

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
ESCUELA DE POSGRADO



**APLICACIÓN DE UNA METODOLOGÍA ÁGIL EN EL DESARROLLO
DE UN SISTEMA DE INFORMACIÓN**

Tesis para optar el grado de Magíster Informática con mención en
Ingeniería de Software que presenta

JAIME HUMBERTO SAMAMÉ SILVA

Dirigido por
DR. JOSÉ ANTONIO POW SANG PORTILLO

Pando, 2013

DEDICATORIA:

A mis queridos padres
Julio Samamé y Merly Silva,
Regina, Milka, mis sobrinas,
y a mis amigos y profesores.



AGRADECIMIENTOS:

Al Profesor Maynard Kong
por su orientación y paciencia.

Al Profesor Antonio Pow Sang Portillo
por su orientación y amistad.

A mis papás
por su apoyo eterno.

A mis compañeros y docentes de la Universidad
por compartir sus experiencias y actitudes,
conocimientos y consejos



“Así que yo les digo: pidan, y se les dará;
busquen, y encontrarán;
llamen, y se les abrirá la puerta.
Porque todo aquel que pide, recibe;
y el que busca, encuentra;
y al que llama, se le abrirá”
Jesucristo

RESUMEN

Las aplicaciones informáticas han contribuido enormemente en el escenario de nuestras vidas, están directas o indirectamente presentes en nuestro día a día, y somos consumidores o desarrolladores de ellas. Cuando cumplimos un papel de desarrolladores, se presentan diversas metodologías al momento de empezar un proyecto de software, dentro de ellas nos ofrecen un punto de vista alternativo a las clásicas y duras, las metodologías ágiles.

El presente proyecto aplica una de estas metodologías ágiles, Programación Extrema (Extreme Programming), en un pequeño proyecto de software, utilizando herramientas de software libre como Java, y como repositorio de datos el estándar XML. El resultado de esta investigación aporta una guía del uso de la metodología ágil en un pequeño proyecto de software que tiene aplicabilidad dentro del ciclo de inteligencia de la información.

Contenido

Parte I:	
Introducción al Estudio	8
Capítulo 1 Introducción	9
Motivación del Estudio.....	10
Propósito del Proyecto.....	13
Contenido de la tesis	19
Capítulo 2 Estado de Arte	21
Introducción al Modelo Ágil	22
Manifiesto Ágil. Valores, Prácticas y Roles	23
Ciclo de vida de un Proyecto XP	30
Otros Métodos Ágiles	32
Dificultades en los principios ágiles	34
Metodologías Ágiles vs Tradicionales.....	39
Estado de Herramientas Similares	43
Ejecución del Proyecto	46
Capítulo 3 Desarrollo del Proyecto	47
Exploración.....	48
Planificación de las Entregas.....	58
Iteraciones.....	60
Producción.....	71
Conclusiones del Proyecto de Tesis	73
Conclusiones	73
Beneficios	74
Dificultades y Observaciones.....	75
Observaciones de Usuarios.....	76
BIBLIOGRAFÍA	78
ANEXOS	81
ANEXO A: Fases de un proyecto en eXtreme Programming	82
ANEXO B: Historias de usuarios	83
ANEXO C: Tareas de Ingeniería	87
ANEXO D: Estructura de datos en XML	98
ANEXO E: Pruebas de Aceptación	101
ANEXO F: Funciones y procesos importantes.....	121
ANEXO G: Estadísticos de Proyectos de Software	123

INDICE DE GRÁFICOS

Figura 1: Muestra el flujo de información entre los módulos y la Organización. Los módulos representan una región del país, donde captan sus eventos para su análisis.....	18
Figura 2: Encuesta de VersionOne (2007): Qué metodología ágil sigue más de cerca.	37
Figura 3: Encuesta de VersionOne (Agosto 2012): Qué metodología ágil sigue más de cerca.	38
Figura 4: Evolución de los largos ciclos de desarrollo en cascada (a) a ciclos iterativos más cortos (b) y a la mezcla que hace XP.(AMARO 2007)	41
Figura 5: Costo del cambio en la ingeniería de software tradicional.	42
Figura 6: Costo del cambio en extreme programming.	42
Figura 7: Página Web que registra incidencias de la ciudadanía.	43
Figura 8: Proceso de atención al vecino a través del Centro de Control de Vigilancia	44
Figura 9: Módulo de Observación del Delito de la Municipalidad de Miraflores	45
Figura 10: Arquitectura de los componentes que se emplearán. El JSDK nos proporciona las librerías más básicas de las cuales se soportan cualquier proyecto Java.....	50
Figura 11: Diseño de tres regiones. Formado por el logo, la navegación y el contenido.....	51
Figura 12: El menú se conforma del Registro, Mantenimiento, y Reportes. Todas las opciones dependen del perfil que corresponda.....	53
Figura 13: Los formularios principales tendrán esta presentación, con los filtros para búsquedas a la izquierda, los resultados a la derecha y la botonera a la derecha.	53
Figura 14: Los formularios que son del tipo cabecera detalle, como en los eventos, presentan los registros de cabecera en la parte superior, y su detalle se cargará en la parte inferior.	54
Figura 15: El diagrama de clases, está conformado por las clases, sus relaciones y la cardinalidad.	55
Figura 16: Diagrama de estados de los eventos. Un evento es registrado, al terminar pasa a Por Procesar, aquí es exportado a XML, en el estado Procesado, se recibe el XML y se revisa su contenido, para pasar al estado Final.	57
Figura 17: La primera iteración considera la creación del formulario de eventos. La funcionalidad de los íconos marcados es desarrollada en la siguiente iteración.	62
Figura 18: El perfil de Informante del módulo de Ica, muestra por defecto el formulario de eventos.....	63
Figura 19: Se muestra los datos del Evento 003. Y la pre-visualización de una foto adjunta.....	63
Figura 20: Si se acepta ver el adjunto, el sistema reconoce el tipo de archivo y lo muestra.....	64
Figura 21: Se selecciona la ubicación del archivo, para exportar los eventos preparados.	64
Figura 22: El archivo normal (que no se genera) y su versión encriptado que se enviará.....	65
Figura 23: El administrador carga el/los archivo(s) enviado por los módulos, visualizando su contenido. Y permitiendo su ingreso a la red interna.	65
Figura 24: El formulario de Eventos pertenece al perfil de Administrador, permite recuperar eventos registrados previamente por los agentes. Los íconos marcados se desarrollan en la última iteración.	66
Figura 25: Mantenimiento de los agentes por módulo. En la figura se pueden visualizar los agentes del módulo de Ica.	67
Figura 26: Mantenimiento de módulos, en la figura se está editando los datos del módulo de Iquitos.	67
Figura 27: Se selecciona el motor de la base de datos, en esta versión se usa Oracle. Para ello se selecciona la ubicación donde se generará el archivo.	68
Figura 28: Versión final ya desarrollado y probado del formulario de ingreso	69
Figura 29: Estructura que define el perfil Administrador o Informante. El archivo posee la lista de usuarios y sus contraseñas junto con el módulo al que pertenecen.....	69
Figura 30: El sistema lee la información del archivo config.xml, e identifica el módulo del usuario. Cada módulo posee su propio archivo config.xml donde se guarda la lista de los usuarios para ese módulo o región.	70
Figura 31: Versión final ya desarrollado y probado del menú principal, en este formulario se muestra el nombre del usuario y el módulo al que pertenece (parte inferior) y las opciones según perfil del menú (parte superior).....	70
Figura 32: Prototipo de pantalla de reporte.....	71

Figura 33: Archivo .jar que contiene la lógica del sistema. El IDE NetBeans proporciona una opción de construir desde un proyecto el archivo jar, y su contenido adicional. 72

INDICE DE TABLAS

Tabla 1: Lista de las principales empresas que siguen la metodología ágil.	12
Tabla 2: Diferencias entre Metodologías Ágiles y no Ágiles.	40
Tabla 3: Ranking de “agilidad” (Los valores más altos representan una mayor agilidad)	41
Tabla 4: Lista de Historias de Usuarios que se atenderán dependiendo de la prioridad y complejidad que presenten cada uno.	49
Tabla 5: Muestra la aplicación de las notaciones en los diferentes elementos que se utilizan al momento de la codificación.	52
Tabla 6: La tabla muestra los roles que se considerarán en la aplicación, cada uno está asociado a opciones de menú. En el caso del rol Informante solo puede manipular sus eventos.	54
Tabla 7: Estructura de datos, donde se detallan las tablas que conforman el sistema. Como información adicional se muestran los campos, el tipo de datos y la descripción.	56
Tabla 8: Indica los aspectos que se deben esperar al momento de la generación de las pruebas de aceptación.	58
Tabla 9: Categorización de las historias de usuarios, por parte del equipo XP.	59
Tabla 10: Plan de Entrega, donde se muestran las iteraciones que se efectuarán sobre las historias priorizadas. Son 3 iteraciones que están formados de 3,3 y 2 historias de usuarios, finalizando en el primer y único release.	59



PARTE 1 Introducción al Estudio

La meta de esta parte del informe es ofrecer una introducción general de la metodología ágil y su utilidad en un mundo cambiante y competitivo. Los argumentos que presentan las metodologías tradicionales frente a la presencia de las ágiles. Y las cifras estadísticas que inclinan la balanza hacia un cambio en el modo de desarrollar software.

Esta parte se compone de los siguientes capítulos:

El capítulo 1 explica los antecedentes y la motivación que llevó a realizar este trabajo, se explican algunos conceptos generales como metodologías ágiles, XML, JAVA, Inteligencia.

En el capítulo 2 se detalla los conceptos y principios de la programación extrema, sus principales procesos en el momento de desarrollar un proyector de software y su comparación con otras metodologías ágiles.

Capítulo 1

Introducción

Objetivos

El objetivo de este capítulo es brindar un marco conceptual para entender el trabajo realizado. Así, en este capítulo:

- Conocerá el propósito de la tesis y la motivación que llevó a realizar este trabajo.
- Sabrá cuál será el aporte de la tesis para los interesados en metodologías ágiles y tecnologías de software libre.
- Se entenderá los conceptos de ciclo de inteligencia, qué procesos lo conforman y cuál de estos procesos será el proceso que desarrollaremos.
- Se explicará el contenido de la tesis, y como está estructurada.

Contenido

1.1 Motivación del Estudio

1.2 Propósito del proyecto

1.3 Contenido de la tesis

1.1 Motivación del Estudio

El incremento de popularidad que están adquiriendo las metodologías ágiles en los últimos años, por empresas de diversos campos, sea de banca, salud, entretenimiento, hardware, defensa, etc. y la experiencia que se tiene al utilizar metodologías tradicionales, han motivado el desarrollar este trabajo.

Hay un interés de saber por qué estas metodologías poco a poco están desplazando o generando híbridos con las metodologías tradicionales, porque es que empresas tan importantes a nivel mundial han enfocado su interés en aplicar metodologías ágiles, cómo se están defendiendo las tradicionales, qué dificultades existen alrededor de estas metodologías, los mitos y paradigmas que se han generado sobre estas.

Se sabe que hemos heredado el uso de metodologías tradicionales de los primeros proyectos de software, los cuales eran grandes proyectos de riguroso control, por lo que se exigía un planeamiento muy calculado acompañado de una documentación muy robusta. Sistemas gubernamentales y aeroespaciales que tenían un desarrollo de hasta 10 años, son claros ejemplos de sistemas pioneros, que dieron las primeras pautas para las metodologías tradicionales. Pero ahora las empresas operan en un entorno global que cambia rápidamente. Debido a esto deben responder frente a nuevas oportunidades y mercados, al cambio en las condiciones económicas, así como al surgimiento de productos y servicios competitivos.

El software es parte de todas las operaciones industriales, de modo que el nuevo software se desarrolla rápidamente para aprovechar las actuales oportunidades, respondiendo así a la amenaza competitiva. Esto conlleva que los sistemas de software consideren por lo general como requerimiento fundamental entregas y desarrollo rápidos.

Debido a que las empresas funcionan en un entorno cambiante, a menudo es prácticamente imposible derivar un conjunto completo de requerimientos de software estable. Los requerimientos iniciales cambian de modo inevitable, porque los clientes encuentran imposible predecir cómo un sistema afectará sus operaciones, cómo

interactuarán con otros sistemas y qué operaciones de usuarios se automatizarán. Es posible que sea sólo hasta después de entregar un sistema, y que los usuarios se acostumbren a éste, cuando se aclaren los requerimientos reales. Incluso, es probable que debido a factores externos, los requerimientos cambien rápida e impredeciblemente. Si esto ocurre el software podría ser obsoleto al momento de su entrega.

Los procesos de desarrollo de software que buscan especificar por completo los requerimientos y, luego, diseñar, construir y probar el sistema, no están orientados al desarrollo rápido de software. A medida que los requerimientos cambian, o se descubren problemas en los requerimientos, el diseño o la implementación del sistema tienen que reelaborarse y probarse nuevamente. Por lo tanto, un proceso convencional en cascada o uno basado en especificación se prolonga con frecuencia, en tanto que el software final se entrega al cliente mucho después de lo que se especificó originalmente.

Existen tipos de software como los sistemas de control crítico para la seguridad, donde es esencial un análisis completo del sistema, resulta oportuno un enfoque basado en un plan. Sin embargo, en un ambiente empresarial de rápido movimiento, esto llega a causar verdaderos problemas. Son esenciales en particular los procesos de diseño que se enfocan en el desarrollo y la entrega de software rápidos.

Durante algún tiempo, se reconoció la necesidad de desarrollo y de procesos de sistema rápidos que administraran los requerimientos cambiantes. IBM introdujo el desarrollo incremental en los 80 (Mills et al., 1980). La entrada de los llamados lenguajes de cuarta generación¹, también de la misma década, apoyó la idea del software de desarrollo y entrega rápidos (Martin, 1981). Sin embargo, la noción prosperó realmente a finales de los 90, con el desarrollo de la noción de enfoques ágiles como el DSDM (Stapleton, 1997), Scrum (Schwaber y Beedle, 2001) y la programación extrema (Beck, 1999; Beck, 2000).

¹ Los lenguajes de cuarta generación son entornos de desarrollo de aplicaciones constituidos por herramientas, tales como compiladores, editores, sistemas de acceso a bases de datos, etc.

La **tabla 1** demuestra el gran interés que está tomando la metodología ágil en las más importantes empresas a nivel mundial. Las metodologías ágiles ofrecen una filosofía que se acomoda a un mundo de constante cambios y alta competencia.

Tabla 1: Lista de las principales empresas que siguen la metodología ágil.
Fuente: João Gama,2012

Nº	Sectores	Usuario
1	Media y Telcos	BBC, BellSouth, British Telecom, DoubleYou, Motorola, Nokia, Palm, Qualcomm, Schibsted, Sony/Ericsson, Telefonica I+D, TeleAtlas, Verizon
2	Software, Hardware	Adobe, Autentia, Biko2, Central Desktop, Citrix, Gailén, IBM, Intel, Microfocus, Microsoft, Novell, OpenView Labs, Plain Concepts, Primavera, Proyectalis, Softhouse, Valtech, VersionOne.
3	Internet	Amazon, Google, mySpace, Yahoo
4	ERP	SAP
5	Banca e Inversión	Bank of America, Barclays Global Investors, Key Bank, Merrill Lynch
6	Sanidad y Salud	Patientkeeper, Philips Medical
7	Defensa y Aeroespacial	Boeing, General Dynamics, Lockheed Martin
8	Juegos	Blizzard, High Moon Studios, Crytek, Ubisoft, Electronic Arts
9	Otros	3M, Bose, GE, UOC, Ferrari

1.2 Propósito del Proyecto

De lo mencionado en el punto 1.1 vemos el auge que están teniendo las metodologías ágiles en el ámbito tecnológico. Es el propósito principal de este trabajo aplicar una metodología ágil en un pequeño módulo, para verificar e identificar sus bondades y dificultades. Para ello es necesario primero identificar algunos conceptos que se utilizarán en el proyecto.

1.2.1 Conceptos Generales

Como metodologías que aplicamos para el desarrollo de proyectos de software, tenemos las tradicionales y las ágiles. La metodología ágil intenta evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados. Es más adaptativo y flexible al proyecto que se desarrolla. De estas metodologías se encuentra la Programación Extrema o eXtreme Programming (XP), desarrollada por Kent Beck en 1990.

Como lenguaje de programación utilizaremos Java, que es un lenguaje de programación de alto nivel orientado a objetos, desarrollado por James Gosling en 1995. El lenguaje en sí mismo toma mucha de su sintaxis de C, Cobol y Visual Basic, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La memoria es gestionada mediante un recolector de basura.

Como elemento de almacenamiento de datos y de configuración del software utilizaremos XML, que es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C). XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. (VariableX, 2009)

Es necesario saber, en que se aplicaran estos conceptos. Para eso debemos definir un término que se usará, que es el de *Inteligencia*.

La Inteligencia es el conjunto de actividades basadas en un ciclo de producción consistente en la obtención, recolección, búsqueda, acopio, procesamiento y difusión de información destinada a un usuario o consumidor final, para la toma de decisiones en materia de seguridad nacional, con el objeto de prevenir sobre amenazas, riesgos y oportunidades¹.

Una de las aplicaciones de la Inteligencia se refleja en la seguridad del país, frente a riesgos internos (económicos, políticos, militares, sicosociales, etc.) y externos (por parte de los países vecinos). Otra aplicación de la inteligencia es el desarrollo tecnológico.

El país cuenta con organismos que aseguran la defensa de una manera preventiva (organismos de inteligencia) como operativa (fuerzas militares).

El país cuenta con el Sistema de Inteligencia Nacional (SINA) que es un conjunto de instituciones del Estado funcionalmente vinculados, que actúan coordinadamente en la producción de inteligencia, para la toma de decisiones, frente a las amenazas y/o riesgos actuales y potenciales contra la seguridad nacional. El SINA forma parte del Sistema de Seguridad Nacional.

Estas instituciones del Estado siguen una doctrina para la generación de inteligencia, la cual se denomina “el ciclo de inteligencia”.

Dentro del ciclo de Inteligencia, para la defensa y prevención de la seguridad del País, se encuentran las siguientes fases o etapas:

Captura y recolección de información, en esta etapa los agentes y colaboradores que forman parte del sistema de inteligencia, proporcionan información de eventos y

¹ DINI, Dirección Nacional de Inteligencia. “Definición.”. Lima 2007

actividades a los órganos de inteligencia, valiéndose de diversos medios para comunicar la información captada. Esta etapa será objeto de nuestro estudio, y aplicación de la metodología ágil y el uso de JAVA y XML. Actualmente esta etapa no contempla un sistema de información, se apoyan en el empleo de correos electrónicos, discos, usb's, escaneos, copias, etc.

Categorización de la Información, una vez que la información es enviada por parte de los agentes y colaboradores, los órganos de inteligencia se encargan de registrarla nuevamente, pero dentro de una base de datos estructurada. Posteriormente se categoriza dependiendo del factor al que pertenezca la información (político, sicosocial, económico, amenazas internas, etc.).

Procesamiento de la información, la información ya categorizada y registrada en una base de datos, es procesada mediante la vinculación de temas, personas, organizaciones, además de validar el contenido de la información origen y distribuir la información procesada a los órganos de análisis.

Análisis de la Información, consiste en determinar futuros escenarios, conclusiones, proyecciones, sugerencias sobre las informaciones que fueron procesadas, generando documentos de inteligencia (estudios de inteligencia, apreciaciones, notas de inteligencia, informes, etc.)

Distribución a los Órganos decisores, es el envío de los documentos de inteligencia a las instituciones decisoras (ministerios, presidente, fuerzas armadas, etc.).

1.2.1 Breve descripción del Problema

Por una parte, se debe conocer a fondo el marco de trabajo de la metodología ágil que hemos seleccionado, que en este caso es XP, para ello se ha compilado información teórica sobre Programación Extrema (XP), y práctica de algunos trabajos y ejemplos desarrollados con XP.

Por otra parte, con respecto al módulo que se desarrollará, debemos saber que existe una gran cantidad de información que se registra y envía por parte de colaboradores a nivel del interior del país, este volumen de información en bruta constituye un “cuello de botella” dentro del proceso de Inteligencia (etapa de captura de información). Dicha información es enviada actualmente por correos convencionales, teléfono, fax, etc. No cuenta con una herramienta informática estructurada que agilice el proceso de registro. Una vez que la información es enviada por los medios mencionados, los órganos de inteligencia deben nuevamente registrar la información a una base de datos interna estructurada y ordenada.

Este proceso toma mucho tiempo debido a que la mayoría de los órganos de inteligencia cuentan con 2 redes (externa e interna) por motivos de seguridad. Lo cual obliga al personal respectivo a abrir correos y descargarlos en un archivo de texto (por el lado de la red externa), y luego registrar la información en una base de datos, que se encuentra en la red interna, copiando y pegando los segmentos del archivo de texto y el contenido multimedia que pueda adjuntarse como audios, videos, fotos, otros archivos (ppt, doc, pdf, xls, etc.).

1.2.3 Propósito del Proyecto de Tesis

El propósito del proyecto, es aplicar una metodología ágil junto con herramientas y tecnologías estándares (Java y XML) para desarrollar una interface que satisfaga la etapa de Captura de Información. La metodología de análisis y diseño estará basada en eXtreme Programming y el lenguaje utilizado para describir las aplicaciones es UML.

Este proyecto está enfocado en analizar, diseñar e implementar una interface que sistematice la etapa de Captura de Información, mediante una herramienta estándar (JAVA) que recoja la información origen, que son los eventos e incidentes que acontecen en determinada región del país, y que son de interés para la seguridad y desarrollo nacional, por ejemplo el seguimiento de una huelga, toma de puentes o

carreteras, mitin, ataque terrorista, etc. Esta información se guardará dentro de una plantilla XML con los campos necesarios (informante, módulo y lugar, fecha de evento, número del informe, título y descripción del evento) y complementarios (informe respuesta, documentos adjuntos, título y breve descripción del adjunto), y genere finalmente un archivo XML protegido (encriptado), que será enviado al órgano de inteligencia que centralizará los archivos, para luego de forma automática ingresar la información a una base de datos convencional. Los documentos adjuntos mencionados pueden ser fotos, audios, videos, afiches/volantes escaneados, otros archivos, a los cuales se les registrará un título o encabezado y una breve descripción.

Se ha escogido una metodología ágil, debido al tamaño del proyecto, el cual implica pocas iteraciones para obtener el producto final.

De igual modo, se almacena el contenido de los eventos e incidentes en archivos XML, los cuales brindan una estructura flexible, ordenada y segura para nuestros datos. La tecnología actual utiliza como estándar este tipo de archivos para enviar información por internet, existe todo un soporte en los lenguajes de programación que administran eficientemente estos archivos. Y la arquitectura de Java, no es ajena a manejar archivos XML.

El proyecto que se desarrolla será una primera versión de una herramienta que ayudará a los informantes (agentes y colaboradores) que están esparcidos en diferentes regiones del país, llamado *módulo* los cuales registran su información de una manera clásica (con un procesador de texto e internet). No cuentan con algún tipo de herramienta informática que los organice y enfoque su trabajo. Por lado del receptor, el beneficio será mucho mayor, actualmente el tiempo de registrar una información con adjuntos toma alrededor de 10 minutos, debido a que la información de un módulo llega en un solo archivo de texto, el cual hay que ordenarlo, separar los adjuntos (mayormente fotos) que llegan insertados en el documento, relacionar a algún pedido de ampliación (en caso que la información que el informante envía es por un pedido explícito de la organización) y registrar los eventos por separado dentro de la base de datos interna.

Con esta herramienta las organizaciones podrán seleccionar los archivos enviados desde provincia o *módulo*, verificar la información y podrán generar las

sentencias para ingresar en la base de datos de la red interna. La **figura 1**, esquematiza el flujo de la información, los módulos envían la información de interés mediante un archivo encriptado en formato XML. El cual será consolidado por la Organización para su procesamiento e ingreso en la base de datos relacional.

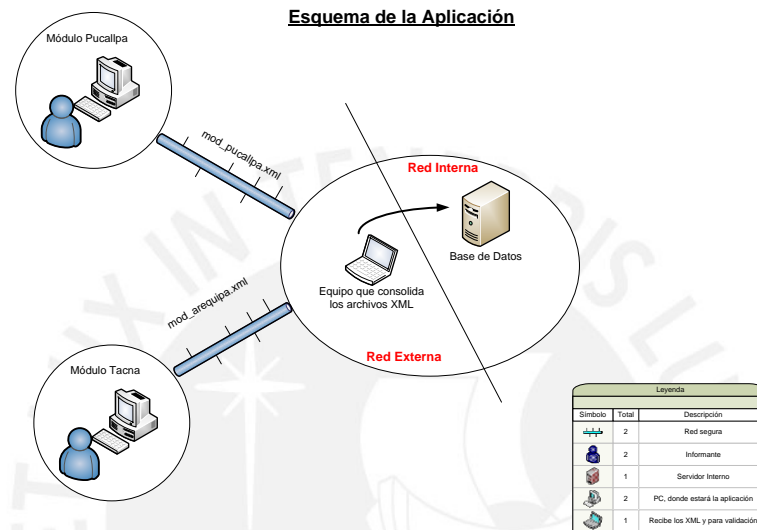


Figura 1: Muestra el flujo de información entre los módulos y la Organización. Los módulos representan una región del país, donde captan sus eventos para su análisis.

1.2.4 Objetivos de la Investigación

La investigación presenta los siguientes objetivos:

Objetivo General

Analizar el enfoque expuesto por las metodologías ágiles en el desarrollo de una aplicación pequeña, utilizando herramientas estándares.

Objetivos Específicos

- Conocer toda la filosofía que envuelve a las metodologías ágiles, particularmente la Programación Extrema (XP).
- Conocer las dificultades que han sido encontradas al aplicar metodologías ágiles. Y en particular identificar estas dificultades en el proyecto de software.

- Aplicar la metodología analizada en una necesidad de proyecto de software, aplicada en la Captura de información, que forma parte del ciclo de inteligencia de la información.
- Identificar semejanzas y diferencias, mejoras o dificultades con respecto a las metodologías tradicionales.

Caso de Estudio

Se aplica el ciclo de vida de la Programación Extrema (XP) en un proyecto de software mediano/pequeño para comparar si efectivamente esta metodología es más adecuada para proyectos de este alcance. Se cuenta con el apoyo de 2 usuarios que participarán en el desarrollo de la aplicación, la cual consiste en registrar eventos o situaciones de riesgo para la seguridad del país. Dicha aplicación no existe, por lo que será una oportunidad para sistematizar dicho proceso.

1.3 Contenido de la tesis

La Tesis está organizada en 3 capítulos, la bibliografía consultada seguida de las conclusiones y los anexos que complementan el informe.

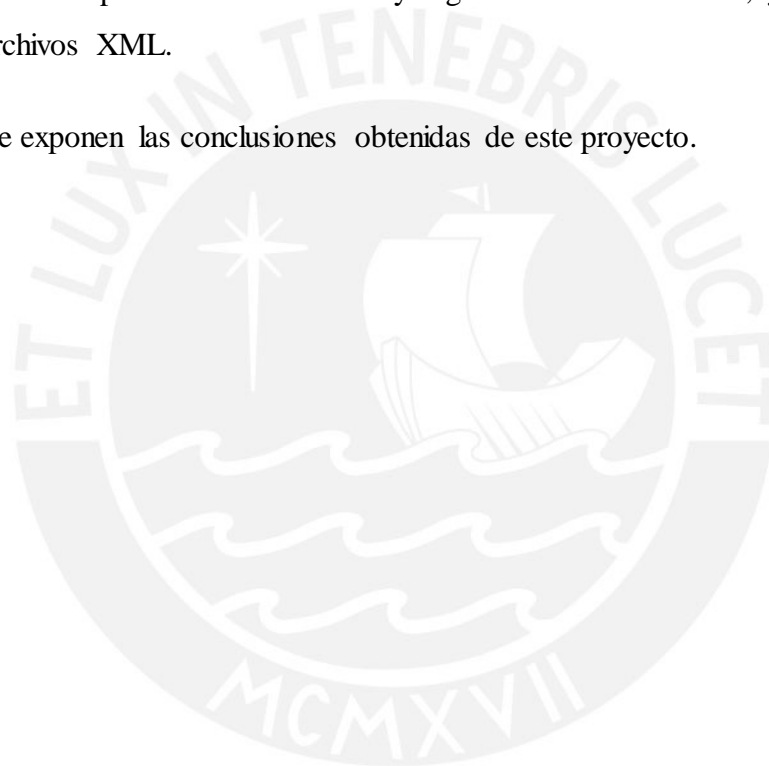
El primer capítulo, se describe la motivación que llevó a desarrollar el informe, así como describe el problema que se pretende resolver, que es aplicar una metodología ágil en la solución del problema, se revisa algunos términos de la metodología y conceptos utilizados de tal forma que el lector pueda entender el contexto del informe.

El segundo capítulo, hace una introducción a las metodologías ágiles, se revisa los principios, ciclo de vida y la filosofía que rodea a dicha metodologías. También se revisa brevemente otras metodologías ágiles y sus características. El capítulo evalúa los

inconvenientes y dificultades que presenta el manifiesto ágil. Finalmente se hace una comparación entre la propuesta ágil y las metodologías tradicionales.

El tercer capítulo describe el desarrollo del proyecto, la definición del marco de desarrollo, se obtienen las historias de los usuarios, y se definen las etapas que se seguirán de acuerdo a la metodología aplicada, obteniendo finalmente el producto deseado. Además, se hará una breve demostración del funcionamiento de la aplicación mostrando por rol o perfil la administración y registro de la información, y como esta se deposita en archivos XML.

Finalmente, se exponen las conclusiones obtenidas de este proyecto.



Capítulo 2

Estado de Arte

Objetivos

El objetivo de este capítulo es brindar un marco conceptual de las metodologías ágiles. Así en este capítulo:

- Se conocerá el origen de las metodologías ágiles, un poco de la historia y propósito de las mismas.
- Se sabrá cuál es la filosofía y composición de las metodologías ágiles.
- Se enfocará en la metodología ágil que se escogió para el desarrollo de la tesis, que es Extreme Programming (XP).
- Se revisará algunas otras opciones de metodologías ágiles y se comparará con las metodologías tradicionales.

Contenido

- 2.1 Introducción al Modelo Ágil
- 2.2 Manifiesto Ágil. Valores, Prácticas y Roles
- 2.3 Ciclo de Vida de un Proyecto XP
- 2.4 Otros Métodos Ágiles
- 2.5 Dificultades en los principios ágiles
- 2.6 Metodologías Ágiles vs Tradicionales
- 2.7 Estado de Herramientas Similares

2.1 Introducción al Modelo Ágil

En la década de 1980 y a inicios de la siguiente, había una visión muy difundida de que la forma más adecuada para lograr un mejor software era mediante una cuidadosa planeación del proyecto, aseguramiento de calidad formalizada, el uso de métodos de análisis y el diseño apoyado por herramientas CASE, así como procesos de desarrollo de software rigurosos y controlados. Esta percepción proviene de la comunidad de ingeniería de software, responsable del desarrollo de grandes sistemas de software de larga duración como los sistemas antes mencionados: aeroespaciales y gubernamentales. (SOMMERVILE 2011).

Para este tipo de sistemas se requiere grandes equipos de desarrolladores que trabajan para diferentes compañías. A menudo los equipos estaban geográficamente dispersos y laboraban por largos periodos. Un ejemplo de este tipo de software es el sistema de control de una aeronave moderna, que puede tardar hasta 10 años desde la especificación inicial hasta la implementación. (SOMMERVILE 2011).

Estos enfoques basados en un plan incluyen costos operativos significativos en la planeación, el diseño y la documentación del sistema. Dichos gastos se justifican cuando debe coordinarse el trabajo de múltiples equipos de desarrollo, cuando el sistema es un sistema crítico y cuando numerosas personas intervendrán en el mantenimiento del software a lo largo de su vida.

Sin embargo, cuando este engorroso enfoque de desarrollo basado en la planeación se aplica a sistemas de negocios pequeños y/o medianos, los costos que se incluyen son tan grandes que dominan el proceso de desarrollo del software. Se invierte más tiempo en diseñar el sistema, que en el desarrollo y la prueba del programa. Conforme cambian los requerimientos del sistema, resulta esencial la reelaboración y, en principio al menos, la especificación y el diseño deben modificarse con el programa.

En la década de 1990 el descontento con estos enfoques engorrosos de la ingeniería de software condujo a algunos desarrolladores de software a proponer nuevos “métodos ágiles”, los cuales permitieron que el equipo de desarrollo se enfocara en el software en lugar del diseño y la documentación. (SOMMERVILE 2011).

Los métodos ágiles se apoyan en el enfoque incremental para la especificación, el desarrollo y la entrega del software. Son más adecuados para el diseño de aplicaciones en que los requerimientos del sistema cambian, por lo general, rápidamente durante el proceso de desarrollo. Tienen la intención de entregar con prontitud el software operativo al cliente, quienes seguramente pondrán requerimientos nuevos y variados para incluir en posteriores iteraciones del sistema. Se dirigen a simplificar el proceso burocrático al evitar trabajo con valor dudoso a largo plazo, y a eliminar documentación que quizá nunca se emplee. (AMARO 2007)

2.2 Manifiesto Ágil. Valores, Prácticas y Roles

La filosofía detrás de los métodos ágiles se refleja en el manifiesto ágil, que acordaron muchos de los desarrolladores líderes de estos métodos. Este manifiesto afirma:

Estamos descubriendo mejores formas para desarrollar software, al hacer y al ayudar a otros a hacerlo. Gracias a este trabajo llegamos a valorar:

- *A los individuos y las interacciones sobre los procesos y las herramientas.*
- *Al software operativo sobre la documentación exhaustiva.*
- *La colaboración con el cliente sobre la negociación del contrato.*
- *La respuesta al cambio sobre el seguimiento de un plan.*

Esto es, aunque exista valor en los objetos a la derecha, valoraremos más los de la izquierda. (SOMMERVILE 2011)

La programación extrema es uno de los procesos ágiles de desarrollo de software más populares. 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. Creer que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.(PARDO 2010)

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

Además de aplicar una metodología, se cuenta con plataformas, herramientas y tecnologías poderosas que nos permiten cubrir los requerimientos. En este proyecto emplearemos como lenguaje de desarrollo JAVA y como manejo de la información el estándar XML.

Extreme Programming (XP) es una disciplina de desarrollo de software basada en los valores de simplicidad, comunicación, retroalimentación y valor. En XP cada participante del proyecto es una parte integral del Equipo. El equipo se forma alrededor de un representante llamado el Cliente, que se sienta con el equipo y trabaja con ellos diariamente. Los equipos de XP usan una forma simple de planificación y seguimiento para decidir qué se debe hacer a continuación y para predecir cuando el proyecto será finalizado. Focalizado en el valor del negocio, el equipo produce software en una serie de pequeños entregables integrados, que aprueban todos los tests que ha definido el Cliente. Los programadores de XP trabajan juntos en pares y como un grupo, con un código testado de forma obsesiva y de diseño simple, mejorando el diseño continuamente para mantenerlo siempre acorde a las necesidades actuales. (CANÓS 2004)

Se considera que utilizar una metodología tradicional como RUP es muy dura y compleja para un proyecto como este. XP promueve los siguientes valores:

- **Comunicación:** XP se enfoca en el entendimiento persona-a-persona de los problemas que van presentando en el ciclo de desarrollo, minimizando la documentación formal y maximizando la interacción cara-a-cara. A modo de

ejemplo, se puede mencionar que a través de la programación en pareja se aumenta la comunicación entre los programadores, mientras que la presencia del cliente On-site, facilita el intercambio de ideas con el equipo de desarrollo.

- **Coraje:** Todo cambio requiere coraje. Algunos definen al coraje como la capacidad de hacer lo correcto aún con la presión de hacer algo distinto. En muchos casos, los integrantes de un grupo toman las decisiones equivocadas no por falta de coraje, sino por falta de convicción. Los valores del equipo deben estar fuertemente alineados. XP intenta crear un soporte liviano con mucha atención a los valores de comunicación, sencillez y retroalimentación de forma que la confianza y el coraje entre los actores fluya naturalmente.
- **Simplicidad:** XP le pide a cada miembro del equipo que construya lo más sencillo que funcione hoy. XP se basa en hacer algo simple hoy y crear un ambiente donde el costo del cambio sea lo más bajo posible.
- **Feedback:** Los programadores obtienen el estado del software minuto a minuto a través de los Test de Unidad y la Integración Continua. El cliente consigue ver el estado del proyecto a lo largo de la iteración a través de los Test de Aceptación. (CANÓS 2004).

XP además está basada en diferentes prácticas de programación, de las que enunciaré las 12 principales:

- **Testeos continuos (pruebas unitarias/ de aceptación):** Se tiene que establecer un tiempo para efectuar las pruebas de aceptación del sistema donde se definirán las entradas al sistema y los resultados esperados de estas entradas. Se recomienda automatizar este tipo de pruebas para poder generar simulaciones del funcionamiento del sistema. Para realizar estas simulaciones automatizadas, se pueden utilizar Ambientes de Prueba. Un buen ejemplo de un marco de trabajo de pruebas unitarias es el JUnit para Java y QUnit para JavaScript.
- **Planificación (historias de usuarios, el equivalente a los requerimientos o requisitos):** El usuario tendrá que realizar lo siguiente: escribir sus necesidades y

definir las actividades que efectuará el sistema. Con esa información se crean las historias del usuario (User Stories). Dependiendo de la complejidad del problema se consideran suficientes para formar el llamado Plan de Liberación, el cual define de forma específica los tiempos de entrega de la aplicación para recibir retroalimentación por parte del usuario. Por regla general, cada una de las Historias del usuario suelen necesitar de una a tres semanas de desarrollo. Son muy importantes y tienen que ser una constante las reuniones periódicas durante esta fase de planificación. Estas pueden ser a diario, con todo el equipo de desarrollo para identificar problemas, proponer soluciones y señalar aquellos puntos a los que se les ha de dar más importancia por su dificultad o por su punto crítico.

- Pequeñas mejoras (frecuentes entregas): Colocan un sistema sencillo en producción rápidamente que se actualiza de forma rápida y constante permitiendo que el verdadero valor de negocio del producto sea evaluado en un ambiente real. Estas entregas no pueden pasar las 2 o 3 semanas como máximo.
- Sistema de metáforas (nombres claros): desarrollada por los programadores al inicio del proyecto, define una historia de cómo funciona el sistema completo. XP estimula historias, que son breves descripciones de un trabajo de un sistema en lugar de los tradicionales diagramas y modelos UML (Unified Modeling Language). La metáfora expresa la visión evolutiva del proyecto que define el alcance y propósito del sistema. Las tarjetas CRC (Clase, Responsabilidad y Colaboración) también ayudarán al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos y define sus responsabilidades (lo que ha de hacer) y las colaboraciones con las otras clases (cómo se comunica con ellas). La metáfora es una descripción evocativa simple sobre cómo funciona el programa; por ejemplo "este programa funciona como una colmena de abejas, que salen a buscar polen y lo traen de vuelta a la colmena", podría ser una descripción para un sistema de recuperación de información a través de agentes.
- Diseño simple (más rápido, funciones necesarias): Se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que

cumpla los requerimientos. *Simple Design* se enfoca en proporcionar un sistema que cubra las necesidades inmediatas del cliente, ni más ni menos. Este proceso permite eliminar redundancias y rejuvenecer los diseños obsoletos de forma sencilla. Refactorización del código (simplificarlo): Permite a los equipos de programadores XP mejorar el diseño del sistema a través de todo el proceso de desarrollo. Los programadores evalúan continuamente el diseño y recodifican lo necesario. La finalidad es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente.

- Programación por parejas (código/marco global): Uno de los principios más radicales y en el que la mayoría de gerentes de desarrollo pone sus dudas. Requiere que todos los programadores XP escriban su código en parejas, compartiendo una sola máquina. De acuerdo con los experimentos, este principio puede producir aplicaciones más buenas, de manera consistente, a iguales o menores costes. Aunque el pair-programming puede no ser para todo el mundo.
- Integración continua (trabajar y renovar la versión): Los equipos mantiene integrado al sistema todo el tiempo. Los equipos XP realizan construcciones muchas veces por día. El beneficio de esta práctica lo vemos al pensar en esos proyectos que todos oímos (o incluso fuimos parte) en donde la construcción se realizaba semanalmente (o incluso con menos frecuencia), y que usualmente termina en un "infierno de integración", donde todo se rompe y nadie sabe por qué. La integración infrecuente lleva a problemas serios en el proyecto de software. Primero, aunque la integración es crítica para entregar buen software, el equipo no la práctica y a menudo la delega a personas que no están familiarizadas con el sistema completo. Segundo, la integración infrecuente suele generar código con errores. Al momento de integrar aparecen problemas que no se detectaron en ninguna de las pruebas del software sin integrar. Tercero, los procesos de integración débiles generan largos períodos de "congelamiento" del código. El congelamiento de código (o "code freeze") significa que por largos períodos los programadores no pueden agregar nuevas

características, aunque sea necesario. Esto debilita nuestra posición en el mercado y con los usuarios finales.

- Programación estandarizada/simple (cumplir función/seguir estándar): Los equipos XP usan un estándar de código en común, de manera que el código del sistema se vea como si fuera escrito por una única persona muy competente. No importa mucho el estándar en sí mismo: lo importante es que el código se vea familiar, para permitir la propiedad colectiva.
- Ritmo sostenible (no sobrecargar al equipo/sin picos de trabajo): La semana de 40 horas: la programación extrema sostiene que los programadores cansados escriben código de menor calidad. Minimizar las horas extras y mantener los programadores frescos, generará código de mayor calidad. Como dice Beck, está bien trabajar tiempos extra cuando es necesario, pero no se ha de hacer durante dos semanas seguidas.
- Relación con el cliente ("el cliente no ayuda al equipo, es parte de él): Se le dará poder para determinar los requerimientos, definir la funcionalidad, señalar las prioridades y responder las preguntas de los programadores. Esta fuerte interacción cara a cara con el programador disminuye el tiempo de comunicación y la cantidad de documentación, junto con los altos costes de su creación y mantenimiento. Este representante del cliente estará con el equipo de trabajo durante toda la realización del proyecto.

En XP se definen los siguientes roles:

- **Cliente:** Determina la funcionalidad que se pretende en cada iteración y define las prioridades de implementación según el valor de negocio que aporta cada Historia. El cliente también es responsable de diseñar y ejecutar los Test de Aceptación. En resumen tenemos:
 - Escribe "Historia de Usuarios" y especifica las pruebas funcionales.
 - Establece prioridades y explica las historias de usuarios.
 - Puede ser o no un usuario final.
 - Tiene autoridad para decidir cuestiones relativas a las historias.

- **Programador:** Es responsable de implementar las Historias solicitadas por el cliente. Además, estima el tiempo de desarrollo de cada Historia para que el cliente pueda asignarle prioridad dentro de alguna iteración. Cada iteración incorpora nueva funcionalidad de acuerdo a las prioridades establecidas por el cliente. El Programador también es responsable de diseñar y ejecutar los Test de Unidad del código que ha implementado o modificado. En resumen:
 - Hace estimaciones sobre las historias.
 - Define tareas a partir de las historias y hace estimaciones.
 - Implementa las historias y las pruebas unitarias
- **Tracker:** Consiste en seguir la evolución de las estimaciones realizadas por los programadores y compararlas con el tiempo real de desarrollo. De esta forma, puede brindar información estadística en lo que refiere a la calidad de las estimaciones para que puedan ser mejoradas. Otra de las tareas que merece ser señalada, consiste en visitar a todos los programadores durante la iteración y analizar cuanto tiempo de trabajo le falta para implementar sus Historias y cuanto es que se había estimado para ellas. Con esta información, pueden tener una idea global del progreso de la iteración y evaluar las acciones que se deben tomar.
- **Coach:** Es el responsable del proceso en general. Se encarga de iniciar y de guiar a las personas del equipo en poner en marcha las 12 prácticas.
- **Manager:** Se encarga de organizar las reuniones (Planificación de la iteración, de la entrega, etc.), se asegura que el proceso de desarrollo se esté cumpliendo y registra los resultados de las reuniones para ser analizados en el futuro. Es de alguna forma el que responde al inversionista en lo que respecta a la evolución del desarrollo. (CANÓS 2004).

Esta metodología es simple y muy dinámica, durante los estudios llevados en la maestría ha habido un interés en aplicarla en un proyecto, y conocerla a profundidad. Se tiene mucha experiencia trabajando con RUP y UML y se ha visto muchas veces esfuerzos innecesarios en la elaboración de documentos que son tediosos, y rígidos. En

XP se ha encontrado una mayor adaptabilidad de las iteraciones y entregables que se presentarán para el proyecto.

2.3 Ciclo de vida de un Proyecto XP

El ciclo de vida de un proyecto XP, se compone de las siguientes 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.
- **Planificación de la Entrega (*Release*):** En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por *tiempo*, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos

puntos se pueden completar. Al planificar según *alcance* del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Velocidad del Proyecto: Cantidad de historias de usuarios por Iteración

Si tuviéramos:

- Iteración 1: 20 historias de usuarios.
- Iteración 2: 22 historias de usuarios.
- Iteración 3: 17 historias de usuarios.

La velocidad (promedio) del proyecto estaría dado por: $(20+22+17)/3 = 20 \text{ hu/iteración}$

○ Planeación por tiempo: Tenemos la siguiente fórmula

Cantidad de historias a ser completadas = #iteraciones * velocidad de proyecto

Si tuviéramos 17 iteraciones * $20 \text{ hu/iteración} = 340$ historias a realizarse

○ Planeación por alcance: Tenemos la siguiente fórmula

Cantidad de iteraciones = total semanas estimadas / velocidad del proyecto

Cantidad de iteraciones = 12 semanas (3 meses) / $20 \text{ hu/iteración} = 1$ iteración

Cada iteración debe durar entre 1 y 2 semanas.

- **Iteraciones:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de menos de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fueren la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.
- **Producción:** La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible

que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento). (WELLS 2009).

2.4 Otros Métodos Ágiles

Pero XP no es la única alternativa con respecto a los métodos ágiles, otra opción interesante es Scrum, que tiene mucha presencia en el medio, se presenta como contrapunto a PMBOK y PRINCE2. Otros ejemplos de metodologías ágiles son por ejemplo:

- **Dynamic Systems Development Method (DSDM):** Metodología ágil más veterana y la que más se aproxima a los métodos tradicionales, su implantación incluso permitiría alcanzar un nivel 2 de madurez según CMMI, al igual que muchas de las metodologías ágiles.
- **Agile Modeling:** Metodología para el modelado y la generación de documentación que se encuentra alineado con los principios del desarrollo ágil y que puede ser utilizado como sustituto del UML estándar.
- **Feature Driven Development (FDD):** Metodología de desarrollo de software orientada a la generación de valor para el cliente. (CANÓS 2004).
- **Scrum:** Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el ScrumMaster, que mantiene los procesos y trabaja de forma similar al director de proyecto, el ProductOwner, que representa a los stakeholders (interesados externos o internos), y el Team que incluye a los desarrolladores.
- **Cristal:** la cual identifica con colores diferentes cada método, y su elección debe ser consecuencia del tamaño y criticidad del proyecto, de forma que los de mayor tamaño, o aquellos en los que la presencia de errores o desbordamiento de

agendas implique consecuencias graves, deben adoptar metodologías más pesadas. De esta forma se pretende obtener mayor rentabilidad en el desarrollo de proyectos de software, Los métodos Crystal no prescriben prácticas concretas, porque están en continuo cambio.

- **Desarrollo Adaptable de Software (ASD):** Metodología desarrollada por Jim Highsmith, después de trabajar muchos años con metodologías predictivas, concluyó que son defectuosas. Reconoce que en cada iteración se producirán cambios e incluso errores.
- **Kanban:** No es una técnica específica del desarrollo software, su objetivo es gestionar de manera general como se van completando tareas, pero en los últimos años se ha utilizado en la gestión de proyectos de desarrollo software, a menudo con Scrum (lo que se conoce como Scrumban). Kanban es una palabra japonesa que significa “tarjetas visuales” (kan significa visual, y ban tarjeta). Esta técnica se creó en Toyota, y se utiliza para controlar el avance del trabajo, en el contexto de una línea de producción.
- **Desarrollo dirigido por características:** Es un proceso ágil que se basa en construir iteraciones cortas que produzcan incrementos funcionales en el software que los clientes y personas encargadas de la gestión del proyecto puedan ver, analizar y aprobar.

En cada iteración se trata de construir una característica, definida como un incremento en la funcionalidad de un producto con significado para el cliente. Esta metodología de desarrollo consiste en cinco procesos secuenciales, los cuales se citan a continuación:

- Especificación de la arquitectura global del sistema.
- Construcción de la lista de características que componen el sistema.
- Planificación por característica.
- Diseño por característica.
- Construcción por característica.

La principal ventaja de esta metodología es que está orientada y gobernada por características, que son un concepto clave en el desarrollo de líneas de productos software.

- **Desarrollo Dirigido por Casos de Pruebas:** Es un proceso ágil, iterativo e incremental, en el que el sistema se desarrolla para satisfacer unos casos de prueba bien definidos antes de construir cada iteración.

El primer paso a realizar de acuerdo con esta metodología es definir una lista de requisitos. A continuación, se aplica el siguiente proceso de forma iterativa hasta completar el proyecto:

- Seleccionar uno o más requisitos.
- Escribir casos de prueba que verifiquen que dichos requisitos se satisfacen.
- Desarrollar el sistema de la forma más sencilla posible para pasar los casos de prueba.
- Verificar que el conjunto de pruebas funciona correctamente.
- Refactorizar la solución desarrollada en caso de que fuese necesario.
- Actualizar la lista de requisitos.

Este proceso tiene dos importantes virtudes: (1) está muy orientada a satisfacer los requisitos; y (2) genera software fiable, si los casos de prueba están bien diseñados. Al desarrollar el software por grupos de requisitos, podría considerarse que en cierto modo está orientado o dirigido por características.

2.5 Dificultades en los principios ágiles

En la práctica, los principios que envuelven a los métodos ágiles son a veces difíciles de cumplir:

- Aunque es atractiva la idea de involucrar al cliente en el proceso de desarrollo, su éxito radica en tener un cliente que desee y pueda pasar tiempo con el equipo con el equipo de desarrollo, y éste represente a todos los participantes del sistema. Los representantes del cliente están comúnmente sujetos a otras presiones, así que no intervienen por completo en el desarrollo del software.

- Quizás algunos miembros del equipo no cuenten con la personalidad adecuada para la participación intensa característica de los métodos ágiles y, en consecuencia, no podrán interactuar adecuadamente con otros miembros del equipo.
- Priorizar los cambios sería extremadamente difícil, sobre todo en sistemas donde existen muchos participantes. Cada uno por lo general ofrece diversas prioridades a diferentes cambios.
- Mantener la simplicidad requiere trabajo adicional. Bajo la presión de fechas de entrega, es posible que los miembros del equipo carezcan de tiempo para realizar las simplificaciones deseables al sistema.
- Muchas organizaciones, especialmente las grandes compañías, pasan años cambiando su cultura, de tal modo que los procesos se definan y continúen. Para ellas, resulta difícil moverse hacia un modelo de trabajo donde los procesos sean informales y estén definidos por equipos de desarrollo.

La mayoría de los proyectos de software incluyen prácticas de los enfoques ágiles como los basados en un plan. Para decidir sobre el equilibrio entre ambos enfoques se deben responder algunas preguntas técnicas, humanas y organizacionales:

- ¿Es importante tener una especificación y un diseño muy detallados antes de dirigirse a la implementación? Si es así entonces un enfoque basado en un plan es recomendable.
- ¿Es práctica una estrategia de entrega incremental, donde se dé el software a los clientes y se obtenga así una rápida retroalimentación de ellos? Si es así, un enfoque basado en método ágil es recomendable.
- ¿Qué tan grande es el sistema que se desarrollará? Los métodos ágiles son más efectivos cuando el sistema logra diseñarse con un pequeño equipo asignado que se comunique de manera **informal**. Esto sería imposible para los grandes sistemas que precisan equipos de desarrollo más amplios, de manera que tal vez se utilice un enfoque basado en un plan.

- ¿Qué tipo de sistema se desarrollará? Los sistemas que demandan mucho análisis antes de la implementación, por lo general, necesitan un diseño bastante detallado para realizar este análisis. En tal circunstancia, quizá sea mejor un enfoque basado en un plan.
- ¿Cuál es el tiempo de vida que se espera del sistema? Los sistemas con lapsos de vida prolongados podrían requerir más documentación de diseño, para comunicar al equipo de apoyo los propósitos originales de los desarrolladores del sistema. Pero los defensores de las metodologías ágiles argumentan que la documentación no se encuentra actualizada, ni se usa mucho para el mantenimiento del sistema a largo plazo.
- ¿Qué tecnologías se hallan disponibles para apoyar el desarrollo del sistema? Los métodos ágiles se auxilian a menudo de buenas herramientas para seguir la pista en un diseño en evolución. Si se desarrolla un sistema con un IDE sin contar con buenas herramientas para visualización y análisis de programas, entonces probablemente se requiera más documentación en el diseño.
- ¿Cómo está organizado el equipo de desarrollo? Si el equipo de desarrollo está distribuido, o si parte del desarrollo se subcontrata, entonces tal vez se requiera elaborar documentos de diseño para comunicarse a través de los equipos de desarrollo. Quizá se necesite planear por adelantado.
- ¿Existen problemas culturales que afecten el desarrollo del sistema? Las organizaciones de ingeniería tradicionales presentan una cultura de desarrollo basada en un plan, pues es una norma en ingeniería. Esto requiere comúnmente una amplia documentación de diseño, en vez del conocimiento informal que se utiliza en los procesos ágiles.
- ¿Qué tan buenos son los diseñadores y programadores en el equipo de desarrollo? Se argumenta en ocasiones que los métodos ágiles requieren niveles de habilidades superiores a los enfoques basados en un plan, en que los programadores simplemente traducen un diseño detallado en un código. Si contamos con un equipo con niveles de habilidades bajos, es probable que se necesite del mejor personal para el diseño, y otros los responsables de la programación.

- ¿El sistema está sujeto a regulación externa? Si un regulador externo tiene que aprobar el sistema. Por ejemplo la Agencia de Aviación Federal de EEUU aprueba el software que es crítico para la operación de una nave. Entonces requiera una documentación detallada. (SOMMERVILE 2011).

La empresa VersionOne realizó una encuesta en el 2007, con respecto al uso de las metodologías ágiles, los resultados de dicha encuesta se muestran en la **Figura 2**.

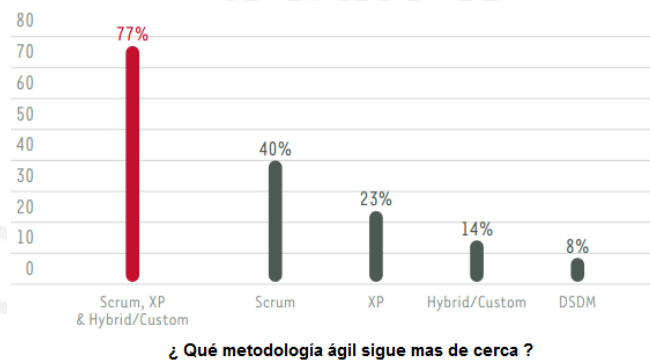


Figura 2: Encuesta de VersionOne (2007): Qué metodología ágil sigue más de cerca.

Fuente: <http://www.versionone.com/pdf/stateofAgiledevelopmentsurvey.pdf>

VersionOne generó una encuesta entre agosto y noviembre del 2012, en su 7ma edición publicó los resultados de dicha encuesta, teniendo como novedad la presencia de Kanban y metodologías híbridas (ver **Figura 3**)

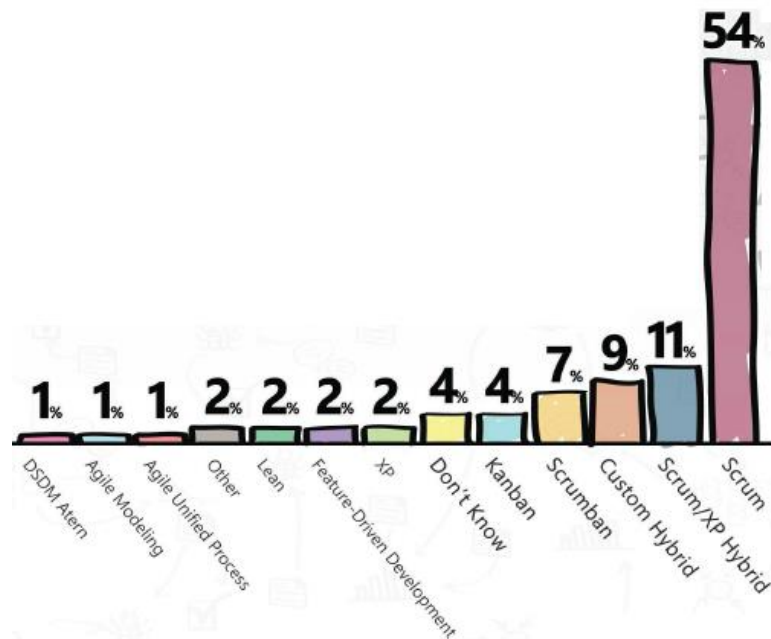


Figura 3: Encuesta de VersionOne (Agosto 2012): Qué metodología ágil sigue más de cerca.
 Fuente: <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf>

Existe un trabajo desarrollado por Ariel Erijman y Alejandro Goyén de la Universidad Católica de Uruguay – Dámaso Antonio Larrañaga, en la que se desarrolla un conjunto de preguntas que yacen de los blogs y foros, preguntas como:

- ¿Cómo se debe adoptar XP en sistemas legados?
- ¿Qué se debe hacer para que los Test de Unidad puedan acceder a los métodos privados?
- ¿Cuál debe ser el orden de implementación de las 12 prácticas de XP? ¿Cómo se deben testear las entradas y salidas de los GUI? ¿Se pueden automatizar estas pruebas?
- ¿Qué tipo de actividades pueden realizar aquellos programadores que han finalizado el trabajo planificado para la iteración?
- ¿Qué tipo de estrategia Test-Code-Refactor (TCR) mejor se ajusta a la metodología XP?
- ¿Los defectos encontrados deben ser ingresados como Historias en la iteración? ¿Cómo se debe definir la prioridad entre la corrección de defectos y el desarrollo de nueva funcionalidad?

- ¿La metáfora de XP es un sustituto válido a lo que tradicionalmente se conoce como la arquitectura de un producto de software?
- ¿Cómo debe ser interpretado el concepto de simplicidad en XP?
- ¿Es aceptable desarrollar la documentación una vez terminado con el desarrollo o debe ser desarrollada en paralelo?
- ¿Es viable utilizar a XP en la construcción de cualquier tipo de software?

Estas preguntas forman parte de los problemas o “huecos” que presenta el manifiesto de XP.

2.6 Metodologías Ágiles vs Tradicionales

No existe una metodología universal o como se dice la piedra filosofal de las metodologías, para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc.) (SOMMERVILE 2011).

Históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobretodo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Esto ha llevado hacia un interés creciente en las metodologías ágiles como ya se vio en el capítulo 1 (tabla 1). Sin embargo, hay que tener presente los inconvenientes y restricciones que se enumeraron en el capítulo anterior.

La **Tabla 2** recoge esquemáticamente las principales diferencias de las metodologías ágiles con respecto a las tradicionales (“no ágiles”). Estas diferencias que afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización.

Tabla 2: Diferencias entre Metodologías Ágiles y no Ágiles.

Fuente: Amaro,2007

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

La **Tabla 3** compara las distintas aproximaciones ágiles en base a tres parámetros: vista del sistema como algo cambiante, tener en cuenta la colaboración entre los miembros del equipo y características más específicas de la propia metodología como son simplicidad, excelencia técnica, resultados, adaptabilidad, etc. También incorpora como referencia no ágil el Capability Maturity Model (CMM).

Tabla 3: Ranking de “agilidad” (Los valores más altos representan una mayor agilidad)
Fuente: Luna,2012

	CMM	ASD	Crystal	DSDM	FDD	LD	SCRUM	XP
Sistema algo Cambiante	1	5	4	3	3	4	5	5
Colaboración	2	5	5	4	4	4	5	5
Características Metodología (CM)								
Resultados	2	5	5	4	4	4	5	5
Simplicidad	1	4	4	3	5	3	5	5
Adaptabilidad	2	5	5	3	3	4	4	3
Excelencia Técnica	4	3	3	4	4	4	3	4
Prácticas de Colaboración	2	5	5	4	3	3	4	5
Media CM	2.2	4.4	4.4	3.6	3.8	3.6	4.2	4.4
Media Total	1.7	4.8	4.5	3.6	3.6	3.9	4.7	4.8

Como se observa en la **Tabla 3**, todas las metodologías ágiles tienen una significativa diferencia del índice de agilidad respecto a CMM y entre ellas destacan ASD, Scrum y XP como las más ágiles.

La **figura 4**, muestra una comparación entre los tipos de ciclos que siguen las metodologías tradicionales, y XP:

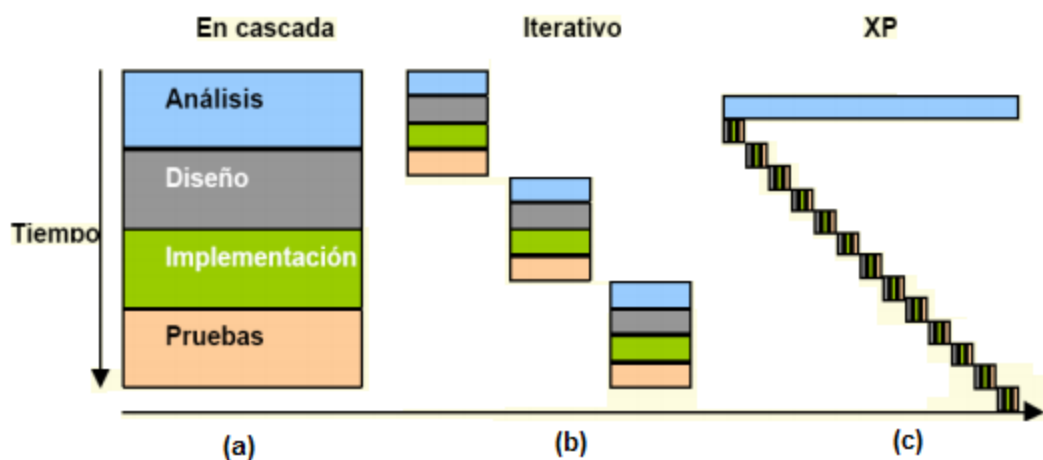


Figura 4: Evolución de los largos ciclos de desarrollo en cascada (a) a ciclos iterativos más cortos (b) y a la mezcla que hace XP.(AMARO 2007)

Las siguientes figuras muestran los costos con respecto al cambio que se alcanzan en las diferentes etapas dentro de las metodologías tradicionales (ver **figura 5**). Vemos que un cambio en la etapa de Requisitos no es significativo comparado con cambios en la etapa de producción.

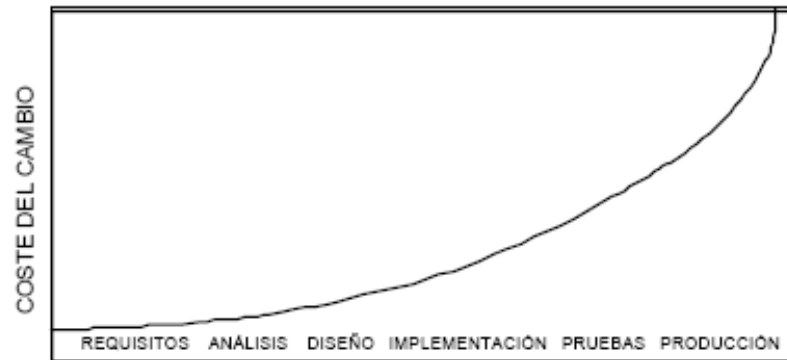


Figura 5: Costo del cambio en la ingeniería de software tradicional.

Fuente: Luna,2012

Para el caso de XP, que considera los cambios constantemente el costo por cambio en cada etapa se mantiene constante, como se demuestra en la **figura 6**:



Figura 6: Costo del cambio en extreme programming.

Fuente: Luna,2012

En el **anexo G** podemos apreciar valores estadísticos de proyectos de software, las causas y factores que influyen en que sean exitosos o fracasos.

2.7 Estado de Herramientas Similares

Actualmente son pocas las instituciones que desarrollan o adquieren herramientas similares, a la que el proyecto plantea. Debido a que es una herramienta muy especializada. Sin embargo, instituciones que trabajan en el rubro de seguridad como la Policía y algunas municipalidades cuentan con herramientas web que registran ocurrencias de incidencias, como pueden ser robos, alteración del orden, pandillaje, venta ilícita de drogas y alcohol, etc.

- Iniciativa de la empresa privada: La **figura 7** muestra una aplicación para el registro de incidencias de seguridad, desarrollado por una iniciativa de la empresa privada.



The screenshot shows a web application titled "Seguridad Ciudadana". The main content area is a registration form with the following sections:

- Requisitorios**: A text input field with the instruction: "Juzgado que ordena requisitoria, en caso de no ser requisitoria, Documento que ordena la intervención."
- EDAD**: A text input field with the instruction: "Si no lo proporciona, no coloque *N/E, *NO INDICA*, *S/N*", sólo déjelo en blanco."
- FOTOGRAFIA FACIAL**: A text input field with the instruction: "Ingresa la URL del servicio web de álbumes fotográficos digitales (Picasa) donde cargaste el archivo de imagen. Ejemplo: (http://picasaweb.google.com/lh/photo/sjO6P3bVVo-MJgvrJV5fiQ?authkey=Gv1sRgCLbMq6vG-Sv_3AE&feat=directlink)."
- MOTIVO ***: A list of radio button options:
 - Omisión a la Asistencia Familiar
 - Robo agravado
 - Hurto agravado
 - Violación Sexual
 - Estafa
 - Violación Sexual a menores
 - Falsificación de Monedas y Billetes
 - TID
 - Microcomercialización de Drogas

On the left side, there is a navigation menu with options like "Página principal", "Formularios", "Ver Resultados", "MiniBlog", and "UTN-FRGP". At the bottom left, there is a promotional banner for AdWords.

Figura 7: Página Web que registra incidencias de la ciudadanía.

Referencia: <https://sites.google.com/site/seguridadciudadanaypnp/>

Este es un esfuerzo de integración tecnológica orientada a la rápida, simple y eficiente captura de información sobre novedades de seguridad ciudadana en diferentes eventos que se desarrollan en el país.

El sitio web nace como una iniciativa ciudadana y la decidida colaboración de la empresa privada como Masautoya.com para ayudar, integrar, convocar e insertar a los diferentes sectores de la sociedad comprometidos en esta causa.

- Herramienta de la Policía Nacional: Por otro lado, la policía Nacional en conjunto con algunas municipalidades cuenta con un software para el registro de incidencias de seguridad, ejemplo de ello se refleja en la **figura 8**, donde se aprecia el flujo de registro.

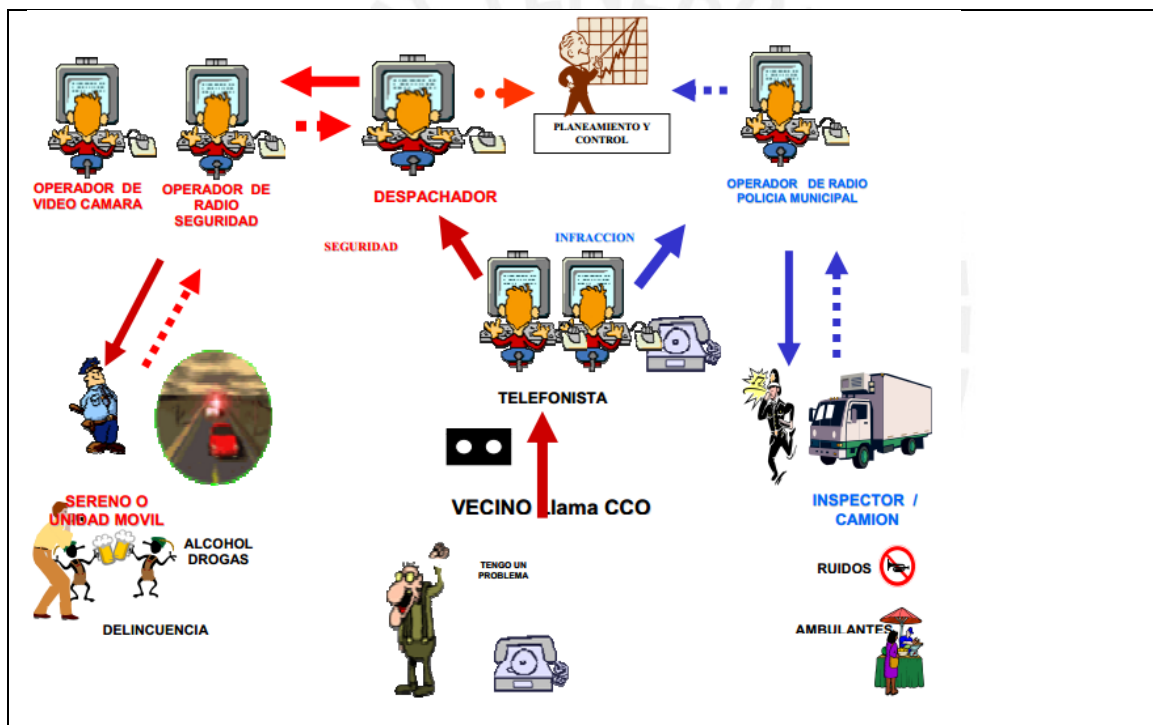
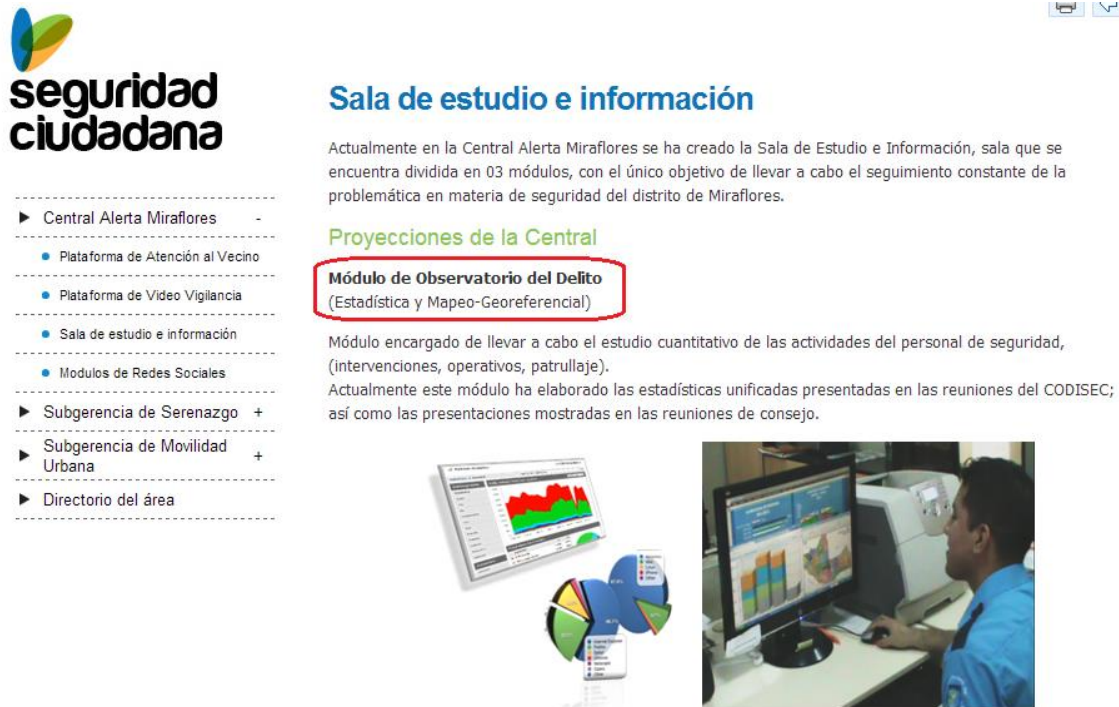


Figura 8: Proceso de atención al vecino a través del Centro de Control de Vigilancia

Fuente:

<http://www.munisurco.gob.pe/municipio/surcoSeguro/planDistritalSeguridadCiudadana/planDistritalSeguridadCiudadana.pdf>

- Municipalidades: Muchas de las municipalidades cuentan actualmente dentro de sus portales con módulos de asistencia al vecino, para el registro de algún delito. Estos módulos cuentan con registro de videos, conexión con agentes de serenazgo y servicios similares de seguridad. La **figura 9**, muestra este servicio.



seguridad ciudadana

- ▶ Central Alerta Miraflores -
 - Plataforma de Atención al Vecino
 - Plataforma de Video Vigilancia
 - Sala de estudio e información
 - Modulos de Redes Sociales
- ▶ Subgerencia de Serenazgo +
- ▶ Subgerencia de Movilidad Urbana +
- ▶ Directorio del área

Sala de estudio e información

Actualmente en la Central Alerta Miraflores se ha creado la Sala de Estudio e Información, sala que se encuentra dividida en 03 módulos, con el único objetivo de llevar a cabo el seguimiento constante de la problemática en materia de seguridad del distrito de Miraflores.

Proyecciones de la Central

Módulo de Observatorio del Delito
(Estadística y Mapeo-Georeferencial)

Módulo encargado de llevar a cabo el estudio cuantitativo de las actividades del personal de seguridad, (intervenciones, operativos, patrullaje). Actualmente este módulo ha elaborado las estadísticas unificadas presentadas en las reuniones del CODISEC; así como las presentaciones mostradas en las reuniones de consejo.




Figura 9: Módulo de Observación del Delito de la Municipalidad de Miraflores

Fuente: http://www.miraflores.gob.pe/_contenTemp12.asp?idpadre=5300&idhijo=5301&idcontenido=5451

La mayoría de estas aplicaciones son muy elaboradas, cuentan con integración a sistemas de cámaras de vigilancia, componentes GPS, y vínculos con centrales telefónicas. Además se instalan en servidores y su alcance es a nivel distrital o nacional.

La idea del software que se ha implementado es contar con una herramienta que sea portable, que registre eventos, vínculos y genere archivos encriptados que sean fáciles y seguros de recopilar. No requiere instalar un servidor, sino que funcione de manera local.



PARTE 2 Ejecución del Proyecto

En esta parte se detalla el proceso de Desarrollo del Proyecto, siguiendo la metodología XP. En esta etapa del Proyecto tenemos la Exploración, Planificación de entregas, Iteraciones y la puesta en Producción. Estas fases nos van a permitir obtener las necesidades de los usuarios llamadas historias de usuarios, definir el marco de desarrollo, identificar las tecnologías que aplicaremos, priorizar las historias de usuarios para su desarrollo en pequeñas iteraciones incrementales, generar la documentación y finalmente colocar en producción la aplicación.

En el capítulo 4 se hará una breve demostración del software desarrollado.

Capítulo 3

Desarrollo del Proyecto

Objetivos

El objetivo de este capítulo es aplicar la metodología XP en el proyecto de software que se ha definido. Así en este capítulo:

- Examinaremos los detalles de la fase de exploración para el proyecto en cuestión.
- Se generarán las historias de usuarios las cuales serán analizadas y priorizadas.
- Como parte de la exploración, analizaremos las herramientas que utilizaremos para codificar, diseñar y probar.
- Efectuaremos las iteraciones que serán definidas en la planificación.

Contenido

- 3.1 Exploración
- 3.3 Planificación de las Entregas
- 3.4 Iteraciones
- 3.5 Producción

3.1 Exploración

En esta fase los 2 principales clientes de la aplicación nos plantean a grandes rasgos, y con sus propias palabras y términos las historias de usuario. Conociendo sus historias, podremos identificar las tecnologías que apoyarán a las ya fijadas (Java y XML), determinaremos la interfaz de desarrollo para JAVA, las librerías que necesitaremos para administrar los archivos XML y todo el entorno de desarrollo.

Los objetivos que buscaremos en esta etapa son los siguientes:

- Se obtendrán las **historias de usuarios**, en tarjetas físicas, con sus propias palabras y términos de los clientes.
- Se analizarán las herramientas y estándares que emplearemos.
- Se desarrollarán los prototipos de pantallas.
- Se definen los roles del sistema.
- Se generarán las clases y la estructura de datos.
- Se generará el diagrama de estados para los eventos.

3.1.1 Historia de Usuarios

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado.

Se resumen las historias de pedidos en la **Tabla 4**: La columna usuarios indica el usuario que generó la historia, y que participará en el análisis de las mismas indicando su prioridad y validando su funcionamiento.(COHN, 2004)

Tabla 4: Lista de Historias de Usuarios que se atenderán dependiendo de la prioridad y complejidad que presenten cada uno.

Nº	Nombre de Historia	Usuario
1	Gestión de Eventos	David Ramírez
2	Generación de Archivo de Eventos	David Ramírez
3	Carga de Eventos	Herbert Calderón
4	Registro en Base de Datos	Herbert Calderón
5	Gestión de Informantes	David Ramírez
6	Gestión de Módulos	David Ramírez
7	Control de Acceso de Usuarios	David Ramírez
8	Consultas y reportes estadísticos	Herbert Calderón
9	Exportación de Informantes	Herbert Calderón
10	Exportación de Módulos	Herbert Calderón

Nota: Las Historias 9 y 10 se agrupan dentro de las historias 6 y 7 respectivamente. El detalle de las fichas de las historias se puede apreciar en el **anexo B**.

3.1.2 Herramientas

Definiremos el marco de desarrollo, donde se establecerán las herramientas para modelar, manejo de archivos XML, la interfaz de desarrollo para la programación, los componentes adicionales que se necesitarán, los estándares que seguiremos y los patrones de prototipos.

Desarrollo:

Utilizaremos como framework o marco de trabajo las librerías proporcionadas por el JDK, en este caso la versión estándar SDK (Standard Development Kit). Como interfaz de desarrollo de Java, el NetBeans, y dos aplicaciones libres para los diagramas UML y el manejo de los archivos XML.

- Java™ 2 SDK, Standard Edition 1.6.0_13 Sun Microsystems.
- Poseidon for UML Community Edition 2.4.1 para la realización de diagramas UML.
- Altova XMLSpy, para la administración de archivos XML.
- Netbeans 6.9.1 ml java, como entorno de desarrollo de todo el proyecto.

Ejecución

Las librerías requeridas para la ejecución de nuestros archivos de java y formularios en el ambiente de producción será mediante:

- JRE (Java Runtime Environment) En su forma más simple, el entorno en tiempo de ejecución de Java está conformado por una Máquina Virtual de Java o JVM, un conjunto de bibliotecas Java y otros componentes necesarios para que una aplicación escrita en lenguaje Java pueda ser ejecutada. El JRE actúa como un "intermediario" entre el sistema operativo y Java.

Arquitectura de Desarrollo

Tras la selección de tecnologías se establece la arquitectura general del sistema que se puede ver en la **figura 10**:

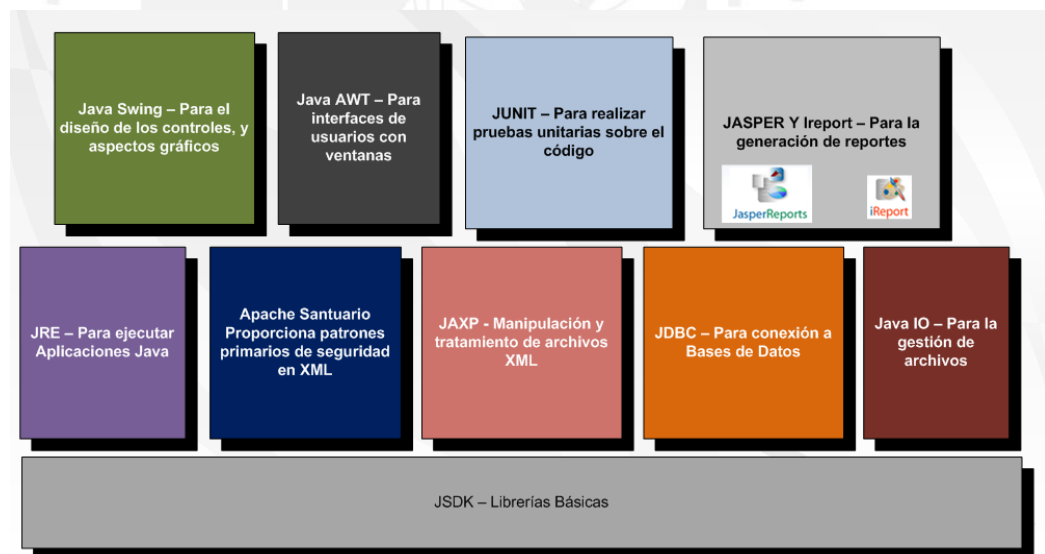


Figura 10: Arquitectura de los componentes que se emplearán. El JSDK nos proporciona las librerías más básicas de las cuales se soportan cualquier proyecto Java.

Todos los componentes de la aplicación se basan en las librerías básicas de Java que se encuentran en el JSDK (Java Standard Development Kit). Sobre estas librerías se ejecutan las siguientes librerías:

- JRE (Java Runtime Environment), se encarga de la ejecución de la aplicación.
- Las librerías del Apache Santuario nos permiten encriptar archivos XML.
- Las librerías de JAPX (Java Api for XML Processing), se utilizan para la manipulación de archivos XML.
- Las librerías IO nos permiten gestionar archivos de sistema. Para la lectura y escritura de archivos
- La parte de Interfases las proporcionan Java Swing, Awt y Jasper. Estas librerías serán el soporte para la aplicación con respecto a la parte visual como formularios, grillas, botones, etc.
- Las librerías JUnit nos permiten generar las pruebas unitarias de las clases.

Estándares de Desarrollo

Definiremos los estándares tanto visuales como de programación del código.

- En el diseño de los formularios se usará el patrón “diseño de tres regiones”.

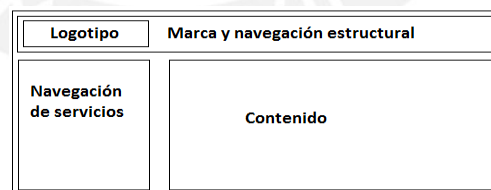


Figura 11: Diseño de tres regiones. Formado por el logo, la navegación y el contenido.

Fuente: DAVID PARSONS, Desarrollo de Aplicaciones Web Dinámicas con XML y JAVA. Anaya Multimedia España, p. 84.

Se compone del logo de la aplicación en la parte superior izquierda. La Marca y Navegación estructural se compone en este caso del menú de opciones y títulos de los formularios. La navegación de servicios, ofrece

opciones como filtros para las consultas. Y el contenido muestra los registros o información propiamente dicho.

- En el código, se empleará la notación Pascal y Camel Casing. Se puede apreciar el modo de uso de estas notaciones en la **tabla 5**.

Tabla 5: Muestra la aplicación de las notaciones en los diferentes elementos que se utilizan al momento de la codificación.

Identificador	Uso de mayúsculas o minúsculas	Ejemplo
Clase	Pascal	AppDomain
Tipo de enumeración	Pascal	ErrorLevel
Valores de enumeración	Pascal	FatalError
Evento	Pascal	ValueChanged
Clase de excepción	Pascal	WebException
Campo estático de sólo lectura	Pascal	RedValue
Interfaz	Pascal	IDisposable
Método	Pascal	ToString
Espacio de nombres	Pascal	System.Drawing
Parámetro	Camel	typeName
Propiedad	Pascal	BackColor

En la columna 1 de la **tabla 5** tenemos el Identificador, que será el tipo de elemento que encontraremos al programar como por ejemplo una clase, en la columna 2 la definición del estándar a utilizar sea Pascal o Camel, y en la columna 3 un ejemplo por cada tipo de identificador.

3.1.3 Prototipos

Los prototipos de pantallas vienen ser un esbozo del modo de interactuar de los usuarios con la aplicación, explican cómo se presentarán y se distribuirán las opciones, botones, registros, menús, etc.

El menú de opciones se conformará de Registro, donde se registrarán los eventos o se cargarán los eventos, dependiendo del perfil con que se ingrese.

Mantenimiento, para la gestión de módulos y sus agentes. Esta opción la tendrá los administradores. Reportes, para generar algunas estadísticas. Esta opción también será mostrada al perfil de administradores.

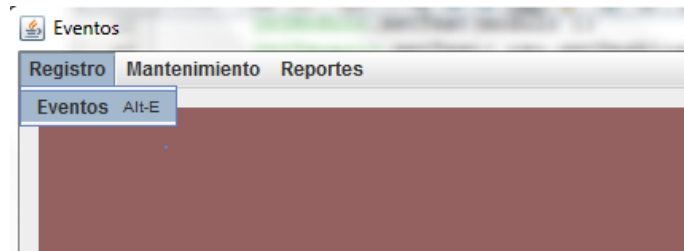


Figura 12: El menú se conforma del Registro, Mantenimiento, y Reportes. Todas las opciones dependen del perfil que corresponda.

Las consultas tendrán la siguiente presentación respetando el diseño de 3 regiones, las consultas muestran los filtros al lado izquierdo y los resultados al lado derecho. Este formato se presentará en los mantenimientos como módulo, agentes, estados, etc. Ver **figura 13**

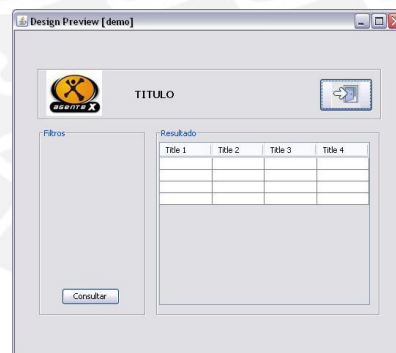


Figura 13: Los formularios principales tendrán esta presentación, con los filtros para búsquedas a la izquierda, los resultados a la derecha y la botonera a la derecha. Las cabeceras y detalles tendrán la siguiente presentación:



Figura 14: Los formularios que son del tipo cabecera detalle, como en los eventos, presentan los registros de cabecera en la parte superior, y su detalle se cargará en la parte inferior.

3.1.4 Definición de Roles

Los roles nos permiten restringir las opciones de la aplicación, dependiendo de las funciones que cumplen sus operadores. Para nuestra aplicación definimos los siguientes roles:

Tabla 6: La tabla muestra los roles que se considerarán en la aplicación, cada uno está asociado a opciones de menú. En el caso del rol Informante solo puede manipular sus eventos.

Rol	Descripción
Administrador Interno	Es el responsable de insertar los eventos exportados en el archivo XML a la base de datos del sistema interno. Además de gestionar los informantes y módulos.
Administrador Externo	Es el responsable de verificar los eventos enviados por los agentes.
Informante	Es el responsable de gestionar los eventos.

3.1.5 Diagrama de Clases

El diagrama de clases, permitirá identificar las clases que conforman la aplicación y como estas clases se relacionan entre sí. Mostrando estos componentes y sus relaciones en un diagrama. Dicho modelo recoge la especificación detallada de cada una de las clases, es decir, sus atributos, operaciones, métodos, y el diseño preciso de las relaciones establecidas entre ellas, bien sean de agregación, asociación o jerarquía.

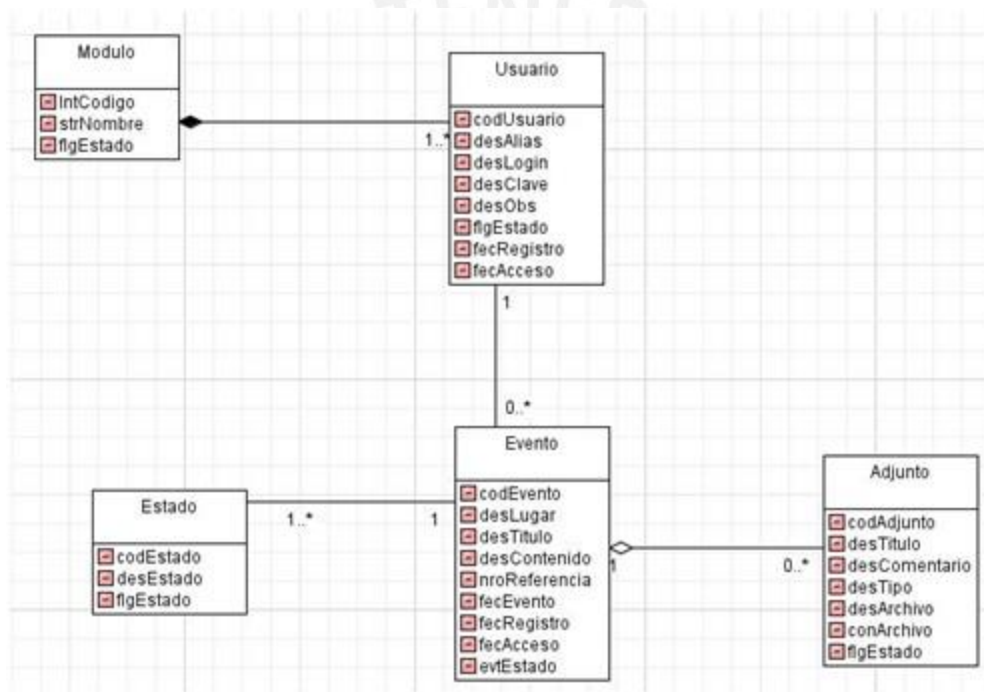


Figura 15: El diagrama de clases, está conformado por las clases, sus relaciones y la cardinalidad.

Podemos apreciar que la clase Módulo tiene una relación de composición con la clase usuario, ya que un módulo se compone de al menos un usuario, pudiendo tener muchos agentes cada módulo.

Los usuarios o agentes generan eventos, los cuales presentan en cualquier momento un estado. Y estos eventos pueden o no contar con documentos adjuntos, por ello es que mantiene una relación de asociación.

3.1.6 Estructura de Datos

Se muestra a continuación la estructura de datos que se utilizará. En el **anexo D** se puede ver aplicada esta estructura en los archivos XML que utilizaremos como datos.

Tabla 7: Estructura de datos, donde se detallan las tablas que conforman el sistema. Como información adicional se muestran los campos, el tipo de datos y la descripción.

Tabla: Módulo , administra la información de los módulos que hay en cada región. Ejemplo el módulo de Pucallpa, Cuzco, Puno.			
Campo	Tipo	Clase	Descripción
intModulo	Primaria	Numérico	Identificador del módulo.
strNombre		Cadena	Descripción o nombre del lugar del módulo.
flgEstado		Cadena	Bandera que indica si el módulo esta activo.
IstUsuario		Lista Usuario	Lista de los usuarios que conforman el módulo
Tabla: Usuario , administra la información de los informantes de cada módulo.			
Campo	Tipo	Clase	Descripción
codUsuario	Primaria	Numérico	Identificador del informante.
desAlias		Numérico	Descripción del nombre o alias del informante
desLogin		Cadena	Descripción del nombre de ingreso al sistema.
desClave		Cadena	Clave de acceso del informante al sistema.
desObs		Cadena	Breve comentario u observación del informante.
flgEstado		Cadena	Bandera que indica si el informante esta activo.
fecRegistro		Cadena	Fecha de registro del informante.
fecAcceso		Cadena	Fecha del último acceso a la información.
Tabla: Evento , administra la información de los eventos e incidentes registrados por los informantes.			
Campo	Tipo	Clase	Descripción
codEvento	Primaria	Numérico	Identificador del evento.
desLugar		Cadena	Descripción del lugar del evento.
desTitulo		Cadena	Descripción del título del evento.
desContenido		Cadena	Contenido del evento.
nroReferencia		Cadena	Número de un documento de referencia.
fecEvento		Cadena	Fecha del evento.
fecRegistro		Cadena	Fecha de registro del evento.
fecAcceso		Cadena	Fecha del último acceso al registro de evento.
evtEstado		Estado	Clase Estado que indica la situación del evento: Registrado, Por Exportar, Exportado.
IstAdjunto		Lista Adjunto	Lista de los documentos adjuntos.
Tabla: Adjunto , administra los documentos adjuntos vinculados a un evento.			
Campo	Tipo	Clase	Descripción

codAdjunto	Primaria	Numérico	Identificador de la región
desTitulo		Cadena	Título del documento adjunto.
desComentario		Cadena	Breve descripción del documento.
conArchivo		Cadena	Contenido del archivo en formato hexadecimal
desTipo		Cadena	Tipo de archivo
desArchivo		Cadena	Nombre completo del archivo adjunto.
flgEstado		Cadena	Bandera que indica si el adjunto esta activo.
Tabla: Estado , administra los documentos adjuntos vinculados a un evento.			
Campo	Tipo	Clase	Descripción
codEstado	Primaria	Numérico	Identificador del estado.
desEstado		Cadena	Descripción del estado.
flgEstado		Cadena	Bandera que indica el estado activo.

3.1.7 Diagrama de Estados

Muestra los estados por los que pasa un evento, desde que es registrado por el agente, hasta que es procesado por el administrador. Pudiendo ser anulado o por procesar antes de ser encriptado.

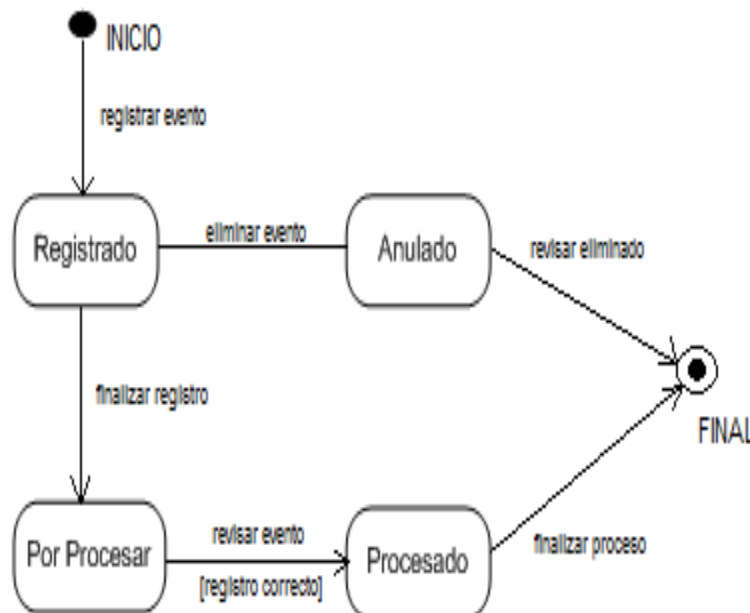


Figura 16: Diagrama de estados de los eventos. Un evento es registrado, al terminar pasa a Por Procesar, aquí es exportado a XML, en el estado Procesado, se recibe el XML y se revisa su contenido, para pasar al estado Final.

3.1.8 Consideraciones para las Pruebas de Aceptación

Las pruebas de aceptación tienen como objetivo la validación de las funcionalidades y rendimientos del sistema que fueron establecidas en las historias de los usuarios, dichas pruebas son realizadas por el cliente.

Tabla 8: Indica los aspectos que se deben esperar al momento de la generación de las pruebas de aceptación.

Nº	Descripción	Resultado
1	Identificar todos los posibles resultados observables de las historias de usuarios.	Listado de resultados observables.
2	Identificar los resultados que terminan la historia y los que permiten continuar dentro la historia	Listado de resultados observables clasificados en historias terminales y no terminales.
3	Identificar todos los caminos de ejecución posibles.	Listado de caminos de ejecución posibles y a cuál de los resultados identificados conduce.
4	Asignar un conjunto de valores válidos y valores del entorno a cada camino de ejecución para obtener el resultado esperado.	Listado de caminos de ejecución, con sus resultados esperados y los valores que permiten obtener dicho resultado.
5	Eliminación de caminos redundantes.	Listado de caminos de ejecución, valores de prueba y resultados que se convertirán en pruebas de aceptación.

3.2 Planificación de las Entregas

En esta fase realizamos un trabajo de identificación y estimación tanto de las historias y sus prioridades como del tiempo que tomará su desarrollo. Definimos las iteraciones y que historias se desarrollarán en cada iteración, como los entregables y que iteraciones lo conformarán. Comparando con RUP, esta fase equivaldría a la Elaboración.

Nuestros objetivos son los siguientes:

- Se analizarán las **historias**, y se estimarán los tiempos, esfuerzos y riesgos que demanden su implementación. Una historia que demande más de 3 días implica dividir la historia. Las historias de menos de un día serán agrupadas.

- Se presentarán las historias al cliente para que priorice su implementación. La priorización está dada por el cliente y se trabaja con valores de alto, medio y bajo.
- Se realizará la **planificación de las entregas**, determinando el tiempo que nos tomará en implementar una entrega.
- Cada **entrega** estará compuesta de **iteraciones** que demandan de 1 a 3 semanas por iteración, las cuales son planificadas con el cliente, dependiendo de su prioridad.
- Cada iteración está compuesta por **historias de usuarios**.

3.2.1 Priorización de las Historias

El proyecto a desarrollar consta de un solo entregable, con 3 iteraciones. En esta etapa se realiza la priorización de las historias por parte del cliente.

Tabla 9: Categorización de las historias de usuarios, por parte del equipo XP.

Nº	Nombre de Historia	Prioridad	Riesgo	Esfuerzo	Iteración
1	Gestión de Eventos	Alta	Bajo	3	1
2	Generación de Archivo de Eventos	Media	Medio	3	1
3	Carga de Eventos	Alta	Bajo	3	1
4	Registro en Base de Datos	Alta	Bajo	2	2
5	Gestión de Informantes	Baja	Bajo	2	2
6	Gestión de Módulos	Baja	Bajo	2	2
7	Control de Acceso de Usuarios	Baja	Bajo	1	3
8	Consultas y reportes estadísticos	Baja	Bajo	1	3
9	Exportación de Informantes	Baja	Bajo	0.5	3
10	Exportación de Módulos	Baja	Bajo	0.5	3

A continuación se muestra el Plan de Entrega con las iteraciones, historias (H) y tareas involucradas:

Tabla 10: Plan de Entrega, donde se muestran las iteraciones que se efectuarán sobre las historias priorizadas. Son 3 iteraciones que están formados de 3,3 y 2 historias de usuarios, finalizando en el primer y único release.

PLAN DE ENTREGA		
Iteración 1	Iteración 2	Iteración 3
H1: Gestión de Eventos Tareas - Comprobación de las estructuras	H4: Registro en la estructura de datos. Tareas	H7: Control de Acceso de Usuarios(Perfiles) Tareas

<p>de datos (archivos XML)</p> <ul style="list-style-type: none"> - Crear formulario de eventos. - Mostrar eventos registrados. - Registrar cabecera de evento. - Registrar adjuntos al evento. - Modificar un evento seleccionado. - Eliminar un evento seleccionado. <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias - Aceptación 	<ul style="list-style-type: none"> - Crear los procedimientos de inserción en las estructuras de base de datos relacional. - Verificación de los registros ingresados a la base de datos. <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias. - Aceptación 	<ul style="list-style-type: none"> - Creación del formulario de ingreso - Verificación de datos ingresados. - Opciones de menú por perfil. <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias. - Aceptación
<p>H2: Generación de Archivo de Eventos</p> <p>Tareas</p> <ul style="list-style-type: none"> - Exportar eventos seleccionados. - Encriptar archivos exportados. <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias - Aceptación 	<p>H5: Gestión de Informantes</p> <p>Tareas</p> <ul style="list-style-type: none"> - Consulta de Informantes. - Creación de Informantes. - Modificación de Informantes. - Eliminación de Informantes. - Exportación a XML. <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias. - Aceptación 	<p>H8: Consultas y reportes estadísticos</p> <p>Tareas</p> <ul style="list-style-type: none"> - Creación del formulario de consultas. - Generación del reporte. <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias. - Aceptación
<p>H3: Carga de Eventos</p> <p>Tareas</p> <ul style="list-style-type: none"> - Importar archivos encriptados. - Mostrar eventos importados. <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias. - Aceptación 	<p>H6: Gestión de Módulos</p> <p>Tareas</p> <ul style="list-style-type: none"> - Consulta de módulos. - Creación de módulos. - Modificación de módulos. - Eliminación de módulos. - Exportación a XML <p>Pruebas</p> <ul style="list-style-type: none"> - Unitarias. - Aceptación 	

El detalle de las tareas se encuentra en el **anexo C**.

3.3 Iteraciones

Es en esta parte donde tenemos el proyecto dividido en unidades de trabajos, y describimos las pruebas que debemos realizar para mantener un código libre de errores. Las tareas a realizar en esta fase son las siguientes:

- El programador divide las historias en tareas más pequeñas llamadas **Tareas de Ingeniería**, las cuales son analizadas a mayor detalle estimando su tiempo de desarrollo.
- Se definirán las pruebas de aceptación por parte del cliente, y las pruebas unitarias por parte del programador.
- Se realizan las correcciones si es que hay, se ajustan las historias si son necesarias, se implementa la entrega al software final, y si hay más historias se repite la etapa de entrega.

Según lo planificado se inicia la etapa de codificación y pruebas, para las historias de usuarios que han sido ordenadas en iteraciones. Por tanto tenemos:

3.3.1 Primera Iteración

Según lo planificado se debe desarrollar las historias:

H1: Gestión de Eventos.

H2: Generación de Archivo de Eventos

H3: Carga de Eventos.

Lo cual implica cumplir con los siguientes objetivos:

- Comprobación de las estructuras de datos (archivos XML)
- Crear formulario de eventos.
- Mostrar eventos registrados.
- Registrar cabecera de evento.
- Registrar adjuntos al evento.
- Modificar un evento seleccionado.
- Exportar eventos seleccionados.
- Encriptar archivos exportados.
- Importar archivos encriptados.
- Mostrar eventos importados.
- Registrar algunos valores.

- Realizar validaciones de formatos y campos nulos.
- Se procede a realizar pruebas unitarias.
- Y se efectúan las pruebas de aceptación (ver **anexo E.**)

La **figura 17**, muestra el formulario que se codifica en la primera iteración, las opciones del formulario como nuevo, editar y eliminar quedan pendientes para la segunda iteración.

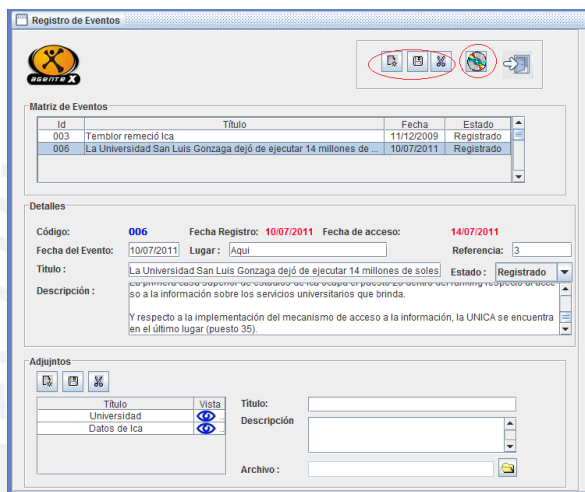


Figura 17: La primera iteración considera la creación del formulario de eventos. La funcionalidad de los íconos marcados es desarrollada en la siguiente iteración.

Al finalizar la primera iteración se tiene los siguientes avances planificados, en el **anexo F** se pueden visualizar las principales funciones y procedimientos que se generan en esta iteración:

Ingreso de Información: Se ingresa como un Informante de Ica, para lo cual el sistema carga automáticamente los eventos que haya registrado, en caso que tenga eventos registrados, de lo contrario se mostrará en blanco, listo para registrar eventos e incidentes, como se ve en la **figura 18**.

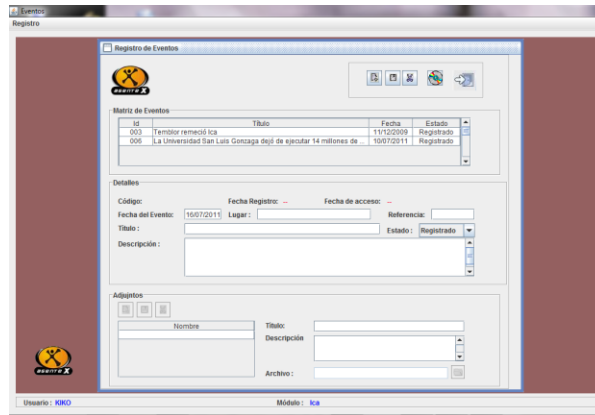


Figura 18: El perfil de Informante del módulo de Ica, muestra por defecto el formulario de eventos.

Gestión de los Registros Ingresados: Con los botones de nuevo, editar y eliminar podemos gestionar los eventos, sobre estos eventos podemos registrar documentos adjuntos (videos, audios, fotos, otros documentos). Los documentos adjuntos son seleccionados y registrados en la parte inferior.



Visualizar Adjuntos: Una vez que tenemos nuestros eventos, estos se muestran como cabeceras en la parte superior, y su detalle se muestra en la parte central. Los adjuntos se pueden visualizar al seleccionar los adjuntos de la parte inferior, como se muestra en la **figura 19**.

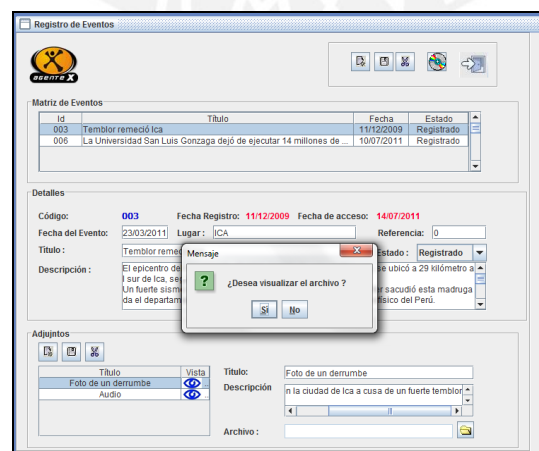



Figura 19: Se muestra los datos del Evento 003. Y la pre-visualización de una foto adjunta.

Si se escoge visualizar el archivo adjunto, el sistema abrirá un formulario para visualizar los distintos formatos de archivos, tal como se muestra en la **figura 20**:



Figura 20: Si se acepta ver el adjunto, el sistema reconoce el tipo de archivo y lo muestra

Cambio de Estado: Una vez que el evento esté completo, se cambia el estado a:

Por Exportar: Estado: Luego se presiona el botón exportar . El sistema procederá a reunir todos los eventos que tengan el estado de “Por Exportar”, para generar un archivo XML encriptado.

El botón nos solicitará la ubicación del archivo, que tendrá como nombre el código del módulo concatenado con el año, mes, día y hora del sistema. Esto se muestra en la **figura 21**.

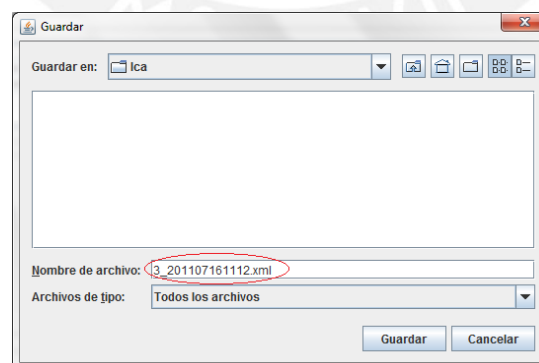


Figura 21: Se selecciona la ubicación del archivo, para exportar los eventos preparados.

Generando el archivo encriptado: Una vez que se presiona el botón Guardar, el sistema generará un archivo encriptado con todos los eventos que estuvieron con el estado Por Procesar, eliminándolos posteriormente.

La **figura 22** muestra el archivo encriptado, y su versión normal:

```
<?xml version="1.0" encoding="UTF-8"?>
<java class="java.beans.XMLDecoder" version="1.6.0_13">
  <object class="java.util.ArrayList">
    <void method="add">
      <object class="Entidades.Evento">
        <void property="codEvento">
          <int>2</int>
        </void>
        <void property="desContenido">
          <string>fhj jh fgjh fhj fgjh </string>
        </void>
        <void property="desLugar">
          <string>Ica</string>
        </void>
        <void property="desTitulo">
          <string>Par de Transportistas 13</string>
        </void>
        <void property="evtEstado">
          <void property="codEstado">
            <int>2</int>
          </void>
          <void property="desEstado">
            <string>Por Exportar</string>
          </void>
        </void>
        <void property="fecAcceso">
          <string>22/01/2010</string>
        </void>
        <void property="fecEvento">
          <string>01/01/2010</string>
        </void>
      </object>
    </void>
  </object>
</java>
```

```
<?xml version="1.0"?>
<java version="1.6.0_13" class="java.beans.XMLDecoder">
  <object class="java.util.ArrayList">
    <void method="add">
      <object class="Entidades.Evento">
        <void property="codEvento">
          <int>2</int>
        </void>
        <void property="desContenido">
          <string>fhj jh fgjh fhj fgjh </string>
        </void>
        <void property="desLugar">
          <string>Ica</string>
        </void>
        <void property="desTitulo">
          <string>Par de Transportistas 13</string>
        </void>
        <void property="evtEstado">
          <void property="codEstado">
            <int>2</int>
          </void>
          <void property="desEstado">
            <string>Por Exportar</string>
          </void>
        </void>
        <void property="fecAcceso">
          <string>22/01/2010</string>
        </void>
        <void property="fecEvento">
          <string>01/01/2010</string>
        </void>
      </object>
    </void>
  </object>
</java>
```

Figura 22: El archivo normal (que no se genera) y su versión encriptado que se enviará.

Carga del archivo encriptado: El archivo que se genera es enviado al administrador externo, que abrirá el archivo y validará su contenido. Si todo está correcto, el archivo es entregado al administrador interno, quien finalmente generará las sentencias para que el contenido del archivo ingrese a una base de datos convencional. En este caso, se generarán sentencias para la base de datos Oracle. La **figura 23** muestra la carga del archivo que envió el informante.

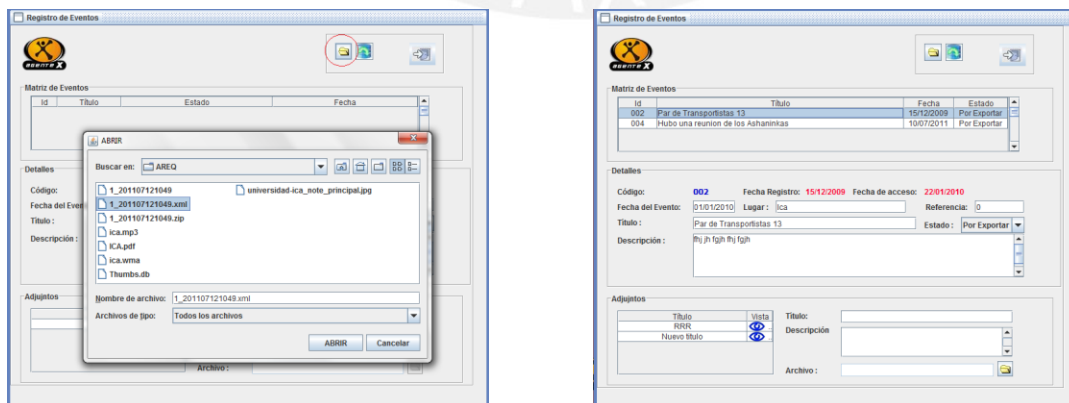


Figura 23: El administrador carga el/los archivo(s) enviado por los módulos, visualizando su contenido. Y permitiendo su ingreso a la red interna.

3.3.2 Segunda Iteración

Según lo planificado debemos desarrollar las historias:

H4: Registro en la estructura de datos.

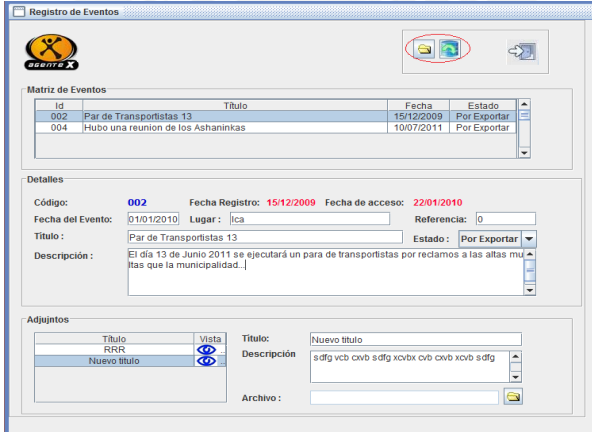
H5: Gestión de Informantes.

H6: Gestión de Módulos.

Los objetivos que persiguen la segunda iteración son los siguientes:

- Se completa las funcionalidades, como carga e inserción en la base de datos.
- Se crea los formularios para la gestión de módulos e informantes.
- Se registra algunos valores.
- Se realizan validaciones de formatos y campos nulos.
- Se procede a realizar pruebas unitarias.
- Y se efectúan las pruebas de aceptación del **anexo E**

El registro de los eventos en el archivo XML, se aprecia en la **figura 24**. La gestión de eventos con todas sus funcionalidades: En esta iteración ya se han completado las funciones de registro, actualización, eliminación y procesamiento.



Registro de Eventos

Matriz de Eventos

Id	Título	Fecha	Estado
002	Par de Transportistas 13	15/12/2009	Por Exportar
004	Hubo una reunion de los Ashaninkas	10/07/2011	Por Exportar

Detalles

Código: **002** Fecha Registro: 15/12/2009 Fecha de acceso: 22/01/2010

Fecha del Evento: 01/01/2010 Lugar: lca Referencia: 0

Título: Par de Transportistas 13 Estado: Por Exportar

Descripción: El día 13 de Junio 2011 se ejecutará un para de transportistas por reclamos a las altas multas que la municipalidad.]

Adjuntos

Título	Vista	Título:
RRR		Nuevo título
Nuevo título		Descripción

Archivo:

Figura 24: El formulario de Eventos pertenece al perfil de Administrador, permite recuperar eventos registrados previamente por los agentes. Los íconos marcados se desarrollan en la última iteración.

Gestión de Informantes: En esta iteración el mantenimiento de agentes por módulo está finalizado:



Figura 25: Mantenimiento de los agentes por módulo. En la figura se pueden visualizar los agentes del módulo de Ica.

Gestión de Módulos: En esta iteración se tiene finalizado el mantenimiento de módulos.



Figura 26: Mantenimiento de módulos, en la figura se está editando los datos del módulo de Iquitos.

Generación de sentencias de base de datos: Finalmente validado el contenido de los archivos, el administrador genera el script o segmento de inserción a la base de datos, como se aprecia en la **figura 27**.

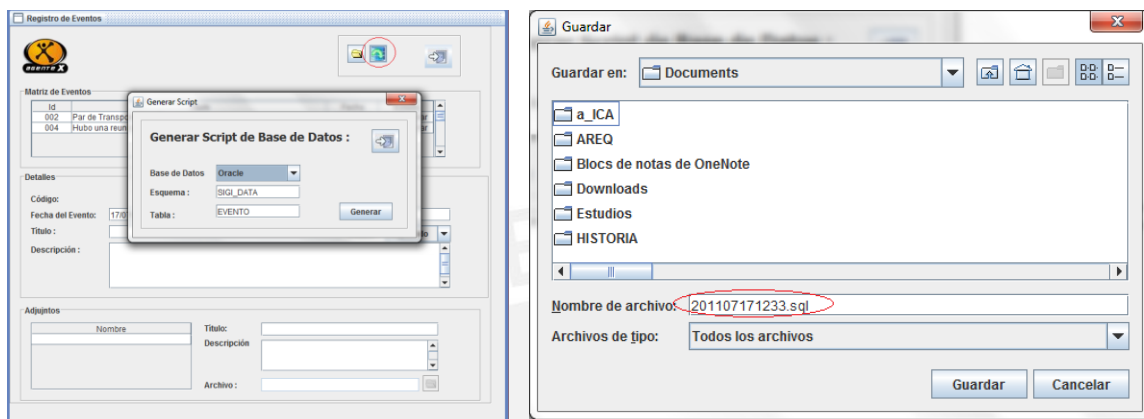


Figura 27: Se selecciona el motor de la base de datos, en esta versión se usa Oracle. Para ello se selecciona la ubicación donde se generará el archivo.

Al finalizar la iteración 2, se ha podido completar la gestión de los módulos y de los informantes, así como generar el archivo script que permite ingresar los eventos en la base de datos estructural.

3.3.3 Tercera Iteración

La tercera iteración debe desarrollar las historias:

H7: Control de Acceso de Usuarios (Perfiles).

H8: Consultas y reportes estadísticos.

Tiene como objetivos:

- Se crea el formulario de acceso al sistema, junto con el menú
- Se crea el formulario para los reportes.
- Se realizan las exportaciones para los formatos más comunes.
- Se procede a realizar pruebas unitarias.
- Y se efectúan las pruebas de aceptación del **anexo E**

El formulario de Acceso: Para el ingreso y la distinción de los usuarios por el perfil que les corresponde:

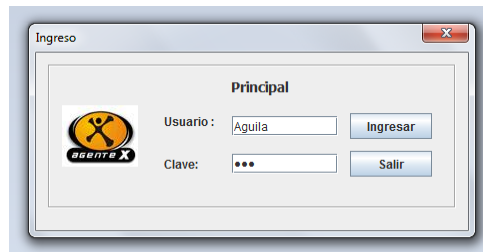


Figura 28: Versión final ya desarrollado y probado del formulario de ingreso

Archivo de configuración, El perfil de la aplicación lo define el archivo config.xml, el cual posee la siguiente estructura:

```

<?xml version="1.0" encoding="UTF-8"?>
- <java class="java.beans.XMLDecoder" version="1.6.0_22">
- <object class="Entidades.Modulo">
- <void property="lstUsuario">
- <void method="add">
- <object class="Entidades.Usuario">
- <void property="codUsuario">
<int>1</int>
</void>
- <void property="desAlias">
<string>Aguila 1</string>
</void>
- <void property="desClave">
<string>asd</string>
</void>
- <void property="desLogin">
<string>Aguila</string>
</void>
</object>
</void>
- <void method="add">
- <object class="Entidades.Usuario">
- <void property="codUsuario">
<int>2</int>
</void>
- <void property="desAlias">
<string>Falcon X</string>
</void>
- <void property="desClave">
<string>asd</string>
</void>
- <void property="desLogin">
<string>Falcon</string>
</void>
</object>
</void>
- <void property="strNombre">
<string>Principal</string>
</void>
</object>
</java>
    
```

Annotations in the image: Blue circles highlight 'lstUsuario', 'add', 'Entidades.Usuario', 'codUsuario', 'desAlias', 'desClave', 'desLogin', and 'Principal'. Blue arrows point from 'Aguila' to 'Usuario 1', from 'Falcon' to 'Usuario 2', and from 'Principal' to 'Nombre del Módulo'.

Figura 29: Estructura que define el perfil Administrador o Informante. El archivo posee la lista de usuarios y sus contraseñas junto con el módulo al que pertenecen.

En la **figura 29**, se muestra la estructura que define el perfil del sistema, además que contiene la lista de los usuarios, en este caso se muestra la estructura para el perfil Administrador tanto interno como externo. La diferencia con la estructura para

informantes, es que los informantes no dice “Principal”, sino el nombre de su módulo. Al momento de ingresar el formulario de ingreso reconoce al módulo o perfil administrador, como se ve en la **figura 30**.



Figura 30: El sistema lee la información del archivo config.xml, e identifica el módulo del usuario. Cada módulo posee su propio archivo config.xml donde se guarda la lista de los usuarios para ese módulo o región.

El formulario de menú: Presenta el marco de trabajo, donde se cargarán los formularios como mantenimientos y registro.

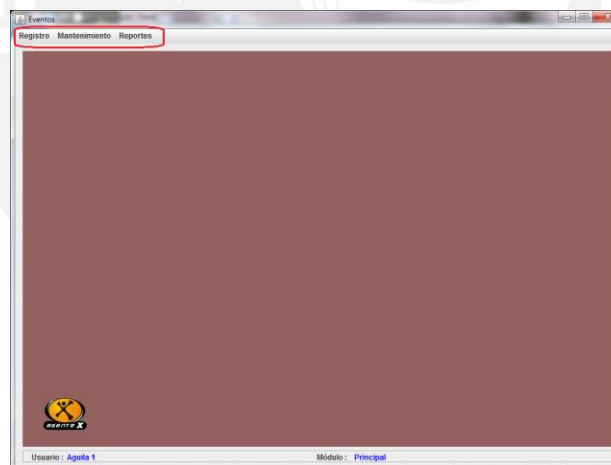


Figura 31: Versión final ya desarrollado y probado del menú principal, en este formulario se muestra el nombre del usuario y el módulo al que pertenece (parte inferior) y las opciones según perfil del menú (parte superior)

Formulario para las consultas y reportes estadísticos:



Figura 32: Prototipo de pantalla de reporte.

La aplicación desarrollada, permite registrar los eventos que suceden en el interior del país, vincular imágenes o documentos, generar un archivo que contiene los eventos de una región del país con sus eventos y adjuntos en un XML encriptado. Luego un administrador consolida los archivos encriptados que han sido enviados por las diferentes regiones.

Para diferenciar entre un usuario que registra sus eventos locales, de un administrador que consolida, se tiene un archivo de configuración XML llamado config.xml, el cual se muestra en la **figura 29**.

3.3 Producción

En esta etapa se procede a instalar y configurar la aplicación en el ambiente del cliente. En este caso, se configura en cada módulo del país y en los equipos de los administradores externo e interno. Los objetivos en esta etapa son:

- Al finalizar cada entrega se pondrá a disposición del usuario la nueva versión del software.
- Si es la última entrega se estará contando con la versión final del software.

En esta etapa se configura la entrega en el ambiente de producción. (SANCHEZ 2004)

Java nos proporciona un mecanismo de generar archivos JAR, o en este caso utilizaremos el JRE (Java Runtime Environment). Para ello se crea un acceso directo al ejecutador de java:

C:\Archivos de programa\Java\jre6\bin

Con referencia a nuestro proyecto, en la **figura 33** podemos apreciar el archivo jar generado y los archivos de apoyo al proyecto:






 lib	17/07/2011 10:50	Carpeta de archivos	
 config.xml	09/07/2011 23:30	Documento XML	2 KB
 estados.xml	09/07/2011 23:30	Documento XML	2 KB
 modulos.xml	17/07/2011 11:57	Documento XML	5 KB
 Proyecto.jar	17/07/2011 11:55	Executable Jar File	5.876 KB

Figura 33: Archivo .jar que contiene la lógica del sistema. El IDE NetBeans proporciona una opción de construir desde un proyecto el archivo jar, y su contenido adicional.

La línea para ejecutar por línea de comando sería:

```
java -jar "C:\CARPETA\Proyecto.jar"
```

En este capítulo se ha desarrollado las fases que expone la metodología aplicada, gran parte del trabajo se encuentra en los anexos. La parte principal del proyecto radica en este capítulo. En cada etapa hemos visto como se va construyendo la aplicación iterativamente e incremental, hasta obtener finalmente un producto deseado. La metodología para este proyecto ha sido muy sencilla y efectiva, debido a la cantidad de historias que se han desarrollado y debido al conocimiento que se posee de las herramientas tanto de desarrollo como de diseño. El propósito de esta tesis es poder proyectar el trabajo realizado en desarrollos más complejos utilizando Extreme Programming.



PARTE 3 Conclusiones del Proyecto de Tesis

4.1 Conclusiones

El trabajo realizado deja las siguientes conclusiones:

1.- La aplicación de la metodología ágil utilizada (Programación Extrema) se ha ajustado perfectamente al tamaño del sistema implementado. Ha tomado un costo de tiempo saber aplicar los principios que esta metodología expone, la documentación de esta metodología es amplia, pero su uso en un caso práctico es muy escasa. Por esto es que este trabajo puede ser usado como guía para proyectos más complejos que quieran utilizar una alternativa a las metodologías tradicionales.

2.- El uso de herramientas de desarrollo libre y estándares, como son Java y XML, han proporcionado todo su potencial y flexibilidad al momento de desarrollar el sistema, lo cual nos ha permitido obtener un sistema con características de portabilidad, donde no importa la plataforma del sistema operativo, ni los recursos del equipo que lo ejecute.

Gracias a la gran información que se cuenta hoy en día de estas herramientas, es que se ha podido obtener este producto.

3.- La filosofía de programar y probar es un gran aporte de esta metodología, el buscar sintetizar o Refactorizar el código genera un código más limpio y fácil de mantener. La experiencia en el desarrollo de software señala que los errores o fallos no detectados a tiempo consumen tiempo y recursos posteriormente, y esta filosofía se anticipa a esto.

4.- El costo por desarrollar un sistema como este ha sido muy reducido, gracias a alternativas libres que encontramos para el desarrollo, como son Apache, NetBeans, Gentleware y otros. Gracias a estas organizaciones se puede contar con las herramientas que se necesitan en el ciclo de desarrollo (análisis, diseño, desarrollo, implementación, pruebas, etc.)

5.- Este proyecto ha sintetizado muchas funcionalidades que se puedan requerir al momento de manejar documentos XML, como son la serialización, el encriptamiento, extracción de información, etc. Sirviendo una vez más como guía para otros proyectos que implementen este estándar de información.

6.- Finalmente, este proyecto tuvo como uno de sus objetivos brindar una herramienta que facilite el flujo de información que se da en el ciclo de Inteligencia para la toma de decisiones, en pro del desarrollo y seguridad nacional. Muchas de las instituciones que se encargan de la seguridad del país no cuentan con una herramienta informática fuera de los límites de su institución, y este proyecto ha apuntado a esta necesidad.

4.2 Beneficios

El trabajo de investigación encontró los siguientes beneficios:

1.- Satisfacción por parte del cliente, debido a que participa constantemente en los avances del software, él prioriza las historias según su criterio, y obtiene un producto de calidad y en el tiempo aproximado.

2.- Satisfacción por parte del equipo, debido a que la metodología no implica mucho conocimiento, existe una amplia difusión de la metodología en la web, documentos, ejemplos y avances sobre problemas o dificultades encontradas.

3.- La metodología se adapta sin ningún problema con las herramientas que se han utilizado. Las pruebas unitarias, refactorización, diseño sencillo, etc. Son actividades que ya vienen incluidas en las actuales herramientas de desarrollo.

4.3 Dificultades y Observaciones

1.- La tesis fue desarrollada de forma individual, lo cual viola una restricción de “Programación por parejas”. Esta nos indica que la programación se realiza en pares, uno codifica y el otro testea. En este trabajo se ha codificado y probado por la misma persona.

2.- No se emplearon metáforas en el proyecto, debido a la poca información que se encontró respecto a este punto, actualmente hay mayor información con respecto a la identificación de las metáforas desde las historias de usuarios. En la página 26 se define este principio.

3.- De igual modo no se dio seguimiento a la estimación de la velocidad que toma para un desarrollador finalizar una iteración. Debido que no se efectuó el rol de Tracker, quien da seguimiento a las estimaciones de los programadores. Para el proyecto se tiene la siguiente estimación utilizando la fórmula de la página 28.

# Iteración	# Historias
1	3
2	3
3	2

$$\text{Velocidad promedio} = (3+3+2) / 3 = 2.67 \text{ hu/iteración}$$

$$\text{Planeación por tiempo} = 3 \text{ iteraciones} * \text{velocidad}$$

$$= 3 * 2.67 = 8 \text{ historias a realizarse}$$

$$\text{Planeación por alcance} = 12 \text{ semanas} / \text{velocidad}$$

$$= 12 / 2.67 = 1.5 \text{ semana/iteración}$$

4.- Si bien la información teórica abunda con respecto a la Programación Extrema, los casos prácticos de su uso son pocos o son muy rebuscados.

5.- Los roles que se cubrieron fueron los siguientes:

- Cliente, efectuado por los 2 usuarios que participaron en el desarrollo del proyecto.
- Programador, quien implementó las historias de usuarios, priorizar las historias junto con el cliente y diseñar y efectuar las pruebas unitarias.
- Coach, quien a medida de su responsabilidad se encargó de efectuar y cumplir las 12 prácticas de la metodología.
- Manager, quien es el coordinador principal para las reuniones con el cliente.

4.4 Observaciones de Usuarios

Las reuniones con los usuarios se realizaron periódicamente, se procuró seguir la recomendación de la metodología donde el método más eficiente y efectivo de compartir información dentro del equipo de desarrollo y los usuarios es con la conversación cara a cara, nos quedamos parados para mantener corta la reunión (15 minutos o menos).

Los usuarios que participaron tuvieron los siguientes comentarios con respecto a la metodología aplicada y el producto obtenido:

- Involucró a los usuarios en el proyecto, y lo asumieron como propio.
- Cambió la manera de desarrollar los proyectos, los cuales empezaban con una entrevista y coordinaciones posteriores, donde el producto se veía al final y nuevamente se concertaba una reunión. A una nueva manera donde van participando y recibiendo versiones útiles.
- Se ha visto la necesidad de formalizar los procesos de desarrollo de software, de tal manera que se respeten las fechas de revisión de los avances, o sean insertadas en los calendarios de trabajo. Así se podrá cumplir con las reuniones pactadas, y asegurar un avance fluido.

- Con respecto a la herramienta tecnológica que se ha elaborado, es una gran iniciativa, ayuda mucho la medición de los colaboradores y ahorra mucho tiempo con respecto al ingreso de la información.
- Da el control de poder registrar a los usuarios y restringir sus opciones. Se sugiere reforzar esta herramienta y masificar su uso en todos los módulos del país.
- Esta forma de trabajar las aplicaciones ha superado a la forma anterior de encuestas y charlas de lo que se iba a obtener.
- Con respecto a la herramienta, cumple a un nivel básico, quizá se deba mejorar el aspecto de poder cargar archivos más pesados.
- Por el lado de la seguridad, la encriptación es una medida más que nos ayuda a proteger la información que nos envían.
- La aplicación ayuda a poder ingresar la información mucho más rápido que el proceso anterior de volcar algo que ya estaba trabajado.
- Esta vez no se ha necesitado de una mayor capacitación de la aplicación puesto que se ha visto en este tiempo su funcionamiento y uno mismo puedo mostrar y enseñar a otros operadores.
- El tiempo en reparar los errores que se han presentado ha sido muy rápido, quizá porque es un piloto pequeño.

Las pruebas de aceptación se realizaron en ambiente de desarrollo siguiendo el formato del anexo E.

BIBLIOGRAFÍA

- AMARO, Sarah y Jorge VALVERDE
2007 “Metodologías Ágiles”
Universidad Nacional de Trujillo
Consulta: 17 de octubre de 2012
<<http://www.seccperu.org/files/Methodologias%20Agiles.pdf>>
- CANÓS, José H. , Patricio LETELLER y Carmen PENADÉS
2004 “Metodologías Ágiles en el Desarrollo de Software”
Universidad Politécnica de Valencia
Consulta: 20 de enero de 2011
<<http://www.willydev.net/descargas/prev/TodoAgil.pdf>>
- COHN, Mike
2004 “User Stories Applied: For Agile Software Development”
Consulta: 04 de Octubre de 2012
<<http://www.mountaingoatsoftware.com/system/asset/file/259/User-Stories-Applied-Mike-Cohn.pdf>>
- DINI, Dirección Nacional de Inteligencia
2007 “Definición.”
Consulta: 17 de enero de 2011
<<http://www.dini.gob.pe/definicion.html>>
- GAMA, João
2012 “Proyectos Ágiles”
Consulta: 10 de octubre de 2012
<<http://www.proyectosagiles.org/historia-de-scrum>>

LETELIER, Patricio

1999

“Desarrollo de un sistema de gestión de una empresa de confecciones”

Universidad Politécnica de Valencia

Consulta: 28 de enero de 2011

<http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemploxp/Gestion_Proyecto.html>

LUNA, Linda

2012

“Metodología Ágil vs Metodología Tradicional”

Universidad Gabriel René Moreno

Consulta: 10 de octubre de 2012

<<http://es.scribd.com/doc/91676941/Metodologias-agiles-vs-tradicionales>>

PARDO, Mario Enrique y Emilio MURADO

2010

Metodologías de desarrollo ágiles: SCRUM y eXtreme Programming[diapositivas].

Consulta: 25 de enero de 2011

<<http://www.slideshare.net/ejordi/metodologas-de-desarrollo-giles-scrum-xp>>

PARSON, David

2009

Desarrollo de Aplicaciones web dinámicas con XML y Java. Primera Edición. Madrid: Ediciones Anaya Multimedia

SANCHEZ, Carlos

2004

“ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras.”

Universidad de Coruña

Consulta: 25 de enero de 2011

<<http://oness.sourceforge.net/proyecto/html/index.html>>

SOMMERVILLE, Ian

2011

Ingeniería de Software. Novena Edición. México:

Ediciones Perason

VariableX.net

2009

“Tecnologías XML“

Consulta: 18 de enero de 2011

<[http://desarrollowebs.net/index.php?option=com_content
&view=article&id=47&Itemid=63](http://desarrollowebs.net/index.php?option=com_content&view=article&id=47&Itemid=63)>

WELLS, Don

2009

“Extreme Programming: A gentle introduction“

Consulta: 20 de enero de 2011

<<http://www.extremeprogramming.org/rules.html>>