UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

PROYECTO DE GRADO

"SISTEMA DE SEGUIMIENTO Y CONTROL DE VISITADORES MÉDICOS"
CASO: FARMEDICAL SRL

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

Postulante: Wankar Daniel Calliconde Gutierrez
Tutora: MSc. Fátima Consuelo Dolz de Moreno
Revisor: Lic. Aldo Ramiro Valdez Alvarado

La Paz – Bolivia
2011

DEDICATORIA

A mis padres Moises Calliconde y Rosse Mary Gutierrez por haberme dado la oportunidad de existir, brindándome siempre su incondicional apoyo.

A mi hermano Abdul Alí, quien siempre confió en mí.

AGRADECIMIENTOS

A mis familias en general por siempre recordarme de dónde vengo y hasta donde puedo llegar.

A mi Docente Tutor, M.Sc. Fátima Dolz, por la valiosa guía que me brindo a lo largo del desarrollo de este trabajo.

A mi Docente Revisor, Lic. Aldo Valdez, por haberme dado parte del gran conocimiento que posee para la buena elaboración del presente proyecto.

A la Universidad Mayor de San Andrés por darnos la posibilidad de poder ser mejores personas en la vida.

Al Sr. Carlos Fernández Aguilar, Gerente General de FARMDEDICAL SRL, por haber puesto su confianza en mí.

A todos mis amigos y compañeros, no solamente de la universidad, con quienes compartí años inolvidables a lo largo de mi formación académica.

RESUMEN

Día a día la información de toda empresa tiene un constante crecimiento, llegando a un punto en el cual es insostenible realizar los procesos con los que cuenta de forma manual. Además que ninguna empresa tiene la opción de brindar ventajas, en este caso ventajas tecnológicas, a sus directos competidores. Por estas razones y haciendo uso de las nuevas tecnologías de información es que se desarrolla el presente proyecto denominado **"Sistema de Seguimiento y Control de Visitadores Médicos" Caso: FARMEDICAL SRL.**

FARMEDICAL SRL, una empresa farmacológica reconocida a nivel nacional, tiene como misión proveer salud a la población Boliviana a través de la comercialización de productos farmacéuticos a nivel nacional, con altos estándares de calidad, seguridad y accesibilidad de costo tratamiento.

El presente proyecto tiene como objetivo resolver los problemas que aquejan a FARMEDICAL SRL en el área de Marketing, más específicamente en el de Visitas Médicas. Para dicho fin hará uso de los conocimientos adquiridos en el estudio dentro de la carrera de Informática de la Universidad Mayor de San Andrés.

Al momento en el área ya mencionada los procesos se los realizan de forma manual, teniendo así problemas en la emisión de reportes en cuanto a visitas médicas y el alcance de las mismas. A partir del desarrollo e implementación del sistema se podrá acortar el tiempo de asignación de visitas médicas por medio de interfaces amigables y fáciles de utilizar, también se obtendrán reportes mucho más exactos del alcance que la visitas médicas tienen.

El sistema está desarrollado en lenguaje Java con la ayuda del IDE Netbeans 6.9.1. Para la gestión de base de datos se utilizó el servidor Postgres 9.0.3 con la ayuda de su administrador pgAdmin III. Para el diseño de la base de datos utilizamos DBDesignerFork. Y para la emisión de reportes utilizamos iReports 4.1.2 con las librerías de JasperReports 4.1.2.

Después del uso del sistema se logró una considerable reducción tanto en el tiempo de asignación de visitas médicos que se utilizaba, se redujo el volumen de información física y se obtuvieron mejores reportes a nivel gerencial.

INDICE

CAPITULO II "MARCO TEÓRICO"

CAPITULO V "ANÁLISIS DE COSTO BENEFICIO"

CAPITULO VI "CONCLUSIONES Y RECOMENDACIONES"

INDICE DE FIGURAS

INDICE DE TABLAS

CAPITULO I

MARCO INTRODUCTORIO

1.1. INTRODUCCIÓN.

La información que hoy en día se produce en magnitudes considerables obliga tanto a empresas públicas como privadas a organizar su información, pero estas, no cumplen los objetivos propuestos por las mismas para satisfacer la demanda exigida por los usuarios tanto internos como externos de una entidad en específico, es por eso que se ven obligadas a adoptar nuevos mecanismos que administren la información de una manera más eficiente y que les ayude a obtener la información requerida para su mejor administración de una forma más sencilla y rápida.

La base sobre la cual se mueven las empresas o también llamadas laboratorios farmacológicos es la promoción de los productos con los que cuentan para su posterior comercialización, pero a este proceso de promoción de sus productos se las denominan Visitas Médicas. El objetivo de esta tarea es hacer conocer a los profesionales médicos acerca del stock con el que se cuenta en dicha empresa. De ahí que los doctores harán uso de los mismos por medio de las recetas médicas que prescriben. Por consiguiente la empresa pondrá a disposición de los pacientes los productos requeridos a través de farmacias autorizadas que tengan el convenio con la empresa.

Las características que este tipo de sistemas deben cumplir para que los usuarios tengan confianza en ellos son: la confiabilidad de la misma, una interfaz amigable fácil de utilizar, los resultados finales deben mostrar realmente el trabajo que en este caso los Visitadores Médicos están realizando con los médicos asignados a su especialidad. En sí toda esta información es muy necesaria para la buena toma de decisiones a nivel gerencial ya que esta actividad es el punto de partida para toda empresa farmacológica a nivel mundial.

1.2. ANTECEDENTES.

A lo largo que este tiempo que se dio un desarrollo a pasos agigantados dentro de las tecnologías de la información, se han desarrollado numerosos sistemas destinados a distintas áreas y objetivos. Es así que viendo la empresa FARMEDICAL SRL y su desarrollo podemos observar lo que sigue.

1.2.1.Antecedentes Institucionales.

FARMEDICAL SRL está constituida por accionistas con mucha trayectoria en la comercialización dentro del campo Farmacéutico y de Cosméticos, así como en las áreas más fundamentales para su desarrollo.

FARMEDICAL SRL como empresa privada se compromete con la salud pública de nuestro país, y mediante la administración eficiente de sus recursos, planeación, y mejora continua logra una mayor efectividad en el cumplimiento de sus objetivos, contribuyendo al acceso de medicamentos por parte de nuestra sociedad, en forma equitativa y a precios razonables, logrando contactos con laboratorios proveedores extranjeros cuya calidad está certificada por las Buenas Prácticas de Manufactura en cada país.

Participa en Licitaciones Públicas y Compras Directas de todas las Instituciones de Salud en Bolivia: Caja Nacional de Salud, Cossmil, Banca Privada, Caja Petrolera de Salud, Caja de Salud de la Banca Privada, Seguro Social Universitario, Caja Cordes, etc., etc., con la finalidad de dar a conocer la calidad de nuestros productos a nivel nacional.

1.2.1.1. Historia.

FARMEDICAL SRL, fue constituida el 6 de Noviembre de 2002. Con el fin de afianzar el Prestigio – Confiabilidad - Competitividad y presencia en el mercado, se establecen alianzas comerciales únicamente con Laboratorios Farmacéuticos garantizados y certificados por sus autoridades reguladoras.

Durante las gestiones 2003 y 2004, se realizan todos los trámites de constitución, estudio de mercado, alianzas estratégicas de fabricación y representación, registros de marca y registros sanitarios.

El año 2005, FARMEDICAL inicia sus actividades oficialmente, introduciendo en el mercado productos Biotecnológicos, Hemoderivados, Anestésicos, Antibióticos, Antiinflamatorios y moléculas nuevas en forma paulatina e impulsora.

El personal motivado por su propia satisfacción contribuye al crecimiento día a día, atendiendo sus diversas tareas, sustentándose en la implementación de un sistema de garantía de calidad, conforme a la normativa vigente, que promueve el desarrollo y participación activa en la mejora continua de los procesos propios de la empresa en el tiempo, logrando un mejor servicio al cliente y mayor accesibilidad a medicamentos por parte de la población en general.

Hoy en día la empresa cuenta con una central ubicada en la ciudad de La Paz y cinco regionales distribuidas en las ciudades de La Paz, Cochabamba, Santa Cruz, Sucre y Beni. Así también la creación de la nueva FARMEDICAL PERU, ubicada en la ciudad de Lima de ese país.

1.2.1.2. Misión.

"Proveer salud a la población Boliviana a través de la comercialización de productos farmacéuticos a nivel nacional, con altos estándares de calidad, seguridad y accesibilidad de costo tratamiento".(Farmedical, 2011)

1.2.1.3. Visión.

"Constituir una empresa del rubro farmacéutico líder a nivel internacional con participación en toda la cadena productiva desde los laboratorios de producción hasta el consumidor final; generando imagen y desarrollo de la industria farmacéutica boliviana con proyección y liderazgo mundial". (Farmedical, 2011)

1.2.2.Antecedentes del Proyecto.

La empresa FARMEDICAL SRL al momento no cuenta con un sistema automatizado que le ayude a hacer un seguimiento y control de todas las visitas médicas que se realizan día tras día, es por tal motivo que se pretende desarrollar un sistema que brinde toda la información necesaria en un momento dado para así poder conocer la situación en la que se encuentra el área de Visitas Médicas dentro de la empresa.

Ya que la industria farmacológica tiene ya mucho tiempo trabajando en el mundo, se desarrollaron diferentes sistemas que ayudan a dichas empresas a hacer una mejor administración de las mismas, entre estas podemos mencionar:

- **PMS v. 1.0**, desarrollado por Programmings SF el año 2007 y presentado en el concurso mundial de programación Top Coder. Ayuda a los visitadores médicos a planificar sus visitas médicas de una forma sencilla por medio de una lista de médicos administrada por ellos mismos.
- **CM Visit**, desarrollado por Integra América el año 2008, es un sistema de apoyo a las visitas médicas por medio de dispositivos móviles. Asiste tanto a Visitadores como a vendedores en su diario operar.
- **Mobile Lab**, desarrollado por InnovaAge el año 2009, apoya a Visitadores médicos a verificar las visitas médicas que realizan también por medio de dispositivos móviles.

1.3. PROBLEMA.

El sistema que se quiere implementar resolverá los siguientes problemas detectados:

1.3.1. Problema Central.

De acuerdo a lo observado dentro de la empresa se encontró es el siguienteproblema central:

- **¿De qué manera podemos obtener una mejor y más exacta información de cómo los visitadores médicos realizan su trabajo dentro de la empresa FARMEDICAL SRL?**

1.3.2. Problemas secundarios.

Pero al mismo tiempo también se pudieron encontrar los siguientes problemas secundarios y sus respectivas causas y efectos:

- El cronograma de visitas médicas se las realiza de forma manual, lo que lleva mucho tiempo para los visitadores médicos.
- El control de las visitas médicas se las realiza mediante el conteo fichas impresas, esto ocasiona errores de omisión, repetición y otros.

- La asignación de muestras médicas se las realiza en base a experiencias pasadas únicamente, lo que incide en el mal uso de las mismas.
- Los reportes que emite en este caso el supervisor están basados en la documentación física con la que cuenta en ese momento, lo que causa perdida de información y un reporte no muy fiable.
- La asignación de especialidades a un Visitador médico se la realiza en base a fichas, lo que ocasiona que un visitador pueda extraviar alguna de ellas y así bajar su rendimiento en cuanto a las visitas médicas.

1.4. OBJETIVOS.

Definir el objetivo central y los objetivos secundarios de un proyecto es la razón de existir del proyecto. Para resolver el problema anteriormente planteado se tiene lo siguiente:

1.4.1. Objetivo General.

El objetivo general es:

- **Desarrollar e implementar un sistema de información que nos brinde la suficiente información acerca del trabajo realizado por los visitadores médicos para la buena toma de decisiones a nivel gerencial en la empresa FARMEDICAL SRL.**

1.4.2. Objetivos Específicos.

Los objetivos específicos serán:

- Reducir el tiempo de la programación de visitas médicas.
- Saber el número exacto de las visitas médicas realizadas.
- Optimizar el uso de las muestras médicas.
- Generar reportes actualizados de acuerdo a estadísticas reales y de acuerdo a las necesidades de la empresa.
- Filtrar los médicos correspondientes a un Visitador médico acorde a la especialidad que este posee.
- Obtener estadísticas actualizadas.
- Diseñar una base de datos segura y confiable.

- ❖ Capacitar a los empleados para el correcto uso del sistema.

1.5. JUSTIFICACIÓN.

El desarrollo del proyecto permite adquirir información detallada y actualizada para el control y seguimiento de las visitas médicas realizadas por cada visitador médico que realizan cotidianamente, de esta manera se obtendrán datos confiables y precisos.

1.5.1. Justificación Económica.

El presente proyecto en desarrollo, dentro de la empresa tiene una importancia alta, ya que así se contará con un control más estricto de las visitas que realizan los visitadores médicos que en si es el punto de partida para la misma. Este influirá directamente a la toma de decisiones por parte del nivel gerencial, del nivel administrativo y por supuesto a un mejor uso de los recursos humanos, económicos y materiales de la empresa. De esta manera se espera que la empresa FARMEDICAL SRL pueda reducir los gastos asignados hacia esta área al mínimo, y así también a la larga generar un ahorro de dinero que se lo puede invertir ya sea en la misma área o en otras actividades que ayudaran al crecimiento de la misma.

1.5.2. Justificación Social.

Dentro de la empresa los Gerentes Regionales, Supervisores y Visitadores Médicos contarán con un sistema que les ayude directamente a la administración de su área de una forma más rápida y confiable. Fuera de la empresa, todos los médicos a los cuales la empresa tiene alcance, contarán con una mejor información y constante actualización acerca de los nuevos productos farmacéuticos que se están manejando a nivel mundial, los cuales la empresa promociona. De tal forma que los mismos al tener mayor conocimiento de estos productos puedan brindar un mejor servicio a todos los pacientes con los que interactúan y también al pueblo en general.

1.5.3. Justificación Tecnológica.

Se justifica técnicamente porque en el presente trabajo se aplican diferentes técnicas para el control y seguimiento de la empresa, además que se emplearan técnicas de modelaje y diseño de sistemas. Para ello es necesario la investigación y la evaluación de las nuevas

tecnologías dentro de esta área, las cuales nos pueden ser muy útiles y podemos incluir al desarrollo y mejorar la complejidad de nuestro sistema.

En este caso en particular haremos uso de un sistema de información basado en los datos que la empresa nos proporcione. Utilizaremos un lenguaje de programación y una base de datos que nos proporcionen todo lo necesario para poder desarrollar e implementar lo deseado, así como técnicas y lineamientos para la programación e implementación de la base de datos. Todas estas estarán implementadas en un servidor que proporcionara la información necesaria a todos los usuarios tanto regionales como departamentales.

1.6. LIMITES, ALCANCES Y APORTES.

El sistema a realizarse dentro de la empresa FARMEDICAL SRL ayudara directamente a la parte gerencial de la misma. De ahí que podemos deducir los siguientes límites y alcances.

1.6.1. Límites.

Se tendrán los siguientes límites:

- El sistema se preocupara exclusivamente de dar reportes acerca de las visitas médicas realizadas más no de las muestras médicas, literaturas y obsequios que un visitador médico pueda regalar a su médico.
- El sistema emitirá reportes estadísticos de las visitas médicas realizadas solo a partir de la fecha en el que este sea implementado y no tomará en cuenta datos anteriores al mismo si es que estos existieran.
- El sistema podrá tener acceso a la base de datos correspondientes a los empleados, mas no modificarlos ya que esta parte es de exclusiva competencia del área de Recursos Humanos.
- El sistema brindará datos generales acerca de los médicos relacionados con la empresa, más no así los datos personales de los mismos, aunque se cuente con ellos y solo se podrá hacer uso de los mismos en caso de alguna emergencia.

1.6.2. Alcances.

Se tendrán los siguientes alcances:

- El sistema ayudara tanto los gerentes regionales, supervisores y visitadores médicos a realizar un mejor y más exacto seguimiento de su propio trabajo en base a reportes estadísticos precisos.
- El sistema ayudara a los supervisores a administrar las visitas médicas de una forma conveniente a la empresa ya que el tendrá acceso completo al mismo.
- El sistema hará automáticamente el filtrado de médicos que corresponden a un visitador médico de acuerdo a la especialidad que este posee.
- El sistema ayudara a todos los visitadores médicos a realizar su planificación de visitas mensuales de una forma más rápida y exacta.
- El sistema emitirá reportes gerenciales para la buena toma de decisiones a nivel gerencial.
- El sistema podrá emitir reportes de seguimiento de actividades tanto diarias, semanales y hasta mensuales.

1.6.3. Aportes.

FARMEDICAL SRL será una de las primeras empresas de este tipo en implementar un sistema de información automatizado para su correspondiente área de visitas médicas convirtiéndola así en una de las mejores a nivel nacional y porque no poder decirlo internacionalmente, ya que esta también tiene su filial FARMEDICAL PERU[1].

Dentro de la Universidad Mayor de San Andrés podemos observar que no se desarrollaron proyectos similares a éste, convirtiendo el mismo en un aporte valioso para futuros proyectos que se relacionen con esta temática.

1.7. METODOLOGÍA.

Para el desarrollo del presente proyecto se aplicaran diferentes metodologías y herramientas que contribuyan al mejor entendimiento y desarrollo del mismo. La metodología utilizada para este trabajo está sujeta a la modelización y abstracción de datos, adquiriendo información necesaria, segura y confiable que el cliente requiera y para el funcionamiento de la empresa.

---

[1] FARMEDICAL PERU: Filial extendida de FARMEDICAL BOLIVIA

1.7.1. Método de investigación.

El método científico es un proceso destinado a explicar fenómenos, establecer relaciones entre los hechos y enunciar leyes que expliquen los fenómenos físicos del mundo y permitan obtener, con estos conocimientos, aplicaciones útiles al hombre.

Los científicos emplean el método científico como una forma planificada de trabajar. Sus logros son acumulativos y han llevado a la Humanidad al momento cultural actual.

1.7.2. Tipo de Investigación.

1.7.2.1. Investigación exploratoria.

Sirve para determinar y precisar campos de información poco conocidos al iniciar una investigación; puede tener utilidad inmediata para recoger información para proyectos.

Averiguar sobre grupos humanos:

- ¿De dónde viene? (Historia)
- ¿Quiénes son y cómo son? (Observación Participante o Relevante)
- ¿Cuántos son? (Investigaciones cuantitativas, estadísticas, bibliográficas)
- ¿Cuántos hay? (Id.)
- ¿Dónde obtengo la información? (investigación bibliográfica y de fuentes)

Un ejemplo de investigación exploratoria es obtener los antecedentes de una comuna para realizar un diagnóstico socioeconómico de la comuna. En nuestro caso en particular realizamos una investigación exploratoria directamente con la empresa en la que se desarrollará el proyecto de grado.

1.7.2.2. Investigación descriptiva.

Los estudios descriptivos se usan cuando se ha escogido un concepto (variable) o ítem y deben explorarse sus cualidades o variables internas.

La investigación descriptiva nos permite averiguar:

- ¿Cuánto (o cuánto) son (es) exactamente?

- ¿Cómo se subdividen?
- ¿Cuántos son afectados? (por las variables)
- ¿Cuánto consumen, gastan, usan, ocupan?
- ¿Cuánto necesitan? (y otros)

A partir de estas preguntas obtenemos la información necesaria para empezar a diseñar la arquitectura del proceso al cual queremos dar solución con nuestro sistema.

1.7.3. Metodología de desarrollo.

La metodología de desarrollo que se utilizará será la de Programación Extrema (XP[2]).

1.7.3.1. Programación Extrema (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea. Esta metodología de desarrollo será más ampliamente descrita en el Capítulo II.

1.7.4. Plataforma y herramientas de desarrollo.

Algunas de las herramientas de desarrollo que se utilizarán en el desarrollo del proyecto son:

1.7.4.1. Plataforma.

La plataforma tanto del sistema operativo y del gestor de base de datos son las siguientes:

---

[2] XP: Extreme Programming

- Sistema Operativo. Windows XP (Service Pack 3).
- Gestor de Base de Datos. PostgreSQL 9.0.3

1.7.4.1. Herramientas.

Las herramientas que hoy en día nos brinda la tecnología son esenciales para un mejor diseño, desarrollo e implementación de un proyecto de software. Es por tal razón que utilizaremos las siguientes.

- pgAdmin III para PostgreSQL
- iReports para JasperReports
- DB DesignerFork
- IDE Netbeans 6.9.1

CAPITULO II

MARCO TEÓRICO

2.1.INTRODUCCION.

El presente capitulo tiene la finalidad de aclarar y describir conceptos y definiciones que serán utilizados en el desarrollo de los diferentes capítulos del proyecto.

Los diferentes conceptos que se desarrollaran a continuación son necesarios para el buen entendimiento del mismo, tanto para personas que estén familiarizadas como las que no lo estén con el desarrollo de software.

2.2. INTRODUCCION A LA INGENIERIA DE SOFTWARE.

Cuando un software se desarrolla con éxito, cuando satisface las necesidades de las personas que lo utilizan; cuando funciona de forma impecable durante mucho tiempo; cuando es fácil de modificar o incluso es más fácil de utilizar, puede cambiar todas las cosas y de hecho cambiar para mejor. Ahora bien, cuando un software falla, cuando los usuarios no quedan satisfechos, cuando es propenso a errores, cuando es difícil de cambiar e incluso más difícil de utilizar, pueden ocurrir y de hecho ocurren verdaderos desastres. Todos queremos desarrollar un software que haga bien las cosas, evitando que esas cosas malas aparezcan. Para tener éxito al diseñar y construir un software necesitaremos disciplina. Es decir, necesitaremos un enfoque de ingeniería.

2.2.1. Software.

En primer lugar se va a tratar un concepto tan importante como es el software. Es importante entender este concepto para poder pasar a definir a continuación lo que es la ingeniería del software.

Algunas definiciones de software:

- IEEE[3]Std. 610 define el software como "Programas, procedimientos y documentación y datos asociados, relacionados con la operación de un sistema informático"
- Según el Webster's New CollegiateDictionary (1975), "Software es un conjunto de programas, procedimientos y documentación relacionada asociados con un sistema, especialmente un sistema informático".

El software se puede definir como el conjunto de tres componentes:

- Programas (instrucciones): este componente proporciona la funcionalidad deseada y el rendimiento cuando se ejecute.
- Datos: este componente incluye los datos necesarios para manejar y probar los programas y las estructuras requeridas para mantener y manipular estos datos.
- Documentos: este componente describe la operación y uso del programa.

Figura 2.1: Componentes del Software

Fuente:[INTECO, 2009]

2.2.1.1. Componentes del Software.

Es importante contar con una definición exhaustiva del software ya que de otra manera se podrían olvidar algunos componentes. Una percepción común es que el software sólo consiste en programas. Sin embargo, los programas no son los únicos componentes del software.

[3] IEEE: Instituto de Ingenieros Eléctricos y Electrónicos(Institute of Electrical and ElectronicsEngineers)

a) Programas

Los programas son conjuntos de instrucciones que proporcionan la funcionalidad deseada cuando son ejecutadas por el ordenador. Están escritos usando lenguajes específicos que los ordenadores pueden leer y ejecutar, tales como lenguaje ensamblador, Basic, FORTRAN, COBOL, C... Los programas también pueden ser generados usando generadores de programas.

b) Datos

Los programas proporcionan la funcionalidad requerida manipulando datos. Usan datos para ejercer el control apropiado en lo que hacen. El mantenimiento y las pruebas de los programas también necesitan datos. El diseño del programa asume la disponibilidad de las estructuras de datos tales como bases de datos y archivos que contienen datos.

c) Documentos

Además de los programas y los datos, los usuarios necesitan también una explicación de cómo usar el programa.

Documentos como manuales de usuario y de operación son necesarios para permitir a los usuarios operar con el sistema.

Los documentos también son requeridos por las personas encargadas de mantener el software para entender el interior del software y modificarlo, en el caso en que sea necesario.

2.2.1.2. Características del Software.

A lo largo de los años, se han evolucionado muchas formas de producir bienes de mejor calidad en el sector de las manufacturas. Este conocimiento puede extenderse a la construcción de productos software de mejor calidad si los profesionales del software entienden las características propias del software.

Para poder comprender lo que es el software (y consecuentemente la ingeniería del software), es importante examinar las características del software que lo diferencian de otras cosas que el hombre puede construir.

El software es esencialmente un conjunto de instrucciones (programas) que proporcionan la funcionalidad requerida, los datos relacionados y documentos. Por lo tanto, el software es un elemento lógico y se diferencia del hardware, un elemento físico, en sus características.

El software se desarrolla, no se fabrica en el sentido clásico. Aunque existen similitudes entre el desarrollo del software y la construcción del hardware, ambas actividades son fundamentalmente distintas.

Cada producto software es diferente porque se construye para cumplir los requisitos únicos de un cliente. Cada software necesita, por lo tanto, ser construido usando un enfoque de ingeniería.

Construir un producto software implica entender qué es necesario, diseñar el producto para que cumpla los requisitos, implementar el diseño usando un lenguaje de programación y comprobar que el producto cumple con los requisitos. Todas estas actividades se llevan a cabo mediante la ejecución de un proyecto software y requiere un equipo trabajando de una forma coordinada.

El proceso usado para construir software es diferente de la fabricación del hardware, donde las máquinas se usan para producir partes y cada trabajador sólo necesita realizar la tarea asignada o usar una máquina.

En el software, el recurso principal son las personas. No es siempre posible acelerar la construcción de software añadiendo personas porque la construcción de software requiere un esfuerzo en equipo. El equipo tiene que trabajar de forma coordinada y compartir un objetivo de proyecto común. Se necesita comunicación efectiva dentro del equipo.

Un nuevo miembro del equipo no es inmediatamente productivo y necesita la iniciación adecuada al equipo y la formación para realizar el trabajo. Esto requiere una inversión de tiempo y esfuerzo por parte de los miembros del equipo existentes y les puede distraer de su propio trabajo.

Otra característica del software es que no se estropea. Los defectos no detectados harán que falle el programa durante las primeras etapas de su vida. Sin embargo, una vez que se corrigen (suponiendo que no se introducen nuevos errores) los fallos disminuyen.

El software no se estropea, pero se deteriora. Durante su vida, el software sufre cambios (mantenimiento). Conforme se hacen los cambios, es bastante probable que se introduzcan nuevos defectos, lo que hace que el software se vaya deteriorando debido a los cambios.

Otro aspecto del software es que, debido a que la industria del software es nueva, el software se diferencia del hardware en el aspecto de uso de componentes. Aunque la mayoría de la industria tiende a ensamblar componentes, la mayoría del software se construye a medida.

Los componentes reutilizables se han creado para que el ingeniero pueda concentrarse en elementos verdaderamente innovadores de un diseño. En el mundo del software es algo que sólo ha comenzado a lograrse en productos lanzados a gran escala.

El componente software debería diseñarse e implementarse para que pueda volver a ser reutilizado en muchos programas diferentes. Hoy en día, se ha extendido la visión de la reutilización para abarcar tanto algoritmos como estructuras de datos, permitiendo al ingeniero del software crear nuevas aplicaciones a partir de las partes reutilizables.

El hardware usa componentes estándar con funciones e interfaces bien definidas. El uso de estos componentes ayuda a evitar reinventar la rueda. La fase de diseño en el ciclo de vida de un producto hardware implica seleccionar los componentes disponibles más adecuados y decidir el enfoque para montarlos. Los componentes de hardware estándar son útiles porque conducen a:

- Reducir el coste y el tiempo de lanzamiento al mercado.
- Buena calidad.
- Ingeniería rápida.
- Fácil mantenimiento.
- Fácil mejora.

El software se crea normalmente desde cero. Con frecuencia se construye de acuerdo a los requisitos específicos de un cliente y no se crea por la unión de componentes existentes.

Como la industria del hardware, la industria del software está intentando adoptar el mecanismo de reutilizar para hacer más fácil y más rápida la construcción. Las ventajas de la reutilización de software están siendo entendidas y apreciadas. Existen algunos elementos

reutilizables a través de librerías de funciones y objetos reutilizables que combinan funciones y datos.

Mientras que la reutilización y el montaje basado en componentes se están incrementando, la mayoría del software continua siendo construido de forma personalizada, y los niveles de reutilización actuales están lejos de los que deberían ser. Además, la tarea de identificar componentes reutilizables potenciales es difícil porque cada producto software es único y distinto.

La industria del software tiene procesos bien definidos para la reutilización de componentes. Esto incluye procesos para la construcción de componentes, almacenamiento de los mismos en librerías de donde se pueden extraer para su reutilización y entonces incorporarlos.

2.2.1.3. Tipos de Software.

El software puede dividirse en dos grandes categorías:

a) Software de aplicaciones.

Se usan para proveer servicios a clientes y ejecutar negocios de forma más eficiente. El software de aplicaciones puede ser un sistema pequeño o uno grande integrado. Como ejemplos de este tipo de software están: un sistema de cuentas, un sistema de planificación de recursos, etc.

b) Software de sistemas.

El software de sistemas se usa para operar y mantener un sistema informático. Permite a los usuarios usar los recursos del ordenador directamente y a través de otro software. Algunos ejemplos de este tipo de software son: sistemas operativos, compiladores y otras utilidades del sistema.

2.2.1.4. Aplicaciones del Software.

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales (es decir, un algoritmo) (excepciones notables a esta regla son el software de los sistemas expertos y de redes neuronales). El contenido y determinismo de la información son factores importantes a considerar para determinar la

naturaleza de una aplicación software. El contenido se refiere al significado y a la forma de la información de entrada y salida. Por ejemplo, muchas aplicaciones bancarias usan unos datos de entrada muy estructurados (una base de datos) y producen informes con determinados formatos. El software que controla una máquina automática (por ejemplo:un control numérico) acepta elementos discretos con una estructura limitada y produce órdenes concretas para la máquina en rápida sucesión.

El determinismo de la información se refiere a la predictibilidad del orden y del tiempo de llegada de los datos. Un programa de análisis de ingeniería acepta datos que están en un orden predefinido, ejecuta algoritmos de análisis sin interrupción y produce los datos resultantes en un informe o formato gráfico. Un sistema operativo multiusuario, por otra parte, acepta entradas que tienen un contenido variado y que se producen en instantes arbitrarios, ejecuta algoritmos que pueden ser interrumpidos en condiciones externas y produce una salida que depende de una función del entorno y del tiempo. Las aplicaciones con estas características se dice que son indeterminadas.

Algunas veces es difícil establecer categorías genéricas para las aplicaciones del software que sean significativas. Conforme aumenta la complejidad del software, es más difícil establecer compartimentos nítidamente separados. Las siguientes áreas del software indican la amplitud de las aplicaciones potenciales:

a) Software de sistemas

El software de sistemas es un conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistemas (por ejemplo: compiladores[4], editores y utilidades de gestión de archivos) procesan estructuras de información complejas pero determinadas. Otras aplicaciones de sistemas (por ejemplo: ciertos componentes del sistema operativo, utilidades de manejo de periféricos, procesadores de telecomunicaciones) procesan datos en gran medida indeterminados. En cualquier caso, el área del software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una

[4] Compilador: Programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar.

planificación, una compartición de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas.

b) Software de tiempo real.

El software que coordina/analiza/controla sucesos del mundo real conforme ocurren. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo y un componente de monitorización que coordina todos los demás componentes, de forma que pueda mantenerse el respuesta en tiempo real.

c) Software de gestión.

El proceso de la información comercial constituye la mayor de las áreas de aplicación del software. Los sistemas discretos (por ejemplo: nóminas, cuentas de haberes-débitos, inventarios, etc.) han evolucionado hacia el software de sistemas de información de gestión (SIG) que accede a una o más bases de datos que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamiento de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en puntos de venta).

d) Software de ingeniería y científico.

Este tipo de software está caracterizado por los algoritmos de manejo de números. Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática. Sin embargo las nuevas aplicaciones del área de ingeniería/ciencia se han alejado de los algoritmos convencionales numéricos. El diseño asistido por computadora (CAD[5]), la simulación de sistemas y otras aplicaciones interactivas,

[5] CAD: Computer – AidedDesign

han comenzado a coger características del software de tiempo real e incluso de software de sistemas.

e) Software empotrado.

Los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (por ejemplo: el control de las teclas de un horno microondas) o suministrar una función significativa y con capacidad de control (por ejemplo: funciones digitales en un automóvil, tales como control de la gasolina, indicadores en el salpicadero, sistemas de frenado, etc.)

f) Software de computadoras personales.

El mercado del software de computadoras personales ha germinado en las pasadas décadas. El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, multimedia, entretenimiento, gestión de bases de datos, aplicaciones financieras, de negocios y personales y redes o acceso a bases de datos externas son algunas de los cientos de aplicaciones.

g) Software basado en web.

Las páginas web buscadas por un explorador son software que incorpora instrucciones ejecutables y datos.

h) Software de inteligencia artificial.

El software de inteligencia artificial hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Los sistemas expertos, también llamados sistemas basados en el conocimiento, reconocimiento de patrones[6] (imágenes y voz), redes neuronales artificiales, prueba de teoremas y los juegos son representativos de las aplicaciones de esta categoría.

---

[6] Patrones: Conjunto de entidades que comparten alguna característica que las diferencia del resto.

2.2.2. Ingeniería del Software.

El término ingeniería de software se define en el DRAE[7] como:

1. Conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía.
2. Profesión y ejercicio del ingeniero

Y el término ingeniero se define como:

1. Persona que profesa o ejerce la ingeniería.

De igual modo, la Real Academia de Ciencias Exactas, Físicas y Naturales de España define el término Ingeniería como: conjunto de conocimientos y técnicas cuya aplicación permite la utilización racional de los materiales y de los recursos naturales, mediante invenciones, construcciones u otras realizaciones provechosas para el hombre.

Otras definiciones:

1. Ingeniería del Software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas de software (Zelkovitz, 1978)
2. Ingeniería del Software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producción de software (Bohem, 1976)
3. Ingeniería del Software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea fiable y trabaje en máquinas reales (Bauer, 1972)
4. La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y mantenimiento del software; es decir, la aplicación de ingeniería al software. (IEEE[8], 1993)

---

[7] DRAE: Diccionario de la Real Academia Española

[8] IEEE: Instituto de Ingenieros Eléctricos y Electrónicos(Institute of Electrical and ElectronicsEngineers)

La definición de IEEE describe la ingeniería del software como un enfoque sistemático cubriendo los aspectos del desarrollo, operación y mantenimiento. Este enfoque es disciplinado y cuantificable.

2.2.2.1. Historia.

El término ingeniería del software apareció por primera vez en la conferencia de ingeniería de software de la OTAN[9] en 1968 y fue mencionado para provocar el pensamiento sobre la crisis de software del momento. Desde entonces, ha continuado como una profesión y campo de estudio dedicado a la creación de software de alta calidad, barato, con capacidad de mantenimiento y rápido de construir. Debido a que el campo es todavía relativamente joven comparado con otros campos de la ingeniería, hay todavía mucho trabajo y debate sobre qué es realmente la ingeniería del software, y si se merece el título de ingeniería. Ha crecido orgánicamente fuera de las limitaciones de ver el software sólo como programación.

Mientras que el término ingeniería del software fue acuñado en una conferencia en 1968, los problemas que intentaba tratar empezaron mucho antes. La historia de la ingeniería del software está entrelazada con las historias contrapuestas de hardware y software.

Cuando el ordenador digital moderno apareció por primera vez en 1941, las instrucciones para hacerlo funcionar estaban conectadas dentro de la máquina. Las personas relacionadas con la ingeniería rápidamente se dieron cuenta de que este diseño no era flexible e idearon la arquitectura de programa almacenado o arquitectura von Neumann. De esta forma la primera división entre "hardware" y "software" empezó con la abstracción usada para tratar la complejidad de la computación.

Los lenguajes de programación empezaron a aparecer en la década de 1950 y este fue otro paso importante en la abstracción. Los lenguajes principales como Fortran, Algol y Cobol se lanzaron a finales de los 50s para tratar con problemas científicos, algorítmicos y de negocio respectivamente. Dijsktra escribió "GotoStatementConsideredHarmful" en 1968 y David Parnas introdujo el concepto clave de la modularidad y encapsulación en 1972 para ayudar a los programadores a tratar con la complejidad de los sistemas de software. Un sistema

---

[9] OTAN: Organización del Tratado Atlántico Norte.

software para gestionar el hardware, denominado sistema operativo también se introdujo, más notablemente por Unix en 1969. En 1967, el lenguaje Simula introdujo el paradigma de la programación orientada a objetos.

Estos avances en software se encontraron con más avances en el hardware. A mediados de los 70s, la microcomputadora fue introducida, haciendo económico a los aficionados a obtener una computadora y escribir software para él. Esto sucesivamente condujo al famoso ordenador personal o PC y Microsoft Windows. El ciclo de vida de desarrollo de software o SDLC también empezó a aparecer como un consenso para la construcción centralizada de software a mediados de los 80s. A finales de los 70s y principios de los 80 se vieron varios lenguajes de programación orientados a objetos inspirados en Simula, incluyendo C++, Smalltalk y Objective C.

El software open source empezó a aparecer a principios de los 90s en la forma de Linux y otros software introduciendo el "bazaar[10]" o el estilo descentralizado de construcción de software. Después aparecieron Internet y la World Wide Web a mediados de los 90s cambiando de nuevo la ingeniería del software. Los sistemas distribuidos ganaron dominio como forma de diseñar sistemas y el lenguaje de programación Java se introdujo como otro paso en la abstracción, teniendo su propia máquina virtual. Varios programadores colaboraron y escribieron el manifiesto ágil que favoreció procesos más ligeros para crear software más barato y en menos tiempo.

2.2.2.2. Etapas.

La ingeniería de software requiere llevar a cabo numerosas tareas, dentro de etapas como las siguientes:

a) Análisis de requisitos

Extraer los requisitos de un producto software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere habilidad y experiencia en la ingeniería del software para reconocer requisitos incompletos, ambiguos o contradictorios. El resultado del análisis de requisitos con el cliente se plasma en el

[10]Bazaar: Sistema de control de versiones distribuido.

documento Especificación de Requisitos. Asimismo, se define un diagrama de entidad/relación, en el que se plasman las principales entidades que participarán en el desarrollo de software.

La captura, análisis y especificación de requisitos (incluso pruebas de ellos), es una parte crucial; de esta etapa depende en gran medida el logro de los objetivos finales. Se han ideado modelos y diversos procesos de trabajo para estos fines. Aunque aún no está formalizada, se habla de la Ingeniería de Requisitos.

La IEEE Std. 830-1998 normaliza la creación de las especificaciones de requisitos software.

b) Especificación

Es la tarea de escribir detalladamente el software a ser desarrollado, en una forma matemáticamente rigurosa. En la realidad, la mayoría de las buenas especificaciones han sido escritas para entender y afinar aplicaciones que ya estaban desarrolladas. Las especificaciones son más importantes para las interfaces externas, que deben permanecer estables.

c) Diseño y arquitectura

Se refiere a determinar cómo funcionará el software de forma general sin entrar en detalles. Consisten en incorporar consideraciones de la implementación tecnológica, como el hardware, la red, etc. Se definen los casos de uso para cubrir las funciones que realizará el sistema, y se transformarán las entidades definidas en el análisis de requisitos en clases de diseño, obteniendo un modelo cercano a la programación orientada a objetos.

d) Programación

Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería del software, pero no necesariamente es la que demanda mayor trabajo ni la más complicada. La complejidad y la duración de esta etapa está íntimamente relacionada al o a los lenguajes de programación utilizados, así como al diseño previamente realizado.

La programación es el proceso de diseñar, escribir, depurar y mantener el código fuente de programas computacionales. El código fuente es escrito en un lenguaje de programación.

e) Prueba

Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación del problema. Una técnica de prueba es probar por separado cada módulo del software y luego probarlo de forma integral, para así llegar al objetivo. Se considera una buena práctica que las pruebas sean efectuadas por alguien distinto al desarrollador que la programó.

f) Mantenimiento

Mantener y mejorar el software para solventar errores descubiertos y tratar con nuevos requisitos. El mantenimiento puede ser de cuatro tipos: perfectivo (mejorar la calidad interna de los sistemas), evolutivo (incorporaciones, modificaciones y eliminaciones necesarias en un producto software para cubrir la expansión o cambio en las necesidades del usuario), adaptativo (modificaciones que afectan a los entornos en los que el sistema opera, por ejemplo, cambios de configuración del hardware, software de base, gestores de base de datos, comunicaciones) y correctivo (corrección de errores).

2.2.2.3. Objetivo de la Ingeniería de Software.

Hemos visto que todas las definiciones de la ingeniería del software se centran en el uso de un enfoque sistemático para la construcción de software.

El objetivo primario de la ingeniería del software es construir un producto de alta calidad de una manera oportuna. Trata de conseguir este objetivo primario usando un enfoque de ingeniería.

Para conseguir el objetivo de construir productos de alta calidad dentro de la planificación, la ingeniería del software emplea una serie de prácticas para:

- Entender el problema
- Diseñar una solución
- Implementar la solución correctamente
- Probar la solución
- Gestionar las actividades anteriores para conseguir alta calidad

La ingeniería del software representa un proceso formal que incorpora una serie de métodos bien definidos para el análisis, diseño, implementación y pruebas del software y sistemas. Además, abarca una amplia colección de métodos y técnicas de gestión de proyectos para el aseguramiento de la calidad y la gestión de la configuración del software.

2.2.2.4. Relevancia de la Ingeniería de Software.

Hoy en día, los productos software se construyen con un nivel de urgencia que no se veía en años anteriores. La prioridad más alta de las compañías es reducir el tiempo de salida al mercado, que es la base del desarrollo rápido.

La ingeniería de software es percibida por algunos como demasiado formal, que consume demasiado tiempo y demasiado estructurada para la flexibilidad necesaria durante el desarrollo de hoy en día. Las personas que hacen estas críticas exponen que no se pueden permitir la formalidad de un enfoque de ingeniería para construir software porque necesitan desarrollar productos de forma rápida. Las personas que lanzan tales objeciones ven la ingeniería como una disciplina estática y piensan que no se puede adaptar a las necesidades cambiantes del negocio y la industria. La verdad es, sin embargo, que la ingeniería del software es adaptativa y por lo tanto, relevante para cualquiera que construya un producto software.

La ingeniería del software es adaptativay no una metodología rígida. Es una filosofía que puede ser adaptada y aplicada a todas las actividades y dominios de aplicación del desarrollo de software.

La ingeniería del software proporciona una amplia colección de opciones que los profesionales pueden elegir para construir productos de alta calidad. Sin embargo, no hay un enfoque de ingeniería individual o un conjunto de procesos, métodos o herramientas de ingeniería del software para construir un producto software.

El enfoque de ingeniería del software, incluyendo los procesos, métodos y herramientas puede y debería ser adaptada al producto, la gente que lo construye y el entorno del negocio.

Los profesionales del software, por lo tanto, no deberían ser dogmáticos sobre la ingeniería del software. No es una religión y no hay verdades absolutas.

2.3.METODOLOGIAS DE DESARROLLO DE SOFTWARE.

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en muchos otros. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales.

Un objetivo de décadas ha sido encontrar procesos y metodologías, que sean sistemáticas, predecibles y repetibles, a fin de mejorar la productividad en el desarrollo y la calidad del producto software.

La evolución de la disciplina de ingeniería del software ha traído consigo propuestas diferentes para mejorar los resultados del proceso de construcción. Las metodologías tradicionales haciendo énfasis en la planificación y las metodologías ágiles haciendo énfasis en la adaptabilidad del proceso, delinean las principales propuestas presentes.

Los requisitos son una lista de cosas que queremos que haga nuestro programa. Lo normal es que recopilemos dicha lista hablando con todas las personas que podamos: usuarios de nuestro programa, expertos en el tema de que trata el programa (ajedrez), etc, etc.

2.3.1. Definición de Metodología.

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo.

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, incremental...). Definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas.

La metodología para el desarrollo de software en un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Una definición estándar de metodología puede ser el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea.

Si esto se aplica a la ingeniería del software, podemos destacar que una **metodología**:

- Optimiza el proceso y el producto software.
- Métodos que guían en la planificación y en el desarrollo del software.
- Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Una metodología define una estrategia global para enfrentarse con el proyecto. Entre los elementos que forman parte de una metodología se pueden destacar:

- Fases: tareas a realizar en cada fase.
- Productos: E/S de cada fase, documentos.
- Procedimientos y herramientas: apoyo a la realización de cada tarea.
- Criterios de evaluación: del proceso y del producto. Saber si se han logrado los objetivos.

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo han evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. Una metodología de desarrollo de sistemas no tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo.

Una metodología de desarrollo de software o metodología de desarrollo de sistemas en ingeniería de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información.

El marco de trabajode una metodología de desarrollo de software consiste en:

- Una filosofía de desarrollo de software, con el enfoque o enfoques del proceso de desarrollo de software.
- Múltiples herramientas, modelos y métodos para ayudar en el proceso de desarrollo de software.

Estos marcos de trabajo están con frecuencia vinculados a algunos tipos de organizaciones, que se encargan del desarrollo, soporte de uso y promoción de la metodología. La metodología con frecuencia se documenta de alguna manera formal.

2.3.2. Ventajas

Son muchas las ventajas que puede aportar el uso de una metodología. A continuación se van a exponer algunas de ellas, clasificadas desde distintos puntos de vista.

Desde el punto de vista de gestión:

- Facilitar la tarea de planificación.
- Facilitar la tarea del control y seguimiento de un proyecto.
- Mejorar la relación coste/beneficio.
- Optimizar el uso de recursos disponibles.
- Facilitar la evaluación de resultados y cumplimiento de los objetivos.
- Facilitar la comunicación efectiva entre usuarios y desarrolladores.

Desde el punto de vista de los ingenieros del software:

- Ayudar a la comprensión del problema.
- Optimizar el conjunto y cada una de las fases del proceso de desarrollo.
- Facilitar el mantenimiento del producto final.
- Permitir la reutilización de partes del producto.

Desde el punto de vista del cliente o usuario:

- Garantía de un determinado nivel de calidad en el producto final.
- Confianza en los plazos de tiempo fijados en la definición del proyecto.
- Definir el ciclo de vida que más se adecue a las condiciones y características del desarrollo.

2.3.3. Metodologías Tradicionales y Agiles.

Desarrollar un buen software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto es trascendental para el éxito del producto.

Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado.

2.3.3.1. Metodologías tradicionales

Las metodologías tradicionales son denominadas, a veces, de forma peyorativa, como metodologías pesadas.

Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil.

Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.)

2.3.3.2. Metodologías ágiles

Este enfoque nace como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se basa en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa. Basan su fundamento en la adaptabilidad de los procesos de desarrollo.

Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan.

2.3.3.3. ¿Metodologías ágiles o metodologías tradicionales?

En las metodologías tradicionales el principal problema es que nunca se logra planificar bien el esfuerzo requerido para seguir la metodología. Pero entonces, si logramos definir métricas que apoyen la estimación de las actividades de desarrollo, muchas prácticas de metodologías tradicionales podrían ser apropiadas. El no poder predecir siempre los resultados de cada proceso no significa que estemos frente a una disciplina de azar. Lo que significa es que estamos frente a la necesidad de adaptación de los procesos de desarrollo que son llevados por parte de los equipos que desarrollan software.

Tener metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta una idea interesante. Estas metodologías pueden involucrar prácticas tanto de metodologías ágiles como de metodologías tradicionales. De esta manera podríamos tener una metodología por cada proyecto, la problemática sería definir cada una de las prácticas, y en el momento preciso definir parámetros para saber cuál usar.

Es importante tener en cuenta que el uso de un método ágil no vale para cualquier proyecto. Sin embargo, una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero y por eso las personas que no estén acostumbradas a seguir procesos encuentran estas metodologías bastante agradables.

Tabla 2.1: Metodologías Agiles vs Tradicionales

Metodologías Tradicionales	Metodologías Agiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios.
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Más artefactos	Pocos artefactos
Más roles	Pocos roles
Grupos grandes y posiblemente distribuidos	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
La arquitectura del software es esencial y seexpresa mediante modelos	Menos énfasis en la arquitectura del software
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible

Fuente:[INTECO, 2009]

2.3. PROGRAMACION EXTREMA XP.

La programación extrema (XP) es un enfoque de la ingeniería del software formulado por Kent Beck. Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto y aplicarlo de manera dinámica durante el ciclo de vida del software.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. A Kent Beck se le considera el padre de XP.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre.

2.3.1. Elementos de la Metodología.

2.3.1.1. Las historias de usuario:

Son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas. Las historias de usuario se descomponen en tareas de programación y se asignan a los programadores para ser implementadas durante una iteración.

2.3.1.2. Roles XP:

Los roles de acuerdo con la propuesta de Beck son:

- Programador: el programador escribe las pruebas unitarias y produce el código del sistema
- Cliente: escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide

cuáles se implementan en cada iteración centrándose en apoyar mayor valor al negocio.

- Encargado de pruebas (tester): ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para las pruebas.
- Encargado de seguimiento (tracker): proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- Entrenador (coach): es el responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor: es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- Gestor (bigboss): es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

2.3.1.3. Proceso XP:

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El programador construye ese valor.
4. Vuelve al paso 1.
5. En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden.

No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse de que el sistema tenga el mayor valor de negocio posible.

El ciclo de vida ideal de XP consisten en 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto.

2.3.2. Ciclo de vida de XP.

El ciclo de vida ideal de XP consiste de seis fases como podemos ver en la siguiente figura:

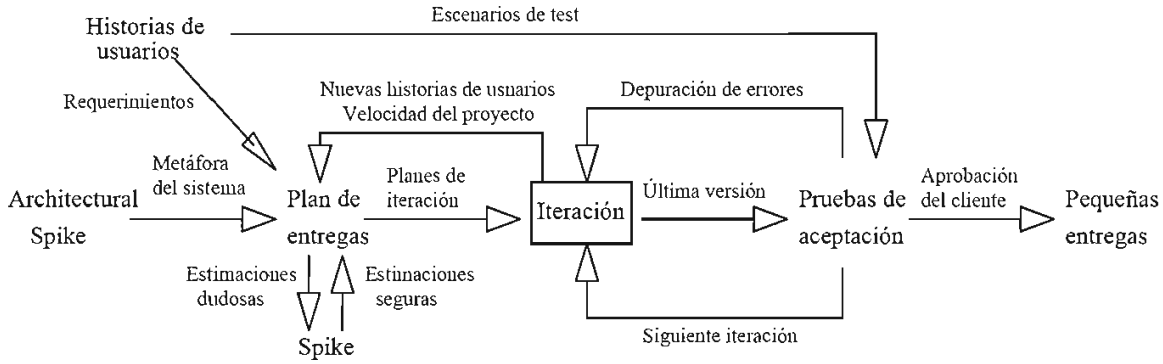

Figura 2.3: Ciclos de vida de XP

Fuente: [ONESS, 2005]

2.3.2.1. Exploración.

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.3.2.2. Planificación de la Entrega (*Release*).

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un

registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.3.2.3. Iteraciones.

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

2.3.2.4. Producción.

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

2.3.2.5. Mantenimiento.

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

2.3.2.6. Muerte del Proyecto.

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

2.3.3. Prácticas XP.

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del coste del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas:

- **El juego de la planificación.** Hay una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- **Entregas pequeñas.** Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más de 3 meses.
- **Metáfora.** El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombre que

actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).

- **Diseño simple.** Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- **Pruebas.** La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.
- **Refactorización (refactoring).** Es una actividad constante de reestructuración del código con el objetivo de evitar duplicación del código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.
- **Programación en parejas.** Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores...). Esta práctica se explicará más en profundidad en apartados sucesivos.
- **Propiedad colectiva del código.** Cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- **Integración continúa.** Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Esta práctica se desarrollará más en profundidad en apartados sucesivos.
- **40 horas por semana.** Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
- **Cliente in-situ.** El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor al negocio y los programadores pueden resolver de manera más inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.
- **Estándares de programación.** XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

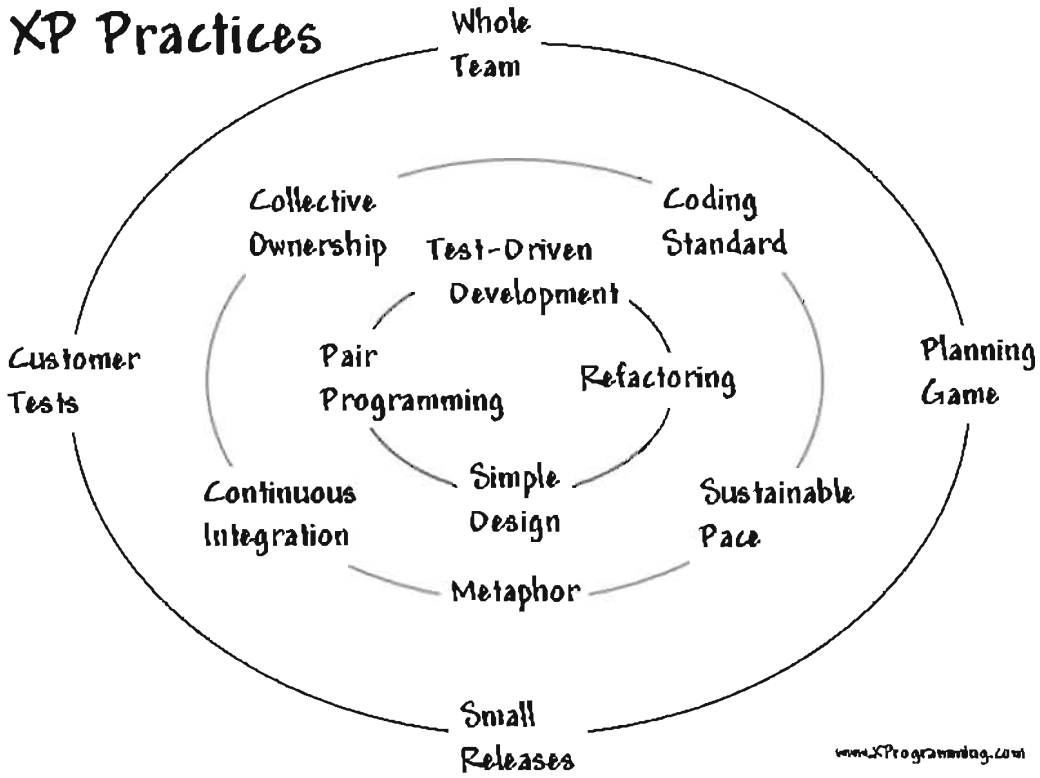

Figura 2.4: Prácticas XP

Fuente: [ONESS, 2005]

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. La mayoría de las prácticas propuestas por XP no son novedosas sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

2.3.4. Principios XP.

Los principios básicos de la programación extrema son: simplicidad, comunicación, retroalimentación y coraje.

- **Simplicidad**: la simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando eso sí que el código esté autodocumentado. Para ello se deben elegir adecuadamente los nombres de las variables, métodos y clases. Los nombres largos no disminuyen la eficiencia del código ni el tiempo de desarrollo gracias a las herramientas de autocompletado y refactorización que existen actualmente. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas se asegura que mientras más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.
- **Comunicación**: la comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor mientras más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse sólo aquello que no va a variar, por ejemplo, el objetivo de una clase o la funcionalidad de un método. Las pruebas unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de cómo utilizar su funcionalidad. Los programadores se comunican constantemente gracias a la programación por parejas. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide qué características tienen prioridad y siempre debe estar disponible para solucionar dudas.
- **Retroalimentación** (feedback): al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en los que es más importante. Considérense los problemas que derivan de tener ciclos muy largos. Meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las

pruebas unitarias informan sobre el estado de salud del código. Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

- **Coraje o valentía**: para los gerentes la programación en parejas puede ser difícil de aceptar, parece como si la productividad se fuese a reducir a la mitad ya que sólo la mitad de los programadores está escribiendo código. Hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la productividad. La simplicidad es uno de los principios más difíciles de adoptar. Se requiere coraje para implementar las características que el cliente quiere ahora sin caer en la tentación de optar por un enfoque más flexible que permite futuras modificaciones. No se debe emprender el desarrollo de grandes marcos de trabajo mientras el cliente espera. En ese tiempo el cliente no recibe noticias sobre los avances del proyecto y el equipo de desarrollo no recibe retroalimentación para saber si va en la dirección correcta. La forma de construir marcos de trabajo es mediante la refactorización del código en sucesivas aproximaciones.

2.3.5. Actividades de XP.

XP describe cuatro actividades que se llevan a cabo dentro del proceso de desarrollo de software.

2.3.5.1. Codificar

Los defensores de XP argumentan que el único producto realmente importante del proceso de desarrollo de sistemas es el código (un concepto al que dan una definición más amplia que la que pueden dar otros). Sin el código no se tiene nada. La codificación puede ser dibujar diagramas que generarán código, hacer scripts de sistemas basados en web o codificar un programa que ha de ser compilado.

La codificación también puede usarse para entender la solución más apropiada. Por ejemplo, XP recomendaría que si nos enfrentamos con varias alternativas para un problema de programación, uno debiera simplemente codificar todas las soluciones y determinar con pruebas automatizadas qué solución es la más adecuada. La codificación puede ayudar también a comunicar pensamientos sobre problemas de programación. Un programador que

trate con un problema de programación complejo y encuentre difícil explicar la solución al resto, podría codificarlo y usar el código para demostrar lo que quería decir. El código, dicen los partidarios de esta posición, es siempre claro y conciso y no se puede interpretar de más de una forma. Otros programadores pueden dar retroalimentación de ese código codificando también sus pensamientos.

2.3.5.2. Probar

Nadie puede estar seguro de algo si no lo ha probado. Las pruebas no es una necesidad primaria percibida por el cliente. Mucho software se libera sin unas pruebas adecuadas y funciona. En el desarrollo de software, XP dice que esto significa que uno no puede estar seguro de que una función funciona si no la prueba. Esto sugiere la pregunta de definir de lo que uno puede no estar seguro.

- No puedes estar seguro de si lo que has codificado es lo que querías significar. Para probar esta incertidumbre, XP usa pruebas unitarias. Son pruebas automatizadas que prueban el código. El programador intentará escribir todas las pruebas en las que pueda pensar que puedan cargarse el código que está escribiendo; si todas las pruebas se ejecutan satisfactoriamente entonces el código está completo.
- No puedes estar seguro de si lo que querías significar era lo que deberías. Para probar esta incertidumbre, XP usa pruebas de aceptación basadas en los requisitos dados por el cliente.

2.3.5.3. Escuchar

Los programadores no saben necesariamente todo sobre el lado del negocio del sistema bajo desarrollo. La función del sistema está determinada por el lado del negocio. Para que los programadores encuentren cual debe ser la funcionalidad del sistema, deben escuchar al negocio. Tienen que escuchar las necesidades de los clientes. También tienen que intentar entender el problema del negocio y dar a los clientes retroalimentación sobre el problema, para mejorar el propio entendimiento del cliente sobre el problema.

2.3.5.4. Diseñar

Desde el punto de vista de la simplicidad, uno podría decir que el desarrollo de sistemas no necesita más que codificar, probar y escuchar. Si estas actividades se desarrollan bien, el

resultado debería ser un sistema que funcionase. En la práctica, esto no ocurre. Uno puede seguir sin diseñar, pero un momento dado se va a atascar. El sistema se vuelve muy complejo y las dependencias dentro del sistema dejan de estar claras. Uno puede evitar esto creando una estructura de diseño que organice la lógica del diseño. Buenos diseños evitarán perdidas de dependencias dentro de un sistema; esto significa que cambiar una parte del sistema no tendrá por qué afectar a otras.

2.3.6. Artefactos XP.

A continuación describimos los artefactos de XP, entre los que se encuentran: Historias de Usuario, Tareas de Ingeniería, Tarjetas CRC y Casos de prueba.

2.3.6.1. Historias de Usuario.

Representan una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

	HISTORIA DE USUARIO
Número:	**Nombre Historia de Usuario:**
Usuario:	**Iteración Asignada:**
Prioridad en Negocio: (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción:	
Observaciones:	

Figura 2.4: Tarjeta "Historia de Usuario"

Fuente:[Beck, 2000]

Estas deben proporcionar sólo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere la implementación de la historia, difiere de los casos

de uso porque son escritos por el cliente, no por los programadores, empleando terminología del cliente. "Las historias de usuario son más "amigables" que los casos de uso formales".

Las Historias de Usuario tienen tres aspectos:

- Tarjeta: en ella se almacena suficiente información para identificar y detallar la historia.
- Conversación: cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación)
- Pruebas de Aceptación: permite confirmar que la historia ha sido implementada correctamente.

2.3.6.2. Tareas de Ingeniería.

Las tarjetas de Tareas de Ingeniería nos permiten presentar el o los objetivos que una historia de usuario nos brindará. Puede haber una o más tareas de ingeniería por cada historia de usuario. Esto facilitara a la realización de sus objetivos dividiendo su complejidad en el caso de que lo fueran.

	TAREA DE INGENIERÍA
Número:	**Historia de Usuario (Nro. Y Nombre):**
Nombre Tarea:	
Tipo de Tarea : Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción:	

Figura 2.5: Tarjeta "Tarea de Ingeniería"

Fuente: [Beck, 2000]

2.3.6.3. Tarjetas CRC[11].

Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. En la siguiente figura se muestra cómo se distribuye esta información.

Figura 2.6: Tarjeta CRC

Fuente: [Beck, 2000]

Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realiza, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

En la práctica conviene tener pequeñas tarjetas de cartón, que se llenarán y que son mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas.

Los pasos a seguir para llenar las tarjetas son los siguientes:

- Encontrar clases
- Encontrar responsabilidades
- Definir colaboradores
- Disponer las tarjetas

[11] CRC: Clase – Responsabilidad - Colaboradores

Para encontrar las clases debemos pensar qué cosas interactúan con el sistema (en nuestro caso el usuario), y qué cosas son parte del sistema, así como las pantallas útiles a la aplicación (un despliegue de datos, una entrada de parámetros y una pantalla general, entre otros). Una vez que las clases principales han sido encontradas se procede a buscar los atributos y las responsabilidades, para esto se puede formular la pregunta ¿Qué sabe la clase? y ¿Qué hace la clase? Finalmente se buscan los colaboradores dentro de la lista de clases que se tenga.

2.3.6.4. Casos de Prueba.

Las tarjetas de Casos de Prueba se utilizan para ver la aceptación de la implementación de una historia de usuario por parte del cliente.

Caso de Prueba de Aceptación	
Código:	**Historia de Usuario (Nro. y Nombre):**
Nombre:	
Descripción:	
Condiciones de Ejecución:	
Entrada / Pasos de ejecución:	
Resultado Esperado:	
Evaluación de Prueba:	

Figura 2.7: Tarjeta "Casos de Prueba"

Fuente: [Beck, 2000]

2.4. MODELADO DE DATOS.

En la informática, un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base).

En un enfoque más amplio, un modelo de datos permite describir los elementos que intervienen en una realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí.

Por lo general, un modelo de datos presenta dos sublenguajes: un Lenguaje de Definición de Datos o DDL (Data DefinitionLanguage), cuya función es describir, de una forma abstracta, las estructuras de datos y las restricciones de integridad; y un Lenguaje de Manipulación de Datos o DML (Data ManipulationLanguage), que se orienta a describir las operaciones de manipulación de los datos. A la parte del DML enfocada a la recuperación de datos, se la suele conocer como Lenguaje de Consulta o QL (QueryLanguage).

La clasificación de los modelos de datos se realiza de acuerdo al nivel de abstracción. Los modelos de datos conceptuales son aquellos que describen las estructuras de datos y restricciones de integridad. Se utilizan durante la etapa de análisis de un problema dado y están orientados a representar los elementos que intervienen y sus relaciones.

Los modelos de datos lógicos se centran en las operaciones y se implementan en algún manejador de base de datos.

Por último, podemos mencionar a los modelos de datos físicos, que son estructuras de datos a bajo nivel implementadas dentro del propio manejador.

2.4.1. Diseño Conceptual.

También conocido como representación abstracta, nos da un panorama conceptual del problema planteado, y cuanta con los siguientes elementos:

- **Propósito:** describir el contenido de información de la BD[12], más que las estructuras de almacenamiento
- **Esquema Conceptual:** descripción de alto nivel de la estructura de la BD, independiente del DBMS[13] que la manipula.
- **Modelo Conceptual:** lenguaje usado para describir esquemas conceptuales
- **Especificación de Req. + Diseño Conceptual → Esquema Conceptual de la BD**

[12] BD: Abreviación de Base de Datos.

[13] DBMS: Denominativo que se le da un Gestor de Base de Datos.

2.4.2. Diseño Lógico.

También lo podemos mencionar como una representación en computadora del Diseño Conceptual.

- **Esquema Lógico:** descripción de la estructura de la BD que puede procesar un DBMS.
- **Modelo Lógico:** lenguaje usado para especificar esquemas lógicos. Los más usados: relacional, de redes, jerárquicos.
- Depende de la clase de modelo de datos usado por el DBMS, pero no del DBMS usado (se efectúa de igual forma para todos los DBMS relacionales, porque todos usan el modelo relacional).
- **Esquema Conceptual + Diseño Lógico → Esquema Lógico de la BD**

2.4.3. Diseño Físico.

En este punto se determina las estructuras de almacenamiento físico.

- Describe las estructuras de almacenamiento y métodos usados para tener acceso efectivo a los datos.
- **Esquema Físico:** descripción de la implantación de una BD en memoria secundaria.
- Se adapta a un DBMS específico.
- **Esquema Lógico + Diseño Físico → Esquema Físico**

En el presente Proyecto de Grado haremos un poco de énfasis en el desarrollo del diseño Lógico y del Diseño Físico. Esto lo veremos en el Capítulo III "Marco Aplicativo".

2.5. CALIDAD DE SOFTWARE.

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad.

La calidad del software es medible y varía de un sistema a otro o de un programa a otro. Un software elaborado para el control de naves espaciales debe ser confiable al nivel de "cero fallas"; un software hecho para ejecutarse una sola vez no requiere el mismo nivel de

calidad; mientras que un producto de software para ser explotado durante un largo período (10 años o más), necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas deriva dos de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

La calidad se software se la realizara utilizando las métricas de calidad de software definidas en la ISO-9126.

2.5.1. ISO 9126.

El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y sub-características de la siguiente manera:

Funcionalidad: Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen las necesidades implícitas o explícitas.

Fiabilidad: Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.

Usabilidad: Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.

Eficiencia: Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

Mantenibilidad: Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

Portabilidad: Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.

CAPITULO III

MARCO APLICATIVO

3.1. INTRODUCCION.

En el presente capítulo daremos a conocer las diferentes fases de la metodología XP, juntamente con sus diferentes prácticas, principios y procesos. Haremos de cada fase algo más interactiva incluyendo en las mismas tablas y figuras que nos ayudaran a obtener una mejor comprensión de las mismas.

También veremos el desarrollo del proyecto de una forma más detallada, lo que nos ayudará a tener una mejor compresión de cómo se desarrolló el "Sistema de seguimiento y control de Visitadores Médicos".

Los entregables que esperamos obtener de cada fase los resumiremos en la siguiente tabla:

Tabla 3.1: Entregables XP Esperados

FASE	ENTREGABLE(S) ESPERADOS
Exploración	✓ Definición del equipo de Trabajo. ✓ Historias de Usuario.
Planificación	✓ Plan de Entregas. ✓ Velocidad del proyecto. ✓ Reuniones.
Iteraciones	✓ Plan de Iteraciones.

	✓ Metáfora del Sistema. ✓ Tarjetas CRC. ✓ Soluciones Puntuales.
Producción	✓ Diseño Lógico. ✓ Diseño Físico. ✓ Diseño Simple. ✓ Disponibilidad del Cliente. ✓ Programación por Parejas. ✓ Integración. ✓ Iteraciones. ✓ Pruebas.
Mantenimiento	✓ Reciclaje de Código. ✓ Estándares de Código. ✓ Rotaciones.
Muerte del Proyecto	✓ Código común. ✓ Conclusión de tiempos estimados.

Fuente: [Elaboración propia]

3.2.FASE DE EXPLORACION.

En esta fase se tratará de establecer los requerimientos del cliente y se definirá el equipo del proyecto que desarrollara el sistema. Al mismo tiempo los programadores se familiarizarán con las herramientas con las que se cuenta.

3.2.1. Definición del equipo de Trabajo.

El proyecto se lo realizará en los predios de FARMEDICAL SRL, lo que cumple con una de las prácticas de XP "Equipo en el mismo lugar."

Tabla 3.2: Definición del equipo de trabajo

DESCRIPCION	RESPONSABLE	ROL
Jefe de Proyecto	Lic.Alberth Heredia	Responsable del proyecto
Coordinador de Actividades	Sr. Raul Vargas Univ. Daniel Calliconde	Coordinar reuniones entre el cliente y el desarrollador.
Cliente	Sr. German Pacheco	Jefe de Marketing
Equipo de Desarrollo	Univ. Daniel Calliconde	Coordinar el desarrollo de las diferentes fases con las que contara el desarrollo e implementación del sistema.

Fuente: [Elaboración propia]

3.2.2. Historias de Usuario.

A continuación se detallan las historias de usuario que se obtuvieron de la plática directa cliente. De manera que pudimos plasmar las necesidades del mismo.

Tabla 3.3: Historia de Usuario No. 1

HISTORIA DE USUARIO	
Número:1	**Nombre Historia de Usuario:**Registro de médicos
Usuario: Jefe de Marketing	**Iteración Asignada:**
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**1
Riesgo en Desarrollo: Bajo (Alto / Medio / Bajo)	**Puntos Reales:**1

Descripción: Se necesita registrar los nuevos médicos relacionados a las promociones que se realicen en determinado tiempo, junto con sus respectivas características.
Observaciones: Un médico se registra solo una vez

Fuente: [Elaboración propia]

HU1: Cuando un visitador médico realiza un nuevo contacto, o sea un nuevo médico, el mismo debe registrar el nombre, dirección del consultorio, especialidad, ciudad, referencias de contactos y su categoría. De esta forma se le hará mucho más fácil la planificación de visitas al Médico introducido.

Tabla 3.4: Historia de Usuario No. 2

HISTORIA DE USUARIO	
Número:2	**Nombre Historia de Usuario:**Registro de Visitadores Médicos
Usuario: Jefe de Marketing	**Iteración Asignada:**
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**1
Riesgo en Desarrollo: Bajo (Alto / Medio / Bajo)	**Puntos Reales:**1
Descripción: Se necesita ingresar a nuevos visitadores médicos con sus respectivas características.	
Observaciones: Un nuevo visitador médico solo se registra una única vez.	

Fuente: [Elaboración propia]

HU2: Cada cierto tiempo, y gracias al constante crecimiento de la empresa el Supervisor de visitadores médicos, necesita realizar la incorporación de nuevos visitadores médicos. Los cuales cuentan con ciertas características específicas de un visitador médico. Estas pueden ser nombre, dirección, celular corto[14], especialidad y otros. Un visitador en específico tendrá una relación directa con los doctores y productos que la empresa posee directamente a través de la o las especialidades con las que el mismo cuenta. También se le deberá registrar dentro de la regional correspondiente a la ciudad donde realiza su trabajo.

Tabla 3.5: Historia de Usuario No. 3

HISTORIA DE USUARIO	
Número:3	**Nombre Historia de Usuario:**Registro de especialidades
Usuario: Jefe de Marketing	**Iteración Asignada:**
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**1
Riesgo en Desarrollo: Baja (Alto / Medio / Bajo)	**Puntos Reales:**1
Descripción: Se precisa registrar las especialidades de los productos que la empresa posee.	
Observaciones: Las especialidades pueden ser especialidades en si o subespecialidades.	

Fuente: [Elaboración propia]

HU3: Como ya vimos en el Capítulo I, la empresa FARMEDICAL se dedica a la importación directa de diferentes productos farmacéuticos del exterior. Estos productos pueden ser utilizados para una o más especialidades de la medicina. Es por eso que es necesario registrar las especialidades con las que la empresa cuenta asociadas a sus productos.

[14] Celular corto: Es la asignación de un número de celular empresarial, que la empresa proporciona al visitador.

Tabla 3.6: Historia de Usuario No. 4

HISTORIA DE USUARIO	
Número:4	**Nombre Historia de Usuario:**Asignación de Especialidades
Usuario: Jefe de Marketing	**Iteración Asignada:**
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**1
Riesgo en Desarrollo: Bajo (Alto / Medio / Bajo)	**Puntos Reales:**1
Descripción: Asignar una o más especialidades a un determinado visitador médico de acuerdo a la especialización con el mismo presenta.	
Observaciones: Se puede tener como máximo diez especialidades por visitador médico.	

Fuente: [Elaboración propia]

HU4: Todos los visitadores antes de empezar a realizar sus visitas médicas pasan por una etapa de especialización en una o varias áreas específicas de la medicina, por ejemplo: anestesiología, pediatría, UTI, etc. Un visitador médico puede tener desde una como mínimo y hasta diez especialidades como máximo. Estas especialidades también van ligadas a los productos con que cuenta FARMEDICAL.

Tabla 3.7: Historia de Usuario No. 5

HISTORIA DE USUARIO	
Número:5	**Nombre Historia de Usuario:**Ruteo semanal
Usuario: Jefe de Marketing	**Iteración Asignada:**

Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados: 3**
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	**Puntos Reales:**3
Descripción: Asignar un horario en un día y semana específico para el ruteo semanal y/o mensual de un visitador médico.	
Observaciones: Los médicos deberán estar asociados al visitador médico por medio de sus especialidades.	

Fuente: [Elaboración propia]

HU5: Los visitadores médicos planifican sus visitas en unas hojas de ruteo semanal y diario. En estas hojas ellos asignan una visita específica en un día y hora ya establecido. Un visitador médico puede llegar a hacer un número de 20 visitas médicas diarias en el mejor de los casos. También cada médico a visitar cuenta con una categoría dentro de la empresa, esta categoría define el número de veces que el visitador asociado a su especialidad debe visitarlo. Uno de los objetivos de estas hojas de ruteo es el de no olvidar ni exceder el número de visitas asignado a un doctor de acuerdo a su categoría. Otro objetivo es el de organizar el tiempo de un visitador.

Tabla 3.8: Historia de Usuario No. 6

		HISTORIA DE USUARIO
Número:6	**Nombre Historia de Usuario:**Fichas de Ruta	
Usuario: Jefe de Marketing		**Iteración Asignada:**
Prioridad en Negocio: Alta (Alta / Media / Baja)		**Puntos Estimados:**3
Riesgo en Desarrollo: Medio		**Puntos Reales:**3

(Alto / Medio / Bajo)	
Descripción: Se necesita tener los comprobantes (fichas) de la ruta planificada por un visitador médico.	
Observaciones: Las fichas deberán ser diarias.	

Fuente: [Elaboración propia]

HU6: Una vez que un visitador médico realiza la planificación de sus visitas médicas, este cuenta con unas fichas de control de visita. Estas fichas deberán ser firmadas por el doctor visitado a la culminación de la misma. Estas fichas sirven para el control del trabajo de cada visitador médico y son cuantificadas por el supervisor medico a la culminación de una semana laboral. Esto nos sirve para verificar a que porcentajes de visitas médicas cumplidas se eleva cada visitador médico y también el correspondiente seguimiento del mismo.

Las historias de usuario presentadas hasta el momento serán la base y guía para el proyecto en desarrollo.

3.3. FASE DE PLANIFICACION.

En la fase de planificación se clasifican las historias de usuario, se dan prioridades, se estiman tiempos y se publica un Plan de Publicaciones. En otras palabras en que Iteración podría una HU[15] ser implementada.

Algunas de las HU son más importantes que otras, en esta fase se determina la prioridad de cada HU.

3.3.1. Plan de entregas.

En el plan de entregas daremos a conocer el número de iteración en la cual una o varias historias de usuario serán producidas. De esta manera el plan de entregas será como sigue:

[15] HU: Abreviación que se le da a "Historia de Usuario"

Tabla 3.9: Plan de entregas

HISTORIA DE USUARIO	ITERACION ASIGNADA	OBJETIVO
HU1	IT1	Crear la interfaz o interfaces para la inserción de médicos, con sus respectivas características y validaciones.
HU2	IT2	Crear la interfaz o interfaces para la inserción de Visitadores Médicos, juntamente con sus respectivas características y validaciones.
HU3	IT3	Crear la interfaz o interfaces para la inserción de Especialidades Médicas con las que FARMEDICAL trabaja.
HU4	IT4	Crear la interfaz para o interfaces para la correcta asignación de especialidad a un visitador ya registrado.
HU5	IT5	Crear la interfaz o interfaces para la planificación semanal de visitas médicas un visitador médico que cuente con al menos una especialidad.
HU6	IT6	Crear la interfaz o interfaces para la correspondiente verificación de las fichas de control de las visitas ya programadas de un visitador médico.

Fuente: [Elaboración propia]

3.3.1. Velocidad del proyecto.

Como pudimos observar en la Tabla 3.x contaremos con un total de 6 iteraciones, es así que dividiremos el tiempo que tenemos para desarrollar el sistema, aproximadamente 4 meses, en tiempos de iteraciones relativamente iguales. Esto a fin de no acelerar ni ralentizar el proyecto y que el mismo se desarrolle de una manera prácticamente continua y sin mucha presión.

Por otro lado también podemos promediar la velocidad del proyecto utilizando:

Velocidad del Proyecto = Número de Historias de Usuario / Número de Iteraciones

Velocidad del Proyecto = 6 / 6

Velocidad del Proyecto = 1

Este indicador se lo puede interpretar de la siguiente manera: En cada semana que se trabaje en el proyecto se avanzará en promedio 1 Historia de Usuario.

3.3.2. Reuniones.

Las reuniones con todo el equipo de trabajo se las realizará una vez a la semana, más específicamente cada viernes, esto con el fin de mostrar avances en cuanto al sistema y aplicar las cuatro actividades propias de XP: codificar, probar, escuchar, diseñar. Esto al final se vuelve en una retroalimentación directa hacia el desarrollo del sistema. De esta manera aseguraremos que el cliente se sienta parte del proceso del proyecto, así como su entera satisfacción.

3.4. FASE DE ITERACIONES.

La fase de iteraciones es una de las fases más importantes dentro del desarrollo de software bajo la metodología XP. Ya que en esta definiremos la planificación de las iteraciones, la metáfora del sistema, el llenado de las tarjetas CRC[16] y el llenado de las tarjetas de ingeniería[17].

3.4.1. Planificación de Iteraciones.

La planificación de las iteraciones ayudará tanto al cliente como al equipo de programación para saber de una forma general la duración de cada iteración y de sus respectivas presentaciones. Ayudará al equipo de trabajo a una mejor administración de su tiempo.

De acuerdo a las distintas historias de usuario y las estimaciones que hicimos en la fase de planificación tenemos la siguiente tabla de planificación de iteraciones:

---

[16] CRC: Abreviación de Class - Responsibilities – Collaborators.

[17] Tarjetas de Ingeniería: también conocidas como TaskCards.

	AGOSTO				SEPTIEMBRE				OCTUBRE				NOVIEMBRE			
Iteración	1°Semana	2°Semana	3°Semana	4°Semana	1°Semana	2°Semana	3°Semana	4°Semana	1°Semana	2°Semana	3°Semana	4°Semana	1°Semana	2°Semana	3°Semana	4°Semana
1																
2																
3																
4																
5																
6																

Figura 3.1: Planificación de Iteraciones

Fuente: [Elaboración propia]

3.4.2. Metáfora del Sistema.

La metáfora del sistema nos ayudara a la fácil comunicación entre el cliente y el equipo de programación. En si el objetivo de la misma es el de poder llegar a obtener un lenguaje común para ambas partes.

Tabla 3.10: Metáfora del Sistema

METAFORA DEL PROYECTO
"Toda visita médica gira alrededor de tres componentes principales: el profesional médico, el visitador médico y el producto farmacéutico que se está promocionando.

Estos tres componentes están ligados entre sí por medio de la especialidad para la cual fueron capacitados o bien fueron creados".

Fuente: [Elaboración propia]

A partir de la metáfora del sistema y del intercambio verbal de ideas con el cliente se llega a generar un glosario de términos. En este definiremos términos que ayudara a ambas partes a una comunicación más sencilla y fluida.

Tabla 3.11: Glosario de términos

GLOSARIO	
Término	**Descripción**
Visita médica	Pequeña reunión programada por un visitador médico a fin de promocionar un cierto producto.
Profesional médico	Es el doctor a cual se le hace la visita médica.
Visitador médico	Es el empleado de la empresa que es el responsable de realizar las visitas médicas programadas.
Producto farmacéutico	Es el producto que la empresa importa directamente de los fabricantes en el exterior. Por ejemplo: analgésicos, antibióticos, etc.
La Empresa	Es el denominativo que también le daremos a FARMEDICAL SRL, donde se desarrolla el presente proyecto.
Muestra médica	Muestra de regalo que un visitador médico obsequia al doctor visitado.
Ruta	Denominativo que se le da a la planificación diaria de visitas de un visitador médico.
Target médico	Se denomina así al conjunto de médicos a los cuales se quiere llegar con las visitas médicas dentro de la empresa.

Fuente: [Elaboración propia]

3.4.3. Tarjetas CRC.

Las tarjetas CRC tienen la finalidad de facilitar la implementación de las clases que serán utilizadas en el desarrollo del sistema. También nos ayudan a identificar que clases interactúan directa e indirectamente dentro del sistema. En el presente sistema se realizaron las siguientes tarjetas CRC.

Tabla 3.12: Tarjeta CRC Persona

PERSONA	
Realizar la correcta administración de las nuevas personas, ya sean visitadores médicos o visitadores.	• Medico • Visitador

Fuente: [Elaboración propia]

La clase persona es la primera clase que implementaremos ya que de ella surgirán las clases Medico y Visitador.

Tabla 3.13: Tarjeta CRC Medico

MEDICO	
Realizar la correcta administración de los nuevos médicos, validando sus respectivas características.	• Persona

Fuente: [Elaboración propia]

Es la segunda clase a implementar, ya que la empresa no tendría razón de ser sin tener doctores a los cuales poder visitar.

Tabla 3.14: Tarjeta CRC Visitador

VISITADOR	
Realizar la correcta administración de los nuevos visitadores médicos con los que cuenta el sistema.	• Persona

Fuente: [Elaboración propia]

El visitador médico es parte fundamental de la empresa, pues es el que da la cara por la misma, es por eso quelos datos a almacenar pertenecientes a un visitador médico deberán ser verídicos y sin errores. Los visitadores médicos en realidad son empleados directos de la empresa y dependen del área de RRHH[18]. Pero por fines de provecho al sistema los incluiremos en nuestra BD.

Tabla 3.15: Tarjeta CRC Especialidad

ESPECIALIDAD	
Administrar las diferentes especialidades con las que trabaja la empresa.	• Medico • Visitador

Fuente: [Elaboración propia]

[18] RRHH: Se refiere al área de Recursos Humanos con la que cuenta toda empresa.

La especialidad es el enlace principal entre visitadores y médicos, sin esta clase no se podría llegar a obtener la relación entre los mismos.

Tabla 3.16: Tarjeta CRC Visita

VISITA	
Administrar las visitas médicas de forma ordenada, impidiendo su duplicidad y/o su pérdida.	• Visitador • Medico • Especialidad

Fuente: [Elaboración propia]

La visita médica es el producto de una extensa planificación de parte del visitador médico. Es el punto de inflexión entre el éxito o fracaso de la empresa.

Tabla 3.17: Tarjeta CRC Regional

REGIONAL	
Validar la existencia de una regional o en su defecto la inserción y/o modificación de la misma.	• Medico • Visitador

Fuente: [Elaboración propia]

Una regional es el lugar donde se realiza netamente el aspecto operacional de la empresa. Es la fortaleza de los visitadores médicos. Una regional puede administrar las visitas de más de una ciudad.

Tabla 3.18: Tarjeta CRC Ciudad

CIUDAD	
Validar la existencia de una ciudad en específico tomando en cuenta su regional asignada.	• Medico • Visitador • Pais

Fuente: [Elaboración propia]

Es la región delimitada de trabajo de los visitadores médicos, estos se mueven exclusivamente dentro la misma, realizando las visitas diarias que ya tienen programadas. Una ciudad depende de una regional.

Tabla 3.19: Tarjeta CRC País

PAIS	
Verificar que las ciudades y regionales estén dentro del país que les corresponde.	• Ciudad

Fuente: [Elaboración propia]

Ya que la empresa FARMEDICAL está en constante crecimiento se vio por conveniente agregar esta clase, para la mejor organización tanto de los doctores y visitadores médicos que trabajan en un País dado. Como enunciamos en el capítulo I "Historia de la empresa", actualmente la empresa hizo la apertura de una nueva filial en la ciudad de Lima, en el país vecino del Perú.

Tabla 3.20: Tarjeta CRC Reporte

REPORTE	
Generar reportes en orden cronológico haciendo un seguimiento de los mismos.	• Visita • Medico • Visitador • Especialidad

Fuente: [Elaboración propia]

El reporte es el fin tangible de todo sistema. Gracias a él se pueden tomar decisiones que beneficien a la propia empresa. Por medio de los reportes, el cliente, en nuestro caso la empresa puede tomar decisiones a nivel gerencial que sean de provecho para FARMEDICAL SRL.

3.4.4. Soluciones Puntuales.

Para poder dar soluciones puntuales a las diferentes historias de usuario, hacemos uso de otro de los artefactos que XP nos proporciona. En este caso estamos hablando de lasTaskCards o también conocidas como Tareas de Ingeniería.

Para obtener las Historias de Ingeniería podemos o no dividir las Historias de Usuario de acorde a su complejidad. De esta manera simplificaremos el trabajo de programación.

A continuación las TaskCards obtenidas para cada Historia de Usuario.

Tabla 3.21: Tarea de Ingeniería No. 1

	TAREA DE INGENIERÍA
Número:1	**Historia de Usuario (Nro. Y Nombre):**1. Registro de médicos
Nombre Tarea: Registro de médicos	

Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**1
Fecha Inicio:1/8/11	**Fecha Fin:** 12/8/11
Programador Responsable: Daniel Calliconde	
Descripción: Realizar una interfaz de usuario por medio de la cual se pueda ingresar un nuevo médico.	

Fuente: [Elaboración Propia]

En la TI1[19] nos dan la tarea de elaborar una o más interfaces necesarias para el correcto ingreso de médicos. Cabe recalcar que para evitar futuros errores en la manipulación de datos, todos los datos ingresados deberán estar en mayúsculas y validados de acuerdo al tipo al que pertenezcan.

Tabla 3.22: Tarea de Ingeniería No. 2

	TAREA DE INGENIERÍA
Número:2	**Historia de Usuario (Nro. Y Nombre):**2. Reg. de Visitadores Médicos
Nombre Tarea: Interfaz registro visitadores médicos	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**1
Fecha Inicio: 15/8/11	**Fecha Fin:** 26/8/11
Programador Responsable: Daniel Calliconde	
Descripción: Realizar una interfaz de usuario por medio de la cual el supervisor de área pueda ingresar a	

[19] TI: Abreviación que se utilizará para representar "Tarea de Ingeniería".

los nuevos visitadores médicos.

Fuente: [Elaboración Propia]

En la TI2 nos dan la tarea de elaborar algo similar a la TC1 pero esta vez para el ingreso de nuevos visitadores médicos. En esta interfaz que será algo así como un formulario todos los campos deberán ser obligatorios ya que asumimos el conocimiento de todas las características de un visitador. Este proceso puede ser realizado solo por el supervisor.

Tabla 3.23: Tarea de Ingeniería No. 3

TAREA DE INGENIERÍA	
Número:3	**Historia de Usuario (Nro. Y Nombre):**3. Registro de Especialidades
Nombre Tarea: Interfaz registro especialidades	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**1
Fecha Inicio: 29/8/11	**Fecha Fin:** 9/9/11
Programador Responsable: Daniel Calliconde	
Descripción: Realizar una interfaz de usuario por medio de la cual el supervisor del área pueda introducir las distintas especialidades que están asociadas a los productos de la empresa.	

Fuente: [Elaboración Propia]

En la TI3 nos dan la tarea de elaborar una interfaz por medio de la cual el supervisor pueda administrar las distintas especialidades con las que trabaja la empresa. Como ya dijimos con anterioridad, la empresa cuenta con distintas especialidades de acuerdo a los productos farmacéuticos con los que la misma cuenta. Estas especialidades deben ser administradas de una forma responsable y con mucho tino. Es por eso que a esta interfaz solo podrá acceder el supervisor de visitadores médicos, mas no así los propios visitadores médicos que en algún momento estén introduciendo un nuevo contacto médico.

Tabla 3.24: Tarea de Ingeniería No. 4

TAREA DE INGENIERÍA	
Número:4	**Historia de Usuario (Nro. Y Nombre):**4. Asig. de Especialidades
Nombre Tarea: Interfaz asignación de especialidades	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**1
Fecha Inicio: 12/9/11	**Fecha Fin:** 23/9/11
Programador Responsable: Daniel Calliconde	
Descripción: Realizar una interfaz por medio de la cual se pueda asignar especialidades a los visitadores ya registrados.	

Fuente: [Elaboración Propia]

En la TI4 nos dan la tarea de elaborar una interfaz por medio de la cual solo el supervisor pueda agregar o eliminar una cierta especialidad asociada a un visitador médico ya registrado.

Tabla 3.25: Tarea de Ingeniería No. 5

TAREA DE INGENIERÍA	
Número:5	**Historia de Usuario (Nro. Y Nombre):**5. Ruteo semanal
Nombre Tarea: Interfaz Ruteo semanal	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**3
Fecha Inicio: 26/9/11	**Fecha Fin:** 14/10/11
Programador Responsable: Daniel Calliconde	

Descripción:
Realizar una interfaz, por medio de la cual los visitadores médicos puedan planificar sus rutas de visitas médicas.

Fuente: [Elaboración Propia]

En la TI5 nos dan la tarea de elaborar, a mi parecer la interfaz más importante, pero a la misma vez la más dificultosa del sistema, una interfaz por medio de la cual tanto un supervisor o un visitador médico pueda planificar las visitas médicas a realizar en las próximas semanas. Esta deberá desplegar la lista de los médicos a los cuales el visitador médico está asociado, así como la frecuencia de visita de los mismos. La frecuencia de visitas está asociada a la categoría con la que cuenta un médico.

Tabla 3.26: Tarea de Ingeniería No. 6

TAREA DE INGENIERÍA	
Número:6	**Historia de Usuario (Nro. Y Nombre):**6. Fichas de Ruta
Nombre Tarea: Informes de Ruta	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**3
Fecha Inicio: 17/10/11	**Fecha Fin:** 28/10/11
Programador Responsable: Daniel Calliconde	
Descripción: Realizar la impresión de fichas de ruta que vayan acorde a la ruta planificada diaria de un visitar medico	

Fuente: [Elaboración Propia]

Por último, la TI6, después de haber realizado planificación correspondiente a un visitador médico, se deberá contar con una opción por medio de la cual el visitador tengo las visitas programadas a mano y así poder hacerlas firmar por el medico visitado.

Tabla 3.27: Tarea de Ingeniería No. 7

	TAREA DE INGENIERÍA
Número:7	**Historia de Usuario (Nro. Y Nombre): 5**
Nombre Tarea: Filtrado de médicos de acuerdo a la especialidad de un Visitador	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**1
Fecha Inicio:31/10/11	**Fecha Fin:** 11/11/11
Programador Responsable: Daniel Calliconde	
Descripción: Realizar una pequeña interfaz que filtre a los médicos de acuerdo a la o las especialidades de un visitador médico.	

Fuente: [Elaboración Propia]

La TI7 es una extensión la HU5, ya que para poder hacer el ruteo semanal correspondiente debemos previamente tener conocimiento de que médicos son los que corresponde a un Visitador Médico X. Esta pequeña interfaz o tabla deberá mostrar como campos principalmente el nombre completo del profesional médico, la especialidad con la que cuenta, la categoría a la que pertenece dentro de la empresa y por último el número de veces que un visitador médico debe visitarlo dentro de un ciclo.

Teniendo ya a mano las tareas de ingeniería necesarias es que podemos pasar a la fase de producción.

3.5. FASE DE PRODUCCION.

En esta fase es donde se inicia la programación de las diferentes Tareas de Ingeniería y sus respectivas Clases de acuerdo a las tarjetas CRC. Para tal propósito presentaremos esta fase por cada iteración ya planificada. Pero antes para una mejor comprensión de los datos veremos el modelo lógico y el modelo físico del sistema.

3.5.1. Modelo Lógico.

Para un mejor entendimiento de como las distintas entidades que interactúan en nuestro sistema es que mostramos el siguiente diagrama Entidad – Relación.

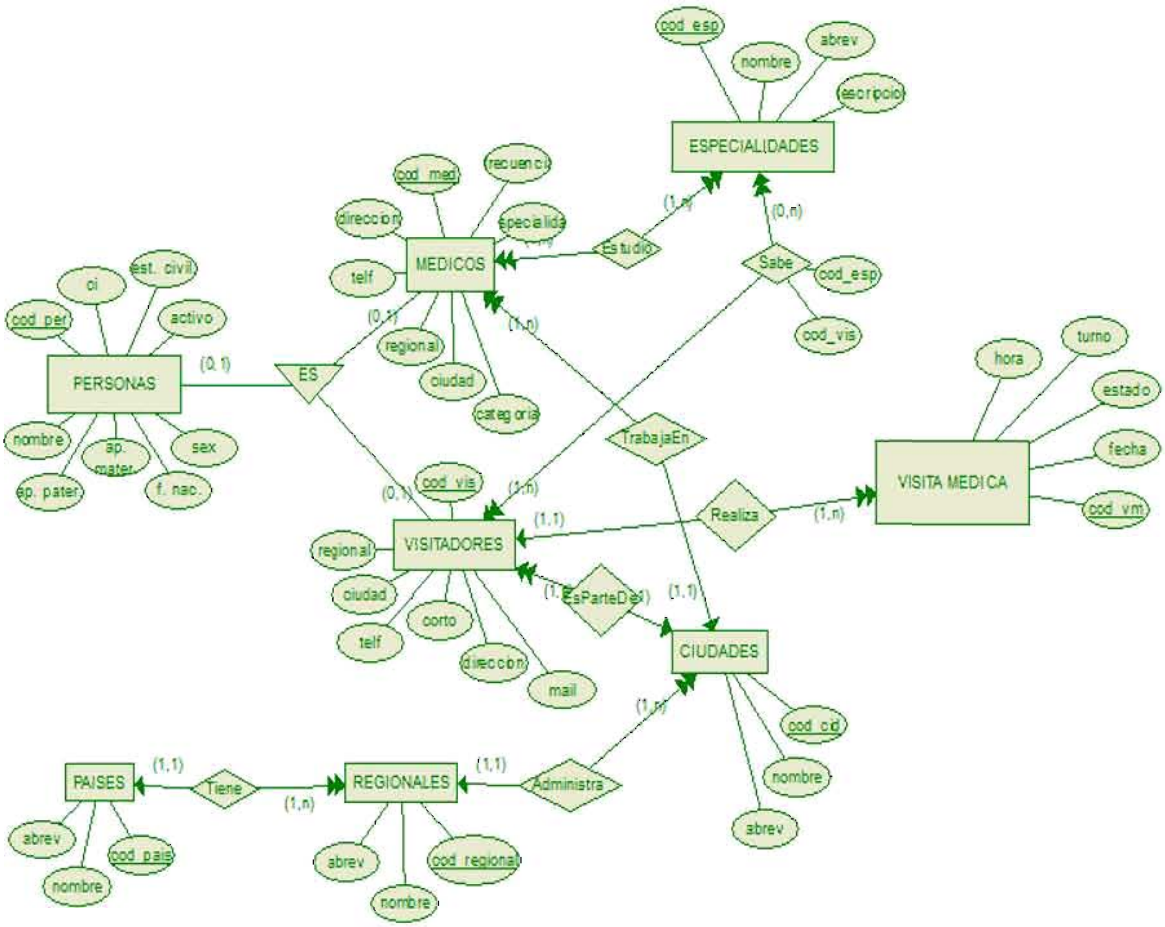

Figura 3.2: Modelo Lógico del Sistema

Fuente: [Elaboración Propia]

3.5.2. Modelo Físico.

El modelo Físico nos muestra la forma de cómo está estructurada nuestra base de datos en nuestro motor de base de datos. Esto nos ayudará tanto a la modificación y/o mantenimiento del mismo cuando sea necesario. No tomamos en cuenta el modelo lógico ya que el modelo físico lo contiene ya estructurado dentro del mismo.

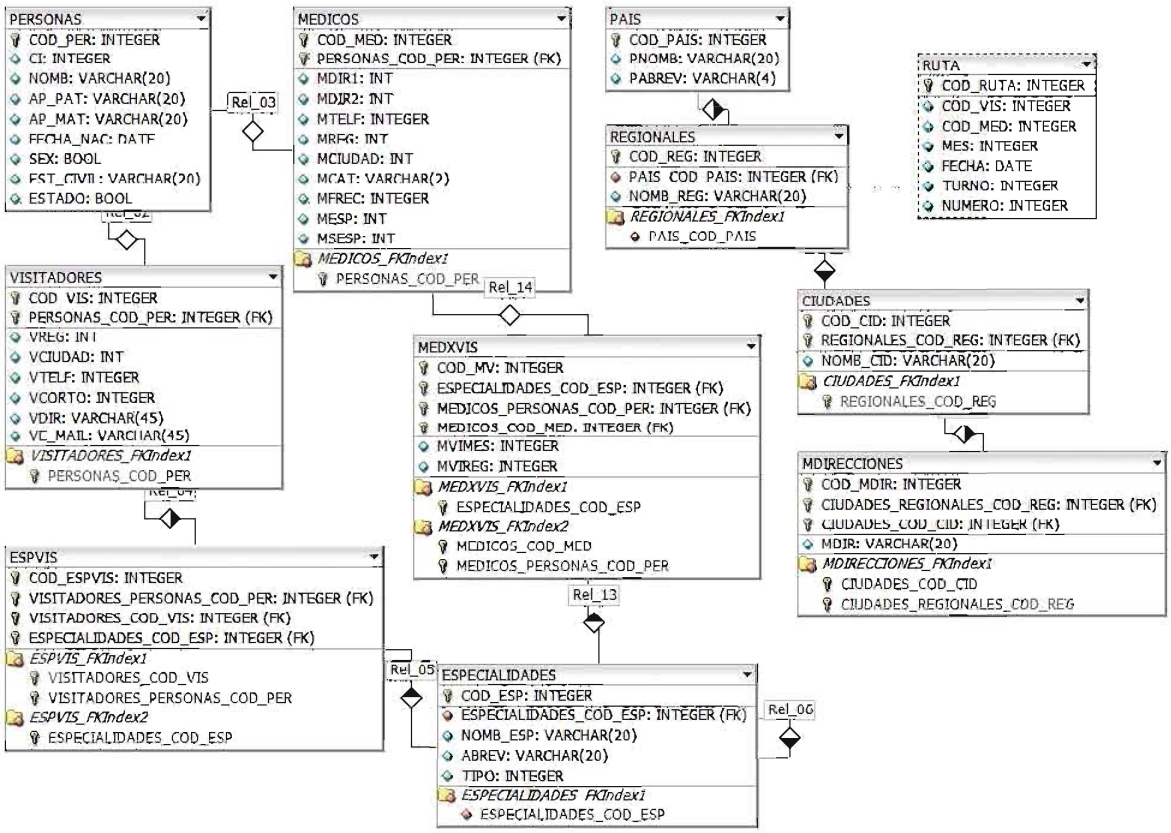

Figura 3.3: Modelo Físico del Sistema.

Fuente: [Elaboración propia]

3.5.2. Diseño Simple.

El diseño simple es una de las prioridades de XP en cuanto a la fase de producción. Y haciendo caso de esta práctica es que realizamos la programación de las diferentes interfaces necesarias para la conformación del sistema de una forma práctica y sencilla. Esta práctica se verá reflejada en cada una de las interfaces de las diferentes iteraciones con las que cuenta el sistema.

3.5.4. Disponibilidad del Cliente.

Gracias que el desarrollo del sistema se lo realizo en predios de la misma empresa. Es que se pudo contar con la presencia del cliente en todo momento. Esto también se verá reflejado en la producción de las diferentes iteraciones con las que cuenta el sistema.

3.5.4. Programación por Parejas.

Para este punto en específico, y que concierne a una de las prácticasmás importantes de XP, se vio por conveniente tomar como pareja de programación al directo cliente que es el jefe de marketing. Él nos asesorará en el diseño de las interfaces para hacer de estas más amigables y simples de utilizar. También nos ayudará a definir la validación de datos, ya que en el tiempo que viene trabajando en el área pudo ver algunas fallas dentro de ladocumentación físicacausadas por la mala introducción de datos, y esto es exactamente lo que queremos evitar.

También sabemos que la metodología XP asigna roles a cada uno de los integrantes del equipo de trabajo, en muchas ocasiones el número de integrantes es menor al número de roles que XP propone, por lo cual muchas veces un individuo puede tener más de un rol a la vez, sin que esto cause problemas dentro del desarrollo del sistema.

3.5.5. Integración.

En algunos momentos de la programación se ve por conveniente dividir las tareas que debemos realizar. Es así que por ese motivo en esos momentos se tiene la impresión de que se está yendo por caminos diferentes, y es un riesgo que se puede correr al trabajar de esa manera. Es por eso que para que no suceda eso se debe hacer una integración continua del código y de los módulos que se vayan desarrollando. De esta manera aseguraremos la coherencia entre ellos y que estos trabajen sin ningún problema con nuevos elementos que se les pueda ir añadiendo. La integración es uno de los aspectos fundamentales de la Programación Extrema.

Al mismo tiempo gracias a la integración continua que se tiene con el cliente y su entorno, el desarrollador puede llegar a estar muy inmerso dentro de las actividades de la empresa que se sentirá parte de la misma. Esto ayudará de una forma considerable al mejor desarrollo del proyecto.

3.5.6. Iteraciones.

Para un mejor y más fácil entendimiento que lo que se está desarrollando dividiremos la presentación por iteraciones.

3.5.6.1. Primera Iteración.

En la primera iteración en la cual está incluida la primera Historia de Usuario y que corresponde a la primera Tarea de Ingeniería se nos pidió poder realizar el correcto ingreso de médicos. Para lo cual se diseñaron y programaron las siguientes Interfaces.

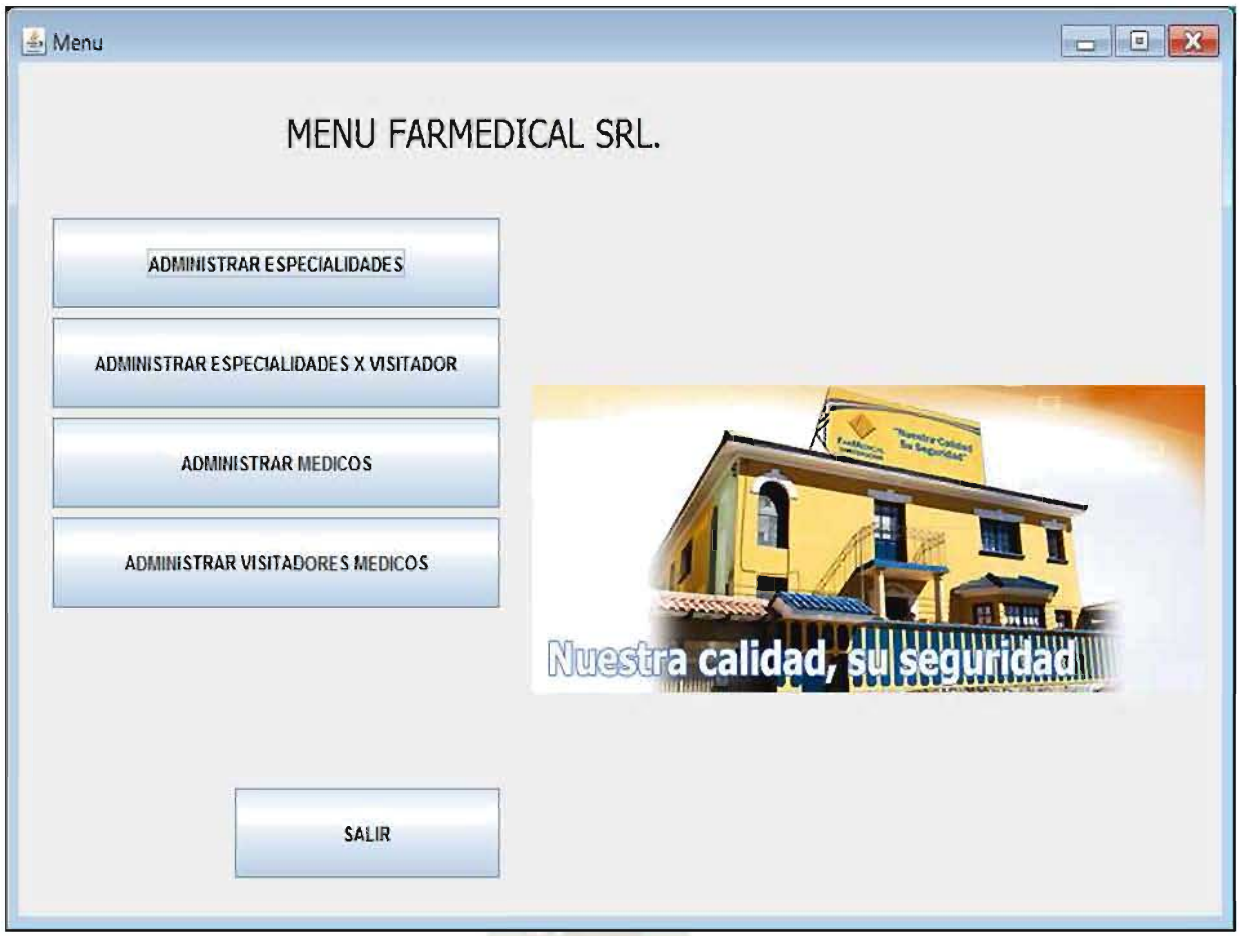

Figura 3.4: Interfaz de Usuario "Menú Inicio"

Fuente: [Elaboración propia]

En la Interfaz inicio podemos ver los diferentes módulos con los cuales contará inicialmente el sistema. A través de esta interfaz podremos ingresas a los diferentes procesos con los que se cuenta. En el caso de la primera iteración ingresar a "Administrar Médicos". Esto nos desplegará la siguiente interfaz:

Figura 3.5: Interfaz de Usuario "Ingreso Médicos"

Fuente: [Elaboración propia]

Una vez ingresado, se nos despliega la interfaz de Ingreso Médicos. Los campos que tienen el símbolo '*' son de carácter obligatorio, ya que estos son parte de la información básica necesaria para un profesional médico. También podemos ver que la mayoría de los campos no son editables, esto porque las opciones que estas presentan se la obtendrá directamente de un listado pre-programado o bien directamente de la base de datos, esto a fin de reducir el error de transcripción que se pueda presentar al momento del ingreso de los datos. En esta interfaz todos los datos son validados con respecto a su tipo, por ejemplo: el campo de Fecha Nac. no admitirá fechas no validas, el campo de Telf/Cel no admitirá letras dentro de su validación. Cuando no se cuenta con la dirección que se está buscando esta se la podrá agregar haciendo click en el botón Nuevo que aparece al lado del campo, esto nos desplegara una pequeña ventana.Ocurrirá lo mismo con los campos de Dirección2, Regional, Ciudad y Subespecialidad. Se cuenta con el campo de Dirección2 ya que la mayoría de los doctores pueden trabajar tanto en un hospital o clínica y al mismo tiempo contar con un consultorio privado, es por eso que se cuentan con dos campos para las direcciones. Se podrán también agregar nuevas Regionales, Ciudades y Subespecialidades.

Figura 3.6: Interfaz de Usuario "Nueva Dirección"

Fuente: [Elaboración propia]

En la ventana "Nueva Dirección" ingresamos la dirección a añadirse a nuestra base de datos. Esta será validada, se verificara que no exista duplicidad y se la transformara a mayúsculas automáticamente antes de su inserción a la BD. Lo mismo ocurrirá para las interfaces Nueva Ciudad y Nueva Regional.

Figura 3.7: Interfaz de Usuario "Nueva Ciudad"

Fuente: [Elaboración propia]

La interfaz "Nueva Ciudad" nos brinda la facilidad de poder agregar una nueva ciudad a nuestro banco de ciudades. Esta puede ser llamada desde cualquier interfaz ya que es independiente.

Figura 3.8: Interfaz de Usuario "Nueva Regional"

Fuente: [Elaboración propia]

La interfaz "Nueva Regional" permite al usuario agregar una nueva regional en cualquier momento de uso del sistema siempre y cuando el usuario tenga nivel de administrador. Tanto las ventanas Nueva Ciudad y Nueva Regional hacen la respectiva validación de los datos introducidos antes de mandarlos a la BD.

Figura 3.9: Interfaz de Usuario "Busca Médico"

Fuente: [Elaboración propia]

La interfaz "Busca Medico" nos ayuda a buscar de una forma muy sencilla la existencia de un médico en específico. Solo debemos ingresar o el nombre o cualquiera de los apellidos del doctor a ser buscado. Los resultados se despliegan en el campo tabla de la ventana de forma automática, sin necesidad oprimir ningún otro botón. Así tener la información necesaria de un profesional médico o simplemente saber de su existencia.

Las interfaces: Nueva Ciudad, Nueva Dirección, Nueva Regional surgieron de la normalización de la Base de Datos y son reutilizables en cualquier etapa de operación del sistema. De esta manera también estamos cumpliendo con una de las prácticas de XP "Código Reutilizable".

3.5.6.2. Segunda Iteración.

Una vez habiendo ingresado por el menú inicio y seleccionando el botón de Administración de Visitadores Médicos, se desplegará la interfaz Registro de Visitadores Médicos. La cual detallaremos a continuación.

Figura 3.10: Interfaz de Usuario "Registro de Visitadores Médicos"

Fuente: [Elaboración Propia]

Esta interfaz es muy similar a la de "Ingreso Médicos" con ciertos cambios en algunos de los campos. Por ejemplo podemos observar que no se ve por ningún lado el símbolo '*', esto sucede porque en esta ventana todos los campos son de carácter obligatorio, se supone que un visitador médico es ya un empleado de la empresa y se cuentan con todos sus datos.

Figura 3.11: Interfaz de Usuario "Busca Visitador"

Fuente: [Elaboración Propia]

La ventana Busca Visitador nos facilita la búsqueda de un Visitador, de igual manera como se realizaba la búsqueda en la ventana Busca Médico se la realiza aquí.

3.5.6.3. Tercera Iteración.

En la Tercera Iteración se generó la interfaz "Especialidades", la cual observamos en la Figura 3.11.

Esta interfaz de un uso muy sencillo ya que solo se debe empezar a escribir las primeras letras de la especialidad que queremos agregar, al mismo tiempo en el campo tabla de la ventana se empezaran a desplegar las especialidades ya existentes en la base de datos. Esto nos ayudará a ver en forma sencilla y directa la existencia de dicha especialidad.

Figura 3.12: Interfaz de Usuario "Especialidades"

Fuente: [Elaboración propia]

3.5.6.4. Cuarta Iteración.

En la cuarta iteración se generó la siguiente interfaz "Visitadores X Especialidad". A continuación la interfaz generada.

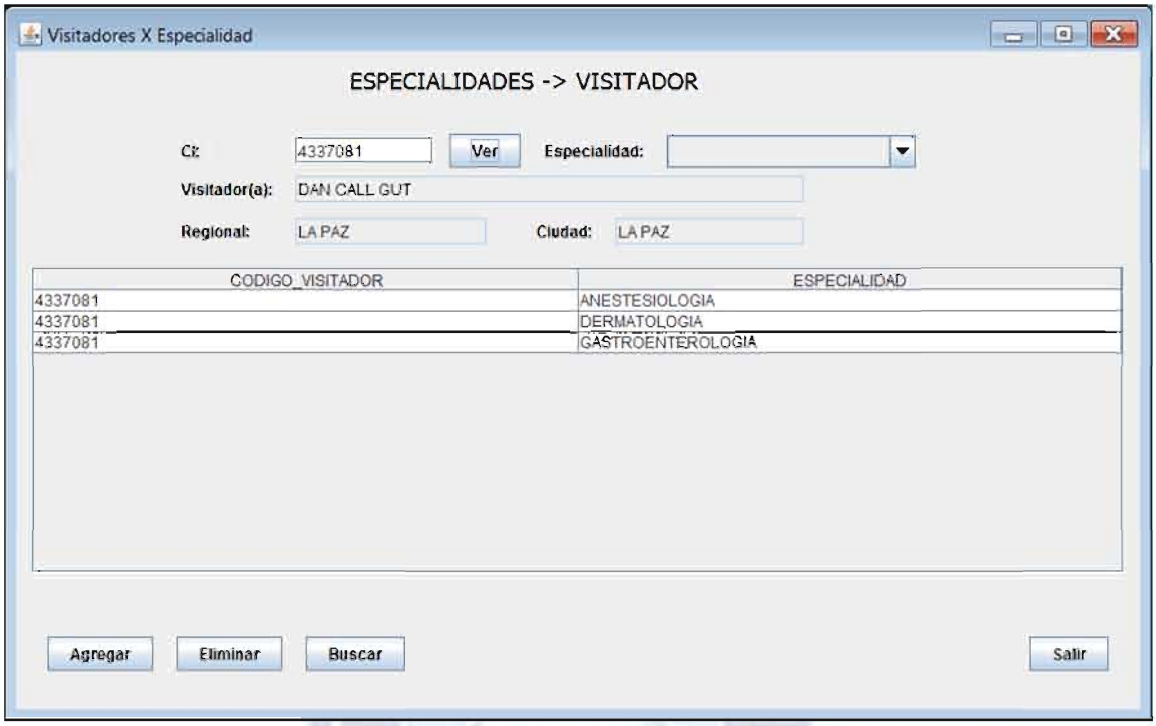

Figura 3.13: Interfaz de Usuario "Visitadores X Especialidad"

Fuente: [Elaboración propia]

En esta ventana el supervisor de visitadores médicos una vez verificadas las especialidad con las que un visitador cuenta pasa a la asociación de esas especialidades al mismo. Solo se debe tener conocimiento del CI del visitador, en caso de desconocer el CI se lo puede buscar por medio del botón Buscar, el cual ya explicamos con anterioridad. El uso de esta ventana es muy sencillo pero fundamental para los siguientes procesos.

3.5.6.5. Quinta Iteración.

En la quinta iteración se generó la interfaz "Ruteo". En esta interfaz un visitador médico puede programar la ruta que seguirá en la semana.

Esta ventana tiene principalmente dos secciones. Una donde se pueden observar a la lista de médicos a los cuales el visitador médico deberá de visitar en un ciclo[20] dado. En esta

[20] Ciclo: Periodo de tiempo igual a un mes utilizado por los visitados médicos para evaluar su rendimiento.

parte también se pueden observar el número de visitas que se debe realizar al médico. Este número reducirá en uno cada vez que un visitador médico acomode al mismo dentro del ruteo diario. En las casillas de ruteo se colocará únicamente el nombre completo del doctor. Estos nombres serán extraídos directamente de la base de datos, esto para evitar errores en la introducción de sus nombres.

3.5.6.6. Sexta Iteración.

En la sexta iteración se pasa a la realización de las fichas de control en formas de reportes, estas serán impresas para luego ser repartidas a sus respectivos visitadores médicos y estas deberán ser firmadas por el médico visitado. Esta es la constancia de que un visitador médico está cumpliendo con su trabajo. La impresión de estas fichas de control ahorra mucho tiempo al visitador médico, ya que este la realizaba con anterioridad de forma manual, esto implicaba pérdida de tiempo y errores humanos por apuros y/o presión.

3.5.7. Pruebas.

Para esta fase haremos uso de otro artefacto de XP. Hablamos de las tarjetas de evaluación. Se realizaron la misma cantidad de tarjetas de evaluación como las historias de usuario que obtuvimos. Estas se las entregó al cliente y a tres visitadores para que puedan realizar las pruebas respectivas y el llenado de las mismas.

Tabla 3.28: Caso de Prueba HU1

	Caso de Prueba de Aceptación
Código: 01	**Historia de Usuario (Nro. y Nombre):** 1
Nombre: Registro de Médicos	
Descripción: Realizar la prueba de aceptación de la HU1	
Condiciones de Ejecución: Servidor PostgreSQL instalado, ejecutable Vimed compilado	
Entrada / Pasos de ejecución: Primero el usuario debe autenticarse en el sistema, debe ingresar al módulo de administración de médicos, debe hacer el llenado de todos los campos obligatorios del formulario y por último debe guardar el trabajo realizado.	
Resultado Esperado: El sistema valida todos los datos ingresados de acuerdo a su tipo, esto reduce los errores humanos ocasionados casualmente.	

Evaluación de Prueba: Aceptado

Fuente: [Elaboración propia]

En la tabla anterior podemos observar los pasos que el usuario final debe de seguir para ver el nivel de aceptación que tuvo el mismo al utilizar el sistema hasta este punto.

Tabla 3.29: Caso de Prueba HU2

	Caso de Prueba de Aceptación
Código: 02	**Historia de Usuario (Nro. y Nombre): 2**
Nombre: Registro de Visitadores Médicos	
Descripción: Evaluar las interfaces relacionadas a la HU2	
Condiciones de Ejecución: Idénticas al caso de prueba 01	
Entrada / Pasos de ejecución: El usuario debe autenticarse en el sistema, ingresa al módulo de administración de Visitadores Médicos, hace el correspondiente llenado de todos los campos ya que en esta interfaz todos los campos son obligatorios, guarda el trabajo realizado.	
Resultado Esperado: Los datos son almacenados en la BD, estos se los puede verificar con el buscador de Visitadores con el que cuenta el sistema.	
Evaluación de Prueba: Aceptado	

Fuente: [Elaboración propia]

La Tabla 3.29 nos indica el nivel de aceptación de la implementación de la HU2 desde el punto de vista de un usuario final.

Tabla 3.30: Caso de Prueba HU3

	Caso de Prueba de Aceptación
Código: 03	**Historia de Usuario (Nro. y Nombre): 3**
Nombre: Registro de Especialidades	

Descripción: Evaluar la interfaz relacionada con la HU3
Condiciones de Ejecución: El servidor PostgreSQL debe estar instalado, el IP de la terminal usuario debe estar registrado en el archivo de configuración de PostgreSQL, el ejecutable VIMED debe estar compilado, las especialidades básicas con las que cuenta la empresa deben estar pre-cargadas en la BD.
Entrada / Pasos de ejecución: El usuario se autentica en el sistema, ingresa al módulo de administración de especialidades, al teclear cualquier letra el sistema automáticamente muestra las especialidades que empiecen con la letra ingresada, en ese momento el supervisor puede decidir si la especialidad necesitada está registrada, sino procede al ingreso de la misma.
Resultado Esperado: El sistema no permite ingresar la duplicidad de una especialidad, esto reduce el error de ingreso de especialidades.
Evaluación de Prueba: Aceptada

Fuente: [Elaboración propia]

La Tabla 3.30 nos da un parámetro de cómo ve el usuario final el sistema implementado hasta la HU3. Como podemos observar que después de haber realizado la correspondiente autenticación de un usuario, el mismo tiene la posibilidad de realizar las actividades para las cuales se relataron las HU.

Tabla 3.31: Caso de Prueba HU4

	Caso de Prueba de Aceptación
Código: 04	**Historia de Usuario (Nro. y Nombre): 4**
Nombre: Registro de visitadores médicos	
Descripción: Evaluación de la interfaz por medio de la cual se realizan la adición de los visitadores médicos que componen el sistema.	
Condiciones de Ejecución: Tener instalado el ejecutable de java y la ip de la máquina cliente registrada en la lista de conexiones aceptables en postgresql.	
Entrada / Pasos de ejecución: El supervisor médico se autentica al sistema y procede a la inserción de uno o más visitadores médicos.	
Resultado Esperado: Los visitadores médicos que pertenecen a la empresa quedaran debidamente registrados en el sistema.	
Evaluación de Prueba: Aceptada.	

Fuente: [Elaboración propia]

El caso de prueba 04 describe el nivel de aceptación que se obtiene desde el punto de vista de usuario final.

Tabla 3.32: Caso de Prueba HU5

	Caso de Prueba de Aceptación
Código: 05	**Historia de Usuario (Nro. y Nombre): 5**
Nombre: Ruteo Semanal	
Descripción: Evaluar la interfaz relacionada con la HU5	
Condiciones de Ejecución: Similares al caso de prueba 3	
Entrada / Pasos de ejecución: Un Visitador debe autenticarse en el sistema, ingresar al módulo de ruteo, planificar sus visitas, guardar el trabajo e imprimir sus fichas de control.	
Resultado Esperado: El sistema indica si se planificaron las visitas correspondientes a todos los médicos asociados al visitador médico.	
Evaluación de Prueba: Aceptada pero con algunas modificaciones por realizar.	

Fuente: [Elaboración propia]

En la tabla de caso de prueba de HU5 vemos que la evaluación de prueba no fue aceptada del todo, esto lo mejoraremos en la fase de mantenimiento, ya que los cambios son pequeños.

3.6. FASE DE MANTENIMIENTO.

La fase de mantenimiento abarca a todas las anteriores fases de XP. Esto porque la metodología permite poder realizar cambios y/o mejoras en cualquiera de ellas. Para la mejor comprensión de esta fase tendremos los siguientes puntos.

3.6.1. Reciclaje de Código.

El reciclaje de Código o también llamado "CodeRefactoring" es el proceso de poder volver utilizable e código reutilizable para cualquiera de las clases que conforman el sistema o bien por sistemas futuros, de esta manera se ahorra tiempo y esfuerzo, lo cual conlleva también a un ahorro económico.

Una de las formas más fáciles de reciclar el código es el de transformar ciertos procesos o código de algoritmos a procedimientos y/o funciones. De esta manera simplemente declarando el tipo de clase en la cual se encuentra el procedimiento o función que queremos invocar queda a total disposición de la clase que la invoca. Este proceso también ayuda a reducir el código redundante y a minimizar las líneas de código que el sistema presenta.

3.6.2. Estándares de Código.

La estandarización de código es una parte fundamental para la programación dentro de un equipo de trabajo. Esto ayuda de gran manera a la fácil comprensión del mismo. Como vimos en XP cualquiera de los programadores puede realizar cambios en el código de otro programador. Esto es factible cuando se cuenta con un estándar de código. Esto conlleva a mejoras constantes del código fuente de un sistema.

3.6.3. Rotaciones.

La metodología XP recomienda la rotación de roles del equipo de trabajo, siempre y cuando el miembro a rotar pueda cumplir con las nuevas funciones que se le puedan asignar, por ejemplo es muy viable que el cliente en si se ponga a programar, pero si puede llevar a cabo la coordinación de las diferentes reuniones de avance dentro del proceso de desarrollo del sistema.

Las rotaciones nos ayudan a poder ver un problema desde otro punto de vista, de esta manera también poder imaginar nuevas y mejores soluciones. En nuestro caso en particular se dejó al cliente interactuar con el sistema de forma individual, así él pudo ver ciertas debilidades tanto de comprensión, como de diseño. Con esto logramos en muchas ocasiones la modificación de las diferentes interfaces que el sistema presenta. Al mismo tiempo se logró ver la utilización del sistema desde el punto de vista del cliente, esto ayudó a la mejor implementación del mismo.

3.7. MUERTE DEL PROYECTO.

El presente sistema llegó a su fase de muerte por los siguientes aspectos:

- Se pudo llenar las expectativas del cliente, el mismo quedó satisfecho con el sistema.

- ❖ Se cumplió el tiempo estimado de desarrollo del sistema para la defensa del mismo. Ver Anexo A.

3.8. SEGURIDAD.

Todo sistema debe contar con un nivel de seguridad, esto depende del fin para el cual fue creado el sistema, no es lo mismo un sistema de operaciones militares que uno implementado para la inscripción de alumnos de un instituto X.Por medio de la seguridad se debe resguardar tanto la utilización, el código y los datos del sistema. Es por eso que desglosaremos este las siguientes partes.

3.8.1. Seguridad del sistema.

Para mejorar la seguridad del sistema con respecto a los usuarios que puedan utilizarla se hizo la siguiente interfaz de Autentificación. Este es el primer nivel de seguridad del sistema en si para con los usuarios.

Figura 3.14: Interfaz de Usuario "Autentificación de Usuarios"

Fuente: [Elaboración propia]

Gracias a esta interfaz, solo los usuarios designados podrán utilizar el sistema. Además después de haberse autentificado les saldrá un mensaje de bienvenida con el nombre completo del usuario y su respectivo tipo (administrador, supervisor o visitador médico).

3.8.2. Seguridad de los datos.

Para incrementar la seguridad de los datos que están almacenados se adoptaron las siguientes políticas:

- Solo el encargado de la administración de la BD dentro de la empresa podrá tener acceso a la misma.
- La estación de trabajo que servirá de servidor deberá estar exclusivamente dentro del área de sistemas de la empresa, y no deberá ser implementada individualmente en ninguna de las regionales con las cuales cuenta la empresa.
- Para que una maquina cliente tenga acceso al servidor, el IP[21] de la misma deberá estar registrada en la tabla de accesos permitidos del gestor de la base de datos.
- Se deberán realizar copias de seguridad de la BD periódicamente.

[21] IP: Internet Protocol.

CAPITULO IV

CALIDAD DE SOFTWARE

4.1. INTRODUCCION.

Medir la calidad de un determina software es una de las tareas más difíciles que se presenta en el desarrollo de un sistema. Pero gracias a esta necesidad se fueron creando diferentes formas de medición de las mismas. A estas las llamamos métricas y entre algunas podemos mencionar las siguientes:

- Modelo de McCall.
- Modelo de Boehm.
- Modelo ISO 9126.

Para nuestro proyecto como lo definimos en el Capítulo II utilizaremos el modelo de calidad de la ISO 9126[22].

4.2. ISO 9126.

La norma ISO 9126 nos ayudará a medir la calidad de nuestro sistema siguiendo los criterios ya explicados en el Capítulo II.

4.2.1. Funcionalidad.

Funcionalidad del sistema está dada por el dominio de la información al cual está asociado un valor de complejidad. Los dominios de información son:

---

[22] ISO: Organización Internacional de Normalización. (International StandardOrganization)

- **Número de entradas de usuario.** Se cuenta cada entrada de usuario que proporciona al software diferentes datos orientados a la aplicación. Las entradas deben ser restringidas de las peticiones que se contabilizan por separado.
- **Número de salidas de usuario.** La salida se refiere a informes, pantallas, mensajes de error.
- **Número de peticiones de usuarios.** Una petición está definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva.
- **Número de archivos.** Se cuenta cada archivo maestro lógico. En otras palabras las tablas existentes en la base de datos.
- **Número de interfaces externas.** Se cuenta todas las interfaces legibles por el ordenador que son utilizados para transmitir información a otro sistema.

Tabla 4.1: Factor de ponderación para la Funcionalidad

PARAMETRO DE MEDICION	Factor de ponderación				
	CUENTA	SIMPLE	MEDIO	COMPLEJO	TOTAL
Nro. de entradas de usuario	12	3	4	6	36
Nro. de salidas de usuario	15	4	5	7	60
Nro. de peticiones de usuario	25	3	4	6	75
Nro. de archivos o tabla	10	7	10	15	70
Nro. de interfaces	12	5	7	10	60
Cuenta total					**301**

Fuente: [Elaboración propia]

Para nuestro proyecto tomaremos en todos los casos en Factor de Ponderación simple. Al tratarse de un sistema de no muy alta complejidad.

A continuación para ajustar la complejidad de los puntos de función según el factor de ajuste se asignan pesos de acuerdo a los siguientes casos como se ve en la tabla 10.

Tabla 4.2: Pesos de los puntos Función

FACTOR	VALOR
Sin importancia	0
Incidental	1
Moderado	2
Medio	3
Significativo	4
Esencial	5

Fuente: [Pressman, 2002]

Los ajustes de complejidad con sus respectivos pesos se ven a continuación:

Tabla 4.3: Factores de evaluación

FACTOR	PESO
1. Requiere el sistema copias de seguridad y de recuperación fiable.	5
2. Se requiere comunicación de datos.	4
3. Existen funciones de procesos distribuidos.	1
4. Es crítico el rendimiento.	4
5. Será ejecutado el sistema en un S.O. existente.	5
6. Requiere el sistema entrada interactiva.	3
7. Requiere entrada de datos interactiva sobre múltiples ventanas.	3
8. Se actualiza los archivos de forma interactiva.	4
9. Son complejas las salidas de los archivos a las peticiones.	3

10. Es complejo el procesamiento interno.	3
11. Se ha diseñado el código para ser reutilizable.	4
12. Están incluidas en el diseño la conversión la conversión e instalación.	3
13. Se ha diseñado el sistema para soportar múltiples instalaciones.	4
14. Se ha diseñado la aplicación para facilitar los cambios y ser fácilmente utilizada por el usuario.	4
ΣFi	**50**

Fuente: [Elaboración propia]

Utilizando los resultados obtenidos hasta el momento y reemplazándolos en la siguiente ecuación tenemos:

PF = (Cuenta total) * [0.65 + (0.01 * ΣFi)]

PF = 301 * [0.65 + (0.01 * 50)]

PF = 346.15

Veamos la siguiente tabla:

Tabla 4.4: Escala de Punto Función

ESCALA	OBSERVACION
PF > 300	Optimo
200 < PF < 300	Bueno
100 < PF < 200	Suficiente
PF < 100	Deficiente

Fuente: [Pressman, 2002]

Viendo la Tabla 4.4 podemos observar que nuestro sistema tiene una funcionalidad optima, ya que los puntos de función encontrados tienen el valor de 346.15.

A continuación calculamos el ajuste, el ajuste se lo obtiene de la utilización de la ecuación anterior pero utilizando los factores de evaluación con sus pesos máximos. De ahí que tenemos:

$$PF_{AJUSTE} = (\text{Cuenta total}) * (\text{Valor máximo de complejidad})$$

$$PF_{AJUSTE} = 301 * 1.35$$

$$PF_{AJUSTE} = 406.35$$

Con estos resultados podemos ahora sí calcular la funcionalidad del sistema, esto lo vemos a continuación.

$$F = PF / PF_{AJUSTE}$$

$$F = 346.15/406.35$$

$$F = 0.8518$$

Dónde:

F: Funcionalidad del sistema.

Si expresamos F en porcentajes obtendremos una funcionalidad del 85%, lo que quiere decir que tomando en cuenta todos los parámetros de medición expresados en la Tabla 4.1 nuestro sistema tiene una funcionalidad aceptable.

4.2.2. Fiabilidad.

La Fiabilidad es el Tiempo Medio Entre Fallos (TMEF) y la fórmula es:

$$TMEF = TMDF + TMDR$$

Dónde:

TMDF = Tiempo medio de fallo.

TMDR = Tiempo medio de reparación.

Para calcular TMDR, hacemos uso de la siguiente fórmula:

TMDR = TMAC + TMIC + TMPC + TMDC

Dónde:

TMAC = Tiempo medio de analizar los cambios.

TMIC = Tiempo medio de implementar los cambios.

TMPC = Tiempo medio de probar los cambios.

TMDC = Tiempo medio de distribuir los cambios.

Tomando una muestra de 10 mediciones por día y en base ala media aritmética se obtuvo la siguiente tabla:

Tabla 4.5: Mediciones de Fiabilidad

TMAC	TMIC	TMPC	TMDC
Σ ti /n	Σ ti /n	Σ ti /n	Σ ti /n
15/10	10/10	20/10	10/10
1.5	1	2	1

Fuente: [Elaboración propia]

De los datos obtenidos en la Tabla 4.4obtenemos el TMDR:

TMDR = TMAC + TMIC + TMPC + TMDC

TMDR = 5.5

De las 10 muestras que se hicieron se estima una falla aproximadamente cada 30 horas de uso continuas. Esto lo expresamos como sigue:

TMDF = 30/10 = 3

Ya contando con TMDF y TMDR procedemos a calcular TMEF:

TMEF = 3 + 5.5 = 8.5

Si multiplicamos el resultado de TMEF por 10 obtendremos 85. Este resultado se lo puede expresar de la siguiente manera: De cada 100 veces que se ejecute el sistema, 85 de las veces este correrá de forma correcta. Esto representa un 85% de Fiabilidad lo que quiere decir que nuestro sistema está dentro del rango de aceptabilidad.

Además de medir la Fiabilidad también tenemos que medirla Disponibilidad, que es la probabilidad de que un sistemafuncione de acuerdo con los requisitos exigidos en un momentodado. La Disponibilidad se la puede obtener a través de la siguiente fórmula:

D = [TMDR/(TMDF+TMDR)]* 100%

Dónde:

D: Disponibilidad del sistema.

Reemplazando los datos obtenidos anteriormente tenemos:

D = [5.5/(3+5.5)]* 100

D = 64.67 = 65%

Por lo tanto la probabilidad de que el sistema se encuentredisponible para su uso en un momento dado es del 65%. Esta medida está dentro de una calificación de satisfactoria, pero se debe procurar mejorarla aunque este no sea un sistema de tipo militar o algo parecido.

La medida de disponibilidad es algo más sensible al tiempomedio de respuesta, por lo tanto es una medida indirecta dela facilidad de mantenimiento del software.

4.2.3. Usabilidad.

La Usabilidad representa Facilidad de Uso que el usuario final percibirá del sistema.Éstamétrica nos muestra el esfuerzo necesario para aprender a manipular el sistema. Se lo obtiene a través de los datos obtenidos de una pequeña encuesta que se la realiza directamente a un grupo determinado de usuarios finales.

La tabla de preguntas para la encuesta será la siguiente:

Tabla 4.6: Preguntas de Facilidad de Uso

N°	PREGUNTA	VALOR
1	¿El sistema satisface los requerimientos de manejo de información?	
2	¿Las salidas del sistema están de acuerdo a sus requerimientos?	
3	¿Cómo considera el ingreso de datos al sistema?	
4	¿Cómo considera los reportes que elabora el sistema?	
5	¿El sistema facilita el trabajo que realiza?	
TOTAL		

Fuente:[Elaboración, propia]

Y la escala de ponderación es la siguiente:

Tabla 4.7: Escala de Evaluación para la Usabilidad

DESCRIPCION	VALOR
Pésimo	1
Malo	2
Regular	3

Bueno	4
Muy Bueno	5

Fuente: [Elaboración propia]

La encuesta se la realizo a 5 usuarios finales, el cliente directo y 4 visitadores médicos, y los datos obtenidos los expresamos en la siguiente tabla:

Tabla 4.8: Resultados de la encuesta de Usabilidad

Pregunta	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Promedio
1	4	3	5	5	4	4,2
2	4	4	4	5	4	4,2
3	4	4	3	4	5	4
4	5	4	3	4	3	3,8
5	4	3	4	4	4	3,8
Σx_i						**20**

Fuente: [Elaboración propia]

Con el resultado obtenido y utilizando la siguiente ecuación tendremos:

$$U= [(\Sigma x_i/n)*100]/n$$

Dónde:

U: Usabilidad.

Reemplazando Σx_i y n = 5 porque fueron 5 encuestas, tenemos:

$$U= [(20 / 5)*100]/ 5$$

$$U = 80\ \%$$

Por lo tanto, concluimos que la facilidad de uso del sistema es del 80% y que está dentro del rango aceptabilidad de la norma ISO 9126. También podemos interpretar este resultado como: que de cada 10 personas que utilicen el sistema 8 encontraran al mismo fácil de utilizar.

4.2.4. Mantenibilidad.

El estándar [IEE94] sugiere un Índice deMadurez del Software (IMS) que proporciona un indicadorde la estabilidad de un producto, basándose en los cambiosque ocurren en cada versión del producto. Cuando el IMS se aproxima a 1 se dice que el sistema es más estable. Se lo determinacon la siguiente fórmula:

$$IMS = [Mt - (Fa + Fc + Fd)] / Mt$$

Dónde:

Mt: Número de módulos en la versión actual

Fc: Número de módulos en la versión actual que se han cambiado

Fa: Número de módulos en la versión actual que se han añadido

Fd: Número de módulos en la versión anterior que se han borrado en la versión actual.

A partir del sistema se obtuvieron los siguientes valores necesarios para el cálculo del IMS. Tomamos en cuenta los cambios que se realizaron en la fase de mantenimiento del sistema, después de la puesta en prueba del mismo. Se los expresa en la siguiente tabla:

Tabla 4.9: Valores IMS

Variable	Valor
MT	5
Fc	1
Fa	0

Fe	0

Fuente: [Elaboración propia]

Ahora sí, reemplazando los valores obtenidos en la Tabla 4.9, tenemos lo siguiente:

IMS = [7- (1 + 0 +0)]/ 7

IMS = 0.85

A medida que el IMS se aproxima a 1 el producto empezará aestabilizarse.

Además podemos expresar la Mantenibilidad con la siguiente ecuación:

M = IMS x 100

M = 85%

Dónde:

M: Mantenibilidad del sistema.

Por lo tanto podemos concluir que nuestro sistema cuenta con una Mantenibilidad del 85%. También podemos interpretar este resultado como: que de cada 10 casos en los que sean necesarios un mantenimiento, en 8 de las veces se podrá realizar el mismo de forma sencilla y práctica, para el resto de las veces se necesitara un análisis más complejo.

4.2.5. Portabilidad.

La portabilidad mide la facilidad con la que un sistema puede moverse de un entorno a otro. Es por eso que tomamos en cuenta los siguientes aspectos necesarios para evaluar la portabilidad del mismo.

4.2.5.1. Servidor

Los aspectos necesarios para implementar el sistema del lado del servidor son los siguientes:

- ❖ Tanto como para Windows o para Linux primero debemos instalar el servidor de base de datos, en este caso PostgreSQL en su versión para Windows o para Linux.
- ❖ Descargar el controlador pgJDBC dentro del mismo PostgreSQL.
- ❖ Agregar las direcciones IP de las máquinas cliente que se precisen que accedan al servidor.
- ❖ Instalar el Java RuntimeEnvironment (JDK).

4.2.5.2. Cliente

Ya que el sistema se lo realizó en lenguaje JAVA, la única necesidad que se necesita en el equipo cliente es tener instalado el Java RuntimeEnvironment.

Con el fin de obtener una medida más aproximada de la portabilidad del sistema se realizaron 10 pruebas de instalación tanto para clientes como para servidores. Utilizando la siguiente ecuación, obtuvimos:

$$P = IE / n$$

Dónde:

P: Portabilidad del sistema.

IE: Instalaciones exitosas.

n: Número de instalaciones totales.

Los resultados obtenidos son los siguientes después de 10 instalaciones practicadas.

$$P = 9 / 10$$

$$P = 0.9$$

Se tuvo una instalación no exitosa ya que al momento no se contaba con conexión a internet, y no se pudo descargar el controlador pgJDBC mencionado anteriormente. Es necesario tener una conexión por la cual se pueda bajar el controlador necesario.

Del resultado final obtenido podemos concluir que nuestro sistema tiene una Portabilidad del 90%. Lo cual lo hace prácticamente adaptable a cualquiera de los dos sistemas operativos más conocidos como lo son Windows y Linux en sus diferentes distribuciones.

4.2. CALIDAD GLOBAL.

Para poder obtener la calidad global de nuestro sistema, sacaremos la media de todas las medidas expresadas en porcentajes hasta el momento. Esto lo expresaremos en la siguiente tabla:

Tabla 4.10: Calidad Global

CRITERIOS	RESULTADO
Funcionalidad	85
Fiabilidad	85
Usabilidad	80
Mantenibilidad	85
Portabilidad	90
Calidad Global	**85**

Fuente: [Elaboración Propia]

Con el resultado obtenido de la Tabla 4.10 decimos que nuestro sistema tiene una calidad del 85%, en otras palabras tiene una calidad aceptable.

CAPITULO V

ANALISIS DE COSTO / BENEFICIO

5.1. INTRODUCCION.

El objetivo de este capítulo es demostrar a los usuarios, al igual que al área administrativa de FARMEDICAL que con la utilización de este sistema se obtendrán muchos beneficios que los esperados.

Para tal efecto haremos uso de ciertas herramientas y heurísticas que nos ayudaran a calcular el VAN (Valor Actual Neto), el C/B (Costo Beneficio) y el TIR (Taza Interna de Retorno). Para poder alimentar el VAN también haremos uso de COCOMO II, una herramienta que nos ayuda a estimar el costo de un producto de software basado en el tamaño del mismo.

Después de realizar los cálculos necesarios para la obtención de los resultados esperados estaremos en la capacidad de afirmar si el proyecto es recomendable o no, y en qué medida.

5.2. COCOMO.

El Modelo Constructivo de Costes (o COCOMO, por su acrónimo del inglés COnstructiveCOstMOdel) es un modelo matemático de base empírica utilizado para estimación de costes[1] de software. Incluye tres submodelos, cada uno ofrece un nivel de detalle y aproximación, cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado.

Este modelo fue desarrollado por Barry W. Boehm a finales de los años 70 y comienzos de los 80, exponiéndolo detalladamente en su libro "Software EngineeringEconomics"

COCOMO consta con tres modelos de estimación. En los tres modelos de estimación se utilizan las tres siguientes ecuaciones:

$E = a(KLDC)^b * m(X)$, en persona-mes

$D = c(E)^d$, en meses

$P = E / D$, en personas

Dónde:

E: es el esfuerzo requerido por el proyecto, en persona-mes.

D: es el tiempo requerido por el proyecto, en meses.

P: es el número de personas requerido por el proyecto.

a, b, c y d: son constantes con valores definidos en una tabla, según cada submodelo.

KLDC: es la cantidad de líneas de código, en miles.

m(X): Es un multiplicador que depende de 15 atributos.

A la vez cada modelo se subdivide en modos. Estos modos son:

- **Modo orgánico**: un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía desde unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles (medio).
- **Modo semi - libre** o **semi - acoplado**: corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.
- **Modo rígido** o **empotrado**: el proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Tabla 5.1: Coeficientes a, b, c y d de COCOMO II

Proyecto de Software	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semi – Acoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

Fuente: [Pressman, 2002]

5.3. COSTO DEL SISTEMA.

El costo del sistema lo dividiremos principalmente en tres partes.

5.3.1. Costo de desarrollo del software.

El cálculo de puntos función lo realizamos en el anterior capítulo, de donde tenemos los puntos función no ajustado:

PF = 301

Este resultado se debe convertir a KLDC[23], para eso utilizaremos la siguiente tabla:

Tabla 5.2: Conversión de puntos función a KLDC

LENGUAJE	NIVEL	FACTOR LDC/PF
C	2,5	128
Ansi Basic	5	64
Java	**6**	**53**

[23] KLDC: Kilos de líneas de código.

PL/I	4	80
Ansi Cobol 74	3	107
Visual Basic	7	46
ASP	9	36
PHP	11	29
Visual C+ +	9,5	34

Fuente: [Pressman, 2002]

Como nuestro sistema está desarrollado en JAVA utilizaremos el Factor LDC/PF 53. Reemplazando este dato en la siguiente ecuación tendremos:

$$\text{LDC} = \text{PF} * \frac{\text{Factor LCD}}{\text{PF}}$$

$$\text{LDC} = (301 * 53)$$

$$\text{LDC} = 15953$$

Para convertirlo a KLDC dividimos LDC entre 1000:

$$\text{KLDC} = \frac{15953}{1000} = 15.95$$

A continuación haremos el cálculo del esfuerzo necesario para la programación del sistema. La ecuación que nos ayudara hallar el esfuerzo, viene dada de la siguiente manera:

$E = a * (KLDC)^b$,en persona - mes

Dónde:

E:es el esfuerzo expresado en personas por mes.

a,b:son constantes empíricas

KLDC:es un número estimado de código fuente en miles distribuidas.

Como nuestro proyecto es del tipo Orgánico utilizaremos a = 2.4 y b = 1.05. Reemplazando estos valores en la ecuación, tenemos:

$$E = 2.4 * (15.95)^{1.05} = 2.148650$$

$$E = 43.96$$

Calculando D con c = 1.05 y d = 0.38 tenemos:

$$D = c(E)^{d}, \quad \text{en meses}$$

$$D = 1.05\ (43.96)^{0.38}$$

$$D = 4.32 = 4$$

El proyecto deberá tener un desarrollo de aproximadamente 4 meses. Para calcular la cantidad de programadores utilizamos la siguiente fórmula:

$$P = E / D, \quad \text{en personas}$$

Reemplazando los datos ya conocidos tenemos:

$$P = 43.96 / 4.32$$

$$P = 10.17 = 10 \text{ [programadores]}$$

El Salario promedio de un programador oscila entre los 200 – 300 $us, en nuestro caso tomaremos el valor de 200 $us. A partir de este monto podemos calcular el costo total del desarrollo del software:

Costo mensual de desarrollo = Número de programadores * Salario promedio

Costo mensual de desarrollo = 10 * 200

Costo mensual de desarrollo = 2000 $us

Como el desarrollo de software se lo estima en 4 meses, tendremos:

Costo total de desarrollo = Costo del software x mes * Número de meses

Costo total de desarrollo = 2000 * 4

Costo total de desarrollo = 8000 $us

5.3.2. Costo de implementación.

La empresa cuenta con un área de sistemas por lo cual cuentan con una red interna funcional y con servicio de internet. Por lo tanto el único costo de implementación que se tendrá será el de la configuración de la parte del servidor y de los clientes requeridos por el cliente. Este tendrá un costo de 100 $us.

5.3.3. Costo de elaboración del proyecto.

Los costos de elaboración del proyecto se refieren principalmente a los gastos que se realizaron a lo largo de las diferentes fases de XP. Estas las podemos ver expresadas en la siguiente tabla:

Tabla 5.3: Costo de elaboración del proyecto

DETALLE	IMPORTE ($us)
Análisis y diseño del proyecto	250
Material de Escritorio	50
Internet	50
Otros	50
TOTAL	**400**

Fuente: [Elaboración Propia]

5.3.4. Costo Total del Software.

El costo total del software se lo obtiene de la sumatoria de los diferentes costos vistos hasta el momento, costo de desarrollo, costo de implementación, costo de elaboración del proyecto. Esto lo vemos expresado en la siguiente tabla:

Tabla 5.4: Costo Total del Producto de Software.

DETALLE	IMPORTE ($us)
Costo de desarrollo	8000
Costo de implementación	100
Costo de elaboración del proyecto	400
TOTAL	**8500**

Fuente: [Elaboración propia]

5.2. VALOR ACTUAL NETO.

El VAN o Valor Actual Neto es un procedimiento que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión. La metodología consiste en descontar al momento actual (es decir, actualizar mediante una tasa) todos los flujos de caja futuros del proyecto. A este valor se le resta la inversión inicial, de tal modo que el valor obtenido es el valor actual neto del proyecto.

La fórmula que utilizaremos para hallar nuestro Valor Actual Neto será:

$$VAN = \sum \frac{Ganancias}{(1+k)^n} - \sum \frac{Costos}{(1+k)^n}$$

Dónde:

VAN: Valor Actual Neto

Ganancias: Ingreso de flujo anual

Costos: Salidas de flujo anual

n: Número de periodo

k: tasa de descuento o tasa de interés al préstamo

Los Gastos y Ganancias que se estiman en un lapso de 4 años los mostramos en la tabla 5.5, para este caso en particular utilizaremos una tasa de descuento del 12% ya que es la tasa actual de interés de préstamo en las entidades financieras.

Tabla 5.5: Calculo del VAN

Año	Costos	Ganancias	Costos/$(1+i)^n$	Ganancias/$(1+i)^n$	Resultado
1	8500	0	7589,28571	0	
2	800	2500	637,755102	1992,98469	
3	400	4500	284,712099	3203,01112	
4	200	6500	127,103616	4130,86751	
∑	9900	13500	8638,85653	9326,86332	
$VAN = \sum \frac{Ganancias}{(1+12\%)^n} - \sum \frac{Costos}{(1+12\%)^n}$					**688,006788**

Fuente: [Elaboración Propia]

Interpretación del VAN.

Para ver si un proyecto es rentable utilizamos la siguiente tabla:

Tabla 5.6: Interpretación del VAN

VALOR DEL VAN	INTERPRETACION
VAN > 0	El proyecto es rentable.
VAN = 0	El proyecto también es rentable, porque ya está incorporada ganancia de la Tasa de Interés.
VAN < 0	El proyecto no es rentable.

Fuente: [Elaboración propia]

Como el resultado que obtuvimos es de VAN = 688 y esto es mayor a 0, podemos decir que nuestro proyecto es rentable.

Costo / Beneficio.

Para hallar el costo/beneficio de un proyecto aplicamos la siguiente ecuación:

$$C/B = \sum \text{Ganancias} / \sum \text{Costos}$$

De ahí, reemplazando la ecuación con los valores que ya conocemos, tenemos:

$$C/B = 13500 / 9900$$

$$C/B = 1.36 \text{ [\$us]}$$

Este resultado lo interpretamos de la siguiente forma: Cada dólar invertido para el proyecto es recuperado y además que de cada dólar invertido para el proyecto se obtiene una ganancia de 0.36 dólares.

5.3. TASA INTERNA DE RETORNO.

Cuando el VAN toma un valor igual a 0, *k* pasa a llamarse TIR (tasa interna de retorno). La TIR es la rentabilidad que nos está proporcionando el proyecto.

La ecuación que utilizaremos es la siguiente:

$$TIR = \sum \frac{Ganancias - Costos}{(1-i)^n}$$

Dónde:

TIR: Tasa Interna de Retorno

Ganancias: Flujo de entrada de un periodo

Costos: Flujo de salida de un periodo

i: Tasa de interés al ahorro

n: Número de periodos

Para una más fácil comprensión de esta ecuación, la expresaremos en la siguiente tabla:

Tabla 5.6: Tasa Interna de Retorno

Año	Costos	Ganancias	$\frac{Ganancias - Costos}{(1-i)^n}$
1	8500	0	-8252,4272
2	800	2500	1602,41305
3	400	4500	3752,0808
4	200	6500	5597,4684
TIR			**2699,53507**

Fuente: [Elaboración propia]

El proyecto nos dará una rentabilidad de 2699.5 $us. Si hubiéramos ahorrado la inversión inicial de 8500 $us en cualquier entidad a una tasa de interés del 3% que es la tasa actual al ahorro, al cabo de 4 años contaríamos con la suma de 9288.17 $us, lo que hace una diferencia de 788.17 $us con nuestra inversión inicial. Claramente vemos que la rentabilidad encontrada en la Tabla 5.6 triplica al valor anterior. Por lo tanto esto demuestra que el porcentaje de interés ganado es mayor al 3% y es del 13% Lo que nos indica que nos conviene invertir en el proyecto mucho más que dejar nuestro dinero en el banco.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. INTRODUCCION.

Durante la realización del presente proyecto "Sistema de Seguimiento y Control de Visitadores Médicos" llegamos a tener las siguientes conclusiones y recomendaciones.

6.2. CONCLUSIONES.

El proyecto de grado "Sistema de Seguimiento y Control de Visitadores Médicos" desarrollado e implementado en la empresa FARMEDICAL SRL ha demostrado que se cumplieron dar solución a los problemas planteados en el capítulo I. Así también demuestra que es un sistema de fácil uso y mantenimiento. Para mejor entendimiento de esto señalamos lo siguiente:

- Se pudo brindar una solución efectiva y funcional al problema que aquejaba al área de Visitas Médicas de FARMEDICAL SRL, más específicamente a los Visitadores Médicos que la comprenden.
- Se redujo el volumen de documentación física en más del 30%.
- Se redujo el tiempo que los Visitadores Médicos empleaban para planificar sus visitas en un 20%.
- Se redujo el problema de omisión de un cierto doctor en la planificación de visitas para el mismo en un 30%.
- A partir de la implementación del sistema se pueden obtener reportes mucho más fiables de cómo va el progreso de las visitas médicas y de sus visitadores médicos.
- Se demostró una vez más que la metodología XP es muy buena para el desarrollo de proyecto de pequeño y mediano tamaño.

6.3. RECOMENDACIONES.

Las recomendaciones que se tienen son las siguientes:

- Realizar copias de seguridad si es posible diariamente o en su defecto semanalmente, ya que la información que contiene la BD del sistema es relevante para la existencia y trabajo de la misma empresa.
- Adquirir un servidor exclusivo para los sistemas administrativos, financieros y de seguimiento con los cuales cuenta la empresa.
- Desarrollar un módulo de asignación de productos farmacéuticos a las distintas especialidades con las que esta cuenta.
- Desarrollar un módulo que nos brinde información de cómo los productos farmacéuticos promocionados por los visitadores médicos son recetados por los doctores a los cuales se les realizó una visita médica.

Bibliografía

Extreme Programming. (27 de 11 de 1997). Recuperado el 22 de 11 de 2011, de http://www.extremeprogramming.org/

Agile Alliance. (5 de 3 de 2001). Recuperado el 27 de 11 de 2011, de http://www.agilealliance.org

InnovaAge. (29 de 4 de 2004). Recuperado el 7 de 5 de 2011, de InnovaAge: http://www.innovaportal.com

IntegraAmerica. (2 de 2 de 2006). Recuperado el 7 de 5 de 2011, de IntegraAmerica: http://www.integraamerica.com

Beck, K. (2000). *Extreme Programming Explained*. Addison - Wesley.

Brealey, M. y. (2006). *Principios de Finanzas Corporativas*. Mc Graw Hill.

Cortez, O. (2011). *Sistema de Educación a Distancia para Idiomas Nativos*. La Paz - Bolivia.

Duran, G. M. (2007). *Sistema de Compras e Información Finaciera Facultativa Virtual*. La Paz - Bolivia.

Farmedical. (2 de 2 de 2011). *FARMEDICAL CORP*. Recuperado el 30 de 11 de 2011, de FARMEDICAL SRL.: http://farmedicalcorp.com

INTECO. (2009). *Ingeniería de software: Metodologías y ciclos de vida*. Madrid: Instituto Nacional de Tecnologías de la Comunicación.

ONESS, P. (12 de 1 de 2005). *ONESS PROJECT*. Recuperado el 10 de 9 de 2011, de http://oness.sourceforge.net/

Pressman, R. (2002). *Ingeniería de Software, Un enfoque práctico*. Madrid: Mc. Graw Hill.

Sommerville, I. (2005). *Ingeniería de Software*. Prentice Hall.

ANEXOS

TABLA DE ANÁLISIS DE INVOLUCRADOS.

GRUPOS	INTERESES	PROBLEMAS	RECURSOS Y MANDATOS
Supervisor de Visitadores Médicos	Contar con información actualizada sobre el progreso de los visitadores médicos	Desconocimiento de algún tipo de sistema que realice las funciones requeridas	Predisposición a brindar la información necesaria
Visitadores Médicos	Tener a la mano datos necesarios para organizar de una mejor forma su ruteo	Desconocimiento de algún tipo de sistema que les ayude a llevar su ruteo mensual	Predisposición a colaborar con el proyecto
FARMEDICAL SRL	Tener implementado un sistema que ayude al área de visitadores médicos	Información incompleta con respecto a todas sus áreas de trabajo	Infraestructura adecuada para la realización del proyecto
Gerencia General	Contar con sistema que le ayude a tomar decisiones con respecto al área	Información incompleta para la toma de decisiones.	Predisposición a compartir la experiencia adquirida
Jefe de Sistemas	Ampliar los sistemas de control con los que se cuentan en la empresa	Falta de un sistema que controle el área de visitas médicas	Colaboración en puntos necesarios del proyecto
Médicos	Tener un cronograma de visitas acorde a sus necesidades	Desconocimiento del record de visitas de las q fueron objeto	Colaboración con respecto al tiempo libre con el que cuentan
Población en General	Poder beneficiarse con promociones y descuentos de parte de la empresa	Desconocimiento de promociones y descuentos de parte de la empresa	Predisposición a trabajar con la empresa

MATRIZ DEL MARCO LÓGICO

RESUMEN NARRATIVO	INDICADORES VERIFICABLES	MEDIOS DE VERIFICACIÓN	SUPUESTOS
FIN Mejorar el control y administración de las visitas médicas.	Administración automatizada del área de visitas médicas.	Mejoramiento en las utilidades de la empresa FARMEDICAL SRL.	- Que el funcionamiento de la empresa sea normal. - Que las personas que interactúan con el sistema brinden los datos necesarios y correctos para el eficaz trabajo del sistema.
PROPOSITO Obtención de información necesaria para la buena toma de decisiones en el área de visitas médicas mediante la implementación de un sistema de seguimiento y control en la empresa FARMEDICAL SRL	- Implementación del sistema en un tiempo de 4 meses. - Reducción de tiempo en la realización de cronogramas -Se hace un mejor uso de las muestras médicas.	- Reportes diarios, semanales y mensuales emitidos por el sistema. - Toma de decisiones con información confiable y segura. - Reportes sobre las visitas médicas realizadas.	- Se utiliza un cronograma de visitas médicas de acuerdo a los requerimientos de la empresa. - Que el funcionamiento de la empresa sea normal. - Que los datos que requiera el sistema sean reales.
PRODUCTOS - Módulo de reportes. - Módulo de rutas. - Base de datos confiable y segura. - Estadísticas actualizadas hasta la fecha.	- Los reportes se los realizará más rápidamente en un 90% - Los cronogramas se realizan de forma más rápida en un 95%. - El conteo de visitas médicas realizadas se realizara de forma más rápida en un 85%	- Reportes de situación actualizados. - Informes y fichas emitidos por el sistema. - Informes individuales de cada visitador médico y su desarrollo.	- Capacitación al personal sobre el sistema a ser implementado. - Se cuenten con el equipo de computación y una webhost necesarios para la implementación del sistema. - Se cuente con el equipo necesario para la emisión de reportes, fichas y todo lo que el sistema necesite emitir.
ACTIVIDADES - Recopilación de información. - Análisis. - Diseño. - Implementación.	Recopilación 500Bs Análisis 1500Bs Diseño 2500Bs Implementación 4000Bs Equipos 12000Bs	- Cronogramas realizados en cada regional. - Encuestas dirigidas a los visitadores médicos por regional.	- No se escatime en gastos para la implementación del sistema. - Posibilidad de tener acceso a la documentación necesaria para el respectivo análisis.