



**UNIVERSIDAD TÉCNICA DE AMBATO**

**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA  
E INDUSTRIAL**

**CARRERA DE INGENIERÍA EN SISTEMAS  
COMPUTACIONALES E INFORMÁTICOS**

**Tema:**

---

“ESTUDIO DE METODOLOGÍAS PARA ESTANDARIZAR EL  
DESARROLLO DE SOFTWARE EN EL DEPARTAMENTO DE  
INFORMÁTICA EN LA PASTORAL SOCIAL CARITAS DE LA DIÓCESIS  
DE AMBATO.”

---

Trabajo de Graduación Modalidad: TEMI Trabajo Estructurado de Manera  
Independiente, presentado previo la obtención del título de Ingeniero en Sistemas  
Computacionales e Informáticos.

AUTORA: Diana Gabriela Ulloa Ulloa  
TUTOR: Ing. Mg. Carlos Israel Núñez Miranda

Ambato – Ecuador  
Noviembre 2014

## **APROBACIÓN DEL TUTOR**

En mi calidad de Tutor del trabajo de investigación, sobre el tema: “ESTUDIO DE METODOLOGÍAS PARA ESTANDARIZAR EL DESARROLLO DE SOFTWARE EN EL DEPARTAMENTO DE INFORMÁTICA EN LA PASTORAL SOCIAL CARITAS DE LA DIÓCESIS DE AMBATO”, presentado por la señorita, Diana Gabriela Ulloa Ulloa, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos y méritos suficientes para ser sometido a la evaluación del jurado examinador que el H. Consejo de la Facultad ha asignado.

Ambato, Noviembre 2014.

.....  
Ing. Mg. Carlos Israel Núñez Miranda  
TUTOR

## **AUTORÍA DE TESIS**

El presente trabajo de investigación titulado “ESTUDIO DE METODOLOGÍAS PARA ESTANDARIZAR EL DESARROLLO DE SOFTWARE EN EL DEPARTAMENTO DE INFORMÁTICA EN LA PASTORAL SOCIAL CARITAS DE LA DIÓCESIS DE AMBATO”, es absolutamente auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, Noviembre del 2014

.....

Diana Gabriela Ulloa Ulloa

CI: 1804394649

## **APROBACIÓN DE LA COMISIÓN CALIFICADORA**

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Franklin Mayorga e Ing. Renato Urvina, revisó y aprobó el Informe Final del trabajo de graduación titulado “ESTUDIO DE METODOLOGÍAS PARA ESTANDARIZAR EL DESARROLLO DE SOFTWARE EN EL DEPARTAMENTO DE INFORMÁTICA EN LA PASTORAL SOCIAL CARITAS DE LA DIÓCESIS DE AMBATO”, presentado por la señorita Diana Gabriela Ulloa Ulloa de acuerdo al Art. 18 del Reglamento de Graduación para Obtener el Título Terminal de Tercer Nivel de la Universidad Técnica de Ambato.

.....  
Ing. Mg. José Vicente Morales Lozada  
**PRESIDENTE DEL TRIBUNAL**

.....  
Ing. Mg. Franklin Oswaldo Mayorga Mayorga  
**DOCENTE CALIFICADOR**

.....  
Ing. Mg. Klever Renato Urvina Barrionuevo  
**DOCENTE CALIFICADOR**

## **DEDICATORIA**

*A Dios nuestro señor por bendecirme en cada instante de mi vida y con su guía alcanzar una más de mis metas.*

*A mi madre Marianita quien me dejó su ejemplo de lucha y valentía para seguir siempre adelante y ahora desde el cielo me cuida y me protege.*

*A mis padres Ángel y María quienes fueron los que también lograron esta meta por su esfuerzo dedicación y apoyo en toda mi vida.*

*A mis hermanos, familiares, novio y amigos quienes siempre me apoyaron y estuvieron conmigo ayudándome para lograr esta meta.*

***Diana Ulloa***

## **AGRADECIMIENTO**

*A Dios por permitir que todas las cosas sean posibles y por las bendiciones recibidas.*

*A la Universidad Técnica de Ambato y en especial a la Facultad de Ingeniería en Sistemas Electrónica e Industrial por acogerme y permitir cumplir mis metas*

*A todos mis profesores que me guiaron para llegar a la meta deseada que es de ser una profesional y a mis padres que me brindaron todo el apoyo necesario en mi vida de estudiante.*

***Diana Ulloa***

## ÍNDICE GENERAL

CONTENIDO	PÁGINA
Caratula	i
Aprobación del Tutor	ii
Autoría de Tesis	iii
Aprobación de la Comisión Calificadora	iv
Dedicatoria	v
Agradecimiento	vi
Índice	vii
Resumen Ejecutivo	xii
Introducción	1
<b>CAPÍTULO I</b>	
El Problema de la investigación	3
1.1. Tema	3
1.2. Planteamiento del Problema	3
1.3. Delimitación	5
1.4 Justificación	5
1.5 Objetivos	6
1.5.1. Objetivo General	6
1.5.2 Objetivo Específicos	6
<b>CAPÍTULO II</b>	
Marco Teórico	7
2.1 Antecedentes Investigativos	7
2.2 Fundamentación Teórica	8
2.2.1 Ingeniería de Software	8
2.2.2 Metodologías de Desarrollo de Software	9
2.2.3 Ventajas del Uso de Metodología	9

2.2.4 Metodologías Ágiles	9
2.2.5 Metodologías Tradicionales	10
2.2.6 El Manifiesto Ágil	10
2.2.7 SCRUM	12
2.2.7.1 Beneficios	13
2.2.7.2 Características	13
2.2.7.3 Roles	14
2.2.7.4 Elementos	16
2.2.7.5 Reuniones	17
2.2.8 Extreme Programming(XP)	19
2.2.8.1 Fundamentos	19
2.2.8.2 Roles	20
2.2.8.3 Características	21
2.2.8.4 Fases	22
2.2.8.5 Proceso	29
2.2.9 RUP(Rational Unified Process)	30
2.2.9.1 Principios	31
2.2.9.2 Ciclo de vida	31
2.2.9.3 Fases	33
2.3 Propuesta de la Solución	34

### **CAPÍTULO III**

Metodología	
3.1 Modalidad básica de investigación	35
3.2 Población y Muestra	35
3.3 Recolección de Información	36
3.4 Procesamiento y Análisis de Datos	36
3.5 Desarrollo del Proyecto	36



## CAPÍTULO IV

Propuesta	37
4.1 Identificación de Procesos	37
4.1.1 Análisis del sistema ASES	39
4.1.2 Procesos	40
4.1.3 Módulos del sistema ASES	41
4.1.4 Estimación de tiempos	42
4.1.5 Tiempo real invertido	43
4.1.6 Burn Down	44
4.2 Estudio de las metodologías	45
4.3 Elección de la metodología	45
4.3.1 Características d las metodologías	45
4.3.2 Comparación de las metodologías	48
4.3.2.1 Criterios de comparación	49
4.3.2.1.1 Ciclo de vida del proyecto	51
4.3.2.1.2 Planificación	54
4.3.2.1.3 Calidad	57
4.3.2.1.4 Herramientas	59
4.3.2.1.5 Matriz comparativa	61
4.3.2.2 Adaptación de la metodología XP	63
4.3.2.2.1 Valores XP	63
4.3.2.2.2 Prácticas XP	63
4.3.2.2.3 Esfuerzo de desarrollo	64
4.3.2.2.4 Burn Down en XP	68
4.4 Guía de la metodología XP	69
4.4.1 Exploración	70
4.4.2 Planificación	73
4.4.3 Iteración	77
4.4.3.1 Análisis	77
4.4.3.1.1 Plan de Entregas	77
4.4.3.1.2 Reuniones	77

4.4.3.2 Diseño	78
4.4.3.2.1 Tarjetas CRC	78
4.4.3.2.2 Soluciones Rápidas	81
4.4.3.2.3 Recodificación	81
4.4.3.3 Codificación	82
4.4.3.3.1 Programación en parejas	82
4.4.3.3.2 Integraciones	83
4.4.3.3.3 Estándares	83
4.4.3.3.4 Propiedad Colectiva	84
4.4.3.4 Pruebas	85
4.4.3.4.1 Pruebas Unitarias	85
4.4.3.4.2 Pruebas de Aceptación	86
4.4.4 Producción	88
4.4.5 Mantenimiento	88

## **CAPÍTULO V**

Conclusiones y Recomendaciones	89
5.1 Conclusiones	89
5.2 Recomendaciones	90
Bibliografía	91
Glosario de Términos	93
Anexos	94

## **ÍNDICE DE TABLAS**

Tabla 2.1 Comparación metodologías ágiles y tradicionales	11
Tabla 3.1 Personal del departamento de la Pastoral	35
Tabla 4.1 Módulo del sistema ASES	41
Tabla 4.2 Estimación de tiempos del sistema ASES	42
Tabla 4.3 Tiempo total invertido del sistema ASES	43

Tabla 4.4 Matriz comparativa de metodologías	48
Tabla 4.5 Escala de evaluación de la metodologías	51
Tabla 4.6 Criterio “Ciclo de vida del proyecto” Metodología SCRUM	52
Tabla 4.7 Criterio “Ciclo de vida del proyecto” Metodología XP	52
Tabla 4.8 Criterio “Ciclo de vida del proyecto” Metodología RUP	53
Tabla 4.9 Resultado del criterio “Ciclo de vida del proyecto”	53
Tabla 4.10 Criterio “Planificación” Metodología SCRUM	54
Tabla 4.11 Criterio “Planificación ” Metodología XP	55
Tabla 4.12 Criterio “Planificación ” Metodología RUP	55
Tabla 4.13 Resultado del criterio “Planificación”	56
Tabla 4.14 Criterio “Calidad” Metodología SCRUM	57
Tabla 4.15 Criterio “Calidad” Metodología XP	58
Tabla 4.16 Criterio “Calidad” Metodología RUP	58
Tabla 4.17 Resultado del criterio “Calidad”	58
Tabla 4.18 Criterio “Herramientas” Metodología SCRUM	60
Tabla 4.19 Criterio “Herramientas” Metodología XP	60
Tabla 4.20 Criterio “Herramientas” Metodología RUP	60
Tabla 4.21 Resultado del criterio “Herramientas”	61
Tabla 4.22 Matriz General de resultados	62
Tabla 4.23 Tiempo estimado del proyecto con XP	66
Tabla 4.24 Tiempo real invertido	67
Tabla 4.25 Identificación de roles	70
Tabla 4.26 Historias de usuario	72
Tabla 4.27 Ejemplo de historias de usuario	74
Tabla 4.28 Estimación de tiempo con XP	75
Tabla 4.29 Velocidad Estimada	76
Tabla 4.30 Modelo de Tarjeta CRC	78
Tabla 4.31 Ejemplo de tarjeta CRC	79
Tabla 4.32 Ejemplo de Prueba unitaria	85
Tabla 4.33 Plantilla de prueba de aceptación	86
Tabla 4.34 Ejemplo de prueba de aceptación	87

## ÍNDICE DE GRÁFICOS

Figura 2.1 Modelo de desarrollo SCRUM	12
Figura 2.2 Ciclo de SCRUM	14
Figura 2.3 Ciclo Sprint	17
Figura 2.4 Ciclo XP	22
Figura 2.5 Planificación de entrega	24
Figura 2.6 Modelo de desarrollo XP	30
Figura 2.7 Ciclo RUP	32
Figura 2.8 Fases RUP	34
Figura 4.1 Grafico de Burn Down	44
Figura 4.2 Criterio “Ciclo de vida del proyecto”	54
Figura 4.3 Criterio “Planificación”	57
Figura 4.4 Criterio “Calidad”	59
Figura 4.5 Criterio “Herramientas”	62
Figura 4.6 Resultado final	64
Figura 4.7 Gráfico Burn Down	68
Figura 4.8 Ejemplo de asignación de historias de usuario	73
Figura 4.9 Ejemplo de diagrama de planificación	76
Figura 4.10 Presentación de capas de desarrollo	79
Figura 4.11 Ejemplo de interfaz de usuario	80
Figura 4.12 Herramienta de programación	82

## ÍNDICE DE ANEXOS

ANEXO A Encuesta dirigida al personal de la Pastoral	94
ANEXO B Árbol del problema	96
ANEXO C Listado de clientes de la Pastoral	97
ANEXO D Solicitud de crédito	98
ANEXO E Movimiento de cuenta grupal	99
ANEXO F Movimiento de cuenta individual	100
ANEXO G Información del Sistema ASES	101

## **RESUMEN EJECUTIVO**

La investigación sobre “Estudio de metodologías para estandarizar el desarrollo de software en el departamento de informática en la Pastoral Social Caritas de la Diócesis de Ambato” tiene como objetivo elaborar una guía de aplicación que mejore el desarrollo de software en la empresa. En la actualidad podemos encontrar mucha información sobre metodologías para desarrollar software y cómo implementarlas, el estudio se centra en la selección de la mejor metodología de desarrollo de software que se acople a las necesidades de la empresa para que facilite la elaboración de los sistemas en la Pastoral.

La empresa carece de la utilización de metodologías que son necesarias para el desarrollo de software, por cuanto se presentan inconvenientes en las fases previas de análisis y diseño, los mismos que generan retrasos en los procesos por no haber tenido desde el inicio una planificación previa por ello se desarrolla una guía de aplicación para que se reduzca los retrasos y mejore la calidad de los sistemas en la empresa.

Con el estudio de las metodologías la empresa tendrá un crecimiento significativo en el desarrollo de sistemas, en cuanto al aspecto tecnológico y económicamente al ahorrar recursos, tiempo y así como también al brindar mejor atención al cliente y un poder de auge para la empresa.

Este tema enfrenta el problema de realizar un estudio comparativo de las metodologías más actuales de acuerdo al proceso que se esté llevando a cabo en el desarrollo de sistemas y así poder presentar criterios y analizar la mejor metodología para utilizarse y estandarizar en todos los sistemas que se desarrollen en la Pastoral.

## INTRODUCCIÓN

En los últimos años la industria del software ha precisado replantear los cimientos sobre los que se sustenta el desarrollo software y las metodologías ágiles han irrumpido con fuerza como un intento de despojar al desarrollo software de la forma como se ha planteado por las metodologías convencionales y en este entorno se pretende satisfacer la necesidades del cliente en el menor tiempo posible y aumentar la productividad.

El presente proyecto llamado “Estudio de metodologías para estandarizar el desarrollo de software en el departamento de informática en la Pastoral Social Caritas de la Diócesis de Ambato”, es de suma importancia para un mejor desarrollo de software en la empresa, esto implica mejorar los procesos y las planificaciones previas al desarrollo, el tiempo, los recursos y todos los beneficios que la investigación de este proyecto trae a la misma.

El primer capítulo pone en evidencia el problema real que tiene la Pastoral en lo que se refiere a un estudio de metodologías ágiles, su planteamiento, delimitación, justificación y objetivos con el fin de clarificar el contexto sobre el cual se va a desarrollar esta investigación.

En el segundo capítulo se dan a conocer antecedentes que se han encontrado sobre la presente investigación y fundamentos teóricos sobre los que la investigación se basa para desarrollarse. Aquí se menciona la propuesta de solución del trabajo.

En el tercer capítulo se describen los diferentes tipos de investigación que se utilizaron por parte del investigador y se detalla la población y a la muestra, Además, se plantean los planes de recolección, procesamiento y análisis de datos y se enumeran los pasos del desarrollo del proyecto.

En el capítulo cuatro se desarrolla la propuesta de tal manera que se describe punto por punto lo mencionado en el capítulo tres, con la finalidad de obtener la mejor metodología para el desarrollo de software de la Pastoral.

En el quinto capítulo se dictan las conclusiones y recomendaciones obtenidas luego de un largo análisis de las metodologías.

## **CAPÍTULO I: EL PROBLEMA**

### **1.1 TEMA**

Estudio de metodologías para estandarizar el desarrollo de software en el departamento de informática en la Pastoral Social Caritas de la Diócesis de Ambato.

### **1.2 PLANTEAMIENTO DEL PROBLEMA**

Las metodologías de desarrollo están adquiriendo gran popularidad en los últimos años y las actuales características de la industria del software han precisado replantear los cimientos sobre los que se sustenta el desarrollo software convencional. En este entorno inestable la ventaja competitiva se encuentra en aumentar la productividad y satisfacer las variantes necesidades del cliente en el menor tiempo posible, para proporcionar un mayor valor al negocio. Ante esta situación, cabe reflexionar sobre el grado de adaptación de las metodologías convencionales a estas circunstancias. La mayoría de los estudios coinciden en que el carácter normativo y la fuerte dependencia de planificaciones previas al desarrollo que definen a las metodologías convencionales, implican que resulten excesivamente pesadas para cubrir las necesidades de un amplio porcentaje del mercado software actual.

En nuestro país las metodologías ágiles han irrumpido con fuerza como un intento de despojar al desarrollo de software planteado por las metodologías convencionales, y son muchas las organizaciones con bastante interés en las mismas. Estas metodologías hacen posible ajustarse rápidamente a las necesidades del cliente y usuarios optimizando la



economía de los proyectos minimizando los riesgos y optimizando recursos.

En la provincia de Tungurahua los avances tecnológicos han evolucionado con rapidez y al mismo tiempo las metodologías ágiles se han impuesto en el desarrollo de software con el fin de hacerlo más predecible y eficiente; con ello ha surgido la necesidad de estudiar las mismas, en busca de un proceso que ayude al desarrollo del software proporcionando un esfuerzo simplificado y siendo bastante claro para presentar los resultados obtenidos, en aspectos tales como: las características del producto, la adaptabilidad a la empresa para que tenga una acogida satisfactoria al mercado.

En la ciudad de Ambato, la utilización de metodologías ágiles es escasa según comentarios de programadores que explican que todavía desarrollan sistemas con metodologías tradicionales, no obstante esto hace que utilicen procesos con extensa documentación y los desarrolladores trabajen invirtiendo mayor tiempo ocasionando pérdida de recursos.

En la Pastoral se presentan inconvenientes en las fases previas de especificación de requisitos, análisis y diseño del software ya que estas fases no fueron planificadas inicialmente en forma correcta y sobrepasaron el tiempo establecido y como resultado genera un retraso en la entrega de las aplicaciones.

Al no usar ninguna metodología se tendrá dificultad para resolver problemas y la mayoría de los casos el resultado, serán aplicaciones bajas en calidad y los programadores pasan por una pérdida de tiempo analizando los proyectos de gran complejidad intentado resolver errores cuya corrección a futuro tiene un costo alto.

### 1.3 DELIMITACIÓN

**ÁREA ACADÉMICA:** Software

**LÍNEA DE INVESTIGACIÓN:** Ingeniería de software

**SUB-LÍNEA DE INVESTIGACIÓN:** Métodos avanzados de producción de software

**ESPACIAL:** La presente investigación se realizará en la ciudad de Ambato, específicamente en la Pastoral Social Caritas de la Diócesis de Ambato.

**TEMPORAL:** El presente proyecto de investigación tendrá una duración de 6 meses, a partir de la aprobación por el Consejo Directivo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

Proyecto de investigación aplicada.

### 1.4 JUSTIFICACIÓN

El estudio de las metodologías para estandarizar el desarrollo de los sistemas en la empresa, está enfocado para facilitar la elaboración y desarrollo de los mismos y con esto obtener un producto de calidad incorporando las mejores prácticas de ingeniería de software, que cumpla con las características de funcionalidad, usabilidad, y fiabilidad.

El mercado competitivo de los productos tecnológicos, además de los conceptos básicos de calidad, coste, exige también rapidez y flexibilidad esto ha hecho que la empresa requiera la utilización de metodologías ágiles para poder brindar un mejor desempeño en el desarrollo de proyectos de la empresa y así poder alcanzar los objetivos planteados a corto plazo para poder tener más competitividad.

Se ha constatado que en la Pastoral no se han realizado ningún estudio sobre la temática planteada, por lo que es de gran interés e importancia el desarrollo de la misma.

Actualmente en la organización no se utiliza ninguna metodología por lo cual es necesario el estudio de una metodología ágil que esté de acuerdo con el avance tecnológico y permita el desenvolvimiento óptimo con su aplicación, reduciéndose así costos.

## **1.5 OBJETIVOS**

### **1.4.1 OBJETIVO GENERAL:**

Realizar un estudio de metodologías para la estandarización del desarrollo de software en el departamento de informática en la Pastoral Social de la Diócesis en la ciudad de Ambato.

### **1.4.2 OBJETIVOS ESPECÍFICOS:**

Estudiar los procesos de desarrollo de software que actualmente se utiliza en la pastoral.

Realizar un estudio comparativo entre las metodologías de desarrollo de software.

Identificar la metodología que se ajuste con los requerimientos de la pastoral.

Elaborar una guía de aplicación de la metodología de desarrollo de software adecuada para la Pastoral Social.

## **CAPÍTULO II: MARCO TEÓRICO**

### **2.1 ANTECEDENTES INVESTIGATIVOS**

En base a los estudios realizados en la empresa se ha llegado a determinar que no ha existido alguna investigación o estudio de las metodologías para el desarrollo de sistemas de la Pastoral en la ciudad de Ambato.

Dentro de los registros bibliográficos que reposan en la Biblioteca de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, no se logró encontrar ningún trabajo investigativo que podrían guardar relación alguna tema propuesto.

Se han encontrado en los registros bibliográficos de la biblioteca de la Escuela Politécnica del Ejército los siguientes trabajos investigativos que podrían guardar relación con el tema propuesto.

“Método ágil Scrum, aplicado a la implantación de un sistema informático para el proceso de recolección masiva de información con tecnología móvil”. Realizada por: Toapanta Kleber, Vergara Marco, Campaña Mauricio. Año 2010.

El presente estudio se enfocó en el análisis del método ágil SCRUM para la implementación de una metodología aplicada al desarrollo de software y busca básicamente rapidez, calidad y reducción de costos en la ejecución de sus proyectos; para asumir estos retos es necesario tener agilidad y flexibilidad; en la recolección masiva de información con dispositivos móviles.

“Desarrollo dirigido por test (tdd) utilizando el framework junit en un sistema web de asignación de aulas de los laboratorios generales de computación de la ESPE, aplicando la metodología agile unified process (aup)”. Realizada por: Aucancela Carlos, Pozo Tatiana. Año 2010.

En el presente trabajo se estudia sobre la técnica del Desarrollo Dirigido por Tests (TDD) con enfoque en el proceso de pruebas utilizando el framework junit y aplicar en el caso práctico, “Sistema web de asignación de aulas de los laboratorios generales de computación de la ESPE”, en base a los lineamientos de la metodología ágil AUP.

## **2.2 FUNDAMENTACIÓN TEÓRICA**

Los requerimientos de la tecnología de la información que demanda en la actualidad y así mismo el interés en las características y funciones que brinda una aplicación y se crea opiniones distintas en cuanto a la funcionabilidad que debería tener el sistema, sus aplicaciones y características con las mismas que se debería hacer un esfuerzo concentrado antes de desarrollar una aplicación.

Los individuos, negocios y gobiernos, dependen cada vez más del software para tomar decisiones estratégicas y tácticas, así como para sus operaciones y control cotidiano. Si el software falla, las personas y empresas grandes pueden experimentar desde un inconveniente menor hasta fallas catastróficas, entonces el software debe tener alta calidad. [1]

### **2.2.1 INGENIERÍA DE SOFTWARE**

Disciplina y profesión enfocada a la aplicación de conocimiento científico y técnico y que utiliza recursos físicos para diseñar e implementar diversas estructuras, máquinas, dispositivos, sistemas y procesos para realizar un objetivo deseado y alcanzar criterios especificados. [2]

## **2.2.2 METODOLOGÍAS DE DESARROLLO DE SOFTWARE**

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.

## **2.2.3 VENTAJAS DEL USO DE METODOLOGÍA**

Desde el punto de vista de gestión: Facilitar la tarea de planificación la tarea de control y seguimiento de un proyecto, mejora la relación coste/beneficio y optimiza el uso de recursos disponibles para facilitar la evaluación de resultados y cumplimiento de los objetivos además de la comunicación efectiva entre usuarios y desarrolladores.

Desde el punto de vista de los ingenieros del software: Ayuda a la comprensión del problema y optimiza el conjunto y cada una de las fases del proceso de desarrollo así como el mantenimiento del producto final y la reutilización de partes del producto.

Desde el punto de vista del cliente o usuario: Garantiza un determinado nivel de calidad en el producto final y da confianza en los plazos de tiempo fijados en la definición del proyecto además de definir el ciclo de vida que más se adecue a las condiciones y características del desarrollo. [3]

## **2.2.4 METODOLOGÍAS ÁGILES**

Las Metodologías Ágiles o “ligeras” constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales, debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.[4]

## **2.2.5 METODOLOGÍAS TRADICIONALES**

Hay una serie de metodologías que solemos llamar tradicionales propuestas casi todas ellas con anterioridad a los años 90 que pretendían ayudar a los profesionales indicando pautas para realizar y documentar cada una de las tareas del desarrollo del software.

## **2.2.6 EL MANIFIESTO ÁGIL**

En el manifiesto ágil se valoran los siguientes puntos: Al individuo, las interacciones del equipo de desarrollo, el proceso y las herramientas: dado que el recurso humano es uno de los principales factores para el éxito del proyecto, es de suma importancia lograr que el equipo de trabajo cree su propio entorno y no que éste se adapte a uno ya prefabricado, el mismo que no podría cumplir precisamente con todas sus necesidades.

La colaboración del cliente más que la negociación de un contrato; es de vital importancia que exista una constante comunicación con el cliente, de forma que él se sienta un miembro más del grupo del trabajo y no solamente quien en muchas ocasiones impone las reglas, procurando con esto garantizar el éxito de proyecto.

Responde a los cambios más que seguir estrictamente un plan: es claro que se debe realizar una planificación previa pero esta debe ser lo suficientemente flexible para adaptarse al entorno que podría variar drásticamente. [5]

## Metodologías Ágiles vs Metodologías Tradicionales

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

**Tabla Nro. 2.1 Comparación de Metodologías**

**Fuente: [www.sisman.utm.edu.ec](http://www.sisman.utm.edu.ec)**



## 2.2.7 SCRUM

Scrum es un proceso ágil que se puede usar para gestionar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales. Es un proceso incremental iterativo para desarrollar cualquier producto o gestionar cualquier trabajo.

En Scrum un proyecto se ejecuta en bloques temporales (iteraciones-sprints) de un mes (pueden ser de dos o tres semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite.

### Modelo de desarrollo aplicando SCRUM



**Figura Nª 2.1 Modelo de desarrollo SCRUM**

*Elaborado por: JOSKOWICZ, José*

El Sprint es el ritmo de los ciclos de Scrum. Está delimitado por la reunión de planificación del sprint y la reunión retrospectiva. Una vez que se fija la duración del sprint es inamovible. La mayoría de los equipos eligen dos, tres o cuatro semanas de duración. Diariamente durante el sprint, el equipo realiza una reunión de seguimiento muy breve. Al final del sprint se entrega el producto al cliente en el que se incluye un incremento de la funcionalidad que tenía al inicio del sprint.

El proceso parte de la lista de requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente ha priorizado los requisitos

balanceando el valor que le aportan respecto a su coste y han sido divididos en iteraciones y entregas. [7]

#### **2.2.7.1 Beneficios**

- Potenciación responsable de organizar el trabajo por parte del equipo, que es quien mejor conoce como realizarlo.
- Define las tareas necesarias para poder completar cada requisito, creando la lista de tareas de la iteración.
- Realiza una estimación conjunta del esfuerzo necesario para realizar cada tarea.
- Es el equipo quien asume la responsabilidad de completar en la iteración los requisitos que selecciona.
- Es cada una de las personas la que se responsabiliza de realizar las tareas a las que se asigna.
- Una estimación conjunta es más fiable, dado que tiene en cuenta los diferentes conocimientos, experiencia y habilidades de los integrantes del equipo.

SCRUM es un método ágil de gestión de proyectos cuyo objetivo principal es elevar al máximo la productividad del equipo de desarrollo. Define un marco para la gestión de proyectos, que se han utilizado con éxito durante los últimos años. Esta especialmente indicada para proyectos con un cambio de requisitos.

Scrum distingue entre dos elementos principales: Actores y acciones. Los actores son las personas que ejecutan las acciones, y las acciones son las distintas fases del ciclo de desarrollo de Scrum.

#### **2.2.7.2 Características**

- Conseguir una mejor aproximación entre las funcionalidades del software y los requerimientos del cliente.
- Comenzar el trabajo lo más rápidamente posible

- Manejo más eficiente de los requerimientos cambiantes en un proyecto.
- Mejorar la comunicación entre el cliente y el equipo desarrollador.

## Componentes

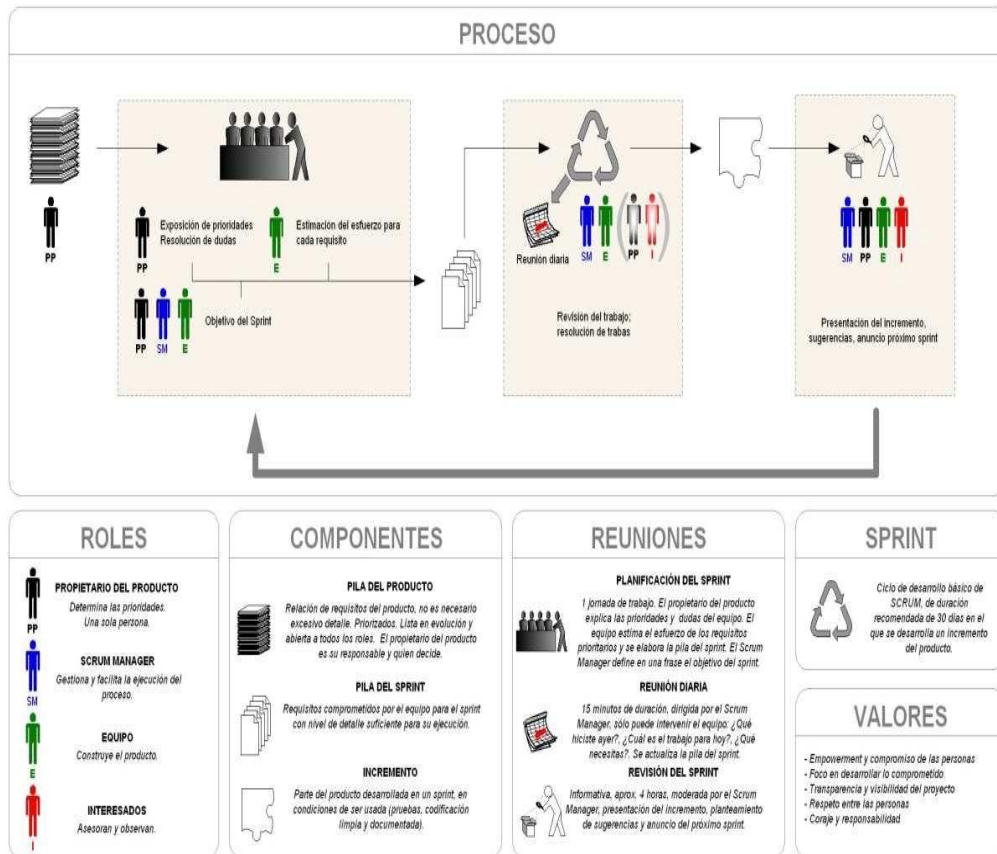


Figura Nª 2.2 Ciclo de SCRUM

Fuente: [www.metodologiasagiles.es](http://www.metodologiasagiles.es)

### 4.2.1.1.3 Roles o Responsabilidades

**Responsables del producto: “Product Owner” (propietario del producto)**

Persona concedora del entorno de negocio del cliente y de la visión del producto.

Representa a todos los interesados en el producto final, financia el proyecto, provee los requerimientos del sistema, prioriza las funcionalidades, adapta los incrementos de cada Sprint, define los objetivos del proyecto.

Es el responsable del “Product Backlog” (pila de producto), y proponiendo los requisitos más prioritarios a desarrollar.

Debe estar disponible durante la iteración para responder las posibles preguntas que pudieran aparecer.

### **Responsables del desarrollo: “Scrum Team” (Equipo)**

Equipo multidisciplinar que cubre todas las habilidades necesarias para generar el resultado requerido por el “Product Owner”.

Se auto-gestiona y auto-organiza de máximo de 5 a 10 personas.

Dispone de atribuciones suficientes para toma de decisiones sobre cómo realizar su trabajo y seleccionan los requisitos que pueden completar en cada iteración, realizando al cliente las preguntas necesarias

### **Responsables del funcionamiento de Scrum: “Scrum Master” (Director)**

Garantiza el funcionamiento y cumplimiento de los procesos y metodologías que se emplean es el director del proyecto. Es decir se encarga de servir como guía para los integrantes del equipo a través de las reuniones.

Dirección de la empresa, con el conocimiento de gestión y desarrollo ágil y facilitando los recursos necesarios.”Responsables del Departamento y del área de gestión de proyectos.

## **Usuarios**

Son los usuarios finales de la aplicación. A partir del progreso de la aplicación, pueden aportar nuevas ideas de modo que la aplicación final se adapte a sus necesidades personales.[8]

### **4.2.1.1.4 Elementos**

#### **“Product Backlog” (Pila del producto)**

Es una lista de todas las tareas, funcionalidades o requerimientos del proyecto. El Product Owner se encarga de priorizar estas tareas y de actualizar la lista con los objetivos conseguidos con la ayuda del Scrum Master. Se parte del resultado que se desea obtener evolucionando durante el desarrollo.

Todos los integrantes del equipo de desarrollo podrán acceder a él aportando ideas. Pero solo el dueño puede tomar la decisión final.

#### **“Sprint Backlog” (Pila del sprint)**

Lista de trabajos que realizará el equipo durante el sprint. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite.

Compromiso de ejecución. Asignación de tareas de forma personal con estimación de tiempos y recursos necesarios

#### **“Scrum Daily” (Reunión equipo)**

Demostración de los objetivos alcanzados en cada sprint. Asistencia de todos los roles, “Product Owner” e incluso usuarios.

Es una tarea iterativa que se realiza todos los días que dure el “Sprint Backlog” con el equipo de desarrollo o de trabajo. Se trata de una reunión

operativa, informal y ágil de un tiempo máximo de 30 minutos, donde se da respuesta a 3 preguntas que se hacen a cada integrante.

- ¿Qué se ha hecho desde la última reunión?
- ¿Qué se va hacer?
- ¿Qué ayuda necesita?

#### **4.2.1.1.5 Reuniones**

### **Sprint**

Un sprint es un periodo de tiempo durante el que se desarrolla un incremento de funcionalidad. Constituye el núcleo de Scrum, que divide de esta forma el desarrollo de un proyecto en un conjunto de pequeñas carreras.

Duración máxima del sprint: 30 días, pero se recomienda 15 días.

Durante el sprint no se puede cambiar el curso de un sprint, abortándolo, y solo lo puede hacer el Scrum Master si decide que no es viable por alguna de las siguientes razones:

- La tecnología acordada no funciona
- Las circunstancias del negocio han cambiado
- El equipo ha tenido interferencias



**Figura N° 2.3 Ciclo Sprint**  
**Fuente: [www.guiadeingenieria.com](http://www.guiadeingenieria.com)**

- **Planificación del Sprint**

El equipo planifica la iteración, dado que ha adquirido un compromiso, es el responsable de organizar su trabajo y es quien mejor conoce cómo realizarlo.

Intervienen todos los roles y define las tareas necesarias para poder completar cada requisito, creando la lista de tareas de la iteración.

Se genera el “Sprint Backlog” o lista de tareas que se van a realizar.

Se determina el “objetivo del Sprint” (funcionalidad del negocio que se va a generar).

Una estimación conjunta es más fiable, dado que tiene en cuenta los diferentes conocimientos, experiencia y habilidades de los integrantes del equipo.

- **Seguimiento del Sprint**

Breve reunión diaria para repasar cada una de las tareas y el trabajo previsto de la jornada. Sólo interviene el equipo de desarrollo.

Cada miembro responde a tres preguntas:

¿Trabajo realizado desde la reunión anterior?

¿Trabajo que se va a realizar hasta la próxima reunión de seguimiento?

¿Problemas que se deben solucionar para realizar el trabajo propuesto?

- **Revisión del Sprint**

El objetivo de la reunión de revisión es presentar el producto o porción del producto desarrollado por el equipo a los usuarios y la misma se utiliza para detectar inconformidades y se los resuelve en el siguiente sprint.

Análisis y revisión del incremento generado en si constituye la presentación de resultado. [9]

### **2.2.8 EXTREME PROGRAMMING (XP)**

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 es 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.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. El ciclo de vida ideal de XP consisten en 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto. [10]

#### **2.2.8.1 Fundamentos**

La programación extrema es una metodología recientemente utilizada en el desarrollo de software. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso, lo integra como una parte más del equipo de desarrollo.

XP fue inicialmente creada para el desarrollo de aplicaciones dónde el cliente no tiene una concepción clara de las funcionalidades que tendrá la aplicación que se desarrollará. Este desconocimiento podría provocar un cambio constante en los requisitos que debe cumplir la aplicación por lo que es necesaria una metodología ágil como XP que se adapta a las



necesidades del cliente y dónde la aplicación se va revisando constantemente.

XP está diseñada para el desarrollo de aplicaciones que requieren un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

### **2.2.8.2 Roles**

Existen diferentes roles (actores) y responsabilidades en XP para diferentes tareas y propósitos durante el proceso:

Programador (Programmer)

- Responsable de decisiones técnicas
- Responsable de construir el sistema
- Define tareas a partir de las historias y hace estimaciones.
- En XP, los programadores diseñan, programan y realizan las pruebas.

Cliente (Customer)

- Determina las historias de usuarios y establece prioridades.
- Determina qué construir y cuándo
- Escribe tests funcionales para determinar cuándo está completo un determinado aspecto

Tutor /Entrenador (Coach)

- El líder del equipo - toma las decisiones importantes
- Principal responsable del proceso
- Observa todo, Identifica señales de peligro, se asegura que el proyecto se mantenga en curso.

Encargado del seguimiento (Tracker)

- Monitoriza el progreso del programador, toma acción si las cosas tienden a salir de su senda.
- Verifica el grado de acierto entre las estimaciones realizadas.
- Realiza el seguimiento del proceso de cada iteración.

Verificador (Tester)

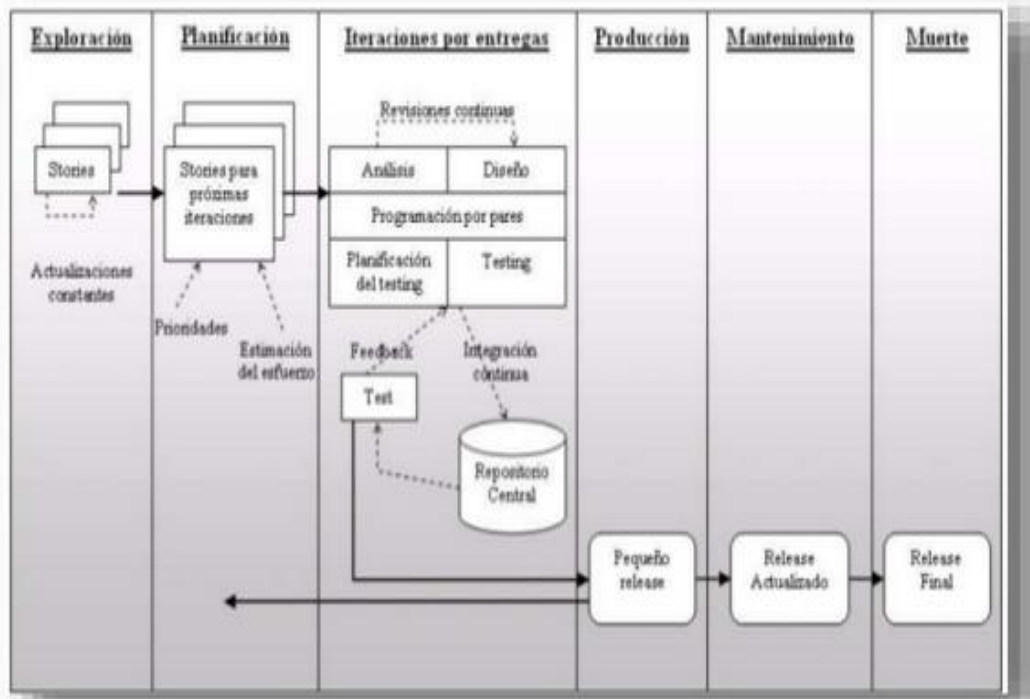
- Ayuda al cliente con las pruebas funcionales.
- Se asegura de que los tests funcionales se ejecutan

### 2.2.8.3 Características

Las principales características de esta metodología XP son las siguientes:

- **Comunicación:** Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.
- **Simplicidad:** Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se requiere mantenimiento o ampliación resulta imposible hacerlo y se tienen que desechar y partir de cero.
- **Realimentación (Feedback):** Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema adecuado a sus necesidades. Se le va mostrando el proyecto a tiempo para sugerir cambios y poder retroceder a una fase anterior para rediseñarlo a su gusto. [11]

- **Tenacidad:** Se debe ser tenaz para cumplir los tres puntos anteriores. Hay que tener valor para comunicarse con el cliente y enfatizar algunos puntos a pesar de que esto pueda dar sensación de ignorancia por parte del programador; hay que ser decidido para mantener un diseño simple y no optar por lo que pudiera parecer mejor o un camino más fácil y por último hay que enfatizar que la realimentación será efectiva.



*Figura N° 2.4 Ciclo XP*

*Fuente: [www.guiadeingenieria.com](http://www.guiadeingenieria.com)*

#### 2.2.8.4 Fases

##### 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.

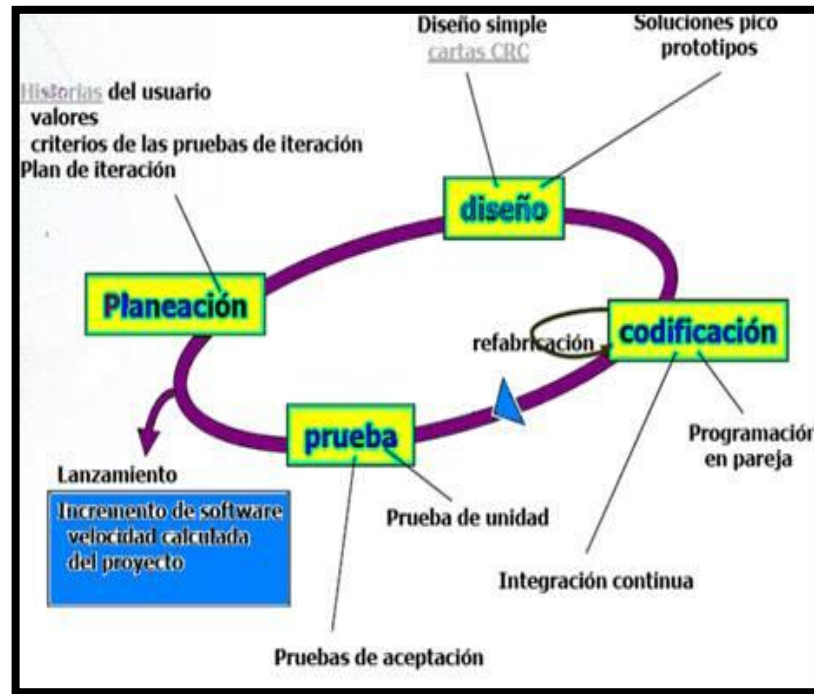
### **Planeamiento**

Se priorizan las historias de usuario y se acuerda el alcance de cada entrega. Los programadores estiman cuánto esfuerzo requiere cada historia y a partir de allí se define el cronograma. La duración del cronograma de la primera entrega no excede normalmente dos meses. La fase de planeamiento toma un par de días. El cronograma fijado en la etapa de planeamiento se realiza a un número de iteraciones, cada una toma de una a cuatro semanas en ejecución. La primera iteración crea un sistema con la arquitectura del sistema completo. Esto es alcanzado seleccionando las historias que harán cumplir la construcción de la estructura para el sistema completo. El cliente decide las historias que se seleccionarán para cada iteración. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. Al final de la última iteración el sistema está listo para producción.

### **Planificación de Entregas (Iteraciones)**

Es un espacio frecuente de comunicación 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. Esta práctica se puede ilustrar como un juego, donde existen dos tipos de jugadores: Cliente y Programador. El cliente establece la prioridad de cada historia de usuario, de acuerdo con el valor que aporta para el negocio. Los programadores estiman el esfuerzo asociado a cada historia de usuario. Se ordenan las historias de usuario según prioridad y esfuerzo, y se define el contenido de la entrega y/o iteración, apostando por enfrentar lo de más valor y riesgo cuanto antes. Este juego se realiza durante la planificación

de la entrega, en la planificación de cada iteración y cuando sea necesario reconducir el proyecto. [12]



*Figura N° 2.5: Planificación de Entrega*  
*Fuente: [www.metodologiasdedesarrollo.com](http://www.metodologiasdedesarrollo.com)*

- **Historias de usuario**

Las “Historias de usuarios” sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas,

debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

- **Plan de iteraciones**

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.

Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir.

- **Reuniones diarias de seguimiento**

El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones. En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar. Para no quitar tiempo innecesario del equipo, se sugiere realizar estas reuniones en círculo y de pie.

## **Diseño**

El diseño moldea la estructura que ordenará la lógica de la aplicación. Un correcto diseño brinda la posibilidad de que el sistema crezca con cambios en un solo lugar, lo hace extensible y reutilizable. Los diseños deben de ser sencillos, si alguna parte del sistema es de desarrollo complejo, lo apropiado es dividirla en varias partes.

Si hay fallas en el diseño o malos diseños, estas deben ser corregidas cuanto antes porque de lo contrario se verán plasmadas en el producto disminuyendo su calidad o en ocasiones, no cumpliendo los requerimientos para los cuales ha sido creado el producto.

- **Metáfora**

En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume en forma evolutiva. Los posibles inconvenientes que se generarán por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. Consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema.

Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema. Esta práctica simplemente nos dice que a la hora de abordar un problema complejo, debemos atacar al mismo estableciendo una analogía entre dicha problemática y una historia o situación que nos permita visualizarla y resolverla con una mayor facilidad.

- **Diseño simple**

Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente.

Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos.

### **Codificar**

El proceso de codificación se basa en plasmar las ideas y funcionalidades del sistema a través del código. En programación, el código expresa la interpretación del problema en términos de los programadores. De esta

forma podemos utilizar el código para comunicar, para hacer comunes las ideas y también para aprender y mejorar el nivel de los mismos recursos involucrados en el desarrollo del proyecto.

El código es el idioma de comunicación de los programadores. Es por ello que se recomienda que el mismo sea sencillo y legible para todos los integrantes del equipo. [13]

- **Refactorización (Refactoring)**

La refactorización es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios.

La refactorización mejora la estructura interna del código sin alterar su comportamiento externo. No se puede imponer todo en un inicio, pero en el transcurso del tiempo este diseño evoluciona conforme cambia la funcionalidad del sistema. Para mantener un diseño apropiado, es necesario realizar actividades de cuidado continuo durante el ciclo de vida del proyecto.

De hecho, este cuidado continuo sobre el diseño es incluso más importante que el diseño inicial. Un concepto pobre al inicio puede ser corregido con esta actividad continua, pero sin ella, un buen diseño inicial se degradará.

- **Programación en parejas**

Toda la producción de código debe realizarse con trabajo en parejas de programadores. XP promueve que todo el código sea escrito en parejas trabajando en el mismo ordenador. La programación en parejas incrementa la calidad del código sin impactar en la fecha de entrega.

En contra de lo que parece, dos personas que trabajan en un mismo equipo añadirán la misma funcionalidad que dos personas trabajando por separado, excepto que el código será de mucha mayor calidad.



- **Propiedad colectiva del código**

Esta práctica establece que cualquier programador puede tener acceso y cambiar cualquier parte del código en cualquier momento que así lo desee.

Esta práctica motiva a todos a contribuir con nuevas ideas en todos los segmentos del sistema, evitando a la vez que algún programador sea imprescindible para realizar cambios en alguna porción de código.

### **Pruebas**

La producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial.

### **Fase de Producción**

Requiere prueba y comprobación extra del funcionamiento del sistema antes de que éste se pueda liberar al cliente. En esta fase, los nuevos cambios pueden todavía ser encontrados y debe tomarse la decisión de si se incluyen o no en el release (iteración) actual. Durante esta fase, las iteraciones pueden ser aceleradas de una a tres semanas. Las ideas y las sugerencias postpuestas se documentan para una puesta en práctica posterior por ejemplo en la fase de mantenimiento. Después de que se realice el primer release productivo para uso del cliente, el proyecto de XP debe mantener el funcionamiento del sistema mientras que realiza nuevas iteraciones. [14]

### **Fase de Mantenimiento**

Requiere de un mayor esfuerzo para satisfacer también las tareas del cliente. Así, la velocidad del desarrollo puede desacelerar después de que el sistema esté en la producción. La fase de mantenimiento puede requerir la incorporación de nueva gente y cambiar la estructura del equipo.

### **2.2.8.5 PROCESO XP**

Un proyecto XP tiene éxito cuando el equipo de desarrollo cumple con todas las expectativas del cliente, es decir el producto final realiza todo aquello para lo que fue pensado.

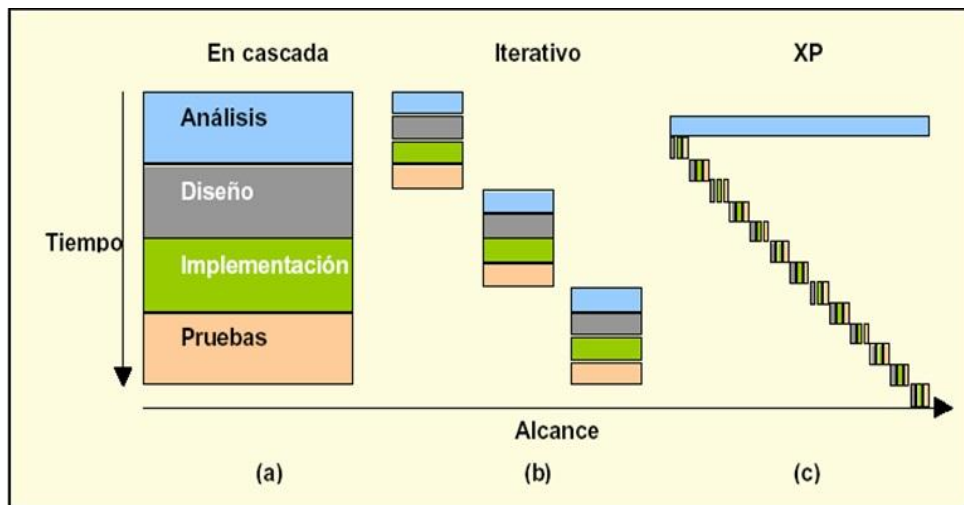
Para poder llevar a cabo esto, el cliente debe seleccionar el valor de negocio a implementar (funcionalidad) basándose en la habilidad del equipo para medir la funcionalidad que puede entregar dentro de un determinado período de tiempo. Es decir, el cliente prioriza las funcionalidades y las agrupa acorde a la capacidad del equipo y la importancia que éstas poseen para él.

El ciclo de desarrollo de una iteración 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 cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo que presenta el equipo junto con el entorno de desarrollo.
4. El programador construye ese valor de negocio.
5. Se vuelve al paso 1.

En todos los “release” o iteraciones tanto el programador como el cliente aprenden de sus errores. El desarrollador no debe ser presionado para hacer una mayor cantidad de trabajo que la estimada previamente, ya que no sólo se perderá calidad en el software o no se cumplirán los plazos; sino que también podría haber una desmotivación en el mismo y su rendimiento en las próximas iteraciones podría llegar a disminuir.

De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración. [15]



*Figura Nª 2.6 Modelo de desarrollo XP*  
*Fuente: [www.ingenierisoftware.obolog.es](http://www.ingenierisoftware.obolog.es)*

### 2.2.9 RUP (Rational Unified Process)

RUP es una metodología que tiene como objetivo ordenar y estructurar el desarrollo de software, en la cual se tienen un conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema Software (Amo, Martínez y Segovia, 2005). Inicialmente fue llamada UP (Unified Process) y luego cambió su nombre a RUP por el respaldo de Rational Software de IBM. Ésta metodología fue lanzada en 1998 teniendo como sus creadores a Ivar Jacobson, Grady Booch y James Rumbaugh. El RUP nació del UML (Unified Modeling Language) y del UP (Sommerville, 2005).

El Proceso Racional Unificado es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

### 2.2.9.1 Características del RUP

El RUP es un proceso basado en los modelos en Cascada y por Componentes, el cual presenta las siguientes características: Es dirigido por los casos de uso, es centrado en la arquitectura, iterativo e incremental (Booch, Rumbaugh y Jacobson, 2000), lo cual es fundamental para el proceso de desarrollo de software. A continuación se explican las tres características de RUP. [11]

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Originalmente se diseñó un proceso genérico y de dominio público, el Proceso Unificado, y una especificación más detallada, el Rational Unified Process, que se vendiera como producto independiente.

### 2.2.9.2 Fases

La duración de cada iteración puede extenderse desde dos semanas hasta seis meses. Las fases son:

**Iniciación.** Se especifican los objetivos del ciclo de vida del proyecto y las necesidades de cada participante. Esto entraña establecer el alcance y las condiciones de límite y los criterios de aceptabilidad. Se identifican los casos de uso que orientarán la funcionalidad.

**Elaboración.** Se analiza el dominio del problema y se define el plan del proyecto. Al cabo de esta fase, debe estar identificada la mayoría de los casos de uso y los actores, debe quedar descripta la arquitectura de software y se debe crear un prototipo de ella. Al final de la fase se realiza un análisis para determinar los riesgos y se evalúan los gastos hechos contra los originalmente planeados.

**Construcción.** Se desarrollan, integran y verifican todos los componentes y rasgos de la aplicación. Los resultados de esta fase (las versiones alfa, beta y otras versiones de prueba) se crean tan rápido como sea posible. Se debe

compilar también una versión de entrega. Es la fase más prolongada de todas.

**Transición.** La fase consiste en prueba beta, piloto, entrenamiento a usuarios y despacho del producto a mercadeo, distribución y ventas. Se produce también la documentación. Se llama transición porque se transfiere a las manos del usuario, pasando del entorno de desarrollo al de producción.



*Figura N° 2.7 Ciclo de RUP*

*Fuente: [www.cesarmontenegrofarias.blogspot.com](http://www.cesarmontenegrofarias.blogspot.com)*

### **2.2.9.3 Principios**

#### **Adaptar el proceso**

El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

#### **Equilibrar prioridades**

Los requerimientos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un

equilibrio que satisfaga los deseos de todos. Gracias a este equilibrio se podrán corregir desacuerdos que surjan en el futuro.

### **Demostrar valor iterativamente**

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

### **Colaboración entre equipo**

El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requisitos, desarrollo, evaluaciones, planes, resultados, etc.

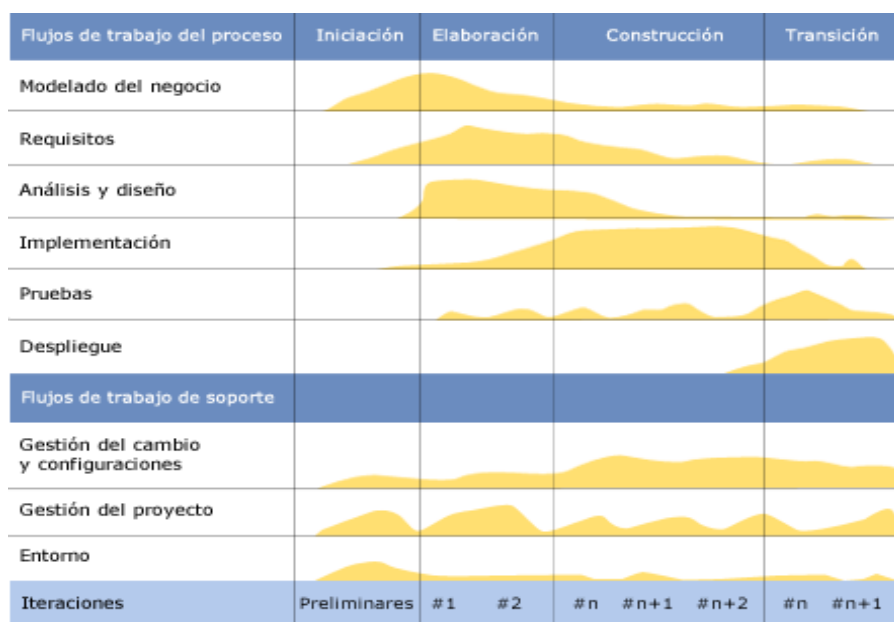
### **Elevar el nivel de abstracción:**

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, marcos de referencia (frameworks) por nombrar algunos.

Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requisitos y sin comenzar desde un principio pensando en la reutilización del código.

### **Enfocarse en la calidad:**

El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente. [16]



**Figura N° 2.8 Fases RUP**  
*Fuente: [www.guiametodologiasagies.com](http://www.guiametodologiasagies.com)*

### 2.3 PROPUESTA DE SOLUCIÓN

La presente investigación propone la Elaboración de una guía de aplicación de la metodología ágil seleccionada para la estandarización del desarrollo de software en la Pastoral en la ciudad de Ambato.

## CAPÍTULO III: METODOLOGÍA

### 3.1 MODALIDAD BÁSICA DE INVESTIGACIÓN

La presente investigación se contextualizará en la modalidad de campo y documental – bibliográfica.

De campo porque se realizará un estudio sistemático de los procesos en la Pastoral donde se producen los acontecimientos y documental-bibliográfico porque se buscará información en libros, informes, revistas porque se tiene como propósito detectar, profundizar y ampliar diferentes enfoques, teorías, conceptualizaciones y criterios en todo lo relacionado a sistemas de comunicación.

### 3.2 POBLACIÓN Y MUESTRA

Para la presente investigación, no se utilizará población y muestra pero se presenta un aporte del personal del departamento de sistemas.

<b>PUESTO</b>	<b>CANTIDAD</b>
JEFE DE SISTEMAS	1
MANTENIMIENTO	1
COMUNICACIONES	1
ANALISTA PROGRAMADOR	1
PROGRAMADOR	1
<b>TOTAL</b>	<b>5</b>

*Tabla N° 3.1 Personal de Dep. Sistemas Pastoral  
Realizado por: El Investigador*



### **3.3 RECOLECCIÓN DE INFORMACIÓN**

Para la recolección de información se hará uso de Guías de Observación, Análisis de Documentos y Entrevistas.

### **3.4 PROCESAMIENTO Y ANÁLISIS DE DATOS**

Se analizará y procesará la información obtenida en cuadros resumen, Gráficos, cuadros comparativos.

### **3.5 DESARROLLO DEL PROYECTO**

1. Identificación de procesos e información que se llevan a cabo.
2. Análisis y estudio de los procesos.
3. Documentación de procesos.
4. Estudio comparativo de las metodologías.
5. Selección de la metodología.
6. Desarrollo de los procesos de la metodología seleccionada.
7. Aplicación práctica de los procesos desarrollados.

## CAPÍTULO IV: LA PROPUESTA

### 4.1 IDENTIFICACIÓN DE PROCESOS

En la actualidad el desarrollo de software en la Pastoral no tiene ninguna metodología de desarrollo de software para el desarrollo de sistemas. Por ello se investigará el proceso de desarrollo de software que se ha estado realizando en el departamento de la Pastoral.

<b>PUESTO</b>	<b>DESCRIPCIÓN DE LAS FUNCIONES</b>
<b>Jefe del departamento de sistemas</b>	Comunicar los planes, objetivos, metas, políticas, normas y procedimientos al personal a su cargo.  Dirigir procesos de evaluación y cambios tecnológicos.  Evaluar sistemas y procesos.  Administrar los recursos que estén bajo su responsabilidad.
<b>PUESTO</b>	<b>DESCRIPCIÓN DE LAS FUNCIONES</b>
<b>Comunicaciones</b>	Controlar los enlaces y el monitoreo de los equipos.  Administrar servidores para el buen funcionamiento de la red en general.  Realizar respaldos de información de los equipos y servidores.  Control de acceso en los perfiles definidos.

<b>PUESTO</b>	<b>DESCRIPCIÓN DE LAS FUNCIONES</b>
<b>Mantenimiento</b>	<p>Realizar mantenimiento de los equipos.</p> <p>Brindar soporte técnico preventivo y correctivo a nivel de software y hardware</p> <p>Reportar e informar del estado de los equipos en forma periódica.</p> <p>Contactar con proveedor frente a fallas mayores de los equipos.</p> <p>Coordinar la instalación y configuración de los equipos informáticos,</p> <p>Coordinar la atención y resolución de problemas y requerimientos.</p>

<b>PUESTO</b>	<b>DESCRIPCIÓN DE LAS FUNCIONES</b>
<b>Analista</b>	<p>Analizar un sistema ya existente para comprender, mejorar, ajustar su comportamiento.</p> <p>Realiza el análisis para ver lo que se quiere hacer inicialmente y después darle al usuario nuevas opciones de uso.</p> <p>Describir todos los procesos y transmite al programador.</p>

<b>PUESTO</b>	<b>DESCRIPCIÓN DE LAS FUNCIONES</b>
<b>Programador</b>	Programar los diseños propuestos por analista Capacitar sobre el uso de la tecnologías Interpretar especificaciones de diseños de los sistemas. Verificar los componentes programados. Corrección de errores de compilación, ejecución.

#### **4.1.1 Análisis del Sistema ASES de la Pastoral**

En este punto se identifica todo el proceso y los pasos que han utilizado para el desarrollo de sistemas de la Pastoral tomando como caso de estudio, el sistema ASES y será utilizado en la investigación.

En la primera parte del desarrollo el analista empieza recolectando información de los problemas que tiene la Pastoral y se analiza los procesos, el alcance del proyecto, una estimación del tiempo y los recursos a utilizar.

Se analiza los procesos que tienen definidos y los alcances funcionales del sistema informático que se implementara como parte de la ejecución del proyecto.

Se hace varias reuniones con el asesor de crédito para recolectar información de todos los procesos que se realiza en el departamento de la Pastoral.

#### **4.1.2 Procesos**

ASES Básicamente maneja la cuenta interna de los bancos comunales, el sistema genera préstamos de los fondos de los socios.

Además se conecta al sistema financiero COBIS, y extrae las deudas de cada persona presentando listados consolidados de deudas tanto, el sistema ASES genera préstamos de los fondos de los socios.

Existen dos tipos de préstamos el uno normal y el otro extraordinario, presenta también balance y estado de resultado del movimiento económico del grupo, solicitud de crédito, resumen de ahorros.

El sistema controla las tasas de interés, utilidades, regularización de ciclos y genera balances.

Los clientes aperturan sus cuentas en la Pastoral iniciando con un grupo de 6 de personas como mínimo, siendo registrada una cuenta para el grupo y una cuenta individual. [ANEXO C]

Las solicitudes de crédito se las hacen con una reunión previa donde cada una de las personas llena una solicitud con el monto que desee y se nombra una directiva la cual es el representante legal que responde a la Pastoral por el crédito del grupo. [ANEXO D]

Los movimientos de las cuentas internas se los realiza después de cada reunión q con su grupo y directiva respectivamente si sea que vaya a realizar un depósito, retiro o pago de préstamo. [ANEXO E]

Maneja una cuenta interna de ahorros con la diferencia de que los movimientos individuales afectan a una cuenta grupal que es el resumen de todos los integrantes del grupo. [ANEXO F]

### 4.1.3 Módulos de Sistema

A continuación se presentan los módulos del sistema de la Pastoral del análisis anterior.

<b>Módulos</b>	<b>Nro.</b>	<b>Descripción</b>
<b>1 Administración</b>	01	Registrar clientes
	02	Registrar asesores
	03	Registrar usuarios
	04	Registrar grupos
	05	Registrar cuenta individual
	06	Registrar cuenta grupal
	07	Registrar carnet
<b>2 Préstamos</b>	08	Registrar solicitud
	09	Verificar datos del cliente
	10	Autorizar solicitud
	11	Generar ciclos
<b>3 Movimientos</b>	12	Automatizar el flujo de ahorros
	13	Controlar deudas consolidadas
	14	Controlar movimientos de la cuenta interna
	15	Controlar movimientos de cuenta grupal
	16	Controlar utilidades
<b>4 Reportes</b>	17	Reporte por grupo
	18	Reporte por cliente
	19	Generar Balance
	20	Generar estados de resultados

*Tabla N<sup>o</sup> 4.1 Módulos del Sist. ASES  
Realizado por: El Investigador*

### 4.1.4 Estimación de Tiempos

Para el desarrollo del presente proyecto se tomó un tiempo de 14 meses, 1 mes de 4 semanas, 1 semana de 5 días y 1 día de 3 horas.

Módulos	Nro.	Tareas	Tiempo Estimado		
			Semanas	Días	Horas
<b>Módulo 1</b>	01	Registrar clientes	1,6	8	24
	02	Registrar asesores	1,6	8	24
	03	Registrar usuarios	1,6	8	24
	04	Registrar grupos	2	10	30
	05	Registrar cuenta individual	1,6	8	24
	06	Registrar cuenta grupal	2,4	12	36
	07	Registrar carnet	1,6	8	24
<b>Módulo 2</b>	08	Registrar solicitud	3	15	45
	09	Verificar datos del cliente	3	15	45
	10	Autorizar solicitud	4	20	60
	11	Generar ciclos	4	20	60
<b>Módulo 3</b>	12	Automatizar el flujo de ahorros	5	25	75
	13	Controlar deudas consolidadas	5	25	75
	14	Controlar movimientos de la cuenta interna	5	25	75
	15	Controlar movimientos de cuenta grupal	6	30	90
	16	Controlar utilidades	3	15	45
<b>Módulo 4</b>	17	Reporte por grupo	3	15	45
	18	Reporte por cliente	2	10	30
	19	Generar Balance	5	25	75
	20	Generar estados de resultados	5	25	75
			56,4	327	981

**Tabla N<sup>o</sup> 4.2 Estimación de tiempos del Sist. ASES**  
**Realizado por: El Investigador**

#### **4.1.5 Tiempo real invertido**

Al inicio del proyecto se elabora un plan de trabajo y se estima tiempos pero en muchas ocasiones el tiempo invertido es diferente al tiempo de planificación.

	Nro.	Tareas	Esfuerzo estimado (semanas)	Esfuerzo real invertido	Días reales
<b>Módulo 1</b>	01	Registrar clientes	1,6	2	10
	02	Registrar asesores	1,6	2	10
	03	Registrar usuarios	1,6	2	10
	04	Registrar grupos	2	2,2	11
	05	Registrar cuenta individual	1,6	2,2	11
	06	Registrar cuenta grupal	2,4	2,8	14
	07	Registrar carnet	1,6	1,4	7
<b>Módulo 2</b>	08	Registrar solicitud	3	4	20
	09	Verificar datos del cliente	3	3,4	17
	10	Autorizar solicitud	4	4,6	23
	11	Generar ciclos	4	5	25
<b>Módulo 3</b>	12	Automatizar el flujo de ahorros	5	5,4	27
	13	Controlar deudas consolidadas	5	5	25
	14	Controlar movimientos de la cuenta interna	5	5,4	27
	15	Controlar movimientos de cuenta grupal	6	7	35
	16	Controlar utilidades	3	3,8	19
<b>Módulo 4</b>	17	Reporte por grupo	3	3,4	17
	18	Reporte por cliente	2	2,8	14
	19	Generar Balance	5	5,2	26
	20	Generar estados de resultados	5	5,4	27
			56,4	75,1	375

**Tabla N<sup>o</sup> 4.3 Tiempo total invertido**  
**Realizado por: El Investigador**



#### 4.1.6 Burn Down

Primero se considera que si la curva de evolución del proyecto está por encima de los días ideales planificados, el proyecto está retrasado. Y si está por debajo se considera un sobre estimación del proyecto.

**Gráfico del Burn Down: Esfuerzo Estimado Vs Esfuerzo Invertido**



*Figura N<sup>o</sup> 4.1 Gráfico del Burn Down  
Realizado por: El Investigador*

Se observa en el gráfico que la curva está por encima del tiempo estimado es decir vemos un retraso desde la primera iteración.

En la segunda iteración se aleja aún más del tiempo estimado y observamos que el esfuerzo invertido es mayor que el esfuerzo estimado.

Se muestra una sobrecarga de trabajo en la tercera iteración y culmina el proyecto con un retraso de 48 días.

La velocidad del proyecto es menor al tiempo previsto.

Según el jefe del departamento de sistemas de la Pastoral indicó que hubo problemas de comunicación entre los miembros del equipo, retraso en cuanto a planificación de los módulos, los requisitos no estuvieron claros y se desarrolló en un tiempo mayor al estimado.

## **4.2 ESTUDIO COMPARATIVO DE LAS METODOLOGÍAS**

En este punto se va a comparar las metodologías, seleccionando las más importantes para objeto de nuestro estudio, conociendo los requerimientos que tenga el desarrollo de software en la Pastoral.

Se han elaborado cuadros comparativos con las metodologías escogidas donde se permitirá conocer la metodología ágil que mejor se adapta al marco de referencia de la Pastoral.

### **4.2.1 Metodologías Ágiles**

Se han seleccionado tres enfoques de metodologías ágiles: SCRUM, XP y RUP que han tenido más uso y reconocimiento en la industria del desarrollo de software. Además son las más populares por el éxito en los proyectos que han sido empleadas.




## **4.3 ELECCIÓN DE LA METODOLOGÍA**

Según el estudio realizado de cada metodología se puede extraer características de cada una de ellas y poder analizar con criterios de comparación cual es la que mejor se adapta a la Pastoral.

### **4.3.1 Características de las metodologías ágiles**

#### **Metodología SCRUM**

##### **Características**

-  Se orienta más a las personas que a los procesos.
-  Acepta requisitos cambiantes.
-  Basado en iteraciones y revisiones.

- ✚ Enfocado a conseguir pequeños incrementos de software.
- ✚ Su prioridad es la satisfacción al cliente

### **Ventajas**

- ✚ Incremento de la productividad.
- ✚ Mejoras Constantes.
- ✚ Existe una gran comunicación en el equipo.
- ✚ El cliente siempre está presente en cada mejora del producto.
- ✚ Entrega de un producto funcional después de cada sprint.
- ✚ Capacidad de aceptar modificaciones.
- ✚ Trabaja con iteraciones cortas.
- ✚ Equipos integrados y comprometidos.

### **Desventajas**

- ✚ Dependencia de las personas.
- ✚ No es apto para todos los proyectos.
- ✚ Delega responsabilidades fijas al equipo de trabajo.
- ✚ Problemas con las fechas de entrega bien restringidas mediante un contrato.

### **Metodología XP (Extremen Pogramming)**

#### **Características**

- ✚ Basada en valores y prácticas.
- ✚ Metodología liviana.
- ✚ Costos reducidos y altos estándares de calidad.

#### **Ventajas**

- ✚ Comunicación entre el cliente y los programadores.
- ✚ Facilidad a cambios .
- ✚ Pruebas continuas durante todo el proceso.
- ✚ Programación organizada.
- ✚ Menor tasa de errores.

- ✚ Satisfacción del programador.
- ✚ El cliente tiene el control de las prioridades.
- ✚ Código sencillo y entendible.
- ✚ Programación en parejas.

### **Desventajas**

- ✚ Es recomendable utilizar en proyectos pequeños.
- ✚ No define costo ni tiempo de desarrollo.

## **Metodología RUP (Rational Unified Process)**

### **Características**

- ✚ Se caracteriza por ser incremental.
- ✚ Usa un enfoque iterativo.
- ✚ Dirigida por casos de uso.
- ✚ Centrado en la arquitectura.
- ✚ Programación por equipos.

### **Ventajas**

- ✚ Asigna tareas de forma disciplinada.
- ✚ Proceso de software configurable.
- ✚ Configuración y control de cambios.
- ✚ Verifica la calidad de software.
- ✚ Mayor documentación.

### **Desventajas**

- ✚ Método pesado.
- ✚ Grado de complejidad.
- ✚ Proyectos grandes.

### 4.3.2 Comparación de Metodologías

**Tabla Comparativa de las Metodologías Ágiles**

Características	METODOLOGÍAS ÁGILES		
	SCRUM	XP	RUP
Tamaño de los proyectos	Pequeños medianos y grandes	Pequeños y medianos	Grandes
Complejidad del proyecto	Alta	Baja	Alta
Tamaño del equipo	Grande	Pequeño, mediano	Grande
Estilo de desarrollo	Iterativo y rápido	Iterativo y rápido	Iterativo
Documentación	Simplificada	Simplificada	Extensa
Comunicación con el cliente	En todo el desarrollo	En todo el desarrollo	En el inicio
Resultados	Rápidos	Rápidos	No muy rápidos
Respuesta a cambios	Alta	Alta	No
Cumplimiento de los requisitos	Se cumple	Se cumple	Se cumple
El respeto de un nivel de calidad	Alto	Alto	Alto
Satisfacción del usuario final	Alto	Alto	Alto
Aumento de la productividad	Medio	Alto	Medio
Iteraciones cortas	Si	Si	No
Colaboración	Si	Si	Si
Integración de los cambios	Si	Si	No
Los requisitos funcionales pueden cambiar	No siempre	Siempre	Sin respuesta

Definición de requisitos.	Lo más útil	Los más útil	Lo más útil
Modelado.	Simplificado	Simplificado	Extenso
Código.	Normal	Sencillo	Normal
Pruebas unitarias.	No constantes	Constantemente	Sin respuesta
Prueba de aceptación.	No	Si	No

*Tabla N<sup>a</sup> 4.4 Matriz comparativa de metodologías  
Realizado por: El Investigador*

Según Kent Beck el creador de la programación extrema dice que es una metodología adecuada para proyectos medianos y pequeños donde los equipos de desarrollo son más 3 y menor a 10.

#### **4.3.2.1 Criterios de Comparación**

Los criterios que se va a utilizar para analizar las metodologías SCRUM, XP y RUP son los siguientes: Ciclo de vida proyecto, planificación, calidad, procesos y herramientas, los mismos que ha sido tomados como base del estudio de la investigación “Herramientas y modelo de desarrollo, metodologías Ágiles” [20] realizada por el MsC. Carlos Carvajal, y adaptada por la autora de esta tesis.

#### **CICLO DE VIDA DEL PROYECTO**

Se consideran las fases, actividades, prácticas de cada metodología como parte de ciclo de vida de un proyecto.

**Fases.-** El proceso de desarrollo de software, que contiene las actividades y las tareas involucradas en la realización del proyecto.

**Actividades.-** Conjunto de acciones planificadas que consisten en la ejecución de procesos que se lleva acabo para crear un producto.

**Prácticas.-** Acciones tangibles utilizadas para mejorar el desarrollo del proyecto.

**Artefactos.-** Instrumentos empleados para comprender el análisis del desarrollo del proyecto.

## PLANIFICACIÓN

Se considera la forma de organización de cada metodología y como son distribuidos los roles en el proyecto además los valores que promueven.

**Roles.-** Distribución de funciones entre los miembros del proyecto.

**Reuniones.-** Junta de los miembros para planificar, discutir y analizar los problemas en el desarrollo del proyecto.

**Valores.-** Promover a cada miembro del equipo ciertos valores en su comportamiento con el propósito de llevar con éxito el proyecto.

## CALIDAD

Se analiza si cada metodología cumple con ciertos parámetros de calidad.

**Fiabilidad.-** Simplicidad en los diseños y en cada etapa de desarrollo del proyecto.

**Usabilidad.-** Se trata de evaluar los beneficios que el equipo de desarrollo y el cliente obtienen utilizando este tipo de metodología.

**Mantenibilidad.-** Si la metodología promueve que los proyectos desarrollados sean simples al momento de mantenerse.

**Aplicabilidad.-** Se considera cuando es un entorno favorable para la aplicación de las metodologías ágiles.

## HERRAMIENTAS

Se da a conocer las distintas herramientas que la metodología posee para cumplir las tareas necesarias en el desarrollo.

**Gestión.-** Se describe como un proceso de planeamiento, ejecución de un proyecto.

**Medición, Seguimiento.-** La forma de llevar un control de lo que se está realizando en el proyecto.

**Refactorización.-** Modificación del código sin cambiar su comportamiento para mejorar la facilidad de comprensión.

### **Escala de Evaluación**

Se evalúa las tres metodologías en base a cuatro criterios ya mencionados y así obtener resultados que nos permitan escoger la metodología para el desarrollo de software de la Pastoral.

Variable	Equivalencia	Puntaje
EM	Evaluación Mala	0
ER	Evaluación Regular	1
EB	Evaluación Bueno	2
EMB	Evaluación Muy Bueno	3
EE	Evaluación Excelente	4

*Tabla N° 4.5 Escala de evaluación de las metodologías  
Realizado por: El Investigador*

La calificación definitiva de la metodología en base a cada criterio del análisis de obtiene sumando los puntajes obtenidos utilizando las siguientes fórmulas.

$$PT_{scrum} = \Sigma(P)$$

$$PT_{xp} = \Sigma(P)$$

$$PT_{rup} = \Sigma(P)$$

**Dónde:**

**P=** Es el valor de cada parámetro

**PT<sub>scrum</sub>**= Puntaje total de Scrum

**PT<sub>xp</sub>**= Puntaje total de XP

**PT<sub>rup</sub>**= Puntaje total de RUP



#### 4.3.2.1.1 CICLO DE VIDA DEL PROYECTO

N <sup>a</sup>	Parámetros	Metodología SCRUM	Puntaje
1	Fases	Compuesta de tres fases: Pre- Juego, Juego, Post-Juego	4
2	Actividades	Elaborar, integrar, ajustar, dirigir	3
3	Prácticas	Centrada un poco más en la gestión de proyectos	3
4	Artefactos	Propone cinco artefactos: historias de usuario, pila del producto, pila del sprint, Burn Down Burn Up para mantener organizados los proyectos.	4

*Tabla N<sup>a</sup> 4.6 Criterio “Ciclo de vida del proyecto” Metodología SCRUM  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología XP	Puntaje( PP)
1	Fases	Compuesta de tres fases: Exploración planificación, iteración, producción, mantenimiento.	4
2	Actividades	Escuchar, diseñar, codificar, hacer pruebas.	4
3	Prácticas	Equipo completo, diseño incremental, historias de usuario, integración continua código compartido, reuniones diarias, entregas pequeñas, desarrollo en parejas.	3
4	Artefactos	Propone cinco artefactos: historias de usuario, pila del producto, pila del sprint, Burn Down Burn Up para mantener organizados los proyectos.	4

*Tabla N<sup>a</sup> 4.7 Criterio “Ciclo de vida del proyecto” Metodología XP  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología RUP	Puntaje(PP)
1	Fases	Compuesta de tres fases: Inicio, elaboración, construcción, transición.	4
2	Actividades	Análisis, diseño, modelado de negocio, despliegue	3
3	Prácticas	Control de cambios, desarrollo iterativo, Arquitectura basada en componentes.	3
4	Artefactos	Casos de uso del sistema, especificación de requisitos, diagrama de implementación.	3

*Tabla N<sup>a</sup> 4.8 Criterio “Ciclo de vida del proyecto” Metodología RUP  
Realizado por: El Investigador*

### Interpretación de Resultados

PM= puntuación máxima

PM= 16

Parámetros	Metodologías		
	SCRUM	XP	RUP
Fases	4	4	4
Actividades	3	4	3
Prácticas	3	3	3
Artefactos	4	4	3
TOTAL Σ(PP)	14	15	13
Porcentaje	87,5 %	93,8 %	81,3 %

*Tabla N<sup>a</sup> 4.9 Resultado del Criterio “Ciclo de vida del proyecto”  
Realizado por: El Investigador*

$$P_{Scrum} = (PT_{scrum}/PM) * 100\%$$

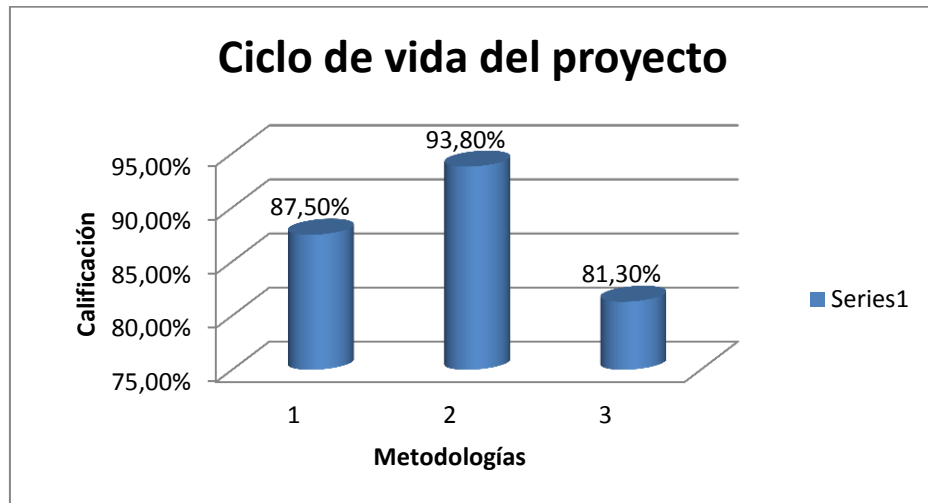
$$= (14/16) * 100\% = 87,5 \%$$

$$P_{XP} = (PT_{xp}/PM) * 100\%$$

$$= (15/16) * 100\% = 93,8 \%$$

$$PRUP = (P_{Trup}/PM) * 100\%$$

$$= (13/16) * 100\% = 81,3 \%$$



*Figura N<sup>a</sup> 4.2 Criterio “Ciclo de vida del proyecto”  
Realizado por: El Investigador*

De acuerdo con los puntajes alcanzados para cada uno de los parámetros de evaluación se puede concluir que XP es la que posee un mejor ciclo de vida para el desarrollo de Software de la Pastoral, alcanzando un porcentaje de 93,8 % seguida por Scrum con un porcentaje de 87,5 % y al final RUP con 81,3 %.

#### 4.3.2.1.2 PLANIFICACIÓN

N <sup>a</sup>	Parámetros	Metodología SCRUM	Puntaje(PP)
1	<b>Roles</b>	Posee los siguientes roles: dueño del producto, Scrum master, equipo scrum.	4
2	<b>Reuniones</b>	Se realizan reuniones diarias, al iniciar cada sprint (plannig meeting) y finalizar cada sprint (sprint review), (sprint retrospective).	3
3	<b>Valores</b>	Fundamenta el compromiso, enfoque auto-organización, coraje respeto y honestidad.	3

*Tabla N<sup>a</sup> 4.10 Criterio “Planificación” Metodología SCRUM  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología XP	Puntaje(P P)
1	<b>Roles</b>	Posee los siguientes roles: Programador, entrenador, tracker, tester(verificador), customer	3
2	<b>Reuniones</b>	Se realizan reuniones diarias, al iniciar cada sprint (daily stand up meeting), al planificar (planning game) y al inicio de cada iteración (iteration meeting).	4
3	<b>Valores</b>	Fundamenta la simplicidad, retroalimentación, comunicación y coraje.	3

*Tabla N<sup>o</sup> 4.11 Criterio “Planificación” Metodología XP  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología RUP	Puntaje(PP)
1	<b>Roles</b>	Posee los siguientes roles: Test manager, test analista, designer, Tester. Coordinador, especialista.	3
	<b>Reuniones</b>	Hace reuniones periódicas.	3
3	<b>Valores</b>	Fundamenta el compromiso, enfoque auto-organización.	3

*Tabla N<sup>o</sup> 4.12 Criterio “Planificación” Metodología RUP  
Realizado por: El Investigador*

### Interpretación de Resultados

PM= puntuación máxima

PM= 12

Parámetros	Metodologías		
	SCRUM	XP	RUP
<b>Roles</b>	4	3	3
<b>Reuniones</b>	3	4	3
<b>Valores</b>	3	4	3
<b>TOTAL <math>\Sigma</math>(PP)</b>	10	11	9
<b>Porcentaje</b>	83,3 %	91,7%	75 %

**Tabla N° 4.13 Resultado del Criterio “Planificación”**  
**Realizado por: El Investigador**

$$P_{Scrum} = (PT_{scrum}/PM) * 100\%$$

$$= (10/12) * 100\%$$

$$= 83,3 \%$$

$$P_{XP} = (PT_{xp}/PM) * 100\%$$

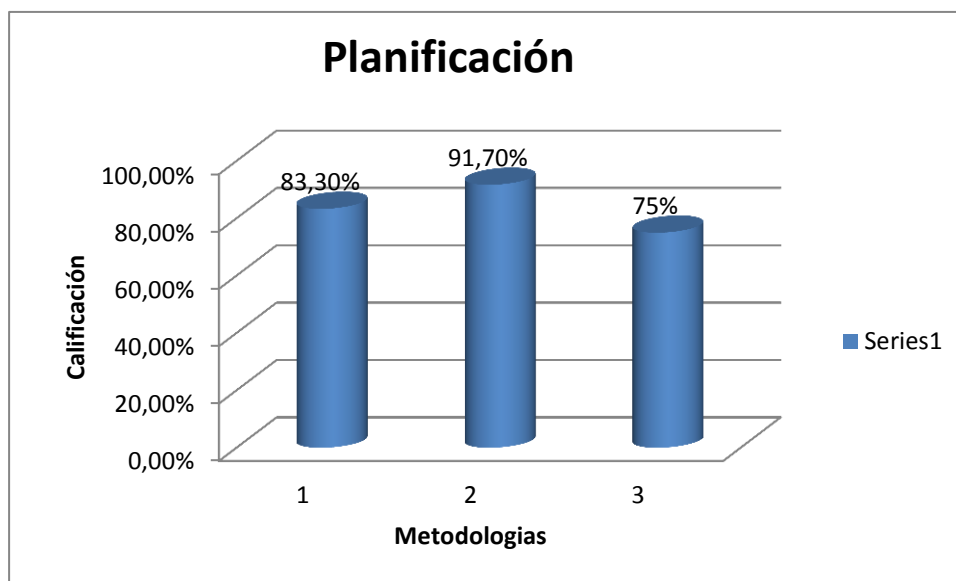
$$= (11/12) * 100\%$$

$$= 91,7 \%$$

$$P_{RUP} = (PT_{rup}/PM) * 100\%$$

$$= (9/12) * 100\%$$

$$= 75\%$$



*Figura N° 4.3 Criterio “Planificación”  
Realizado por: El Investigador*

De acuerdo con los puntajes alcanzados para cada uno de los parámetros de evaluación se puede concluir que Scrum es la que posee un mejor ciclo de vida para el desarrollo de Software de la Pastoral, alcanzando un porcentaje de 91,7 % seguida por XP con un porcentaje de 83,3 % y al final RUP con 75 %.

#### 4.3.2.1.3 CALIDAD

N <sup>a</sup>	Parámetros	Metodología SCRUM	Puntaje(PP)
1	<b>Fiabilidad</b>	Simplicidad de los diseños.	4
2	<b>Usabilidad</b>	Evita la burocracia y generación documental	3
3	<b>Mantenibilidad</b>	No promueve la mantenibilidad.	3
4	<b>Aplicabilidad</b>	Baja complejidad de proyectos, alta iteración con el cliente, equipo grande.	4

*Tabla N° 4.14 Criterio “Calidad” Metodología SCRUM  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología XP	Puntaje(PP)
1	<b>Fiabilidad</b>	Simplicidad en el código y en los diseños.	4
2	<b>Usabilidad</b>	Evita la burocracia y generación documental.	3
3	<b>Mantenibilidad</b>	Evita mayor número de líneas y duplicación de código.	4
4	<b>Aplicabilidad</b>	Baja complejidad de proyectos, alta iteración con el cliente, equipos, pequeños.	4

*Tabla N<sup>a</sup> 4.15 Criterio “Calidad” Metodología XP  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología RUP	Puntaje(PP)
1	<b>Fiabilidad</b>	Complejidad en los diseños.	3
2	<b>Usabilidad</b>	Extenso proceso de documentación	3
3	<b>Mantenibilidad</b>	No promueve la mantenibilidad.	3
4	<b>Aplicabilidad</b>	Alta complejidad de proyectos, alta iteración con el cliente, equipo grande.	3

*Tabla N<sup>a</sup> 4.16 Criterio “Calidad” Metodología RUP  
Realizado por: El Investigador*

### Interpretación de Resultados

PM= puntuación máxima

PM= 16

Parámetros	Metodologías		
	SCRUM	XP	RUP
<b>Fiabilidad</b>	4	4	3
<b>Usabilidad</b>	3	3	3
<b>Mantenibilidad</b>	3	4	3
<b>Aplicabilidad</b>	4	4	3
<b>TOTAL <math>\Sigma</math>(PP)</b>	14	15	12
<b>Porcentaje</b>	87,5 %	93,8 %	81,3 %

*Tabla N<sup>a</sup> 4.17 Resultado del Criterio “Calidad”  
Realizado por: El Investigador*

$$PScrum = (PTscrum/PM) * 100\%$$

$$= (14/16) * 100\%$$

$$= 87,5 \%$$

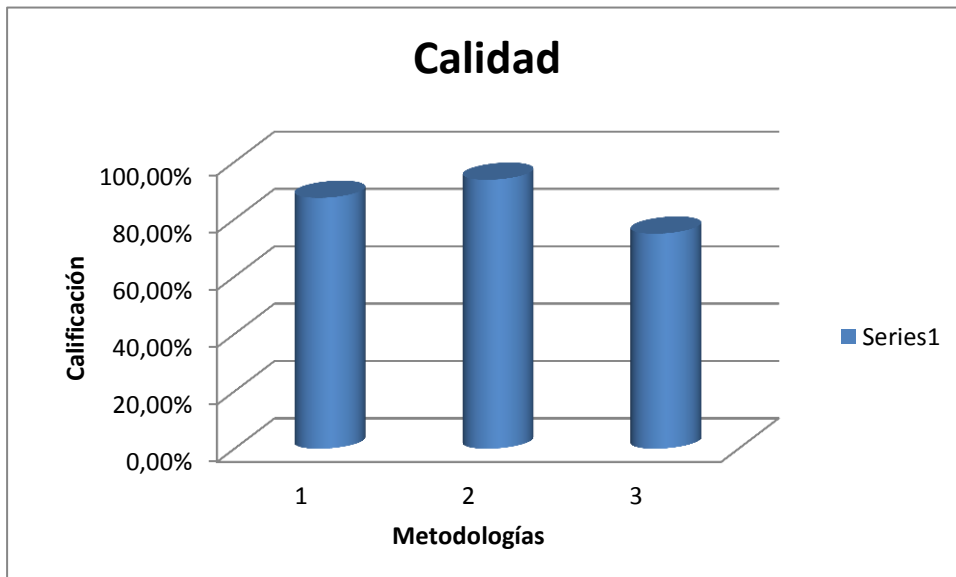
$$PXP = (PTxp/PM) * 100\%$$

$$= (15/16) * 100\% = 93,8 \%$$

$$PRUP = (PTRup/PM) * 100\%$$

$$= (12/16) * 100\%$$

$$= 75 \%$$



**Figura N° 4.4 Criterio "Calidad"**

**Realizado por: El Investigador**

De acuerdo con los puntajes alcanzados para cada uno de los parámetros de evaluación se puede concluir que XP es la que posee un mejor calidad para el desarrollo de Software de la Pastoral, alcanzando un porcentaje de 93,8 % seguida por Scrum con un porcentaje de 87,5 % y al final RUP con 75 %.



#### 4.3.2.1.4 HERRAMIENTAS

N <sup>a</sup>	Parámetros	Metodología SCRUM	Puntaje(PP)
1	Gestión	Herramientas: Sprintometer, Scrum Desk.	4
2	Medición y Seguimiento	Scrum Timer, Pango Scrum	4
3	Refactorización	No se ha encontrado alguna herramienta que utilice la metodología.	2

*Tabla N<sup>a</sup> 4.18 Criterio “Herramientas” Metodología SCRUM  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología XP	Puntaje(PP)
1	Gestión	Herramientas: Maven, Sprintometer, Ant.	4
2	Medición y Seguimiento	Herramientas: JMeter	3
3	Refactorización	Utiliza IDEs como Eclipse y NetBeans.	4

*Tabla N<sup>a</sup> 4.19 Criterio “Herramientas” Metodología XP  
Realizado por: El Investigador*

N <sup>a</sup>	Parámetros	Metodología RUP	Puntaje(PP)
1	Gestión	Herramientas: Rational Rose, Rational ClearCase.	4
2	Medición y Seguimiento	Rational RequisitePro	3
3	Refactorización	No se ha encontrado alguna herramienta que utilice la metodología.	2

*Tabla N<sup>a</sup> 4.20 Criterio “Herramientas” Metodología RUP  
Realizado por: El Investigador*

## Interpretación de Resultados

PM= puntuación máxima

PM= 12

Parámetros	Metodologías		
	SCRUM	XP	RUP
<b>Gestión</b>	4	4	4
<b>Medición y Seguimiento</b>	4	3	3
<b>Refactorización</b>	2	4	2
<b>TOTAL Σ(PP)</b>	10	11	9
<b>Porcentaje</b>	83,3 %	91,7 %	75 %

*Tabla N<sup>o</sup> 4.21 Resultado del Criterio “Herramientas”  
Realizado por: El Investigador*

$$P_{Scrum} = (PT_{scrum}/PM) * 100\%$$

$$= (10/12) * 100\%$$

$$= 83,3 \%$$

$$P_{XP} = (PT_{xp}/PM) * 100\%$$

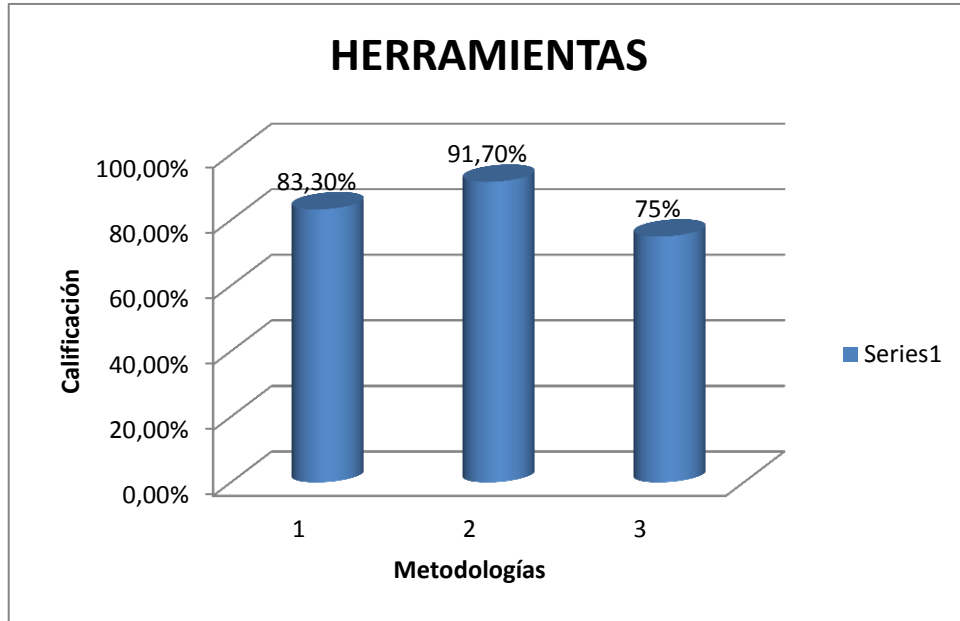
$$= (11/12) * 100\%$$

$$= 91,7 \%$$

$$P_{RUP} = (PT_{rup}/PM) * 100\%$$

$$= (9/12) * 100\%$$

$$= 75\%$$



*Figura N° 4.5 Criterio “Herramientas”*

*Realizado por: El Investigador*

De acuerdo con los puntajes alcanzados para cada uno de los parámetros de evaluación se puede concluir que XP es la que posee mejores herramientas para el desarrollo de Software de la Pastoral, alcanzando un porcentaje de 91,7 % seguida por Scrum con un porcentaje de 83,3 % y al final RUP con 75 %.

#### 4.3.2.1.5 Matriz General de Resultados

PARÁMETRO	CRITERIO	Metodologías Agiles		
		SCRUM	XP	RUP
CICLO DE VIDA DEL PROYECTO	Fases	4	4	4
	Actividades	3	4	3
	Practicas	3	3	3
	Artefactos	4	4	3
PLANIFICACIÓN	Roles	4	3	3
	Reuniones	3	4	3
	Valores	3	4	3
CALIDAD	Fiabilidad	4	4	3

	Usabilidad	3	3	3
	Mantenibilidad	3	4	3
	Aplicabilidad	4	4	3
HERRAMIENTAS	Gestión	4	4	4
	Medición y Seguimiento	4	3	3
	Refactorización	2	4	2
	TOTAL	48	52	43

*Tabla N° 4.22 Matriz General de Resultados  
Realizado por: El Investigador*

### **Puntaje Total del Análisis**

PT=56

### **Puntaje Total de Scrum**

$PT_{Scrum} = (\sum(PTA_{Scrum}/PT) * 100\%)$

$= (48/56) * 100\%$

$= 85,7\%$

### **Puntaje Total de XP**

$PT_{Xp} = (\sum(PTA_{xp}/PT) * 100\%)$

$= (52/56) * 100\%$

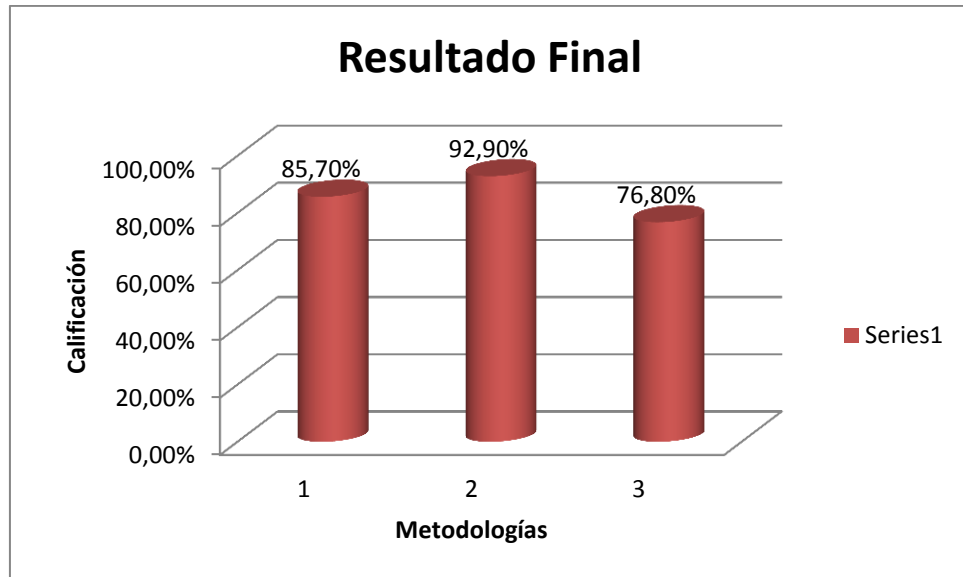
$= 92,9\%$

### **Puntaje Total de RUP**

$PT_{Rup} = (\sum(PTA_{rup}/PT) * 100\%)$

$= (43/56) * 100\%$

$= 76,8\%$



**Figura N° 4.6 Resultado Final**  
**Realizado por: El Investigador**

Después de haber comparado las tres metodologías en base a los criterios generales vemos que las tres son se basan en desarrollo iterativo incremental y las tres son muy buenas; SCRUM con un porcentaje final de 85,7 % y RUP con 76,8 % pero en este caso sobresale XP con un porcentaje final de 92,9% por su capacidad de agilidad, enfocándose en el código y asegurando la calidad del proyecto.

#### **4.3.2.2 ADAPTACIÓN DE LA METODOLOGÍA XP**

A continuación se realiza adaptación de la metodología XP en los sistemas de la Pastoral en base al análisis anteriormente realizado.

XP facilitará el desarrollo de sistemas aplicando las prácticas y valores que posee la metodología.

##### **4.3.2.2.1 Valores**

Simplicidad.- El diseño debe ser sencillo, solo se deben crearse diagramas útiles y se puede optimizar el código y haciéndolo más claro y simple.

Comunicación.- La comunicación debe ser permanente entre los miembros del equipo a lo largo de todo el proyecto.

Retroalimentación.- Esta ligada con la comunicación y es la mejor manera de conocer el estado actual del proyecto haciendo pruebas funcionales, esto proporcionara información real.

Respeto.- Las personas tenemos diferentes formas de pensar hay que trabajar respetando los diferentes criterios de los miembros del equipo.

#### **4.3.2.2.2 Prácticas**

Historias de usuario.- Se trata de una breve descripción de las funcionalidades del sistema, por lo general escrito por el cliente.

Programación por pares.- XP propone el desarrollo en parejas con esto conlleva ventajas como la disminución de errores, costos y mejores diseños.

Refactorización.-Es reestructurar el código en forma constante con el objetivo de mejorar la legibilidad, simplicidad; es decir hacerlo más flexible para posteriores cambios.

En la planificación de cualquier proyecto hay tres conceptos que se manejan:

**Alcance**.- Se define las áreas necesarias para alcanzar las características que se desea obtener del producto.

**Tiempo**.- Previsión de la duración que llevará el proyecto.

Una iteración 10 días laborables (2 semanas).Desarrolladores 2.Horas diarias 5.

**Costo**.- Dinero y recursos que se destinaran al proyecto.

Costo de ejecución de las actividades=Cantidad de horas del programador\*costo de hora del programador (sueldo mensual/media de horas trabajadas por mes)

Para mantener los objetivos de calidad determinados, si se hace una modificación en una de las tres variables implica la modificación de alguna de las otras dos.

#### 4.3.2.2.3 Esfuerzo de desarrollo

XP propone mayor esfuerzo de trabajo es decir se aumentan las horas de trabajo a la semana.

Personas en el equipo: 3 PERSONAS, horas de esfuerzo de desarrollo: 3 horas por día, días de esfuerzo de desarrollo: 15 días, semanas de esfuerzo de desarrollo: 3 semanas.

	Nr o.	Descripción	Tiempo Estimado		
			Semanas	Días	Horas
<b>Iteración 1</b>	01	Registrar clientes	1,4	7	21
	02	Registrar asesores	1,4	7	21
	03	Registrar usuarios	1,4	7	21
	04	Registrar grupos	2	10	30
	05	Registrar cuenta individual	2	10	30
	06	Registrar cuenta grupal	2,4	12	36
	07	Registrar carnet	1,4	7	21
<b>Iteración 2</b>	08	Registrar solicitud	3,6	18	54
	09	Verificar datos del cliente	3,2	16	48
	10	Autorizar solicitud	4	20	60
	11	Generar ciclos	4,4	22	66
<b>Iteración 3</b>	12	Automatizar el flujo de ahorros	5	25	75
	13	Controlar deudas consolidadas	4,4	22	66
	14	Controlar movimientos de la cuenta interna	5	25	75
	15	Controlar movimientos de cuenta grupal	6,4	32	96
	16	Controlar utilidades	3	15	45
<b>Iteración 4</b>	17	Reporte por grupo	3	15	45
	18	Reporte por cliente	2,4	12	36
	19	Generar Balance	5	25	75
	20	Generar estados de resultados	5	25	75
			66,4	332	996

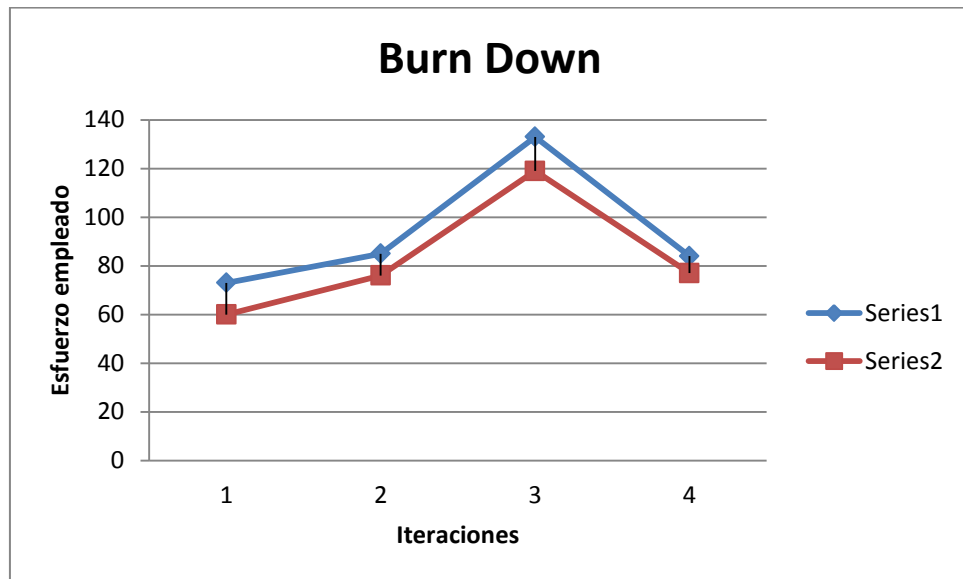
**Tabla N° 4.23 Tiempo Estimado del proyecto**  
Realizado por: El Investigador

	Nro	Tareas	Estado de Desarrollo	Esfuerzo real invertido Sist. ASES	Esfuerzo estimado XP sema	Días estimados
<b>Iteración n 1</b>	01	Registrar clientes	completo	2	1,4	7
	02	Registrar asesores	completo	2	1,4	7
	03	Registrar usuarios	completo	2	1,4	7
	04	Registrar grupos	completo	2,2	2	10
	05	Registrar cuenta individual	completo	2,2	2	10
	06	Registrar cuenta grupal	completo	2,8	2,4	12
	07	Registrar carnet	completo	1,4	1,4	7
<b>Iteración n 2</b>	08	Registrar solicitud	completo	4	3,6	18
	09	Verificar datos del cliente	completo	3,4	3,2	16
	10	Autorizar solicitud	completo	4,6	4	20
	11	Generar ciclos	completo	5	4,4	22
<b>Iteración n 3</b>	12	Automatizar el flujo de ahorros	completo	5,4	5	25
	13	Controlar deudas consolidadas	completo	5	4,4	22
	14	Controlar movimientos de la cuenta interna	completo	5,4	5	25
	15	Controlar movimientos de cuenta grupal	completo	7	6,4	32
	16	Controlar utilidades	completo	3,8	3	15
<b>Iteración n 4</b>	17	Reporte por grupo	completo	3,4	3	15
	18	Reporte por cliente	completo	2,8	2,4	12
	19	Generar Balance	completo	5,2	5	25
	20	Generar estados de resultados	completo	5,4	5	25
				<b>75,1</b>	<b>66,4</b>	<b>332</b>

**Tabla N° 4.24 Tiempo Real invertido**  
**Realizado por: El Investigador**



#### 4.3.2.2.4 Burn Down en XP



*Figura N<sup>o</sup> 4.7 Gráfico del Burn Down con XP  
Realizado por: El Investigador*

Se observa en el gráfico que aplicando XP en la primera iteración se puede terminar en el tiempo estimado, mientras que en la segunda iteración se observa un retraso pero podemos ver que esto es superado en la tercera iteración, y se puede terminar en el tiempo previsto, aunque XP no es riguroso en cuanto al tiempo de culminación pero se puede mejorar en cuanto a las prácticas se hacían antes en la Pastoral.

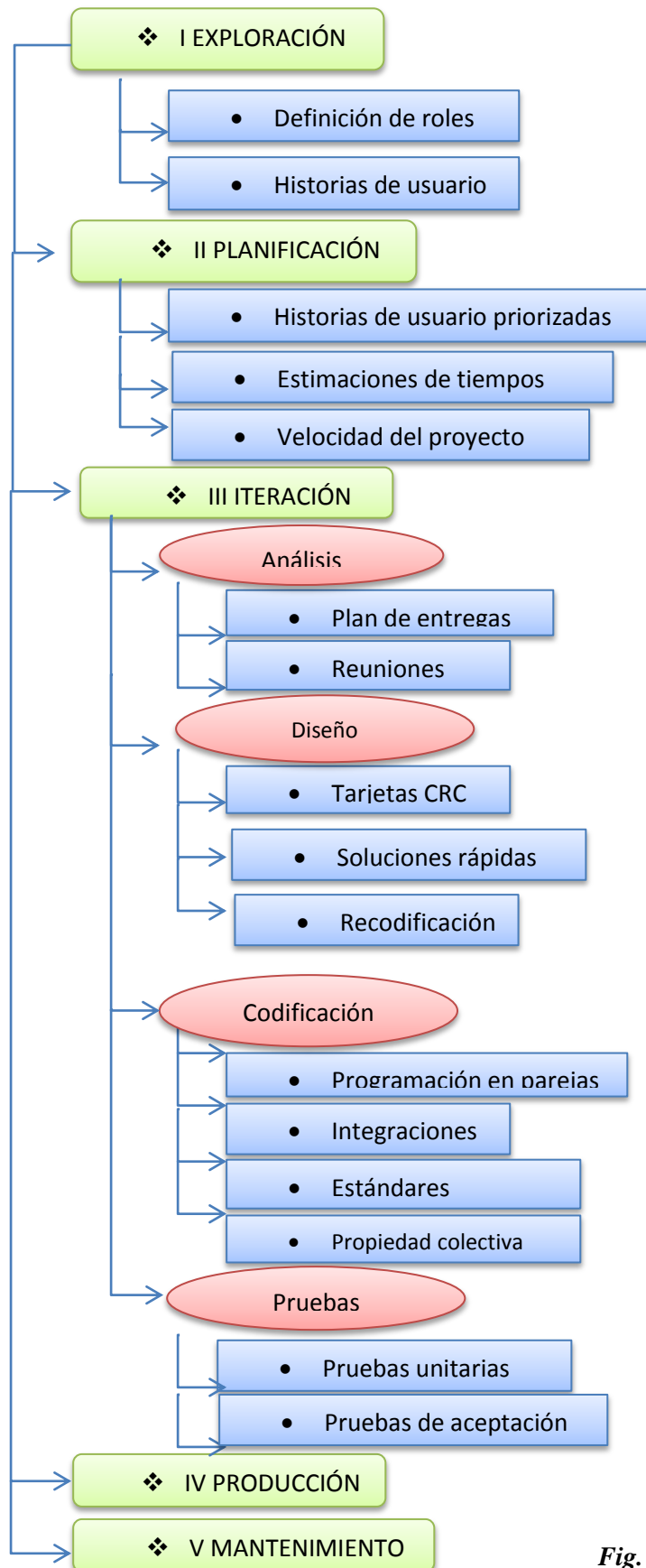
#### **Conclusión**

Se pudo mejorar con los valores y prácticas de XP en cuanto a la comunicación, la simplicidad del en el momento de codificar, la participación del cliente en todo el proyecto y el valor para resolver los inconvenientes en cuanto se presente.

XP se ha adaptado a la empresa en primer lugar porque los proyectos que se maneja en la misma son pequeños, el grupo de trabajo tiene pocos integrantes y es adecuado para la empresa.

Por otro lado la documentación para el desarrollo XP propone algo simple y sencillo; solo lo necesario para ser utilizado posteriormente.

#### 4.4 GUÍA DE LA METODOLOGÍA XP (EXTREME PROGRAMMING)



*Fig. Nª 4.8 Fases de XP  
Elaborado por: El Investigador*

#### 4.4.1 EXPLORACIÓN

Inicialmente se comenta sobre cada uno de los aspectos que XP propone para la etapa de planeación.

Para cada uno de los elementos se enuncia lo que la teoría sobre XP recomienda contrastándola con la experiencia real en la realización del proyecto.

El Jefe del proyecto debe convocar a una reunión inicial donde se debe contar con la participación de todos los integrantes y se definen en primer lugar los roles.

##### 1.-Definición de Roles

Roles	Persona	Área	Descripción
Cliente	Francisco Freire	Director de la Pastoral	Determina las historias de usuario las prioriza y él está en todo el proyecto.
Programador	David Ruiz	Programador	Es el responsable de construir el sistema y aportaciones técnicas.
Director	Marco Mayorga	Jefe del Dep. Sistemas	Es el líder del grupo y toma decisiones importantes.
Encargado del Seguimiento	Danilo Montalvo	Analista programador	Realiza el seguimiento de los procesos de cada iteración
Verificador	Víctor López	Comunicaciones	Ayuda al cliente con las pruebas que se ejecuten correctamente.

*Tabla N° 4.25 Identificación de roles  
Realizado por: El Investigador*

2.- El Director enuncia a cada uno de los integrantes lo que se va hacer en esta fase y con la participación principal del cliente se escuchan las ideas que para el proyecto que se pondrá en marcha.

3.- Se determinan las herramientas con las que se desarrolla el proyecto en este caso se ha estado utilizando las siguientes:

- Servidor de Base de datos. SQL 2008
- Herramienta de Desarrollo de Software: VISUAL ESTUDIO 2010

4.- Los desarrolladores construyen uno o varios prototipos del sistema.

5.- Con el director de proyecto se realiza una idea general que tendrá el proyecto para determinar el alcance y se procede a redactar historias de usuario.

### **2.- Historias De Usuario**

Las “Historias de usuarios” sustituyen a los documentos de especificación funcional, y a los “casos de uso”. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido.

Si alguna de ellas tiene riesgos que no establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba “spikes”, para reducir estos riesgos.

#### **Características**

- Las historias de usuario deben ser independientes unas de otras, de ser necesario se combinan con otras dependientes o buscar la manera para que las historias sean independientes.
- Valoradas por el dueño del producto es decir cada historia debe ser importante para el dueño del producto que para el equipo.
- Deben ser estimables porque un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomara completarla.

- Deben ser pequeñas no tan extensas porque son difíciles de estimar e imponen restricciones. Pero si son muy cortas se recomienda unir las en una sola.
- Las historias de usuario cubren requerimientos funcionales por lo que generalmente son verificadas en cada entrega.
- Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

<b>Historias de Usuario</b>	
<b>Numero: 1</b>	Nombre Historia de Usuario: REGISTRAR CLIENTES
<b>Modificación o extensión de Historia de Usuario (Nro. y Nombre) :</b>	
<b>Usuario: ADMINISTRADOR</b>	Iteración Asignada: PRIMERA
<b>Prioridad en Negocio: MEDIA</b> (Alta/ Media/ Baja)	Puntos Estimados : 7 días
<b>Riesgo en el Desarrollo:</b> MEDIO (Alto/ Medio/ Bajo)	Puntos Reales:
<b>Descripción :</b> Se realiza el registro de la información de los clientes (Nombre, Apellido, Estado civil, nacionalidad, dirección, teléfono)	
<b>Observaciones:</b> El cliente se registra al momento de aperturar una cuenta	

*Tabla N° 4.26 Ejemplo de historia de usuario  
Elaborado por: El Investigador*

#### 4.4.2 PLANIFICACIÓN

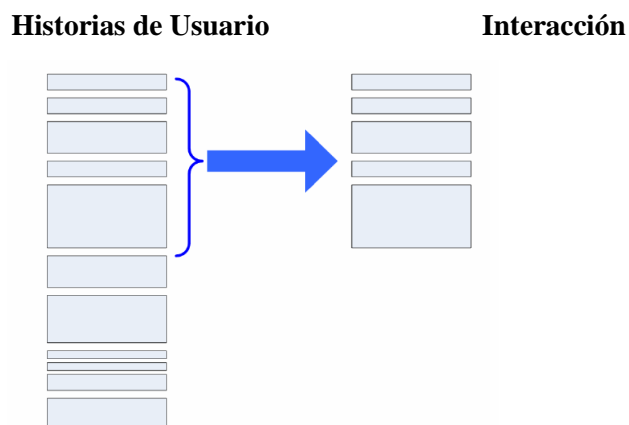
Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

Entre los elementos a discutir se encuentran las historias de usuario, el plan de entregas, el esfuerzo total y la velocidad del equipo.

Se recoge información para poder hacer una valoración económica del posterior desarrollo. Para ello se discutirá sobre el funcionamiento general de la empresa, los distintos departamentos que la componen, los tipos de empleados, etc.

1.- Se reúnen Todos los integrantes del proyecto y como anteriormente ya elaboraron las historias de usuario ahora se las va priorizar.

- El cliente establece la prioridad de cada historia de usuario de acuerdo con el valor que aporta para la empresa.
- El equipo puede realizar preguntas e incluso aportar sobre la prioridad de las historias de usuario.
- Se desglosa cada una de las historias de usuario en tareas funcionales.
- La iteración se formará de acuerdo a como el cliente agrupe las historias de usuario como se ve en la figura 4.9.



*Figura N<sup>o</sup> 4.8: Ejemplo de Asignación de Historias de usuario a una Iteración  
Elaborado por: El Investigador*

	<b>Nro.</b>	<b>Descripción</b>
<b>Iteración 1</b>	01	Registrar clientes
	02	Registrar asesores
	03	Registrar usuarios
	04	Registrar grupos
	05	Registrar cuenta individual
	06	Registrar cuenta grupal
	07	Registrar carnet
<b>Iteración 2</b>	08	Registrar solicitud
	09	Verificar datos del cliente
	10	Autorizar solicitud
	11	Generar ciclos
<b>Iteración 3</b>	12	Automatizar el flujo de ahorros
	13	Controlar deudas consolidadas
	14	Controlar movimientos de la cuenta interna
	15	Controlar movimientos de cuenta grupal
	16	Controlar utilidades
<b>Iteración 4</b>	17	Reporte por grupo
	18	Reporte por cliente
	19	Generar Balance
	20	Generar estados de resultados

*Tabla Nª 4.27: Ejemplo de Historias de usuario por Iteración  
Elaborado por: El Investigador*

2.- Se estima la duración de la iteración y la fecha de entrega de la funcionalidad a obtener.

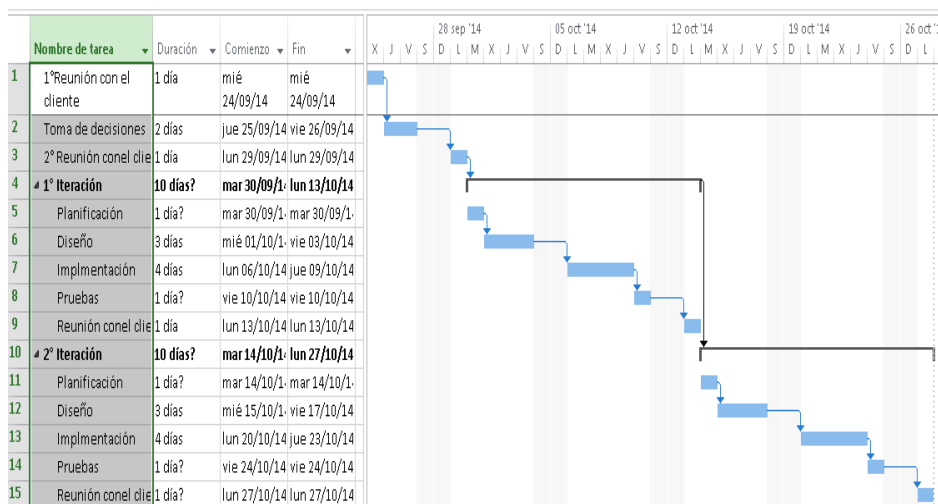
- Se puede hacer de 2 a 2 y 30 por cada semana de iteración.
- Debe quedar claro que las estimaciones realizadas en esta fase son primarias, y podrían variar cuando se analicen en más detalle en cada iteración.

	Nro.	Descripción	Tiempo Estimado		
			Semanas	Días	Horas
<b>Iteración 1</b>	01	Registrar clientes	1,4	7	21
	02	Registrar asesores	1,4	7	21
	03	Registrar usuarios	1,4	7	21
	04	Registrar grupos	2	10	30
	05	Registrar cuenta individual	2	10	30
	06	Registrar cuenta grupal	2,4	12	36
	07	Registrar carnet	1,4	7	21
<b>Iteración 2</b>	08	Registrar solicitud	3,6	18	54
	09	Verificar datos del cliente	3,2	16	48
	10	Autorizar solicitud	4	20	60
	11	Generar ciclos	4,4	22	66
<b>Iteración 3</b>	12	Automatizar el flujo de ahorros	5	25	75
	13	Controlar deudas consolidadas	4,4	22	66
	14	Controlar movimientos de la cuenta interna	5	25	75
	15	Controlar movimientos de cuenta grupal	6,4	32	96
	16	Controlar utilidades	3	15	45
<b>Iteración 4</b>	17	Reporte por grupo	3	15	45
	18	Reporte por cliente	2,4	12	36
	19	Generar Balance	5	25	75
	20	Generar estados de resultados	5	25	75
			66,4	332	996

**Tabla N° 4.28 Estimaciones de tiempo con XP**  
**Realizado por: El Investigador**



3.-Se realiza la planificación de acuerdo a las iteraciones que se han formado.



**Figura Nª 4.9 Ejemplo de diagrama de planificación**  
**Realizado por: El Investigador**

Después de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.

#### 4.-Velocidad del Proyecto

La velocidad del equipo es igual a las semanas estimadas por iteración basándose en el número de historias de usuario implementadas para cada iteración, de modo que la velocidad inicial del proyecto (primera iteración) constituirá la base de evaluación de las siguientes iteraciones y se pueda realizar un seguimiento adecuado de la estimación inicial de la velocidad de implementación del equipo.

	Iteración 1	Iteración 2	Iteración 3	Iteración 4
Horas	150	200	300	250
Semanas	10	14	18	16
Horas Semanales	25	23	21	30
Historias de usuario (velocidad del proyecto)	7	4	5	4

**Tabla Nª 4.29: Ejemplo de Velocidad estimada**  
**Elaborado por: investigador**

### 4.4.3 ITERACIONES

#### 4.4.3.1 Análisis

##### 4.4.3.1.1 Plan de entrega de iteración

Al comienzo de cada iteración, se realiza una reunión de planificación de la iteración en donde se centra específicamente asignar tareas y distribuir el trabajo a desarrollarse en la iteración.

- El tiempo máximo de reunión es de 2 horas por cada semana que dure la iteración.
- Cada historia de usuario se traduce en tareas específicas de programación.
- Asimismo, para cada historia de usuario se establecen las pruebas de aceptación.
- Pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

##### 4.4.3.1.2 Reuniones Diarias de Seguimiento

El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones.

- Todo los integrantes participarán dando opiniones si está bien o algo hay que cambiar.
- Para optimizar el tiempo del equipo, se sugiere realizar estas reuniones en círculo y de pie con un tiempo máximo de 15 minutos.
- Evitar discusiones largas.

#### 4.4.3.2 Diseño

El diseño se realiza en todo el tiempo de vida del proyecto, siendo constantemente revisado y muy probablemente modificando debido a los cambios presentados durante el desarrollo.

Entre los elementos más importantes que menciona XP referentes al diseño están las tarjetas CRC, las metáforas, la refactorización y soluciones pequeñas.

##### 4.4.3.2.1 Tarjetas CRC

Se crean las tarjetas CRC y el modelo Entidad Relación, del cual surgen varias versiones en la medida que se incorpore funcionalidades a la aplicación.

Diseño de las tarjetas CRC

Tarjeta CRC		
<b>Número:</b>	Escenario:	
<b>Nombre CRC:</b>		
<b>Responsabilidades:</b>	Colaboradores:	Métodos:
<b>Observaciones:</b>		

*Tabla N<sup>o</sup> 4.30: Modelo de Tarjeta CRC  
Elaborado por: El Investigador*

- **Clases** (Nombre de clase: define un vocabulario)
- **Responsabilidades** para cada clase: muestran los problemas que van a ser resueltos, mediante la utilización de una frase que inicia con un verbo activo (Una responsabilidad es algo que la clase sabe o hace, Indicadores de las acciones son los verbos).

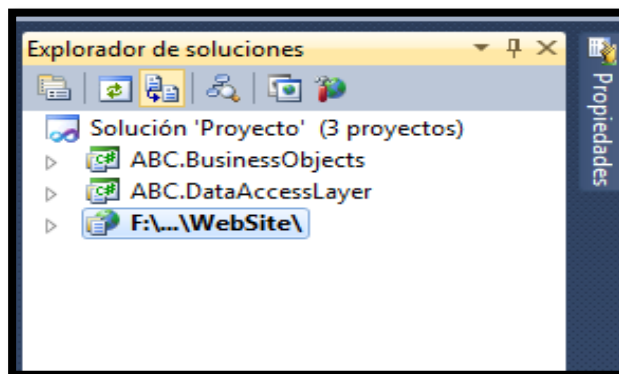
- **Colaboradores:** expresa dependencias entre objetos(Colaboradores son los que envían o reciben mensajes, se les pide información o realizar alguna acción)

Para la documentación de tarjetas CRC, a continuación se presenta la siguiente plantilla.

Tarjeta CRC		
<b>Número: 1</b>	Escenario: Registro de clientes	
<b>Nombre CRC:</b> clientes		
<b>Responsabilidades:</b>	<b>Colaboradores:</b>	<b>Métodos:</b>
.Mantener un registro de los clientes		ObtenerCliente
.Obtener cliente		GuardarCliente
.Guardar cliente		
<b>Observaciones:</b>		

*Tabla N° 4.31: Ejemplo de Tarjeta CRC  
Elaborado por: El Investigador*

La Arquitectura que se va a implementar para el desarrollo de la aplicación posee una distribución del sistema de tres capas, de la siguiente forma:



*Figura N° 4.10: Presentación de capas de desarrollo  
Elaborado por: El Investigador*

### **CAPA 1: Componente Arquitectónico de acceso de Datos:**

En este componente se gestionan todos los objetos de tipo de dato, manejados por medio de procedimientos almacenados que interactúan directamente con la base de datos.

### **CAPA 2: Componente Arquitectónico Lógica de Negocios:**

En este componente se gestiona todo lo referente a la lógica de negocio e interactúa directamente con la capa de Acceso de Datos.

Componente Arquitectónico Objetos de Negocio: Este componente maneja los atributos necesarios para que interactúe la capa de Acceso de Datos con la interfaz gráfica de usuario, permite encapsular cualquier tipo de dato u objeto manejado como una variable privada.

### **CAPA 3: Componente Arquitectónico Lógica de Negocios:**

Este componente basado en Web Forms contiene la implementación de la interfaz del sistema.

Diagrama de una interfaz de usuario



*Figura N<sup>o</sup> 4.11: Ejemplo de interfaz de usuario  
Elaborado por: Investigador*

### 4.4.3.2 Solución rápida “Spikes”

Cuando aparecen problemas técnicos, o cuando es difícil de estimar tiempo para implementar una historia de usuario, pueden utilizarse en pequeños programas de prueba llamados “spikes” para explorar diferentes soluciones.

- Si es necesario se utiliza “spike” únicamente para probar o evaluar una solución y luego se desecha.
- XP recomienda asignar estos programas o “spike” por parejas. El objetivo de lograr una comprensión rápida de cada uno de estos asuntos, asegurando el cumplimiento de los plazos del proyecto.

### 4.4.3.3 Recodificación ”Refactoring”

Consiste en escribir nuevamente una parte del código de un programa sin cambiar su funcionalidad, para hacerlo más efectivo, conciso y entendible.

Sin embargo la metodología XP sugiere recodificar cada vez que sea necesario,

- Los programadores pueden cambiar los tipos de datos no sólo en la base de datos sino también en los métodos donde existían estas variables, en un tiempo corto.
- Los resultados de esta práctica nos ayudan en las siguientes iteraciones, cuando sea necesario ampliar o cambiar la funcionalidad
- La recodificación se logra con éxito gracias a la estructura del programa que permite la adición del código faltante, y al plan de organización que se ideó en dicha modificación.

Un ejemplo que se puede aplicar “refactoring” es:

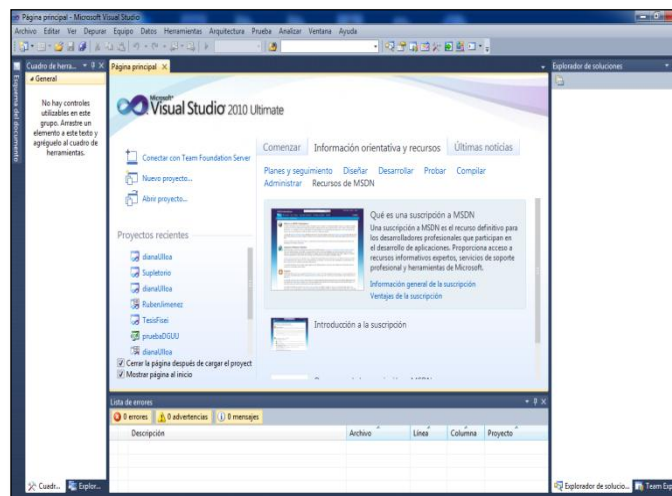
Cambiar el tipo de datos de las llaves primarias de algunas de las tablas, casi al finalizar el proyecto.

Dicha modificación afecta aproximadamente la mitad de las clases y métodos de la aplicación, entre las cuales se encontraban las relacionadas con otra tabla.

### 4.4.3.3 Codificación

En XP la codificación se inicia prácticamente desde el principio, favoreciendo el objetivo de estar haciendo entregas frecuentemente al cliente.

Algunos de los elementos más importantes en cuanto a la codificación son: la programación en pares, las integraciones, estándares, se debe trabajar en parejas y debe haber una propiedad colectiva del código.



*Figura N°4.12: Herramienta de programación*

*Elaborado por: Investigador*

#### 4.4.3.3.1 Programación en pares

XP propone que se desarrolle en pares programados, ambos trabajando juntos en un mismo ordenador.

- El código es permanente revisado por dos personas donde se puede identificar con mayor facilidad los errores.
- Se codifica de manera conjunta haciéndolo lo más simple posible.
- Si se presenta algún problema se resuelve de forma más rápida.
- El proyecto termina con más personas que conocen los detalles de cada parte del código.

- Se requiere que ambos programadores tengan concepciones similares para que sean productivos.

Es muy probable que dos programadores que no dominan la herramienta en la cual estén desarrollando sean mucho más productivos trabajando bajo un mismo computador que estando solos.

### **4.4.3.3.2 Integraciones permanentes**

- Se deben hacer integraciones cada pocas horas o en lo posible no tardar más de un día entre una y otra integración. Entre más se tarde en encontrar un problema, resultará más costoso resolverlo.
- Integrar frecuentemente evita problemas como el trabajar sobre una clase obsoleta. Generalmente se deben hacer una o dos integraciones diarias.
- Se debe garantizar en todo momento se está trabajando sobre la última versión del proyecto.

### **4.4.3.3.3 Uso de Estándares**

XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación.

- La estandarización del código debe ser asumida desde el mismo momento en que se inicie la codificación.
- Desde el inicio siguiendo una disciplina se debe tener un esquema de estándares acordados de forma tácita,

Se debe resaltar que el programar empleando estándares es una práctica que no solo se recomienda en XP, es una buena práctica que debe ser seguida en cualquier metodología de desarrollo lo que no implica algo muy novedoso en XP. Se trata de un medio con el cual se pretende facilitar la propiedad colectiva del código.



### 4.4.3.3 Propiedad Colectiva del Código

Estrategias como la rotación del personal, el empleo de estándares y la programación en parejas van destinadas a la consecución de la propiedad colectiva del código, de modo tal que solo se logrará este objetivo en la medida que las estrategias planteadas sean ejecutadas cuidadosamente.

- Se debe procurar rotar a los programadores no solo de compañero, también de partes del proyecto a desarrollar.
- Cualquier programador debería poder continuar la codificación que alguien más empezó sin muchas dificultades.

#### Cliente Presente

Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP.

- Los desarrolladores requieren especificaciones y detalles que solo el cliente puede proporcionar.
- No se requieren de largos documentos de especificaciones, sino solo detalles.
- El cliente ayuda a los desarrolladores a prevenir a tiempo situaciones no deseables y corregirlas a tiempo.

#### No trabajar horas Extras

- El dedicar horas extras a un proyecto retrasado, no lo va a poner al día, es preferible replantear los plazos a trabajar horas extras.
- La metodología XP indica debe llevarse un ritmo de 40 horas semanales, se puede variar entre 35, 40, 45 por semana. Lo importante es que se debe planificar el trabajo de manera que se lleve un ritmo constante y razonable sin sobrecarga de trabajo al equipo.

- En la medida posible se debe negociar el plan de entrega, realizando una nueva reunión de planificación con el cliente, los desarrolladores y los gerentes.

#### 4.4.3.4 PRUEBAS

##### 4.4.3.4.1 Pruebas unitarias

La metodología XP propone un modelo inverso, en el que, lo primero que se escribe son los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas.

#### Diseño de Prueba Unitaria

Prueba	1
<b>Descripción</b>	Registrar un cliente
<b>Objetivos</b>	Verificar que todos los datos del cliente se ingresen de manera correcta.
<b>Condiciones</b>	Se presiona el botón nuevo y activaran todos los campos para que ingresen todos los datos personales del cliente y al final se guarden.
<b>Resultado Esperado</b>	Los datos del cliente deben estar guardados sin ningún error en los campos.
<b>Resultado Obtenido</b>	La prueba se realizó satisfactoriamente ya que se pudo ingresar los datos sin ningún problema y el cliente quedo registrado.

*Tabla N<sup>o</sup> 4.32: Ejemplo de prueba unitaria  
Elaborado por: El investigador*

- Se debe escribir primero las pruebas, esto identificará los posibles casos especiales que deberá pasar el código.
- El cliente y los desarrolladores participan sobre las pruebas de aceptación.

#### 4.4.3.4.2 Pruebas de Aceptación

Se deben diseñar las pruebas de aceptación con base en las historias de usuario

Mediante la planificación de iteraciones y en base a las especificaciones de historias de usuario, se crea las pruebas de aceptación,

Para la documentación formal de la pruebas de aceptación, se procede a utilizar la siguiente plantilla.

<b>PRUEBAS DE ACEPTACIÓN</b>	
<b>Caso de Prueba: Registro Clientes</b>	
<b>Numero de Caso de Prueba:1</b>	Número Historia de Usuarios:1
<b>Nombre de Caso de prueba:</b>	
<b>Descripción:</b>	
<b>Condiciones de ejecución</b>	
<b>Entradas:</b>	
<b>Resultado Esperado:</b>	
<b>Evaluación:</b>	

*Tabla Nª 4.33: Plantilla de prueba de aceptación*

*Elaborado por: El Investigador*

**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.

<b>PRUEBAS DE ACEPTACIÓN</b>	
<b>Caso de Prueba:</b> Registro Clientes	
<b>Numero de Caso de Prueba:1</b>	Número Historia de Usuarios:1
<b>Nombre de Caso de prueba:</b> Registrar Cliente	
<b>Descripción:</b> Permitirá ingresar los datos de los clientes y almacenarlos en la base de datos.	
<b>Condiciones de ejecución:</b> Se llevara a cabo si el cliente desea aperturar una cuenta.	
<b>Entradas:</b> Los parámetros de entrada.  Nombres del cliente  Apellidos del cliente  Cedula  Teléfono  Dirección	
<b>Resultado Esperado:</b> Los datos se ingresaron con éxito.	
<b>Evaluación:</b> Se tiene que hacer una actualización cada vez que se ingresa un nuevo registro.	

*Tabla Nª 4.34: Prueba de aceptación*

*Elaborado por: El Investigador*

Las pruebas de aceptación también representa una salida del sistema que espera el cliente sea fundacional además de ayudar a realizar un seguimiento del código a emplear.

XP enfatiza en la realización de un sin número de pruebas a lo largo del proyecto, con el fin de asegurar en todo momento la realización de lo planteado en el diseño.

#### **4.4.4 PRODUCCIÓN**

El cliente en conjunto con los desarrolladores realizan pruebas adicionales y revisiones de rendimiento de cada entrega del sistema antes de poner en funcionamiento.

Al mismo tiempo se debe tomar decisiones sobre la inclusión de nuevas características a versión actual.

Las ideas que ha sido propuestas son documentadas para su posterior implementación esto se lo hace en un documento y se guarda en un cd en el departamento de sistemas.

#### **4.4.5 MANTENIMIENTO**

En esta fase se debe agregar una nueva funcionalidad así como mantener el sistema corriendo para poder comprobar si todo está bien.

Los desarrolladores y todo el equipo comienzan una nueva entrega con lo que se regresa a fase de planificación.

## CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

### 5.1 CONCLUSIONES

- ✓ El estudio de procesos que se realizó en la Pastoral demuestran que no son los adecuados para el desarrollo de software ya que se invierte mayor tiempo en su realización; por su extensa documentación lo cual representa una pérdida para la Pastoral.
- ✓ El realizar un estudio comparativo de las metodologías permitió encontrar las ventajas y desventajas que poseen las mismas e identificar con claridad la metodología adecuada para la Pastoral.
- ✓ Concluimos que la metodología XP en el departamento de sistemas en la Pastoral nos ayudará con la disminución de errores por la comunicación con el cliente en todo el desarrollo, además de su programación organizada, pruebas continuas, documentación simplificada y principalmente porque se adapta trabajando con equipos pequeños.
- ✓ La aplicación de la guía de la metodología XP mejorará la eficiencia y calidad del desarrollo de software en el departamento de sistemas de la Pastoral en base a las estimaciones realizadas, donde el tiempo empleado es menor.(Tabla 4.24)
- ✓ El estudio de la metodología XP permitió establecer un cambio representativo para la Pastoral ya que se puede obtener un producto garantizado y que cumpla con todos los requerimientos del cliente.

## 5.2 RECOMENDACIONES

- ✓ Aplicar las metodologías ágiles para estandarizar el desarrollo de software en el departamento de la Pastoral permitirá disminuir el tiempo utilizado por los programadores en la producción de software y así mismo reducir los procesos.
- ✓ Se recomienda trabajar en parejas, esto ayudará a la aplicación a estar sometida a constantes cambios de trabajo, mayor visibilidad y funcionabilidad para la empresa.
- ✓ Mantener la comunicación entre el cliente y desarrolladores durante el proceso de implementación de la aplicación, con la finalidad de lograr una retroalimentación concreta y frecuente que permita desplegar resultados funcionales que cumplan con las expectativas del usuario.
- ✓ Realizar las pruebas del sistema de forma continua y oportuna de manera que ayuden a reducir los posibles problemas que pueden presentarse a lo largo de una implementación de una iteración para lograr un producto funcional y de calidad.
- ✓ Se recomienda aplicar la guía de la metodología XP para el departamento de sistemas de la Pastoral con porque permitirá que los desarrolladores puedan trabajar de formas organizada, reduciendo esfuerzo de trabajo y obteniendo la satisfacción del cliente.

## BIBLIOGRAFÍA

[1]	PRESSMAN, Roger, Phd, “Ingeniería del Software” Un enfoque práctico, Séptima Edición, México Sep, 2005. [Accedido: Sep. 01, 2013].
[2]	CERVANTES, Humberto “Ingeniería de software”, <i>humbertocervantes.net</i> , Sep, 2008. [Online]. Disponible en: <a href="http://www.humbertocervantes.net/cursos/ingsoft/PresentacionCurso.pdf">http://www.humbertocervantes.net/cursos/ingsoft/PresentacionCurso.pdf</a> [Accedido: Sep. 10, 2013].
[3]	INTECO, Instituto Nacional de Tecnologías para la Comunicación “Ingeniería del Software. Metodologías y Ciclos de Vida”, <i>intenco.com</i> , Mar, 2009. [Online]. Disponible en: <a href="http://www.inteco.es/file/N85W1ZWfHifRgUc_oY8_Xg">http://www.inteco.es/file/N85W1ZWfHifRgUc_oY8_Xg</a> [Accedido: Sep. 12, 2013].
[4]	LETPEELIER, Patricio; PENADES, Carmen “Metodologías Agiles para el Desarrollo de Software”, <i>willydev.net</i> , Dic. 30, 2011. [Online]. Disponible en: <a href="http://www.willydev.net/descargas/masyxp.pdf">http://www.willydev.net/descargas/masyxp.pdf</a> [Accedido: Sep. 20, 2013].
[5]	COHM, Mike “Evolución de las Metodologías”, <i>usmp.edu.pe</i> , Sep 2012. [Online]. Disponible en: <a href="http://www.usmp.edu.pe/vision2012_lima/SEMINARIOS/seminarios/Evolucion_de_las%20metodologias.pdf">http://www.usmp.edu.pe/vision2012_lima/SEMINARIOS/seminarios/Evolucion_de_las%20metodologias.pdf</a> [Accedido: Sep. 23, 2013].
[6]	CALDERÓN, Amaro “Metodologías Agiles”, <i>sisman.utm.edu.ec</i> , Julio 2007. [Online]. Disponible en: <a href="http://www.sisman.utm.edu.ec/libros/.../METODOLOGIAS%20AGILES.pdf">www.sisman.utm.edu.ec/libros/.../METODOLOGIAS%20AGILES.pdf</a>



	[Accedido: Sep. 23, 2013].
[7]	GALLEGOS, Manuel “Metodología Scrum”, <i>openaccess.uoc.edu</i> Feb. 2012. [Online]. Disponible en: <a href="http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTF_C0612memoria.pdf">http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTF_C0612memoria.pdf</a> [Accedido: Sep. 24, 2013].
[8]	JOSKOWICZ, Jose “Reglas y Practicas de Extreme Programming”, <i>iie.fing.edu.uy</i> [Online]. Disponible en: <a href="http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf">http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf</a> [Accedido: Sep. 25, 2013].
[9]	SCOTT W. Ambler, “Modelo Agil”, <i>www.agilemodeling.com</i> , 2013. [Online]. Disponible en: <a href="http://www.agilemodeling.com/essays/fdd.htm">http://www.agilemodeling.com/essays/fdd.htm</a> [Accedido: Sep. 26, 2013].
[10]	CANAZA, Benedicto, “Metodologías Ágiles RUP”, <i>www.ingenieriadesoftware.mex.tl</i> , 2009. [Online]. Disponible en: <a href="http://ingenieriadesoftware.mex.tl/images/18149/P5_METODOS%20AGILES%20RUP_2_.pdf">http://ingenieriadesoftware.mex.tl/images/18149/P5_METODOS%20AGILES%20RUP_2_.pdf</a> [Accedido: Mar. 16, 2014].
[11]	ECHEVERY, Luis. DELGADO, Luz “Metodología Ágil XP al desarrollo de software”, Pereira 2007. [Tesis online]. T-054E18 Universidad Tecnológica de Pereira [Accedido: Abr. 10, 2014].
[12]	GARCÍA, Daniel “Adaptación de metodologías de Ingeniería de software Orientadas a objeto al mantenimiento evolutivo de aplicaciones”, 2008. [Tesis online]. Memoria-009 Universidad Politécnica de Cataluña [Accedido: Mayo. 23, 2014].

[13]	PULLAS, Tatiana “Desarrollo de un sistema para voto electrónico de voto aplicando metodología XP”, 2010. [Tesis online]. T-CM900 Escuela Politécnica Nacional [Accedido: Junio. 03, 2014].
[14]	TOAPANTA, Kleber “Metodología ágil SCRUM aplicado a un sistema de información con tecnología móvil”, 2012. [Tesis online]. T-ESPE-0344 Escuela Política del Ejército [Accedido: Junio. 07, 2014].
[15]	DÍAZ, De Santos “Dirección de Proyectos XP”, Segunda Edición 2013. [Libro online]. [Accedido: Junio. 08, 2014].
[16]	SOMMERVILLE, Ian “Ingeniería de Software”, Séptima Edición Madrid-España 2006. [Libro online]. [Accedido: Junio. 14, 2014].
[17]	PRIOLO, Sebastián “Métodos Ágiles”, Primera Edición 2009. [Libro online]. [Accedido: Junio. 10, 2014].
[18]	KNIBERG, Henrik “SCRUM y XP desde las Trincheras”, 2007. [Libro online]. Disponible en: <a href="http://infoq.com/minibooks/scrum-xp-from-the-trenches">http://infoq.com/minibooks/scrum-xp-from-the-trenches</a> [Accedido: Junio. 20, 2014].
[19]	ZAPATA, Javier “Explicación de la Programación Extrema”, Segunda Edición 2002. [Libro online]. [Accedido: Junio. 23, 2014].
[20]	CARVAJAL, Carlos, “Metodologías Ágiles”, 2008. [Tesis online]. Disponible en: <a href="http://upcommons.upc.edu/pfc/bitstream/2099.1/5608/1/50015.pdf">http://upcommons.upc.edu/pfc/bitstream/2099.1/5608/1/50015.pdf</a> [Accedido: Agosto. 16, 2014].

## GLOSARIO DE TÉRMINOS

**Metodología ágil.-** Método que permite incorporar cambios con rapidez en el desarrollo de software.

**Refactorización.-** Es el proceso que consiste en mejorar el código una vez escrito cambiando su estructura interna sin modificar su comportamiento externo.

**Product Backlog.-** Lista de objetivos o requisitos priorizada que representa la visión y expectativas del cliente respecto a los objetivos y entregas del producto o proyecto.

**Sprint.-** Es el período en el cual se lleva a cabo el trabajo.

**Sprint Backlog.-** Se describe el como el equipo va a implementar los requisitos durante el sprint.

**Historias de Usuario.-** Es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario

**Spike.-** Es una pequeña solución que se podía decir para realizar una función independiente del mecanismo existente.

**Iteración.-** Es un conjunto de periodos de tiempo dentro de un proyecto.

**Pruebas de aceptación.-** Tiene como propósito demostrar al cliente el cumplimiento de un requisito del software.

**Tarjetas CRC.-** Son una herramienta usado para el diseño de software el cual permite determinar las clases.

**Pruebas unitarias.-** Es una forma de comprobar el correcto funcionamiento de un módulo.

**Burn Down.-** Es un diagrama con una representación gráfica del trabajo por hacer en un proyecto.

**Manifiesto Ágil.-** Es un documento que está compuesto por 12 principios asociados a cuatro valores para el desarrollo de software.

## **ANEXOS**

### **ANEXO A**

#### **UNIVERSIDAD TÉCNICA DE AMBATO**

##### **FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL (FISEI)**

###### **Encuesta dirigida para los desarrolladores del departamento de sistemas de la Pastoral**

**OBJETIVO:** Recolectar información sobre la condición actual en que se está desarrollando software en la pastoral

#### **INSTRUCTIVO:**

- Procure ser lo más objetivo posible
- Marque con una X en el paréntesis de la alternativa que usted eligió.

**1. ¿La pastoral cuenta con una metodología de desarrollo de software adecuado para la empresa?**

**1.1 Si ( )**

**1.2 No ( )**

**1.3 No sé ( )**

**2. ¿Los procedimientos y métodos que utiliza la pastoral son los adecuados para el desarrollo d software?**

**2.1 Si ( )**

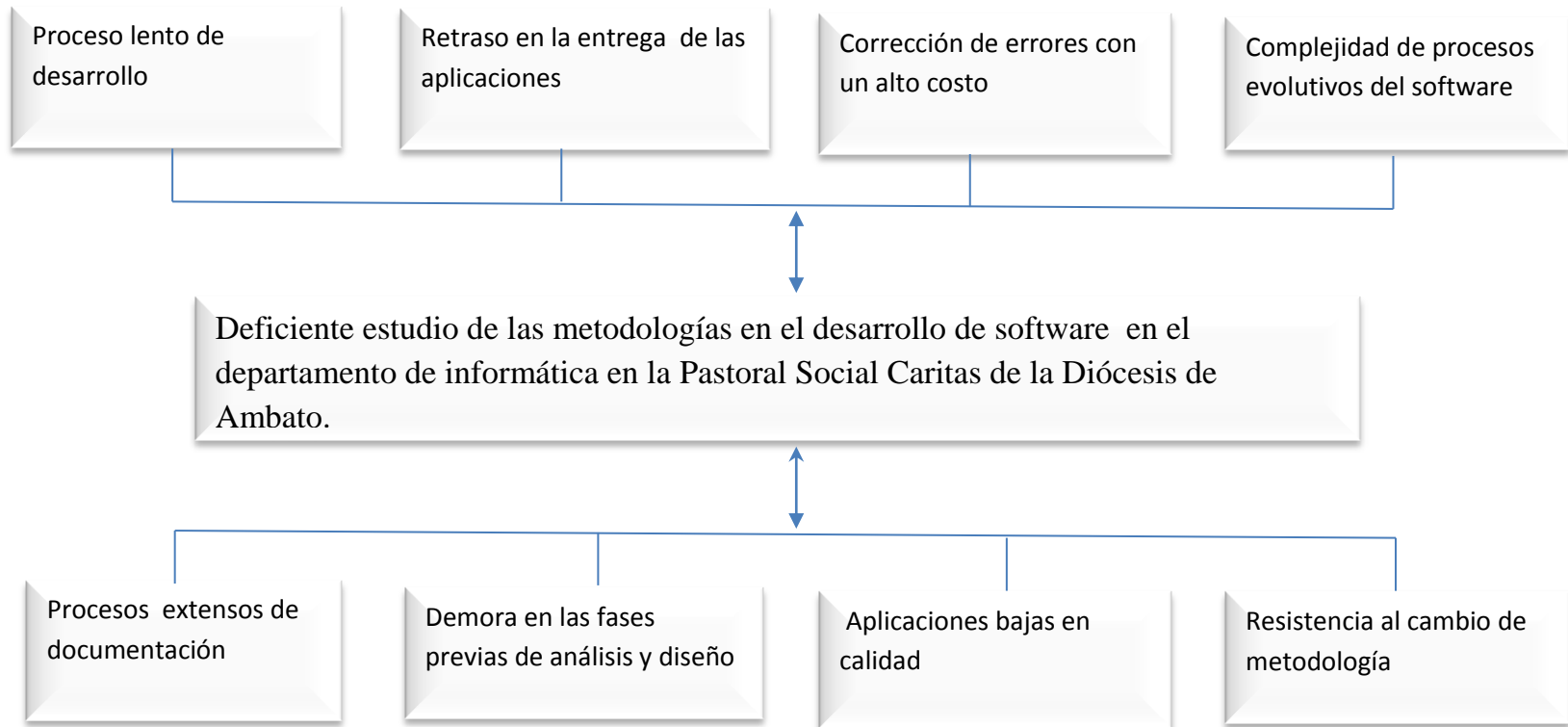
**2.2 No ( )**

**2.3 No sé ( )**

**3. ¿La calidad de desarrollo de software en la empresa es la esperada?**



ANEXO B: ÁRBOL DEL PROBLEMA



ANEXO C



MICROFINANZAS SOLIDARIAS

SOCIAS POR GRUPO

GRUPO:	25	EL PARAISO DE TISALEO
ASESOR:	Arevalo Villacis Sandra Elizabeth	

No	CEDULA	CODIGO	CLIENTE
1	1801370717	1199	BONILLA BORJA ELVIA LUZMILA
2	1800420331	1200	CAPUZ MANOTOA MARIA ROSARIO
3	1800429225	1203	GUEVARA BONILLA BERTHA MARIA
4	1802234755	1204	NU-EZ BAYAS MARGOTH DEL ROCIO
5	1801348010	1205	MAZABANDA ILVAY ANA MERCEDES
6	1800470526	1207	CAPUZ MANOTOA MARIA DOLORES
7	1802220887	1209	CAPUZ MANOTOA MARIA SOLEDAD
8	1802460947	1210	BONILLA GUEVARA BLANCA CECILIA
9	1803607157	1213	YUGCHA BONILLA FATIMA MARIBEL
10	1802700201	1218	YUGCHA BONILLA MERY CECILIA
11	1801676626	6399	FUENTES CHICAIZA MARIA GLORIA
12	1704811270	8013	ZAMORA FRIAS LIDIA AMERICA
13	0902850528	9615	BAYAS RODRIGUEZ ROSA EMPERATRIZ
14	0400976221	9618	QUITIAQUEZ QUIGUNTAR MIRIAM DEL CARMEN
15	1801801547	10308	QUISPILEMA BALLADARES EMMA BEATRIZ
16	1804163515	12067	LOPEZ ANCHALUIZA EVELYN GENOVEVA
17	1804037651	13958	BONILLA BORJA NELLY CUMANDA
18	1802386035	14829	ANALUISA VAYAS EUGENIA DEL ROCIO
19	1802222438	14830	ANCHALUIZA VILLEGAS NORMA PIEDAD
20	0913156931	14832	PORTERO YUGCHA LAURA BEATRIZ
21	1800187187	16499	MOPOSITA FREIRE LAURA ELENA
22	1804058293	16500	CAIZA MORALES SANDRA CAROLINA
23	1803302353	17448	GUAPISACA CHICAIZA MARIA ISABEL
24	1803553773	17449	GUAPISACA CHICAIZA ANA PATRICIA
25	1804138814	18646	TISALEMA ANALUISA LOURDES DEL ROCIO
26	1804661179	18686	CAIZA MAZABANDA MARTHA CECILIA
27	1805139381	18709	BONILLA BORJA LUZ MARIA
28	1803440104	18710	PULLUGANDO MANOBANDA GLADYS MAGDALENA
29	1804613907	19658	CHIPANTIZA MASABANDA MAYRA ALEJANDRA
30	1802899797	19659	CAIZA MORALES MONICA NOEMI

ANEXO D



MICROFINANZAS SOLIDARIAS  
SOLICITUD DE CREDITO PRELIMINAR

GRUPO	35	JESUS DEL GRAN PODER QUINCHICOTO
ASESOR	Yanchaluiza Pimboza Luis Raul	
CICLO	21	

No	Nombre	ANOR	ACRC	VIV	ACI	AHO	NNOR	NCRC	NCI	RET
1	CARRERA ORTIZ MERCY NEREIDA	0.00	0.00	0.00	0.00	643.49				
2	FREIRE ZURITA OLGA CARMELA	500.00	100.00	0.00	500.00	172.34				
3	LOPEZ RAMIREZ LILIA DEL CARMEN	0.00	0.00	0.00	0.00	500.21				
4	ORTIZ ALARCON WILMA MAGDALENA	1500.00	500.00	0.00	1000.00	662.31				
5	ORTIZ ROSERO MARIA DELFINA	1000.00	0.00	0.00	245.44	44.59				
6	RAMOS ORTIZ EMERITA GUADALUPE	1400.00	1000.00	0.00	600.00	204.90				
7	RAMOS VILLEGAS CECILIA MAGDALENA	1200.00	0.00	0.00	300.00	51.86				
8	SANCHEZ SANCHEZ EULOGIA MACRINA	500.00	0.00	0.00	0.00	229.35				
9	VILLACRES SANCHEZ ZOILA MARGARITA	0.00	0.00	0.00	0.00	397.27				
10	VILLACRES SANCHEZ PRISCILA PIEDAD	0.00	0.00	0.00	0.00	0.00				
11	VILLACRES ORTIZ BLADIS LUZMILA	0.00	0.00	0.00	0.00	24.07				
12	VILLACRES SANCHEZ JEANNETTE DEL SOCORRO	500.00	100.00	0.00	700.00	83.62				
13	VILLEGAS VILLACRES FATIMA TRINIDAD	0.00	0.00	0.00	0.00	14.15				
14	ORTIZ FREIRE MARIA ELEVACION	100.00	0.00	0.00	200.00	463.15				
15	TOAPANTA MEJIA MARIA CARMEN	500.00	0.00	0.00	0.00	474.93				
16	ORTIZ FREIRE MARIA FABIOLA	500.00	0.00	0.00	200.00	85.43				
17	RAMOS VILLEGAS EULOGIA DELFILIA	0.00	0.00	0.00	0.00	146.80				
18	RAMOS LLERENA REBECA ELIZABETH	0.00	0.00	0.00	0.00	1.49				
19	RAMOS ORTIZ MERCEDES IRENE	1900.00	1000.00	0.00	1100.00	29.78				
20	ORTIZ IBARRA JANETH MARGOTH	0.00	0.00	0.00	0.00	170.94				
21	GAVILANES FREIRE MARICELA LOURDES	0.00	0.00	0.00	0.00	221.22				
22	VILLACRES SANCHEZ NARSISA DEL ROCIO	500.00	0.00	0.00	100.00	238.45				
23	VILLEGAS TISALEMA VIOLETA MATILDE	0.00	0.00	0.00	0.00	254.99				
24	ESPINOZA MORENO LIDIA ENRIQUETA	0.00	0.00	0.00	0.00	77.35				
25	ORTIZ CASTRO MARIA CARMELINA	500.00	0.00	0.00	0.00	14.10				
26	VILLACRES ORTIZ RUTH MARIBEL	0.00	0.00	0.00	0.00	10.46				



ANEXO E



MICROFINANZAS SOLIDARIAS

MOVIMIENTO CUENTA GRUPAL

GRUPO: 25 EL PARAISO DE TISALEO

ASESOR: Arevalo Villacis Sandra Elizabeth

CICLO: 21 No CUENTA 45

CAPITALIZACION 434.76 EMERGENCIA 0.00 FONDO S. 287.50  
 AHORROS 767.39 BANCOS 90.78

No	FECHA	CAUSAL	VALOR	SALDO
1	6/28/2011	RET/PRES	1,120.00	134.05
2	7/27/2011	N/C/IntBan	0.22	134.27
3	7/27/2011	N/D/GasBan	1.75	132.52
4	7/27/2011	N/D/GasBan	1.00	131.52
5	7/27/2011	DEP/PRES	1,080.00	1,211.52
6	7/26/2011	N/C/P/EX	29.67	1,211.52
7	7/27/2011	N/D/P/CO	1,022.28	189.24
8	8/30/2011	DEP/PRES	805.00	994.24
9	8/23/2011	N/C/P/EX	28.81	994.24
10	8/30/2011	N/D/P/CO	805.00	189.24
11	9/6/2011	N/D/GasBan	1.75	187.49
12	9/6/2011	N/D/GasBan	1.75	185.74
13	9/23/2011	DEP/PRES	920.00	1,105.74
14	9/20/2011	N/C/P/EX	82.58	1,105.74
15	9/23/2011	N/D/P/CO	920.00	185.74
16	10/20/2011	DEP/PRES	420.00	605.74
17	10/21/2011	N/D/P/CO	420.00	185.74
18	11/8/2011	N/D/GasBan	1.75	183.99
19	11/8/2011	N/D/GasBan	1.75	182.24
20	11/16/2011	DEP/PRES	675.00	857.24
21	11/16/2011	N/D/P/CO	733.95	123.29
22	11/30/2011	DEP/PRES	100.00	223.29
23	11/30/2011	N/D/P/CO	100.00	123.29
24	12/6/2011	N/D/GasBan	1.75	121.54
25	12/14/2011	DEP/PRES	1,103.00	1,224.54
26	12/15/2011	DEP	30.00	1,254.54
27	12/15/2011	N/D/P/CO	1,103.30	151.24
28	12/15/2011	DEP	60.00	211.24
29	12/15/2011	N/D/Pago	238.30	211.24

**ANEXO F**



**PASTORAL SOCIAL CÁRITAS AMBATO**  
**DISPONIBLE INDIVIDUAL POR GRUPO**

<b>GRUPO:</b>	<b>25</b>	<b>EL PARAISO DE TISALEO</b>
<b>ASESOR:</b>	<b>Arevalo Villacis Sandra Elizabeth</b>	

No	COD	No CTA	NOMBRES	DISP.	BASE	TOTAL
1	1199	1095	BONILLA BORJA ELVIA LUZMILA	13.06	0.00	13.06
2	1200	1096	CAPUZ MANOTOA MARIA ROSARIO	12.89	0.00	12.89
3	1203	1097	GUEVARA BONILLA BERTHA MARIA	41.19	0.00	41.19
4	1204	1098	NU-EZ BAYAS MARGOTH DEL ROCIO	29.20	0.00	29.20
5	1205	1099	MAZABANDA ILVAY ANA MERCEDES	4.20	0.00	4.20
6	1207	1100	CAPUZ MANOTOA MARIA DOLORES	6.09	0.00	6.09
7	1209	7728	CAPUZ MANOTOA MARIA SOLEDAD	33.03	0.00	33.03
8	1210	1101	BONILLA GUEVARA BLANCA CECILIA	23.63	0.00	23.63
9	1213	1102	YUGCHA BONILLA FATIMA MARIBEL	300.93	0.00	300.93
10	1218	1103	YUGCHA BONILLA MERY CECILIA	0.06	0.00	0.06
11	6399	1104	FUENTES CHICAIZA MARIA GLORIA	39.52	0.00	39.52
12	8013	1105	ZAMORA FRIAS LIDIA AMERICA	0.00	0.00	0.00
13	9615	1106	BAYAS RODRIGUEZ ROSA EMPERATRIZ	2.98	0.00	2.98
14	9618	1107	QUITIAQUEZ QUIGUANTAR MIRIAM DEL CARMEN	5.49	0.00	5.49
15	10308	1113	QUISPILEMA BALLADARES EMMA BEATRIZ	0.00	0.00	0.00
16	12067	1108	LOPEZ ANCHALUIZA EVELYN GENOVEVA	0.00	0.00	0.00
17	13958	1114	BONILLA BORJA NELLY CUMANDA	0.94	0.00	0.94
18	14829	1112	ANALUISA VAYAS EUGENIA DEL ROCIO	16.40	0.00	16.40
19	14830	1116	ANCHALUIZA VILLEGAS NORMA PIEDAD	3.78	0.00	3.78
20	14832	1109	PORTERO YUGCHA LAURA BEATRIZ	28.60	0.00	28.60
21	16499	1110	MOPOSITA FREIRE LAURA ELENA	0.00	0.00	0.00
22	16500	1111	CAIZA MORALES SANDRA CAROLINA	31.41	0.00	31.41
23	17448	1115	GUAPISACA CHICAIZA MARIA ISABEL	4.12	0.00	4.12
24	17449	1117	GUAPISACA CHICAIZA ANA PATRICIA	20.50	0.00	20.50
25	18646	7730	TISALEMA ANALUISA LOURDES DEL ROCIO	8.12	0.00	8.12
26	18686	7729	CAIZA MAZABANDA MARTHA CECILIA	11.15	0.00	11.15
27	18709	7889	BONILLA BORJA LUZ MARIA	15.00	0.00	15.00
28	18710	7888	PULLUGANDO MANOBANDA GLADYS MAGDALENA	50.10	0.00	50.10
29	19658	9496	CHIPANTIZA MASABANDA MAYRA ALEJANDRA	40.00	0.00	40.00
30	19659	9497	CAIZA MORALES MONICA NOEMI	25.00	0.00	25.00
31	19888	9771	TOA MANOTOA TANNIA JANETH	0.00	0.00	0.00
<b>TOTAL GRUPAL:</b>				<b>767.39</b>	<b>0.00</b>	<b>767.39</b>

## ANEXO G

### Recopilación de información del sistema ASES

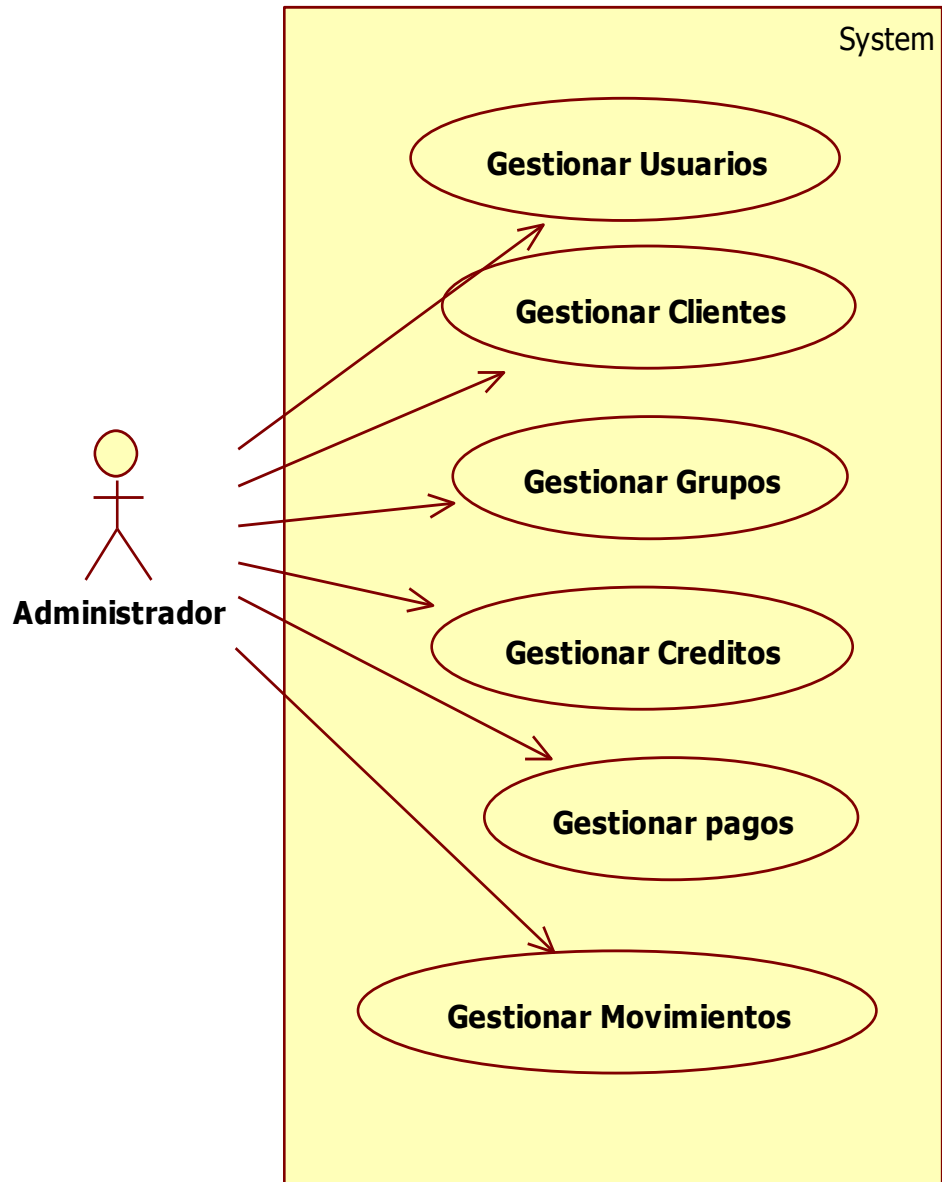
#### Definición de usuarios

A continuación se denominan los usuarios del sistema

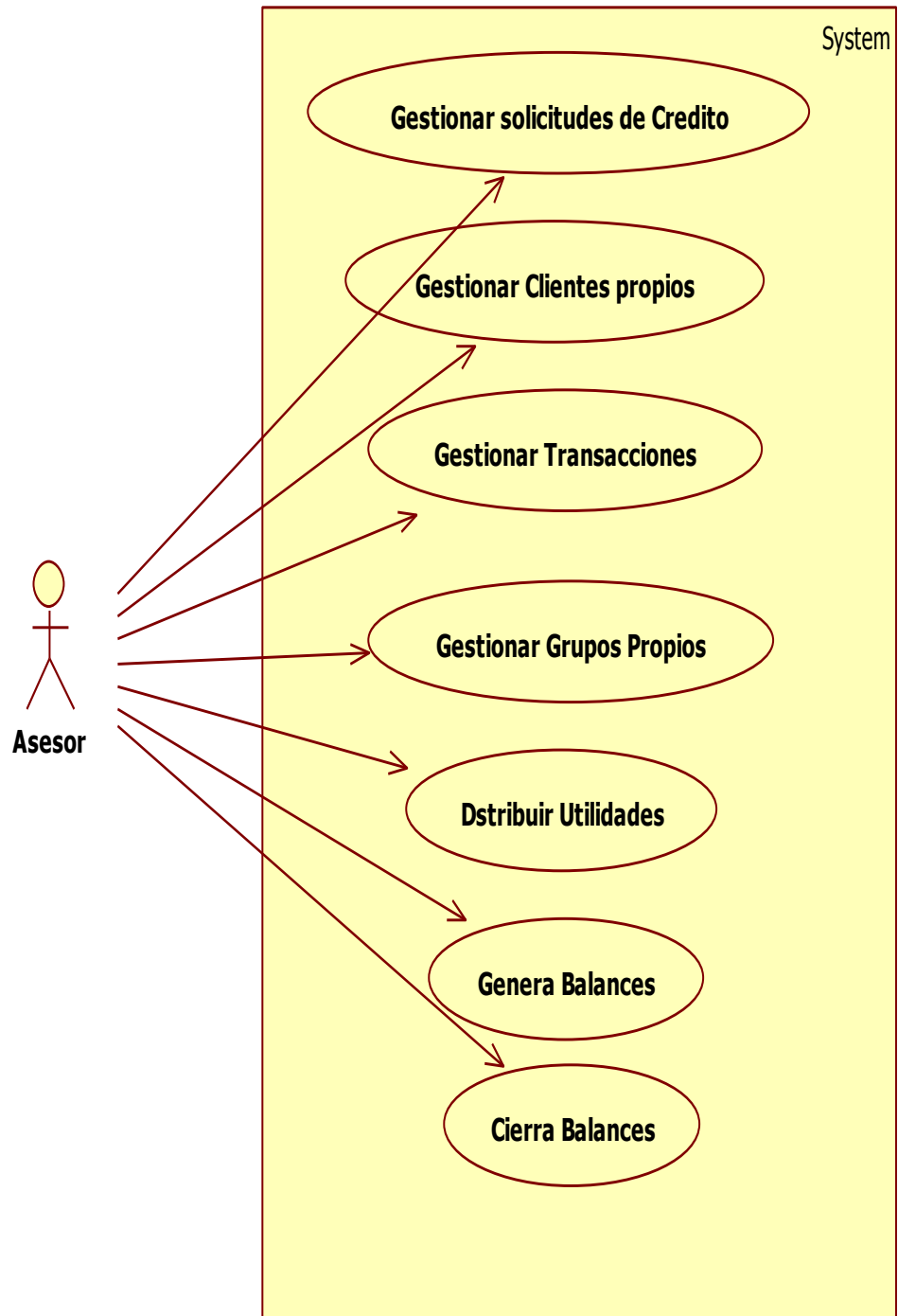
Usuario	Descripción
Administrador	Usuario con privilegios de administrador en el sistema.  Este tipo de usuario, puede gestionar todos los recursos utilizados por el sistema.
Supervisor	Solo puede mirar movimientos de cualquier persona y de cualquier grupo pero no pueden realizar transacciones, pueden mirar todos los reportes, y autorizan excepciones de movimientos.
Asesor	El ingresa las transacciones de los integrantes de los grupos que administra, cierra balances, ciclos, distribuye utilidades, ingresa socios, cambia de grupos, genera créditos, ingresa pagos, etc todo lo referente a el movimiento interno del grupo.

<p>Coordinador</p>	<p>Solo puede mirar movimientos de cualquier persona y de cualquier grupo pero no pueden realizar transacciones, pueden mirar todos los reportes, y autorizan excepciones de movimientos y puede mirar los reportes de las deudas consolidadas.</p>
<p>Operativos</p>	<p>Al inicio eran personas encargadas de realizar los débitos de las cuentas de las socias, luego pasamos a que lo hagan directo los asesores.</p> <p>Usuario no importante.</p>
<p>Operador Medico</p>	<p>Existe un módulo especial donde se lleva el control de un seguro de salud de las socias.</p>
<p>Salud</p>	<p>Solo tienen acceso a unos reportes especiales de socias para capacitaciones de salud.</p>

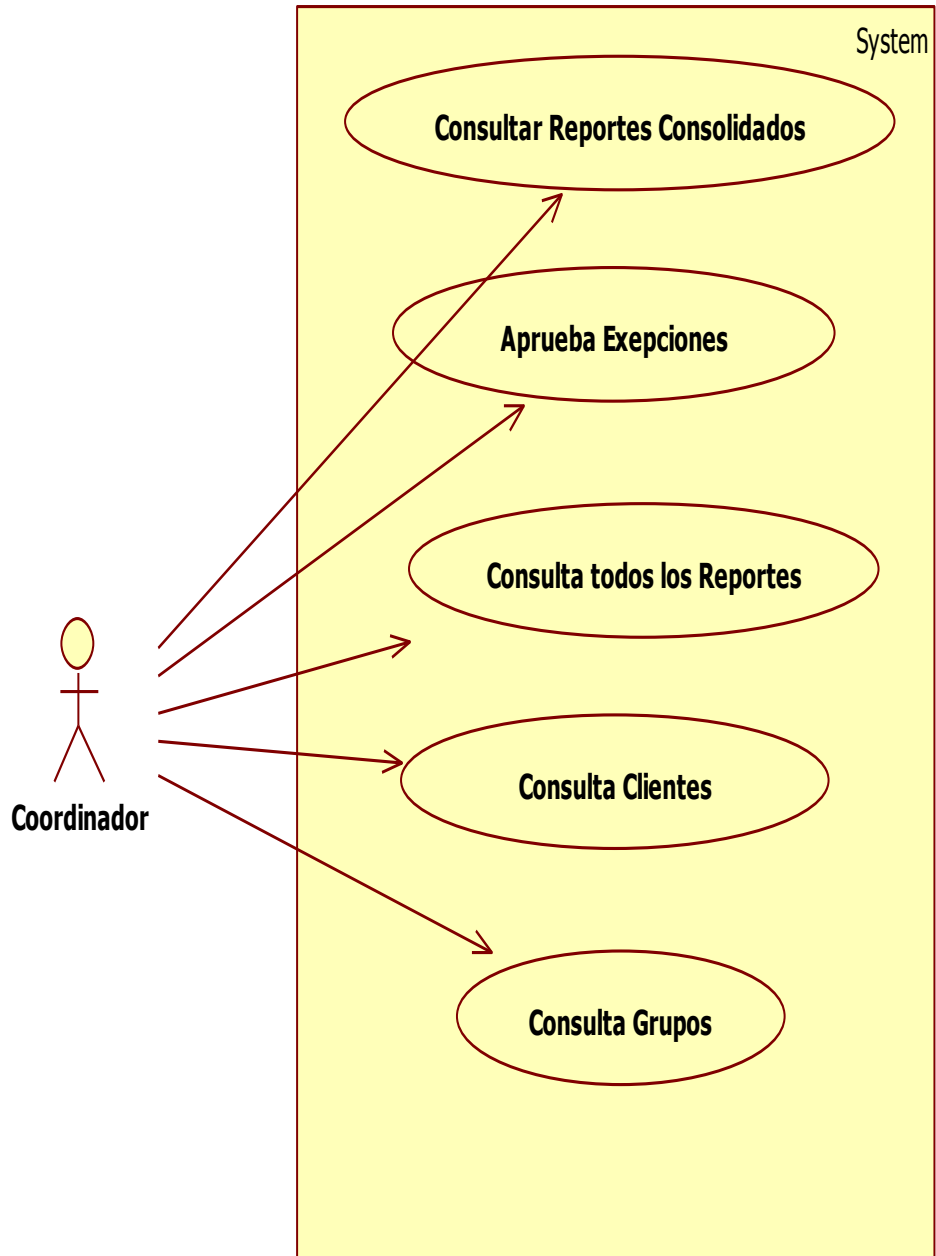
## Diagrama de caso de uso



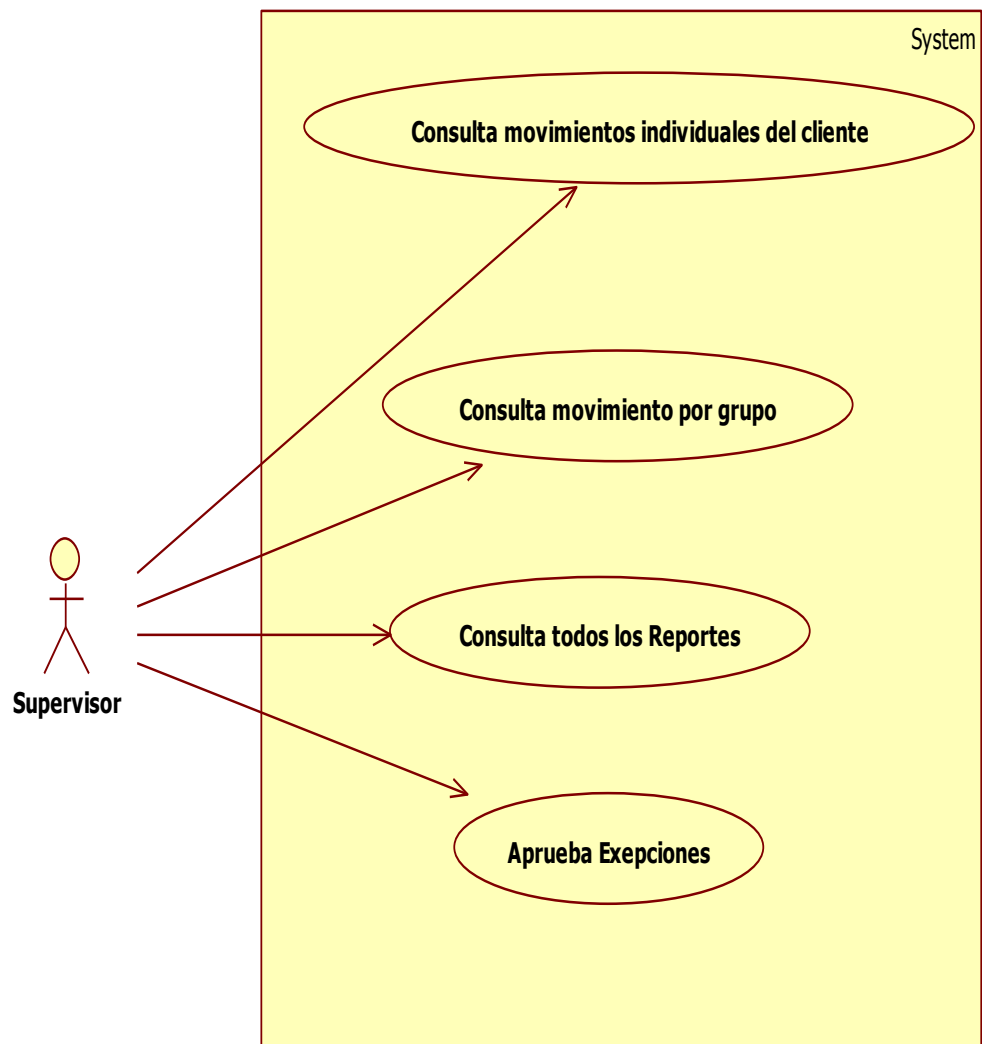
## Diagrama del rol de Asesor



## Diagrama del Rol Coordinador

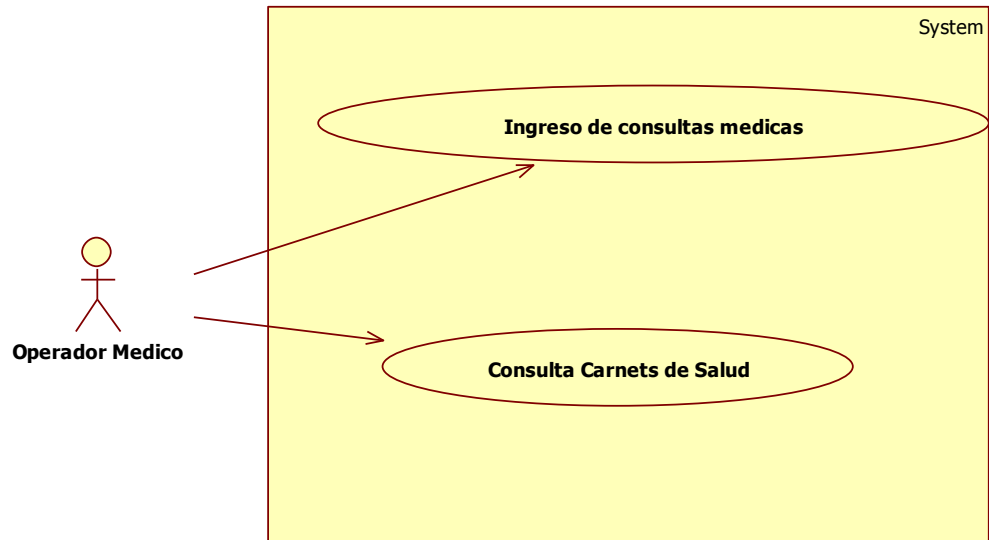


## Diagrama del Rol Supervisor

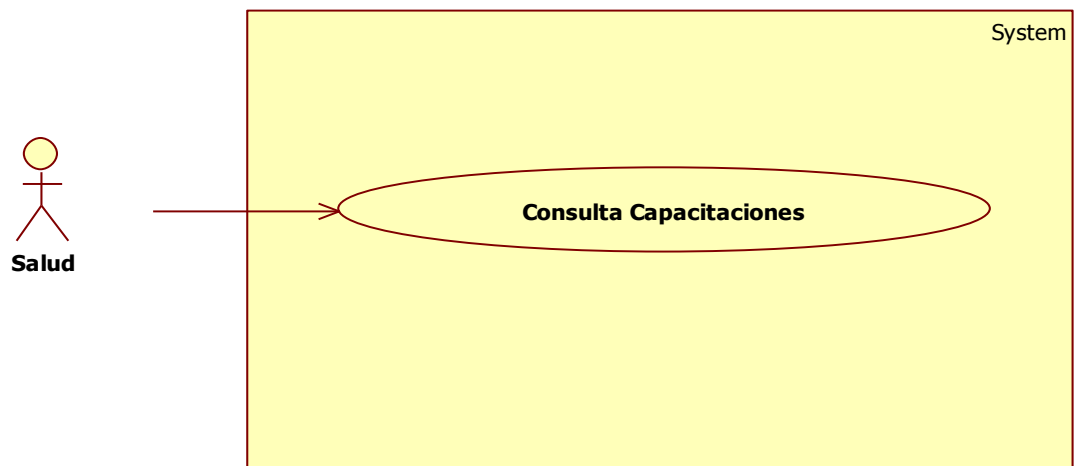




## Diagrama del Rol Operador Médico



## Diagrama del Rol Salud



	Administrador	
Descripción	Proporcionar información para dar mantenimiento a los usuarios del sistema y a los clientes	
Precondición	Registrarse en el sistema.	
Poscondición	Salir	
Flujo Normal		
	1	El usuario utiliza el formulario para mantenimiento de usuarios
	2	El sistema solicita los datos del nuevo usuario.
	3	El sistema almacena los datos proporcionados.
Flujos alternos		
	*	<b>Usuario ya registrado</b>
Excepciones		Si el usuario ya existe en el sistema y es necesario modificar la información actual, el sistema proporciona la funcionalidad para realizarlo.
	1	Si se intenta registrar un usuario con login el sistema debe advertir la situación y evitará que se registre el usuario.
Notas	1	Se requiere los siguientes datos del usuario:
		1.1 Login (*): identificador único de cada usuario con el cual se autenticara en el sistema. 1.2 Clave de acceso (*): código para acceso al sistema. 1.3 Nombres(*): 1.4 Apellidos(*): 1.5 Código(*): 1.6 Identificación 1.6.1 Tipo de identificación 1.7 Dirección de domicilio 1.8 Teléfono y celular (*): 1.9 Dirección de correo electrónico

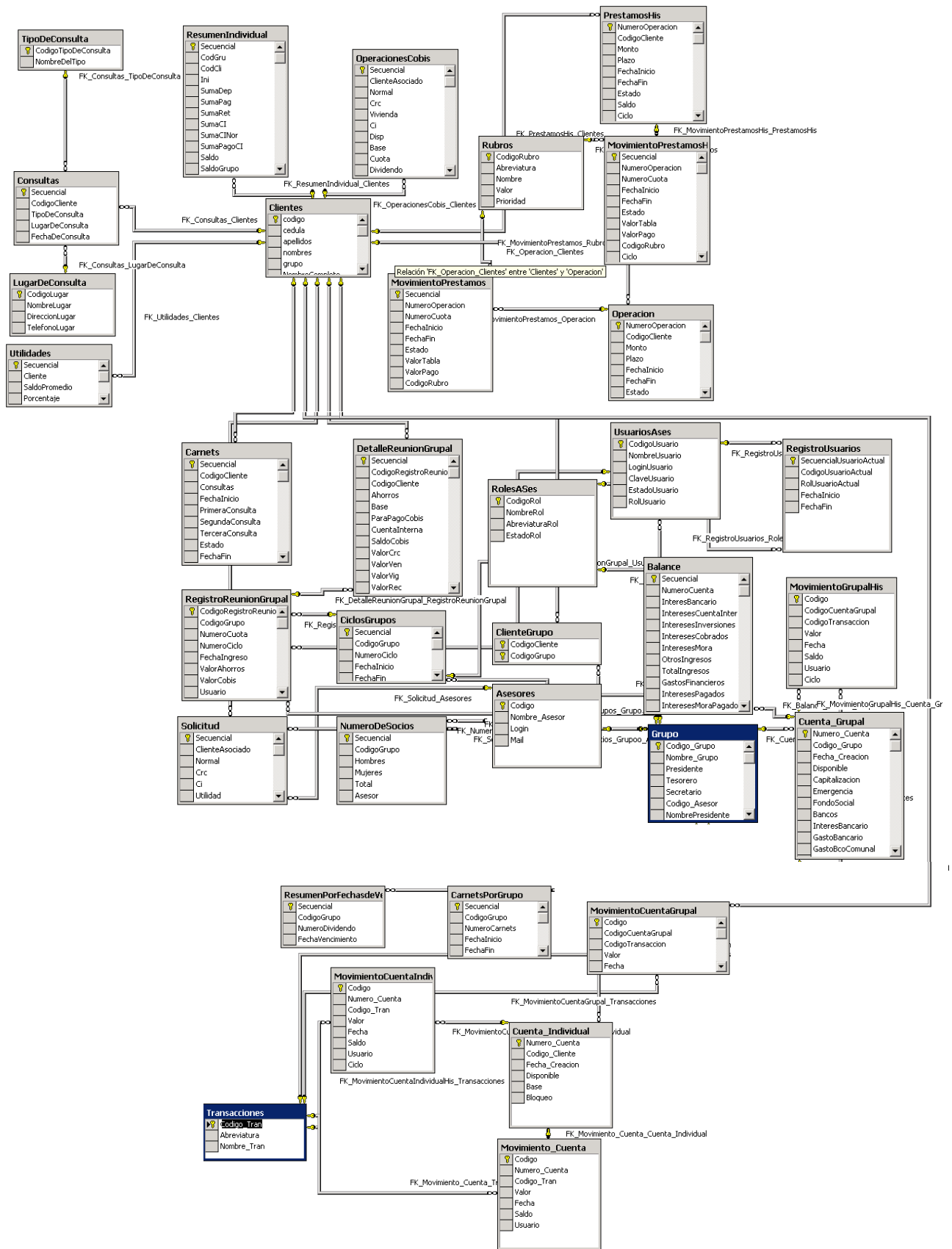
	Asesor	
Descripción	Proporcionar información para crear, modificar a los clientes	
Precondición	Registrarse en el sistema.	
Postcondición	Salir	
Flujo Normal		
	1	El usuario utiliza el formulario para mantenimiento de usuarios
	2	El sistema solicita los datos del nuevo cliente.
	3	El sistema almacena los datos proporcionados.
Flujos alternos		
	*	<b>Cliente ya registrado</b>
		Si el cliente ya existe en el sistema y es necesario modificar la información actual, el sistema proporciona la funcionalidad para realizarlo.
Excepciones		
	1	Si se intenta registrar a un cliente que ya exista el sistema debe advertir la situación y evitara que se registre al cliente.
Notas	1	Se requiere los siguientes datos del cliente:
		1.10 Código (*): identificador único de cada usuario con el cual se autenticara en el sistema. 1.11 Cedula (*): código para acceso al sistema. 1.12 Nombres(*): 1.13 Apellidos(*): 1.14 Número de cuenta(*): 1.15 Identificación 1.6.1 Tipo de identificación 1.16 Dirección de domicilio 1.17 Teléfono y celular (*): 1.18 Dirección de correo electrónico

	Supervisor	
Descripción	Proporciona información para consultas sobre los clientes y los grupos y generar informes.	
Precondición	Registrarse en el sistema.	
Poscondición	Salir	
Flujo normal	1	Los clientes deben existir para poder hacer consultas
	2	El sistema solicita los datos de los clientes o grupos.
	3	El sistema almacena los datos proporcionados.
Flujos alternos	*	<b>Cliente no registrado</b>
		Se consulta los movimientos del cliente o grupo con su número de cuenta o de grupo asignado por el sistema.
Excepciones	1	Si se intenta consultar un cliente que no existe el sistema debe advertir la situación y saldrá un mensaje.
Notas	1	Se requiere los siguientes datos del usuario:
		<p>1.1 Login (*): identificador único de cada usuario con el cual se autentificara en el sistema.</p> <p>1.2 Clave de acceso (*): código para acceso al sistema.</p> <p>1.3 Identificación</p> <p>1.6.1 Tipo de identificación</p>

	Operador Médico	
Descripción	Proporciona información sobre el seguro médico de cada cliente.	
Precondición	Registrarse en el sistema.	
Poscondición	Salir	
Flujo normal	1	Los clientes acceden al servicio de salud
	2	
Flujos alternos	*	
		Registra en la especialidad que fue atendido el cliente.
Excepciones	1	
Notas	1	Se requiere los siguientes datos del usuario:
		1.4 Login (*): identificador único de cada usuario con el cual se autenticara en el sistema. 1.5 Clave de acceso (*): código para acceso al sistema.

	Salud	
Descripción	Proporciona visualizar reportes de los clientes	
Precondición	Registrarse en el sistema.	
Poscondición	Salir	
	1	El sistema solicita los datos del cliente
	2	El sistema visualiza el reporte del cliente
Flujos alternos	*	
Excepciones	1	
Notas	1	Se requiere los siguientes datos del usuario:
		1.6 Login (*): identificador único de cada usuario con el cual se autenticara en el sistema. 1.7 Clave de acceso (*): código para acceso al sistema.

# MODELO DE DATOS DEL SISTEMA ASE



## Arquitectura

Después de haber modelado la base de datos procedemos a generar la arquitectura que se implementara en las siguientes capas:

**Capa de presentación (GUI):** presenta el sistema al usuario, muestra la información existente y captura la información del usuario con un mínimo de proceso.

**Capa de negocio (BLL):** Donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Aquí es donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación y con la capa de datos, para solicitar al gestor de la base de datos almacenar o recuperar los datos al sistema.

**Capa de Datos (DAL):** Es donde residen los datos y es la encargada de acceder a los mismos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

**Capa de Entidades de negocio (BEL):** Es la representación de los objetos manejados en el sistema y también de las tablas de la base de datos. Permite el transporte de los datos desde afuera hacia la base de datos y viceversa.

### 4.3.1.7 Pruebas

**Pruebas de Unidad.-** Sirven para comprobar el correcto funcionamiento de un componente concreto del sistema.

En este tipo de pruebas se busca situaciones que expongan las limitaciones de la implementación del componente puede ser tratado como una prueba de caja negra.

También se puede hacer un análisis de su estructura interna que son pruebas de caja blanca.

**Pruebas de integración.-** Son las que se realizan cuando se van juntando los componentes que conforman el sistema y sirven para detectar errores en sus interfaces.

**Pruebas Alfa.**- Una vez realizado el sistema, los responsables del desarrollo realizan pruebas desde el punto de vista de un usuario final, este tipo de pruebas pueden ayudar principalmente a pulir aspectos de la interfaz de usuario del sistema y validaciones que pudieron haberse pasado por alto durante la codificación.

En la siguiente tabla se probará algunas de las características de cada requisito.

Requisito	Característica a probar	Tipo de prueba
<b>Gestión clientes</b>	Crear un cliente y almacenar datos. Crear un cliente con código existente. Crear un cliente con campos obligatorios vacíos. Crear un cliente con valores que no admiten los campos. Modificar los datos de un cliente y actualizar los datos. Modificar los datos de un cliente con valores que no admiten los campos. Buscar un grupo y mostrar todos sus clientes. Buscar un grupo con código no existente. Buscar un grupo con código vacío.	Prueba de caja negra
<b>Gestionar crédito</b>	Generar un crédito con valores que no admiten los campos. Buscar los créditos de un cliente con código no existente. Buscar los créditos de un cliente con código vacío. Modificar los créditos de un cliente.	Prueba de caja negra



## Herramientas

### Visual Estudio 2010

Es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación en este caso Visual C#.

Visual estudio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET, así se pueden crear aplicaciones que se intercomunican entre estaciones de trabajo, pagina web.

Se escogió esta herramienta porque desde hace años atrás ha demostrado ser una excelente plataforma de desarrollo de sistemas.

### Sql server 2000

Es una potente base de datos que aumenta y expande las posibilidades, rendimiento, fiabilidad, calidad y facilidad Ofrece escalabilidad y presenta nuevas soluciones para reducir la carga del servidor, asegurar el funcionamiento continuo y sin cuelgues, y nos aporta una administración avanzada y alta capacidad de configuración. Por todas esas características se eligió este motor de base de datos y además por su gratuidad.

Todas las herramientas están con sus licencias respectivas.

## Hardware

### Clientes

Sistema Operativo	PCS
Windows 8	5
Windows 7	10
Windows XP	5

### Servidor

Sistema Operativo	
Windows XP	I5

Se muestra las pantallas principales del sistema ASES.



Ingresamos con el usuario Administrador.

