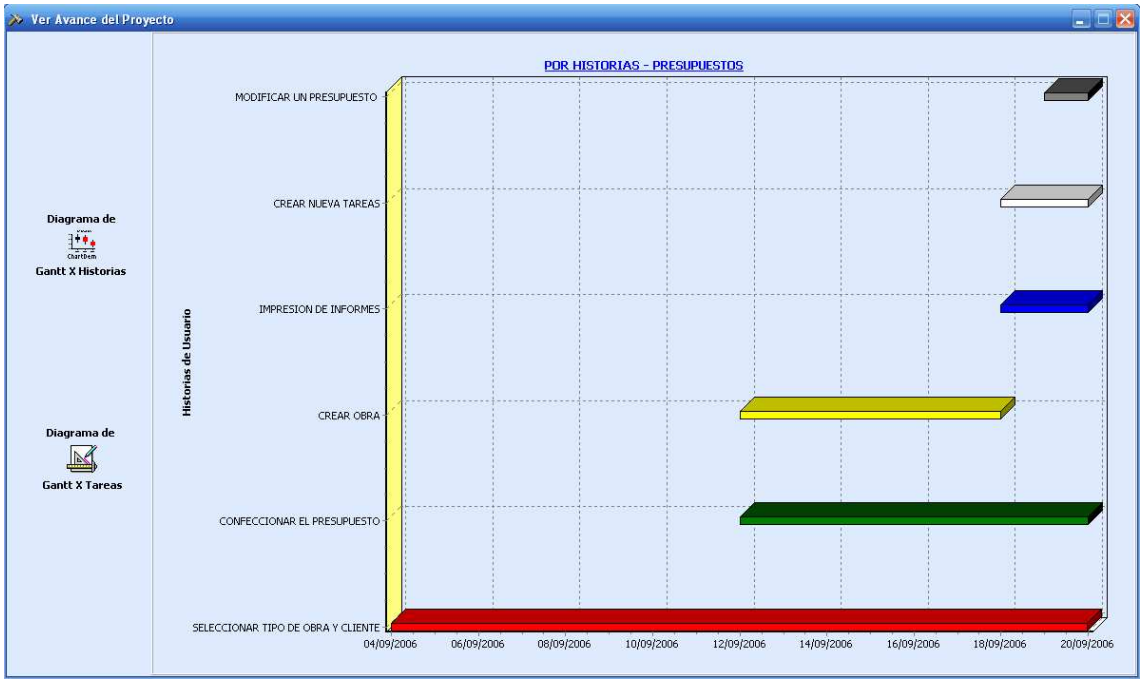

Figura 5.15. Diagrama de Gantt por Historias

En el caso de querer obtener la visualización por tareas, se selecciona la opción *"Diagrama de Gantt por Tareas"* (ver Figura 5.16). De este modo se puede ver las duraciones reales de las tareas cuyas historias hayan finalizado o no.

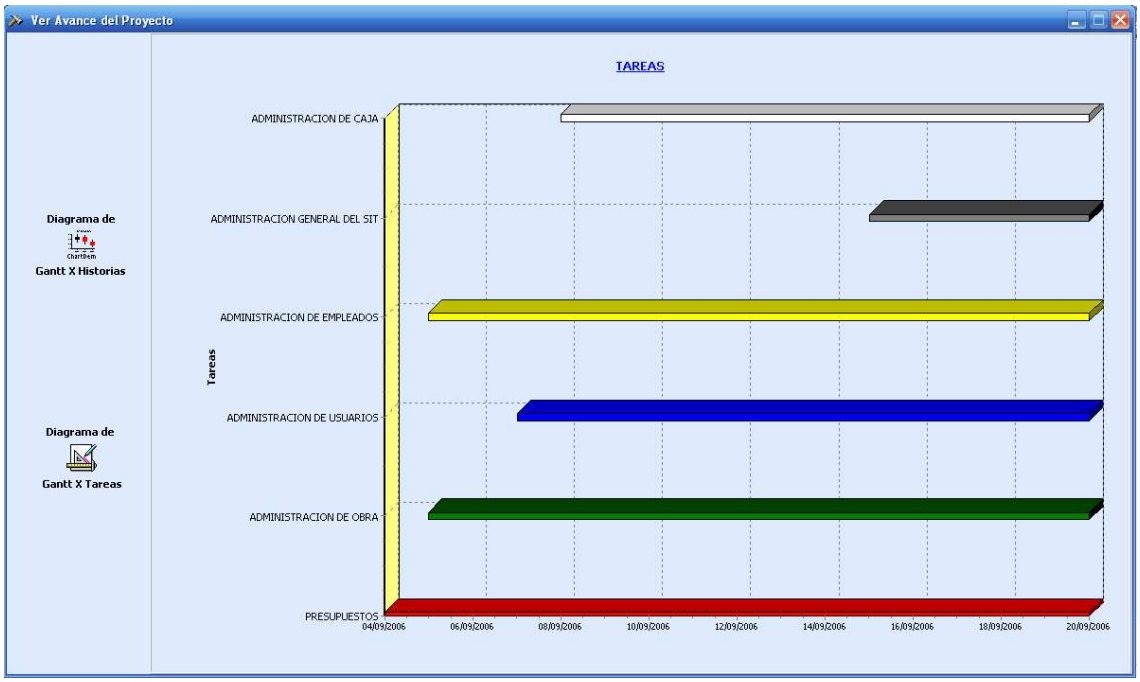

Figura 5.16. Diagrama de Gantt por Tareas

V.1.4.4. Imprimir Especificaciones

Todo lo que se capturó durante la fase de requisitos puede obtenerse en formato impreso de modo que desarrolladores, clientes, usuarios y demás personas involucradas en el proyecto puedan tener una idea global de la tarea realizada.

Para ello, el prototipo de la herramienta provee una ventana que le permite seleccionar el tipo de reporte a obtener (ver Figura 5.17), el cual puede ser por Usuarios (ver Figura 5.18) o por Tareas (ver Figura 5.19). Obviamente se puede obtener reportes por referencias cruzadas, prioridades y estimaciones.

Figura 5.17. Selección de forma de Impresión

Figura 5.18. Impresión por Usuarios

Figura 5.19. Impresión por Tareas

V.1.4.5. Archivo de Ayuda

El prototipo consta de un archivo de ayuda[1] el cual brinda asistencia al usuario facilitando la instalación y el uso del prototipo de HGH. Este archivo se divide las siguientes páginas de ayuda:

- Menú Principal
- Administración de Usuarios
- Tareas
- Historias y Versiones
- Consultas del Estado del Proyecto
- Detalles de Instalación.

[1] Archivo de Ayuda fue generado usando DotHLP Professional

CAPÍTULO VI

EVALUACIÓN DE RESULTADOS

Obtener requisitos de alta calidad es necesario para lograr proyectos software exitosos. Por lo general, en la fase de Análisis de Requisitos, se obtienen los requisitos y se establece el fundamento para el diseño. En realidad, existen pocas métricas para el análisis y la especificación de requisitos, sin embargo se puede utilizar las métricas aplicables a los proyectos y adaptarlas a la especificación de requisitos, como por ejemplo los Puntos de Función [28, 33].

No obstante, usar métricas específicas de análisis y especificación de requisitos, es importante para entender cuáles son los requerimientos esenciales y, además, proporcionan una visión interna de la calidad del modelo de análisis.

VI.1. APLICACIÓN DEL PROTOTIPO Y EVALUACIÓN DE RESULTADOS

Es necesario detectar los errores de las Especificaciones de Requerimientos Software[1] (ERS) durante la fase de requisitos, sino el costo de repararlos crecerá exponencialmente a medida que se avanza en el proyecto.

Obtener especificaciones de calidad es importante, pues las consecuencias si se ignora la calidad son las siguientes:

- El software resultante puede no satisfacer las necesidades del cliente.
- Múltiples interpretaciones, pueden causar discrepancias entre clientes y desarrolladores.
- Puede resultar imposible probar el sistema completo.

Para corroborar que el prototipo de HGH y el uso de etnografía en captura de requisitos proporcionan especificaciones de calidad, entonces, se someterán los

[1] Ver definiciones en Marco Teórico.

resultados obtenidos en dos sistemas a las métricas de calidad de la especificación definidas por Davis [10, 11], quien propone una lista de características que se usan para evaluar la calidad del análisis y la especificación de requisitos.

Concretamente los atributos que se evalúan son: ausencia de ambigüedad, corrección, compleción, capacidad de verificación, facilidad de comprensión, capacidad de modificación, trazabilidad, independencia del diseño, concisión, capacidad de logro, consistencia interna y externa, ausencia de redundancia y capacidad de reutilización de las especificaciones. También se puede evaluar que estén almacenadas electrónicamente las especificaciones o no, que estén los requisitos anotados por importancia, estabilidad relativa y con su versión correspondiente, organizados y con referencias cruzadas.

Los sistemas[2] que se utilizarán para evaluar los resultados son: *"Sistema de Control de Deposito de Materiales Eléctricos"* (SCDME) y *"Sistema de Inscripción y Puntajes en Tiro a la Hélice"* (SIPTH).

VI.2. MÉTRICAS [10, 11, 32, 33, 39]

La ecuación de calidad de las ERS es una simplificación; lo que realmente es significativo son los valores de Q_i

$$Q = \frac{\sum W_i Q_i}{\sum W_i}$$ donde W_i es el peso, el cual esta predefinido por Davis [10].

A. ***Ausencia de ambigüedad (especificidad)*:** los requisitos no son ambiguos si, y sólo si, tienen una única interpretación. Una forma de medir la ambigüedad es con el porcentaje de requerimientos que se han interpretado de manera única. El peso asignado es W_1 =1.

Fórmula	SCDME	SIPTH	Evaluación
$Q_1 = \frac{N_{ui}}{N_r}$	1,00	0,88	0: Requerimientos Ambiguos 1: Ausencia de Ambigüedad

[2] Ver descripciones de los sistemas en el Marco Empírico

N_{ui}: número de requisitos por el cual todos los examinadores de los requisitos presentan interpretaciones idénticas.

N_r: número total de requisitos.

B. ***Corrección****:* los requerimientos son correctos si, y sólo si, representan alguna necesidad del sistema a construir. Una forma de medir si los requerimientos son correctos, es el porcentaje de de requerimientos que se han validado. El peso asignado es W_2 =1.

Fórmula	SCDME	SIPTH	Evaluación
$Q_2 = \frac{N_c}{N_r}$	0,96	0,83	0: Totalmente Incorrecto 1: Totalmente Correcto

N_c**:** número de requisitos correctos.

$N_r = N_c + N_{nv}$. Donde N_{nv} es el número de requisitos no validados.

C. ***Compleción***: se dice que las especificaciones están completas si, y sólo si, todo lo que se supone que el software hace está incluido en ella, además no hay requisitos marcados como no determinados.

Una forma de medir la compleción es teniendo en cuenta el modelo que se muestra en la Figura 6.1. La *Zona A* representa los requerimientos que se conocen y que se han capturado, la *Zona B* representa los requisitos capturados, pero están pobremente especificados y aún no se han validado, la *Zona C* representa requisitos que se conocen pero que no se han especificado; y la *Zona D* representa requisitos potenciales que no están entendidos y no se han especificado. Las flechas en la figura simbolizan el movimiento de requisitos de una zona a otra; como se puede apreciar, la tendencia es migrar hacia la *Zona A*.

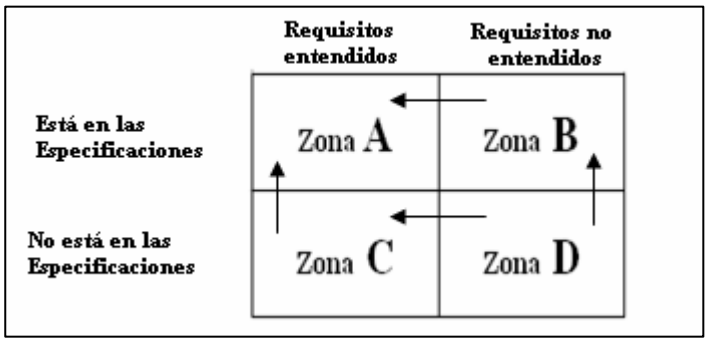

Figura 6.1. Modelo de Compleción de Requerimientos

Con este modelo se puede medir la compleción como el porcentaje de requisitos en las ERS que se han entendido y se han capturado. El peso asignado es W_3 =7.

Fórmula	SCDME	SIPTH	Evaluación
$Q_3 = \frac{N_A}{N_r}$	0,88	0,72	0: Incompleto 1: Completo

N_A**:** número de requisitos en la *Zona A.*

$N_r = N_A + N_B$. Donde N_B número de requisitos en la *Zona B.*

D. ***Capacidad de Verificación*****:** un requisito es verificable si, y sólo si, existe un proceso que le permita a una persona o máquina probar que el producto software satisface los requisitos. El hecho de que un requisito sea verificable, significa que esto reducirá el costo de la prueba. El peso asignado es W_4 =7.

Fórmula	SCDME	SIPTH	Evaluación
$Q_4 = \frac{N_r}{N_r + \sum C(R_i) + \sum T(R_i)}$	0,83	0,9	0: No Verificable 1: Verificable

$C(R_i)$**:** es el costo necesario para verificar la presencia del requerimiento *Ri*.

$T(R_i)$**:** es el tiempo necesario para verificar la presencia del requerimiento *Ri*.

E. ***Facilidad de comprensión***: los requisitos deben escribirse en lenguaje natural, de modo que sean entendibles tanto para clientes como desarrolladores. Incrementar la entendibilidad para desarrolladores, puede decrementar la entendibilidad de clientes. El peso asignado es W_5 =1.

Fórmula	SCDME	SIPTH	Evaluación
$Q_5 = \frac{N_{ur}}{N_r}$	1	1	0: No hay requerimientos entendidos 1: Todos los requerimientos entendidos

N_{ur}**:** número de requisitos que los examinadores (clientes, desarrolladores) pensaron que entendieron.

F. ***Capacidad de Modificación***: una especificación es modificable si, y sólo si, su estructura y estilo son tales que cualquier cambio puede ser hecho fácilmente, completamente y de forma consistente. Esto significa que el

usuario puede cambiar o agregar otros requisitos útiles. El peso asignado W_6 depende de la aplicación.

Fórmula	SCDME	SIPTH	Evaluación
Q_6	1	1	1: Si la tabla de contenidos e índice están contenidos en la ERS (fácil de modificar) 0: Otro

G. ***Trazabilidad***: una ERS es trazable si, y sólo si, está escrita de una manera que facilita la referencia a cada requisito individual. El peso asignado es W_7 depende de la aplicación. Las características que debe presentar para ser trazable son:

- Numerar cada requisito jerárquicamente.
- Se puede referir a cualquier requisito a través del número asignado.
- Usar una convención para indicar los requerimientos.

Fórmula	SCDME	SIPTH	Evaluación
Q_7	1	1	1: Si describe las características de arriba 0: No lo hace

H. ***Independencia del Diseño***: una especificación es independiente del diseño, si puede existir más de un diseño del sistema. El peso asignado es W_8 =5.

Fórmula	SCDME	SIPTH	Evaluación
$Q_8 = \frac{D(R_e \cup R_i)}{D(R_e)}$	1	1	0: Diseño Dependiente 1: Diseño Independiente

R_e**:** es el total de requerimientos que describen el comportamiento externo.

R_i**:** es el total de requerimientos que directamente dirigen la arquitectura o algoritmos de la solución.

$R = R_e \; U \; R_i$, R es el conjunto de todos los requerimientos.

$D(Re \; U \; Ri)$**:** es el número de soluciones de diseño que satisfacen todos los requerimientos.

$D(R_e)$**:** es el número de soluciones de diseño del sistema real, que satisface aquellos requerimientos del comportamiento externo.

I. ***Concisión***: si se tiene dos especificaciones, cada una de las cuales tienen niveles idénticos de calidad, la especificación más corta será la mejor. Si la

especificación es concisa, entonces los usuarios pueden leer fácilmente toda la documentación. Una de las medidas de concisión es contar las páginas, cuanto más páginas peor. El peso asignado es W_9 =2.

Fórmula	SCDME	SIPTH	Evaluación
$Q_9 = \frac{1}{Size+1}$	0,1	0,14	0: Peor SRS 1: Mejor SRS

*Size***:** es el número de páginas de la especificación.

J. ***Capacidad de Logro***: una ERS es alcanzable si, y sólo si, podría existir al menos un diseño e implementación del sistema, que implemente correctamente todos los requerimientos expresados en las ERS. La medida de capacidad de logro se la puede expresar como la existencia de un sistema. Peso asignado es W_{10} =1.

Fórmula	SCDME	SIPTH	Evaluación
Q_{10}	1	1	1: Existe un sistema 0: No existe un sistema

K. ***Consistencia Interna***: una ERS es internamente consistente si, y sólo si, no hay un conjunto de requerimientos que expresen conflicto. La medida de la consistencia interna es el porcentaje de funciones únicas que son determinísticas. El peso asignado es W_{11} =1.

Fórmula	SCDME	SIPTH	Evaluación
$Q_{11} = \frac{n_u - n_n}{n_u}$	0,92	0,88	0: 100% Inconsistente 1: 100% Consistente

n_u**:** es la cantidad de funciones reales únicas.

$n_u <= n_f$.Donde n_f son los requerimientos funcionales.

n_n**:** es la cantidad de funciones que no son determinísticas.

L. ***Consistencia Externa****:* una ERS es externamente consistente si, y sólo si, no hay requerimientos que expresen conflictos con líneas base ya proyectadas en la documentación. La mejor medida es el porcentaje de requerimientos que son consistentes con los documentos. El peso asignado es W_{12} =1.

Fórmula	SCDME	SIPTH	Evaluación
$Q_{12} = \frac{n_{EC}}{n_{EC} + n_{EI}} = \frac{n_{EC}}{n_r}$	1	0,94	0: 100% Inconsistente 1: 100% Consistente

n_{EC}**:** número de requisitos consistentes con todos los documentos.

n_{EI}**:** número de requisitos inconsistentes.

Donde $n_r = n_{EC} + n_{EI}$.

M. ***Ausencia de redundancia***: una ERS es redundante si el mismo requerimiento está expresado más de una vez. La medida es el porcentaje de funciones únicas que no están repetidas. El peso asignado es W_{13} =0.

Fórmula	SCDME	SIPTH	Evaluación
$Q_{13} = \frac{n_f}{n_u}$	1	1	0: Redundante 1: No redundante

n_f**:** cantidad de funciones especificadas.

N. ***Capacidad de reutilización***: una ERS es reusable si, y sólo si, sus sentencias, párrafos y secciones pueden ser fácilmente adoptados o adaptados a otras ERS subyacentes. Una alternativa es medir la reusabilidad de las ERS como potencial especificación a reusar. Existen técnicas para optimizar el potencial de reuso, ellas son:

- Escribir las ERS usando constantes simbólicas.
- Usar modelos formales.
- Crear bibliotecas de tipos de requerimientos abstractos.

Fórmula	SCDME	SIPTH	Evaluación
No definido	En realidad la reusabilidad es difícil de medir pues las especificaciones se construyen a partir de las historias definidas por los usuarios de cada sistema en particular.		No definido

O. ***Electrónicamente Almacenado***: si los requerimientos se almacenan electrónicamente en una base de datos, entonces la ERS puede generarse automáticamente. Se puede medir el porcentaje del volumen de la ERS que se ha almacenado electrónicamente. El peso asignado W_{14} depende de la aplicación.

Fórmula	SCDME	SIPTH	Evaluación
Q_{14}	Los requisitos se almacenan electrónicamente en una base de datos, lo cual permite generar automáticamente las especificaciones		1: Electrónicamente Almacenado 0: Otro

P. ***Anotado por importancia relativa***: permite a los lectores conocer cuáles requerimientos son los más importantes para el cliente. De esta manera se puede satisfacer las necesidades mínimas del cliente. La medida es el porcentaje de requerimientos que son registrados en las especificaciones. El peso asignado W_{15} depende de la aplicación.

Fórmula	SCDME	SIPTH	Evaluación
Q_{15}	Como se solicita al usuario de la herramienta el ingreso de la prioridad de las historias de usuario, entonces es posible ordenar las especificaciones por importancia.		1: Anotado 0: No Anotado

Q. ***Anotado por estabilidad relativa***: permite a los lectores conocer cuáles requerimientos cambian con mayor probabilidad. La medida es el porcentaje de requerimientos que son anotados. El peso asignado W_{16} depende de la aplicación.

Fórmula	SCDME	SIPTH	Evaluación
Q_{16}	Como se solicita al usuario de la herramienta el ingreso de modificaciones y versiones de las historias de usuario, entonces es posible conocer cuales son los requerimientos que cambian con mayor frecuencia.		1: Anotado 0: No Anotado

R. ***Anotado por versión***: le dice al lector que se han cumplido los requisitos en diferentes fases (versiones) del desarrollo. La medida es el porcentaje de requerimientos que son anotados. El peso asignado W_{17} depende de la aplicación.

Fórmula	SCDME	SIPTH	Evaluación
Q_{17}	La herramienta permite visualizar las historias terminadas o no, fechas de inicio y fin, y las historias que el desarrollador terminó pero que tiene que volver a desarrollarlas pues fueron modificadas.		1: Anotado 0: No Anotado

S. ***Organizado***: si la especificación es organizada, entonces el usuario puede localizar la información fácilmente y entender las relaciones lógicas entre las diferentes secciones. Alguna alternativas para considerar a la ERS organizada son:

- Agrupar los requerimientos funcionales por clases de usuarios.

- Agrupar los requerimientos funcionales por estímulos comunes.
- Agrupar los requerimientos funcionales por respuestas comunes.
- Agrupar los requerimientos funcionales por características.
- Agrupar los requerimientos funcionales por objetos.

Fórmula	SCDME	SIPTH	Evaluación
No definido	El prototipo de la herramienta permite organizar la información por tareas, por usuarios y por historias.		No definido

T. ***Referencias Cruzadas***: una ERS es de referencias cruzadas si, y sólo si, las referencias cruzadas se usan en la ERS para relacionar secciones que contienen requerimientos a otras secciones que incluyen: requerimientos idénticos, más descripciones detalladas de los mismos requerimientos ó requerimientos que dependen de ellos o de cuales ellos dependen.

Fórmula	SCDME	SIPTH	Evaluación
No definido	La herramienta permite trazar referencias entre las historias.		No definido

El resultado de la ecuación de calidad de los sistemas evaluados es:

$$Q_{SCDME} = 0{,}88 \qquad \text{y} \qquad Q_{SIPTR} = 0{,}85$$

VI.3. EVALUACIÓN

Como se expresó anteriormente, la ERS constituye un documento que describe el comportamiento externo observable y las características esperadas del sistema. Generalmente, una ERS de calidad contribuye exitosamente a la creación de un sistema software que resuelve las necesidades reales de los usuarios.

Al evaluar los resultados de la ecuación general de calidad de las ERS, obtenidas a partir del uso de los lineamientos de aplicación de etnografía en la captura de requisitos y el prototipo de HGH, se puede observar que en ambos sistemas el valor de *Q* es próximo a 1, lo cual indica que se trata de especificaciones de calidad.

Sin embargo, este valor no dice mucho a cerca de las características de calidad de las especificaciones presentadas al inicio del capítulo, por lo cual se considera que los valores de Q_i son más significativos a la hora de evaluar las mismas. Además, se puede observar en la lista de características de calidad que alguna de ellas tiene métricas que permiten evaluar si son de calidad o no, mientras que hay otras que son condiciones que si se cumplen permiten afirmar que las especificaciones son de calidad. A continuación se explican algunos resultados de las características de calidad mencionadas anteriormente.

Por ejemplo, en cuanto a la *no ambigüedad* de las especificaciones, en el SCDME todos los requerimientos se interpretaron de una única manera, mientras que en el sistema SIPTH, el 88% de los requerimientos se interpretaron de una sola forma; no obstante, como este valor es próximo al 100% del total de requerimientos, se considera que en ambos sistemas las especificaciones no son ambiguas.

En cuanto a la *compleción* de las especificaciones, en el sistema SCDME el 88% de los requisitos han sido entendidos y capturados, mientras que en el sistema SIPTH el 72% cumplen esta condición; por lo tanto, se puede decir que en ambos sistemas las especificaciones cumplen el requisitos de calidad de *compleción*, pues el número de requisitos entendidos y capturados es próximo al 100% de los requisitos totales del sistema.

En lo que se refiere a si las especificaciones están *organizadas* o no, se puede apreciar que el diseño de HGH y de la base de datos, permite que la información se organice por usuarios, por tareas, por historias, por versiones, por fechas, etc., aún cuando en el prototipo sólo se prevé la organización por usuarios y por tareas; por lo tanto, se pode decir que las especificaciones de ambos sistemas son organizadas.

Por último, se puede decir que las especificaciones están *anotadas por importancia relativa*, ya que a cada historia el usuario le puede asignar una prioridad lo cual permite poder ordenar las ERS; es decir, al analista puede saber fácilmente cuales son las más importantes para el cliente.

CONCLUSIÓN

En el proceso de desarrollo de software de los MT cada etapa depende de la anterior, lo cual implica que el costo de realizar cambios cuando el producto está próximo a entregarse es mucho mayor que en etapas tempranas. Por lo tanto, es importante determinar con exactitud los requisitos del sistema, para evitar realizar cambios cuando el proceso ya está en marcha. Sin embargo, esto no siempre es posible puesto que en algunas ocasiones se producen cambios de las necesidades del cliente durante el desarrollo, o bien, cambios ajenos al proyecto pero que lo afectan directamente (por ejemplo, cambios en las tecnologías, limitaciones financieras, etc.).

Por otra parte, los MA plantean que se puede evitar el costo del cambio retrasando la toma de decisiones. El más popular de estos métodos es XP, el cual propone la comunicación directa con el cliente, las fichas para asentar las historias de usuario, la refactorización, y considera al código fuente como la documentación. No obstante, su aplicación tiene serios desaciertos, como la falta de buenas especificaciones y de gestión de requisitos.

A pesar de existir varias herramientas de gestión de requisitos, hay pocas aplicables a un entorno de desarrollo XP, por lo tanto, se propuso el diseño de una herramienta de gestión de historias de usuario llamada HGH, la cual consta de un sistema interno de control de versiones, apta para que los usuarios la utilicen en forma concurrente para actualizar historias. Se formuló una serie de lineamientos o recomendaciones de aplicación de etnografía en la fase de captura, para ayudar a los desarrolladores a realizar una exhaustiva recolección de requisitos e incorporar toda la información obtenida a HGH. Así, con la aplicación de etnografía se podrá obtener la información necesaria para definir en forma correcta las tareas y actualizar las versiones.

Para probar que la etnografía y la utilización HGH colaboran con usuarios y desarrolladores a lograr una adecuada captura, especificación y control de requisitos,

manteniendo los principios de los MA, se construyó un prototipo que se utilizó durante el desarrollo de dos sistemas.

Los resultados de la utilización se sometieron a una lista de características para valorar la calidad de las especificaciones obtenidas. Como se expresó al final del último capítulo, en ambos sistemas se obtuvieron especificaciones de calidad.

Por ejemplo, en lo que se refiere a la especificidad, se puede decir que, gracias a la etnografía, se definieron correctamente las tareas, las historias y las versiones, permitiendo así que el número de requerimientos no ambiguos se aproxime al número total de requerimientos del sistema. Además, como se trata de una herramienta software, las especificaciones están almacenadas electrónicamente, organizadas, no hay redundancia, se pueden establecer referencias cruzadas y se puede acceder a ellas con facilidad.

Se puede concluir que en los sistemas sometidos a prueba, tanto los lineamientos de la etnografía, que resultaron fáciles de aplicar, como el uso del prototipo de la herramienta desarrollado, permitieron obtener especificaciones de calidad, ayudando a los desarrolladores (analistas, diseñadores, programadores, etc.) y al cliente a incorporar la información del sistema, a saber en todo momento el estado del proyecto, a controlar las actualizaciones de requisitos, facilitando las pruebas y el mantenimiento del sistema. Cabe destacar que se trata de un prototipo, lo cual implica que no presenta todas las funcionalidades de la herramienta diseñada, sin embargo fue útil para probar las características mencionadas en el párrafo anterior.

Como trabajo futuro se propone terminar la construcción de la herramienta HGH, y se espera que facilite la visualización del estado de las tareas, las historias y las versiones, y permita obtener las especificaciones por prioridades, prioridades técnicas, duraciones, referencias cruzadas, etc. (además de las ya provistas por el prototipo).

BIBLIOGRAFÍA

[1] **Abian,** Miguel Ángel. "Promesas incumplidas: sobre los métodos ágiles" 2003.

[2] **Álvarez Sánchez,** José Ramón; **Calleja,** Manuel Arias. "Análisis, Diseño y Mantenimiento del Software". Dpto. de Inteligencia Artificial - ETSI Informática. UNED. Diciembre 2002.

[3] **Barbero,** Juan Manuel. "Trabajando con programación extrema – Experiencias reales"

[4] **Beck,** Kent. "Extreme Programming Explained: Embrace Change". 1999.

[5] **Blanqué,** Javier. "Nuestra Metodología de Desarrollo de Software". http://www.dinamitec.com

[6] **Burke,** Jason; **Kira,** Andrea. "Ethnographic Methods". University of Maryland. Octubre 2001.

[7] **Calero Solís,** Manuel. "Una explicación de la programación extrema".

[8] **Campos,** Mauricio. "Extreme Programming". Universidad Tecnológica Nacional. Facultad Regional Buenos Aires. Julio 2004

[9] **Carvallo,** Judit; **Rivas,** Lornel; **Villegas,** Maria del Carmen. "Programación Extrema". Junio de 2002.

[10] **Davis,** Alan. "Identifying and Measuring Quality in a Software Requirements Specification". Mayo 1993

[11] **Davis,** Alan. "Software Requirements. Objects, functions & stages". 1993.

[12] **Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software** Facultad de Informática - Universidad Politécnica de Madrid. "Definición de Perfiles en Herramientas de Gestión de Requisitos". Septiembre de 2005.

[13] **Durán Toro,** Amador**; Bernárdez Jiménez,** Beatriz**.** "Metodología para la Elicitación de Requisitos de Sistemas Software". Abril de 2002.

[14] **Floría Cortés,** Alejandro. "Métodos de Indagación". Febrero 2000.

[15] **Fowler,** Martin. "¿Ha muerto el diseño?" Mayo 2004.

[16] **Fowler,** Martin. "La nueva Metodología".

[17] **Granollers,** Toni. “MPIU+A. Una metodología que integra la ingeniería del software, la interacción persona-ordenador y la accesibilidad en el contexto de equipos de desarrollo multidisciplinares”. Julio 2004.

[18] **Garrido,** J. L.; **Gea,** M.; **Noguera,** M.; **González,** M.; **Ibáñez,** J. A. “Una Propuesta Arquitectónica para el Desarrollo de Aplicaciones Colaborativas”. Departamento de Lenguajes y Sistemas Informáticos - Universidad de Granada.

[19] **Guersenzvaig,** Ariel. “El usuario arquetípico: Creación y uso de personajes en el diseño de productos interactivos”. Agosto de 2005.

[20] **Hom,** James. “Estudio Etnográfico / Observación de Campo”. 1996

[21] **IEEE Std 1233**, **1998** (incluye IEEE Std 1233–1996 e IEEE Std 1233a-1998). “Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas”.

[22] **IEEE Std 610.12**, **1990**. “Glosario de Terminologías de Ingeniería de Software”.

[23] **IEEE Std 830, 1998. “**Especificaciones de los requisitos del software”.

[24] **ISO 9241-11, 1998.** “Ergonomic requirements for office work with visual display terminals (VDTs) – Guidance on usability ”.

[25] **Jacobson,** Ivar; **Booch,** Grady; **Raumbaugh,** James. “El lenguaje Unificado de modelado”. 1999.

[26] **Jacobson,** Ivar; **Booch,** Grady; **Raumbaugh,** James. “El proceso Unificado de desarrollo de software”. 2000.

[27] **Kähkönen,** Tuomo; **Abrahamsson,** Pekka. “Digging into the Fundamentals of extreme Programming - Building the Theoretical Base for Agile Methods”. Publicado en la IEEE Computer Society 2003.

[28] **Kandula,** Ganesh; **Sathrasala,** Vinay Kumar. “Product and Management Metrics for Requirements”. Marzo 2005

[29] **Letelier,** Patrício; **Penadés,** Maria Carmen. “Metodologías Ágiles para el desarrollo de software: eXtreme Programming (XP)”.

[30] **Paulk,** M. “Extreme Programming from CMM Perspective”. Diciembre 2001.

[31] **Pérez Sánchez,** Jesús. “Metodologías Ágiles: La ventaja competitiva de estar preparado para tomar decisiones lo más tarde posible y cambiarlas en cualquier momento”. http://www.agile-spain.com

[32] **Pflleger,** Shari Lawrence. “Ingeniería de Software. Teoría y Practica”. 2002.

[33] **Pressman,** Roger. “INGENIERÍA DEL SOFTWARE. Un enfoque práctico”. Quinta Edición. 2002.

[34] **Reifer,** Donald J. “How Good Are Agile Methods?”. IEEE SOFTWARE Julio / Agosto 2002.

[35] **Rouncefield,** Mark. “G53DBC Ethnography and workplace studies”. Lancaster University.

[36] **Sánchez,** Emilio A; **Letelier,** Patricio; **Canos,** José H. “Mejorando la gestión de historias de usuario en eXtreme Programming”.

[37] **Sánchez González,** Carlos. ”ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras”. Septiembre de 2004.

[38] **Santana,** Lourdes Denis; **Gutiérrez Borobia,** Lidia. “La investigación etnográfica: experiencias de su aplicación en el ámbito educativo”.

[39] **Sommerville,** Ian. “Ingeniería del Software”. 7ma Edición 2005

[40] **SQS, SA.** “Herramienta que implementa extreme Programming para la gestión de requisitos”.

[41] **Stark,** John A.; **Crocker,** Ron. “Trends in Software Process: The PSP and Agile Methods”. Publicado en la IEEE Computer Society 2003.

[42] **Titulo de la página**: “Agile Requirements” Disponible: http://www.agilexp.com. Acceso: Marzo 2006.

[43] **Titulo de la página**: “CVS”. Disponible en: http://es.wikipedia.org/wiki/CVS. Acceso: Febrero 2006.

[44] **Titulo de la página**: “Extreme Programming”. Disponible en: http://www.planetacodigo.com/wiki/glosario:extreme_programming?dobacklink. Acceso: Febrero 2006.

[45] **Titulo de la pagina**: “Fundamentos de la Ingeniería de Software”. Disponible en: http://www.miguelherrero.tk. Acceso: Febrero 2005.

[46] **Titulo de la página**: “Gestión de requisitos: CaliberRM™”. Disponible en: http://www.danysoft.com/asp/download.asp?sFichero=caliberv.zip&sDir=free. Acceso: Febrero 2006.

[47] **Titulo de la página**: “Herramienta para Gestión de Requisitos”. Disponible en: http://www.irqaonline.com. Acceso: Febrero 2006.

[48] **Titulo de la página**: “Refactorización en la práctica: peligros y soluciones”. Disponible en: http://www.agile-spain.com. Acceso: Enero de 2006.

[49] **Titulo de la página**: “Requirements Management with Reconcile”. Disponible en: http://www.compuware.com/products/reconcile.htm. Acceso: Enero de 2006.

[50] **Titulo de la página**: “Sistema de control de versión”. Disponible en: http://es.wikipedia.org/wiki/Sistema_de_control_de_versi%C3%B3n. Acceso: Febrero 2006.

[51] **Titulo de página**: “Subversión”. Disponible en: http://es.wikipedia.org/wiki/Subversion. Acceso: Febrero 2006.

[52] **Titulo de la página**: “XPlanner”. Disponible: http://www.xplanner.org. Acceso: Febrero 2006.

[53] **Tsvetkova,** Nikolina; **Karastateva,** Violeta. “Ethnography? (What) Does it Have to Do with Language Education?”.